Linguistic Databases

Linguistic

edited by

CSLI Lecture Notes Number 77

Databases

John Nerbonne

CSLI PUBLICATIONS
Center for the Study of
Language and Information
Stanford, California

CSLI Publications
Center for the Study of Language and Information
Leland Stanford Junior University
Printed in the United States
02 01 00 99 98 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data

Linguistic databases / edited by John Nerbonne.
p. cm. — (CSLI lecture notes ; no. 77)
Papers presented at a conference held Mar. 23–24, 1995, University of Groningen.
Includes bibliographical references and index.

ISBN 1-57586-093-7 (hardback : alk. paper)
ISBN 1-57586-092-9 (pbk. : alk. paper)

1. Linguistics — Databases — Congresses. 2. Linguistic analysis (Linguistics) — Congresses. 3. Computational linguistics — Congresses. I. Nerbonne, John A., 1951– . II. Series.

P128.D37L56 1997
410'.285'574 — dc21 97-15734
CIP

Contents

Contributors

ANDREW BREDENKAMP is a Senior Research Fellow, who joined the CL/MT group at Essex in October 1993; he studied Modern Languages (specialising in Translation) at the Polytechnic of Central London and is a qualified translator (PgDip) of German, French, Dutch and Italian. He obtained an MA in Linguistics from the University of London (UCL/SOAS), before coming to Essex to write a Ph.D. thesis on nominal anaphora in HPSG. His main interests include Syntax, particularly HPSG, and Machine Translation. He is currently involved in developing industrial prototypes for NLP and MT applications.

GERARD CHOLLET is Director of Research at CNRS (Centre National de la Recherche Scientifique). He is a member of the Signal Department of ENST (Ecole Nationale Superieure des Telecommunications) in Paris since 1983 where he coordinates and conducts research in Automatic Speech Processing, Pattern Recognition and Multimedia Systems. Since 1991, he participates to the development of IDIAP (Institut d'Intelligence Artificielle Perceptive) in Martigny, Switzerland. From the start of the EU research programme in 1981, he has been and is involved in a number of ESPRIT, ACTS, Telematics and COST projects. He studied Physics and Computer Science at the University of Paris and earned a Ph.D. in Computer Science and Linguistics from the University of California, Santa Barbara, in 1978. Gerard Chollet is the author of over twenty book chapters and articles in international journals and holds several patents.

OLIVER CHRIST earned a M.Sc. (diploma) in Computer Science at University of Stuttgart, Germany, in 1992. He worked at the Institute for Natural Language Processing (IMS) of the University of Stuttgart from late 1992 until early 1997 and is the primary author of the IMS Corpus Toolbox (`http://www.ims.uni-stuttgart.de/CorpusToolbox`). He is now with Trados, a software company specialised in translation support tools and terminological databases.

JEAN-LUC COCHARD earned a degree of Engineer in Mathematics in 1983 and a Ph.D. in Computer Science in 1988 at Swiss Federal Institute of Technology, Lausanne, Switzerland. He spent two years as a post-doc at the Centre for In-

Linguistic Databases.
John Nerbonne, ed.

formation Technology Innovation, Montreal, carrying out research in Machine Translation. Since 1991, he is with IDIAP, where he took part to the set-up of the Speech Processing Group. Since 1995, he is leader of this research group and since end of 1996, he is also assistant-director of IDIAP. His first scientific interest is to investigate how the concept of multi-agent systems could be used to include multiple (statistical and non-statistical) heterogeneous knowledge sources in state-of-the-art speech recognition systems.

ANDREI CONSTANTINESCU obtained his degree in electronic engineering and information engineering from the Technical University Munich in 1994. He passed his final year at the AI lab of the Massachusetts Institute of Technology, where he performed research in signal matching through evolutionary algorithms and in automatic signal description, under the guidance of Prof. Patrick Winston. Since spring 1995, he is with IDIAP's Speech group, where he is currently involved among others in the SpeechDat-II project (multilingual telephone speech databases) and in COST 249 (Continuous Speech Over the Telephone). At the same time he is enrolled as Ph.D. student at ENST Paris.

WERNER A. DEUTSCH is head of the Acoustics Laboratory of the Austrian Academy of Sciences. He is lecturer in psychoacoustics and acoustic phonetics at the University of Vienna and served as consultant to several industry and governmental organisations. He has published over 50 papers in psychoacoustics of speech and music, digital audio and signal processing. His current interests are in the area of research and development in auditory perception, man-machine communication, sound analysis and resynthesis and digital audio networks. He currently is involved in HARMONICA, an EC concerted action on music information systems in digital libraries (`http://www.kfs.oeaw.ac.at`).

SIOBHAN DEVLIN received a B.A. (Hons) degree in Classics from the University of Liverpool in 1988. She was awarded an M.Sc. in Computer Based Information Systems from the University of Sunderland in 1993, after which she began her Ph.D. research which is entitled "An investigation into the Use of a Natural Language Paraphrasing Tool as a means of aiding Aphasics in the Reading Process".

ERIK FUDGE studied Mathematics, then Modern Languages at the University of Cambridge. After a period teaching in schools, he re-entered the academic world to research into computational syntax, partly under Fred Householder at Indiana University. This brush with computers led to a decision not to have anything further to do with them: lectureships at Edinburgh and Cambridge led him into the fields of phonology and phonetics. He has held Chairs in Hull and (currently) Reading. Eventually, however, in the mid-1980s, he renounced his earlier vow and embraced the computer again; with Linda Shockey he is now working towards a database of syllable structure.

SYLVIANE GRANGER is Professor of English Language and Linguistics at the University of Louvain (Belgium). She is also director of the university's Cen-

tre for English Corpus Linguistics, a research centre which specialises in the linguistic analysis of learner and bilingual corpora. In 1990 she launched the International Corpus of Learner English project, an international collaborative venture to compile a corpus of writing by learners of English from 13 different mother tongue backgrounds. Her publications include Dictionnaire des Faux Amis Français-Anglais English-French (with J. Van Roey and H. Swallow) (1991), Perspectives on the English Lexicon (ed.) (1991) and Learner English on Computer (ed.) (Addison Wesley Longman, in press).

EDGAR HAIMERL is in charge of the software development within the ALD project and responsible for the technical realisation of the linguistic atlas of Dolomitic Ladin and its neighbouring dialects. He holds Masters in Mathematics, German Philology and Philosophy from the University of Freiburg, Germany, from 1984 to 1986 and a Ph.D. in German Philology from 1990. Since 1991 he is with the ALD team where he designed and wrote various programs for the publication of dialect data (automatic correction of dialect data; generation of phonetic dialect maps; index retrieval system). He is now involved in investigations on automatic taxation and statistical evaluation of dialect data (details at `http://www.sbg.ac.at/rom/people/proj/ald/eh/eh.htm`).

CEDRIC JABOULET obtained a degree of Engineer in Electronics and Information Technology from the ESIEE Engineering School, Paris, France in 1994. Since 1994, he is with the IDIAP's Speech Processing Group where he was involved in the Polyphone project (collection of a large speech database). He is now involved in CAVE (CAller VErification), a European project, dedicated to Speaker Verification for Banking and Telecom applications. His main interests are Speech and Speaker Recognition Over the Telephone.

JUDITH KLEIN studied Information Science and Computational Linguistics at the University of the Saarland. For several years she has been working as a student assistant at the German Research Centre for Artificial Intelligence (DFKI) in Saarbrücken mainly in projects concerned with diagnostic evaluation tools (e.g. TSNLP Test Suites for NLP). Her MA thesis discusses the information system TSDB developed in TSNLP. She will continue her work in the DiET project (Diagnostic and Evaluation Tools for NL Applications).

PHILIPPE LANGLAIS earned a Ph.D. in computer Science in 1995 at the University of Avignon (France), working on integrating Prosody into Speech Recognition. He spent three years (from 1992 to 1995) at the IDIAP Institute, Martigny, (Switzerland), as a researcher in Automatic Speech Recognition. Since 1995, he is working as a researcher at the Computer Science Laboratory of Univ. of Avignon carrying out research in Automatic Natural Language Processing. His main scientific interest is to investigate human-computer communication problems.

JACQUES LE MAITRE is a Professor of Computer Science at the University of Toulon (France) where he manages a research team on multimedia information systems. Previously he was a researcher at the Centre National de la Recherche

Scientifique. He is a database specialist and these last ten years his main works concern the development of functional database languages and textual databases.

SYLVIA MOOSMÜLLER earned a Ph.D. in General and Applied Linguistics at the University of Vienna. Her main field of research has been the (socio)phonological and phonetic variation in Austrian German. In 1993, she was awarded the Sandoz-price for humanities. Since 1992, she has worked at the Acoustics Research Laboratory of the Austrian Academy of Sciences in the fields of acoustic phonetics and speaker recognition.

ELISABETH MURISASCO is an Assistant Professor in Computer Science at the University of Toulon (France) for four years. Its current research topics are textual databases and distributed hypertext databases. She is member of a research team working on multimedia information systems. Previously, she prepared a Ph.D. on the development of an object manager for a functional database programming language.

JOHN NERBONNE earned an M.S. in Computer Science and a Ph.D. in Linguistics at Ohio State University. He worked at Hewlett-Packard Labs for five years and the German AI Center (Saarbrücken) for three years before becoming Professor of Computational Linguistics and Head of Humanities Computing in Groningen. He is the author of over thirty articles on computational linguistics and is currently the coordinator of the GLOSSER project, which applies morphological analysis to computer-assisted language learning. Nerbonne is Chair of the European Chapter of the Association for Computational Linguistics 1997–98.

KLAUS NETTER is a senior researcher and project manager in the Language Technology Lab at the German Research Center for Artificial Intelligence (DFKI GmbH) in Saarbrücken. He studied theoretical linguistics, German and English at the Universities of Munich and Reading (GB) and earned a Ph.D. in Computational Linguistics at the University of the Saarland. He previously worked at the University of Munich and at the Institut für Maschinelle Sprachverarbeitung (IMS) at the University of Stuttgart, where he focussed on Machine Translation (and other applications) in the framework of Lexical Functional Grammar (LFG). His basic research areas comprise computational grammar development, lexicon and morphology, with a specific focus on German syntax and constraint-based grammar frameworks, such as HPSG or TAG. He has also been involved in several projects and activities on the testing and evaluation of NLP components and applications and is currently also working on projects in the area of information retrieval.

ANTON NOLL is a computer scientist and electronic engineer. He currently is studying statistics and informatics. He has worked at the Austrian Academy of Sciences since 1982 where he leads the hardware and software development of the S$_T$OOLS acoustics workstation.

STEPHAN OEPEN has studied linguistics, computer sciences, and Russian at Berlin and Volgograd. From 1992 to early 1997 he worked as a staff scientist at DFKI Saarbrücken (the national AI research institute in Germany) in several research projects including a large-scale NL dialogue system, a state-of-the-art HPSG grammar engineering environment (the DFKI PAGE system), grammar writing for German and English, and the construction and use of generic NLP test suites. He recently joined Saarbrücken University as a lecturer in the Computational Linguistics department to complete a dissertation project on adequate performance models for HPSG-type grammars.

MONIQUE ROLBERT is an Assistant Professor at the University of Aix-Marseille III (France) and makes her research in the Laboratoire d'Informatique de Marseille. She works in the field of Natural Language Processing and more particularly on pronouns and discourse treatment.

LOUISA SADLER has taught theoretical and computational linguistics at the Uni verity of Essex since 1986, and was previously at the University of East Anglia. She has been involved in a number of externally funded projects in machine translation and natural language processing. Her research interests include HPSG and LFG syntax, the syntax of the Celtic languages and argument structure as well as computational linguistics.

LINDA SHOCKEY holds a Ph.D. in Linguistics from Ohio State, and she lectures in that discipline at the University of Reading. She has worked on a variety of computer-based linguistic issues (speech recognition, text-to-speech synthesis): the syllable database represents a new direction for her research.

GARY SIMONS is director of the Academic Computing Department at the international headquarters of the Summer Institute of Linguistics in Dallas, Texas. Prior to taking up this post, he did field work with SIL in Papua New Guinea (1976) and the Solomon Islands (1977–1983). In 1979 he received a Ph.D. in general linguistics (with minor emphases in computer science and classics) from Cornell University.

ANDREW SPENCER has taught linguistics at the University of Essex since 1990. His research interests have included phonological theory (especially Slavic phonology), child phonology, and morphology. He is the author of Morphological Theory (1991) and Phonology (1996) both published by Blackwells. He is currently working on an extension of a database of Russian verbs and their nominalizations relating lexical semantics to argument structure and morphology.

JOHN TAIT received his B.Sc. (Hons) degree in Computing Science from the University of Essex in 1976 and his Ph.D. from the University of Cambridge in 1983. Following further research there, he pursued an industrial career in a wide variety of organisations. He was appointed as a Senior Lecturer at the University of Sunderland in 1991, and Reader in Intelligent Information Systems in 1995. Dr. Tait's main research interests are in natural language

engineering, especially as applied to information retrieval and management, and support for the disabled. Dr. Tait is a Joint Editor of the Cambridge University Press journal Natural Language Engineering.

JOHN THOMSON has been on staff with the Academic Computing Department of the Summer Institute of Linguistics (Dallas, Texas) since 1987. Most recently he has served as the lead programmer on the team that developed CELLAR. Prior to that he developed two widely-used programs for the Macintosh: Conc (a concordance builder) and IT (an interlinear text processor). In 1985 he received a Ph.D. in computer science from Monash University (Melbourne, Australia).

MARTIN VOLK is a lecturer and research assistant in the Computational Linguistics group at the University of Zurich. He holds a Masters in Artificial Intelligence from the University of Georgia and a Ph.D. in Computer Science from the University of Koblenz. His main research interest are in linguistic engineering focusing on grammar engineering and evaluation techniques, as well as structured document processing. (Details under: `http://www.ifi.unizh.ch/staff/volk.html`)

RALF VOLLMANN studied Linguistics, Tibetan Studies and Ethnology at the Universities of Graz and Vienna. He received a M.A. degree in linguistics in 1991. He worked at the Austrian Academy of Sciences in sociophonetics, sociophonology, morphology, first language acquisition and database design. He is now working at the University of Graz on an electronic and paper edition of Wilhelm von Humboldt's Linguistic Writings.

1

Introduction

JOHN NERBONNE

This is a selection of papers on the use of databases in linguistics. All of the papers were originally presented at a conference entitled "Linguistic Databases", held at the University of Groningen on March 23–4, 1995. This introduction reviews the motivation for a special examination of linguistic databases, introduces the papers themselves, and then, briefly, suggests both how the knowledge is useful to working linguists, as well as how databases might evolve to be used for linguistic data more easily.

Motivation

Linguistics is a data-rich study. First, there is a great deal of purely linguistic data. Linguistics sets itself the task of describing and analyzing the structure of language as this is evidenced in billions of speakers, each making hundreds of utterances a day for linguistic lifetimes of several decades. Factors important to the structure of these utterances include thousands of languages, each with tens to hundreds of thousands of words, which in turn may be found in dozens to hundreds of different word forms. The results of linguistic analysis show great variety in the particular coding rules at all levels—sounds, words and phrases. The

The editor wishes to thank the program committee, Tjeerd de Graaf, Tette Hofstra and Herman Wekker for scientific and organizational advice; Duco Dokter and Edwin Kuipers, who handled local arrangements; and Sietze Looyenga, who managed finances.

The conference was supported financially by grants from the Dutch National Science Foundation (NWO), Royal Dutch Academy of Science (KNAW), the Groningen Center for Language and Cognition (CLCG), and the Behavioral and Cognitive Neurosciences Graduate School, Groningen (BCN). Our thanks to all of these organizations. The editor further wishes to thank the Japanese Ministry of Post and Telecommunications, who sponsored a visit during which this introduction was written.

Linguistic Databases.
John Nerbonne, ed.

rules are not individuated clearly enough to allow reliable comparisons, but they seem to number in the thousands in current formulations. The many factors studied in linguistic variation—including geography, sex, social and educational status, pathology, and situational "register" (as for telegrams, graffiti, etc.)—only add to this embarrassment of riches.

Second, various subfields of linguistics have developed experimental methodologies involving a great deal of data which, although not purely linguistic, are used crucially in linguistic theorizing or in applications. Some examples of these would be physical measurements such as air pressure and pitch, records of systematic errors in production and apprehension (in acquisition and pathology), the quasi-linguistic ill-formed examples of generative grammar, or psychological measurements such as reaction time or the movements of eyes in reading.

Third, applications of linguistics need to keep track of further data categories such as successful (and unsuccessful) processings, degree of ambiguity in results, use of particular knowledge sources and heuristics, user reactions, and comparisons among alternative system configurations.

Tasks of Databases

Given this amount of data, it is not surprising that a good number of linguists have investigated software for managing data. Databases have long been standard repositories in phonetics (see Liberman 1997, UCLA Phonetics Laboratory 1996) and psycholinguistics (see MacWhinney 1995) research, but they are finding increasing further use not only in phonology, morphology, syntax, historical linguistics and dialectology but also in areas of applied linguistics such as lexicography and computer-assisted language learning. Normally, they serve as a repositories for large amounts of data, but they are also important for the organization they impose, which serves to ease access for researchers and applications specialists.

The term 'database' refers to collections of electronic records of linguistic data. As a simple example, this might be a file or set of files of sentences. Furthermore, a database records data in a DECLARATIVE form, i.e. independent of the particular procedures needed to interpret, display or modify it. This means that the creators and maintainers of databases have avoided storage forms like the files of word-processing packages, which rely on extensive programs for display, etc.

Similarly, something like a parser or generator of sentences is not a database, even if it could be argued to contain the same information. Even if that were so, these forms are by definition abstract and require either a special procedural interpretation, in which case they fail to be

declarative, or, in the case of logic grammars, they require sophisticated inference if they are to be used as characterizations of data.[1] The grammars used in parsers and generators are hypotheses about data, and thus serve a completely different purpose from databases—even if ideal grammars would characterize the same information. But in fact the information is never the same—the data is not exactly as even the best hypotheses predict.

Furthermore, a database attempts to ensure the CONSISTENCY or INTEGRITY of data by eliminating redundant specifications. By eliminating the chance of double specification, we reduce the chance of inconsistent specification. Oepen et al. (below) discuss this requirement in more detail. To give a simple example, if one wished to record the finite verb (of the main clause) in a database of sentences, this would NOT best be done by recording it separately (as 'works' in 'Dan works.') since this would introduce the possibility of inconsistency (one might write 'work' or 'wrok'). A preferable solution is to record an index ('2nd'), which is also required to be less than sentence length ($0 < i \leq \mathit{length}$). Of course, database theory, a vibrant branch of computer science, is only suggested by the remarks above. Further issues arise in making data entry and retrieval easy and fast, in organizing data description (metadata), and in adding logical power (deductive db's) or flexibility (object-oriented db's). Brookshear (1997) contains an excellent introductory chapter on databases—including various data models; and Ullman (1988) and Date (1995) are very popular textbooks, both focusing on relational databases.

The point of the remarks above has been to clarify what's meant by 'database'. If they suggest that the main issues facing the developers and users of linguistic databases today are conceptual, then they are misleading. The most important issues are overwhelmingly practical, as the call for papers suggested. We turn now to a review of the points suggested from the CFP, which called for a forum on the exchange of information and views on the proper use of databases within the various subfields of linguistics. We add notes to each of the points below, suggesting how the conference answered the questions raised. Of course, these are (subjective) assessments.

1. Databases vs. annotated corpora, pros and cons.
   The tendency was less to ask which is correct, and more to ask how we can have both. LeMaitre et al. and Volk present schemes

[1] Since the data is always in some sense encoded—there is no direct representation of letters or sounds in electronic circuitry—there are always procedures involved when we look at or otherwise work with the data. But the encoding schemes used are standard and very simple.

for viewing annotated corpora as databases. Oepen et al., while adducing advantages of databases, identify the relationship to corpora as a future goal.

2. Needs with respect to acoustic data, string data, temporal data. Existing facilities.
   As the papers by Chollet et al. and Deutsch et al. demonstrate, phonetic databases are already multimedia (and must be), and as Simon and Thompson, Oepen et al. show, string handling is highly desirable in language software and syntactic databases, respectively.
3. Developing (maximally) theory-neutral db schemas for annotation systems.
   The general consensus was that this issue was interesting, but not pressing---as the example of the Penn Treebank (Black et al., 1991) showed, even very theory specific annotations are useful. One can use them as comparisons in those parts of analysis that seem right, and they are to some degree translatable into other systems. Even though generality is to be preferred, the work cannot wait until perfect schemes arise.
4. Commercially available systems v. public domain systems. What's available?
   While some developers were at pains NOT to rely on commercial packages (hoping to keep the threshold low for users), this was not general. The down-side of this decision is of course the extra development effort (and its duplication). Deutsch et al. and Haimerl discuss packages they use.
5. Uses in grammar checking, replication of results.
   This is a focus in the papers by Volk and Oepen et al.
6. Needs of applications such as lexicography.
   While there was unfortunately no representation from lexicography, Granger and Devlin and Tait present other applications, and applications are discussed in the papers by Oepen et al., Volk, Deutsch et al. and Chollet.
7. Making use of CD-ROM technology.
   Again, little focused discussion even though it figures in the work of Chollet et al. and in the work of the data collection organizations (see following point).
8. The existing professional expertise: Linguistic Data Consortium

(LDC),[2] Text Encoding Initiative (TEI) (see Sperberg-McQueen and Burnard, 1994).

Here we find not only a new organization, the European Language Resources Association (ELRA),[3] but also a very general consensus on the importance of coordinating the overwhelming amount of work being done and yet to be done.

There was very little discussion about a question often heatedly raised in recent meetings between theoreticians and applications specialists: what sorts of data should be compiled and organized—"natural data" versus other. Perhaps this should have been unsurprising since the scope of the meeting was much broader than theoretical linguistics, the only subfield in which a RESTRICTION to intuitive data is (sometimes) upheld. Bredenkamp, Sadler and Spencer's contribution demonstrates the value of databases for researchers using intuitive data in generative theory, however, and they, Volk, and Oepen et al. all employ databases for intuitive data. Those references should be enough to correct the misconceptions that the use of databases is tied to particular views of what linguistic data must be or that their use is incompatible with including data about ill-formedness.

On the side of those sceptical about the "natural data" of the data debate, it is worth noting the example of the speech community, who progressed for many years while focusing on the artificial data of speech read from texts with pauses between words. None of this may be interpreted to mean that there was consensus about the status of intuitive vs. natural data, but only that there was no sense of resolving the issue for either side.

A further distinction in the use of databases was deliberately left underspecified in the CFP, and that was the distinction between data and hypotheses (as databases may organize them). Electronic dictionaries and NLP lexicons are in some sense databases, but they are (largely) not databases of data directly, but rather linguistic descriptions of data. Lexicography was mentioned in the CFP in order to solicit contributions from those using database technology to organize linguistic hypotheses. The papers by Chollet et al. and by Deutsch et al. describe systems for the handling of data AND hypotheses in phonetics, and the (unpublished) talks by Gebhardi and Sutcliffe et al. examine the application of database technology to managing lexical information. There was, how-

[2]See "Introduction to the Linguistic Data Consortium" on the World-Wide Web. There's a pointer from the LDC's homepage at `http://www.cis.upenn.edu/~ldc`

[3]ELRA's director promises that its address shall be ELRA / ELDA, 87, Avenue D'ITALIE, 75013 PARIS, FRANCE, email: `elra@calvanet.calvacom.fr`

ever, a suggestion that the management of hypotheses (or knowledge bases) puts harsher demands on the need to be able to revise (even though all databases must allow changes).

Papers

In this section we attempt to summarize the individual papers and assess their contributions.

Syntactic Corpora and Databases

Although these papers all arise from the natural language processing field, it is easy to see their relevance to the non-applied, pure science of syntax. There are certainly signs of theoreticians turning to corpora for evidence (Dalrymple et al. 1997).

TSNLP—**Test Suites for Natural Language Processing** by Stephan Oepen , Klaus Netter and Judith Klein presents a detailed methodology for building a test suite of linguistic examples to be used in evaluating the linguistic coverage of natural language processing systems, and then reports on a large-scale implementation of such test suites for three languages. The authors motivate the development of hand-built test material to complement corpus-based evaluation tools, also suggesting design criteria for test suites. They propose a concrete annotation scheme which tries to remain as neutral as possible with respect to particular linguistic theories. They discuss the broad range of constructions in their test material, and also their portable software for storage and retrieval. In a final section they report on a connecting the test suite to an HPSG grammar and parser.

Markup of a Test Suite with SGML by Martin Volk begins with the Standard Generalized Markup Language (SGML), an international standard which enjoys growing popularity in textual software. Its developers described SGML as "a database language for text", suggesting that it ought to be up to the task of declaratively representing a test suite--an annotated corpus of test sentences, both grammatical and ungrammatical. Volk shows that SGML is indeed suited to this task (discussing five different models) and chooses a model which minimizes redundancy by taking into account the inheritance of syntactic features. The article concludes that SGML is advantageous, but that an optimal inheritance mechanism is still lacking.

From Annotated Corpora to Databases: The SgmlQL Language by Jacques LeMaitre, E. Murisasco and Monique Rolbert describes the SgmlQL query language for SGML documents. This language is an extension of SQL, the standard query language for relational databases. It allows for the retrieval of information from SGML docu-

ments and for the modification of markup in SGML documents. The retrieval and modification commands are sensitive to SGML structure and they are made flexible through extensions using regular expressions. The applicability of SgmlQL to various tasks in NLP is discussed.

Phonetic Databases

An Open-Systems Approach for an Acoustic-Phonetic Continuous Speech Database by Werner Deutsch, Ralf Vollmann, Anton Noll and Sylvia Moosmüller was developed to support the design and evaluation of speech recognition. It is a database of acoustic files with a library of database routines to support entry, annotation, editing and update, and retrieval as well as specialized functions for speech such as segmentation, fast-fourier transformations, fundamental and formant frequency extraction, and linear-predictive coding. It is described here as applied to databases of Austrian German (still under construction) and child language. Its user-interface is implemented as a hypertext system, and, although the work to-date has been confined to the PC platform, the developers are confident of its portability.

Swiss French PolyPhone and PolyVar: Telephone Speech Databases to Model Inter- and Intra-Speaker Variability by Gerard Chollet, J.-L. Cochard, A. Constantinescu, C. Jaboulet and Ph. Langlais reports on the construction of PolyPhone, a database of the telephone speech of 5,000 speakers of Swiss French. The current report focuses on (i) the sampling discipline used, especially how the phonetic material solicited was varied in an effort to minimize collection needed (while guaranteeing a sufficient amount of material in each category); (ii) the transcription of material, including an interface built for this purpose; (iii) labeling procedures, including some automated support. PolyPhone and PolyVar are instances of speech databases being distributed to the speech research community by the LDC and ELRA.

Applications in Linguistic Theory

Three papers demonstrate applications to questions in linguistic theory.

A Database Application for the Generation of Phonetic Atlas Maps by Edgar Haimerl describes CARD, a system for managing phonetic data to be used to create dialect atlas maps. CARD is a front end to a fairly large (350K entries) PC database (XBase) of data gathered for Salzburg's ALD project (Dolomitic Ladinian). Although CARD handles the basic management of the survey data (input, update, and retrieval), its particular strength lies in its capacity to produce printed maps for arbitrary selections of the data. Because users need to build atlas maps displaying a variety of different features, flexibility is of the

utmost importance. CARD achieves this by using the database itself for meta-information. Most application details are stored in database files instead of being hard-coded into the program, and indexes and new databases are created on the fly as the user configures an application. Such techniques help the lexicographer tailor the programme to accommodate specific requirements vis-a-vis selection of data, order of multiple entries, physical appearance of maps, and compound (accented) characters. CARD is expected to be made available when the ALD is published on CD-ROM.

The Reading Database of Syllable Structure by Erik Fudge and Linda Shockey describes a database of syllable structures. The aim of the work is to allow the user to query whether certain syllable types or tokens occur in a language, or indeed to identify languages in the database for which a specified sequence is possible. Thus the database attempts to go beyond the databases such as the UCLA Phonological Segment Inventory to include the possible combinations of these segments within words, i.e., the phonotactic characteristics of the language. In this paper, the authors report on work in progress. They set up a general database for phonotactic statements for (at the moment) more than 200 languages. Queries supported include examples such as 'Is [qi] a possible syllable in a certain language or language family?' or 'What percentage of languages allow onsetless syllables?'

Investigating Nominal Argument Structure: the Russian Nominalization Database by Andrew Bredenkamp, Louisa Sadler and Andrew Spencer focuses on the interplay of aspect and argument structure in deverbal nominalization. Following an introduction to aspect and argument linking in syntactic theory and a sketch of Russian verbal morphology, the authors review Grimshaw's distinction between complex event nominalizations on the one hand, and result nominalizations and simple event nominalizations on the other, in which only complex event nominalizations are claimed to have argument structure. Russian is selected to test the hypothesis because some tests rely on aspect, which is overtly marked in Russian. To organize the relevant data, the authors have organized it into a database which contains 2,000 Russian verbs and their corresponding nominalizations. In the last section of the paper, the authors discuss a first result: while underived imperfective verbs and simplex and derived perfective verbs give rise to both complex event nominalizations and result and simple event nominalizations, derived imperfective verbs give rise to complex event nominalizations only. This had not been noticed in purely theoretical work.

Applications

The Computer Learner Corpus: A Testbed for Electronic EFL Tools, an invited paper by Sylviane Granger, reports on progress in language instruction based on the International Corpus of Learner English, a corpus collected from the (of course imperfect) language of learners. Earlier improvements in language instruction had resulted from the careful use of native-speaker corpora, e.g., more attention to frequent words in introductory texts. The present report demonstrates the use of corpora in identifying common errors and in assessing the worth of computerized tools, especially grammar checkers, for language learners—where there seems to be a need for more specialized software. The paper concludes with a plea for the adaptation of existing software to this application area.

The Use of a Psycholinguistic Database in the Simplification of Text for Aphasic Readers by Siobhan Devlin and J. Tait reports on the implementation of a program to simplify text which makes use of both the Oxford Psycholinguistic Database as well as WordNet, a dictionary database structured semantically. The program is designed to replace difficult words by more common synonyms. The Oxford Psycholinguistic Database contains information on frequency, concreteness and other measures which contribute to the difficulty of words, and WordNet contains information on synonyms. The program is intended for use in testing what factors improve the ease with which aphasics can read.

Extending Basic Technologies

Two papers provide suggest some of the directions in which linguistic databases of the future may lie.

Linking WordNet to a Corpus Query System by Oliver Christ addresses the general problem of increasing the utility of text corpora. The strategy proposed is to consult knowledge bases external to the corpus to allow more precise queries. The author notes that the standard approach to annotating corpora is to add static attributes to the word forms, and that his proposal differs in that it defines "dynamic attributes" which query external sources to compute more annotations on-the-fly. The approach is exemplified by showing how a dynamic link to the WordNet thesaurus makes it possible to constrain searches on a text corpus by semantic information computed from WordNet for the word forms.

Multilingual Data Processing in the CELLAR Environment, an invited paper by Gary F. Simons and John V. Thomson addresses a problem which researchers have wished would go away more than

they have tried to solve: the atavistic "monolinguality" of the computer world. CELLAR is a "comprehensive environment for linguistic, literary and anthropological research", including innovative ideas on what is needed in linguistic databases, but, at the urging of the program committee, the authors agreed to focus here on the multilingual issues in system design. In spite of encouraging recent improvements (mostly in displaying foreign texts), we still have trouble with inputing from the keyboard, spell-checking, sorting (e.g., in alphabetic order), searching, embedding within other languages, or cross-referencing to elements (alignment, as used in glossing or parallel texts). A fundamental requirement within CELLAR is that each string data element must be marked for the language it is written in. Multilingualism extends beyond data: the user interface is likewise multilingual, so that menus and button labels, error messages etc. are available in several languages.

Going on

How can the information here be useful to working linguists? Most directly, of course, if the papers here provide some ideas on how to get data and on how to organize what data you have so that you and others can get the most out of it. The organizations to turn to have been mentioned above: LDC, TEI and ELRA.

More databases in general:

```
http://www.cis.ohio-state.edu/hypertext/faq/usenet
        /databases/free-databases/faq.html
```

is a catalogue of free database systems.

The Conference

There were several presentations at the conference which do not appear below. The novelty of the topic made it was impossible to be certain beforehand whether there would be work of sufficient interest and quality to warrant publication. The original CFP therefore explicitly refrained from promising publication, but the meeting demonstrated that this would be valuable. Unfortunately, some authors had commitments elsewhere, etc. which prevented their contributions from being included here. Undoubtedly the collection would be even more valuable if more of the conference contributions below could have been included:

Susan Armstrong (ISSCO, Geneva) and Henry Thompson (Edinburgh) A Presentation of MLCC: Multilingual Corpora for Cooperation

Dietmar Zaefferer (Munich) Options for a Cross-Linguistic Reference Grammar Database

Masahito Watanabe (Meikai, Yokohama) A Better Language Database for Language Teaching

Gunter Gebhardi (Berlin) Aspects of Lexicon Maintenance in Computational Linguistics

Richard Sutcliffe et al. (Limerick) From SIFT Lexical Knowledge Base to SIFT Lexical Data Base: Creating a Repository for Lexicological Research and Development

Pavel A. Skrelin (St. Petersburg) The Acoustic Database for Studying Language Changes

Kamel Bensaber, Jean Serignat, and Pascal Perrier (ICP, Grenoble) BD_ART: Multimedia Articulatory Database

Lou Boves and Els den Os (SPEX, Leidschendam) Linguistic Research using Large Speech Corpora

In particular, the conference benefited from the presentations of Jan Aarts and Mark Liberman, who were invited to speak on their decades of accomplishments in this area:

Jan Aarts Prof. of English, Nijmegen, leader of the TOSCA, and Linguistic Databases projects: "Annotation of Corpora: General Issues and the Nijmegen Experience"

Mark Liberman Prof. of Linguistics & Computer Science, University of Pennsylvania and Director, Linguistic Data Consortium "Electronic Publication of Linguistic Data"

Finally, there were demonstrations by Judith Klein (TSNLP), Gunter Gebhardi (LeX4), Edgar Haimerl (ALD), Werner Deutsch (S_TOOLS) and Hans van Halteren (LDB). The TSNLP, ALD and S_TOOLS work is reported in the papers (coauthored) by the demonstrators.

References

Black, E., S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jellinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. In *Proc. of the Feb. 1991 Speech and Natural Language Workshop.* San Mateo: Morgan Kaufmann.

Brookshear, G. 1997. *Computer Science: An Overview.* Reading, Massachusetts: Addison-Wesley. 5th edition.

Dalrymple, M., M. Kanazawa, Y. Kim, S. Mchombo, and S. Peters. 1997. Reciprocal Expressions and the Concept of Reciprocity. *Linguistics and Philosophy.* accepted for publication.

Date, C. J. 1995. *An Introduction to Database Systems.* Reading, Massachusetts: Addison-Wesley. 6th edition.

Liberman, M. 1997. Introduction to the Linguistic Data Consortium. available via the LDC site: `http://www.ldc.upenn.edu/`.

MacWhinney, B. 1995. *The CHILDES Project: Tools for Analyzing Talk.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Sperberg-McQueen, C. Michael, and L. Burnard. 1994. *Guidelines for Electronic Text Encoding and Interchange.* Oxford and Chicago: Text Encoding Initiative. Version 3 (first publicly available version).

UCLA Phonetics Laboratory. 1996. The Sounds of the World's Languages. See `http://www.humnet.ucla.edu/humnet/linguistics/faciliti/`.

Ullman, J. D. 1988. *Principles of Database and Knowledge-Base Systems.* Rockville, Maryland: Computer Science Press.

2

TSNLP — Test Suites for Natural Language Processing

STEPHAN OEPEN, KLAUS NETTER, & JUDITH KLEIN

1 Introduction

Given the growing number of natural language applications, there is an increasing demand for reference data other than the type of data found in large collections of text corpora. Test suites have long been accepted in the NLP context, because they provide for controlled data that is systematically organized and documented. However, with increasing and realistic applications of NLP technology a generation of general-purpose test suites will be required that (i) have broad coverage, (ii) are multilingual, and (iii) incorporate consistent and highly structured linguistic annotations. Linguistic databases of this kind will not only facilitate the deployment of test suites as diagnostic tools, but as well support different kinds of evaluation and serve as repositories for developers.

The objective of the TSNLP project[1] is to construct test suites in

Most of the TSNLP results presented originate from joint research of the TSNLP consortium; besides, many of the underlying ideas — especially relating to the kind of linguistic resource and type of annotations required — and the initial project stimulus are due to to Hans Uszkoreit.

The authors are most grateful for the constructive cooperation and acknowledge the contributions of the partners Doug Arnold, Lorna Balkan, Frederik Fouvry, and Siety Meijer (University of Essex, UK), Dominique Estival, Kirsten Falkedal, and Sabine Lehmann (ISSCO, Geneva, Switzerland) and Eva Dauphin, Veronika Lux, and Sylvie Régnier-Prost (Aerospatiale, France). Major parts of the test data construction and implementation work at DFKI have been and still continue to be carried out by diligent research assistants; the authors gratefully appreciate the enthusiasm of Judith Baur, Tom Fettig, Fred Oberhauser, and Andrew P. White.

[1]The project was started in December 1993 and completed in March 1996; the consortium is comprised of three academic partners — viz. the University of Essex (UK), the Istituto Dalle Molle per gli Studii Semantici e Cognitivi ISSCO (Geneva,

Linguistic Databases.
John Nerbonne, ed.

three different languages building on a common basis and methodology. Specifically, TSNLP addresses a range of issues related to the construction and use of test suites. The main goals of the project are to:

- provide guidelines for the construction of test suites to facilitate a coherent and systematic data collection;
- develop a rich annotation schema that is maximally neutral with respect to particular linguistic theories or specific evaluation and application types (section 2.1);
- construct substantial test fragments in English, German, and French that are applicable to different NLP application types (viz. parsers, grammar and controlled language checkers) (section 2.2);
- design and implement an extensible linguistic database to store, access, and manipulate the test data (section 2.3); and to
- investigate methods and tools for (semi-) automated test suite construction.

Both the methodology and test data developed currently are validated in a testing and application phase (section 2.4).

To ensure the wide distribution and dissemination of TSNLP test suites, the results of the project will be made available to the public domain. Ideally, the TSNLP annotation schema and database technology will serve as a starting and reference point for future test suite efforts, thus for the first time allowing the interchange of test data and the comparison of test and diagnostic results.

In the present paper the authors take the opportunity to present some of the recent outcome of TSNLP to the community of language technology developers as well as to potential users of NLP systems. Accordingly, the presentation puts emphasis on practical aspects of applicability and plausibility rather than on theoretically demanding research topics; the TSNLP results presented are of both methodological and technological interest.

Switzerland), and DFKI Saarbrücken (Germany) who have strong backgrounds in machine translation, evaluation, and natural language processing respectively — and the Common Research Center of Aerospatiale (Suresnes, France) as an industrial partner.

Most of the project results (documents, bibliography, test data, and software) as well as on-line access to the test suite database (see section 2.3) can be obtained through the world-wide web from the TSNLP home page `http://tsnlp.dfki.uni-sb.de/tsnlp/`.

The TSNLP project was funded within the Linguistic Research Engineering (LRE) programme of the European Commission under research grant LRE-62-089; special thanks to Dan Flickinger and John Nerbonne for the constructive peer reviews and Roger Havenith of the CEC.

1.1 Test Suites vs. Text Corpora

Test suites consisting of systematically constructed individual test items are not the only source of reference data for testing and evaluating NLP applications. In particular they clearly cannot replace test corpora drawn from naturally occurring texts but rather have to be seen as serving different and complementary purposes. Among the key differences between test suites and corpora are the following (see Balkan et al. 1994 for more details):

- **Control over test data:** While test suites allow the construction of focussed test data using small vocabulary and illustrating specific phenomena in isolation or in controlled interaction, corpora will always comprise arbitrary combinations of different phenomena. Although the combinations of phenomena found in corpora may be empirically relevant, they do not allow for a focussed and fine-grained diagnosis of system performance.
- **Systematic coverage:** Test suites can provide test items which illustrate systematic variations over a specific phenomenon. In a corpus such variations are likely to occur only accidentally and they will not be related to each other.
- **Non-redundant representation:** While well-constructed test suites will list every phenomenon as concisely as possible, corpora are clearly redundant, in that one and the same phenomenon can occur over and over again.
- **Inclusion of negative data:** Ungrammatical examples do not occur naturally (or occur unsystematically) in corpora. However, for diagnostic purposes testing for negative data may be relevant (especially, for example, with grammar and controlled language checkers).
- **Coherent annotation:** As is exemplified in section 2.2, manually constructed test data can be associated with different kinds of systematic and coherent annotations. With few exceptions such annotations are hard to find in combination with corpora.

For most of these aspects it could be argued that, for example, the selection and combination of phenomena and the degree of variation within some particular phenomenon as they are found in corpora reflect what is actually relevant in the real world. However, corpora — regardless of their size — will only give a more or less representative sample and do not allow for a systematic diagnosis. Ideally, test suites and corpora should therefore stand in a complementary relation, with the former building on the latter wherever possible and necessary (section 3

briefly sketches the plans for future research on relating test suites and text corpora to each other).

1.2 TSNLP Objectives

The TSNLP project aims to construct test suites that address all of the desiderata listed in section 1. There is a special emphasis on the collection of minimal sets of test items reflecting the systematic variation of a single parameter while others remain controlled.

In order to encourage the distribution and assessment of TSNLP results, the annotation schema and selection of test data has to be maximally neutral with respect to a particular theory of grammar or individual languages. The test data construction and database technology developed shall encourage developers and users of NLP technology to tailor the test suites to their needs and to extend the database with user or application specific information.

2 Results of TSNLP

In both test data construction and database design issues TSNLP could — among others — build on experiences from the DiTo project previously carried out at DFKI (Nerbonne et al. 1993). The DiTo effort produced some 1500 systematically constructed and annotated sentences for four syntactic phenomena of German; the test data were organized into a simple relational database.[2]

The annotation schema and parts of the data developed in DiTo (for German at least) served as a starting point for work done in TSNLP. At the same time, however, the DiTo project had revealed two important problems that prompted the TSNLP consortium to pursue more elaborate approaches:

(i) Data construction in DiTo was directly carried out in the internal format defined by the database: the test data authors had to know and understand some of the database internals and produce the appropriate format; the method, obviously, is inefficient, laborious, and error-prone.

(ii) Although the interpreted pattern matching language `awk(1)` as the DiTo query engine provided for high-level string manipulation facilities, the hybrid combination with Un*x scripts and `yacc(1)` programs imposed severe limitations on efficiency as well as interfaces, portability, and simplicity.

[2]Basically, the database software was implemented through the Un*x commands `sort(1)`, `uniq(1)` and `join(1)` combined with a simple query engine (an `awk(1)` script) and a query processor (realized in `yacc(1)` and `lex(1)`).

Building on the experiences from the DiTo project, TSNLP put strong emphasis on tool building — to support test data construction independent of the database internal representation — as well as on scalable database technology. Sections 2.1, 2.2, and 2.3 present the TSNLP strategy in test data construction and database design respectively. Finally, in section 2.4 some initial results of the ongoing testing phase of test data and methodology are reported.

2.1 The TSNLP Annotation Schema

Based on a survey of existing test suites (Estival 1994a) and the types of annotations (if any) found, a detailed annotation schema has been designed which neither presupposes some token linguistic theory nor a particular evaluation type nor a limited range of suitable applications (see Estival et al. 1994b for a complete presentation).

Test data and annotations in TSNLP test suites, basically, are organized at four distinct representational levels:

- **Core data:** The core of the data collection are the individual *test items* (sentences, phrases et al.) together with all general, categorial, and structural information that is independent of phenomenon, application, and user parameters. Besides the actual string of words, annotations at this level include (i) bookkeeping records (date, author, origin, and item id), (ii) the item format, length, category, and wellformedness code, (iii) the (morpho-)syntactic categories and string positions of lexical and — where uncontroversial — phrasal constituents, and (iv) an abstract dependency structure. Encoding a dependency or functor-argument graph rather than a phrase structure tree avoids the need to impose a specific constituent structure but still can be mapped onto one (see section 2.4.3).
- **Phenomenon related data:** Based on a hierarchical classification of *phenomena* (e.g. verb valency being a subtype to general complementation) each phenomenon description is identified by its phenomenon identifier, supertype(s), interaction with other phenomena, and presupposition of such. Moreover, the set of (syntactic) parameters that is relevant in the analysis of a token phenomenon (e.g. the number and type of complements in the case of verb valency) is determined. Individual test items can be assigned to one or several phenomena and annotated according to the respective parameters.
- **Test sets:** As an optional descriptive level in between the test item and phenomenon levels, pairs or sets of grammatical and ill-formed

Test Item

item id: *24033*	author: *dfki-klein*	date: *jan-94*
register: *formal*	format: *none*	origin: *invented*
difficulty: *1*	wellformedness: *1*	category: *S_v2*
input: *Der Manager sieht den Präsidenten*		length: *5*
comment:		

position	instance	category	function	domain
0:2	*Der Manager*	*NP_nom-sg*	*subj*	*2:3*
2:3	*sieht*	*V*	*func*	*0:5*
3:5	*den Präsidenten*	*NP_acc-sg*	*obj*	*2:3*

Phenomenon

phenomenon id: *24*	author: *dfki-klein*	date: *jan-94*
name: *C_Complementation*		
supertypes: *Complementation*		
presupposition: *C_Agreement, NP_Agreement*		
restrictions: *neutral*	interactions: *none*	purpose: *test*
comment:		

FIGURE 1 Sample instance of the TSNLP annotation schema for one test item: annotations are given in tabular form for the *test item*, *analysis*, and *phenomenon* levels. The *function* and *domain* attributes encode the abstract dependency structure using (zero-based) substring positions to refer to parts of the test item.

items can be grouped into *test sets*. Thus, it can be encoded how, for example, test items that originate from the systematic variation of a single phenomenon parameter relate to each other.

- **User and application parameters:** Information that will typically correlate with different evaluation and application types in the use of a test suite is factored from the remainder of the data into *user & application profiles*. Currently, there are foreseen (i) a centrality measure on a per phenomenon, test set, or even test item basis (e.g. user x may in general consider phenomenon y to be central but test set z to be marginal within y) and (ii) judgments of relevance with respect to a specific application. To match the results obtained from a token NLP system (a parser, say) against a test suite, a formal or informal output specification can be added at the profile level.

In addition to the parts of the annotation schema that follow a more or less formal specification, there is room for textual comments at the various levels in the above hierarchy to accommodate information that cannot be formalized.

Phenomenon	English	French	German	Total
C_Complementation	117 | 845	225 | 639	218 | 246	560 | 1730
C_Agreement	68 | 55	104 | 183	224 | 175	396 | 413
C_Modification		329 | 63		329 | 63
NP_Complementation		12 | 28		12 | 28
NP_Agreement	201 | 993	272 | 1082	299 | 1732	772 | 3807
NP_Modification	302 | 509		53 | 60	355 | 569
Diathesis	201 | 61	176 | 118	147 | 148	524 | 327
Tense Aspect Modality	157 | 39	77 | 275	186 | 134	420 | 448
Sentence Types	80 | 100	322 | 454	105 | 14	507 | 568
Coordination	147 | 186	379 | 320	105 | 429	631 | 935
Negation	289 | 129	62 | 106	82 | 210	433 | 445
Word Order		7 | 7	60 | 160	67 | 167
Extragrammatical	24 | 34		253 | 0	277 | 34
Total	1586 | 2951	1965 | 3275	1732 | 3308	5283 | 9534

FIGURE 2 Status of the data construction as of March 1996: relevance and breadth of individual phenomena differ for the three languages (the figures are grammatical vs. ungrammatical items respectively).

2.2 Test Data Construction

Following the TSNLP test suite guidelines (Balkan et al. 1996) and using the annotation schema sketched in section 2.1, the construction of test data was based on a classification of (syntactic) phenomena to be covered. From judgments on linguistic relevance and frequency for the individual languages[3] the following list of *core phenomena* for TSNLP was compiled:

- complementation;
- agreement;
- modification;
- diathesis;
- modality, tense, and aspect;
- sentence and clause types;
- topology and word order;
- coordination;
- negation; and
- extragrammatical (e.g. parentheticals and temporal expressions).

A further sub-classification of phenomena relates to the relevant *syntactic domains* in which a phenomenon occurs; for the above list these

[3]Given the very limited resources for corpora studies in TSNLP (see Dauphin et al. 1995), the choice of syntactic phenomena for the three languages was primarily based

are sentences (S), clauses (C), noun phrases (NP), adjectival phrases (AP), prepositional phrases (PP), and adverbial phrases (AdvP). Cross-classified phenomena names are composed by attaching the syntactic domain as a prefix to the phenomenon name (e.g. *C_Complementation, NP_Agreement* et al.) and can be further sub-classified according to phenomenon-internal dimensions.

Figure 2 gives a survey of the test material constructed in TSNLP (as of March 1996); although — because of language-specific differences and the parallel construction of the data — coverage varies across the three languages, all three test suites have a comparable breadth of analysis for each of the phenomena (each totaling some 4500 test items). The vocabulary used in the TSNLP test suites in general is drawn from the business domain and aims for a minimally-sized lexicon (complete vocabulary lists for the three languages are given in Lehmann et al. 1996); current lexicon sizes range between 550 and 900 inflected forms of (on average) some 400 lexemes.

As the identification of relevant parameters for each of the phenomena turned out to be difficult and arguable in several cases, not all of the test data are annotated to the same depth on the phenomenon-specific level. Nevertheless, for the phenomena agreement, complementation, and topology and word order (and in part for sentence and clause types as well) the set of relevant parameters has been determined and used in annotating the test data. The ungrammatical test items for each phenomenon are coarsely classified according to the source of ungrammaticality (ideally resulting from the variation of a single parameter).

Following is an example for the C_Complementation (verbal government) phenomenon which is characterized by two parameters: (i) the total number of (obligatory) complements; and (ii) a summary of the actual valency frame.[4] For the sentence *Der Manager sieht den Präsidenten.* ('the manager sees the president.'), for example, the parameter values are:

number of complements: *2*
valency frame: *subj (NP_nom)* , *obj (NP_acc)*

---

on subjective assessments obtained from experienced linguists and grammar engineers. Naturally, the current list is incomplete and cannot be considered exhaustive.

[4]Obviously, both of the C_Complementation parameters are redundant with respect to the information that is already encoded in the phenomenon-independent *analysis* part of the *test item* annotations. However, making the valency information explicit as two separate per phenomenon parameters allows one to explain the source of ungrammaticality in ill-formed test items quite straightforwardly. Additionally, the parameters (at least intuitively) correspond to the procedures that can be applied to grammatical test items in order to derive ill-formed examples: in the case of verbal government ungrammatical items either result from the deletion of complements or from a violation of the governed properties.

Category	German Example
S_imp-v1	*Denke an den Test!*
S_non-imp-v1	*Denkt der Manager an den Test?*
S_wh-v2	*Wer kommt?*
S_v2	*Der Manager denkt an den Test.*
S_cpl(dass)-vfinal	*Daß der Student nur ja an den Test denkt.*
C_cpl(dass)-vfinal	Der Manager glaubt, *daß der Student arbeitet.*
C_cpl(ob)-vfinal	Es ist fraglich, *ob der Student arbeitet.*
C_v2	Der Manager sagt, *der Chef halte den Vortrag.*
C_wh-v2	Der Manager fragt: *Wer hält den Vortrag?*
C_wh-vfinal	Der Manager fragt, *wer den Vortrag hält.*
C_rel-vfinal	Der Manager, *der kompetent ist*, arbeitet.

FIGURE 3 Excerpt from the list of categories used in annotations at the sentential and clausal levels: lexical properties (e.g. the type of complementizer) appear in parentheses; language-specific dimensions (e.g. the verb-initial vs. verb-second vs. verb-final distinction for German) are cast into suffixes.

The corresponding ill-formed test items derived from this example

**Der Manager sieht.*

**Der Manager sieht [dem Präsidenten]$_{dat}$.*

are classified according to the respective parameter violated and linked to the positive example into a *test set* (see section 2.1).

In order to enforce consistency of annotations across the three languages — even though the test data construction is carried out in parallel — a canonical list of categories and functions used in the description of categorial and dependency structure had to be established; figures 3 and 4 present a (small) excerpt from the two lists. The dimensions chosen in the classification attempt to avoid properties that presuppose very specific assumptions of a particular theory of grammar (or language), and rather try to capture those distinctions that evidently are relevant across the set of TSNLP core phenomena; thus, the subclassification of German clause types (see figure 3), for example, mostly corresponds to what is actually realized as clausal complements for the verbal government phenomenon.

To ease the laborious test data construction and to reduce erratic variations in filling in the TSNLP annotation schema, a graphical test suite construction tool (tsct) has been implemented. The tool instantiates the annotation schema as a form-based input mask and provides for (limited) consistency checking of the field values. Additionally, tsct allows the reuse of previously constructed and annotated data, as often

Function	German Example
func	Der Manager *arbeitet.*
func	Es liegt *daran*, daß der Manager arbeitet.
subj	*Der Manager* arbeitet.
p-obj(um)	Der Manager bittet *um den Test.*
p-obj-loc	Der Manager wohnt *in der Stadt.*
subj-pred-adj	Der Manager ist *krank.*
mod-loc-place	Der Manager arbeitet *in der Stadt.*
mod-temp-duration	Der Manager arbeitet *den ganzen Tag.*
mod-manner	Das *alte* Projekt
spec	Alle *diese* Manager arbeiten.

FIGURE 4 Excerpt from the list of functions used in dependency structure annotations: examples of functors, arguments, modifiers, and specifiers.

when constructing a series of test items it will be easier to duplicate and adapt a similar item rather than producing annotations from scratch.

2.3 The Test Suite Database

One of the primary technological objectives of TSNLP was to design and implement a linguistic database that can ease the storage, maintenance and retrieval of natural language test data.

2.3.1 Motivation

According to the specific TSNLP requirements (see section 1.2), the key desiderata of the database design are:

- **usability**: to facilitate the application of the methodology, technology and test data developed in TSNLP to a wide variety of diagnosis and evaluation purposes of different applications by developers or users with varied backgrounds;
- **suitability**: to meet the specific necessities of storing and maintaining natural language test data (e.g. in string processing) and to provide maximally flexible interfaces;
- **extensibility**: to enable and encourage users of the database to add test data and annotations according to their needs without changes to the underlying data model;
- **portability and simplicity**: to make the results of TSNLP available on several different hard- and software platforms free of royalties and easy to use.

Although the hierarchical organization of phenomena and the concept of per-phenomenon parameters (see section 2.1) seem to suggest a data model incorporating inheritance relations (as they are known from

object-oriented or taxonomical databases), hierarchical relations are of only limited significance in the annotation schema. Therefore — and also to most closely approximate the five desiderata — the TSNLP database takes a conservative approach building on plain relational database technology.

Because it is expected that the applicability and acceptance of the tools (i.e. the technology) produced by the project will be one of the keys to the wide distribution and deployment of the methodology and test data developed, a special emphasis in the database design was put on aspects of flexibility and interfaces (to both people and programs). As a matter of fact, however, the technological requirements of developers or users from an academic background typically differ substantially from those in an industrial environment. To account for the two different areas of application, the TSNLP database was implemented taking a dual strategy.

Firstly, to produce a tool that is well tailored to the purpose of storing and retrieving TSNLP test data, highly flexible in its interfaces and portable and free of royalties at the same time, a small and simple relational database engine in plain ANSI C (which we presently call tsdb_1 using the subscript to distinguish between the two parallel implementations of tsdb; see section 2.3.4) was implemented. Since the expected size of the database as well as major parts of the database schema are known in advance, it seemed plausible to impose a few restrictions on the full generality of the relational algebra that allow fine-tuning and simplification of both the program and especially the query language.

Secondly, an alternative implementation was carried out through the exploitation of a commercial database product (called tsdb_2; see section 2.3.5) to produce a tool that meets the demands of potential users, especially in a non-academic context. Because commercially available database systems have different assets (and typically many more features) than the home-grown implementation tsdb_1, the dual realizations of the TSNLP database carried out in parallel nicely complement each other and provide for valuable feedback and comparison.[5]

2.3.2 The Database Schema

While the data format deployed in the test data construction tool tsct mirrors the rough four-level classification given in section 2.1 and is primarily motivated by data construction considerations, the require-

[5] Recently, in an extension to the original TSNLP work plan, a third implementation of the test suite database has been started at the University of Essex; building on the popular Microsoft Access database package, this version of tsdb has very similar design goals and features to the tsdb_2 realization.

