Scaling Connectionist Compositional Representations

John C. Flackett
Faculty of Technology
Southampton Institute
Southampton, SO14 0RD, UK
john.flackett@solent.ac.uk

John Tait
School of Computing and Technology
University of Sunderland
Sunderland, SR6 0DD, UK
john.tait@sunderland.ac.uk

Guy Littlefair
Faculty of Technology
Southampton Institute
Southampton, SO14 0RD, UK
guy.littlefair@solent.ac.uk

Abstract

The Recursive Auto-Associative Memory (RAAM) has come to dominate connectionist investigations into representing compositional structure. Although an adequate model when dealing with limited data, the capacity of RAAM to scale-up to real-world tasks has been frequently questioned. RAAM networks are difficult to train (due to the moving target effect) and as such training times can be lengthy. Investigations into RAAM have produced many variants in an attempt to overcome such limitations. We outline how one such model ((S)RAAM) is able to quickly produce context-sensitive representations that may be used to aid a deterministic parsing process. By substituting a symbolic stack in an existing hybrid parser, we show that (S)RAAM is more than capable of encoding the real-world data sets employed. We conclude by suggesting that models such as (S)RAAM offer valuable insights into the features of connectionist compositional representations.

Introduction

Connectionist architectures, such as the feed-forward back-propagation network (Rumelhart & McClelland 1986), can form non-linear associations between input and output data. However, they are limited inasmuch as they fail to capture any combinatorial structure in which the data belongs. This compositional gauntlet laid at the feet of connectionists (Fodor & Pylyshyn 1988), sparked a multitude of techniques that allow recursive structures to be encoded (Pollack 1990; Smolensky 1990; Touretzky 1990; Plate 1995). Of these techniques, it is Pollack's (1990) Recursive Auto-Associative Memory (RAAM) that has dominated research activities involving connectionist compositional representations. Many studies concerned with achieving a better understanding of RAAM based models and the properties of their representations have been undertaken (Chalmers 1990; Blank, Meeden, & Marshall 1992; Balogh 1994; Bodén & Niklasson 1995; Kwasny & Kalman 1995; Hammerton 1998). Even so, it is well known that connectionism is a difficult medium to work with when dealing with large amounts of data. Long and difficult training may ultimately lead to poor learning and limited, if any, generalization capabilities. In order to utilize recursive, structured connectionist representations, in non-trivial applications (such as natural language processing), we must address both the underlying compositional representation and the storage capacity of the network.

Recursive Auto-Associative Memory - RAAM

Essentially a feed-forward network, RAAM is able to encode symbolic structures, such as sequences and trees, by deriving compressed internal representations recursively. The network learns to produce on the output layer whatever is presented on the input layer, and in doing so, constructs a compressed representation of the input on the hidden layer. This compressed representation is then fed back into the network with more input, which creates a further compressed representation that is a composition of the initial compressed representation and the new input. Retrieval of encoded structures is achieved by presenting the compressed representation to the decoder (hidden layer + output layer), which then performs the decode operation (recursively). Unfortunately, the recursion within RAAM results in the network having to learn input patterns that are constantly changing. This property, known as the *moving target effect*, means that the output layer can never perfectly match what is presented on the input layer. It is therefore necessary to cease training when a certain tolerance is reached. Achieving this stopping condition can be difficult and is, for the most part, inordinately time consuming. Another problem with RAAM is that the identification of terminals (leaf nodes of a structure) within the structured representations can be difficult upon reconstruction, and this problem seems to increase with the depth and complexity of a structure that is encoded (Blair 1997). A final limitation with RAAM is that it is only able to encode fixed-valance trees, thus precluding the use of certain types of data.

