Learning Context-Free Grammars with a Simplicity Bias

Pat Langley and Sean Stromsten

DaimlerChrysler Research and Technology Center
1510 Page Mill Road, Palo Alto, CA 94304 USA
langley@isle.org sean@psych.stanford.edu

Abstract. We examine the role of simplicity in directing the induction of context-free grammars from sample sentences. We present a rational reconstruction of Wolff's SNPR – the GRIDS system – which incorporates a bias toward grammars that minimize description length. The algorithm alternates between merging existing nonterminal symbols and creating new symbols, using a beam search to move from complex to simpler grammars. Experiments suggest that this approach can induce accurate grammars and that it scales reasonably to more difficult domains.

1 Introduction

In this paper we focus on the task of inducing context-free grammars from training sentences. Much recent work on this topic has dealt with learning finite-state structures, but there is considerable evidence that human language involves more powerful grammatical representations. In context-free grammar induction, the learner must find not only a set of grammatical rewrite rules but also the nonterminal symbols used in those rules. For example, in addition to deciding that an English sentence can be composed of a noun phrase and a verb phrase, it must also create definitions for these intermediate concepts.

A central challenge of grammar induction involves the generative nature of language. The learner must somehow create a knowledge structure that produces an infinite number of sentences from a finite set of training cases. Typically, this requires recursive or iterative structures, which can cause overgeneralizations. Effective induction of context-free grammars requires strong constraints on search through the space of candidates. One that often recurs in the literature is a bias toward *simple* grammars.

This bias helps avoid one sort of trivial grammar that has a separate rule for each training sentence and that does not generalize at all to new sentences. However, a naive notion of simplicity leads to another sort of trivial grammar that admits any string of words and overgeneralizes drastically. A more useful variation on this idea views the grammar as a code and seeks to compress the sample sentences, minimizing the summed description length of the grammar and it derivations of training sentences. By 'simplicity' then, we mean that of the grammar *and* the derivations of the training sentences under the grammar.

In the following pages, we examine the extent to which this notion of simplicity can successfully direct the grammar-induction process. We explore this idea in the context of GRIDS, a rational reconstruction of Wolff's (1982) SNPR system. We first describe GRIDS' representation, performance component, learning algorithm, and evaluation function, then present experimental studies designed to evaluate the system's learning behavior. In closing, we discuss related work on grammar induction and outline directions for future research.

2 Grammar Induction Driven by Simplicity

As noted above, GRIDS represents grammatical knowledge as context-free rewrite rules, using a top-level symbol (S), a set of nonterminals, and a set of terminal symbols corresponding to words. Each rewrite rule includes one nonterminal symbol on the left-hand side and one or more symbols on the right, indicating that one can replace the former with the latter in recognizing or generating a sentence. Following VanLehn and Ball (1987), we restrict GRIDS' grammars so that no rule has an empty right-hand side, the only rules of the form $X \rightarrow Y$ are those in which Y is a terminal symbol, and every nonterminal appears in the derivation of some sentence. This restriction does not limit representational power, as one can transform any context-free grammar into this form.

The performance component of GRIDS is a top-down, depth-first parser that repeatedly substitutes the first nonterminal X in its string with the right-hand side of a rewrite rule having X on the left. We do not view this performance algorithm as part of our theoretical framework, and its implementation is far from efficient. However, it does let GRIDS determine whether a given grammar parses a given string of words, and thus whether that grammar is overly general, overly specific, or accurate for the language at hand.

2.1 Learning Operators and Search Organization

GRIDS' approach to grammar induction, as in Wolff's earlier system, relies on two learning operators. The first creates a nonterminal symbol X and an associated rewrite rule that decomposes X into its constituents. In grammars for natural languages, such symbols and their rules correspond to specific phrases and clauses. The introduction of phrasal terms should be useful when certain combinations of symbols tend to occur together in sentences. Table 1 (a) gives a simple example of this operator's effect.

