LEXICAL ACQUISITION IN THE CORE LANGUAGE ENGINE

David M. Carter
SRI International Cambridge Research Centre
23 Millers Yard, Mill Lane
Cambridge CB2 1RQ, U.K.

Keywords: computational lexicography; lexical acquisition

ABSTRACT

The SRI Core Language Engine (CLE) is a general-purpose natural language front end for interactive systems. It translates English expressions into representations of their literal meanings. This paper presents the lexical acquisition component of the CLE, which allows the creation of lexicon entries by users with knowledge of the application domain but not of linguistics or of the detailed workings of the system. It is argued that the need to cater for a wide range of types of back end leads naturally to an approach based on eliciting grammaticality judgments from the user. This approach, which has been used to define a 1200-word core lexicon of English, is described and evaluated.

1 INTRODUCTION

The SRI Core Language Engine (CLE; Alshawi *et al*, 1988a,b) is a domain independent system for translating English sentences into formal representations of their literal meanings which are capable of supporting reasoning. It is designed to be used as a major component of interactive advisor systems such as interfaces to database management systems and diagnostic expert systems. The main contribution of the CLE is intended to be substantial coverage of English constructions in both syntax and semantics that is well motivated and hence extensible.

The CLE makes use of three main types of lexicon entry.

- A *syntactic* entry for a word consists of one or more complex categories, each specified by a principal category symbol augmented by a set of constraints on the values of syntactic features. Such categories also appear in the CLE's grammar, and matching and merging of the information encoded in them carried out by unification during parsing.
- *Word sense* entries for words are specified in the same way, but involve semantic as well as syntactic features. Semantic interpretation, which takes place in tandem with parsing, works by unification of feature values in word sense entries and semantic interpretation rules.
- *Sortal* (selectional) *restrictions* are defined for logical form predicates (i.e. word senses). After possible semantic interpretations are constructed, the CLE applies these restrictions with reference to a user-definable hierarchy of sortal classes, to reject any interpretations in which the sort expected by some argument

of a predicate is inconsistent with that of the object filling that argument.

The CLE lexical acquisition tool VEX (for Vocabulary EXpander) allows the creation of CLE lexicon entries by users with knowledge both of English and of the application domain, but not of linguistic theory or of the way lexical entries are represented in the CLE. It asks the user for information on the grammaticality of example sentences, and for selectional restrictions on arguments of predicates, and writes to disc a set of instructions that can immediately be used by the CLE to create appropriate lexical entries automatically in main memory.

2 THE TASK OF LEXICAL ACQUISITION

VEX's task is to aid in the creation of lexical entries that will allow the CLE to map certain English expressions into appropriate logical form predicates. These predicates are expected then to receive further application-specific processing. A crucial factor in designing VEX was that virtually no assumptions can be made about the nature of this subsequent processing or about the representations, if any, into which predicates will be mapped; indeed, the main use of VEX so far, one which suggests its viability, has been to construct the CLE's 1200-word core lexicon, which is intended to be application-independent.

This situation contrasts with that obtaining in, for example, the TEAM system (Grosz *et al*, 1987). Whereas the CLE is intended to interface to a range of back end systems, TEAM was designed specifically as a front end for databases of a particular kind. This means that lexical acquisition in TEAM is essentially a matter of determining the English counterparts of particular database relations, and that the possibilities for word behaviours are constrained by the kinds of relations that exist. Furthermore, TEAM's coverage of verb subcategorization is rather more limited than that of the CLE. Thus TEAM is able to allow the user to volunteer a sentence from which, with the help of some hard-wired auxiliary questions, it infers the syntactic and semantic characteristics of the way a verb and its arguments map into the database.

However, because of the CLE's wide syntactic coverage and the lack of constraints from any known application, it is too risky to allow the user to volunteer sentences to VEX. Instead, VEX itself presents example sentences to the user and asks whether or not they are acceptable. In addition, the logical forms produced are of a fairly neutral, conservative nature, and correspond one-to-one to the individual surface syntactic subcategorization(s) that are identified; for example, related usages like the transitive and intransitive uses of "break" ("John broke the window" vs. "The window broke") will be mapped onto different predicates, leaving it to the back end to make whatever it needs to of the relationship between them. Thus apart from eliciting selectional restrictions, virtually all of VEX's processing is done at the level of syntax.