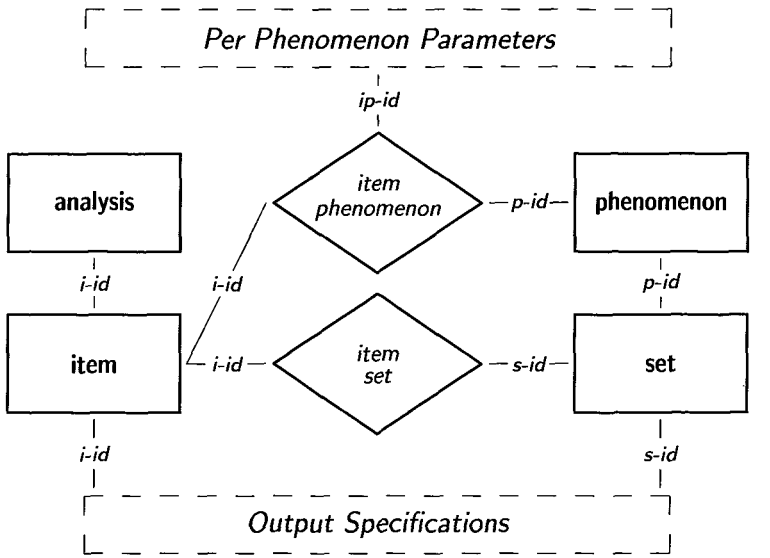


FIGURE 5 Graph representation of the TSNLP database schema. Distinct relationship types (viz. *item-set* and *item-phenomenon* are postulated for $m \times n$ relations where necessary.

ments for the abstract TSNLP data model (and analogously for the tsdb database schema) were somewhat different. Although the classification into four basic tables (corresponding to the four tsct windows) serves very well for data input purposes, it falls short from a data storage and maintenance perspective in that (because of the tsct reuse and duplicate facilities) the resulting data files are highly redundant (i.e. repetitive information is stored over and over with the individual items).

Accordingly, during the database import the four-level classification given in section 2.1 had to be refined and cast into more specific relations that allow for the elimination of redundancy and (i) factor the basic entity (or object) types from the relationship types and (ii) make this distinction explicit, in order to allow for variable logical views on the data and to enable the retrieval engine to compute the appropriate combined (or virtual) relations required to process complex queries. As another interesting facet of the import procedure, list- or multi-valued attributes which typically are not foreseen in plain relational databases are normalized as multiple records in a separate relation.[6]

[6]For example the tsct representation of test sets — relating a comma-separated list of positive test items (or rather their identifiers) to a set of negative items — is

Figure 5 presents the core of the database schema (in an abstract entity-relationship graph representation) as it is currently in use; still missing in the presentation is an account of how *user & application profiles* are linked to the core data (see section 2.4 for an example).

2.3.3 Query and Retrieval: Some Examples

For the TSNLP test data the basic purpose of the relational database retrieval facilities are:

- to dynamically construct test suite instances according to an arbitrary selection from the data; and
- to employ the query engine to compute appropriate virtual (join) relations that contain everything relevant for a particular task, but at the same time hide all internal, bookkeeping or irrelevant data.

— Following are a few retrieval examples — formulated in the simplified SQL-like query language that is interpreted by the home-grown implementation tsdb_1 (see section 2.3.4) — that demonstrate some of the flexibility of the annotations and retrieval engine (see Oepen et al. 1996 for more examples and complete documentation on the database and query language):[7]

- find all grammatical verb-second sentences relevant to the phenomenon of clausal (i.e. subject verb) agreement:

```
retrieve i-input
  where i-wf = 1 && i-category = "S_v2" &&
        p-name = "C_agreement"
```

- collect all grammatical sentences with pronominal subjects together with their identifiers and categories:

```
retrieve i-id i-input i-category
  where i-wf = 1 &&
        a-function = "subj" && a-category ~ "PRON"
```

transformed into multiple data records that relate the individual items to one and the same test set (again, its identifier). Because the lists of positive and negative items may not match up pairwise, the polarity (positive or negative) for each of the items is cast as an attribute of its own (the new *polarity* attribute rather than the *i-wf* grammaticality judgment) in order to allow for test sets in which well-formed items can serve as negative examples or vice versa; in testing a controlled language checker, for example, it may be desirable to construct test sets containing grammatical items that still are assigned negative polarity because they are not part of the controlled target language (e.g. based on a distinction between active vs. passive voice).

[7] Attribute names incorporate the first character of the defining (or base) relation as a prefix: thus, *i-input*, for example, corresponds to the *input* field of the test *item* relation (see figures 1 and 5).

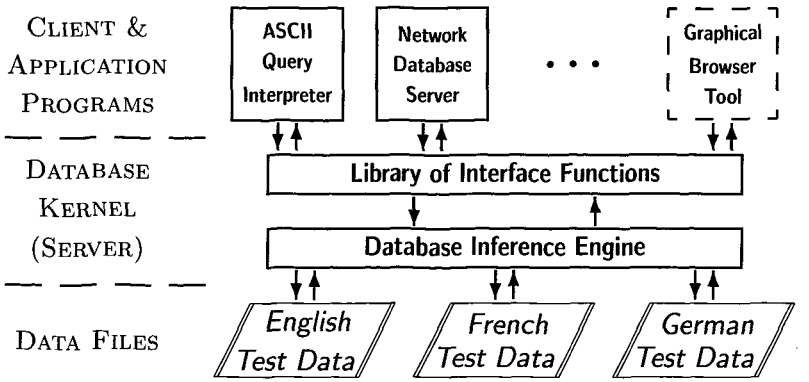


FIGURE 6 Rough sketch of the modular tsdb_1 design: the database kernel is separated from client programs through a layer of interface functions.

2.3.4 Implementing tsdb_1: Building it Ourselves

Although TSNLP had no intention to compete with commercial database products in terms of efficiency and generality, there are good reasons speaking in favour of the proprietary implementation of tsdb_1 within the project:

+ the database software and query language can be fine-tuned to the test suite extraction task; because the focus of natural language test data retrieval is on string manipulation, the tsdb_1 implementation is enriched with regular expression matching (a concept that is often unavailable in standard relational database products);
+ both data *and* software can be made available free of royalties;
+ the database can be given a flexible and well-documented interface allowing the connection to arbitrary (client) applications within or outside of TSNLP; the applicability of the interface so far has been exemplified in three cases (viz. the ASCII query interpreter, the network server, and the DFKI HPSG system);
+ the data format can be made transparent (similar to the tsct ASCII files) and thus accessible to standard Un*x text processing tools.

Figure 6 gives an overview of the tsdb_1 architecture: the database kernel is constituted by the inference engine and a separate layer of interface functions that are organized into a library of C functions.

Any application that is capable of interfacing to a library of external functions (e.g. programs that can be statically linked with the library or Common-Lisp or Prolog applications by use of the foreign function interface) can be given full bidirectional access to the database. Within

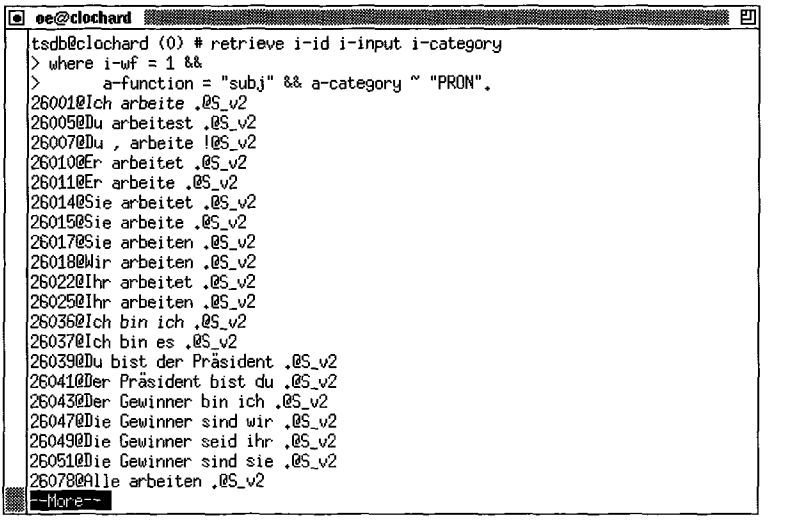


FIGURE 7 Screen dump of a tsdb_1 session: the query interpreter (topmost three lines) has command line editing, input history and relation and attribute name completion facilities.

TSNLP, two of the client applications have been developed: (i) an ASCII-based query interpreter that allows the retrieval of data in a subset of the SQL query language (figure 7 shows a screen dump from a retrieval session using tsdb_1) and (ii) a network database server that allows (read-only) access to the database from remote machines in a TCP/IP network. The tsdb_1 query language provides for the standard numeric and string comparison operations plus regular expression matching, boolean connectives (conjunction, disjunction, and negation of conditions), and a dynamic query result storage.

Other applications to connect to the TSNLP database include a lexical replacement tool built by Essex (see Arnold et al. 1994) which so far reads and writes data files in the tsct format and possibly a graphical browser for test data derived from tsct itself.[8]

The implementation of tsdb_1 is in ANSI C and supports (among others) Un*x workstations as well as Macintosh and Intel x86-based personal computers. The interface specification is as a C header file to be included by client programs.

[8]However, it rather seems that the implementation of tsdb based on the commercial database software Microsoft FoxPro (see section 2.3.5 for details) will naturally be superior in terms of convenient graphical browsing of the test data, as FoxPro comes with high-level support for graphical interfaces.

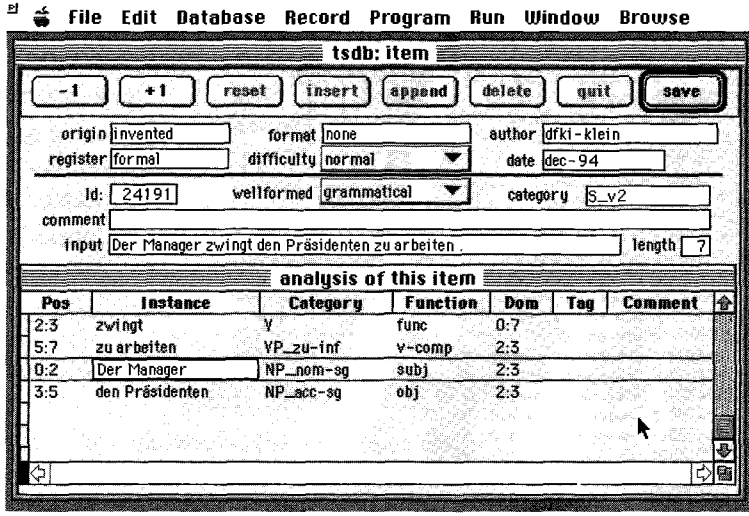


FIGURE 8 Screen dump of the development version of tsdb_2: the FoxPro database software has high-level support for graphical browsing of the data.

2.3.5 Implementing tsdb_2: Commercial Database Software

Even though the ANSI C implementation of tsdb_1 allows one to deploy the tool in a standard personal computer environment, an alternative implementation of tsdb based on a commercial database product was carried out in parallel in order to produce a tool that has a *look + feel* familiar to personal computer users from typical office software; additionally, building on common standard software the TSNLP results will be better accessible to users who have to rely on the availability of on-site technical support.

Based on the evaluation of several low- and medium-priced commercially available database systems for Macintosh and Intel x86-based personal computers running under Microsoft Windows, the product of choice was Microsoft FoxPro (in its current version 2.6) because it (i) is available for both platforms, (ii) comes with high-level interface building support for a graphical front end to the TSNLP database, and (iii) is reasonably priced. The parallel implementation of two tsdb versions is possible because both tsdb_1 and tsdb_2 use exactly the same database schema (see section 2.3.2); thus the two resulting databases come out as one-to-one replicas of each other such that data can be freely exchanged between them at any time.

While the home-grown implementation of tsdb_1 scores high for its flexible functional interface (to client software), the FoxPro version tsdb_2

has a more advanced user interface (see figure 8 for a screen dump of one of the windows in the current development version). Since FoxPro encourages the implementation of graphical, mouse- and menu-based forms that hide away technical details of the underlying database, tsdb_2 comes with a sophisticated graphical browser and editor for the TSNLP test data that is (at least) equivalent in its functionality to the X11-based test data construction tool.

2.4 Putting it to the Test

To validate the TSNLP annotation schema and test data, the project results have been tested against three different application types: a grammar checker (*Le Correcteur* for French), a controlled language checker (*SECC* for English) and a parser (the HPSG system developed at DFKI[9] for German). Following is a brief summary of some of the testing results obtained for the German test suite (for testing results for the other two test suites and applications see Dauphin et al. 1995a, 1995b).

2.4.1 A Glass Box: Connecting TSNLP to an HPSG System

Whereas for the grammar and controlled language checkers there was no or only limited access to internal data structures or the control flow within the application (i.e. the application as a whole seen from the outside is a *black box*), the evaluation situation was different for the PAGE parser for German: since the test suite and the parser have been developed by the same group, there was full access to the internal structures of individual modules as well as to the flow of control. Testing TSNLP results in this setup amounted to an ideal *glass box* evaluation.

In order to take full advantage of this favourable evaluation situation when connecting the PAGE system to the TSNLP test suites and diagnosing the German HPSG grammar, individual steps were to

- identify and tailor relevant test data (including lexical replacement where necessary);

[9]The PAGE (Platform for Advanced Grammar Engineering) system has continuously been developed over more than five years by the DFKI Computational Linguistics group. The current development version combines

- a powerful state-of-the-art grammar development system including a rich typed feature formalism and unifier, a parameterizable parser and generator, a command shell, and several development tools;
- linguistic competence for German (morphology, syntax and, semantics);
- new methods for linguistic performance modeling; and
- an interesting sample application for appointment and calendar management.

The system has been delivered to several international research institutions (including CSLI Stanford, Simon Fraser, Ohio State, and Brandeis Universities, and German sites taking part in the international VerbMobil machine translation project); see Uszkoreit et al. 1994 for more details.

- establish a bidirectional interface to the TSNLP database allowing the application to retrieve *and* store data (through the Common-Lisp foreign function interface and the tsdb_1 interface library);
- supplementarily: add output specifications (the number of readings, phrase structural or semantic representations et al.) to a PAGE *user & application profile* (see section 2.4.2);
- at the application side: investigate a mapping between structures derived from the grammar and the TSNLP morpho-syntactic and relational annotations (see section 2.4.3);
- run automated *retrieve, process, and compare* cycles in grammar diagnosis and progress evaluation; and to
- deploy the TSNLP database to store additional application-specific data (e.g. performance measures).

Given this bidirectional connection to the test suite, the DFKI PAGE system presumably is one of the few grammar development platforms seamlessly integrated with a multi-purpose, multi-lingual test suite,[10] thus enriching the grammar engineering environment with a validated and comparable evaluation facility and highly valuable coverage and performance measures for the ongoing grammar development for (currently) German and English.[11]

Additionally, initial work done on the automated mapping of HPSG feature structures as they are derived by the PAGE grammar into the TSNLP annotations at lexical and phrasal levels (see section 2.4.3 below) into allowed for the validation of the type and depth of annotations chosen.

2.4.2 Diagnostic Results Obtained

In order to facilitate a continuous progress evaluation (i.e. the automated comparison of actual system performance in terms of coverage and efficiency after changing the software or grammar to the expected or former behaviour) for grammar development in PAGE, the German test suite was customized and extended following the *user & application profile* approach (see section 2.1): all information specific to PAGE and the German HPSG grammar is stored into a designated place in the

[10] Although in the Hewlett-Packard NL system already there was a close connection to the Hewlett-Packard test suite, (see Flickinger et al. 1987), the link was unidirectional rather than bidirectional: since the HP test suite (similar to most first generation test suites) is organized as a set of ASCII files, the NL system would retrieve a set of test items, process them in a batch job, and compare the results to the (shallow) annotations available.

[11] A similar degree of integration is currently aimed for in a cooperation with the developers of the ALEP (Advanced Linguistic Engineering Platform) grammar development system funded by the European Commission.

i-id	system	version	readings	first	total	user	date	error
11001	page	4711	1	0.9	2.0	oe	23-sep-1995	
11006	page	4711	2	3.6	8.8	oe	23-sep-1995	
11178	page	4711	-1			oe	23-sep-1995	bus error

i-id	nll (normalized meaning representation)
11001	direct(theme:^(∃ ?x ?y addressee(instance:?x) work(instance:?y agent:?x)))
11006	direct(theme:^(∃ ?x ?y addressee(instance:?x) work(instance:?y agent:?x)))
11006	ask(theme:^(∃ ?x ?y addressee(instance:?x) work(instance:?y agent:?x)))

FIGURE 9 Information added in customizing the German test suite for testing the PAGE HPSG grammar: the PAGE *user & application profile* records processing measures (e.g. the grammar version and date of testing, the number of readings obtained, the processing times for finding the first reading and overall total, and the type of processing error where applicable) for each of the test items. Rather than bracketed phrase structure trees, a normalized semantic representation serves as an output specification allowing judgments on the quality of analyses and to identify spurious ambiguities.

database (viz. two specific and separate relations) such that the generality of the overall test suite is not affected; figure 9 gives an excerpt from the resulting PAGE *user & application profile.*

Using the customized German test suite and the test data available at the beginning of the testing phase (3674 test items for 9 phenomena) three complete test cycles (each of more than a full day in processing time[12]) were carried out in order to (i) debug the interface and database technology, (ii) determine the system coverage and precision and to eliminate data flaws, and (iii) compute reference data for future diagnosis (including manual inspection and correction of the $\mathcal{NLL}$ output specifications).

Feedback for the Test Suite The overall feedback from testing the German test suite with the PAGE system was very positive: both, the test data and the database technology have proven highly adequate and sufficiently flexible to provide for a valuable diagnostic tool in the (progress) evaluation of a constraint-based parser. The test suite, annotations, and software could be taken as-is, yet allowing the smooth and seamless integration into a complex grammar development system.

[12]In order to obtain comparable efficiency measures, the system is run in development mode and reset to its initial state for each of the test items; hence, even though the actual parsing time totals less than two hours, the batch processing mode adds a substantial time overhead.

Besides the feedback for the application, comparing the German test data to the analyses derived by the PAGE HPSG grammar revealed some encoding mistakes in the test data (often resulting from human failures — typing errors and misclassifications — that were not detected during the import and consistency checking), concerning above all unexpected and unintended ambiguities detected by the system as well as other inconsistencies. However, even without this additional validation and feedback the test data already turned out to be of very high quality.

Feedback for the Application Although the testing within TSNLP proper had its focus on the validation of the project methodology, test data, and technology, naturally, several diagnostic results for the application were obtained at the same time.

Fundamental in both judging the current status of the application as well as for the continuous progress evaluation while the system evolves is the classification of *output mismatches*, i.e. test items for which the expected performance of the application is different from the actual result obtained. Using the information stored in the PAGE *user & application profile* and the results obtained in the second and third test cycles, the database retrieval engine allows one to extract and count classes of output mismatches:

mismatches of expected vs. actual output	**2nd cycle**	**3rd cycle**
wellformed items without analysis	868	751
illformed items with analysis	235	43
spurious ambiguities ($\geq$ 3 readings)	27	19
fatal system errors	7	0
total reduction of mismatches	**29 %**	

While the first class of output mismatches (grammatical test items that could not be analyzed by PAGE) is mostly determined by lexical gaps in the grammar[13] and restrictions on the treatment of coordination, the second class (supposedly ill-formed test items that nevertheless have one or more analyses) predominantly indicates overgeneration in the application and allowed the grammar writer to identify two systematic sources of failure (viz. spurious rule application and lexical misclassification) Besides, two serious software failures were identified that in several years of (unsystematic) manual testing had not manifested.

[13]Since the testing for different complementation patterns (e.g. verbal complementation in the *C_Complementation* phenomenon) is partly a matter of lexical rather than of syntactic overage, obviously further test suite or grammar customization will be required.

2.4.3 Mapping HPSG structures into TSNLP Categories

In order to maximally profit from the TSNLP annotations when testing the HPSG analyses produced by the PAGE parser, it becomes necessary to map the HPSG typed feature structures onto the atomic category names used in the TSNLP database and to compute appropriate dependency relations from a derivation tree (a parsing result).

Building on a typed feature structure rewrite engine (the zebra system (see Konrad 1995) developed at DFKI) that allows the compilation and recursive application of individual or groups of rewrite rules over typed feature structures, an incomplete version of a suitable mapping procedure for PAGE result structures has been implemented. Without caring for the details of HPSG (see Pollard and Sag 1994), in the following the method and technology are illustrated by a few simplified examples.

In general, by partitioning rewrite rules into disjoint sets that are applied sequentially, it is possible to process heavily nested feature structures inside-out; for example for the mapping into TSNLP annotations it seems plausible (i) to rewrite embedded categories (i.e. lexical and phrasal nodes) first, then (ii) to deploy another set of rewrite rules to collect the results from step (i), and (iii) finally to compute the dependency relations in a third stage.

Example (1) is a simplified version of the rule that rewrites the HPSG representation of a (dative singular) saturated noun phrase into the TSNLP equivalent.[14] The context for an application of rule (1) (i.e. the pattern on the left hand side) is strictly local: it is restricted to the token phrase being rewritten.

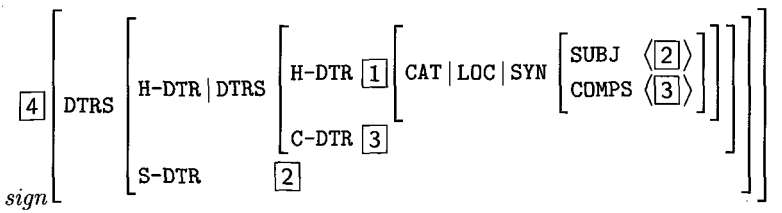
$$(1)\quad \begin{bmatrix} \textit{sign} \\ \text{CAT} \mid \text{LOC} \mid \text{SYN} \begin{bmatrix} \text{SPEC} & \langle\rangle \\ \text{COMPS} & \langle\rangle \\ \text{HEAD} & \textit{noun} \begin{bmatrix} \text{AGR} \begin{bmatrix} \text{CASE} & \textit{dat} \\ \text{NUM} & \textit{sg} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \longrightarrow \textit{NP\_dat-sg}$$

For the extraction of dependency structures from larger phrases and sentences, typically a larger context is required. In order to identify the subject and object(s) of a verbal head (a functor in the TSNLP sense), for example, the entire phrase structure tree headed by the verb (the domain

[14] Although in the rewrite rule examples for expository reasons type names are omitted where they are not strongly required to constrain the context, in the zebra system each of the embedded feature structures is assumed to bear an appropriate type. Because the types used in the specification of rewrite rules can be taken from exactly the same hierarchy that constitutes the parsing grammar itself, the set of TSNLP rewrite rules can be interpreted as a proper and declarative extension to the PAGE grammar.

of the functor) has to be taken into account: even though the grammatical functions can be determined from the valency features of the verbal head (structure [1] in example (2)), the remainder of the derivation tree (represented in (2) as a HPSG DTRS structure) contains the information about the actual realization of the subcategorized complements and the corresponding string positions.[15]

(2)



→

a-position	a-category	a-function	a-domain
|[1]|	[1]	*func*	|[4]|
|[2]|	[2]	*subj*	|[1]|
|[3]|	[3]	*obj*	|[1]|

As of March 1996 a categorial mapping for root categories of TSNLP test items (i.e. the *category* field in the *item* relation) has been established and used to automatically compare the results obtained from PAGE with the annotations found in the TSNLP test suite. Analogously to the grammaticality results, the categorial information in several cases has proven to be valuable in testing the PAGE application. Regarding feedback for the test suite, the mapping of HPSG feature structures into TSNLP categories made it possible to detect two systematic error sources in the data: firstly, it greatly helped in spotting incorrect or inconsistent categorial annotations (e.g. two different test data authors had used slightly distinct categories for the same lexical item and type of construction); secondly, it provided the basis for the identification of categorial ambiguities that test data authors had not been aware of (e.g. the use of the form *arbeiten* ('work') as both a verb and a noun).

However, unless the similar exercise is completed for TSNLP functional annotations, it remains difficult to judge to what extent an exhaustive mapping onto the TSNLP annotations can be established and to what extent the granularity of test data and level of annotations suffice

[15] In example (2) the notation '|[1]|' is used to indicate the position of the substring corresponding to the feature structure [1]. String positions in HPSG can in general be read off the PHON feature (a glossed list-valued representation); however, since the PAGE system incorporates a chart-based parsing algorithm, a more straightforward option is to extract the position values immediately from the chart.

for the thorough and automated matching against a highly formalized theory of grammar such as HPSG.

3 Future Work: Beyond TSNLP

An issue that has been discussed several times is that general-purpose test suites normally list items on a par without rating them in terms of frequency or even relevance with respect to an application. Clearly, test suites without this information cannot serve as the basis of an evaluation in the same way as application-, task- or domain-specific corpora do, unless a human evaluator assigns a personal ranking to the test data.

As to relevance, presumably a manual linguistic judgment will always be required, since a relevant phenomenon need not necessarily be a frequent phenomenon.[16]

As to frequency, one of the major challenges in constructing and applying test suites is the combination of the controlled test suite environment with information about the frequency of individual constructions and classes of phenomena to be found in text corpora. One approach envisaged to extend the TSNLP approach into this direction is to match tagged test items against analogously tagged corpora, such that the matching abstracts away from additional material in the corpus, taking into account only the relevant parts of a sentence. Quite obviously, relating test suites to text corpora along these lines is a long-term scientific goal that will require future research.

Additional straightforward extensions of TSNLP results will be to increase the coverage of the test data and to apply the methodology to more languages.

References

Arnold, D., M. Rondell, and F. Fouvry. 1994. Design and Implementation of Test Suite Tools. Report to LRE 62-089 D-WP5.1. University of Essex, UK.

Balkan, L., F. Fouvry, and S. Regnier-Prost (eds.). 1996. TSNLP User Manual. Volume 1: Background, Methodology, Customization, and Testing. Technical report. University of Essex, UK.

Balkan, L., K. Netter, D. Arnold, and S. Meijer. 1994. Test Suites for Natural Language Processing. In *Proceedings of the Language Engineering Convention.* Paris.

Dauphin, E., V. Lux, S. Regnier-Prost (principal authors), D. Arnold, L. Balkan, F. Fouvry, J. Klein, K. Netter, S. Oepen, D. Estival, K. Falkedal,

[16]The correct analysis of negated constructions, for example, can be crucial to an application even though the type of construction may be relatively infrequent.

and S. Lehmann. 1995a. Checking Coverage against Corpora. Report to LRE 62-089 D-WP3.2. University of Essex, UK.

Dauphin, E., V. Lux, S. Regnier-Prost, L. Balkan, F. Fouvry, K. Falkedal, S. Oepen (principal authors), D. Arnold, J. Klein, K. Netter, D. Estival, K. Falkedal, and S. Lehmann. 1995b. Testing and Customisation of Test Items. Report to LRE 62-089 D-WP4. University of Essex, UK.

Estival, D., K. Falkedal, S. Lehmann (principal authors), L. Balkan, S. Meijer, D. Arnold, S. Regnier-Prost, E. Dauphin, K. Netter, and S. Oepen. 1994. Test Suite Design — Annotation Scheme. Report to LRE 62-089 D-WP2.2. University of Essex, UK.

Estival, D. (principal author), K. Falkedal, L. Balkan, S. Meijer, S. Regnier-Prost, K. Netter, and S. Oepen. 1994. Survey of Existing Test Suites. Report to LRE 62-089 D-WP1. University of Essex, UK.

Flickinger, D., J. Nerbonne, I. A. Sag, and T. Wasow. 1987. Toward Evaluation of NLP Systems. Technical report. Hewlett-Packard Laboratories. Distributed at the 24th Annual Meeting of the Association for Computational Linguistics (ACL).

Konrad, K. 1995. Abstrakte Syntaxtransformation mit getypten Merkmalstermen. Technical report. Deutsches Forschungszentrum für Künstliche Intelligenz. forthcoming.

Lehmann, S., D. Estival, K. Falkedal, H. Compagnion, L. Balkan, F. Fouvry, J. Baur, and J. Klein. 1996. TSNLP User Manual. Volume 3: Test Data Documentation. Technical report. Istituto Dalle Molle per gli Studii Semantici e Cognitivi (ISSCO) Geneva, Switzerland.

Nerbonne, J., K. Netter, K. Diagne, L. Dickmann, and J. Klein. 1993. A Diagnostic Tool for German Syntax. *Machine Translation* 8:85–107.

Oepen, S., F. Fouvry, K. Netter, T. Fettig, and F. Oberhauser. 1996. TSNLP User Manual. Volume 2: Core Test Suite Technology. Technical report. Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI) Saarbrücken, Germany.

Pollard, C., and I. A. Sag. 1994. *Head-Driven Phrase Structure Grammar.* Studies in Contemporary Linguistics. Stanford, California: CSLI Publications and The University of Chicago Press.

Uszkoreit, H., R. Backofen, S. Busemann, A. K. Diagne, E. A. Hinkelman, W. Kasper, B. Kiefer, H.-U. Krieger, K. Netter, G. Neumann, S. Oepen, and S. P. Spackman. 1994. DISCO — An HPSG-based NLP System and its Application for Appointment Scheduling. In *Proceedings of the 15th Conference on Computational Linguistics (COLING).* Kyoto.

3

From Annotated Corpora to Databases: the SgmlQL Language

JACQUES LE MAITRE, ELISABETH MURISASCO, & MONIQUE ROLBERT

1 Introduction

Natural language processing is becoming more and more data oriented, and this involves the utilization of large text corpora. A part of this utilization consists either in extracting text elements or in enriching the texts by marking elements recognized by means of lexical, syntactic or semantic processing. To this end, marking languages have been developed. The most widely used is SGML (see Goldfarb 1990 and Van Herwijnen 1994). Moreover, due to their high volume and structural complexity, annotated corpora are real databases but they are rarely managed as such. This paper deals with the management of SGML annotated corpora as databases.

To build databases of SGML texts, two approaches can be considered. The first consists in translating SGML into a classical database model (relational or object oriented) and then manipulating the database by means of the languages provided by the target DBMS. Several experiments have been conducted in this line (see Blake 1994, Christophides 1994, Yan 1994 and Volz 1996). The second approach consists in considering SGML as the data definition language and developing a specific data manipulation language. The advantage of the second approach is that it gives to the user a uniform interface for annotating and manipulating texts, while the first approach forces the user to switch between two levels of representation (SGML and the database model). The choice

This work is partially supported by the MULTEXT European project (LRE 62-050).

Linguistic Databases.
John Nerbonne, ed.

of this second approach led us to develop a new query language: SgmlQL described in this paper. As its name indicates, SgmlQL is based on SQL. Its specific purpose is to manipulate trees. This allows to consider the structure of an SGML text, which is a derivation tree of the grammar which defines its type.

This paper is organized as follows. Section 2 is a brief survey of SGML. Section 3 is a typology of document manipulation operators. Section 4 describes the language and illustrates it through examples. Section 5 describes NLP applications of SgmlQL. Section 6 deals with SgmlQL's implementation and section 7 with related works. Section 8 concludes.

2 SGML

SGML is an international standard for document description (see Goldfarb 1990). A SGML document is split up into elements. An element has the following form:

$$< n\ a_1 = v_1 \ldots a_k = v_k > c < /n >$$

where $< n \ldots >$ is its start tag, $< /n >$ is its end tag, n is the name of its type, c is its content and each $a_i = v_i$ is an attribute-value pair. The content of an element may be either empty, or a string, or an element, or a sequence of elements or strings. Tags are used to mark the global structure of a document (chapters, sections, paragraphs, footnotes, etc.) as well as its detailed structure (emphasizing, cross-references, etc.). Attributes are used to qualify an element independently of its content (identification, reference link, formatting indication, etc).

The generic structure of a document is described in a Document Type Definition (DTD). A DTD defines entities and element types which can be used to describe a document. An entity is a name given to a part of a DTD or of a document. An entity can be used as an abbreviation or as a reference to another SGML document. An element type is defined by its name, its content model and its attributes. The content model is a regular expression recursively built from names of type, from *#PCDATA* (strings) and from operators which define various kinds of element compositions: ordered (",") or not ("&"), disjunctive ("|"), repeated ("*", "+"), and optional ("?"). An attribute is defined by its name and by the type of values it has. Figure 1 gives an example of a DTD and Figure 2 shows a document which is an instance of this DTD.

```
<!DOCTYPE ScientificBook [
  <!ELEMENT - - Book          (Tit, Aut+, Chap+)                                  >
  <!ELEMENT - - Aut           (#PCDATA)                                           >
  <!ELEMENT - - Chap          (Tit, Par*, Sec+)                                   >
  <!ELEMENT - - Sec           (Tit, Par+)                                         >
  <!ELEMENT - - Par           (Sent)+                                             >
  <!ELEMENT - - (Tit | Sent)  (Art | Noun | Adj | Verb | Cli | Prep | Punct)*     >
  <!ELEMENT - - (Art | Noun | Adj | Verb | Cli | Prep | Punct)  (#PCDATA)        > ]>
```

FIGURE 1 A DTD

3 Annotated Text Manipulation

Before defining the SgmlQL language, we shall describe briefly the operations we want to accomplish with it, knowing that these operations will take place in the field of annotated text corpora utilization.

Texts corpora are mostly used to extract information or build new documents. Because SGML texts are tree structured, the units on which these manipulations work are subtrees or leaves. From this point of view, one may want, for example, to modify the marking (or eventually to delete it locally) if the units it defines are not more well-suited or precise enough for further treatment. For example, retrieval of words collocation in sentences may not be easy if the words in the sentence are tagged with lexical categories. In this case operations which delete all marking in a given subtree would be useful.

More generally, we think that usable transformations for annotated text corpus utilizations can be characterized as follows:

bracketing which replaces a sequence of brother subtrees by a tree whose root has to be given and whose immediate subtrees are those in the sequence,

unbracketing which replaces a subtree by its immediate subtrees,

leaf expansion which replaces a substring in a leaf of a tree by a subtree whose root has to be given and such that the substring is its unique subtree,

compression which replaces a tree by the string obtained by concatenation of its leaves,

extraction which extracts subtrees of a tree,

tree pruning which removes branches.

Figures 3, 4, and 5 shows some examples of these types of operations.

Annotated text corpus manipulation can also consist of statistical and quantitative operations (word frequency, syntactic category lists, etc.).

All of the above operations must be preceded by location of the subtrees on which they work. This location can be done by means of pattern-matching primitives: path patterns (see Christophides 1994),

```
<ScientificBook>
  <Book>
   <Tit>
    <Noun>Bases</Noun><Prep>de</Prep><Noun>données</Noun> ...
   </Tit>
   <Aut>Claude Delobel</Aut>
   <Aut>Michel Adiba</Aut>
   ...
   <Chap>
    <Tit>
     <Art>Les</Art><Noun>problèmes</Noun> ...
    </Tit>
    <Par>
     ...
     <Sent>
      ...
      <Prep>dans</Prep>
      <Art>un</Art>
      <Noun>état</Noun>
      <Adj>cohérent</Adj>
      ...
     </Sent>
     ...
    </Par>
    ...
    <Sec>
     <Tit><Noun>Notes</Noun><Adj>bibliographiques</Adj></Tit>
     <Par>
      <Sent>
       <Art>Le</Art><Noun>problème</Noun><Prep>de</Prep> ...
      </Sent>
      ...
     </Par>
    </Sec>
   </Chap>
   ...
  </Book>
 </ScientificBook>
```

FIGURE 2 An instance of the DTD in Figure 1

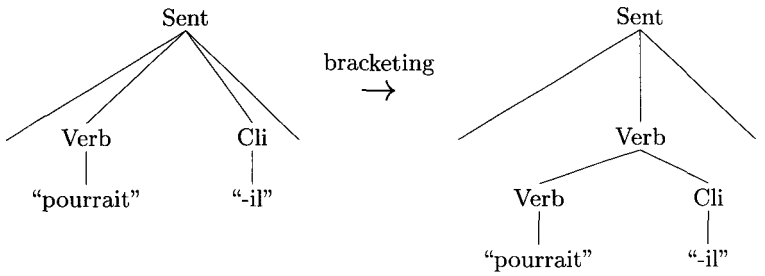


FIGURE 3 Marking up a verb + clitic sequence

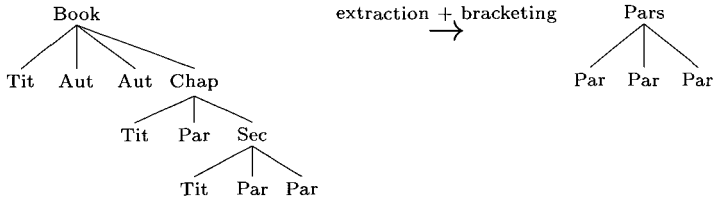


FIGURE 4 Selecting all the paragraphs of a book

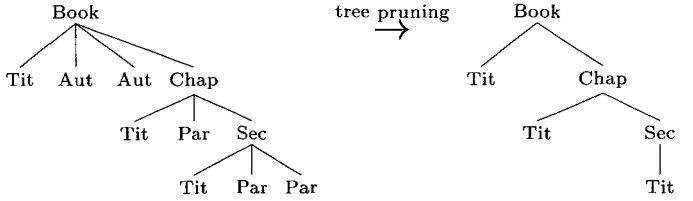


FIGURE 5 Building a book plan (the titles alone are kept)

tree patterns (see Kilpeläinen 1993) and string patterns. Moreover, it is important to build these primitives so that querying can be done without knowing the exact structure of the queried documents. As we shall see below, SgmlQL offers more sophisticated tree patterns than those proposed by Kilpeläinen (1993), and also offers string patterns.

4 SgmlQL

SgmlQL provides operators which integrate tree and string patterns, to build new documents from elements extracted from SGML documents either putting them together or modifying their structure. Rather than building a new language, we have based SgmlQL on SQL, the most popular query language, and more particularly on OQL the object-oriented SQL of the ODMG (see Cattell 1993) because texts are complex objects.

4.1 From SQL to SgmlQL

To make clear the specificity of SgmlQL compared to SQL and OQL, let us see the example of the figure 6 which gives for these three languages the expression of the same query on a given database. The database describes a set of books by their titles, their authors and their date of publishing and the query finds the title and the authors of the books published in 1995.

A relational schema is a set of table definitions. An SQL query is applied to tables and returns a table. **select... from... where** is a filter used to select attributes (*Title* and *Name*) got from the cartesian product of the table Book by the table Author, according to a given condition (*Year* = 1995).

An object oriented schema is a set of classes of objects. The **from** clause of an OQL query browses nested sets of objects (*mybooks* and *authors* of one of *mybooks*). Each object is binded to a variable (*b* and *a*). The **select** clause builds a structured object (a tuple of a title and an author name). The result of the query is a set of such objects.

SgmlQL can directly query documents formatted in SGML. The **from** clause browses subelements of nested SGML elements (*mybooks*, *Name* of each *Author* of each *Book* of *mybooks*). Each subelement is matched against a pattern (here *Title*(*t*), *Author*+(*as*), *Year*(*y*)) and some of its components are binded to variables (*t*, *as* and *y*) if the matching succeeds. The **select** clause builds the content of the answer element.

In conclusion, the originality of SgmlQL is to manipulate SGML elements. From SQL, it takes the **select... from... where...** operator and from OQL the browsing of nested sets and its clean functional se-

SQL

data schema

Book(Title: string, Year: integer)
Author(Name: string, Nickname: string, Title: string)

Titles and authors of the books published in 1995

```
          b.Title a.Name
select    b.title, a.name
from      Book b, Author a
where     b.Title = a.Title and b.Year = 1995
```

OQL

data schema

class Book tuple (title : string, authors : list(Author), year : integer)
class Author type tuple (name : string, nickname : string)
name mybooks : set(Book)

Titles and authors of the books published in 1995

```
select    tuple(book: b.title, author: a.name)
from      b in mybooks
          a in b.authors
where     b.year = 1995
```

SgmlQL

data schema

```
<!DOCTYPE Books [
<!ELEMENT - - Books              (Book*)                    >
<!ELEMENT - - Book               (Title, Author+, Year)     >
<!ELEMENT - O Author             (Name, Nickname)           >
<!ELEMENT - O (Title | Name | Nickname | Year) (#PCDATA)   > ]>
```

Titles and authors of the books published in 1995

```
<answer>
 select t, n
 from Title(t), Author+(as), Year(y) in mybooks
       Name(n) in as
 where content(y) = 1995
</>
```

FIGURE 6 An example with SQL, OQL, and SgmlQL

mantics. Its main additional functionality is to deal with pattern matching integrated in this operator.

4.2 Accessing the components of an SGML element

To access the type and the content of an element, two functions are predefined: **type** and **content**. And to access the value of an attribute, the classical "." operator is used.

Let e be the element $< t\ a_1 = v_1 \ldots a_n = v_n > c < /t >$, then

type$(e) = t$
content$(e) = c$ if c is a string or an element
list$(c_1, \ldots, c_n)$ if c is the sequence $c_1, \ldots, c_n$
$e.a_1 = v_1$
...
$e.a_n = v_n$

Therefore the content of an element is either a string, or an element or, a list of strings or elements. Hereafter the term "mixed list" is used to refer such a list. For example, let b be the following element:

```
<Book edition=2>
 <Title>Practical <Em>SGML</Em></Title>
 <Author>Eric van Herwijnen</Author>
</Book>
```

then

type(b) = "Book"
b.edition = "2"
content(b) = list(t, a)
type(t) = "Title"
content(t) = list("Practical", e)
type(e) = "Em"
content(e) = "SGML"
type(a) = "Author"
content(a) = "Eric van Herwijnen"

4.3 Queries

A query is a functional expression which is evaluated in an environment composed of SGML elements, and whose value (or answer) is an SGML element. An expression is an atom, a variable, or an operator expression. Atoms are strings, numbers and boolean values. Operators are predefined functions: some of them are described below.

The value *val*(*e*) of an expression *e* is computed as follows:

- If *e* is an atom, *val*(*e*) is that atom. For instance, *val*(1) is 1.
- If *e* is a variable, *val*(*e*) is the value of that variable in the current environment. For instance, if *mybook* is the variable which denotes the document of figure 2, *val*(*mybook*) is that document.
- If *e* is an operator expression, the definition of that operator gives *val*(*e*). For instance, *val*(Tit **in** mybook) is the list of all the elements of the document of the figure 2 whose type is *Tit*. The definition of the operator **in** is detailed in section 4.4.1.

Hereafter, the expressions "the value of *e* is *v*" and "*e* is reduced to *v*" are equivalent.

4.4 Main Operators

In this section, we present the SgmlQL operators which perform the basic operations enabling the utilization of a corpus as presented in section 3: pattern-matching, extraction and modification. All classical operators on numbers, strings, boolean values, tuples, and lists are available in SgmlQL. For the sake of conciseness, they will not be presented.

4.4.1 Pattern-matching

SgmlQL offers two pattern-matching operators. The matching of an object (element, list or string) with a pattern is achieved by the **match** operator. The **in** operator finds all subobjects of an object (element, list or string) which match a given pattern. The **in** operator makes it possible to cross an object without giving the vertical or horizontal position of the subobjects searched for. These operators use complex patterns which can recognize a string, an element or a list. SgmlQL offers many patterns, we present most of them below (*p* denotes a pattern):

string patterns

- *#PCDATA* matches with any string
- "*r*" matches with any string matching the UNIX regular expression *r*. For example, '<*SGML*>' matches with any sentence containing the word SGML.

element patterns

- _ matches with any element or string
- T matches with any element of type *T*
- <>*p*</> matches with any element whose content matches *p*
- <T>*p*</> matches with any element of type *T* whose content matches *p*. For example, <*Tit*>*#PCDATA*</> matches

with any element of type *Tit* whose content matches with any string,

list patterns

- p^* matches with any list $c_1, \ldots, c_n$ such that each c_i matches with p or else the empty list
- ... is an abbreviation of _*
- p_1, p_2 matches with any list c_1, c_2 such that c_1 matches with p_1 and c_2 matches p_2. For example, *Tit, Par* matches with any list of two contiguous elements, the first of type *Tit*, the second of type *Par*.

Obviously, patterns can be combined, as in *<Chap> Tit,Par</ Chap>* which matches with any element of type *Chap* whose the content matches with the list pattern *Tit,Par*.

It is possible to attach a variable to each pattern which does not appear in a repetition ("*"). This variable will be instantiated with the string, element or list which matches this pattern.

We give now the definition of the **match** and **in** operators. Let e be an expression and p a pattern, the expression

$$e \textbf{ match } p$$

has the value:

true	if p has no variable and *val*(e) matches p,
false	if p has no variable and *val*(e) does not match p,
$\{x_1 = v_1, \ldots, x_2 = v_n\}$	if p has variables and *val*(e) matches p instantiating the variables $x_1, \ldots, x_n$ by $v_1, \ldots, v_n$ ($\{x_1 = v_1, \ldots, x_n = v_n\}$ is an *environment*),
$\{\}$	if p has variables and *val*(e) does not match p.

Let p be a pattern and e an expression such that $o = val(e)$, the value of the expression:

$$p \textbf{ in } e$$

is a boolean if p is without variable. Otherwise, it is a list of environments computed as follows:

- if p is a string pattern then p **in** e returns the list of x **match** p for each string x of *val*(e) such that the matching succeeds,
- if p is an element pattern then p **in** e returns the list of x **match** p for each subelement x of *val*(e) such that the matching succeeds,

- if p is a list pattern then p **in** e returns the list of x **match** p for each sublist x of the content of each subelement of $val(e)$ such that the matching succeeds.

The following figures illustrate the evaluation of an **in** operator expression. They show the matched objects. It is assumed in these examples that a global variable *mybook* exists which denotes a document of type *ScientificBook* (see Figures 1 and 2). In these figures each nonterminal node is numbered and in the value of an **in** expression a number i represents the tree whose root is numbered i.

Figure 7 shows the value of the expression

Noun(n), <Prep>"de"</> **in** mybook

which is $list(\{n = 3\}, \{n = 48\})$.

The searched pattern describes a list of two contiguous elements, one of type *Noun* (the variable n is attached to it), one of type *Prep* whose content matches with the string "de". Each time a matching is achieved, the variable n is instantiated with the element of type *Noun* found. The result is the list of all the environments created so (the trees numbered 3 and 48).

Figure 8 shows the value of the expression

(Tit(t), Par*(p)) **in** mybook

It is $list(\{t = 2, p = list()\}, \{t = 9, p = list(12, 30)\}, \{t = 42, p = list(45)\})$. The searched pattern describes a list of two contiguous constituents, an element of type *Tit* and a list (eventually empty) of elements of type *Par*.

4.4.2 Element Construction

Let n be a type name, let $a_1, \ldots, a_n$ be attribute names, and let $e_1, \ldots, e_n, f$ be expressions. The expression:

$$< n\ a_1 = e_1 \ldots\ a_n = e_n > f < / >$$

is reduced to the element $< n\ a_1 = val(e_1) \ldots\ a_n = val(e_n) > val(f) < /n >$

4.4.3 Extraction

Let $e_1, \ldots, e_n, f$, and g be expressions, and let $p_1, \ldots, p_n$ be patterns. The expression:

select g **from** p_1 **in** $e_1, \ldots, p_n$ **in** e_n [**where** f]

is reduced to the value v given by the following algorithm, where *env* is the current environment:

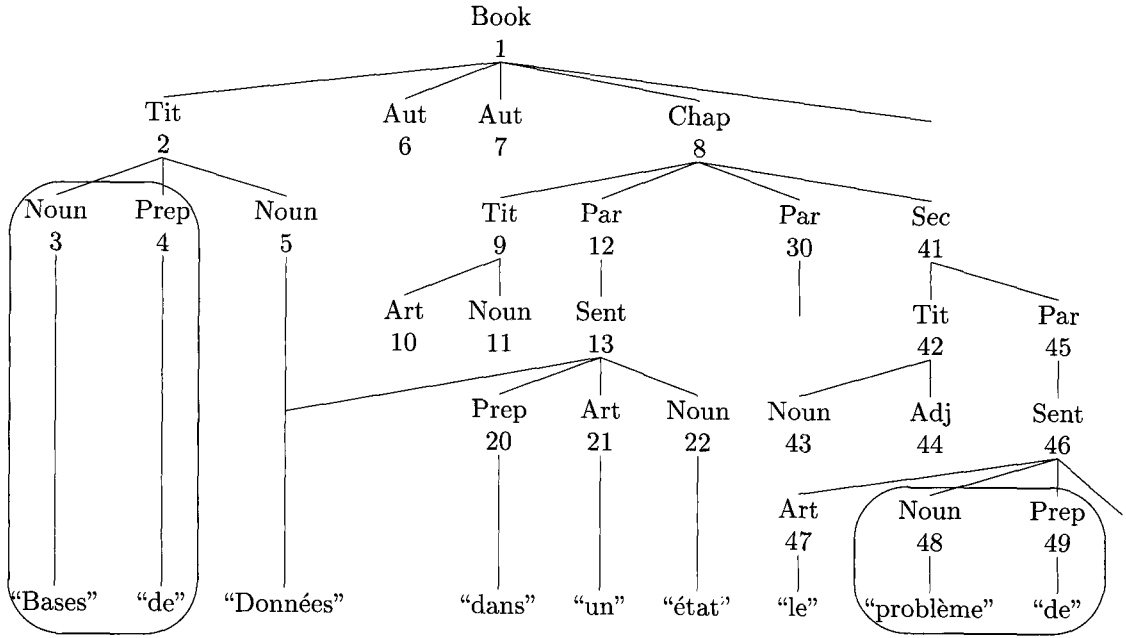


FIGURE 7 Evaluation of Noun(n), <Prep>"de"</> in mybook

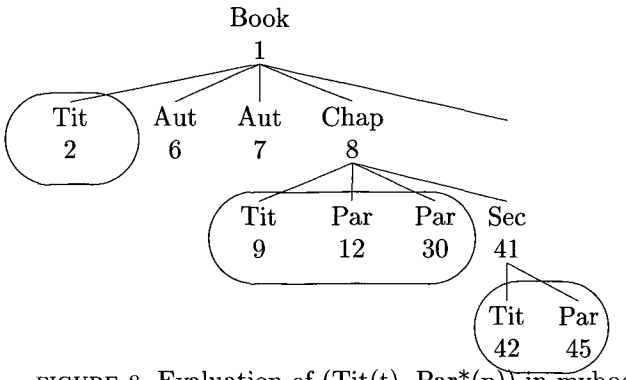


FIGURE 8 Evaluation of (Tit(t), Par*(p)) in mybook

```
v ← nil
for each env_1 in val(p_1 in e_1)
  env ← env ∪ env_1
  for each env_2 in val(p_2 in e_2)
   ...
   env ← env ∪ env_{n-1}
   for each env_n in val(p_n in e_n)
    env ← env ∪ env_n
    if val(f) = true then v ← v, val(g)
   end_for
   ...
  end_for
end_for
```

In this algorithm and in the examples below, we use the operator "," that we have define to concatenate strings, elements and mixed lists. Let e_1 and e_2 be two elements, l_1 and l_2 be two lists, s_1 and s_2 be two strings:

- if e_1 is an empty element and e_2 is any element then $e_1, e_2 = e_2$ and conversely,
- if e_1 and e_2 are strings then e_1, e_2 is their concatenation,
- if e_1 is a string and e_2 an element (or conversely), then e_1, e_2 is $list(e_1, e_2)$,
- if e_1 is a string and e_2 is $list(e_{21}, \ldots, e_{2n})$ then e_1, e_2 is $list(e_1, e_{21}, \ldots, e_{2n})$ and conversely,
- if e_1 is $list(e_{11}, \ldots, e_{1m})$ and e_2 is $list(e_{21}, \ldots, e_{2n})$ then e_1, e_2 is $list(e_{11}, \ldots, e_{1m}, e_{21}, \ldots, e_{2n})$.

The two following queries illustrate the construction and the extraction operators. The first one finds the title and the introductory paragraphs (at level 1) of each chapter of mybook whose title contains SGML:

```
<Answer>
  select t, ps
  from <Chap>Tit(t), Par*(ps), ...</> in mybook
  where t match 'SGML'
</>
```

The second one finds the title and all paragraphs (whatever their level) in each chapter of mybook.

```
<Answer>
  select t , select p from Par(p) in c
  from <Chap>Tit(t), ...</>(c) in mybook
</>
```

4.4.4 Modification

Modifications of an element are handled by two operators: **replace** and **remove**. Theyare very powerful operators which in particular carry out the tree-transforming operations described in section 3: bracketing, unbracketing and leaf expansion for **replace**, tree pruning for **remove**.

Let e, f, and g be expressions and let p be a pattern. The expression:

replace p **from** e [**where** f] **by** g

is reduced to the object produced by replacing in *val(e)*, each subobject o which matches p and such that *val(f)* = *true*, by *val(g)*. The expressions f and g are evaluated in the environment produced by the matching of o with p. The following query replaces each chapter of mybook by the title of that chapter:

```
<Answer>
  replace <Chap><Tit>_(t)</Tit>,...</Chap> from mybook
  by <ChapTit> t </>
</>
```

Let e and f be expressions and let p be a pattern. The expression:

remove p **from** e [**where** f]

is reduced to the object produced by deleting from *val(e)* each subobject o which matches p and such that *val(f)* = *true*. The expression f is evaluated in the environment produced by the matching of o with p.

This query performs the transformation of Figure 5. It removes from mybook all branches which do not contain an element of type *Tit*:

```
<Answer>
  remove _(e) from mybook
  where not( (Tit in e) or (e match #PCDATA) )
</>
```

4.4.5 Compression

The compression (see section 3) is handled by the **text** operator. Let e be an expression such that $o = val(e)$, the expression:

$$\textbf{text}(e)$$

is reduced to o if o is a string, or to the concatenation of the strings appearing in o if o is an element or a list. The following query extracts all the sentences of mybook and removes all tags in these sentences:

```
<Answer>
  text(Sent in mybook)
</>
```

5 Using SgmlQL for Natural Language Processing

Due to the increase of the corpus-based approach in Natural Language Processing, text-oriented software tools are in dire need. In this chapter, we shall show some particular uses of SgmlQL for the utilization of annotated corpora in this field.

5.1 Search and Retrieval Based on Pattern-matching

Many NLP utilizations of corpora do pattern-matching on texts. Their goal is to extract some specific information from the texts (terminological extraction (see Bourigault 1992), automatic acquisition of the hyponymy lexical relation (see Hearst 1992), classifier selection (see Sornlertlamvanich 1994)) and more generally location of occurrences of a given morpho-syntactic pattern (see Silberstein 1994).

Because SgmlQL is based on pattern-matching and offers a strong extraction operator (**select**... **from**... **where**), it is particulary well suited to this kind of use and offers several advantages:

- Patterns are regular expressions which can be embedded inside a **from** clause. They can also contain string patterns and tree patterns. SgmlQL is thus rigourously formal and highly expressive.
- Matching can be done at any level of a tree, without having to specify the structure that the query goes through. But it is also possible to specify a local matching context inside the tree. This can be very useful when working on complex syntactic structures.

In the following examples, we are going to show some searching based on pattern-matching with SgmlQL queries.

Hearst (1992) identifies a set of lexico-syntactic patterns that indicate semantic lexical relations. The author gives an example of such a pattern for the hyponymy relation, which is:

NP_0 such as $\{NP_1, NP_2, \ldots, (\text{and / or})\}$ NP_n.

NPs occurring in this pattern are such that every NP_i has NP_0 as a hyponym. The following query shows one use of SgmlQL in this case. Assume the query is applied to an SGML text which has the DTD of Figure 1 (i.e., only the lexical categories have been marked). The query gives every pair of nouns which appears in the following sequences: a noun followed by "*such as*" optionally followed by an article and necessarily followed by a noun. These sequences are taken from my book (a text whose type is *ScientificBook* and which is bound to the variable *mybook*).

```
<Answer>
  select gen, spec
  from
    (Noun(gen), <>"such"</>, <>"as"</>, Art?, Noun(spec))
    in mybook
</>
```

This query finds the "generic terms/specific terms" pairs which appear in an expression like " ...thing such as a penknife ...". We can see that the possibility of using both string patterns and tree patterns gives SgmlQL a great deal of expressive ease. If NPs have been marked in the corpus, we can simply and directly do the searching expressed in Hearst 1992. This is achieved by:

```
<Answer>
  select gen, liste_specs
  from
   (NP(gen), ...(x), NP*(liste_specs), ...(y), NP(last_spec))
   in mybook
  where text(x) match '\<such as\>' and
    ((text(y) match 'or') or (text(y) match 'and'))
</>
```

The following query is taken from Sornlertlamvanich 1994. This query makes it possible to search Thaï sentences which contain occurrences of the pattern N–CL-DET. This pattern represents sequences made of a noun consecutively or non-consecutively followed, by a classifier and consecutively followed by a determinant.

```
<Answer>
  select s
  from Sent(s) in mybook
  where (Noun, ..., Class, Det) in s
</>
```

It is interesting to see in this example that, contrary to string pattern-matching tools, SgmlQL can be used to specify the tree context (here, the sentence) in which the matching is done. This avoids considering data items which exceed two sentences. This also constrains the data length.

5.2 Modification of Corpus Annotation

The SgmlQL **replace** operator allows for numerous modifications in the structure of a tree or its markup. Some are usable in NLP applications: extension of markup (annotation) corresponding to the use of local grammars, modification or deletion of markup. These modifications can occur at any level of the tree, even on its leaves. Let us look at three examples of this type of modification.

