In: *Proceedings of CoNLL-2000 and LLL-2000*, pages 7–12, Lisbon, Portugal, 2000.

Corpus-Based Grammar Specialization

Nicola Cancedda and **Christer Samuelsson**
Xerox Research Centre Europe
6, chemin de Maupertuis
38240, Meylan, France
{Nicola.Cancedda, Christer.Samuelsson}@xrce.xerox.com

Abstract

Broad-coverage grammars tend to be highly ambiguous. When such grammars are used in a restricted domain, it may be desirable to specialize them, in effect trading some coverage for a reduction in ambiguity. Grammar specialization is here given a novel formulation as an optimization problem, in which the search is guided by a global measure combining coverage, ambiguity and grammar size. The method, applicable to any unification grammar with a phrase-structure backbone, is shown to be effective in specializing a broad-coverage LFG for French.

1 Introduction

Expressive grammar formalisms allow grammar developers to capture complex linguistic generalizations concisely and elegantly, thus greatly facilitating grammar development and maintenance. Broad-coverage grammars, however, tend to overgenerate considerably, thus allowing large amounts of spurious ambiguity. If the benefits resulting from more concise grammatical descriptions are to outweigh the costs of spurious ambiguity, the latter must be brought down. We here investigate a corpus-based compilation technique that reduces overgeneration and spurious ambiguity without jeopardizing coverage or burdening the grammar developer.

The current work extends previous work on corpus-based grammar specialization, which applies variants of explanation-based learning (EBL) to grammars of natural languages. The earliest work (Rayner, 1988; Samuelsson and Rayner, 1991) builds a specialized grammar by chunking together grammar rule combinations while parsing training examples. What rules to combine is specified by hand-coded criteria.

Subsequent work (Rayner and Carter, 1996; Samuelsson, 1994) views the problem as that of cutting up each tree in a treebank of correct parse trees into subtrees, after which the rule combinations corresponding to the subtrees determine the rules of the specialized grammar. This approach reports experimental results, using the SRI Core Language Engine, (Alshawi, 1992), in the ATIS domain, of more than a 3-fold speedup at a cost of 5% in grammatical coverage, the latter which is compensated by an increase in parsing accuracy. Later work (Samuelsson, 1994; Sima'an, 1999) attempts to automatically determine appropriate tree-cutting criteria, the former using local measures, the latter using global ones.

The current work reverts to the view of EBL as chunking grammar rules. It extends the latter work by formulating grammar specialization as a global optimization problem over the space of all possible specialized grammars with an objective function based on the coverage, ambiguity and size of the resulting grammar. The method was evaluated on the LFG grammar for French developed within the PARGRAM project (Butt et al., 1999), but it is applicable to any unification grammar with a phrase-structure backbone where the reference treebank contains all possible analyses for each training example, along with an indication of which one is the correct one.

To explore the space of possible grammars, a special treebank representation was developed, called a *folded treebank*, which allows the objective function to be computed very efficiently for each candidate grammar. This representation relies on the fact that all possible parses returned by the original grammar for each training sentence are available and the fact that the grammar specialization never introduces new parses; it only removes existing ones.

The rest of this paper is organized as follows: Section 2 describes the initial candidate grammar and the operators used to generate new candidate grammars from any given one. The function to be maximized is introduced and motivated in Section 3. The *folded treebank* representation is described in Section 4, while Section 5 presents the experimental results.

2 Unfolding and Specialization

The initial grammar is the grammar underlying the subset of correct parses in the training set. This is in itself a specialization of the grammar which was used to parse the treebank, since some rules may not show up in any correct parse in the training set; experimental results for this *first-order* specialization are reported in (Cancedda and Samuelsson, 2000). This grammar is further specialized by inhibiting rule combinations that show up in incorrect parses much more often than in correct parses.

