

---

# NLP Based on Artificial Neural Networks: Introduction

**HERMANN MOISL**

*University of Newcastle, Newcastle-upon-Tyne, England*

## 1. INTRODUCTION

This introduction aims to provide a context for the chapters that comprise Part 3 of the handbook. It is in two main sections. The first section introduces artificial neural network (ANN) technology, and the second gives an overview of NLP-specific issues.

The quantity of research into the theory and application of ANNs has grown very rapidly over the past two decades or so, and there is no prospect of being able to deal exhaustively either with ANN technology or with ANN-based NLP in a relatively few pages. The discussion is therefore selective. Specifically,

1. The introduction to ANN technology makes no attempt to survey the current state of knowledge about ANNs, but confines itself to presentation of fundamental ANN concepts and mathematical tools.
2. The overview of ANN-based NLP surveys the historical development of the subject and identifies important current issues. It does not, however, go into detail on specific techniques or systems; such detail is provided by the chapters that follow. What is important in any research area is to some extent subjective, but the topics chosen for inclusion are well represented in the literature.

Citation of the relevant research literature is also necessarily selective, because it is, by now, very extensive. In general, the aim has been to provide a representative sample of references on any given topic.

## 2. ANN TECHNOLOGY

There are numerous general textbooks on ANNs [Refs. 7,23,26,36,114,124,160,164, 278,281 are a representative selection]. The appearance of the *Parallel Distributed Processing* volumes [281] sparked off the current wave of interest in ANNs, and they are still an excellent introduction to the subject; there is also a clear and very accessible account [[60]; see Chap. 3]. Further information on what follows is available from these and other textbooks.

The quintessential neural network is the biological brain. As the name indicates, an artificial neural network is a man-made device that emulates the physical structure and dynamics of biological brains to some degree of approximation. The closeness of the approximation generally reflects the designer's motivations—a computational neuroscientist aiming to model some aspect of brain structure or behaviour will, for example, strive for greater biological fidelity than an engineer looking for maximum computational efficiency—but all ANNs have at least this much in common:

- Structurally, they consist of more or less numerous interconnected artificial neurons, also called “nodes” or “units.”
- They communicate with an environment by means of designated input and output units.
- Signals received at the input units are propagated to the remaining units through the interconnections and reappear, usually transformed in some way, at the output units.

Figure 1 shows a generic ANN; here the circles represent the units, and the arrows directional connections between units. The generic ANN is rarely if ever used in practice, but numerous variations on it have been developed; more is said about this later in the chapter.

Many researchers argue that, although biological brain structure need not, and in practice usually does not, constrain ANN architecture except in the most general terms just listed, what is known of brain structure should inform development and application of ANN architectures, for two main reasons. Firstly, it can be productive of new ideas for ANN design, and secondly, because the human brain is the only device known to be capable of implementing cognitive functions, it is sensible to emulate it as closely as possible [1E 9, 218]. The case for this is unanswerable, but in view of the space limitations of this handbook there is simply no room for anything beyond a brief synopsis of ANN technology. (The reader who wants an accessible path into the (voluminous) literature on brain science is referred to [9] and [60].)

### A. Architecture

Here, ANN architecture is taken to mean the combination of its topology, that is, the physical layout and behavioural characteristics of its components, and its learning mechanism. Topology and learning are discussed separately.

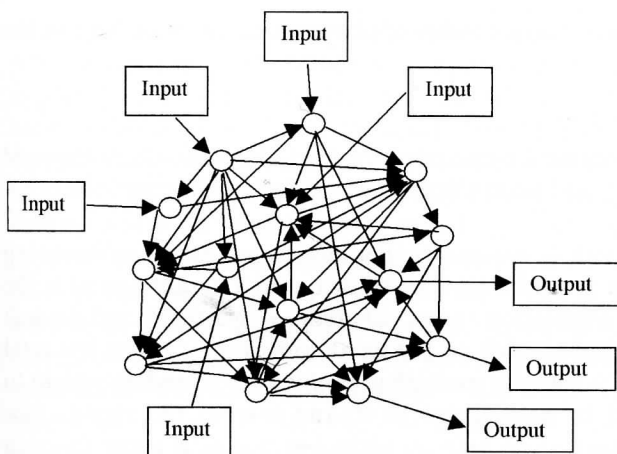


Fig. 1 A generic ANN.

### 1. Topology

This subsection develops the generic net mentioned earlier. It first of all looks more closely at the generic net, and then considers restrictions on its topology and the effects of these restrictions on network behaviour.

#### a. Generic ANN Characteristics

- The generic net is a physical device.
- It consists of units and interconnections among units.
- The units are partitioned into three types: input units that receive signals from an environment, output units that make signals available to an environment, and “hidden” units that are internal to the net and not directly accessible from outside.
- Each unit has at least one and typically many connections through which it receives signals. The aggregate of these signals at any time  $t$  elicits a response from the unit that, in the case of input and hidden units, is propagated along all outgoing connections to other units in the net, and in the case of output units is made available to the environment.
- Units can be of different types in the sense that they may respond differently to their inputs.
- Any given connection between units may be more or less efficient in transmitting signals; this relative efficiency is referred to as “connection strength.” There is typically a significant variation in strength among the connections in a net.
- Signals applied at the input units are propagated throughout the net via connections and emerge at the output units, usually transformed in some way. The transformation of the input signals, and therefore the response of the net to environmental signals in general, is conditioned by the nature of the constituent units’ response to incoming signals, the pattern of interconnection among units together with the strengths of those connections, and

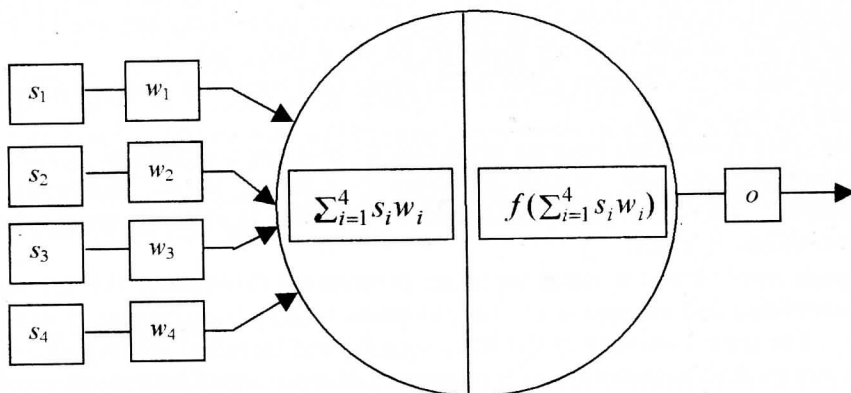
the processing dynamics; more is said of processing dynamics later in the chapter.

*b. Restrictions on the Generic Net*

As noted, the generic net is never used in practice. Instead, restrictions are imposed on aspects of network topology, and these restrictions crucially affect network behaviour:

**UNIT RESPONSE.** A single unit in the generic net has some number of incoming connections. Designate the unit as  $n$ , the number of incoming connections as  $k$ , the strength of any given incoming connection  $i$  as  $w_i$  (where  $w$  = "weight"), and the unit output as  $o$ . Signals  $s_i$  are applied to each of the  $w_i$  at some time  $t$ . Then the total input to  $n$  is  $\sum_1^k s_i w_i$ ; that is, each signal is multiplied by the corresponding weight to determine how strongly it will be propagated to  $n$ , and  $n$  sums the signals thus weighted. This sum is then transformed into an output signal  $o$  as some function of the inputs:  $o = f(\sum_1^k s_i w_i)$ , as shown in Fig. 2. The output function  $f$  may be linear, so that  $\sum_1^k s_i w_i$  is increased or decreased by some constant amount or simply output unaltered; a net consisting of units with such a linear output function can implement only a restricted range of input-output behaviours. Alternatively,  $f$  may be nonlinear. There is an arbitrary number of possible nonlinear functions that might be applied, but in practice the nonlinearity is restricted to the binary step function and approximations of it. The binary step function says that if  $\sum_1^k s_i w_i$  exceeds some specified threshold the unit outputs a signal, usually represented as 1, and if it does not exceed the threshold it fails to respond, where failure is typically represented as 0. Continuous approximations of the binary step function such as the frequently used logistic sigmoid  $f(x) = 1/1 + e^{-x}$  (where  $x = \sum_1^k s_i w_i$ ) are S-shaped. Here, the unit response to  $\sum_1^k s_i w_i$  is not restricted to binary values, but can take real values in some interval, most often 0..1 or -1..1.

**NUMBER OF UNITS.** The number of input and output units of an ANN is determined by the dimensionality of the input and output data:  $n$ -dimensional input data requires  $n$  input units, one for each dimension, and so also for outputs, where  $n = 1, 2, \dots$ . There is no limit in principle on  $n$ , but there are practical limits, the most important of which is that, as the dimensionality grows, network training becomes



**Fig. 2** A single ANN unit.



more and more time-consuming and, eventually, intractable for commonly used ANN architectures. It is, therefore, important to keep the dimensionality as small as possible consistent with the need to preserve essential application-specific information; various ways of preprocessing data to achieve information-preserving data dimensionality reduction have been developed (see below).

The number of hidden units required for a net to learn a given data set and then generalize acceptably well is one of the most researched issues in the theory and application of ANNs. Theoretical and empirical results have consistently shown that, as the size of the data set that the ANN is required to learn grows, so the number of hidden units must increase. However, there is still no reliable way to determine with any precision how many hidden units are required in a particular application. The traditional approach has been heuristic search: try a variety of hidden unit sizes, and use the one that works best. Several more principled and efficient approaches to determining the optimum number of hidden units have since been developed. Again, more is said about this later.

**CONNECTIVITY.** In the generic net the units are arbitrarily interconnected. Such a topology is rarely if ever used. Instead, some systematic restriction is typically imposed on connectivity. Some of the connections in the generic net can be removed, as in Fig. 3. The units can then be arranged so that all the input units form a group, all the units connected to the input units form another group, all the units connected to the second group form a third group, and so on, as shown in Fig. 4. A layered structure has emerged, based on connectivity. Such layering is a fundamental concept in ANN design, and it raises important issues.

How many layers should a net have?

In the literature, some authors count layers of units and others count layers of connections. The latter is in the ascendant, for good reason, and we adopt that convention here.

A net can have any number of layers, but for economy of design and processing efficiency the aim is to have as few as possible. The single-layer net is maximally

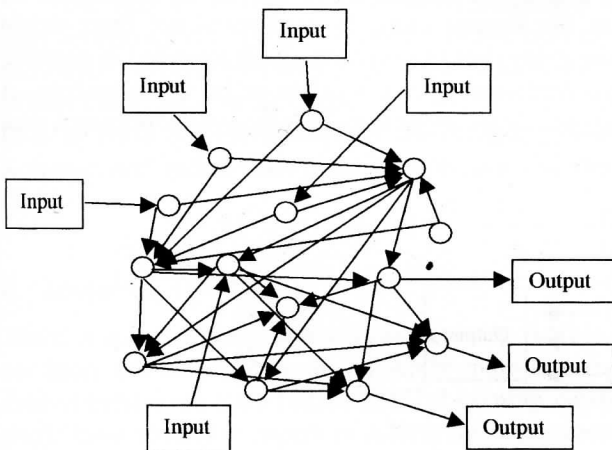
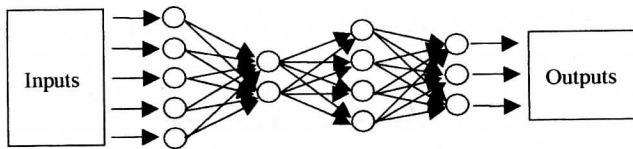


Fig. 3 Generic ANN with connections selectively removed.



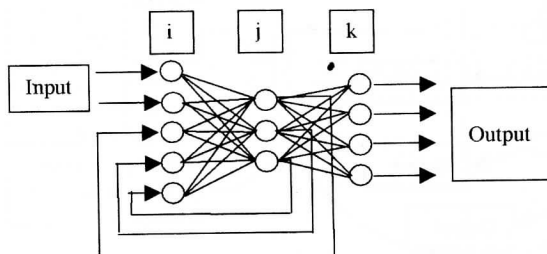
**Fig. 4** Rearrangement of Fig. 3 into a layered structure.

simple. Early ANNs were single-layered, and they are still used in certain applications, but there are input–output behaviours that no single layer net can implement [229]. Two-layer nets can overcome this restriction; in fact, for any input–output behaviour that a net with more than two layers can implement, there is always a two-layer net that implements the same behaviour. In theory, therefore, more than two layers are never required, but in practice it may be convenient or even necessary to use more than two.

What should the pattern of connectivity between layers be?

In the foregoing layered modification of the generic net, all the connections point in the same direction, so that input signals are propagated from the input units through intermediate layers of units to the output units. ANNs with this arrangement of connections are called feedforward nets. Feedback connections are also possible, however. Assume a two-layer net, that is, one with three layers of units labelled  $i, j, k$ , as depicted in Fig. 5. Here, in addition to the connections between layers  $i, j$  and  $j, k$ , there are connections from layer  $j$  back to layer  $i$ , so that outputs from the units in layer  $j$  are propagated not just to layer  $k$ , but also become input to layer  $i$ . An ANN with one or more feedback connections is called a recurrent net.

Feedforward and recurrent nets differ in how they process inputs. To see the difference, assume an encoding scheme in which each alphabetic character and each digit 0..9 is assigned a unique binary representation. Assume also a net that outputs the representation of 1 if an input string consists entirely of alphabetic characters, and the representation of 0 otherwise. Now input (the encoding of) the string *abc* to a feedforward net, as in Fig. 6. The signal is applied to the input units, and the corresponding signal emerges at the output units. In a physical net there would necessarily be a small propagation delay, but this is routinely disregarded in practice: the input–output behaviour of a feedforward net is taken to be instantaneous. A recurrent net, on the other hand, processes input sequentially over continuous time



**Fig. 5** A layered ANN with feedback connections.

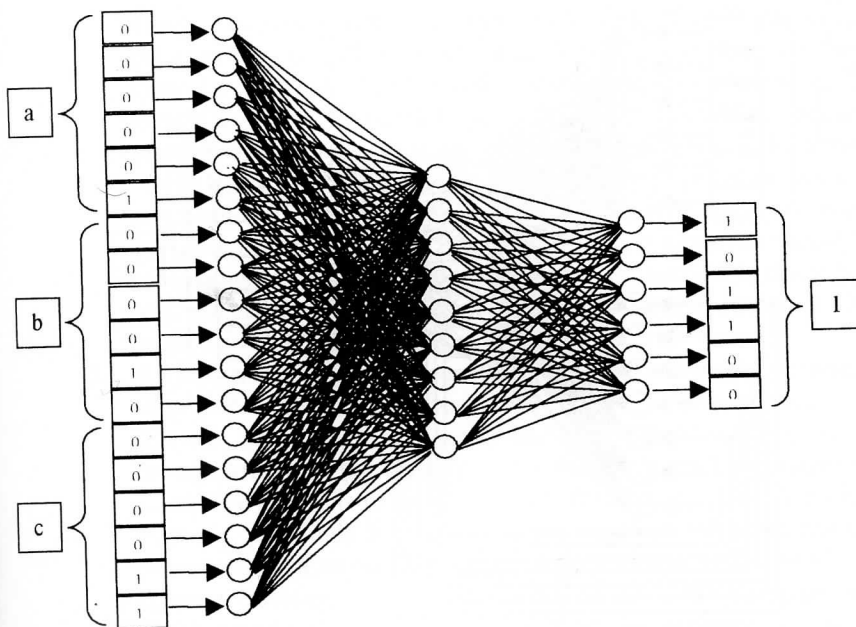


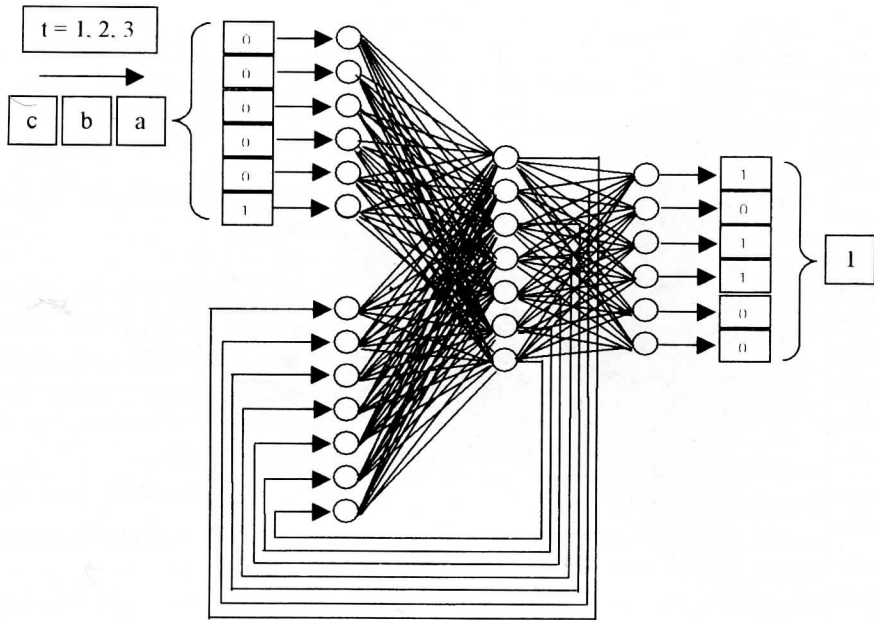
Fig. 6 A feedforward ANN for string processing.

or in discrete time steps, as in Fig. 7. Assuming discrete time at  $t_0$  input to the net is the representation of  $a$  together with the initialized outputs of the layer  $j$  units fed back by the recurrent connections. This generates output in layer  $j$  which, together with  $b$ , is input at time  $t_1$ , and so on to the end of the string. Because speech and NL text are inherently sequential, recurrent nets are important in ANN-based NLP. Figure 7 shows a frequently used feedback topology, but various others are possible. These varieties, and issues such as higher-order feedback connections, are discussed in the cited textbooks.

The two ANN topographies most frequently used in NLP work are the two-layer feedforward net, with sigmoid hidden unit activation function known as the multilayer perceptron (MLP), and the MLP with feedback connections from the hidden units back to the input, known as the simple recurrent network or SRN. Both of these are exemplified in the foregoing. Various other topologies, such as Kohonen and radial basis function nets, are also extensively used, however [see e.g., 114].

## 2. Learning

There is no necessary connection between ANNs and learning. It is possible to configure the connections in an ANN manually in such a way as to give it some desired behaviour [i.e., 121,122,246]. Learning capability is, however, probably the single most attractive aspect of ANNs for reserachers, and the overwhelming majority of ANN-based work uses it.



**Fig. 7** A recurrent ANN for string processing.

Given a sufficiently complex net, that is, one with enough hidden units, the aim of ANN learning is to find a set of connection weights that will allow the net to implement some desired behaviour. An ANN learns by adjusting its connection strengths in response to signals from an environment such that, once training is deemed to be complete by the designer, the net responds in predictable ways to signals that it encountered in the course of training, and in reasonable ways to signals that it did not encounter during training, where the designer is the judge of what is reasonable. Such learning assumes a broadly stationary environment. For example, a net that is to function as a speech processor is trained on speech signals for which the probability distribution does not change significantly over time. Once trained, the net is expected to behave reasonably in response to novel speech input from the same distribution, but not to, say, visual signals, which bear no systematic relation to speech.

