In: Proc. 13th Ann. Conf. Cognitive Science Soc., Chicago, Ill., August 1991, pp. 908-912

# Syntactic Category Formation with Vector Space Grammars

**Andreas Stolcke**

Computer Science Division
University of California
Berkeley, CA 94720

International Computer Science Institute
1947 Center St., Berkeley, CA 94704
stolcke@icsi.berkeley.edu

**Abstract**

A method for deriving phrase structure categories from structured samples of a context-free language is presented. The learning algorithm is based on adaptation and competition, as well as error backpropagation in a continuous vector space. These connectionist-style techniques become applicable to grammars as the traditional grammar formalism is generalized to use vectors instead of symbols as category labels.

More generally, it is argued that the conversion of symbolic formalisms to continuous representations is a promising way of combining the connectionist learning techniques with the structures and theoretical insights embodied in classical models.

## Introduction

Connectionism, and especially Parallel Distributed Processing (PDP) has developed an array of models of learning systems (backpropagation, Boltzmann machines, competitive learning (Rumelhart, McClelland, & The PDP Research Group 1986)). These models typically operate on representations at a rather low and unstructured level (unit activations, bit vectors, microfeatures) relative to the structures used in traditional linguistic descriptions (trees and graphs, case frames, grammar rules, stacks). This is a necessary feature, the algorithms used are powerful and general precisely because they operate on simple and homogeneous representations. Simplicity and uniformity of the representations also allows for mathematical analysis, leading to theoretically well-founded methods such as gradient decent or simulated annealing.

A second prerequisite for these connectionist learning algorithms is that representations be *continuous*. Continuity of the representation space (ofter paired with requirement that some performance measure be *differentiable* with respect to the representations), ensures that *adaptive learning* can take place, i.e., gradual adjustment towards a specified goal. Again, continuity and differentiability are typically not found in traditional linguistic constructs, which tend to be inherently discrete (an exception are Fuzzy Languages (Zadeh 1972)).

It seems desirable, then, to investigate ways to combine connectionist (usually vector-based) representations with structures and concepts developed in traditional theories, especially in cases where those theories have a strong empirical or intuitive appeal. The goal would be to use one or several of the learning techniques mentioned above to learn or develop representations that are meaningful in the framework of the theory at hand.

The remainder of the paper gives an example of this general approach, applied to the theory of context-free grammars and the problem of learning category labels for the phrase types in a language. First we introduce the the hybrid symbolic/connectionist formalism on which the algorithm is based, and then discuss learning by way of a detailed example.

## Vector Space Grammars

The formalism presented here has been dubbed *Vector Space Grammar* (VSG) because it represents a generatization of traditional context-free grammars (CFGs) in which the nonterminals are represented by points in a continuous vector space rather than by symbols. Like in CFG, nonterminal rules (in Chomsky Normal From, CNF) are of the form

$$\mathbf{x} \rightarrow \mathbf{y}\ \mathbf{z} \tag{1}$$

and lexical (terminal) rules of the form

$$\mathbf{x} \rightarrow \mathrm{a} \tag{2}$$

But whereas in CFG categories ($\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$) are symbols in a space with a binary metric (equality/nonequality), VSG uses *vectors* as nonterminals. This gives a continuous metric on the category space, thus fulfilling one of the prerequisites for an adaptive learning mechanism. Terminals (words) in VSG are still unanalyzed atomic entities, and strings of terminals form the domain over which a language is defined.

A standard non-terminal rule maps two specific symbolic categories into a third symbolic category (the left-hand side of the rule). Similarly, a VSG rule maps two vectors onto a third. From a bottom-up parsing point of view, a traditional CFG rule is applicable if and only if its two right-hand side categories match exactly two other categories (roots of partial parses). In VSG, rule applicability becomes a graded notion, and every rule

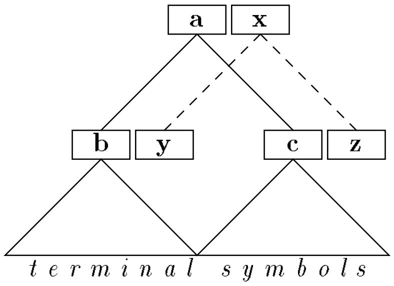


Figure 1: Vectors involved in VSG rule application. The new root vector $\mathbf{a}$ is a function of the subtree root vectors $\mathbf{b}$ and $\mathbf{c}$ and the vectors in the rule $\mathbf{x} \rightarrow \mathbf{y}\ \mathbf{z}$, e.g., $\mathbf{a} = (\mathbf{b} \cdot \mathbf{y})(\mathbf{c} \cdot \mathbf{z})\mathbf{x}$.