In more detail, we considered *downward unfolding* of grammar rules (see Fig.1).[1] A grammar rule is unfolded downwards on one of the symbols in its right-hand side if it is replaced by a set of rules, each corresponding to the expansion of the chosen symbol by means of another grammar rule. More formally, let $G = \langle \Sigma, \Sigma_N, S, R \rangle$ be a context-free grammar, and let $r, r' \in R, k \in \mathcal{N}^+$ such that $\mathrm{rhs}(r) = \alpha A \beta$, $|\alpha| = k-1$, $\mathrm{lhs}(r') = A$, $\mathrm{rhs}(r') = \gamma$. The *rule adjunction* of r' in the k^{th} position of r is defined as a new rule $\mathrm{RA}(r,k,r') = r''$, such that:

$$\begin{array}{l} \mathrm{lhs}(r'') = \mathrm{lhs}(r) \\ \mathrm{rhs}(r'') = \alpha\gamma\beta \end{array}$$

For unification grammars, we instead require

$$\begin{array}{l} \mathrm{lhs}(r') \sqcup \mathrm{rhs}(r)(k) \\ \mathrm{lhs}(r'') = \theta(\mathrm{lhs}(r)) \\ \mathrm{rhs}(r'') = \theta(\alpha\gamma\beta) \end{array}$$

where $\mathrm{rhs}(r)(k)$ is the kth symbol of $\mathrm{rhs}(r)$, where $X \sqcup Y$ indicates that X and Y unify, and where θ is the most general unifier of $\mathrm{lhs}(r')$ and $\mathrm{rhs}(r)(k)$.

The *downward rule unfolding* of rule r on its k^{th} position is then defined as:

$$\mathrm{DRU}(r,k) = \begin{cases} \{r' | \exists r'' [r' = \mathrm{RA}(r,k,r'')]\} & \text{if} \neq \emptyset \\ \{r\} & \text{otherwise} \end{cases}$$

It is easy to see that if all $r' \in \mathrm{DRU}(r,k)$ are retained then the new grammar has exactly the same coverage as the old one. Once the rule has been unfolded, however, the grammar can be specialized. This involves inhibiting some rule combinations by simply removing the corresponding newly created rules. Any subset $X \subseteq \mathrm{DRU}(r,k)$ is called a *downward specialization* of rule r on the k^{th} element of its rhs.

Given a grammar, all possible (downward) unfoldings of its rules are considered and, for each unfolding, the specialization leading to the best increase in the objective function is determined. The set of all such best specializations defines the set of candidate successor grammars. In the experiments, a simple hill-climbing algorithm was adopted. Other iterative-refinement schemes, such as simulated annealing, could easily be implemented.

3 The Objective Function

Previous research approached the task of determining which rule combinations to allow either by a process of manual trial and error or by statistical measures based on a collection *of positive examples only*: if the original grammar produces more than a single parse of a sentence, only the "correct" parse was stored in the treebank. However, we here also have access to all incorrect parses assigned by the original grammar. This in turn means that we do not need to estimate ambiguity through some correlated statistical indicator, since we can measure it directly simply by checking which parse trees are licensed by every new candidate grammar G. There are many possible ways of combining the counts of correct and incorrect parses in a suitable objective function. For the sake of simplicity we opted for a linear combination. However, simply maximizing correct parses and minimizing incorrect ones would most likely lead to overfitting. In fact, a grammar with one large flat rule for each correct parse in the treebank would achieve a very high score during training, but most likely perform poorly on unseen data. A way to avoid overfitting consists in penalizing large grammars by introducing an appropriate term in the linear combination. The objective

[1] The converse operation, upward unfolding, was not used in the current experiments.

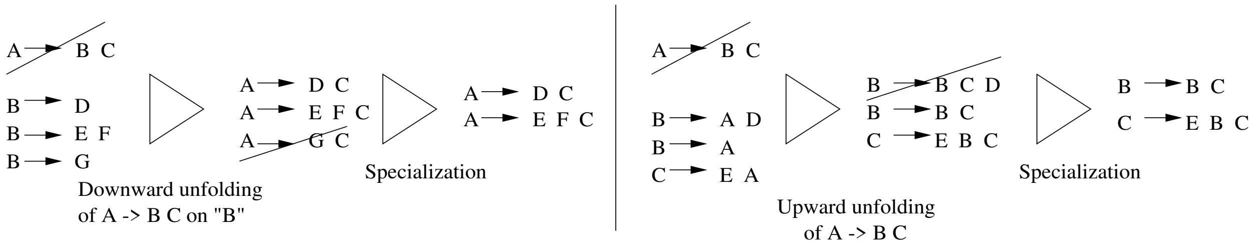


Figure 1: Schematic examples of upward and downward unfolding of rules.

