Large Lexicons For Natural Language Processing: Utilising The Grammar Coding System of LDOCE

Bran Boguraev

University of Cambridge Computer Laboratory
Corn Exchange Street
Cambridge, CB2 3QG, England
and

Ted Briscoe

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

This article focusses on the derivation of large lexicons for natural language processing. We describe the development of a dictionary support environment linking a restructured version of the Longman Dictionary of Contemporary English to natural language processing systems. The process of restructuring the information in the machine readable version of the dictionary is discussed. The Longman grammar code system is used to construct 'theory neutral' lexical entries. We demonstrate how such lexical entries can be put to practical use by linking up the system described here with the experimental PATR-II grammar development environment. Finally, we offer an evaluation of the utility of the grammar coding system for use by automatic natural language parsing systems.

1 Introduction

The grammar coding system employed by the *Longman Dictionary of Contemporary English* (henceforth LDOCE) is the most comprehensive description of grammatical properties of words to be found in any published dictionary available in machine readable form. This paper describes the extraction of this, and other, information from LDOCE and discusses the utility of the coding system for automated natural language processing.

Recent developments in linguistics, and especially on grammatical theory — for example, Generalised Phrase Structure Grammar (GPSG) (Gazdar et al., 1985), Lexical Functional Grammar (LFG) (Kaplan and Bresnan, 1982) — and on natural language parsing frameworks — for example, Functional Unification Grammar (FUG) (Kay, 1984a), PATR-II (Shieber, 1984) — make it feasible to consider the implementation of efficient systems for the syntactic analysis of substantial fragments of natural language. These developments also emphasise that if natural language processing systems are to be able to handle the grammatical and semantic idiosyncracies of individual lexical items elegantly and efficiently, then the lexicon must be a central component of the parsing system. Real-time parsing imposes stringent requirements on a dictionary support environment; at the very least it must allow frequent and rapid access to the information in the dictionary via the dictionary head words. The research described below is taking place in the context of three collaborative projects (Boguraev, 1987; Russell et al., 1986; Phillips and Thompson, 1986) to develop a general-purpose, wide coverage morphological and syntactic analyser for English. One motivation for our interest in machine readable dictionaries is to attempt to provide a substantial lexicon with lexical entries containing grammatical information compatible with the grammatical framework employed by the analyser.

The idea of using the machine readable source of a published dictionary has occurred to a wide range of researchers, for spelling correction, lexical analysis, thesaurus construction, and machine translation, to name but a few applications. Most of the work on automated dictionaries has concentrated on extracting lexical or other information, essentially by batch processing (eg. Amsler, 1981; Walker and Amsler, 1986), or

0362-613X/87/030203-218$03.00

on developing dictionary servers for office automation systems (Kay, 1984b). Few established parsing systems have substantial lexicons and even those which employ very comprehensive grammars (eg. Robinson, 1982; Bobrow, 1978) consult relatively small lexicons, typically generated by hand. Two exceptions to this generalisation are the Linguistic String Project (Sager, 1981) and the IBM CRITIQUE (formerly EPISTLE) Project (Heidorn et al., 1982; Byrd, 1983); the former employs a dictionary of approximately 10,000 words, most of which are specialist medical terms, the latter has well over 100,000 entries, gathered from machine readable sources. In addition, there are a number of projects under way to develop substantial lexicons from machine readable sources (see Boguraev, 1986 for details). However, as yet few results have been published concerning the utility of electronic versions of published dictionaries as sources for such lexicons. In this paper we provide an evaluation of the LDOCE grammar code system from this perspective.

We chose to employ LDOCE as the machine readable source to aid the development of a substantial lexicon because this dictionary has several properties which make it uniquely appropriate for use as the core knowledge base of a natural language processing system. Most prominent among these are the rich grammatical subcategorisations of the 60,000 entries, the large amount of information concerning phrasal verbs, noun compounds and idioms, the individual subject, collocational and semantic codes for the entries and the consistent use of a controlled 'core' vocabulary in defining the words throughout the dictionary. (Michiels (1982) contains further description and discussion of LDOCE.) In this paper we focus on the exploitation of the LDOCE grammar coding system; Alshawi et al. (1985) and Alshawi (1987) describe further research in Cambridge utilising different types of information available in LDOCE.

The information available in the dictionary is both very rich and diverse, but also typically only semi-formalised, as it is intended for human, rather than machine, interpetation. As a consequence the programs we are developing, both to restructure and to exploit this information, need to undergo constant revision as they are being used. The system we describe is not intended for off-line use, where one might attempt to derive, completely automatically, a lexicon for natural language analysis. Rather than trying to batch process the electronic source, lexicon development from the LDOCE tape is more incremental and interactive. Our system is designed as an integral part of a larger grammar (and lexicon) development environment, where new lexical entries are automatically generated from the on-line version of the dictionary, checked for correctness and consistency and only then added to the 'final' lexicon.

The problem of utilising LDOCE in natural language processing falls into two areas. Firstly, we must provide an environment in which the machine readable source is linked to the development environment in an appropriate fashion and secondly, we must restructure the information in the dictionary, using the development environment, in such a way that natural language processing systems are able to utilise it effectively. As an example, we demonstrate how the LDOCE grammar codes can be put to practical use by linking up the system with the experimental PATR-II parsing system. Finally, we offer an evaluation of the utility of the LDOCE grammar coding system from the perspective of natural language processing.

2 The Access Environment

There is a well recognised problem with providing computational support for machine readable dictionaries, in particular where issues of access are concerned. On the one hand, dictionaries exhibit far too much structure for conventional techniques for managing 'flat' text to apply to them. On the other hand, the equally large amounts of free text in dictionary entries, as well as the implicitly marked relationships commonly used to encode linguistic information, makes a dictionary difficult to represent as a structured database of a standard, eg. relational, type. In addition, in order to link the machine readable version of LDOCE to our development environment, and eventually to our natural language processing systems, we need to provide fast access from Lisp to data held in secondary storage. Lisp is not particularly well suited for interfacing to complex, structured objects, and it was not our intention to embark on a major effort involving the development of a formal model of a dictionary (of the style described in, eg., Tompa 1986); on the other hand a method of access was clearly required, which was flexible enough to support a range of applications intending to make use of the LDOCE tape.

The requirement for having the dictionary entries in a form convenient for symbolic manipulation from within Lisp was furthermore augmented by the constraint that all the information present in the typesetting tape should be carried over to the on-line version of LDOCE, since it is impossible to say in advance which records and fields of an entry would, or would not, be of potential use to a natural language processing program. Finally, the complexity of the data structures stored on disc should not be constrained in any way by the method of access, as we do not have a very clear idea what form the restructured dictionary may eventually take.

Given that we were targeting all envisaged access routes from LDOCE to systems implemented in Lisp, and since the natural data structure for Lisp is the s-expression, we adopted the approach of converting the tape source into a set of list structures, one per entry. Our task was made possible by the fact that while far from being a database in the accepted sense of the word, the LDOCE typesetting tape is the only truly computerised dictionary of English (Michiels, 1983).

The logical structure of a dictionary entry is reflected on the tape as a sequence of typed records (see Figure 1), each with additional internal segmentation, where records and fields correspond to separate units in an entry, such as headword, pronunciation, grammar code, word senses, and so forth.

```
(Record-type homograph
      (Seq-number E-code I-code))
(Record-type headword (Serial-no Main-entry))
(Record-type pronunciation (Phonetic))
(Record-type variant (Spelling Pronunciation))
(Record-type part-of-speech (Category Inflection))
(Record-type grammar-code (G-code Label))
(Record-type def-code
      (Seq-number G-code Subj-code Box-codes))
(Record-type entry-text
      (Phrase Label Definition Examples X-ref))
(Record-type def-word (Basic-word Morphology
      Homograph Word-sense))
(Record-type cross-reference (Type Pointers))
(Record-type word-sense (Def-code Def-text))
(Record-type usage (Text X-ref))
```

Figure 1

The "lispification" of the typesetting tape was carried out in a series of batch jobs, via a program written in a general text editing facility. The need to carry out the conversion without any loss of information meant that special attention had to be paid to the large number of non-printing characters which appear on the tape. Most of these signal changes in the typographic appearance of the printed dictionary, where crucial information about the structure of an entry is represented by changes of typeface and font size. All control characters were translated into atoms of the form ***AB**, where **A** and **B** correspond to the hexadecimal digits of the ASCII character code. Information was thus preserved, and readily available to any program which needed to parse the implicit structure of a dictionary entry or field, and the lispified source was made suitable for transporting between different software configurations and operating systems. Figure 2 illustrates part of an entry as it appears in the published dictionary, on the typesetting tape and after lispification.