ANN learning mechanisms are standardly categorized into supervised and unsupervised algorithms:

1. A *supervised learning algorithm* requires predefinition of a training set of (input, target output) signal pairs: for each input, the net must learn to output the target signal. In other words, the net is taught a prespecified input–output behaviour. The essence of supervised learning is as follows. The input component of a training pair is presented at the input units of the ANN. The signal is propagated through the net, and the response that emerges at the output units is compared with the target output component of the training pair. If the target and actual outputs correspond within tolerances, no change is made to the network weights. Otherwise, the

weights are adjusted in a way such that they decrease the difference between the actual response and the target output the next time the input in question is presented. This adjustment process continues incrementally until actual and target outputs coincide. How exactly the appropriate weight adjustments are made is the province of the various available ANN learning algorithms. By far the most popular of these is back-propagation, which was introduced in its present form by Rumelhart and McClelland in 1986 [281], since which time numerous modifications and extensions have been proposed. Back-propagation is described in numerous places, including the textbooks cited at the outset of this section; a good introductory account is given [108].

2. An *unsupervised learning algorithm* does not use target outputs, and the net thus does not learn a prespecified input–output behaviour. Rather, an ANN is presented with inputs and expected to self-organize in response to regularities in the training set without external guidance, such that the learned response is in some sense interesting or useful. A Kohonen net, for example, is a single layer feedforward topology the input units of which are connected to a two-dimensional grid of output units, as shown in Fig. 8. Given a training set of  $n$ -dimensional vectors, the Kohonen learning algorithm adjusts the connections so that the distance among the training vectors in  $n$ -dimensional space is reflected in unit activation in two-dimensional output space: once a Kohonen net is trained, each input vector is associated with a particular region of output unit activation, and the distance among regions is proportional to the distance among inputs in vector space. The result is called a topographic map. No target output is involved: the Kohonen learning algorithm allows the net to self-organize.

## B. Mathematical Modelling

The foregoing discussion has tried to give an account of ANNs as physical devices that can be trained to display interesting physical behaviours. An intuitive understanding of ANNs at this level is useful for most people, but it is not sufficient. The limitations become apparent as one works with ANNs and has to confront a range of design and analysis issues. Which ANN architecture is appropriate for a given application? How much data is required? How large does the net have to be? How

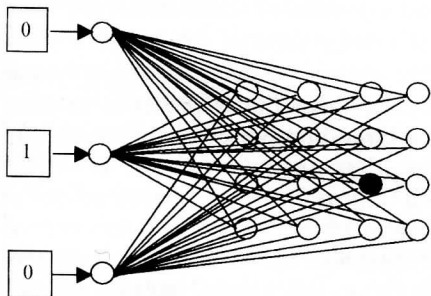


Fig. 8 A Kohonen net.

long will training take? Once trained, how can the net's behaviour be understood? One can approach these and other issues experimentally, but that is at best inefficient. The alternative is to model ANNs mathematically, and then to apply established mathematical methods to the models in addressing matters of design and analysis [103,317]. The latter approach is now standard; some essentials are outlined in what follows.

To undertake mathematical modelling, physical nets have to be represented as mathematical objects. Linear algebra is fundamental in this:

- Assemblies of input, hidden, and output units are represented as vectors the components of which take numerical values, so that the  $i$ th unit activation in an  $n$ -unit assembly corresponds to the  $i$ th element in a vector of length  $n$ .
- Connections between an  $m$ -unit and an  $n$ -unit assembly are represented as a two-dimensional  $m \times n$  matrix, and numerical matrix values represent connection strengths: the value in cell  $m_i n_j$  represents the connection strength between the  $i$ th unit in assembly  $m$  and the  $j$ th unit in assembly  $n$ .
- In the evaluation of a unit output  $o = f(\sum_1^k s_i w_i)$ , the calculation of the sum in brackets is represented by the inner product of the input vector  $s$  and the corresponding row of the weight matrix  $w$ .

There are currently three main approaches to mathematical modelling of ANNs: as dynamical systems, as computational systems, and as statistical algorithms [317]. We look briefly at each.

### 1. ANNs as Dynamical Systems

A physical system that changes over some time span is a physical dynamical system [9,38,88,140,165,166,243,244,310,317,346,361]. Dynamical systems theory is the branch of mathematics that studies the long-term behaviour of physical dynamical systems; this is done by constructing mathematical models of physical systems, and then analyzing the properties of the models.

Analysis of ANNs as dynamical systems generally uses state-space modelling, the key concepts of which are:

#### a. State Space

A state space is an  $n$ -dimensional euclidean or noneuclidean space. Assuming the former, the dimensionality of the space is determined by the number of variables in the model:  $n$  variables determine an  $n$ -dimensional space, where each orthogonal axis measures a different variable. Let  $x_1, x_2, \dots, x_n$  be variables of some model  $M$ . Then the vector of variable values  $x_1(t), x_2(t), \dots, x_n(t)$  is the state of  $M$  at time  $t$ . Now assume an initial state at time  $t_0$  and plot the state vector in state space as the system changes for  $t > 0$ . The result is a trajectory in state space that represents the state evolution of the model. If this is repeated for a large number of initial states, the result is a collection of state trajectories called a phase portrait.

#### b. Invariant Manifolds

A manifold is a  $k$ -dimensional region in  $n$ -dimensional space, where  $k$  is strictly smaller than  $n$ . In a dynamical system model, the  $n$ -dimensional volume defined by the vector of initial values collapses onto a smaller-dimensional volume or manifold as the system evolves over time. The manifold to which the system evolves from

a given starting condition is called an invariant manifold or attractor. Typically, more than one initial vector will collapse onto the same manifold; the set of vectors that evolve onto a given manifold is called the manifold's basin of attraction. Invariant manifolds thereby define subspaces of state space to which trajectories evolve, and within which they remain, for specific initial conditions.

There are various types of manifold. The simplest is the fixed-point attractor, a 0-dimensional manifold that represents a physical system that evolves to, and in the absence of disturbance remains in, a fixed and unchanging state; the classic example is a pendulum, which eventually comes to rest and remains so unless moved. Limit cycles are trajectories that loop back on themselves, and represent oscillatory motion in the physical system being modelled. In addition, more complex attractors are possible, including fractal and chaotic ones.

Invariant manifolds may or may not be stable when the system has settled down and is subsequently disturbed. If the system settles back down to the pre-disturbance attractor, the manifold is stable. The state may, however, move away from the attractor as a result of the disturbance and go to another attractor in the state space.

A general dynamical system model is a triple  $M = (S, T, f)$ , where  $S$  is a state space,  $T$  is a temporal domain, and  $f = S \times T \rightarrow S$  is a state transition function that describes how the state of the model evolves over time. Depending on how  $S$ ,  $T$ , and  $f$  are defined, dynamical models can be subcategorized in various ways.  $S$  can, for example, be a continuous or discrete state space;  $T$  can be continuous or discrete time;  $f$  can be linear or nonlinear. In addition, the model can be autonomous or nonautonomous. For the last of these, dynamical systems theory draws a distinction between modelling a physical system's intrinsic behaviour, and modelling the effect which an external environment has on the system; such environmental influences are called inputs. In an autonomous system the effects of inputs are not modelled, whereas they are for nonautonomous systems. An autonomous system is started with known initial state values and then allowed to evolve over time without interaction with an environment, that is, without input. In many applications, however, nonautonomous systems are more interesting because the physical systems being modelled are generally subject to environmental influences.

There are two sorts of ANN dynamical systems to consider: activation dynamics and weight dynamics. For activation dynamics, assume a net containing  $n$  units. These  $n$  units are the state variables of the activation dynamical system, and the state of the system at time  $t$  is the vector of unit activation values  $u_1, u_2, \dots, u_n$ . If the weights and initial conditions or input are held constant, the evolution of the unit activation vector is the activation dynamics of the net. The state space of the weight dynamics, on the other hand, is the space of weight matrices, the dimension of which is that of the number of trainable weights in the net. Given some initial weight matrix  $W$ , the weight dynamics is the evolution of  $W$  as the net is trained using a learning algorithm. The key issue in weight dynamics is convergence to a point attractor, that is, for the weights to evolve so that they stop changing, as this indicates that learning is complete. The activation dynamics of various types of ANN exhibit the range of long-term behaviours characteristic of dynamic systems, from fixed-point to chaotic. The aim in most applications is to have the activation dynamics converge to point attractors, but not invariably; for example, there is some work that exploits fractal dynamics for NLP.



## 2. ANNs as Computational Systems

Given arbitrary sets  $A$  and  $B$ , a function  $f$  is a subset  $S$  of the Cartesian product  $A \times B$  such that, for each pair  $(a, b) \in S$ ,  $a \in A$  is uniquely associated with  $b \in B$ . An algorithm is said to compute  $f$  if, given the first component  $a$  of a pair  $(a, b) \in S$ , it returns the second component  $b$ , that is, if it generates the set of pairs that constitutes  $f$ . Now, if the physical inputs and outputs of a net  $N$  are interpreted as  $A$  and  $B$ , and if  $N$ 's physical input–output behaviour is consistent with  $S$  under that interpretation, then  $N$  can be seen as an algorithm for computing  $f$ , that is, as a computational system [for discussions of computation see Ref. 60, Chap. 3, and the journal *Minds and Machines* 1994; 4:377–488].

### a. Computability

Computability in standard automata theory studies the classes of function that can be computed by various types of automata. The corresponding study of ANN computability does the same for various types of ANN topology [120,278,317]. The key results thus far for present purposes are as follows:

1. Under unboundedness assumptions analogous to those made in automata theory, certain ANN topologies have been shown to be Turing-equivalent, that is, for any function that a Turing Machine computes, there is an ANN with one of these topologies that computes the same function, and vice versa. These topologies include some of the most frequently used types of net [see survey in Ref. 285]: multilayer perceptrons with any one of a broad class of nonlinear hidden-layer activation functions, including the more or less standard sigmoid [372,373, reprinted in Ref. 374], radial basis function networks [137], and recurrent networks [308; but see Ref. 321]. Single-layer feedforward nets, on the other hand, are known not to be Turing-equivalent: there are some functions that no such net can compute [i.e., 26].
2. ANNs that are bounded in terms of the number of units or the precision of connection strengths are computationally equivalent to finite-state automata. Because all physically implemented ANNs are necessarily bounded, they are all limited to finite state computational power. This does not compromise ANNs relative to automata; however, because the same applies to the latter—every implemented Turing Machine is equivalent to a finite-state machine. In real-world applications automata are given sufficient memory resources to compute the finite subset of the required function in practice, whereas ANNs are given more units or higher-resolution connections to achieve the same end.

These results say only that, for some function  $f$ , there exists an ANN topology and a set of connections that compute  $f$ . There is no implication that the requisite set of connections can be learned with a net of tractable size, or within a reasonable time; see further discussion in what follows.

### b. Computational Complexity of Learning

Every computation has time and space resource requirements, and these vary from algorithm to algorithm. Computational complexity theory [353] studies both the relative resource requirements of different classes of algorithm for a given function, and also the rate at which resource requirements increase as the size of particular



instances of the function being computed grows, where “size” can be informally understood as the amount of input data needed to describe any particular instance of a function, or, more formally, as the number of  $n$  of bits used to encode the input data. As  $n$  increases, so do the resource requirements. The question is how quickly they increase relative to  $n$ —linearly, say, or polynomially. If the increase is too rapid, the function becomes intractable to compute within reasonable time or space bounds.

Computational complexity theory has a direct and important application to ANN learning [5,31,36,81,187,188,278,317]: the problem of scaling. Given some ANN topography, the aim of learning is to find a set of connection weights that will allow the net to compute some function of interest; the size of the problem is the number of weights or free parameters in the net. The number of weights in a net is a function of the number of input, output, and hidden units, which, as we have seen, are determined by the data-encoding scheme adopted in any given application, and the size of the data set that the net is required to learn. As the number of parameters used to encode the data or the number of data items increases, therefore, so does the number of connections. Now, empirical results have repeatedly shown that the time required to train an ANN increases rapidly with the number of network connections, or network complexity, and fairly quickly becomes impractically long. The question of how ANN learning scales with network complexity, therefore is, crucial to the development of large nets for real-world applications. For several commonly used classes of ANN, the answer is that learning is an NP-complete or intractable problem—*We cannot hope to build connectionist networks that will reliably learn simple supervised learning tasks* [187].

This result appears to bode ill for prospects of scaling ANNs to large, real-world applications. It is all very well to know that, in theory, ANNs with suitable topologies and sufficient complexity can implement any computable function, but this is of little help if it takes an impractically long time to train them. The situation is not nearly as bad as it seems, however. The intractability result is maximally general in that it holds for all data sets and all ANN topologies. However theoretically interesting, such generality is unnecessary in practice [19], and a variety of measures exist that constrain the learning problem such that intractability is either delayed or circumvented; these include the following [23,26,271]:

- Restricting the range ANN topologies used [187,278].
- Developing mechanisms for determining the optimal network complexity for any given learning problem, such as network growing and pruning algorithms [26,364].
- Biasing the net toward the data to be learned. Empirical evidence has shown that biasing network topology in this way can substantially speed up learning, or even permit data to be loaded into a fixed-size net which, without biasing, could not load the data at all [33,106]. Such biasing involves incorporation of prior knowledge about the problem domain into the net in various ways [80,122,183,245,345].
- Explicit compilation of knowledge into initial network weights [121,134,246,248,249].
- Preprocessing of inputs by feature extraction, where the features extracted reflect the designer’s knowledge of their importance relative to the problem.

Feature extraction also normally involves reduction of the number of variables used to represent the data and, thereby, the number of input units, with consequent reduction in network complexity [23,26,114].

- Incremental training, during which the net is trained with data that is simple initially, and that increases in complexity as learning proceeds [64,65,106].
- Transfer: the weights of a net  $N_1$  to be trained are initialized using weights from another net  $N_2$  that has already been successfully trained on a related problem [128,269,302,337–340].

### c. Automata Induction

From a computational point of view, the activation dynamics of a feedforward net are of no interest because, for a given initial condition or input, it always converges to the same point attractor. The dynamics of feedforward nets, therefore, are disregarded for computational purposes, and the input–output mapping is taken to be instantaneous. The dynamics of recurrent ANNs (RANNs) are on the other hand of considerable computational interest, because the state space trajectory of a RANN in response to a sequence of input signals is computationally interpretable as the state transitions of an automaton in response to an input symbol string, and as such the dynamics of a RANN in response to a set of input signal sequences is interpretable as an automaton processing a language  $L$ . If the dynamics are learned from input–output data rather than explicitly compiled into the net, moreover, the RANN, from a computational point of view, can be taken to have approximated an automaton that defines  $L$  or, equivalently, to have inferred the corresponding grammar. RANNs are therefore amenable to analysis in terms of a well-developed formal language and automata theory [39,318,342,354].

Since the mid-1980s there has been a good deal of work on the use of RANNs for grammatical inference [4,130–136,142,185,221,246–248,266,297,298,362,379,380]. The training of a Simple Recurrent Network (SRN) as a finite-state acceptor is paradigmatic: given a language  $L$  and a finite set  $T$  of pairs  $(a, b)$ , where  $a$  is a symbol string and  $b$  is a boolean that is true if  $a \in L$  and false otherwise, train an SRN to approximate a finite-state acceptor for  $L$  from a proper subset  $T'$  of  $T$ . The SRN, like various other RANN topologies used for grammatical inference, is a discrete-time, continuous-space dynamical system. To extract discrete computational states, the continuous ANN state space is partitioned into equivalence classes using, for example, statistical clustering algorithms based on vector distance, and each cluster is interpreted as a single computational state [39,132,142,250,341,362; see also Ref. 81]. Any finite state machine extracted in this way is a possible computational interpretation of the RANN, but it is not unique, because the number of states extracted depends on the granularity of the continuous space partitioning and on the partitioning algorithm used [94,95].

Grammatical inference using RANNs is not without its problems. The most important of these are the following:

1. A RANN can be interpreted as a finite-state automaton. Nevertheless, it remains a dynamical system that approximates, but does not implement, the automaton in the strict sense of implementation. For short strings the approximation is usually quite close, but as string length increases,

instabilities in the network dynamics can cause the net's behaviour to diverge from that of the abstracted machine [39,142,193–195].

2. There can be systematic dependencies between symbols that occur at different sequential positions in strings; subject–verb number agreement in English is an example. A problem with using RANNs for grammatical inference has been that, as the sequential distance between lexical items in a dependency pair increases, so does the difficulty of learning the dependency [22,23]. An SRN processes an input string  $a_1, a_2, \dots, a_n$  sequentially, and represents the lexical processing history up to any given symbol  $a_i$  in state  $S_i$ ; that is, in the configuration of hidden layer units when  $a_i$  is the current input. Assume now that there is a dependency between  $a_i$  and  $a_j$  later in the string. When processing arrives at  $a_j$ , the farther back in the sequence  $a_i$  is, the weaker its representation in  $S_j$ , until the resolution in the hidden units is insufficient to retain a memory of the net's having seen  $a_i$  and the dependency is lost. A variety of solutions have been proposed [102,168,214,248,254,291,360].
3. Theoretically, grammatical inference is an intractable problem. Gold [138] showed that even the simple class of languages, the regular languages, cannot be learned in reasonable time from a finite set of positive examples. On this basis one might assume that, whatever its successes relative to particular and usually quite small or simple languages, general RANN-based grammatical inference is hopeless. That would be an overinterpretation, however. In practice, there are measures that render grammatical inference tractable, such as provision of negative as well as positive examples, or incorporation of prior knowledge of the grammar to be inferred into the learning mechanisms [52,108,215].

Work on grammatical inference is not confined to finite-state acceptors or indeed to more general finite-state machines. Pushdown automata and Turing machines have also been inferred [78–80,327,376,380]. Chapter 29 provides a detailed discussion of grammatical inference and automata induction.