function *Score* was thus formulated as follows:

$$Score_G = \lambda_{corr} Corr_G - \lambda_{inc} Inc_G - \lambda_{size} Size_G$$

where *Corr* and *Inc* are the number of correct and incorrect parses allowed by the grammar, and *Size* is the size of the grammar measured as the total number of symbol occurrences in the right-hand sides of its rules. λ_{corr} and λ_{inc} are weights controlling the pruning aggressiveness: their ratio $\lambda_{corr}/\lambda_{inc}$ intuitively corresponds to the number of incorrect trees a specialization must disallow for each disallowed correct tree, if the specialization is to lead to an improvement over the current grammar. The lower this ratio is, the more aggressive the pruning is. The relative value of λ_{size} with respect to the other λs also controls the depth to which the search is conducted: most specializations result in an increase in grammar size, which tends to be more and more significant as the number and the size of rules grows; a larger λ_{size} thus has the effect of stopping the search earlier. Note that only two of the three weights are independent.

4 Treebank Representation

A *folded treebank* is a representation of a set of parse trees which allows an immediate assessment of the effects of inhibiting specific rule combinations. It is based on the idea of "folding" each tree onto a representation of the grammar itself. Any phrase-structure grammar can be represented as a concatenation/or graph — a directed bipartite multigraph with an or-node for each symbol and a concatenation-node for each rule in the grammar. The present description covers context-free grammars, but the scheme can easily be extended to any unification grammar with a context-free backbone by replacing symbol eqality with unification.

Given a grammar $G = \langle \Sigma, \Sigma_N, S, R \rangle$, we can define a relation η_Σ and a (partial) function η_R:

- $\eta_\Sigma \subseteq \Sigma_N \times R$ s.t. $\langle A, r \rangle \in \eta_\Sigma$ iff $A = \mathrm{lhs}(r)$
- $\eta_R : R \times \mathcal{N}^+ \to \Sigma$ s.t. $\eta_R(r, i) = X$ iff $\mathrm{rhs}(r) = \beta X \gamma$, $|\beta| = i - 1$

Figure 2 shows the correspondence between a simple grammar fragment and its concatenation/or graph.

Each tree can be represented by folding it onto the concatenation/or graph representing the grammar it was derived with, or, in other words, by appropriately annotating the graph itself. If N is the set of nodes of a parse tree obtained using grammar G, the corresponding folded tree is a partial function f

$$f : N \times N \to R \times \mathcal{N}^+$$

such that $f(n, n') = \langle r, k \rangle$ implies that node n was expanded using rule r, and that node n' is its k^{th} daughter in the tree (Fig.3). In the following, we will use the inverse image of $\langle r, k \rangle$ under f, which we denote $\phi(r, k) = f^{-1}(\langle r, k \rangle) = \{\langle n, n' \rangle | f(n, n') = \langle r, k \rangle\} \subset N \times N$. This can in turn be seen as a partial function

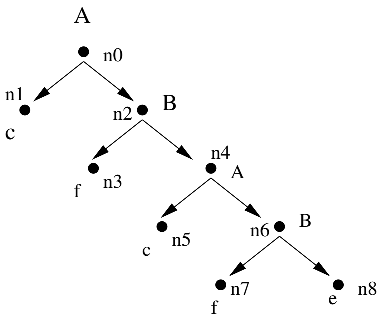
$$\phi : R \times \mathcal{N}^+ \to 2^{N \times N}$$

Disallowing the expansion of the k^{th} element in the right-hand side of rule r by means of rule r' (assuming symbols match, i.e., $\langle \eta_R(r, k), r' \rangle \in \eta_\Sigma$) results in suppressing a tree where:

$$\exists n, n', n'' \in N, k' \in \mathcal{N}^+$$
$$[\langle n, n' \rangle \in \phi(r, k) \wedge \langle n', n'' \rangle \in \phi(r', k')]$$

