Tree Matching with Recursive Distributed Representations*

Andreas Stolcke[†] Dekai Wu[‡]

TR-92-025

April 1992

Abstract

We present an approach to the structure unification problem using distributed representations of hierarchical objects. Binary trees are encoded using the recursive auto-association method (RAAM), and a unification network is trained to perform the tree matching operation on the RAAM representations. It turns out that this restricted form of unification can be learned *without hidden layers* and producing good generalization if we allow the error signal from the unification task to modify both the unification network and the RAAM representations themselves.

*Paper to be presented at the Workshop on *Integrating Neural and Symbolic Processes—the Cognitive Dimension* at the National Conference on Artificial Intelligence, San Jose, CA, July 1992.

[†]Computer Science Division, University of California at Berkeley, and International Computer Science Institute, 1947 Center St., Berkeley, CA 94704, `stolcke@icsi.berkeley.edu`. Research supported by an IBM Graduate Fellowship and by the International Computer Science Institute.

[‡]Department of Computer Science, University of Toronto, Canada M5S 1A4, `dekai@cs.toronto.edu`. Beginning Fall 1992: Dept. of Computer Science, Hong Kong University of Science and Technology. This research was done at the Computer Science Division, University of California at Berkeley and the International Computer Science Institute, and was sponsored in part by the Defense Advanced Research Projects Agency (DoD), monitored by the Space and Naval Warfare Systems Command under N00039-88-C-0292, the Office of Naval Research under contract N00014-89-J-3205, and the Sloan Foundation under grant 86-10-3. Preparation of this document was partially supported by the Natural Sciences and Engineering Research Council of Canada.

1 Introduction

Representation and manipulation of hierarchical, recursively embedded structures pose considerable problems to connectionist models, which generally rely on fixed-width representations implemented in limited neural hardware. Recursive distributed representations formed by recursive auto-association (RAAM) have been proposed as a solution (Pollack 1988; Pollack 1990). Pollack's original work was mainly aimed at demonstrating the feasibility of the representational mechanism, and since then little has been done to investigate whether RAAM representations can actually support the kinds of structure-sensitive functions and operations that are typically applied to hierarchical representations in the symbolic domain (traversal, retrieval of substructures, unification, etc.).[1]

We believe that the goal of RAAMs (recursive, compositional representation) is a valid and important one in bridging the gap between current connectionist modeling and higher-level cognitive modeling (language, reasoning). In an attempt to narrow this gap 'from below', we have investigated extensions to the RAAM framework that allow it to perform non-trivial, interesting functions of the kind generally used in higher-level models.

The particular task investigated in this study is that of *unification*, a general function on structures like terms, trees, and graphs that implements a recursive matching and merging operation. One of us (Stolcke 1989) has previously proposed a structured (localist) connectionist model for unification and has argued that unification represents an operation that should have a natural connectionist implementation due to the intuitive notions if formalizes (feature integration, pattern matching). Unification has diverse applications but is particularly widespread in theorem proving and in research on natural language parsing and interpretation. See (Wu 1990; Wu 1992) for a unifying view of parallel parsing and interpretation, pattern completion, and unification.

2 Tree Matching

As a task suitable for evaluating the usefulness of RAAM encodings we consider an abstracted special case of unification, namely *tree matching*. The example domain is the set of binary trees with two kinds of leaves: variables (represented as '0') and non-variables or features (in the simplest case, there is only one feature, represented by '1' leaf). Matching (or unification) is an operation that combines two trees into a new tree. A variable node matches any tree and produces that tree as result. A feature node matches another (identical) feature node, which is then the result. Two complex (non-leaf) trees match if the corresponding subtrees match, and the subtree unifications form the subtrees of the result. See Fig. 1 for examples.

3 RAAM encoding

The method proposed by Pollack to encode arbitrarily deep trees in a fixed-width vector works by recursive compression of representations. Each leaf node has a predetermined

[1]Some attempts in this direction have not been convincing (Chalmers 1990); but see (Blank *et al.* 1992) for some very recent work.

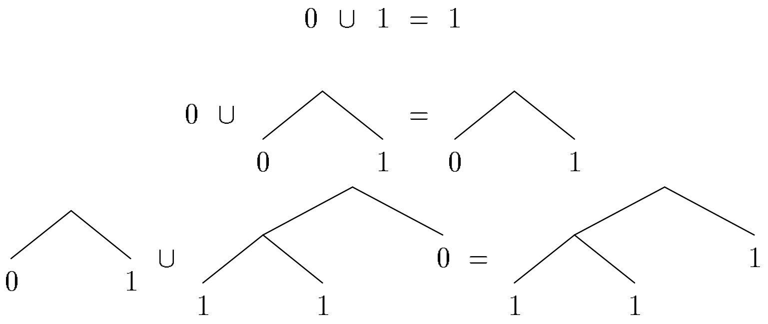


Figure 1: Some sample tree matchings and unification results. A '0' matches any leaf of subtree and leaves it unchanged. A '1' matches only itself. Matching and merging proceeds recursively from the root nodes.

