A Multi-Purpose Interface to an On-line Dictionary

Branimir Boguraev and David Carter
University of Cambridge, Computer Laboratory
Corn Exchange Street, Cambridge CB2 3QG, England

Ted Briscoe
Department of Linguistics, University of Lancaster
Bailrigg, Lancaster LA1 4YT, England

Abstract

We argue that there are two qualitatively different modes of using a machine-readable dictionary in the context of research in computational linguistics: batch processing of the source with the purpose of collating information for subsequent use by a natural language application, and placing the dictionary on-line in an environment which supports fast interactive access to data selected on the basis of a number of linguistic constraints. While it is the former mode of dictionary use which is characteristic of most computational linguistics work to date, it is the latter which has the potential of making maximal use of the information typically found in a machine-readable dictionary. We describe the mounting of the machine-readable source of the *Longman Dictionary of Contemporary English* on a single user workstation to make it available as a development tool for a number of research projects.

1 Introduction

A growing mass of work at present, both within the narrower field of computational linguistics and in the wider context of building knowledge-based systems, is focussed on making use of the lexical resources to be found in a number of (monolingual) dictionaries of the style exemplified by e.g. *The Longman Dictionary of Contemporary English, The Collins English Dictionary,* or *Webster's Seventh New Collegiate Dictionary.* These contain a wealth of information relevant to a wide range of natural language processing functions — a fact which is hardly surprising, given that such sources typically (and almost by definition) contain the results of substantial efforts to collate and analyse data about real language, to elicit collocational and distributional properties of words and to apply common principles of defining their meaning.

The availability of dictionary sources on-line, in the form of machine-readable dictionaries (henceforth MRDs) and encyclopaedias, makes it possible to view these as repositories of large amounts of information, linguistic and extra-linguistic, which can be brought to bear at various points of the natural language understanding process. Developments in hardware, as well as research in computational linguistics, offer the technology both to process lexical resources and to extract from them what is relevant to computer programs concerned with various natural language processing activities. A number of recent projects have extracted data from publishers' tapes and subsequently used it to support activities such as syntactic parsing, speech synthesis, lexical disambiguation, semantic interpretation in context, spelling correction and machine translation. The common denominator in these projects is that the end product incorporates in some form appropriate fragments derived from the machine-readable source.

There are essentially two different modes in which MRDs can be used (see Boguraev, 1987, for more details). The predominant technique to date involves an arbitrary amount of pre-processing, typically in batch, of the on-line source. Those parts of the dictionary entries which contain useful data for the task at hand are extracted and suitably represented in a form directly usable by a client program. Such a model of dictionary use does not in any way rely on the original source being available at the time the language processing application is active, and thus a batch derivation of the appropriate information is a suitable way of transforming the raw data into a usable repository of lexical knowledge.

But the reality of trying to adapt information, originally packaged according to lexicographic and typographic conventions for visual presentation and not at all intended for automated natural language processing, suggests a different model of dictionary use. The non-trivial task of developing suitable procedures for pre-processing of the machine-readable source typically requires careful analysis of the properties of the particular MRD, and is best aided by having fast interactive access to appropriate fragments of it.

In addition, many research projects of a more experimental nature focus on investigating the ways in which the availability of an MRD can aid the development of particular natural language processing systems. The assumption is that an analysis of the accumulated data in the dictionary will reveal regularities which can then be exploited for the task at hand. Just one example, from a number of projects currently under way in Cambridge, illustrating this point is the work of Alshawi (1987), who has analysed definition texts across an entire dictionary to produce a "definition grammar" together with an associated technique for parsing the natural language descriptions of words into semantic structures.

Such projects depend critically not only on the availability of a machine-readable equivalent of a published dictionary, but also on a software system capable of providing fast interactive access into the on-line source through various access routes. Operational natural language processing systems clearly will have well-defined requirements as far as their lexicons are concerned, and once the format of lexical resources has been settled, retrieval of individual entries can be implemented fairly efficiently using standard computational and linguistic techniques (see e.g. Russell et al., 1986). The placing of a dictionary on-line, however, with the intention of making it available to a number of different research projects which need to locate and

collate dictionary samples satisfying a wide range of constraints, requires an efficient and flexible system for management and retrieval of linguistic data.