Note that as a result of the lispification, brackets have been inserted at suitable points, both to delimit entries and indicate their internal structure; in addition characters special to Lisp have been appropriately escaped. Thus an individual dictionary entry can now be made available to a client program by a single call to a generic **read** function, once the Lisp reader has been properly positioned and 'aligned' with the beginning of

```
rivet² υ 1 [T1;X9] to cause to fasten with RIVETs¹:...

28289801<R0154300<rivet
28289902<02<  <
28290005<v<
28290107<0100<T1;X9<NAZV<    H   XS
28290208<to cause to fasten with
28290318<{*CA}RIVET{*CB}{*46}s{*44}{*8A}:

((rivet)
 (1 R0154300 !< rivet)
 (2 2 !< !<)
 (5 v !<)
 (7 100 !< T1 !; X9 !< NAZV !< ----H---XS)
 (8 to cause to fasten with
   *CA RIVET *CB *46 s *44 *8A : .....))
```

Figure 2

the s-expression encoding the required entry. In the lispified entry in Figure 2 the numbers at the head of each sublist indicate the type of information stored in each field within the overall entry. For example, "5" is the part of speech field, and "8" is the word sense definition.

The 60,000 or so complete entries of the processed dictionary require of the order of 20 MBytes to store. The problem of access, from Lisp, to the dictionary entry s-expressions held on secondary storage cannot be resolved by ad hoc solutions, such as sequential scanning of files on disc or extracting subsets of such files which will fit in main memory, as these are not adequate as an efficient interface to a parser. (Exactly the same problem would occur if our natural language systems were implemented in Prolog, since the Prolog 'database facility' refers to the knowledge base that Prolog maintains in main memory.) In principle, given that the dictionary is now in a Lisp-readable format, a powerful virtual memory system might be able to manage access to the internal Lisp structures resulting from reading the entire dictionary; we have, however, adopted an alternative solution as outlined below.

We have mounted LDOCE on-line under two different hardware configurations. In both cases the same lispified form of the dictionary has been converted into a random access file, paired together with an indexing file from which the disc addresses of dictionary entries for words and compounds can be computed.

A series of systems in Cambridge are implemented in Lisp running under Unix™. They all make use of an efficient dictionary access system which services requests for s-expression entries made by client programs. A dictionary access process is fired off, which dynamically constructs a search tree and navigates through it from a given homograph directly to the offset in the lispified file from where all the associated information can be retrieved. As Alshawi (1987) points out, given that no situations were envisaged where the information from the tape would be altered once installed in secondary storage, this simple and conven-

tional access strategy is perfectly adequate. The use of such standard database indexing techniques makes it possible for an active dictionary process to be very undemanding with respect to main memory utilisation. For reasons of efficiency and flexibility of customisation, namely the use of LDOCE by different client programs and from different Lisp and/or Prolog systems, the dictionary access system is implemented in the programming language C and makes use of the inter-process communication facilities provided by the Unix operating system. To the Lisp programmer, the creation of a dictionary process and subsequent requests for information from the dictionary appear simply as Lisp function calls.

Most of the recent work with the dictionary, and in particular the decompacting and analysis of the grammar codes has been carried out in Interlisp-D on Xerox 1100 series workstations. The same lispified form of the dictionary was used. Originally it was installed on a single workstation and only available locally. Instead of a separate process building a search tree, the access method relies on a precompiled, multilevel indexing structure which allows direct hashing into the on-line source. In addition, the powerful Interlisp-D virtual memory allows the access system to be significantly enhanced by caching most of the working subset of the dictionary at any given turn in main memory. It turns out that for a single user workstation, specially tuned for Lisp and operations optimised at the microcode level for random file access and s-expression I/O, this strategy offers remarkably good results.

More recently, a dictionary server, of the kind described by Kay (1984b), was implemented and installed as a background process on a Xerox workstation networked together with the rest of the equipment dedicated to natural language processing applications (Boguraev et al., 1987). Again, the same lispified form of the machine readable source of LDOCE was used. From the point of view of providing a centralised service to more than one client, efficiently over a packet switching network, disc space on the server processor was not an issue. This made it possible to construct a larger, but more comprehensive, index for the dictionary, which now allows the recovery of a word in guaranteed time (typically less than a second).

The main access route into LDOCE for most of our current applications is via the homograph fields (see Figure 1). Options exist in the access software to specify which particular homograph (or homographs) for a lexical item is required. The early process of lispification was designed to bring together in a single group all dictionary entries corresponding not only to different homographs, but also to lexicalised compounds for which the argument word appears as the head of the compound. Thus, the primary index for *blow* allows access to two different verb homographs (eg. *blow*^{3}) , two different noun homographs (eg. *blow*^{2}), 10 compounds (eg. *blow off* and *blow-by-blow*), or all 14 of the dictionary entries (not necessarily to be found in subsequent positions in the dictionary) related to *blow*. While no application currently makes use of this facility, the motivation for such an approach to dictionary access comes from envisaging a parser which will operate on the basis of the on-line LDOCE; and any serious parser must be able to recognise compounds before it segments its input into separate words.

From the master LDOCE file, we have computed alternative indexing information, which allows access into the dictionary via different routes. In addition to headwords, dictionary search through the pronunciation field is available; Carter (1987) has merged information from the pronunciation and hyphenation fields, creating an enhanced phonological representation which allows access to entries by broad phonetic class and syllable structure (Huttenlocher and Zue, 1983). In addition, a fully flexible access system allows the retrieval of dictionary entries on the basis of constraints specifying any combination of phonetic, lexical, syntactic, and semantic information (Boguraev et al., 1987). Independently, random selection of dictionary entries is also provided to allow the testing of software on an unbiased sample.

3 The Format Of The Grammar Codes

The lispified LDOCE file retains the broad structure of the typesetting tape and divides each entry into a number of fields — head word, pronunciation, grammar codes, definitions, examples, and so forth. However, each of these fields requires further decoding and restructuring to provide client programs with easy access to the information they require (see Calzolari (1984) for further discussion). For this purpose the formatting codes on the typesetting tape are crucial since they provide clues to the correct structure of this information. For example, word senses are largely defined in terms of the 2000 word core vocabulary, however, in some cases other words (themselves defined elsewhere in terms of this vocabulary) are used. These words always appear in small capitals and can therefore be recognised because they will be preceded by a font change control character. In Figure 1 above the definition of *rivet* as verb includes the noun definition of "RIVET^1", as signalled by the font change and the numerical superscript which indicates that it is the first (i.e. noun entry) homograph; additional notation exists for word senses within homographs. On the typesetting tape, font control characters are indicated by hexadecimal numbers within curly brackets. In addition, there is a further complication because this sense is used in the plural and the plural morpheme must be removed before RIVET can be associated with a dictionary entry. However, the restructuring program can achieve this because such morphology is always italicised, so the program knows that, in the context of non-core vocabulary items, the italic font control character signals the

```
((pair)
 (1 P0008800 < pair)
 (2 1 < <)
 (3 peER)
 .....
 (7 200 < C9 !, esp !. *46 of < CD-- < ----J---Y)
 (8 *45 a *44 2 things that are alike or of the same
   kind !, and are usu !. used together : *46 a pair of
   shoes !| a beautiful pair of legs *44 *63 compare
   *CA COUPLE *CB *8B *45 b *44 2 playing cards of
   the same value but of different *CA SUIT *CB *46
   s *8A *44 (3) : *46 a pair of kings)
 (7 300 < GC < ---- < --S-U---Y)
 (8 *45 a *44 2 people closely connected : *46 a pair
   of dancers *45 b *CA COUPLE *CB *8B *44 (2)
   (esp !. in the phr !. *45 the happy pair *44) *45 c
   *46 sl *44 2 people closely connected who cause
   annoyance or displeasure : *46 You!'re a fine pair
   coming as late as this !!        .....)

(Word-sense (Number 2)
  ((Sub-definition
    (Item a) (Label NIL)
    (Definition 2 things that are alike or of the
     same kind !, and are usually used together)
    ((Example NIL (a pair of shoes))
     (Example NIL (a beautiful pair of legs)))
    (Cross-reference
     compare-with
      (Ldoce-entry (Lexical COUPLE)
       (Morphology NIL)
       (Homograph-number 2)
       (Word-sense-number NIL)))
   (Sub-definition
    (Item b) (Label NIL)
    (Definition 2 playing cards of the same value
     but of different
      (Ldoce-entry (SUIT)
       (Morphology s)
       (Homograph-number 1)
       (Word-sense-number 3))
     ((Example NIL (a pair of kings))))))
(Word-sense (Number 3)
  ((Sub-definition
    (Item a) (Label NIL)
    (Definition 2 people closely connected)
    ((Example NIL (a pair of dancers))))
   (Sub-definition
    (Item b) (Label NIL)
    (Definition
     (Ldoce-entry (Lexical COUPLE)
      (Morphology NIL)
      (Homograph-number 2)
      (Word-sense-number 2))
     (Gloss:
       especially in the phrase the happy pair )))
   (Sub-definition
    (Item c) (Label slang)
    (Definition 2 people closely connected who
     cause annoyance or displeasure)
    ((Example NIL
      (You!'re a fine pair coming as late as this!)))))))
```

Figure 3

occurrence of a morphological variant of a LDOCE head entry.