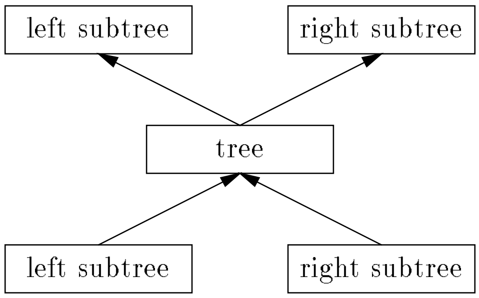


Figure 2: RAAM network architecture. The network is trained to reproduce the two input vectors (subtree encodings) at the output layer (auto-association). The hidden layer activations serves as compressed encoding for both subtrees. The process is repeated recursively to encode an entire tree.

vector representation of width n. Vector codes for complex trees are formed by compressing the two n component vectors for the subtrees into a new n component vector. This is accomplished using an auto-associator network with $2n$ inputs and outputs, and n hidden units, which will carry the compressed representations (Fig. 2). During training, one auto-associator is trained to reliably reproduce all trees in question. In the performance phase, the lower half of the network serves as an encoder, the upper half as a decoder, each of which have to be used iteratively to encode/decode an entire tree.

A important issue in RAAM-based networks is proper evaluation of the network's performance. Some of the literature suggests that it is sufficient to train networks up to a point where the subtree vectors after decoding are within a 'sufficient small' tolerance ϵ of the encoded vectors. This is misleading, however, since there is no way to predict how small ϵ should be. Rather, repeated recursive decoding can amplify any error to a point where the leaves of an encoded tree cannot be recovered reliably. We therefore used complete accurate decoding of the entire tree as the only error criterion in all our experiments.

Another critical problem in decoding RAAM representations is the identification of terminal codes, i.e., those representing leaf nodes of the tree. If a leaf node is mistaken for an internal node and decoded further, the result is guaranteed to be garbage. The most common approach to this problem is to choose terminal codes among binary (0/1 valued) vectors. A code vector is then deemed terminal if all its components are within some ϵ of 0 or 1. A somewhat more elegant solution is to train the network to explicitly dedicate a bit to the distinction between terminals and nonterminals. This can be achieved during training of the auto-associator by injecting an extra error in one of the bits of the hidden layer, forcing it to be 1 for all nonterminal nodes, while choosing all terminal codes to have a 0 in that bit. We found this method to give more reliable results and used it in all experiments reported below.

4 Unifying RAAM-encoded trees

To investigate the power of RAAM encodings, we deliberately restrict the network that learns the unification task to a particularly simple architecture with no hidden layers (Fig. 3). Given a pair of input vectors representing two unifiable input trees (using RAAM encoding), the network is to produce an output vector representing their unification (as verifiable using RAAM decoding).

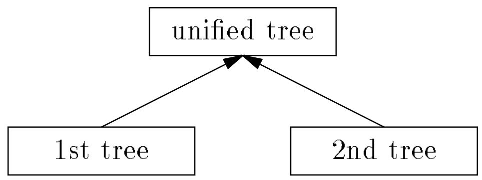


Figure 3: Unification network. A simple one-layer network takes two RAAM-encoded trees and maps them onto the RAAM code for the unified tree.

We have investigated two modes of operation upon RAAM representations. In the simpler case, representations for trees are first learned using standard RAAM. The unification network is then trained with instances comprised of two unifiable input trees and the desired (unified) output tree, all of which are encoded using the RAAM network.

In the second *parallel training* mode we allow the representation of trees to evolve dynamically to fit the task. This is accomplished by training the RAAM and unification networks simultaneously. The error between the RAAM-encoded result and the output of the unification network is backpropagated both through the unification network and also recursively through the RAAM network. The RAAM encoder develops tree encodings as moving targets for unification, which in turn helps shape representations so that an efficient (linearly separable) solution to the unification problem is possible. Parallel training allows shifting part of the work of unification from the unification network itself to the representation. This contrasts with other work (e.g., Blank *et al.* 1992) where multi-layer networks are used to process RAAM codes for tasks which appear less demanding than that investigated here.