It is interesting in some cases to complete the existing markup of a corpus to make it easier for subsequent use of the corpus. The following query makes the transformation shown in Figure 3 of Section 3. It consists of using the *Verb* tag to mark the sequences made up of a verb followed by a clitic, in all the sentences of mybook:

```
replace Sent(s) from mybook by
  replace (Verb, Cli)(x) from s by <Verb> x </>
```

The list pattern *(Verb, Cli)(x)* captures all the occurrences of two brother nodes whose types are *Verb* and *Cli*, respectively.

One may want instead to simplify the markup of a text, for example in order to make it easier to apply some types of searching to it. The combined use of the **replace** and **text** operators allows one to switch directly from a complex tree to a tree which is reduced to its root and has as its sole leaf the string obtained by concatenation of all of the leaves of the initial tree. This type of operation is done by the following query for example:

```
replace Sent(s) from mybook by <Sent >text(s)</>
```

By applying this query, every tree of type *Sent* which appears in *mybook*, whatever its structure, will be replaced by a tree of the same type and whose only component is the string produced by **text**(s).

Other kinds of modification are possible, in particular modifications that do not change the tree structure of the document but only change

some of its markups. For example, it is possible to modify the word marked up in such a way that every word which is marked with a part of speech tag, becomes an element of type *Word* which has its part of speech tag as an attribute.

replace Sent(s) **from** mybook **by**
  replace <>_(s)</>(w) **from** s **by** <Word cat = type(w) > s </>

For example, this query transforms the element

<Verb>eat</Verb>

into

<Word cat=Verb>eat</Word>

5.3 Statistical and Quantitative Utilization

Corpus utilization often involves generating lists and statistics (basic statistics for words, collocates (pattern or part of speech) such as frequency, etc. (see Ide 1994)). Classical operators of SQL make it possible to achieve a large part of these uses in parallel with retrieval. The first example below calculates the number of word sequences corresponding to the pattern "Noun of Noun" (characteristic pattern for compound noun or technical term retrieval):

<Answer>
  group nn **in**
   select <NN>n1, "of", n2</>
   from (<Noun>_(n1)</>, <Prep>"of"</>, <Noun>_(n2)</>))
   (nn) **in** mybook)
   by nn
  with <Count>**count**(partition)</>
</>

group ... **by** ... **with** is an OQL operator which is a functional version of the **group** ... **by** operator of SQL. The result of this query is the list of all occurrences of the structure "Noun of Noun" in *mybook* and their frequency. For example:

<Answer>
   <NN>assignment of value</NN><Count>18</Count>
   <NN>variety of queries</NN><Count>2</Count>
  ...
</Answer>

Much of the work on concordances relies on the notions of proximity and neighborhood (see Yarowsky 1992, Dagan 1993, or Niwa 1994). At this point, when these notions use counting on words, they are not di-

rectly (or not easily) usable in SgmlQL. It is possible to use "structural neighborhood" such as "in the same sentence" or "in the preceding or following sentence". To take into account other types of neighborhoods, we plan to develop operators which manipulate positions (relative or absolute addresses) of constituents matched by a pattern. SgmlQL is already able to associate to a variable x of a pattern a variable *x_loc* which contains the position of the component which binds x.

6 Implementation

A first prototype of SgmlQL is available: it enables to query normalized SGML documents stored into files. This version, developed with the language C and the UNIX environment, implements a part of the patterns defined in section 4.4.1 (string patterns) and does not use the DTD of the queried documents. The next version of SgmlQL will treat element and list patterns and will use the DTD of a document to control the correctness of the queries and to produce the DTD of the answer.

This prototype is a part of the set of tools delivered for the European project MULTEXT by Jean Veronis's team in the laboratory Parole et Langage in Aix-en-Provence. In this context, SgmlQL has been tested and it appears as an appropriate tool with satisfying performances. SgmlQL can be linked to other linguistic tools on the UNIX command line: for instance, it can be used to filter documents then treated with a linguistic tool (a segmenter...); it can also be used to build a new document from elements treated before by linguistic tools.

So SgmlQL appears as a stable prototype (see Le Maitre and Murisasco 1996) at the following URL: `http://www.lpl.univ-aix.fr/projects/multext/`

7 Related works

The pioneer work in this field is Gonnet and Tompa's (see Gonnet and Tompa 1987). They have shown that a grammar could be used as a database schema and they have defined a very complete set of operators to manipulate the derivation trees of these grammars.

There exists two categories of SGML query languages: specific languages or extensions of existing query languages. The languages of the first category, although they can be highly efficient, are often incomplete and are based on non-strict formalisms. The most complete among them is HyQ, the HyTime query language (see DeRose 1994). As SgmlQL, HyQ has a functional semantic and a basic operator which browses every element of a tree or of a list. It also offers a very complete set of location operators. Unfortunately, its LISP-like syntax is very heavy and it does not enable pattern matchings on trees or on lists of subtrees.

The languages of the second category have been developed within the context of projects intended to integrate a textual component into classical DBMS: relational or object-oriented. Blake (1994) describes an extension of SQL providing access to SGML documents. A SGML document is viewed as a set of three tables which respectively contains: the elements of this document (type, ancestor, subsumed text), their attributes and their hierarchical relationships. Elements of a document can be expressed by an SQL query operating on these virtual tables and then extracted into a true table by means of a specific operator. To simplify some queries, the ".." operator has been introduced which makes it possible to access to elements of a document by partially specifying their path from the root of this document (by example *Book..Par* retrieves every paragraph of a book). The interest of this SQL extension is its simplicity. The resulting language is powerful but the expression of some queries is complex because of the tabular representation of the tree-like structure of a document.

Christophides (1994) proposes an extension of O_2SQL (the O_2-DBMS's query language) which enables to query SGML documents first modeled with O_2. This extension consists mainly in the introduction of path patterns. Path patterns can be used in the **from** clause of a **select**... **from**... **where** expression to extract from a tree some parts of the paths which match these patterns. As in the precedent language, the ".." operator can be used to denote parts of paths not relevant to the query. The proposed language is very powerful. However we think that it is less natural to query a document using path patterns than using tree patterns, as we propose in SgmlQL. Moreover, path patterns do not make it possible to describe horizontal links between elements of a document. Of course, these links can be expressed with constraints on the ranks of the paths to which the elements belong, but such expressions can be very complex.

Yan (1994) presents an integration of a text retrieval system, TextMachine, into the OpenOODB system. TextMachine is based on tree inclusion primitives (see Kilpeläinen 1993). These primitives have been integrated into OSQL, the OpenOODB's query language. The resulting language is elegant but it suffers from two limitations. The first limitation is that one can test only the existence of a pattern, so it is not possible to extract the recognized parts of a text; SgmlQL enables this. The second limitation is that pattern-matching can only take into account the relative positions of the elements (below, on the left or on the right). This requires, when more precise links are necessary, to express them in the **where** clause of a query.

Volz (1996) proposes to apply a loose coupling of existing DBMS and Information Retrieval System (IRS) to structured document handling.

The system offers a complete management of texts supported by the OODBMS, and the coupling with the IRS makes it possible to access to textual aspects. The query language based on SQL, uses at the same time methods from the OODBMS and retrieval operations issued from the IRS. This makes it possible to query on structural and content-oriented aspects of a text. An application has been developed to handle SGML documents. Each document corresponds to a tree of database objects.

Finally, we must specify that no one of these languages offers operators to modify the structure of a document such as SgmlQL's **remove** or **replace**.

8 Conclusion

A first version of SgmlQL, which has less functionalities than the one described in this paper, has been implemented. It is a part of the MULTEXT tools set and is mainly used to extract data from SGML corpora for further treatments by linguistic tools.

The development of SgmlQL is in progress in the following directions:

Hypertext manipulation. We shall add to SgmlQL an addressing mechanism to locate the elements of a text and to manipulate these locations. The interest of such a mechanism in NLP is to consider contextual aspects like those described in the previous section.

Type inference. Since SgmlQL makes it possible to build new SGML documents, it is necessary for it to produce the DTD of these documents. This could be done with a classical mechanism of type inference.

Implementation. We are currently implementing a complete version of SgmlQL conceived as a uniform interface to query SGML texts stored as files or in databases (relational or object oriented).

We are convinced that the rigorous semantic on which SgmlQL is founded will make it possible to extend its functionalities according to the needs of various application fields.

References

Blake, G. E. , M. P. Consens, P. Kilpeläinen, P.-A. Larson, T. Snider, and F. W. Tompa. 1994. *Text / Relational Database Management Systems: Harmonizing SQL and SGML.* In Proceedings of ADB'94

Bourigault, D. 1992. *Surface Grammatical Analysis for the Extraction of Terminological Noun Phrases.* In Proceedings of the 14th Conference on Computational Linguistics (Coling'92), 977–81. Nantes, France.

Cattell, R. C. G, et al. 1993. *The Object Database Standard ODMG-93.* San Mateo: Morgan Kaufmann Publishers.

Christophides, V., S. Abiteboul, S. Cluet, and M. Scholl. 1994. *From Structured Documents to Novel Query Facilities.* In Proceedings of ACM-SIGMOD'94, 313–24. Minneapolis.

Dagan, I., M. Shaul, and M. Shaul. 1993. *Contextual word similarity and estimation from sparse data.* In Proceedings of the 31st annual Meeting of the ACL, 164–71. Columbus, Ohio.

DeRose, S. J., and D. G. Durand. 1994, *Making Hypermedia Work. A User's Guide to HyTime.* Kluwer Academic Publishers.

Goldfarb, C. F. 1990. *The SGML Handbook.* Oxford Clarendon Press.

Gonnet, G., and F. Tompa 1987. Mind Your Grammar: A New Approach to Modeling Text. In Proceedings of the 13th International Conference on Very Large Data Bases (VLDB'87), 339–46. Sept. 1–4, 1987, Brighton, England. San Mateo: Morgan Kaufmann.

Hearst, M. A. 1992. *Automatic Acquisition of Hyponyms from Large Text Corpora.* In Proceedings of the 14th Conference on Computational Linguistics (Coling'92), 539–45. Nantes, France.

Ide, N., and J. Veronis. 1994. *MULTEXT: Multilingual Text Tools and Corpora.* In proceedings of the 15th Conference on Computational Linguistics (Coling'94), 588–92. Kyoto, Japan.

Kilpeläinen, P., and H. Mannila. 1993. *Retrieval from Hierarchical Texts by Partial Patterns.* In Proceedings of ACM-SIGIR'93, 214–22. Pittsburgh.

Le Maitre, J., and E. Murisasco. 1996, *SgmlQL User's manual,* Internal Report, Feb.

Niwa, Y., and Y. Nitta. 1994. *Co-occurrence vectors from corpora vs. distance vectors from dictionaries.* In Proceedings of the 15th Conference on Computational Linguistics (Coling'94), 304–9. Kyoto, Japan.

Siberstein, M. 1994. *INTEX: A Corpus Processing System.* In Proceedings of the 15th Conference on Computational Linguistics (Coling'94), 579–83. Kyoto, Japan.

Sornlertlamvanich V., W. Pantachat, and S. Meknavin. 1994. *Classifier Assignment by Corpus-Based Approach.* In Proceedings of the 15th Conference on Computational Linguistics (Coling'94), 556–61. Kyoto, Japan.

Van Herwijnen, E. 1994. *Practical SGML.* Kluwer Academic Publishers.

Volz, M., K. Abere, and K. Böhm. 1996. *Applying a flexible OODBMS-IRS-coupling to Structured document handling.* In Proceedings of the 12h International Conference on data engineering (ICDE), New Orleans, USA.

Yan, T. W., and J. Annevelink. 1994. *Integrating a Structured-Text Retrieval System with an Object-Oriented Database System.* In Proceedings of the 20th Conference on Very Large Data Bases (VLDB'94), 740–9. Santiago, Chile.

Yarowsky, D. 1992. *Word-sense disambiguation using statistical models of roget's categories trained on large corpora.* In Proceedings of the 14th Conference on Computational Linguistics (Coling'92), 454–60. Nantes, France.

4

Markup of a Test Suite with SGML

MARTIN VOLK

1 Introduction

Recently, there have been various attempts to set up a test suite covering the syntactic phenomena of a natural language (cp. Flickinger et al. 1989, Nerbonne et al. 1991). The latest effort is the TSNLP project (Test Suite for Natural Language Processing) within the Linguistic Research and Engineering (LRE) framework sponsored by the European Union (cp. Balkan et al. 1994). These test suites are meant for the testing of NLP software regarding their coverage of syntactic phenomena. Volk (1995) showed that a well-organised test suite can also be used to support incremental grammar development and grammar documentation. The key issues in the organisation of a test suite are the ease of extensibility and interchangeability as well as the avoidance of redundancy. We have implemented a test suite, which is optimized for the avoidance of redundancy and we report on the trade-off for extensibility and interchangeability.

We define a test suite as a collection of syntactically well-formed natural language sentences and contrastively selected ungrammatical sentences (termed: non-sentences). The non-sentences can be used to check for overgeneration of the grammar. The collection is built according to the following principles:

1. The vocabulary for the sentences and non-sentences is derived from a controlled set of words.
2. The sentences and non-sentences are annotated with respect to

The work reported in this paper is in part joint work done with Arne Fitschen and Stefan Pieper at the University of Koblenz. In addition, I would like to thank Michael Hess and Dominic Merz for comments on earlier versions of this paper.

Linguistic Databases.
John Nerbonne, ed.

their syntactic properties. The annotation aims at being as theory-neutral as possible.

3. The sentences differ among each other in at least one feature. The same holds for the non-sentences.
4. A non-sentence is identical to a sentence up to one feature. This particular feature is the reason for the ungrammaticality.

The vocabulary restriction (principle 1) guarantees that problems encountered in parsing a test sentence do not derive from word form problems, i.e. that they are truly syntax problems rather than morphology problems. The annotation of test suite entries (principle 2) is needed to document why a test sentence was entered into the test suite and to maintain the test suite's extensibility. The minimal difference (principle 3) ensures that the redundancy in testing is kept minimal. Principle 4 governs the selection of non-sentences such that they are only minimally deviant from a grammatical item.

The simplest way of storing the sentences in a test suite is by sequentially listing them in a file. The annotation can then simply be added after a special symbol. The obvious disadvantage is that this becomes hard to understand and control after few sentences. Furthermore, it is difficult to assign the various non-sentences to a related well-formed sentence in such an unstructured setup. Example 1 shows some sentences taken from a test suite for German relative clauses (see Krenn and Volk 1993) with simple annotations demonstrating that such a format is inappropriate because of the arbitrariness of the annotations.

```
 Der Mann, der den Jungen sieht, ist zufrieden.
    / Relative Pronoun: 'der'
*Der Mann, die den Jungen sieht, ist zufrieden.
    / Problem: Agreement mismatch
*Der Mann, der der Junge sieht, ist zufrieden.
    / Problem: Wrong function
 Der Mann hat das Haus gesehen, das gross ist.
    / Position: moved
*Der Mann hat, das gross ist, das Haus gesehen.
    / Problem: Wrong position
```

Example 1

Since a test suite is meant to support testing of NLP software, it is necessary to ensure a uniform and system independent representation language. The language must provide for structuring the test suite and for annotating the sentences, while at the same time allowing a database

view on the collection. We propose using SGML (Standard Generalized Markup Language) as representation language since it is "a database language for text" (Goldfarb 1990). SGML is an abstract representation scheme not bound to any hardware or system platform. It is thus superior to any database system in view of its interchangeability. Various ways of employing SGML for the markup of a test suite will be compared in this paper.

In order to demonstrate the feasibility of using SGML for a test suite, we have collected a test suite of 350 German sentences and represented them with SGML. The test suite is integrated into the GTU system (German: *Grammatik-Testumgebung*; grammar test environment), a tool for the development and testing of natural language grammars. GTU was prototyped on DOS-PCs and reimplemented under UNIX using SICStus-Prolog (see Volk et al. 1995 and section 3 of this paper).

2 Different SGML markup schemes

SGML was developed as a markup language for textual documents (Goldfarb 1990). It was meant for the markup of different types of documents allowing their interchange across implementations and systems. In 1986 it was approved as ISO standard 8879. But it has only gained widespread acceptance in the early 1990s as tools for working with SGML became available. Recently, there has been an increasing interest in SGML due to the interest in HTML (which is an instance of SGML) in the World Wide Web.

SGML allows to define tag sets for different types of documents. It becomes most useful when the tag sets are standardized or at least interchanged together with the documents. The Text Encoding Initiative (TEI) has defined SGML tag sets for various document types such as prose, verse, drama, spoken language transcriptions, dictionaries, terminological databases and many more (Sperberg-McQueen and Burnard 1994). Because a test suite is a rather specific document type there exists no standard tag set. One can at most borrow from tag sets such as defined by the TEI for general header information (tags for author, title, funder, revision history etc.) and for units such as sentence or intra-sentential phrases. But even the TEI tag set for corpora that can be assumed to come closest to a test suite is insufficient since it is concerned with corpora details not relevant to test suites (such as contextual information) and lacks test suite specific details (such as special annotations). One could, of course, embed any of the SGML tagging schemas that are introduced in this paper into the TEI schema with the modification provisions specified therein. But this is solely a technical move and can therefore be omitted here.

In this paper we discuss various approaches of defining SGML tag sets for a test suite. We evaluate every approach with respect to

Extensibility How difficult is it to add more sentences to the test suite and to keep it consistent?

Interchangeability How difficult is it to port the test suite to another system and to use it for testing another grammar?

Redundancy What amount of redundant annotation does the test suite's representation format require?

The goal is to maximize the first two criteria while minimizing redundancy.

For a document to be processable by an SGML parser there must be three units of information (all of whom are typically ASCII files):

1. The **SGML document** contains the text with its markup tags. In our case this is the file holding the test sentences and annotations with the markup.
2. The **Document Type Definition** (DTD) defines which tags and attributes are allowed or required in a given SGML document. A DTD can be seen as a context-free grammar stating how an SGML document is to be marked up.
3. The **SGML declaration** defines parameters for the tag set such as maximum name length for tags and their attributes or the character sets used in a document.

We will concentrate on the conceptual level and will therefore only be concerned with the SGML document and its DTD. The SGML declaration is processing specific and can thus be omitted here.

2.1 Non-hierarchical markup

Using SGML we can define tags that account for the definition of a test suite and help in structuring it. The objects in the test suite are the sentences and their annotations as pairs of feature name and feature value. In addition we must assign non-sentences to a sentence and we must document the reason for their ungrammaticality. The following DTD shows a possible definition.

```
<!DOCTYPE flatTS1 [

<!ELEMENT flatTS1      - -  (entry+)                          >
<!ELEMENT entry        - -  (sentence,
                             (feature, value)+, comment?,
                             (nonsentence, problem)+ )        >
<!ELEMENT sentence     - O  (#PCDATA)                         >
<!ELEMENT feature      - O  (#PCDATA)                         >
<!ELEMENT value        - O  (#PCDATA)                         >
```

```
<!ELEMENT comment     - O  (#PCDATA)                    >
<!ELEMENT nonsentence - O  (#PCDATA)                    >
<!ELEMENT problem     - O  (#PCDATA)                    >

<!ATTLIST entry   sID   ID   #Required                  >
]>
```

Example 2

This DTD defines a test suite as a sequence of elements of type `entry`. Every `entry` consists of a `sentence`, one or more `feature-value` pairs, an optional `comment` and one or more `nonsentences` together with a `problem` field. All of these elements are defined to take strings as content, i.e. parsable character data (`#PCDATA`). In addition an `entry` gets a unique identifier which is realized via the attribute `sID`.

This format allows an entry as shown in example 3. The end tags for most of the elements have been omitted since the strong sequentiality of the elements makes it obvious where an end tag has to be inserted by an SGML parser. The example shows a German sentence which contains a relative clause (translating as *The man who sees the boy is content*). The entry also includes two non-sentences that were derived from the sentence. Every `problem` field holds information on why the respective non-sentence is ungrammatical.[1]

```
<entry sID=s1>
<sentence> Der Mann, der den Jungen sieht, ist zufrieden.

<feature> Type of relative pronoun     <value> 'der'
<feature> Case of relative pronoun     <value> nominative
<feature> Type of reference word       <value> noun
<feature> Position of relative clause  <value> direct
<feature> Number of relative clauses   <value> simple
<feature> Function of relative clause  <value> attribute

<nonsentence> Der Mann, die den Jungen sieht, ist zufrieden.
  <problem> agreement mismatch between relative pronoun and
            reference word
<nonsentence> Der Mann, der der Junge sieht, ist zufrieden.
  <problem> wrong form or function of relative pronoun
</entry>
```

Example 3

[1]Note that most of our examples cannot be processed using a standard SGML declaration. One needs to adapt the NAMELEN variable and to extend the characters permitted in attribute names to include the apostrophe.

The advantage of such a format is its transparency and therefore its straight forward interchangeability. The test suite can be given to other researchers and they can easily extract the information they desire. Moreover, the order of entries in the test suite is not of any significance and therefore the test suite can be easily extended. The major shortcoming of this format lies in the amount of redundant annotations that needs to be copied from entry to entry. Imagine an entry that needs identical annotation up to the feature "Type of relative pronoun". It would require all the other annotations to be copied in its entry.

2.1.1 Using SGML attributes with a default value

A first step towards reducing this redundancy lies in the use of a default mechanism that inserts default values for the annotation features if nothing else is specified. That way one would only need to specify the features specific to a given sentence. SGML provides a default mechanism for attributes, attributes being properties of SGML elements. The modified DTD for the test suite could be defined as in example 4.

```
<!DOCTYPE flatTS2 [

<!ELEMENT flatTS2     - - (entry+)                          >
<!ELEMENT entry       - - (sentence, comment?,
                           (nonsentence, problem)+ )        >
<!ELEMENT sentence    - O (#PCDATA)                         >
<!ELEMENT comment     - O (#PCDATA)                         >
<!ELEMENT nonsentence - O (#PCDATA)                         >
<!ELEMENT problem     - O (#PCDATA)                         >

<!ATTLIST entry
   sID         ID                                  #Required >
<!ATTLIST sentence
   relprontype ('der' | 'welcher' | 'was' | 'wo')      'der'
   relproncase (nom | gen | dat | acc)                  nom
   refwordtype (noun | name | pronoun | adverb)        noun
   position    (direct | pre | moved)                direct
   number      (simple | embedded | conjoined)       simple
   function    (attribute | subject | object)     attribute
>
]>
```

Example 4

The definitions for the elements `feature` and `value` have been eliminated. Instead, all features are added as SGML attributes to the element

sentence. The features (and some values) have been replaced by abbreviations corresponding to the features in example 3. This format has the additional advantage that an attribute's domain can be enumerated (middle column of the attribute list) and a default value can be defined (rightmost column). For example, the feature relproncase stands for "case of the relative pronoun" and has as its domain nominative, genitive, dative, and accusative with the default set to nominative.[2] With the domain information the SGML parser can check whether a given value is valid according to the DTD thus providing a free consistency check for the test suite's feature system. In the SGML document we now have to specify only those features whose values deviate from the default values. Example 5 is identical to example 3 except for a different type of relative pronoun.[3]

```
<entry sID=s2>
<sentence  relprontype = 'welcher'>
    Der Mann, welcher den Jungen sieht, ist zufrieden.
 <nonsentence>
    Der Mann, welche den Jungen sieht, ist zufrieden.
  <problem> agreement mismatch between relative pronoun and
            reference word
<nonsentence>
    Der Mann, welcher der Junge sieht, ist zufrieden.
  <problem> wrong form or function of relative pronoun
</entry>
```

Example 5

The SGML parser will turn this entry into a fully specified entry in the following format where the attributes are given before the element.[4]

```
Attr SID = S2
(ENTRY
Attr RELPRONTYPE = WELCHER
Attr RELPRONCASE = NOM
Attr REFWORDTYPE = NOUN
Attr POSITION = DIRECT
Attr NUMBER = SIMPLE
```

[2]The meanings of the other features and their domains are documented in Krenn and Volk (1993).

[3]*der* and *welcher* are synonymous relative pronoun forms, the former is much more frequently used (Grebe 1973 claims that *der* and its corresponding feminine and neuter forms account for 85% of all relative pronoun occurrences).

[4]This output was produced by the sgmls-parser, a validating SGML parser available in the public domain. The output was slightly modified to facilitate readability.

```
Attr FUNCTION = ATTRIBUTE
(SENTENCE Der Mann, welcher den Jungen sieht, ist zufrieden.
)SENTENCE
(NONSENTENCE Der Mann, welche den Jungen sieht, ist
             zufrieden.
)NONSENTENCE
(PROBLEM - agreement mismatch between relative pronoun and
           reference word
)PROBLEM
(NONSENTENCE Der Mann, welcher der Junge sieht, ist
             zufrieden.
)NONSENTENCE
(PROBLEM - wrong form or function of relative pronoun
)PROBLEM
)ENTRY
```

Example 6

In the parser output every attribute for the sentence is present with its default value except for the attribute `relprontype` which has its value as specified in the SGML document. Please note that this format cuts back on redundancy whenever default values are appropriate (which is the case in a substantial number of entries). In addition, the default attributes are assigned to every element of type `sentence`. If the test suite contains sentences (i.e. entries of type `sentence`) that have nothing to do with relative clauses, these default values will still be assigned. It is not possible to assign an attribute only if some other attribute is present. Attributes are independent of each other. To remedy this problem one needs to define a special tag for every syntactic phenomenon that has its own set of attributes. Instead of having a tag for `sentence` we have to have a tag for `sentence-with-relative-clause`, `sentence-with-coordinated-clauses`, `sentence-with-complex-np` and so on. The attributes defined in example 4 for `sentence` will then be defined for `sentence-with-relative-clause` only. Using the SGML default mechanism thus cuts back on the redundancy of the annotation system but results in an unacceptable proliferation of special purpose tags.

2.1.2 Using SGML attributes with a "Current" value

Another means of using inheritance within SGML attributes is the use of the `#Current` value instead of the default value. Specifying `#Current` says that the value of an attribute needs to be given at the first occurrence of the element and that it will be copied to every subsequent occurrence until a new value is explicitly given which is copied from

there on. The attribute list for sentence in example 4 could be turned into the format in example 7.

```
<!ATTLIST sentence
   relprontype ('der' | 'welcher' | 'was' | 'wo')  #Current
   relproncase (nom | gen | dat | acc)             #Current
   refwordtype (noun | name | pronoun | adverb)    #Current
   position    (direct | pre | moved)              #Current
   number      (simple | embedded | conjoined)     #Current
   function    (attribute | subject | object)      #Current
>
```

Example 7

If we sort the test suite entries according to the first attribute, we have to specify the attribute only once for every value which will then be inherited by all subsequent entries. Of course the same limitations mentioned for the default value also apply to the #Current value. Since the attributes are specific to sentences with relative clauses, one has to make sure that they are not copied to other test suite entries. In addition, using #Current forces the test suite developer into carefully ordering the entries. Extending the test suite thus becomes an error prone process. For this reason the usage of #Current is generally discouraged (cp. Rieger 1995, p. 134).

Another approach to reducing redundancy consists of inheriting neither default nor current values but all general values through an inheritance hierarchy. For this we need something resembling an object-oriented class system where properties are inherited from one level to another unless otherwise specified. But switching to a real object-oriented framework would undermine our basic assumption of keeping the test suite in a format for easy interchangeability. We therefore have experimented with describing an inheritance tree with SGML.

2.2 Layered Markup

Organizing the test suite into a tree structure requires, above all, to identify the core syntactic phenomena that serve as the nodes in the tree. Let's assume that we want to model a tree as depicted in figure 1. The nodes represent syntactic phenomena like main clauses, subordinate clauses or relative clauses. A node in the tree inherits all features from the phenomena on the path that connects this node to the root of the tree. Thus, a node is more general than its daughter nodes. This inheritance obviously provides a means to compact the representation.

There are two principally distinct ways of expressing a tree structure in SGML. One is by using the built-in reference links as edges between

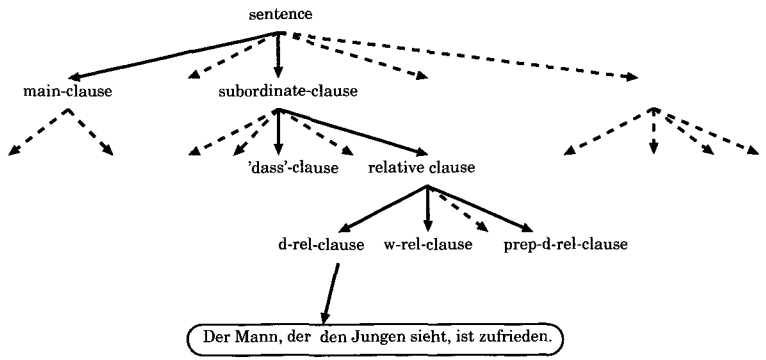


FIGURE 1 Simple test suite tree

the nodes. The other is mapping the nested structure generically or directly into the DTD.[5]

2.2.1 SGML tree structure via reference links

Reference links in SGML are special purpose attributes to indicate references within a document. A reference consists of a pointer and a designated goal. In the DTD the pointer is marked with the `IDREF` value whereas the reference goal is marked with `ID`. Attaching these values to two attributes called `nextnode` and `nodeID` for an element `node` provides the means to point from one node to another. However, this allows only to point from a node to exactly one other node. But in order to account for a tree structure we need to point to more than one node. We therefore introduce an empty element within `node` called `nodelink` which can occur more than once. This empty element holds nothing but an `IDREF` pointer to another node. The DTD will then look like example 8. The definition for `entry` has been omitted. It can be assumed to be the same as in example 2.

```
<!DOCTYPE layeredTS1 [

<!ELEMENT layeredTS1      - -  (node+, entry+)                >
<!ELEMENT node            - -  (name, feature, value,
                                 (nodelink+ | entrylink+))  >
<!ELEMENT name            - O  (#PCDATA)                      >
<!ELEMENT feature         - O  (#PCDATA)                      >
<!ELEMENT value           - O  (#PCDATA)                      >
```

[5]Note that in a similar manner the TEI Guidelines provide both built-in reference links and a nested generic structure for the representation of tree structures (cp. Sperberg-McQueen and Burnard 1994, sections 21.2 and 21.3).

```
<!ELEMENT nodelink       - O  EMPTY                          >
<!ELEMENT entrylink      - O  EMPTY                          >

<!ATTLIST node          nodeID      ID       #Required      >
<!ATTLIST nodelink      nextnode    IDREF    #Required      >
<!ATTLIST entrylink     nextentry   IDREF    #Required      >
]>
```

Example 8

The DTD in example 8 states that a layered test suite consists of a sequence of nodes followed by a sequence of entries. Every node is required to have a **name**, a **feature**, and a **value** element as well as at least one link to another node or at least one link to an entry. The links are empty elements consisting only of the attributes **nextnode** and **nextentry** respectively. The nodes and entries in example 9 model part of the tree in figure 1.

```
<node  nodeID = n2>
<name>    subordinate clause
<feature> finite verb position
<value>   final
<nodelink   nextnode = n21>
<nodelink   nextnode = n22>
<nodelink   nextnode = n23>
</node>

<node  nodeID = n21>
<name>    'dass'-clause
<feature> conjunction type
<value>   'dass'
<nodelink   nextnode = n211>
<nodelink   nextnode = n212>
</node>

<node  nodeID = n22>
<name>    relative clause
<feature> conjunction type
<value>   relative pronoun
<nodelink   nextnode = n221>
<nodelink   nextnode = n222>
<nodelink   nextnode = n223>
</node>
```

```
<node  nodeID = n221>
<name>     d-relative clause
<feature>  type of relative pronoun
<value>    'der'
<entrylink   nextentry = s1>
</node>
```

Example 9

Node n2 subsumes subordinate clauses. In the example it has links to the nodes n21, n22 and n23, which are nodes for *'dass'*-clauses, relative clauses and whatever other subordinate clause one wants to specify. n2 is defined by the feature "finite verb position" which has the value "final" which stands for the fact that in German subordinate clauses the finite verb is at the end whereas in German main clauses it is at the second position. This feature is meant to be inherited to all nodes below n2. There is an inheritance chain from n2 to n22 to n221 to the test suite entry s1. This entry contains a relative clause starting with a "der" relative pronoun and will inherit all the features specified in the chain for "finite verb position", "conjunction type", and "type of relative pronoun".

The format in example 9 is very transparent but it puts the burden of organizing the inheritance completely on the software developer, because SGML does not define any inheritance along the lines of reference links. Additional software is needed that takes the output of the SGML parser, traces the inheritance links, collects the features and values, and adds them to the entries.

This approach has been implemented within the GTU project at the University of Koblenz. A tree with 35 nodes has been set up and 350 sentences were attached as entries to this tree. A C-program was added to ensure the inheritance of features and to transform the output into a Prolog database so that the test suite can easily be accessed from GTU. In addition the tree was mapped into GTU as an ordering mechanism in accessing the test suite. The user can select a particular node and is presented with all the sentences subsumed by it (see section 3).

2.2.2 SGML tree structure via generic nested tags

The tree structure of an inheritance tree can not only be represented via reference links but also by nested tags. Since SGML allows the recursive definition of tags we can define a node as consisting of name, feature, and value followed by either some (other) nodes or some entries. The DTD for this format is shown in example 10.

```
<!DOCTYPE layeredTS2 [

<!ELEMENT layeredTS2      - - (node)                          >
<!ELEMENT node            - - (name, feature, value,
                               (node+ | entry+) )             >
<!ELEMENT name            - O (#PCDATA)                       >
<!ELEMENT feature         - O (#PCDATA)                       >
<!ELEMENT value           - O (#PCDATA)                       >
]>
```

Example 10

Compared with the nested structure in example 8 we can now do without the reference links since the mother-child relation between nodes is defined by their nesting. Within the SGML document we have to keep track of the nesting of nodes which is bothersome if not supported by special editing tools. The entries are within the most deeply embedded nodes.

```
<node>
  <name> ...   <feature> ...   <value> ...
  <node>
    <name> ...   <feature> ...   <value> ...
    <node>
      <name> ...   <feature> ...   <value> ...
      <entry> ... </entry>
      <entry> ... </entry>
    </node>
    <node>
      <name> ...   <feature> ...   <value> ...
      <entry> ... </entry>
      <entry> ... </entry>
    </node>
  </node>
</node>
```

Example 11

As in the example 8/9 the inheritance of annotation features has to be performed by add-on software. The traversal of the tree as defined by example 10 is easier since a mother node directly precedes all children nodes in the linear sequence within the SGML document whereas in example 9 the inheritance program needs to follow the reference links which can result in jumps to anywhere in the file.

2.2.3 SGML tree structure via specific nested tags

Finally, it is possible to use the linguistically motivated node names directly as SGML element names. In this way we get an explicit coding of the inheritance tree in the DTD.

```
<!DOCTYPE layeredTS3 [

<!ELEMENT layeredTS3        - - (main-clause, sub-clause)
                                        +(feature, value)>
<!ELEMENT main-clause       - - (transitive-clause,
                                 intransitive-clause)       >
<!ELEMENT sub-clause        - O ('dass'-clause,
                                 relative-clause)           >
<!ELEMENT relative-clause   - O (d-rel-clause, w-rel-clause)>

<!ELEMENT 'dass'-clause     - O (entry+)                    >
<!ELEMENT d-rel-clause      - O (entry+)                    >
<!ELEMENT w-rel-clause      - O (entry+)                    >
]>
```

Example 12

In this format a subordinate clause is a `'dass'-clause` or a `relative-clause` (among other things not listed here). The `feature-value` pair is given as an included element in the top level tag and is therefore allowed in all subelements. The resulting SGML document is very short since most of the information is already in the DTD. An entry in such an SGML document consists only of the sentence, the non-sentences and the problem field. The obvious disadvantage of this approach is that for any extension of the test suite related to a new phenomenon the DTD must be modified. A new element has to be inserted into the DTD and its attributes have to defined. It is easy to imagine, how complex a DTD we can expect when covering more than a dozen syntactic phenomena. Even simple extensions can become a major project requiring SGML experts to work on the DTD. The principle of defining a markup language for easy extensibility is thus violated and interchangeability can no longer be entertained.

3 The test suite format in the GTU system

GTU is a workbench for the development and testing of unification based grammars. It supports three different grammar formalisms (LFG, ID/LP, DCG) and two different parsing modes (top-down and bottom-up chart parsing). GTU has been interfaced to huge lexical resources (the CELEX database and the Gertwol analyzer). It comes with special

purpose grammar editors and a hypertext help system. GTU was originally designed as a tutoring system for linguistics students. Therefore special care has been taken in the design of a robust and user friendly interface including visualization and automatic comparison of parsing results. Natural language sentences can be manually entered and fed to one of the parsers or they can be selected from the test suite.

GTU contains a test suite in a format similar to example 8/9, that is, it has an inheritance tree with reference links. The sentences are attached to the tree's leaves and inherit all the features from the leaves. In contrast to example 8/9 it is legitimate in the GTU test suite to attach a sentence to more than one leaf. This explains why we could not use the solution with nested tags as given in example 10/11 unless we added sentence identifiers and reference links. In GTU, a test sentence can be a representative for different phenomena. E.g. a sentence like *Der Mann sieht den Jungen an.* (The man looks at the boy.) will be attached to the "separable prefixes" node (it contains a separated verb prefix) and to the "transitive verbs" node. In order to minimize redundancy we have attached the GTU test suite to a tree structure as depicted in figure 2.

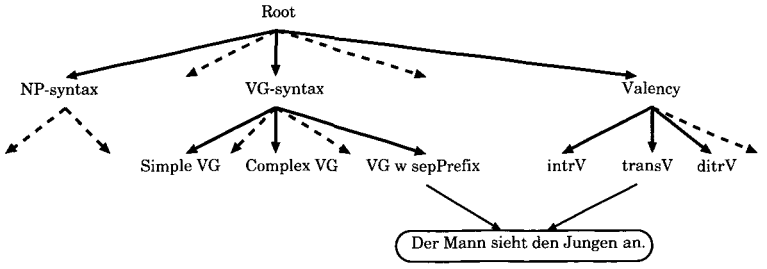


FIGURE 2 Part of GTU's test suite tree

When building up the test suite, we first had to design the test suite tree and a corresponding DTD. The tree includes nodes for noun phrases, prepositional phrases, verb groups, coordination, sentence types and so on. For all these phenomena sentences were collected from grammar books and normalized until they conformed to the requirements of the test suite definition. Non-sentences had to be made up according to the principle that they differ from a given sentence in exactly one feature. In a next step the tree itself and all sentences were annotated with SGML tags.

The complete test suite document was validated with an SGML parser. This ensures that all the sentences were tagged in conformance with the DTD. The parser output is again a structured SGML docu-

ment with all our test sentences. We translated the parser output (via lex and yacc) into a Prolog database. Since we envisioned our test suite to grow to several thousand sentences we were looking for a mechanism that could handle a large database efficiently. We chose the SICStus-Prolog external database facility which comes with indexing schemes while keeping the data on disk storage (unlike regular Prolog data that are loaded into the knowledge base in working memory).

4 Accessing the test suite

So far we have looked at the advantages and drawbacks when building up and extending a test suite. We still have to consider what the different representation formats mean for accessing the information. Retrieving information from a database can in principle be done in two different ways by either using a database query language or by using task specific menus. Query languages are more powerful and flexible while menus allow faster access for predefined purposes and are easier to handle. SGML documents can be translated into relational databases. But because of the recursive structure in SGML documents this is very tedious. Therefore, Böhm et al. (1994), on examining the relation between databases, SGML documents and information retrieval, concluded that object-oriented databases are a better choice.

It is obvious that a database query language can be used to access an SGML test suite only if the tags are sufficiently general. (If the tags were specific, how would the user know what to ask for?) For instance, it is better to have `<sentence>` rather than `<sentence-with-relative-clause>`. Representing syntactic features as SGML attributes (as in example 4 but without defaults) rather than as SGML elements (as in example 2) helps in keeping the annotations consistent and compact which is also important for using a query language. For instance, it is more compact to query for the attribute `relprontype` rather than for the element `feature` with the content `Type of relative pronoun`. So, one can imagine a database query to a test suite with non-hierarchical markup and attributes as

```
find <sentence> where relprontype = 'der'
```

Note that the validity of a query can be decided before searching the database by checking in the DTD whether `relprontype` is a valid attribute for `<sentence>` and whether `'der'` is a valid value for this attribute. This fits in nicely with findings by Böhm et al. (1994): "The key for applying IR retrieval techniques to SGML documents has to be found in using the DTD to guide the retrieval process." When elaborate

and user friendly query languages for SGML documents become available this will be the way to go for a test suite with SGML markup.

In the GTU project we did not have access to any such software and therefore decided to implement specific menus. For every node in the test suite tree the interface provides a screen display of all subsumed sentences. The sentences can be selected one at a time or in groups and fed to the GTU natural language parser with a mouse click. The user interface also allows the selection of multiple nodes at the same time to obtain the intersection of the sentence sets subsumed by the selected nodes. For instance, we can get all the sentences that have a separable prefix verb and a direct object.

Experience has shown that it is important to have groups of sentences that can be tested repeatedly, be it for a grammar in different development phases or for grammars in different formalisms. Test results can only be compared if sentence groups are stable. Therefore it is important that groups of sentences can be saved for future usage. For details on GTU see Volk (1995); the development of GTU's test suite is documented in Fitschen and Pieper (1994).

5 Summary and conclusion

A test suite should be easy to extend and to interchange. Ease of extension requires that a transparent representation format is used. Ease of interchange requires that a system and platform independent format is used. SGML fulfills both of these requirements at first sight. But there are some drawbacks if one wants to minimize redundancy in annotating the test suite entries.

Using inheritance in the way it is built into SGML, such as default values or `#Current` values for attributes, results in problems of keeping the attributes to the relevant elements which can only be resolved with a proliferation of element names. A different approach consists in forcing hierarchic structure into the SGML markup by means of reference links or nested tags. But then the inheritance mechanism has to be performed by add-on programs which are interpreting SGML markup in an idiosyncratic way.

The lesson: If extensibility and interchangeability are considered most important then one has to keep it simple, and a format like in example 2 is the best choice. This should, however, be accompanied with features represented as SGML attributes in order to get a consistency check by the SGML parser. Redundancy must then be taken into account and should be hidden from the user with the help of appropriate

editing and viewing tools. Validating SGML parsers are freely available, and can be used to perform checks on the markup.

On the practical side: In the future we would like to have tools that provide for efficient browsing and retrieval from SGML databases. This would considerably ease the usability of a test suite tagged with SGML.

References

Balkan, L., K. Netter, D. Arnold, and S. Meijer. 1994. TSNLP. Test Suites for Natural Language Processing. In *Proceedings of Language Engineering Convention, Paris.* Centre for Cognitive Science, Edinburgh: European Network in Language and Speech.

Böhm, K., A. Müller, and E. Neuhold. 1994. Structured document handling—a case for integrating databases and information retrieval. In *Proceedings of 3rd ACM International Conference of Information and Knowledge Management (CIKM).* Maryland. Also published as DIMSYS report by GMD Darmstadt under http://www.darmstadt.gmd.de.

Fitschen, A., and S. Pieper. 1994. WAITS—Weiterentwickelte Alternative Testsatzsammlung. (Studienarbeit) Universität Koblenz-Landau.

Flickinger, D., J. Nerbonne, I. Sag, and T. Wasow. 1989. Toward Evaluation of NLP Systems. Research report. Palo Alto, CA: Hewlett-Packard Laboratories.

Goldfarb, C. F. 1990. *The SGML Handbook.* Oxford: Clarendon Press.

Grebe, P. (ed.). 1973. *DUDEN. Grammatik der deutschen Gegenwartssprache.* Mannheim: Bibliographisches Institut. 3rd edition.

Krenn, B., and M. Volk. 1993. DiTo-Datenbank. Datendokumentation zu Funktionsverbgefügen und Relativsätzen. DFKI-Document D-93-24. Saarbrücken: DFKI.

Nerbonne, J., K. Netter, A. K. Diagne, L. Dickmann, and J. Klein. 1993. A diagnostic tool for German syntax. *Machine Translation (Special Issue on Evaluation of MT Systems), (also as DFKI Research Report RR-91-18)* 8(1-2):85–108.

Rieger, W. 1995. *SGML für die Praxis. Ansatz und Einsatz von ISO 8879.* Berlin: Springer.

Sperberg-McQueen, C. M., and L. Burnard (ed.). 1994. *Guidelines for Electronic Text Encoding and Interchange (TEI P3).* Chicago, Oxford: ACH, ACL and ALLC.

Volk, M., M. Jung, and D. Richarz. 1995. GTU—A workbench for the development of natural language grammars. In *Proc. of the Conference on Practical Applications of Prolog*, 637–660. Paris.

Volk, M. 1995. *Einsatz einer Testsatzsammlung im Grammar Engineering.* Sprache und Information, Vol. 30. Tübingen: Niemeyer Verlag.

5

An Open Systems Approach for an Acoustic-Phonetic Continuous Speech Database: The S_Tools Database-Management System (STDBMS)

WERNER A. DEUTSCH, RALF VOLLMANN, ANTON NOLL, & SYLVIA MOOSMÜLLER

1 Introduction

An open concept of an acoustic-phonetic database system was developed to combine advanced sound storage, analysis and retrieval functions as well as features which are normally available either in relational or in hypertext systems only. This paper describes an ongoing project on the implementation of technology and interactive tools which are used to produce speech corpora from running speech. Sound and related data sets are stored for usage in designing and evaluating algorithms for acoustic and psychoacoustic models of speech, for phonetics and speech recognition. The sound database system, its associated acoustic I/O and sound analysis workstations may be extended to non-speech applications such as experimental music and noise research, as well as multimedia techniques.

2 System Architecture

The hardware of a sound database system closely resembles a general purpose multimedia storage server connected to client sites located in a Local Area/Metropolitan Area Network (LAN/MAN) environment (cf. Deutsch 1994). Specific software for multistream retrieval, disk scheduling, advanced reading and buffering are required in order to assure con-

Linguistic Databases.
John Nerbonne, ed.

tinuous recording and playback. Audio samples are classified as continuous media (CM). The workstations at the client sites are equipped with CM I/O devices such as microphones, AD/DA converters, loudspeakers etc. which transfer the data to and from the main memory as well as to and from the file server.[1] The recording of speech sounds requires the storage of a continuous stream of sound data without interruption for signal durations from a few seconds up to several hours. Speech data playback in reverse requires the manipulation and retrieval of similar large quantities of data at high speeds. The network architecture itself has to let interact multiple users at different locations with the server at reasonable short response times and retrieve arbitrary sound information at real-time.

Therefore, the design of a sound database system differs significantly from general purpose database systems supporting textual and numerical data only. Sophisticated reading and buffering including the use of FIFO's at the clients as well as at the file server's site has to be applied in order to meet real-time performance characteristics.

The server's disk storage capacity limits the amount of sound information totally available in the LAN. Medium-sized applications need mass storage for 50 to 200 hours of sound;[2] larger sound-archive installations range up to 100 Terabytes and more. Today, storage requirements up to several hundred Gigabytes can be implemented by using RAID[3] technology. RAID improves both the reliability and fault tolerance as well as the I/O throughput for system storage in a PC-based local area network. Larger storage capacities call for special and expensive hardware. In the long run, backup problems—which today still exist in handling file sizes of typically 580 or 681 MByte—will likely be solved, allowing the use of general-purpose distributed computer systems in the future.

3 Data Acquisition

The data acquisition task includes activities performing the transfer of analog signals to digital format. Single channel speech signals are digitized at a sampling rate of 16 kHz with a binary word length of 16 bit,[4] comprising a total dynamic range of about 96 dB.[5] The continuous

[1] Local workstations may also be equipped with sufficient disk storage capacities in order to run medium size applications at the user site.

[2] 1 hour of uncompressed two channel high fidelity sound (sampling rate 48 kHz, stereo) can be stored on approximately 0.75 Gbyte.

[3] RAID: Redundant Array of Inexpensive Disks.

[4] Recent developments of AD converters comprise 20 bit quantization with noise levels down to 135 dB.

[5] The dynamic range is given by: $R = 20 * \log 2^{16}$ dB.

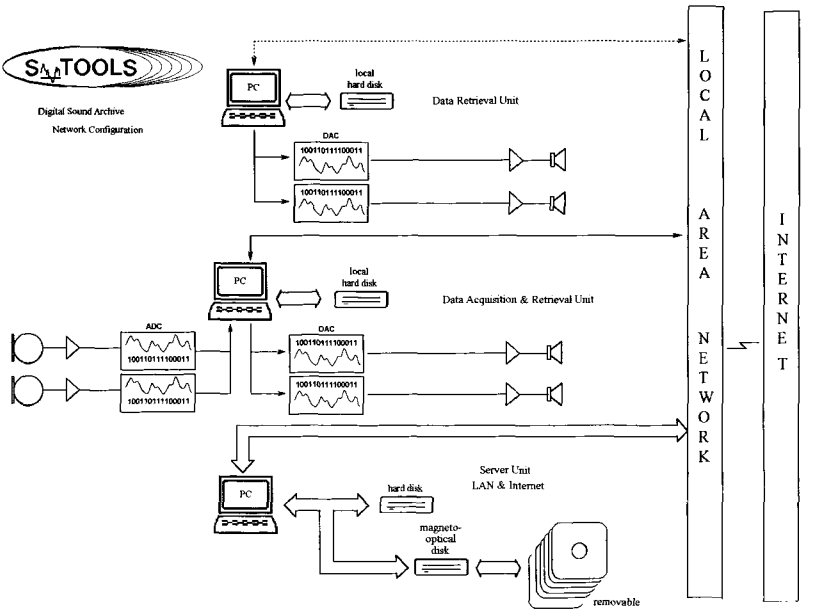


FIGURE 1 Database network configuration, consisting of data acquisition and retrieval units and a LAN server. One or more clients can be equipped with DSP subsystems for computational intensive sound analysis-resynthesis and parameter extraction.

data stream of 32 kByte/s is stored on sound files in a standard WAVE FORMAT[6] structure. During the transfer, the audio signal volume control has to be monitored to prevent AD-converters from overloading. It is of advantage to adjust input level control to the absolute peak level of the audio signals which are to be digitized in order to keep the level constant during the whole recording session. Prerecorded signals on digital carriers such as on DAT[7] tapes, hard disk recorders etc. can be transferred directly via the digital AES/EBU[8] interface. A sampling rate converter can be used to convert recordings from arbitrary sampling rates up and down to the desired application sampling rate.

[6]WAVE FORMAT: see Microsoft Multimedia Audio Waveform. Usually, file formats containing samples with linear quantization can be converted into arbitrary signal formats without loss of data.

[7]DAT: Digital Audio Tape recordings are generally recorded at sampling rates of 32, 44.1 or 48 kHz, stereo.

[8]AES/EBU: Audio Engineering Society—European Broadcasting Union.

3.1 Data Reduction

Consumer digital audio tape recorders as well as audio network services and radio broadcast productions make use of data reduction and non-linear coding in order to save bandwidth. A standard method of perceptual coding consists in the elimination of all psychoacoustical masked spectral components of the audio signal to be transmitted. Because data reduction is lossy and does not provide the regeneration of the original code, it should not be applied on archival material. Loss-free data compression is acceptable.

4 Sound Editing, Segmentation and Classification

The acoustic workbench includes interactive graphic-acoustical editing tools which support the identification of sound events in the running data stream and enables the user to enter sound segment specifications. Segments are created by means of an interactive graphical editor which provides easy bracketing of signals, the determination of size, starting and ending address of individual sound events. Cue in and cue out points are labeled accordingly and written into an edit list which is used to create a signal file directory. Reference to a sound segment can be obtained by specifying a directory entry (i.e. segment name.ext) or a segment boundary and alternatively, by relative address specification. Combined addressing and address expressions such as segment name.ext+5s-1200 samples are accepted.

One main concern for future research on the segmentation of acoustical signals is the implementation of automatic content and programme material-specific segmentation and classification procedures, as for the annotation of speech and music, singing, jazz music, opera etc. Systems should automatically perform as much of the segmentation and classification process as possible. The automatic classification is an open research problem which requires the description of the incoming documents on the logical and conceptual level. At least three levels of document description should be distinguished: the top level defines the document type hierarchy, the conceptual structure is given by the second level and finally, on the third level, the contents of the documents are represented. Ideally all relevant components using syntactic, linguistic and background knowledge should contribute to classification rules that constitute the basis for classification decisions. For the acoustics of speech, in a first step the comparison of low level spectral parameters, such as spectral dominance (formant frequencies), voiced/unvoiced decisions, pauses and prosodic features are applied as candidates for segment boundaries. These experimental solutions are still speaker depen-

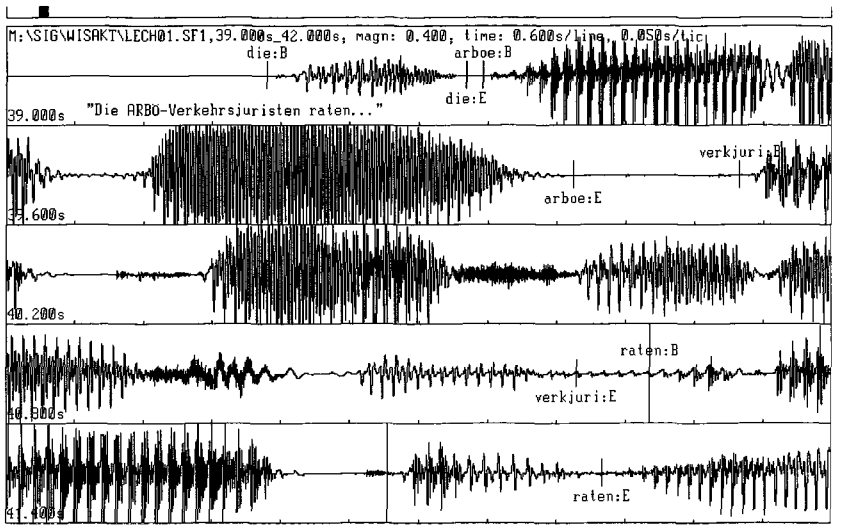


FIGURE 2 Segmentation in the time domain ("Die ARBÖ-Verkehrsjuristenraten..."). The display shows the waveform of free running speech with a duration of 2.4 seconds.

dent and work sufficiently reliable on good sound quality speech signals containing long vowel segments only. As further speech material can be processed, the work with automatic segmentation algorithms will improve.

The segmentation and classification of sound signal data has to be considered as one of the most important steps in data acquisition. It requires careful inspection of the data and is, above all, very time-consuming. However, any further processing or the retrieval of data is dependent on proper segmentation.

In phonetics, the duration of relevant speech signal segments (i.e. phonemes, words, utterances) varies from few milliseconds up to several minutes and longer. Thus, hundreds of segments have to be created in one single sound signal file. The introduction of signal file directories being attached to—or included in—the sound file has been proven very useful. It facilitates the handling of several hundred thousands of labeled sound signal segments in a database system. Moreover, sound file directory address and segment data provide the basis to establish further links necessary to integrate segmented sound information in a distributed multimedia database which can be stored on local or remote disks in a network environment.

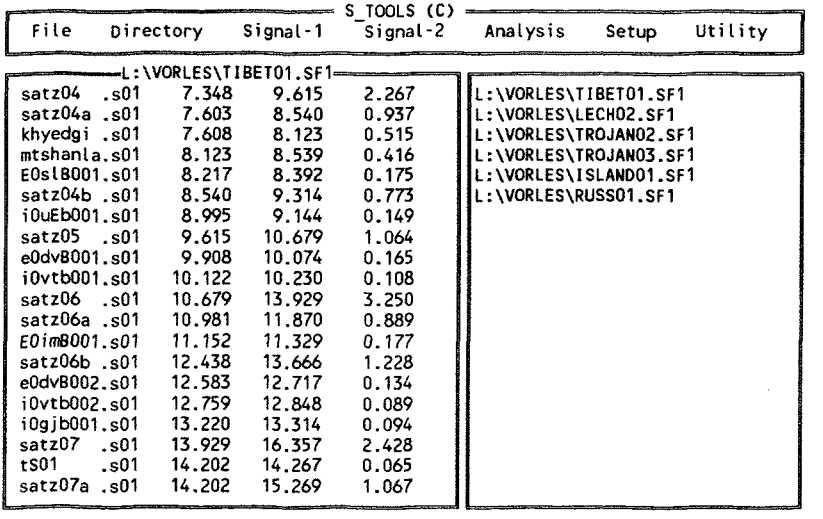


FIGURE 3 Sound Signal File Directory: right table contains the operating file system information of signal files currently opened for use. Left table shows signal segment name.ext, starting of the segment, ending of the segment and duration in seconds. The signal segment addresses may also be specified in sample indices.

5 Speech Analysis Tools

Among the requirements that must be addressed by any speech data base system are the interconnections to signal processing applications which usually are implemented on dedicated workstations. Tasks like FFT, Cepstrum, LPC analysis, formant and fundamental frequency extraction as well as analysis by synthesis procedures and psychoacoustical experimental setups including listening tests are substantially supported by having access to large signal storage volumes. Recent developments in digital signal processing (DSP) technology permit the computation of most algorithms in real-time. For example, the amplitude spectrum of a 1024 point FFT based on 32ms of speech can be computed and displayed graphically in less than 15 ms.