A suite of programs to unscramble and restructure all the fields in LDOCE entries has been written which is capable of decoding all the fields except those providing cross-reference and usage information for complete homographs. Figure 3 illustrates a simple lexical entry before and after the application of these programs. The development of the restructuring programs was a non-trivial task because the organisation of information on the typesetting tape presupposes its visual presentation, and the ability of human users to apply common sense, utilise basic morphological knowledge, ignore minor notational inconsistencies, and so forth. To provide a test-bed for these programs we have implemented an interactive dictionary browser capable of displaying the restructured information in a variety of ways and representing it in perspicuous and expanded form.

In what follows we will discuss the format of the grammar codes in some detail as they are the focus of the current paper, however, the reader should bear in mind that they represent only one comparatively constrained field of an LDOCE entry and therefore, a small proportion of the overall restructuring task. Figure 4 illustrates the grammar code field for the third word sense of the verb *believe* as it appears in the published dictionary, on the typesetting tape and after restructuring.

```
believe v ... 3 [T5a,b;V3;X(to be)1, (to be)7]

(7 300 !< T5a !, b !; V3 !; X (*46 to be
   *44) 1 !, (*46 to be *44) 7 !< )

sense-no 3   head: T5a
             head: T5b
             head: V3
             head: X1 right optional (to be)
             head: X7 right optional (to be)
```

Figure 4

LDOCE provides considerably more syntactic information than a traditional dictionary. The Longman lexicographers have developed a grammar coding system capable of representing in compact form a non-trivial amount of information, usually to be found only in large descriptive grammars of English (such as Quirk et al., 1985). A grammar code describes a particular pattern of behaviour of a word. Patterns are descriptive, and are used to convey a range of information: eg. distinctions between count and mass nouns (*dog* vs. *desire*), predicative, postpositive and attributive adjectives (*asleep* vs. *elect* vs. *jokular*), noun complementation (*fondness, fact*) and, most importantly, verb complementation and valency.

Grammar codes typically contain a capital letter, followed by a number and, occasionally, a small letter, for example [T5a] or [V3]. The capital letters encode information "about the way a word works in a sentence

or about the position it can fill" (Procter, 1978: xxviii); the numbers "give information about the way the rest of a phrase or clause is made up in relation to the word described" (ibid.). For example, "T" denotes a transitive verb with one object, while "5" specifies that what follows the verb must be a sentential complement introduced by *that*. (The small letters, eg. "a" in the case above, provide further information typically related to the status of various complementisers, adverbs and prepositions in compound verb constructions: eg. "a" indicates that the word *that* can be left out between a verb and the following clause.) As another example, "V3" introduces a verb followed by one NP object and a verb form (V) which must be an infinitive with *to* (3).

In addition, codes can be qualified with words or phrases which provide further information concerning the linguistic context in which the deserbed item is likely, and able, to occur; for example [D1(*to*)] or [L(*to be*)1]. Sets of codes, separated by semicolons, are associated with individual word senses in the lexical entry for a particular item, as Figure 5 illustrates. These sets are elided and abbreviated in the code field associated with the word sense to save space. Partial codes sharing an initial letter can be separated by commas, for example [T1,5a]. Word qualifiers relating to a complete sequence of codes can occur at the end of a code field, delimited by a colon, for example [T1;I0: (DOWN)]. Codes which are relevant to all the word senses in an entry often occur in a separate field after the head word and occasionally codes are elided from this field down into code fields associated with each word sense as, for example, in Figure 6. Decompacting and restructuring grammar code entries into a format more suitable for further automated analysis can be done with knowledge of the syntax of the grammar code system and the significance of punctuation and font changes. However, discovering the syntax of the system is difficult since no explicit description is available from Longman and the code is geared more towards visual presentation than formal precision; for example, words which qualify codes, such as "to be" in Figure 4, appear in italics and therefore, will be preceded by the font control character ***45.** But sometimes the thin space control character ***64** also appears; the insertion of this code is based solely on visual criteria, rather than the informational structure of the dictionary. Similarly, choice of font can be varied for reasons of appearance and occasionally in-

feel[1] ***v*** **1 [T1,6] to get the knowledge of by touching with the fingers: ... 2 [Wv6;T1] to experience (the touch or movement of something): ... 3 [L7] to experience (a condition of the mind or body); be consciously: ... 4 [L1] to seem to oneself to be: ... 5 [T1,5;V3] to believe, esp. for the moment 6 [L7] to give (a sensation): ... 7 [Wv6;I0] to (be able to) experience sensations: ... 8 [Wv6;T1] to suffer because of (a state or event): ... 9 [L9 (*after, for*)] to search with the fingers rather than with the eyes: ...**

Figure 5.

see off ***v adv*** **[T1] 1 [(*at*)] to go to the airport, station, etc., with (someone who is beginning a trip): *saw his friend off at the bus station* 2 to remain unharmed until (something or someone dangerous) has ceased to be active; WITHSTAND: *They saw off 3 enemy attacks within 3 days***

Figure 6

formation normally associated with one field of an entry is shifted into another to create a more compact or elegant printed entry.

In addition to the 'noise' generated by the fact that we are working with a typesetting tape geared to visual presentation, rather than a database, there are errors and inconsistencies in the use of the grammar code system. Examples of errors, illustrated in Figure 7, include the code for the noun *promise* which contains a misplaced comma, that for the verb *scream*, in which a colon delimiter occurs before the end of the field, and that for the verb *like* where a grammatical label occurs inside a code field.

promise *n* ...	**1**	[C(of),C3,5; *under*+U]
scream *v* ...	**3**	[T1,5; (OUT); I0]
like *v* ...	**2**	[T3,4; *neg.*]

Figure 7

In addition, inconsistencies occur in the application of the code system by different lexicographers. For example, when codes containing "to be" are elided they mostly occur as illustrated in Figure 4 above. However, sometimes this is represented as [L(to be)1,9]. Presumably this kind of inconsistency arose because one member of the team of lexicographers realised that this form of elision saved more space.

This type of error and inconsistency arises because grammatical codes are constructed by hand and no automatic checking procedure is attempted (see Michiels, 1982, for further comment). One approach to this problem is that taken by the ASCOT project (Akkerman et al., 1985; Akkerman, 1986). In this project, a new lexicon is being manually derived from LDOCE. The coding system for the new lexicon is a slightly modified and simplified version of the LDOCE scheme, without any loss of generalisation and expressive power. More importantly, the assignment of codes for problematic or erroneously labelled words is being corrected in an attempt to make the resulting lexicon more appropriate for automated analysis. In the medium term this approach, though time consuming, will be of some utility for producing more reliable lexicons for natural language processing.

However, in the short term, the necessity to cope with such errors provides much of the motivation for our interactive approach to lexicon development, since this allows the restructuring programs to be progressively refined as these problems emerge. Any attempt at batch processing without extensive initial testing of this kind would inevitably result in an incomplete and possibly inaccurate lexicon.

4 The Content Of The Grammar Codes

Once the grammar codes have been restructured, it still remains to be shown that the information they encode is going to be of some utility for natural language processing. The grammar code system used in LDOCE is based quite closely on the descriptive grammatical framework of Quirk et al. (1972, 1985). The codes are doubly articulated; capital letters represent the grammatical relations which hold between a verb and its arguments and numbers represent subcategorisation frames which a verb can appear in. Most of the subcategorisation frames are specified by syntactic category, but some are very ill-specified; for instance, 9 is defined as "needs a descriptive word or phrase". In practice many adverbial and predicative complements will satisfy this code, when attached to a verb; for example, *put* [X9] where the code marks a locative adverbial prepositional phrase vs. *make* under sense 14 (hereafter written *make*(14)) is coded [X9] where it marks a predicative noun phrase or prepositional phrase.

The criteria for assignment of capital letters to verbs is not made explicit, but is influenced by the syntactic and semantic relations which hold between the verb and its arguments; for example, I5, L5 and T5 can all be assigned to verbs which take a NP subject and a sentential complement, but L5 will only be assigned if there is a fairly close semantic link between the two arguments and T5 will be used in preference to I5 if the verb is felt to be semantically two place rather than one place, such as *know* versus *appear*. On the other hand, both *believe* and *promise* are assigned V3 which means they take a NP object and infinitival complement, yet there is a similar semantic distinction to be made between the two verbs; so the criteria for the assignment of the V code seem to be purely syntactic. Michiels (1982) and Akkerman et al. (1985) provide a more detailed analysis of the information encoded by the LDOCE grammar codes and discuss their efficacy as a system of linguistic description. Ingria (1984) comprehensively compares different approaches to complementation within grammatical theory providing a touchstone against which the LDOCE scheme can be evaluated.

Most automated parsing systems employ grammars which carefully distinguish syntactic and semantic information, therefore, if the information provided by the Longman grammar code system is to be of use, we need to be able to separate out this information and map it into a representation scheme compatible with the type of lexicon used by such parsing systems.