This is not the computationally straightforward issue it appears to be, as conventional database management systems (DBMS) are not well suited for on-line dictionary support, particularly when the entire dictionary is viewed as a *lexical knowledge base*, more complex in structure and facing more taxing demands in a natural language research environment. This paper addresses the problem in greater detail, by placing it into the wider context of research into computational linguistics and highlighting those issues which pose a challenge for the current DBMS wisdom. We propose a solution adequate to handle most of the lexical requirements of current systems, which is generalisable to a range of MRDs, and describe a particular implementation for single user workstations used in a number of on-going research projects at the universities of Cambridge and Lancaster.

2 The nature of the problem

Several factors put the task of mounting a machine-readable dictionary as a proper development tool beyond the scope of current DBMS practice and make its conversion into a database of e.g. a standard relational kind quite difficult.

Firstly, there is the nature of the data in a dictionary: typically, it contains far too much free text (definitions, examples, cross-reference pointers, glosses on usage and so forth) to fit easily into the concept of structured data. On the other hand, the highly structured and formalised encoding of other types of information (found in e.g. the part of speech, syllabification or pronunciation fields) makes a dictionary equally unsuitable for on-line access by information retrieval methods.

The second factor is due to the nature of the only source of machine-readable dictionaries so far available — namely the publishers' typesetting tapes, originally constructed for the production of a printed version. The organisation of data there, aimed at visual presentation, carries virtually no explicit structure; a tape is simply a character stream containing an arbitrary mixture of typesetting commands and real data. This not only introduces the difficult problem of "parsing" a dictionary entry (addressed in detail by e.g. Kazman, 1986), but also raises the issue of devising a suitable representation for the potentially huge amount of linguistic data; one which does not limit in any way the language processing functions that could be supported or constrain the complexity of the computational counterpart of a dictionary entry.

Finally, there is the nature of the data structures themselves. A text processing application, typically written in Lisp or Prolog, requires that its lexical data is represented in a compatible form, say Lisp s-expressions of arbitrary complexity. Therefore, even if we choose to remain neutral with respect to representation details, we still face the problem of interfacing to a vast number of symbolic s-expressions, held in secondary storage. This problem arises from the unsuitability of conventional data models for handling the complex data structures underlying any sophisticated symbolic processing. Partly, this is due to the inherent restrictions such models impose on the class of data structure they can represent easily — namely records of fixed format. But more importantly, conventional database systems make strong assumptions about the status and use of data they have to hold: databases are taken to consist of a large number of data records taken from a small number of rigidly-defined classes. It is not clear that a lexical "knowledge base", derived from a dictionary and intended to support a wide range of language processing applications, fits this model well.

Some solutions to these problems will no doubt be offered by dedicated efforts to develop special purpose data models, capable of computationally representing a dictionary and amenable to flexible and efficient DBMS support. The work, at the University of Waterloo, on computerising the *Oxford English Dictionary* (Tompa, 1986) is a good example here; similarly, the desire to be able to mount computerised dictionaries on-line for in-house research motivates Byrd's work on a general purpose dictionary access method (Byrd et al., 1986). In the short run, alternative approaches reduce the complexity of the problem by limiting themselves to applying the machine readable source of a dictionary to a small class of similar tasks, and building customised interfaces offering relatively narrow access channels into the on-line data. Thus IBM's WordSmith system (Byrd and Chodorow, 1985) is concerned primarily with providing a browsing functionality which supports retrieval of words "close" to a given word along the dimensions of spelling, meaning and sound, while a group at Bell Labs has several large dictionaries on-line used only for research on stress assignment (Church, 1985). Alshawi et al. (1985) have used a machine-readable source directly for syntactic analysis of texts; however, the approach taken there — namely that of simple pre-indexing by orthography — does not generalise easily for applications which require the rapid locating and retrieval of entries satisfying more than one selection criterion.