At this point, a change of traditional methodology has to be emphasized: since computational intensive algorithms were time-consuming in the past, an uneconomical storage of partial results was necessary in order to have amplitude spectra, parameter files etc. available for statistical analysis and further processing. Today, most speech parameters can be computed in real-time by means of DSP-technology, provided the

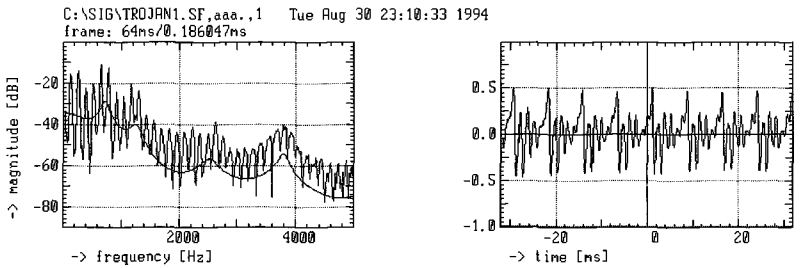


FIGURE 4 left: amplitude spectrum and LPC-smoothed spectrum of the vowel /a/, male speaker; right: waveform display. Both can be created in real-time.

computational algorithms and the signal segment data have been stored properly. This procedure requires much less storage capacity than the administration of a complete set of parameter files. Parameters can be reproduced any time they are needed on line in an application. Moreover, this new approach allows easy comparison of different analysis procedures applied on the same speech material and does not impose any constraints on cumulative processing of new speech signals.

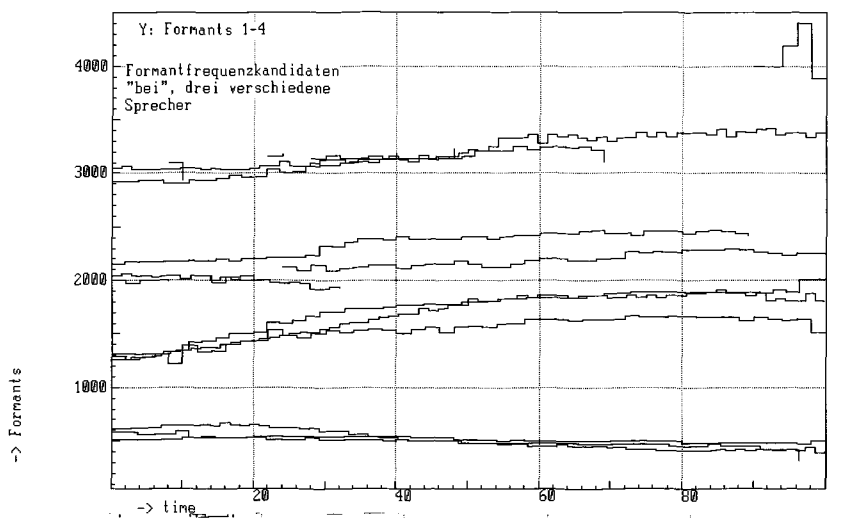


FIGURE 5 Graphical comparison of formant frequencies of voiced speech (segment /bei/: 3 male speakers).

6 Database Contents

6.1 Contents of the Austrian German database

The Austrian German database is under cumulative construction and currently contains the following corpora:

a) The Standard Austrian German Corpus is built from recordings of free running speech (open interviews, everyday topics) and standardized readings of sentence-lists of Austrian German native speakers. The sentences were designed to provide different phonological variations of diphthongs, lenitions, accent, stress etc. Each speaker provided 5 to 10 tokens of each sentence. The speech was recorded under living room conditions and in a sound-treated room for high quality recordings, respectively. The recordings have been collected within the framework of several sociophonological research projects (cf. Moosmüller 1991) as: *"Standard German in Austria varieties and norms I and II"*; *"language variation in phonetics"*; *"phonetic and phonological variations in dependence of individual psychophysiological activation"*.[9]

b) Forensic Speech Data Corpus: telephone quality speech material has been collected on different occasions including criminal cases. It contains field recordings of telephone speech, telephone transmission simulations and laboratory recordings of high sound quality for comparison. The purpose of this corpus is to evaluate intra-speaker and inter-speaker variance for specified sets of speech parameters. The Forensic Speech Corpus includes recordings from L2 German speakers with native languages (L1): Albanian, Serbian, Croatian, Russian, Polish, Italian etc. as well as speech signals from several foreign languages (L1).

The overall number of speakers currently stored in the Austrian German database sums up to 220. In average, 180 signal segmentations per speaker have been created. The database contains more than 36.000 sound signal segments in total. The number of segmentations per speaker depends on the level of segmentation already performed. Several recordings are segmented on sentence level only. Word types can vary from 10 to 400 tokens. The number of tokens on the phonetic level is considerably higher, e.g. vowels and diphthongs, (cf. Vollmann 1996). Approximately the same amount of segmented speech data is prepared to be processed in the next future.

Lexica of graphemic and phonemic forms of all words in the corpora will be built when transcriptions are completed.

[9]These projects have been funded by the Fonds zur Förderung der wissenschaftlichen Forschung: cf. P5624, P5930G; P6940-SPR; P8551-SPR, headed by Wolfgang U. Dressler.

6.2 Contents of the Child Language Database[10]

The purpose of the Child Database is to provide speech material for phonetic and morphological research in language acquisition. The speech signals are recorded at regular time intervals (two-weekly) at every day life situations (longitudinal and cross-sectional data). Children are recorded from age 1;5 to 3;0.

The database currently involves 6 corpora (i.e. children) in 4 languages (2x Austrian German, 1x Swiss French, 2x Polish, 1x Russian). Two Austrian German corpora will be processed to their total extent including utterance, word and phoneme level segmentation of the speech signal within the next future.

6.3 Aspects of Cumulative Data Acquisition

If an entire recording session is stored on disk, in practice the progress of segmentation, transcription and annotation is limited by man power mainly. One approach to overcome this limitation is to facilitate cumulative and simultaneous editing in workgroups. For that reason the S_TOOLS concept (cf. Deutsch & Noll 1994) allows the export and reimport of sound signal file directories and supports concurrent accesses to the same soundfile. Creating multiple copies of the sound file directory on different workstations improves workgroup interaction. At the end of several simultaneous editing sessions, it is necessary to keep track of whichever work of annotation belongs to which portion of a soundfile. Nevertheless, larger quantities of sound data can be processed at once by this approach.

Different segmentation strategies may be useful for different applications. For instance, acoustic phonetic analysis of speech needs narrow segmentation, whereas linguistic data analysis is primarily based on larger contiguous portions of a sound file. Consequently, the database system is capable to keep several versions of segment descriptions (several different sound file directories) for one and the same sound signal. The choice between segmentation strategies enables the user to access specified sound segments more adequately.

When speech material has been collected over a longer period of time in the framework of different research projects, various file naming conventions are likely to be used for directories and sound signal files. Therefore, standardisation throughout file and segment identifiers has to be performed in order to integrate the data into a common

[10]The child language data is being recorded within the framework of the International Child Language Acquisition Project on the "Acquisition of Pre- and Protomorphology". The Austrian workpackage of the project is funded by the Fonds zur Förderung der wissenschaftlichen Forschung, P-10250SPR.

database. This task can be accomplished by renaming segment identifiers (name.ext) as well as by restructuring sound file directories without touching or destroying the contiguity of the sound files.

7 Database Access to Speech Signal Segments

The contents of signal file directories (see figure 3) which were created during the editing and segmentation process are exported into ASCII (delimited) files; segment boundaries (starting and ending address in seconds), segment duration and segment identifiers (name.ext) are imported into the database and linked to the corresponding text entries which were written during transliteration and transcription. Although there are no commonly agreed standards of transcription for spoken language corpora up to now, the convention used is compatible to most transcription conventions in linguistics.[11] At this level of processing, the basic segment configuration can be expanded into a hierarchical architecture such as represented by phonemes as a portion of a word or words being contained in an utterance (see TYPE: below). Additional descriptors such as morphological encodings can be included by adding new field definitions. For the integration of non speech sounds and noises, an arbitrary number of fields can be opened if necessary. Links between fields and their content may be established as temporal or permanent.

In order to reproduce sound segments from the audio database via the sound system, a list of segments (selected by a specific query) is created which enters a batch process and a control structure on the acoustic workstation. Similar batch processing can be applied when sound analysis procedures and speech parameter extraction, or experimental designs have to be supplied with sound segments.

8 Structure of the Database Management System

To avoid any dependence on specific software development, the sound database concept cooperates with off-the-shelf standard database software packages which functionally are as different as for instance askSam and MS-Access.[12] Moreover the concept is independent from the specific procedure of sound signal segmentation and the acoustic workstation used as long as a minimal set of descriptors and sound segment addresses are provided:

[11] Frequently, transcriptions already exist, as they have been done with different text processing software. Text processor macros that replace field delimiters can be used to create a delimited format representation for convenient import of text into the database(s).

[12] MS-Access: Microsoft Access® .

1a. **Record session identifier:** describes the record number and establishes a link to the master tape identifier as well as to corpus and transcriptions.

1b. **Master (tape) identifier:** describes the origin of the analog or digital master tape. Usually, a master tape of analog recordings is stored in a sound or multimedia archive.

2. **Database identifier and session type:** are entered in the corpus and record session tables, respectively.

3. **Workstation identifier:** identifies the type of workstation used for data acquisition and signal processing (if applicable).

4a. **Corpus identifier:** a corpus integrates a collection of recordings belonging to one speaker, to one project or to a comparable classification. Usually, a corpus contains sets of data which were collected and/or assembled together.

4b. **Corpus type identifier:** contains the definitions of portions of different corpora which can be prepared for direct comparison.

4c. **Language identifier:** the language identification is stored in the corpus table.

5a. **Transcription:** A transcription is linked to a single contiguous record and is deassembled into individual utterances.

5b. **Word list:** each utterance can be split into word units. Each entry is either linked to a transcription line or to general record frame information (if no transcription is available).

5c. **Phoneme list:** each phonemic (or phonic) segment will be linked to a word (cf. Kiel Corpus, Kohler (ed.) 1994), to a sentence (cf. DARPA TIMIT database, Fisher et al. 1987, Pallett 1988) or to record frame information (if no transcription is available).

6. **Sound segment identifier:** specifies segment (name.ext), starting, ending address and segment duration. These data can be imported from S_TOOLS sound file directories.

7a. **Sound signal file identifier:** specifies the physical and/or logical location of sounds stored on a local or remote file system (archive, server; drive; directory; filename.ext),

7b. **File type identifier:** the file (data) type identifier enables the mapping of signal file types (and formats), such as .WAV, .SF etc. to applications and is stored in the sound signal file table.

The basics described above establish a minimal set of fields, descriptions and links which provide the capability to easily navigate through a variety of sound documents, realized by the concept of pathnames, segment names and identifiers, used to describe documents and their

components. The representation can be used for unifying and structuring material from different speech projects as well as from other sources of audio signals. It provides an open framework which allows the creation of an architecture, reflecting the logical and conceptual structure of arbitrary sound material.

8.1 Implementation in askSam®

Hypertext systems originally were designed as an interface to facilitate the access to a large amount of unstructured text in electronic form. For use with sound data, the most important feature is the access time for users to listen to the sound segments they want. AskSam (cf. North American Software 1991, Cyffka 1991, Cyffka & Bahr 1992) proved to be useful as a combined xBase-Hypertext system. AskSam simply stores text information as free text which is segmented into data sets. Database field names and field contents are entered in plain ASCII as text according to syntax rules and conventions which can be introduced simply by textual editors (e.g. "WORD[aber][13] IPA[aßɐ[14]]"). Specifying queries for specific document contents and conceptual items can be realized like free text searches (e.g. "aber {in} WORD["). Context searches are performed by specifying "aber nicht" which returns sentences like "**Aber** das habe ich gar **nicht** gesagt" etc.

In order to avoid the necessity to establish a lot of relations between several databases, a single askSam file can contain data sets with various data structures. AskSam programs combine queries which select contents according to multilevel and multistructure document models. This feature can be used to unify and standardize arbitrary data sets of different origin and structure. It enables the user to prepare the transfer of standardized data sets into an xBase environment, which accepts predefined document structures only. The extension of the procedure described includes the integration of several distributed sound and multimedia databases to larger archive systems.

In the askSam-S_TOOLS approach signal segment description as soundfile_name.ext, segment boundaries and sentence/word/phoneme descriptions are integrated into a single data set (see figure 6). Document descriptions may contain additional fields:

[13]For explanatory reasons, data base field names have been expanded in order to refer to the field content. In the actual application, field names have been restricted to two-letter-words ("SF[" for Sound File).

[14]The phonetic encoding was done due to a phonetic IPA (cf. IPA 1993) font to be used under MS-Windows applications; it may easily be converted to, e.g. SAMPA coding (cf. Wells et al. 1992), if necessary.

TYPE[WORDS] START[0,00000] END[1,00000]
DURATION[1,00000] TASKNAME[SEGMENT1.SP1]
SOUNDFILE[SPEECH.WAV] DIRECTORY[\DATA]
ORTHOGR[aber] PHONET[##?'aßW#] INFO[noise]
SPEAKER[B102] **AGE[23] RECDATE[12.04.1993]** .....$$

FIGURE 6 Example of database entries created to describe a speech sound segment in an askSam[15]-compatible representation.

A Hypertext menu-driven retrieval surface has been developed in order to provide users with a convenient interface for query formulation. Each user query is logged and monitored for further adaptation and development of improved querying possibilities resulting in advanced askSam programs and subroutines.

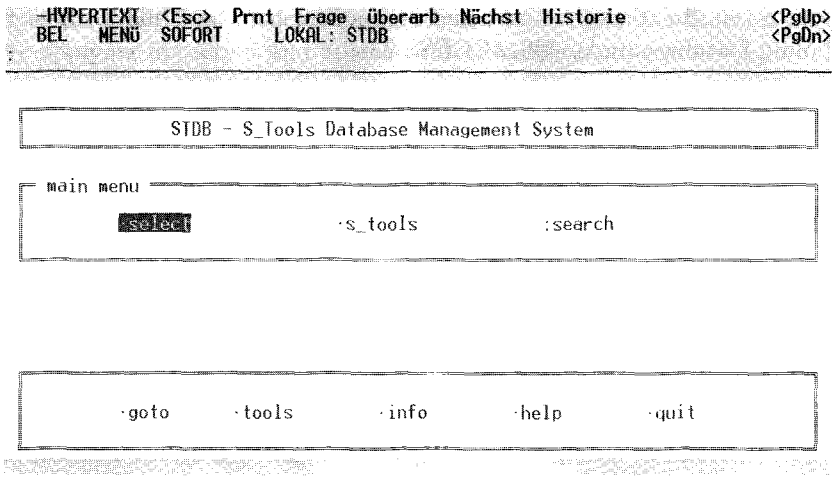


FIGURE 7 STDB Main Menu under askSam 5.0

8.2 Implementation in MS-Access

Due to the fact that the basic concept of STDBMS was planned to be independent on a specific database software, the portability of the data into different database systems is facilitated. The import of data into MS-Access environment (Microsoft Access 2.0) needed no change of the overall data structure. MS Access provides greater relationality. The current implementation integrates sound signals, still images and textual data.

[15] AskSam® is a registered trademark of North American Software.

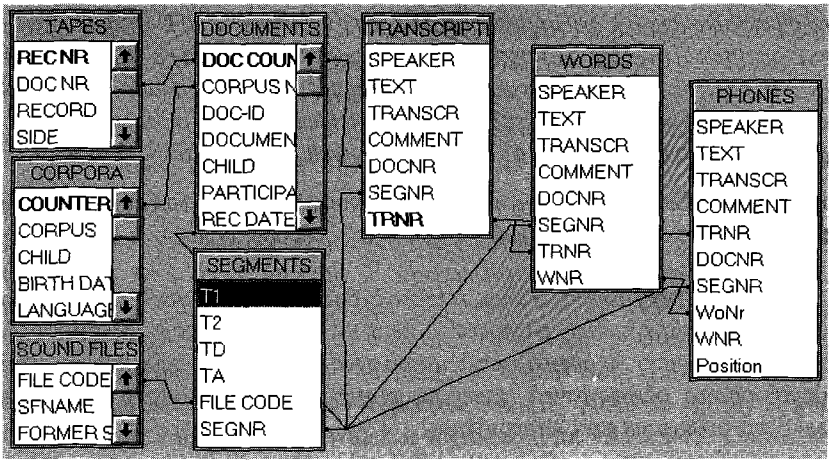


FIGURE 8 Relational schema and components of the Child Language Sound Database under MS-ACCESS 2.0

For the use of special sound (digital signal processing) hardware, an external player is necessary:

```
AppName="c:\S_TOOLS\play.exe" & [SoundFiles.Drive] &
  [SoundFiles.Subdirectory] & [SoundFiles.File] &
  [Segments.Segment]
Shell(AppName)
```

FIGURE 9 MS-Access basic commands in order to execute an external (sound) program.

The following example shows the database entry of an utterance with its morphological and phonological description. Phonetic transcriptions are given according to IPA conventions by using a MS-Windows IPA font.

Speaker	Text	Phonetic	Note
S001	Was sagst du?	v5as sakst d5u	laughs

FIGURE 10 Transcription line as represented in MS-Access: the [Phonetic] fields contain the transcriptions according to IPA convention.

9 Conclusions

One of the advantages of the implementation of sound databases in a DOS/Windows environment is the availability of a wide range of development tools and application programs including multimedia extensions. Capabilities like dynamic data interchange (DDE), dynamic linked libraries (DLL), object linking and embedding (OLE) can be used to integrate still images or to interleave audio signals and video data (AVI: audio-video interleaved) in such a way that video can be played in synchrony with sound.

Future work will integrate knowledge based systems components and multilevel semantic document models as an effective way to improve the efficiency of sound document-retrieval. Further improvements of semi-automatic data acquisition procedures are necessary in order to facilitate the transfer of large quantities of analog documents into digital format. The database concept described attempts to increase the functionality by applying hypertext and relational database components.

References

Cyffka, G. 1991. askSam: Funktionen, Befehle, Musterbeispiele. Vaterstetten: IWT-Verlag (= IWT kompakt: Datenbanken).

Cyffka, G., and H. Bahr. 1992. Erfolgreich arbeiten mit askSam: Vom einfachen Notizbuch bis hin zur komplexen Datenverwaltung. (mit Diskette) Vaterstetten: IWT-Verlag.

Deutsch, W. A. 1994. Vom Archiv zum Multimedia-Netzwerkservice. *Das audiovisuelle Archiv* 35/36: 54–61.

Deutsch, W. A., and A. Noll. 1994. Datenerfassung, Speicherung und digitale Signalverarbeitung für Akustik, Lärm, Sprache und Musik. Wien: Österreichische Akademie der Wissenschaften.

Fisher, W., V. Zue, J. Bernstein, and D. Pallett. 1987. An acoustic phonetic database. *JASA*, Suppl. A 81, S92.

IPA. 1993. The International Phonetic Alphabet (revised to 1993). *Journal of the IPA* 23: 1.

Kohler, K. J., ed. 1994. Lexica of the Kiel PHONDAT Corpus. Read Speech. Vol. I and II (= Arbeitsberichte des Instituts für Phonetik und digitale Sprachverarbeitung, Universität Kiel (AIPUK) Nr. 27, Nr. 28)

Meghini, C., F. Rabitti, and C. Thanos. 1991. Conceptual Modeling of Multimedia Documents. *IEEE Computer*, Vol. 24, No. 10, Oct. 1991: 23–30.

Moosmüller, S. 1991. Hochsprache und Dialekt in Österreich. Wien: Böhlau.

North American Software. 1991. AskSam Referenzhandbuch. München.

Pallett, D. S. 1988. Documentation for Part I of the DARPA Acoustic-Phonetic Database. ms.

Vollmann, R. 1993. Die "klingende" Datenbank—Multimedia in der Forschung. *Monitor* 9/93: 58–64.

Vollmann, R. 1996. Phonetics of Informal Speech: the Viennese Monophthongization. *Studia Phonetica Posnaniensia* 5: 87–100.

Vollmann, R., W. A. Deutsch, A. Noll, and S. Moosmüller. 1995. Konzeption einer Tondatenbank. Integration von Ton, Text, Bild und Daten in S_TOOLS_DB. *Arbeitsberichte der Forschungsstelle für Schallforschung* 1: 73–87.

Wells, J., W. Barry, M. Grice, A. Fourcin, and D. Gibbon. 1992. Standard computer-compatible transcription. Esprit project 2589 (SAM), Doc. no. SAM-UCL-037. London: Phonetics and Linguistics Dept., UCL.

6

The Reading Database of Syllable Structure

ERIK FUDGE & LINDA SHOCKEY

Every description of a language includes statements of vowel and consonant inventories. Databases to deal with such matters (e.g. UPSID) have been in existence for some time.

Some language descriptions also include statements of what sequences and other combinations may or may not occur in words of the language: phonotactic statements. Not all language descriptions actually include phonotactic statements: many treatments restrict themselves to stating inventories of vowels and consonants, and possibly tones and/or accents. Even so, such statements can in fact be made for all languages.

1 Syllable-structure

The content of phonotactic statements varies greatly in detail from language to language: Figures 1, 2, and 3 show diagrams of the basic syllable-structure for three languages—Yagua, Lhomi, and English. It can easily be seen how different they are: simple vs. complex branching, small vs. large inventories.

The first author has been collecting relevant data, and formulating such statements where possible, for a number of years (we now have these diagrams for 200+ languages). In spite of the variation between languages, it soon became clear that a common general phonotactic framework can be set up. It also became clear (a little later!) that the computer provided a logical way to store this information.

Research for this database was partially funded by the Research Endowment Fund of the University of Reading, to whom we are very grateful.

Linguistic Databases.
John Nerbonne, ed.

The aim of our database, then, is to set up such a framework for phonotactic statements for as many languages as possible. The most important units for establishing this framework are syllables. Terminology for parts of syllable structure has been worked out: Onset, Rhyme, Peak (or Nucleus), Coda (as in the diagrams). Phrase-structure relationships analogous to those of syntax are recognised between and within these parts; in fact, phrase-structure rules can be written to generate the occurring combinations.

Each structural place may be occupied by one element selected from the inventory of available sounds. Rather than a single inventory of sounds, or even an inventory divided into vowels and consonants, syllable structure may require different inventories to be available at different places in the structure (as in the diagram for English). Typically, relations of overlapping or inclusion will hold between these different inventories (e.g. m, n, w, l, r, j in both Initial and Post-initial for English).

2 Above the Syllable: the Word

Syllable-structure is, of course, only part of the story: there are larger units, particularly the word, which are needed for a full statement of the constraints. As yet this has not been incorporated in the database. Some examples of word-based phonotactic statements would be:

(a) Some Coda phenomena are restricted to word-final position (Yagua: no Codas at all except when the syllable is word-final); some are excluded from word-final position (Lhomi: /l/ as Coda in syllables other than word-final).

(b) Stressed syllables may permit bigger vowel inventories than unstressed syllables (English: only [ə] or [ɪ] are found in fully unstressed syllables).

(c) Corresponding parts of successive syllables may exhibit constraints for or against co-occurrence (e.g. vowel harmony).

(d) Some Codas may need to refer to the Onset of the following syllable (Lhomi: the element labelled 'Geminate').

3 Further Restrictions

In addition to postulating these larger units, we have to impose further restrictions on possibilities of co-occurrence: for English the sequence /smlasr/ could be fitted into Figure 3 (Pre-initial /s/, Initial /m/, Post-initial /l/, etc.), but is not in fact a permitted syllable. In the context of Pre-initial /s/ and Post-initial /l/, the inventory of possible Initials reduces to just the two-element set /p/ and /k/.

As yet, the database takes no account of restrictions of this kind.

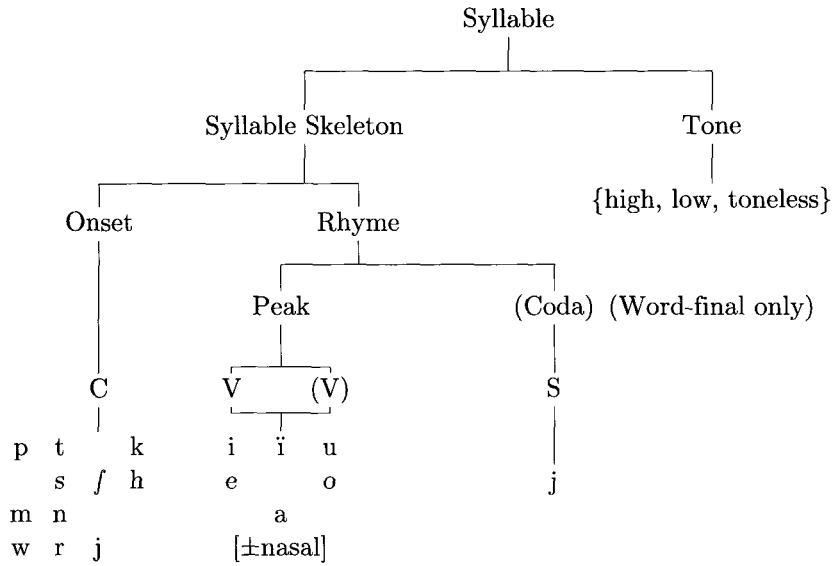


FIGURE 1 Syllable-structure of Yagua (Peru)

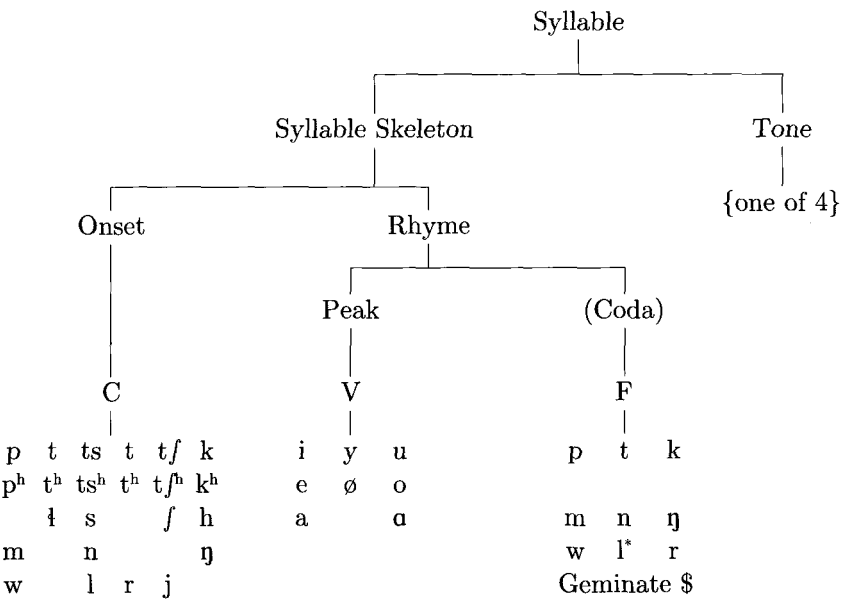


FIGURE 2 Syllable-structure of Lhomi (Nepal)

* In syllables other than word-final

$ Not word-final; consonant identical with Onset of the next syllable (only certain Onsets permitted in this case)

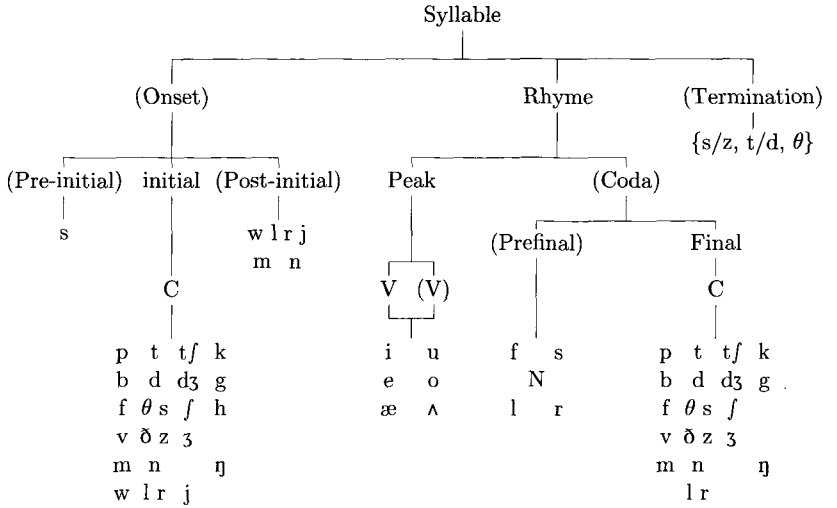


FIGURE 3 Syllable-structure of Southern British English

4 Problems

Three main problem areas are worth special attention: (a) form of source descriptions; (b) systematic vs. accidental gap?; (c) status of loanwords. Each will be discussed further in a subsection.

4.1 Form of Source Descriptions

In spite of our claim that phonotactic statements can be made for all languages, there is no guarantee that any descriptive article(s) on which the phonotactic description is to be based will be cast in anything like the same form as the ideal phonotactic statements. Where a description contains no statement at all of limitations on what sequences of consonants may occur, word-by-word inspection of data cited becomes necessary to establish such statements. In order to maintain consistency of descriptive format, it has therefore been necessary to devise a standard method for proceeding from source material to standard format (diagrams like those given here).

4.2 Systematic or Accidental Gap?

It is sometimes also necessary to reach a decision on whether absence of some combination represents a systematic gap or merely an accidental gap. For example, English words can begin with /st-/ or /tw-/: is the absence of /stw-/ systematic or accidental? For all other consonants X and Y, if /sX-/ and /XY-/ are both possible, then /sXY-/ is also

possible—this suggests the absence of /stw-/ is accidental (perhaps due to the low frequency of /tw-/).

4.3 Loanwords

The status of 'loanwords' is always difficult. Some loans are immediately 'assimilated' to the patterns of the 'borrowing' language, e.g. any word beginning with /st-/ borrowed into Spanish will split this unpermitted cluster by prefixing /e/, in effect putting the /s/ into a separate syllable of its own. Others, however, cause new phonotactic patterns to arise, e.g. many languages of the Philippines permit only single-consonant Onsets, but have imported loanwords from Spanish and English with clusters /pl-/, /tr-/ etc.: at what stage of the loaning process can we conclude that the language has developed branching Onsets?

So much for the theoretical background and the aims of this database. What of the methods by which these aims have been achieved?

5 Preparation

Two aspects of this are discussed: (a) the format of language files, and (b) the database software used.

5.1 Language Files

Information about permissible syllable structures in over 200 languages was gathered from linguistic literature as described above. These languages were from a wide variety of language families and from all over the globe.

Information was initially stored on filing cards as formulae of the type normally used by linguists. The phonemic inventory for each language was noted as well. These cards were then used as the basis for language files which were entered on a computer. Only 191 of these files were created, as complete information was not available in all cases. Entering new languages is an ongoing and (from the present point of view) nearly endless task.

5.2 Database Software

Commercially-available database software is designed to create tables and to perform arithmetic operations on the material stored in the rows and columns.

Queries about whether something is present in the database are answered by a simple lookup procedure. Clearly this form of data representation is not amenable to the storage of formulae. In using the latter it is necessary to ascertain whether a sequence or structure which is be-

ing sought can be generated by the syllable grammar, i.e. by expanding the formula.

To know whether the syllable [qi] is possible in a language or language family, one has to find out whether these phonemes are present in the inventory and whether each of them is possible in combination with the other and in that order, as well as making sure that the CV structure is permitted. ([q] might, for example, not be permitted syllable-initially or before front vowels, even if it does exist in the phonemic inventory).

It would, of course, have been possible to expand each grammar ourselves and put the resulting list in the database. The file "English" would then contain all the syllable nuclei which can stand on their own, all CV possibilities, all CCVs, all CCCVs, and so forth. There is, after all, a finite number of possible syllables in each language. For a language allowing only CV sequences and with fifteen consonants and five vowels, there would be 15×5 or 75 entries only. If both CV and CVC structures were possible, there would be 75+15×5×15 or 1200 entries. English allows anything from V to CCCVCCC, so a considerably higher number would be generated. This list could be done automatically by computer, and, as storing and searching are becoming daily easier, it would provide a viable but brute-force solution. Generating all possible syllables would also be a practical way to avoid writing rules for co-occurrence restrictions.

In the end, we decided to use software which worked with syllable grammars, even though it may be more difficult. We wished to take advantage of linguistic knowledge in the system, using what is known about natural classes and universal phonological constraints. The program which was used is based on finite-state automata and matches input against a grammar. We would like to thank Ewan Klein, Steven Bird, and Mark Ellison of the Centre for Cognitive Science of the University of Edinburgh for the use of their program (cf. Bird & Ellison 1992) for research purposes only.

6 Data Entry

At the outset of the project, we were faced with having to represent the symbols of the International Phonetic Alphabet using an ordinary typewriter keyboard. No commercially-available font could be found which would allow both for storage of data and editing of data: we could create files containing non-ASCII symbols within specified programs, but couldn't search for these symbols using either another program (such as an editor) or an operating system.

In order that our results be maximally usable by other scientists,

we chose to use the three-number codes suggested by the International Phonetic Association. These codes allow one to represent all symbols and diacritics of the IPA: 101 = [p]; 102 = [b]; 103 = [t], for example. The numbers increase with the backness of the tongue in this case.

Vowel numbers have a semi-transparent relationship with height, backness, and rounding. As these organisational principles can be used as mnemonics, the system is not hard to learn and use, though the numbers for little-used sounds and modifiers must be reconsulted from time to time.

7 Setting up a Data Base

Before you make a query, you need several kinds of information entered into files as well as the software for matching patterns with syllable grammars:

(a) A list of all the symbols you wish you use for any and all languages, and the classes they fall into, for example, vowels, high vowels, front vowels. This list must include modified symbols, so that nasalised, breathy, velarised, and all other options count as separate units (i.e. [i] will not match with nasalised [i]). This makes an enormous list and accounts to some degree for the slowness with which the program runs.

(b) A phonemic inventory for each language.

(Both (a) and (b) are stored in the same file, which contains all the classes you are ever planning to work with. We refer to it as the Class File).

(c) A syllable grammar for each language, written in a prescribed format. Here is the grammar for English:

"{ ((132) C (Engel)) V ((Engel1) Engc1) (Engf (Engf)) & [English]* & $1}"

where (132) is /s/, C is any English consonant, and Engel, Engel1, etc. are sub- classes of phonemes which are listed in the Class File.

Each grammar is kept in a separate file, which is named after the language.

8 Using the Database

The naming convention for the language grammar files allows us to recall them using any of several fields. "French.Romance.IndoEuropean.all" would be accessed in a query involving any of its subparts or in a query about all languages in the database.

It is possible to enquire about structures, specific sounds, sounds with a given feature, or any combination of the above. For example:

(a) IE "CVC" looks for CVC structures in all of the Indo-European languages

(b) all "110 H" looks for /g/ + high vowel sequences in the whole database

(c) Romance "C 103 H" looks for a consonant followed by a /t/ + high vowel in the Romance languages.

9 Results

Following are some generalisations derived from our database:

Configuration of initial clusters: (S = stop, F = fricative, N = nasal, G = glide)

shape	no. of languages	%
SS	18	9
SF	26	14
SN	20	10
SG	67	35
FN	29	15
FF	45	29
FS	34	18
FG	75	39

28 or 15% of languages allow syllable-initial three-consonant clusters.

86 or 45% of languages allow initial 2-consonant clusters.

Configuration of final clusters:

shape	no. of languages	%
SS	9	5
NS	11	6
NF	9	5
FF	13	7
GSF	8	4
NSF	8	4
FS	9	5
GC	10	5

7 or 4% of languages allow final 3-consonant clusters

17 or 9% of languages allow final 2-consonant clusters.

131 or 69% have an obligatory syllable-initial consonant.

None has an obligatory null onset.

15 or 8% have an obligatory syllable-final consonant.

23 or 12% have an obligatory null coda.

It will also be possible to ask "which languages allow both X syllable-initially and X or Y syllable-finally" and other questions about overall syllable structure.

10 Some Minor Problems

(a) We have not yet adequately confronted the details of phonotaxis, though the broad generalisations are in place. In English, for example, our current grammar gives "sdring" and "ssming" the stamp of approval, though it clearly should not. It will be possible to include constraints in the grammars to filter out false hits. We are somewhat hampered in less well-known languages by lack of information about which sequences do not occur, as mentioned in section 4.2.

(b) The program which does the matching of input with syllable grammars came to us in a compiled form, and this prevents us from making some changes which we hope to incorporate soon. These have largely to do with output format. For example, the program at present outputs not only the fact that a particular language can satisfy structure requirements specified in a query, but also lists all the ways the structure can be satisfied. So, for example, if you ask whether English has the structure CCV, it replies "spi, sti, ski, sfi, pri, pli.....". Though there are cases where we want all of this information (especially for debugging), we would often be happy with a simple 'yes,' (i.e. the program could stop searching the first time the pattern is satisfied). However, we do not have access to the code.

A change of this sort might help to make the program run faster as well. Searches are at present very time-consuming and need to be run in batch mode overnight.

11 Conclusion

We are now in a position to investigate the type of syllable universal suggested by Greenberg (1969) and to re-evaluate the work of Hooper (1976) on the sonority hierarchy of syllable structure as well as to ask a variety of new questions. We hope to improve our ability to specify co-occurrence restrictions within the syllable, though this is necessary for only a fairly small subset of the languages included. The problems we confront at the moment seem more related to implementation than to content.

References

Bird, S., and M. Ellison. 1992. *One Level Phonology: Autosegmental Representations and Rules as Finite-State Automata.* Research Paper EUCCS/RP51, University of Edinburgh Centre for Cognitive Science.

Esling, J., and H. Gaylord. 1993. Computer Codes for Phonetic Symbols. *Journal of the International Phonetic Association* 23:83–97.

Greenberg, J. 1969. Some Generalisations Concerning Initial and Final Consonant Sequences. *Linguistics* 18:5–34.

Hooper, J. B. 1976. *An Introduction to Natural Generative Phonology.* Academic Press.

7

A Database Application for the Generation of Phonetic Atlas Maps

EDGAR HAIMERL

Introduction

ALD I is a project at Salzburg University (Prof. H. Goebl) that started in 1985 aiming at the production of a phonetic dialect atlas of Dolomitic Ladinian and its neighbouring dialects. ALD stands for "atlante linguistico del ladino dolomitico e dialetti limitrofi". The dialect atlas consists of 217 inquiry points with about 1,600 responses each.

From the beginning ALD I was designed to make use of all the technical advantages of computers; the questionnaire for the ALD I has a clear structure that made it possible to convert all information into a database. From 1989 to 1991 the 217 questionnaires were entered into 217 databases in DBase-format and have meanwhile been converted into one big database with about 350,000 entries using 50 MByte of disk space.[1] Figure 1 is a screen dump showing the form to view and edit the ALD-database. There you can see the database's structure:

- ORTNR is the number of the location.
- FRAGENR contains the number of each paradigm (the ALD has 806 paradigms).
- FRVERSION counts the responses in each paradigm including multiple response.

[1] For detailed information on the project's progress see the annual publications in *Ladinia* II (Goebl 1978), *Ladinia* X (Kattenbusch and Goebl 1986), *Ladinia* XI (Szekely et al. 1987), *Ladinia* XII...XVI (Bauer et al. 1988...1991, Bauer and Goebl 1992). *Ladinia* XVII (Bauer et al. 1993) gives an overview of the project's hard- and software development. There are also presentations of the project in French (e.g. Goebl 1992 and 1994) and Italian (e.g. Bauer 1992).

Linguistic Databases.
John Nerbonne, ed.

The combination of these three numbers allows a unique identification of every entry in the database.

- STIMULUS contains the question in Italian as given in the questionnaire.
- HILFSTRANS gives the approximate response in ASCII-code.
- DEFTRANS is the ALD-coding for the phonetic transcription. We need about 350 characters, which are all built from one font. In this coding a nearly infinite number of characters is possible. The transcription is similar to the one used in other Romance dialect atlases, for instance in the AIS, the dialect atlas of Italy and the southern part of Switzerland (see Jaberg and Jud 1928).
- BEDEUTUNG allows to specify a different semantic connotation.
- PRAGMA and REGISTER contain additional linguistic information.
- NUMERUS, GENUS, GRAMMATIK are self-explanatory, in the field GRAMMATIK verbs, substantives and so on are coded.
- BEMERKUNG contains codings for multiple response.

1 Generating phonetic atlas maps with CARD

Since 1993 we have been developing a program called CARD that generates phonetic atlas maps for the ALD. The abbreviation CARD stands for '**CA**rtography and **R**etrieval of **D**ialect data'. Apart from generating maps the program can be used to enter, search and correct data, define characters and vary the layout of the maps. Figure 6 shows a clipping of an ALD map, only containing the Dolomitic Ladinian area.[2] It has been created automatically as PostScript file with CARD.

Besides generating maps for the ALD, CARD serves another task: It is a retrieval system for the ALD-database, which is to be published on CD-ROM. This dialect atlas on CD will be the first to give dialectologists the facilities of electronic data retrieval and generation of phonetic maps satisfying their specific interests.[3]

1.1 Difficulties on the way from the database to the map

Three major problems had to be solved when converting phonetic dialect data from a table to a map, i.e. a map as PostScript file:

[2] *Ladinia* XVII (Bauer et al. 1993) contains the complete map in format A2 together with five other maps.

[3] Kirk and Kretzschmar (1992) introduced software for interactive linguistic mapping. This system is based on FoxBASE+\MAC and has a well-devised architecture. Despite these advantages it cannot be used in the ALD because it does not print phonetic transcription and cannot output PostScript files for printing. Geisler (1993) introduced a program that generates atlas maps but only works on selected subsets of data.

1. How to handle the huge amount of data contained in most linguistic atlases?
2. How to decide automatically the order of multiple responses at one location?
3. How to spatialise the linguistic data taken from a table and keep the layout open to modifications?

The generation of phonetic fonts is an extra topic and is dealt with in section 2—generating phonetic characters.

1.1.1 Selecting a subset of data

The first difficulty to overcome is to select the data needed for one map out of the 350,000 entries in the ALD database within a minimum of time. As we use XBase databases only indexed access offers reasonable performance. If all the entries you need for one map succeed one another then building a subset is only sequentially copying entries as long as a condition returns true. When we use indexed data accessthe generation of a subset of data for one map takes 10 to 20 seconds (on an 80 Mhz 486 PC). Searching entries without using indexed data access the same procedure would take about 20 minutes.

Two disadvantages have to be dealt with when using indexed data access:

1. The query system is not flexible because the index files have to be generated in advance.
2. The query condition has to match exactly the data format defined in the index file, otherwise no entries are found.

AD 1—flexibility: Database applications cannot only work on databases but can also use databases to configure themselves. Instead of fixing all details in the program's code, most application details can be reduced to table structures and then stored externally. This technique is known as data driven technology. In CARD not only values can be stored externally but also names of functions which are to be called from the application. Even code blocks are possible. As all those details will be learned at run-time, the program allows a wide variety of configurations.

CARD uses a database which contains—among other entries—names of all index files and function calls to build new files or rebuild missing ones. Based on this index management CARD offers those index files as menu items which you previously defined in your configuration. Adding, deleting or changing entries in the index management table enables you to make the query possibilities fit your needs.

AD 2—data format of entries: According to the selected criterion CARD provides a form to enter the search conditions. This form is made

up in accordance with the information in the index file and transforms the input into the right format. This ensures that the query condition exactly matches the format of the fields.

Figure 5 illustrates data driven technology in CARD and shows the interdependency between the user interface in the top row, the details of the configuration stored in external databases in the middle row and the definition files at the bottom, which point to the configuration databases being used. The general definition file—at the bottom—defines options and colours. It configures editors and communicates at runtime the names of all necessary files and databases to the program. In the middle row you can find the database which contains the names of the index files and the functions used by the index management. If you follow the structure chart from top to bottom you'll see how the possibilities of selecting subsets are determined by the database of index files and function names, which itself can easily be exchanged by changing the name given in the general definition file.

1.1.2 Deciding about the order of multiple entries at one location

The second difficulty we had to face was to design a module that makes reasonable suggestions for the succession of multiple entries at one location. The difficulty arises for example in a map that contains singular and plural forms of one paradigm. Suppose you want the singular automatically precede the plural. There could be an additional response in the singular classified as rarely used which should be the last within the responses in the singular.

CARD has to check different fields of the ALD-database to establish the correct order. In a technique similar to the index management, CARD decides the order according to information stored in an external database. Figure 5 illustrates this: The program's suggestion for the correct order is based on the information provided by a database. The name of the actually used database is defined in the general definition file. The same database also holds all the information to convert numeric code into readable text; for example the numeric code 9 in the field REGISTER is to be converted to the text *rar*—meaning 'rare'—which will then appear on the map.

Figure 2 shows that this complex procedure is easy to be handled in the program: In the window FELDWAHL the user selects the fields and their hierarchical order. In our example the information in the field NUMERUS is the first criterion to decide about succession, the information in the field REGISTER is subordinate. With this selection a map will be generated where responses in the singular precede those in the

plural and where within the singular those responses restricted by extra information in REGISTER come last.

CARD collects this dispersed information, generating a new database with the same amount of entries as the subset selected for one map but with a different structure. It contains all the information needed for one entry on the map:

- The field DEFTRANS is equal to DEFTRANS in the ALD-database and codes the phonetic transcription of one response.
- The field ZUTEXT may contain additional linguistic information taken from various fields. Numerical code in the ALD-database is already resolved.
- ORDNUNG is a numeric field. Its entry is calculated by the program to propose the succession of multiple response. CARD allows to modify the field ORDNUNG and thus changes the succession or moves any entry into the legend, as we did with the map in figure 6.

1.1.3 Creating a PostScript file

Now we have collected all the information needed for one map but the biggest problem is still unsolved: A file for printer output is to be generated. CARD can generate output files in different output languages but only PostScript or TEX allow to generate the characters needed and to spatialise without restriction to lines. As with the index management and the calculation of the entries' order all details of the map definition should be stored externally. Different layouts of the map, e.g. variations in paper size, in position of the locations, in the character set or in the output language, turn out to be too different in format to be reduced to the structure of only one database. We therefore organize these parameters hierarchically by using one definition file for each layout pointing to several databases for detailed definitions and including function calls or code blocks.

The structure chart (figure 5) explains the details: A layout definition file (bottom row)

1. defines the name of the database with the co-ordinates of all locations.
2. It points to the database defining all phonetic characters. Based on this database a parser translates the coding of the ALD transcription to PostScript or TEX.
3. It points to the different files needed, for example the PostScript header file which includes the definition of all PostScript routines and the font of the character elements.

4. There is one section in the layout definition file where the details of every entry on the map are defined. Such details are
   - the fields printed on the map and their layout
   - the separators between multiple answers
   - the character that refers to the legend.

1.2 CARD can be used in similar projects

Storing configuration data in external files instead of fixing them in the program code allows a wide variety of inquiries and map layouts. This makes CARD a flexible tool not only for our purpose but allows to use the program in projects with similar aims. The forms for entering data and correcting them, the character definitions, the language of the user interface including context sensitive online help can be edited and varied without modification of the program code to fit the needs of similar projects or different sets of data.

As can be seen from the clipped map (figure 6) we made the following definitions for ALD maps:

- Answers are separated by semicolons.
- The field DEFTRANS is converted by the font module so that the phonetic transcription appears on the map.
- The content of the field ZUTEXT is put between brackets after the transcription.
- The entries are aligned left or right according to the information in the location database.

Certainly we do not expect dialectologists to create layout definition files if they only want to print ALD maps. They simply choose from a set of layout definition files by selecting a name from a list. Figure 3 shows the list in which CARD provides different layouts for ALD maps. Users familiar with the program may create new layout definitions and will find them listed in the CARD window.

Once a map exists as PostScript file, you can either use a PostSript printer to print maps or produce films, use Ghostscript to view the output or clip parts and paste them into a desktop publishing application.

These details may have caused the impression that generating maps is a time consuming procedure. The program does all this in one step using default values and takes not more than 30 seconds for selecting data from the ALD-database, calculating the succession and generating the PostScript file.

2 Generating phonetic characters

The main problem with phonetic transcriptions on a PC is the limitation to 256 characters in one font due to the 8 bit-coding in ASCII.[4] The transcription of the ALD needs about 350 characters. In our fist approach to this problem in 1988 we used several fonts and made the printer switch to the font needed before printing the character. We organized the characters in 7 fonts, one similar to ASCII, one font for each vowel and its variations and one font for the rest of the phonetic characters.[5] The program LETTRIX did the switching between fonts using the command $\backslash n$ to print all characters following this command from font number n. For example the command $\backslash 3j$ would make LETTRIX switch to font number *3* (containing the vowel "*e*" and its variations) and print the character on position *j*, which would be printed as $\dot{\acute{e}}$. The ALD-data are still coded in this way, though we don't use the program LETTRIX any more.

This first solution to the font problem allows an infinite number of characters, but switching between fonts makes printing very slow. The solution using LETTRIX was improved by our colleague Mr. G. Schiltz. He developed a TEX macro based on the LETTRIX coding that uses single character elements to combine phonetic characters. The combinations of the characters are defined in an external table in ASCII format.[6] This second solution needs only one font containing all character elements; it is not surprising that printing with TEX is much faster than using LETTRIX.

The only disadvantage of this second solution is TEX itself:

1. During map generation we would permanently have to switch between CARD generating the TEX-file and TEX compiling and printing the map.
2. If you wanted to use CARD on CD-ROM you would have to install the ALD-TEX fonts and the macro first and use TEX to print the maps.

This disadvantages in mind we developed a third solution to the font problem: it uses the transcription's coding from LETTRIX and combines it with the "one font & one table" idea of the TEX macro but directly generates PostScript output. The table's structure has been improved: Now the table is a DBase database and holds additional information about the exact position of the diacritics.

[4] Kretzschmar (1989) and Geisler (1993:148s.) suggested solutions to the problem of displaying and entering phonetic characters.

[5] See Bauer et al. (1988:30–9).

[6] See Schiltz in Bauer et al. (1991:238–244) and Kelle and Schiltz (1993).

CARD provides a form (figure 4) to define or edit characters. The field ZEICHENCODE contains the code, \3j as in our example. The column entitled ASCII-WERT contains the ASCII coding of the character elements used to compose the phonetic character. Each of the diacritics can be exactly adjusted by moving it horizontally or vertically (see values in the columns entitled X-RICHTUNG and Y-RICHTUNG). The maximum number of 8 diacritics serves all needs of the ALD transcription (see the fields AKZENTO1 ... AKZENTO5; AKZENTU1 ... AKZENTU3) but could easily be enlarged if necessary.

The entry in the field PS-CODE at the bottom of the form is automatically created by CARD and shows the internal PostScript code used in the map files. The function A at the end of the code expects three entries on the stack:

1. a character (here *e* on ASCII position 101, which is the BASIS-ZEICHEN)
2. a function (here 'scale', redefined to SC with two parameters)
3. an array of arrays, each array containing the information for one diacritic and its x- and y-translation. In the example the diacritic on position 196 is moved 3/100 of the font size to the right and 5/100 upwards.

The system allows to print out single characters in A4 size to control the exact position of the elements. All you have to do is to enter a filename in the field AUSDRUCK IN DATEI and to send this PostScript file to the printer.

This third solution to the problem of font generation for phonetic transcriptions has three advantages:

1. The number of characters is only limited by the maximum number of character elements (256 elements are possible, the ALD transcription needs only 83 elements).
2. The diacritics have an exact position fitting the character's width.
3. It is easy to generate new characters and thus make the transcription system match the need of various phonetic phenomena.

We have been using CARD successfully for generating maps for two years now and keep improving the program. Latest announcements on the ALD project and it's software development are available via WWW at `http://www.edvz.sbg.ac.at/rom/home.htm`—section *Projekte.* By the end of 1996 it should be possible to download a demo version of CARD from the ALD-page.