will be applicable to every two categories to some extent. However, the formalism is designed such that well-matching rules give a 'high' output, and poorly matching rules result in a vector close to the zero vector. This is accomplished by the following 'activation function' for VSG rules. Let $\mathbf{x} \rightarrow \mathbf{y}\ \mathbf{z}$ be the rule applied to two categories $\mathbf{b}$ and $\mathbf{c}$ (we use bold letters to denote vector quantities). Then the category resulting from the rule application is defined as

$$\mathbf{a} = (\mathbf{b} \cdot \mathbf{y})(\mathbf{c} \cdot \mathbf{z})\mathbf{x} \qquad (3)$$

where $\cdot$ denotes the inner product of the vector space. The two inner products on the right express the match between the categories specified by the rule and the categories assigned to the substrings. Match values range between $-1$ and $+1$ if the category vectors are normalized. Since the right-hand side terms in a context-free rule work conjunctively (all have to match), the match values are multiplied. Choosing the inner product as the measure of matching partly determines the structure of the category space: categories will behave differently to the extent that they are orthogonal. The elements involved in rule application are depicted schematically in Fig. 1.

Traditional CFGs can be mapped into the vector representation such that equation 3 works precisely as traditional grammar rules do (using an all-or-nothing match between categories). This isomorphism maps each nonterminal in the CFG to a dimension in the VSG category space, and demonstrates that VSGs are indeed a formal generalization of symbolic phrase-structure grammars.

*Acceptance of strings* by a grammar can be defined analogously to traditional grammars, although acceptance becomes a non-discrete function (as in fuzzy languages). Since these definitions are not directly relevant to the learning procedure we will omit them here and turn immediately to the learning algorithm (see (Stolcke 1991) for details).

## Learning with Vector Space Grammars

The problem of learning to parse strings of a language can be broken down into two subproblems: finding the structure of the parse tree (i.e., the phrase bracketing), and assigning category labels to the nodes in the tree (the phrases). Current work with VSGs addresses mainly the second of these problems, for several reasons.

There are indications that the two problems might in fact be handled separately by natural language speakers, and that there are cues independent from category assignment that allow speakers to derive the phrase bracketing information. It has been shown that there are very effective statistical methods to find phrase boundaries (without phrase type classification) in text (Magerman & Marcus 1990). Secondly, psycholinguistic data indicates that humans can learn language structures successfully only when they can draw from a rich set of universal intra- and extra-sentential cues to induce phrase structure independently (Morgan, Meier, & Newport 1987; Morgan, Meier, & Newport 1989). Morgan (1986) has also argue for the prior availability of phrase structure information on learnability grounds.

Another source of independent phrase structure information comes from strong correlations between syntactic and semantic structure, i.e., the fact that syntax usually exhibits a structure parallel to one of the conceptual dependencies it expresses. This fundamental 'iconic relationship' between syntax and concepts is understood by some linguists as the very essence of language (Langacker 1985). A learner could capitalize on this principle if one assumes that certain general cognitive capacities are available prior to syntax learning.

For the purpose of this paper, then, we will assume that a learning system has access to phrase-bracketing information from independent sources. We will discuss how the category system and the rules for a language can be learned within the formal framework provided by VSG, given positive (and possibly negative) instances of the language along with their phrase structure boundaries. The kinds of structures available to the learning algorithm are familiar from Levy and Joshi's (1978) *skeletal structural descriptions*, and have been shown to be sufficient for syntax learning (Fass 1983). These learnability results, however, use automata induction techniques with very complex data structures (equivalence classes of trees structures), and are therefore not directly comparable to the methods employed here. VSG learning uses only very simple vector data structures and works on-line, i.e., no history of past samples has to be stored. All the primitive computations performed could be implemented using standard connectionist hardware, except for the global control and structure allocation mechanism.

Two global parameters of the system are the dimension of the category space and the number of rules to be used. These parameters should be set 'large enough' for a given language, and have an effect similar to the number of hidden units in a backpropagation network.

With too little resources, the system will not converge on a solution, and with too many degrees of freedom the solution might be redundant and not express certain generalizations about the input.

At the outset of learning, then, a fixed number of nonterminal rule 'templates' of the form (1) (with a given vector space dimension) are allocated. Additionally, for each terminal symbol, a rule of the form (2) is created. All category vectors, in all rules, are set to random unit-length vectors.