Many variants of the original RAAM model have been proposed in an effort to overcome the initial shortcomings of the technique. For instance, Sequential RAAM (SRAAM)(Kwasny & Kalman 1995) uses pre-processing to transform any input tree into a sequence. However, although this addresses the fixed-valance constraint and reduces structure complexity it does not resolve the depth issue - indeed, it actually exacerbates the problem. More recent work on RAAM identifies the main failing being with the terminal test (usually a simple distance measure) which, during the decode operation tests to see if the output from a RAAM is a terminal or whether it is a sub-tree representation that requires further decoding. This lead to the development of Infinite RAAM (IRAAM) (Melnik, Levy, & Pollack 2000) which uses fractal geometry to remove the ambiguity of terminal identification and therefore allows much larger data sets to be encoded.

In conclusion, a useful RAAM model would be one that not only has the capacity to encode large amounts of data, but also trains relatively quickly.

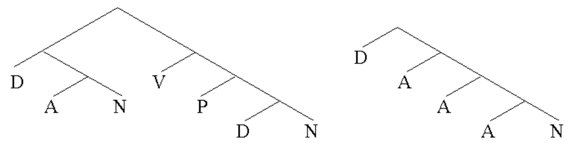
*Simplified*RAAM -(S)RAAM

Pollack (1990) cited, as a concern for further research into RAAM, the need for:

> Developing...a general representational scheme which could be analytically derived for a particular representational task without relying on slow, gradient-descent learning. (p.101)

In an effort to realize this, *Simplified*RAAM ((S)RAAM) was developed (Callan 1996). (S)RAAM offers a simplification to the training of a RAAM by analytically deriving internal network representations via Principal Component Analysis (PCA). The (S)RAAM model derives two matrices which imitate the first and second layers of weights (encoder and decoder) in a RAAM. These are called the *constructor* and *reconstructor* matrices. The constructor matrix in (S)RAAM is basically the set of eigenvectors retained from performing PCA on an initial training matrix[1]. Given the two trees shown in Figure 1 as our data set, we generate the training matrix by replacing all terminals in each of the trees with unique bit patterns. As opposed to RAAM, (S)RAAM does not use an external stack to re-circulate sub-trees back to the input layer therefore we must provide all trees and sub-trees as part of the training matrix (with smaller trees padded with zeros so all initial tree patterns are the same width). PCA is then performed on the training matrix and all principal components retained. The product of the training matrix and eigenvectors is then taken to produce internal (S)RAAM representations for each [sub-]tree. A new training matrix is then defined with

[1] The reconstructor is an approximate inverse of the constructor therefore we concentrate here on derivation of the constructor.



(a) D-V-A-N-P-D-N **(b)** D-A-A-A-N

Figure 1: The trees ((D(AN))(V(P(DN)))) and (D(A(A(AN)))). A (S)RAAM training representation is generated by performing a breadth-first scan of each tree ((a) and (b)) and replacing each terminal with a unique bit pattern.

any sub-trees being replaced with the internal representation for that sub-tree. PCA is then applied again and the whole process continues until the number of eigenvalues stabilizes. The final set of eigenvectors become the constructor matrix. Since (S)RAAM does not rely on normal gradient decent learning, the amount of time it takes to learn is decreased considerably (as an example, whereas RAAM took approximately 50 minutes to learn a small data set, (S)RAAM took just 1 minute). (S)RAAM exhibits some favorable properties which makes it an interesting vehicle for deriving compositional connectionist representations:

- analytical technique which facilitates representational analysis;
- reliable reconstruction of constituent parts;
- representational width is discovered automatically;
- representations are suited to holistic processing;
- structure generalization can be safely predicted without the need for explicit testing as (S)RAAM generalizes to linear combinations of structures seen during training; and
- no theoretical limit on size of data that may be encoded (only computational limit).

Some of the work that we have carried out to investigate these characteristics is now detailed.

Constituents?

Experiments outlined in Bodén & Niklasson (1995) can be used to show how it is possible to perform structure sensitive processing on (S)RAAM representations. A set of trees are created using data that is fully representative and complete. These trees contain the six different constituents *a*, *b*, *c*, *d*, *e* and *f* along with a terminating *nil* token. Each constituent is assigned a 1-in-6 binary bit pattern (with *nil* as all zeros). Constituents appear in every possible position in a tree and there are 3 tree structures (left, right or well-balanced). Each tree may have a depth of either 2 or 3 e.g., the left-balanced tree (((*e nil*)*a*)*c*) with depth 3. This gives a total of 540 unique trees.