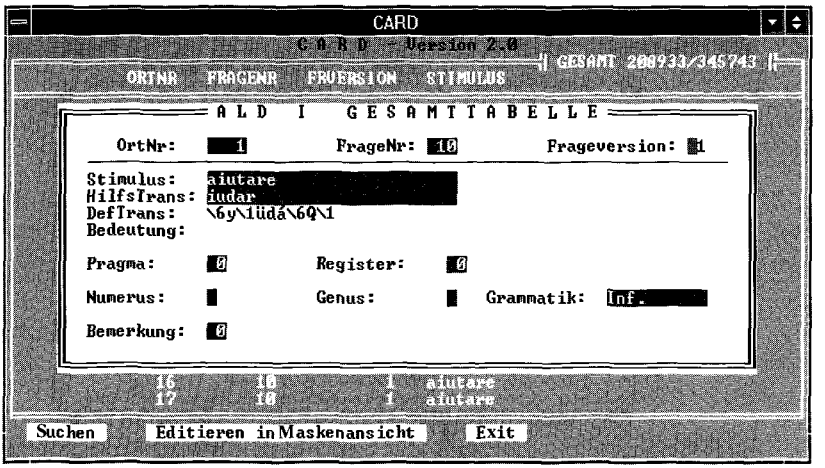


Fig. 1: CARD can be used to view and edit the ALD database

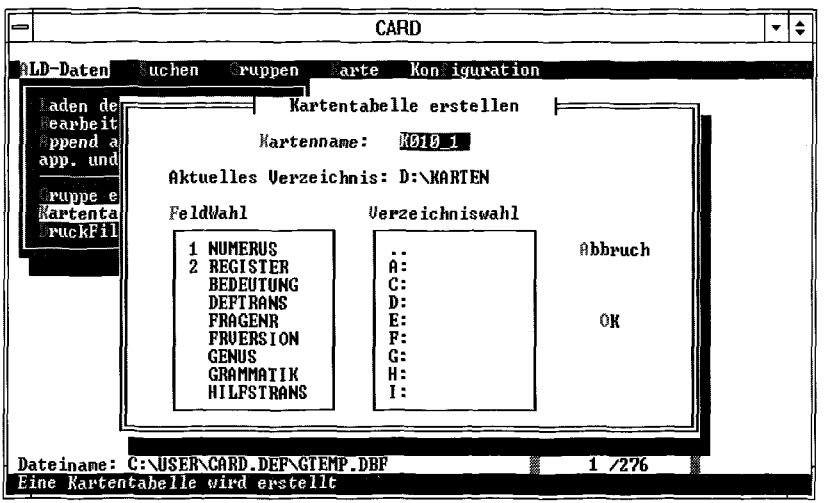


Fig. 2: Based on the selected fields (here Numerus, Register) CARD calculates the order of multiple entries

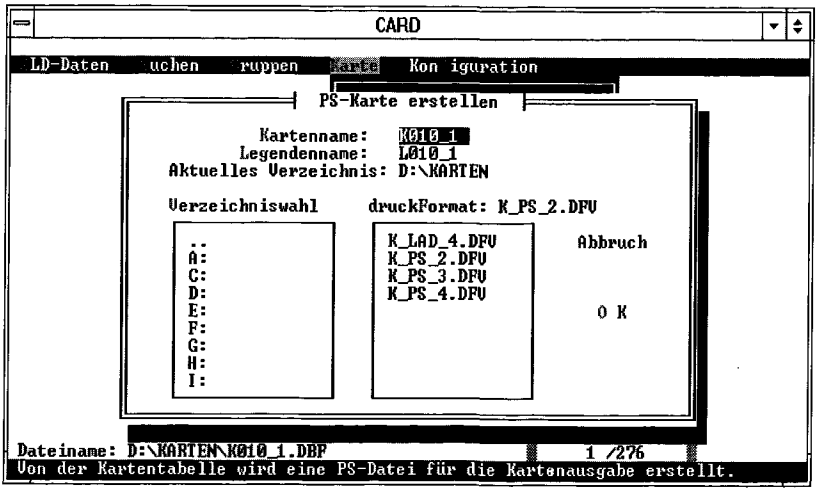


Fig. 3: A map is generated; the layout depends on the layout definition file selected in the DRUCKFORMAT- window

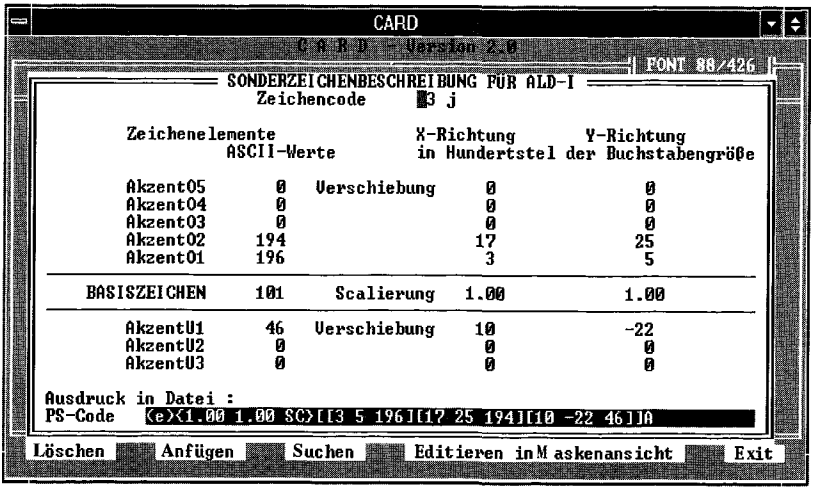


Fig. 4: Defining phonetic characters in CARD

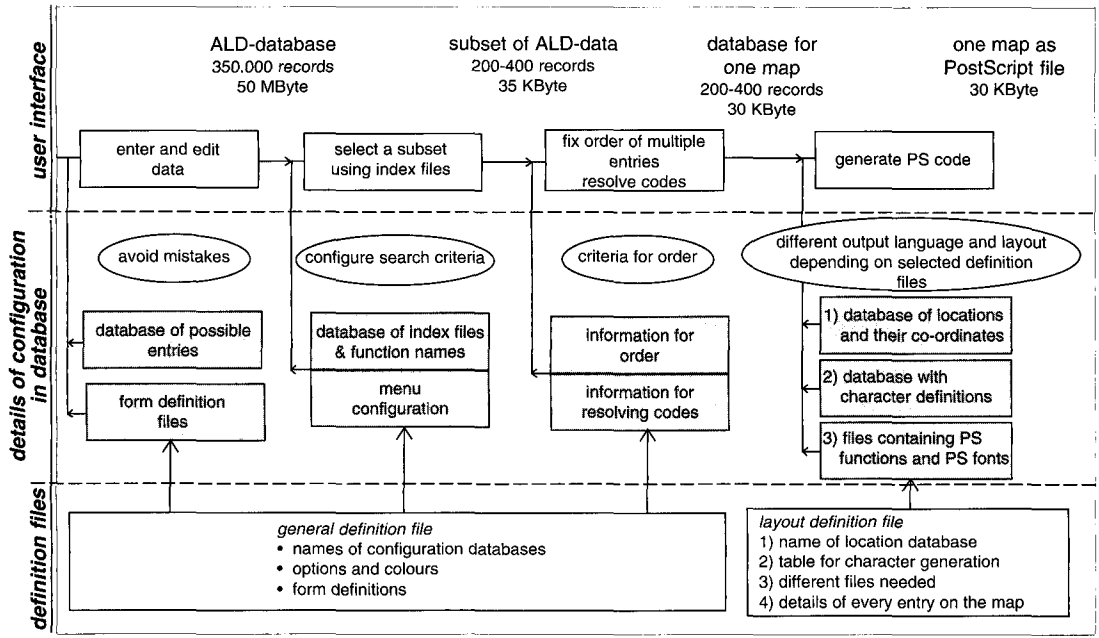


Fig. 5: Data Driven Technology as used in CARD

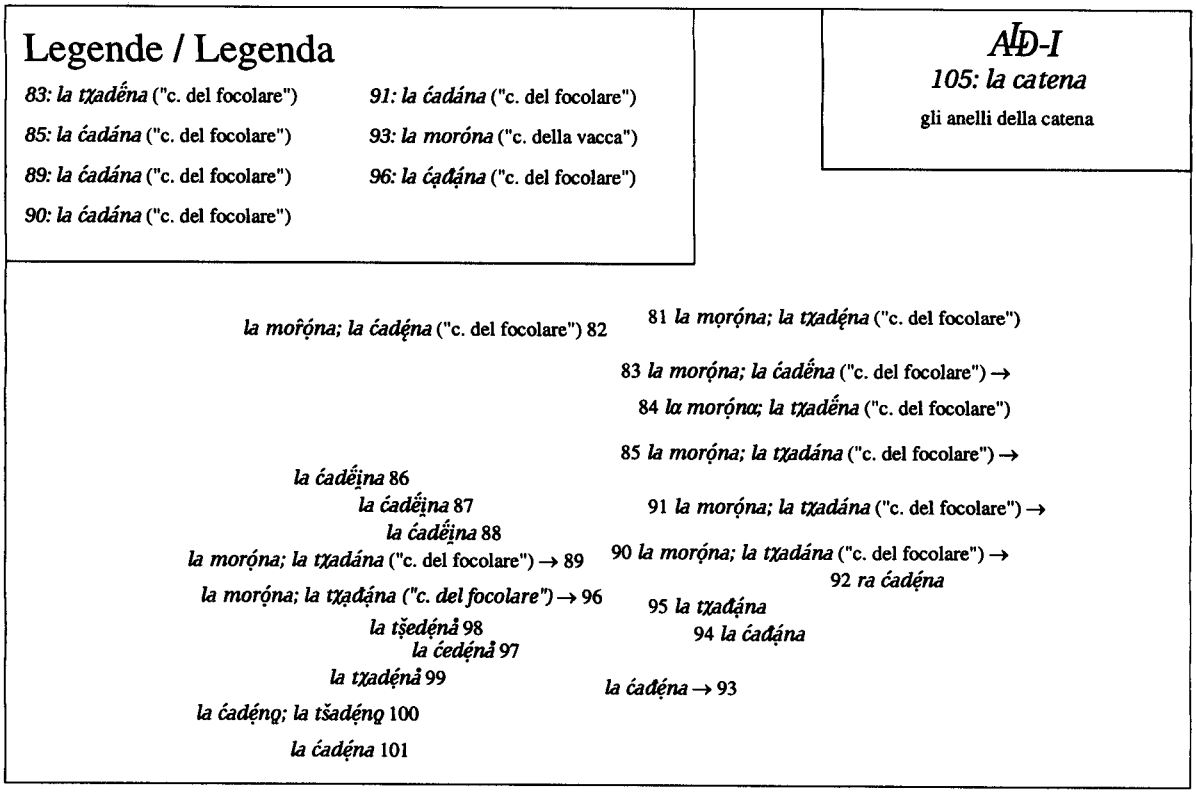


Fig.6: Clipping of an ALD-map; the Dolomitic Ladinian Area

References

Bauer, R. 1992. L'informatizzazione dell'atlante linguistico sonoro ALD I (Atlante linguistico del ladino centrale e dialetti limitrofi I). *Linguistica* XXXII:197–212.

Bauer, R., H. Böhmer, S. Gislimberti, H. Goebl, R. Köhler, M. Schleusser, T. Szekely, and H. Tyroller. 1990. Arbeitsbericht 5 zum ALD I—Relazione di lavoro 5 per l' ALD I. *Ladinia* XIV:259–304.

Bauer, R., H. Böhmer, H. Goebl, E. Haimerl, G. Schiltz, and H. Tyroller. 1991. Arbeitsbericht 6 zum ALD I. *Ladinia* XV:203–54.

Bauer, R., S. Gislimberti, E. Perini, T. Szekely, and H. Goebl. 1988. Arbeitsbericht 3 zum ALD I—Relazione di lavoro 3 per l' ALD I. *Ladinia* XII:17–56.

Bauer, R., and H. Goebl. 1992. Arbeitsbericht 7 zum ALD I. *Ladinia* XVI:169–84.

Bauer, R., H. Goebl, T. Szekely, S. Gislimberti, and E. Perini. 1989. Arbeitsbericht 4 zum ALD I—Relazione di lavoro 4 per l'ALD I. *Ladinia* XIII:185–229.

Bauer, R., E. Haimerl, and H. Goebl. 1993. Arbeitsbericht 8 zum ALD I. *Ladinia* XVII:125–52.

Geisler, H. 1993. Dialect data processing with personal computers. In Wolfgang Viereck, ed., *Verhandlungen des Internationalen Dialektologenkongresses [Bamberg, 29 July–4 August, 1990]*, Vol.1:147–62. Stuttgart: Steiner.

Goebl, H. 1978. Ein Sprach- und Sachatlas des Zentralrätoromanischen (ALD). *Ladinia* II:19–33.

Goebl, H. 1992. L'atlas parlant dans le cadre de l'Atlas linguistique du ladin central et des dialectes limitrophes (ALD). In *Actes du congrès international de dialectologie / Nazioarteko dialektologia biltzarra. Agiriak [Bilbao / Bilbo 21–25 October, 1991]*, 397–441. Bilbao / Bilbo: Académie de la langue basque / Euskaltzaindia.

Goebl, H. 1994. L'Atlas linguistique du ladin central et des dialectes limitrophes (première partie, ALD I). In Pilar Garcia Mouton, ed., *Geolingüística. Trabajos europeos*, 155–68. Madrid: Consejo Superior de Investigaciones Científicas.

Jaberg, K., and J. Jud. 1928. *Der Sprachatlas als Forschungsinstrument: Kritische Grundlegung und Einführung in den Sprach- und Sachatlas Italiens und der Südschweiz.* Halle (Saale): Niemeyer.

Kattenbusch, D., and H. Goebl. 1986. Die ersten Enqueten für den ALD I—Erfahrungen und Ergebnisse (ALD-Arbeitsbericht 1). *Ladinia* X:5–32.

Kelle, B., and G. Schiltz. 1993. Die Wiedergabe phonetischer Schriftzeichen in der automatisierten Sprachatlas- und Druckvorlagenherstellung. In Wolfgang Viereck, ed., *Verhandlungen des Internationalen Dialektologenkongresses [Bamberg, 29 July–4 August, 1990]*, 1, 240–52. Stuttgart: Steiner.

Kirk, J. M., and W. A. Kretzschmar. 1992. Interactive linguistic mapping of dialect features. *Literary and Linguistic Computing,* Vol. 7, No. 3:168–75.

Kretzschmar, W. A. 1989. Phonetic display and output. *Journal of English Linguistics-special issue: Computer Methods in Dialectology,* 47–53.

Szekely, T., E. Perini, S. Gislimberti, and H. Goebl. 1987. Arbeitsbericht 2 zum ALD I—Relazione di lavoro 2 per l'ALD I. *Ladinia* XI:183–218.

8

Swiss French PolyPhone and PolyVar: Telephone Speech Databases to Model Inter- and Intra-speaker Variability

GERARD CHOLLET, JEAN-LUC COCHARD, ANDREI CONSTANTINESCU, CEDRIC JABOULET, & PHILIPPE LANGLAIS

Introduction

Following the demand of the speech technology market, a number of companies and research laboratories joined their forces in order to produce valuable and reusable resources, especially speech databases. Examples of such existing consortia are **COCOSDA** or **SPEECHDAT**, where the latter aims at developing multilingual PolyPhone speech databases recorded over the telephone. Other existing PolyPhone database are MacroPhone (Bernstein et al. 1994), Dutch Polyphone, Voice across Hispanic America. MacroPhone consists of approximately 200,000 utterances by 5000 speakers, who were recorded in the United States. For the distribution of these and many other speech corpora, different national and international database centers have emerged. **LDC** (**L**inguistic **D**ata **C**onsortium), **ELRA** (**E**uropean **L**anguage **R**esources **A**ssociation), and **BAS** (**B**avarian **A**rchive for **S**peech Signals), shall be named here as examples.

Serving their purpose, the collected databases are used for developing, testing, enhancing and evaluating speech technology products, like interactive voice servers, listening typewriter, speaker verification and identification systems (Sorin et al. 1994, Niedermair 1994, Choukri

Linguistic Databases.
John Nerbonne, ed.

1994, MACIF 1994, Strawe 1994), *etc.* Especially for capturing intra-speaker variability, the PolyVar database was designed and recorded at IDIAP, as a complement to the Swiss French PolyPhone database, which addresses inter-speaker variability issues.

1 General overview

We will detail in the following the specific problems of speech database collection (sampling the speaker population, selection of vocabulary items, ...), and will present actual development we carried out at IDIAP through the PolyPhone and PolyVar databases.

Figure 1 shows the overall processing of database collection that we will describe in the following sections.

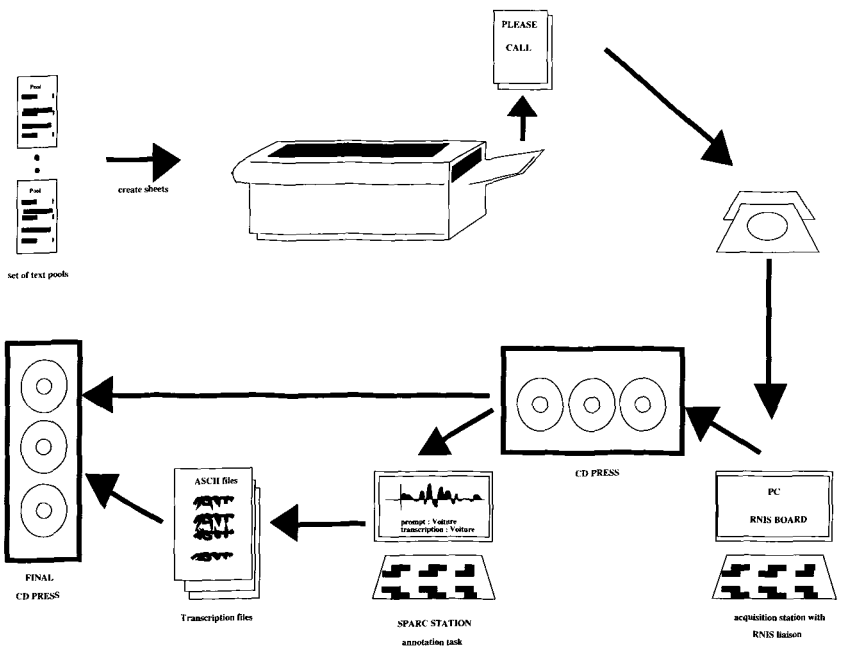


FIGURE 1 Overview of the collection database process.

2 Database specifications

2.1 Contents

The goal of the Swiss French PolyPhone database, was to record 5000 speakers over the telephone network. Therefore, persons from all parts of French speaking Switzerland were approached and received calling

sheets. Each sheet distributed to the speakers was made up of 38 items. The desired utterances were:

1 sequence of 6 single digits including the hash (#) and the star ($\star$) symbols
1 sheet id number (5 connected digits / 1 natural number)
1 telephone number (spontaneous)
1 16-digit credit card number
2 natural numbers (1 + sheet id)
2 currency amounts
1 quantity
3 spelled words
1 time phrase (prompted, word style)
1 date (spontaneous)
1 date (prompted)
1 yes/no question
1 city name (prompted)
1 city name (spontaneous)
5 application words from a list of 105
1 name (spelling table)
1 mother tongue (spontaneous)
1 education level
1 telephone type
10 sentences (read)
1 query to telephone directory (given the name and the city of subject)
1 free comment on the call

The application word list, with a total of 105 entries, showed control words for different possible voice servers, e.g. telebanking, information kiosk, and ticket reservation.

As a complement to the PolyPhone database, the collection of the PolyVar database was decided and started. The contents of the latter corpora, which was designed with regards to intra-speaker variability, are similar to those of its sibling, PolyPhone.

2.2 Speakers

Speaker recruitment for the Swiss PolyPhone database was taken care of by *LINK*, a marketing firm in Lausanne. They addressed people from all over French speaking part of Switzerland. The coverage of regions is proportional to their size: the number of speakers from every city

depends on the population size. Other sampling criteria applied in order to obtain a broad demographic coverage were:

- Sex
- Age (–29, 30–49 and 50+ years)
- Employment status (full-time, part-time, unemployed)

The participation rate of the people approached, was rather high and resulted as follows:

- 66% agreed to call the server when asked to participate,
- 63% of those who agreed did really call,
- which gives a total of 42% effective participants.

Unlike for the PolyPhone database, the speakers participating in the collection of the PolyVar database were not selected by any demographic means. It was rather the availability criteria which imposed fundamental constraints, delimiting candidates mainly to members of IDIAP, their families, and befriended researchers. However, it was tried at least to balance the number of male and female speakers to some degree. The demographic unbalance, which at first might be rated negative, does not appear grave at a second thought: it is expectable, that intra-speaker variability will hardly depend on the person's age or social background, if recordings are collected during a period of a few months.

2.3 Generation of calling sheets

As already mentioned, each speaker was asked to read 10 sentences. In order to ensure good phonetic coverage for the resulting database, a **greedy** algorithm was used to select the sentences from some different corpora (Newspapers, Internet texts,...). The sheet generation program computes the phonetic value of possible sentences and compares them to the actual phonetic coverage (see figure 2) resulting from the annotated calls. The phonetic value of a sentence is defined as the sum of phoneme, diphone, triphone, and polyphone enhancements obtained, when compared to the actual coverage. In this context, the expected enhancement brought to a class (phonemes, diphones, triphones, polyphones) is the square of the subtraction result of each element in the class to the mean average of all elements (see figure 3).

A similar greedy algorithm is applied to numbers, since their pronunciations vary considerably depending on the context surrounding them. Therefore, the left and right adjacent phonemes are taken into account for each number word. Thus, by selecting numbers with respect to the context in which the included number words are found, a good sample is obtained.

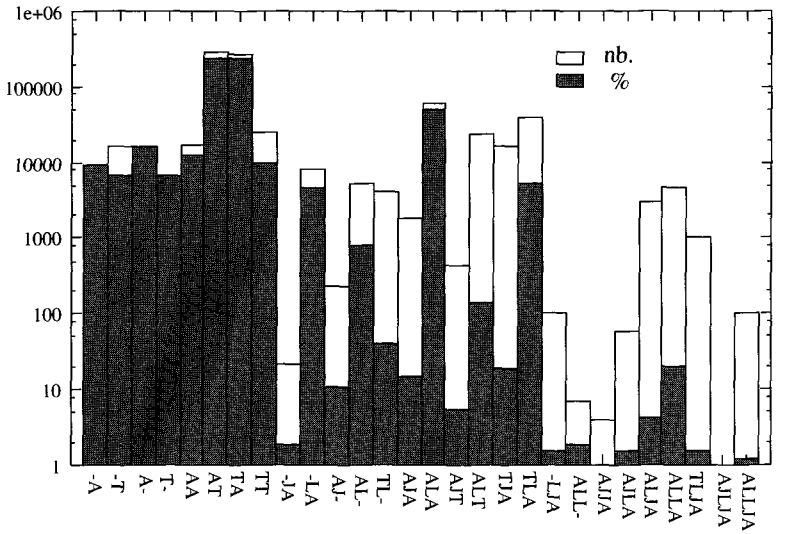


FIGURE 2 Example of polyphones coverage.

The spelled words are selected from a list of names and titles. Here again, a greedy algorithm ensures a quite homogeneous distribution of letters and special characters (e.g. accents) in several contexts used for spelling.

Finally, the function words to appear in the sheets, are determined randomly from the function word list.

As can be seen in figure 2, even with large amount of sentences (more than 20,000) there is a lack of several units (here polyphones). This points out a crucial problem from speech recognition point of view, since even rare units need to be modeled. In the near future, we will need to take into account sentence generation technology in order to fill the gaps of phonetic units and also to be able to learn linguistic structures (syntactic, semantic, ...) that will be integrated to comprehension task. In order to be aware of the problem of phonetic coverage, we show in figure 4 the evolution of the polyphones coverage with the number of sentences.

3 Technical details

3.1 Recording

For the recording of the PolyPhone calls, a Rhetorex platform was used. The recording site was located at Bern and was operated by the **Swiss Telecom PTT** on SwissNet II (Swiss ISDN). Thanks to this platform, 15 ISDN lines could be monitored and recorded from in parallel. The

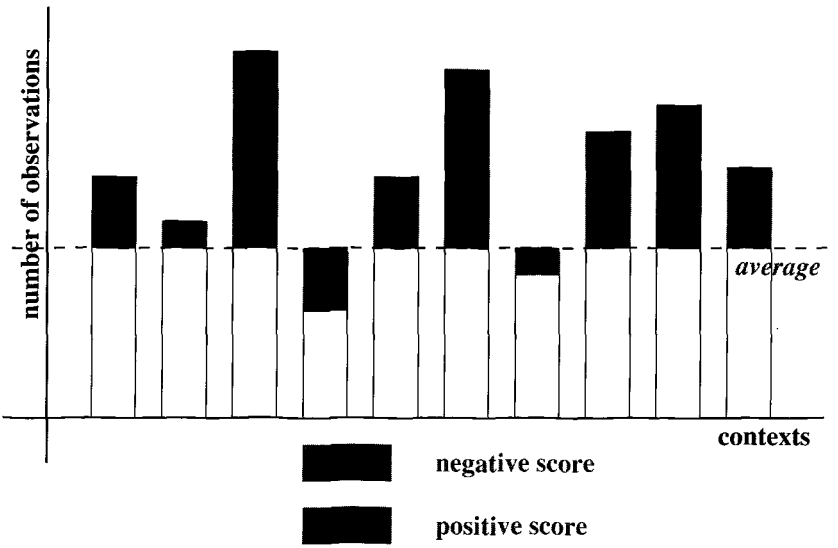


FIGURE 3 Principle of the greedy algorithm for coverage purpose.

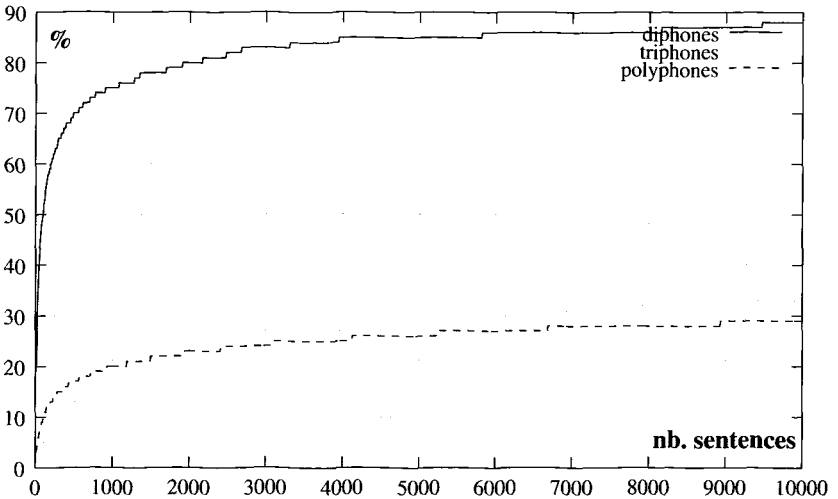


FIGURE 4 diphones, triphones and polyphones coverages compared to the number of sentences analysed.

recordings were stored on CD-ROM, using 8 bit A-law, and then sent to IDIAP for post-processing.

The PolyVar calls were recorded at IDIAP on a PC based platform, using a DialSys board commercialised by **ACSYS**. This board offers an analog telephone interface.

We observed in several experiments differences in the two kinds of recordings that show the importance of telephone line adaptation.

3.2 Final format of speech database

The processed and annotated calls are compressed with the SHORTEN algorithm (which proved to give a profit of 20% for PolyPhone recordings and 40% for PolyVar calls) and stored on CD-ROMs. Each item of the prompt sheet is stored in a separate file with a NIST header.

This is an example of a NIST header for an item of a call. All but the first 8 fields are specified by NIST standard:

```
database_id -s26 Swiss French Polyphone 0.0
recording_site -s17 Swiss Telecom PTT
sheet_id -i 13946
prompt -s26 Informations consommateurs
text_transcription -s26 Informations consommateurs
speaking_mode -s4 read
sample_begin -r 0.200000
sample_end -r 2.725125
sample_count -i 23401
sample_n_bytes -i 2
channel_count -i 1
sample_coding -s26 pcm,embedded-shorten-v1.09
sample_rate -i 8000
sample_byte_format -s2 10
sample_checksum -i 12379
```

Field one names the database, field two stores the recording site, the third field value is the calling sheet id. In the two following fields the prompted text and the orthographic transcription of the real utterance are included. The sixth field explains whether the utterance was "read" or "spontaneous". Field seven and eight, label beginning and end of the usable speech (in seconds). When possible, the recorded signal was cut 200 ms before and 200 ms after the usable speech segment.

4 Transcription

In the context of the SpeechDat project, international annotation guidelines for speech databases were developed as a common basis, allowing

for supplementary individual extensions to the standards agreed upon. For the Swiss French PolyPhone and PolyVar databases, verification of orthographic transcription of the recorded utterances was performed at IDIAP after receiving collected calls on CD-ROMs. For further processing, the calls were converted to 16 bits linear.

4.1 General guidelines for orthographic transcription

The orthographic transcription is intended to be done at a lexical level. Only few acoustic events are mentioned in the transcription: unintelligible words, unusual pronunciation of words or letters, hesitations on parts of sentences. The transcriber should write what was really pronounced, and not what the speaker was supposed to say.

The transcription is intended to be a quick procedure. Transcribers should not have to agonize over decisions, but rather realize that their transcription is intended to be a rough guide that others may examine further for details.

All items (even numbers, date, time, ...) are transcribed in letters; abbreviations are not used, unless they are spoken in their abbreviated form.

One objective of the transcription is to keep as much speech files as possible in the corpus. Thus we try to avoid deleting items due to extra noises or disfluencies.

Moreover, general rules for written French should be respected during the transcription. For example, the transcription is case sensitive, and all proper names are to be capitalized.

4.2 Transcription interface

We developed a tool called **ANNOTATOR** for the purpose of verifying and correcting the transcription of utterances. We are about to include the phonetic transcription, which at present is performed in a separate step, directly into the annotation tool.

The ANNOTATOR interface (see figure 5) works under SunOS or Solaris on Sun workstations. Those must have audio equipment. Moreover, the annotation tool requires an installed version of Xwaves from **Entropic** and the public domain package TCL-TK.

5 Automatic processing

The system $\mathsf{ETC}_{\mathsf{vérif}}$ (Cochard and Froidevaux 1995), for "Environment for Cooperative Treatment applied to the verification of speech samples" ("*E*nvironnement de *T*raitement *C*oopératif appliqué à la *vérif*ication d'échantillons de parole"), under development in our laboratory, is in-

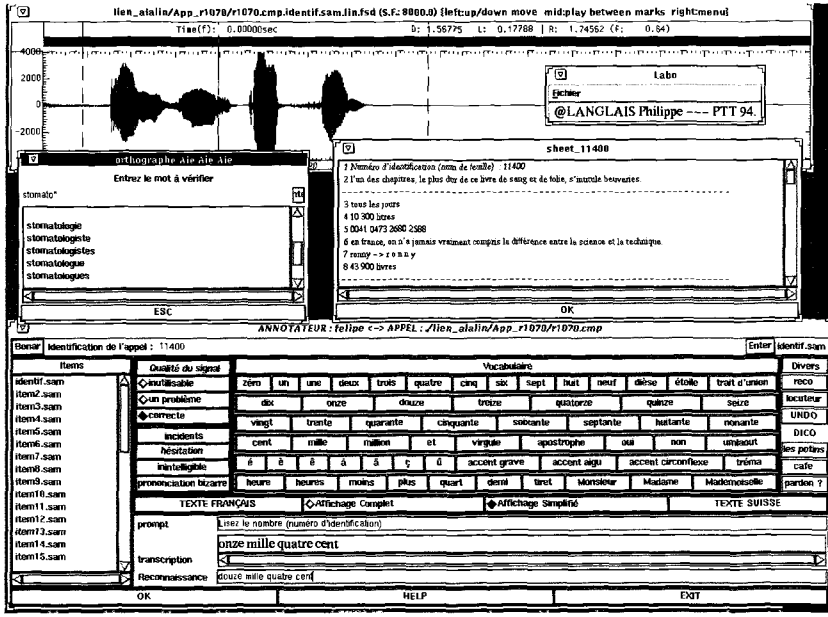


FIGURE 5 ANNOTATOR interface for transcription purpose.

tended to provide valuable help for the difficult task of phonetic labeling of speech corpora.

The task dedicated to $\mathsf{ETC}_{\mathsf{vérif}}$ is to compare a given speech sample to a given written text, and to decide whether the utterance is or is not correct. This decision is based on a processing of two streams of input data: the speech sample, on one hand, and the prompted text, on the other hand, using different knowledge sources (KSs). $\mathsf{ETC}_{\mathsf{vérif}}$ is defined as a multi-agent system, decomposed into two layers (see figure 6): a kernel, a general purpose platform called ETC that is application-independent, and a periphery that bears all the knowledge of the application domains. The ETC platform provides different services to the host application:

- a model of Ω-*agents* that defines a standard interface to distinct entities representing specific areas of expertise;
- a model of μ-*agents* that gives rise to the notion of active data, allowing a decentralized structuring of local hypotheses;
- a framework for the definition of the system evolution dynamics.

The original intention behind the selection of a multi-agent approach is two-fold. Firstly, we consider this approach as the most promising one to be successful in building a system that offers the capability to exchange or add KSs. Secondly, this approach gives us the means to pre-

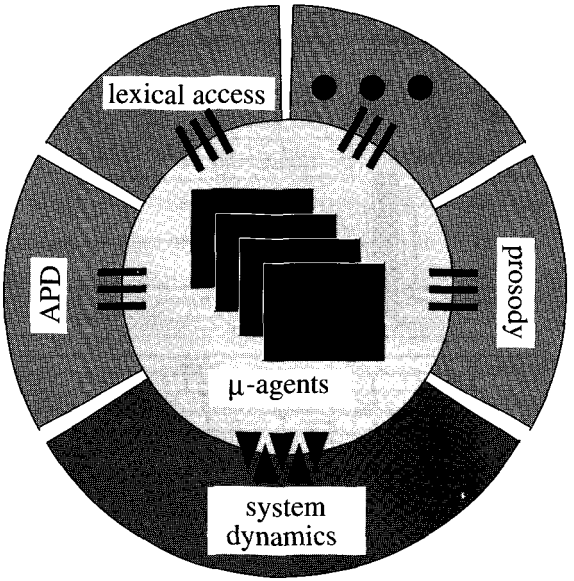


FIGURE 6 General overview of $\mathsf{ETC}_{\text{vérif}}$ architecture.

vent the combinatory explosion problem that occurs with an exhaustive search in a huge solution space.

Now, we clearly see that it is possible to favorably combine $\mathsf{ETC}_{\text{vérif}}$ approach and human interaction. One can find here an illustration of our global approach and its capability to integrate new agents actively contributing to the resolution process. In this context, the user is in position, through a graphical interface, to intervene in the resolution process, as any other agent can do, by making local propositions that point the system to a distinct area of the solution space.

Thus the phonetic transcription of large speech corpora will become a reasonable task as the system can provide online assistance with a phonetic alignment that can integrate some local user-made modifications. In a fully automatic or in a user-assisted mode, the system keeps the control of the final solution. This particular feature of $\mathsf{ETC}_{\text{vérif}}$ if we are able to set up robust evaluation strategies, will provide the guarantee of the reliability of annotated databases.

5.1 Ω-agent, provider of solutions

An *Ω-agent* is a conventional problem solving module surrounded by a uniform interface. Each Ω-agent represents a specific area of expertise; it is thus able to provide local solutions to problems falling in the scope

of its competence. As introduced by Lander (1994, p. 12), each Ω-agent can be queried in three different ways:

initial solution – the agent competence is requested to generate a proposal that solves a local sub-problem and that can be used as part of a composite solution.

local alternative – the agent competence is requested to provide an alternative solution with a focus of attention on what was conflicting in the previous proposal(s).

criticism – the agent competence is requested to evaluate a proposal in order to see if it can fit its own requirements.

These three primitives are sufficient to possibly ensure an entire covering of each agent solution space. But, due to the focus mechanism that plays a role of a heuristic function, there is no guarantee that, for a particular problem, every position in a particular solution space will be reached.

In $\mathsf{ETC}_{\text{vérif}}$, the set of Ω-agents currently available are:

- HMMs for phonemes and words (built from the Polyphone),
- a bottom-up, rule-based, acoustic-phonetic decoder (Méloni and Gilles 1991),
- an hybrid LVQ-HMM system for phonemes recognition (Torkkola 1994),
- a grapheme to phoneme transcription based on BDLEX (Pérennou et al. 1992),
- a macro-prosodic module that labels syntactic constituents boundaries (Langlais and Cochard 1995).

5.2 μ-agent, element of a composite solution

Each atomic result provided by an Ω-agent, e.g. a phonetic label, a word-form, a prosodic label, etc., is encapsulated in a framework that defines its interacting capabilities, called a *μ-agent*. Each type of atomic result is assigned a distinct set of behaviors. In the Figure 7, an aspect of each μ-agent is addressed, namely its capability to weave relations with other μ-agents. In this artificial situation, three kind of relations are presented:

solid line – express neighborhood of phonetic labels based on temporal information or of word-forms based on writing convention.

solid line with white arrows – shows a case of phonetic clustering. Two distinct answers have been provided within the same class of phonemes on compatible temporal intervals.

dashed line – denotes the fact that a local solution has been used by an Ω-agent to produce other solutions.

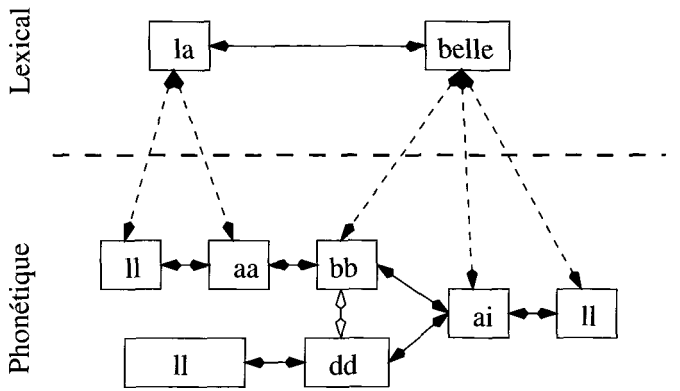


FIGURE 7 A partial state of a composite solution during a simulated processing of the sentence "*La belle ferme le voile*".

The purpose of all these relations is two-fold. Firstly, they are used to define a simple path traversing algorithm, in order to combine atomic solutions in a coherent composite solution. Secondly, the relations play a predominant role in the dynamics of the system as they are central in the cycles of evaluation-improvement of the current solution with the final objective to have a solution with a high enough confidence rate.

5.3 The dynamics of the system

For each global configuration of μ-agents, the system has to evaluate the current set of composite solutions. If a particular composite solution (a path from left to right in Figure 7) get a high confidence score, this solution is provided as *the* answer to the global problem. Otherwise, a local improvement process is started on relatively low confidence μ-agents areas of the best solution. This cycle is performed as long as no sufficiently good solution is available and as far as improvements are still possible.

The determination of a good solution is directly related to the fact that an atomic result is assigned a reliability estimate. The computation of a reliability measure is based on the following formulas 1–3:

$$(1) \qquad C(A) = \lim_{i \to \infty} C_i(A)$$

$$(2) \qquad C_i(A) = 1 - \sum_{j=1}^{i} \frac{1}{(2 + Ce_{j-1}(A))^j}, \text{ for } i \geq 1$$

$$C_0(A) = 1$$

$$(3) \qquad Ce_i(A) = \sum_{B=1}^{N} V(A,B)\, C_i(B)$$

The reliability of a μ-agent A, $C(A)$, is defined as the value on which

$C_i(A)$ is converging. The definition of $C_i(A)$ insures that successive values converge in the interval $]0,1[$ since $Ce_i(A) > 0$. $Ce_i(A)$ denotes the contribution of A neighbourhood in the computation of A reliability. The definition of $Ce_i(A)$ relies on the assumption that all μ-agents are denoted by natural numbers from 1 to N and $V(A,B) = 1$ if A is directly connected to B, $V(A,B) = 0$ otherwise.

Figure 8 gives a graphical illustration of the evolution of $C_i(A)$ for all μ-agents mentioned in the figure 7.

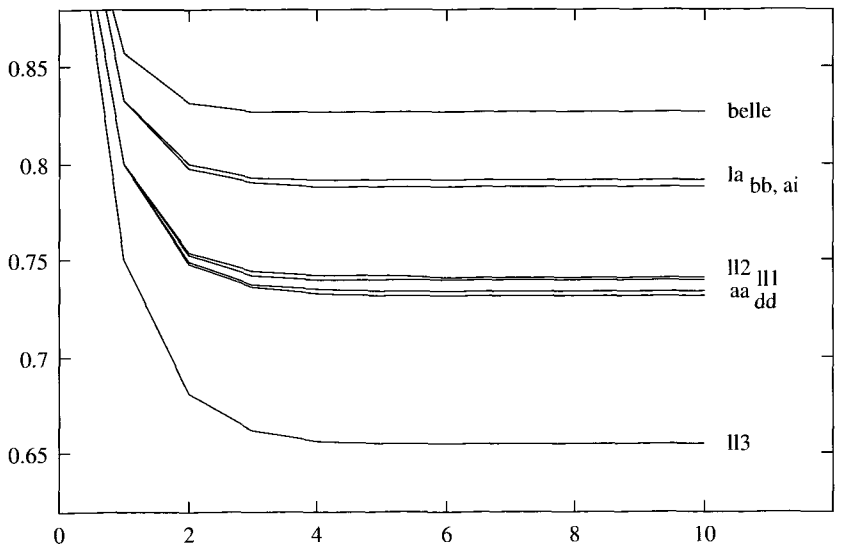


FIGURE 8 Graphical illustration of the convergence property of the currently implemented reliability measure.

Once the reliability measure is computed for all μ-agents currently present in the system, an average reliability value $\bar{C}$ is computed for each path that covers the entire speech sample. The path with the highest $\bar{C}$, called $\bar{C}_M$ is selected as a focus of interest. Two situations may occur at this point, either $\bar{C}_M$ is higher than a predefined threshold, or it is lower. In the first case, nothing has to be done; the system is considered to be successful in solving the problem. In the second case, an attempt to improve the current best composite solution is performed by requesting local alternatives to Ω-agents responsible of low reliable elements of this solution.

It is precisely at this point that an end-user can intervene. As the system is strongly asynchronous, a human intervention, by means of adding hypotheses (phonetic, prosodic, or any other that is related to a

class of μ-agent) will be automatically considered as a new element to be integrated in a new iteration of reliability measure, selection, decision. The figure 9 gives an overview of the first version of a GUI that allows the user to build graphical realizations of local hypotheses with temporal boundaries. All the relations are automatically computed as requested by the various classes of μ-agents.

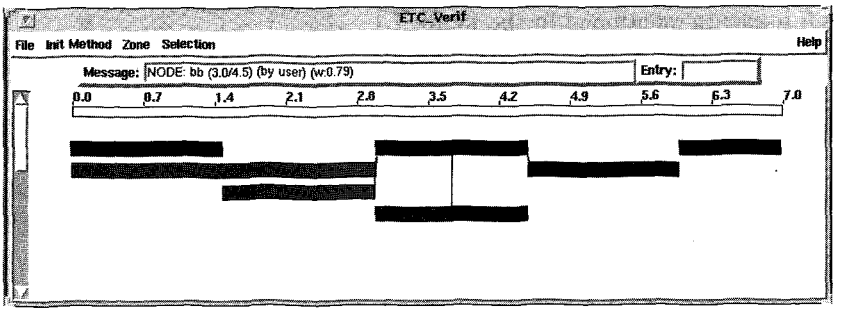


FIGURE 9 First prototype of a GUI that lets the user interact with a process of problem solving based on the multi-agent philosophy implemented in $\mathsf{ETC}_{\mathsf{vérif}}$.

6 Applications

Partially, the speech databases described here were already used to evaluate and enhance speech technology systems of IDIAP. As a first application, parts of the subcorpora of pronounced digits were extracted and used as training and testing material for our HMM based digits recognizer. Though only a fraction of the available utterances was used, the usefulness and necessity of such speech databases become obvious by looking at figure 10, which illustrates the enhancement of performance in relation to the amount of training material.

Since IDIAP is presently involved in several projects concerned with speech recognition at the level of digits (ATTACKS), spelling, keyword spotting (CERS), as well as with speaker verification tasks (CAVE, M2VTS), we will continue to exploit the developed PolyPhone and PolyVar databases in order to improve and test existing speech technology systems and to develop new ones. Moreover, we intended to implement our research results into real world applications, like telebanking and other voice accessed security sensitive domains. It is also foreseen to develop new voice driven servers for services like information kiosk, or personal assistants in collaboration with partners, e.g. Swiss Télécom PTT.

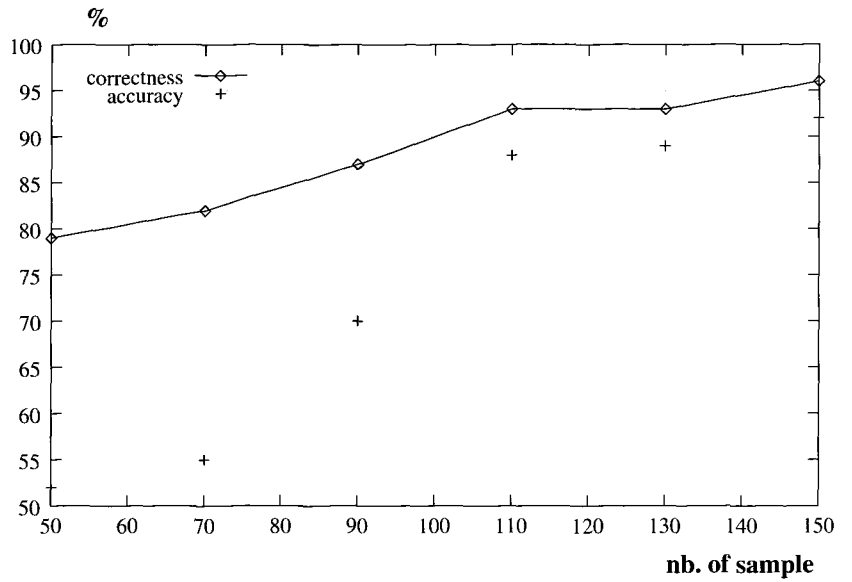


FIGURE 10 Average performance of our digits models compared to the number of learning examples used to build each model.

7 Conclusions

The need of standardized, reusable speech databases is obvious. The benefits they offer are immediate: a large set of data allows to enhance existing speech technology and to develop new products. Thanks to such speech databases, it is easier to assess speaker verification and speech recognition systems, and to evaluate new approaches, as they arise, e.g. flexible vocabulary. The efforts of different database consortia with the goal of collecting these costly reusable resources are worthy to be undertaken. Upon completion, these speech corpora are made available through organisations like LDC or ELRA. The former, for instance, is already distributing several speech and text databases, where the range of items on the ordering form varies from corpora of spoken digits (TIDIGITS), over continuous speech databases (ARPA), to speaker identification data (YOHO).

This working structure of producers, distributors and users, will continue to grow, offering more and more new resources. Resources, which at turn will help enhancing performance of speech technology systems — thus bringing us step by step closer to our goal.

Annexe: hard copy of a prompting sheet

feuille d'appel n° 13946

Votre appel téléphonique au N° gratuit 155 28 14

Tout d'abord, au nom de la recherche scientifique suisse, de l'économie en général et plus particulièrement des Télécom PTT et de l'institut LINK, nous tenons à vous remercier de votre précieuse collaboration.

Directives sur la façon de procéder

Lisez attentivement les quelques directives suivantes :

- **lors de l'appel :**
  - **la voix enregistrée du répondeur annonce les numéros en gras (colonne de gauche) de chaque rubrique**
  - **le répondeur enchaîne avec l'énoncé explicatif de la demande;** à titre purement informatif cet énoncé est mentionné dans la colonne du milieu (et aux rubriques 35-36 un choix de réponses est proposé entre parenthèses)
  - **un "bip sonore" vous indiquera que c'est-à-vous de parler: lisez alors le texte de la rubrique concernée**
- **avant de faire le N° gratuit** 155 28 14, parcourez une première fois les 38 rubriques de ce document
- si vous souhaitez une information complémentaire, vous pouvez téléphoner au N° 031 / 338 64 57 (MM. Mury ou van Kommer pendant les heures de bureau)

Ci-dessous commence votre tâche effective

N°	Explication / texte annoncé par l'automate	Ce que vous devez lire (juste après le "Bip sonore")
1	Lisez le nombre :	*13946*
2	Lisez la phrase :	*La fermière élève des oies.*
3	Lisez le(s) mot(s) :	*oui*
4	Lisez la quantité :	*4 032 mètres*
5	Lisez le numéro de carte de crédit :	*6493 3578 0602 1217*
6	Lisez la phrase :	*Déchausse-toi avant d'entrer.*
7	Lisez le nom puis épelez-le :	*Bexen* *B e x e n*
8	Lisez la somme :	*28 011 Dollars U S*

page suivante s.v.p

9	Lisez la phrase :	*L'épervier a saisi un moineau dans ses serres.*
10	Lisez le(s) mot(s) :	*informations consommateurs*
11	Lisez la phrase 11 :	*Le coupable a fait des aveux spontanés.*
12	Lisez la phrase 12 :	*Il nous a noyés dans des explications invraisemblables.*
13	Lisez l'heure :	*22:23*
14	Lisez le mot :	*Nicolas*
15	Lisez la phrase :	*Ce musicien a du génie.*
16	Lisez le(s) mot(s) :	*réservation*
17	Lisez la somme :	*14 811 Lires*
18	Lisez la date :	*Mercredi 4 octobre 2022*
19	Lisez la phrase :	*Les hameçons, les filets sont des engins de pêche.*
20	Lisez le nombre :	*12 028.711*
21	Lisez le(s) mot(s) :	*Le temps*
22	Lisez le nom puis épelez-le :	*Valaisan* *V a l a i s a n*
23	Lisez la phrase :	*Je sens une douleur à l'épine dorsale.*
24	Lisez le(s) mot(s) :	*l'heure*
25	Lisez un à un les chiffres et signes :	** 0 7 9 5 4*
26	Lisez le nom puis épelez-le :	*Scacco* *S c a c c o*
27	Lisez la phrase 27 :	*Cette maison a été le théâtre d'un crime.*
28	Lisez la phrase 28 :	*Il communie tous les dimanches.*

page suivante s.v.p

29	Lisez le nom de ville :	*schwytz*
30	Veuillez poursuivre en donnant un numéro de téléphone que vous connaissez par cœur :	.....
31	Indiquez votre langue maternelle :	.....
32	Répondez par Oui ou par Non à cette question : êtes-vous de sexe féminin ?	.....
33	Indiquez votre date de naissance :	.....
34	Indiquez la ville dans laquelle vous avez commencé votre formation scolaire, à savoir la 1ère année d'école primaire :	.....
35	Indiquez le niveau final de votre formation scolaire (école primaire, école professionnelle ou école supérieure) :	.....
36	Indiquez le type de téléphone que vous utilisez en ce moment précis (standard Tritel, téléphone sans fil, Natel C, Natel D, un appareil importé, ou une cabine téléphonique publique) :	.....
37	Veuillez maintenant faire comme si vous étiez en ligne avec le 111 ...pour demander le n° de téléphone de la personne imaginaire dont les coordonnées se trouvent en face :	*... PIPOZ-PILET NICOLE BOIS-NOIR 18 CERNIER...*
38	Votre éventuel commentaire sur l'ensemble de cet entretien :	.....

Nous vous remercions de votre collaboration et espérons que la taxcard vous sera utile ... et la chance favorable lors du tirage au sort du voyage !

References

Bernstein, J., K. Taussig, and J. Goldfrey. 1994. Macrophone: An American English Speech Corpus for the POLYPHONE Project. In *ICASSP.*

Choukri, K. 1994. Traitement automatique de la parole, application aux serveurs vocaux interactifs. *Vocatel (Journal du téléphone).*

Cochard, J.-L., and P. Froidevaux. 1995. Environnement multi-agents de reconnaissance automatique de la parole en continu. In *Actes des 3èmes Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-agents*, 101–110. mars.

Lander, S. E. 1994. *Distributed Search and Conflict Management Among Reusable Heterogeneous Agents.* Doctoral dissertation, Dept of Computer Science, University of Massachusetts Amherst, May.

Langlais, P., and J.-L. Cochard. 1995. The use of prosodic agents in a cooperative automatic speech recognition system. In *International Congress of Phonetic Sciences.* Stockholm, Sweden, August 13–19.

MACIF. 1994. MACIF IVR system: One year of exploitation. Internal document, June.

Méloni, H., and P. Gilles. 1991. Décodage acoustico-phonétique ascendant. *Traitement du signal* 8(2):107–114.

Niedermair, G. 1994. Sprachgesteuerte VoiceMail- Anforderungen an die Spracherkennung und Datensammlung. Interne Siemens Studie, München.

Pérennou, G., M. de Calmès, I. Ferrané, and J. M. Pécatte. 1992. Le projet BDLEX de base de données lexicales du français écrit et parlé. In *Actes du Séminaire Lexique*, 41–56. IRIT-UPS Toulouse.

Sorin, C., D. Jouvet, M. Toularhoat, D. Dubois, B. Cherbonnel, D. Bigorne, and C. Cagnoulet. 1992. CNET Speech Recognition and Text-To-Speech in Telecommunications applications. In *IEEE Workshop on Interactive Voice Technology for Telecommunications applications.* Piscataway, NJ, October 19–20. IEEE Wor.

Strawe. 1994. CTI in the European Marketplace. In *Computer Telephony Conference Voice'94.* October.

Torkkola, K. 1994. New ways to use LVQ-codebooks together with hidden Markov models. In *Proceedings of the IEEE 1994 International Conference on Acoustics, Speech, and Signal Processing.* Adelaide, Australia, Apr.

9

Investigating Argument Structure: The Russian Nominalization Database

ANDREW BREDENKAMP, LOUISA SADLER, & ANDREW SPENCER

1 Introduction

The interface between the lexical semantics of a predicate and its (morpho-) syntactic properties continues to be a major focus of linguistic research within a wide range of different frameworks. It is evident that the lexico-semantic properties of predicates largely determine the syntactic environments in which syntactic heads appear, and a variety of specific proposals have been put forward to account for the mapping into syntax (Bresnan and Zaenan 1990, Tenny 1994).

While there is agreement at this very general level, widely disparate proposals have been put forward as to precisely which levels of representation are necessary and what (linguistic or conceptual) notions are relevant to linking. In more surface oriented theories such as LFG and HPSG the "syntactic side" of linking is a level of surface syntax, while in other theories, the "syntactic side" of linking is akin to a level of deep syntax in which e.g. passive subjects and the single arguments of monovalent unaccusative predicates are internal arguments (e.g. the work of Levin and Rappaport-Hovav 1995). In addition, some authors assume a level of lexical conceptual structure (LCS) as the "semantic side" of link-

The work reported on here was carried out with the support of the ESRC under grant R00023 4783, to Andrew Spencer and Louisa Sadler. This support is gratefully acknowledged. The dataset is freely available for academic purposes: contact andrewb@essex.ac.uk for further information.

Linguistic Databases.
John Nerbonne, ed.

ing (Levin and Rappaport-Hovav 1994, Jackendoff 1990, Tenny 1994), others a level of proto-roles (Dowty 1991, Alsina 1992, Zaenen 1993) and still others a level of thematic roles (Bresnan and Kanerva 1989).

An unfortunate ambiguity surrounds the term *argument structure* in this context. Many authors assume a distinct level of *predicate argument structure* which mediates the projection of lexico-semantic structures into the syntax (Levin and Rappaport-Hovav 1995, Grimshaw 1990), giving a three tiered architecture, with scope for operations and processes to be captured in a variety of ways. Three (linearly ordered) levels clearly allows for two interlevel mappings as well as operations at each level, though in most proposals, the mapping between argument structure and syntax is more or less trivial. For proponents of such a view, "argument structure", "predicate argument structure (PAS)" and "a-structure" are terms which refer to a *level of representation.* Other proposals, such as some articulated within LFG and HPSG as well as Jackendoff (1990), avoid such a level, mapping directly from the semantic level (however formalized) into the syntax. In this case, the term "argument structure" is used more generally to refer to the syntactically relevant predicate-argument (style) relations holding between a semantic head and its dependents. We will use the term in this looser sense, except when specific proposals for a level of argument structure are under discussion, precisely to avoid commitment to one or another theoretical model. The intuition is clear — if a predicate has (projects) an argument structure, then it has (obligatory) syntactic dependents corresponding to semantic participants.

One major focus of work on argument structure, then, is that of determining in what ways the lexical semantics of predicates determines the surface realizations of syntactic arguments. An important, and open, question is precisely which types of property the mapping to syntax is sensitive to. While most early work assumed that the properties in question were basically thematic in nature, recent research has pointed to an important aspectual dimension to the semantics-syntax interface. In Grimshaw (1990) a thematic hierarchy derived from LCS representations combines with an aspectual tier in predicting the syntactic realization of arguments. The nature of the aspectual tier is somewhat unclear, but Grimshaw appears to have in mind some indexing of roles according to which sub-event of a complex event they participate in, and according to whether they encode certain salient (aspectually relevant) properties such as causation. Van Valin (1990) derives many of the effects of linking from semantic representations which include reference to aspectual structure together with a hierarchy of primitive thematic roles. In this case, the system is sensitive to a classification (of predicates) as accom-

plishments, achievements, activities or states, and hence to aspectual properties such as telicity. Other work embodies the claim that *only* aspectual properties are relevant in the syntax-lexical semantics interface (Tenny 1987, Tenny 1994, Mulder 1993). Tenny's system, for example, makes reference to aspectual notions such as path, measure and endpoint.

The notion of argument structure and the nature of the lexico-semantic syntax interface are particularly fascinating when the predicate undergoes some morphological alternation, loosing some of its (canonically verbal) predicative properties, as in derivational processes such as nominalization and deverbal adjective formation. In general, the argument structure and interface properties of non-verbal predicatives are far less studied than those of verbs. Until recently the standard position was that nouns only optionally take syntactic dependents (Simpson 1983), but in a detailed study of English nominalizations, Grimshaw (1990) claims that many derived nominals are systematically ambiguous between a reading on which the arguments are syntactically obligatory (and correspond to LCS participants), and a reading on which they do not take arguments at all, although they may sometimes appear with modifiers, adjuncts, argument adjuncts, or (in her terminology) complements.

We see then that two key issues in current work on the interface between lexical semantics and (morpho-)syntax are the argument structure of non-verbal predicates, especially nominals, and the role of aspectual notions in determining when and how dependents are syntactically expressed. Given this, the Slavic languages are of particular interest. First they exhibit highly productive morphological processes of nominalization. Second, aspectual properties are highly prominent in these languages, which grammaticize the distinction between perfective and imperfective aspects. We will discuss this matter at greater length below, but roughly speaking, a verb in the perfective aspect treats an event as *bounded* and often thereby makes implicit reference to the endpoints of an event, implying completion or totality (see Smith 1991 for some illuminating discussion and Durst-Andersen 1992 for an accessible survey of traditional views). A verb in the imperfective aspect generally lacks these components of meaning and so can refer to a continuous state, a habitual activity or an iterated action or activity, or can name an event without commitment to its boundedness. Nominals can be derived both from stems which are formally (or morphologically) perfective and from those which are imperfective. This provides us with a rich area of nominalizations in which to investigate claims about the relevance of aspectual properties such as boundedness and telicity to the syntax-semantics

interface and specifically, to the projection of argument structure. The conventional wisdom in the Slavic tradition (see for example Comrie 1980 and Khokhachova 1969) is that grammatical aspect is *semantically* neutralized under nominalization, but this seemed to us, in advance of carrying out this investigation, to be only partly true. For example, the open class of secondary imperfectives permit productive nominalization (rather like English *-ing*) and appear to preserve the processual meaning associated with imperfective aspect in verbs.

Against the background of the theoretical claims and insights briefly reviewed above, our aim in this project was to investigate systematically the argument structure and syntactic properties of derived nominals in Russian. In doing this we hoped to shed some light on the following:

- What is the relationship between grammatical aspect (the perfective/imperfective distinction of Slavic) and the notion of situation aspect (or lexical Aktionsart), which involves the property of telicity?
- Derived nominals exhibit formal/morphological perfectivity or imperfectivity. To what extent can they be said to preserve the associated semantic properties?
- Which classes of derived nominals, if any, systematically have an argument structure? How is this related to situation aspect and/or grammatical aspect?

Using corpus resources, native speaker intuitions and dictionary/thesaural resources, we collected a significant dateset of Russian verbs, derived verbs and associated deverbal nominals in order to investigate these questions. Some clarifications are perhaps in order at this point concerning our data collection processes. The phenomenon that we have investigated is somewhat restricted in register — the deverbal nominalizations we focused on are found predominantly in scientific texts. Dictionaries generally fail to provide the full array of arguments a nominalization is capable of taking, and for a given nominalization, especially of a low frequency verb, a corpus is likely to underestimate the argument taking potential of a nominal in precisely those aspects which we are investigating. Moreover speaker intuitions are clearly required in very many cases to distinguish specific readings of nominalizations — a notoriously difficult area. For these reasons, the primary data come from dictionary entries supplemented by native speaker intuitions. There is a good deal of variation in acceptability judgments for nominalizations just as there is in English. Recognising that this represents an insoluble problem given current knowledge and resources, we therefore recorded what seemed to us to be the consensus judgment on difficult cases taken

from the intuitions of a native speaker research assistant and from other educated Russians resident in Britain and St. Petersburg. A detailed investigation of speaker variability in this area has yet to be carried out.

To increase systematicity and ease querying, we built a database application for the dataset. This is necessary because, given the morphological complexity of the data, we wanted to be able to pull out different sets of examples according to different settings of the different parameters. With a view to future reuse by other researchers, or within a large-scale computational lexicon, it was also important to us to store our data in a format that would not prejudice future reuse.

The rest of this paper is organized as follows. We first provide a brief introduction to relevant aspects of Russian morphology and nominalization. We then turn to the question of the argument structure of deverbal nominals, providing a short summary of Grimshaw's analysis of English nominalizations. The following sections focus on the Russian data and the database application. A final section presents our findings. The reader who is interested primarily on our linguistic findings may find a preliminary analysis of these in Sadler, Spencer and Zaretskaya (1994).