3 THE STRATEGY ADOPTED

VEX adopts a *copy and edit* strategy in constructing lexical entries. It is provided with pointers to entries in a "paradigm" lexicon for a number of representative word usages and declarative knowledge of the range of sentential contexts in which these usages can occur. For example, it knows that a phrasal verb such as "rely on" that takes a compulsory prepositional phrase complement can be the main verb

in a sentence of the form "np *verb prep* np" (e.g. "John *relies on* Mary") but not in one of the form "np *verb* np *prep*" (e.g. *"John *relies* Mary *on*"). Entries in the paradigm lexicon are distinguished not only by the type and number of arguments they take, but also by phenomena such as "tough-movement", subject raising and equi-NP deletion. VEX elicits grammaticality judgments from the user to determine which paradigm (or set of paradigms) occurs in the same contexts as the word being defined, and then constructs the new entries by making substitutions in these paradigm entries. Each use of a paradigm will give rise to one distinct predicate.

An alternative to this copy and edit strategy would be a more detailed, knowledge-based method in which VEX was equipped with knowledge of the function of every feature and other construct in the representation, and asked the user questions in order to build entries in a bottom-up fashion. However, such an approach has several drawbacks.

The complexity of the representation would make a bottom-up approach unwieldy and time-consuming, both for the builder of VEX and for the user, who would have to answer an inordinately long list of questions for every new entry. Furthermore, interaction at the level of individual linguistic features would allow genuinely novel entries to be created, which, given that the user is a non-linguist, would almost certainly lead to inconsistencies. In addition, endowing VEX directly with knowledge of the representation would mean that as the representation developed, VEX would continually have to be updated.

The copy and edit approach, on the other hand, makes VEX independent of most changes to the representation. Furthermore, the fact that its knowledge is specified at the level of word behaviours, means that as the CLE's coverage increases, modifications to this knowledge are easy to make. It also makes robust and (relatively) succinct interaction with the user easier to achieve.

4 ASSUMPTIONS BEHIND THE STRATEGY

The appropriateness of VEX's strategy depends on a number of assumptions, including the following.

Firstly, it assumes that the syntactic behaviours of arbitrary words are describable in terms of a fixed, manageably small set of paradigms. The alternative view, which has been argued for by Gross (1975), is that in fact *every* word is in some way idiosyncratic. I offer no counterarguments to that position here, but merely observe that as far as copy-and-edit lexical acquisition is concerned, it is a counsel of despair; if every word has its peculiarities, then every lexical entry must be constructed from scratch by a trained linguist (either by hand or using a bottom-up lexical acquisition tool of the kind dismissed above for use by non-linguists). VEX's approach, on the other hand, can be expected to work if the *approximate* regularities that undoubtedly do exist are strong enough that the exceptions will not cause major problems, and this indeed seems to be the case for open class words. VEX does not attempt to deal with closed class words, as these are more idiosyncratic, and in any case are few enough for entries to be written for them by hand as part of the development of the CLE.

Secondly, however, even once we accept the use of a finite paradigm set, there is the question of what those paradigms are. One might at first think that paradigms would be represented by "typical" tran-

sitive verbs, count nouns and so on; but in fact, such typical words are very hard to find, because in practice almost every word has a range of behaviours that it shares with various other words. If we imagine an ideal hand-coded lexicon for the whole language, then the *entry* for a word will consist of a set of *categories*, each of which allows a number of syntactic *patterns* of use. The mappings between words and categories, and between categories and patterns, are both many-to-many; indeed, one category may allow the same set of patterns as a collection of other categories by virtue of leaving unspecified a feature value which the other categories collectively enumerate.

We define a *paradigm* as any minimal non-empty intersection of entries, or, equivalently, as any maximal set of categories with the same distribution among entries. That is, every category in a paradigm will occur in exactly the same set of entries in the ideal lexicon as every other category (if any) in that paradigm; and every entry will be a disjoint union of paradigms. The reason this "grain size" for paradigms is correct is as follows. Any smaller grain size would result in some pairs of paradigms always occurring together in entries, thereby multiplying the number of distinct predicate names and losing generality. A larger grain size, however, would mean that some words either could not be assigned a consistent set of paradigms, or would be assigned the same category more than once, leading to spurious multiple analyses.

