The LEXSYS Project

John Carroll, Nicolas Nicolov, Olga Shaumyan, Martine Smets & David Weir
University of Sussex
Brighton, BN1 9QH, UK

1 Introduction

We present an overview of the ongoing LEXSYS project[1]. The aim is to bring together, and evaluate, a variety of current NLP techniques, including the organisation of grammars into inheritance hierarchies for compact representation, exploitation of diverse precompilation techniques for efficient parsing, and use of statistical analysis to disambiguate parse results. In conjunction with this we are using several existing tools and resources, such as the lexicon developed in the Alvey Natural Language Tools project (Briscoe et al., 1987), lexical frequency information from the SPARKLE project[2], and an established lexical knowledge representation language DATR (Evans and Gazdar, 1996a) to represent the grammar. The overall architecture of LEXSYS is shown in Figure 1 and the following sections discuss each of the system's main components.

2 The morphological analyser

The text is first *tokenised* and then a *sentence-splitter* is applied to it to determine likely sentence boundaries. The resulting sentences are tagged with extended part-of-speech (PoS) labels using a first-order HMM *tagger* (Elworthy, 1994) trained on the SUSANNE corpus (Sampson, 1995). The SUSANNE lexicon is augmented with open-class words from the LOB corpus and the tagger incorporates a part-of-speech guesser that empirically achieves around 85% label assignment accuracy for unknown words. For each word the tagger returns multiple-label hypotheses, but filters out any whose probabilities are below a preset factor of the most probable. The thresholding technique allows us to fine-tune the trade-off between the costs of incorrect tagging and processing complexity due to lexical ambiguity.

After tagging, a *lemmatiser* finds the lemma, or base form, corresponding to each word-label pair, using an enhanced version of the GATE project stemmer (Cunningham et al., 1995). Finally, the lemma and PoS label are combined with syntactic information associated with the word's morphological form (e.g. number for nouns).

3 The grammar

Lexicalized D-Tree Grammar (LDTG) (Rambow et al., 1995) is a variant of LTAG. The primitive elements of LDTG are called elementary d-trees and are combined together to form larger structures during a derivation. Although, for convenience, we present d-trees graphically as though they were conventional trees, they are more correctly thought of as expressions in a tree description logic (Rogers and Vijay-Shanker, 1992). These expressions *partially* describe trees by asserting various relationships between nodes: parenthood, domination, precedence (indicating that one node is to the left of another), equality and inequality.

There are two substitution-like operations for composing d-trees, both of which involve combining two descriptions while equating exactly one node from each description. One of the operations is always used to add complements and involves equating a frontier node (in the d-tree that is getting the complement) with the root of some component (in the d-tree that is providing the complement), such that the two nodes

[1]This work is supported by UK EPSRC project GR/K97400 and by an EPSRC Advanced Fellowship to Carroll. Thanks to Roger Evans, Gerald Gazdar & K. Vijay-Shanker for helpful discussions.

[2]CEC Telematics Applications Programme project LE1-2111 "SPARKLE: Shallow PARsing and Knowledge extraction for Language Engineering".

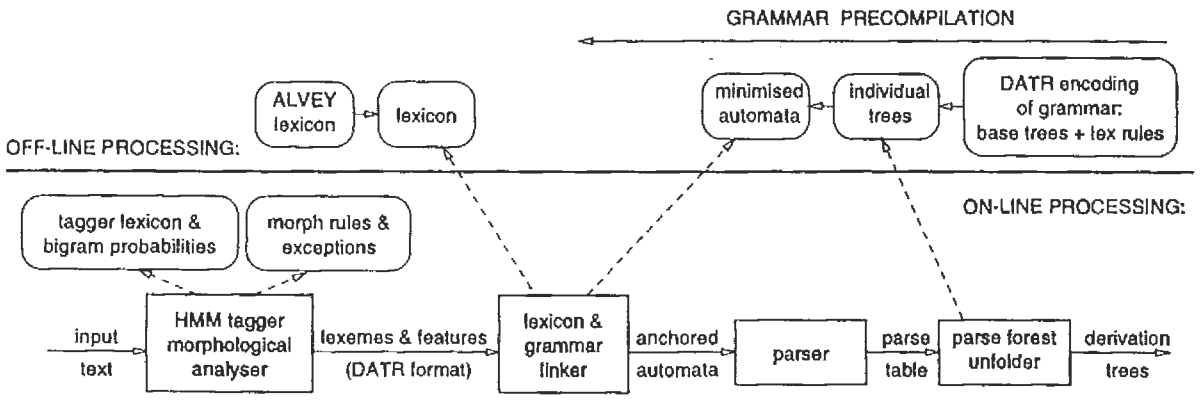

Figure 1: System architecture

being equated are compatible. Two schematic examples of this operation are shown at the top of Figure 2. These are the two cases that appear in our grammar for English[3]: at the top left is the case in which the entire complement d-tree appears below the point of substitution; the top right gives the case in which the complement involves extraction where the extracted component is placed at the top of the d-tree.