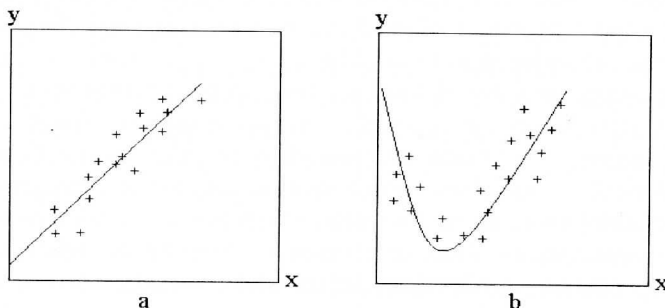
### 3. ANNs as Statistical Algorithms

Inferential statistics and ANN learning both discover regularities in data and model the processes that generate that data. It is not, therefore, surprising that ANNs are now seen as one approach to statistical modelling [6,26–28,48,186,277,278,284,317,358,375]; ANNs are in essence statistical devices for inductive inference [364]. For present purposes, the most important aspect of ANN-based statistical modelling is regression analysis. Given a set of independent variables  $x_1, x_2, \dots, x_n$ , for example the various factors affecting computer network loading, and a dependent variable  $y$ , for example, the average length of time it takes for any given user to obtain a response at some level of loading, regression analysis aims to infer the relation between independent and dependent variables from a sample data set of (independent variable, dependent variable) value pairs. The inferred relation is intended as a model for the system that generated the data: it is taken to describe the relation between independent and dependent variable values not only for the sample data set, but also for the population as a whole. For example, assuming the case in which there is only one independent and one dependent variable, data plotted on  $x, y$  axes

might look like (a) or (b) in Fig. 9. For (a), it is clear that the relation between independent and dependent variables is basically linear, and as such, the general form for a suitable model would be a linear polynomial  $y = \alpha_0 + \alpha_1 x^1$ , where the  $\alpha_i$  are constants chosen so that the line that the resulting expression describes best fits the data, and where “best” is usually defined as the line for which the sum of the squared distances from itself to each data point is a minimum. This is the regression line. The line does not pass through all or even many of the data points; there is a random scatter on either side. The relation between independent and dependent variables that the regression line and the corresponding polynomial model is thus probabilistic: the value the model predicts for  $y$  given some  $x$  not in the data set is the mean of a random scatter to either side of  $y$ , as in Fig. 9. Similarly, the shape of (b) suggests a quadratic polynomial of the general form polynomial  $y = \alpha_0 + \alpha_1 x^1 + \alpha_2 x^2$  as a suitable model, where, again, the values chosen for the parameters are such that the resulting expression describes the line of best fit through the data.

To see how ANNs can be used as regression models, we concentrate on the best-studied topology: the two-layer feedforward MLP with sigmoid activation hidden units and real-valued output. The input layer corresponds to the independent variables, the output layer to the dependent variable(s), the network weights to the parameters  $\alpha_i$ , and the training set of (input, target output) pairs to the (independent variable, dependent variable) data set. ANN learning then becomes an algorithm for determining the parameters in the expression for the regression line through the data or, in other words, a statistical method for function approximation. The resulting model takes the form not of a multivariate polynomial, but of an artificial neural network.

As noted, a conventional (i.e., non-ANN) statistical model must represent not only the relations between independent and dependent variables for the data sample on which it is based, but also for the population from which the sample was drawn. The same is true of ANN-based regression models: loading is not enough. For loading, all that is required is that a net learn a given set  $T$  of data pairs, such that, when the first element of a pair  $(a, b) \in D$  is input to the loaded net, the corresponding second element is output. But, in almost all applications, the net is expected to generalize. Given a population  $D$  of (input, target output) pairs and a proper subset  $T$  of  $D$  as a training set, then if  $T$  is successfully loaded, the expectation is that for every pair  $(a_j, b_j) \in D$  and  $\notin T$ ,  $a_j$  input to the net will output  $b_j$  within



**Fig. 9** Linear and quadratic relations between independent and dependent variables.

required tolerances. The generalization property of ANNs is much cited in the literature, but it is by no means automatic. To design a net that generalizes well, the relation between network complexity—the number of connections in the net—and the size of the training data set has to be understood.

Given a training data set, how complex does the net that learns that data need to be for adequate generalization? A net that is either too complex or not complex enough will fail to generalize adequately. To see why this is so, we consider once again a univariate data set and fit three different regression lines to it, as in Fig. 10.

The first regression line corresponds to some linear polynomial  $y = \alpha_0 + \alpha_1 x^1$ , the second to some cubic polynomial  $y = \alpha_0 + \alpha_1 x^1 + \alpha_2 x^2 + \alpha_3 x^3$  and the third to some higher-order polynomial  $y = \alpha_0 + \alpha_1 x^1 + \alpha_2 x^2 + \dots + \alpha_n x^n$ . The linear curve fits the data poorly, the cubic one fits the general trend of the data, but actually passes through only a few points, and the higher-order one gives the best fit in that it passes through each point exactly. The linear expression is clearly not a good model for the data. Surprisingly, however, neither is the higher-order one; the best of them is the cubic curve. Consider what happens for some value of  $x$  not in the sample. The cubic polynomial returns the corresponding  $y$  value on the regression curve, and it is consistent with the random scatter of data points around the curve; that is, the model has predicted a  $y$  value in response to  $x$  that is in the same distribution as the sample data. Under identical circumstances, the higher-order polynomial may return a  $y$  value that is relatively far from the sample data points, and thus does not predict a  $y$  value in the same distribution as the sample. The problem is that the higher-order polynomial has fit the sample data too exactly and thus failed to model the population well. In short, given a data set, there is an optimal number of terms in the polynomial for regression modelling. In ANN terms, this corresponds to network complexity: too complex a net overfits the data, whereas a net that is not complex enough underfits it, similar to the foregoing linear polynomial. It also goes without saying that it is not only the size of the data set that is important, but also the distribution of the samples. Even a large number of data points will not result in good generalization if, for example, they are all clustered in one isolated region of the population distribution. Training data must be chosen in such a way that it represents the population data distribution well.

The results of theoretical work on the interrelation of network complexity, data set size, and generalization capability [19,26,173,175,375] have, in practical terms,

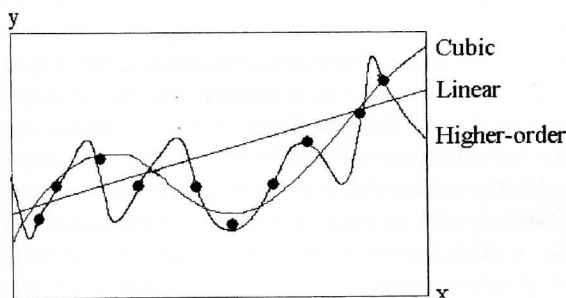


Fig. 10 Three different fits to a data distribution.

yielded a rule of thumb: good generalization is attained when the number of weights in the net is small compared with the number of data items in the training set. Experimental results, however, have shown that good generalization can be achieved with more complex nets or, equivalently, fewer data than this rule of thumb would indicate; clearly, additional factors, such as the nature of the function being approximated and the characteristics of the learning algorithm, are also involved [208,209].

In practical terms, determination of network complexity for a given data set has historically been a matter of trial and error, and to a large extent remains so: try nets of varying complexity, and use the one that generalizes best. More principled approaches have been developed, however, such as algorithms that increase or decrease the number of connections in the course of network training, with the aim of thereby arriving at an optimum topology; these and other methods are reviewed [26,271].

### 3. ANN-BASED NLP

This section is in two main parts. The first outlines the motivation for using ANN technology in NLP, and the second gives an overview of the history and current state of ANN-based NLP.

#### A. Motivation

In general, a new technology is adopted by a research community when it offers substantial advantages over what is currently available, and any associated drawbacks do not outweigh those advantages. NLP has, from the outset, been dominated by a technology based on explicit design of algorithms for computing functions of interest, henceforth called “symbolic NLP” for reasons that will emerge, and implementation of those algorithms using serial computers. It is only since the early 1980s that alternative technologies have become available, one of which is ANNs. In what follows we look at the main advantages and disadvantages of ANNs relative to symbolic NLP.

##### 1. Advantages

Various advantages of ANNs are cited in the literature [14,23,35,93,124,160,251]. The three discussed in what follows are the most frequently cited and, in the case of the first and second, arguably the most important.

###### a. Function Approximation

We have seen that ANNs can approximate any computable function as closely as desired. The function  $f$  that a given ANN approximates is determined by the parameter values, or weights, associated with the ANN, and these parameter values are learned from a data set  $D \subset f$ . In principle, therefore, NLP functions can be approximated from data using ANNs, thereby bypassing the explicit design of algorithms.

What should one want to dispense with explicit design? Looking back on several decades’ work on symbolic artificial intelligence (AI), some researchers have come to feel that a variety of problems, including some NLP ones, are too difficult to be solved by explicit algorithm design, given the current state of software technology [93,124,218,305], and have instead turned to function approximation



techniques, such as ANNs; Arbib [9], referring to the class of adaptive systems to which ANNs belong, writes: *The key motivation for using learning networks is that it may be too hard to program explicitly the behavior that one sees in a black box, but one may be able to drive a network by the actual input/output behavior of that box. . . to cause it to adapt itself into a network which approximates that given behavior* (p.20). A, and probably the, chief advantage of ANN technology for NLP, therefore, is that it offers an alternative way of implementing NLP functions that have thus far proved difficult to implement using explicit algorithm design.

It should be noted that function approximation from data is not exclusive to ANNs. There is renewed interest in non-ANN machine learning in the AI community [146,169,170,307,355,371]. The claim here is not that ANNs are the only or even the best approach to function approximation in the AI/NLP domain [206,210], but rather that they offer one possible technology for it.

#### b. Noise Tolerance

Practical NLP systems must operate in real-world environments, and real-world environments are characterized by noise, which can for present purposes be taken as the presence of probabilistic errors in the data—spelling or syntax errors, for example. A frequent criticism of symbolic AI/NLP systems is that they are brittle in the sense that noisy input which the designer has not taken into account can cause a degree of malfunction out of all proportion to the severity of the input corruption [i.e., 218]. The standard claim is that ANNs are far less brittle, so that the performance of an ANN-based system will “degrade gracefully” in some reasonable proportion to the degree of corruption.

Noise tolerance is a by-product of ANN function approximation by nonlinear regression. We have seen that ANNs approximate a function from data by fitting a regression curve to data points, and that the best approximation—the one that generalizes best—is not the curve that passes through the data points, but the one that captures the general shape of the data distribution. Most of the data points in a noisy environment will be at some distance from the regression curve; if the input corruption is not too severe, the regression model will place the corresponding output in or near the training data distribution.

Symbolic NLP systems have historically been heavily dependent on generative linguistic theory. This was and is reasonable: linguistic theory has been intensively developed for several decades, and it constitutes a substantial body of knowledge about how natural language works. But mainstream generative linguistic theory concerns itself with what has traditionally been called competence [3,63] models of the human language faculty, disregarding, as a matter of principle, the myriad contingencies of real-world linguistic usage. To the extent that it is based on generative linguistic theory, therefore, a symbolic NLP system is an intrinsically competence design that has to be supplemented with mechanisms that allow it to operate with an acceptable level of resilience in response to noisy real-world input. There is no theoretical obstacle to this: the process of adapting a competence virtual machine is a matter of software development. In practice, though, the often-cited brittleness of symbolic systems indicates that this has so far proved problematical. ANN function approximation does away with the need for this adaptation process. There is no a priori competence model, only performance data from which the net learns the required mapping. What the net learns is not an implementation of a competence

virtual machine, but rather an implementation of some (nonexplicitly specified) performance virtual machine [52].

### *c. ANNs and Biological Brains*

A frequently cited advantage of ANNs is that they are brain-like to greater or lesser degrees and, therefore, more suitable for difficult AI problems such as vision, speech, and natural language understanding than are conventional AI methods. The argument goes like this. We know that the biological brain implements functions such as vision, speech, and so on. For artificial implementation of such functions it makes practical sense to work with a processing architecture that is as close as possible to that of the brain. ANNs are closer to brains in terms of processing architectures than serial computers. Consequently, ANNs are to be preferred for AI applications. There is no theoretical justification for this, because (certain types of) ANN and the class of Turing Machines are computationally equivalent. The argument is a pragmatic one that will, presumably, be decided on empirical grounds one day. For the moment, its validity is a matter of personal judgment.

A final note. ANNs had a resurgence of popularity in the mid-1980s, and from some overenthusiastic claims made at the time one might have thought that they were a philosopher's stone for the problems that beset a wide range of computationally oriented disciplines. Since then, it has become increasingly clear that they are not. The observation that the class of ANNs can implement the computable functions directly parallels the one that Turing Machines are capable of the same thing: it is reassuring that the solution to any given (computable) problem exists, but that does not help one find it. For Turing Machine-based computation, this is a matter of identifying an appropriate algorithm. For ANNs it is a matter of training. But, as we saw earlier, successful function approximation and consequent noise tolerance involves appropriate choice of data and network architecture. The function approximation and noise tolerance here cited as technological advantages are not automatic properties of all ANNs in all applications. Rather, they must be attained by theoretically informed design and experiment.

## *2. Disadvantages*

Two of the main problems associated with ANNs, scaling and generalization, have already been discussed, together with possible solutions. To these must be added:

### *a. Inscrutability*

A network that has loaded training data and generalizes well realizes a desired behaviour, but it is not immediately obvious how it does so; the set of connection strengths determine the behaviour, but direct examination of them does not tell one very much [16]. Because of this, ANNs acquired a reputation as "black box" solutions soon after their resurgence in the early-mid-1980s, and have consequently been viewed with some suspicion, particularly in critical application areas such as medical diagnosis expert systems, where unpredictable behaviour, or even the possibility of unpredictable behaviour, is unacceptable. ANNs are, however, no longer the black boxes they used to be. We have looked at mathematical tools for understanding them, and various analytical techniques have been developed [16,37,64,77,144]. Inscrutability has, in short, become less of a disadvantage in the application of ANNs.



### *b. Representation of Structure*

A foundational assumption of current mainstream thinking about natural language is that sentences have syntactic and semantic hierarchical structure. It is easy to represent such structure using symbolic AI/NLP techniques, but is problematic in ANNs. In fact, the ability of ANNs to represent structure has been a major research issue, of which more is said later in this chapter.

### *3. Discussion*

It needs to be stressed that, in the foregoing comparison of symbolic and ANN technology, the intention was not to argue that one is necessarily “better” than the other in a partisan sense. They are alternative technologies, each with its strengths and weaknesses and, in an NLP context, can be used pragmatically in line with one’s aims [99,141,169,213,333]. The current position relative to NLP is that ANN-based systems, *while becoming ever more powerful and sophisticated, have not yet been able to provide equivalent (let alone alternative superior) capabilities to those exhibited by symbolic systems* [101, p 391].

## **B. History and Current State of ANN-Based NLP**

In the 1930s and 1940s, mathematical logicians formalized the intuitive notion of an effective procedure as a way of determining the class of functions that can be computed algorithmically. A variety of formalisms were proposed—recursive functions, lambda calculus, rewrite systems, automata, artificial neural networks—all of them equivalent in terms of the class of functions they can compute [126]. Automata theory was to predominate in the sense that, on the one hand, it provided the theoretical basis for the architecture of most current computer technology and, on the other, it is the standard computational formalism in numerous science and engineering disciplines. The predominance was not immediate, however, and artificial neural networks in particular continued to be developed throughout the 1950s and 1960s [124,160,169,222,264,335]. Indeed, the perceptron, introduced in the late 1950s, caused considerable scientific and even popular excitement because it could learn from an environment, rather than having to be explicitly configured. But, in 1969, Minsky and Papert [229] showed that there were computable functions that perceptrons could not compute [see discussion in Ref. 261], as a consequence of which ANN-based research activity diminished significantly, and throughout the 1970s automata theory became the dominant computational formalism. Some researchers persevered with ANNs, however, and by the early 1980s interest in them had begun to revive [111]. In 1986 Rumelhart and McClelland published their now-classic *Parallel Distributed Processing* [281] volumes. Among other things, these proposed the backpropagation learning algorithm, which made it possible to train multilayer nets and thereby to overcome the computational limitations that Minsky and Papert had demonstrated for perceptrons. The effect was immediate. An explosion of interest both in the theory and application of ANNs ensued, and that interest continues today.

One of the ANN application areas has been NLP, and the purpose of this section is to survey its development. Unfortunately, the application of ANN technology to NLP is not as straightforward to document as one might wish. The Preface noted that the development of NLP is historically intertwined with that of several

other language-oriented disciplines—cognitive science, AI, generative linguistics, computational linguistics [on this see Ref. 9, pp. 11–17 and 31–34]. In general, the interaction of these disciplines has been and continues to be important, because insights in any one of them can and often do benefit the others. It remains, however, that each discipline has its own research agenda and methodology, and it is possible to waste a good deal of time engaging with issues that are simply irrelevant to NLP. Now, it happens that ANN-based research into natural language has historically been strongly cognitive science-oriented, and the cognitive science agenda has driven work on NL to a large extent. For present purposes it is important to be clear about the significance of this for NLP. This section, therefore, is in two parts. The first considers the significance of the cognitive science bias for ANN-based NLP, and the second then gives an overview of work in the field.

### 1. ANN-based NLP and Cognitive Science

The formalisms invented in the 1930s and 1940s to define computable functions were soon applied to modelling of aspects of human intelligence, including language [17,159; see also discussion in *Minds and Machines* 1994; 4:377–490]. There have been two main strands of development. One is based on automata and formal language theory, and has come to be known as the “symbolic” paradigm. The other is based on ANNs and is known as the “connectionist” or “subsymbolic” paradigm. Until fairly recently, the symbolic paradigm dominated thinking about natural language in linguistics, cognitive science, and AI. It reached its apotheosis in the late 1970s, when Newell and Simon proposed the Physical Symbol System Hypothesis (PSSH), in which “physical symbol system” is understood as a physical implementation of a mathematically stated effective procedure, the prime example of which is the Turine Machine [see discussion in Ref. 334]:

The necessary and sufficient condition for a system to exhibit general intelligent action is that it be a physical symbol system. Necessary means that any physical system that exhibits general intelligence will be an instance of a physical symbol system. Sufficient means that any physical symbol system can be organized further to exhibit general intelligence [240].

The PSSH was based on existing results in linguistics, cognitive science, and AI, and was intended both as an agenda for future work in those disciplines—it *sets the terms in which we search for a scientific theory of mind* [240]—and was also widely accepted as such. Thus, by 1980, these disciplines were all concerned with physical symbol systems: in essence, the first two proposed cognitive virtual architectures, and the third implemented them. At this time, however, interest in ANNs was being revived by cognitive scientists who saw them as an alternative to the dominant symbolic paradigm in cognitive modelling, and a debate soon arose on the relative merits of the PSSH- and ANN-based approaches to cognitive modelling. The PSSH case was put in 1988 by Fodor and Pylyshyn (FP) [116], who labelled what we are here calling the symbolic as the “classical” position, and Smolensky (SM) [310] argued the ANN-based case; between them they set the parameters for subsequent discussion. It is important to be as clear as possible about the issues, so a summary is presented here.

FP and SM agreed on the following (all quotations in what follows are from [116] and [310]):

- For any object of scientific study, there are levels of description, all potentially true of that object. The level of description chosen depends on the sort of explanation required.
- The level of interest in the debate between what will henceforth be referred to as the symbolists and the subsymbolists is the cognitive level, and the aim is to specify models of (aspects of) human cognition.
- The cognitive level is defined by the postulation of representational mental states, which are states of the mind that encode states of the world; mental states have semantics. Discussions at the cognitive level, therefore, must address mental architectures based on representational states.
- There is a fundamental disagreement between symbolists and subsymbolists about the nature of mental representations and the processes that operate on them.

Thereafter, they differed. We consider their positions separately.