Given a sample string from the language and a parse tree skeleton, we construct a labeled parse tree from the current set of rules. To assign a category vector to a node, the rule whose right-hand side represents the best match for the child node categories is selected and equation (3) is used to compute the output category for that node. 'Best match' is defined according to the same inner product metric as used in equation (3), i.e., using the value $(\mathbf{b} \cdot \mathbf{y})(\mathbf{c} \cdot \mathbf{z})$. Only the rules selected at some node will later participate in the learning process. Since only the currently best rules get selected the whole process strongly resembles the method of *competitive learning* (Rumelhart & Zipser 1985).

By working from the terminal nodes to the root we arrive at a category label for the entire string. If the training sample is a positive instance of the language we know what the target category for the parse should be: the sentence category 'S' that every grammar has to provide. Without loss of generality we can fix S throughout training to be a particular vector, e.g., the unit vector $(1, 0, \ldots, 0)$.

The second idea adapted from connectionist learning methods is that of *error backpropagation* (Rumelhart, Hinton, & Williams 1986). At the root node we can immediately compute an error term for the discrepancy between the desired output and the actual output. For positive examples this is just the difference between S and the root category, for negative examples we compute an error term which tends to make the output category and S orthogonal. A recursive procedure (based on the chain rule) can then compute the derivative of that error with respect to every category vector occurring in some rule (left of right-hand side) applied somewhere in the tree. The computation of error derivatives is straightforward because of the simple linear operations used in equation 3 but omitted here for lack of space (see (Stolcke 1991)).

Derivatives for each category vector are then added up and multiplied by some constant (the 'learning rate') to give the adjustment to be applied to that category. All rules are updated accordingly, all categories are rescaled to unit-length, and the next training example is processed. The algorithm cycles through the training set until the error becomes negligible or no further improvement is observed over a long period of time.

It is important to realize that the backpropation step can not assign errors that are due to choosing the 'wrong' rule at some point, because rule selection is a discrete step that allows no differentiation. Unfortunate rule selection has to be overcome by changing the rules themselves, and competitive rule selection is a heuristic to minimize rule selection errors.

```
+ ((a circle) (touches (a square)))
+ ((a square) (touches (a circle)))
+ ((a circle) (is (below (a square))))
+ ((a square) (is (below (a circle))))
+ ((a circle) (is (above (a square))))
+ ((a square) (is (above (a circle))))
- (a square)
- (a circle)
- (above (a circle))
- (below (a square))
- (touches (a circle))
- (touches (a square))
- (is (above (a square)))
- (is (above (a square)))
- (is (below (a circle)))
- ((a circle) (below (a square)))
- ((a square) (above (a circle)))
- ((a circle) (is (touches (a square))))
- ((is circle) (touches (a square)))
- ((a circle) (a (a square)))
- ((a square) (is (below (is circle))))
- ((a square) (touches (below (a circle))))
- ((a circle) (is (a square)))
- ((a square) (a (above (a circle))))
```

Figure 2: Training set used for the VSG learning experiment. The data is drawn from a fragment of English generated by the grammar given in the text. Positive training instances are labeled with +, negatives ones with -.

## A Sample Grammar

We have run simulations of the algorithm described above to verify that is can indeed converge onto working VSG grammars for a variety of small artificial and 'natural' languages. As indicated in the introduction, one of our main goals was to not only attest learning success (as defined by the error function) but to try to understand how the category vectors formed collaborate to produce a meaningful system of rules.

As an example consider a fragment of English consisting of transitive sentences ('A circle touches a square') and copula sentences ('A circle is below a square') involving the nouns *circle, square*, the verbs *is, touches*, the prepositions *above, below* and the determiner *a* (this fragment is borrowed from the $L_0$ project domain (Feldman et al. 1990), a sample grammar for it is given below).

The algorithm was run over a set of 6 positive and 18 negative samples, listed in Fig. 2. the number of rules was set to 5 and the category dimension to 15. At a constant learning rate of 0.5 the error was typically negligible after 50 passes over the training set.

As a method for analyzing the resulting VSG we used *cluster analysis*, which groups vectors according to a distance metric in a hierarchical fashion. Fig. 3 shows the result of clustering all vectors occurring in rules as well as the fixed S vector.