The second operator involves merging two nonterminal symbols into a single symbol. The resulting sets of rules with the same left-hand side correspond, in grammars for natural languages, to word classes (e.g., nouns and verbs) and phrasal classes (e.g., noun phrases). Their introduction should be useful when certain symbols tend to occur in similar contexts within the language. We should note one important side effect of the merge operator. Given the rewrite rule $X \rightarrow Y \ldots Z$, merging X and Z produces the rule $X \rightarrow Y \ldots X$, which involves a recursive call. Table 1 (b) illustrates this outcome in a simple grammar, though merging can also produce indirect recursions.

Table 1. The learning operators used in GRIDS include (a) creating a new symbol and rewrite rule based on two existing symbols, and (b) merging two existing symbols, which can lead to redundant (and thus removed) rules, as well as to recursive grammars.

(a) Creating symbol *AP1*	(b) Merging *AP1* and *AP2*
	$NP \rightarrow ART\ AP1$
$NP \rightarrow ART\ ADJ\ NOUN$	$NP \rightarrow ART\ AP2$
$NP \rightarrow ART\ ADJ\ ADJ\ NOUN$	$AP1 \rightarrow ADJ\ NOUN$
	$AP2 \rightarrow ADJ\ AP1$
$\Downarrow$	$\Downarrow$
$NP \rightarrow ART\ AP1$	$NP \rightarrow ART\ AP1$
$NP \rightarrow ART\ ADJ\ AP1$	$AP1 \rightarrow ADJ\ NOUN$
$AP1 \rightarrow ADJ\ NOUN$	$AP1 \rightarrow ADJ\ AP1$

GRIDS starts by transforming the sample sentences into an initial 'flat' grammar that contains only rules of the form $S \rightarrow X \ldots Y$ (one for each observed sentence) and $X \rightarrow W$ (for each word W). Thus, each S rewrite rule and its associated word rules correspond to a single training instance, so that the initial grammar covers all (and only) the training sentences. Symbol creation does not change the coverage of a grammar, and symbol merging can never decrease the coverage. Thus, as GRIDS proceeds, it only considers grammars with the same or greater generality than the current hypothesis. The current version uses beam search, with a beam size of three, to control its steps through the resulting space.

The learning process in GRIDS alternates between two modes, each relying on a different operator. First the system considers all ways of merging pairs of nonterminal symbols in each current grammar, producing a set of successor grammars. When this action produces a new grammar that contains identical rewrite rules, all but one of the redundant rules are removed. Next the system uses an evaluation function, which we will discuss shortly, to select the best b grammars from the successors, breaking ties among candidates at random. If the evaluation metric indicates that at least one of the successors constitutes an improvement over the current best grammar, the new grammars become the current best set and the system continues in this mode.

However, if none of the new grammars scores better than the current best candidate, GRIDS switches from 'merge' mode into 'create' mode. Here the algorithm considers all ways of creating new terms, and their associated rules, from pairs of nonterminal symbols that occur in sequence within the grammars. GRIDS then substitutes the new term for all occurrences of the sequence in the prospective grammar. Again, it selects the best alternatives and, if some score better than the current best grammar, the best b candidates become the current set and the program continues in 'create' mode; if not, GRIDS changes modes and again considers merging. The algorithm continues in this manner, alternating between modes until neither leads to improvement, in which case it halts.

2.2 Directing Search with Description Length

We have seen that Grids carries out a beam search through the space of context-free grammars, starting with a specific grammar based on training sentences and moving toward more general candidates. However, the space of grammars is large and the system needs some evaluation metric to direct search toward promising candidates. To this end, it applies the principle of minimum description length, measuring the simplicity of each candidate grammar G in terms of the description length for G plus that for the training sentences, encoded as derivations in G.

In this formulation, a hypothetical 'receiver' must know how to interpret the string of bits that encode the model and data. Grids encodes the rules of the grammar as strings of symbols separated by tokens of a STOP symbol. Each nonterminal token requires $\log(N+1)$ bits, where N is the number of nonterminal types, and the terminals each require $\log P_i$, where P_i is the number of words with the same part of speech. The derivations are strings of rewrite rules. The left-hand side of each is known, at each point, given the previous rules, so it need only distinguish among the R right-hand sides, which requires $\log R$ bits.

