Experiments with Corpus-based LFG Specialization

Nicola Cancedda and Christer Samuelsson

Xerox Research Centre Europe,
6, chemin de Maupertuis
38240 Meylan, France
{cancedda|samuelsson}@xrce.xerox.com

Abstract

Sophisticated grammar formalisms, such as LFG, allow concisely capturing complex linguistic phenomena. The powerful operators provided by such formalisms can however introduce spurious ambiguity, making parsing inefficient. A simple form of corpus-based grammar pruning is evaluated experimentally on two wide-coverage grammars, one English and one French. Speedups of up to a factor 6 were obtained, at a cost in grammatical coverage of about 13%. A two-stage architecture allows achieving significant speedups without introducing additional parse failures.

1 Introduction

Expressive grammar formalisms allow grammar developers to capture complex linguistic generalizations concisely and elegantly, thus greatly facilitating grammar development and maintenance. (Carrol, 1994) found that the empirical performance when parsing with unification-based grammars is nowhere near the theoretical worst-case complexity. Nonetheless, directly parsing with such grammars, in the form they were developed, can be very inefficient. For this reason, grammars are typically compiled into representations that allow faster parsing. This does however not solve the potential problem of the grammars overgenerating considerably, thus allowing large amounts of spurious ambiguity. Indeed, a current trend in high-coverage parsing, especially when employing a statistical model of language, see, e.g., (Collins 97), is to allow the grammar to massively overgenerate and instead disambiguate by statistical means during or after parsing. If the benefits resulting from more concise grammatical descriptions are to outweigh the costs of spurious ambiguity, the latter must be brought down.

In such a situation, corpus-based compilation techniques can drastically improve parsing performance without burdening the grammar developer. The initial, and much seminal work in this area was been carried out by Rayner and coworkers, see (Rayner 1988), (Samuelsson and Rayner 91) and (Rayner and Carter 1996). In the current article,

we apply similar ideas to Lexical Functional Grammar (LFG) in the incarnation of the Xerox Linguistic Environment (XLE). The goal is to investigate to what extent corpus-based compilation techniques can reduce overgeneration and spurious ambiguity, and increase parsing efficiency, without jeopardizing coverage. The rest of the article is organized as follows: Section 2 presents the relevant aspects of the LFG formalism and the pruning strategy employed, Section 3 describes the experimental setup, Section 4 reports the experimental results and Section 5 relates this to other work.

2 LFG and Grammar Pruning

The LFG formalism (Kaplan and Bresnan, 1982) allows the right-hand sides (RHS) of grammar rules to consist of a regular expression over grammar symbols. This makes it more appropriate to refer to the grammar rules as *rule schemata*, since each RHS can potentially be expanded into a (possibly infinite) number of distinct sequences of grammar symbols, each corresponding to a traditional phrase-structure rule. As can easily be imagined, the use of regular-expression operators such as *Kleene-star* and *complementation* may introduce a considerable amount of spurious ambiguity. Moreover, the LFG formalism provides operators which — although not increasing its theoretical expressive power — allow rules to be written more concisely. Examples of such operators are the *ignore* operator, which allows skipping any sequence of grammar symbols that matches a given pattern; the *shuffle* operator, which allows a set of grammar symbols to occur in any order; and the *linear precedence* operator, which allows partially specifying the order of grammar symbols.

The pruning method we propose consists in eliminating complex operators from the grammar description by considering how they were actually instantiated when parsing a corpus. In LFGs, each rule scheme corresponds to a particular grammar symbol, since different expansions of the same symbol are expressed as alternatives in the regular expression on its RHS. We can define a specific path through the RHS of a rule scheme by the choices

made when matching it against some sequence of grammar symbols. Our training data allows us to derive, for each training example, the choices made at each rule expansion. By applying these choices to the rule scheme in isolation, we can derive a phrase-structure rule from it.