The graph shows that the vectors fall into nine major clusters of left-hand side and right-hand side rule vectors. We can reconstruct a symbolic rule system from

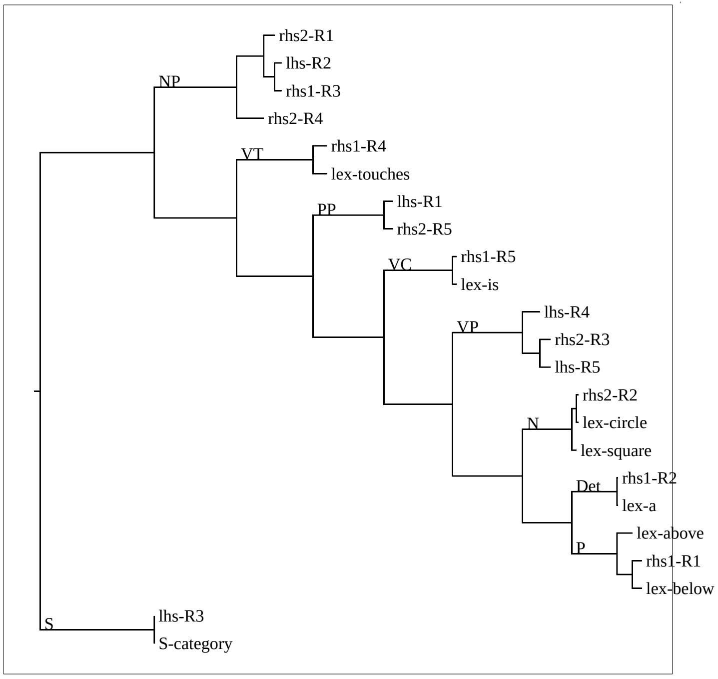


Figure 3: Clusters of category vectors derived from sample language. Left-hand side and right-hand side vectors in non-terminal rules are labeled by one of 'lhs', 'rhs1' and 'rhs2', and the rule number (R1–5). Left-hand side vectors of terminal rules are labeled as 'lex' and the terminal they generate. 'cat-S' is the fixed S category vector. Branches leading to clusters that can be identified with non-terminals from the context-free grammar given in the text are labeled with the corresponding symbol.

the diagram by identifying these clusters with (initially arbitrary) symbols and then filling in the rule templates used. Once this is done, we can interpret the rules on the basis of their interaction with other rules and rename the symbols according to our traditional names for syntactic categories if an unambiguous interpretation is possible.

Analysis of cluster diagram for this example shows not only that a rule system has been formed that accounts precisely for the input sample, but also that these rules and categories can be put into a one-to-one correspondence with a natural standard CFG for the language at hand, such as:

| | | | | | |
|---|---|---|---|---|---|
| S | $\rightarrow$ | NP VP | N | $\rightarrow$ | square\|circle |
| NP | $\rightarrow$ | Det N | VT | $\rightarrow$ | touches |
| VP | $\rightarrow$ | VT NP | VC | $\rightarrow$ | is |
| VP | $\rightarrow$ | VC PP | P | $\rightarrow$ | above\|below |
| PP | $\rightarrow$ | P NP | *Det* | $\rightarrow$ | a |

(Fig. 3 explains how CFG symbols map to vector clusters.)

## Discussion

The details of the resulting rule and category structure are highly dependent on the training environment. For the example in the previous section, extreme conditions were intentionally chosen to generate the perfect correspondence between the structure learned and the traditional CFG. Specifically, constraining the number of rules to five forced a parsimonious use of categories. With more rules to work with either redundancies would have developed (several rules serving the same function) or some rules stay useless (never winning a competition and not converging onto meaningful categories). Also, the relatively large number of negative examples ensured that the categories formed were sufficiently discriminatory. With less or no negative examples a grammar develops that accounts for all the positive examples but fails to exclude all the negative ones, due to overly general rules.

The need for negative examples is the most bothersome problem if one is looking for a plausible mechanism for natural language acquisition (and widely acknowledged as a major challenge for many theories of acquisition, see, e.g., (Pinker 1989)). Although our current system is certainly too impoverished to claim to be a model of natural language acquisition (it handles only syntax, for one thing), it would be nice to obviate the need for negative examples.

Experiments show that just dropping the negative examples from the training set produces grammars with too few rules. They account for the training data by clustering a large number vectors together, resulting in categories that are too general and have too little discriminatory power to rule out false negative examples.

This situation arises partly due to a well-known problem with competitive learning schemes. A small number of rules win most of the competitions early, thereby pulling all category vectors into a few large pools. Many rules never get applied and never learn to be useful as a result. To counteract this tendency we have recently modified the learning algorithm to incorporate an idea from learning in *genetic systems* (Holland 1975). In the modified learning schedule, rules that never are used are periodically eliminated from the rule set and replaced by copies ('clones') of rules that are heavily used. These are the ones that tend to be overly general, and duplicating them allows the two copies to specialize into different roles in the grammar. A modified algorithm based on this heuristic does much better in positive-only training, and is able to derive a category structure similar to that in Fig. 3 with only one overly general cluster (merged VP and PP vectors, an error that is in fact motivated by the similar syntactic functions of these two categories).