The program which transforms the LDOCE grammar codes into lexical entries utilisable by a parser takes as input the decompacted codes and produces a relatively theory neutral representation of the lexical entry for a particular word, in the sense that this representation could be further transformed into a format suitable for most current parsing systems. For example, if the input were the third sense of *believe*, as in Figure 4, the program would generate the (partial) entry shown in Figure 8 below. The four parts correspond to different syntactic realisations of the third sense of the verb *believe*. **Takes** indicates the syntactic category of the subject and complements required for a particular realisation. **Type** indicates aspects of logical semantics discussed below.

```
((Takes NP SBar) (Type 2))
((Takes NP NP Inf) (Type 2 ORaising))
(or ((Takes NP NP NP) (Type 2 ORaising))
    ((Takes NP NP AuxInf) (Type 2 ORaising)))
(or ((Takes NP NP AP) (Type 2 ORaising))
    ((Takes NP NP AuxInf) (Type 2 ORaising)))
```

Figure 8

At the time of writing, rules for producing adequate entries to drive a parsing system have only been developed for verb codes. In what follows we will describe the overall transformation strategy and the particular rules we have developed for the verb codes. Extending the system to handle nouns, adjectives and adverbs would present no problems of principle. However, the LDOCE coding of verbs is more comprehensive than elsewhere, so verbs are the obvious place to start in an evaluation of the usefulness of the coding system. No attempt has been made to map any closed class entries from LDOCE, as a 3,000 word lexicon containing most closed class items has been developed independently by one of the groups collaborating with us to develop the general purpose morphological and syntactic analyser (see the Introduction and Russell et al., 1986).

Initially the transformation of the LDOCE codes was performed on a code-by-code basis, within a code field associated with each individual word sense. This approach is adequate if all that is required is an indication of the subcategorisation frames relevant to any particular sense. In the main, the code numbers determine a unique subcategorisation. Thus the entries can be used to select the appropriate VP rules from the grammar (assuming a GPSG-style approach to subcategorisation)

and the relevant word senses of a verb in a particular grammatical context can be determined. However, if the parsing system is intended to produce a representation of the predicate-argument structure for input sentences, then this simple approach is inadequate because the individual codes only give partial indications of the semantic nature of the relevant sense of the verb.

The solution we have adopted is to derive a semantic classification of the particular sense of the verb under consideration on the basis of the complete set of codes assigned to that sense. In any subcategorisation frame which involves a predicate complement there will be a non-transparent relationship between the superficial syntactic form and the underlying logical relations in the sentence. In these situations the parser can use the semantic type of the verb to compute this relationship. Expanding on a suggestion of Michiels (1982), we classify verbs as Subject Equi, Object Equi, Subject Raising or Object Raising for each sense which has a predicate complement code associated with it. These terms, which derive from Transformational Grammar, are used as convenient labels for what we regard as a semantic distinction; the actual output of the program is a specification of the mapping from superficial syntactic form to an underlying logical representation. For example, labelling *believe*(3) (Type 2 ORaising) indicates that this is a two place predicate and that, if *believe*(3) occurs with a syntactic direct object, as in

(1) *John believes the Earth to be round*

it will function as the logical subject of the predicate complement. Michiels proposed rules for doing this for infinitive complement codes; however there seems to be no principled reason not to extend this approach to computing the underlying relations in other types of VP as well as in cases of NP, AP and PP predication (see Williams (1980), for further discussion).

The five rules which are applied to the grammar codes associated with a verb sense are ordered in a way which reflects the filtering of the verb sense through a series of syntactic tests. Verb senses with an [*it* + I5] code are classified as **Subject Raising.** Next, verb senses which contain a [V] or [X] code and one of [D5], [D5a], [D6] or [D6a] codes are classified as **Object Equi.** Then, verb senses which contain a [V] or [X] code and a [T5] or [T5a] code in the associated grammar code field, (but none of the D codes mentioned above), are classified as **Object Raising.** Verb senses with a [V] or [X(*to be*)] code, (but no [T5] or [T5a] codes), are classified as **Object Equi.** Finally, verb senses containing a [T2], [T3] or [T4] code, or an [I2], [I3] or [I4] code are classified as **Subject Equi.** Figure 9 gives examples of each type.

The Object Raising and Object Equi rules attempt to exploit the variation in transformational potential between Raising and Equi verbs; thus, in the paradigm case, Object Raising verbs take a sentential complement and Object Equi verbs do not, as examples (2) and (3) illustrate.

happen(3) [Wv5;*it*+I5]
(Type 1 SRaising)
warn(1) [Wv4;I0;T1:(*of,against*),5a;D5a;V3]
(Type 3 OEqui)
assume(1) [Wv4;T1,5a,b;X(*to be*)1,7]
(Type 2 ORaising)
decline(3) [T1,3;I0]
(Type 2 SEqui)

Figure 9

(2) *John believes that the Earth is round.*

(3) **John forces that the Earth is round.*

Secondly, if a verb takes a direct object and a sentential complement, it will be an Equi verb, as examples in (4) and (5) illustrate.

(4) *John persuaded Mary that the Earth is round.*

(5) **John believed Mary that the Earth is round.*

Clearly, there are other syntactic and semantic tests for this distinction, (see eg. Perlmutter and Soames, 1979:472), but these are the only ones which are explicit in the LDOCE coding system.

Once the semantic type for a verb sense has been determined, the sequence of codes in the associated code field is translated, as before, on a code-by-code basis. However, when a predicate complement code is encountered, the semantic type is used to determine the type assignment, as illustrated in Figures 4 and 8 above. Where no predicate complement is involved, the letter code is usually sufficient to determine the logical properties of the verb involved. For example, T codes nearly always translate into two-place predicates as Figure 10 illustrates.

In some cases important syntactic information is conveyed by the word qualifiers associated with particular grammar codes and the translation system is therefore sensitive to these correlations. For example, the Subject Raising rule above makes reference to the left

hate[2] v ... 1 [T1,3,4; V3,4] to have a great dislike of
.....

```
(hate
 ((Sense 1)
  ((Takes NP NP) (Type 2))
  ((Takes NP Inf) (Type 2 SEqui))
  ((Takes NP Ing) (Type 2 SEqui))
  ((Takes NP NP Inf) (Type 3 OEqui))
  ((Takes NP NP Ing) (Type 3 OEqui))
```

Figure 10

context qualifier "it". Another example where word qualifiers can be utilised straightforwardly is with ditransitive verbs such as *give* and *donate*. *Give* is coded as [D1(*to*)] which allows us to recover the information that this verb permits dative movement and requires a prepositional phrase headed by "to":

(Takes NP NP ToPP) and (Takes NP NP NP).

On the other hand, donate is coded [T1 (*to*)], which tells us that it does not undergo dative movement but does require a prepositional phrase headed by "to":

(Takes NP NP ToPP).

There are many more distinctions which are conveyed by the conjunction of grammar codes and word qualifiers (see Michiels, 1982, for further details). However, exploiting this information to the full would be a non-trivial task, because it would require accessing the relevant knowledge about the words contained in the qualifier fields from their LDOCE entries.

5 Lexical Entries For PATR-II

The output of the transformation program can be used to derive entries which are appropriate for particular grammatical formalisms. To demonstrate that this is possible we have implemented a system which constructs dictionary entries for the PATR-II system (Shieber, 1984 and references therein). PATR-II was chosen because it has been reimplemented in Cambridge and was therefore, available; however, the task would be nearly identical if we were constructing entries for a system based on GPSG, FUG or LFG. We

```
word storm:
 w_sense => <head trans sense-no> = 1
            V TakesNP Dyadic


worddag storm:
 [cat: v
 head: [aux: false
      trans: [pred: storm
          sense-no: 1
              arg1: <DG15> = []
              arg2: <DG16> = []]]
syncat:[first: [cat: NP
               head: [trans: <DG15>]]
         rest: [first: [cat: NP
                       head:
                         [trans: <DG16>]]
                 rest: [first: lambda]]]]
```

Figure 11

intend to use the LDOCE source in the same way to derive most of the lexicon for the general purpose, morphological and syntactic parser we are developing. The latter employs a grammatical formalism based on GPSG; the comparatively theory neutral lexical entries that we construct from LDOCE should translate straightforwardly into this framework as well.

The PATR-II parsing system operates by unifying directed graphs (DGs); the completed parse for a sentence will be the result of successively unifying the DGs associated with the words and constituents of the sentence according to the rules of the grammar. The DG for a lexical item is constructed from its lexical entry whichcontains a set of templates for each syntactically distinct variant. Templates are themselves abbreviations for unifications which define the DG. For example, the basic entry and associated DG for the verb *storm* are illustrated in Figure 11.