3 System functionality

The motivation for the design described here is divided equally between the diverse nature of MRD-based projects in Cambridge and Lancaster and the unique properties of the particular dictionary that they use. The suitability of the *Longman Dictionary of Contemporary English* (LDOCE) for research into computational linguistics has been discussed at length elsewhere (see, in particular, Michiels, 1982); below we will outline several projects undertaken in Cambridge as a context for highlighting its particularly useful characteristics insofar as they are relevant to this paper.

LDOCE carries special lexical and linguistic information which is useful for a number of natural language processing tasks.

1. The dictionary is unique in tagging word senses with *grammar codes* which provide very elaborate syntactic subcategorisation information; a procedure has been developed for mapping the grammar codes into feature clusters (in the style of e.g. Generalised Phrase Structure Grammar), subsequently to be used by a syntactic parser (Boguraev and Briscoe, 1987, describe this in detail). The transformation program is about to be integrated in a software system for gram-

mar support and development, both as a lexicon generator and as a tool for grammar debugging (Boguraev and Ritchie, 1987).

2. The *pronunciation* information in LDOCE has provided the basis for a study, in the larger context of speech recognition, of the implications of the phonetic structure of the English lexicon for different methods for lexical access (Carter, 1986). Again in the context of speech recognition, we intend to tackle the problem of word identification from a lattice of phonemes by constructing a parser that uses information about both phoneme collocations and syntactic predictions derived from independent analyses of the phonetic and grammar coding fields in the dictionary.
3. Furthermore, LDOCE carries special tags, known as *subject* and *box codes*, which encode semantic notions like the overall context in which a word sense is likely to appear (e.g. politics, religion, language) and selectional restrictions on verbs, nouns and compound phrases; we intend to use this information for further guidance during the word recognition process. Independently, an algorithm has been developed for analysing the semantic content proper of the dictionary entries by converting the definition texts in LDOCE into fragments of semantic networks (Alshawi, 1987); this opens opportunities for building a comprehensive and robust semantic component which could then be incorporated into any of the projects mentioned above.

It is clear that in order to make full use of the computerised LDOCE, we need a dictionary access system with proper DBMS functionality, capable of efficient retrieval of entries satisfying selection criteria applying at various levels of linguistic description. The design of the system described here allows precisely such heterogeneous requests. What we offer is a software environment buffering the user from the typically baroque and idiosyncratic format of the raw dictionary source and allowing, via a carefully crafted interface, multiple entry points and arbitrarily complex access paths into the on-line lexical knowledge base.

4 Requirements for the dictionary database

Three main requirements can be identified if the database is to perform the functions intended for it.

Firstly, the source tape of the dictionary must be converted into a format to which fast access can be coupled. This involves, at the very least, overall segmentation of the original character stream into records corresponding to gross lexical categories such as head word, pronunciation and part of speech. This may be a highly complex task, as in Kazman's (1986) project to restructure the text of the OED, or it may be conceptually fairly straightforward, as in the case of LDOCE where considerable segmentation is already present. But in either case, given that the on-line dictionary is intended to support more than one application, a more elaborate structuring of the entries' individual records might turn out to be unsuitable for further unforeseen use. Fortunately, it is clear from work with computerised dictionaries in general that once an application has located the relevant fragment of a dictionary entry, local "parsing" into whatever format is needed can be fast and reliable, and can therefore be done "on the fly" by functions which manipulate individual entries on demand and have no permanent effect on the underlying source. Thus we should aim at incorporating the segmented version of the source intact into the database, to serve directly as its "bottom layer" in the sense that all access paths ultimately point to complete dictionary entries, which are then returned as the results of queries.

Secondly, it should be possible to execute queries involving information of as many different types as possible. Even if the machine-readable source used is a comparatively structured one such as LDOCE, the creation of access paths will involve, for at least some types of information, the non-trivial (but fast) construction of an intermediate and temporary representation by means of the local parsing already mentioned. For example, subcategorisation information is often specified in a rather elliptical form in LDOCE, for the sake of human readability; this must be made explicit by a parsing process, as described in Boguraev and Briscoe (1987). Also, it is desirable to impose a phonologically motivated structure on pronunciations, which are typically given as a string of phonemes and stress markers. This will allow the user to specify a constraint on, say, "the onset of the second syllable of the word", whose position in the phoneme string will not be the same for all words. The straight indexing approach used by e.g. Boguraev and Briscoe (1987) for headword-based access cannot in general provide sufficiently flexible access routes.