Before concluding, we would like to contrast the general line followed here with some of the pure PDP approaches to language. Many of these take the view that learning networks have to 'discover' whatever structure is implicit in language, and are reluctant to provide the network with clues to this effect. In this approach, interpreting the results and representation obtained through successful learning (as well as their theoretical implications) becomes a major problem. Frequently researchers resort to post hoc analyses hoping to find familiar structures in their data using techniques similar to the ones used here (Elman 1988; Pollack 1990).

Part of the motivation underlying VSGs is that familiar structures (e.g., context-free rules) can be built into connectionist representations as a bias, allowing a more straightforward interpretation of the results. In the example, traditional categories emerged as a discrete approximation to the cluster structures developed in learning, thereby guiding their interpretation. (The case for symbolic representations as approximations to sub-symbolic entities has been made by Smolensky (1987).)

## Conclusions

We have argued for generalizations of traditional symbolic representations and models to benefit from some of the learning power found in connectionist systems without completely discarding the structural properties and intuitions embodied in traditional theories. As an example, we have introduced a generalization of phrase structure grammars, Vector Space Grammars, that is based on vectors instead of symbols to represent grammatical categories. An algorithm using VSGs based on the principles of adaptation of categories and competition between rules can be used to derive a syntactic category system from exposure to phrase-bracketed sample sentences. The results of learning can be interpreted in traditional notions by interpreting vector clusters as category symbols.

## Acknowledgements

I thank Jerry Feldman, Terry Regier, and Subutai Ahmad for valuable discussions of the ideas presented here. This work is supported by an IBM Graduate Fellowship.

## References

Elman, J. L. 1988. Finding Structure in Time. CRL Technical Report 8801, Center for Research in Language, University of California at San Diego, La Jolla, Calif.

Fass, L. F. 1983. Learning Context-Free Languages from their Structured Sentences. *ACM SIGACT News* 15(3).

Feldman, J. A., Lakoff, G., Stolcke, A., and Weber, S. H. 1990. Miniature Language Acquisition: A touchstone for cognitive science. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society*, 686–693, MIT, Cambridge, Mass.

Holland, J. 1975. *Adaption in natural and artificial systems.* Ann Arbor, Mich.: University of Michigan Press.

Langacker, R. 1985. *Foundations of Cognitive Grammar. Vol. 1: Theoretical Prerequisites.* Stanford: Stanford University Press.

Magerman, D. M., and Marcus, M. P. 1990. Parsing a Natural Language Using Mutual Information Statistics. In *Proceedings of the 8th National Conference on Artificial Intelligence*, Boston, Mass.

Morgan, J. L. 1986. *From Simple Input to Complex Grammar.* Cambridge, Mass.: MIT Press.

Morgan, J. L., Meier, R. P., and Newport, E. L. 1987. Structural Packaging in the Input to Language Learning: Contributions of Prosodic and Morphological Marking of Phrases to the Acquisition of Language. *Cognitive Psychology* 19:498–550.

Morgan, J. L., Meier, R. P., and Newport, E. L. 1989. Facilitating the Acquisition of Syntax with Cross-Sentential Cues to Phrase Structure. *Journal of Memory and Language* 28:360–374.

Pinker, S. 1989. Language Acquisition. In Posner, M. I., ed., *Foundations of Cognitive Science.* Cambridge, Mass.: MIT Press.

Pollack, J. B. 1990. Recursive Distributed Representations. *Artificial Intelligence* 46:77–105.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. 1986. Learning Internal Representations by Error Propagation. In (Rumelhart, McClelland, & The PDP Research Group 1986), 318–362.

Rumelhart, D. E., McClelland, J. L., and The PDP Research Group 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1: *Foundations.* Cambridge, Mass.: Bradford Books (MIT Press).

Rumelhart, D. E., and Zipser, D. 1985. Feature Discovery by Competitive Learning. *Cognitive Science* 9:75–112. Reprinted in (Rumelhart, McClelland, & The PDP Research Group 1986), pp. 151–193.

Smolensky, P. 1987. On variable binding and the representation of symbolic structures in connectionist systems. Technical Report CU-CS-355-87, University of Colorado, Boulder, Colo.

Stolcke, A. 1991. Vector Space Grammars and Grammatical Category Acquisition. Technical report, International Computer Science Institute, Berkeley, Calif. in preparation.

Zadeh, L. A. 1972. Fuzzy Languages and their Relation to Human and Machine Intelligence. In *Proceedings of the Conference on Man and Computer, Bordeaux, France, June 1970*, 130–165. Basel: S. Karger.