2 Russian Deverbal Nominalization

This section provides a very brief review of those aspects of Russian morphology that are relevant to the nominalization data we discuss. We begin with a few brief remarks concerning the verbal morphology of Russian and the conjugation system and then discuss the expression of perfective aspect and lexeme formation. The second subsection gives a brief introduction to process of deverbal nominalization in Russian. By the end of this section, then, we have set up all the distinctions of relevance to our data collection, although we will be no means have done justice to the morphological niceties in our crude thumbnail sketch.

2.1 Verbal Morphology

There are two main conjugation classes in Russian (I and II), with several subclasses and a degree of allomorphy connected with morphophonemically conditioned palatalization. A further complication is that in certain morphosyntactic contexts certain stems from certain conjugation classes exhibit the Iotation alternation in the final consonant of the stem.

Russian distinguishes morphologically between the perfective and imperfective aspects.[1] Laying to one side some minor complications, a

[1] We abbreviate perfective and imperfective as PERF and IMPF respectively.

simplex (monomorphemic) verb lexeme will usually be imperfective and will generally form a perfective by prefixation. The perfective prefixes also have lexemic uses in which they derive new lexemes (with perfective aspect), but it is possible to isolate some cases in which a prefix is semantically *empty*, apart from introducing perfectivity (boundedness).[2] This dual function of many Slavic prefixes is extensively discussed (for an accessible discussion, see for example Piñón 1993 and Piñón 1994.)

(1) *On pisal pis'mo.*
he write(IMPF) letter

He was writing a letter. (letter not necessarily finished)

(2) *on na-pisal pis'mo.*
he PERF-write letter

He wrote a letter. (letter necessarily finished)

The derivation of new lexemes from simplex ones by prefixation is a very significant type of word formation in Russian. Many of the prefixes used have a fairly constant meaning, though others (or other uses) can be highly idiosyncratic. Lexeme formation by prefixation (*lexical prefixation*) has a number of important aspectual properties:

–The prefix adds an extra meaning component to create a new lexeme, often adding a notion of Change of State to give an accomplishment from a simplex activity lexeme.
–Lexical prefixation almost always produces a perfective base.
–The imperfective member of the aspectual pair is then formed by secondary imperfectivization (SIMPF)

Since the secondary imperfectives have turned out to be of particular interest in our investigation of nominalizations, we will say a little about how they are formed. Broadly speaking, there are two ways: by suffixation of *-yv-* or *-iv-* and shift to Class I, or by simple shift of conjugation class to Class I (iotated). The *-iv* form is a variant of *-yv-* conditioned by the palatalization of the previous consonant. The 'o' vowel of many roots ablauts to 'a' under secondary imperfectivization (eg. *ugovar'vat'*). Some illustrative examples are provided below, where 'Iath' stands for the 'athematic' subclass of Class I, lacking a theme vowel.

[2]It is not possible to predict the prefix selected by a given lexeme for the "pure" perfective, though the prefix *po-* seems to be a default.

(3)

Perfective	Imperfective	
ukrep'it' (II)	ukrepl'at' (I)	consolidate
sobrat' (Iath)	sobirat' (I)	collect
peredelat' (I)	peredel-yv-at' (I)	re-do
ugovor'it' (II)	ugovar'ivat' (I)	persuade

2.2 Nominalization

Space precludes detailed discussion of the derivational processes of nominalization in Russian. The commonest types of morphological operation deriving deverbal nominals are conversion (sometimes accompanied by palatization of the final consonant of the stem), and suffixation of *-ka* or *-nie* (see Townsend 1975:152ff for further details):

(4)

conversion			
razgromit'	to raid, rout (PERF)	razgrom	rout
obyskat'	to search (PERF)	obysk	search
-ka			
pobelit'	to whitewash (PERF)	pobelka	whitewashing
sxvatit'	to fight (PERF)	sxvatka	skirmish
perepisat'	copy (PERF)	perepiska	copying
-nie			
razrušit'	destroy (IMPF)	razrušenie	destruction
ukrepit'	consolidate (PERF)	ukrepl'enie	consolidation
raspisat'	write out (PERF)	raspisanie	timetable
raspisyvat'	write out (SIMPF)	raspisyvanie	writing out
starat's'a	try (IMPF)	staranie	endeavour
perepisyvat'	copy (SIMPF)	perepisyvanie	copying

From these examples, it is clear that nominalization of perfective and imperfective (including secondary imperfective) stems is possible. Some forms are ambiguous, but in general it is possible to distinguish nominals based on imperfective stems from nominals based on perfective stems. Roughly speaking, most verb lexemes give some sort of nominalization. Secondary imperfective verb forms almost always give a nominal and perfective verb forms very often do so.

Recalling that the focus of our study is on the relationship between the nominals and aspectual structure/properties, two questions present themselves as of primary interest. The first is whether the viewpoint or aspectual contrast marked on verbs is semantically as well as morphologically preserved under nominalization. On the assumption that the relevant notion is boundedness one might ask whether perfective nominals focus on endpoints or bounded events and imperfective nominals on the process denoted by the nominalization.

The question which is at issue here is whether in Russian, viewpoint aspect, that is, the *semantic property* distinguishing imperfective and perfective aspects is restricted to verbs or whether it is also inherited into derived nominals. For English, the data is rather unclear, but it appears that both perfective and imperfective interpretations may be given to nominalizations such as *destruction* and *felling* (they may be vague as to viewpoint, or viewpoint may simply be semantically irrelevant to nominal forms) whereas in Polish, by contrast, perfective and imperfective pairs have been reported to give distinct nominalizations with retention of viewpoint in quite a systematic fashion (see Rozwadowska 1995).

The second question is whether or not the Russian data lends any support to Grimshaw's claim, which we discuss very briefly below, that all and only nominals which have an associated event structure project a syntactic argument structure. This is also a prototypically verbal property; again, what is at issue is precisely which (semantic) verbal properties are preserved under nominalization. In order to address this second question, we need some further background, which we provide in the following section.

In the longer term, we would like to get a clearer understanding of the relationship between grammatical or viewpoint aspect, and lexical Aktionsart (situation aspect) in the Slavic languages, and in Russian in particular. This is necessary to a thorough investigation of nominalization. In our view, the significance of Grimshaw's work will not be in the rather fuzzy claims about the preservation of event structure (see below) under nominalization, but in the articulation of the view that some nominal forms have an argument structure precisely because they maintain some prototypically verbal aspectual properties. If this is so, then those languages in which aspectual distinctions (whether situation aspect or viewpoint aspect) are highly salient can be expected to provide a rich source of data.

3 Argument Structure and Nominalization

In a recent detailed study of English nominalizations, Grimshaw (1990) argues that the systematic ambiguity of many nominal forms hides a basic regularity according to which nominals which have an *event structure* have an argument structure (and therefore obligatory syntactic dependents), while other nominal forms obligatorily lack an argument structure. The former group of nominals are called the Complex Event Nominals, while Result and Simple Event Nominals are those which lack an argument structure (henceforth, CEN, RN and SEN respectively). A cru-

cial point is that a given form, such as *examination*, can be ambiguous between a complex event reading and a result or concrete object reading — according to Grimshaw, it has an argument structure only in the former case.

(5) John's examination of the patients .... (agent) [CEN]

(6) John's examination was long (possessor, author) [RN]

Distinguishing between these different readings is not always straightforward, since there are not necessarily any surface clues (in English, for example, both obligatory arguments and optional modifiers of derived nominals may be marked with the preposition *of* and appear in the same canonical positions with respect to the head. Similar facts are reported for other languages). This means that it is not simply possible to rely solely on corpus data without linguist intuitions.

The following examples further illustrate the contrast between these different types of deverbal nominal:

(7) The constant expression *(of one's feelings) is desirable. [CEN]

(8) John's journey to China broadened his mind. [SEN]

(9) There was a strange expression on her face. [RN]

Grimshaw shows that a number of diagnostics may be used to distinguish CENs from non-argument taking nominals. A number of these diagnostics test straightforwardly for the occurrence of aspectual structure.

(a) Aspectual modifiers, such as *frequent* and *constant*, occur only with eventive, argument bearing nominals. These elements must be licensed by event structure.[3] In the examples below, the obligatoriness of the *of* complement in (10 b) is diagnostic of the presence of an argument structure — in this case, but not in (10 a), which is a concrete result nominal, the adjective *frequent* is permitted:

(10) (a) The (*frequent) expression is desirable.
(b) The frequent expression of one's feelings is desirable.

(b) 'event' control is possible with a complex event nominal, but not with result or simple event nominals. CENs also permit aspectual modifiers such as *in an hour*, *for 6 weeks*.

(11) The examination of the patient in order to determine whether ...
The destruction of the city in two weeks/*for two weeks

[3]Note however that this test must be treated with some delicacy — *constant* in fact has several meanings and some of these combine felicitously with result nominals.

(c) *possessives* with a subject-like interpretation are licensed by argument structure, while other nominals license possessive phrases standing in a looser relation to the head noun. If the possessive has a subject-like interpretation, then the argument which would map to the object of the corresponding verb will also be obligatory.

(12) the teacher's assignment *(of the exercise)
the student's assignment was too long

(d) Like a prenominal possessive phrase, in CENs the *by-phrase* also receives a subject-like interpretation and renders the object obligatory. This contrasts with the modifier by-phrase found with RNs and SENs:

(13) the examination *(of the papers) by the instructor

(14) an examination by a competent instructor will reveal

(e) There are widespread differences in the determiner system, which is highly restricted in the case of complex event nominals. On the other hand, result and simple event nominals permit *a, one, that, the* and also pluralize.

(15) The/*a/*that shooting of rabbits is illegal.
Those trips to the beach were fun.

(f) Complex event nominals do not occur predicatively or with equational *be*, but result nominals do:

(16) *The examination was of the patients.
The exam was much too long.

While the notion of event structure remains irritatingly unclear in Grimshaw's work, there are persuasive indications that the distinction between CEN and other nominals is closely related to aspectual properties, as evidenced by the range of tests Grimshaw proposes. A number of the proposed tests involve aspectual modifiers (*frequent, constant, in a week, for an hour*) which are sanctioned by the internal aspectual nature of the event described by the nominal. The fact that the determiner system is heavily constrained for CENs may also be taken as evidence suggestive of the preservation of verbal properties.

4 The Data

The picture which emerges from this brief review of Russian morphology and Grimshaw's analysis of English deverbal nominals is a rather complex one. We have a large number of different parameters to control for

in the data collection. Our initial starting point was to assume that we should investigate a sample of nominalizations which might permit us to isolate the contribution of a number of factors, including the following:

- the Aktionsart of the verbal base, that is, membership in the Aktionsart classes (situation aspect) of State, Activity, Accomplishment and Achievement[4]
- the derivational element involved in the nominalization
- conjugation class of the verbal base
- the grammatical aspect of the verbal base, that is, the imperfective/perfective distinction, which relates to the parameter of boundedness. Recall however that because prefixation of an imperfective stem (to derive a perfective) may also entail a meaning shift, we must also investigate the behaviour of a further class of derived nominals, those from secondary imperfective bases.

As noted above, although there are some important differences between grammatical aspect (boundedness) and situation aspect (telicity), the two are closely connected. Our data bears out the observation that simplex verbs overwhelmingly refer to Activities or States and are grammatically imperfective. A small number of simplex verbs denote Accomplishments or Achievements, and these are grammatically perfective (and for the most part form an imperfective pair by shift into the default Class 1). Prefixed verbs are overwhelmingly Achievements or Accomplishments. For some discussion, see Brecht (1984) who tries to derive the properties of grammatical aspect from situation aspectual (Aktionsart) properties.

In order to determine, according to Grimshaw's classification, what sorts of nominals are found across all these different types of verbal base, we needed to elaborate a battery of tests to distinguish for Russian, CENs from SENs and RNs. We briefly discuss these tests in the next subsection.

4.1 Tests for CENs

It turns out that Grimshaw's diagnostics for English may be rather straightforwardly adapted to Russian, and serve to pick out the class of CENs (cf. Schoorlemmer 1995:298). Thus, for example, while a nominal may pluralize in a result or simple event reading, this is impossible on a complex event reading:

[4]For Russian, and other Slavic languages, it might be appropriate to distinguish the Semelfactives as an Aktionsart or situation aspect class, as explicitly argued by Smith (1991).

(17) *grubye vyraženija*
rude expressions

rude expressions

(18) *vyraženie naseleniem nedovol'stva bespokoit*
expression population-INSTR dissatisfaction-GEN worries
Partiju
Party

the expression of dissatisfaction by the population worries the Party

(19) * *vyraženija naseleniem nedovol'stva*
expressions population-INSTR dissatisfaction-GEN

Below we illustrate the application of a number of Grimshaw's tests to the Russian data, but we will not provide here extensive discussion of the diagnostics.

(20) Event modification:

postojannoe vyraženie naseleniem nedovol'stva
continual expression population-INSTR discontent-GEN

continual expression of discontent by the population

(21) Subject effect on Object:

"bardak" – eto ego vyraženie, ne mojo
"pig-sty" – that's his expression not mine

*ego (postojannoe) vyraženie *(nedovol'stva) razdražaet*
his (continual) expression *(dissatisfaction-GEN) irritates
menja
me

his (continual) expression *(of dissatisfaction) irritates me

(22) Restrictions on Determiners:

eti (grubye) vyraženija
these (rude) expressions

* *takoe vyraženie naseleniem nedovol'stva*
such expression population-INSTR dissatisfaction-GEN

such expression of dissatisfaction by the population

5 The Database Application

In the previous section we have tried to give an indication of the range of factors which informed our data collection. We have attempted to systematically investigate all these parameters as far as possible in the work carried out so far. This has resulted in a quite systematically described lexical fragment containing around 2,000 verbs and 4,500 associated nominal forms. These lexical descriptions are stored in a database, which we describe in this section. Verbs were chosen to reflect a variety of argument structure and semantic characteristics and in certain cases the morphological properties discussed in Sadler, Spencer and Zaretskaya (1994).

5.1 Choice of Platform

The construction of the database was undertaken in Microsoft Access running under Microsoft Windows. The main justification for this approach was the relative ease with which a graphical interface, with concomitant ease of use, could be constructed. Microsoft Access provides a graphical interface "for free", allowing for prototypes to be rapidly built and developed in cooperation with users, who, in this instance, were linguists not generally familiar with issues in database design.

The use of Microsoft Access was also driven by the desire to make the data available to as wide a set of users as possible. Microsoft Access allows for the development of stand-alone applications, licences for which are the property of the developer(s). Unlike databases constructed in INGRES, ORACLE or even some dialects of PROLOG, this application could be made freely available (e.g. via ftp) to the academic community at little extra cost.

This option was made more attractive given the attempts being made as part of the project to exchange information with the Former Soviet Union, where the IBM PC platform is by far the most common platform for computational linguistics research.[5] A further advantage of this choice is the relative ease with which the database can incorporate Cyrillic. It is a relatively trivial task to redesign the database such that certain sections of the data, or even all data and application components are translated into Russian, the only restriction being the absence of a

[5] The Microsoft Access database itself requires about 8 Mb of disk space (depending of the installation type selected). The database application developed here occupies currently approximately another 2 Mb. The minimal specification machine required to run the application effectively would be a 50 MHz 486 with 8 Mb of RAM (working memory).

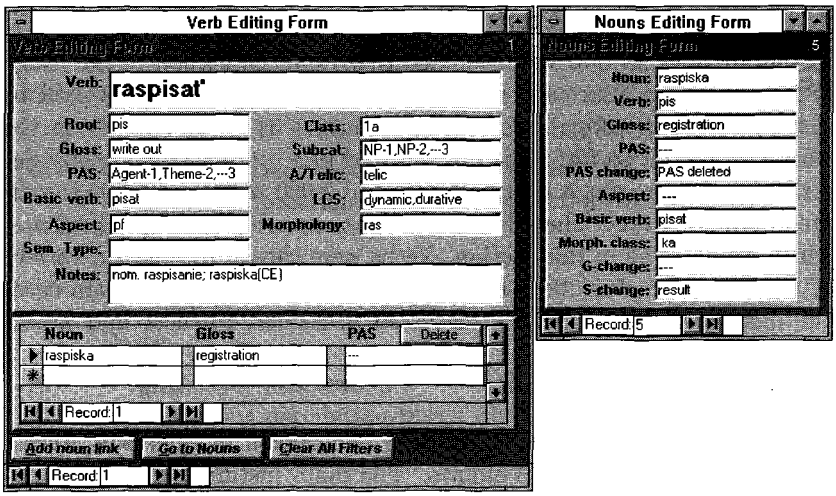


FIGURE 1 The Russian Nominalizations Database main editing forms

standard (and public domain) Russian font for the Microsoft Windows environment.[6]

At the outset the decision was thus made to transliterate the Cyrillic data according to what is more or less a *de facto* academic standard.

5.2 Database construction

The database is relational, consisting of three interlinked tables. The first contains entries for verbs, with, among other things, approximate English glosses, predicate-argument structures, aspectual information, and certain morphological information, for instance regarding the verb root and prefixation. The noun table has a roughly parallel structure, with entries having their own distinct argument structure and aspectual features. Finally a table is used to set up links between verbs and their nominalizations. In form view, an entry for a verb shows those nouns to which the current verb is linked, that is, which nominalizations are derived from this verb. Researchers can therefore easily interrogate the database to investigate, for instance, the way in which a particular kind of affixation affects the argument structure of the resulting nominalization. Microsoft Access offers a sophisticated built-in query engine, which allows for the application of complex search criteria over the data.

[6]This was the situation when we started work, but standards in this area are emerging.

6 Querying the database

6.1 Stating Criteria

The querying system of Microsoft Access allows for the translation of all standard SQL queries. Some level of interactive query building is also supported. For our purposes, standard criteria involve simple expressions which range over strings:

```
Like ''*<string>*''
Like ''*[<string>,<string>]''
Not ''*<string>*''
Is Not Null
```

Furthermore, criteria can refer to the beginning or the end of the string. The following expressions succeed if `<string>` equates with the `<num>` characters at the beginning or end respectively of the value of `<Field>`:

```
Left ([<Field>,num] = ''<string>''
Right ([<Field>,num] = ''<string>''
```

The following are equivalent expressions stating disjunctive criteria over two (or more) strings:

```
In (''<string>'', ''<string>'')
''<string>'' Or ''<string>''
```

Queries can be constructed which prompt the users for values at run time. The criterion:

```
[Enter <Field> value:]
```

will create a dialogue box for entering the value of the field for which the criterion was given.

6.2 Using the expression builder

Microsoft Access also has a graphical tool, called the **Expression Builder**, which allows the user to select the various components of a query, namely datasources (references and expressions) and operators from buttons and lists. The tool will then build the query for the user.

6.3 Translating Natural Language queries into Access

The database was conceived and developed in such a way as to facilitate its use by linguists not necessarily well-versed in the intricacies of database design and use. It was therefore important to show the relative ease with which the natural language queries of the linguist could be "translated" into database queries.

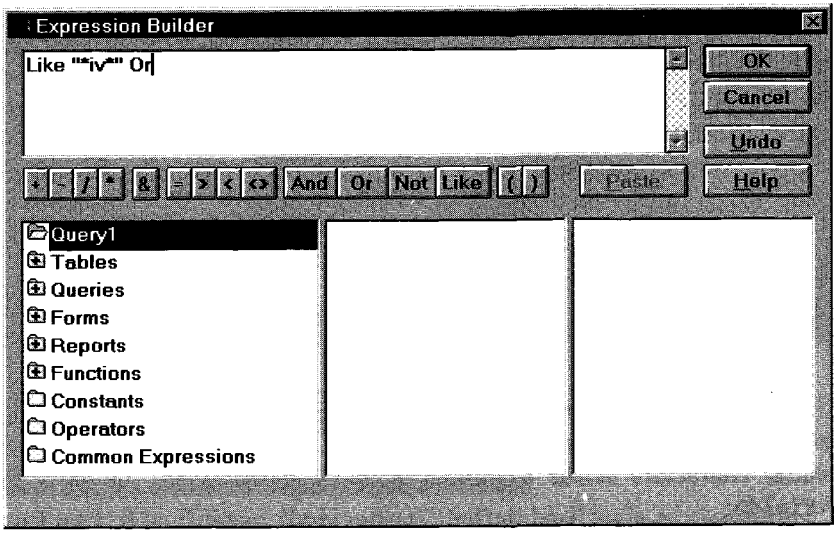


FIGURE 2 The Access Expression Builder

A query of the form:

"List all nominalizations from Class 2 perfective bases"

can be made by setting the field criteria as follows:

field	criterion
class	"*2*"
aspect	"pf"

More specifically, we needed to compile a query which reported on the form of the nominalisation suffix produced by certain types of imperfective verbs which contain the characteristic - *iv* or - *yv* suffix, whose perfective forms fell into a certain conjugation class which we have labeled 2i. In this case the query was constructed as follows.

Two copies of the verbs table were loaded into the query, and a link established between the `root`, `gloss` and `basic verb` fields. Furthermore a link was established between the second verbs table and its nouns by means of the noun-verbs link table. Filters (or "criteria") are then set on the verbs tables such that the `aspect` and `conj class` fields of the first verb table are constrained to match "pf" and "2i" respectively. The aspect value of the second verbs table is constrained to show only those with the values "impf" or "2nd" (the two ways in which the imperfective was encoded). Finally, the nouns table was constrained such that

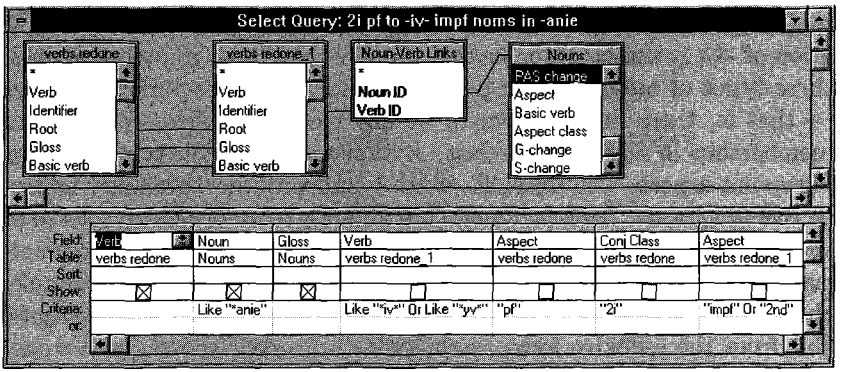


FIGURE 3 A complex query

only those nouns which were suffixed with "-anie" would be shown. The results of this query showed that secondary imperfective verbs suffixed by *-iv/-yv*, which are aspectual pairs to perfectives in class "2i", all have a nominalization in *-nie*. (The importance of this is that it contrasts with secondary imperfectives lacking this suffix.)

6.4 Automatic entry of data

In addition to the standard uses of the database query engine, detailed above, it was possible in some cases to make use of the **update query** facility of Microsoft Access to automate data entry. In the case of the Russian nominalizations database, a new field was required that included information on the conjugation class of the verbs in the database. This requirement was not discovered until after a large part of the data had already been entered into the database. Update queries can be used to set the value of a field, according to some condition on another field. In this case, the update query simply checked the conjugation class of the verbs by looking at their inflectional suffixation.

7 The Results

We are now in a position to consider briefly the findings of our study (for a preliminary analysis, see Sadler, Spencer and Zaretskaya 1994).

Recall that there are three productive ways in which deverbal nominals can be formed, by conversion (zero derivation), and by adding the suffixes *-ka* and *-nie* . There are a number of reasons to consider *-nie* to be the default choice: it has the fewest morphological restrictions; it is the only suffix that regularly gives a "pure" nominalization; it is used with the more recently formed verbalizers and it induces no phonological change on its base.

The behaviour of nominals formed by zero derivation and by suffixation of *-ka* is similar. Our basic finding is that these processes give all three types of nominal from perfective and simplex imperfective bases — that is, there do not seem to be any clear differences traceable to membership in conjugation class, Aktionsart (situation type) class or viewpoint aspect. The *-ka* suffix is not found with any secondary imperfective in *-yv-,-iv-*. However there are about a score of reflexive verbs in *-yv-,-iv-*, based on a small number of verb roots, which have nominalizations in *-ka*, in which the *-yv-,-iv-* suffix is truncated. For example, the secondary imperfective from *perepisat'* 'to copy out' is *perepisyvanie*. But from the reflexive form *perepisyvat's'a* 'to correspond (with someone)' we get the nominal *perepiska*, which has both a RE and a CEN reading. It is not yet clear what, if anything, is responsible for this exceptionality. Accordingly, we have tentatively assumed that *-ka* suffixation is impossible with secondary imperfectives formed with the *-iv/-yv* suffix. Zero nominals are never formed from *-iv/-yv* imperfectives.

(23)

a)	podnosit'	bring (IMPF)	
	podnos	tray	RN
b)	obyskat'	search (PERF)	
	obysk	search	SEN/CEN
c)	xodit'	walk, move (IMPF)	
	xod	move	RN/SEN
d)	čitat'	read (IMPF)	
	čitka	reading	SEN/CEN
e)	pobelit'	whitewash (PERF)	
	pobelka	whitewashing	SEN/CEN
f)	otbelit'	bleach (PERF)	
	otbelka	bleaching	CEN
g)	primanit'	lure (PERF)	
	primanka	bait	RN

The picture with nominalization by *-nie* suffixation is more interesting, however, in that it suggests that a sharp distinction should be drawn between the behaviour under nominalization of perfective and simplex imperfectives on the one hand and secondary imperfectives on the other. Briefly, our data suggests that the following situation holds.

Simplex (imperfective) stems and perfective stems appear to give rise to nominals in *-nie* belonging to all three categories, complex event, result and simple event. The fact that perfective stems *can* give rise to complex event readings, which we take to be inherently processual, would appear to indicate that the the semantic/aspectual notion of boundedness is not retained under nominalization (cf. Schoorlemmer

1995). As with nominalization with *-ka* and ∅, some illustrative data are provided below:

(24)

a)	pisat′	write (IMPF)	
	pisanie	writing	RN/CEN
b)	starat′s′a	try (IMPF)	
	staranie	endeavour	SEN
c)	pet′	sing (IMPF)	
	penie	singing	SEN/CEN
d)	raspisat′	write out (PERF)	
	raspisanie	timetable	RN
e)	spisat′	write off (PERF)	
	spisanie	writing off	CEN
f)	zatverdet′	harden (PERF)	
	zatverdenie	hardening	RN/CEN
g)	izgotovit′	manufacture (PERF)	
	izgotovlenie	manufacture	SEN/CEN

On the other hand, secondary imperfectives behave with very considerable regularity in three ways: firstly, they very nearly always form a nominalization, secondly, this nominalization almost always has a purely CEN interpretation, and thirdly the morphological realization is almost always affixation of *-nie*.

(25)

a)	perepisyvat′	copy (SIMPF)	
	perepisyvanie	copying	CEN
b)	raspisyvat′	write out (SIMPF)	
	raspisyvanie	writing out	CEN
c)	spisyvat′	copy (SIMPF)	
	spisyvanie	copying	CEN
d)	zatverdevat′	harden (SIMPF)	
	zatverdevanie	hardening	CEN
e)	izgotavl′ivat′	manufacture (SIMPF)	
	izgotavl′ivanie	manufacture	CEN
f)	navarivat′	weld on (SIMPF)	
	navarivanie	welding on	CEN
g)	sobirat′	collect (SIMPF)	
	sobiranie	collecting	CEN

These nominalizations are complex event nominalizations in the sense of Grimshaw, in that they support eventive adverbials, agent oriented adverbials, do not appear in the plural, and permit event control (see Schoorlemmer 1995:298 for exemplification). In addition like English

complex event nominalizations, they do not cooccur with demonstratives or other determiners.

There is one further complication which emerges from the data which we have considered. Recall that some Class II verbs form secondary imperfectives not in *-yv, -iv* but by simple shift to the Class I form (with iotation of the root). Some example perfective/secondary imperfective pairs are given below:

(26)

PERF	SIMPF	
provozglašit′	provozglašat′	proclaim
poset′it′	poseščat′	visit
soobsščit′	soobščat′	communicate
postanov′it′	postanovl′at′	RNolve, decree
ukrep′it′	ukrepl′at′	consolidate
razrušit′	razrušat′	destroy
razor′it′	razor′at′	destroy

It seems that these lexemes behave exceptionally in that only the perfective stem may form the base of the nominalization — the resultant nominals very nearly always have a CEN reading, but in addition often have RN and/or SEN readings, seemingly displaying the diversity we associate with nominals in *-ka* and ∅ and nominals in *-nie* from perfectives and simplex imperfectives. While this *is* consistent with one of the observations we made above - that the combination of the secondary imperfective with *-nie* always and only gives rise to complex event readings, it is nonetheless somewhat curious if, for this subclass of verbs, the suffixation of *-nie* to the secondary imperfective stem is somehow blocked. Recall that it was precisely this combination (secondary imperfectivity and nominalization in *-nie*) which gave rise to CEN nominals in a very regular way. The table (27) shows *-nie* nominalization from the (apparently) perfective stem (grammatical) and the ungrammatical form produced by *-nie* suffixation to the secondary imperfective base.

(27)

PERF Nom		Nominal Type	SIMPF Nom
provozglašenie	proclamation	CEN	*provozglašanie
poseščenie	visit(ation)	SEN/CEN	*poseščanie
soobščenie	communication	RN/SEN/CEN	*soobščanie
postanovl′enie	resolution	RN	*postanovl′anie
ukrepl′enie	consolidation	RN/CEN	*ukrepl′anie
razrušenie	destruction	SEN/CEN	*razrušanie
razor′enie	destruction	CEN	*razor′anie

There is no purely phonological restriction which will account for the lack of **razrušanie*, **ukrepl′anie* and such like. Similar nominals are regularly

formed from Class Ia, Iaj simplex verbs (e.g. *pisanie*, *napisanie*, *brosanie* (Class Iaj *brosat'* 'throw'), *staranie*), and of course, there is a potentially unlimited number of nominals in *-nie* from Class Iaj verbs with the *-yv-*, *-iv-* secondary imperfective suffix.

Although we cannot elaborate here, we do not think that there is in fact a gap in the paradigm here. We argue elsewhere that the *-enie* forms in (27) above are in fact ambiguous between perfective and imperfective. Rather we believe that there are good morphological reasons for assuming a rule of referral under which, say, *ukrepl'enie* is simultaneously the nominalization of *ukrepit'* (PERF) and *ukrepl'at'* (IMPF). (see Sadler, Spencer and Zaretskaya 1994 for further discussion).

8 Conclusion

In our investigation of the argument structure of Russian deverbal nominalization, we have built a database application which contains a significant number of verbs and their associated nominalizations. We have attempted to provide a dataset large enough to systematically explore all the relevant parameters, as outlined briefly in this paper.[7] For the reasons that we have explained above, the data in the resource comes from a number of different sources, including native speaker intuitions. The dataset that we have relatively large and systematic. Because it has been developed as a database application, the data is systematic and queryable. It is our view that using a database for linguistic research has a number of advantages when the data, and particular its analysis, is as complex as it is in this case.

Some preliminary results have emerged from our work so far. A systematic investigation of a quite substantial set of data has shown that that there is one island of really systematic behaviour — nominalizations in *-nie* from secondary imperfectives always provide CEN readings. We think that this is consistent with the findings from other languages that a prototypically verbal property (having an event structure) may be maintained under nominalization. In our view, the data suggests an analysis in which the aspectual property of unboundedness is inherited into nominal forms under nominalization in *-nie*.[8]

References

Alsina, A. 1992. On the Argument Structure of Causatives. *Linguistic Inquiry* 23(4):517–555.

[7] Currently the database comprises some 7,000 verbal and nominal entries.

[8] This view requires that the real grammatical aspectual opposition in Russian is between the perfective and the secondary imperfective. We believe that this is certainly a defensible position, but cannot discuss it here.

Brecht, R. 1984. The Form and Function of Aspect in Russian. In *Issues in Russian Morphosyntax*, ed. Michael Flier and Richard Brecht. Columbus: Slavica Publishers.

Bresnan, J., and J. M. Kanerva. 1989. Locative Inversion in Chichewa: A Case Study of Factorization in Grammar. *Linguistic Inquiry* 20:1–50.

Bresnan, J., and A. Zaenen. 1990. Deep Unacccusativity in LFG. In *Grammatical Relations: A Cross-Theoretical Perspective*, ed. Patrick Farrell Katarzyna Dziwirek and Errapel Mejías-Bikandi. 45–58. Stanford, CA: CSLI Publications.

Comrie, B. 1980. Nominalizations in Russian: Lexical noun phrases or transformed sentences. In *Morphosyntax in Slavic*, ed. C. Chvany and R. Brecht. Columbus, Ohio: Slavica Publishers.

Dowty, D. 1991. Thematic Proto-Roles and Argument Selection. *Language* 67:547–619.

Durst-Andersen, P. 1992. *Mental Grammar: Russian Aspect and Related Issues.* Columbus, Ohio: Slavica Publishers.

Grimshaw, J. 1990. *Argument Structure.* Cambridge, MA: MIT Press.

Jackendoff, R. 1990. *Semantic Structures.* Cambridge, MA: MIT Press.

Khokhachova, V. N. 1969. *K istorii otglagol'nogo slovoobrazovanija sushchestvitel'nyx v russkom literaturnom jazyke novogo vremeni.* Moskow: Nauka.

Levin, B., and M. Rappaport-Hovav. 1994. A Preliminary Analysis of Causative Verbs in English. *Lingua* 92:35–77.

Levin, B., and M. Rappaport-Hovav. 1995. *Unaccusativity.* Cambridge, MA: MIT Press.

Mulder, R. 1993. *The Aspectual Nature of Particles.* Amsterdam: HIL.

Piñón, C. 1993. Nominal Reference and the imperfective in Polish and English. In *Proceedings of NELS 23*, 383–97. Ottawa.

Piñón, C. 1994. Accumulation and Aspect in Polish. In *NELS 24*, 491–506.

Rozwadowska, B. 1995. The duality of Polish *-nie/-cie* nominals. In *Licensing in Syntax and Phonology*, ed. E. Gussmann. Lublin: Wydawnictwo Folium.

Sadler, L., A. Spencer, and M. Zaretskaya. 1994. The Third Stem in Russian. Working Paper 44. University of Essex: Department of Language and Linguistics.

Schoorlemmer, M. 1995. Participial Passive and Aspect in Russian. Onderzoeksinstituut voor Taal en Spraak, University of Utrecht.

Simpson, J. 1983. Resultatives. In *Papers in LFG*, ed. Annie Zaenen Lori Levin, and Malka Rappaport. 143–57. IULC.

Smith, C. 1991. *The Parameter of Aspect.* Dordrecht: Kluwer Academic Publishers.

Tenny, C. 1987. *Grammaticalizing Aspect and Affectedness.* Doctoral dissertation, MIT, Boston, MA.

Tenny, C. 1994. *Aspectual Roles and the Syntax-Semantics Interface.* Dordrecht: Kluwer Academic Publishers.

Townsend, C. 1975. *Russian Word Formation.* Columbus, Ohio: Slavica Publishers.

Van Valin, R. 1990. Semantic Parameters of Split Intransitivity. *Language* 66.

Zaenen, A. 1993. Unaccusativity in Dutch: integrating syntax and lexical semantics. In *Semantics and the Lexicon*, ed. James Pustejovsky. 129–63. Dordrect: Kluwer Academic Publishers.

10

The Use of a Psycholinguistic Database in the Simplification of Text for Aphasic Readers

SIOBHAN DEVLIN & JOHN TAIT

1 Introduction

Aphasia is the term given to a broad range of disorders characterised by the total or partial loss of language faculties following brain damage resulting from a stroke or head injury. Recently, there has been much interest in the social rather than medical implications of aphasia (see Parr 1992, 1993). This trend has influenced current work at the University of Sunderland concerning the development of a computerised tool that simplifies text, specifically newspaper reports, by lexical and syntactical substitution according to empirically proven criteria. A major part of this system is a commercially available tool: the Oxford Psycholinguistic Database which contains information on the semantics, syntax, phonology and orthography of almost 99,000 words. Of particular importance to this research are the psycholinguistic properties held in the database that influence word level comprehension i.e. word frequency, familiarity, concreteness, imageability, meaningfulness and age of acquisition.[1] Together with WordNet, an on-line lexical reference system holding English nouns, verbs, adjectives and adverbs in synonym sets, the psycholinguistic database provides the basis for word level simplification.

Sincere thanks are expressed to Professor Ruth Lesser of the University of Newcastle upon Tyne, to Carrie Jarvis and her staff at Sunderland City Hospitals' Speech and Language Therapy Service and to Prue Hayes and her staff at The Sanderson Centre, Gosforth, Newcastle upon Tyne.

[1] Within the time constraints of this research it is not possible to address all of these factors. Only word frequency is addressed at this stage.

Linguistic Databases.
John Nerbonne, ed.

The effects of aphasia can have a huge impact on the lives of sufferers in that people who, prior to the onset of the condition, were perfectly literate can suddenly find everyday use of language almost impossible. Commonly,[2] they will have difficulty evaluating sentences that deviate from the canonical word order (in English: subject-verb-object) and necessitate the mapping of grammatical roles onto thematic roles. In this way passive sentences that are semantically reversible such as

1. "the boy was kicked by the girl"

may be interpreted with the boy as the subject, despite the presence of the grammatical morphemes "was", "-ed" and "by". On the lexical level, a common problem is that words with a low frequency count are less easily identified than those with high counts. Lesser (1978) has said that there is a close link between word frequency and age of acquisition. Thus a word such as "corporation" is learned at a much later stage in a person's life than a word like "group". Although these words cannot be said to be exactly synonymous there are instances when one might acceptably be substituted for the other. "Group" has a much higher frequency and in a word substitution exercise may greatly aid an aphasic person's understanding of a particular situation, whether read or heard.

2 The Nature of Aphasia

Aphasia can affect both the expressive and receptive modes of language so that speech and writing may be altered as well as auditory comprehension and reading: this includes repetition and writing to dictation. While the condition may be restricted to only one of these channels, equally it can affect several or all of them. In this way then, it is difficult to conceive of building a natural language simplifying system that could cater for the needs of every individual. Rather, the intention of our work is to build a general tool that is likely to help most users to some degree. Primarily, the aim is to provide simplification to the most basic level possible. Eventually, the tool may be adaptable to suit each user's level of ability.

3 The Use of Microcomputers in Aphasiology

The use of computer technology for helping aphasics is not a new idea. In 1992 the journal *Aphasiology* devoted a whole issue to the subject and there are many other useful papers on the subject including Guyard et al. (1990), and Petheram (1991). Generally, work in this field concerns therapeutic exercises and speech aids. Examples of such work may be

[2]The examples given in the paper pertain to the English language. The effects of aphasia differ according to the language spoken by the patient.

found in numerous applications developed by the Aphasia Computer Team at The Frenchay Hospital in Bristol, England (Enderby 1993) and in the TalksBack project at The University of Dundee, Scotland (Waller & Dennis 1993).

The research described in this paper focuses on the reading process and in particular on the problems encountered by aphasics with everyday reading material with the aim of making such material more accessible.

4 Increasing Readability

In order to determine how to facilitate reading by aphasics, it is first of all necessary to discuss what makes any given text readable. The concept of readability may differ from person to person. It has been viewed variously as legibility, interest and ease of reading. According to Bently (1985):

2. "Readability is an attempt to match the reading level of written material to the 'reading with understanding' level of the reader."

It is possible to measure the readability of a piece of writing by applying one of the many available formulae to it e.g., Flesch (1948) and Lorge (1959). These formulae vary in their criteria but most agree that high readability scores are effected by the use of short, familiar content words and short sentences and some, such as Lorge (1959) and Elley (1975), also consider that the more words used that are absent from lists of so-called "easy" words, the lower the readability score will be. One factor that they cannot measure, though it is pertinent to this research, is the effect of type size and illustrations. However, any such discussion is beyond the scope of this paper.

So far, evidence of the sort of simplification necessary to aid aphasic reading has been taken from the literature on aphasia (e.g. Goodglass 1993, Lesser 1978, Naeser et al. 1987). However, as a good deal of the literature is concerned with auditory rather than written comprehension we are currently gathering empirical evidence that will hopefully support our proposals, a discussion of which will follow shortly. In the meantime it is possible to speak with a certain amount of confidence about the simplification procedures chosen.

Naeser et al. (1987) have observed that auditory sentence comprehension failure in aphasics is attributable to impairments in both syntactic decoding and lexical comprehension. Therefore, any simplification that is to take place must be on both the word level and the sentence level. Although we are not concerned here with simplification at the sentence level, it is perhaps wise to mention briefly the criteria we are adopting.

4.1 Syntactic Simplification

Since aphasic readers are more likely to comprehend sentences that are simple than those that are complex, sentences should

1. follow the simplest logical form i.e., subject-verb-object (SVO),
2. be active in voice,
3. be kept as short as possible, containing only one clause and one base proposition so as not to overload the aphasic's processing capacity,
4. contain only one adjective per noun, although for emphasis the same adjective should perhaps be repeated to avoid the introduction of extra modifiers e.g., "the red, red rose" not "the bright red rose",
5. be chronological—i.e., the order of propositions in the text should be in the same order as the events they describe in time,
6. be semantically non-reversible—because in a phrase such as "husband's sister" the likely perception will be of a man since he appears first in the phrase. Likewise, the instruction "point to the ruler with the pen" may well cause the patient to point with the ruler. This phenomenon is what Luria (1970) called the impairment of logico-grammatical relationships.

Having touched on the methods employed to achieve syntactical simplification of each sentence we can now turn our attention to the main thrust of the paper: lexical simplification by synonym substitution.

4.2 Lexical Simplification

Discourse comprehension demands processing on various levels: micro, macro and super (Van Dijk & Kintsch 1983). The focus of this paper is the micro level which deals with lexical semantics. Several factors may influence word processing in aphasia. Lesser (1978:111) reports that people retain words they have learned at an early age (generally high frequency words) far more readily than those they have learned in later life (generally lower frequency words). Other psycholinguistic factors are familiarity, concreteness, imageability and meaningfulness. It is not the intention of this paper to discuss these factors at any great length since they are not (currently, at any rate) considered in the research: further information can be found in Paivio et al. (1968), Gilhooly & Logie (1980), and Toglia & Battig (1978). For the moment, we address only word frequency.

4.3 Empirical Evidence

As stated previously, we plan to substantiate the simplification criteria chosen (i.e. lexical and syntactic simplification) with evidence from tests that we are currently conducting within the Speech and Language Therapy departments of the local hospitals. The format of the test being administered to diagnosed aphasic patients is as follows: each patient, during the course of their normal therapy is asked to read six, totally unconnected, short passages. The passages have been taken from the local daily Newspaper, *The Sunderland Echo.* Half of the texts have been manually simplified according to the simplification criteria; half taken directly from the newspaper. The order in which the patients receive the passages varies so that some receive three unchanged texts followed by three simplified texts, while some receive the texts in an alternate fashion beginning with an unchanged text and ending with a simplified text. After the patient has read each passage, either silently or aloud, they are given four questions on the text. The questions are of three types: two multiple choice, one yes/no answer and one requiring the answer to be pulled directly from the text, and the answers given are either correct or incorrect (although the exact responses are also recorded). Referral back to the text is permitted in the course of answering each question.

As yet, the administering of the test is not complete. However, some preliminary results are available and are reported below.

4.4 Preliminary Results

The findings of our two pilot tests provided some indications of the problems likely to be encountered. One instance of this was the very figurative nature of newspaper text: an actual example being

3. "A major travel company has collapsed, leaving thousands of Wearsiders with a holiday *headache*" (our emphasis).

Another, unexpected, problem was the presence of verb particles in instances such as

4. "Union chiefs look set to call *off* a one-day strike at Newcastle Airport."

In the pilot test, the patient actually registered the verb as being "call" and not "call off"; consequently, when we simplified the text we substituted this for "cancel".

To date we have tested nine patients. Because of the nature of the disorder, it has been extremely difficult to find many patients who have retained a sufficiently high reading ability to be able to even attempt the test. Consequently, we are now looking at testing discharged patients

whose language is significantly above the level experienced immediately post-onset.

Our preliminary results, though, are encouraging. Of the nine subjects we tested, one performed equally well on the simplified and unchanged texts, one performed better on the unchanged texts, [3] and seven performed better on the simplified texts. We can therefore reject, at the 5% level, the null hypothesis that the subjects will perform equally well on the two types of text (McNemar's test, Campbell & Machin 1990: 139–41).

5 Newspaper Text

At this point, it is probably worthwhile taking a little time to discuss the nature of newspaper text with the aim of demonstrating why it is beyond the comprehension of aphasic people. The language of newspaper reports is subject to a number of constraints and these must be taken into consideration in the building of the system. The labels "journalese" (for newspapers generally) and "tabloidese" (restricted to the tabloids) have been coined to classify this type of language. In actual fact, however, there seem to be several "journaleses" depending on the readership that the newspaper is aimed at. All newspapers need to compress a great deal of information into a limited space and need to be clear and unambiguous. They must be comprehensible to people from very diverse educational and social backgrounds. As a result they are, in theory, designed to be simple. Nevertheless they are still, like all written texts, beyond the full comprehension of most aphasic readers.

5.1 Observations

What, then, are the characteristics of newspaper text that are likely to be a cause for concern for the aphasic reader?

1. Long sentences—As mentioned in section 5, aphasics can have difficulty processing long sentences. Yet in the broadsheets, sentences tend to be around 32–35 words long and even in the tabloids and local papers the average is 16–20 words (Keeble 1994). Shortening sentences and clauses would undoubtedly aid comprehension.
2. Compounds and adjectives—Tabloidese is full of compounds, such as
    5. "Twenty-five-year-old blonde-haired mother-of-two Jane Smith..."

[3] It was noted by this patient's therapist that his severe short term memory problems contributed to this result.

which is not the type of language that an aphasic can easily understand. Adjectives should be restricted to one per noun.

3. Position of adverbials—Adverbials commonly appear at the start of a sentence in a strongly emphatic role rather than in their more normal post-verbal position. While this repositioning does not affect the order of the subject, verb and object, it is possible that long adverbial phrases placed at the beginning of a sentence could contribute to an overload in the aphasic reader's processing capacity before the action of the sentence is even encountered. Note the following example from Crystal & Davy (1969:176):

   6. "For the first time in routine procedure an electronic computer contributed to the forecast chart published today on this page."

4. Opening paragraphs—The pith of a story is meant to be conveyed in the opening sentence or paragraph and then gradually expanded. This often means long, convoluted sentences containing multiple predicates and clauses. For an aphasic reader, these must be simplified into simple sentences containing only one predicate each.
5. Opening words—the first words of any introduction are vital in attracting the reader's attention and leading him into the story (Storey 1991). This often means that the passive is best employed. Thus instead of

   7. "The council today accepted a bid to build an incinerator on local wasteland"

   a more sensational

   8. "A bid to build an incinerator on local wasteland was today accepted by the council"

   would be employed. Although the example given is non-reversible (so the passive itself should not be problematic) the large number of function words and the low imageability and frequency of the content words would negatively influence an aphasic's comprehension of the sentence.
6. Punctuation—most style books advise against the overuse of punctuation that would hinder the reader's progress through the text. Thus we commonly see commas omitted after initially placed adverbials, between coordinations and between sequences of adjectives. While a normal reader can determine the break between structures from the grammar of the piece, it is possible that an aphasic would have difficulty in doing the same.

5.2 The Style Book

Most newspapers have a style book that dictates the way copy is to be written and presented. In this paper, we refer specifically to that of Portsmouth and Sunderland Newspapers PLC (Storey 1991). The style book urges a "precise use of the English Language" and advocates that "simple English is best". Staff are directed to George Orwell's "Politics and the English Language" in which he advises:

9. - Never use a metaphor, simile, or other figure of speech which you are used to seeing in print.
   - Never use a long word where a short word will do.
   - If it is possible to cut out a word, always cut it out.
   - Never use the passive where you can use the active.
   - Never use a foreign phrase, a scientific word, or a jargon word, if you can think of an everyday English equivalent.
   - Break any of these rules sooner than say anything outright barbarous.

These rules do in fact provide an excellent basis for writing for aphasics. Unfortunately, it is often the case that in the writing of news reports, they are not strictly adhered to.

5.3 Headlines

A final word on newspapers is necessary since headlines seem to operate under a law of their own. Crystal & Davy (1969) wrote:

10. "Headlines have to contain a clear, succinct and if possible intriguing message, to kindle a spark of interest in the potential reader".

What is interesting to this research is that headlines omit the very words that aphasics have difficulty understanding i.e., function or closed-class words. We alluded to this in section 1 where it was suggested that an aphasic would fail to recognise the reversal of the roles of subject and object by the presence of the morphemes "was", "-ed" and "by" in (1).

It remains to be seen from the ongoing tests whether or not the headlines were easier to understand than the main body of the texts.

6 Practical Implementation of the Simplification Tools

In this section we will describe the tools in current use and explain how we have so far integrated them. The tools being used for word level simplification are the Oxford Psycholinguistic Database and WordNet. Investigations are underway in order to determine those Natural Language Processing Tools that are most suited to the task of sentence level simplification, though this won't be discussed here.

6.1 The Oxford Psycholinguistic Database

We have already mentioned, in section 4.2, the factors affecting comprehension at word level (frequency, familiarity, concreteness, imageability, meaningfulness and age of acquisition). All of these factors can be found in the Oxford Psycholinguistic Database. Formerly the MRC Psycholinguistic database, conceived by Max Coltheart in 1981, the database was revamped in 1987 by Philip Quinlan of the Department of Psychology, University of York, and currently runs on an Apple MacIntosh. It comprises 98,538 English words and information on the spelling, syntactic category and number of letters for each of these as well as information on the phonetics, syllabic count, stress patterns and aforementioned criteria affecting comprehension.

There are various uses to which the database might be put. The two suggested by its author are:

a. the selection of sets of words for use in psycholinguistic experimentation and
b. the direct analysis of the linguistic information combined in the database.

The database was combined using many sources: the Associative Thesaurus (Kiss et al. 1973), Jones' Pronouncing Dictionary of the English Language (Jones 1963), Paivio's ratings of the concreteness, imagery and meaningfulness of words (Paivio et al. 1968), Gilhooly and Logie's ratings based on age of acquisition, imagery, concreteness, familiarity and ambiguity measures (Gilhooly & Logie 1980), the Colorado norms which deal with word meaningfulness (Toglia & Battig 1978), the word frequency counts of Kucera and Francis (1967) and Thorndike and Lorge (1944) and the Shorter Oxford English Dictionary database (Dolby et al. 1963).

For the work here at Sunderland, it has been necessary to change the platform of the database for a number of reasons. First of all, for our purposes, wherein we require speedy access to the data, the database was found to be painfully slow. Secondly, the other tools with which we hope to link the database run on a Unix platform. The methods by which we transferred the database will be detailed shortly, but first we shall discuss the other database employed in the research.

6.2 WordNet

The WordNet database (as described in Miller et al. 1993) is a database of synonymous relations. The word categories it contains are noun, verb, adjective and adverb. It was developed at Princeton University under the direction of Professor George Miller and follows current psycholin-

guistic theories of human lexical memory. For this research, WordNet acts as an on-line thesaurus, providing us with a choice of synonyms for the input words. These are sent to the psycholinguistic database for frequency measurements and the synonym with the highest rating is selected as the replacement word.

7 Linking the Tools

What follows is an overview of the work we have achieved to date in integrating the two databases.

First of all, the Oxford Psycholinguistic Database was transferred from Apple MacIntosh to PC format and from there to a Sun Sparc Station. We then removed the phonetic information from the database as it was causing the records to be of varying lengths which significantly increased both the size of the database and the problems of interrogating it. As the phonetic information was felt to be superfluous to this project (in its current state) it seemed reasonable to discard it in an effort to make database access more expedient.

A program was written, in C, to interrogate the database. Although much faster than on its MacIntosh platform, it was still relatively slow at retrieving information. Consequently, we adopted a simple indexing scheme. This meant that instead of performing a sequential search whereby the program looked at every word in the database in alphabetical order until it arrived at the correct one, it looked only at those words with the appropriate initial letter. This measure drastically reduced access time.

There follows an example of how effective the transfer has been. The first word in the database is "aardvark"; the last is "zymurgy". Before the database was transferred from the MacIntosh, the time taken to retrieve information on one field only e.g., word frequency, for aardvark was four and a half minutes and for zymurgy was more than six minutes. If all possible fields were requested, then the response time was simply unworkable. Once transferred to Unix, the time taken to access the information was less than one second and less than ten seconds respectively. After indexing, the time taken was less than one second for any word in the database, for any number of requested fields.

The C program was then expanded to incorporate the necessary functionality to interrogate the WordNet database. This was achieved by using the command line version of WordNet. As the C program was developed on a Sun Sparc Station we were able to exploit the environment's ability to multi-task by creating a child process from the original program (i.e., the parent process). The child process interrogates WordNet

and directs its output (i.e., a selection of synonyms from the database) to a specified text file. On completion of this task, the child process dies, thus awakening the parent process which then interrogates the child's output file. After obtaining the synonyms from the file, the original procedures to interrogate the psycholinguistic database are executed for each of them as well as for the original word. In this way, synonyms are proffered for each input word, and the highest frequency option is selected.

When the tool is complete, the psycholinguistic database will be consulted at the end of the parsing process, once the simplest logical sentence structure has been reached (the user will be able to choose the level of simplification required). However, we would also contend that the tool as it stands i.e., purely as a lexical simplification device, may serve to increase comprehension in its own right.

8 Conclusion

The focus of this paper has been a discussion concerning the integration of various tools, chiefly the Oxford Psycholinguistic Database and the synonym database WordNet, in an attempt to produce a computerised text simplifying device to help aphasic people to read. Maximum use has been made of existing linguistic resources in an effort to ensure the greatest possible coverage of English text.

The paper has also demonstrated that the contributory factors in a person's reduced ability to read are impairments in both syntactic decoding and in lexical comprehension. Thus a device that would compensate for these two factors by simplifying at both sentence and word level, would be of undoubted value. Tests on aphasic volunteers are ongoing with the aim of substantiating the evidence for using syntactic and lexical substitution. Preliminary results have been referred to in section 4.4. We await further results and hope to publish them fully in the near future. In the meantime our work continues, based on indications from the available data and on our interpretation of the aphasia literature.

Ultimately, it is hoped that the simplification tool will be further developed beyond the remit of the current research to embrace not only aphasics but also deaf people and those with language disorders displaying similar linguistic symptoms.

References

Bently, D. 1985. *How and Why of Readability.* University of Reading, UK.

Campbell, M. J., and D. Machin. 1990. *Medical Statistics: a commonsense approach.* England: Wiley & Sons.

Coltheart, M. 1981. The MRC Psycholinguistic Database. *Quarterly Journal of Experimental Psychology*, 33A: 497–505

Coltheart, M. 1981. *MRC Psycholinguistic Database: User Manual*, Ver. 1.

Crystal, D., and D. Davy. 1969. *Investigating English Style.* Longman.

Dean, E. C. 1992. Microcomputers and aphasia. *Aphasiology* 6, 2: 151–3.

Dolby, J. L., H. L. Resnikoff, and F. L. MacMurray. 1963. A tape dictionary for linguistic experiments. In *Proceedings of the American Federation of Information Processing Societies: Fall Joint Conference* 24. Baltimore, MD: Spartan Books.

Elley, W. 1975. *Assessing the difficulty of Reading Materials. The Noun Frequency Method.* New Zealand Centre for Educational Research.

Enderby, P. 1993. *Speech and Language Therapy Research Unit: Annual Report*, April 1992–March 1993. Bristol, UK: Frenchay Hospital.

Flesch, R. 1951. *The Art of Plain Talk.* Collier MacMillan Publishers.

Gilhooly, K. J., and R. H. Logie. 1980. Age of Acquisition, imagery, concreteness, familiarity and ambiguity measures for 1944 words. *Behavioural Research Methods and Instrumentation* 12: 395–427.

Goodglass, H. 1993. *Understanding Aphasia.* Academic Press.

Guyard, H., V. Masson, and R. Quiniou. 1990. Computer-based aphasia treatment meets artificial intelligence. *Aphasiology* 4, 6: 599–613.

Jones, D. 1963. *Everyman's English Pronouncing Dictionary* (12th ed.). Dent, London.

Keeble, R. 1994. *The Newspapers Handbook.* Routledge.

Kiss, G. R., C. Armstrong, R. Milroy, and J. Piper. 1973. An associative thesaurus of English and its computer analysis. In A. J. Aitken, R. Bailey, and N. Hamilton-Smith, eds., *The Computer and Literary Studies.* Edinburgh: Edinburgh University Press.

Kucera, H., and W. N. Francis. 1967. *Computational Analysis of Present-day American English.* Providence, RI: Brown University Press.

Lesser, R. 1978. *Linguistic Investigations of Aphasia.* Edward Arnold Ltd.

Lorge, I. 1959. *The Lorge Formula for Estimating the Difficulty of Reading Materials.* New York: Teachers College Press.

Luria, A. R. 1970. *Traumatic Aphasia.* The Hague: Mouton.

Miller, G., R. Beckwith, C. Fellbaum, D. Gross, K. Miller, and R. Tengi. 1993. *Five Papers on WordNet.* Princeton, N.J.: Princeton University Press.