Thirdly, the user or client program should be free to specify different types of constraint in any combination. We cannot assume in advance that information of a given type will always be present in great enough quantities to allow efficient retrieval. For example, if the system is being used by an automatic speech recogniser, then at one point in the signal significant information on pronunciation may be available, but few syntactic or semantic constraints may be present; at another point, the situation may be reversed, with the speech signal itself yielding little phonological information but with an expectation-driven parser providing quite specific higher-level constraints. In each case, the stronger, more specific constraints must be used for access, and the weaker ones only for checking the entries retrieved. To achieve this, the system must clearly be able to estimate in advance what the most efficient search strategy will be. This ability to perform maximally efficient searches given many different kinds of constraint will also be important if the database is being used interactively to investigate properties of the language. If the system's claim to be interactive is to be justified, it must be able to tell the user in advance roughly how long a prospective query would take to evaluate, and roughly how many entries would be returned as a result.

5 Design and implementation

The design and implementation of the database system described here reflects the three requirements just identified.

The machine-readable source of LDOCE serves as the bottom layer of the database after undergoing a

"lispification" process described in detail in Boguraev and Briscoe (1987). This process preserves all the information, lexical and typographic, on the tape, and involves little restructuring, serving primarily to reformat the source in a bracketed form in which it can be much more easily read by Lisp programs. The link between the user or client program and the lispified dictionary is provided by a *pointer file* and a *constraint file* whose nature and motivation will be described below.

5.1 Analysing dictionary entries

Information of six different types is analysed for the construction of access paths: semantic features classifying the meanings of words and their dependents; semantic subject area; grammatical part of speech; grammatical subcategorisation; British English pronunciations; and definition texts. All these types can be mixed together in constructing search queries. Entries can also be accessed by spelling patterns.

The codes used for the first three of these types of information have a fairly simple structure, and are hence trivial to extract. The fourth, subcategorisation, is indicated by a complex and highly discriminatory set of codes; the extraction of these codes from the elliptical form in which they occur in LDOCE is described in Boguraev and Briscoe (1987). We will therefore discuss here only the structuring of pronunciations and the treatment of definition texts.

Pronunciations are represented in the dictionary as strings of *phonemes* and primary and secondary *stress markers*. Syllable boundaries are not reliably indicated. Therefore, in order to allow the syllable-based access that a speech recogniser would probably require, pronunciation fields are parsed into *syllables* and, within a syllable, into *onset, peak* and *coda*, using the phonotactic constraints given in Gimson (1980) and employing a *maximal onset principle* (Selkirk, 1978) where these yield ambiguous syllable boundaries. Thus for example the internal syllable boundary in the pronunciation of "constraint" is placed before the "s".

The parser used for analysing pronunciations is a special-purpose one whose (very simple) grammar is incorporated into its code. This allows pronunciations to be parsed many times faster than by a general-purpose parser with a declarative grammar. It also allows constraints on relationships between syllable constituents to be relaxed when necessary. For example, the LDOCE pronunciation of "bedouin" is 'beduin, which violates the constraint that a syllable whose peak is u (as in "put") cannot have a null coda; this constraint is therefore relaxed to obtain a parse.

The strategy used for indexing entries according to the words their definition texts was designed to reflect the fact that it is the semantic content of these words that is likely to be of interest to the user. This has two main consequences:

(1) It is more appropriate to take root forms of words as keys than to treat inflectional variants differently, because it is the root that holds most of the semantic content. Indeed, the inflection used with a particular word often depends on the largely arbitrary choice of syntactic constructions used in the definition. Thus for example, entries whose definitions contain any of the words "film", "films" and "filmed" should all be indexed under "film".

(2) Closed class words are unlikely to be useful as keys because their semantic content is limited and often highly context-dependent. In addition, many of them occur too often to be sufficiently discriminating for efficient lookup. Therefore only open class words are made available as keys.