The symbolist position which FP articulated descends directly from the PSS hypothesis, and thus proposes cognitive architectures that compute by algorithmic manipulation of symbol structures. On this view of cognitive modelling, the mind is taken to be a symbol-manipulation machine. Specifically:

- There are representational primitives: symbols. FP refer to these as atomic representations.
- Being representational, symbols have semantic content; that is, each symbol denotes some aspect of the world.
- A representational state consists of one or more symbols, each with an associated semantics, in which (i) there is a distinction between structurally atomic and structurally molecular representations, (ii) structurally molecular representations have syntactic constituents that are themselves either structurally molecular or structurally atomic, and (iii) the semantic content of a representation is a function of the semantic contents of its syntactic parts, together with the syntactic structure.
- Input–output mappings and the transformation of mental states *are defined over the structural properties of mental representations. Because these have combinatorial structure, mental processes apply to them by virtue of their form.*

Together, the foregoing features define cognitive architectures that are intended to be taken literally in the sense that they constrain their physical realization. *In particular, the symbol structures in a classical model are assumed to correspond to real physical structures in the brain, and the combinatorial structure of a representation is supposed to have a counterpart in the structural relations among physical properties of the brain. This is why Newell (1980) speaks of computational systems such as brains as physical symbol systems. This bears emphasis because the classic theory is committed to there being not only a system of physically instantiated symbols, but also the claim that the physical properties onto which the structure of the symbols is mapped are the very properties that cause the system to behave as it does. A system which has symbolic expressions, but whose operations does not depend on the structure of these expressions does not qualify as a classical machine.*

Therefore, *if in principle syntactic relations can be made to parallel semantic relations, and if in principle you have a mechanism whose operation on expressions are sensitive to syntax, then it is in principle possible to construct a syntactically driven machine whose state transitions satisfy semantic criteria of coherence. The idea that the brain is such a machine is the foundational hypothesis of classical cognitive science.*

SM distinguished symbolic and subsymbolic paradigms in cognitive science. The symbolic paradigm corresponds directly to the classic position just outlined, whereas the subsymbolic paradigm is the one that he himself proposed. The subsymbolic paradigm defines models that are *massively parallel computational systems that are a kind of dynamical system*. Specifically:

- The representational primitives are called “subsymbols.” They are like classic symbols in being representational, but unlike them in being finer-grained: they *correspond to constituents of the symbols used in the symbolic paradigm. . . Entities that are typically represented in the symbolic paradigm as symbols are typically represented in the subsymbolic paradigm as a large number of subsymbols*. A subsymbol in a subsymbolic ANN model corresponds directly to a single processing unit.
- Being representational, subsymbols have a semantic content, that is, each subsymbol denotes some aspect of the world. The difference between symbolic and subsymbolic models lies in the nature of the semantic content. SM distinguishes two semantic levels, the conceptual and the subconceptual: *The conceptual level is populated by consciously accessible concepts, whereas the subconceptual one is comprised of finer-grained entities beneath the level of conscious concepts*. In classic models, symbols typically have conceptual semantics, that is, semantics that correspond directly to the concepts that the modeller uses to analyze the task domain, whereas subsymbols in subsymbolic models have subconceptual semantics; the semantic content of a subsymbol in a subsymbolic ANN model corresponds directly to the activity level of a single processing unit.
- In the symbolic paradigm, as noted, input–output mappings and the transformation of mental states *are defined over the structural properties of mental representations. Because these have combinatorial structure, mental processes can apply to them by virtue of their form*. This is not so in the subsymbolic case. Subsymbolic representations are not operated on by processes that manipulate symbol structures in a way that is sensitive to their combinatorial form because subsymbolic representations do not have combinatorial form. Instead, they are operated on by numeric computation. Specifically, a subsymbolic ANN model is a dynamical system the state of which is a numerical vector of the activation values of the units comprising the net at any instant  $t$ . The evolution of the state vector is determined by the interaction of (a) the current input, (b) the current state of the system at  $t$ , and (c) a set of numerical parameters corresponding to the relative strengths of the connections among units.

How do the symbolic and subsymbolic paradigms relate to one another as cognitive models? FP dismissed the subsymbolic paradigm as inadequate for cognitive modelling. SM was more accommodating.

FP argued as follows. Symbolists posit constituency relations among representational primitives, that is, among symbols, which allows for combinatorial representations. For subsymbolists the representational primitives are units or aggregates of units, but only one primitive relation is defined among units: causal connectedness. In the absence of constituency relations, subsymbolic ANN models cannot have representational states with combinatorial syntactic and semantic structure. Because subsymbolic representations lack combinatorial structure, mental processes cannot operate on them in the structure-sensitive way characteristic of symbolic models. *To summarize, classical and connectionist [= subsymbolic] theories disagree about the nature of mental representations. For the former but not the latter, representations characteristically exhibit combinatorial constituent structure and combinatorial semantics. Classical and connectionist theories also disagree about the nature of mental processes: for the former but not the latter, mental processes are characteristically sensitive to the combinatorial structure of the representations on which they operate. These two issues define the dispute about the nature of cognitive architecture.* Now, any adequate cognitive model must explain the productivity and systematicity of cognitive capacities (on systematicity see [143,241,242,304]). Symbolic models appeal to the combinatorial structure of mental representations to do this, but subsymbolists cannot: *Because it acknowledges neither syntactic nor semantic structure in mental representations, it treats cognitive states not as a generated set but as a list, and among other things lists lack explanatory utility. Because they cannot explain cognitive productivity and systematicity, subsymbolic models are inadequate as cognitive models. Subsymbolic models may be useful as implementations of symbolically defined cognitive architectures, but this has no implications for cognitive science.*

SM proposed his view of the relation between symbolic and subsymbolic paradigms in the following context: Given a physical system  $S$  and two computational descriptions of its behaviour—a “lower level” description  $\mu$ , say an assembly language program, and a “higher-level” description  $M$ , say a Pascal program, what possible relations hold between  $\mu$  and  $M$ ? Three possibilities are proposed:

1. Implementation: Both  $\mu$  and  $M$  are *complete, formal, and precise* accounts of the computation performed by  $S$ . Here  $\mu$  can be said to implement  $M$  in that there is nothing in the lower-level account that is not also in the higher-level one.
2. Elimination:  $\mu$  is a *complete, formal, and precise* account of  $S$ , and  $M$  bears no systematic relation to it. Here,  $M$  has no descriptive validity, and  $\mu$  eliminates  $M$ .
3. Refinement:  $\mu$  is a *complete, formal, and precise* account of  $S$ , and  $M$  is not. There are, however, systematic relations between  $\mu$  and  $M$ , so that  $M$  can be said to approximately describe  $S$ . Here  $\mu$  is said to refine  $M$ .

SM's proposal was that subsymbolic models refine symbolic ones, rather than, as FP had suggested, implementing them. In a now-famous analogy, he likened the relation between symbolic and subsymbolic paradigms to that which obtains between the macrophysics of Newtonian mechanics and the microphysics of quantum theory. Newtonian mechanics is not literally instantiated in the world according to the

microtheory, because fundamental elements in the ontology of the macrotheory, such as rigid bodies, cannot literally exist according to the microtheory. In short, *in a strictly literal sense, if the microtheory is right, then the macrotheory is wrong.* This does not, however, mean that Newtonian mechanics has to be eliminated, for it has an explanatory capacity that is crucial in a range of sciences and branches of engineering, and that a (strictly correct) quantum mechanical account lacks; such explanatory capacity is crucial in SM's view. Thus, cognitive systems *are explained in the symbolic paradigm as approximate higher-level regularities that emerge from quantitative laws operating on a more fundamental level—the subconceptual—with different semantics.* Or, put another way, symbolic models are competence models that idealize aspects of physical system behaviour, whereas subsymbolic models are performance models that attempt to describe physical systems as accurately as possible.

As noted, these two positions set the parameters of a debate that was to continue to the present day. It began with a long series of peer commentaries appended to Smolensky's article, and both Smolensky and Fodor subsequently expanded on their positions [117–119,311,312,314]. In addition, numerous other researchers joined the discussion; a representative sample is [21,30,40,42,59,60–62,64,66,67,89,92,93,99,108,169,171,172,177,241,267,304,347,350–352,355–357,365]; see also the discussion in *Minds and Machines* 1994. There is no way we can follow the debate any further here, or presume to judge the complex issues it has raised. We do, however, need to be clear about its implications for ANN-based NLP.

Firstly, the debate has forced a reexamination of fundamental ideas in cognitive science and AI [i.e., 64,67,93], and its results are directly relevant to NLP. It has, moreover, already been noted that much of the ANN-based research on NL is done within a cognitive science framework. ANN-based NLP cannot, therefore, afford to ignore developments in the corresponding cognitive science work. This is not bland ecumenism, but a simple fact of life.

Secondly, notwithstanding what has just been said, it remains that the cognitive science focus of the debate can easily mislead the NLP researcher who is considering ANNs as a possible technology and wants to assess their suitability. The debate centres on the nature of cognitive theories and on the appropriateness of symbolist and subsymbolist paradigms for articulation of such theories. These issues, however intrinsically interesting, are orthogonal to the concerns of NLP as this handbook construes them. Cognitive science is concerned with scientific explanation of human cognition (on explanation in cognitive science see [63,64] and *Minds and Machines* 1998, 8), including the language faculty, whereas NLP construed as language engineering has no commitment to explanation of any aspect of human cognition, and NLP systems have no necessary interpretation as cognitive models. The symbolist argument that the ANN paradigm is inadequate in principle for framing cognitive theories is, therefore, irrelevant to NLP as understood by this handbook, as are criticisms of particular ANN language-processing architectures in the literature on the grounds that they are “cognitively implausible,” or fail to “capture generalizations,” or do not accord with psycholinguistic data.

Thirdly, once the need for cognitive explanation is factored out, the debate reduces to a comparison of standard automata theory and ANNs as computational technologies [169,171]. So construed, the relation is straightforward. We have taken the aim of NLP to be design and construction of physical devices that have specific behaviours in response to text input. For design purposes, the stimulus–response



behaviour of any required device can be described as a mathematical function, that is, as a mapping from an input to an output set. Because, moreover, the stimulus–response behaviour of any physical device is necessarily finite, the corresponding input–output sets can also be finite, with the consequence that every NLP function is computable; in fact, the sizes of the I/O sets are specifiable by the designer and, therefore can be defined in such a way as to make the function not only finite and thereby theoretically computable, but also computationally tractable. As such, for any NLP mapping, there will be a Turing Machine—a PSS—that computes it. But we have seen that certain classes of ANN are Turing equivalent, so there is no theoretical computability basis for a choice between the two technologies. The choice hinges, rather, on practical considerations such as ease of applicability to the problem in hand, processing efficiency, noise and damage tolerance, and so on. A useful illustrative analogy is selection of a programming language to implement a virtual machine that computes some function. All standard-programming languages are equivalent in terms of the functions they can implement, but some are more suitable for a given problem domain than others in terms of such things as expressiveness or execution speed: assembler would be much harder to use in coding some complex AI function than, say LISP, but once done it would almost certainly run faster [169,171].

And finally, the debate has set the agenda for ANN-based language-oriented research in two major respects: the paradigm within the research is conducted, and the ability of ANNs to represent compositional structure. Both these issues are discussed in what follows.

## 2. ANN-Based NLP: An Overview

ANN-based NL research [46,55,101,276,296,299,301] began, fairly slowly, in the early 1980s with papers on implementing semantic networks in ANNs [167], visual word recognition [139,216,279], word–sense disambiguation [72–74], anaphora resolution [275], and syntactic parsing [109,294,295,309]. In 1986, Lehnert published a paper [212] on the implications of ANN technology for NLP, an indication that this early work had by then attracted the attention of mainstream work in the field. Also, 1986 was the year in which the *Parallel Distributed Processing* volumes [281] appeared, and these contained several chapters on language: McClelland and Kawamoto on case role assignment, McClelland on word recognition, and Rumelhart and McClelland on English past-tense acquisition. All of these were to be influential, but the last-named had an effect out of all proportion to the intrinsic importance of the linguistic issue it dealt with. Rumelhart and McClelland [280] presented an ANN that learned English past-tense morphology from a training set of (past-tense, present-tense) verb form pairs, including both regular (“-ed”) and irregular formations. They considered their net as a cognitive model of past-tense morphology acquisition on the grounds that its learning dynamics were in close agreement with psycholinguistic data on past-tense acquisition in children and, because it was able to generalize the regular-tense formation to previously unseen present-tense forms after training, that it had learned an aspect of English morphology. Crucially, though, the net did this without reference to any explicit or implicit PSS architecture. This was quickly perceived as a challenge by symbolist cognitive scientists, and it became a test case in the symbolist vs. subsymbolist debate outlined earlier. Pinker and Prince [252] made a long and detailed critique, in response to

which the Rumelhart and McClelland model was refined by a succession of researchers [64,82,83,108,219,258,259]; see also 220 and the discussion in Chap. 31].

From an NLP point of view, the chief importance of Rumelhart and McClelland's work and its successive refinements is not in its validity as a cognitive model, but in the impetus that it gave to ANN-based NL research. It made 1986 a watershed year, in the sense that the number of language-oriented papers has increased dramatically since then. Disregarding speech and phonology because of this handbook's focus on text processing, there has been further work on a wide variety of topics, a representative selection of which follows:

Morphology [64,82,83,108,128,129,147–149,219,220,234,258,259,268]  
 Lexical access [257,293]  
 Lexical category learning [104–106,115]  
 Noun phrase analysis [370]  
 Anaphora resolution [1]  
 Prepositional phrase attachment [71,343]  
 Grammaticality judgment [3,205,207,210,378]  
 Syntax acquisition [45,53,54,56–58, 104–106,108,144,362,377]  
 Parsing [47,73,145,161–163,174,178–180,198–200,227,231,270,320,324,334,  
 359,363]  
 Case role assignment [217,224,228,324,325]  
 Lexical semantics [11,12,69,90,110,113,235,238,239,272–274,286–290,319,  
 323,366]  
 Lexical disambiguation [34,190]  
 String semantics [3,71,144,228,233,323,325,326,367]  
 Metaphor interpretation [236,369]  
 Reasoning [201–203,306,328–331]  
 Full text understanding [35,223,225,226]  
 Language generation [76,85–87,127,150,197,286]

A range of practical NLP applications have also been developed. Chapters 33–37 give a representative sample.

The extent of NLP-related work since 1986 precludes any attempt at a useful summary here. Instead, the rest of this chapter discusses several important general issues in ANN-based NLP, leaving detailed consideration of specific topics and research results to the chapters that follow.

#### *a. Research Paradigms*

The symbolist–subsymbolist debate has resulted in a trifurcation of ANN-based natural language-oriented research, based on the perceived relation between PSSH- and ANN-based cognitive science and AI:

- The symbolic paradigm accepts FP's view of the position of ANNs relative to cognitive science. It considers ANNs as an implementation technology for explicitly specified PSS virtual machines, and it studies ways in which such implementation can be accomplished. NL-oriented work in this paradigm is described in Chap. 30.
- The subsymbolic paradigm subdivides into what is sometimes called “radical connectionism,” which assumes no prior PSSH analysis of the problem



domain, but relies on inference of the appropriate processing dynamics from data, and a position that, in essence, considers prior PSSH analysis of the problem domain as a guide to system design or as an approximate or competence description of the behaviour of the implemented system. References [91–97,356] exemplify the radical position, and Smolensky, in a manner of speaking the father of subsymbolism, has in various of his writings [310,314–316] taken the second. The subsymbolic paradigm is described in Chap. 31.

- The hybrid paradigm, as its name indicates, is a combination of the symbolic and the subsymbolic. It uses symbolic and subsymbolic modules as components in systems opportunistically, according to what works best for any given purpose. A subsymbolic module might, for example, be used as a preprocessor able to respond resiliently to noisy input, whereas the data structures and control processes are conventional PSS designs [99,169,171,218,332,333,367,368]. The hybrid paradigm is described in Chap. 32.

Interest in the hybrid paradigm has grown rapidly in recent years and, to judge by relative volumes of research literature, it is now the most often used of the foregoing three alternatives in engineering-oriented applications such as NLP. It is not hard to see why this should be so. The hybrid paradigm makes full use of theoretical results and practical techniques developed over several decades of PSSH-based AI and NLP work, and supplements it with the function-approximation and noise-resistance advantages of ANNs when appropriate. By contrast, the symbolic and subsymbolic paradigms are in competition with established PSSH-based theory or methodology. On the one hand, the symbolic paradigm has yet to demonstrate that it will ever be superior to conventional computer technology as an implementation medium for PSS virtual machines [i.e., 99]. On the other hand, the subsymbolic paradigm essentially disregards existing PSSH-based NLP theory and practice, and starts afresh. Of the three paradigms, therefore, it is the least likely to generate commercially exploitable systems in the near future, although it is the most intriguing in pure research terms.

### *b. Representation*

The most fundamental requirement of any NLP system is that it represent the ontology of the problem domain [64,114,124,300,303,304]; see also Chap. 28]. One might, for example, want to map words to meanings, or strings to structural descriptions: words, meanings, strings, and structures have to be represented in such a way that the system can operate on them so as to implement the required mapping. Most ANN-based NL work has been directly or indirectly concerned with this issue, and this section deals with it in outline.

Before proceeding, a particular aspect of the representation issue has to be addressed. As we have seen, the symbolist–subsymbolist debate made representation of compositional structure a major topic in ANN-based cognitive science. FP claimed that ANNs were incapable of doing so and, accordingly, dismissed them as inadequate for cognitive modelling. In response, adherents of ANN-based cognitive science have developed a variety of structuring mechanisms. Now, FP insists on representation of compositional structure in cognitive modelling on explanatory

grounds: it captures the productivity and systematicity of cognitive functions such as language. But NLP is not interested in cognitive explanation. The aim is to design and implement devices having some specified I/O behaviour. Whatever its importance for cognitive science, therefore, is representation of compositional structure is an issue for ANN-based NLP?

Generative linguistic theory provides a well-developed and, by now, natural way of thinking about NL, a foundational assumption of which is that sentences are compositionally structured objects. Symbolist AI and NLP have shared this assumption, and designed systems in which compositional data structures are manipulated by structure-sensitive processes. This is one way of approaching the design and implementation of NLP devices, but not the only way [64]. System identification theory poses the black box problem, which asks: Given a physical device with an observable input–output behaviour, but for which internal mechanism is not open to inspection, what is its internal mechanism? 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, but 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* [8, pp. 38–39 for discussion see Refs. 42 and 43]. Now, we have taken the aim of NLP to be design and construction of physical devices with a specified I/O behaviour, so the identification problem applies directly: given a desired I/O behaviour, what mechanism should be used to produce it? One answer is a Turing Machine, as discussed earlier, but it is not the only answer [196]. It is possible to use automata of complexity classes lower than that of Turing Machines, and even finite state machines to compute NLP functions [51–55, 230]: we have already seen that NLP functions are necessarily finite, and any finite function can be computed by a finite-state machine. But finite-state machines can represent only trivial compositionality that reduces to simple sequentiality; it was because of their inability to represent nontrivial structure—that is, simultaneous left- and right-branching dependency—that Chomsky originally rejected them as models for NL sentence structure [49, p.24]. Indeed, one could even use a continuous space dynamical system, for which compositional discrete symbol structures are undefined. All three mechanisms are theoretically capable of generating the required observable behaviour and, assuming they are appropriately configured, they are equivalent for NLP purposes.