Naeser, M. A., P. Mazurski, H. Goodglass, M. Peraino, S. Laughlin, and W. C. Leaper. 1987. Auditory syntactic comprehension in nine aphasic groups (with CT scans) and children: Differences in degree, but not order of difficulty observed. *Cortex*, 23: 359–80.

Orwell, G. 1946. Politics and the English Language. *Collected Essays.* London: Secker & Warburg. 1961. 2nd.

Paivio, A., J. C. Yuille, and S. A. Madigan. 1968. Concreteness, imagery and

meaningfulness values for 925 words. *Journal of Experimental Psychology Monograph Supplement* 76, 3, Part 2.

Parr, S. 1992. Everyday reading and writing practices of normal adults: implications for aphasia assessment. *Aphasiology*, 6, 3, 273–83.

Parr, S. 1993. Coping with aphasia: conversations with 20 aphasic people. *Aphasiology*, 8, 5: 457–66.

Petheram, B. 1991. Microcomputers as a supplement to Aphasia Therapy. *Journal of Neurolinguistics*, 6, 2: 177–95.

Quinlan, P. T. 1992. *The Oxford Psycholinguistic Database.* Oxford University Press.

Storey, R. 1991. *Style Book: a guide for the staff of Portsmouth and Sunderland Newspapers PLC.*

Thorndike, E. L., and I. Lorge. 1944. *The Teacher's Word Book of 30,000 Words.* New York: Teachers College Press.

Toglia, M. P., and W. R. Battig. 1978. *Handbook of Semantic Word Norms.* Lawrence Erlbaum Assoc., New York.

Van Dijk, T. A., and W. Kintsch. 1983. *Strategies of Discourse Comprehension.* New York: Academic Press.

Waller, A., and F. Dennis. 1993. Using TalksBack with Dysphasic Adults. *Augmentative Communication in Practice.* Scotland: Collected Papers.

11

The Computer Learner Corpus: A Testbed for Electronic EFL Tools

SYLVIANE GRANGER

1 Current EFL Tools

Most current EFL (English as a Foreign Language) tools, be they dictionaries, grammars, grammar and style checkers or CALL (Computer Assisted Language Learning) software, have two things in common. Firstly, they no longer use invented examples, opting instead for examples taken from corpora of authentic language. And secondly, the tools seem in general to be designed for all learners of English, irrespective of their mother-tongue.

The focus on authenticity originated with the Collins Cobuild dictionary project, which gave rise to a whole range of EFL tools based on authentic data. Underlying the Collins Cobuild approach was the firm belief that better descriptions of authentic native English would lead to better EFL tools and indeed, studies which have compared materials based on authentic data with traditional intuition-based materials have found this to be true. In the field of vocabulary for example, Ljung (1991) has found that traditional textbooks tend to over-represent concrete words to the detriment of abstract and societal terms and therefore fail to prepare students for a variety of tasks, such as reading quality newspapers and report-writing. The conclusion is clear: textbooks are more useful when they are based on authentic native English.

As for the 'generic' nature of most EFL tools, there seem to be a variety of reasons for this, some theoretical, some practical and not

We acknowledge the support in this research provided by the Belgian National Scientific Research Council, the Ministry of Education of the French-speaking Community of Belgium and the University of Louvain Research Fund.

Linguistic Databases.
John Nerbonne, ed.

least of which is the principle of unilingualism in ELT—i.e. the exclusive use of English in the teaching/learning process—which has undoubtedly played a large role. Although this principle may no longer be the dogma that it once was, it still nevertheless dominates the ELT scene. The general focus on universal features of learner language may also play a part. But the main reason is probably a practical, a commercial one. There is a much bigger market for all-round ELT tools than for L1-specific tools, which are obviously much more of a commercial risk for publishers. But this attitude is no longer acceptable at a time when all specialists recognize the importance of transfer in second language acquisition. Luckily, there are signs that things are changing, with the recent appearance of several bilingual reference books aimed at specific language groups.[1] It has to be said though that these bilingual tools are still the exception rather than the rule.

This creates something of a paradox. On the one hand is the belief that ELT materials should be based on solid, corpus-based descriptions of *native* English. On the other hand, materials designers are content with a very fuzzy, intuitive, non-corpus-based view of the needs of an archetypical *learner*. However, the efficiency of EFL tools could be improved if materials designers had access not only to authentic native data but also, as suggested in Figure 1, to authentic learner data, with the NS (native speaker) data giving information on what is typical in English, and the NNS (non-native speaker) data highlighting what is difficult for learners in general and for specific groups of learners.

FIGURE 1 Source data for EFL tools

The aim of this paper is to demonstrate the usefulness of using computer learner corpora in the design of EFL tools, with specific reference to electronic grammar and style checkers.

[1]Cambridge University Press has recently brought out a new series called Cambridge Word Routes, which is a sort of bilingual thesaurus. And Collins has recently published a student's dictionary specially developed for Brazilian learners of English.

2 Computer Learner Corpora

The main advantage of the computer learner corpus (CLC) is that, benefiting from the techniques and tools developed in corpus linguistics, it can provide the EFL sector with a much more solid and versatile empirical foundation than has previously been available. CLCs are a relatively new development in corpus linguistics, and very few have been compiled to date, but the enthusiasm with which existing CLCs have been met suggests there will be rapid proliferation.

2.1 Corpus design criteria

As in other areas of corpus linguistics, it is important to ensure that certain criteria are established for the building and analysis of CLCs. In the discussion of these criteria which follows, particular reference will be made to the **International Corpus of Learner English (ICLE)**, a computerised corpus of writing of learners from different backgrounds.[2]

In their seminal 1992 article Atkins, et al. laid down the basic standards for corpus design. Their aim was to establish a set of criteria "necessary to foster the creation of high-quality compatible corpora of different languages, for different purposes, in different locations, and using different types of software and hardware" (p. 1). The need for explicit corpus design criteria is particularly acute in the case of learner language because of the inherent heterogeneity of learner output. Among the text attributes listed by Atkins, et al., the following are of particular relevance to EFL corpus building: *mode* (written vs. spoken), *genre* (essay, letter, conversation, etc.), *function* (narrative, expository, reflective, etc.) and *technicality* (general, technical, semi-technical). Most current EFL corpora cover the same mode—writing—but differ in other features of the writing assignment. The **Longman Learner Corpus** for example, contains a variety of genres, functions and degrees of technicality. **ICLE** on the other hand, a smaller, specialized corpus, contains only one task type, non-technical argumentative essay writing. Thus while it cannot make statements about all learner writing, it can be used to make reasonably definitive statements about a particular genre of learner writing.

Because of the major role played by transfer in second language acquisition, the language of the writer, an attribute which is, according to Atkins, et al. (1992:8) "in many cases unavailable or irrelevant", assumes particular importance in the CLC framework. An L1-undifferentiated EFL corpus would therefore be of relatively little interest. EFL corpus

[2]For more information on the ICLE database, see Granger (1993, 1994, 1996, and in press).

builders should either focus on one specific group of learners (the **Hong Kong University of Science and Technology Learner Corpus**, for instance, covers writing by Chinese (primarily Cantonese) learners) or cover a variety of mother tongue backgrounds (the option chosen by both the compilers of the Longman Learner Corpus and ICLE).[3] The advantage of the multi-language background learner corpus is that it makes it possible to distinguish between areas of difficulty specific to one language group and features common to several, perhaps all language groups, a distinction which would make it possible to improve both generic and bilingual EFL tools.

2.2 A computer-based methodology

A machine-readable raw (unannotated) learner corpus can be submitted to a whole range of text-handling software tools, thereby providing EFL analysts with a wealth of quantitative and qualitative data which has hitherto been inaccessible. The quantitative measures which are most easily provided are frequency counts of morphemes, words and phrases. Comparisons with NS (native speaker) frequency counts highlight patterns of over- and underuse which often provide impetus for further analysis. Concordancing software enables the researcher to take a more qualitative look at the learner data and detect cases of syntactic, semantic or stylistic misuse. Though research in this area is still in its infancy, this computer-aided methodology has already shed new light on several aspects of learner language: use of connectors (Milton & Tsang 1993; Granger & Tyson 1996), collocations and lexical phrases (Chi, et al. 1994; Granger forthcoming), verbs of saying (Tribble 1991), modal expressions (Dagneaux 1995). Figure 2, which displays concordances of the word **anyway** extracted from the ICLE database (French learner subcorpus) and a comparable corpus of native English writing, demonstrates clearly that while native speakers use the word **anyway** primarily in clause or sentence final position, French learners distinctly favour sentence initial position.

NS/NNS concordance-based comparisons frequently highlight erroneous patterns in learner writing, such as the use of the marked infinitive after **accept** alongside the grammatical **that**-clause complementation in Figure 3.

Results of such comparisons can be used by materials designers to draw users' attention to frequent pitfalls. Specialists in EFL pedagogy

[3]The International Corpus of Learner English contains thirteen subcorpora which correspond to the following mother tongue backgrounds: French, Dutch, German, Spanish, Swedish, Finnish, Polish, Czech, Russian, Bulgarian, Italian, Japanese and Chinese.

Non-native writing

some time dreaming or not.	>Anyway we cannot affirm
some kind of bygone days.	>Anyway, a solid amount
ecological policies.	>Anyway, they do not have
orientations has to be found.	>Anyway, "wait and see" see
am I going to far?	>Anyway, what must be kept in
labyrinth with reality	>Anyway, when the dreams
death, life will go on	>anyway. Those numbers have
whether it is possible and	>anyway, it is not true

Native writing

The fact that he married her	>anyway, and the photos of
should have thrown it	>anyway. By saving those two
death to a crime of passion,	>anyway, as yet another
they're going to die	>anyway. This shows that
when we are going to die	>anyway. The laugh causes
being looked on favorably	>anyway, after accepting the
were not eligible for it	>anyway. Then, as we can
Greenland was an exception	>anyway because it's reason

FIGURE 2 Concordances of **anyway** in NS and NNS writing

have also suggested using such concordances in the classroom as starting points for discussion of grammar and usage in the context of data-driven learning (see Tribble & Jones 1990; Pickard 1994; Johns1994).

Concordancers are of great value in highlighting marked and/or erroneous use in learner data. However current concordancers have limitations. As Kirk (1994: 25) points out, one of their major weaknesses is that they operate on the basis of unlemmatized word forms. This means for example, that anyone interested in NNS use of the verb **be** will have to search for 9 different forms (**be, 's, is, isn't, was, wasn't, were, weren't, been**). A lemmatizer would automatically group together all inflected forms of one and the same lemma. Regrettably, none of the currently available concordancing packages contain a fully automatic lemmatizer, although *WordSmith*, the new package from Oxford University Press, contains a lemmatization facility, which enables users to link several entries.

Another difficulty faced by EFL analysts working with raw learner corpora is that words in English frequently belong to more than one word category. A search for the major prepositions in learner writing will bring up all the occurrences of **in, over, after, etc.** as adverbial particles, thus forcing the analyst to undertake a long and tedious process of manual disambiguation. This can be avoided if the learner

Non-native writing

families, the parents accept	>that new visions of things
don't always accept	>that their children also
nor the children accept	>to recognize that
young. He could never accept	>to be inferior
Feminists have to accept	>to be treated as men

Native writing

not being able to accept	>that fulfilment of
the act. Hugo cannot accept	>that the party line has

FIGURE 3 Concordances of **accept** in NS and NNS writing

V (montr,*) + ADV (ge/intens/partic) + ART/N(com)
Verb (monotransitive, pres/past/-ed participle) + Adverb (general, intensifier or particularizer) + Article or Common Noun.

Examples of NNS structures:
...and to select carefully the programmes.
...people like maecenas able to support materially artists.
...This justifies naturally the success of all these games.

Example of NS structure:
This illustrates emphatically the folly of l'Optimisme.

FIGURE 4 Combination-of-tags search in NS and NNS data

corpus is tagged. The ICLE corpus is currently being tagged with the TOSCA analysis system.[4] Preliminary investigation brings out the advantages of a tagged CLC over a raw CLC: it is both more efficient, in that it only gives the analyst what he/she wants (prepositions and not adverbial particles, for instance), and more powerful, in that it allows for grammatical as well as lexical searches. A search for the grammatical tag AUX (modal) in the French subcorpus of ICLE by Meunier (1995) and a similar-sized native corpus using the same tags brings out a significant overuse of modal auxiliaries by the learners. A subsequent lexical search proves this to be to a large extent due to overuse of **can**. Such comparisons are sure to shed new light on learners' use of grammar. It is also possible to search for sequences of tags. Figure 4 (taken from Meunier 1995) illustrates a search for the verb-adverb-noun sequence in

[4]The TOSCA analysis system was developed by Jan Aarts and his team at the University of Nijmegen within the framework of the International Corpus of English (ICE) project, with which the ICLE corpus has close links.

NS and NNS data, a structure which is both overused and misused by French learners of English.

3 A Computational Model of EFL Performance

Though it is rarely stated explicitly, the underlying assumption behind most grammar checkers is that they are intended for any user, native or otherwise. However, several studies have demonstrated that existing grammar checkers are "not reliable writing aids for non-native speakers" (Milton 1994:66) and indeed, if one analyses their performance in detail, it becomes apparent that they cater primarily for the needs of the native writer.

This was confirmed by a recent study which compared the effectiveness of two widely-used checkers, Correct Grammar and Grammatik, in analyzing samples of native and non-native data (Granger & Meunier 1994). Whilst problems shared by both native and non-native speakers such as concord or double negation had a reasonable success rate, the rate of detection for typically non-native problems was generally low. The whole area of lexico-grammar is a case in point. The success rate of dependent preposition errors (**discuss *about sth, depend *to sth**) or non-finite verb complementation errors (**prevent sb *to do sth, insist *to do sth**), for instance, proved to be extremely low.

What is lacking is an adequate computational model of EFL performance. According to Catt & Hirst (1990) it is this that is holding back the whole field of computer-assisted language instruction (CALI). Current CALI systems, which merely match students' answers with a predefined list of responses, should be replaced by more intelligent CALI (ICALI) systems which can deal with learners' free-form input. For that however, it is necessary to construct a computational model of EFL performance: "Only if the learner's competence can be modeled computationally is there hope of developing CALI systems capable of dealing intelligently with learner language" (Catt & Hirst 1990:6).

The *Scripsi* system developed by Catt & Hirst remedies this deficiency "by incorporating a credible model of the second language learner's linguistic competence, one that takes into account the phenomena of transfer and overgeneralization" (1990:22). This rule-based error diagnostic tool detects transfer errors by allowing the application of L1 rules in the parsing process. It detects overgeneralization errors by relaxing constraints on feature values (for a full description of the system, see Catt 1988:27–53). *Scripsi* is a prototype system, which only caters for a very limited number of errors from French- and Chinese-speaking learners of English. Other error-diagnosing parsers, which combine 'cor-

rect' and 'lenient' rules, are being developed (Covington & Weinrich 1991; Luomai 1994), but they suffer from the same weakness as *Scripsi*, i.e. they have a very low lexical and syntactic coverage.

If they are to be truly effective, error diagnostic tools need to cater for the most typical errors in a particular learner population. As Tschichold (1994:198) rightly points out: "The fact that a grammar checker can correct an error the user is not ever likely to make may be impressive, but it is not very useful to users". In constructing their model of learner competence, Catt & Hirst have used data from well-known EA (error analysis) studies. Such data suffer from two major weaknesses: size and heterogeneity. Most EA studies are based on very small samples of learner language, sometimes no more than 2,000 words representative of a dozen or so learners. In addition, EA researchers have often not paid attention to the variety of factors which can influence learner language, such as mother tongue background, medium, level of proficiency, etc. EA data may therefore be a good source of data to build a prototype but will be of little use beyond that stage. In the following section I will show how computer learner corpora, which contain large quantities of carefully collected learner data, can contribute to improving the diagnostic capabilities of grammar and style checkers.

4 Contribution of CLC Data to Grammar and Style Checkers

This section reports on some preliminary results of an ongoing two-year project to adapt current English grammar and style checkers for French-speaking users. The data used in the project is the 300,000 word component of the ICLE database, which contains samples of writing from 450 different advanced French-speaking learners. With a view to widening the proficiency range of the data this corpus was supplemented with a 75,000 word corpus of writing by intermediate learners.

The first stage of the project involved manually correcting and error tagging a 150,000 word subset of the overall corpus, evenly distributed between intermediate and advanced. The error tagging system devised for the project is based on the following principles. Each error tag carries a general category tag: G for grammatical, L for lexical, F for formal, etc. and one or more specific codes. For instance, the GVT code refers to grammatical verb tense errors. Figure 5 shows a sample of error tagged material: errors are preceded by the relevant error tag and followed by the corrected form.[5]

[5]The insertion of the corrected form seems to contradict a statement made in a previous article according to which "Errors should not be normalised, as this involves

```
The first positive aspect would be the age of the child.  (CLS)
Actually $In fact$ studies carried out by eminent linguists have
proved that the ideal (FS) adge $age$ for learning a foreign
language is between 3-10 years.  (CLC) Of course $0$ no consensus
(GVAUX) could be reached $has been reached$ about this subject as
others (LSF) pretend $claim$ that this trend would lead to mental
fatigue, overworking and that it could disturb the children's
(GNN) mind $minds$.  In their opinion, it would be preferable to
learn a second language only after having a (FS) throughout
$thorough$ knowledge of the first.
```

FIGURE 5 Sample of error-tagged text

the fact that we could	>(GVPR) argue on $argue about$ the definition
want to be parents, do not	>(GVPR) care of $care about$ the sex
is rising. These people who	>(GVPR) come in $come to$ Belgium
Family planning	>(GVPR) consists on $consists of$
have the possibility to	>(GVPR) discuss about $discuss$ their problems
which the purchaser cannot	>(GVPR) dispense of $dispense with$
the health. Nobody	>(GVPR) doubts about $doubts$ that.
harvest they got is often	>(GVPR) exported in $exported to$ countries

FIGURE 6 Error tag search: dependent prepositions

Once a text has been tagged for errors, it is possible to draw up comprehensive inventories of specific error types. Figure 6 shows the concordance of error-tagged data sorted on the error tag GVPR, which marks errors made with dependent prepositions following verbs. Figure 7 contains a list of LSF errors, i.e. lexical single word errors due to the influence of a formally similar word in the user's mother tongue (the so-called 'faux amis') and Figure 8 contains count/uncount noun errors.

By running these lists of errors through current checkers, it is possible to give a precise assessment of which rules—be they grammatical, lexical, orthographic or stylistic—need to be refined or added to the existing stock. One area among many where improvement is badly needed

a high degree of subjectivity, given that many errors—particularly lexical, stylistic and textual ones—can be corrected in many different ways" (Granger, Meunier & Tyson 1994:105). Corrected forms were inserted within the framework of this project because it was useful for the researchers to have access to them when analysing and categorizing the error data.

seems to us, thanks to its	>(LSF) actual $modern$ style of
West leads the	>(LSF) actual $present$ world of economy
fall into despair, the truth will	>(LSF) conduct $drive$ him to suicide
goes to his or her	>(LSF) course $lecture$ but because
to others, it is rather	>(LSF) deceiving $disappointing$
it is not impossible that such	>(LSF) experiences $experiments$
employ an	>(LSF) important $large$ number of
be shown thanks to a certain	>(LSF) mark $brand$ of cigarettes
since they are allowed to	>(LSF) penetrate $enter$ our country
health of stressed	>(LSF) pork $pigs$

FIGURE 7 Error tag search: faux amis

of advice on	>(GNUC) a 0 better health care
for years. Undoubtedly	>(GNUC) a 0 big progress has been made
you from breathing	>(GNUC) a 0 pure air
seems to be different.	>(GNUC) A 0 clear evidence is the percentage
characteristic	>(GNUC) behaviours $behaviour$
It provides	>(GNUC) employments $employment$
combining study life and	>(GNUC) leisures $leisure facilities$
a balance between work and	>(GNUC) spare times $spare time$
need to do some	>(GNUC) works $work$ or simply for your personal

FIGURE 8 Error tag search: uncount nouns

is that of the count/uncount distinction illustrated in Figure 8. This distinction is responsible for many errors in learner writing: it affects number (***leisures** vs **books**), article usage (***without car** vs **without passion**), use of indefinite determiners (***many leisure** vs **many books**) or quantifying expressions (***a great deal of books** vs **a great deal of money**), etc.

Our research, however, is not limited to the analysis of error lists. Besides identifying errors, a good grammar and style checker should also detect and correct stylistic infelicities in learner writing. The clichés, circumlocutions and other infelicitous words and expressions highlighted by current checkers are not very useful because they are typical of native usage. A phrase such as **I don't doubt but that** is not likely to be used by learners. Here too CLC data prove very useful. The whole 300,000 word database can be scanned for recurring words and phrases and the results compared with a corpus of native writing of the same genre. Such a search brings out overused phrases such as **we can say that, we must not forget that, from the point of view of, as far as (x is concerned)** which make learner writing sound verbose and pompous.

Though the project is not complete, some preliminary conclusions can already be drawn.

First of all, the high number of L1-dependent errors detected would seem to require that separate programs be developed for each mother tongue background. The LSF category alone, i.e. the so-called 'faux amis' illustrated in Figure 7, makes up one third of all the lexical errors in the corpus. Only when several mother tongue backgrounds have been covered will we be able to determine the nature and extent of the common core of EFL problems.

Secondly, it would seem to be essential that an effective grammar and style checker should cater for learners' lexical errors. Our research demonstrates that a significant number of EFL errors are lexical. A key objective of the project will be to assess how best to cater for lexical errors.

In general, it seems reasonable to say that CLC—both in their tagged and untagged versions—prove to be a particularly rich source of data for improving grammar checkers. Error-tagged CLC bring out the typical errors in a given learner population, informing us of the relative weight of each category type and spelling out the most error prone members in each category. Untagged CLC bring out stylistic infelicities in learner writing.

It should be noted, however, that although our research so far shows clearly that all components of grammar checkers can be substantially

improved by the use of CLC data, there is still a large proportion of errors which will remain beyond the capabilities of grammar checkers for the foreseeable future, because they are semantic in nature. This restriction affects lexis as well as grammar: many grammatical errors are undetectable because they display no formal sign of error. For instance, article errors due to the generic vs specific distinction (**I like the carrots** vs **I like carrots**), modal auxiliary errors (the difference between **I can do it** and **I may do it**) or discourse-level pronoun errors (**it is true** vs **this is true**) all escape detection. As a consequence, grammar checkers need to be viewed as one component in a wider writing workstation, which would include a series of interactive lookup facilities, such as an online grammar, an online learners' dictionary and an online collocational database. As Milton (1994:68) rightly points out: "students need more than just pruning shears to cut away errors (...) They also require, in the absence of human assistance, a better online expert than the type that is now available".

5 Conclusion

The native speaker perspective occupies a central position in current ELT material. It is typicality of usage in native English that determines syllabus design; it is native speaker language that serves as a testbed for grammar checkers. This is understandable since native proficiency is what all learners are after. In fact, EFL materials designers need to have access to more detailed descriptions of native English use than are currently available, enabling improvements along the lines of the frequency data which has been incorporated in the latest editions of the Collins Cobuild Dictionary and the Longman Dictionary of Contemporary English. However, pedagogic practice shows that the native perspective needs to be supplemented with a learner perspective. Computer learner corpora are the best way of finding out about learners' difficulties and they will undoubtedly play a major role in the development of future EFL tools.

References

Aarts, J., P. de Haan, and N. Oostdijk, eds. 1993. *English Language Corpora: Design, Analysis and Exploitation.* Amsterdam/Atlanta: Rodopi.

Atkins, S., J. Clear, and N. Ostler 1992. Corpus Design Criteria. *Literary and Linguistic Computing* 7/1: 1–16.

Catt, M. 1988. *Intelligent Diagnosis of Ungrammaticality in Computer-Assisted Language Instruction.* Technical Report CSRI-218. Computer Systems Research Institute: University of Toronto.

Catt, M., and G. Hirst. 1990. An Intelligent CALI System for Grammatical Error Diagnosis. *CALL*, Volume 3, 3–26.

Chi, A. M., K. W. Pui-yiu, and M. W. Chau-ping. 1994. Collocational problems amongst ESL learners: a corpus-based study, in Flowerdew & Tong, eds., 157–65.

Covington, M., and K. Weinrich. 1991. Unification-based Diagnosis of Language Learners' Syntax Errors, *Literary and Linguistic Computing* 6/3: 149–54.

Dagneaux, E. 1995. *Epistemic modal expressions in native and non-native writing.* Unpublished MA Dissertation, Université Catholique de Louvain: Louvain-la-Neuve.

Flowerdew, L., and A. K. K. Tong, eds. 1994. *Entering Text.* The Hong Kong University of Science and Technology.

Fries, U., G. Tottie, and P. Schneider, eds. 1994. *Creating and using English language corpora.* Amsterdam/Atlanta: Rodopi.

Granger, S. 1993. International Corpus of Learner English, in J. Aarts, P. de Haan & N. Oostdijk eds., 57–71.

Granger, S. 1994. The Learner Corpus: a Revolution in Applied Linguistics. *English Today* 10/3: 25–9.

Granger, S. 1996. Learner English around the World, in S. Greenbaum, ed. *Comparing English World-wide.* Clarendon Press: Oxford, pp. 13–24.

Granger, S., ed. In press. *Leaner English on Computer.* Addison Wesley Longman.

Granger, S. Forthcoming. Prefabricated patterns in advanced EFL writing: collocations and formulae, to appear in: T. Cowie, ed., *Phraseology.* Oxford: Oxford University Press.

Granger, S., and F. Meunier 1994. Towards a grammar checker for learners of English, in Fries, et al., eds. pp. 79–91.

Granger, S., F. Meunier, and S. Tyson 1994. New Insights into the Learner Lexicon: a preliminary report from the International Corpus of Learner English, in L. Flowerdew & A. Tong, eds. pp. 102–113.

Granger, S., and S. Tyson 1996. Connector usage in the English essay writing of native and non-native EFL speakers of English. *World Englishes*, Volume 15, No. 1: 17–27.

Johansson, S., and A. B. Stenström, eds. 1991. *English Computer Corpora.* Mouton de Gruyter: Berlin & New York.

Johns, T. 1994. From printout to handout: Grammar and vocabulary teaching in the context of Data-driven Learning, in T. Odlin ed. 293–313.

Kirk, J. 1994. Taking a byte at Corpus Linguistics, in L. Flowerdew & A. Tong, eds. pp. 18–49.

Ljung, M. 1991. Swedish TEFL meets reality, in Johansson & Stenström, eds. pp. 245–56.

Luomai, X. 1994. Smart Marker—an efficient GPSG parser, in L. Flowerdew & A. Tong, eds. pp. 144–56.

Meunier, F. 1995. Tagging and parsing interlanguage, in: L. Beheydt, ed., *Linguistique appliquée dans les années nonante*, ABLA Papers 16:21–9.

Milton, J. 1994. A Corpus-Based Online Grammar and Writing Tool for EFL Learners: A Report on Work in Progress, in Wilson & McEnery: 65–77.

Milton, J., and K. S. T. Tong, eds. 1991. *Text Analysis in Computer-assisted Language Learning.* The Hong Kong University of Science and Technology.

Milton, J., and E. S. C. Tsang 1993. A corpus-based study of logical connectors in EFL students' writing: directions for future research, in Pemberton & Tsang, eds. pp. 215–46.

Odlin, T., ed. 1994. *Perspectives on Pedagogical Grammar.* Cambridge: Cambridge University Press.

Pemberton, R., and E. S. C. Tsang, eds. 1993. *Studies in Lexis.* The Hong Kong University of Science and Technology.

Pickard, V. 1994. Producing a concordanced-based self-access vocabulary package: some problems and solutions, in Flowerdew & Tong, eds. pp. 215–26.

Tribble, C. 1991. Electronic Text Analysis in Curriculum Design and Evaluation, in Milton & Tong, eds. pp. 4–14.

Tribble, C., and G. Jones 1990. *Concordances in the Classroom.* Longman.

Tschichold, C. 1994. Evaluating second language grammar checkers. *TRANEL* 21: 195–204.

Wilson, A., and T. McEnery, eds. 1994. *Corpora in Language Education and Research.* Unit for Computer Research on the English Language: Lancaster.

12

Linking WordNet to a Corpus Query System

Oliver Christ

This paper describes how the on-line thesaurus WordNet[1] (Miller et al. 1993a) can be linked to a corpus query system in order to use the semantic information provided by WordNet (especially information about super- and subordination relations in the vocabulary of nouns and verbs) for the retrieval of concordances from a corpus.

We show that a corpus query system capable of using an on-line thesaurus supports an interesting class of queries and concordances. The mechanism which is used to link WordNet to the corpus query system is very flexible, so that the same mechanism can be used to link almost arbitrary sources of information to the query system in a modular and flexible way.

1 Introduction

The IMS corpus query system (Christ 1994) is an efficient and modular corpus query system capable of handling an arbitrary number of corpus annotations of different types. Annotations are usually *statically annotated*, that is, the corpus query system itself maintains the data and the indices and has built-in routines to access the information. Annotations like the "corpus text" proper, part-of-speech tags, sentence boundaries etc. usually belong to this group.

In order to use knowledge from external (linguistic) knowledge sources in corpus queries, *"dynamic attributes"* can be declared for a corpus.

I am greatly indebted to an anonymous reviewer who suggested I include a discussion of the approach in terms of "precision" and "recall" and who gave a lot of other important comments on an earlier version of this paper.

[1] WordNet™ is a registered trademark of Princeton University.

Linguistic Databases.
John Nerbonne, ed.

The data of such dynamic attributes (or "virtual attributes") is not maintained by the corpus query system (or the corpus data manager) itself, but rather by external tools (for example a database or even a shell script). When data from such dynamic attributes is needed, for example when a corpus query is being evaluated, the corpus query system makes a data connection to the external tool which then in turn is responsible to compute and return an appropriate value. This value is then further processed by the corpus query system or the corpus data manager. This mechanism therefore supports the reuse of "corpus-external" linguistic information in corpus exploration, with the hope that additional information allows the linguist to express more detailed "filters" on the corpus data so that less material is being retrieved and has to be browsed manually.

From a conceptual point of view, dynamic attributes are modeled as function calls: each dynamic attribute has an argument list which is filled by actual parameter values at the time of access. Then, an external tool is called with these parameters by using the Unix `pipe` mechanism of the C library. The value returned by the call of the external tool is then passed back to the query evaluator, where it is used and processed in the same way as values retrieved from internally stored annotations (this concept is illustrated in figure 1).

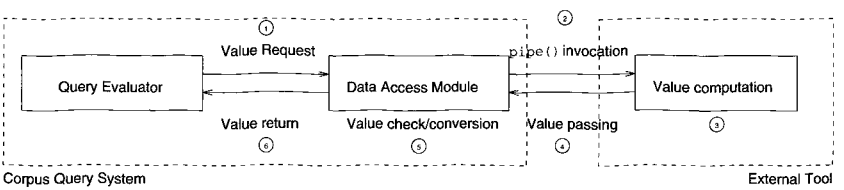


FIGURE 1 The idea of "dynamic" or "virtual" attributes

Dynamic attributes are mainly useful in cases where the query system does not directly support the internal (static) representation of the information in question. This is, for example, the case for thesaurus-like relations where the information must be stored whether two (arbitrary) words are in a given (semantic) relation or not. The data structures which would be most suitable to store this kind of information are, however, not directly supported by our query system.

In the experiments reported in this paper, the lexical knowledge base WORDNET (an online-thesaurus) was linked to the IMS Corpus Query System by using dynamic attributes to circumvent the limitation mentioned above.

Section 2 describes the types of information requested from WORD-

NET in the experiments. Section 3 then explains the general concept by which external knowledge sources are linked to our corpus query system. A first example is presented in section 4. A more complex example, the retrieval of verb-noun combinations which have certain semantic properties, is introduced in section 5. The queries which can be used to retrieve examples for these verb-noun combinations are then shown in section 6, some of the results which are produced by these queries are listed in section 7. A central problem of the concept of linking external knowledge sources to a corpus query system is access efficiency. In a first naive approach, access was very slow, so that more efficient models of data exchange had to be implemented, which are outlined in section 8. The paper finishes with a short discussion in section 9.

2 The subset of WORDNET information used in the experiments

WORDNET provides information about hypernym, hyponym, synonym and antonym relations as well as some other lexical/semantic relations between English nouns, verbs, and partially between adjectives and adverbs (refer to Miller et al. 1993b for a detailed description of the information provided by WORDNET).[2]

In the experiments reported here, we only use hyponym and hypernym relations between verbs and nouns, although the other WORDNET relations could also be used. Further, we restrict the type of information requested from WORDNET: for our purposes, we assume that we are merely interested in whether two words are in a given WORDNET relation or not. This can be modeled by computing the smallest number of "steps" which have to be taken from any of the senses of the subordinate word to reach the superordinate word when going "upwards" in the hierarchy.[3] For easier processing, we assume that a zero distance means that two words are not within the given relation, and a value of one means that the two words are equal. Any value $n > 1$ means that $n - 1$ steps have to be taken to reach the superordinate.

3 The declaration of dynamic attributes

Dynamic attributes must be declared so that the query processor knows about the name of the attribute, the arguments to be passed to the external tool, their types, the return value type, and the external tool which is to be called upon a value request. This declaration is done

[2]In this paper, we use WORDNET version 1.4, although version 1.5 is meanwhile available.

[3]Of course we do not claim that this simple measure is a measure of "semantic distance".

in a central file (the "registry file") which exists once for each corpus the query system knows about. Such a registry file lists the physical properties of the corpus (for example, where the data is stored) and the set of annotations which are available for the corpus (part-of-speech tags, lemmas, sentence boundaries, etc.).[4] The declaration for a dynamic attribute may look as follows:

```
DYNAMIC wndist(STRING,STRING,STRING):INT
   "/corpora/bin/wnreq -s '$1' '$2' '$3'"
```

This means that the dynamic attribute `wndist` takes three string-type arguments and returns an integer value (the "distance" between two words in a given relation). The part surrounded by double quotes describes which external tool is called when a value of the dynamic attribute is requested (`/corpora/bin/wnreq`), and how the (actual) parameters of the dynamic attribute are passed to the external tool. `$1`, `$2` and `$3` refer to the first, second and third parameter of the dynamic attribute, respectively. The three arguments are the hyponym word, the relation name, and the hypernym word. The return value, an integer number, is computed from the output of the call of the external program `wnreq`, called with these three arguments.

Dynamic attributes can be assigned to (or removed from) a corpus at any time, simply by changing the registry file. Instead of WORDNET, other external knowledge resources could be linked to the query system in a similar way, for example, MRDs (or lexical databases) with suitable filters and access programs. The advantage of the declarative specification of external knowledge sources in the registry file is that the query processor does not need to have inherent knowledge about how to access the external resource (as it would be the case when the access procedures would be linked into the query processor program). The only link between the query processor and the external resource is the shell call which is defined in the registry file. The disadvantage of this approach is poor access efficiency, as will be discussed below in section 8.

4 A first example

For a first example, we investigate occurrences of the verb *to kill* with a human object.[5] The following query is a simple formalization of this search problem:

[4]The IMS Corpus Query System supports an arbitrary number of annotations of certain types. Refer to Christ (1994) for an overview.

[5]In the examples presented in this paper, we use a subset of the part-of-speech tagged portion of the Penn Treebank Corpus (see, for example, Marcus et al. 1993) and the subset of the Brown corpus which is part of the WordNet 1.4 distribution.

```
[lemma="kill" & pos="V.*"] [pos="DT"]?
  [pos="N.*" & wndist(lemma,"hypen","human")>0]
```

This means that we are looking for verb occurrences of the lemma *kill*, optionally followed by a determiner, and then followed by a noun where the "WORDNET distance" between the noun and the concept "human" in the relation "hypen" (the hypernym relation for nouns) is greater than zero. The following concordance is a part of the result of this query (the "match" is enclosed in angle brackets):

```
in one botched job <killed a client> . Her remorse was sh
lem group vowed to <kill Americans> if the U.S. implement
ld single-handedly <kill the BART> funds unless the congr
 sugar , which can <kill diabetics> . .PP The usual warni
 of threatening to <kill Mr.> Laff . .PP Mr. Laff 's atto
 way the U.S. will <kill a madman> is by making sure we t
 used primarily to <kill worms> in the intestinal tracts
o blame because he <killed the woman> without `` will or
```

When looking for the nouns which are not classified as "human" (by using the condition `wndist(lemma,"hypen","human")=0`) we not only find non-human objects (such as *project, inflation, proposal, ...*), but also nouns like *people*, which WORDNET does not classify as "+HUMAN".[6] Figure 2 shows part of the concordance of non-human objects of the lemma *kill*.

5 Retrieving verb-noun combinations with certain semantic properties

The objective of the queries presented in this section is to extract verb-noun collocation candidates from a corpus of English where the verb is in a hyponym relation to a given "more general" verb (or "concept") and the noun is in a hyponym relation to a given more general noun.[7] The two combinations we were interested in are shown in table 1.

	Verb	Noun
1.	utter	message
2.	create_by_mental_act	message

TABLE 1 V/N combinations used in our experiments

In the second experiment, for example, concordances were sought in the corpus where the verb is a hyponym of the WORDNET verb entry *cre-*

[6] Instead, *people* (as well as, for example, *police*) is a subordinate of "group", which in turn does not have any superordinates.

[7] These experiments were motivated and conducted by Leo Wanner, formerly at the Institute for Romance Linguistics, University of Stuttgart.

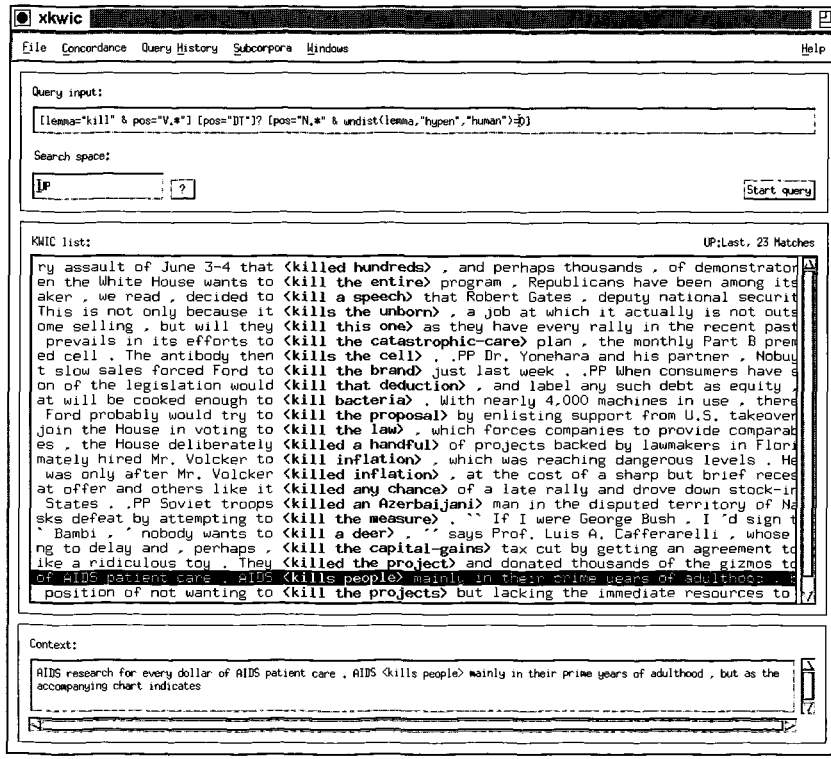


FIGURE 2 The retrieval tool XKWIC, showing part of the concordance generated by a query for non-human objects of the lemma *kill* when run on the part-of-speech tagged portion of the Penn Treebank.

ate_by_mental_act and the noun is a hyponym of the WORDNET noun entry *message*. These WORDNET verb/noun "classes" were determined manually. The second example therefore should retrieve verb-noun combinations where the verb is some way of "creating [something] by mental act" and the noun is a kind of "message".

The next section shows how these search problems can be formalized.

6 Formalization into corpus queries

The two noun/verb combinations listed in table 1 have been searched in the corpus with at most 10 words in between (disallowing punctuation marks, prepositions and subordinating conjunctions) with the (obviously too simple) assumption that the first noun following the verb is its direct object. The verb and the (following) noun had to be in the same sentence. The dynamic attribute `wndist` computes the "distance" between two given words in a given WORDNET relation (here, the verb-hypernym

relation hypev and the noun-hypernym relation hypen are used). In our corpus query system, the formalization for the first search can be expressed in the following way (the hash sign # introduces a comment which expands until the end of the line):

```
v:[pos="V.*"]                                  # 1: a verb
[pos!="N.*|V.*|IN" & word!="\.|,|;|:"]{0,10}   # 2: up to 10 tokens
n:[pos="N.*"]                                  # 3: a noun
:: wndist(v.word,"hypev","utter")>0 &          # 4: verb <= "utter"
   wndist(n.word,"hypen","message")>0          # 5: noun <= "message"
within s;                                      # 6: all within 1 sentence
```

Example 1

In line 1, a token is searched which has a part-of-speech tag which begins with a capital "V" (which is the case for all verb classes in the underlying part-of-speech tagset). The token is labeled with a "v" (by preceding the left square bracket with "v:". In the query language, it is possible to refer back to values at labeled positions). Line 2, then, expresses the form of the tokens which may be between the verb and the noun in question. In order to reduce noise, we do not allow nouns (N.*), verbs (V.*), and prepositions or subordinating conjunctions (IN).[8] Further, we exclude punctuation marks. Between zero and ten of such tokens are allowed ({0,10}). In line 3, we finally request that a noun follows, which again is labeled in order to be able to refer to that noun later. Lines 4 and 5, then, express further conditions on the semantic properties of the verb and the noun. These conditions are specified in a "general constraint" which is evaluated after all other conditions have been successfully evaluated by the query evaluator. This "general constraint" is introduced by "::". In line 4, the wndist dynamic attribute (or "function") is used to check whether the verb (v.word) is a hyponym of "utter", and line 5 requires that the noun (n.word) is a hyponym of "message". Finally, line 6 requires that the whole matching token sequence lies within one sentence.

The query can also be expressed without using labeled tokens and without the "general constraint". Instead, the wndist calls can be written as part of the conditions of the verb and the noun, respectively:

```
[pos="V.*" & wndist(word,"hypev","utter")>0]
[pos!="N.*|V.*|IN" & word!="\.|,|;|:"]{0,10}
[pos="N.*" & wndist(word,"hypen","message")>0]
within s;
```

Example 2

[8] Excluding all prepositions between the verb and the noun is probably too restrictive. With a different tagset, a finer distinction (e.g. excluding only relative pronouns) could have been expressed.

Although query (2) is shorter and possibly reflects a more intuitive way to express the query conditions, it is less efficient than query (1) above: since the "general constraint" (the conditions written after the "::") is only evaluated *after* the other conditions (verb, followed by at most 10 restricted tokens, followed by a noun) have been (successfully) tested for a given token sequence in the corpus, the external knowledge source has to be called only for this restricted set of candidates, and not for all verbs in the corpus, as it is the case in query (2). In the present case, it is thus possible to reduce the number of "external calls" by ≈2/3, which reduces the time to evaluate the query from ≈1 hour to 20 minutes.[9]

The other query is expressed accordingly:

```
v:[pos="V.*"] [pos!="N.*|V.*|IN" & word!="\.|,|;|:"]{0,10} n:[pos="N.*"]
:: wndist(v.word,"hypev","create_by_mental_act")>0 &
   wndist(n.word,"hypen","message")>0 within s;
```

Example 3

Table 2 illustrates the number of matches for each of the two queries. Some randomly selected lines of the resulting concordances are shown in the following section 7.

Query	Nr. of Matches
utter/message (query 1)	44
create_by_mental_act/message (query 3)	27
(unrestricted verb-noun combinations)	11,960

TABLE 2 Number of matches (concordance lines) for the queries. "unrestricted verb-noun combinations" means V-N combinations with the same restrictions on the tokens between the verb and the noun as in the queries above, but without any semantic restrictions.

7 Query results

This section lists, for each of the queries listed in section 6 above, some automatically and randomly selected concordance lines from the output produced by the query processor. The range of words matched by the corresponding query is surrounded by angle brackets. Thus, the opening angle bracket directly precedes the verb of the verb-noun combination and the closing angle bracket directly follows the noun.

[9] An intelligent query optimizer could apply such transformations automatically; however, such an optimizer is not yet built into the query system, so that query efficiency is mainly controlled by the way the queries are written.

7.1 Query 1 (utter/message)

(20 matches out of 44 total)

o this will continue to <place a disproportionate burden> '' on Fulton taxpa
_Rep._B._D._Pelham will <offer a resolution> Monday in the House to rescind
ecifically asked him to <offer the resolution> . The resolution , which Barb
that we can continue to <preach the gospel> in a form that makes it seem inc
is what we mean when we <say this demand> must be accepted without condition
central significance to <draw the details> into relevance and meaningfulness
ny photographic meteors <give evidence> of a fluffy , loosely bound meteorit
ght and the gegenschein <give some evidence> for such a dust blanket , a phe
se , enough sampling to <give a satisfactory description> of the vertical di
ion , we should like to <give an example> which illustrates some of the idea
 to me untrue . When we <repeat the remark> that such suffering was a bad th
ient was_not_allowed_to <move his body> in_any_way at_all , the following st
e found it difficult to <return the fire> . Noticing Russell 's horse in_fro
facts and attempting to <falsify history> . While the Daily_Post continued t
 it was the only way to <affirm his history> and identity in the torpid , be
feet so that they would <give substantive support> to Christ ' s body , to i
ingly . ' ' how can you <say such_a thing> ? There will be thousands who wil
im groping in the mud . <Say three minutes> to make it sporting . Still the
ith trembling hands and <read the formal report> in Hillman ' s beautiful ,
 . I made inquiries , I <read a book> of etiquette . In December I wrote her

7.2 Query 3 (create by mental act/message)

(20 matches out of 27 total)

ar as possible , we may <develop this argument> at greater length . We may s
nd is a major effort to <make a record> that sounds like a real orchestra ra
 feeling that one might <make some mistake> or that in every previous execut
central significance to <draw the details> into relevance and meaningfulness
, authorizes the SBA to <make a complete inventory> of the productive facili
ORITE FLUX # One cannot <make a very satisfactory guess> about the micromete
will find it helpful to <think_of the special case> when the primes are of d
ence . (For_instance , <see Example> 2 of Section 5 - 5 , on red cards in h
e of us might naturally <make such a statement> . We come_upon a rabbit that
er-by happened later to <discover the body> and feel repugnance or not . If
ome_upon the rabbit and <make our remark> about its suffering being a bad th
e roof until they could <see a body> on the bed covered by a blanket .
n be made , it will not <find support> or countenance from any of the men
ly () '' Let 's go and <see what it_'s> like '' . Annisberg was about seven
lace of Fontainebleau . <Imagine the honor> of it ! ' ' What was your answer
t his enemies could not <find his body> and have it dug_up and burned . The
 doing . But he did n't <figure Ma> for much . He did n't figure her at_all
Under the heading of it <'s an ill wind> , et_cetera '' . Backstage was tomb
on , Lincoln refused to <make any further statement> . '' I fear explanation
oceros '' , I said , '' <think_of the effect> it would have on an audience o

8 Access strategies

In a first approach, we used the (textual) output of the tool `wn` (which is part of the WORDNET distribution) to compute the value of the `wndist` dynamic attribute. The `wn` tool is normally called with a "start word" and one of the supported relations and returns a hierarchically structured list of words which are in the desired relation (or in its transitive closure). The information needed in our experiment (the "distance" of two words in a given relation) is thus not directly provided by WORDNET, but can be computed from the textual output of `wn`.

At the beginning of our experiments, we used various tools to filter the information textually returned by a call to `wn` in order to compute the distance. So, for each verb for which the distance to, for example, "utter" had to be computed, `wn` had to be called (which requires the loading of all WORDNET data files) and the output of that call had to be passed through a set of piped-together `sed`, `awk` and `grep` filters. This procedure turned out to be much too slow for extensive retrieval.[10] In order to improve evaluation efficiency, first a C function was written which directly accesses the WORDNET libraries to compute the distance of two words within a given relation.

The distance measuring function improved access efficiency a lot, since it was no longer necessary to analyse the textual output of `wn`. However, in order to further improve access efficiency, a WORDNET data server daemon was implemented in C in order to circumvent the necessity of loading WORDNET's data files on each value request. This daemon needs to be started only once (therefore, the data files only have to be loaded once) and waits for data requests which come in from the (local) network. When a data request is received, the server computes the requested information (in our case, the distance of two words within a given relation) and returns it to the calling process (given that the calling process is authorized to request information).[11] Together with

[10]The queries listed in section 6 were evaluated on a 230,000 word corpus with 11,960 verb-noun combinations which are potential candidates for the collocations in question. That is, `wn` and the whole set of filters had to be called 11,960 times (for each verb-noun cooccurrence) and additionally for those nouns where the information computed by consulting WORDNET did not filter out the verb, leading to more than 100,000 Unix processes.

[11]This model of information exchange is usually called a *client-server model*. The server process is the daemon which holds the data, and the calling process (the client) is an arbitrary process. Although in the present case, the client and the server run on the same local network, this need not be the case; in principle, the data request may come from any host connected to the Internet, thus adding "remote" information to a (local) corpus where the remotely accessed information probably couldn't be stored locally at all.

the distance measuring function, the queries are computed reasonably fast (about 20 times faster than in the first approach), although not with an efficiency which is suitable for interactive use.[12] However, this highly depends on the query: in many cases more restrictive query conditions can be expressed so that less requests of "external data" are necessary (the examples in section 4 are computed in about 15–20 seconds on a two-million word corpus).

It should be noted that in the example queries shown above, the *dynamic* computation of the semantic relationship between a word in the corpus and a certain "concept" can be avoided: one can first use WORDNET to compute the set of all words which are in a given relation to the concept (e.g. by running "`wn message -treen`" to compute all nouns which are hyponyms of "message") and then automatically compose a query where all these words are enumerated as a disjunction over word forms (possibly with the help of a morphological tool to either produce all inflectional forms of the words or to lemmatize the corpus beforehand):

```
[pos="V.*" & (word="utter"|word="emit"|word="speak"|...)]
[pos!="N.*|V.*|IN" & word!="\.|,|;|:"]{0,10}
[pos="N.*" & (word="message"|word="request"|word="demand"|...)]
within s;
```

Example 4

The queries produced by this approach are rather large, but can be evaluated by the query evaluator very efficiently.[13] However, the intention of this paper is to exemplify the general idea of "dynamic annotations", and not to show the most efficient way of how to use WORDNET information in corpus querying. Moreover, in some cases, such a pre-computation of disjunctions is impossible (for example when the conditions on a word would in some way depend on the word's context or on other dynamically computed information, or when the number of hyponyms is "unlimited", e.g. due to compounding in German).

9 Discussion

It is clear that the evaluation efficiency of corpus queries involving the consultation of an external thesaurus like WORDNET is still too bad for

[12] Each of the queries described in section 6 took about twenty minutes to process on the rather small corpus. Without the evaluation of the WORDNET information (that is, the computation of all verb-noun sequences with at most 10 (restricted) words in between, within one sentence), the query takes about 15 seconds and yields 11,960 matches.

[13] This approach is taken in another component of the IMS Corpus Toolbox, namely the Macro Processor (MP) developed by Bruno M. Schulze within the DECIDE (MLAP 93-19) project (see Schulze 1996).

interactive querying, even after implementing the techniques described in section 8. But the additional constraints which can be expressed by using WORDNET information lead to small concordance lists, which — in the case of our 230,000 word corpus — are then small enough for manual postprocessing. In a batch processing mode, the same queries could be evaluated on a larger corpus, leading to concordance lists which are then large enough for performing statistical tests, if necessary. Efficiency could be improved by additional restrictions on the queries by using part-of-speech values or other (non-dynamic) corpus attributes. The lexicographical relevance of the use of additional knowledge sources in corpus querying lies in the reduction of the material to be browsed manually and in the possibility to express queries more precisely, thus leading to more relevant concordances.

The method presented in this paper increases the *precision* of the query result when compared with unrestricted V-N combinations.[14] As illustrated above, this goal is actually achieved, since the proportion of relevant items in the unfiltered 11,960 V-N combinations is very small, whereas it is considerably higher in the concordances which are produced by additionally using WORDNET information for filtering out irrelevant matches.

Precision can be further increased by using additional tools or information sources to determine the WORDNET sense for each word in the corpus: the `wndist` function computes the "distance" between a word and a given concept for *any* of the word's senses, not only for the sense in which the word is actually used.

By using an external information source, recall may decrease when compared with unfiltered V-N combinations (mainly due to limitations of the underlying thesaurus): not all words which occur in the corpus have an entry, and not all the words which have an entry are fully and correctly classified. The retrieval result is at most as good as the underlying information source(s).

Without an underlying thesaurus, a linguist would of course not search for unrestricted V-N combinations and filter out irrelevant ones manually. Rather, s/he would think of a number of e.g. "uttering" words and a number of "message" words and express a query for combinations of those. However, this approach is cumbersome, and probably would miss several relevant word forms. When compared with this method,

[14] "Precision" and "recall" are used in the standard information retrieval sense here, where precision is defined as the proportion of all retrieved items that are actually relevant, and recall is defined as the proportion of all relevant items in the corpus that are actually retrieved.

the approach described in this paper increases recall, since more relevant items are actually retrieved.

The use of WORDNET as an external, additional knowledge source for the corpus query processor makes it possible to express semantic constraints for concordances and to use the knowledge provided by WORDNET (or other on-line thesauri) in corpus queries. As shown in this paper, it is relatively easy to express queries which show some of the semantic properties of typical verb arguments or to retrieve collocation candidates based on semantic information.

WORDNET is only one example how a lexical database can be reused for corpus querying. Through the modular and general mechanism in which knowledge sources can be attached to our query system, very different kinds of (linguistic) resources can at least partially be reused in corpus exploration (provided that the resource is able to provide the information by using the data types which are currently supported by the query system) without the need to link access knowledge into the query system proper, although special care has to be taken to make the data interchange efficient.

By (re-)using external knowledge sources in corpus querying, it is possible to follow a bootstrapping strategy for the creation or augmentation of a corpus-based lexical knowledge base: knowledge gained from corpus evidence (possibly through the use of other knowledge sources, such as WORDNET) can be recursively used in subsequent queries by linking the resource under construction to the query system and a corpus, thus enabling one to incrementally increase the quality or the coverage of the lexical knowledge source.

Although in the present case the implementation of a client-server model was mainly motivated by efficiency considerations, it should be noted that such models have a number of important advantages (however, a thorough discussion of advantages and disadvantages is beyond the scope of this paper). Using client-server models for the network-based interchange of linguistic information makes it possible to use information which is not physically available at the client's computer. Client-server models can therefore be used as a basis for distributed, shareable linguistic resources.

References

Christ, O. 1994. A modular and flexible architecture for an integrated corpus query system. In *Proceedings of COMPLEX'94: 3rd Conference on Computational Lexicography and Text Research (Budapest, July 7–10 1994)*. Budapest, Hungary. CMP-LG archive id 9408005.

Marcus, M. P., B. Santorini, and M. A. Marcinkiewicz. 1993. Building a

large annotated corpus of English: the Penn Treebank. *Computational Linguistics* 19(2):313–330.

Miller, G. A., R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. 1993a. Introduction to WordNet: An On-line Lexical Database. In *Miller et al.: Five papers on WordNet,* Miller et al. 1993b.

Miller, G. A., R. Beckwith, C. Fellbaum, D. Gross, K. Miller, and R. Tengi. 1993b. Five Papers on WordNet. Technical Report CSL Report 43 (revised March 1993). Cognitive Science Laboratory, Princeton University, March.

Schulze, B. M. 1996. Designing and Evaluating Extraction Tools for Collocations in Dictionaries and Corpora: MP User's Guide. Technical Report, IMS-CL (internal draft).

13

Multilingual Data Processing in the CELLAR Environment

GARY F. SIMONS & JOHN V. THOMSON

This paper describes a database system that has been developed by the Summer Institute of Linguistics to be truly multilingual. It is named CELLAR—Computing Environment for Linguistics, Literary, and Anthropological Research. After elaborating some of the key problems of multilingual computing (section 1), the paper gives a general introduction to the CELLAR system (section 2). CELLAR's approach to multilingualism is then described in terms of six facets of multilingual computing (section 3). The remainder of the paper (sections 4 and 5) describes details of how CELLAR supports multilingual data processing by presenting the conceptual models for the on-line definitions of multilingual resources.

1 The Problem of Multilingual Computing

Ours is a multilingual world and the promise of universal computing cannot be realized until we have an adequate solution to the problem of multilingualism in the computing environment. In the same way that

We are indebted to many for making the research reported herein possible. We have collaborated (Simons as project director and Thomson as lead programmer) in the design of CELLAR ever since the project began in 1989. The following team of Smalltalk programmers (listed with length of service) have worked to implement these designs: Larry Waswick (5 years), Sharon Correll (4 years), Marc Rettig (2 years), Jim Heimbach (2 years), Nathan Miles (1 year), and David Harrison (6 months). Three other colleagues—Jonathan Kew, Randy Regnier, and John Wimbish—have contributed materially to the development of our conceptual model for multilingual computing as they have attempted to use CELLAR to build applications. The project has been funded in large part through a grant from Wycliffe Bible Translators (Huntington Beach, Calif.).

