5 Recurrent neural networks and natural language processing

HERMANN MOISL

5.1 Introduction

One of the many application areas of Artificial Neural Networks (ANNs) has been natural language processing (NLP), and various ANN approaches to language processing have been proposed (see Sharkey & Reilly (1992) for an overview and recent work). This paper concentrates on one which has been prominent in recent research: induction of language processing devices in synchronous recurrent distributed ANNs by exposure to symbol string environments (Castano et al. 1993; Elman 1990, 1991; Giles et al. 1990, 1991, 1992; Giles & Omlin 1993; Pollack 1991; Sanfeliu & Alquezar 1992; Servan-Schreiber et al. 1989, 1991; Sharkey & Sharkey 1993; Watrous & Kuhn 1992). Such nets have yielded promising results in language processing and in NLP more particularly, but there appears to be a problem in principle which renders them inadequate for general NLP. In terms of their architecture and string-processing dynamics these nets look very like strict finite-state automata (FSAs), where "strict" refers to FSAs in which there is no distinction between processor and memory (on which see, further, Schwarz 1992).

But one of the earliest and most enduring results in generative linguistics is that finite-state devices are inadequate for generating natural languages, with the consequence that the nets in question appear to be inadequate on theoretical grounds for NLP. The general argument of this paper is that the theoretical problem is only apparent, and that the adequacy of recurrent distributed nets for NLP is a purely empirical issue. The discussion is in two main parts. The first considers and rejects standard arguments against finite-state NLP, and the second sketches a general approach to finite-state NLP using one particular kind of recurrent ANN: the simple recurrent network (SRN).

5.2 Finite-state processing in NLP

NLP research has been and continues to be naturally associated with other disciplines that concern themselves with the study of human language: theoretical linguistics most obviously, but also the range of disciplines that come under the umbrella of cognitive science, for which theoretical characterization of natural language is widely seen as paradigmatic for understanding of cognition more generally. The argument of this section is based on the assumption that the aims of NLP and of these associated disciplines are fundamentally different. Mainstream linguistics and cognitive science regard natural language as an abstract object, and aim to explain it by developing maximally expressive and economical theories about it. NLP, on the other hand, is concerned with the design and construction of physical devices to process physical strings for some purpose. On the basis of this assumption, the current section argues that the objections in principle to finite-state NLP stem from a failure to keep these aims separate.

The more or less standard approach in non-ANN NLP, exemplified in Gazdar & Mellish (1989), is to implement a physical device on the basis of some preferred linguistic theory. That is, the grammatical categories, syntactic structures, and any structure-manipulating devices such as movement rules of the linguistic theory are incorporated into a language-processing algorithm, and that algorithm can then either be directly physically realized or simulated on a general-purpose computer. The result is a physical device whose behaviour is both completely specified by the linguistic theory and as fully understood as the theory itself. Moreover, as a purely practical matter, when simulation rather than direct realization is chosen – and it invariably is – the construction of the NLP device becomes no more difficult than design and coding of the algorithm: a compiler takes care of the rest. These are considerable advantages, but the fact remains that direct instantiation of linguistic theories is not a necessary approach to construction of NLP devices.

Given a physical system with an observable behaviour, what internal mechanisms produce that behaviour? This is the identification problem, and the answer, in Arbib's words, is this:

Even if we know completely the function, or behavior, of a device, we cannot deduce from this a unique structural description. . . . The process of going from the behavior of a system to its structural description is then not to be thought of as actually identifying the particular state variable form of *the* system under study. It is, rather, that of identifying a state variable description of *a* system that will yield the observed behavior, even though the mechanism for generating that behavior may be different from that of the observed system. (Arbib 1987: 38–9, emphasis original).

True, theoretical linguistics is concerned with natural language in the abstract, not with the language behaviour of individual systems/humans, but the identification problem nevertheless applies in that a class of systems is at issue: linguistic theory is a structural description of the class of natural language speakers.

This applies straightforwardly to the present discussion. Assume some function, say, a mapping from sentences to meanings. An NLP device realizes the function by pairing physical sentence representations with physical meaning representations. Such a device would be describable in terms of a processor which uses linguistic-theoretic ontology even though it did not physically instantiate that ontology but rather used some other mechanism. In other words, linguistic theory has no necessary implications for the design of NLP processors. In principle, then, the way is open for candidate NLP technologies, including devices with finite-state architecture such as recurrent ANNs. There are, however, some standard objections to finite-state architectures for NLP, and these will now be dealt with in turn.