A second operation is used to add modifiers. In terms of tree descriptions, this operation is similar to the complement-adding operation since it also involves combining two d-trees while equating a pair of nodes. In this case, however, it involves equating an *internal* node (in the d-tree that is getting the modifier) with the root of some component (in the d-tree that is providing the modifier), such that the two nodes being equated are compatible. Two schematic examples are shown at the bottom of Figure 2. As in the case of the complement-adding operation, these are the two cases that appear in our grammar for English: at the bottom left is the case in which the entire modifying d-tree appears below the point of modification; the bottom right gives the case in which the modifier involves extraction, where the extracted component is placed at the top of the d-tree[4].

We are in the process of developing a wide-coverage LDTG based on the XTAG grammar. There are a number of differences between the formalisms and the analyses they allow. One of the main differences is that the LDTG formalism allows the existence of VPcomplements for main verbs, and this has a number of consequences: e.g. the grammar does not assume the existence of *PRO*, auxiliary and main verbs anchor the same type of tree, there are no predicative trees, passive participles anchor VPtrees [5]. See Smets (1998) for more details.

As in the XTAG system, ES'S are grouped into families. Currently we have 44 families with around 60 families expected in total. The total number of (unanchored) ES'S in the current grammar is 650 with approximately 1000 ES'S expected. The grammar is encoded using the lexical knowledge representation language DATR (Evans and Gazdar, 1996b), based on the scheme proposed for LTAG by Evans, Gazdar and Weir (1995). Encoding is compacted through the use of 36 lexical rules and non-monotonic inheritance. Details are presented in Smets and Evans (1998).

4 The lexicon

The lexicon is a reworked version of the Alvey Natural Language Tools (ANLT) lexicon (Carroll and Grover, 1989) where category and feature assignments are expressed in DATR notation to conform to the encoding used for the grammar and the results of morphological anal-

[3]The general case is explained in Rambow et al. (1995).

[4]In the examples shown at the bottom of Figure 2 the modifier d-tree is placed to the left of the subtree it modifies. It is also possible for modification to take place on the right.

[5]The analyses that we are able to implement are also adopted in a number of theories: GPSG, HPSG, LFG, CG.

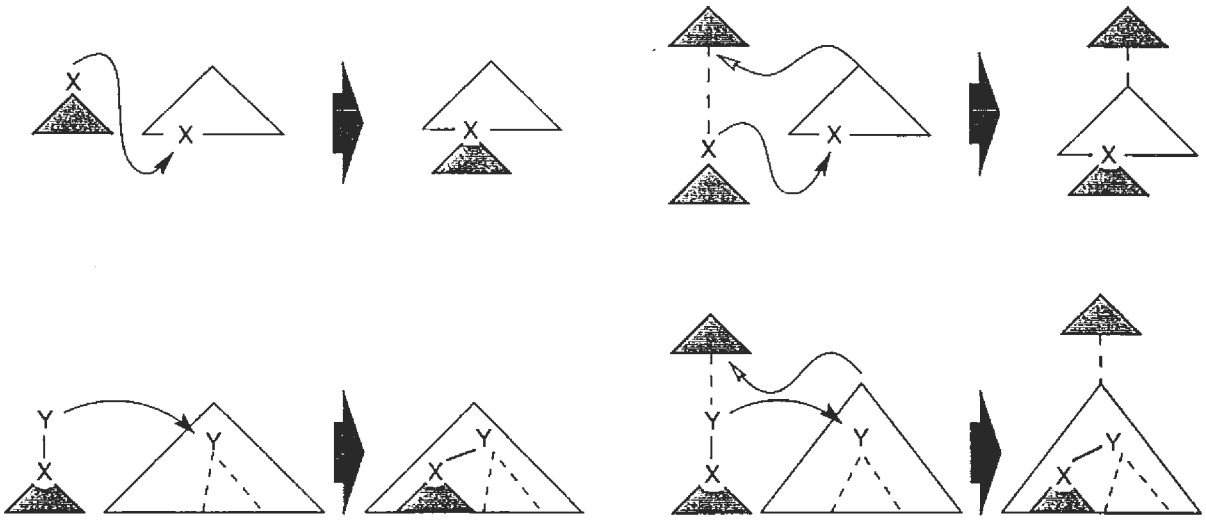

Figure 2: Composition operations

ysis. Although not currently exploited, this uniform notation would permit the lexicon to form the leaf nodes in the grammar hierarchy and so inherit automatically any of the syntactic information (such as default feature assignment) contained there. The lexicon contains only lemmas, with wordform information supplied by the morphological analyser. It should be noted that the morphological form of a linguistic datum affects how much of a family is selected: so the *ing* form of the verb will not inherit all of the ES's associated with the verb, but only the forms stipulated as *ing* or *non-finite*.