Linguistic Databases.
John Nerbonne, ed.

people can be multilingual, our computing systems need to be multilingual. They need the potential to work in any language and to work simultaneously with many languages at the same time.

Many computerists have conceived of the multilingual computing problem as a problem of *special characters*. This approach considers the multilingualism problem to be solved when all the characters needed for writing the languages of interest can be both displayed on the screen and printed. In the early days of computing, when character generation was hard-wired into video terminals and molded directly into print chains and type fingers, this was impossible. Now that graphical user interfaces are the norm, this is not a problem; font-building tools can be used to create customized fonts that hold the needed character shapes.

The writing system is the most visible aspect of language data; that is probably why special characters and fonts have been such a focus of solutions to multilingual computing. But now that graphical user interfaces have made arbitrary fonts commonplace, it is clear that there is much more to multilingual computing than displaying the data. Different languages also have different conventions for many aspects of data processing, such as sorting, keyboarding, spell checking, hyphenation, finding word and sentence breaks, and the like. A truly multilingual computing environment would handle all of these processes in a manner appropriate to the language of the data.

The major software vendors have recognized this problem and, in an effort to find a wider market for their products, have produced versions of their products that are customized for languages other than the one supported by the original version. This is a process generally referred to by the industry as *internationalization* of software. This process confronts many of the problems of multilingual computing, but does not provide a solution. The results are still monolingual (or sometimes bilingual) products which are just as limited as the original monolingual program. But in the same way that people are multilingual, our computing systems need to be multilingual.

We need computers, operating systems, and programs that can potentially work in any language and can simultaneously work with many languages at the same time. A most promising recent development along these lines is the incorporation of the World Script component into version 7.1 of the Macintosh operating system (Ford and Guglielmo 1992). In the late 80s, Apple developed an extension to their font manager called the script manager (Apple 1988). It handles particularly difficult font problems like the huge character inventory of Japanese and the context-sensitive rendering of consonant shapes in Arabic. A script system, in conjunction with a package of "international utilities," is able to

handle just about all the language-dependent aspects of data processing mentioned above (Davis 1987). The original script manager's greatest failing was that only one non-Roman script system could be installed in the operating system. World Script has changed this. It is now possible to install as many script systems as one needs.

World Script begins to make the resources needed for truly multilingual computing available in the operating system, but these resources need to find their way into application software. At this point, programming a new script system or developing an application that takes advantage of the script manager is a low-level job for a skilled system programmer, and progress in this area is slow. What is more, the application environment adds some higher-level requirements for multilingual computing, such as the need for the user to create and manipulate multiple language versions of the same datum, and to have multiple language versions of software interfaces. These additional requirements are developed in section 3 on "Facets of Multilingual Computing."

The Summer Institute of Linguistics (through its Academic Computing Department) has embarked on a project to build a computing environment that is truly multilingual. The environment is called CELLAR—for Computing Environment for Linguistic, Literary, and Anthropological Research. It is, at the heart, an object-oriented database management system. It includes a high-level programming language that is designed to make it relatively easy to build application programs that are truly multilingual. Before discussing CELLAR's approach to multilingual computing, we begin by giving an introduction to CELLAR as a database system. This is necessary because the multilingual resources of CELLAR are modeled as objects in the database. Thus the definitions of those resources in sections 3 and beyond depend on an understanding of CELLAR's basic object model.

2 An Introduction to CELLAR

This introduction to CELLAR proceeds in three parts. Section 2.1 describes the major requirements for a linguistic computing environment which motivated the development of CELLAR. Section 2.2 gives an overview of the major components of the system. Section 2.3 goes into greater depth on the component that will be used throughout the paper to define the formal model of multilingual resources, namely, the conceptual modeling language.

2.1 The Requirements of a Computing Environment for Linguistic Research

At the heart of a computing environment for linguistic research must lie a database of linguistic information. The requirements for such a database system must begin with the nature of the data it seeks to store and manipulate. The following are five of the most fundamental features of the data we work with (both the primary textual data and our analyses of them) and the demands they put on the database:

1. The data are multilingual; the database must therefore be able to keep track of what language each datum is in and process each in a manner appropriate to its language.
2. The data in a text unfold sequentially; the database must therefore be able to represent text in proper sequence.
3. The data are hierarchically structured; the database must therefore be able to express hierarchical structures of arbitrary depth.
4. The data elements bear information in many simultaneous dimensions; the database must therefore be able to vest data objects with many simultaneous attributes.
5. The data are highly interrelated; the database must therefore be able to encode associative links between related pieces of data.

It is possible to find software systems that meet some of these requirements for data modeling, but we are not aware of any that meet them all. Some word processors (like Nota Bene, Multilingual Scholar, and those that use the Macintosh World Script system) deal well with multilingualism (point 1). All word processors deal adequately with sequence (point 2). Some word processors handle arbitrary hierarchy well (point 3), but most do not. The areas of multidimensional data elements and associative linking (points 4 and 5) do not even fall within the purview of word processors. This is where database management systems excel, but they typically do not support multilingualism, sequence, and hierarchy adequately.

The academic community has begun to recognize that SGML (ISO 1986) can be used to model linguistic data; indeed, it is possible to devise data markup schemes that meet all of the above requirements. The Text Encoding Initiative (TEI) guidelines (Sperberg-McQueen and Burnard 1994) give many examples. But SGML-based representations of information are too abstract and too cumbersome for the average researcher to work with directly. This suggests a sixth fundamental requirement for a computing environment that will meet the needs of the linguistic researcher:

6. Users need to be able to manipulate data with tools that present conventionally formatted displays of the information; the database must therefore be supported by a complete computing environment that can provide user-friendly tools for creating, viewing, and manipulating information.

Until this requirement is met, it will be difficult for the community of linguistic researchers to realize the promise of the kind of data interchange that SGML and the the TEI guidelines make possible. See Simons (1997) for a fuller discussion of the role of visual models in a database system.

2.2 A Computing Environment for Linguistic Research

CELLAR is a computing environment that is being built expressly to meet the above requirements. See Simons (in press) for a fuller discussion of these requirements and how CELLAR seeks to meet them. At the heart of CELLAR is an object-oriented database and programming system; see Rettig, Simons, and Thomson (1993) for an informal description of its features.

To build an application in CELLAR, one does not write a program in the conventional sense of a structure of imperative commands. Rather, one builds a declarative model of the problem domain. A complete *domain model* contains the following four components:

- *Conceptual model.* Declares all the object classes in the problem domain and their attributes, including integrity constraints on attributes that store values and built-in queries to implement those that compute their values on-the-fly.
- *Visual model.* Declares one or more ways in which objects of each class can be formatted for display to the user.
- *Encoding model.* Declares one or more ways in which objects of each class can be encoded in plain text files so that users can import data from external sources.
- *Manipulation model.* Declares one or more tools which translate the interactive gestures of the user into direct manipulation of objects in the knowledge base.

Because CELLAR is an object-oriented system, every object encapsulates all the knowledge and behavior associated with its class. Thus, any object can answer questions, whether from the programmer or the end user, like: "What queries can you answer about yourself?" "What ways do you have of displaying yourself?" "What text encoding formats do you support?" "What tools do you offer for manipulating yourself?" For programmers this greatly enhances the reusability of previously coded

classes. For end users this greatly enhances the ability to explore all the capabilities of an application without having to read manuals.

2.3 The Conceptual Modeling Language

Linguists have been using computer systems for decades to model the things in the "real world" which they study. A conceptual model is a formal model in which every entity being modeled in the real world has a transparent and one-to-one correspondence to an object in the model. Relational databases do not have this property; they spread the information about a single entity across multiple tables of a normalized database. Nor do conventional programming languages; even though the *records* of Pascal and the *structures* of C offer a means of storing all the state information for a real world entity in a single data storage object, other aspects of the entity like its behavior and constraints on its relationships to other objects are spread throughout the program.

A conceptual modeling language, like an object-oriented language, encapsulates all of the information about a real world entity (including its behavior) in the object itself. A conceptual modeling language goes beyond a standard object-oriented language (like Smalltalk) by replacing the simple instance variables with attributes that encapsulate integrity constraints and the semantics of relationships to other objects. Because conceptual modeling languages map directly from entities in the real world to objects in the computer-based model, they make it easier to design and implement systems. The resulting systems are easier to use since they are semantically transparent to users who already know the problem domain. See Borgida (1985) for a survey of conceptual modeling languages and a fuller discussion of their features.

A database in CELLAR is defined by a conceptual model. Both when designing conceptual models and when presenting finished ones, we use a graphical notation. When entering the model into CELLAR, we use a formal language. We first present the graphical notation along with an actual example of a conceptual model for a bilingual dictionary. As a conclusion to this section, the syntactic form of the conceptual modeling language is illustrated with the same example.

A key to the graphical notation for conceptual modeling is given in figure 1. The rectangles represent classes of objects. The class name is given at the top of the rectangle; the attributes are listed inside. When the rectangle is dashed, the class is *abstract.* This means that there are no instances of this class; rather, the class is a higher-level node in an is-kind-of hierarchy. All of its subclasses inherit all of its attributes. When the rectangle is solid, the class is *concrete* and it does have instances. Vertical lines link superclasses on top with their subclasses below. The

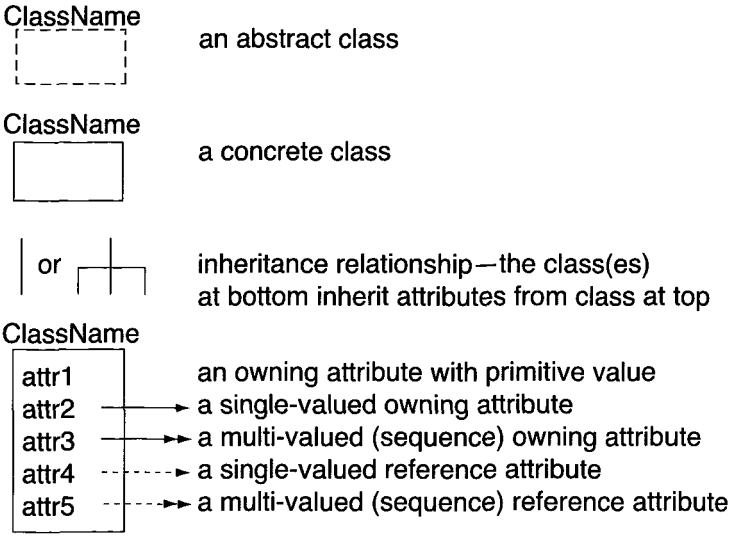


FIGURE 1 Key to conceptual modeling notation

subclass *inherits* all the attributes of the superclass. Thus its full list of attributes consists of the attributes listed for its superclass plus its own attributes.

The *attributes* of a class are listed inside the rectangle. When nothing follows the name of an attribute, its value is a simple string, number, or boolean. Arrows indicate that the attribute value is an instance of another class. A single-headed arrow means that there is only one value; a double-headed arrow indicates that a sequence of values is expected. Solid arrows represent *owning* attributes; these model the part-whole hierarchy of larger objects composed of smaller objects. Dotted arrows represent *reference* attributes; these model the network of relationships that hold between objects. In a CELLAR knowledge base, every object is owned by exactly one object of which it is a part, but it may be referred to by an arbitrary number of other objects to which it is related.

Figure 2 shows a sample conceptual model for a basic bilingual dictionary. The top-level class in the model is BilingualDictionary. It has attributes identifying the two languages, the compiler, and the categories that are allowed for parts of speech and other kinds of information that

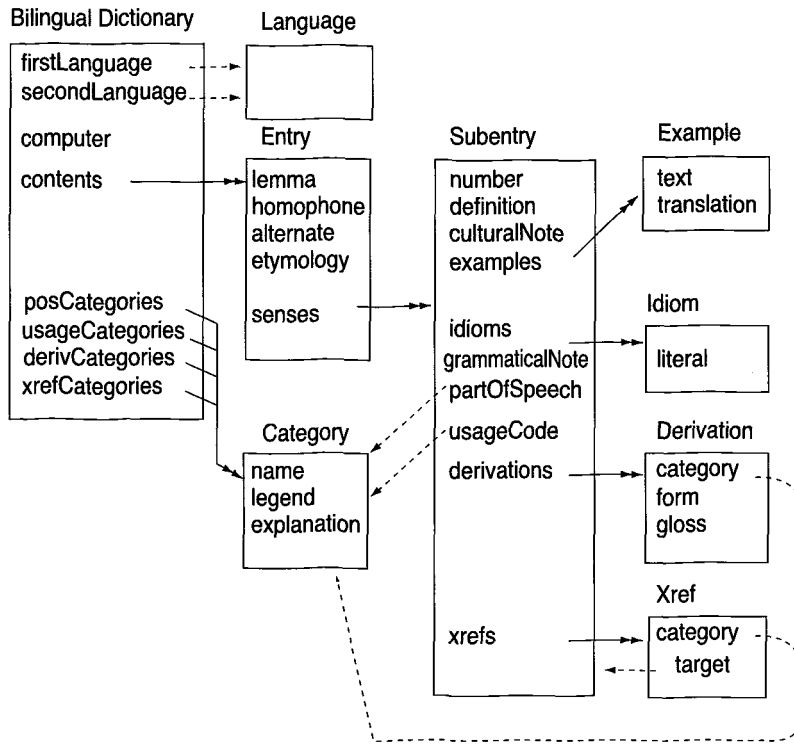


FIGURE 2 Sample conceptual model of a basic bilingual dictionary

draw from closed sets of values. The main contents of a BilingualDictionary are a sequence of Entries, one for each lexical item. The Entry has attributes that give the form of the lemma, alternate forms, and the etymology, but the bulk of the information is in a sequence of Subentries, one for each sense of meaning for the word. A Subentry gives grammatical information as well as definition and cultural notes. Furthermore, it may contain Examples, Idioms, morphological Derivations, and Xrefs (semantic cross-references), each of which is modeled as a complex object with further attributes. Idiom is a subclass of Example; it adds a literal translation to the text and free translation that an Example has. The target of a cross-reference (Xref) points to the semantically related Subentry.

For an illustration of the conceptual modeling language, consider the following definition of the class Subentry and some of its attributes:

```
class Subentry has
    definition : String
    examples   : sequence of Example
    partOfSpeech : refers to Category in
      posCategories of my owning(!BilingualDictionary)
    number     : Integer means
      positionIn(''senses'', self) of my owner
```

This defines the class Subentry to have four attributes. An attribute definition consists minimally of the attribute name and its signature, separated by colon. The signature declares what class of objects are allowed to be attribute values (if the attribute is being set) or are returned as values (if the attribute is being queried). The phrase *sequence of* (or simply *seq of*) indicates that the attribute takes a sequence of values.

The phrase *refers to* indicates that the attribute stores a reference to an object that is part of some other object. The *in* phrase which follows gives a query that retrieves all the possible target objects in the system. In this case the query is *posCategories of my owning(!BilingualDictionary)*. *X of* Y means, "Return all the objects generated by getting the X attribute of each of the objects generated by the query Y." *Self* is a query that returns the object itself; *my X* is a shorthand for *X of self*. *My owner* returns the Entry of which the Subentry is a part; *my owning(!BilingualDictionary)* goes up the ownership chain until it finds the BilingualDictionary of which the Subentry is a part. The possible values for the *partOfSpeech* of a Subentry are thus any of the Category objects that are stored in the *posCategories* attribute of the BilingualDictionary.

The final attribute in the example is a virtual attribute. It is so called because the value is not really there in the object. It is calculated on-the-fly by executing a query. The *means* keyword indicates that the attribute is virtual. The expression that follows is the query to be executed when the object is asked for the value of the virtual attribute. In this case, the *number* of the sense of meaning is computed by asking the Entry of which this is a sense (that is, *my owner*) to tell us the position in its *senses* attribute of the current Subentry (that is, *self*).

3 Facets of Multilingual Computing

In our multilingual world, a person may use many languages during the course of a day. When people are multilingual, they do not isolate their use of one language from another; rather, they may mix the languages in a single utterance, or use one language to explain the meaning of an utterance in another. In order to use a computer effectively, these

same people need systems that can cope in a similar manner with all the languages they use. For a computing system to be truly multilingual, it needs to cope with multiple languages in the following ways:

- It must be possible for different bits of text to be in different languages.
- It must be possible for a string of text in one language to embed spans of text in other languages.
- It must be possible for the system to make valid assumptions about what language a string is in without requiring a declaration from the user.
- It must be possible for a string of text in one language to be associated with equivalent expressions (that is, translations) in other languages.
- It must be possible for users to control which version they prefer to see when equivalents in many languages are present.
- It must be possible for the interface to a computer system to change in response to the user's language needs and preferences.

The next five subsections discuss these requirements of a multilingual computing system in greater depth and describe how CELLAR meets them.

3.1 System as a Repository of Multilingual Resources

The entry point for CELLAR's model of multilingual computing is the string as an individual data item. String is a fundamental data type in virtually every programming language and database system, and in every system known to us, String is modeled as just a sequence of numerical character codes. But consider the following two propositions which we take to be axiomatic:

1. Any bit of textual data stored in a computer is expressed in some language (whether it be natural or artificial).
2. Text-related information processing functions (like keyboarding, spell checking, sorting, hyphenation, finding word boundaries, and so on) are language dependent.

For instance, the conventional keyboard layout on English typewriters has QWERTY in the upper left, while the conventional layout for French has AZERTY. The string *noch* would be a misspelled word in English, but is fine in German. In English *ll* sorts between *lk* and *lm*, but in Spanish it sorts between *lz* and *ma*.

These points lead to the following two conclusions:

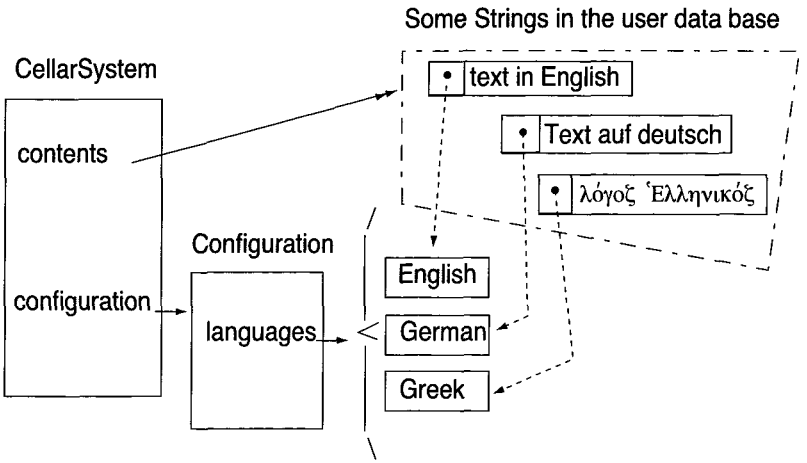


FIGURE 3 The system as a repository of multilingual resources

3. A computer system cannot correctly process a bit of textual data unless it knows what language it is in.
4. A string, as the fundamental unit of textual data, is not fully specified unless it identifies the language it is in.

CELLAR's model of multilingual computing is based on these two points. Figure 3 illustrates the basic model. Every string of textual data in the user database contains not only its characters but also a tag for the language it is in. These tags take the form of pointers to language definitions that are stored in the system's configuration. These language definitions are objects that declare everything the system knows about how to process data in that language. (The details of these language definitions are explored below in sections 4 and 5.) Thus the computing system is a repository of multilingual resources in two respects: each piece of text is an information resource in some (identified) language, and the definitions of the languages and their processing behavior are key information resources.

3.2 String as a Multilingual Object

It is not enough that the system as a whole can cope with multiple languages. So must an individual string, since:

5. A string of text in one language may include spans of text in another language, and so on recursively.

For instance, a paragraph in English might quote a sentence in German which discusses a Greek word. Such embedding of language within lan-

guage is a fundamental property of the way humans use language and must be reflected in our computing systems.

CELLAR's conceptual model of String as a data object is therefore a recursive one, in that the value of a string can contain another string. And each of these strings has a built-in *language* attribute to store the pointer to the language it is in. The recursion stops with "primitive strings." These are sequences of character codes with no explicit pointer to language; a primitive string is in the language of the String it is a part of. The value of a string can then be a sequence of a mixture of any number of primitive strings and strings. Thus,

```
class String has
    language : refers to Language
    value    : sequence of PrimitiveString or String

class PrimitiveString has
    value    : sequence of CodePoint
```

Note that PrimitiveString is actually a secret of the implementation; the CELLAR user can see and create only Strings. But the internal structure of strings is accessible through two built-in attributes of String:

isMonolingual Returns *true* if the String contains only a PrimitiveString; *false* if it contains at least one embedded String.

substrings Returns the sequence of PrimitiveStrings and Strings at the top level of the String's internal structure. The PrimitiveStrings are returned as Strings with the language explicitly set.

In a typical computer application, mixing of languages is handled by font switches in a flat string. But this flat model is not adequate because it fails to encode the logical structure of the string. Consider the two multilingual strings diagrammed in figure 4. The first might represent a sentence in English which embeds two phrases from Arabic. The second could represent an English sentence that quotes an Arabic sentence that contains an English word. Both multilingual strings, if they are flattened, involve an alternating sequence and English and Arabic primitive strings, but the difference in logical structures is encoded by the different patterns of recursive embedding.

It is essential to encode logical structure explicitly in order to get the correct display of these strings on the screen. In the first case, the left-to-right order of English dominates the whole display with the characters of the embedded Arabic strings being displayed in right-to-left order (as indicated by the reversed order of characters in *2sp* and *4sp*). In the second case, the right-to-left order of Arabic dominates the whole

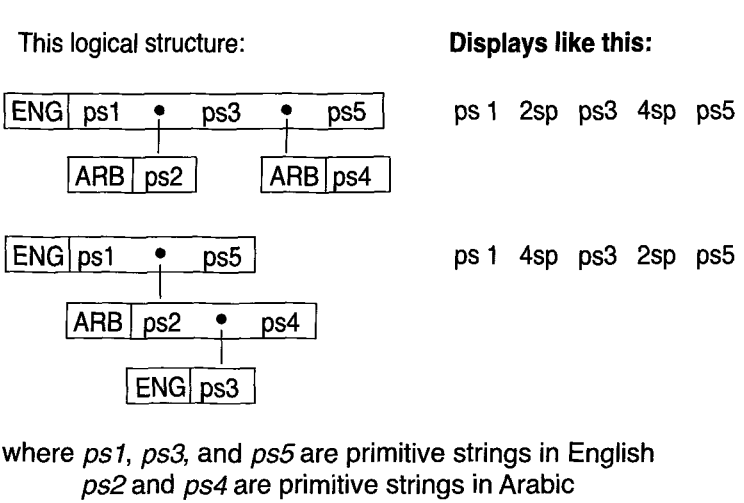


FIGURE 4 Necessity for embedding in multilingual strings

embedded string, so that primitive strings 2 through 4 are displayed in right-to-left order. As a result, the relative positions of strings 2 and 4 are flipped in the two displays. (See Knuth and MacKay 1987 for a complete discussion of text formatting of multilingual documents with mixed direction.) Without support of arbitrary embedding of languages within multilingual strings it is impossible to guarantee that the correct display can be generated when the layout must be changed, such as to accommodate a different column width.

3.3 Conceptual Model as a Source of Multilingual Assumptions

We have proposed that every string in the database should be tagged for what language it is in. To ask the user to explicitly tag each string would be an onerous task; it would not seem at all necessary since:

6. The language of a string of text is largely predictable from its context in the database.

For instance, once we know that a complete text is in the Latin language, we can assume that all of its component parts are also in Latin unless we are told otherwise. Once we know that we have a bilingual dictionary of German to French, we can assume that any headword we see is in German and any definition is in French. In the same way, a database implementation of a bilingual dictionary should be able to assume that text entered into the headword field is in the first language and that

text entered into the definition field is in the second language. When the two languages involved use completely different writing systems the necessity for automatic switching of language based on context is even more apparent.

CELLAR supports this requirement of multilingual computing by allowing the attribute definitions in a conceptual model to take a *default language* declaration. Indeed, the expectations as to what parts of a data object are in what language are an integral part of its conceptual model. For instance, the bilingual dictionary of section 2.3 (see figure 2) already has attributes at its top-level to record the *firstLanguage* and the *secondLanguage* of the dictionary. The following attribute definitions are then used to tell the system that definitions are assumed to be in the second language, and that the text of an example is in the first language while the translation is in the second:

```
class Subentry has
    definition  : String default language is
                    secondLanguage of my dictionary
    examples    : sequence of Example

class Example has
    text        : String default language is
                    firstLanguage of my dictionary
    translation : String default language is
                    secondLanguage of my dictionary
```

Note that the identity of the default language is retrieved by a query. It would be possible to express these as literal references to particular languages, but then the conceptual model would not be reusable for dictionaries of other languages. The queries used here assume that a virtual attribute named *dictionary* is implemented; it goes up the ownership hierarchy to find the top-level BilingualDictionary object. It has a very simple recursive definition: on BilingualDictionary it is defined to mean *self*, and on all other objects in the conceptual model it is defined to mean *dictionary of my owner.*

3.4 Attribute as a Multilingual Alternation

We have already seen that strings may be multilingual by virtue of embedding text in multiple languages. Another dimension of multilingualism for strings is that:

7. A string of text in one language may have equivalent ways of being expressed (that is, translated) in other languages.

For instance, consider the following situations:

- A programmer writing an application for a multilingual audience wants all the strings of text displayed by the program to have versions in every language of the target audience.
- A linguist writing a dictionary of a vernacular language spoken in an area that has a number of official languages as well as other languages of wider communication wants the definitions and other explanatory notes to be recorded in each of these languages.
- A linguist analyzing a text wants to record the words of the text in three alternate orthographies: a phonemic orthography, a traditional orthography, and a proposed new practical orthography. (In our model these are alternate encodings of the string; see section 5.3.)

Note the fundamental difference between these needs and those discussed in the previous section. When a string embeds strings in other languages, we are dealing with a single author's and reader's ability to handle data in multiple languages; the data in different languages are typically meant to be read consecutively. When we deal with multiple translations or encodings of the same data, we are dealing with the possibility that different readers have different abilities to cope with particular languages, and that different authors may be able to write in different languages. A typical reader will use one version or another, not all of them, although sometimes comprehension might be aided by using more than one.

The need to store multilingual alternatives is so fundamental that it is not confined to strings; it could occur at virtually any level in the conceptual model.

8. A constructed object expressing information in one language may require alternate constructions to express equivalent information in other languages.

For instance, if a text is modeled as a hierarchical structure of objects rather than being one long string, then translating a paragraph object that is made up of sentence objects may require that a whole new paragraph object be built so that the translated sentences can be reordered into a more appropriate order for the target language.

Due to point (8), we treat the problem of multilingual alternatives as not being confined to the contents of a string, but as being a matter of alternations among related objects. We treat the attribute as the point of alternation; that is, an attribute, even though it is declared to have a single value, is allowed to store language-dependent alternatives for that single value. But not every attribute should allow an alternation; it is a decision the designer of a conceptual model should make. Thus,

CELLAR provides this capability through the use of the *multilingual* keyword in the signature of an attribute definition. For example, to turn the bilingual dictionary of section 2.3 into a multilingual dictionary in which we did not have to define separate attributes for all the possible defining languages, we would give definitions like the following:

```
class Subentry has
    definition  : multilingual String default language
                    is secondLanguage of my dictionary
    examples    : sequence of Example

class Example has
    text        : String default language
                    is firstLanguage of my dictionary
    translation : multilingual String default language
                    is secondLanguage of my dictionary
```

Note that in Examples, the *text* is still confined to a single language, but that the *translation* could be in many different languages. Note, too, that multilingual alternations do not interfere with the notion of default language. Unless the attribute is told otherwise (such as by the explicit language tagging of the object), an object is assumed to be the alternative for the default language.

3.5 User as a Controller of Multilingual Preferences

Once attributes store multilingual alternatives for the data, it becomes important to determine which alternative to show at any given time. This is not something the conceptual model can determine, for:

9. Different users of a computer system have different abilities to read and use various languages.
10. A given user may need to present the same information to different audiences who have different abilities to read and use various languages.

For instance, one user of a program may prefer to read its instructions in French while another prefers to read them in English. Or, a linguist may want to publish annotated texts of a vernacular language with Spanish glosses for one audience, but English glosses for another. Thus the fundamental control over which language alternatives should be displayed rests with the user.

CELLAR supports this at the system level by maintaining the user's list of *preferredLanguages* in the Configuration (see figure 5). This is a list of the languages which the user has some competence to read or use. The languages must be chosen from among those that are defined in the

system configuration, and they are ranked in order of the user's preference. For example, an English speaker with a good working knowledge of German and a smattering of French might list those three languages in that order. A native speaker of Swahili who understands French well and English less well might list those three.

When the user interface asks an attribute for its value and multiple alternatives are present, the system consults the preferred languages list and returns the first available alternative from the user's preference list. For example, if the preferences list is Swahili, French, English, then if a Swahili alternative is present the system shows that. If not, it tries for a French alternative. Failing that, it tries English, and failing that, it picks one of the available alternatives arbitrarily. A programmer may want to control the selection of alternatives more precisely in a particular instance. For these purposes, CELLAR provides a refinement to the normal *of* operator for getting the value of an attribute. For instance, whereas *X of Y* is the normal syntax for asking object Y to return its value for attribute X, the query *choose L for X of Y* specifically requests the alternative stored in that attribute for language L. Given that *L* is a single language and *LL* is an ordered list of languages, the *choose ... for* construction has the following variant forms:

choose L for Return the alternative for language *L*.

choose LL for Return the first alternative from the given list.

choose all LL for Return all alternatives from the given list in the order listed.

choose L ... for Return the alternative for language *L* if present; otherwise, the first alternative according to system preferences.

choose LL ... for Return the first alternative from the list *LL*, or if none present, the first alternative according to system preferences.

choose all ... for Return all the alternatives that are present in system preference order.

choose all LL ... for As above, but put the explicitly listed alternatives at the front of the result.

L and *LL* above may give literal language identifiers, in which case the programmer has full control. Or, they may be queries that retrieve preferences entered by the user, in which case the user has application-specific control. By using the variants above that return multiple values, the programmer can build user interfaces that display many values for an attribute, not just one. This would be desirable, for instance, for the scenario developed in the preceding subsection where a single dictionary includes definitions in many languages.

3.6 Interface as a Multilingual Construction

The selection of multilingual alternatives does not apply only to the treatment of data; a multilingual computing environment also needs to be able to provide tools for *manipulating* the data that communicate to different users in the languages they understand best. There are many aspects to this. For example,

- Menu item and button label text may need translation.
- Explanatory text in a dialog may need translation.
- Error messages may need translation.

A typical approach (such as is followed on the Macintosh) is to put all translatable text in a resource file. However, such resource files are only designed to deal with one language at a time. To make a version for another language it is necessary to copy the resource file and translate every text string. The result is a set of monolingual tools.

CELLAR's approach is to provide truly multilingual tools by using multilingual attributes in the classes that are used to build tools. For example,

- The text of menu items and button labels are multilingual attributes.
- Literal strings placed in windows may be multilingual.
- Exception (error, warning, information) objects use multilingual attributes for their brief message and for their extended explanation.

Simply by manipulating the *preferredLanguages* list in the system configuration, the user can interactively switch the language of the interface. This design means that the user will see a mixed language application when only some of the strings are available in the first preference language. For instance, the brief error messages might be translated into a number of regional languages, while the extended explanations that give more detail about them are available in only a few international languages. CELLAR's preference system insures that users will see the available translation that best meets their needs.

But this only solves part of the problem. Building multilingual user interfaces is more than just translating bits of text, because:

11. Different languages (due to differences in word order, direction of writing, length of words, and the like) may require different layouts to display equivalent information.

Again, the standard solution is to put layout designs in resource files and to copy them and rearrange them as needed for each new language version.

CELLAR solves this problem, too, in a manner that supports development of a single multilingual application. User interfaces are constructed by nesting a structure of declarative templates, each of which specifies a graphic layout for a set of objects. There are templates to lay out horizontal rows, vertical piles, wrapping paragraphs, bordered frames, editable text fields, check boxes, radio buttons, and so on. The problem of alternative layouts for multilingual interfaces is solved by adding another subclass of template: a multilingual template. As a CELLAR object, it has a simple definition:

```
class MultilingualTemplate based on Template has
    template : multilingual Template
```

In other words, MultilingualTemplate is a subclass of Template. Thus, anywhere it is possible to put a template of any kind, it is also possible to put a MultilingualTemplate. The latter has a single multilingual attribute that stores a collection of alternative templates, each of which lays out that substructure of the interface for a different language.

This approach has several advantages over the resource file approach.

- The preference mechanism allows partial translation to work much more easily. For example, a complex system originally created in English might be partly translated into French, and the most frequently-used parts further translated into a local minority language. The preference mechanism allows the minority language speaker to take advantage of whatever is in that language, then whatever is in French, and then see the rest in English.
- It is not necessary to have a separate program to edit the resources. All the translations are present in the system at once, and can be edited like any other multilingual attributes.
- More information is shared. It is not necessary to make a copy for each language of layouts that do not need to vary.
- Changes can be more incremental. If a different layout is needed for just one part of a dialog, the information about the rest of it can be common.
- All the alternatives can be present together. This makes it much easier to switch the system from one language to another for a user who did not fully understand the explanation in one language and wants to consult a second, or for different users of the same system. All that need be changed is the list of preferred languages.

- The original creator of a tool can give less attention to what will need to be translated, because the language for building user interfaces allows multilingual templates to be inserted at whatever level a cross-language difference is later found.
- CELLAR's window design language allows components of a display to be positioned on-the-fly relative to the space they and their neighboring components take up (as opposed, for instance, to the Macintosh resource-editing approach in which window layout is fixed before run time). This greatly reduces the need to build different layouts when the differences between two languages are just a matter of string lengths.

4 Defining Multilingual Behavior as System Resources

The key to CELLAR's ability to function as a multilingual computing environment is that every bit of textual data is explicitly linked to an object that defines its language-specific behavior. The remaining sections of the paper describe these definitions in detail. First, however, it is useful to see an overview.

Figure 5 shows the conceptual model of a CELLAR system and the collection of resources that define its multilingual behavior. The CELLAR system has a number of attributes including:

name Stores a name to distinguish this CELLAR system from others.

contents Contains the knowledge base of user data objects (including thousands of strings).

domainModels Stores domain models (see section 2.2) for all the built-in system classes as well as for user applications.

clipboard Holds the contents of the clipboard for copying, cutting, and pasting CELLAR objects.

integrityAgenda Stores a queue of pending object integrity checks and known integrity violations.

configuration Stores a Configuration object which holds the definitions of all multilingual resources installed in the CELLAR system.

The system configuration stores the following resources:

languages Defines every language known to the system. Any language can be encoded in more than one way. Thus, each Language definition stores a set of LanguageEncodings. See section 5.3 for a detailed discussion.

preferredLanguages The languages the user prefers to see when multiple alternatives are available. See section 3.5.

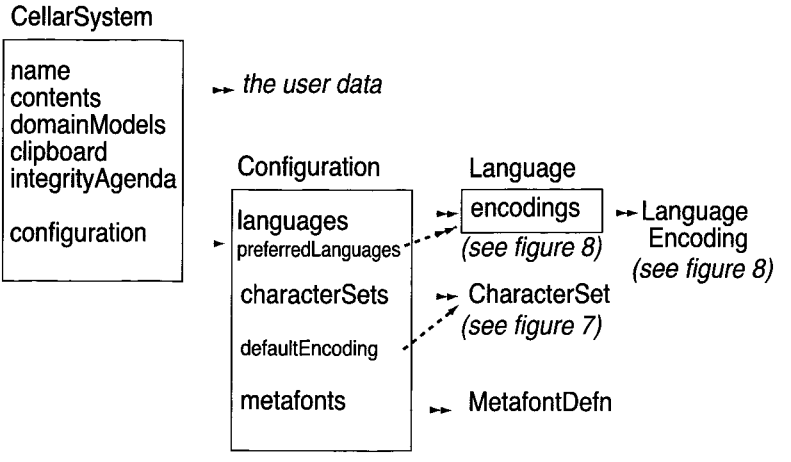


FIGURE 5 Conceptual model of CellarSystem and Configuration

characterSets Defines every character set known to the system. See section 5.2 for a detailed discussion.

defaultEncoding This pointer selects one of the character sets as the default encoding. If user input or file input ever produces a string of characters with no indication of what its encoding is, it will be set to this by default.

metafonts These are abstract fonts that are defined across languages, character sets, and hardware platforms. They are a mechanism for enhancing portability of application code in a multilingual, multiplatform setting.

5 Resources for Encoding

In section 3.1 we established that every string in a multilingual computing environment must know what language it is in, but it is actually more complicated than this: every string needs to know both its language and its method of encoding. Section 5.1 explains why. Sections 5.2 and 5.3 define the conceptual models for language and encoding resources. Finally, section 5.4 explains how the encoding resources can be used to automatically tokenize a string (that is, break it into a stream of meaningful subcomponents).

5.1 The Refined Basic Model: String and Encoding

Section 3.2 gives the following conceptual model for String as a basic data type:

```
class String has
    language : refers to Language
    value    : sequence of PrimitiveString or String
```

But having strings encode their language is not precise enough, for one very basic reason:

12. There are many possible ways to map the characters of a particular language onto a set of character codes.

For instance, German can be encoded in the German reference version (DIN 66 003) of the ISO 646 character set, the Windows ("ANSI") character set, and the Macintosh character set. But the umlauted *u* (*ü*) has a different character code in each case, namely, 125, 252, and 159, respectively. It is therefore not enough to know that a particular string is in German; one must know which of the possible encodings it uses. Therefore:

13. A string, as the fundamental unit of textual data, is not fully specified unless it identifies the character set by which it is encoded.

At this point one might think that the solution is for the string to identify the character set it is encoded in, rather than the language. But as discussed above under point (2) in section 3.1, two identical strings of characters in the same character set may have different processing requirements if they are in different languages. The string must therefore identify both its language and the way it is encoded by a character set. Thus:

14. The information-processing behavior of a string is determined by its method of encoding (which includes its language, its character set, and the way the information units of the language are mapped onto characters).

Therefore, the following statement supercedes (4) and (13):

15. A string, as the fundamental unit of textual data, is not fully specified unless it identifies its method of encoding.

The CELLAR conceptual model of the built-in class String is therefore as follows:

```
class String has
    encoding : refers to Encoding
    value    : sequence of PrimitiveString or String
```

Encoding is thus the object that encapsulates the knowledge the system has about what the characters in the string represent and how they are to be processed.

The language which a given string represents may not be known. Or, it may be a special-purpose notation that does not warrant formal definition as a language. In these cases, identifying the character set is all that can be said about the string's encoding. When the string is in a known language, the encoding involves the relationship between a language and a character set. The class Encoding is thus defined as an abstract class with two subclasses: CharacterSet and LanguageEncoding.

```
abstract class Encoding has
    name          : String
    description   : String

class CharacterSet based on Encoding has
    characters    : sequence of CharacterDefn

class LanguageEncoding base on Encoding has
    language      : Language
    characterSet  : refers to CharacterSet
```

Figure 6 summarizes the basic model in a diagram. It shows that every string stores a pointer to its encoding which may be either a LanguageEncoding or a CharacterSet. If it is a LanguageEncoding, then that object includes an identification of the language and a pointer to the CharacterSet in which it is encoded. Note that many LanguageEncodings may use (that is, point to) the same CharacterSet.

5.2 Defining Character Sets and Their Characters

It is well known that the number of scripts (or writing systems) in the world is much lower than the number of languages, and that many languages thus use the same script. For instance, most of the languages of western Europe use a Roman script; the Slavic languages of eastern Europe use a Cyrillic script. On the other hand, a single language can be written with more than one script. A classic example is Serbo-Croatian: the Serbs write their language with a Cyrillic script, while the Croats write the same language with a Roman script.

Character sets are the computational analog of scripts, and the same relationships to language hold true:

16. Many languages may be encoded using the same character set.
17. A single language may be encoded using multiple character sets.

In other words, there is a many-to-many relation between languages and

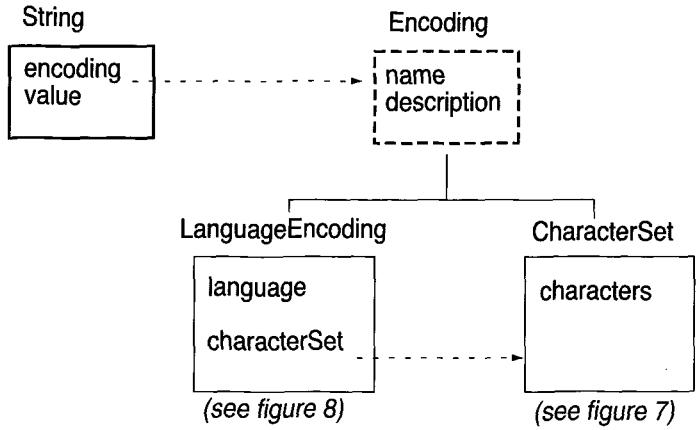


FIGURE 6 Conceptual model of String and Encoding

character sets. Language and CharacterSet are thus modeled independently; the class LanguageEncoding is used to mediate the many-to-many relationship.

The conceptual model of CharacterSet, which is also diagrammed in figure 7, is as follows:

```
class CharacterSet has
    name    : String
    description : String
    codePoints  : seq of CodePoint
    baseBeforeDiacritics : Boolean
```

Name is the name by which the character set is known throughout the system. *Description* is documentation for users of the system; it gives a brief explanation of the purpose and source of the character set. *CodePoints* and *baseBeforeDiacritics* are described below.

Just as a script consists of a collection of graphic symbols (like letters of the alphabet), a character set consists of a collection of characters. But a character is not just a graphic symbol. Rather, it is an abstraction which associates a numerical code stored in the computer with a conventional graphic form. (The actual graphic forms are instantiated by fonts; see section 6.) This association of a numerical code with a graphic form we call a *code point.* But there is more to a character than this alone, because:

18. When two languages are encoded with the same character set,

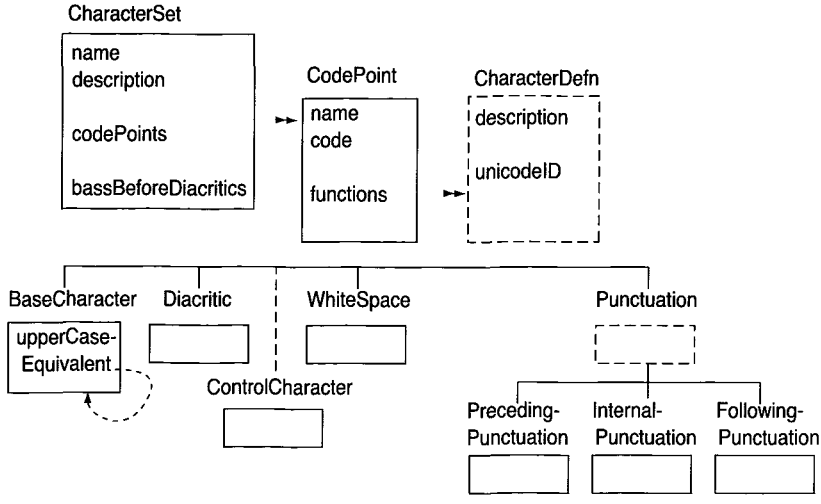


FIGURE 7 Conceptual model of CharacterSet and CharacterDefn

it means that characters at the same code point have the same graphic form but not necessarily the same function.

For instance, whereas one language might use the colon (code 58) of the ASCII character set as a punctuation mark, another might use it to signal length of the preceding vowel. Or one language might use the apostrophe (code 39) as a quotation mark while another uses it for the glottal stop consonant. Thus:

19. A fully specified character associates a code point (that is, a numerical code and a graphic form) with a function.

Therefore, our design formally distinguishes code points from characters. The *codePoints* attribute of CharacterSet is defined to store a sequence of CodePoints (see figure 7), and each CodePoint in turn stores a sequence of CharacterDefns. Each of these character definitions defines an association between its owning code point and the function it represents. When a LanguageEncoding declares which characters it uses, it selects these functionally specified CharacterDefns rather than CodePoints. Even though the CharacterDefns are ultimately motivated by the needs of LanguageEncodings, they are stored as part of the CharacterSet in order to maximize reuse of specifications. For instance, it is not desirable to define a new instance of colon as a punctuation mark for each of the hundreds of languages that can be encoded with a Roman-based character set like ANSI. Rather, the character set owns a single

character definition and all language encodings point to the same character definition.

The formal definitions of CodePoint and CharacterDefn are as follows:

```
class CodePoint has
    name       : String
    code       : Integer
    functions  : seq of CharacterDefn

abstract class CharacterDefn has
    description : String
    unicodeID   : String
```

The *name* is a name for the graphic form associated with the CodePoint, such as "apostrophe" or "lowercase alpha." The *description* describes the function of the character if it is other than what would be assumed by the name of the code point. For instance, colon could be described as "vowel length" for the nonstandard use suggested above. Another way to describe the function of a character is to declare the corresponding character in Unicode. For instance, the apostrophe of ASCII (code 39) could be functioning as "apostrophe" (Unicode 02BC), "opening single quotation mark" (Unicode 2018), or "closing single quotation mark" (Unicode 2019). Note that the *functions* of a CharacterDefn is a sequence attribute so that such multiple functions for a single code point can be defined.

A third way to specify something about the function of a character is by means of the subclass of CharacterDefn that is used. CharacterDefn itself is an abstract class; an instance must use one of the subclasses. The subclasses are BaseCharacter, Diacritic, ControlCharacter, WhiteSpace, PrecedingPunctuation, InternalPunctuation, and FollowingPunctuation. These specifications allow the system to parse a string of characters into its meaningful parts, like the individual words (see section 5.4 below).

Only one subclass adds any attributes, namely, BaseCharacter. It adds a pointer to another BaseCharacter in the same character set that is the upper case equivalent of this one:

```
class BaseCharacter based on CharacterDefn has
    upperCaseEquivalent : refers to BaseCharacter
        in functions of codePoints of owner of my owner
    lowerCaseEquivalent : seq of BaseCharacter means
        refsFromUpperCaseEquivalentOfBaseCharacter
```

There is no need to set the lower case equivalent. It is computed as a virtual attribute that queries the backreference which is automatically set when *upperCaseEquivalent* is set. It allows multiple values since, for instance, both *a* and *á* might declare *A* as their upper case equivalent.

A final note concerns the *baseBeforeDiacritics* attribute of CharacterSet. This stores a Boolean that declares whether the diacritic characters are encoded before or after their base characters. For instance, if *true*, then `a`e` means *àe;* if *false*, it means *aè*. By putting this attribute on CharacterSet we are saying that even if two character sets have identical code points, they are still different character sets if one encodes diacritics after their bases, while the other puts them before. This is because the encoded strings have different meanings. They are also likely to require different fonts to handle the different discipline for placing diacritics on bases.

5.3 Defining Languages and Their Encodings

As noted above in point (17), a single language may be encoded with more than one character set. Figure 8 shows how this is modeled. The Language object has an attribute named *encodings* which may store multiple LanguageEncodings. Each LanguageEncoding stores a pointer to the CharacterSet it uses. The full conceptual model of Language is as follows:

```
class Language has
    name : String
    XXXcode : String
    ISOcode : String
    encodings : seq of LanguageEncoding
    preferredEncoding : refers to LanguageEncoding
                        in my encodings
```

Name is the display name of the language. *XXXcode* is the three-letter code from the *Ethnologue* (Grimes 1992) which uniquely identifies this language from among the 6,000 plus languages of the world. *ISOcode* is the three-letter code from ISO 639 (ISO 1991); the committee draft of this standard specifies unique codes for 404 of the world's major languages. *PreferredEncoding* points to one of the encodings for this language; it is the encoding that the system will use if it is asked to create a string in this language and no specific encoding is requested.
The complete conceptual model for LanguageEncoding is as follows:

```
class LanguageEncoding has
    name : String
    description : String
```

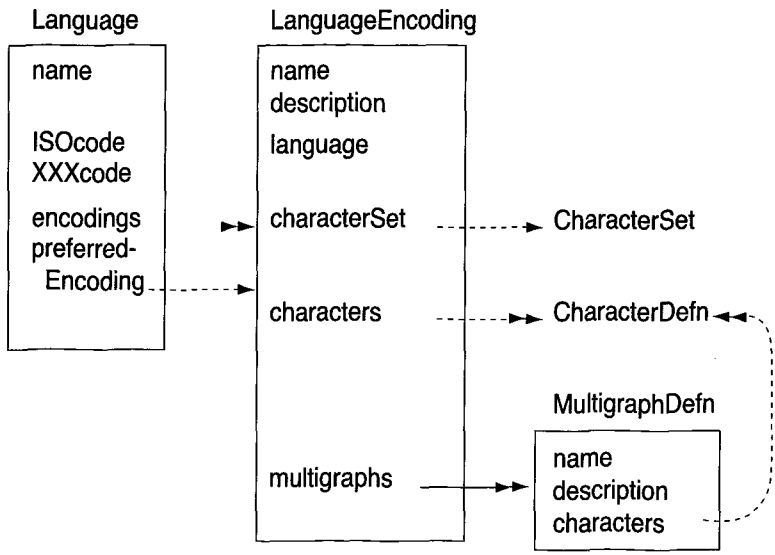


FIGURE 8 Conceptual model of Language and LanguageEncoding

```
language : Language means my owner
characterSet : refers to CharacterSet
   in characterSets of owner of my owner
characters  : refers to seq of CharacterDefn
   in functions of codePoints of my characterSet
multigraphs : seq of MultigraphDefn
```

The first two attributes give a *name* to the encoding and a brief *description* of its purpose or source. Figure 6 shows that LanguageEncoding has an attribute to specify its *language*. Here we see that this is a virtual attribute; the language is not actually stored here but is retrieved by executing the query: *my owner. CharacterSet* indicates which character set is used for this method of encoding the language. It is a pointer to one of the character sets that is defined in the system Configuration (that is, *owner of my owner;* see figure 5).

The *characters* attribute defines the characters that are used in this encoding of the language. It does so by pointing to the appropriate CharacterDefns; note that the *in* clause of the attribute specification limits the selection to only those CharacterDefns that are defined in the character set used by the encoding. By pointing to a selection of the

possible CharacterDefns, this attribute does two things. First, it allows the designer of the encoding to omit (and thus disallow) code points that are not used for encoding this language. Second, as noted above in point (18), the same code point can be used for different functions in encoding different languages. This attribute allows the designer of the encoding to select the particular CharacterDefn (and thus the particular function) that is desired for a particular code point.

The final attribute of a LanguageEncoding declares its *multigraphs*. These are sequences of characters which for some purposes (such as sorting or breaking into sound units) should be considered as a single unit; for instance, in German the trigraph *sch* represents a single sound (the grooved alveopalatal fricative) and the digraph *ng* represents another single sound (the velar nasal). A MultigraphDefn has the following conceptual model:

```
class MultigraphDefn has
    name : String
    description : String
    characters  : refers to seq of CharacterDefn
        in characters of my owner
```

That is, it has a *name* and a *description* (as would a single character). It also specifies the sequence of *characters* that form the multigraph by pointing to the CharacterDefns for the characters. Note that in this case, the *in* clause is even more restricted than it was for *characters* of the LanguageEncoding. Here the possible values are limited to the characters that are defined in *my owner* (that is, the LanguageEncoding).

5.4 Using Encoding Information to Tokenize Strings

One kind of processing that is typically done on text strings is to break them down into meaningful subcomponents, or tokens. For some purposes one might want to view the string as a sequence of characters, for others as a sequence of sound units (where multigraphs are treated as single tokens), and for still others one might want to treat the string as a sequence of wordforms and punctuation marks. The kinds of information stored in CharacterSets and LanguageEncodings make it possible for CELLAR to perform such tokenization of strings using built-in functions. They are as follows:

characterTokenize Returns a sequence of CharacterDefns for the code points in the String.

baseCharacterTokenize Returns a sequence of Strings, each of which is either a base character with associated diacritics or a single character of any other type.

multigraphTokenize Returns a sequence of CharacterDefns and MultigraphDefns.

wordTokenize Returns a sequence of Strings (each representing a base character plus its diacritics) and Punctuation characters. WhiteSpace and ControlCharacters are dropped out.

languageTokenize Returns a sequence of monolingual Strings.

Each of these is implemented as a built-in virtual attribute of class String. For instance, *characterTokenize of "abc"* would return a sequence of three CharacterDefns.

Even the simplest tokenizer, *characterTokenize*, makes use of what is known about the encoding of the string. Suppose that a language encoding for English uses the undifferentiated double-quote character (", ASCII code 34) to mark both opening and closing quotes. The language encoding thus includes two CharacterDefns for code point 34, one as PrecedingPunctuation and the other as FollowingPunctuation. CELLAR uses a simple built-in algorithm that considers position relative to white space and the ends of the string to determine the function of each occurrence of an ambiguous code point. This disambiguated tokenization could be used, for instance, to convert the string to another character set in which different code points and glyphs are used for opening and closing quotes.

More sophisticated tokenizers allow grouping of characters that function together as a single unit at some level. *BaseCharacterTokenize* allows one to identify the base characters of a string and determine which diacritics go with each. CELLAR can do this easily based on its knowledge of which characters are base characters, which are diacritics, which are neither, and whether diacritics attach to the preceding or following base character. *MultigraphTokenize* makes use of CELLAR's knowledge of multigraphs in the string's LanguageEncoding to divide a string into a sequence of single characters and multigraphs.

One of the most useful tokenizers is *wordTokenize*. It divides strings into wordforms (made up of successions of base characters and diacritics) and punctuation characters. *LanguageTokenize* takes apart a multilingual string and returns the sequence of monolingual strings it contains. This differs from *substrings* (section 3.2) in that the latter takes apart only the top layer of structure in a multilingual string, while *languageTokenize* completely flattens the structure converting all layers to a single sequential stream.

6 Conclusion

All of the features of multilingual computing that are described above have been implemented in the CELLAR system. So have user-defined collating sequences for language-specific sorting and cross-language "metafonts" which allow programmers to write applications without worrying about what languages will occur in the data and what fonts they will use for rendering, but space does not allow us to describe these in detail. And much more can be done. Hyphenation, spell checking, and language-specific treatment of numbers and dates are some features we have not begun to work on. In progress is the specification of a finite-state transduction component that can be used to implement language-specific mapping from keyboard to code points, transliteration from one encoding of a language to another, and context-sensitive rendering from character to font glyph (Simons 1989).

These are some of the details of multilingual data processing that can be filled in now that we have built a framework for multilingual computing. Indeed, we believe that this framework is the most significant contribution of our work. This is because the processing details alone cannot give us computing systems that are truly multilingual—they must be part of an environment that addresses all the facets of multilingual computing.

References

Apple Computer. 1988. The Script Manager. *Inside Macintosh* 5:293–322. Reading, Mass.: Addison-Wesley.

Borgida, A. 1985. Features of Languages for the Development of Information Systems at the Conceptual Level. *IEEE Software* 2(1):63–72.

Davis, M. E. 1987. The Macintosh Script System. *Newsletter for Asian and Middle Eastern Languages on Computer* 2(1&2):9–24.

Ford, R., and C. Guglielmo. 1992. Apple's New Technology and Publishing Strategies. *MacWeek*, September 28, 1992: 38–40.

Grimes, B. F. 1992. *Ethnologue: Languages of the World (12th edition).* Dallas, Tex.: Summer Institute of Linguistics.

ISO. 1986. *Information Processing—Text and Office Systems—Standard Generalized Markup Language (SGML).* ISO 8879: 1986 (E). Geneva: International Organization for Standardization.

ISO. 1991. *Code for the Representation of Names of Languages.* ISO CD 639/2:1991. Geneva: International Organization for Standardization.

Knuth, D. E., and P. MacKay. 1987. Mixing Right-to-Left Texts with Left-to-Right Texts. *TUGboat* 8(1): 14–25.

Rettig, M., G. Simons, and J. Thomson. 1993. Extended Objects. *Communications of the ACM* 36(8): 19–24.

Simons, G. F. 1989. The Computational Complexity of Writing Systems. In Ruth M. Brend and David G. Lockwood, eds., *The Fifteenth LACUS Forum,* pp. 538–53. Lake Bluff, Ill.: Linguistic Association of Canada and the United States.

Simons, G. F. 1997. Conceptual Modeling versus Visual Modeling: a Technological Key to Building Consensus. *Computers and the Humanities* 30(4): 303–319.

Simons, G. F. In press. The Nature of Linguistic Data and the Requirements of a Computing Environment for Linguistic Research. To appear in John Lawler and Helen Dry, eds., *Using Computers in Linguistics: a practical guide.* Routledge. A shorter version of this work was also published as follows: 1996. *Dutch Studies on Near Eastern Languages and Literatures*, vol. 2, no. 1, pp. 111–128.

Sperberg-McQueen, C. M., and L. Burnard. 1994. *Guidelines for the Encoding and Interchange of Machine-readable Texts.* Chicago and Oxford: Text Encoding Initiative.

Name Index

Subject Index

T

U

V

W