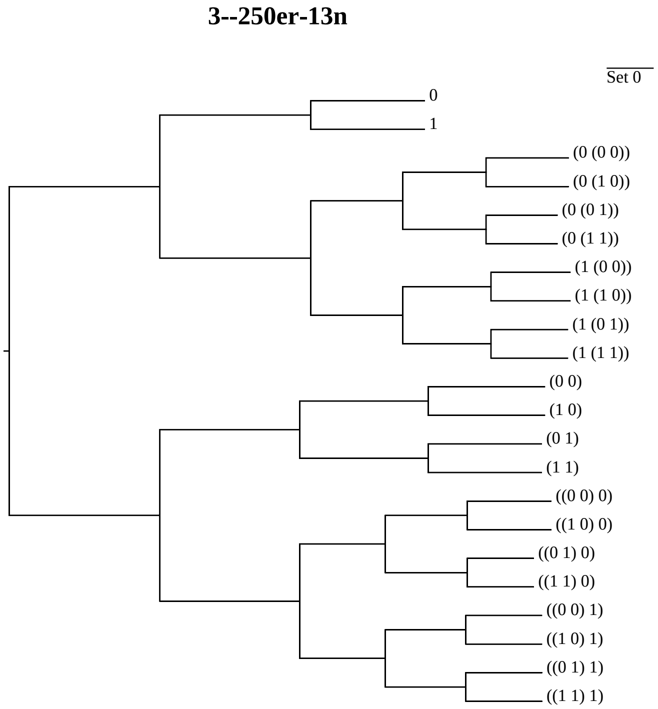


Figure 4: Cluster analysis of RAAM representations for trees with three leaves.

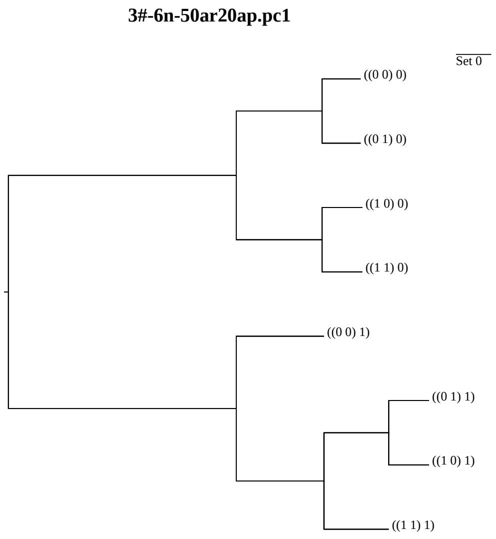
5 Results

Prior to studying unification performance, we studied the performance of RAAM alone on trees of up to five terminal nodes, and obtained good results for both learning and generalization. For example, having been trained on 70% of all trees of four leaf nodes, it consistently encodes and decodes the remaining 30% with 100% accuracy.

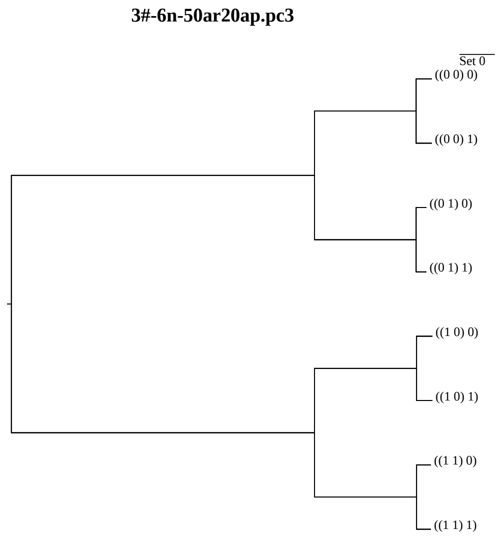
The representations developed exhibit exactly the sort of similarity relationships one would intuitively generate if asked to group tree structures by hand. Fig. 4 shows the Euclidean-distance cluster analysis for a typical case, showing the primary factor to be tree structure.

Within each set of identically-structured trees, the values of the terminals govern the sub-clustering, with terminals closer to the root being more important than deeper terminals. Fig. 5 demonstrates how, for a particular structure, the values of the individual terminals are encoded by different coordinates along the principal component axes. A width-six vector representation was learned using parallel training, and the representation for the subset of tree structures of the form ((x x) x) is analyzed. In (a) the first principal component dimension is shown; it identifies the rightmost terminal, which is also outermost. In (b) the third principal component dimension identifies the left inner terminal. In (c) the sixth principal component dimension identifies the right inner terminal.

