Technical Report

UCAM-CL-TR-224
ISSN 1476-2986

Number 224

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Generalised probabilistic LR parsing of natural language (corpora) with unification-based grammars

Ted Briscoe, John Carroll

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

https://www.cl.cam.ac.uk/

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Generalised Probabilistic LR Parsing of Natural Language (Corpora) with Unification-based Grammars

Ted Briscoe & John Carroll
(ejb / jac @cl.cam.ac.uk)

University of Cambridge, Computer Laboratory
Pembroke Street, Cambridge, CB2 3QG, UK

Abstract

We describe work towards the construction of a very wide-coverage probabilistic parsing system for natural language (NL), based on LR parsing techniques. The system is intended to rank the large number of syntactic analyses produced by NL grammars according to the frequency of occurrence of the individual rules deployed in each analysis. We discuss a fully automatic procedure for constructing an LR parse table from a unification-based grammar formalism, and consider the suitability of alternative LALR(1) parse table construction methods for large grammars. The parse table is used as the basis for two parsers; a user-driven interactive system which provides a computationally tractable and labour-efficient method of supervised learning of the statistical information required to drive the probabilistic parser. The latter is constructed by associating probabilities with the LR parse table directly. This technique is superior to parsers based on probabilistic lexical tagging or probabilistic context-free grammar because it allows for a more context dependent probabilistic language model, as well as use of a more linguistically adequate grammar formalism. We compare the performance of an optimised variant of Tomita's (1987) generalised LR parsing algorithm to an (efficiently indexed and optimised) chart parser. We report promising results of a pilot study training on 151 noun definitions from the *Longman Dictionary of Contemporary English* (LDOCE) and retesting on these plus a further 54 definitions. Finally, we discuss limitations of the current system and possible extensions to deal with lexical (syntactic and semantic) frequency of occurrence.

Contents

1 Wide-coverage Parsing of Natural Language

The task of syntactically analysing substantial corpora of naturally-occurring text and transcribed speech has become a focus of recent work. Analysed corpora would be of great benefit in the gathering of statistical data regarding language use, for example to train speech recognition devices, in more general linguistic research, and as a first step towards robust wide-coverage semantic interpetation. The Alvey Natural Language Tools (ANLT) system is a wide-coverage lexical, morphological and syntactic analysis system for English (e.g. Briscoe et al., 1987). Previous work has demonstrated that the ANLT system is, in principle, able to assign the correct parse to a high proportion of English noun phrases drawn from a variety of corpora. The goal of the work reported here is to develop a practical parser capable of returning a single most likely parse (from the usually large number of possibilities) for material drawn from a specific corpora on the basis of minimal (supervised) training and manual modification.

The first issue in corpus analysis to consider is what the analysis will be used for and what constraints this places on its form. The literature contains a variety of proposals, ranging from part-of-speech tagging to assignment of a unique, sophisticated syntactic analysis. Our eventual goal is to recover a semantically and pragmatically appropriate syntactic analysis capable of supporting semantic interpretation. Two stringent requirements follow immediately: firstly, the analyses assigned must determinately represent the syntactic relations which hold between all constituents in the input; secondly, they must be drawn from an *a priori* defined, well-formed set of possible syntactic analyses (such as the set defined by a generative grammar). Otherwise semantic interpretation of the resultant analyses cannot be guaranteed to be (structurally) unambiguous and the semantic operations defined (over syntactic configurations) cannot be guaranteed to match and yield an interpretation. These requirements immediately suggest that approaches which recover only lexical 'tags' or a syntactic analysis which is the 'closest fit' to some previously defined set of possible analyses, are inadequate (taken alone).

Pioneering approaches to corpus analysis proceeded on the assumption that computationally-tractable generative grammars of sufficiently general coverage could not be developed (see papers in Garside et al., 1987). However, the development of wide-coverage declarative and computationally tractable grammars makes this assumption questionable. For example, the ANLT grammar and lexicon (Grover et al., 1989; Carroll & Grover, 1989) consists of an English lexicon of 35,000 morphemes and a 'compiled' fixed-arity term unification grammar containing 1346 phrase structure rules. Taylor et al. (1989) demonstrate that this grammar is capable of assigning the correct analysis to 96.8% of a corpus of 10,000 noun phrases extracted (without regard for their internal form) from a variety of corpora. However, although Taylor et al. show that the ANLT grammar has very wide coverage, they do not offer solutions to the problems of 1) tuning a grammar to a particular corpus or sublanguage 2) selecting the correct analysis from the set licensed by the grammar and 3) providing reliable analyses of input outside the coverage of the grammar. Firstly, it is clear that vocabulary, idiom and conventionalised constructions used in, say,

legal language and dictionary definitions will differ both in terms of the range and frequency of words and constructions deployed. Secondly, Church & Patil (1982) demonstrate that for a realistic grammar parsing realistic input, the set of possible analyses licensed by the grammar can be in the thousands. Finally, it is extremely unlikely that any generative grammar will ever be capable of correctly analysing all naturally occurring input (if only because of the synchronic idealisation implicit in the assumption that the set of grammatical sentences of a language is well-formed.)

Solutions to these three problems would make syntactic analysis of corpora feasible. In this paper, we describe our approach to the first and second problems and make some preliminary remarks concerning the third (far harder) problem. Our approach to grammar tuning is based on a semi-automatic parsing phase during which modifications to the grammar are made manually and statistical information concerning the frequency of use of grammar rules is acquired. Using this statistical information and modified grammar, a breadth-first probabilistic parser is constructed. The latter is capable of ranking the possible parses identified by the grammar in a useful (and efficient) manner. However, (unseen) sentences whose correct analysis is outside the coverage of the grammar remain a problem. The feasibility and usefulness of our approach has been investigated in a preliminary way by analysing a small corpus of noun definitions drawn from the *Longman Dictionary of Contemporary English* (LDOCE) (Proctor, 1978). This corpus was chosen because the vocabulary employed is restricted (to approximately 2000 morphemes), average definition length is about 10 words (with a maximum of around 30), and each definition is independent, allowing us to ignore phenomena such as ellipsis. In addition, the language of definitions represents a recognisable sublanguage, allowing us to explore the task of specialising a general purpose grammar. The results reported below suggest that probabilistic information concerning the frequency of occurrence of syntactic rules correlates in a useful (though not absolute) way with the semantically and pragmatically most plausible analysis.

2 Probabilistic Approaches to Parsing

Probabilistic approaches to syntactic analysis have mostly involved lexical 'tagging' with extended part-of-speech labels, or parsing using probabilistic context-free grammars, both approaches based on first-order Markov modelling. Supervised and unsupervised learning techniques have been investigated. In the former case, an unambiguous training corpus must be created which is used to derive the probabilities. In the latter, probabilities are acquired automatically using some version of the inside-outside algorithm (e.g. Baker, 1982) and repeated re-estimation of probabilities from an ambiguous, unanalysed corpus. Supervised learning is likely to yield better results but requires manual analysis of a training corpus. Unsupervised learning using the inside-outside algorithm will converge, but not necessarily on the optimal probabilities.

Stochastic first-order Markov models have been used to tag words in corpora with lexical syntactic categories appropriate to their grammatical context (e.g. De Rose,

1988). Each word is associated with one or more categories and a matrix of transition probabilities between two (bigrams) or sometimes three categories (trigrams) is used to find the most probable path through a sequence of words associated with more than one category. This technique has been shown to yield accuracy rates of 95% or better when trained and tested on the same corpus, and several researchers have argued that it represents a useful first stage in the production of a complete parse of corpus material in order to reduce lexical ambiguity (Garside & Leech, 1985; Hindle, 1989). To investigate the utility of lexical tagging, we applied this technique to our corpus of LDOCE noun definitions, using a bigram model with 22 categories trained on a modified version of the tagged Lancaster Oslo/Bergen (LOB) corpus (Garside, 1987). Manual inspection of 94 tagged definitions revealed a success rate (per word) of 92.5%; that is, 67 errors. However, the probability of an error occurring in a definition of 9 or more words remains greater than 50%, and out of 11 definitions containing 20 or more words only 2 were correctly tagged throughout. Errors were quite systematic and tended to occur in constructions involving syntactic dependencies beyond the 'range' of the bigrams, such as coordination, preposed phrases and so forth, reflecting the inadequacy of finite-state models of natural language syntax (Chomsky, 1957). The majority of errors observed are local syntactic ambiguities which would be rejected by a parser computing globally coherent analyses. For example, in the definition of *absolution* given in (2-1) *that* is incorrectly tagged as NP, not COMP, but there is no overall consistent syntactic analysis of (2-1) using this tag.

(2-1) the_DET_ words_N_ said_V_ by_P_ a_DET_ priest_N_ in_P_ a_DET_ church_N_ service_N_ when_PPRO_ he_NP_ declares_V_ that_NP_ the_DET_ people_N_ are_AUX_ forgiven_V_

The same tagging system, but untrained, was applied to the same corpus and repeated re-estimation was used until the system converged. In this unsupervised learning mode, manual inspection revealed a success rate (per word) of about 80%—a very significant decrease in performance over the case where unambiguous data was utilised during a supervised learning phase. Thus, tagging requires manual creation of a training corpus to achieve useful levels of performance. In addition, its usefulness as a first stage in parsing is further reduced by the fact that lexical ambiguity in the ANLT lexicon (and others like it) is largely the result of different subcategorisation possibilities within each major category, and not of part-of-speech ambiguity *per se*; for example, the verb *believe* has eight basic entries representing subcategorisation possibilities. For these reasons, we decided to explore approaches which attempt a full syntactic analysis of untagged material.

Fujisaki et al. (1989) describe a corpus analysis experiment using a probabilistic context-free grammar (CFG). They trained a CFG containing 2118 rules on a corpus of 4206 sentences. The grammar was initially converted into Chomsky Normal Form and then Griebach Normal Form, increasing the number of productions to 7550. These transformations simplify the statement of the training and parsing algorithms. In addition, since the definition of Griebach Normal Form requires that each production contain a terminal category as its leftmost daughter, and since

Fujisaki et al. appear to treat lexemes as terminal categories, this means that the grammar represents substantial lexical information too. For example, two possible productions might be VP → believe Sfin and VP → believe NP; associating different probabilities with these rules will allow the representation of the relative probability of *believe* subcategorising for a sentential complement or noun phrase object in the probabilistic model. The training process involved automatically assigning probabilities to each CF rule on the basis of their frequency of occurrence in analyses. As the training procedure was unsupervised, the data concerning frequency of use was drawn from both correct and incorrect parses of the corpus. The inside-outside algorithm was used to derive successive approximations for the rule probabilities which take into account the probabilities of the different parses of ambiguous sentences in the corpus. The intuitive idea is that if probabilities are continuously re-estimated in a manner which increases the overall probability that the set of sentences in the training corpus were generated (regardless of syntactic analysis) by the grammar, this will assign probabilities to individual rules which accurately reflect their frequency of use in the correct analyses of each sentence. This re-estimation process is guaranteed to converge (Baker, 1982), in the sense that the probabilities assigned to rules will stabilise. Fujisaki et al. suggest that the stable probabilities will model semantic and pragmatic constraints in the corpus, but this will only be so if these correlate with the frequency of rules (and word forms) in correct analyses, and also if the 'noise' in the training data created by the incorrect parses is effectively factored out. Whether this is so will depend on the number of 'false positive' examples, with only incorrect analyses, the degree of hetrogeneity in the training corpus, and so forth. Fujisaki et al. report some results based on testing the parser on the corpus used for training. In 72 out of 84 sentences examined, the most probable analysis was also the correct analysis. 6 of the remainder were false positives and did not receive a correct parse, whilst the other 6 did but it was not the most probable. A success rate (per sentence) of 85% is apparently impressive, but it is difficult to evaluate properly in the absence of full details concerning the nature of the corpus. For example, if the corpus contains many simple and similar constructions, unsupervised training is more likely to converge quickly on a useful set of probabilities.

Sharman et al. (1990) conducted a similar experiment with a grammar in ID/LP format (Gazdar et al., 1985, Sharman, 1989). ID/LP grammars separate the two types of information encoded in CF rules—immediate dominance and immediate precedence—into two rule types which together define (a subset of) the CFLs. This allows probabilities concerning dominance, associated with ID rules, to be factored out from those concerning precedence, associated with LP rules. In this experiment, a supervised training regime was employed. A grammar containing 100 terminals and 16 non-terminals and initial probabilities based on the frequency of ID and LP relations was extracted from a manually parsed corpus of about 1 million words of text. The resulting probabilistic ID/LP grammar was used to parse 42 sentences of 30 words or less drawn from the same corpus. In addition, lexical syntactic probabilities were integrated with the probability of the ID/LP relations to rank parses. 18 of the parses were identical to the original manual analyses, whilst a further 19 were 'similar', yielding a success rate of 88%. What is noticeable about this experiment is

that the results are no better than Fujisaki et al.'s unsupervised learning experiment discussed above, despite the use of supervised training and a more sophisticated language model. It is likely that these differences derive from the corpus material used for training and testing, and that the results reported by Fujisaki et al. will not be achieved with most corpora. The search space involved in parsing with probabilistic CFGs will be much greater, in general, than that involved in lexical tagging. We demonstrated above that there is a considerable advantage to supervised as opposed to unsupervised tagging on an identical corpus with identical category sets. It would be surprising if this advantage were not even more pronounced in the case of the more complex problem of parsing. However, it is also possible that integrating information about lexemes directly into the grammar, and hence probabilistic model, as Fujisaki et al. did, improved performance considerably over the use of probabilities reflecting only the relative likelihood of a lexical (extended) part-of-speech tag, as in Sharman et al. It will only be possible to draw definite conclusions when researchers report more details concerning their (training) corpora (as we do in the appendix).