In principle, therefore, compositional structure is not necessary for NLP. It may, however, be useful in practice. A discrete-time, continuous-space dynamical system, such as a two-layer feedforward ANN, with sigmoid activation function, may theoretically be capable of implementing any computable function, but, for some particular function, is finding the required weight parameters computationally tractable, and will network complexity have reasonable space requirements? It may well turn out to be that compositional structure makes implementation of certain NLP functions easier or indeed tractable; the need for compositional structure in ANN-based NLP is an empirical matter and, consequently, researchers need to be aware of the structuring mechanisms developed by ANN-based cognitive science.

There are two fundamentally different approaches to ANN-based representation [60,64,108,336,348,349]:

1. Local representation: Given some set  $E$  of objects to be represented and a set  $N$  of network units available for representational use, a local representation scheme (or “local scheme” for short) allocates a unique unit or group of units  $\in N$  for each object  $e \in E$ .
2. Distributed representation: Given the same sets  $E$  and  $N$ , a distributed representational scheme uses all the units  $n \in N$  to represent each  $e \in E$  [348,349].

The difference is exemplified in the pair of representational schemes for the integers 0..7 shown in Fig. 11. In the local scheme each bit represents a different integer, whereas in the distributed one, all the bits are used to represent each integer, with a different pattern for each. Because, in the local scheme, each bit stands for one and only one integer, it can be appropriately labelled, but in the distributed scheme, no bit stands for anything on its own; no bit can be individually labelled, and each is interpretable only in relation to all the others.

Local and distributed schemes both have advantages [60,99,108,124,300,305,336]. Much of the earlier work used localist representation, and although the balance has now shifted to the distributed approach, significant localist activity remains [15,35,101,110,133,201,203,274,305,306] (see also the discussion in Chap. 32). In what follows, local and distributed approaches to representation of primitive objects and of compositional structure in ANNs are discussed separately.

#### LOCAL REPRESENTATION

##### Representation of primitives

Local representation of primitive objects is identical with that in the PSSH approach: in a PSS each object to be represented is assigned a symbol, and in local ANN representation each object is assigned a unit in the net.

##### Representation of structure

Local representation of primitive objects is straightforward, but representation of compositional structure is not. The difficulty emerges from the following example [24]. Assume a standard AI blocks world consisting of red and blue triangles and squares. A localist ANN has to represent the possible combinations.

	Local	Distrib
1	0000001	0000001
2	0000010	0000010
3	0000100	0000011
4	0001000	0000100
5	0010000	0000101
6	0100000	0000110
7	1000000	0000111

**Fig. 11** Local and distributed representations of the integers 1–7.

One unit in the net is allocated to represent “red”, another for “blue,” another for “triangle,” and yet another for “square.” If one wants to represent “red triangle and blue square,” the obvious solution is to activate all four units. Now represent “blue triangle and red square” using the same procedure. How can the two representations be distinguished? The answer is that they cannot, because there is no way to represent the different colour-to-shape bindings in the two cases. In other words, there is no way to represent constituency in a net such as this. Localists have developed a variety of binding mechanisms to overcome this problem [24,102,305,306,329]:

#### DISTRIBUTED REPRESENTATION

##### Representation of primitives

In distributed ANNs, each primitive object is represented as a pattern of activity over some fixed-size group  $g$  of  $n$  units, or, abstractly, as a vector  $v$  of length  $n$  in which the value in any vector element  $v_i$  represents the activation of unit  $g_i$ , for  $1 \leq i \leq n$ . Such representation has two properties that localist schemes lack.

Firstly, the relation between a representation and what it represents can be nonarbitrary in a distributed scheme. In a localist scheme, the relation is arbitrary: each node (or node group) in a localist net represents a primitive object, and it does not matter which node is chosen for which object. Such arbitrariness extends to certain kinds of distributed representation as well. ASCII encoding can, for example, be taken over directly into a text-based NLP system to represent alphanumeric characters, such that the 256 eight-bit codes are distributively represented over eight network units. This gives advantages of efficiency and damage resistance, but the representation is arbitrary in the sense that each code assignment depends only on binary number notation and the order in which the characters happen to have been arranged. If one wanted to be perverse, the code-to-character assignment could be altered without affecting the usefulness of the scheme, because there is no reason to prefer any particular code for any particular character. Distributed representation can, however, be made nonarbitrary by using feature vectors. Compare, for example, ASCII and feature–vector representations for alphanumeric characters in Fig. 12.

In the feature–vector scheme, each element of the grid is a pixel; the rows are concatenated from the upper right, assigning value-1 for dark and 0 otherwise, and the resulting vector represents the physical shape of the corresponding letter. As such, the assignment of representation to what it represents is nonarbitrary.

Secondly, a nonarbitrary distributed scheme can represent similarities among the primitive objects. In any distributed scheme, arbitrary or nonarbitrary, the dimensionality  $n$  of the vectors defines an  $n$  dimensional space within which the representations have a similarity structure in the sense that some vectors are closer than others in that space. In a nonarbitrary scheme, however, the similarity structure systematically reflects similarities among represented objects. Referring to Fig. 12, F and T are more visually similar than either O and F or O and T. In the ASCII scheme there is no systematic relation between vector distance among codes and visual similarity among bitmaps: the ASCII codes for O and F are most similar, those for O and T are farthest apart. Distance among feature–vectors, however, corresponds directly to visual similarity.




	ASCII	Bitmap	Bitmap vector
F	01000100		00111001000011000100
O	01001101		01110100011000101110
T	01010010		01110001000010000100

Fig. 12 ASCII and feature-vector representations of alphanumeric characters.

Until fairly recently, feature-vector representations were handcrafted, and were thus conditioned by individual designers' analyses of what is significant in task domains. There has been a move away from such explicitly designed distributed representational schemes to learned ones. For example, a feedforward MLP can be used for this purpose by training it to autoassociate vectors, that is, by making each vector  $v_i$  in some data set both input and target output. If the hidden layer is smaller than the input-output layers, then  $v_i$  at the input of a trained net will generate a "compressed representation" of itself on the hidden units, which can be used as a representation of  $v_i$  in subsequent processing. Examples of learned distributed representations in NLP applications [101] are FGREP [223,225] (described in Chap. 37) and xRAAM [211].

#### Representation of structure

Using a distributed ANN to represent compositional structure is difficult because arbitrarily complex structures have to be represented with a fixed-size resource, that is, over some specific group of units. To see this, assume that the primitive objects in a given domain are represented as feature vectors. An ANN that uses distributed representations by definition uses all the available units for each vector. There would be no difficulty about individually representing "man," for example, or "horse." But how would the net represent "man" and "horse" at the same time? Even more difficult is representation of relations, such as "man on horse." The problem, therefore, has been to find ways of overcoming this difficulty.

The crucial insight came from van Gelder in 1990 [348; discussed in Ref. 64]. He argued that distributed representations can be compositional, but not necessarily in the sense intended within the PSSH paradigm. Van Gelder's paper has become very influential in ANN-based cognitive science generally; its importance for present purposes lies in the distributed compositional representation that it proposes, and that also underlies several distributed ANN-based representational mechanisms. Because of its importance, a reasonably detailed account of it is given here.

Van Gelder states a set of "minimal abstract conditions" that any representational scheme must satisfy for it to be compositional:

There is a set of primitive types (symbols, words, and such)  $P_i$ . For each type there is available an unbounded number of tokens.

There is a set of expression types  $R_i$ . For each type there is available an unbounded number of tokens.

There is a set of constituency relations over these primitive and expression types.

He then focusses on the distinction between the types and tokens in these conditions, and makes that distinction the basis of his proposal for distributed compositional representation. The argument goes like this. Specification of a particular representational scheme is bipartite. On the one hand, the primitive and expression types together with the abstract constituency relations among them have to be stated. On the other hand, it is necessary to state what the tokens of the primitive and expression types look like physically, and how primitive and expression tokens can be combined to generate new expression tokens. The physical specification is necessary because it provides a notation in terms of which the abstract specification can be stated and applied; without a notation, there is no way to talk about the abstract representational scheme. The standard way of specifying compositional representational schemes conflates the abstract and physical specifications. Thus, the specification, *If A and B are wff, then (A&B) is a wff*, uses the physical symbols  $A$ ,  $B$  and  $(A\&B)$  to specify two primitive types and their combination is an expression type without making the distinction between abstract and physical explicit. This is necessarily the case, because notation exists to permit formal statement of abstractions. It does, however, obscure the distinction, maintenance of which is crucial to the argument for distributed compositional representation. In particular, given any set of primitive types, primitive expressions, and constituency relations, and given also a set of primitive tokens, there is more than one way of physically instantiating the abstract constituency relations by combining primitive tokens into expression tokens. Two sorts of such "modes of combination" are cited:

1. Concatenative combination. The tokens of an expression's constituents are physically present in the expression token. Thus, the logical expression  $[(P\&O)\&R]$  contains the expression token  $(P\&Q)$  and the primitive token  $R$ , and  $(P\&Q)$  itself contains the primitive tokens  $P$  and  $Q$ . This concatenative compositionality is spatial: primitive tokens are placed alongside one another in a physical sequence, as in  $[(P\&Q)\&R]$ .

A concatenative representational scheme says two things: *When one is describing a representation as having a concatenative structure, one is making more than just the grammatical point that it stands in certain abstract constituency relations. One also says that it will have a formal structure of a certain kind, that is, a structure such that the abstract constituency relations among expression types find direct, concrete instantiation in the physical structure of the corresponding tokens. This is called syntactic structure. Thus, the syntactic structure of the representation is the kind of formal structure that results when a concatenative mode of combination is used.*

2. Nonconcatenative combination. The familiar compositional schemes—natural languages, programming languages, mathematical and logical languages—are all concatenative. This makes it easy to lose sight of the possibility that there might be alternatives to concatenative combination



of tokens for representation of abstract constituency relations. To represent abstract constituency relations, it is sufficient to specify general, effective, and reliable procedures for generating expression tokens from constituent tokens, and for decomposing expressions into their constituents. One way of doing this is to specify processes operating on concatenative representations, but there is no reason, in principle, why tokens of constituents should be literally present in token expressions. If general, effective, and reliable procedures for generation and decomposition of nonconcatenative expression tokens can be specified, then the scheme can legitimately be said to represent the corresponding abstract constituency relations. Any scheme that specifies such procedures is said to be functionally compositional. Van Gelder cites Goedel numbering for formal languages as an example, the details of which would take us too far afield; the important features for present purposes are that it provides the required general, effective, and reliable procedures for constructing expression tokens from constituent tokens and for decomposing expression tokens into constituent tokens, and that the generated expression tokens do not literally contain the constituent tokens, as they do in concatenative schemes.

Concatenation cannot work with an ANN that uses distributed representation because the size of an expression token must increase with the complexity of the abstract constituency that it represents, but the ANN representational resource is limited to the size of a single primitive token. The importance of nonconcatenative representation is that it breaks the link between abstract complexity and spatial representation size: because it does not require constituent tokens to be physically present in an expression token, it becomes possible, in principle, to represent abstract constituency relations over a fixed-size resource. What is needed in ANN terms are "general, effective, and reliable procedures" to compose constituent tokens into and to decompose them from expression tokens represented over the representational units of the net. Several such nonconcatenative mechanisms have been proposed [101,242,348], chief among them tensor products [313], recursive autoassociative memories (RAAM) [262,263; see also Ref. 41], and holographic-reduced descriptions [254-256]. These are discussed in Chaps. 30 and 31.

### *c. Sequential Processing*

Text processing is inherently sequential in the sense that word tokens arrive at the processor over time. ANN-based work on NL has addressed this sequentiality using three main types of network architecture [23]:

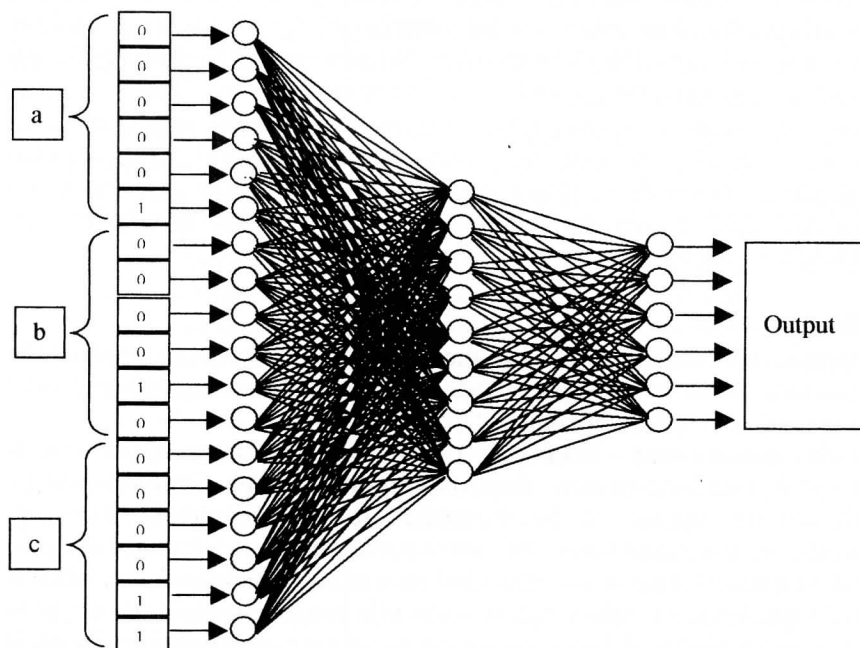
**MULTILAYER PERCEPTRONS.** MLPs are a feedforward architecture, and time is not a parameter in feedforward nets: they map inputs to outputs instantaneously. Consequently, MLPs appear to be inappropriate for sequential processing. Nevertheless, the earlier ANN-based NL work used them for this purpose by, in effect, spatializing time. Given a set of symbol strings to be processed, the MLP is given an input layer large enough to accommodate the longest string in the set, as in Fig. 13. This was unwieldy both because, depending on the input encoding scheme, it could result in large nets that take a long time to train, and because of the inherent variability in the length of NL strings. It is now rarely used.

**TIME-DELAY NEURAL NETWORKS.** A time-delay neural network (TDNN) is an MLP for which the input layer is a buffer  $k$  elements wide, as shown in Fig. 14. It processes input dynamically over time  $t_0, t_1 \dots t_n$  by updating the input buffer at each  $t_i$  and propagating the current input values through the net to generate an output. The problem here is buffer size. For example, any dependencies in a string for which the lexical distance is greater than the buffer size will be lost. In the limiting case, a buffer size equal to the input string length reduces to an MLP. TDNNs have been successfully used for finite-state machine induction [70] and in NLP applications [23,32,33].

**RECURRENT NETWORKS.** Recurrent networks (RANN) use a fixed-size input layer to process strings dynamically; RANNs used for NL work are discrete-time, and input successive symbols in a string at time steps  $t_0, t_1 \dots t_n$ , as in Fig. 15. The net's memory of sequential symbol ordering at any point in the input string is maintained in the current state, which is fed back at each time step.

NL research using RANNs has proceeded in concert with the work on automata induction described earlier [53,54,56–58,68,104–108,185,204,205,207,210,297,298]. Because RANNs are dynamical systems, they can display the range of behaviours that characterize such systems. ANN-based NL work has so far used mainly fixed-point dynamics, but there have been some who exploit more complex dynamics [13,29,177,194,265] (see also discussion in [23,108] and Chap. 31).

Other approaches to dynamic processing of symbol sequences have been developed [232], such as sequential Kohonen nets [44,181], but most existing ANN-based NL work has used the foregoing three varieties.



**Fig. 13** An MLP for string processing.



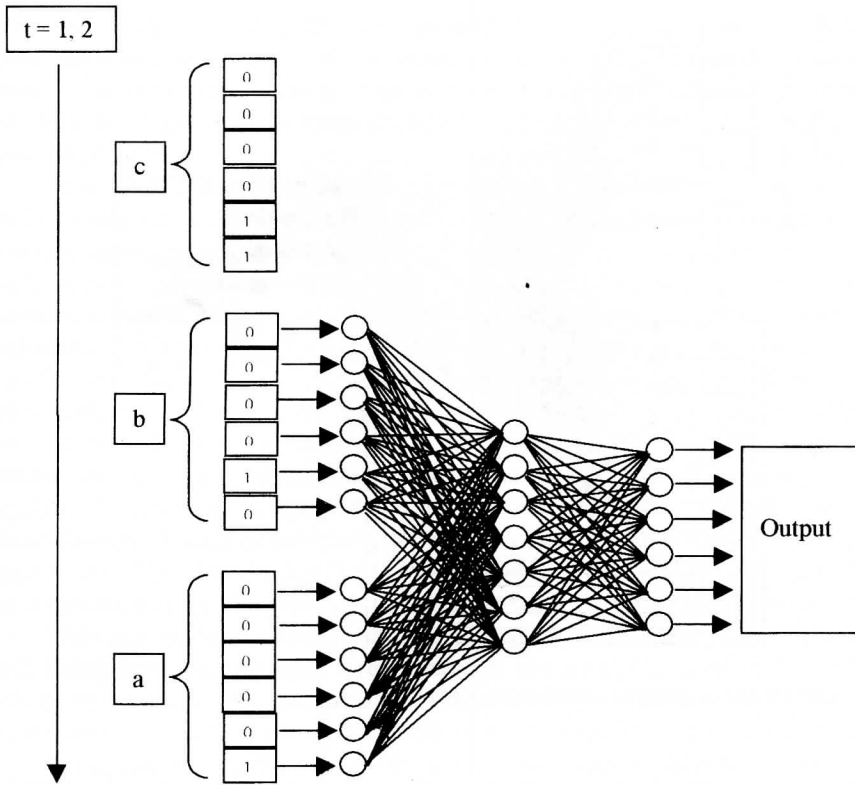
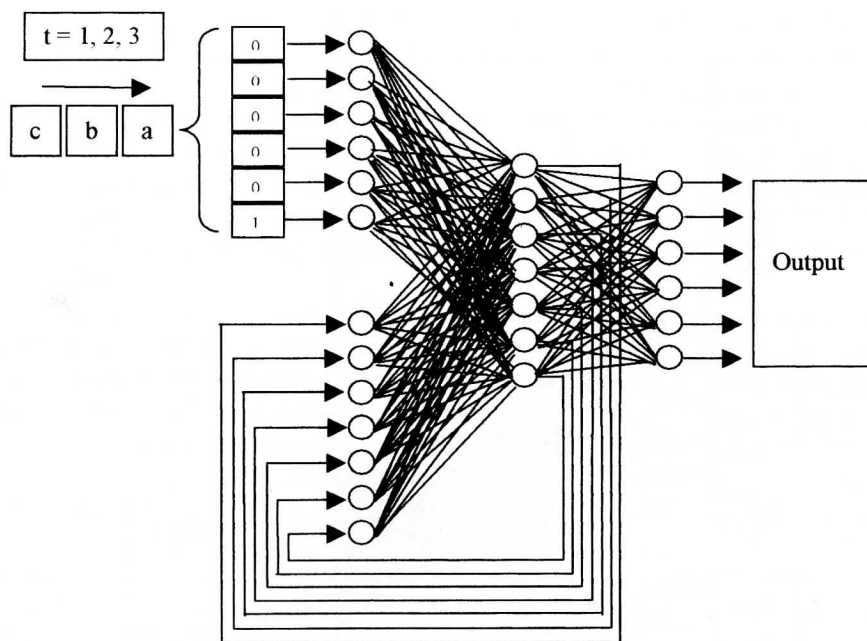


Fig. 14 A TDNN for sequential string processing.