This check can be performed very efficiently once the tree is represented by the ϕ function, i.e., once it is folded, as all this requires is to compare the entries for $\langle r, k \rangle$ and $\langle r', k' \rangle$[2] with a procedure linear in the size of the entries themselves. If we used a more traditional representation, the same check would require traversing

[2] In fact, it suffices to check the entries for $\langle r', 1 \rangle$.



$$
\begin{aligned}
\phi(r_1,1) &= \{\langle n_0,n_1\rangle,\langle n_4,n_5\rangle\}\\
\phi(r_1,2) &= \{\langle n_0,n_2\rangle,\langle n_4,n_6\rangle\}\\
\phi(r_2,1) &= \emptyset\\
\phi(r_2,2) &= \emptyset\\
\phi(r_3,1) &= \{\langle n_2,n_3\rangle\}\\
\phi(r_3,2) &= \{\langle n_2,n_4\rangle\}\\
\phi(r_4,1) &= \{\langle n_6,n_7\rangle\}\\
\phi(r_4,2) &= \{\langle n_6,n_8\rangle\}
\end{aligned}
$$

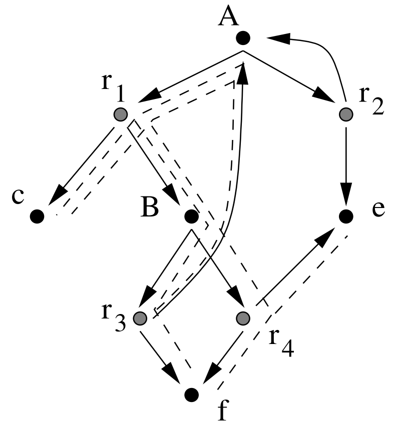


Figure 3: A tree and its folded representation.

the whole tree. The worst-case complexity is still linear in the size of the tree, but in practice, the number of nodes expanded using any given rule is much smaller than the total number of nodes.

Whenever a specialization is performed, all folded trees that are no longer licensed are removed; the concatenation/or graph for the grammar is updated to reflect the introduction and the elimination of rules; and the annotations on the affected edges are appropriately recombined and distributed. If the performed specialization is $X \subseteq \mathrm{DRU}(r,k) \neq \{r\}$, [3] then the concatenation/or graph is updated as follows

$$
\begin{aligned}
\bar{R} &= R \cup X \setminus \{r\}\\
\bar{\eta}_\Sigma &= \eta_\Sigma \cup \{\langle \mathrm{lhs}(r),\bar{r}\rangle | \bar{r} \in X\} \setminus \{\langle \mathrm{lhs}(r), r\rangle\}
\end{aligned}
$$

$$
\bar{\eta}_{\bar{R}}(r'',i) = \\
= \begin{cases}
\eta_R(r'',i), & r'' \notin X\\
\eta_R(r,i), & r'' \in X, i < k\\
\eta_R(r',i-k+1), & r'' = \mathrm{RA}(r,k,r') \in X,\\
 & k \le i \le k+m-1,\\
\eta_R(r,i-m+1), & r'' = \mathrm{RA}(r,k,r') \in X,\\
 & i > k+m-1,
\end{cases}
$$

where $m = \mathrm{arity}(r') = |\mathrm{rhs}(r')|$ is the number of right-hand-side symbols of rule r'. For each tree that is not eliminated as a consequence of the specialization we have