The third assumption on which VEX's strategy is based is that judgments of grammaticality are to a large extent shared between speakers of the language and tend to be absolute, binary ones. Experience has shown, however, that different users have different intuitions, and even the same user can give different answers on different occasions. To deal with this problem, if VEX receives a set of judgments from which it cannot form a consistent paradigm set, it offers the user a choice of ways in which he can change his mind; this process of negotiation usually arrives at a satisfactory conclusion. The user can also choose to backtrack at any time.

In any case, although grammaticality judgments are sometimes variable and indeterminate, they are much less so than judgments of semantic acceptability, which do not play any part in VEX's main decision-making process. In order to remind the user to judge grammaticality rather than semantic well-formedness, VEX presents example sentences containing "nonsense nouns" such as "thingummy" and "whatsit".

5 ELICITING SYNTACTIC INFORMATION

The algorithm for defining a new word or phrase specified by the user is as described here; an example of its operation follows.

First, the user is asked for the part(s) of speech of the new item (noun, verb, etc; no further grammatical knowledge is assumed). The rest of the definition process takes place separately for each part of speech. VEX majors on verb and adjective definitions, and knows about only very gross distinctions between noun types (e.g. count vs. mass nouns), because other distinctions, notably that between relational and non-relational nouns, arguably have as much to do with pragmatics as with syntax and are therefore left for later back-end processing to deal with.

After determining any irregular inflectional forms, VEX elicits grammaticality judgments from the user. In the most recently released version of the system,

VEX knows about 52 different paradigms and their grammaticality in the context of 52 different sentential patterns.[1] Its task is to discover the behaviour of the new word or phrase by presenting as few example sentences to the user as possible, and then to find the minimal subset of the paradigms that between them account for that behaviour. The sets of paradigms and sentences are progressively reduced as follows.

- Paradigms for a different part of speech or number of words from those of the new phrase are eliminated.

- VEX removes sentence patterns which either do not correspond to any surviving paradigms, or whose grammaticality can be deduced from that of other patterns in the subset. For example: *if* sentence pattern S1 is grammatical when (and only when) a word or phrase with paradigm P1 is inserted in it; sentence pattern S2 is grammatical only for paradigm P2; and sentence pattern S3 is grammatical only for P1 and P2: *then* there is no point in presenting S3 to the user if S1 and S2 are also to be presented, because S3 will be grammatical when and only when either S1 or S2 (or both) are grammatical. Thus VEX orders the candidate sentence patterns according to the number of paradigms associated with them, and eliminates from the resulting list any patterns whose paradigm set is exactly the union of those of one or more later items.

- The remaining sentence patterns, with forms of the item being defined substituted in, are presented to the user, who states which of them are grammatical. Because the number of possible word behaviours is quite large, up to 18 sentences may be presented in this way; instead of immediately making a full choice, therefore, VEX allows the user to make a partial choice, and will then provide further guidance by specifying what paradigms might be implied by that choice, and what other sentences would need to be judged grammatical for those paradigms to be acceptable.

- Some of the user's approved sentences may be "false positives" in the sense that they are grammatical only by virtue of resulting from another grammatical sentence by an operation such as pronominalization or addition of an optional prepositional phrase. VEX detects any such sentence pairs and eliminates false positives, sometimes with reference to the user's answer to a yes/no question about any implications holding between the sentences.

- VEX then tries to find a minimal set of paradigms which, together, occur in all and only the contexts the user has marked as grammatical. At this point, one of the following occurs:

(a) There is exactly one minimal set. This set is accepted, and VEX moves on to consider semantic aspects of the new entry (see section 7 below).