Dynamic processing of language is usefully understood in the context of work on the dynamical systems approach to cognitive modelling [20,66,67,107,108,191,267,351].

*d. Tabula Rasa Learning*

The overwhelming majority of ANN-based language work involves learning, and is thus describable as language acquisition research in a broad sense. There is a large generative linguistics and cognitive science literature on language acquisition, the aim of which is to explain how children learn the linguistic knowledge characteristic of adult native speakers. Such explanation is typically stated in terms of a generative linguistic theoretical framework, and based on the “poverty of the stimulus” argument: the examples of language usage available to children is insufficiently rich to allow adult linguistic competence to be inferred without some innate set of constraints that make the acquisition process tractable; this set of constraints is known as Universal Grammar [50,253]. ANN-oriented linguists and cognitive scientists on the other hand argue that *rich linguistic representations can emerge from the interaction of a relatively simple learning device and a structured linguistic environment* [260; see also 3,64,108], although most now agree that some form of innate constraint is necessary [17,18,108,182,302]; on the nature of such constraints; however,



**Fig. 15** A RANN for sequential string processing.

see other discussions [52,53,57,64,108]. The difference between them and generativists is not absolute, in the sense that they support *tabula rasa* learning as a general strategy for language acquisition, but rather one of degree: they differ from generativists on the relative importance of innate constraints and environmental factors, tending to emphasize the latter. On language acquisition see also Chap. 29.

In the discussion of computational complexity of ANN learning we looked briefly at ways of initializing nets with knowledge of the problem domain, thereby constraining the learning process and rendering it more tractable. These are general techniques, and are thus applicable to NL language learning. In the NLP literature, however, two main emphases have emerged [64]:

**INCREMENTAL TRAINING.** As we have seen, domain knowledge can be incorporated into a net by appropriate initialization of weights. NL-oriented work has, however, preferred to achieve this by incremental training, whereby domain knowledge is not explicitly compiled into network weights, but acquired from data. For some language-learning task, the net is initially trained on short, syntactically simple strings, and as these are learned, increasingly longer and more complex strings are introduced into the training set. Elman [106] found that, for an NL set  $S$  containing strings that varied in structural complexity from monoclausal to multiple layers of embedding, a given RANN with randomly initialized weights—that is, a *tabula rasa* net lacking any systematic prior knowledge—could not learn the  $S$  if the whole of  $S$  was used for training, but could if the strings were presented such that the monoclausal ones were presented first, then those with one level of embedding, and so on. In the first phase of training, all 10,000 strings of the training set were monoclausal. Once the net had learned these, the training set was redefined to include 7,500

monoclausal strings and 2,500 with embedded clauses, then 5,000 monoclausal and 5,000 complex, and finally 10,000 complex. Loading of the final 10,000 complex strings was successful, whereas the attempt to train the same *tabula rasa* net on the final training set of 10,000 complex strings had been unsuccessful [see also Refs. 64,65,135].

**TOPOLOGICAL STRUCTURING.** Theoretically, a single MLP with a sufficient number of hidden units can implement any computable function. Therefore, it should be possible to implement any required NLP function using a sufficiently large MLP. Early work in fact did use single MLPs to solve small language-processing problems, but there is now a widespread recognition that this approach will not work for large, real-world applications: *We cannot feed 20 years of raw sensory input to a 3-layer, feed-forward, back-error propagation network and then expect a college graduate to result* [199, p. 46; 101,110]. As NLP systems have grown in size and complexity, they have increasingly used modular architectures, in which modules for specific tasks are interconnected in accordance with the preanalyzed requirements of the problem domain, and thereby compute the required function globally [84,125,184]. Miikkulainen's language-understanding system, described in Chap. 37, is an example [see also 225,226,228]; the  $L_0$ /NTL project is another [110,113]. Hybrid architectures are a special case of modular ones in which ANN and PSS-based modules interact.

Why are modular systems expected to work where large single-ANN ones are not? Essentially because the structuring of the modules relative to one another reflects prior knowledge of the task domain, and is thus a way of incorporating prior constraints into a learning system to aid tractability, as discussed earlier.

An alternative approach to topological structuring focuses on the amount of memory available to the learning system. In the same series of experiments referred to in the preceding subsection on incremental training, Elman [106] presented the entire training set to the net in one batch, but varied the amount of feedback memory, starting small and gradually increasing it in the course of training. The result was that the net was able to learn the training set, but only in stages: initially, when memory was most restricted, it learned only the simplest strings in the training set and, as memory was gradually increased, it was able to learn strings of increasing syntactic complexity [see also 64].

#### e. *Meaning*

There are some NLP applications, such as document search, for which the meaning of the text being processed is not an issue. In others, semantic interpretation of text is necessary, but reasonably straightforward; an example would be an NL command interpreter for a database front end, where both the syntax of input strings and the semantic interpretations to which they are mapped are both well defined and severely restricted relative to normal linguistic usage. When, however, one moves to AI-oriented applications such as (more or less) unrestricted NL-understanding systems, semantic interpretation becomes a difficult and still largely unresolved problem. ANNs do not provide an easy solution, but they do offer a promising alternative to existing PSSH-based approaches.

Meaning is variously understood by different disciplines and by researchers within them. It does, however, seem noncontroversial to say that meaning of NL linguistic expressions has to do with denotation of states of the world, and that semantic interpretation is a mapping from strings to denotations. That, in any

case, is what is assumed here. PSSH-based AI and NLP systems have implemented the mapping by constructing system-internal representations of some aspect of the world—the “domain of discourse”—and then relating input strings to the representation [25,192]. How best to represent the word has become of the research discipline in its own right—knowledge representation—and numerous formalisms exist. Systems that use logic formalisms, for example, transform input strings into logical propositions, and these are then related to the domain representation using a deductive inference mechanism to arrive at a semantic interpretation. Some ANN-based work on semantic interpretation continues in the PSSH tradition, in the sense that they use explicitly designed domain representations. Other work takes a radically different approach, however: input strings are mapped not to explicitly designed representations of the world which, inevitably, reflect a designer’s analysis of what is significant in the task domain, but to representations that are learned from the world through transducers without designer intervention. At its most ambitious, this line of research aims to embed NLP systems in robotic agents that not only receive inputs from an environment by, say, visual, acoustic, and tactile transducers, but also interact with and change the environment by means of effectors. The aim is for such agents to develop internal world representations by integrating inputs and internal states through self-organization based on adaptive interaction with the environment: *Concepts are thus the “system’s own,” and their meaning is no longer parasitic on the concepts of others (the system designer)* [97]. In particular, agents would learn to represent the meanings of words and expressions from their use in specific environment-interactive situations. Work on this is proceeding[11,12,75, 92,93,96–98,100,101,112,113,176,237–239,251,272–274,283,288,319,322,356], although it must be said that, to keep experimental simulations tractable, the goal of real-world interaction is often reduced to explicitly designed microworlds reminiscent of ones like the famous SHRLDU in the PSSH tradition. On these matters see also Chap. 31.

The work just cited is symptomatic of developments in cognitive science and AI that emphasize the role of the real-world environment in the explanation of cognition and in the design and construction of artificially intelligent machines [66,67,93,151–158].

## REFERENCES

The references cited in the text and listed in the following appear in conventionally published books, journals, conference proceedings, and technical reports. As in virtually all other areas of academic study, however, a major new source of research information has developed in recent years: the Internet. Subject-specific Web sites with a wealth of information and resources now exist, and most research groups and individual researchers maintain their own sites, where the status of current work is reported, and from which it is typically possible to download existing research papers and preprints of work to appear in conventional paper format. Because this information source is now effectively indispensable, this list of references is prefaced with a list of Web URLs that are particularly useful for ANN-based NLP. As all Web users know, URLs can be volatile, and those given are correct at the time of writing, but where there has been a change it is usually possible to find the site using a search engine.

## A. Web Resources

### 1. NLP-Specific Sites

- R Cole, J Mariani, H Uszkoreit, A Zaenen, V Zue, eds. Survey of the State of the Art in Human Language Technology. Center for Spoken Language Understanding, Oregon Graduate Institute of Science and Technology:  
<http://www.cse.ogi.edu/CSLU/HLTsurvey/>
- Association for Computational Linguistics:  
<http://www.cs.columbia.edu/~radev/newacl>

### 2. Directories

- Galaxy:  
<http://galaxy.einet.net/galaxy/Engineering-and-Technology/Computer-Technology.html>
- Pacific Northwest National Laboratory  
<http://www.emsl.pnl.gov:2080/proj/neuron/neural>
- Yahoo. Neural Networks:  
[http://www.yahoo.co.uk/Science/Engineering/Electrical\\_Engineering/Neural\\_Networks/](http://www.yahoo.co.uk/Science/Engineering/Electrical_Engineering/Neural_Networks/)
- Yahoo: Natural Language Processing  
[http://www.yahoo.co.uk/Science/Computer\\_Science/Artificial\\_Intelligence/Natural\\_Language\\_Processing/](http://www.yahoo.co.uk/Science/Computer_Science/Artificial_Intelligence/Natural_Language_Processing/)

### 3. Coordinating Organizations

- European Network in Language and Speech (ELSNET):  
<http://www.elsnet.org>
- IEEE Neural Network Council:  
<http://www.ewh.ieee.org/tc/nnc/index.html>
- International Neural Network Society:  
<http://www.inns.org>

### 4. Research Groups

Most research groups maintain Web sites. Space precludes any attempt at an exhaustive listing. What follows is a selection of those that have proven particularly useful; all contain links to other sites. Individual researchers' pages are not included. These can be found using a Web search engine.

- The Neural Theory of Language Project (formerly the L<sub>0</sub> project):  
<http://www.icsi.berkeley.edu/NTL>
- UTCS Neural Networks Research Group:  
<http://net.cs.utexas.edu/users/nn>
- Austrian Research Institute for Artificial Intelligence (ÖFAI):  
<http://www.ai.univie.ac.at/oefai/nn/nnngroup.html>
- Artificial Intelligence Research Laboratory, Department of Computer Science, Iowa State University:  
<http://www.cs.iastate.edu/~honavar/aigroup.html>
- Brain and Cognitive Sciences, University of Rochester:  
<http://www.bcs.rochester.edu/bcs>

- Language and Cognitive Neuroscience Lab, University of Southern California:  
<http://siva.usc.edu/coglab>
- NEC Research Institute:  
<http://external.nj.nec.com/homepages/giles/>
- Center for the Neural Basis of Cognition, Carnegie Mellon University/University of Pittsburgh  
<http://www.cnbc.cmu.edu/>
- Interactive Systems Lab, Carnegie Mellon University/University of Karlsruhe:  
<http://www.is.cs.cmu.edu/ISL.neuralnets.html>
- Neural Network Theory at Los Alamos, Los Alamos National Laboratory:  
[http://www-xdiv.lanl.gov/XCM/neural/neural\\_theory.html](http://www-xdiv.lanl.gov/XCM/neural/neural_theory.html)
- Neural Networks Research Centre, Helsinki University of Technology:  
<http://www.cis.hut.fi/research>
- Neural Computing Research Group (NCRG), Aston University  
<http://www.ncrg.aston.ac.uk>
- Natural Language Processing Group, University of Sheffield:  
<http://www.dcs.shef.ac.uk/research/groups/nlp/>
- Centre for Neural Networks (CNN), King's College, London:  
<http://www.mth.kcl.ac.uk/cnn/>

#### 5. *Publication Archives*

The main electronic print archive is Neuroprose at <ftp://archive.cis.ohio-state.edu/pub/neuroprose/>; a directory of others is at NCRG, <http://neural-server.aston.ac.uk/NN/archives.html>

#### 6. *Discussion Groups*

- Neuron Digest:  
Contact address: [neuron-request@psych.upenn.edu](mailto:neuron-request@psych.upenn.edu)
- Connectionists:  
Contact address: [connectionists-request@cs.cmu.edu](mailto:connectionists-request@cs.cmu.edu)

### **B. Cited References**

1. R Allen. Several studies on natural language and back-propagation. Proceedings of the First Annual International Conference on Neural Networks, 1987.
2. R Allen. Sequential connectionist networks for answering simple questions about microworlds. Proceedings of the Tenth Annual Conference of the Cognitive Science Society, 1988.
3. J Allen, M Seidenberg. The emergence of grammaticality in connectionist networks. In: B Macwhinney, ed. Emergentist Approaches to Language: Proceedings of the 28th Carnegie Symposium on Cognition, Hillsdale, NJ: Lawrence Erlbaum.
4. N Alon, A Dewdney, T Ott. Efficient simulation of finite automata by neural nets. J ACM 38:495–514, 1991.
5. M Anthony. Probabilistic analysis of learning in artificial neural networks: the PAC model and its variants. Neural Comput Surv 1:1–47, 1997.
6. S Amari. Learning and statistical inference. In: M Arbib, ed. The Handbook of Brain Theory and Neural Networks. Cambridge, MA: MIT Press, 1995.



7. J Anderson. *An Introduction to Neural Networks*. Bradford Books/MIT Press, 1995.
8. M Arbib. *Brains, Machines, and Mathematics*. 2nd ed. New York: Springer, 1987.
9. M Arbib, ed. *The Handbook of Brain Theory and Neural Networks*, Cambridge, MA: MIT Press, 1995.
10. M Aretoulaki, G Scheler, W Brauer. Connectionist modelling of human event memorization process with application to automatic text summarization. *AAAI Spring Symposium on Intelligent Text Summarization*, Stanford, CA, 1998.
11. D Bailey, N Chang, J Feldman, S Narayanan. Extending embodied lexical development. *Proceedings of CogSci98*, 1998, Hillsdale, NJ: Lawrence Erlbaum.
12. D Bailey, J Feldman, S Narayanan, G Lakoff. Embodied lexical development. *Proceedings of the 19th Annual Meeting of the Cognitive Science Society*, Stanford University Press, 1997.
13. B Baird, T Troyer, F Eeckman. Synchronization and grammatical inference in an oscillating Elman net. In: S Hanson, J Cowan, C Giles, eds. *Advances in Neural Information Processing Systems 5*, Morgan Kaufman, 1993.
14. J Barnden. Artificial intelligence and neural networks. In: M Arbib, ed. *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press, 1995.
15. J Barnden. Complex symbol-processing in *Composit*, a transiently localist connectionist architecture. In: R Sun, L Bookman, eds. *Computational Architectures Integrating Neural and Symbolic Processes*. Dordrecht: Kluwer, 1995.
16. R Baron. Knowledge extraction from neural networks: a survey. *Neurocolt Technical Report Series NC-TR-94-040*, 1995.
17. E Bates, J Elman. Connectionism and the study of change. In: M Johnson, ed. *Brain Development and Cognition*. Oxford: Blackwell, 1993.
18. E Bates, D Thal, V Marchman. Symbols and syntax: a Darwinian approach to language development. In: N Krasnegor, ed. *The Biological and Behavioural Determinants of Language Development*. Hillsdale, NJ: Lawrence Erlbaum, 1991.
19. E Baum. When are  $k$ -nearest neighbor and backpropagation accurate for feasible-sized set of examples? In: Hanson, G Drastal, R Rivest, eds. *Computational Learning Theory and Natural Learning Systems*. Cambridge, MA: MIT Press, 1994.
20. W Bechtel, A Abrahamsen. *Connectionism and the mind*. Oxford: Blackwell, 1991.
21. W Bechtel. Representations and cognitive explanations: assessing the dynamicist's challenge in cognitive science. *Cogn Sci* 22:295–318, 1998.
22. Y Bengio, P Simard, P Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans Neural Networks* 5:157–166, 1994.
23. Y Bengio. *Neural Networks for Speech and Sequence Recognition*. International Thomson Computer Press, 1996.
24. E Bienenstock, S Geman. Compositionality in neural systems. In: M Arbib, ed. *The Handbook of Brain Theory and Neural Networks*, Cambridge, MA: MIT Press, 1995.
25. L Birnbaum. Rigor mortis: a response to Nilsson's "Logic and artificial intelligence," *Artif Intell* 47:57–77, 1991.
26. C Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.
27. C Bishop. Classification and regression. In: E Fiesler, R Beale, eds. *Handbook of Neural Computation*. Oxford: Oxford University Press, 1997.
28. C Bishop. Generalization. In: E Fiesler, R Beck, eds. *Handbook of Neural Computation*. Oxford: Oxford University Press, 1997.
29. A Blair, J Pollack. Analysis of dynamical recognizers. *Neural Comput* 9:1127–1242, 1997.
30. D Blank, L Meeden, J Marshall. Exploring the symbolic/subsymbolic continuum: a case study of RAAM. In: J Dinsmore, ed. *The Symbolic and Connectionist Paradigms: Closing the Gap*. Hillsdale, NJ: Lawrence Erlbaum, 1992, pp 113–148.

31. A Blum, R Rivest. Training a 3-node neural network is NP-complete. Proceedings of the 1988 IEEE Conference on Neural Information Processing. Morgan Kaufmann, 1988.
32. U Bodenhausen, S Manke. A connectionist recognizer for on-line cursive handwriting recognition. Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, 1994.
33. U Bodenhausen, P Geutner, A Waibel. Flexibility through incremental learning: neural networks for text categorization. Proceedings of the World Congress on Neural Networks (WCNN), 1993.
34. L Bookman. A microfeature based scheme for modelling semantics. Proceedings of the International Joint Conference on Artificial Intelligence, 1987.
35. L Bookman. Trajectories through Knowledge Space. A Dynamic Framework for Machine Comprehension. Dordrecht: Kluwer, 1994.
36. N Bose, P Liang. Neural Network Fundamentals with Graphs, Algorithms, and Applications. New York: McGraw-Hill, 1996.
37. A Browne, ed. Neural Network Analysis, Architectures, and Applications. Institute of Physics Publishing, 1997.
38. T Burton. Introduction to Dynamic Systems Analysis. New York: McGraw-Hill, 1994.
39. M Casey. The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction. *Neural Comput* 8:1135–1178, 1996.
40. D Chalmers. Subsymbolic computation and the Chinese room. In: J Dunsmore, ed. *The Symbolic and Connectionist Paradigms: Closing the Gap*. Hillsdale, NJ: Lawrence Erlbaum, 1992.
41. D Chalmers. Syntactic transformations on distributed representations. *Connect Sci* 2: 53–62, 1990.
42. D Chalmers. A computational foundation for the study of cognition, TR-94-03, Philosophy/Neuroscience/Psychology Program, Washington University, 1994.
43. B Chandrasekaran, S Josephson. Architecture of intelligence: the problems and current approaches to solutions. In: V Honavar, L Uhr, eds. *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration*, Academic Press, 1994.
44. G Chappel, J Taylor. The temporal Kohonen map. *Neural Networks* 6:441–445, 1993.
45. N Chater, P Conkey. Finding linguistic structure with recurrent neural networks. Proceedings of the Fourteenth Annual Meeting of the Cognitive Science Society. Hillsdale, NJ: Lawrence Erlbaum, 1992.
46. N Chater, M Christiansen. Connectionism and natural language processing. In: S Garrod, M Pickering, eds. *Language Processing*. Psychology Press, 1999.
47. C Chen, V Honavar. A neural network architecture for syntax analysis. ISU CS-TR 95-18, Department of Computer Science, Iowa State University, 1995.
48. B Cheng, D Titterton. Neural networks: a review from a statistical perspective. *Stat Sci* 9:2–54, 1994.
49. N Chomsky. *Syntactic Structures*. The Hague: Mouton & Co., 1957.
50. N Chomsky. *Knowledge of Language*. Praeger, 1986.
51. M Christiansen. The (non)necessity of recursion in natural language processing. Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Lawrence Erlbaum, 1992.
52. M Christiansen. Language learning in the full, or why the stimulus might not be so poor after all. TR-94-12, Philosophy/Neuroscience/Psychology Program, Washington University, 1994.
53. M Christiansen, N Chater. Generalization and connectionist language learning. *Mind & Lang* 9:273–287, 1994.