There are several inherent problems with probabilistic CFG (including ID/LP) based systems, whether they are trained in a supervised or unsupervised mode. Firstly, although CFG is an adequate model of the majority of constructions occurring in natural language (Gazdar & Mellish, 1989), it is clear that wide-coverage CFGs will need to be very large indeed, and this will lead to difficulties of (manual) development of consistent grammars and, possibly, to computational intractability at parse time (particularly during the already computationally expensive training phase). Secondly, associating probabilities with CF rules means that information about the probability of a rule applying at a particular point in a parse derivation is lost. This leads to complications distinguishing the probability of different derivations when the same rule can be applied several times in more than one way. Grammar 1 below is an example of a probabilistic CFG, in which each production is associated with a probability and the probabilities of all rules expanding a given non-terminal category sum to one.

Grammar 1

(2-2)

1)	T → S	(1.0)
2)	S → NP VP	(1.0)
3)	VP → Vtr NP	(.4)
4)	VP → Vintr	(.6)
5)	NP → ProNP	(.4)
6)	NP → Det N	(.3)
7)	NP → NP PP	(.3)
8)	N → N N	(.3)
9)	PP → P NP	(1.0)
10)	N → N@	(.7)

The probability of a particular parse is the product of the probabilities of each rule used in the derivation. Thus the probability of parse a) in Figure 1 is .0336. The

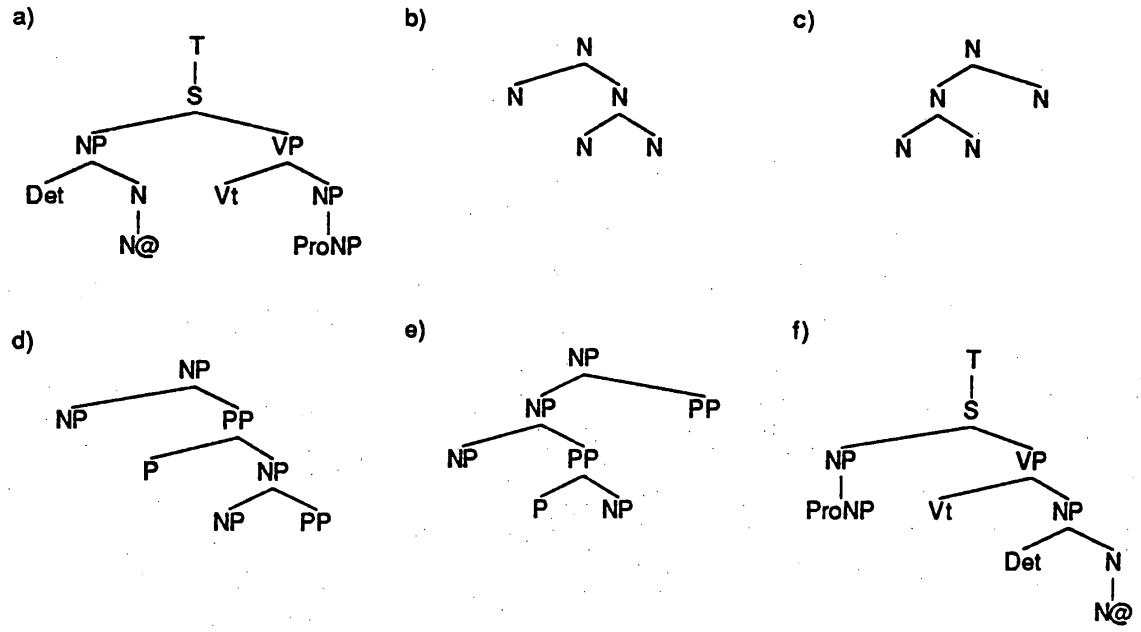

Figure 1: Probabilistic Context-free Derivations

probability of parse b) or c) must be identical though (.09), because the same rule is applied twice in each case. Similarly, the probability of d) and e) is also identical (.09) for essentially the same reason. However, these rules are natural treatments of noun compounding and prepositional phrase (PP) attachment in English, and the different derivations correlate with different interpretations. For example, b) would be an appropriate analysis for *toy coffee grinder*, whilst c) would be appropriate for *cat food tin*, and each of d) and e) yields one of the two possible interpretations of *the man in the park with the telescope*. We want to keep these structural configurations probabilistically distinct in case there are structurally conditioned differences in their frequency of occurrence; as would be predicted, for example, by the theory of parsing strategies (e.g. Frazier, 1988). Fujisaki et al. (1989) propose a rather inelegant solution for the noun compound case which involves creating 5582 instances of 4 morphosyntactically identical rules for classes of word forms with distinct bracketing behaviour in noun-noun compounds. However, we would like to avoid enlarging the grammar and eventually to integrate probabilistic lexical information with probabilistic structural information in a more modular fashion.

Probabilistic CFGs also will not model the context dependence of rule use; for example, a NP is more likely to be expanded as a pronoun in subject position than elsewhere (e.g. Magerman & Marcus, 1991b) but only one global probability can be associated with the relevant CF production. Thus the probabilistic CFG model predicts (incorrectly) that a) and f) will have the same probability of occurrence. These considerations suggest that we need a technique which allows use of a more adequate grammatical formalism than CFG and a more context dependent probabilistic language model. Our approach is to use the LR parsing technique as a

natural way to obtain a finite-state representation of a non finite-state grammar, and in section 8 we demonstrate that LR parse tables do provide an appropriate amount of contextual information to solve the problems described above.

3 LR parsing and Natural Language Processing

An LR parser (Aho, Sethi & Ullman 1986) is a shift-reduce parser guided by a parse table indicating what action should be taken next. Although originally developed for use in the syntax analysis phase of programming language compilers (where the grammar of the language will be unambiguous), in the past few years LR parsing has been advocated as a suitable technology for NL parsing systems (which use ambiguous grammars). Tomita (1987) describes an efficient breadth-first algorithm for non-deterministic parsing with an ambiguous context-free grammar. Tomita's approach uses standard LR parse table precompilation techniques but generalises the shift-reduce parser to operate with a 'graph-structured' stack and find all possible analyses defined by the grammar for some input. That is, he allows the parse table to contain conflicting actions (such as shift or reduce, or reduce with distinct rules) in a single state and thus define a non-deterministic finite-state automaton. Whenever the parser reaches a state in which there is a parse action conflict, the stack divides and both analyses are pursued, conceptually in parallel. The graph-structured stack is similar to a chart (e.g. Gazdar & Mellish, 1989) in that it is intended to be an efficient data structure in which shared sub-analyses are represented only once. However, generalised LR parsing should be more efficient than standard variants of chart parsing because the use of an LR parse table ensures that rule invocation is as constrained as possible given known automatic precompilation techniques (e.g. Tomita, 1987).

In Figure 2 we show the LALR(1) parse table for Grammar 1 above. This grammar is ambiguous, and so the parse table contains action conflicts (in states 6 and 9). The ambiguities correspond to whether right- or left-branching analyses of PP attachment or noun compounds are produced. A breadth-first generalised LR parser will produce all possible analyses for the input by generalising the standard stack based LR parsing algorithm (e.g. Aho et al., 1986) to a graph-structured stack based algorithm in which all paths through the parse table are explored in the event of an action conflict.

The table consists of two parts: an action table and a goto table. The easiest way to understand the table's function is to examine a simple derivation. In Figure 3 we give the derivation for a sentence such as *he loves her* consisting of a pronoun (ProNP), transitive verb (Vt) and a second pronoun. Since there is no ambiguity in the analysis, the derivation can be represented as a single stack (with the top element on the right) and the lookahead item in parentheses to its right.

The parser begins in state zero with an otherwise empty stack and the lookahead item being the terminal symbol associated with the first word of the input. The parse table entry for state zero with lookahead ProNP instructs the parser to shift and move to state two. This causes ProNP to be pushed onto the parse stack followed

Actions

State	$	Det	N@	P	ProNP	Vi	Vt
0		s3			s2		
1	r1						
2	r5			r5		r5	r5
3			s4				
4	r10		r10	r10		r10	r10
5	r6		s4	r6		r6	r6
6	r8		r8/s4	r8		r8	r8
7				s8		s13	s11
8		s3			s2		
9	r9			r9/s8		r9	r9
10	r7			r7		r7	r7
11		s3			s2		
12	r3			s8			
13	r4						
14	r2						
15	acc						

Gotos

State	N	NP	PP	S	T	VP
0		7		1	15	
1						
2						
3	5					
4						
5	6					
6	6					
7			10			14
8		9				
9			10			
10						
11		12				
12			10			
13						
14						
15						

Figure 2: LALR(1) Parse Table for Grammar 1

0)	0	(ProNP)
1)	0 ProNP 2	(Vt)
2)	0 NP	(Vt)
3)	0 NP 7	(Vt)
4)	0 NP 7 Vt 11	(ProNP)
5)	0 NP 7 Vt 11 ProNP 2	($)
6)	0 NP 7 Vt 11 NP	($)
7)	0 NP 7 Vt 11 NP 12	($)
8)	0 NP 7 VP	($)
9)	0 NP 7 VP 14	($)
10)	0 S	($)
11)	0 S 1	($)
12)	0 T	($)
13)	0 T 15	($)

Figure 3: Parse derivation for *he loves her*

by the new state number and the lookahead item to be set to the terminal symbol associated with the next word in the input. In state two with lookahead Vt the table instructs the parser to reduce with rule 5 (NP → ProNP), so the current state is popped and the topmost category is replaced with the mother category of rule 5. The parser next consults the goto part of the table to discover which state it should move to when in state zero with the non-terminal NP. The goto table instructs the parser to move to state seven so this state is pushed onto the stack. Now several shift actions occur (exactly as before) until the end of sentence marker ($) is the lookahead item. In step 6, the second pronoun is reduced to NP, but this time the goto table instructs the parser to move to state 12 (from state 11). In step 8, another reduction occurs from state 12 using rule 3 (VP → Vt NP), but this time two terminals on the stack are popped and replaced with VP because the rule specifies two daughters. Two further reductions bring the parser to the accept configuration and the parse halts successfully. Blank entries in the table correspond to error states; if the parser reaches one of these states it halts and returns a message indicating that the input is not analysable. In the case where more than one analysis is possible, it is conceptually easiest to think of the parse stack being copied and two separate analyses being pursued from thereon; however, this conceptually straightforward approach would lead to a very inefficient implementation (hence the graph-structured stack).

In this paper we describe how we have extended previous work on LR parsing of NL; firstly, by defining an automatic technique for deriving a LR parse table from a unification-based grammar, secondly, by developing a probabilistic version of generalised LR parsing in which probabilities derived from a training phase are associated directly with parse actions and/or state transitions in the LR parse table and these probabilities are used to rank distinct parse derivations, and thirdly, by developing

a semi-automatic LR based parser for constructing a disambiguated training corpus. In each case, we argue that LR techniques are the most appropriate.

4 LR parsing within a Unification-based Grammar Framework

The heart of the LR parsing technique is the parse table construction algorithm which is the most complex and computationally expensive aspect of LR parsing. Much of the attraction of the technique stems from the fact that the real work takes place in a precompilation phase and the run time behaviour of the resulting parser is relatively simple and efficient. An LR parser finds the 'rightmost derivation in reverse', for a given string and CF grammar. The precompilation process results in a parser control mechanism which enables the parser to identify the 'handle', or appropriate substring in the input to reduce, and the appropriate rule of the grammar with which to perform the reduction. The control information is standardly encoded as a parse table with rows representing parse states and columns terminal and non-terminal symbols of the grammar (see Figure 2 above). This representation defines a finite state automaton. If the grammar is an unambiguous CFG, the automaton will be deterministic, however, the standard algorithm for parse table construction also builds non-deterministic automata containing action conflicts for ambiguous CFGs. The parse table construction algorithm involves 3 steps: defining FIRST and FOLLOW lists for the grammar (as for Earley's algorithm), computing a set of item sets containing dotted rules which represent the amount of input 'consumed', and computing the transition functions between item sets. The parse table itself is a transformation of the resulting finite-state network. The relevant algorithms are described in Aho et al. (1986:215f), Chapman (1987) and elsewhere. An introduction to and evaluation of LR parsing for NL is given in Briscoe (1987:107f).

Tomita (1987) describes a system for non-deterministic LR parsing of context-free grammars consisting of atomic categories, in which each CF production may be augmented with a set of tests (which perform similar types of operation available in a unification grammar). At parse time, whenever a sequence of constituents is about to be reduced into a higher-level constituent using a production, the augmentation associated with the production is invoked to check syntactic or semantic constraints such as agreement, pass attribute values between constituents, and construct a representation of the higher-level constituent. The parser is driven by an LR parse table; however, the table is constructed solely from the CF portion of the grammar, and so none of the extra information embodied in the augmentations is taken into account during its construction. Thus the predictive power of the parser to select the appropriate rule given a specific parse history is limited to the CF portion of the grammar which must be defined manually by the grammar writer. This requirement places a greater load on the grammar writer and is inconsistent with most recent unification-based grammar formalisms, which represent grammatical categories entirely as feature bundles (e.g. Briscoe et al., 1987; Pollard & Sag, 1987; Zeevat et al., 1987). In addition, it violates the principle that grammatical formalisms should

be declarative and defined independently of parsing procedure, since different definitions of the CF portion of the grammar will, at least, effect the efficiency of the resulting parser and might, in principle, lead to non-termination on certain inputs in a manner similar to that described by Shieber (1985).