In contrast to RAAM, (S)RAAM automatically generates the size of the internal representations - determined by the number of principal components (eigenvalues) retained. If all principal components are retained (S)RAAM is a lossless representational scheme. Encoding the 540 trees in (S)RAAM produced a network equivalent of 24x36x24. That is, (S)RAAM derived an internal representation width of 36 elements (24 elements larger than Bodén & Niklasson's RAAM). It is important to note however, that (S)RAAM could fully encode and decode all 540 trees successfully and at the first attempt. Given the same set of data to learn, (S)RAAM creates the same external-to-internal mapping every time therefore, we only needed to train one (S)RAAM to provide a full set of internal representations. To identify the properties encoded within the compositional representations (without decoding), Bodén & Niklasson proposed five different experiments on the set of representations:

Tree Depth: determine the depth of an encoded tree,

Balance: determine *balance* of a tree,

Last Leaf: determine final tree leaf constituent,

2nd Last Leaf: determine 2nd to last leaf constituent,

Initial State: determine the initial tree state.

To extract the necessary information from the encodings, 10 feed-forward, backpropagation networks were trained to classify the internal (S)RAAM representations according to the feature being tested. Although all the above experiments were carried out (and the results obtained were much more favorable for (S)RAAM than for RAAM) of most relevance (in the context of this paper) is the identification of the Last Leaf constituent. All ten networks correctly learnt the complete training set (300 of the 540 trees). However, upon testing the networks with the remaining 204 trees[2] there were between 31 and 40 misclassifications. It is known that RAAM models dedicate most effort into retaining information about the last item encoded (Pollack 1990; Callan 1996). Considering this, it is surprising that no network was able to fully identify all the last leaf constituents.

To evaluate this failing we looked closely at the representations created by (S)RAAM. Balogh (1994) showed that RAAM representations are localist in nature therefore, it would follow that (S)RAAM representations also exhibit a similar property. Theoretically, it should be possible to isolate the particular elements of an internal representation, which go to form individual constituents. The question is how?

The answers lies in analysis of (S)RAAMs constructor. A clue as to which elements in an internal representation make up a constituent is given in the form of a large weighting in the constructor matrix. From close analysis we can identify which elements in an (S)RAAM representation must be present for the last terminal to be fully reconstructed. For example, to be able to recover the terminal *c* from the internal representation for the tree (((*nil b*)*d*)*c*), the elements 1, 5, 7, 8, 11 and 21 are required. This means that all the information about the *c* terminal is held in these six elements of the (S)RAAM representation for the whole tree. This makes it possible to extract terminal representations analytically (see Flackett (1998) for full experimental details and results). However, using simple vector subtraction is an easier method to produce an internal (S)RAAM representation for an encoded terminal. For instance, if we encode the sentence *The man put money in the bank*, we can extract an internal representation for the word *bank* by subtracting the encoded sub-tree representation for *The man put money in the* from the original representation. Interestingly, the resulting representation is context sensitive. Using spatial similarity measures (i.e., clustering) of extracted terms, we can show that tokens used in similar contexts have similar representations. In a nutshell, it is possible to extract some kind of semantic information from both trees and tokens encoded in (S)RAAM. This work indicates that (S)RAAM provides us with useful strategy(s) with which to investigate connectionist compositional representations.

Scaling (S)RAAM

As a technique for qualitative analysis of RAAM based models, (S)RAAM proves to be a practical approach and provides a mechanism that is both fast and reliable. However, in order for it to be a useful technique that connectionists may actually apply (rather than just a reductionist tool), the ability to scale-up to large data sets is of the utmost importance. One area in which the capacity to encode large amounts of data can be tested, is in natural language processing (NLP).