The template Dyadic defines the way in which the syntactic arguments to the verb contribute to the logical structure of the sentence, while the template TakesNP defines what syntactic arguments *storm* requires; thus, the information that *storm* is transitive and that it is logically a two-place predicate is kept distinct. Consequently, the system can represent the fact that some verbs which take two syntactic arguments are nevertheless one-place predicates.

The modified version of PATR-II that we have implemented contains only a small dictionary and constructs entries automatically from restructured LDOCE entries for most verbs that it encounters. As well as carrying over the grammar codes, the PATR-II lexicon system has been modified to include word senses numbers, which are derived from LDOCE. Thus, the analysis of a sentence by the PATR-II system now represents its syntactic and logical structure and the particular senses of the words (as defined in LDOCE) which are relevant in the grammatical context. Figures 12 and 13 illustrate the dictionary entries for *marry* and *persuade* constructed by the system from LDOCE.

In Figure 14 we show one of the two analyses produced by PATR-II for a sentence containing these two verbs. The other analysis is syntactically and logically identical but incorporates sense two of *marry*. Thus, the output from this version of PATR-II represents the information that further semantic analysis need only consider sense two of *persuade* and sense one and two of *marry*; this rules out one further sense of each, as defined in LDOCE.

6 Evaluation

The utility of the work reported above rests ultimately on the accuracy of the lexical entries which can be derived from the LDOCE tape. We have not attempted a systematic analysis of the entries which would result if the decompacting and grammar code transformation programs were applied to the entire dictionary. In Section 3 we outlined some of the errors in the grammar

marry *v* **1** [T1; I0] to take (a person) in marriage: *He married late in life / never married.* | (fig.) *She married money* (= a rich man) **2** [T1] (of a priest or official) to perform the ceremony of marriage for (2 people): *An old friend married them* **3** [T1 (*to*)] to cause to take in marriage: *She wants to marry her daughter to a rich man*

```
(marry
 ((Sense 1)
  ((Takes NP NP) (Type 2))
  ((Takes NP) (Type 1)))
 ((Sense 2)
  ((Takes NP NP) (Type 2)))
 ((Sense 3)
  ((Takes NP NP PP) (Type 3))))


word marry:
 w_sense ⇒
   <head trans sense-no> = 1
   V TakesNP Dyadic
 w_sense ⇒
   <head trans sense-no> = 1
   V Takes IntransNP Monadic
 w_sense ⇒
   <head trans sense-no> = 2
   V TakesNP Dyadic
 w_sense ⇒
   <head trans sense-no> = 3
   V TakesNPPP Triadic
```

Figure 12

codes which are problematic for the decompacting stage. However, mistakes or omissions in the assignment of grammar codes represent a more serious problem. While inconsistencies or errors in the application of the grammar coding system in some cases can be rectified by the gradual refinement of the decompacting program, it is not possible to correct errors of omission or assignment automatically. On the basis of unsystematic evaluation, using the programs to dynamically produce entries for the PATR-II parsing system, a number of errors of this type have emerged.

For example, the LDOCE definitions and associated code fields in Figure 15, demonstrate that *upset*(3) needs *it* + D5 which would correspond to its use with a noun phrase and a sentential complement; *suppose*(2) is missing optional "to be" for the X1 and X7 codes listed; *help*(1) needs a T3 code since it does not always require a direct object as well as an infinitive complement; and *detest* needs a V4 code because it can take a direct object as well as a gerund complement.

It is difficult to quantify the extent of this problem on the the basis of enumeration of examples of this type. Therefore, we have undertaken a limited test of both the accuracy of the assignment of the LDOCE codes in the source dictionary and the reliability of the more ambitious (and potentially controversial) aspects of the grammar code transformation rules. It is not clear, in particular, that the rules for computing semantic types for verbs are well enough motivated linguistically or that the LDOCE lexicographers were sensitive enough to the different transformational potential of the various classes of verbs to make a rule such as our one for Object Raising viable.

We tested the classification of verbs into semantic types using a verb list of 139 pre-classified items drawn from the lists published in Rosenbaum (1967) and Stockwell et al. (1973). Figure 16 gives the number of verbs classified under each category by these authors and the number successfully classified into the same categories by the system.

The overall error rate of the system was 14%; however, as the table illustrates, the rules discussed above classify verbs into Subject Raising, Subject Equi and

persuade *v* **1** [T1 (*of*); D5] to cause to feel certain; CONVINCE: *She was not persuaded of the truth of his statement* **2** [T1(*into, out of*); V3] to cause to do something by reasoning, arguing, begging, etc.: *try to persuade him to let us go with him.* | *Nothing would persuade him.* | .....

```
(persuade
 ((Sense 1)
  ((Takes NP NP) (Type 2))
  ((Takes NP NP SBar)
   (Type 3)))
 ((Sense 2)
  ((Takes NP NP) (Type 2))
  ((Takes NP NP Inf)
   (Type 3 ObjectEqui))))


word persuade:
 w_sense ⇒
   <head trans sense-no> = 1
   V TakesNP Dyadic
 w_sense ⇒
   <head trans sense-no> = 1
   V TakesNPSBar Triadic
 w_sense ⇒
   <head trans sense-no> = 2
   V TakesNP Dyadic
 w_sense ⇒
   <head trans sense-no> = 2
   V TakesNPInf
   ObjectControl Triadic
```

Figure 13

```
parse> uther might persuade gwen to marry cornwall

[cat: SENTENCE
 head: [form: finite
        agr: [per: p3 num: sg]
        aux: true
      trans: [pred: possible
          sense-no: 1
              arg1: [pred: persuade
                 sense-no: 2
                     arg1: [ref: uther sense-no: 1]
                     arg2: [ref: gwen sense-no: 1]
                     arg3: [pred: marry
                        sense-no: 2
                            arg1: [ref: gwen sense-no: 1]
                            arg2: [ref: cornwall
                              sense-no: 1]]]]]]
```

Figure 14

Object Equi very successfully. The two Subject Raising verbs which were not so classified by the system were *come about* and *turn out*. *Come about* is coded I5 in LDOCE, but is not given any word qualifier; *turn out* is not given any I5 code. These are clear examples of omissions on the part of the Longman lexicographers, rather than of failure of the rule. Similarly, *trust* is not recognised as an Object Equi verb, because its dictionary entry does not contain a V3 code; this must be an omission, given the grammaticality of

(6) *I trust him to do the job.*

Prefer is misclassified as Object Raising, rather than as Object Equi, because the relevant code field contains a T5 code, as well as a V3 code. The T5 code is marked as 'rare', and the occurrence of *prefer* with a tensed sentential complement, as opposed to with an infinitive, is certainly marginal:

upset ... **3** [T1] to cause to worry, not to be calm, etc.: .....

suppose ... **2** [T5a,b; V3 *often pass.*; X1,7,9] to believe: *I suppose that's true.* | *I supposed him to be a workman, but he was in fact a thief.* | *He was commonly supposed (to be) foolish* .....

help ... **1** [T1; I0; V3, (*esp AmE*) 2] to do part of the work (for someone); be of use to (someone in doing something); AID, ASSIST: *Could you help me up (the stairs)?* | *The stick helps him (to) walk.* | *Your sympathy helps a lot.* | *Can I help (with your work)?* .....

detest ... [T1,4] to hate with very strong feeling: *I detest people who deceive and tell lies.* | *They detest all that shooting and killing.* .....

Figure 15

(7) *I prefer that he come on Monday.*

(8) *?I prefer that he marries Julie.*

This example also highlights a deficiency in the LDOCE coding system since *prefer* occurs much more naturally with a sentential complement if it collocates with a modal such as "would". This deficiency is rectified in the verb classification system employed by Jackendoff and Grimshaw (1985) in the Brandeis verb catalogue.

The main source of error comes from the misclassification of Object Raising into Object Equi verbs. Arguably, these errors also derive mostly from errors in the dictionary, rather than a defect of the rule. 66% of the Object Raising verbs were misclassified as Object Equi verbs, because the cooccurrence of the T5 and V (2, 3, or 4) codes in the same code fields, as predicted by the Object Raising rule above, was not confirmed by LDOCE. All the 14 verbs misclassified contain V codes and 10 of these also contain T5 codes. However, the Longman lexicographers typically define two different word senses, one of which is marked (perhaps among other codes) T5 and the other with a V code. Analysis of

	Published lists	Derived from LDOCE	%
SEqui	31	31	100%
OEqui	58	56	97%
SRaising	7	5	71%
ORaising	42	28	67%

Figure 16

acknowledge ... **1** [T1,4,5 (*to*) to agree to the truth of; recognise the fact or existence (of): *I acknowledge the truth of your statement.* | *They acknowledged (to us) that they were defeated* | *They acknowledged having been defeated* **2** [T1 (*as*); X (*to be*) 1,7] to recognise, accept, or admit (as): *He was acknowledged to be the best player.* | *He was acknowledged as their leader.* | *They acknowledged themselves (to be) defeated* .....

hear ...*v* **1** [Wv6; T1; V2,4; I0] to receive and understand (sounds) by using the ears: *I can't hear very well.* | *I heard him say so.* | *I can hear someone knocking* **2** [Wv6; T1,5a] to be told or informed: *I heard that he was ill* — compare HEAR ABOUT, HEAR FROM, HEAR OF .....