In what follows, we will assume that the unification-based grammars we are considering are represented in the ANLT object grammar formalism (e.g. Briscoe et al., 1987). This formalism is a notational variant of Definite Clause Grammar (e.g. Pereira & Warren, 1980) in which rules consist of a mother category and one or more daughter categories, defining possible phrase structure configurations. Categories consist of sets of feature name-value pairs, with the possibility of variable values, which may be bound within a rule, and of category-valued features. Categories are combined using fixed-arity term unification (Prolog-style). The results and techniques we report below should generalise to many other unification-based formalisms. An example of a possible ANLT object grammar rule is given in (4-1).

(4-1)

[N −, →	[N +,	[N −,
V +,	V −,	V +,
BAR 2,	BAR 2,	BAR 1,
PER x,	PER x,	PER x,
PLU y,	PLU y,	PLU y,
VFM z]	CSE Nom]	VFM z]

This rule provides a (simple) analysis of the structure of English clauses, corresponding to S → NP VP, using a feature system based loosely on that of GPSG (Gazdar et al., 1985). In Tomita's LR parsing framework, each such rule must be manually converted into a rule of the form shown in (4-2) in which some subpart of each category has been replaced by an atomic symbol.

(4-2)

Vb →	Nn	Vb
[BAR 2,	[BAR 2,	[BAR 1,
PER x,	PER x,	PER x,
PLU y,	PLU y,	PLU y,
VFM z]	CSE Nom]	VFM z]

However, it is not obvious which features should be so replaced—why not include BAR and CSE? It will be difficult for the grammar writer to make such substitutions in a consistent way, and still more difficult to make them in an optimal way for the purposes of LR parsing, since both steps involve consideration and comparison of all the categories mentioned in each rule of the grammar. Constructing the LR parse table directly and automatically from a unification grammar would avoid these drawbacks. In this case, the LR parse table would be based on complex categories, with unification of complex categories taking the place of equality of atomic ones in the standard LR parse table construction algorithm (Osborne, 1990). However, this approach is computationally prohibitively expensive: Osborne (1990:26) reports that his implementation (in HP Common Lisp on a Hewlett Packard 9000/350) takes

almost 24 hours to construct the LR(0) states for a grammar of just 75 productions. In addition, this approach will be subject to the same problems with non-termination which Shieber (1985) identifies with respect to Earley's algorithm, for grammars which employ features in recursive, cyclic and other complex ways.

Our approach, described below, not only extracts unification information from complex categories, but is also safe from inconsistency and non-termination. We start with a unification grammar and automatically construct a CF 'backbone' of rules containing categories with atomic names and an associated 'residue' of feature name-value pairs. Each backbone grammar rule is generally in direct one-to-one correspondence with a single unification grammar rule. The LR parse table is then constructed from the CF backbone grammar. This approach results in a computationally tractable parse table construction algorithm for realistic sized grammars. The parser is driven by this table, but in addition when reducing a sequence of constituents the parser performs the unifications specified in the relevant unification grammar rule to form the category representing the higher-level constituent, and the derivation fails if one of the unifications fails. Even though a unification grammar will be, at best, equivalent to a very large (and possibly infinite) set of atomic-category CF productions, in practice we have obtained efficient LR parsers from backbone grammars containing only about 30% more productions than the original unification grammar. The construction method ensures that for any given grammar the CF backbone captures the maximal amount of information available in the categories defined which can be represented in a CFG. Thus the construction method guarantees that the resulting LR parser will terminate and will be as predictive as the source grammar in principle allows.

Building the backbone grammar is a two-stage process:

1. Compute the largest maximally specific set (in terms of subsumption) of disjoint categories covering the whole grammar and assign to each category a distinct atomic category name. That is:

```
initialise disjoint-set to be empty;
for each category C in grammar
   let disjoint-merge be categories in disjoint-set which unify with C;
   if disjoint-merge is empty
      then add C to disjoint-set;
      else replace all elements of disjoint-merge in disjoint-set
         with the single most specific category which unifies with C
         and all categories in disjoint-merge;
assign a distinct name to each category in disjoint-set.
```

2. For each unification grammar rule, create a backbone grammar rule containing atomic categories, each atomic category being the name assigned to the category in the disjoint category set which unifies with the corresponding category in the unification grammar rule:

```
for each rule R of the form C1 --> C2 ... Cn in unification grammar
```

```
add a rule B of the form B1 --> B2 ... Bn to backbone grammar
  where Bi is the name assigned to the (single) category in
    disjoint-set which unifies with Bi, for i=1, n.
```

Algorithms for creating LR parse tables assume that the terminal vocabulary of the grammar is distinct from the non-terminal one, so the procedure described above will not deal properly with a unification grammar rule whose mother category is assumed elsewhere in the grammar to be a lexical category. The modification we make is to automatically associate two different atomic categories, one terminal and one non-terminal, with such categories, and to augment the backbone grammar with a unary rule expanding the non-terminal category to the terminal.

Two other aspects of the ANLT grammar formalism require further minor elaborations to the basic algorithm: firstly, a rule may introduce a gap by including the feature specification [NULL +] on the gapped daughter—for each such daughter an extra rule is added to the backbone grammar expanding the gap category to the null string; secondly, the formalism allows Kleene star and plus operators (Gazdar et al., 1985)—in the ANLT grammar these operators are utilised in rules for coordination. Rules containing Kleene star daughters are expanded out into a rule without the daughter and one with the daughter as a Kleene plus. A new non-terminal category is created for each distinct Kleene plus category and two extra rules are added to the backbone grammar. One (unary) rule expands the new category as the original category, and the other (binary) rule expands the new category to the original category followed by the new one. Without further modifications this would engender a right-branching binary tree structure, rather than the intended flat sequence of categories, but a parser can easily be modified to flatten out the tree structure during processing. For example, in the rule

N2 → N2[CONJ EITHER], N2[CONJ OR]+.

licensing noun phrases of the form *either kim or lee or sandy*, the second daughter may be repeated an indefinite number of times. Corresponding to this unification grammar rule there would be three backbone grammar rules, forming for this noun phrase the (unflattened) structure in Figure 4. Grammars written in other, more low-level unification grammar formalisms, such as PATR-II (Shieber, 1984), commonly employ treatments of the type just described to deal with phenomena such as gapping, coordination and compounding. However, this method both allows the grammar writer to continue to use the full facilities of the ANLT formalism, and allows the algorithmic derivation of an appropriate backbone grammar to support LR parsing.

The major task of the backbone grammar is to encode sufficient information (in the atomic categoried CF rules) from the unification grammar to constrain the application of the latter's rules at parse time. The nearly one-to-one mapping of unification grammar rules to backbone grammar rules described above works quite well for the ANLT grammar, with a single exception which creates spurious shift-reduce conflicts during parsing resulting in an unacceptable degradation in performance. In the ANLT grammar three very general rules are used to form nominal, adjectival

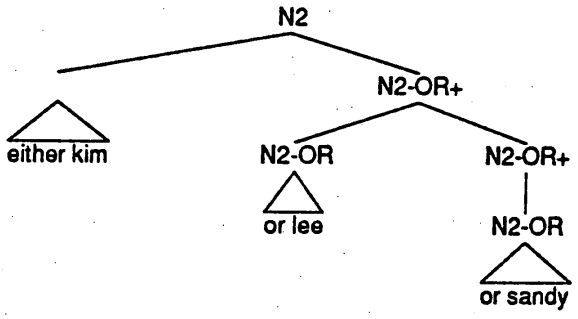

Figure 4: Backbone tree for rule with a Kleene plus daughter

and prepositional phrases following a conjunction; the categories in these rules lead to otherwise disjoint categories for conjuncts being merged, giving rise to a set of overly general backbone grammar rules for co-ordination. For example, the rule in the ANLT grammar for forming a noun phrase conjunct introduced by a conjunction is

N2[CONJ @con] → [SUBCAT @con, CONJN +], H2.

The variable value for the CONJ feature in the mother means that all N2 categories specified for CONJ (e.g. N2[CONJ EITHER], N2[CONJ NULL]) are generalised to the same category. This results in the backbone rules, when parsing *either kim or lee helps*, being unable, after forming a N2[CONJ EITHER] for *either kim*, to discriminate between the alternatives of preparing to iterate this constituent (as in the example *kim, lee or sandy helps* where *kim* would be N2[CONJ NULL]), or shifting the next word *or* to start a new constituent. We solve this problem by declaring CONJ to be a feature which may not have a variable value in an element of the disjoint category set. This forces the system to expand out each unification grammar rule which has a category containing CONJ with a variable value into a number of rules each of which is fully-specified for CONJ, and to create backbone rules for each of these. Although there are of the order of fifty possible values for SUBCAT, since the value of SUBCAT is bound to that of CONJ in these rules and there are only eight possible values for CONJ in the grammar (NULL, BOTH, NEITHER, EITHER, AND, OR, NOR and BUT), the general rule for forming a nominal conjunct given above, for example, ends up being represented by a set of eight specialised backbone grammar rules.

Similar types of inefficient behaviour are likely to occur with grammars which utilise complex patterns of feature value propagation; inference on the patterns of possible unification of these values and appropriate expanding-out of the categories concerned would be necessary for an LR parser to work effectively. We have experimented with a grammar which uses 'gap threading' (Pereira & Shieber, 1987), and in this case inference on the values of the features used for threading the gaps would be necessary to prevent the parser considering analyses containing infinite sequences of gaps. This and other areas of complexity in unification-based formalisms need

further investigation before we can claim to have developed a system capable of producing an optimal LR parse table for any unification-based grammar. In particular, declaring certain category-valued features so that they cannot take variable values may lead to non-termination in the backbone grammar construction algorithm. However, it should be possible to restrict the set of features which are considered in category-valued features in an analogous way to Shieber's (1985) restrictors for Earley's algorithm, so that an optimal parse table can still be constructed.

5 Construction of LR Parse Tables for Large NL Grammars

The backbone grammar generated from the ANLT grammar is large: it contains 575 distinct categories and more than 1750 productions. We therefore chose to construct an LALR parse table in preference to a canonical LR table, since the LALR table for a grammar is typically much smaller: Aho, Sethi & Ullman (1986:236) claim an order of magnitude difference for programming language grammars. For a given grammar, an SLR table always has the same number of states as its LALR counterpart, but the latter handles a slightly larger range of grammars without generating action conflicts. We take the number of symbols of lookahead to be 1, again to control the size of the parse table, since the number of states in the table is doubly exponential on the length of lookahead (Chapman, 1987). In fact, most parser-generators for programming languages also use LALR(1) tables; for example, Johnson (1975), Grosch (1990).

In the LR parsing literature there are essentially two approaches to constructing LALR(1) parse tables. One approach is graph-based (DeRemer & Pennello 1982), transforming the parse table construction problem to a set of well-known directed graph problems, which in turn are solvable by efficient algorithms. Unfortunately this approach does not work for grammars which are not LR(k) for any k (DeRemer & Pennello 1982:633), for example ambiguous grammars. We therefore broadly follow the alternative approach of Aho, Sethi & Ullman (1986), but with a number of optimisations:

1. Constructing the LR(0) sets of items: we first compute LR(0) states containing kernel items (the item [S' $\rightarrow$. S], where S' is the start symbol, and all items which have a symbol to the left of the dot), and then add to each state its non-kernel items by computing the LR(0) closure of the state. The computation of sets of items is performed in two stages because this makes it more efficient to compute whether a new candidate LR(0) state already exists, since the number of items in each set at this stage is much smaller. (With the ANLT grammar, the mean number of kernel items in each LR(0) set is about 8, whereas the mean number of both kernel and non-kernel items per state is more than 400).

2. Computing the LALR(1) lookaheads for each item: the conventional approach is to compute the LR(1) closure of each kernel item in order to determine the lookaheads which are generated *spontaneously* and those which *propagate* from

other items. However, in an initial implementation we found that the LR(1) closure operation as described by Aho et al. was too expensive to be practicable for the number and size of LR(0) states we deal with, even with schemes for caching the closures of non-kernel items once they had been computed. Instead, we have moved to an algorithm devised by Kristensen & Madsen (1981) which avoids performing the LR(1) closure operation. The crucial advantage of this algorithm is the ability, at any stage in the computation, to tell whether the calculation of the lookahead set for a particular item has been completed, is underway, or has not yet started. This means that even partially computed lookahead sets can be cached (with the computation yet to be done explicitly marked), and that items whose lookahead sets are found to subsume those of others are able to just copy the results from the subsumed sets.

