Simple Synchrony Networks : Learning to Parse Natural Language with Temporal Synchrony Variable Binding*

Peter C.R. Lane† and James B. Henderson
Department of Computer Science, University of Exeter
Prince of Wales Road, Exeter EX4 4PT, UK
{pclane,jamie}@dcs.exeter.ac.uk

Abstract

The Simple Synchrony Network (SSN) is a new connectionist architecture, incorporating the insights of Temporal Synchrony Variable Binding (TSVB) into Simple Recurrent Networks. The use of TSVB means SSNs can output representations of structures, and can learn generalisations over the constituents of these structures (as required by systematicity). This paper describes the SSN and an associated training algorithm, and demonstrates SSNs' generalisation abilities through results from training SSNs to parse real natural language sentences.

1 Introduction

Temporal Synchrony Variable Binding (TSVB) [1] extends the representational ability of a connectionist network to include entities. The original motivation behind TSVB was for a network to represent variables and so carry out symbolic reasoning [1]. Henderson [2] argues that this extension further gives connectionist networks an inherent ability to learn generalisations across entities. This ability allows TSVB networks to learn the kinds of regularities that arise from a compositional generative grammar, which [3] uses to describe the property of systematicity. In particular, with tasks involving language, TSVB networks will generalise information learned about one syntactic constituent to other syntactic constituents.

This paper begins by describing the basic idea behind TSVB, which is that with pulsing units entities can be represented using the timing of pulses. The pulsing binary-threshold units of [1] provide a connectionist model of structures and symbolic reasoning, but are not suitable for training with standard algorithms such as backpropagation [4]. To develop an architecture for TSVB networks that can use backpropagation, we start with the Simple Recurrent Network (SRN) architecture [5] and extend it with units that pulse. The resulting Simple Synchrony Network (SSN) architecture has two SRN components, one standard SRN that represents overall context, and one TSVB SRN that represents a set of entities. This information is combined to compute the output for each entity. As for SRNs, SSNs can be trained using Backpropagation Through Time [4]. Finally, we demonstrate the ability of SSNs to

[0]This paper appears in the Proceedings of ICANN 1998, Skövde, Sweden.
†Supported by the Engineering and Physical Sciences Research Council, UK.

represent structure and learn generalisations over structural constituents with results from experiments training SSNs to parse a corpus of natural language sentences.

2 Temporal Synchrony Variable Binding

Temporal Synchrony Variable Binding [1] is a connectionist technique for representing entities. We will adopt the following central characteristics of TSVB:

- the division of each time period into discrete phases,
- pulsing units, to compute within each phase independently of other phases,
- non-pulsing units, to compute across all phases, combining information about several entities.

Thus in each time step the network cycles through the set of entities, using pulsing units to compute about each entity independently. To communicate information between entities there are also non-pulsing units, which compute across all phases and thereby represent information about the overall context. Being able to compute in terms of entities, overall context, and their interactions is a crucial feature of the architecture proposed below.

The use of TSVB in a connectionist network has two important consequences. The first is that the output can represent structure. The activation of output units in a particular phase can represent information about a particular constituent in the structure, including its structural relationships to other constituents. The second consequence is that TSVB networks inherently generalise information learned about one entity to other entities. Because different times (i.e. phases) are used to represent different entities, and the same link weights are used at every time, the same learned information is applied to every entity. This argument is used in [2] to demonstrate that TSVB networks possess inherent systematicity [3].

3 Simple Synchrony Networks

Rather than starting with an existing TSVB architecture and making it compatible with backpropagation, we choose to start with an existing backpropagation architecture and add TSVB. A natural choice is the Simple Recurrent Network (SRN) architecture [5]. This architecture already handles temporal sequences. An SRN accepts a sequence of input patterns and produces a sequence of internal and output patterns. TSVB requires such sequences both for each entity and for the overall context. The known effectiveness of learning in SRNs implies that each of these sequences can be learned effectively, but this is not sufficient, since the sequences are not all independent. The Simple Synchrony Network (SSN) architecture minimizes the amount of interaction between these sequences while maintaining the generality of the computations that can be performed. This approach preserves the effectiveness of learning in these networks.