Figure 17

these word senses suggests that this approach is justified in three cases, but unmotivated in five; for example, *hear* (1),(2) (justified) vs. *acknowledge* (1),(2) (unjustified) (see Figure 17). The other four cases we interpreted as unmotivated were *show, suspect, know, confess* and in the case of *consider*(2), (Figure 18) there is a clear omission of a T5 code, as demonstrated by the grammaticality of

(9) *I consider that it is a great honour to be here.*

Similarly, *expect* is not given a V3 code under sense 1 (Figure 19), however the grammaticality of

(10) *I expect him to pass the exam*

with the relevant interpretation suggests that it should be assigned a V3 code. Alternatively, sense 5, which is assigned a V3 code, seems suspiciously similar to sense 1.

The four verbs which are misclassified as Object Equi and which do not have T5 codes anywhere in their entries are *elect, love, represent* and *require*. None of these verbs take sentential complements and therefore they appear to be counterexamples to our Object Raising rule. In addition, Moulin et al. (1985) note that our Object Raising rule would assign *mean* to this category incorrectly. *Mean* is assigned both a V3 and a T5 category in the code field associated with sense 2 (i.e. "intend"), however, when it is used in this sense it must be treated as an Object Equi verb.

This small experiment demonstrates a number of points. Firstly, it seems reasonable to conclude that the assignment of individual codes to verbs is on the whole relatively accurate in LDOCE. Of the 139 verbs tested, we only found code omissions in 10 cases. Secondly though, when we consider the interaction between the assignments of codes and word sense classification, LDOCE appears less reliable. This is the primary source of error in the case of the Object Raising rule. Thirdly, it seems clear that the Object Raising rule is straining the limits of what can be reliably extracted from the LDOCE coding system. Ideally, to distinguish between raising and equi verbs, a number of syntactic criteria should be employed (Perlmutter and Soames, 1979:460ff.). However, only two of these criteria are explicit in the coding system.

On the basis of the results obtained, we explored the possibility of modifying the Object Raising rule to take account of the cooccurrence of T5 and T5a codes and V or X codes within a homograph, rather than within a word sense. An exhaustive search of the dictionary produced 24 verbs coded in this fashion. Ten of these were listed as Object Raising verbs in the published lists used in the above experiment. Five more verbs were classified as Equi in the published lists. Of the remaining nine verbs which did not appear in the published lists, three were clearly Object Raising, one was clearly Equi, a further two were probably Object Raising, and the last three were very difficult to classify. This demonstrates that modifying the Object Raising rule in this fashion would result in the misclassification of some Equi verbs. In fact, the list is sufficiently small that this set of verbs is probably best coded by hand.

As a final test, we ran the rules for determining the semantic type of verbs over all the 7,965 verb entries in LDOCE. There are 719 verb senses which are coded in the dictionary as having the potential for predicate complementation. Of these 5 were classified as Subject Raising, 53 as Object Raising, 377 as Subject Equi, and 326 as Object Equi by our rules. 42 of the Equi verbs are ambiguous between Subject and Object Equi under the same sense; in the transformation program this ambiguity is resolved by selecting the type appropriate for each individual code. For example, a code which translates

consider ... **2** [Wv5, X (*to be*) 1,7; V3] to regard as; think of in a stated way: *I consider you a fool* (= I regard you as a fool). | *I consider it a great honour to be here with you today.* | *He said he considered me (to be) too lazy to be a good worker.* | *The Shetland Islands are usually considered a part of Scotland* .....

Figure 18

expect ... **1** [T3,5a,b] to think (that something will happen): *I expect (that) he'll pass the examination.* | *He expects to fail the examination.* | *"Will she come soon?" "I expect so."* ..... **5** [V3] to believe, hope and think (that someone will do something): *The officer expected his men to do their duty in the coming battle* .....

Figure 19

as (Takes NP Inf) would select Subject Equi, while (Takes NP NP Inf) would select Object Equi. These sets of verbs together with the relevant LDOCE sense number are listed in the appendix. An exhaustive analysis of the 54 verbs classified as Object Raising revealed two further errors of inclusion in this set; *order*(6) should be Object Equi and *promise*(1) should be Subject Equi. The 42 verbs which the transformation program treats as ambiguous Equi verbs appear to be somewhat heterogeneous; some, such as *want*(1) and *ask*(2), are cases of 'Super-Equi' control verbs where the control relations are determined contextually, whilst others, particularly the phrasal verbs, appear to be better classified as Object Raising. *Allow*(1) and *permit*(1) appear here incorrectly because they are coded [T4] to capture examples such as

(11) *They do not allow / permit smoking in their house.*

In this example the subject of the progressive complement is not controlled by the matrix subject. Again, since the list is small, this set of verbs should probably be coded by hand.

7 Conclusion

Most applications for natural language processing systems will require vocabularies substantially larger than those typically developed for theoretical or demonstration purposes and it is often not practical, and certainly never desirable, to generate these by hand. The evaluation of the LDOCE grammar coding system suggests that it is sufficiently detailed and accurate (for verbs) to make the on-line production of the syntactic component of lexical entries both viable and labour saving. However, the success rate of the programs described above in producing useful lexical entries for a parsing system depends directly on the accuracy of the code assignments in the source dictionary. Correcting the mistakes and omissions in these assignments would be a non-trivial exercise. This is part of the motivation for adopting the interactive, rather than batch mode, approach to using the tape for lexicon development. We envisage eventually using the system to generate lexical entries in a semi-automatic fashion, allowing the user to intervene and correct errors during the actual process of constructing lexical entries, so that gradually a reliable and relatively error-free large lexicon for automated natural language processing systems containing detailed grammatical information can be constructed from LDOCE.

Clearly, there is much more work to be done with LDOCE in the extension of the use of grammar codes and the improvement of the word sense classification system. Similarly, there is a considerable amount of information in LDOCE which we have not exploited systematically as yet; for example, the box codes, which contain selection restrictions for verbs or the subject codes, which classify word senses according to the Merriam-Webster codes for subject matter (see Walker and Amsler (1983) for a suggested use for these). The large amount of semi-formalised information concerning the interpretation of noun compounds and idioms also represents a rich and potentially very useful source of information for natural language processing systems. In particular, we intend to investigate the automatic generation of phrasal analysis rules from the information on idiomatic word usage.

In the longer term, it is clear that neither the contents nor form of any existing published dictionary meet all the requirements of a natural language processing system. A substantial component of the research reported above has been devoted to restructuring LDOCE to make it more suitable for automatic analysis. However, even after this process much of the information in LDOCE remains difficult to access, essentially because it is aimed at a human reader, as opposed to a computer system. This suggests that the automatic construction of dictionaries from published sources intended for other purposes will have a limited life unless lexicography is heavily influenced by the requirements of automated natural language analysis. In the longer term, therefore, the automatic construction of dictionaries for natural language processing systems may need to be based on techniques for the automatic analysis of large corpora (eg. Leech et al., 1983). However, in the short term, the approach outlined in this paper will allow us to produce a relatively sophisticated and useful dictionary rapidly.

8 Acknowledgements

We would like to thank the Longman Group Limited for kindly allowing us access to the LDOCE typesetting tape for research purposes. We also thank Steve Pulman, Graham Russell and Karen Sparck Jones for their comments on the first draft, which substantially improved this paper. Part of the research reported here was funded by the UK Science and Engineering Research Council (Grant No. GR/D/05554) under the Alvey Programme. This paper is a substantially revised and updated version of an earlier presentation at the Second Conference of the European Chapter of the ACL.

References

Akkerman, Erik; Masereeuw, Pieter; and Meijs, Willem. 1985 *Designing a Computerised Lexicon for Linguistic Purposes*. ASCOT Report No. 1, CIP-Gegevens Koninklijke Bibliotheek, Den Haag, Netherlands.

Akkerman, Erik. 1986 A Critical Assessment of the LDOCE Coding System. To appear in: Akkerman, E.; Masereew, P.; and Meijs, W., Eds., *ASCOT Report No 2*, CIP-Gegevens Koninklijke Bibliotheek, The Hague, Netherlands.

Alshawi, Hiyan; Boguraev, Branimir; and Briscoe, Ted. 1985 Towards a Lexicon Support Environment for Real Time Parsing. In *Proceedings of the Second Conference of the European Chapter of the Association for Computational Linguistics*, Geneva, Switzerland: 171-178.

Alshawi, Hiyan. 1987 Processing Dictionary Definitions with Phrasal Pattern Hierarchies. In this issue.