5.2.1 Unbounded-length centre-embedding strings

One of the aims of syntactic theory is to characterize the range of natural language sentence structures as economically as possible without any necessary regard as to how those characterizations might relate to production and understanding of utterances by speakers in the real world. Some of these structures are characterized as recursive; because it imposes no physical realization constraints, syntactic theory permits an arbitrary depth of recursive embedding and consequently sentences of unbounded length, which renders the language in question an infinite string set. Now, Chomsky (1956) long ago demonstrated that finite-state devices are incapable of generating or processing the language $a^n b^n$, that is, a set of strings in which some unbounded number of a given symbol a is followed by exactly the same number of some other symbol b, where n is any positive integer. Since this string pattern is attested in natural languages in what are analysed as recursive centre-embedding structures, and since generative grammars do not specify limits on recursive structures, any NLP device has to be able to deal with unbounded centre-embedded strings. But this is impossible if the device is finite-state, and so finite-state devices are inadequate for NLP.

If, however, one is concerned not with abstract characterization of natural language but with constructing a physical NLP device, it is only necessary to consider the string set which the device can be expected to encounter in practice rather than the set it might encounter in principle. On that view, arguments against the finite-stateness of natural languages simply do not apply. In the real world there is no such thing as an arbitrarily deeply nested recursive structure, and no such thing as a string of unbounded length. These things have finite limits, and, seen in terms of what infinity is conventionally taken to mean, even the longest natural language strings are really very short. The difference between the theoretical linguist's and the NLP researcher's views of natural language comes down to this. A grammar with recursion generates an infinite string set as long as there is no bound on the application of that recursion. As soon as a bound is imposed the language which the grammar generates becomes a finite subset of the one generated by the unbounded version, which can be processed by a finite-state device (Hopcroft & Ullman 1979). There is consequently no reason why a finite-state device should not process any member of the class of natural languages.

None of this is new. Miller & Chomsky (1965: 464–83) recognized that the human language processor had to be finite-state, and the inevitable finite-stateness of physical NLP devices has since been generally accepted though rarely mentioned and sometimes, it seems, forgotten. Despite this, strict finite-state architecture has never, to my knowledge, been seriously entertained for NLP. The reasons for this have to do with the second and third objections, to which we now turn.

5.2.2 Capturing generalizations

Strict finite-state architecture differs from that of higher-order automata in automata theory in that the the higher-order ones have a processor–memory distinction (Schwarz 1992). If a bound is placed on the memory of such a higher-order device, it becomes functionally finite-state, that is, it can be simulated in terms of input–output behaviour by a strict finite-state machine. The only difference is that the two sorts of device use different algorithms to compute any given function on account of their different architectures. Advocates of finite-state NLP have preferred higher-order, bounded-memory architectures to the strict finite-state one explicitly on account of the former's explanatory advantage in relation to generative linguistic theory (Chomsky 1956; Miller & Chomsky 1965; Church 1980; Pulman 1986). If one's aim is explanation, then clearly a higher-order automaton with bounded memory is to be preferred. But the primary aim of NLP is construction of physical devices, not explanation, so the choice between higher-order, bounded-memory and strict finite-state architecture is in principle neutral.

5.2.3 Compositionality

Semantic theory in theoretical linguistics aims to associate linguistic expressions with meanings, given some definition of "meaning". This involves at least:

- assignation of meaning to the primitive expressions – the morphemes – of a given language;

- specification of how the primitive expressions relate to composite expressions – phrases and sentences – in that language;
- specification of how linguistic meaning relates to the world.

There are various approaches to these tasks; the one used for exemplification here, formal or Montague semantics (Cann 1993), has been and continues to be influential. Formal semantics defines the meaning of a sentence as its truth conditions: a sentence means what the world would have to be like for the sentence to be true. Truth conditions are defined relative to a "universe of discourse" which the semantic theory models. Such a model has two main parts:

(a) The entities of the universe of discourse are identified, and the relationships between these entities and the primitive expressions of the language, that is, the denotations of the primitive expressions, are defined; and
(b) How the denotations of composite expressions are constructed from those of primitive expressions is specified.