The task of deriving root forms of words is made much easier by the fact that LDOCE's definition texts are constructed largely from a set of two thousand basic words. When other words are used, they (or, in the case of inflectional variants, their root forms) are shown in a special font. Accurate root extraction for words not so marked can therefore be accomplished simply by stripping off affixes (which are themselves in the basic word list) and applying a few simple rules for spelling changes until a basic word is found. All irregular forms of basic words are stored explicitly.

Distinguishing open and closed class words is also straightforward; a list of closed class words was derived by performing a database lookup using those grammar codes and categories that represent closed classes.

5.2 Constructing access paths

Once the relevant information has been extracted from an entry, constructing acess paths is straightforward in the grammatical, semantic and definition text cases: a list of entry pointers is constructed for every code and every suitable definition word found in the dictionary. Pronunciations, however, are treated differently. To achieve flexibility and efficiency, a pointer list is formed for every distinct syllable in every position in which it occurs (e.g. second syllable in a three-syllable word).

When the whole dictionary has been analysed, a *pointer file* is created containing all the entry pointer lists and, just before each list, its length. As described below, this allows the system to estimate the work involved in evaluating a query without actually having to read the (sometimes very long) list itself.

The next stage is to construct the *constraint file*. This file takes the form of a discrimination net which links every possible constraint on an entry (e.g. a subject area, a grammar code or a constituent of a syllable) to one or, in the pronunciation case, several lists in the pointer file.

5.3 Constructing search queries

A menu-driven graphical interface is provided by means of which the user can construct a search query in the form of a tree whose terminal nodes are constraint values, disjunctions of them, or wild cards. The menus are derived automatically from the constraint file, so that only queries with some chance of being satisfied can be constructed. For example, if the user is constructing a specification of a syllable, the tree at one point may be as in Figure 1.

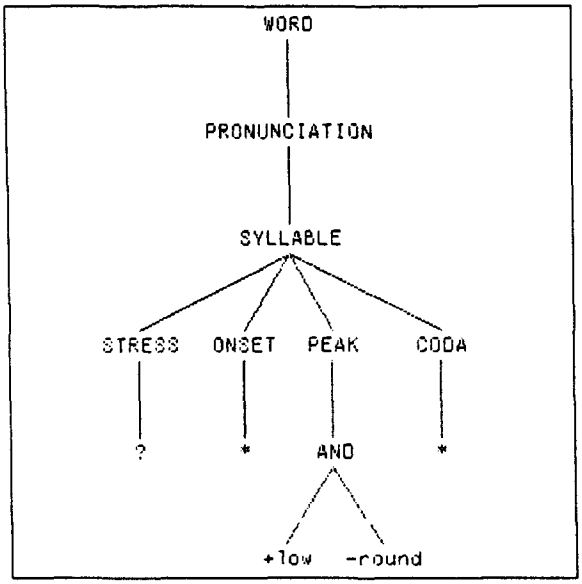

Figure 1

If the user selects the CODA node, the resulting menu, shown in Figure 2, allows him to specify the coda "pst", but not, for example "psm". (In this menu, and in terminal nodes of the PRONUNCIATION subtree of Figure 1, "*" matches any sequence of symbols; "?" matches any single symbol; and all other symbols have the phonetic values defined for them in LDOCE).

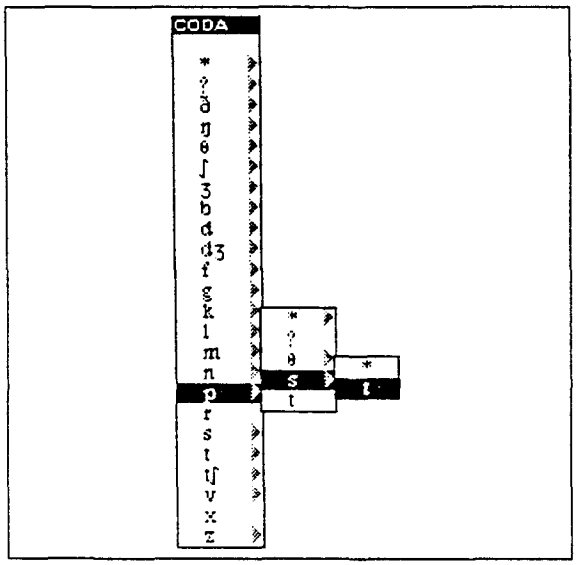

Figure 2

A tree can be constructed either from a WORD node alone, or by instructing the system to build a tree from the entry for a specified word, and then editing it. Once the tree is built, either a *partial search* (to gather statistics) or a *full search* (to retrieve entries) can be requested.