Intuitively, this measure should shun large grammars with overly specific rules, despite their short derivations, because other grammars will have smaller descriptions and do nearly as well on the derivations. The measure avoids very small, overly general grammars because they can describe too many unobserved strings, so that bits must be wasted in encoding the derivations of actual sentences just to distinguish them from these nonsentences. In general, a good code assigns long encodings to rare strings and short encodings to common ones. In our case, a good grammar may also forfeit entirely the ability to encode some (unobserved) strings in exchange for the ability to encode others (observed training sentences) more efficiently.

3 Experimental Studies of Grids' Behavior

The central hypothesis in our work was that simplicity, as measured by description length, is a powerful bias for constraining the process of grammar induction. To evaluate this hypothesis, we carried out a number of experiments, which we report after considering their design and the domains used therein.

3.1 Grammatical Domains and Experimental Design

We decided to use artificial grammars in our experiments, since they let us both control characteristics of the domain and measure the correctness of the induced knowledge structures. In particular, we designed the two subsets of English grammar shown in Table 2. The first (a) includes declarative sentences with arbitrarily long strings of adjectives and both transitive and intransitive verbs, but no relative clauses, prepositional phrases, adverbs, or inflections. The second grammar (b) contains declarative sentences with arbitrarily embedded relative clauses, but has no adjectives, adverbs, prepositional phrases, or inflections.

These two grammars are unsophisticated compared to those required for natural languages, but they involve recursion and generate an infinite class of

Table 2. Two grammars used to generate training and test sentences for experiments with the GRIDS algorithm. The first grammar (a) includes arbitrary strings of adjectives, whereas the second (b) supports arbitrarily embedded relative clauses.

(a)	(b)
$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$VP \rightarrow VERBI$	$VP \rightarrow V\ NP$
$VP \rightarrow VERBT\ NP$	$NP \rightarrow ART\ NOUN$
$NP \rightarrow the\ NOUN$	$NP \rightarrow ART\ NOUN\ RC$
$NP \rightarrow the\ AP\ NOUN$	$RC \rightarrow REL\ VP$
$AP \rightarrow ADJ$	$VERB \rightarrow saw$
$AP \rightarrow ADJ\ AP$	$VERB \rightarrow heard$
$VERBI \rightarrow ate$	$NOUN \rightarrow cat$
$VERBI \rightarrow slept$	$NOUN \rightarrow dog$
$VERBT \rightarrow saw$	$NOUN \rightarrow mouse$
$VERBT \rightarrow heard$	$ART \rightarrow a$
$NOUN \rightarrow cat$	$ART \rightarrow the$
$NOUN \rightarrow dog$	$REL \rightarrow that$
$ADJ \rightarrow big$	
$ADJ \rightarrow old$	

sentences, thus providing tests of GRIDS' ability to generalize correctly. However, one can also state both grammars as finite-state machines, which involve iteration but not recursion, so we also examined two languages that required center embedding. One involved sentences with a string of a's followed by an equal number of b's, whereas the other involved strings of balanced parentheses. Both languages have been used as testbeds in earlier efforts on grammar induction.

For the two English subsets, we created 20 training sets with enough strings in each for the program to reach asymptotic performance, with instances for the adjective phrase domain having a length of ten or less and those for the relative clause grammar length 15 or less. For the parenthesis-balancing and $(ab)^n$ languages, we used the same strategy to generate training sets with maximum lengths of ten and 20, respectively.

The measurement paradigms typically used for supervised learning tasks do not apply directly to grammatical domains. A grammar-induction system can infer the right word classes with relative ease, making the real test whether it forms recursive rules that let it correctly generalize to sentences longer than those in the training sample. Thus, in generating our test sets, we used maximum lengths of 15 and 20 for the adjective phrase and relative clause domains, respectively. For the parenthesis language, we generated all 65 legal strings of length 12 or less as positive test cases, and enumerated all 15 sentences of length 30 or less for the $(ab)^n$ language.