Part (b) is based on Frege's principle of compositionality, which says that the meaning of a composite expression is a function of the meanings of its component primitives and their manner of combination, and is defined using a grammar which both generates the sentences of the language in question, and associates a constituent structure with each sentence. The connection between syntactic structure and meaning is made by the rule-to-rule hypothesis, in which each syntactic rule is associated with a corresponding semantic rule which specifies the meaning of a composite expression in terms of its immediate syntactic constituents. In this way, syntax can be said to drive semantics in the sense that, given the meanings of the morphemes of a language, syntactic structure determines what sentences mean.

Compositionality has also been at the centre of a long-running debate in the cognitive science community, where the rival claims of an established "classical" approach to the study of cognition and those of the "connectionist" challenger are at stake (for recent discussion and extensive references see Dinsmore (1992) and Volume 4 of *Connection Science*). The classical position, forcefully put by Fodor & Pylyshyn (1988), is that, in order adequately to explain certain fundamental aspects of cognition, one requires the notion of the structured representation, that is, of a mental object consisting of primitive symbols arranged in a constituent structure with compositional semantics, and the notion of mental processes which interpret representations in a way that is sensitive to their structure; because they are by nature finite-state devices, ANNs cannot articulate or process representations with a constituent structure adequate for capturing the requisite generalizations about cognition. In response, connectionist cognitive scientists have worked to vindicate ANNs as a suitable alternative paradigm for cognitive theorizing by developing specifically connectionist accounts of compositionality, chief among them Smolensky's (1990) tensor product representation and van Gelder's (1990) functional compositionality, which is based on temporal rather than spatial structuring of constituents. Functional compositionality is exemplified in Pollack's (1990) RAAM architecture, and applied to NLP by Chalmers (1990) and Blank et al. (1992).

Central to both theoretical linguistics and cognitive science, then, is the idea of semantic interpretation of an expression on the basis of its syntactic structure. Now, it is universally agreed that natural language sentences have complex and varied structures. How, therefore, can a finite-state architecture which imposes a single, strictly sequential syntax on sentences be adequate for natural language semantics? The classicists in the cognitive science debate think that it cannot, and the connectionsts agree in that they found it necessary to develop compositional representational methods for ANNs. Nevertheless, I claim that compositionality has no necessary implications for NLP. The basis for this claim is, again, that explanation is one thing, and construction of physical devices another. Compositional semantics in natural language sentences depends crucially on the attribution of more or less complex syntactic structure to sentences. But sentences are abstract objects, and so are the structures attributed to them: both are artefacts of linguistic theory. Physical strings, whether spoken or written, have no structure apart from strict temporal or spatial sequence. One approach to implementing a sentence-to-meaning function is to map strings onto abstract sentences with abstract structures and to process these sentences in accordance with those structures, which yields a processor that is perspicuously related to linguistic theory as well as to structured meaning representations, but the identification problem says that this is not necessary. Since, for NLP, it suffices that the sentence-to-meaning function be realized by a device which pairs strings with physical meaning representations, another legitimate approach is to attempt to process strings in accordance with the structure we know for certain they have – strict sequence – using a processor with strict finite-state architecture.

5.3 Recurrent ANNs and NLP

Assuming the validity of the arguments in the preceding part of the discussion, the way is clear for development of an approach to NLP based on strict finite-state architecture. But, as the Celtic chieftain Calgacus is reputed to have said of the Roman invasion of Britain: "They made a desert and called it peace" (Tacitus, *Agricola*). Notions of complex phrase structure and associated compositional semantics provide intuitively accessible and theoretically well-developed ways of thinking about language and offer a basis for the design of NLP devices via the relationship between linguistics and computation which formal language and automata theory define. If one dispenses with these notions in NLP design, what will replace them? This section sketches a proposal for an alternative.

It is best to be clear at the outset that generative linguistic theory is not being challenged: for present purposes it is accepted as a characterization of what has traditionally been called "human linguistic competence". The issue is the relationship between linguistic theory and the physical processing mechanisms required for

NLP, and the argument is, in essence, that physical instantiation of computational architectures with the processor–memory distinction (Schwarz 1992) is unnecessary.