3.1 The architecture and training algorithm

Figure 1 illustrates a SSN architecture. The two recurrent components at the bottom are each SRNs (minus their output layers). The lefthand SRN

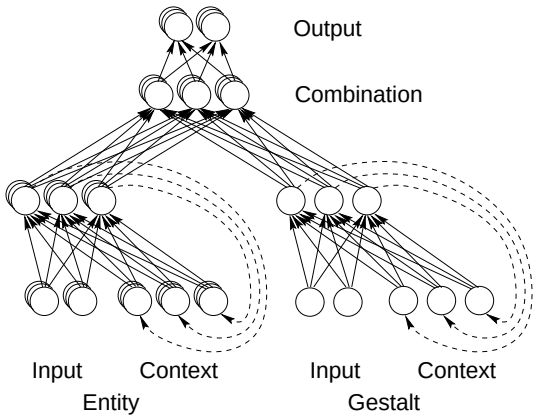

Figure 1: A Simple Synchrony Network. The pulsing units are depicted as several units stacked on top of each other, because they store activations for several entities.

(the entity component) consists of pulsing units, and computes a distributed representation of each entity independently. The righthand SRN (the gestalt component) uses nonpulsing (i.e. standard) units, and computes a distributed representation of the overall context. Thus as a SSN processes a sequence of inputs, it computes one sequence of representations for each entity plus one sequence of representations for the overall context. After each step, the upper hidden layer combines the overall context representation with each entity's representation to produce the output pattern for each entity.

What allows a SSN to learn effectively is the way it combines information between the separate sequences of representations computed by its SRN components. One type of interaction is when information about entities is needed to compute the overall context. In [1] this is done with logical AND and OR combinations across entities, but these cannot be used with backpropagation. Continuous combination operations, such as summation, do not in practice generalise to greater or fewer numbers of entities, and thus cannot be used. The solution adopted in SSNs is to push these dependencies down into the input layer. Any information that is input about an entity and might be relevant to the overall context must also be represented in the input to the gestalt component. As long as information about a fixed number of entities is *input* at any one time (in our application this is one), there will be no need for the input representation to generalise over different numbers of entities.

Another type of interaction between sequences is when information about the overall context is needed to compute information about entities. One approach would be to allow this interaction to take place at every time step. However this would lead to many different times at which any given piece of information could be communicated. In practice these alternatives compete, leading to ineffective learning. The remaining possibilities are communication at the time when the information is input and/or communication at the time when the information is needed in the entity's output. All these options are somewhat effective, but the latter appears to work the best, so that is the one

used in this paper. This is done with the combination layer shown above the two SRN components.

The only remaining type of interaction is between the sequences of representations for different entities. Because we do not presuppose any organization to the set of entities (such as a stack or a tape), it is sufficient for all such interaction to go through the representation of the overall context, as covered above.

SSNs are trained using the same algorithm as can be used for SRNs, Backpropagation Through Time (BPTT) [4]. BPTT works by unfolding the network into one copy per time period, and then applying standard backpropagation to the resulting feed-forward network, using weight sharing to keep the different copies of the network the same. For SSNs, within each time period's copy BPTT must also make a copy of the pulsing units for each entity. Therefore, each link to or from a pulsing unit will be duplicated once per time period and once per entity. During backpropagation training the weight for each such copy of the link must be updated in an identical fashion.

3.2 Representing structure

One of the strengths of the SSN architecture is its ability to identify entities in its output, and so output a representation of structure. To illustrate this we consider a SSN computing the syntactic structure for the sentence "John loves a woman", as shown in figure 2. Each constituent in the syntactic structure is represented as an entity, and the structure is output as a set of relationships between these entities. The pattern of input-output discussed here is used in the experiments reported in the next section.

Firstly, we must define what is to be input to the entity and gestalt components of the SSN. In language processing, every word is relevant to the overall context. Therefore, every word is input to the gestalt component of the SSN, one word per time period, as is done with a standard SRN. Further, each word may introduce a new syntactic constituent. Therefore each word is also input to the entity component of the SSN in a previously unused phase.

The syntactic structure for a sentence can be specified incrementally using three output units, *Grandparent*, *Parent* and *Sibling*. This output format is illustrated in figure 2 as a sequence of pieces of structure. When accumulated together these pieces completely specify the entire syntactic structure, as shown at the bottom of figure 2.

There are two cases to this output format. If the current word's constituent has already been introduced by another word, then the *Parent* output identifies that constituent from those that have been previously introduced, as shown for "woman" in figure 2. Otherwise the *Parent* output identifies the constituent that was introduced with the current word. In this case the structural position of this parent constituent must also be identified. If the parent constituent is part of a constituent that has already been introduced, then the *Grandparent* output unit identifies that constituent during the current time period, as shown for "a". If the parent constituent is part of a constituent that is introduced later, then the *Sibling* output unit will identify the current parent constituent during the time period when the later constituent is introduced. This is shown in the period for "loves", which identifies the parent of "John" as the *Sibling*.