In a partial search, the system follows each constraint to the pointer list(s) it leads to, and sums the lengths of these lists (as recorded explicitly in the pointer file) to display the approximate number of dictionary entries that satisfy it. It also indicates which constraints it would use to look up candidate entries in a full search, which ones it would merely apply as tests to those candidates, and, to allow the user to decide whether or not to order a full search, about how long the process would take. It makes the lookup/test choice using figures for the expected time taken to read (a) a pointer from the constraint file and (b) a complete entry from the dictionary. The most efficient search strategy involves using the most specific few constraints as lookup keys (more specific keys ultimately yielding fewer entries). The optimal number of constraints to use is found by balancing the number of pointers that will have to be read, which increases with the number of lookup keys, against the expected number of entries that will have to be read, which decreases. (An entry will only be read if there is a pointer to it in every pointer list. Therefore if n lookup keys are used, returning pointer lists of lengths L_1, L_2, ... L_n, then the expected number of entries to be read, assuming statistical independence between lists, is $L_1 L_2 ... L_n / D^{n-1}$, where D is the number of entries in the dictionary. This decreases with n because L_i cannot exceed D, and is in fact normally very much smaller).

In a full search, these statistics and choices are not only displayed but are also acted on. The pointer lists for the lookup constraints are intersected, the number of pointers resulting is displayed and, at the user's option, the corresponding entries are read from the dictionary, the test constraints are applied to them, and the surviving entries are displayed. Applying tests to a dictionary entry involves reanalysing the relevant parts of it in the same way as when the database is constructed.

6 An example

As an example, suppose the user wishes to see all entries for three-syllable nouns which describe movable solid objects, whose second syllable has a schwa as peak, and whose third syllable has a coda that is a voiced stop. He constructs the tree in Figure 3 overleaf, and selects the "partial search" option. This returns the information shown in Figure 4.

```
Would look up on these constraints:

(*CODA* (OR b d g) / 3 3)
                        ->   502 items

Would test on these ones:

(*PEAK* E / 2 3)      (->  3629 items)
(@8 5 J)              (->  5179 items)
(*NSYLLS* 3)          (-> 10468 items)
(@C n)                (-> 23835 items)

Estimated pointer+entry reading time:
12.5+85.5=97.8 seconds (502 entries)
```