Generative linguistic theory defines a function from linguistic expressions – words, phrases, sentences – to meanings, given some definition of "meaning". An NLP device implements this function if, given a physical representation of an expression, it returns a physical representation of the associated meaning. Since natural languages are finite sets for NLP purposes, it becomes possible to define the function as a list of <expression, meaning> pairs, and for an NLP device to do nothing more than table lookup on a physical representation of the list. The proposal is to train a recurrent ANN with strict finite-state architecture to implement such a table lookup device. The discussion is in three parts: Section 5.3.1 designs a computer simulation of an ANN to associate strings with meaning representations; Section 5.3.2 presents test results and analysis of that ANN; and 5.3.3 briefly addresses some issues which arise from Sections 5.3.1 and 5.3.2.

5.3.1 Network design and implementation

The aim is to implement a mapping from linguistic expressions to meanings, given a finite set of expressions. This requires pairing of each expression of length 1 with a meaning, each expression of length 2 with a meaning, and so on up to some maximum length, as in Figure 5.1. Here, meanings are labelled "1", "2", "3", . . . ; the expression *the* means "1", *the man* means "2", *the man in* means "3", and so on.

Meanings

	1	2	3	4	5	6
	the					
	the	man				
Strings	the	man	in			
	the	man	in	the		
	the	man	in	the	tent	
	the	man	in	the	tent	slept

Figure 5.1 *String-to-meaning mapping.*

In view of the number of words in English and of the possible combinations of words up to some reasonable maximum sentence length, this approach may appear to require a very large number of meanings, but in fact there are exactly as many as that which generative linguistic theory would posit for the same expression set. Linguistic theory would generate the meanings more elegantly than an explicit listing by building them out of the meaning primitives assigned to morphemes, but elegance is not an issue here.

Like much of the grammatical induction work mentioned in Section 5.1, the net used to implement this mapping is a simple recurrent network (SRN), a discrete-time dynamical system whose architecture and processing dynamics make it straightforwardly interpretable as a finite-state automaton (for example, Arbib 1987: 24–6), as in Figure 5.2.

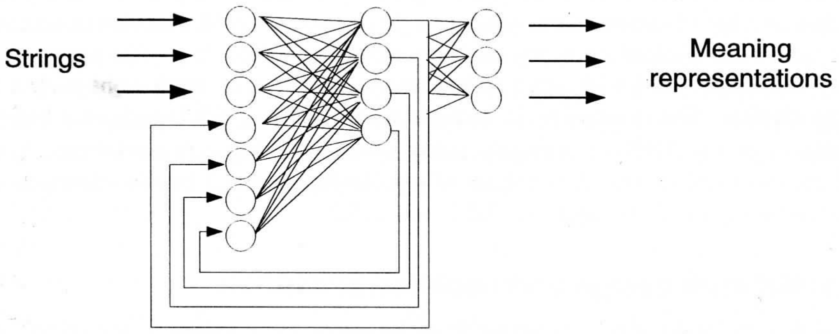

Figure 5.2 *SRN implementation of string-to-meaning mapping.*

The set of hidden-layer configurations is the state set. The connections between the input and hidden layers are the next-state function in that, for every combination of current input and current state, they generate a characteristic associated next state in the hidden layer. The connections between hidden and output layers are the output function in that, for every state of the hidden layer, they generate a characteristic output in the output layer.

The SRN is in principle a physical device with physical input and output signals, but as with virtually all ANN research practice, it is simulated on a conventional computer. The linguistic expressions and meanings, which are in principle to be represented physically, in fact need to be given a representation appropriate to the simulation. This means numerical vectors whose components represent features of the physical input and output signals. The frequently used "one-hot" encoding is adopted here, where each distinct input and output from the net is assigned a unique binary-valued vector in which one component is "1" and all the rest are "0". This is unrealistic for actual NLP work, where one would want to represent features of acoustic or visual input in some detail, but it suffices for present purposes. The result is a list of <binary-valued vector sequence, binary-valued vector> pairs which represent the linguistic <expression, meaning> pairs of the mapping.