The grammar is specialized, or *pruned*, by retaining all and only those phrase-structure rules that correspond to a path taken through a rule scheme when expanding some node in some training example. Since the grammar formalism requires that each LHS occur only in one rule scheme in the grammar, extracted rules with the same LHS symbol are merged into a single rule scheme with a disjunction operator at its top level. For instance, if a rule scheme with the structure

$$A \rightarrow B^* \{C \mid D\}$$

is expanded in the training data only in the following ways

$$A \rightarrow C$$
$$A \rightarrow BC$$
$$A \rightarrow BD$$

then it will be replaced by a rule scheme with the following structure

$$A \rightarrow \{C \mid BC \mid BD\}$$

The same approach is taken to replace all regular-expression operators, other than concatenation, with the actual sequences of grammar symbols that are matched against them. A more realistic example, taken from the actual data, is shown in Figure 1: none of the optional alternative portions following the V is ever used in any correct parse in the corpus. Moreover, the $ADVP$ preceding the V occurs only 0 or 1 times in correct parses.

Like other unification-based formalisms, lexical functional grammars allow grammar rules to be annotated with sets of feature-based constraints, here called "functional descriptions", whose purpose is both to enforce additional constraints on rule applicability and to build an enriched predicate-argument structure called "f-structure", which, together with the parse tree, constitutes the output of the parsing process. As these constraints are maintained verbatim in the specialized version of the rule scheme, this poses no problem for this form of grammar pruning.

3 Experimental Setup

The experiments carried out to determine the effectiveness of corpus-based specialization were performed as illustrated in Figure 2. Two broad-coverage LFG grammars were used, one for French and one for English, both of which were developed within the Pargram project (Butt et al., 1999) during several years time. The French grammar consists of 133 rule schemata, the English grammar of 85 rule schemata.

Each grammar is equipped with a treebank, which was developed for other purposes than grammar specialization. Each treebank was produced by letting the system parse a corpus of technical documentation. Any sentence that did not obtain any parse was discarded. At this point, the French corpus was reduced to 960 sentences, and the English corpus to 970. The average sentence length was 9 for French and 8 for English. For each sentence, a human expert then selected the most appropriate analysis among those returned by the parser.

In the current experiments, each treebank was used to specialize the grammar it had been developed with. A set of 10-fold cross-validation experiments was carried out to measure several interesting quantities under different conditions. This means that, for each language, the corpus was randomly split into ten equal parts, and one tenth at a time was held out for testing while the remaining nine tenths were used to specialize the grammar, and the results were averaged over the ten runs. For each grammar the average number of parses per sentence, the fraction of sentences which still received at least one parse (*anyparse*) and the fraction of sentences for which the parse selected by the expert was still derived (*coverage*) were measured[1]. The average CPU time required by parsing was also measured, and this was used to compute the *speedup* with respect to the original grammar.

The thus established results constitute one data point in the trade-off between ambiguity reduction on one side, which is in turn related to parsing speed, and loss in coverage on the other. In order to determine other points of this trade-off, the same set of experiments was performed where specialization was inhibited for certain rule schemata. In particular, for each grammar, the two rule schemata that received the largest number of distinct expansions in the corpora were determined. These proved to be those associated with the LHS symbols 'VPverb[main]' and 'NP' for the French grammar, and 'VPv' and 'NPadj' for the English one.[2] The experiments were repeated while inhibiting specialization of first the scheme with the most expansions, and then the two most expanded schemata.

Measures of coverage and speedup are important

---

[1] As long as we are interested in preserving the f-structure assigned to sentences, this notion of coverage is stricter than necessary. The same f-structure can in fact be assigned by more than one parse, so that in some cases a sentence is considered out of coverage even if the specialized grammar assigns to it the correct f-structure.

[2] 'VPv' and 'VPverb[main]' cover VPs headed by a main verb. 'NPadj' covers NPs with adjectives attached.