54. M Christiansen, N Chater. Natural language recursion and recurrent neural networks, TR-94-13, Philosophy/Neuroscience/Psychology Program, Washington University, 1994.
55. M Christiansen, N Chater. Connectionist natural language processing: the state of the art. Special issue of *Cogn Sci on Connectionist Models of Human Language Processing: Progress and Prospects*. *Cognitive Science* 23, 1999, 417–437.
56. M Christiansen, N Chater. Toward a connectionist model of recursion in human linguistic performance. *Cognitive Science* 23, 1999, 157–205.
57. M Christiansen, J Devlin. Recursive inconsistencies are hard to learn: a connectionist perspective on universal word order correlations. Proceedings of the 19th Annual Cognitive Science Society Conference. Lawrence Erlbaum, 1997.
58. M Christiansen, M MacDonald. Processing of recursive sentence structure: testing predictions from a connectionist model. work in progress.
59. P Churchland. On the nature of explanation: a PDP approach. *Physica D* 42:281–292, 1990.
60. P Churchland, T Sejnowski. *The Computational Brain*. Cambridge, MA: MIT Press, 1992.
61. A Clark. Beyond eliminativism. *Mind & Lang* 4:251–279, 1989.
62. A Clark. *Microcognition: Philosophy, Cognitive Science, and Parallel Distributed Processing*. Cambridge, MA: MIT Press, 1989.
63. A Clark. Connectionism, competence, and explanation. *J Philos Sci* 41:195–222, 1990.
64. A Clark. *Associative Engines. Connectionism, Concepts, and Representational Change*. Cambridge, MA: MIT Press, 1993.
65. A Clark. Representational trajectories in connectionist learning. *Mind & Mach* 4:317–332, 1994.
66. A Clark. The dynamical challenge. *Cogn Sci* 21:461–481, 1997.
67. A Clark. *Being There. Putting Brain, Body, and World Together Again*. Cambridge, MA: MIT Press, 1997.
68. A Cleeremans, S Servan-Schreiber, J McClelland. Finite state automata and simple recurrent networks. *Neural Comput* 1:372–381, 1989.
69. D Clouse, G Cottrell. Regularities in a random mapping from orthography to semantics. Proceedings of the Twentieth Annual Cognitive Science Conference. Lawrence Erlbaum, 1998.
70. D Clouse, C Giles, B Horne, G Cottrell. Time-day neural networks: representation and induction of finite-state machines. *IEEE Trans Neural Networks* 8:1065–1070, 1997.
71. C Covic, P Munro. Learning to represent and understand locative prepositional phrases. Proceedings of the Tenth Annual Meeting of the Cognitive Science Society. Lawrence Erlbaum, 1998.
72. G Cottrell, S Small. A connectionist scheme for modelling word sense disambiguation. *Cogn & Brain Theory* 6: 1983.
73. G Cottrell. Connectionist parsing. Proceedings of the Seventh Annual Conference of the Cognitive Science Society, 1985, pp 201–212.
74. G Cottrell. *A Connectionist Approach to Word Sense Disambiguation*. Pitman, 1989.
75. G Cottrell, B Bartell, C Haupt. Grounding meaning in perception. Proceedings of the German Workshop on Artificial Intelligence, 1990.
76. G Cottrell, K Plunkett. Acquiring the mapping from meaning to sounds. *Connect Sci* 6:379–412, 1995.
77. M Craven, J Shavlik. Extracting tree-structured representations of trained networks. In: *Advances in Neural Information Processing Systems* 8. Cambridge, MA: MIT Press, 1996.
78. S Das, C Giles, G Sun. Capabilities and limitations of a recurrent neural network with an external stack memory in learning context-free grammars. Proceedings of the

- International Joint Conference on Neural Networks. China: Publishing House of Electronics Industry, 1992, vol 3.
79. S Das, C Giles, G Sun. Learning context-free grammars: capabilities and limitations of a recurrent neural network with an external stack memory. Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society. Lawrence Erlbaum, 1992.
  80. S Das, C Giles, G Sun. Using prior knowledge in an NNPD to learn context-free languages. In: C Giles, S Hanson, J Cowan, eds. *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann, 1993.
  81. S Das, M Mozer. Dynamic on-line clustering and state extraction: an approach to symbolic learning. *Neural Networks* 11:53–64, 1998.
  - 81a. B DasGupta, H Siegelmann, E Sontag. On the complexity of training neural networks with continuous activation functions. *IEEE Trans Neural Networks* 6:1490–1504, 1995.
  82. K Daugherty, M Seidenberg. Rules or connections? The past tense revisited. Proceedings of the Fourteenth Annual Meeting of the Cognitive Science Society. Lawrence Erlbaum, 1992.
  83. K Daugherty, M Hare. What's in a rule? The past tense by some other name might be called a connectionist net. Proceedings of the 1993 Connectionist Models Summer School. Lawrence Erlbaum, 1993.
  84. M de Francesco. Modular topologies. In: E Fiesler, R Beck, eds. *Handbook of Neural Computation*. Oxford: Oxford University Press, 1997.
  85. G Dell. A spreading activation theory of retrieval in language production. *Psychol Rev* 93:283–321, 1986.
  86. G Dell. The retrieval of phonological forms in production: tests of predictions from a connectionist model. *J Mem Lang* 27:124–142, 1988.
  87. G Dell, L Burger, W Svec. Language production and serial order: a functional analysis and a model. *Psychol Rev* 104:123–147, 1997.
  88. R Devaney. *An Introduction to Chaotic Dynamical Systems*. 2nd ed. Addison–Wesley, 1989.
  89. J Dinsmore, ed. *The Symbolic and Connectionist Paradigms: Closing the Gap*. Lawrence Erlbaum, 1992.
  90. G Dorffner. Taxonomies and part-whole hierarchies in the acquisition of word meaning—a connectionist model. Proceedings of the 14th Annual Conference of the Cognitive Science Society. Lawrence Erlbaum, 1992.
  91. G Dorffner. Radical connectionism for natural language processing. TR-91-07, Oesterreichisches Forschungsinstitut fuer Artificial Intelligence, Wien, 1991.
  92. G Dorffner. A step toward sub-symbolic language models without linguistic representations. In: R Reilly, N Sharkey, eds.. *Connection Approaches to Natural Language Processing*. Hillsdale, NJ: Lawrence Erlbaum, 1992.
  93. G Dorffner. *Neural Networks and a New Artificial Intelligence*. Thomson Computer Press, 1997.
  94. G Dorffner. Radical connectionism—a neural bottom-up approach to AI. In: *Neural Networks and a New Artificial Intelligence*. Thomson Computer Press, 1997.
  95. G Dorffner. A radical view on connectionist language modelling. In G Dorffner, ed. *Konnektionismus in Artificial Intelligence und Kognitionsforschung*. Berlin: Springer, 1990.
  96. G Dorffner, E Prem, H Trost. Words, symbols, and symbol grounding. TR-93-30, Oesterreichisches Forschungsinstitut fuer Artificial Intelligence, Wien, 1993.
  97. G Dorffner, E Prem. Connectionism, symbol grounding, and autonomous agents. Proceedings of the 15th Annual Conference of the Cognitive Science Society, 1993.
  98. M Dyer, V Nenov. Language learning via perceptual/motor experiences. Proceedings of the 15th Annual Conference of the Cognitive Science Society. Lawrence Erlbaum, 1993.

99. M Dyer. Symbolic neuroengineering for natural language processing: a multilevel research approach. In: J Barnden, J Pollack, eds. *Advances in Connectionist and Neural Computation Theory 1*, Ablex, 1991.
100. M. Dyer. Grounding language in perception. In: V Honavar, L Uhr, eds. *Artificial Intelligence and Neural Networks: Steps Toward Principle Integration*. New York: Academic Press, 1994.
101. M Dyer. Connectionist natural language processing: a status report. In: R Sun, L Bookman, eds. *Computational Architectures Integrating Neural and Symbolic Processes*. Dordrecht: Kluwer, 1995.
102. S El Hahi, Y Bengio. Hierarchical recurrent neural networks for long-term dependencies. In: M Mozer, DS Touretzky, M Perrone, eds. *Advances in Neural Information Processing Systems 8*. Cambridge, MA: MIT Press, 1996.
103. S Ellacott, D Bose. *Neural Networks: Deterministic Methods of Analysis*. International Thomson Computer Press, 1996.
104. J Elman. Finding structure in time. *Cogn Sci* 14:172–211, 1990.
105. J Elman. Distributed representation, simple recurrent networks, and grammatical structure. *Mach Learn* 7:195–225, 1991.
106. J Elman. Learning and development in neural networks: the importance of starting small. *Cognition* 48:71–99, 1993.
107. J Elman, Language as a dynamical system. In: R Port, T van Gelder, eds. *Mind as Motion: Explorations in the Dynamics of Cognition*, MIT Press, 1995.
108. J Elman, E Bates, M Johnson, A Karmiloff-Smith, D Parisi, K Plunkett. *Rethinking Innateness. A Connectionist Perspective on Development*. Cambridge, MA: MIT Press, 1996.
109. M Fandy. Context-free parsing in connectionist networks. TR174, Department of Computer Science, University of Rochester, 1985.
110. J Feldman. Structured connectionist models and language learning. *Artif Intell Rev* 7:301–312, 1993.
111. J Feldman, D Ballard. Connectionist models and their properties. *Cogn Sci* 6:205–254, 1982.
112. J Feldman, G Lakoff, A Stolcke, W Hollbach, S Weber. Miniature language acquisition: a touchstone for cognitive science. TR-90-009, ICSI, Berkeley, 1990.
113. J. Feldman, G Lakoff, D Bailey, S Narayanan. T Regier, A Stolcke.  $L_0$ —the first five years of an automated language acquisition project. *Artif Intell Rev* 10: 103–129, 1996.
114. E Fiesler, R Beale. *Handbook of Neural Computation*. Oxford University Press, 1997.
115. S Finch, N Chater. Learning syntactic categories: a statistical approach. In: M Oaksford, G Brown, eds. *Neurodynamics and Psychology*. New York: Academic Press, 1993.
116. J Fodor, Z Pylyshyn. Connectionism and cognitive architecture: a critical analysis. *Cognition* 28:3–71, 1988.
117. J Fodor, B McLaughlin. Connectionism and the problem of systematicity: why Smolensky's solution doesn't work. In: T Horgan, J Tienson, eds. *Connectionism and the Philosophy of Mind*. Kluwer, 1991.
118. J Fodor, B McLaughlin, Connectionism and the problem of systematicity: why Smolensky's solution won't work. *Cognition* 35:183–204, 1990.
119. J Fodor. Connectionism and the problem of systematicity (continued): why Smolensky's solution still doesn't work. *Cognition* 62:109–119, 1997.
120. S Franklin, M Garzon. Computation in discrete neural nets. In: P Smolensky, et al., eds. *Mathematical Perspectives on Neural Networks*. Hillsdale, NJ: Lawrence Erlbaum, 1996.

121. P Frasconi, M Gori, G Soda. Injecting nondeterministic finite state automata into recurrent networks. Technical Report, Dipartimento di Sistemi e Informatica, Firenze, 1993.
122. P Frasconi, M Gori, G Soda. Recurrent neural networks and prior knowledge for sequence processing: a constrained nondeterministic approach. *Know Based Syst* 8: 313–322, 1995.
123. P Frasconi, M Gori, M Maggini, G Soda. Representation of finite state automata in recurrent radial basis function networks. *Mach Learn* 23:5–32, 1996.
124. S Gallant. *Neural Network Learning and Expert Systems*. Cambridge, MA: MIT Press, 1993.
125. P Gallinari. Modular neural net systems, training of. In: M Arbib, ed. *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press, 1995.
126. A Galton. The Church–Turing thesis: its nature and status. *AISB Q* 74:9–19, 1990.
127. M Gasser. A connectionist model of sentence generation in a first and second language, TR UCLA-AI-88-13, University of California, 1988.
128. M Gasser. Transfer in a connectionist model of the acquisition of morphology. In: H Baayen, R Schroeder, eds. *Yearbook of Morphology 1996*. Dordrecht: Foris, 1997.
129. M Gasser. Acquiring receptive morphology: a connectionist model. *Proceedings of ACL 94*, 1994.
130. C Giles, D Chen, C Miller, H Chen, G Sun, Y Lee. Learning and extracting finite state automata with second-order recurrent neural networks. *Neural Comput* 4:393–405, 1992.
131. C Giles, C Omlin. Inserting rules into recurrent neural networks. In: S Kung, F Fallside, J Sorenson, eds. *Neural networks for Signal Processing II, Proceedings of the 1992 IEEE Workshop*. IEEE Press, 1992.
132. C Giles, C Miller, D Chen, G Sun, H Chen, Y Lee. Extracting and learning an unknown grammar with recurrent neural networks. In: J Moody, S Hanson, R Lippmann, eds. *Advances in Neural Information Processing Systems 4*. Morgan Kaufmann, 1992.
133. C Giles, C. Omlin. Rule refinement with recurrent neural networks. *IEEE International Conference on Neural Networks*, 1993.
134. C Giles, C Omlin. Extraction, insertion, and refinement of symbolic rules in dynamically driven recurrent neural networks. *Connect Sci* 5:307–337, 1993.
135. C Giles, B Horne, T Lin. Learning a class of large finite state machines with a recurrent neural network, *Neural Networks* 8:1359–1365, 1995.
136. C Giles, G Sun, H Chen, Y Lee, D Chen. Second-order recurrent neural networks for grammatical inference. *Proceedings of the International Joint Conference on Neural Networks*, 1991.
137. F Girosi, T Poggio. Networks and the best approximation property. *Biol Cybernet* 63: 169–176, 1990.
138. E Gold. Language identification in the limit. *Inform Control* 10:447–474, 1967.
139. R Golden. A developmental neural model of visual word perception. *Cogn Sci* 10:241–276, 1986.
140. R Golden. *Mathematical Methods for Neural Network Analysis and Design*. Cambridge, MA: MIT Press, 1996.
141. S Goonatillake, S Khebbal. *Intelligent Hybrid Systems*. New York: Wiley, 1994.
142. M Gori, M Maggini, G Soda. Learning finite state grammars from noisy examples using recurrent neural networks. *NEURAP'96*, 1996.
143. R Hadley. Systematicity in connectionist language learning. *Mind Lang* 9:247–273, 1994.
144. R Hadley, V Cardei. Acquisition of the active-passive distinction from sparse input and no error feedback, CSS-IS-TR97-01, School of Computer Science and Cognitive Science Program, Vancouver, BC: Simon Fraser University, 1997.



145. S Hanson, J Kegl. PARSNIP: a connectionist network that learns natural language grammar from exposure to natural language sentences. Proceedings of the Ninth Annual Conference of the Cognitive Science Society. Lawrence Erlbaum, 1987.
146. S Hanson, G Drastal, R Rivest. Computational Learning Theory and Natural Learning Systems. Cambridge, MA: MIT Press, 1994.
147. M Hare, J Elman. A connectionist account of English inflectional morphology: evidence from language change. Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society, 1992.
148. M Hare, J Elman. Learning and morphological change. *Cognition* 56:61–98, 1995.
149. M Hare, J Elman, K Daugherty. Default generalization in connectionist networks. *Lang Cogn Processes* 10:601–630, 1995.
150. T Harley. Phonological activation of semantic competitors during lexical access in speech production. *Lang Cogn Processes* 8:291–309, 1993.
151. S Harnad. The symbol grounding problem. *Physica D* 42:335–346, 1990.
152. S Harnad. Connecting object to symbol in modeling cognition. In: A Clark, R Lutz, eds. *Connectionism in Context*. Springer, 1992.
153. S Harnad. Problems, problems: the frame problem as a symptom of the symbol grounding problem. *Psychology* 4, 1993.
154. S Harnad. Grounding symbols in the analog world with neural nets—a hybrid model. *Think* 2:12–78, 1993.
155. S Harnad. Symbol grounding is an empirical problem: neural nets are just a candidate component. Proceedings of the Fifteenth Annual Meeting of the Cognitive Science Society. Lawrence Erlbaum, 1993.
156. S Harnad. Grounding symbolic capacity in robotic capacity. In: L Steels, R Brooks, eds. The “Artificial Life” Route to “Artificial Intelligence.” *Building Situated Embodied Agents*. Lawrence Erlbaum, 1995.
157. S Harnad. Does the mind piggyback on robotic and symbolic capacity. In: H Morowitz, J Singer, eds. *The Mind, the Brain, and Complex Adaptive Systems*. Addison-Wesley, 1994.
158. S Harnad. Computation is just interpretable symbol manipulation; cognition isn't. *Minds & Mach* 4:379–390, 1995.
159. P Haugeland. *Artificial Intelligence: The Very Idea*. Cambridge, MA: MIT Press, 1985.
160. S Haykin. *Neural Networks: A Comprehensive Foundation*. New York: Macmillan, 1994.
161. J Henderson. Connectionist syntactic parsing using temporal variable binding. *J Psycholinguist Res* 23:353–379, 1994.
162. J Henderson, P Lane. A connectionist architecture for learning to parse. Proceedings of 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL '98), University of Montreal, Canada, 1998.
163. J Henderson. Constituency, context, and connectionism in syntactic parsing. In: M Crocker, M Pickering, C Clifton, eds. *Architectures and Mechanisms for Language Processing*. Cambridge University Press, to appear.
164. J Hertz, A Krogh, R Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, 1991.
165. J Hertz. Computing with attractors. In: M Arbib, ed. *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press, 1995.
166. R Hilborn. *Chaos and Nonlinear Dynamics*. Oxford: Oxford University Press, 1994.
167. G Hinton. Implementing semantic networks in parallel hardware. In: G Hinton, J Anderson, eds. *Parallel Models of Associative Memory*. Lawrence Erlbaum, 1981.