The SRN is trained by repeated random selection of a pair from the list, and presentation of that pair to the net so that it can learn to associate the components using back-propagation; this continues until the learning error curve stops decreasing significantly. For example, training the net on the list in Figure 5.1 would proceed as follows:

1. Choose a pair, say <*the*, 1>: the vector representation of *the* is presented to the designated units in the first layer of the net, the vector representation of "1" becomes the target output, and back-propagation is applied.
2. Choose another, say <*the man*, 2>: the vector representation of *the* becomes the input and the vector representation of "1" the target output, then the vector representation of *man* becomes the input and the vector representation of "2" the target output, applying back-propagation in each case.
3. Similarly, for <*the man in*, 3>, *the* is associated with "1", *man* is associated with "2", and *in* with "3".

In this way, the net learns a mapping from strings to meanings: in all the Figure 5.1 strings *the* means "1", in strings 2–6 *the man* means "2", in strings 3–6 *the man in* means "3", and so on up to *the man in the tent slept*, which means "6".

5.3.2 Results and analysis

Results

The simulation was trained and tested on sets of up to 24 strings of maximum length 12, all of them of the declarative variety, such as (1), and learned the string–meaning function perfectly.

(1) The cat with the long tail sat on the mat.

Special attention was paid to the net's ability to handle long-distance dependencies, since this is always an issue in NLP research and has often – though erroneously – been regarded as a problem for finite-state processors. The net had no difficulty with maintaining dependencies across the distances so far required of it, (2)–(3), or the centre-embedding (4).

(2) The man in the boat by the shore sees.

(3) The men in the boat by the shore see.

(4) The car the man the woman loves drove stopped.

Analysis

For an SRN with *n* units in its hidden layer, the values which those units assume at any processing step constitute an *n*-component vector. Each such vector defines a point in *n*-space. For a test string of length *a* there are *a* hidden-layer configurations, and for *b* test strings there are *c* configurations as given in (5).

$$(5)\quad c = \sum_{1\ldots b} \sum_{1\ldots a}$$

In the grammatical inference research mentioned in Section 5.1, it is usual to carry out a cluster analysis of these *c* vectors in order to gain some insight into network operation. These analyses have shown that the points are not in general haphazardly distributed in n-space, but cluster in regions of that space such that all the hidden-layer vectors which generate a given output are adjacent. Such clusters are usually interpreted as states of a finite-state machine which the net has inferred from string input, but this is an unnecessary abstraction. The component vectors of a cluster typically differ from one another to greater or lesser degrees, and an alternative interpretation, adopted here, is to assign a separate state to each distinct hidden-layer configuration. This results in a much larger finite-state machine for a given function, but corresponds more directly to physical reality.

The hidden-layer configurations which the trained net assumed in the course of string processing in the results reported above were subjected to cluster analysis, and the results were unsurprising: the hidden-layer vectors clustered in accordance with the target output, as above. Assuming, however, a separate state for every distinct vector, the analysis shows that there is a unique state for every <expression, meaning> pair. Thus, for the three strings (6)–(8), the state sequences for initial *the* and for *man* are identical in all three cases; thereafter (6) and (7)–(8) bifurcate, and (7)–(8) continue with identical state sequences for *in the restaurant ate*, at which point they too bifurcate; *ate his lunch* in (6) and (7), though lexically identical, have different state sequences; there are three distinct states for *lunch*.

(6) The man ate his lunch.

(7) The man in the restaurant ate his lunch.

(8) The man in the restaurant ate a sparing lunch.

By following the trajectory which each string generates in the state space, one can see that the net learns a distinct state sequence for every distinct string.

5.3.3 Discussion

Whether or not the approach proposed in Section 5.3.1 will be practicable, and what sorts of development are required, are empirical matters. At least two issues arise at this stage, however, both of them having to do with the assumed prior existence of an <expression, meaning> list and of a corresponding physical representation of it:

(a) Though theoretically possible on the grounds of finiteness, construction of an <expression, meaning> list for some natural language would be a huge and almost certainly infeasible task.

(b) Linguistic expressions and meanings are abstractions, and compilation of a list implies some physical representation scheme. Representation of linguistic expressions is well understood, but it is not at all clear what a physical representation of a meaning might look like; in the foregoing discussion "meaning representations" were simply variables instantiated by arbitrary vectors for the sake of argument.