The original rule:

$$
VPperfp \rightarrow \begin{array}{ll}
ADVP* & V \\
\downarrow \in (\uparrow ADJUNCT) & \{@M\_Head\_Perfp \ |@M\_Head\_Passp\} \\
(\downarrow ADV\_TYPE) = vpadv & @(Anaph\_Ctrl \uparrow)
\end{array}
$$

$$
\left(\left\{ \begin{array}{l}
ADVP+ \\
\downarrow \in (\uparrow ADJUNCT) \\
(\downarrow ADV\_TYPE) = vpadv
\end{array} \right. \left| \begin{array}{l} PP \\ @PPadjunct \end{array} \right. \left| \begin{array}{l} PPcase \\ @PPcase\_obl \end{array} \right\} \right)
$$

is replaced by the following:

$$
VPperfp \rightarrow \left\{ \begin{array}{ll}
ADVP & V \\
\downarrow \in (\uparrow ADJUNCT) & \{@M\_Head\_Perfp \ |@M\_Head\_Passp\} \\
(\downarrow ADV\_TYPE) = vpadv & @(Anaph\_Ctrl \uparrow) \\
\\
\left| \begin{array}{l} V \\ \{@M\_Head\_Perfp \ |@M\_Head\_Passp\} \\ @(Anaph\_Ctrl \uparrow) \end{array} \right.
\end{array} \right\}
$$

Figure 1: The pruning of a rule from the actual French grammar. The "*" and the "+" signs have the usual interpretation as in regular expressions. A sub-expression enclosed in parenthesis is optional. Alternative sub-expressions are enclosed in curly brackets and separated by the "|" sign. An "@" followed by an identifier is a macro expansion operator, and is eventually replaced by further functional descriptions.
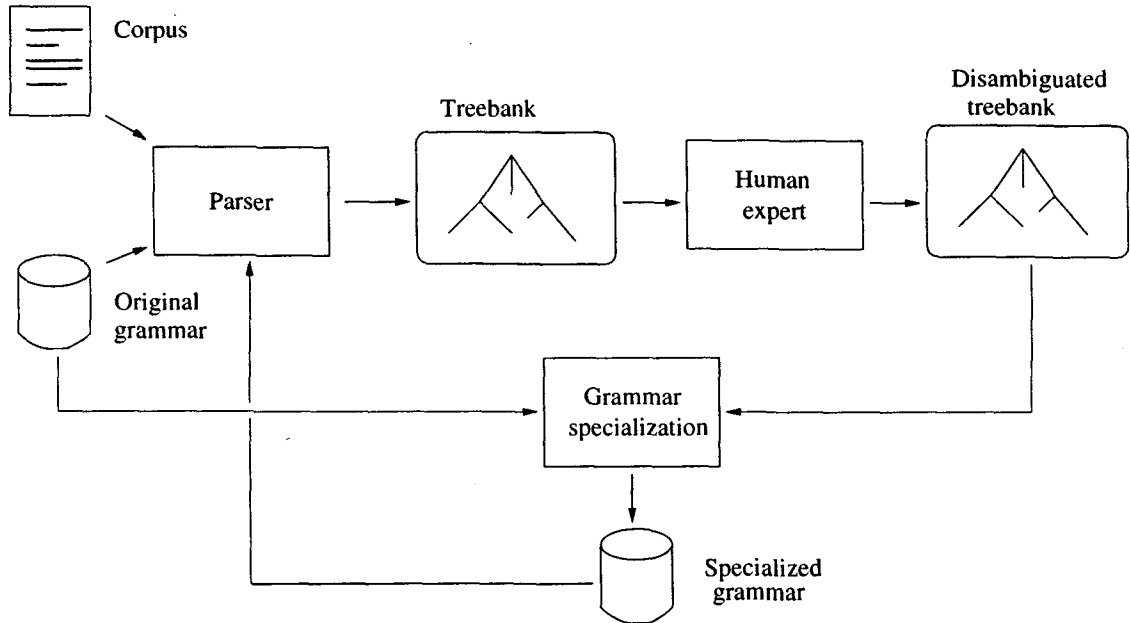

Figure 2: The setting for our experiments on grammar specialization.

indicators of what can be achieved with this form of grammar pruning. However, they could potentially be misleading, since failure times for uncovered sentences might be considerably lower than their parsing times, had they not been out of coverage. If the pruned grammar fails more frequently on sentences which take longer to parse, the measured speedup might be artificially high. This is easily

realized, as simply removing the hardest sentences from the corpus would cause a decrease in the average parsing time, and thus result in a speedup, without any pruning at all. To factor out the contribution of uncovered sentences from the results, the performance of a two-stage architecture analogous to that of (Samuelsson and Rayner, 1991) was simulated, in which the pruned grammar is attempted