In separate but related work (Briscoe and Carroll, 1997), we are acquiring the complementation possibilities for predicates from large amounts of text information about. In that work we distinguish 160 verbal subcategorisation classes—a superset of those found in the ANLT and COMLEX Syntax dictionaries—and we acquire relative frequencies for each class found for each verb. The approach uses a previously-existing phrase-structure parser which yields 'shallow' parses, a subcategorisation class classifier, and *a priori* estimates of the probability of membership of these classes. Carroll et al. (1998a) demonstrate that adding this frequency information to a (non-lexicalised) statistical parser significantly increases its disambiguation accuracy. We intend also to incorporate this information into the system described in this paper, at the point where lemmas are associated with tree families: each lemma / family combination would have a separate probability. Carroll and Weir (1997) outline other alternative probabilistic models, some of which we also intend to investigate.

The same shallow phrase-structure parser is also providing data for the acquisition of selectional preferences, at present again just for verbs, and only for NP and PP subject, direct and indirect verbal complements (McCarthy, 1997). The technique uses the WordNet hypernym hierarchy (Fellbaum, 1998) in tandem with Minimum Description Length learning (Rissanen, 1978) to induce semantic classes of nominal heads at an appropriate level of abstraction. We have results of acquisition from a 10 million word extract from the British National Corpus, and will augment the lexicon with the acquired selectional frequencies and use them during parsing as a further source of disambiguation information.

5 The parser

We have implemented a simple bottom-up parsing algorithm which is being used for grammar development. The parser simulates anchor-up traversal of ES's. This traversal begins at the anchor node with the parser working outwards as it moves upwards towards the root of the ES. When visiting nodes during this traversal,

the parser must perform various actions. Which particular action is required at each node is determined by the type of node (e.g. whether it is a frontier or internal node) and its position relative to the anchor (whether it is to the right or left of the anchor). We refer to each step in this sequence as a parser action and to a sequence of parser actions associated with a ES, as an elementary computation (EC) of that ES.

Prior to parsing, each word of the input is associated with a set of ES'S that it can anchor. Each ES in the grammar can be pre-compiled into a (flat) sequence of parser actions. These sequences, rather than the ES'S themselves, are the objects that the parser manipulates during parsing.

The parser fills a 2-dimensional table (where each cell corresponds to a substring of the input) by advancing through these parser action sequences as actions are executed. In addition to action sequences, the items in cells contain multisets that hold suspended action sequences. In LTAG, adjunction has the effect of embedding one tree within another, where a stack can be used by a parser to control the unbounded nesting of ES'S that can occur in derivations. LDTG also allows embedding of ES'S; however, multisets rather than stacks are used to control this embedding. This difference is due to the limited control provided by LDTG over the relative positioning of the components of two composed ES'S. Each entry in the parse table contains a list of pointers to the entries that caused it to be added. Once the table is complete, top-down pruning is performed to remove entries that do not form part of a complete parse. This produces a parse forest from which phrase structure trees are derived.

Building an efficient parser for a *wide-coverage* LDTG or LTAG grammar represents a challenge. Each word in the input string introduces a large number of ES'S into the parse table: one for each of its possible alternative readings. In the current grammar the words *come*, *break* and *give* anchor around 130, 180 and 340 ES'S, respectively. In fact, if we include ES'S for all alternative feature values, these figures rise by an order of magnitude. There can be substantial overlap in structure among the ES'S associated with a given input word. Existing LTAG parsing algorithms treat each ES as independent, which results in considerable duplication of processing of common structure during parsing. Evans and Weir (1997; 1998) propose that a significant amount of overlapping among EC'S can be pre-compiled out by performing the following steps: (1) compile each ES into a finite state automaton; (2) for each set of ES'S that a single word can anchor, merge the corresponding automata into a single automaton; (3) minimise the number of states in the merged automaton (using standard techniques); and (4) rather than associating each input word with a set of d-trees, associate it with a minimized automaton and parse as usual. A preliminary indication of how the Evans-Weir proposal will work in practise on the LEXSYS grammar is discussed in (Carroll et al., 1998b) where we show that using minimized automata leads to a several-hundred-fold reduction in the number of automata states. Even greater savings are achieved when all feature information from the lexicon is included. In fact, the use of minimized automata appears to provide an efficient solution to processing ES'S whose node labels involve feature structures that might normally be encoded with disjunctive feature values (but which we encode with multiple instances of the ES). We are in the process of implementing a parser that exploits this technique in order to more fully evaluate its practical value.