Despite its theoretical validity, the architecture proposed in Section 5.3.1 emerges as impractical in the light of (a) and (b). The following development of that architecture is intended to address this. An increasingly favoured idea in connectionist research is that ANN-based systems be "hooked up" to the world with a range of sensors which transduce physical signals into a format amenable to ANN processing, thus providing input representations which relate to environmental regularities in a systematic way and thereby, it is hoped, allowing nets to behave in ways interpretable as semantically coherent (e.g. Peschl 1992; Pfeifer & Verschure 1992; Plunkett et al. 1992). The development of the NLP approach being proposed here is based on this idea (Fig. 5.3).

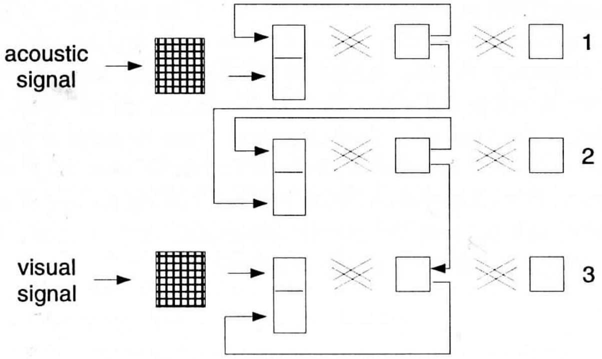

Figure 5.3 *Interconnected finite-state automata.*

The acoustic and visual components are the input sensors. Input to both is via Kohonen nets (represented by grids) which function as transducers from physical signals to 2-D topographic maps. In the acoustic component the map is input to an SRN which processes sequences of maps corresponding to time-sliced acoustic inputs. In the visual component the map of some visually perceived state of the world is input to a feed-forward ANN; this model takes no account of real-world dynamics at present, and an SRN is consequently not required. Between these two components is a feed-forward net ("link net") whose input layer is the hidden layer of the acoustic sequence processor, whose target output is the hidden layer of the visual processor, and whose hidden layer represents the association of the two. Training assumes sources of acoustic and visual input such that the two sorts of input would be perceived by a human observer as naturally related: (9) would be correlated with a state of the world in which a cat is sitting on a mat, and not with a crane lifting bricks.

(9) The cat sat on the mat.

This addresses (a) and (b) above as follows:

(a) Given an appropriate physical environment, the net learns to implement the expression-to-meaning function on-line and incrementally. There is no need to predefine an <expression, meaning> list.

(b) The net generates its own representations: expression representations are abstracted from acoustic input in the hidden layer of the acoustic SRN, and meaning representations develop in the link-ANN hidden layer in the course of training; the intention is that, at any stage of acoustic input, the link-net hidden layer should represent the meaning of the string up to that point, where "meaning" is understood in an impoverished sense appropriate to the restricted range of inputs.

5.4 Conclusion

This paper has argued that pessimistic assessments of the adequacy of recurrent ANNs for NLP on the grounds that they have a finite-state architecture are unjustified, and that their adequacy in this regard is an empirical issue. Whether or not the model can be developed for general NLP remains to be seen, but even as it stands it exemplifies the kind of radical departure from linguistics-based NLP that is possible once the supposed theoretical obstacles to finite-state NLP are removed. In particular, it departs from linguistics-based NLP in making no use of any syntactic or compositional structure beyond the purely sequential, and amounts to table-lookup mapping from strings to meaning representations.

References

Arbib, M. 1987. *Brains, machines, and mathematics*, 2nd edn. New York: Springer.

Blank, D., L. Meeden & J. Marshall 1992. Exploring the symbolic/subsymbolic continuum: a case study of RAAM. In *The symbolic and connectionist paradigms: closing the gap*, J. Dinsmore (ed.), 113–48. Hillsdale, NJ: Lawrence Erlbaum.

Cann, R. 1993. *Formal semantics: an introduction.* Cambridge: Cambridge University Press.

Castano, M., E. Vidal & F. Casacumberta 1993. Inference of stochastic regular languages through simple recurrent networks. In Lucas (1993), 16/1–6.

Chalmers, D. 1990. Syntactic transformations on distributed representations. *Connection Science* **2**, 53–62.

Chomsky, N. 1956. Three models for the description of language. *IRE Transactions on Information Theory* **IT-2**, 109.

Church, K. 1980. *On memory limitations in natural language processing*. Technical Report LCS/TR-45, Department of Electrical Engineering and Computer Science, MIT.