Boguraev, Branimir. 1986 (and forthcoming) Machine Readable Dictionaries and Research in Computational Linguistics. In *Proceedings of a Workshop on Automating the Lexicon*, Grosseto, Italy (to

be published as Walker,D. and Zampolli,A., Eds., *Automating the Lexicon in a Multilingual Environment,* Cambridge University Press, Cambridge, UK).

Boguraev, Branimir. 1987 A Natural Language Toolkit: Reconciling Theory with Practice. In *Proceedings of a Workshop on Word Order and Parsing in Unification Grammars,* Friedenweiler, Germany (to be published as Reyle,U. and Rohrer,C., Eds., "*Word Orders, Parsing, and Unification Grammars*" D. Reidel, Dordrecht, Holland).

Boguraev, Branimir; Carter, David and Briscoe, Ted. 1987 A Multi-Purpose Interface to an On-line Dictionary. In *Proceedings of the Third Conference of the European Chapter of the Association for Computational Linguistics,* Copenhagen, Denmark: 63–69.

Byrd, Roy. 1983 Word Formation in Natural Language Processing Systems. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* Karlsruhe, Germany: 704-706.

Calzolari, Nicoletta. 1984 Machine-Readable Dictionaries, Lexical Data Bases and the Lexical System. In *Proceedings of the 10th International Congress on Computational Linguistics,* Stanford, California: 460-461.

Carter, David. 1987 An Information Theoretic Analysis of Phonetic Dictionary Access, *Computer Speech and Language*, 2:1–11.

Gazdar, Gerald; Klein, Ewan; Pullum, Geoffrey; and Sag, Ivan. 1985 *Generalized Phrase Structure Grammar.* Blackwell, Oxford, UK.

Heidorn, George et al. 1982 The EPISTLE Text-Critiquing System. *IBM Systems Journal,* 21(3): 305-326.

Huttenlocher, Daniel and Zue, Victor. 1983 Phonotactic and Lexical Constraints in Speech Recognition, In *Proceedings of the National Conference on Artificial Intelligence,* Washington, D.C.: 172-176.

Ingria, Robert. 1984 Complement Types in English. Report No. 5684, Bolt Beranek and Newman Inc., Cambridge, Mass.

Jackendoff, Ray and Jane Grimshaw. 1985 A Key to the Brandeis Verb Catalog. Unpublished mimeo, under NSF Grant IST-84-20073, "Information Structure of a Natural Language Lexicon", Program in Linguistics and Cognitive Science, Brandeis University, Waltham, Mass.

Kaplan, Ronald and Bresnan, Joan. 1982 Lexical-Functional Grammar: A Formal System for Grammatical Representation. In: J.Bresnan, Ed., *The Mental Representation of Grammatical Relations.* The MIT Press, Cambridge, Mass: 173-281.

Kay, Martin. 1984a Functional Unification Grammar: A Formalism for Machine Translation. In *Proceedings of the 10th International Congress on Computational Linguistics,* Stanford, California: 75-79.

Kay, Martin. 1984b The Dictionary Server. In *Proceedings of the 10th International Congress on Computational Linguistics,* Stanford, California, 461-462.

Leech, Geoffrey; Garside, Roger; and Atwell, Erik. 1983 The Automatic Grammatical Tagging of the LOB Corpus. Bulletin of the International Computer Archive of Modern English, Norwegian Computing Centre for the Humanities, Bergen, Norway.

Michiels, Archibal. 1982 Exploiting a Large Dictionary Data Base. PhD Thesis, Université de Liège, Liège, Belgium.

Michiels, Archibal. 1983 Automatic Analysis of Texts. In *Informatics 7, Proceedings of a Conference of the ASLIB Informatics Group and the Information Retrieval Group of the British Computer Society,* Cambridge, UK: 103-120.

Moulin, A.; Jansen, J; and Michiels, A. 1985 Computer Exploitation of LDOCE's Grammatical Codes, paper presented at a Conference on Survey of English Language, Lund.

Perlmutter, D.M. and Soames, S. 1979 *Syntactic Argumentation and the Structure of English.* University of California Press, Berkeley, California.

Phillips, John and Thompson, Henry. 1986 A Parser for Generalised Phrase Structure Grammars. To apper in Klein, E. and Haddock, N., Eds., *Edinburgh Working Papers in Cognitive Science,* University of Edinburgh, Edinburgh, Scotland.

Procter, Paul. 1978 *Longman Dictionary of Contemporary English.* Longman Group Limited, Harlow and London, England.

Quirk, Randolph; Greenbaum, Sidney; Leech, Geoffrey; and Svartvik, Jan. 1972 *A Grammar of Contemporary English,* Longman Group Limited, Harlow and London, England.

Quirk, Randolph; Greenbaum, Sidney; Leech, Geoffrey; and Svartvik, Jan. 1985 A Comprehensive Grammar of English, Longman Group Limited, Harlow and London, England.

Robinson, Jane. 1982 DIAGRAM: A Grammar for Dialogues. *Communications of the ACM,* 25(1): 27-47.

Rosenbaum, P.S. 1967 *The Grammar of English Predicate Complement Constructions.* MIT Press, Cambridge, Mass.

Russell, Graham; Pulman, Steve; Ritchie, Graeme; and Black, Alan. 1986 A Dictionary and Morphological Analyser for English. In *Proceedings of the Eleventh International Congress on Computational Linguistics,* Bonn, Germany: 277-279.

Sager, N. 1981 *Natural Language Information Processing,* Addison-Wesley, Reading, Mass.

Shieber, S. 1984 The Design of a Computer Language for Linguistic Information, In *Proceedings of the 10th International Congress on Computational Linguistics,* Stanford, California: 362-366.

Stockwell, R.P.; Schachter, P.; and Partee, B.H. 1973 *The Major Syntactic Structures of English.* Holt, Rinehart and Winston, New York, New York.

Tompa, Frank. 1986 Database Design for a Dictionary of the Future. Preliminary Report, Centre for the New Oxford English Dictionary, University of Waterloo, Waterloo, Ontario.

Walker, D. and Amsler, A. 1986 The Use of Machine-Readable Dictionaries in Sublanguage Analysis. In: R. Grishman and R. Kittredge, Eds., *Analysing Language in Restricted Domains,* Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Williams, E.S., 1980 Predication. *Linguistic Inquiry,* 11(2): 203-238.

Appendix

1. Subject Raising verbs (total number 5)

appear (3)	chance (1)	happen (3)	seem (2)	transpire (2)

2. Object Raising verbs (total number 53)

adjudge (1)	admit (3)	allow (5)	argue (3)	assert (1)
assume (1)	avow (1)	believe (3)	betray (3)	certify (2)
declare (2)	deem (1)	deny (1)	determine (1)	discover (2)
engage (4)	feel (5)	find (8)	foreordain (1)	guess (1)
hold (9)	judge (3)	maintain (5)	make out (5)	mean (2)
mind (2)	notice (1)	observe (1)	order (6)	perceive (1)
predicate (1)	prefer (1)	preordain (1)	presume (1)	presume (2)
proclaim (1)	pronounce (2)	pronounce (3)	prove (1)	recognize (3)
remember (1)	report (1)	reveal (2)	see (2)	smell (2)
smell (3)	suppose (1)	suppose (2)	tell (6)	think (2)
understand (3)	understand (4)	warrant (2)		