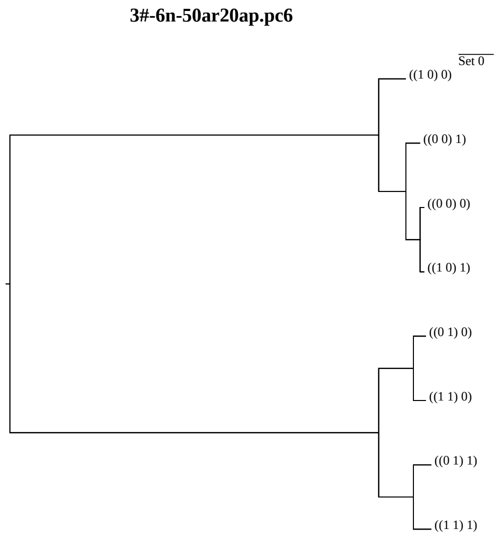
Using parallel training, performance of the unification network on the training set is



(a)



(b)



(c)

Figure 5: Analysis of a width-six RAAM encoding along the principal component axes.

generally at or near 100%. Table 1 shows the results for three runs starting with different initial conditions. Vectors of length 25 were used to encode trees of up to 4 leaf nodes, terminal nodes were encoded by fixed orthogonal unit vectors. The training set consisted of 1001 unifications, formed by taking all unifying pairs among all trees of length ≤ 3 and a random 70% of all trees of length 4 (giving a total of 82 trees). The remaining 20 trees were paired among themselves and with the ones in the training set to form another 404 possible unifications which served as the test set. Training started with a momentum of 0.5 and a learning rate of 1.0, which was successively decreased by half every time the learning curve flattened out. Training stopped after the indicated number of epochs after the learning rate had dropped below a threshold level. The last column shows the single unification result that accounted for all incorrect outputs in both the training and the test sets. In both these cases the network output a tree that, after decoding, showed a '1' instead of the correct '0'.

Run	Training set	Test set	Epochs	Error pattern
1	100% (1001)	100% (440)	50	
2	98.8% (989)	90.5% (398)	49	(x (x (0 x)))
3	98.3% (984)	95.2% (919)	60	(x ((0 x) x))

Table 1: Unification performance and generalization (percentage and number of samples correct) with three different random initial conditions.

Remarkably, performance on the test set (generalization) is always very close to that on the training set. As expected, the errors occur at the most deeply embedded substructure. In each of the two runs with less than 100% accuracy, the same bit at the deepest level accounted for all the errors.

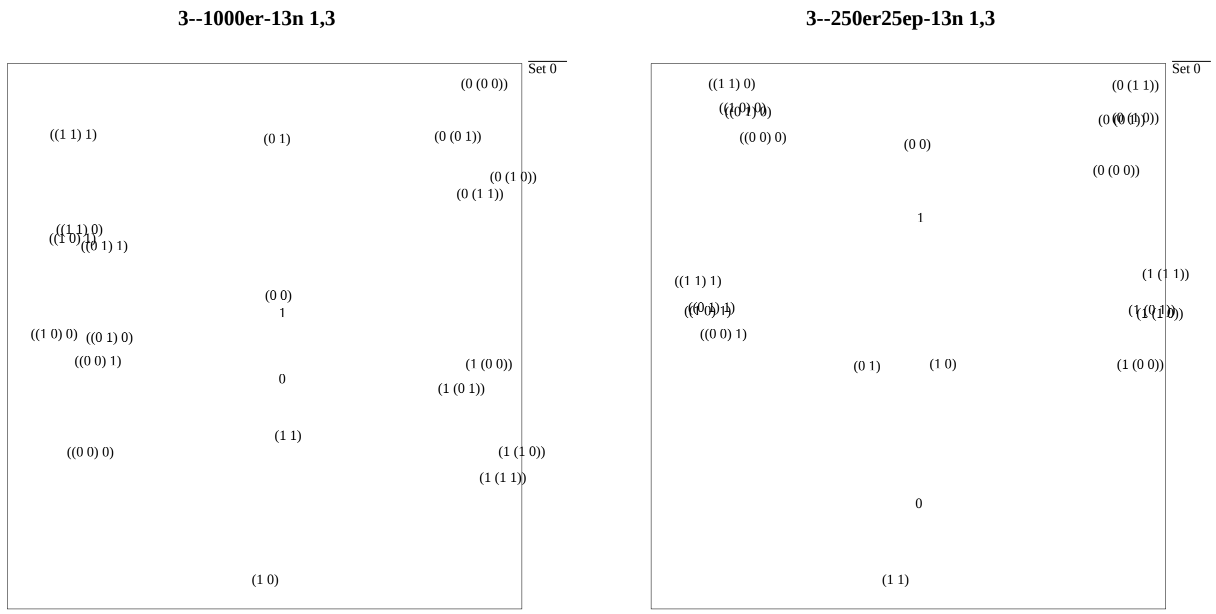


Figure 6: Vector representations for trees, plotting the first against third principal component dimensions, (a) with RAAM training only, and (b) with parallel RAAM and unification training.