Dinsmore, J. 1992. *The symbolic and connectionist paradigms: closing the gap.* Hillsdale, NJ: Lawrence Erlbaum.

Elman, J. 1990. Finding structure in time. *Cognitive Science* **14**, 179–211.

Elman, J. 1991. Distributed representation, simple recurrent networks, and grammatical structure. *Machine Learning* **7**, 195–225.

Fodor, J. & Z. Pylyshyn 1988. Connectionism and cognitive science: a critical analysis. *Cognition* **28**, 3–71.

Gazdar, G. & C. Mellish 1989. *Natural language processing in LISP*. Wokingham, England: Addison-Wesley.

Giles, C. & C. Omlin 1993. Extraction, insertion, and refinement of symbolic rules in dynamically driven recurrent neural networks. *Connection Science* **5**, 307–38.

Giles, C., G. Sun, Y. Lee & D. Chen 1990. Higher order recurrent networks and grammatical inference. In *Advances in neural infromation systems 2*, D. Touretzky (ed.), 380–7. San Mateo, Calif.: Morgan Kaufmann.

Giles, C., D. Chen, C. Miller, H. Chen, G. Sun & Y. Lee 1991. Grammatical inference using second-order recurrent neural networks. In *Proceedings of the International Joint Conference on Neural Networks, IEEE 91*, Seattle, Wash., Vol. II, 273–81.

Giles, C., C. Miller, D. Chen, G. Sun, H. Chen & Y. Lee 1992. Extracting and learning an unknown grammar with recurrent neural networks. In *Advances in neural information processing systems*, J. Moody, S. Hanson & R. Lippmann (eds), 363–71. San Mateo, Calif.: Morgan Kaufmann.

Hopcroft, J., & J. Ullman 1979. *Introduction to automata theory, languages, and computation*. Reading, Mass.: Addison-Wesley.

Lucas, S. (ed.) 1993. *Grammatical inference: theory, applications, and alternatives*. London: IEE.

Miller, G. & N. Chomsky 1965. Finitary models of language users. In *Readings in mathematical psychology II*, R. Bush, E. Galanter & D. Luce (eds), 156–71. New York: John Wiley.

Peschl, M. 1992. Construction, representation, and the embodiment of knowledge, meaning, and symbols in neural structures. *Connection Science* **4**, 327–38.

Pfeifer, R. & P. Verschure 1992. Beyond rationalism: symbols, patterns, and behaviour. *Connection Science* **4**, 313–25.

Plunkett, K., C. Sinha, M. Moeller & O. Strandsby 1992. Symbol grounding or the emergence of symbols? *Connection Science* **4**, 293–312.

Pollack, J. 1990. Recursive distributed representations. *Artificial Intelligence* **46**, 77–105.

Pollack, J. 1991. The induction of dynamical recognizers. *Machine Learning* **7**, 123–48.

Pulman, S. 1986. Grammars, parsers, and memory limitations. *Language and Cognitive Processes* **1**, 197–225.

Sanfeliu, A. & R. Alquezar 1992. Understanding neural networks for grammatical inference and recognition. In *Advances in structural and syntactic pattern recognition*, H. Bunke (ed.), 75–98. Singapore: World Scientific.

Schwarz, G. 1992. Connectionism, processing, memory. *Connection Science* **4**, 207–26.

Servan-Schreiber, D., A. Cleeremans & J. McClelland 1989. Learning sequential structure in simple recurrent networks. In *Advances in neural information processing systems 1*, D. Touretzky (ed.), 643–52. San Mateo, Calif.: Morgan Kaufmann.

Servan-Schreiber, D., A. Cleeremans & J. McClelland 1991. Graded state machines: the representation of temporal contingencies in simple recurrent networks. *Machine Learning* **7**, 161–93.

Sharkey, A. & N. Sharkey 1993. Connectionism and natural language. In Lucas (1993), 20/1–10.

Sharkey, N., & R. Reilly 1992. *Connectionist approaches to natural language processing*. Hillsdale, NJ: Lawrence Erlbaum.

Smolensky, P. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence* **46**, 159–216.

van Gelder, T. 1990. Compositionality: a connectionist variation on a classical theme. *Cognitive Science* **14**, 355–84.

Watrous, R. & G. Kuhn 1992. Induction of finite state languages using second-order recurrent networks. *Neural Computation* **4**, 406–14.