$$
\bar{\phi}(r'',i) = \\
= \begin{cases}
\phi(r'',i),\\
\quad \text{if } r'' \notin X, r'' \neq r'\\
\\
\phi(r'',i) \setminus \{\langle n',n''\rangle | \exists n[\langle n,n'\rangle \in \phi(r,k)]\},\\
\quad \text{if } r'' = r'\\
\\
\{\langle n,n'''\rangle | \exists n',n'',k'[\langle n,n'\rangle \in \phi(r,k)\\
\wedge \langle n',n''\rangle \in \phi(r',k') \wedge \langle n,n'''\rangle \in \phi(r,i)]\}\\
\quad \text{if } r'' = \mathrm{RA}(r,k,r') \in X, i < k\\
\\
\{\langle n,n''\rangle | \exists n'[\langle n,n'\rangle \in \phi(r,k)\wedge\\
\langle n',n''\rangle \in \phi(r',i-k+1)]\}\\
\quad \text{if } r'' = \mathrm{RA}(r,k,r') \in X, k \le i \le k+m-1,\\
\\
\{\langle n,n'''\rangle | \exists n',n'',k'[\langle n,n'\rangle \in \phi(r,k)\wedge\\
\langle n',n''\rangle \in \phi(r',k')\wedge\\
\langle n,n'''\rangle \in \phi(r,i-m+1)]\}\\
\quad \text{if } r'' = \mathrm{RA}(r,k,r') \in X, i > k+m-1,
\end{cases}
$$

where again $m = \mathrm{arity}(r')$. These updates can be implemented efficiently, requiring neither a traversal of the tree nor of the grammar.

5 Experimental Results

We specialized a broad-coverage LFG grammar for French on a corpus of technical documentation using the method described above. The treebank consisted of 960 sentences which were all known to be covered by the original grammar. For each sentence, all the trees returned by

[3] If $X = \mathrm{DRU}(r,k) = \{r\}$, then no update is needed.

$R = \{r_1, r_2, r_3, r_4\}$ s.t.
r_1: A $\rightarrow$ c B
r_2: A $\rightarrow$ e A
r_3: B $\rightarrow$ f A
r_4: B $\rightarrow$ f E

$\eta_\Sigma = \{\langle A, r_1\rangle, \langle A, r_2\rangle, \langle B, r_3\rangle, \langle B, r_4\rangle\}$

$$\eta_R: \begin{array}{l} \langle r_1, 1\rangle \rightarrow c \\ \langle r_1, 2\rangle \rightarrow B \\ \langle r_2, 1\rangle \rightarrow e \\ \langle r_2, 2\rangle \rightarrow A \\ \langle r_3, 1\rangle \rightarrow f \\ \langle r_3, 2\rangle \rightarrow A \\ \langle r_4, 1\rangle \rightarrow f \\ \langle r_4, 2\rangle \rightarrow e \end{array}$$

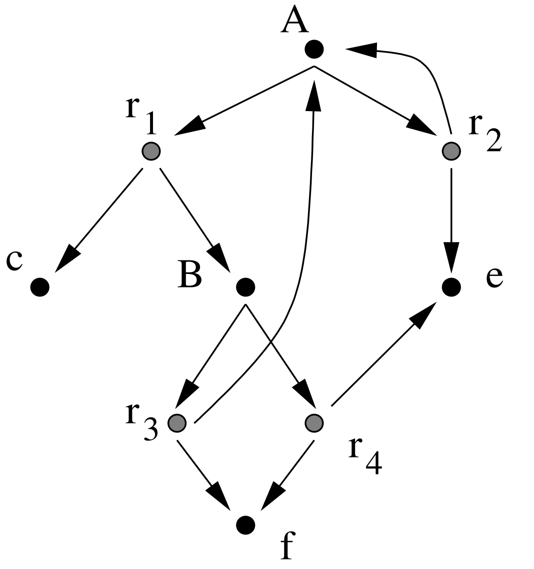


Figure 2: A tiny grammar and the corresponding concatenation/or graph.

the original grammar were available, together with a manually assigned indication of which was the correct one. The environment used, the Xerox Linguistic Environment (Kaplan and Maxwell, 1996) implements a version of optimality theory where parses are assigned "optimality marks" based on a number of criteria, and are ranked according to these marks. The set of parses with the best marks are called the *optimal parses* for a sentence. The correct parse was also an optimal parse for 913 out of 960 sentences. Given this, the specialization was aimed at reducing the number of optimal parses per sentence.