6 Summary

LEXSYS is being developed as a wide-coverage parsing system using a lexicalized grammar formalism. We are employing two techniques to keep the scope of the task under control: (1) encoding grammar nsing DATR to achieve compact representation, and (2) parsing with minimized automata to achieve computation sharing. We feel that this approach allows us to maintain a separation between the issues of linguistic adequacy and processing pragmatics (grammar storage, parsing efficiency, etc.). The future work will also incorporate a stochastic component for parse disambiguation.

References

Ted Briscoe and John Carroll. 1997. Automatic extraction of subcategorization from corpora. In *Proceedings of the 5th ACL Conference on*

Applied Natural Language Processing, pages 356–363, Washington, DC.

Edward Briscoe, Claire Grover, Branimir Boguraev, and John Carroll. 1987. A formalism and environment for the development of a large grammar of English. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 703–708, Milan, Italy.

John Carroll and Claire Grover. 1989. The derivation of a large computational lexicon of English from LDOCE. In B. Boguraev and E. Briscoe, editors, *Computational Lexicography for Natural Language Processing*. Longman.

John Carroll and David Weir. 1997. Encoding frequency information in lexicalized grammars. In *Proceedings of the 5th International Workshop on Parsing Technologies*, pages 8–17.

John Carroll, Guido Minnen, and Ted Briscoe. 1998a. Can subcategorisation probabilities help a statistical parser? In *Proceedings of the 6th ACL/SIGDAT Workshop on Very Large Corpora*, Montreal, Canada.

John Carroll, Nicolas Nicolov, Olga Shaumyan, Martine Smets, and David Weir. 1998b. Grammar compaction and computation sharing in automaton-based parsing. In *Proceedings of the First Workshop on Tabulation in Parsing and Deduction*, pages 16–25.

Hamish Cunningham, Robert Gaizauskas, and Yorick Wilks. 1995. A general architecture for text engineering (GATE) — A new approach to language R&D. Research memo CS-95-21, Department of Computer Science, University of Sheffield, UK.

David Elworthy. 1994. Does Baum-Welch re-estimation help taggers? In *Proceedings of the 4th ACL Conference on Applied Natural Language Processing (ANLP'94)*, Stuttgart, Germany.

Roger Evans and Gerald Gazdar. 1996a. DATR: A Language for Lexical Knowledge Representation. *Computational Linguistics*, 22(2):167–247.

Roger Evans and Gerald Gazdar. 1996b. DATR: A language for lexical knowledge representation. *Computational Linguistics*.

Roger Evans and David Weir. 1997. Automaton-based parsing for lexicalized grammars. In *Proceedings of the 5th International Workshop on Parsing Technologies*, pages 66–76.

Roger Evans and David Weir. 1998. A structure-sharing parser for lexicalized grammars. In *Proceedings of the 36th Meeting of the Association for Computational Linguistics and the 16th International Conference on Computational Linguistics*.

Roger Evans, Gerald Gazdar, and David Weir. 1995. Encoding lexicalized tree adjoining grammars with a nonmonotonic inheritance hierarchy. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics (ACL'95)*, pages 77–84.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Diana McCarthy. 1997. Word sense disambiguation for acquisition of selectional preferences. In *Proceedings of the Proceedings of the ACL/EACL 97 Workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 52–61, Madrid, Spain.

Owen Rambow, K. Vijay-Shanker, and David Weir. 1995. D-Tree Grammars. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics*, pages 151–158.

J. Rissanen. 1978. Modeling by shortest data description. *Automatica*, 14:465–471.

James Rogers and K. Vijay-Shanker. 1992. Reasoning with descriptions of trees. In *Proceedings of the 30th Meeting of the Association for Computational Linguistics*, pages 72–80.

Geoffrey Sampson. 1995. *English for the Computer*. Oxford University Press, Oxford, UK.

Martine Smets and Roger Evans. 1998. A compact encoding of a DTG grammar. In *Proceedings of the TAG+ Workshop*.

Martine Smets. 1998. Comparison of XTAG and DTG. In *Proceedings of the TAG+ Workshop*.