"A Sentence"

Parser with specialized
grammar

Fails   Succeeds

Time = Time$_{specialized}$

Parser with original
grammar

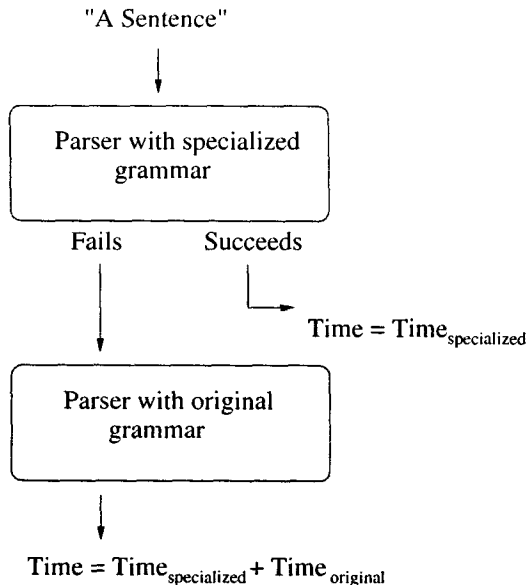Time = Time$_{specialized}$ + Time$_{original}$

Figure 3: A schematic representation of the simulated two-stage coverage-preserving architecture.

first, and the sentence is passed on to the original unpruned grammar whenever the pruned grammar fails to return a parse (see Figure 3). The measured speedup of this simulated architecture, which preserves the *anyparse* measure of the original grammar, takes into account the contribution of uncovered sentences, as it penalizes sweeping difficult sentences under the carpet.

4 Experimental Results

The results of the experiments described in the section above are summarized in the table in Figure 4. The upper part of the table refers to experiments with the French grammar, the lower part to experiments with the English grammar. For each language, the first line presents data gathered for the original grammar for comparison with the pruned grammars. The figures in the second line were collected by pruning the grammar based on the whole corpus, and then testing on the corpus itself. The grammars obtained in this way contain 516 and 388 disjuncts — corresponding to purely concatenative rules — for French and English respectively. *Anyparse* and *coverage* are not, of course, relevant in this case, but the statistics on parsing time are, especially the one on the maximum parsing time. For each iteration in the 10-fold cross-validation experiment, the maximum parsing time was retained, and those ten times were eventually averaged. If pruning tended to leave sentences which take long to parse uncovered, then we would observe a significant difference between the average over maximum times on the grammar trained and tested on the same corpus

(which parses all sentences, including the hardest), and the average over maximum times for grammars trained and tested on different sets. The fact that this does not seem to be the case indicates that pruning does not penalize difficult sentences. Note also that the average number of parses per sentence is significantly smaller than with the full grammar, of almost a factor of 9 in the case of the French grammar.

The third line contains results for the fully pruned grammar. In the case of the French grammar a speedup of about 6 is obtained with a loss in coverage of 13%. The smaller speedup gained with the English grammar can be explained by the fact that here, the parsing times are lower in general, and that a non-negligible part of this time, especially that needed for morphological analysis, is unaffected by pruning. Even in the case of the English grammar, though, speedup is substantial (2.67). For both grammars, the reduction in the average maximum parsing time is particularly good, confirming our hypothesis that trimming the grammar by removing heavy constructs makes it considerably more efficient. A partially negative note comes from the average number of disjuncts in the pruned grammars, which is 501 for French and 374 for English. Comparing this figures to the number of disjuncts in grammars pruned on the full corpus (516 and 388), we find that after training on nine tenths of the corpus, adding the last tenth still leads to an increase of 3-4% in the size of the resulting grammars. In other words, the marginal gain of further training examples is still significant after considering about 900 sentences, indicating that the training corpora are somewhat too small.

The last two lines for each language show figures for grammars with pruning inhibited on the most variable and the two most variable symbols respectively. For both languages, inhibiting pruning on the most variable symbol has the expected effect of increasing both parsing time and coverage. Inhibiting pruning also on the second most variable symbol has almost no effect for French, and only a small effect for English.