Figure 4

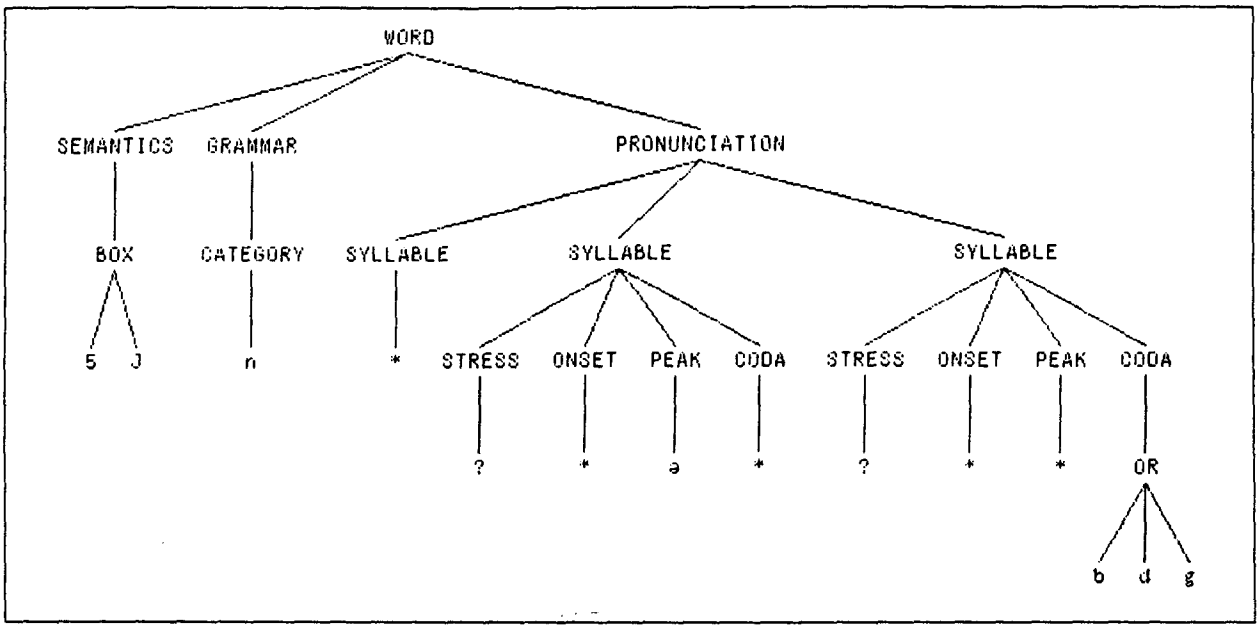

Figure 3

Because of the expected large number of entries in the result and the time that would be taken to read them, the user decides to look only at the entries for such words whose definitions contain the word "camera". He adds the relevant constraint to the tree (the system checking, as he does so, that "camera" is a valid key) and orders another partial search. This time, the statistics are more manageable. A full search is therefore ordered, in which the definition word "camera" is used as the only lookup key, and the other constraints are this time all used as tests. This returns the entries for the words "clapperboard" and "Polaroid", shown in Figure 5.

Entry window

clap.per.board /'klæpəbɔːd || -ərbɔrd/ *n* (subj MP--, box ----J) (when starting to film a scene for the cinema) a board on which the details of the scene to be filmed are written, held up in front of the camera

Po.lar.oid /'pəʊlərɔɪd/*n tdmk* **1** [U] (subj ST--, box ----S---K) a material with which glass is treated in order to make light shine less brightly through it, used in making SUNGLASSES,car windows, etc. **2** [C] (subj PG--, box ----J---A) also (*fml*) **Polaroid cam.e.ra** / ,··· '··· / — a type of camera that produces a finished photograph only seconds after the picture has been taken

Figure 5

7 Conclusion

We have sketched the requirements for, and the design of, a flexible interface to an on-line dictionary, capable of supporting the lexical requirements of a number of active research projects and adaptable to a range of applications. The programs have been developed in Lisp, and make heavy use of the interactive graphic capabilities of Xerox's Lisp workstations. Nothing, however, depends critically on this; the Interlisp-D interactive graphics simply make the task of constructing search specifications very easy for the end user. The design is sufficiently modular to allow easy modification, and the system would be capable of functioning on a conventional minicomputer or mainframe (as long as it supported random access to files) just as well as it does on a single user workstation.

In order to make the lexical knowledge base available to all the projects requiring access to it, the system had to be adapted to fit into the local environment of networked workstations. The database manager was easily repackaged to reflect the model of one server catering for several clients over a network; the Remote Procedure Call Protocol (XSIS, 1981) provided the necessary functionality to incorporate the manager into a *dictionary server* node (of the kind discussed by Kay, 1984) — this bypassed the need for costly fileservers and proved the integrity of the design. We also plan to develop a version of the system running in Franz Lisp under UNIX and accessing the MRC dictionary database (Coltheart, 1981).

While the system implements in effect a linguistically motivated DBMS constructed round a suitable machine-readable source, it stops short of full browsing capability (even though such a capability could easily be added by fully integrating Alshawi's definitions analysis program into the overall design). In this sense the lexical knowledge base discussed here differs

from the concept of a lexical database as described in Calzolari (1986), or underlying Miller's WORDNET (1985). Nonetheless, the methodology described here is sufficiently flexible and powerful to satisfy a substantial proportion of the needs of the computational linguistics community till a proper mix of database and browsing capabilities becomes available.