3. Constructing the parse table: the LALR(1) parse table is derived straightforwardly from the lookahead sets, although to keep the size of the parse table within reasonable bounds we chose appropriate data structures to represent the goto entries and shift and reduce actions. For the ANLT backbone grammar there are 133,467 goto entries (non-terminal – state pairs), 300,378 shift actions (terminal – state pairs), and 720,482 reduce actions (terminal – rule-number pairs); however, of the goto entries only 2,257 are distinct and of the shift actions only 848 are distinct, most states contain just reduce or just shift actions, and in any one state very few different rules are involved in reduce actions. Taking advantage of the characteristics of this distribution, in each state we represent (in Common Lisp)

   (a) a set of goto entries as a list of (non-terminal – state) conses sorted into a canonical order, list elements and tails of lists shared where possible between states,

   (b) a set of shift actions as a list containing a single (large) integer (the list shared when possible between states), where if the state shifts to state s on lookahead t, the element indexed by t in an auxiliary array will contain s together with a number n, and bit n in the binary representation of the integer will be 1,

   (c) a set of reduce actions as, for each rule involved, a cons whose second element is the rule number and whose first is a bit-vector (shared when possible between states) whose nth bit is 1 if the reduce should occur with the nth terminal as lookahead,

   (d) an accept action as a cons with the first element being the lookahead symbol.

For the grammars we have investigated, this representation achieves a similar order of space saving to the *comb vector* representation suggested by Aho et al. (1986:244ff) for unambiguous grammars (see Klein and Martin (1989) for a survey of representation techniques). The parse table for the ANLT grammar occupies approximately 550Kbytes of memory, and so represents each goto entry and action

Grammar	Number of atomic rules/categories	Number of states	Number of LR(0) items	Number of actions
Pascal	158 / 124	275	?	2604
Tomita, Japanese	800 / ?	?	?	?
ANLT (1346 PS rules)	1758 / 575	3106	1345946	1020860

Table 1: Sizes of grammar and LALR(1) parse tables

Grammar	Backbone computation	LR(0) state construction	lookahead computation	parse table construction
ANLT	250	1650	5600	2100

Table 2: Timings for LALR(1) parse table construction (in seconds of CPU time excluding overheads on a DEC 3100 running Allegro Common Lisp)

in an average of less than 4 bits. In contrast to conventional techniques, though, we maintain a faithful representation of the parse table, not replacing error entries with more convenient non-error ones in order to save extra space. Our parsers are thus able to detect failures as soon as theoretically possible, an important efficiency feature when parsing non-deterministically with ambiguous grammars, and a time-saving feature when parsing interactively with them (see next section). Table 1 compares the size of the LALR(1) parse table for the ANLT grammar with several others reported in the literature.

¿From these figures, the ANLT grammar is more than twice the size of Tomita's (combined morphological and syntactic) grammar for Japanese (Tomita 1987:45). The grammar itself is about one order of magnitude bigger than that of a typical programming language, but the LALR(1) parse table, in terms of number of actions, is two orders of magnitude bigger (figures for Pascal from Klein & Martin (1989)); thus although Tomita (1984:357) anticipates LR parsing techniques being applied to large NL grammars written in formalisms such as GPSG, the sizes of parse tables for such grammars grow more rapidly than he predicts.

As might be expected, and Table 2 illustrates, parse table construction for large grammars is CPU-intensive. As a rough guide, LALR(1) table construction for a programming language grammar can take from about 5 to 50 seconds (figures for Modula-2 from Grosch (1990)), so scaling up two orders of magnitude, our timing figures for the ANLT grammar fall in the expected region.

6 Interactive Incremental Deterministic Parsing

The major problem with attempting to use a disambiguated training corpus is to find a way of constructing this corpus in an error-free and resource efficient fashion. Even manual assignment of lexical categories is slow, labour intensive and error prone. The greater complexity of constructing a complete parse makes the totally manual approach very unattractive, if not impractical. Sampson (1987:83) reports that it took 2 person / years to produce the 'LOB treebank' of 50,000 words. Furthermore, in that project, no attempt was made to ensure the analyses were well-formed with respect to a generative grammar. Attempting to manually construct analyses consistent with a grammar of any size and sophistication would place an enormous additional load on the analyst. Leech & Garside (in press) discuss the problems which arise in manual parsing of corpora concerning accuracy and consistency of analyses across time and analyst, the labour intensive nature of producing detailed analyses, and so forth. They advocate an approach in which simple 'skeleton' parses are produced by hand from previously tagged material, with checking for consistency between analysts. These skeleton analyses can then be augmented automatically with further information implicit in the lexical tags. Whilst this approach may well be the best that can be achieved with fully manual techniques, it is still unsatisfactory in several respects. Firstly, the analyses are crude, whilst we would like to automatically parse with a grammar capable of assigning sophisticated semantically interpretable ones; but it is not clear how to train an existing grammar with such unrelated analyses. Secondly, the quality of any grammar obtained automatically from the parsed corpus is likely to be poor because of the lack of any rigorous checks on the form of the skeleton parses. Such a grammar might, in principle, be trained from the parsed corpus, but there are still likely to be small mismatches between the actual analysis assigned manually and any assigned automatically. For these reasons, we decided to attempt to produce a training corpus using the grammar which we wished ultimately to train. As long as the method employed ensured that any analysis assigned was a member of the set defined by the grammar, these problems during training should not arise.

Following our experience of constructing a substantial lexicon for the ANLT grammar from unreliable and indeterminate data (Carroll & Grover, 1989), we decided to construct the disambiguated training corpus semi-automatically, restricting manual interaction to selection between alternatives defined by the ANLT grammar. One obvious technique would be to generate all possible parses with a conventional parser and to have the analyst select the correct parse from the set returned (or reject them all). However, this approach places a great load on the analyst, who will routinely need to examine large numbers of parses for given sentences. In addition, computation of all possible analyses is likely to be expensive and, in the limit, intractable.

Briscoe (1987) demonstrates that the structure of the search space in parse derivations makes a left-to-right, incremental mode of parse selection most efficient. For example, in noun compounds analysed using a recursive binary-branching rule ($N \rightarrow N\ N$) the number of analyses correlates with the Catalan series (Church &

Patil, 1982), so a 3 word compound has 2 analyses, 4 has 5, 5 has 14, 9 has 1430, and so forth. However, Briscoe (1987:154f) shows that with a simple bounded context parser (with one word lookahead) set up to request help whenever a parse indeterminacy arises, it is possible to select any of the 14 analyses of a 5 word compound with a maximum of 5 interactions and any of the 1430 analyses of a 9 word compound with about 13 interactions. In general, resolution of the first indeterminacy in the input will rule out approximately half the potential analyses, resolution of the next, half of the remaining ones, and so on. For 'worst case' CF ambiguities (with $O(n^3)$ complexity) this approach to parse selection involves numbers of interactions which increase at 'little more than linear rate' with respect to the length of the input. It is possible to exploit this insight in two ways. One method would be to compute all possible analyses represented as a chart or graph-structured stack and ask the user to select between competing sub-analyses which have been incorporated into a successful analysis of the input. In this way, only genuine global syntactic ambiguities would need to be considered by the user and, by tracing the dependencies between sub-analyses, all further sub-analyses dependent for incorporation into a complete analysis on a rejected sub-analysis could themselves be rejected without further interaction with the user. The disadvantage of this approach is that it relies on prior computation of the full set of analyses. The second method involves incremental interaction with the parser during the parse to guide it through the search space of possibilities. This has the advantage of being computationally efficient, but the potential disadvantage of requiring the user to resolve many local syntactic ambiguities which will not be incorporated into a successful analysis. Nevertheless, using LR techniques this problem can be minimised and, because we do not wish to develop a system which must be able to compute all possible analyses (at some stage) in order to return the most plausible one, we have chosen the latter incremental method.

The interactive, incremental parsing system that we implemented asks the user for a decision at each choice point during the parse. However, to be useable in practice, such a system must avoid, as far as possible, presenting the user with spurious choices that could be ruled out either by using more of the left context, or by looking at words yet to be parsed. A system based around an LR parser satisfies the first point, since the parse table encodes a maximal amount of left context information. As for the second point, an LALR(1) parser is able to look one word ahead to resolve ambiguities (although, of course, the resolution of a local ambiguity may potentially involve an unlimited amount of lookahead, e.g. Briscoe, 1987:125ff). In fact, LR parsing is the most powerful parsing technique for which an automatic compilation procedure is known. (Extensions to the LR technique, for example those using LR-regular grammars (Culic & Cohen, 1973; Bermudez, 1991), might be used to further cut down on interactions; however, computation of the parse tables to drive such extended LR parsers may prove intractable for large NL grammars. We are currently investigating this issue.)

An LR parser faces an indeterminacy when it enters a state in which there is more than one possible action, given the current lookahead. In a particular state there cannot be more than one shift or accept action, but there can be several reduce actions each specifying a reduction with a different rule. When parsing, each shift

or reduce choice must lead to a different final structure, and so the indeterminacy represents a point of syntactic ambiguity (although it may not correspond to a genuinely global syntactic ambiguity in the input, on account of the limited amount of lookahead).

In the ANLT grammar and lexicon, lexical ambiguity is at least as pervasive as structural ambiguity. A naive implementation of an interactive LR parser would ask the user the correct category for each ambiguous word as it was shifted; many open-class words are assigned upwards of twenty lexical categories by the ANLT lexicon with comparatively fine distinctions between them, so this strategy would be completely impracticable. To avoid asking the user about lexical ambiguity, we use the technique of *preterminal delaying* (Shieber, 1983), in which the assignment of an atomic preterminal category to a lexical item is not made until the choice is forced by the use of a particular production in a later reduce action. After shifting an ambiguous lexical item, the parser enters a state corresponding to the union of states that would be entered on shifting the individual lexical categories. Since, in general, several unification grammar categories for a single word may be subsumed by a single atomic preterminal category, we extend Shieber's technique so that it deals with a grammar containing complex categories by associating a set of alternative analyses with each state (not just one), and letting the choice between them be forced by later reduce actions, just as with atomic preterminal categories.

In order not to overload the user with spurious choices concerning local ambiguities, the parser does not request help immediately it reaches a parse action conflict. Instead the parser pursues each option in a limited breadth-first fashion and only requests help with analysis paths which remain active. In our current system this type of lookahead is limited to up to four indeterminacies ahead. Such checking is cheap in terms of machine resources and very effective in cutting down both the number of choice points the user is forced to consider and also the average number of options in each one. Table 3 shows the reduction in user interaction achieved by increasing the amount of lookahead in our system. Computation of the backbone grammar generates extra rules (as previously described, to deal with lexical categories used as rule mothers and daughters specified to be repeatable an indefinite number of times) which do not correspond directly to single unification grammar rules. At choice points, reductions involving these rules are not presented to the user; instead the system applies the reductions automatically, proceeding until the next shift action or choice point is reached, including these options in those presented to the user.

The final set of measures taken to reduce the amount of interaction required with the user is to ask if the phrase being parsed contains one or more gaps or instances of co-ordination before presenting choices involving either of these phenomena, blocking consideration of rules on the basis of the presence of particular feature-value pairs. The example in (6-1) shows the system parsing the sentence *in an anxious mood he helped the abbot* with a four choice lookahead (user input in bold type). The resulting parse tree is displayed with atomic category aliases substituted for the actual complex categories.

lookahead	number of choices	mean number of options in each choice
none	9	2.8
1 choice	6	2.8
2 choices	5	2.2
3 choices	2	2.0
4 choices	1	2.0

Table 3: Number of choices presented to the user and the mean number of options in each one, parsing *in an anxious mood he helped the abbot* with the ANLT grammar using different amounts of lookahead

(6-1) **Parse>> in an anxious mood he helped the abbot**

```
Are there any gaps in this phrase?  n

Ambiguity in state 89/102/67/58 with (mood he helped the abbot $)
remaining in buffer.  Analysis so far is in, an, anxious.
1:  Shift word 'mood' onto stack.
2:  Reduce end 1 analyses with rule A1/A (giving category
'A1-617').

Which choice (1 - 2 / abort / finish)?  2
8052 msec CPU
3875 unifications, 3226 failures
1 parse

(T
   (S
      (P2
          (P1 in
             (N2 an
                (N2 (N1 (A2 (A1 anxious)) (N1 mood))))))
      (S (N2 he)
         (VP helped (N2 the (N2 (N1 abbot)))))))
```

The requests for manual selection of the analysis path are displayed to the analyst in as terse a manner as possible and require knowledge of the ANLT grammar and lexicon to be resolved effectively. The LDOCE definition for *aconite*, shown in (6-2), consists of 8 words and has 46 parses in the ANLT grammar.

(6-2) a medicine made from one of these plants

For this definition, it takes 6 interactions to construct the correct analysis (illustrated in Figure 5) from the 46 possible parses.