The table in Figure 5 summarizes the measures on the simulated two-stage architecture. For both languages the best trade-off, once the distribution of uncovered sentences has been taken into account, is achieved by the fully pruned grammars.

5 Related Work

The work presented in the current article is related to previous work on corpus-based grammar specialization as presented in (Rayner, 1988; Samuelsson and Rayner, 1991; Rayner and Carter, 1996; Samuelsson, 1994; Srinivas and Joshi, 1995; Neumann, 1997).

	Parses/sentence	Anyparse	Coverage	Avg. time (secs.)	Max. time (secs.)	Speedup
French						
original grammar	1941	1.00	1.00	1.52	78.5	1
test = training	219	1.00	1.00	0.28	5.62	5.43
First-order pruning	164	0.91	0.87	0.25	5.69	6.08
no pruning on 'VPverb[main]'	1000	0.94	0.91	0.42	8.70	3.62
no pruning on 'Vpverb[main]' and 'NP'	1279	0.94	0.92	0.42	8.42	3.62
English						
original grammar	58	1.00	1.00	0.56	31.73	1
test = training	24	1.00	1.00	0.23	3.92	2.43
First-order pruning	21	0.94	0.88	0.21	3.92	2.67
no pruning on 'VPv'	25	0.96	0.91	0.32	11.06	1.75
no pruning on 'Vpv' and 'NPadj'	31	0.96	0.93	0.35	11.16	1.60

Figure 4: The results of the experiments on LFG specialization.

	Avg. CPU time (secs.)	Speedup
French		
First-order pruning	0.570	2.67
no pruning on 'VPverb[main]'	0.616	2.47
no pruning on 'VPverb[main]' and 'NP'	0.614	2.48
English		
First-order pruning	0.311	1.81
no pruning on 'VPv'	0.380	1.47
no pruning on 'VPv' and 'NPadj'	0.397	1.40

Figure 5: Results for the simulated two-stage architecture.

The line of work described in (Rayner, 1988; Samuelsson and Rayner, 1991; Rayner and Carter, 1996; Samuelsson, 1994) deals with unification-based grammars that already have a purely-concatenative context-free backbone, and is more concerned with a different form of specialization, consisting in the application of explanation-based learning (EBL). Here, the central idea is to collect the most frequently occurring subtrees in a treebank and use them as atomic units for parsing. The cited works differ mainly in the criteria adopted for selecting subtrees from the treebank. In (Rayner, 1988; Samuelsson and Rayner, 1991; Rayner and Carter, 1996) these criteria are handcoded: all subtrees satisfying some properties are selected, and a new grammar rule is created by flattening each such subtree, i.e., by taking the root as left-hand side and the yield as right-hand side, and in the process performing all unifications corresponding to the thus removed internal nodes. Experiments carried out on a corpus of 15,000 trees from the ATIS domain using a version of the SRI Core Language Engine resulted in a speedup of about 3.4 at a cost of 5% in grammatical coverage, which however was compensated by an increase in parsing accuracy.

Finding suitable tree-cutting criteria requires a considerable amount of work, and must be repeated for each new grammar and for each new domain to which the grammar is to be specialized. Samuelsson (Samuelsson, 1994) proposes a technique to automatically selects what subtrees to retain. The selection of appropriate subtrees is done by choosing a subset of nodes at which to cut trees. Cutnodes are determined by computing the *entropy* of each node, and selecting only those nodes whose entropy exceeds a given threshold. Intuitively, nodes with low entropy indicate locations in the trees where a given symbol was expanded using a predictable set of rules, at least most of the times, so that the loss of coverage that derives from ignoring the remaining cases is low. Nodes with high entropy, on the other hand, indicate positions in which there is a high uncertainty in what rule was used to expand the symbol, so that it is better to preserve all alternatives. Several schemas are proposed to compute entropies, each leading to a different trade-off be-

tween coverage reduction and speedup. In general, results are not quite as good as those obtained using handcoded criteria, though of course the specialized grammar is obtained fully automatically, and thus with much less effort.