Another issue concerns the need to distinguish errors of omission (failures to parse sentences in the target language), which indicate an undergeneral grammar, from errors of commission (failures to generate only sentences in the target

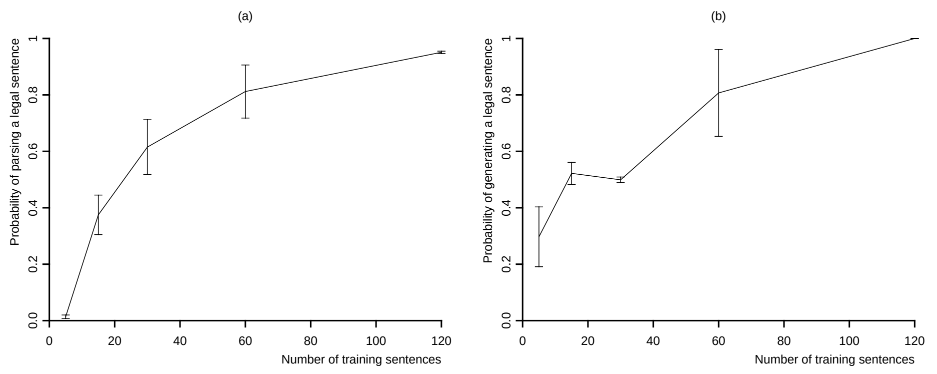

Fig. 1. Average learning curves for the adjective phrase grammar from Table 2, with (a) measuring the probability of parsing a legal test sentence and (b) the probability of generating a legal sentence.

language), which indicate an overgeneral one. To estimate these terms, we used the target grammar T and each learned grammar L to generate sentence samples, and then determined their overlap. We estimated errors of omission from the fraction of sentences generated by T that were parsed by L, and errors of commission from the fraction of sentences generated by L that were parsed by T. On the average, an undergeneral grammar will produce a low score on the first measure, whereas an overgeneral one will produce a low score on the second.

3.2 Experimental Results

We intended our initial study to show that GRIDS could actually induce accurate grammars for all four domains. However, we were also interested in the *rate* of learning, so we explicitly varied the number of training sentences available to the system, at each level measuring the two accuracies of the learned grammar, averaged over 20 different training sets.

Figure 1 presents the learning curves for the adjective phrase grammar from Table 2, with (a) showing results on the first measure, the probability of parsing a legal test sentence, and (b) showing those for the second, the probability of generating a sentence parsed by the target grammar. The curves show both the average accuracy and 95% confidence intervals as a function of different numbers of training sentences. After 120 training cases, the learned grammars cover 95% of the positive test set, and all generated strings are legal.

Somewhat different results occurred with the relative clause language, as shown in Figure 2. As before, the probability of parsing the 500 legal test sentences increases with experience, though with many fewer examples, reaching 100% after only 15 training items. However, in this case GRIDS' probability of generating a legal sentence starts at 100%, falls to below 60% by the fourth case, then rebounds to perfect accuracy after processing 11 training sentences.

Experimental results for the parenthesis balancing language (not shown here) are analogous to those for adjective phrases, and the learning curves for the $(ab)^n$ language follow a very similar pattern, though the slopes are different. Clearly,

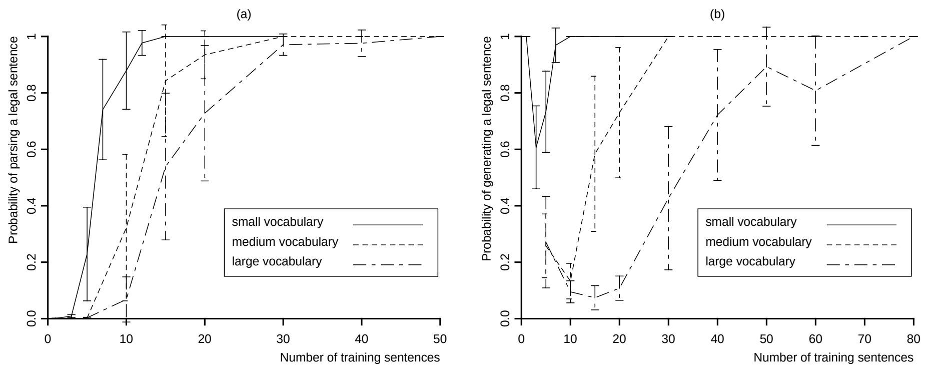

Fig. 2. Learning curves for the relative clause grammar from Table 2, and for analogous grammars that involve larger word classes, with (a) measuring the probability of parsing a legal test sentence and (b) the probability of generating a legal sentence.

one goal of future research should be to explain the underlying causes of these distinctive patterns, as well as the widely differing rates of learning.