Using an annotated natural language corpora, Tepper (Tepper, Powell, & Palmer-Brown 2002) presented a hybrid, connectionist, deterministic, shift-reduce parser. This architecture (shown in Figure 2) parses in a right-to-left fashion and consists of the following modules:

RLD: right-to-left delimiter. Recurrent network that identifies the start of a phrase to be reduced.

LRD: left-to-right delimiter. Recurrent network that identifies the end of a phrase to be reduced.

PSR: feed-forward network that reduces the group of words (identified above) to the appropriate phrase.

Input Stack: pre-tagged sentence to be parsed. Words are 'popped' to the RLD and reduced phrases 'pushed' from the PSR.

[2] You cannot test for the existence of last leaf constituent(s) with well-balanced trees therefore they are removed from the data set.

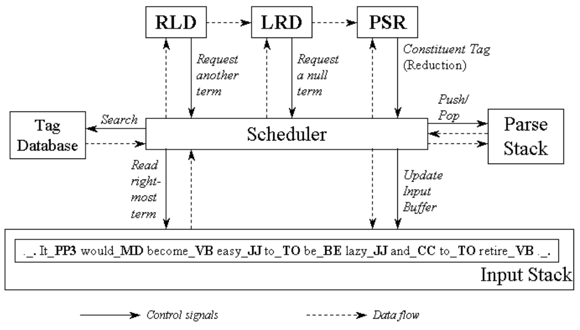


Figure 2: Tepper's shift-reduce, hybrid parsing architecture. Taken from (Tepper, Powell, & Palmer-Brown 2002, p.98).

Tag Database: a list of all possible words and phrases and their associated bit-pattern representations.

Scheduler: controls the transfer of data around the parser.

Parse Stack: used to hold the current state of the parse. Reductions made by the PSR are stored in this module.

The stacks, database and scheduler are all symbolic components, whilst the RLD, LRD and PSR are connectionist modules. The implementation of a traditional stack to maintain the current parse state (Parse Stack) is justified by Tepper on the grounds that

> ...existing connectionist approaches to structure representation...have not been shown to generalize effectively [and]...do not offer significant advantages over symbolic approaches. (p.97)

We disagree and argue that (S)RAAM representations lend themselves well to the task of shift-reduce parsing and may replace the Parse Stack in the current architecture. Substituting the traditional symbolic component for (S)RAAM in maintaining the current parse state, removes the requirement for a profusion of stack operations (pushes and pops) during the parsing process. The PSR can be taught to output an internal (S)RAAM representation that models the required reduction of terms. More importantly these internal representations carry context sensitive information that may be used by the parser (RLD, LRD and PSR) to aid the deterministic parsing process. Instead of re-circulating meaningless symbolic tokens back onto the Input Stack, internal (S)RAAM representations may be employed.

In the first instance however, we need to ensure (S)RAAM is capable of encoding the necessary amount of data required for the application. Tepper makes use of the Lancaster Parsed Corpus (LPC) (Garside & Varadi 1987) with a training set of 654 sentences (a total of 3706 unique trees and sub-trees). (S)RAAM succeeded in learning the data perfectly, converging in just under two weeks. This task would be unthinkable with a conventional RAAM network. On the face of it, the initial stage of scaling (S)RAAM seems complete.

The results presented throughout this paper indicate that (a) meaningful, context-sensitive, compositional, connectionist representations, offer advantages over traditional symbolic approaches, and (b) using real-world data (S)RAAM is able to learn effectively enough to be a useful module in a hybrid parsing architecture. However, certain obstacles remain. The internal representations generated by (S)RAAM are ~1800 elements in length. This obviously poses problems for other networks that attempt to make use of them e.g., LRD, RLD and PSR. Although it is possible to reduce the length of the representations by 'dropping' minor components, it is unclear what effect this actually has on the learning and generalization capabilities when dealing with large data sets. Certainly, initial input encodings play a major part in how (S)RAAM learns (in both time and resources). We are currently investigating whether it is possible to encourage (S)RAAM to represent data structures in a more compact manner by paying close attention to the input.