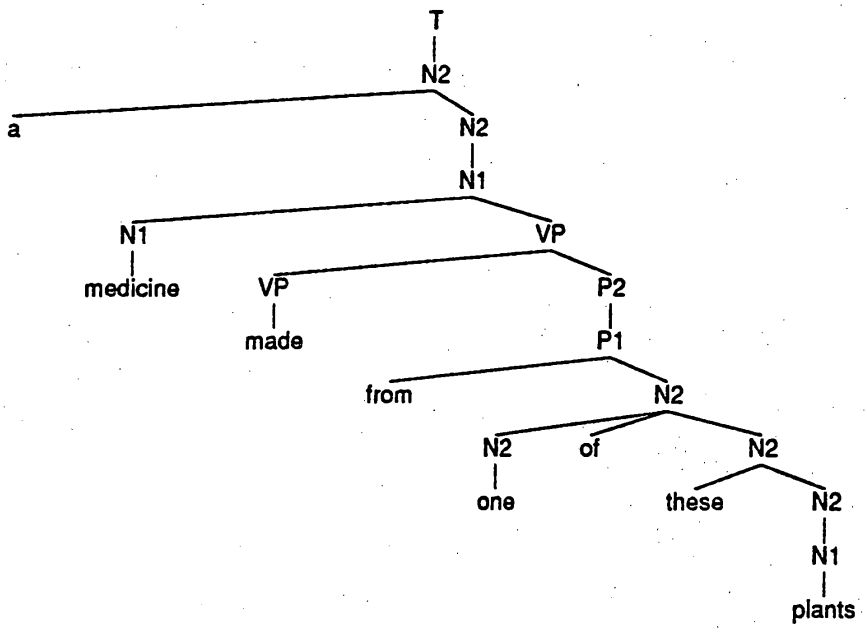

Figure 5: Parse tree for *a medicine made from one of these plants*

This definition is of average complexity in the corpus we are considering; however, definitions can contain more than 30 words, and in these cases there will often be many hundreds or even thousands of parses. In a more general corpus of written material the average sentence length is likely to be 30–40 words. The LDOCE definition for *youth hostel* in (6-3) contains 30 words and is one of the more complex examples we analysed in the experiment reported below.

(6-3) a hostel for usu young people walking around country areas on holiday for which they pay small amounts of money to the youth hostels association or the international yha.

Achieving the correct analysis interactively involved 63 interactions and took the first author about 40 minutes (including the addition of two lexical entries). The parse tree is shown in Figure 6. This example illustrates clearly the problems with any approach based on *post hoc* selection of the correct parse—we have been unable to compute the full set of analyses for this example (on a DEC 3100 Unix workstation). However, using the incremental approach to semi-automatic parsing we have been able to demonstrate that the correct analysis is amongst this set. Furthermore, a probabilistic parser with a beam-search or best-first search regime may well be able to compute this analysis in a tractable fashion. (To date, the largest example for which we have been able to compute all analyses had approximately 2500 analyses.)

The parse histories resulting from semi-automatic parsing are automatically stored and can be used to derive the probabilistic information which will guide the parser after training. We return to a discussion of the manner in which this information is utilised in section 8.

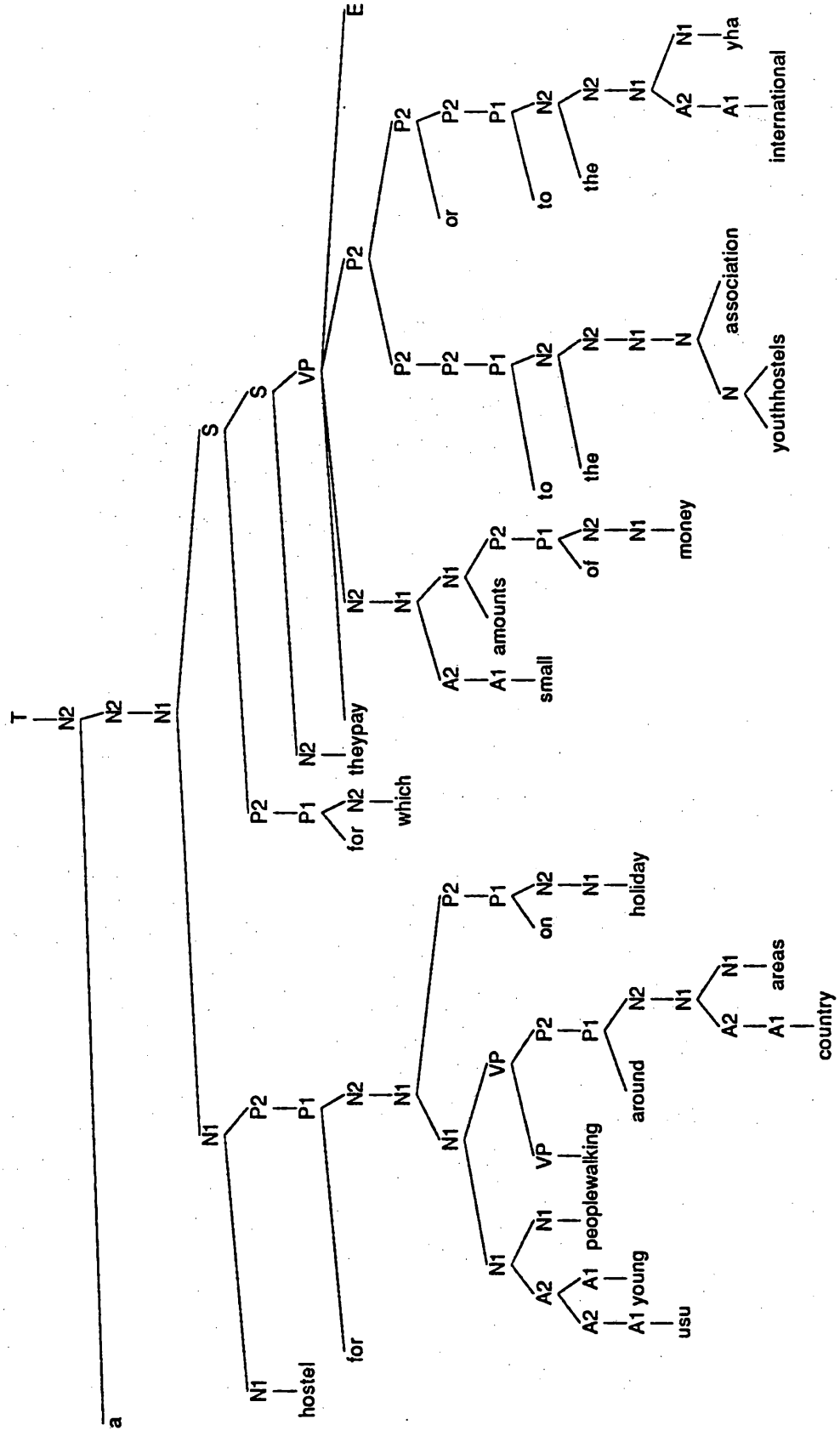

Figure 6: Parse tree for definition of *youth hostel*

7 Non-deterministic Breadth-first LR Parsing with Unification Grammars

As well as building an interactive parsing system incorporating the ANLT grammar (described above), we have also implemented a breadth-first non-deterministic LR parser for unification grammars. This parser is integrated with the Grammar Development Environment (GDE, Carroll et al., 1988) in the ANLT system, and provided as an alternative parser for use with stable grammars for efficient parsing of large bodies of text. The existing chart parser, although slower, has been retained since it is more suited to grammar development, because of the speed with which modifications to the grammar can be compiled and its better debugging facilities (Boguraev et al., 1988).

Our non-deterministic LR parser is based on Tomita's (1987) parsing algorithm and uses a graph-structured stack in the same way. Our parser is driven by the LALR(1) state table computed from the backbone grammar, but in addition on each reduction the parser performs the unifications appropriate to the unification grammar version of the backbone rule involved. The analysis being pursued fails if one of the unifications fails. The parser performs sub-analysis sharing (where if two or more trees have a common sub-analysis, that sub-analysis is only represented once), and local ambiguity packing (in which sub-analyses which have the same top node and cover the same input have their top nodes merged, being treated by higher level structures as a single sub-analysis). However, we generalise the technique of packing described by Tomita, driven by atomic category names, to complex feature-based categories following Alshawi et al. (in press): the packing of sub-analyses is driven by the subsumption relationship between the feature values in their top nodes. An analysis is only packed into one that has already been found if its top node is subsumed by, or is equal to that of the one already found. An analysis, once packed, will thus never need to be unpacked during parsing (as in Tomita's system) since the value of each feature will always be uniquely determined.

We noticed during preliminary experiments with our unification LR parser that it was often the case that the same unifications were being performed repeatedly, even during the course of a single reduce action. The duplication was happening in cases where two or more pairs of states in the graph-structured stack had identical complex categories between them (for example due to backbone grammar ambiguity). During a reduction with a given rule, the categories between each pair of states in a backwards traversal of the stack are collected up and unified with the appropriate daughters of the rule. Identical categories appearing here between traversed pairs of states leads to duplication of unifications. By caching unification results we eliminated this wasted effort and improved the initially poor performance of the parser by a factor of about three.

As for actual parse times, Table 4 compares those for the GDE chart parser, the semi-automatic, user-directed LR parser, and the non-deterministic LR parser. Our general experience is that although the non-deterministic LR parser is only around 30–50% faster than the chart parser, it often generates as little as a quarter

Parser	Parse time
Bottom-up chart	7.4
LR semi-automatic (with 4-choice lookahead)	7.2
LR non-deterministic	5.7

Table 4: Chart and LR parse times for the LDOCE definition *loss of attention to what one is doing* with the ANLT grammar (in CPU seconds on a DEC 3100)

the amount of garbage. (The relatively modest speed advantage compared with the substantial space saving appears to be due to the larger overheads involved in LR parsing). Efficient use of space is obviously an important factor for practical parsing of long and ambiguous texts.

8 LR Parsing with Probabilistic Disambiguation

Several researchers (Wright & Wrigley, 1989; Wright, 1990; Ng & Tomita, 1991; Wright et al., 1991) have proposed using LR parsers as an efficient method of parsing with a probabilistic context-free grammar. This approach assumes that probabilities are already associated with a CFG and describes techniques for distributing those probabilities around the LR parse table in such a way that a probabilistic ranking of alternative analyses can be computed very efficiently at parse time, and probabilities assigned to analyses will be identical to those defined by the original probabilistic CFG. However, our method of constructing the training corpus allows us to associate probabilities with an LR parse table directly, rather than simply with rules of the grammar. An LR parse state encodes information about the left and right context of the current parse. Deriving probabilities relative to the parse context will allow the probabilistic parser to distinguish situations in which identical rules reapply in different ways across different derivations or apply with differing probabilities in different contexts.

Semi-automatic parsing of the training corpus yields a set of LR parse histories which are used to construct the probabilistic version of the LALR(1) parse table. The parse table is a non-deterministic finite-state automaton so it is possible to apply Markov modelling techniques to the parse table (in a way analogous to their application to lexical tagging or CFGs). Each row of the parse table corresponds to the possible transitions out of the state represented by that row and each transition is associated with a particular lookahead item and a parse action. Non-determinism arises when more than one action, and hence transition, is possible given a particular lookahead item. The most straightforward technique for associating probabilities with the parse table is to assign a probability to each action in the action part of the table (e.g. Wright, 1990). If probabilities are associated directly with the parse table, rather than derived from a probabilistic CFG or equivalent global pairing of probabilities to rules, then the resulting probabilistic model will be more sensitive

to context. For example, in the derivation for *he loves her* discussed in section 3, the distinction between reducing the first pronoun and second pronoun to NP (using the same CF production from Grammar 1) can be maintained in terms of the different lookahead items paired with the reduce actions relating to this rule in state 5 of the parse table in Figure 2; in the first case, the lookahead item will be 'Vi', and in the second '$'. However, this approach does not make maximal use of the context encoded into a transition in the parse table and it is possible to devise situations in which the reduction of a pronoun in subject position and elsewhere would be indistinguishable in terms of lookahead alone; for example, if we added appropriate rules for adverbs to Grammar 1, then this reduction would be possible with lookahead 'Adv' in examples like *he passionately loves her* and *he loves her passionately.*

A slightly less obvious approach is to further subdivide reduce actions according to the state reached after the reduce action has applied. This state is used together with the resultant non-terminal to define the state transition in the Goto part of the parse table. Thus, this move corresponds to associating probabilities with transitions in the automaton rather than with actions in the action part of the table. For example, a reduction of pronoun to NP in subject position in the parse table for Grammar 1 in Figure 2 always results in the parser returning to state 0 (from which the Goto table deterministically prescribes a transition to state 7 with non-terminal NP). In the derivation shown in Figure 3 it can be seen that the second pronoun reduction results in the parser returning to state 11. Thus training on a corpus with more subject than non-subject pronominal NPs will now result in a probabilistic preference for reductions which return to 'pre-subject' states with 'post-subject' lookaheads. Of course, this does not mean that it will be impossible to devise grammars in which reductions cannot be kept distinct which might, in principle, have different frequencies of occurrence. However, this approach appears to be the natural stochastic, probabilistic model which emerges from the LR parsing technique. Any further sensitivity to context would require sensitivity to patterns in larger sections of a parse derivation than can be defined in terms of an LR state transition.

The probabilities required to create the probabilistic version of the parse table can be derived from the set of parse histories resulting from the training phase described in section 6, by computing the frequency with which each transition from a particular state has been taken and normalising these such that the probabilities assigned to each transition from a given state sum to one. In Figure 7 we show a probabilistic LALR(1) parse table for Grammar 1 derived from a simple, partial (and artificial) training phase. In this version of the table a probability is associated with each shift action in the standard way, but separate probabilities are associated with reduce actions depending on the state reached after the action; for example, in state 4 with lookahead 'N@' the probability of reducing with rule 10 is .17 if the state reached is 3 and .22 if the state reached is 5. The actions which have no associated probabilities are ones which have not been utilised during the training phase; they have an implicit probability equivalent to one observation for each possible action on a row of the table. For this reason, the explicit probabilities for each row sum to less

than one. The goto part of the table is not shown because it is always deterministic and, therefore, we do not associate probabilities with goto transitions.