Although our test grammars are simple compared to those encountered in natural languages, their complexity is comparable to others reported in the literature. Nevertheless, it would be good to understand the ability of the methods embodied in GRIDS to scale to more difficult induction tasks. To this end, we carried out an additional experiment in which we increased the size of word classes. In particular, we extended the relative clause grammar from Table 2, which included two verbs, three nouns, and one relative pronoun, by doubling and tripling the number of words in each of these categories.

Figure 2 compares the learning curves for these domains, using the two performance measures described earlier. Although increasing the size of the word classes slows down the learning process, the reduction in learning rate seems quite reasonable. Specifically, the number of training sentences required to reach perfect accuracy appears to be no more than linear in the size of the word classes. Also, this factor seems to affect both performance measures equally.

4 Discussion

Our approach to learning shares some of its central features with earlier work on grammar induction. We have already noted GRIDS' debt to Wolff's (1982) SNPR system, which also carried out heuristic search using operators for creating and merging symbols, and which used an evaluation function that traded off a grammar's simplicity and its ability to 'compress' the training data. Cook, Rosenfeld, and Aronson's (1976) early work grammar induction also used an operator for creating nonterminal symbols, combined with hill-climbing search directed by a evaluation function similar in spirit to Wolff's.

Stolcke (1994) has carried out more recent research along similar lines, independently developing a grammar-induction algorithm that shares GRIDS' starting representation and its operations for symbol merging and creation. His system's evaluation metric also trades off a grammar's simplicity with its ability to account for observed sentences, but it learns probabilistic context-free grammars and processes training sentences incrementally. Grünwald (1996) has also developed an algorithm that uses a description-length score to direct search for 'partial' grammars, again invoking operators for term creation and merging.

The bias toward simplicity has arisen in other grammar-induction research, some quite different in overall control structure. Examples include enumerative algorithms that consider simpler grammars before more complex ones, as well as methods that start with a randomly generated grammar and invoke simplicity measures to direct hill-climbing search. Not all work on grammar induction relies on the simplicity bias, but the idea plays a recurring role in the literature. The literature also contains many formal claims about language 'learnability' under various conditions. Neither positive or negative results of this sort are relevant to our work, since we care not about guarantees but about practical methods.

Undoubtedly, we can improve the GRIDS algorithm along many fronts. For instance, it assumes that each word belongs to only one category, whereas in natural languages the same word can serve as several parts of speech. Also, an impediment to larger-scale studies is that the run time of the initial 'merge' operations increases with the square of the number of words. One strategy for dealing with the many possible merges involves trying only pairs with high scores on some heuristic measure, perhaps computed over co-occurrence statistics. Another response would be to develop an incremental version of GRIDS that processes only a few training sentences at a time and expands the grammar as necessary. We plan to explore both approaches to improving computational efficiency.

We cannot yet draw final conclusions about the role played by GRIDS' simplicity bias, as there exist other formulations of this idea not covered by our experimental evaluation. Nor can we yet tell whether other operators, or other organizations of the search process, will yield better or worse results. Clearly, more work remains to be done, but the results to date suggest the notion of simplicity has an important role to play in the acquisition of grammatical knowledge.

References

Cook, C. M., Rosenfeld, A., & Aronson, A. (1976). Grammatical inference by hill climbing. *Informational Sciences*, *10*, 59–80.

Grünwald, P. (1996). A minimum description length approach to grammar inference. In S. Wermter, E. Riloff, & G. Scheler (Eds.) *Connectionist, statistical and symbolic approaches to learning for natural language processing*. Lecture Notes in Computer Science, 1040. Berlin: Springer-Verlag.

Stolcke, A. (1994). *Bayesian learning of probabilistic language models*. Doctoral dissertation, Division of Computer Science, University of California, Berkeley.

VanLehn, K., & Ball, W. (1987). A version space approach to learning context-free grammars. *Machine Learning*, *2*, 39–74.

Wolff, J. G. (1982). Language acquisition, data compression and generalization. *Language & Communication*, *2*, 57–89.