Compositionality and (S)RAAM

Since the inception of (S)RAAM, there has been little (presented) research detailing the capabilities of the technique. One of the major problems faced by connectionists is the long and difficult training of traditional network approaches. (S)RAAM overcomes this problem by deriving internal compositional representations via PCA. Additionally, the network topology is determined automatically and therefore relieves the practitioner of this (sometimes) difficult design task. Incorporating (S)RAAM in an existing parsing framework, allows us to measure the benefits of utilizing structured connectionist representations in place of traditional symbolic techniques. The ability to scale-up to real-world data may also be evaluated. Additionally, if we are to fully understand the features of connectionist generated compositional representations then we need a mechanism that facilitates low-level investigation. (S)RAAM provides one such approach.

References

Balogh, I. L. 1994. An analysis of a connectionist internal representation: Do RAAM networks produce truly distributed representations? Ph.D. Dissertation, New Mexico State University, Las Cruces, NM.

Blair, A. 1997. Scaling-up RAAMs. Tech. Report CS-97-192, University of Brandeis.

Blank, D. S.; Meeden, L. A.; and Marshall, J. B. 1992. Exploring the symbolic/subsymbolic continuum: A case study of RAAM. In Dinsmore, J., ed*., The Symbolic and Connectionist Paradigms: Closing the gap.* Hillsdale, NJ: Lawrence Erlbaum.

Bodén, M., and Niklasson, L. 1995. Features of distributed representations for tree-structures: A study of RAAM. In Current Trends in Connectionism-*Proceedings of the 1995 Swedish Conference on Connectionism*. Skövde: Lawrence Erlbaum.

Callan, R. E. 1996. Netting the Symbol: Analytically Deriving Recursive Connectionist Representations of Symbol Structures. Ph.D. Dissertation, Systems Engineering Faculty, Southampton Institute, England.

Chalmers, D. J. 1990. Why Fodor and Pylyshyn were wrong: The simplest refutation. In Proceedings of The Twelfth Annual Conference of the Cognitive Science Society, Cambridge, MA, July 1990, 340-347. Hillsdale, NJ: Lawrence Erlbaum Associates.

Flackett, J. C. 1998. Features of (S)RAAM representations. Internal Poster Presentation, SERC'98, Southampton Institute.

Fodor, J. A., and Pylyshyn, Z. W. 1988. Connectionism and cognitive architecture: a critical analysis. In Pinker, S., and Mehler, J., eds., *Connections and Symbols*. Cambridge, Mass.: MIT Press.

Garside, R., L. G., and Varadi, T. 1987. Manual of Information to Accompany the Lancaster Parsed Corpus. Department of English, University of Oslo.

Hammerton, J. 1998. Exploiting Holistic Computation: An evaluation of the Sequential RAAM. Ph.D. Dissertation, School of Computer Science, University of Birmingham.

Kwasny, S. C., and Kalman, B. L. 1995. Tail-recursive distributed representations and simple recurrent networks. *Connection Science* 7(1):61-80.

Melnik, O.; Levy, S.; and Pollack, J. B. 2000. RAAM for infinite context-free languages. In *International Joint Conference on Neural Networks*. Como, Italy: IEEE Press.

Plate, T. 1995. Holographic reduced representations. *IEEE Transactions on Neural Networks* 6(3):623-641.

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

Rumelhart, D. E., and McClelland, J. L. 1986*. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. I: Foundations*. Cambridge, MA: MIT Press.

Smolensky, P. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. In Hinton, G., ed., *Artificial Intelligence*, volume 46. 159-216.

Tepper, J. A.; Powell, H. M.; and Palmer-Brown, D. 2002. Corpus-based connectionist parsing. *Connection Science*, 93-114.

Touretzky, D. S. 1990. Boltzcons: Dynamic symbol structures in a connectionist network. *Artificial Intelligence* 46(1&2):5-46.