When ignoring issues related to the elimination of complex operators from the RHS of rule schemata, the grammar-pruning strategy described in the current article is equivalent to explanation-based learning where all nodes have been selected·as cutnodes. Conversely, EBL can be viewed as higher-order grammar pruning, removing not grammar rules, but grammar-rule combinations.

Some of the work done on data-oriented parsing (DOP) (Bod, 1993; Bod and Scha, 1996; Bod and Kaplan, 1998; Sima'an, 1999) can also be considered related to our work, as it can be seen as a way to specialize in an EBL-like way the (initially unknown) grammar implicitly underlying a treebank.

(Srinivas and Joshi, 1995) and (Neumann, 1997) apply EBL to speed up parsing with tree-adjoining grammars and sentence generation with HPSGs respectively, though they do so by introducing new components in their systems rather then by modifying the grammars they use.

6   Conclusions

Sophisticated grammar formalisms are very useful and convenient when designing high-coverage grammars for natural languages. Very expressive grammatical constructs can make the task of developing and maintaining such a large resource considerably easier. On the other hand, their use can result in a considerable increase in grammatical ambiguity. Grammar-compilation techniques based on grammar structure alone are insufficient remedies in those cases, as they cannot access the information required to determine which alternatives to retain and which alternatives to discard.

The current article demonstrates that a relatively simple pruning technique, employing the kind of reference corpus that is typically used for grammar development and thus often already available, can significantly improve parsing performance. On large lexical functional grammars, speedups of up to a factor 6 were observed, at the price of a reduction in grammatical coverage of about 13%. A simple two-stage architecture was also proposed that preserves the *anyparse* measure of the original grammar, demonstrating that significant speedups can be obtained without increasing the number of parsing failures.

Future work includes extending the study of corpus-based grammar specialization from first-order grammar pruning to higher-order grammar pruning, thus extending previous work on explanation-based learning for parsing, and apply-

ing it to the LFG formalism.

References

Rens Bod and Ronald Kaplan. 1998. A probabilistic corpus-driven model for lexical-functional analysis. In *Proceedings of Coling-ACL-98*, Montreal, Canada.

R. Bod and R. Scha. 1996. Data-oriented language processing: An overview. Technical report, ILLC, University of Amsterdam, Amsterdam, The Netherlands.

Rens Bod. 1993. Using an annotated corpus as a stochastic grammar. In *Proceedings of EACL-93*, Utrecht, The Netherlands.

M. Butt, T.H. King, M.E. Niño, and F. Segond. 1999. *A Grammar Writer's Cookbook*. CSLI Publications, Stanford, CA.

John Carrol. 1994. Relating complexity to practical performance in parsing with wide-coverage unification grammars. In *Proceedings of (ACL'94)*, Las Cruces, New Mexico, June.

Ronald Kaplan and Joan Bresnan. 1982. Lexical-functional grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press.

Günter Neumann. 1997. Applying explanation-based learning to control and speeding-up natural language generation. In *Proceedings of ACL-EACL-97*, Madrid, Spain.

Manny Rayner and David Carter. 1996. Fast parsing using pruning and grammar specialization. In *Proceedings of the ACL-96*, Santa Cruz, CA.

Manny Rayner. 1988. Applying explanation-based generalization to natural-language processing. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, Tokyo, Japan.

Christer Samuelsson and Manny Rayner. 1991. Quantitative evaluation of explanation-based learning as an optimization tool for a large-scale natural language system. In *Proceedings of the IJCAI-91*, Sydney, Oz.

Christer Samuelsson. 1994. Grammar specialization through entropy thresholds. In *Proceedings of the ACL-94*, Las Cruces, New Mexico. Available as cmp-lg/9405022.

Khalil Sima'an. 1999. *Learning Efficient Disambiguation*. Ph.D. thesis, Institute for Logic, Language and Computation, Amsterdam, The Netherlands.

B. Srinivas and A. Joshi. 1995. Some novel applications of explanation-based learning to parsing lexicalized tree-adjoining grammars. In *Proceedings of the ACL-95*, Cambridge, MA.

209