(b) There are no minimal sets, because every set of paradigms that together allows the sentences the user has said are grammatical also allows a sentence that was (by implication) judged ungrammatical. This occurs quite often because users frequently ignore sentences, misread them, or simply have different intuitions on them from those embodied in the CLE's data. VEX responds by asking the user to accept one of several additions to, or deletions from, the grammatical set. The user may either accept a revision or reconsider his assumptions and backtrack to some earlier point in the dialogue. The backtracking mechanism is in fact available throughout a VEX session, and allows the user to restart the dialogue from a range of earlier points.

[1] The equality of these numbers is coincidental.

(c) There are several minimal sets of the same size. In this case, VEX prefers less ambiguous sets, i.e. those that minimize the number of occasions that two paradigms in the set *both* account for the grammaticality of a sentence (and hence could lead to apparent ambiguity in parsing). If this does not select a unique paradigm set, VEX chooses a set at random and warns the user of the conflict; such conflicts almost always result from VEX being unable to separate two distinct behaviours for a phrase, a situation which can be remedied by the user presenting the behaviours to VEX in two separate dialogues.

6 AN EXAMPLE

Suppose the user wishes to define the phrasal verb "use up". After morphological information has been supplied, VEX presents the following list of sentences:

```
1 The thingummy used up.
2 The thingummy used the whatsit
    up.
3 The whatsit was used up by the
    thingummy.
4 The thingummy used the boojum
    up very good.
5 The boojum was used up the
    whatsit by the thingummy.
6 The whatsit was used up for the
    boojum by the thingummy.
7 The thingummy used up existing.
8 The thingummy used up the whatsit
    that the boojum existed.
9 The whatsit was used up by the
    thingummy to exist.
```

and invites the user to specify which ones are grammatical in the domain in question. The user would approve sentences 2, 3 and 9 only. VEX then considers the possibility that, because sentence 3 is grammatical, sentence 9 is grammatical only when "to exist" is an optional modifier. This is in fact the case. It asks the user:

```
Does "the whatsit was used up by
      the thingummy to exist"
necessarily imply
  "the whatsit was used up by the
   thingummy IN ORDER TO exist"?
```

When the user answers affirmatively, sentence 9 is dropped from consideration. (Contrast the case of "call on", where "The board called on the chairman to resign" can mean something quite different from "The board called on the chairman in order to resign").

VEX now has enough information to decide that "use up" behaves syntactically as a transitive particle verb.

7 ELICITING SEMANTIC INFORMATION

Once a set of paradigms has been established, VEX asks for a name for the predicate corresponding to each one, and then for sortal restrictions on the predicate and its arguments. Sortal restrictions may be given to VEX directly as a list (interpreted conjunctively) of atoms occurring in the sort hierarchy currently in force, or indirectly as a pointer to sortal restrictions on another predicate or one of its arguments. If an explicit list is provided, they are checked for existence in the sort hierarchy currently in force and for mutual consistency in terms of that hierarchy (e.g. the list "male female" would normally be rejected), but no check is made for the existence of other predicates referred to, since these may not yet have been defined or incorporated into the system.

VEX allows the user to specify any number of alternative sets of restrictions on a predicate. However, the use of more than one set is discouraged, because if the

alternative restrictions are assigned to distinct predicates then the CLE will be able to provide the back-end system with more information than would otherwise be possible.

8 FURTHER PROCESSING

When selectional restrictions have been acquired, VEX writes out to disc a set of "implicit" lexical entries. Implicit lexical entries are instructions interpreted by CLE code that makes substitutions, for words and predicate names, in entries for the paradigms that VEX knows about. The results of these substitutions are explicit, feature-based entries, which are then compiled directly into the format used by the parser itself. Both expansion and compilation happen automatically and are hidden from the user; thus as soon as a word is defined with VEX, it can be used in an input sentence.

The are three main advantages in introducing this "implicit" level of representation. Firstly, implicit entries are much smaller than explicit and compiled ones, which results in considerable saving of space since the latter are only generated on demand. Secondly, if the paradigm entries are later changed, for example because of developments in the feature system, existing implicit entries will usually not need to be altered; their explicit and compiled forms will automatically come to reflect those of the paradigm entries when the system is recompiled. This has occurred many times during the development of the CLE. Thirdly, implicit entries are also rather shorter than explicit ones and are therefore easier to edit by hand where desired. Hand editing is appropriate on those occasions when VEX has not quite produced the desired results, either because of peculiarities in the phrase being defined, or more commonly because the user changes his mind about what detailed responses to VEX are appropriate (for example, changing a predicate name) and does not wish to redefine the phrase from scratch. It can also be useful if, for example, the sort hierarchy is extended after some entries have been defined, and it is necessary to update the sortal restrictions on those entries to take full advantage of the extension.