3. Subject Equi verbs (total number 335)

abide (1)
advocate (1)
aim at (1)
aspire (1)
awake (1)
begrudge (1)
bother (3)
bust out (3)
claim (4)
come (1)
come into (1)
compete (1)
conduce to (1)
consent (1)
contemplate (2)
contrive (3)
dare (1)
declare for (1)
delay (1)
deride (1)
discontinue (1)
dread (1)
enjoy (1)
excuse (1)
fall to (1)
forbear (1)
frown on (1)
get down to (1)
go about (2)
grow (5)
guard against (1)
hinge on (1)
indulge in (1)
justify (1)
knock off (2)
leave (7)
look forward to (1)
merit (1)
neglect (1)
operate (2)
petition (2)
play at (2)
postpone (1)
preclude (1)
pretend (2)
prohibit (1)
provide for (3)
recall (1)
rejoice (1)
resent (1)
resort to (1)
rise above (1)
scorn (1)
seem (1)
send away (2)
shirk (1)
sicken of (1)
start (1)
stop (1)
swear by (1)
tend (2)
tremble (3)
undertake (2)
volunteer (1)
wait (1)
witness to (1)
account for (2)
affect (1)
allude to (1)
assent (1)
bear (5)
bid fair (1)
break off (1)
care (1)
clamour (2)
come (7)
come on (1)
conceal (1)
confess (1)
consider (1)
continue (1)
could (1)
dare (2)
decline (3)
delight (2)
descend to (1)
discourage (2)
duck out of (1)
envisage (1)
expect (1)
finish (1)
forbid (2)
funk (1)
get out of (1)
go in for (2)
grow out of (2)
happen (2)
hit on (1)
inveigh against (1)
keep (11)
know about (1)
like (2)
make (18)
militate magainst (1)
neglect (2)
own to (1)
pine (3)
pledge (1)
practise (4)
prepare (3)
pretend (4)
promise (3)
purport (1)
reckon on (2)
relish (1)
resist (1)
result from (1)
risk (2)
scramble (2)
see (7)
send off (3)
should (1)
smile (2)
stem from (1)
strive (1)
swear off (1)
think of (1)
trouble (3)
unite (2)
volunteer (2)
want (3)
wriggle out of (1)
ache (2)
afford (2)
anticipate (1)
attach to (3)
bear (9)
blanch (2)
burn (6)
cease (1)
clog (1)
come before (1)
come to (1)
conceive of (1)
confess (2)
consist in (1)
continue (3)
covenant (1)
decide (2)
defend (3)
delight in (1)
deserve (1)
disdain (2)
elect (2)
escape (3)
exult (1)
fix (2)
forget (1)
get (3,11)
get round to (1)
go on (5)
grow out of (3)
hasten (2)
hope (1)
involve (2)
keep from (2)
lament (1)
live by (1)
make up for (1)
miss (1,2,5)
negotiate (1)
pant (4)
plan (1)
plot (5)
practise (5)
prepare for (1)
proceed (1)
propose (1)
purpose (1)
recollect (1)
remember (2)
resist (2)
resume (1)
rue (1)
scream (4)
see about (1)
serve (5)
shrink from (1)
stand (8)
stick (8)
subscribe to (1)
swear to (1)
think of (5)
try (1)
use (1)
vote (1)
want (4)
write (4)
acknowledge (1)
agree (2)
appear (2)
attempt (1)
begin (1)
blink at (1)
burst (3)
chance (1)
close (3)
come down to (1)
commence (1)
concur (2)
confide (1)
conspire (1)
contract (1)
cut out (4)
decide on (1)
defy (3)
demand (1)
detest (1)
dislike (1)
endeavour (1)
essay (1)
exult over (2)
fix on (1)
forget about (1)
get around to (1)
give up (1)
go with (3)
grudge (1)
hate (3)
incline (4)
itch (3)
keep on at (1)
lead to (1)
loathe (1)
manage (2)
necessitate (1)
offer (3)
pay for (1)
play (3)
plump for (1)
prate about (1)
presume (4)
profess (2)
propose (2)
put off (1)
refuse (1)
repent (1)
resolve (1)
revel in (1)
say (5)
scruple (1)
see to (1)
set about (1)
shudder (1)
stand (12)
stoop (3)
suggest (2)
take to (2)
threaten (2)
try (2)
venture (2)
vouchsafe (1)
warrant (1)
write back (1)
adore (3)
aim (2)
arrange (2)
avoid (1)
beg (3)
blush (2)
burst out (1)
choose (2)
cloud (3)
come out against (1)
compare with (1)
condescend (1)
connive (1)
conspire (2)
contrive (1)
cry out against (1)
declare against (1)
deign (1)
depose (2)
disclaim (1)
do with (1,2)
endure (1)
evade (2)
fail (1,3)
flick (2)
forswear (1)
get away with (1)
go (5)
go without (1)
guarantee (2)
hesitate (1)
include (1)
jib at (1)
kick against (1)
learn (1)
long (1)
mean (5)
need (1)
omit (2)
pertain to (1)
play at (1)
pooh-pooh (1)
pray (1)
pretend (1)
profit by (1)
provide for (2)
quit (1)
regret (1)
require (1)
resolve (2)
revert to (1)
scheme (1)
seek (3)
send (4)
set out (2)
shun (1)
stand for (2)
stoop to (1)
swear (1)
take up (1)
train (3)
try (3)
venture (4)
vow (1)
watch (3)
yearn (1)

4. Object Equi verbs (total number 284)

acknowledge (2)
allure (1)
assist (1)
beckon (1)
bill (2)
bring (5)
call on (2)
challenge (4)
come down on (1)
condemn (4)
consider (2)
cozen into (1)
dedicate to (1)
deputize (2)
doom (1)
embolden (1)
enable (1)
enggae in (1)
esteem (2)
fancy (3)
forbid (1)
get (8)
habituate to (1)
help (1)
importune (1)
influence (1)
instruct (3)
invite (2)
keep from (1)
legislate for (1)
let off (1)
make (3)
motion (2)
nominate (2,4)
organize (1)
phone (1)
plume upon (1)
predetermine (1,3)
prepare for (3)
prevent (1)
programme (1)
provoke into (1)
put off (1)
reeducate (1)
represent (1.2)
schedule (1)
send (2)
show (1)
spur (2)
summons (1)
take (18)
telegraph (1)
tell (5)
timetable (1)
trouble (2)
watch (1)
yearn for (1)

adjure (1)
appoint (1)
attribute to (1)
behove (1)
bludgeon into (1)
bring in (3)
catch (3)
challenge (5)
command (1)
condition (3)
constrain (1)
credit with (1)
defy (2)
design (2)
dragoon into (1)
employ (1)
enable (2)
entice (1)
excite (2)
figure on (1)
force (1)
give (17)
hail as (1)
help (2)
impute to (1)
inhibit from (1)
intend (2)
invite (3)
know (4)
let (1)
long for (1)
make (5)
motion to (1)
notify (1)
overhear (1)
pick (1)
pray (3)
predispose (1)
press (9)
prevent from (1)
promise (1)
push (3)
put up to (1)
regard as (1)
represent as (1)
school (1)
send (3)
show (9)
spy (3)
summon (1)
talk into (1)
telephone (1)
tell off (2)
time (1)
unfit for (1)
watch (5)

advise (1)
arrange for (1)
authorize (1)
beseech (1)
bluff into (1)
bully (1)
cause (1)
charge (5)
commission (1)
confess (3)
cop (1)
dare (5)
delegate (2)
designate (2)
draw on (3)
employ (2)
encourage (1)
entitle (2)
exhort (1)
find (1)
frighten into (1)
give over to (1)
harden to (1)
hunger (1)
incite (1)
inspire (1)
introduce to (1)
itch for (1)
lead (2)
let (2)
look at (1)
make (6)
motivate (1)
obligate (1)
persuade (2)
pick on (1)
preclude from (1)
preen on (1)
pressure (1)
pride on (1)
prompt (1)
push on (2)
reckon (1)
rely on (2)
request (1)
seduce (2)
set (4)
signal (2)
steel (1)
supplicate (1)
talk out of (1)
telex (1)
tempt (1)
tip (1)
urge (2)
watch for (1)

aid (1)
ask (4)
badger (1)
bestir (1)
bribe (1)
buzz (3)
caution (1)
charge with (1)
compel (1)
conjure (1)
counsel (1)
debar from (1)
depend on (1)
detail (1)
drive (8)
employ in (1)
end in (1)
entreat (1)
expect (5)
find (6)
frighten out of (2)
goad into (1)
hark at (1)
impel (1)
incline (3)
instigate (2)
inure to (1)
join with in (1)
lead on (1)
let (3)
look to (2)
make (7)
move (11)
oblige (1)
pester (1)
plead with (1)
predestinate (1)
prepare (1)
pressurize (1)
profess (3)
prove (3)
put down as (1)
reckon on (1)
remember as (1)
require (2)
select (1)
set (8)
sign (2)
stop (2)
suppose (3)
tax with (1)
tell (2)
tempt (2)
tip off (2)
want (2)
wean from (1)

allow (2)
assign (4)
bargain for (1)
bid (2)
bring (2)
cable (1)
challenge (1)
charge with (2)
condemn (3)
connive at (1)
couple with (1)
decide (4)
depute (1)
direct (3)
egg on (1)
empower (1)
engage (1)
equip (2)
fancy (1)
fit (5)
get (4)
groom (4)
hear (1)
implore (1)
induce (1)
instruct (2)
inveigle into (1)
keep (10)
legislate against (1)
let (4)
lower (3)
mean (4)
name (3)
order (1)
petition (1)
pledge (2)
predestine (1)
prepare (5)
prevail upon (1)
program (1)
provoke (2)
put down to (1)
reduce to (4)
remind (1)
result in (1)
send (1)
shape (1)
slate (2)
suffer (4)
suspect (2)
teach (1)
tell (3)
thank (2)
train (2)
warn (1)
worry at (1)

5. Equi verbs (total number 42)

allow (1)
beg (2)
countenance (1)
hate (1)
imagine (1)
nag (1)
qualify (1)
see (6)
want (1)

allow for (1)
calculate on (1)
count on (1)
hate (2)
intend (1)
need (1)
race (3)
sign (3)
wish (5)

approve of (1)
chance (2)
culminate in (1)
hear about (1)
like (1)
pay (1)
recommend (2)
sign up (1)

ask (2)
choose (1)
desire (1)
hear of (1)
like (3)
permit (1)
rely on (1)
start (3)

bank on (1)
compensate for (1)
enjoin (1)
help (4)
love (2)
prepare (4)
save (4)
visualize (1)