The difference between our approach and one based on probabilistic CFG can be brought out by considering various probabilistic derivations using the probabilistic parse table for Grammar 1. Assuming that we are using probabilities simply to rank parses, we can compute the total probability of an analysis by multiplying together the probabilities of each transition we take during its derivation. In Figure 8, we give the two possible complete derivations for an example like *the winter holiday camp closed* consisting of a determiner, three nouns and an intransitive verb. The ambiguity concerns whether the noun compound is left- or right-branching, and, as we saw in section 2 a probabilistic CFG cannot distinguish these two derivations. The probability of each step can be read off the action table and is shown after the lookahead item in the figure.

In step 8 a shift-reduce conflict occurs so the stack 'splits' whilst the left- and right-branching analyses of the noun compound are constructed. The a) branch corresponds to the right-branching derivation and has a probability of 4.3×10^{-8}, whilst that of the left-branching b) derivation is 5.3×10^{-8}. Since the table was constructed from parse histories with a preponderance of left-branching structures this is the desired result. In practice, this technique is able to distinguish and train accurately on 4 of the 5 possible structures for a 4 word noun-noun compound; but it inaccurately prefers a completely left-branching analysis over structures like *((n (n n)) n)*. Once we move to 5 word noun-noun compounds, performance degrades further. However, this level of performance on such structural configurations is likely to be adequate, because correct resolution of most ambiguity in such constructions is likely to be dominated by the actual lexical items which occur in individual examples. Nevertheless, if there are systematic structural tendencies evident in corpora (for example, Frazier's (e.g. 1988) parsing strategies predict a preference for left-branching analyses of such compounds), then the probabilistic model is sensitive enough to discriminate them.

In practice, we take the geometric mean of the probabilities rather than their product to rank parse derivations. This is done to provide a crude form of normalisation of the length of the derivation required to produce competing analyses (see also Magerman & Marcus, 1991b). Otherwise, it would be difficult to prevent the system always developing a bias in favour of analyses involving less rules or equivalently 'smaller' trees, almost regardless of the training material. Of course, the need for this step reflects the fact that, although the model is more context dependent than probabilistic CFG, it is by no means a perfect probabilistic model of NL. For example, the stochastic nature of the model and the fact that the entire left context of a parse derivation is not encoded in LR state information means that the probabilistic model cannot take account of, say, the pattern of resolution of earlier conflicts in the current derivation. Another respect in which the model is incomplete is that we are associating probabilities with the context-free backbone of the unification grammar and, therefore, the probabilistic model has no access to information concerning feature values or feature propagation. One consequence of this is that it is quite possible that, given a particular input, an otherwise highly probable

State	$	Det	N@	P	ProNP	Vi	Vt
0		s3 (.50)			s2 (.50)		
1	r1 (0 .83)						
2	r5			r5 (8 .33)		r5	r5 (0 .50)
3			s4 (1.00)				
4	r10 (3 .11 6 .11)		r10 (3 .17 5 .22)	r10 (3 .11)		r10 (3 .11 5 .11)	
5	r6 (8 .13 11 .13)		s4 (.33)	r6 (11 .13)		r6 (0 .20)	r6
6	r8 (3 .17 5 .17)		r8 (3 .25) s4 (.17)	r8		r8 (3 .17)	r8
7				s8		s13 (.43)	s11 (.43)
8		s3 (.50)			s2 (.50)		
9	r9 (12 .40)			r9 (12 .40) s8		r9	r9
10	r7 (11 .40)			r7 (11 .40)		r7	r7
11		s3 (.75)			s2		
12	r3 (7 .43)			s8 (.43)			
13	r4 (7 .75)						
14	r2 (0 .84)						
15	acc (1.00)						

Figure 7: A probabilistic version of the parse table for Grammar 1

```
 0) 0                    (Det)
 1) 0 Det 3              (N@1)  .50
 2) 0 Det 3 N@ 4         (N@2) 1.00
 3) 0 Det 3 N            (N@2)
 4) 0 Det 3 N 5          (N@2)  .17
 5) 0 Det 3 N 5 N@ 4     (N@3)  .33
 6) 0 Det 3 N 5 N        (N@3)
 7) 0 Det 3 N 5 N 6      (N@3)  .22

 8a) 0 Det 3 N 5 N 6 N@ 4  (Vi)  .17     8b) 0 Det 3 N'          (N@3)
 9a) 0 Det 3 N 5 N 6 N     (Vi)           9b) 0 Det 3 N' 5        (N@3)  .25
10a) 0 Det 3 N 5 N 6 N 6   (Vi)  .06     10b) 0 Det 3 N' 5 N@ 4  (Vi)   .33
11a) 0 Det 3 N 5 N'        (Vi)          11b) 0 Det 3 N' 5 N     (Vi)
12a) 0 Det 3 N 5 N' 6      (Vi)  .08     12b) 0 Det 3 N' 5 N 6   (Vi)   .11
13a) 0 Det 3 N"            (Vi)          13b) 0 Det 3 N"         (Vi)

14) 0 Det 3 N 5         (Vi)    .17
15) 0 NP                (Vi)
16) 0 NP 7              (Vi)    .20
17) 0 NP 7 Vi 13        ($)     .43
18) 0 NP 7 VP           ($)
19) 0 NP 7 VP 14        ($)     .75
20) 0 S                 ($)
21) 0 S 1               ($)     .83
22) 0 T                 ($)
23) 0 T 15              ($)    1.00
```

Figure 8: Parse derivations for *the winter holiday camp closed*

derivation will fail because of unification failure associated with a reduce operation. As long as we only use the probabilistic model to rank successful analyses, this is not particularly problematic. However, it is possible to envisage parser control regimes which attempted, say, beam search, using probabilistic information associated with transitions, which would not yield the desired result given this property. As with any such probabilistic model, it would be possible to enrich it further to create a better approximation; however, most such enhancements would increase the amount of training material required to the point where supervised training, at least, would be quite impractical.

We intend to develop a version of the parser which uses probabilistic information to define a best-first search regime in order to reduce the average computational cost of finding the desired analysis. Nevertheless, the current breadth-first probabilistic parser is more efficient than its non-probabilistic counterpart described in the previous section, since it is able to probabilistically unpack the packed parse forest. However, due to the possibility of unification failures caused by conflicting feature values percolated up a tree from packed constituents we cannot utilise the efficient unpacking algorithm for CF grammars given by Wright et al. (1991) (which allows the first m most probable derivations to be recovered from the parse forest without the need for exhaustive search). Instead we unpack the parse forest starting from the leaves, retaining at most m alternative sub-trees with identical mother categories at each stage. This method entails an exhaustive traversal of the packed forest, but does have the property of limiting the amount of structure that is built.

9 Parsing LDOCE Noun Definitions

In order to test the techniques and ideas described in previous sections, we undertook a preliminary experiment using a subset of LDOCE noun definitions as our test corpus. (The reasons for choosing this corpus are discussed in the introduction.) A corpus of approximately 30,000 noun definitions was created from LDOCE by extracting the definition fields and normalising the definitions to remove punctuation, font control information, and so forth. A lexicon was created for this corpus by extracting the appropriate lemmas and matching these against entries in the ANLT lexicon. The 10,600 resultant entries were loaded into the ANLT morphological system (Ritchie et al., 1987) and this sublexicon and the full ANLT grammar formed the starting point for the training process.

246 definitions, selected without regard for their syntactic form, were parsed semi-automatically using the parser described in section 6. During this process, further rules and lexical entries were created for some examples which failed to parse. 151 were successfully parsed and 63 lexical entries and 14 rules were added. Some of the rules required reflected general inadequacies in the ANLT grammar; for example, we added rules to deal with new partitives and prepositional phrase and verb complementation. However, 7 of these rules cover relatively idiosyncratic properties of the definition sublanguage; for example, the postmodification of pronouns by relative clause and prepositional phrase in examples beginning *something that...*, *that of...*,

parenthetical phrases headed by adverbs, such as *the period... esp the period*, and coordinations without explicit conjunctions ending with *etc*, and so forth. Further special rules will be required to deal with brackets in definitions to cover conventions such as *a man (monk) or woman (nun) who lives in a monastery* which we ignored for this test. Nevertheless, the number of new rules required is not great and the need for most was identified very early in the training process. Lexical entries are more problematic, since there is little sign that the number of new entries required will tail off. However, many of the entries required reflect systematic inadequacies in the ANLT lexicon rather than idiosyncracies of the corpus. It took approximately 1 person / month to produce this training corpus. As a rough guide, it takes an average of 15 seconds to resolve a single interaction with the parser. However, the time a parse takes can often be lengthened by incorrect choices (and the consequent need to back up manually) and by the process of adding lexical entries and occasional rules.

The resultant parse histories were used to construct the probabilistic parser. We tried both the approaches described in section 8, associating probabilities directly with shift or reduce actions in the action table and associating them with transitions involving actions from the action table (to a particular goto state in the case of reduce actions). The resulting parsers were used to reparse the training corpus and the most highly ranked analyses were automatically compared to the original parse histories. We have been unable to reparse in a breadth-first fashion all 151 of those examples parsed manually because of the size of the search space for many of the examples, combined with the fact that the probabilistic parser does not, as yet, use the probabilities to control this search, but merely to rank the resultant analyses. We have reparsed 63 of the 151 original examples—all those of less than 10 words (mean length 5.3). We report detailed results for the parser based on associating probabilities with transitions and discuss the (minor) differences in results below. Of the 63 reparsed definitions, in 47 cases the correct analysis (as defined by the training corpus) is also the most highly ranked. In 11 of the remaining cases the correct analysis is the second most-highly ranked analysis. Of the total of 16 cases where the correct analysis is not the most-highly ranked, in 4 the most-highly ranked analysis is, in fact, correct and the training data is wrong. In each of these cases there is a discrepancy of only one rule. Of the remaining 12 clearly incorrect parses, in 8 cases there is also a one-rule discrepancy between the most probable analysis and the correct one. In 3 of these cases there was an inappropriate structural preference for 'low' or 'local' attachment (see Kimball, 1973). If these results are interpreted in terms of a goodness of fit measure such as that of Sampson et al. (1989), the measure would be better than 98%. If we take correct parse / sentence as our measure then the result is 81%, which is slightly worse than that of Fujisaki et al. (1989).

We also parsed a further 54 LDOCE noun definitions not drawn from the training corpus, each containing up to 10 words. Of these, in 31 cases the correct parse was the most highly ranked, in 5 cases it was the second most highly ranked, in 1 case the third most highly ranked, and in the remaining 17 cases it was not in the first three analyses. This yields a correct parse / sentence measure of 57.4%. Examination of the failures again revealed that a preference for local attachment of postmodifiers

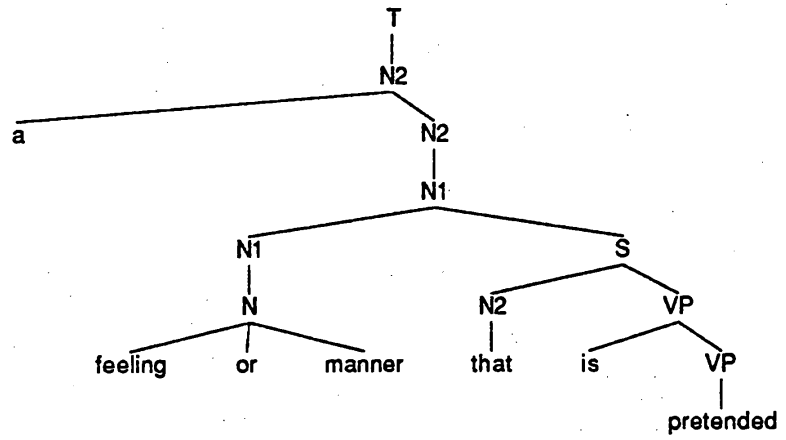

Figure 9: Analysis for *A feeling or manner that is pretended*

accounted for 3 cases, a preference for compounds for 5, and inadequacies in the grammar of coordination for 3. The others are mostly caused by the lack of lexical entries with appropriate SUBCAT features.

Comparison with the more straightforward approach of associating probabilities directly with parse actions revealed a slight advantage for this technique on this test sample – we achieved 85.9% correct parses / sentence for the training sample and 63.8% for the unseen sample. There appears to be no theoretical reason for this result. One possible explanation is that associating probabilities with transitions has the effect of increasing the number of distinct 'operations' to which probabilities are assigned so that given the very limited amount of training material we have created to date, this version of the parser is undertrained by comparison with the one in which probabilities are associated directly with parse actions. Only further experimentation will tell us definitively which is the most successful approach, but the theoretical analysis presented in section 8 suggests that associating probabilities with transitions should provide a more contextually dependent model.

In Figure 9 we show the analysis for the unseen definition of *affectation* which has 45 parses of which the most probable is correct (using both techniques) – this represents one of the most convincing demonstrations of the potential value of the techniques described, so far.