Not only does unification influence the representation under parallel training, but in fact, depending on initial conditions, we have sometimes only been able to achieve good unification performance with parallel training. For one such case, Fig. 6 shows that the RAAM-only representation, which permitted only 67% unification accuracy, was organized much less tightly than the successful parallel-training representation that yielded 99% accuracy.

We have observed that the fastest and most accurate convergence is achieved by initially training the RAAM network to 100% encoding/decoding performance, then simultaneously training the unification and RAAM networks. On the one hand, this indicates that the valley region in weight space surrounding the convergence point of the RAAM-only network contains other easily reached solutions, including those that also work for unification. On the other hand, this also indicates that the combined RAAM-unification weight space contains local minima from which it is difficult to reach the optimal solution. A solution is much more easily attained from a point corresponding to a minimum in the RAAM-only weight space, than from a general random initial point. This supports Elman's 1991 observation that presenting input incrementally facilitates learning more complex functions that are otherwise unlearnable.

6 Extensions

The next step is to consider functional tasks of greater complexity. As a second step toward a more realistic unification function, we have applied the same technique to learning the task of unifying one structure into a predetermined *internal* node of the other. This task is also consistently learned perfectly and with excellent generalization.

We have made preliminary investigations into learning a third, far more ambitious *context-sensitive unification* task. In the version we used, one structure is unified with either the root or an internal node of the other, preferring the root if possible. Context-sensitive unification is needed for general recognition tasks such as natural language parsing or interpretation, where it is not known *a priori* how input fragments should be combined. Although initial runs have yielded up to 98% accuracy on learning this task, generalization appears to be poor. This is not surprising considering the simplicity of the no-hidden-layer unification network, and we would expect to need more structure to implement context switches.

One important limitation of our investigations so far is that we have presented the unification network with positive samples only. In general we would want the network to not only unify (or in the case of trees, match) unifiable structures, but also indicate unification failure. The networks trained on positive samples only seem to produce approximate solutions if asked to unify two non-unifiable structures. This could in fact be a promising property of a connectionist approach to the structure unification problem, but we haven't yet investigated this possibility systematically.

7 Conclusions

When we began this investigation we were rather skeptical about the suitability of RAAM representations as a general-purpose representation for hierarchical objects. We were truly surprised that the trivial network in Fig. 3 was successful in performing the unification task for the limited domain we had given it. We remain skeptical nevertheless, if only for the prohibitive scaling properties of backpropagation learning, which would require very large networks, sizable training sets and long training times to learn a more realistic domain.

That said, however, our study of RAAM-encoded tree unification indicates that at least in principle, *functionally efficient* distributed representations can be learned for hierarchical structures. Using an objective function that simultaneously drives the representation to improve compression and functional performance results in improved performance on both criteria individually.

References

Blank, D. S., L. A. Meeden, & J. B. Marshall (1992). Exploring the symbolic/subsymbolic continuum: A case study of RAAM. In J. Dinsmore, editor, *Closing the Gap: Symbolism vs. Connectionism*. Lawrence Erlbaum Associates. To appear.

Chalmers, D. J. (1990). Transformations on distributed representations. *Connection Science*, 2.

Elman, J. L. (1991). Incremental learning, or the importance of starting small. In *Program of the Thirteenth Annual Conference of the Cognitive Science Society*, pp. 443–448.

Pollack, J. B. (1988). Recursive auto-associative memory: Devising compositional distributed representations. Technical Report MCCS-88-124, Computing Research Laboratory, New Mexico State University, Las Cruces, New Mexico.

Pollack, J. B. (1990). Recursive distributed representations. *Artificial Intelligence*, 46:77–105.

Stolcke, A. (1989). Unification as constraint satisfaction in structured connectionist networks. *Neural Computation*, 1(4):559–567.

Wu, D. (1990). Probabilistic unification-based integration of syntactic and semantic preferences for nominal compounds. In H. Karlgren, editor, *Proceedings of the 13th International Conference on Computational Linguistics*, University of Helsinki, Helsinki, Finland.

Wu, D. (1992). *Automatic Inference: A Probabilistic Basis for Natural Language Interpretation*. PhD thesis, University of California at Berkeley.