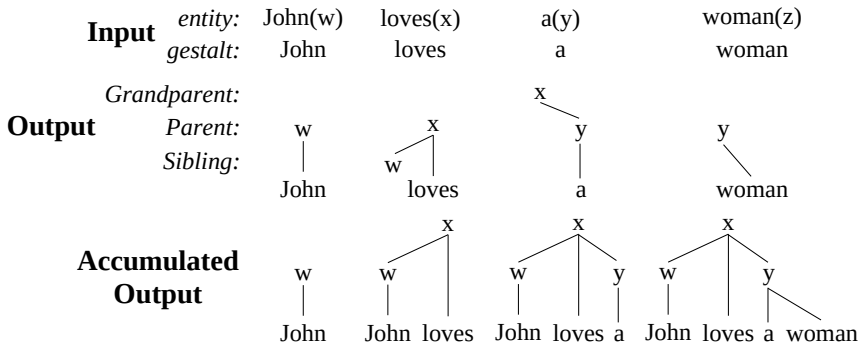

Figure 2: The input and output information for "John loves a woman". Phases are shown as variables (w,x,y, and z).

4 Learning to parse natural language

Although connectionist networks have been applied to tasks involving language learning in the past, there has been no convincing application to learning to parse naturally occurring sentences. We have conducted experiments in training SSNs to parse natural language using the Susanne[1] corpus as a source of preparsed sentences taken from newspaper reports. This section describes these experiments and our current results.[2]

We used the input format described above, but with part-of-speech tags as the input instead of words. For example, the sentence "John loves a woman" would be input as "NP VVZ AT NN". This reduces the overhead of training because less data is required, since there are many fewer part-of-speech tags than words. Both the entity and gestalt inputs use a localist representation of each letter within a part-of-speech tag. Since the part-of-speech tags are at most three letters long, we have three banks of input units per component. The target output is an unlabeled parse tree, represented with the output format also described above. To convert from the continuous outputs of the network to a discrete structure, we first take maximums across competing outputs, then convert the resulting structural relationships to a set of constituents.

We compare our results to the current state-of-the-art for mapping word tags to labeled parse trees, which are Probabilistic Context Free Grammars [7]. The standard evaluation of performance compares the constituents output by the model (the SSN in this case) to the constituents in the corpus, to determine the percentage of the output constituents that are correct (precision), and percentage of the correct constituents that are output (recall). Reported figures for both precision and recall are around 75% [7].

We trained a range of SSNs on a training set formed from sentences of length less than thirty words (13,523 words in total). Each network was trained until the sum-squared error reached a minimum, and results obtained on a cross-validation set. The cross-validation set consisted of 4,700 words with an average

[1]The Susanne corpus is sponsored by the Economic and Social Research Council (UK) with the University of Sussex as grantholder.

[2]These and some related experiments are discussed in more detail in [6].

(unrestricted) sentence length of 21.6 words. The best two of these networks were then selected for testing. They each had 20 units in their gestalt and entity hidden layers, but one had 10 units in its combination layer, and the other had 20. The test set consisted of 4,602 words with an average sentence length of 26.2 words. The average performance of these two networks on the training set was 71.6% precision, 75.8% recall, on the cross-validation set, 68.2% precision and 73.8% recall, and on the test set, 62.6% precision and 69.4% recall.

5 Conclusion

This paper has presented a new connectionist architecture, Simple Synchrony Networks (SSNs), which is trained using an extension of Backpropagation Through Time. SSNs combine the characteristics of Simple Recurrent Networks to learn about patterns across time with the characteristics of Temporal Synchrony Variable Binding to represent entities. We demonstrate the ability of SSNs to represent structure and generalise learned information across entities with experiments training SSNs to parse natural language sentences. We conclude that the SSN is a simple but significant new architecture extending the impressive generalisation abilities of connectionist networks in pattern matching tasks to the more complex domains typical of higher level cognition.

References

[1] L Shastri and V Ajjanagadde. From simple associations to systematic reasoning: A connectionist representation of rules, variables, and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences*, 16:417–494, 1993.

[2] J Henderson. A connectionist architecture with inherent systematicity. *Proceedings of the Eighteenth Conference of the Cognitive Science Society, La Jolla, CA*, 1996.

[3] J A Fodor and Z W Pylyshyn. Connectionism and cognitive architecture: a critical analysis. *Cognition*, 28:3–71, 1988.

[4] D E Rumelhart, G E Hinton, and R J Williams. Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland, (eds.), *Parallel Distributed Processing, Vol 1.* MIT Press, Cambridge, MA., 1986.

[5] J L Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[6] J Henderson and P Lane. A connectionist architecture for learning to parse. *Proceedings of the Association of Computational Linguistics*, 1998.

[7] E Charniak. Statistical Techniques for Natural Language Parsing. *AI Magazine*, forthcoming.