In Figure 10 we show the single analysis assigned to one definition of *anchorage*. This is an example of a false positive which, in this case, is caused by the lack of a lexical entry for *charge* subcategorising it for an infinitival complement. Consequently, the parser finds an analysis in which *charged* is treated as a transitive passive participle, as in *the money charged was wrong*, *to anchor in* is treated as an infinitival relative with a NP gap, as in *a place to anchor in*, and *a harbour* is analysed as a zero relative clause in which *a*, because it can function as a number, is able to occur as a pronoun, and *harbour* is treated as a transitive verb with missing object. Thus the analysis found would be appropriate for *a place visited to anchor*

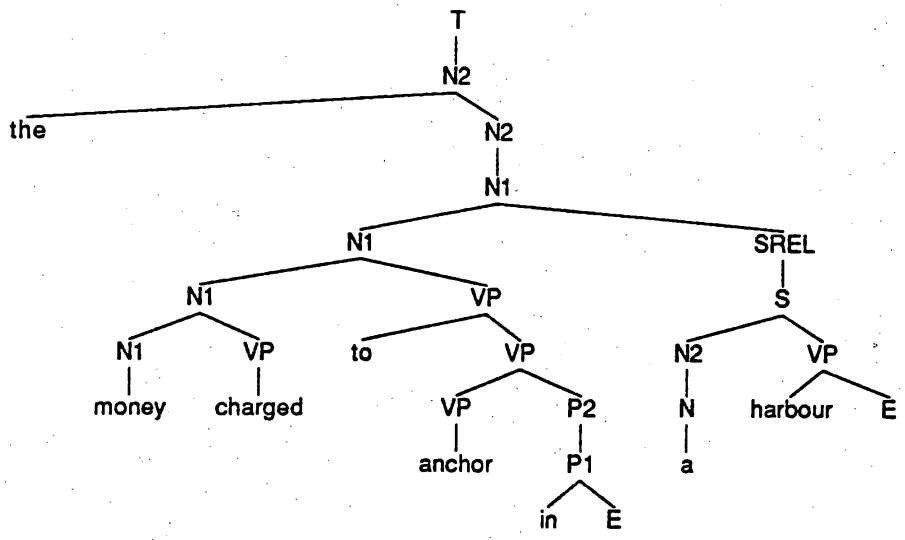

Figure 10: Analysis for *The money charged to anchor in a harbour*

in one likes. Apart for the treatment of *a*, it is difficult to fault this analysis and the same is true for the other (mostly less bizarre) false positives we have looked at. Such false positives present the biggest challenge to the type of system we are attempting to develop. One hopeful sign is that the analyses assigned such examples appear to have low probabilities relative to most probable correct analyses of other examples. However, considerably more data will be required before we can decide whether this trend is robust enough to provide the basis for automatic identification of false positives.

The appendix provides a complete list of the examples parsed and gives the number of parses for each and whether the correct analysis was in the top three most probable.

Using a manually disambiguated training corpus and manually tuned grammar appears feasible with the definitions sublanguage. Results at least comparable to those obtained by Fujisaki et al. (1989) and Sharman et al. (1990) are possible on the basis of a quite modest amount of manual effort and a very much smaller training corpus, because the parse histories contain little 'noise' and do usefully reflect the semantically and pragmatically appropriate analysis in the training corpus, and also because the number of failures of coverage were reduced to some extent by adding the rules specifically motivated by the training corpus. Unlike Fujisaki et al. or Sharman et al., we did not integrate information about lexemes into the rule probabilities or make use of lexical syntactic probability. It seems likely that the structural preference for local attachment might be overruled in appropriate contexts if lexeme (or better, word sense) information were taken into account. The worse results obtained for the unseen data appear to be caused more by the non-existence of a correct analysis in a significant number of cases, rather than by a marked decline in the usefulness of the rule probabilities. This again highlights the need to deal effectively with examples outside the coverage of the grammar.

10 Conclusions and Further Work

The system that we have developed offers partial and practical solutions to two of the three problems of corpus analysis we identified in the introduction. The problem of tuning an existing grammar to a particular corpus or sublanguage is addressed partly by manual extensions to the grammar and lexicon during the semi-automatic training phase and partly by use of statistical information regarding frequency of rule use gathered during this phase. The results of the experiment reported in the last section suggest that syntactic pecularities of a sublanguage or corpus surface quite rapidly, so that manual additions to the grammar during the training phase are practical. However, lexical idiosyncracies are far less likely to be exhausted during the training phase, suggesting that it will be necessary to develop an automatic method of dealing with them. In addition, the current system does not take account of differing frequencies of occurrence of lexical entries; for example, in the LOB corpus the verb *believe* occurs with a finite sentential complement in 90% of citations, although it is grammatical with at least a further five patterns of complementation. This type of lexical information, which will very likely vary between sublanguages, should be integrated into the probabilistic model. This will be straightforward in terms of the model, since it merely involves associating probabilities with each distinct lexical entry for a morpheme and carrying these forward in the computation of the likelihood of each parse. However, the acquisition of the statistical information from which these probabilities can be derived is more problematic. Existing lexical taggers are unable to assign tags which reliably encode subcategorisation information. Therefore, progress on this front must await successful techniques for robust phrasal analysis of corpora (e.g. Magerman & Marcus, 1991a). In addition, it has become clear that the ANLT lexicon and grammar of derivational morphological processes needs to be improved.

The task of selecting the correct analysis from the set licensed by the grammar is also partially solved by the system. It is clear from the results of the preliminary experiment reported in the previous section that it is possible to make the semantically and pragmatically correct analysis highly ranked, and even most highly ranked in many cases, just by exploiting the frequency of occurrence of the syntactic rules in the training data. However, it is also clear that this approach will not succeed in all cases; for example, in the experiment the system appears to have developed a preference for local attachment of prepositional phrases (PPs) which is inappropriate in a significant number of cases. It is not surprising that probabilities based solely on the frequency of syntactic rules are not capable of resolving this type of ambiguity; in an example like *John saw the man on Monday again* it is the temporal interpretation of *Monday* which favours the adverbial interpretation (and thus non-local attachment). Such examples are syntactically identical to ones like *John saw the man on the bus again* in which the possibility of a locative interpretation creates a mild preference for the adjectival reading and local attachment. To select the correct analysis in such cases it will be necessary to integrate information concerning word sense collocations into the probabilistic analysis. We are interested in collocations between the head of a PP complement, a preposition and the head of

the phrase being postmodified. In general, these words will not be adjacent in the text, so it will not be possible to use existing approaches unmodified (e.g. Church & Hanks, 1989), because these apply to adjacent words in unanalysed text.

One way of integrating 'structural' collocational information into the system presented above would be to make use of the semantic component of the (ANLT) grammar. This component pairs logical forms with each distinct syntactic analysis which represent, amongst other things, the predicate-argument structure of the input. In the resolution of PP attachment and similar ambiguities, it is 'collocation' at this level of representation which appears to be most relevant. Integrating a probabilistic ranking of the resultant logical forms with the probabilistic ranking of the distinct syntactic analyses presents no problems, in principle. However, once again, the acquisition of the relevant statistical information will be difficult, because it will require considerable quantities of analysed text as training material. One way to ameliorate the problem might be to reduce the size of the 'vocabulary' for which statistics need to be gathered by replacing lexical items with their superordinate terms (or a disjunction of such terms in the case of ambiguity). Copestake (1990) describes a program capable of extracting the genus term of a definition from an LDOCE definition, resolving the sense of such terms, and constructing hierarchical taxonomies of the resulting word senses. Taxonomies of this form might be used to replace PP complement heads and postmodified heads in corpus data with a smaller number of superordinate concepts. This would make the statistical data concerning trigrams of head-preposition-head less sparse (c.f. Gale & Church, 1990) and easier to gather from a corpus. Nevertheless, it will only be possible to gather such data from determinately syntactically analysed material.

The third problem of dealing usefully with examples outside the coverage of the grammar even after training is not addressed by the system we have developed. Nevertheless, the results of the preliminary experiment for unseen examples indicate that it is a significant problem, a least with respect to lexical entries. A large part of the problem with such examples is identifying them automatically. Some such examples will not receive any parse and will, therefore, be easy to spot. Most though, will receive incorrect parses (one of which will be automatically ranked as the most probable) and can, therefore, only be identified manually for the moment. Jensen et al. (1983) describe an approach to parsing such examples based on parse 'fitting' or rule 'relaxation' to deal with 'ill-formed' input. An approach of this type might work with input which recieves no parse, but cannot help with the identification of those which only receive an incorrect one. In addition, it involves annotating each grammar rule about what should be relaxed and requires that semantic interpretation can be extended to fitted parses.

Garside & Leech (1985) and Sampson et al. (1989) propose more thorough-going probabilistic approaches in which the parser uses a statistically-defined measure of 'closest fit' to the set of analyses contained in a 'treebank' of training data to assign an analysis. This approach attempts to ensure that analyses of new data will conform as closely as possible to existing ones, but does not require that analyses assigned are contained in the set defined by the generative grammar implicit in the treebank analyses. If successful, this approach would solve the three problems of corpus

analysis identified above. Sampson et al. report some preliminary results for a parser of this type which uses the technique of simulated annealing to assign the closest fitting analysis on the basis of initial training on the LOB treebank and automatic updating of its statistical data on the basis of further parsed examples. Sampson et al. give their results in terms of a similarity measure with respect to correct analyses assigned by hand. For a 13 sentence sample the mean similarity measure was 80% and only one example received a fully correct analysis. These results suggest that the technique is not reliable enough for practical corpus analysis, to date. In addition, the analyses assigned, on the basis of the LOB treebank scheme, are not syntactically determinate (for example syntactic relations in unbounded dependency constructions are not represented) and the analyses assigned are not drawn from a prior well-defined set.

In the short term, we intend to continue to resolve the related problems of grammar coverage and false positive examples which have only incorrect parses by manual analysis of the output of the probabilistic parser, manual addition of grammar rules and lexical entries, and incorporation of correct parses into the training materials with periodic re-estimation of probabilities. Using this technique we can also incrementally acquire probabilities concerning lexical entries and structural word sense collocations from the disambiguated training corpus. This 'intermediate' training phase should be feasible because the correct analysis, where it is available, should occur high in the ranked list of possible parses and should, therefore, be identifiable relatively quickly. In addition, once we judge that a good set of initial probabilities has been obtained and the problem of false positives reduced to manageable levels, it should be feasible to move to an unsupervised training regime in which probabilities are automatically re-estimated using the inside-ouside algorithm and only a subset of new material is manually checked in order to build up a growing corpus of correct analyses against which continuing convergence can be repeatedly re-estimated.

In the longer term, we intend to explore techniques for automatic extension of the grammar or parser in the face of failures of coverage in a probabilistic context. The lack of a parse could be used to trigger inductive grammar or parser extension processes within the parse contexts defined by the more highly ranked partial derivations. Within such a 'probabilistic (nearly) deterministic' parse context, induction of appropriate rules may prove feasible (e.g. Berwick, 1985). In addition, experience may show that it is possible to infer the existence of false positives from abnormally low probabilities for the highest ranked parses or alternatively from abnormally low values for sub-analyses within such parses relative to other highly ranked parses for input which does have a correct parse.

In conclusion, the main positive points of the paper are that, 1) LR parse tables can be used to define a more context dependent and adequate probabilistic model of NL, 2) optimally efficient LR parse tables can be constructed automatically from unification-based grammars in standard notation, 3) effective parse table construction and representation techniques can be defined for realistically-sized ambiguous NL grammars, 4) semi-automatic LR based parse techniques can be used to efficiently construct training corpora, and 5) the LR parser and ANLT grammar jointly define an appropriate probabilistic model into which probabilities concerning

lexical subcategorisation and structurally defined word sense collocations could be integrated.

Acknowledgements

This research is supported by SERC / DTI-IED project 4/1/1261 'Extensions to the Alvey Natural Language Tools' and by ESPRIT BRA 3030 'Acquisition of Lexical Information from Machine-readable Dictionaries'. We would like to thank Longman Group Ltd for allowing us access to the LDOCE MRD, David Elworthy for the use of his lexical tagging program, and Ann Copestake and Antonio Sanfilippo for considerable help in the analysis of the LDOCE noun definition corpus. In addition, Hiyan Alshawi and Steve Young have helped clarify our thinking and made several suggestions which have influenced the way this research has developed. All errors and mistakes remain our responsibility.

References

Aho, A., R. Sethi & J. Ullman (1986) *Compilers: Principles, Techniques and Tools,* Addison-Wesley, Reading, Massachusetts.

Alshawi, H. (ed.) (In Press) *The Core Language Engine,* MIT Press, Cambridge, Massachusetts.

Baker, J. (1982) 'Trainable Grammars for Speech Recognition' in D. Klatt & J. Wolf (eds.), *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America,* ASA, pp. 547–550.

Bermudez, M. (1991) 'A unifying model for lookahead LR parsing', *Computer Languages, vol.16.2,* pp. 167–178.

Berwick, R. (1985) *The Acquisition of Syntactic Knowledge,* MIT Press, Cambridge, Massachusetts.

Boguraev, B., J. Carroll, E. Briscoe & C. Grover (1988) 'Software Support for Practical Grammar Development', *Proceedings of the 12th International Conference on Computational Linguistics,* Budapest, Hungary, pp. 54–58.

Briscoe, E. (1987) *Modelling Human Speech Comprehension: A Computational Approach,* Chichester, UK: Ellis Horwood; New York: Wiley.

Briscoe, E., C. Grover, B. Boguraev & J. Carroll (1987) 'A Formalism and Environment for the Development of a Large Grammar of English', *Proceedings of the 10th International Joint Conference on Artificial Intelligence,* Milan, Italy, pp. 703–708.