We ran a series of ten-fold cross-validation experiments; the results are summarized in the table in Fig.4. The first line contains values for the original grammar. The second line contains measures for the first-order pruning grammar, i.e., the grammar with all and only those rules actually used in correct parses in the training set, with no combination inhibited. Lines 3 and 4 list results for fully specialized grammars. Results in the third line were obtained with a value for λ_{corr} equal to 15 times the value of λ_{inc} in the objective function: in other words, during training we were willing to lose a correct parse only if at least 15 incorrect parses were canceled as well. Results in the fourth line were obtained when this ratio was reduced to 10. The average number of parses per sentence is reported in the first column, whereas the second lists the average number of optimal parses. *Coverage* was measured as the fraction of sentences which still receive the correct parse with the specialized grammar. To assess the trade off between coverage and ambiguity reduction, we computed the F-score[4] considering only optimal parses when computing precision. This measure should not be confused with the F-score on labelled bracketing reported for many stochastic parsers; here precision and recall concern perfect matching of whole trees. Recall is the same as coverage: the ratio between the number of correct parses produced by the specialized grammar and the total number of correct parses (equalling the total number of sentences in the test set). Precision is the ratio between the number of correct parses produced by the specialized grammar and the total number of parses produced by the same grammar. The fourth column lists values for the F-score when equal weight is given to precision and recall. Intuitively, however, in many cases missing the correct parse is more of a problem than returning spurious parses, so we also computed the F-score with a much larger emphasis on recall, i.e., with $\alpha = 0.1$. The corresponding values are listed in the last column.

The average number of parses per sentence, both optimal and non-optimal, decreases significantly as more and more aggressive specialization is carried out, and consequently, more coverage is lost. The most aggressive form of spe-

[4]The F-score is the harmonic mean of recall and precision, where precision is weighted α and recall $1 - \alpha$.

	Avg.p/s	Avg. o.p./s.	Coverage (%)	F($\alpha = 0.5$)	F($\alpha = 0.1$)
orig.	1941	4.69	100	35.15	73.05
f.o.pruning	184	3.38	89	40.64	71.89
λ_{corr}=15λ_{inc}	82	2.23	86	53.25	76.58
λ_{corr}=10λ_{inc}	63	2.03	82.5	54.46	74.80

Figure 4: Results of the specialization experiments.

cialization gives the highest F-score for $\alpha = 0.5$, whereas somewhat more conservative parameter settings lead to a better F-score when recall is valued more. A speedup of a factor 4 is achieved already by first-order pruning and remains approximately the same after further specialization.

6 Conclusions

Broad-coverage grammars tend to be highly ambiguous, which may constitute a serious problem when using them for natural-language processing. Corpus-independent compilation techniques, although useful for increasing efficiency, do little in terms of reducing ambiguity.

In this paper we proposed a corpus-based technique for specializing a grammar on a domain for which a treebank exists containing all trees returned for each sentence. This technique, which builds extensively on previous work on explanation-based learning for NLP, consists in casting the problem as an optimization problem in the space of all possible specializations of the original grammar. As initial candidate grammar, the *first-order pruning* of the original grammar is considered. Candidate successor grammars are obtained through the *downward rule unfolding and specialization* operator, that has the desirable property of never causing previously unseen parses to become available for sentences in the training set. Candidate grammars are then assessed according to an objecting function combining grammar ambiguity and coverage, adapted to avoid overfitting. In order to ensure efficient computability of the objective function, the treebank is previously *folded* onto the grammar itself. Experimental results using a broad-coverage lexical-functional grammar of French show that the technique allows effectively trading coverage for ambiguity reduction. Moreover, the parameters of the objective function can be used to control the trade off.

Acknowledgements

We would like to thank the members of the MLTT group at the Xerox Research Centre Europe in Grenoble, France, and the three anonymous reviewers for valuable discussions and comments. This research was funded by the European TMR network Learning Computational Grammars.

References

Hiyan Alshawi, editor. 1992. *The Core Language Engine*. MIT Press.

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

Nicola Cancedda and Christer Samuelsson. 2000. Experiments with corpus-based lfg specialization. In *Proceedings of the NAACL-ANLP 2000 Conference*, Seattle, WA.

Ronald Kaplan and John T. Maxwell. 1996. LFG grammar writer's workbench. Technical report, Xerox PARC. Available on-line as ftp://ftp.parc.xerox.com/pub/lfg/lfgmanual.ps.

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, Australia.

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.