8 Acknowledgements

This work was supported by a research grant (Number GR/D/4217.7) from the UK Science and Engineering Research Council. We are grateful to the Longman Group Limited for kindly allowing us access to the LDOCE typesetting tape for research purposes. We thank Graham Titmus for his assistance in bringing the dictionary server up.

9 References

Alshawi, H.; Boguraev, B.K. and Briscoe, E.J.(1985) 'A dictionary support environment for real-time parsing', *Proceedings of the Second Conference of the European Chapter of the ACL,* Geneva, pp.171–178

Alshawi, H.A.(1987,forthcoming) 'Processing dictionary definitions with phrasal pattern hierarchies', *Computational Linguistics*

Boguraev, B.K.(1987,forthcoming) 'Machine-readable dictionaries and computational linguistics research' in Walker, D. and Zampolli, A. (eds.), *Automating the lexicon: theory and practice in a multilingual environment,* Cambridge University Press, Cambridge

Boguraev, B.K. and Briscoe, E.J.(1987,forthcoming) 'Large lexicons for natural language processing: utilising the grammar coding system of LDOCE', *Computational Linguistics, vol.13*

Boguraev, B.K.; Ritchie, G.D. et al.(1987,forthcoming) 'The lexical component of a natural language toolkit' in Walker, D. and Zampolli, A. (eds.), *Automating the lexicon: theory and practice in a multilingual environment,* Cambridge University Press, Cambridge

Byrd, R.J. and Chodorow, M.S.(1985) 'Using an online dictionary to find rhyming words and pronunciations for unknown words', *Proceedings of the 25th Annual Meeting of the ACL,* Chicago, Illinois, pp.277–284

Byrd, R.J.; Neumann, G. and Andersson, K.S.B.(1986) *DAM — a dictionary access method,* IBM research report, in preparation

Calzolari, N.(1986,forthcoming) 'Machine-readable dictionaries, lexical data bases and the lexical system' in Walker, D. and Zampolli, A. (eds.), *Automating the lexicon: theory and practice in a multilingual environment,* Cambridge University Press, Cambridge

Carter, D.M.(1986) *An information-theoretic analysis of phonetic dictionary access,* Computer Laboratory, University of Cambridge (submitted to Computer Speech and Language).

Church, K.(1985) 'Stress assignment in letter-to-sound rules for speech synthesis', *Proceedings of the 25th Annual Meeting of the ACL,* Chicago, IL, pp.246–254

Coltheart, M.(1981) 'The MRC Psycholinguistic Database', *Quarterly Journal of Experimental Psychology, vol.33A,* 497–505

Gimson, A.C.(1980) *An introduction to the pronunciation of English (3rd edition),* Edward Arnold, London

Kay, M.(1984) 'The dictionary server', *Proceedings of the 10th International Congress of Computational Linguistics,* Stanford, California, pp.461–462

Kazman, R.(1986) *Structuring the text of the Oxford English Dictionary through finite state transduction,* Technical Report TR-86-20, Department of Computer Science, University of Waterloo, Waterloo, Ontario

Michiels, A.(1982) *Exploiting a large dictionary database,* Ph.D. Thesis, Université de Liège, Belgium

Miller, G.(1985) *WORDNET: a dictionary browser,* In Proceedings of the First International Conference on Information in Data, University of Waterloo centre for the New OED, Waterloo, Ontario

Russell, G.J. et al.(1986) 'A dictionary and morphological analyser for English', *Proceedings of the 11th International Congress on Computational Linguistics,* Bonn, pp.277–279

Selkirk, E.O.(1978) *On prosodic structure and its relation to syntactic structure,* Indiana University Linguistics Club, Bloomington, Indiana

Tompa, F.(1986) *Database design for a dictionary of the future,* Preliminary report, Centre for the New Oxford English Dictionary, University of Waterloo, Waterloo, Ontario

XSIS 038112(1981) *Courier: the Remote Procedure Call protocol,* Xerox Systems Integration Standard, Xerox Corporation, Stamford, Connecticut