Carroll, J. & C. Grover (1989) 'The derivation of a large computational lexicon for English from LDOCE' in Boguraev, B. & E. Briscoe (eds.), *Computational Lexicography for Natural Language Processing,* Longman, London, pp. 117–134.

Carroll, J., B. Boguraev, C. Grover & E. Briscoe (1988) *The grammar development environment: a user guide,* Cambridge University, Computer Laboratory, Technical Report 127.

Chapman, N. (1987) *LR Parsing: Theory and Practice,* Cambridge University Press, Cambridge, England.

Chomsky, N. (1957) *Syntactic Structures,* Mouton, The Hague.

Church, K. & P. Hanks (1989) 'Word association norms, mutual information, and lexicography', *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics,* Vancouver, Canada, pp. 76–83.

Church, K. & R. Patil (1982) 'Coping with syntactic ambiguity or how to put the block in the box on the table', *Computational Linguistics, vol.8,* pp. 139–49.

Copestake, A. (1990) 'An approach to building the hierarchical element of a lexical knowledge base from a machine-readable dictionary', *Proceedings of the Workshop on Inheritance in Natural Language Processing,* Tilburg, pp. 19–29.

Culic, K. II & R. Cohen (1973) 'LR-regular grammars – an extension of LR(k) grammars', *Journal of Computer and System Sciences, vol.7,* pp. 66–96.

DeRemer, F. & T. Penello (1982) 'Efficient Computation of LALR(1) Look-Ahead Sets', *ACM Transactions on Programming Languages and Systems, vol.4.4,* pp. 615–649.

De Rose, S. (1988) 'Grammatical category disambiguation by statistical optimization', *Computational Linguistics, vol.14.1,* pp. 31–39.

Frazier, L. (1988) 'Grammar and language processing' in Newmeyer, F. (eds.), *Linguistics: The Cambridge Survey,* Vol 2, Cambridge University Press, Cambridge, England, pp. 14–45.

Fujisaki, T., F. Jelinek, J. Cocke, E. Black & T. Nishino (1989) 'A probabilistic method for sentence disambiguation', *Proceedings of the 1st International Workshop on Parsing Technologies,* Carnegie-Mellon University, Pittsburgh, pp. 105-114.

Gale, W. & K. Church (1990) 'Poor estimates of context are worse than none', *Proceedings of the DARPA Speech and Natural Language Workshop,* Hidden Valley, Pennsylvania, pp. 283–287.

Garside, R. (1987) 'The CLAWS word-tagging system' in Garside, R., G. Leech & G. Sampson (eds.), *The Computational Analysis of English: A Corpus-based Approach,* Longman, London, pp. 30–41.

Garside, R. & F. Leech (1985) 'A probabilistic parser', *Proceedings of the 2nd European Conference of the Association for Computational Linguistics,* Geneva, Switzerland, pp. 166–170.

Garside, R., G. Leech & G. Sampson (1987) *The Computational Analysis of English: A Corpus-based Approach,* Longman, London.

Gazdar, G., E. Klein, G. Pullum & I. Sag (1985) *Generalized Phrase Structure Grammar,* Blackwell, Oxford, England.

Gazdar, G. & C. Mellish (1989) *Natural Language Processing in Lisp,* Addison-Wesley, London.

Grosch, J. (1990) 'Lalr—A generator for efficient parsers', *Software—Practice and Experience, vol.20.11,* pp. 1115–1135.

Grover, C., E. Briscoe, J. Carroll & B. Boguraev (1989) *The Alvey Natural Language Tools Grammar (Second Release),* University of Cambridge, Computer Laboratory, Technical Report 162.

Hindle, D. (1989) 'Acquiring disambiguation rules from text', *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics,* Vancouver, Canada, pp. 118–25.

Jensen, K., G. Heidorn, L. Miller & Y. Ravin (1983) 'Parse fitting and prose fixing: getting a hold on ill-formedness', *Computational Linguistics, vol.9,* pp. 147-153.

Johnson, M. (1989) 'The Computational Complexity of Tomita's Algorithm', *Proceedings of the 1st International Workshop onParsing Technologies,* Carnegie-Mellon University, Pittsburgh, pp. 203–208.

Johnson, S. (1975) *Yacc—yet another Compiler-Compiler,* Computer Science Technical Report 32, Bell Laboratories, Murray Hill, New Jersey.

Kimball, J. (1973) 'Seven principles of surface structure parsing in natural language', *Cognition, vol.2,* pp. 15–47.

Kipps, J. (1989) 'Analysis of Tomita's Algorithm for General Context-Free Parsing', *Proceedings of the 1st International Workshop on Parsing Technologies,* Carnegie-Mellon University, Pittsburgh, pp. 193–202.

Klein, E. & M. Martin (1989) 'The Parser Generating System PGSU', *Software—Practice and Experience, vol.19.11,* pp. 1015–1028.

Kristensen, B. & O. Madsen (1981) 'Methods for Computing LALR(k) Lookahead', *ACM Transactions on Programming Languages and Systems, vol.3.1,* pp. 60–82.

Leech, G. & R. Garside (In Press) 'Running a grammar factory: the production of syntactically analysed corpora or 'treebanks" in *English Computer Corpora: Selected Papers and Bibliography,* Johansson, S. & A. Stenstrom. Mouton de Gruyter, Berlin

Magerman, D. & M. Marcus (1991a) *Parsing a Natural Language Using Mutual Information Statistics,* Pennsylvania University, CIS Dept., Ms..

Magerman, D. & M. Marcus (1991b) 'Pearl: a probabilistic chart parser', *Proceedings of the 2nd International Workshop on Parsing Technologies,* Cancun, Mexico, pp. 193–199.

Ng, S-K. & M. Tomita (1991) 'Probabilistic parsing for general context-free grammars', *Proceedings of the 2nd International Workshop on Parsing Technologies,* Cancun, Mexico, pp. 154–163.

Osborne, M. (1990) *Corpus Parsing,* M.Phil. dissertation, University of Cambridge, England.

Pereira, F. & S. Shieber (1987) *Prolog and Natural Language Analysis,* University of Chicago Press, Chicago.

Pereira, F. & D. Warren (1980) 'Definite clause grammars for language analysis – a survey of the formalism and a comparison with augmented transition networks', *Artificial Intelligence, vol.13.3,* pp. 231–278.

Pollard, C. & I. Sag (1987) *Information-based Syntax and Semantics: Volume 1 – Fundamentals,* Chicago University Press, Chicago.

Proctor, P. (ed.) (1978) *The Longman Dictionary of Contemporary English,* Longman, London.

Ritchie, G., S. Pulman, G. Russell & A. Black (1987) 'A computational framework for lexical description', *Computational Linguistics, vol.13,* pp. 290–307.

Sampson, G. (1987) 'The grammatical database and parsing scheme' in Garside, R., G. Leech & G. Sampson (eds.), *The Computational Analysis of English: A Corpus-based Approach,* Longman, London, pp. 82–96.

Sampson, G., R. Haigh & E. Atwell (1989) 'Natural language analysis by stochastic optimization: a progress report on Project APRIL', *Journal of Experimental and Theoretical Artificial Intelligence, vol.1,* pp. 271–287.

Sharman, R. (1989) *Probabilistic ID/LP Grammars for English,* Report 217, IBM UK Scientific Centre, Winchester, England.

Sharman, R., F. Jelinek & R. Mercer (1990) 'Generating a grammar for statistical training', *Proceedings of the DARPA Speech and Natural Language Workshop,* Hidden Valley, Pennsylvania, pp. 267–274.

Shieber, S. (1983) 'Disambiguation by a Shift-Reduce Parsing Technique', *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics,* MIT, Cambridge, Massachusetts, pp. 113–118.

Shieber, S. (1984) 'The Design of a Computer Language for Linguistic Information', *Proceedings of the 10th International Conference on Computational Linguistics,* Stanford, California, pp. 362–366.

Shieber, S. (1985) 'Using restriction to extend parsing algorithms for complex feature-based formalisms', *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics,* Chicago, pp. 145–152.

Taylor, L., C. Grover & E. Briscoe (1989) 'The syntactic regularity of English noun phrases', *Proceedings of the 4th European Meeting of the Association for Computational Linguistics,* UMIST, Manchester, pp. 256–263.

Tomita, M. (1984) 'Parsers for Natural Languages', *Proceedings of the 10th International Conference on Computational Linguistics,* Stanford, California, pp. 354–357.

Tomita, M. (1987) 'An efficient augmented-context-free parsing algorithm', *Computational Linguistics, vol.13.1,* pp. 31–46.

Wright, J. (1990) 'LR parsing of probabilistic grammars with input uncertainty for speech recognition', *Computer Speech and Language, vol.4,* pp. 297–323.

Wright, J. & E. Wrigley (1989) 'Probabilistic LR parsing for speech recognition', *Proceedings of the 1st International Workshop on Parsing Technologies,* Carnegie-Mellon University, Pittsburgh, pp. 193–202.

Wright, J., E. Wrigley & R. Sharman (1991) 'Adaptive probabilistic generalized LR parsing', *Proceedings of the 2nd International Workshop on Parsing Technologies,* Cancun, Mexico, pp. 154–163.

Zeevat, H., J. Calder & E. Klein (1987) 'Unification Categorial Grammar' in *Categorial Grammar, Unification Grammar, and Parsing,* Edinburgh WP in Cognitive Science 1, Edinburgh University.

Appendix

Below are the results of the experiment with LDOCE noun definitions. Column one: ranking of correct parse; column two: number of analyses.

Examples reparsed from training corpus

```
2 3  freedom from control
1 3  monastery or convent
1 16 the group of people living in such a building
2 3  the alphabet as taught to children
1 2  sudden forgetfulness
1 1  an example of this
0 2  the act of washing oneself
1 3  an australian aborigine
1 4  place where one lives
1 2  the act of putting an end to something
1 3  great hatred
2 25 an example of this by accident or intention
2 4  loss of surface by rubbing
1 1  the act of making shorter
1 4  loss of attention to what one is doing
1 3  a person who stays away
1 1  a glass of this
1 1  a member of an academy
1 2  the act of increasing speed
1 4  the act of accepting or of being accepted
2 3  in business an agreement to pay
1 1  means of entering
1 1  way in
1 1  something added
1 1  an addition
1 7  agreement as to a demand
1 5  strong expressions of approval and praise
1 5  loud shouts of welcome and honour
1 2  an upward slope
1 4  strong praise and approval
1 1  something that helps or makes an action easier
1 3  a person who plays a musical accompaniment
1 8  the act of finishing work completely and successfully
1 2  something completely and successfully done
1 2  a lady-like art
1 11 an agreement between countries businesses etc
1 1  the quality of being accurate
1 15 bitterness of speech temper or manner
1 1  an example of this
```

```
2 5  a chemical made from acetic acid
1 2  a continuous pain
2 6  the quality of being acid
1 6  something esp a liquid with an acid taste
0 31 an antiaircraft gun or fire from such a gun
1 46 a medicine made from one of these plants
1 16 the group of people living in such a building
1 1  the act of making shorter
2 2  powers and skills esp of the mind
1 9  the lowest temperature that is thought to be possible
1 3  the wood of this tree
1 1  a jew
1 2  power control etc
1 29 a simple or foolish country man
2 4  the yellow central part of an egg
1 7  a very long time
2 9  a type of short dog with fairly long hair
2 20 young people considered as a group
1 3  young animals
1 7  a young person esp a boy
1 2  early life
0 63 young men and women considered as a group
0 5  a long loud cry
0 19 something that binds people or things together
```

Parses of unseen material

```
1 1  something needing action
1 3  a happening
0 12 behaviour which is not one 's natural manner
1 45 a feeling or manner that is pretended
1 1  something which causes suffering
1 1  a person from africa
1 10 the power or force which causes a result
1 7  one of the periods of life
1 3  a long time
2 4  something that annoys one
0 29 a group or mass formed by this
1 7  thinking in the same way
0 52 a person or thing that supports or helps
1 19 the act of directing a weapon remark etc
1 16 a dog with a rough coat
2 2  type of large terrier
0 27 the act of allowing fresh air into a room
0 15 any letter that is sent by air
```

1 6 a large passenger aircraft
0 16 letters parcels etc sent by air
1 7 the system of sending things by air
2 3 the speed at which an aircraft travels through the air
1 22 the sky or the space above the ground
2 47 the sky as something through which to fly
1 5 a very gentle wind
2 5 an attack by military aircraft
0 26 a case or example of this work
1 3 a female airman
0 14 an animal or plant that lacks the typical colouring
1 15 the white or colourless part of an egg
0 1 a person who studied or practised alchemy
1 1 the drinks containing this
1 2 false name
1 9 the muslim name for god
0 15 a share as of money or space
0 12 money provided for a special purpose
0 4 a person who helps or supports one
1 1 something that attracts or charms
1 3 quantity or sum
1 10 total quantity or sum
0 90 a sexual relationship esp one that is secret
1 2 large amount
1 4 the state of being amused
3 12 the act or action of producing this state
0 5 a person who believes in anarchism
1 4 absence of government or control
1 1 absence of order
1 1 something hated
0 11 a person skilled in anatomy
1 1 a place where ships may anchor
0 1 the money charged to anchor in a harbour
1 11 a means of making firm
1 2 an old man
0 7 a person who helps another in work