9 SUMMARY AND CONCLUSIONS

The application-independence of the CLE leads to a style of lexical acquisition different from that of earlier, dedicated natural-language front ends. I have argued for a technique based on a limited number of syntactic paradigms, a subset of which are selected for the construction of new entries according to the user's judgments of sentence grammaticality. This allows the lexical acquisition component to avoid strong dependencies on the CLE's linguistic representation, the application domain, the nature of the back end system, or the user's knowledge of linguistics.

VEX's concentration on syntactic paradigms allows a wide range of subcategorization types to be recognised and dealt with, and also permits a non-trivial lexicon to be easily maintained while the system is under development. The use of VEX to define the CLE's 1200 word core lexicon is evidence for the practicality of the approach.

The crucial factor in evaluating VEX, however, is its acceptability to the non-linguist (but application-expert) users for whom it was designed. No formal evaluation of this has been carried out, but informal feedback from members of the companies to whom a version of the CLE was delivered in the summer of 1988 has been

generally positive. It appears that, once they have studied the annotated VEX session transcript distributed with the CLE documentation, those who have so far used the system have had no great difficulty with the idea of using nonsense words or with concepts such as grammaticality and paradigms.

Perhaps the most difficult task faced by the VEX user is to decide which of the sentences presented are grammatical; however, this task is significantly eased by the possibility of backtracking, by the consistency checker, and by the partial choice facility, all of which were implemented in response to comments by users of earlier versions of the system. The difficulties that remain seem largely due to the fact that the CLE is intended to be usable in as wide as possible a range of hardware and software environments, so that the interface cannot assume any graphical facilities such as cursor-addressable displays. Were such facilities to be available, the system could provide step-by-step feedback on the consequences of individual grammaticality judgments.

The fact that VEX is not specific to any one application domain or type of back-end system, and is relatively loosely coupled to the internal characteristics of the CLE, means that the techniques it embodies should in principle be applicable to (even if not always optimal or sufficient in) a wide range of natural language processing contexts. Indeed, it might be possible to produce a version of VEX with clearly-defined interfaces at morphological, syntactic and semantic levels that could simply be "plugged in" to a range of existing systems to provide them with a lexical acquisition capability.

10 ACKNOWLEDGEMENTS

Development of the CLE has been carried out as part of a research programme in natural language processing supported by the UK Department of Trade and Industry under Alvey grant ALV/PRJ/IKBS/105 and by members of the NATTIE consortium (British Aerospace, British Telecom, Hewlett Packard, ICL, Olivetti, Philips, Shell Research, and SRI). I would like to thank the CLE development team, and various members of the consortium organizations, for valuable criticisms and suggestions.

11 REFERENCES

Alshawi, Hiyan, David M. Carter, Jan van Eijck, Robert C. Moore, Douglas B. Moran, Fernando C.N. Pereira, Arnold G. Smith and Steve G. Pulman. 1988a. *Interim Report on the SRI Core Language Engine.* Report CCSRC-005, SRI International Cambridge Research Centre, Cambridge, England.

Alshawi, Hiyan, David M. Carter, Jan van Eijck, Robert C. Moore, Douglas B. Moran and Steve G. Pulman. 1988b. Overview of the Core Language Engine. *Proceedings of the International Conference on Fifth Generation Computer Systems*, Tokyo, pp. 1108-1115; also Report CCSRC-008, SRI International Cambridge Research Centre, Cambridge, England.

Gross, Maurice. 1975. *Méthodes en Syntaxe*, Hermann, Paris.

Grosz, Barbara J., Douglas E. Appelt, Paul Martin, and Fernando C.N. Pereira. 1987. TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces. *Artificial Intelligence*, 32:173–243.