168. S Hochreiter, J Schmidhuber. Recurrent neural net learning and vanishing gradient. Proceedings of the Fuzzy-Neuro Workshop, Soest, Germany, 1997.
169. V Honavar, L Uhr. Artificial Intelligence and Neural Networks: Steps Toward Principled Integration. New York: Academic Press, 1994.
170. V Honavar. Toward learning systems that integrate different strategies and representations. In: V Honavar, L Uhr, eds. Artificial Intelligence and Neural Networks: Steps Toward Principled Integration. New York: Academic Press, 1994.
171. V Honavar. Symbolic artificial intelligence and numeric neural networks: towards a resolution of the dichotomy. In: R Sun, L Bookman, eds. Computational Architectures Integrating Neural and Symbolic Processes. Dordrecht: Kluwer, 1995.
172. T Horgan, J Tienson, eds. Connectionism and the Philosophy of Mind. Dordrecht: Kluwer, 1991.
173. B Horne, D Hush. Bounds on the complexity of recurrent neural network implementations of finite state machines. *Neural Networks* 9:243–252, 1996.
174. T Howells. VITAL—a connectionist parser. Proceedings of the Tenth Annual Conference of the Cognitive Science Society, 1988.
175. G Huang, H Babri. Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. *IEEE Trans Neural Networks* 7:1329–1338, 1996.
176. S Jackson, N Sharkey. Grounding computational engines. *Artif Intell Rev* 10:65–82, 1996.
177. H Jaeger. From continuous dynamics to symbols. Proceedings of the First Joint Conference on Complex Systems in Psychology: Dynamics, Synergetics, and Autonomous Agents, Gstaad, Switzerland, 1997.
178. A Jain. Parsing complex sentences with structured connectionist networks. *Neural Comput* 3:110–120, 1991.
179. A Jain, A Waibel. Parsing with connectionist networks. In: M Tomita, ed. *Current Issues in Parsing Technology*. Kluwer, 1991.
180. A Jain, A Waibel. Incremental parsing in modular recurrent connectionist networks. *International Joint Conference on Neural Networks*, 1991.
181. D James, R Miikkulainen. SARDNET: a self-organizing feature map for sequences. In: G Tesauro, D Touretzky, T Leen, eds. *Advances in Neural Processing Systems* 7, 1995.
182. K Jim, C Giles, B Horne. Synaptic noise in dynamically driven recurrent neural networks: convergence and generalization, UMIACS-TR-94-89, Institute for Advanced Computer Studies, University of Maryland, 1994.
183. W Joerding, J Meador. Encoding a priori information in feedforward networks. *Neural Networks* 4:847–856, 1991.
184. M Jordan, R Jacobs. Modular and hierarchical learning systems. In: M Arbib, ed. *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press, 1995.
185. M Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. Proceedings of the Eighth Conference of the Cognitive Science Society, 1986.
186. M Jordan. Neural networks. In: A Tucker ed. *CRC Handbook of Computer Science*. Boca Raton, FL: CRC Press, 1996.
187. JS Judd. Complexity of learning. In: G Smolensky, et al. *Mathematical Perspectives on Neural Networks*. Hillsdale, NJ: Lawrence Erlbaum, 1996.
188. JS Judd. Time complexity of learning. In: M Arbib, ed. *Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT, Press, 1995.
189. A Karmiloff-Smith. Nature, nurture, and PDP: preposterous developmental predicates? *Connect Sci* 4:253–270, 1992.
190. A Kawamoto. Distributed representations of ambiguous words and their resolution in a connectionist network. In: S Small, G Cottrell, M Tanenhaus, eds. *Lexical Ambiguity Resolution*. Morgan Kaufmann, 1989.

191. S Kelso. *Dynamic Patterns*. Cambridge, MA: MIT Press, 1995.
192. D Kirsch. Foundations of AI: the big issues. *Artif Intell* 47:3–30, 1991.
193. J Kolen. The origin of clusters in recurrent network state space. *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, 1994.
194. J Kolen. Fool's gold: extracting finite state machines from recurrent network dynamics. In: J Cowan, G Tesauro, J Alspector, eds. *Advances in Neural Information Processing Systems 6*. Morgan Kaufmann, 1994.
195. J Kolen. Recurrent networks: state machines or iterated function systems? In: M Mozer, P Smolensky, D Touretzky, J Elman, A Weigend, eds. *Proceedings of the 1993 Connectionist Models Summer School*. Lawrence Erlbaum, 1994.
196. J Kolen, J Pollack. The observer's paradox: apparent computational complexity in physical systems. *J of Exp Theor Artif Intell* 7:253–269, 1995.
197. K Kukich. Where do phrases come from: some preliminary experiments in connectionist phrase generation. In: G Kempen, ed. *Natural Language Generation*. Kluwer, 1987.
198. S Kwasny, K Faisal. Connectionism and determinism in a syntactic parser. *Connect Sci* 2:63–82, 1990.
199. S Kwasny, K Faisal. Symbolic parsing via subsymbolic rules. In: J Dinsmore, ed. *The Symbolic and Connectionist Paradigms: Closing the Gap*. Lawrence Erlbaum, 1992.
200. S Kwasny, S Johnson, B Kalman. Recurrent natural language parsing. *Proceedings of the Sixteenth Annual Meeting of the Cognitive Science Society*, Lawrence Erlbaum, 1994.
201. R Lacher, K Nguyen. Hierarchical architectures for reasoning. In: R Sun, L Bookman, eds. *Computational Architectures Integrating Neural and Symbolic Processes*. Kluwer, 1995.
202. T Lange, M Dyer. High-level inferencing in a connectionist network. *Connect Sci* 1: 181–217, 1989.
203. T Lange. A structured connectionist approach to inferencing and retrieval. In: R Sun, L Bookman, eds. *Computational Architectures Integrating Neural and Symbolic Processes*. Kluwer, 1995.
204. S Lawrence, C Giles, S Fong. On the applicability of neural network and machine learning methodologies to natural language processing. UMIACS-TR-95-64. Institute for Advanced Computer Studies, University of Maryland, 1995.
205. S Lawrence, C Giles, S Fong. Natural language grammatical inference with recurrent neural networks. *IEEE Trans Knowl Data Eng* in press.
206. S Lawrence, A Tsoi, A Back. Function approximation with neural networks and local methods: bias, variance, and smoothness. *Australian Conference on Neural Networks, ACNN 96*, Australian National University, 1996.
207. S Lawrence, C Giles, S Fong. Can recurrent neural networks learn natural language grammars? *Proceedings of the International Conference on Neural Networks, ICNN 96*, 1996.
208. S Lawrence, C Giles, A Tsoi. What size network gives optimal generalization? Convergence properties of backpropagation, UMIACS-TR-96-22, Institute for Advanced Computer Studies, University of Maryland, 1996.
209. S Lawrence, C Giles, A Tsoi. Lessons in neural network training: overfitting may be harder than expected. *Proceedings of the Fourteenth Annual Conference on Artificial Intelligence, AAAI-97*, AAAI Press, 1997.
210. S Lawrence, S Fong, C Giles. Natural language grammatical inference: a comparison of recurrent neural networks and machine learning methods. In: S Wermter, et al., eds. *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*. Springer, 1996.

211. G Lee, M Flowers, M Dyer. Learning distributed representations for conceptual knowledge and their application to script-based story processing. *Connect Sci* 2:313–346, 1990.
212. W Lehnert. Possible implications of connectionism. *Theoretical Issues in Natural Language Processing*. University of New Mexico, 1986.
213. D Levine, M Apariciov. *Neural Networks for Knowledge Representation and Inference*. Hillsdale, NJ: Lawrence Erlbaum, 1994.
214. T Lin, B Horne, C Giles. How embedded memory in recurrent neural network architectures helps learning long-term temporal dependencies. UMIACS-TR-96-28. Institute for Advanced Computer Studies, University of Maryland, 1996.
215. S Lucas. New directions in grammatical inference. In: *Grammatical Inference: Theory, Applications, and Alternatives*. IEEE, 1993.
216. J McClelland, D Rumelhart. An interactive activation model of context effects in letter perception: Part I. An account of basic findings. *Psychol Rev.* 88:375–407, 1981.
217. J McClelland, A Kawamoto. Mechanisms of sentence processing: assigning roles to constituents. In: D Rumelhart, J McClelland, eds. *Parallel Distribution Processing*. Cambridge, MA: MIT Press, 1986.
218. T McKenna. The role of interdisciplinary research involving neuroscience in the development of intelligent systems: In: V Honavar, L Uhr, eds. *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration*. New York: Academic Press, 1994.
219. B MacWhinney, J Leinbach. Implementations are not conceptualizations: revising the verb learning model. *Cognition* 40 (1991), 121–157.
220. G Marcus. The acquisition of the English past tense in children and multilayered connectionist networks. *Cognition* 56:271–279, 1997.
221. P Manolios, R Fanelli. First-order recurrent neural networks and deterministic finite state automata. *Neural Comput* 6:1155–1173, 1994.
222. D Medler. A brief history of connectionism. *Neural Comput Surv* 1:61–101, 1998.
223. R Miikkulainen, M Dyer. A modular neural network architecture for sequential paraphrasing of script-based stories. TR UCLA-AI-89-02, University of California, 1989.
224. R Miikkulainen, M Dyer. Natural language processing with modular PDP networks and distributed lexicon. *Cogn Sci* 15:343–399, 1991.
225. R Miikkulainen. *Subsymbolic Natural Language Processing: An Integrated Model of Scripts, Lexicon, and Memory*. Cambridge, MA: MIT Press, 1993.
226. R Miikkulainen. Integrated connectionist models: building AI systems on subsymbolic foundations. In: V Honavar, L Uhl, eds. *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration*. New York: Academic Press, 1994.
227. R Miikkulainen. Subsymbolic parsing of embedded structures. In: R Sun, L Bookman, eds. *Computational Architectures. Integrating Neural and Symbolic Processes*, Kluwer, 1995.
228. R Miikkulainen. Subsymbolic case–role analysis of sentences with embedded clauses. *Cogn Sci* 20:47–73, 1996.
229. M Minsky, S Papert. *Perceptrons*. Cambridge, MA: MIT Press, 1969.
230. H Moisl. Connectionist finite state natural language processing. *Connect Sci* 4:67–91, 1992.
231. M Mozer, S Das. A connectionist symbol manipulator that discovers the structure of context-free languages. In: S Hanson, J Cowan, C Giles, eds. *Advances in Neural Information Processing Systems*. Morgan Kaufmann, 1993.
232. M Mozer. Neural net architectures for temporal sequence processing. In: A Weigend, N Gershenfeld, eds. *Predicting the Future and Understanding the Past*. Addison-Wesley, 1994.

233. P Munro, C Cosic, M Tabasko. A network for encoding, decoding, and translating locative prepositions. *Connect Sci* 3:225–240, 1991.
234. R Nakisa, U Hahn. Where defaults don't help: the case of the German plural system. *Proceedings of the Sixteenth Annual Meeting of the Cognitive Science Society*. Lawrence Erlbaum, 1996.
235. S Narayanan. Talking the talk *is* like walking the walk: a computational model of verbal aspect. *Proceedings of the Annual Conference of the Cognitive Science Society*, Stanford, 1997.
236. S Narayanan. Moving right along: a computational model of metaphoric reasoning about events. *Proceedings of the National Conference on Artificial Intelligence*, AAAI Press, 1999.
237. S Narayanan. Embodiment in language understanding: modeling the semantics of causal narratives. *AAAI 1996 Fall Symposium on Embodied Cognition and Action*. AAAI Press, 1996.
238. V Nenov, M Dyer. Perceptually grounded language learning: Part 1—a neural network architecture for robust sequence association. *Connect Sci* 5:115–138, 1993.
239. V Nenov, M Dyer. Perceptually grounded language learning: Part 2—DETE: a neural/procedural model. *Connect Sci* 6:3–41, 1994.
240. A Newell. Physical symbol systems. *Cogn Sci* 4:135–183, 1980.
241. L Niklasson, T van Gelder. On being systematically connectionist. *Mind Lang* 9:288–302, 1994.
242. L Niklasson, N Sharkey. Systematicity and generalization in compositional connectionist representations. In: G Dorffner, ed. *Neural Networks and a New Artificial Intelligence*. Thomson Computer Press, 1997.
243. A Norton. Dynamics: an introduction. In: R Port, T van Gelder, eds. *Mind as Motion. Explorations in the Dynamics of Cognition*. Cambridge, MA: MIT Press, 1995.
244. S Olafsson. Some dynamical properties of neural networks. In: R Linggard, D Myers, C Nightingale, eds. *Neural Networks for Vision, Speech, and Natural Language*. Chapman & Hall, 1992.
245. C Omlin, C Giles. Training second-order recurrent neural networks using hints. In: D Sleeman, P Edwards, eds. *Machine Learning: Proceedings of the Ninth International Conference*. Morgan Kaufmann, 1992.
246. C Omlin, C Giles. Constructing deterministic finite-state automata in recurrent neural networks. *J ACM* 43:937–972, 1996.
247. C Omlin, C Giles. Fault-tolerant implementation of finite-state automata in recurrent neural networks. *RPI Computer Science Technical Report 95-3*, 1995.
248. C Omlin. Stable encoding of large finite state automata in recurrent neural networks with sigmoid discriminants. *Neural Comput* 8:675–696, 1996.
249. C Omlin, C Giles. Extraction and insertion of symbolic information in recurrent neural networks. In: V Honavar, L Uhr, eds. *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration*. New York: Academic Press, 1994.
250. C Omlin, C Giles. Extraction of rules from discrete-time recurrent neural networks. *Neural Networks* 9:41–52, 1996.
251. R Pfeifer, P Verschure. Complete autonomous systems: a research strategy for cognitive science. In: G Dorffner, ed. *Neural Networks and a New Artificial Intelligence*. Thomson Computer Press, 1997.
252. S Pinker, A Prince. On language and connectionism: analysis of a parallel distributed processing model of language acquisition. *Cognition* 28:73–193, 1988.
253. S Pinker. *The Language Instinct*. William Morrow, 1994.
254. T Plate. Holographic recurrent networks. In: C Giles, S Hanson, J Cowan, eds. *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann, 1993.

255. T Plate. Holographic reduced representations. *IEEE Trans Neural Networks* 6:623–641, 1995.
256. T Plate. A common framework for distributed representation schemes for compositional structure. In: F Maire, R Hayward, J Diederich, eds. *Connectionist Systems for Knowledge Representation and Deduction*. Queensland University of Technology, 1997.
257. D Plaut. Semantic and associative priming in a distributed attractor network. *Proceedings of the 17th Annual Conference of the Cognitive Science Society*. Lawrence Erlbaum, 1995.
258. K Plunkett, V Marchman. U-shaped learning and frequency effects in a multi-layered perceptron: implications for child language acquisition. *Cognition* 38:43–102, 1991.
259. K Plunkett, V Marchman. From rote learning to system building: acquiring verb morphology in children and connectionist nets. *Cognition* 48:21–69, 1993.
260. K Plunkett. Language acquisition. In M Arbib, ed. *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press, 1995.
261. J Pollack. No harm intended: a review of the *Perceptrons* expanded edition. *J Math Psychol* 33:358–365, 1989.
262. J Pollack. Implications of recursive distributed representations. In: D Touretzky, ed. *Advances in Neural Information Processing Systems 1*. Morgan Kaufmann, 1989.
263. J Pollack. Recursive distributed representations. *Artif Intell* 46:77–105, 1990.
264. J Pollack. Connectionism: past, present, and future. *Artif Intell Rev* 3:3–20; 1989.
265. J Pollack. The induction of dynamical recognizers. *Mach Learn* 7:227–252, 1991.
266. S Porat, J Feldman. Learning automata from ordered examples. *Mach Learn* 7:109–138, 1991.
267. R Port, T van Gelder. *Mind as Motion. Explorations in the Dynamics of Cognition*. Cambridge, MA: MIT Press, 1995.
268. S Prasada, S Pinker. Similarity-based and rule-based generalizations in inflectional morphology. *Lang Cogn Processes* 8:1–56, 1993.
269. L Pratt. Experiments on the transfer of knowledge between neural networks. In: S Hanson, et al., eds. *Computational Learning Theory and Natural Learning Systems*. Cambridge MA: MIT Press, 1994.
270. J Rager. Self-correcting connectionist parsing. In: R Reilly, N Sharkey, eds. *Connectionist Approaches to Natural Language Processing*. Hillsdale, NJ: Lawrence Erlbaum, 1992.
271. R Reed, R Marks. Neurosmoothing: improving neural network learning. In: M Arbib, ed. *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press, 1995.
272. T Regier. Learning object-relative spatial concepts in the  $L_0$  project. *Proceedings of the Thirteenth Annual Meeting of the Cognitive Science Society*, 1991.
273. T Regier. Learning perceptually grounded semantics in the  $L_0$  project. *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, 1991.
274. T Regier. *The Human Semantic Potential*. Cambridge, MA: MIT Press, 1996.
275. R Reilly. A connectionist model of some aspects of anaphor resolution. *Proceedings of the Tenth Annual Conference on Computational Linguistics*, 1984.
276. R Reilly, N Sharkey. *Connectionist Approaches to Natural Language Processing*. Hillside, NJ: Lawrence Erlbaum, 1992.
277. B. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.
278. R Rojas. *Neural Networks: A Systematic Introduction*. Springer, 1996.
279. D Rumelhart, J McClelland. An interactive activation model of context effects in letter perception: Part 2. the contextual enhancement effects and some tests and enhancements of the model. *Psychol Rev* 89:60–94, 1982.



280. D Rumelhart, J McClelland. On learning the past tenses of English verbs. In: D Rumelhart, J McClelland, eds. *Parallel Distributed Processing*, vol 2. Cambridge, MA: MIT Press, 1986.
281. D Rumelhart, J McClelland. *Parallel Distributed Processing*, 2 vols. Cambridge, MA: MIT Press, 1986.
282. D Rumelhart, D Zipser. Feature discovery by competitive learning. In: D Rumelhart, J McClelland, eds. *Parallel Distributed Processing*. Vol 1. Cambridge, MA: MIT Press, 1986.
283. N Sales, R Evans, I Aleksander. Successful naive representation grounding. *Artif Intell Rev* 10:83–102, 1996.
284. W Sarle. Neural networks and statistical models. *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, 1994.
285. F Scarselli, A Tsoi. Universal approximation using feedforward neural networks: a survey of some existing methods, and some new results. *Neural Networks* 11:15–37, 1998.
286. G Scheler. Generating English plural determiners from semantic representations: a neural network learning approach. In: S Wermter, et al., eds. *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*. Springer, 1996.
287. G Scheler. With raised eyebrows or the eyebrows raised? A neural network approach to grammar checking for definiteness. *Proceedings of NEMLAP-2*, Ankara, Turkey, 1996.
288. G Scheler. Computer simulation of language acquisition: a proposal concerning early language learning in a micro-world. Technical Report FKI-214-95, Technische Universität München, Institut für Informatik, 1995.
289. G Scheler, N Fertig. Constructing semantic representations using the MDL principle. *Proceedings of HELNET'97*, Montreux, Switzerland, 1997.
290. G Scheler. Learning the semantics of aspect. In: H Somers, D Jones, eds. *New Methods in Language Processing*. University College London Press, 1996.
291. J Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Comput* 4:234–242, 1992.
292. J Schmidhuber, S Heil. Sequential neural text compression. *IEEE Trans Neural Networks* 7:142–146, 1996.
293. M Seidenberg, J McClelland. A distributed, developmental model of word recognition and naming. *Psychol Rev* 96:523–568, 1989.
294. B Selman. Rule-based processing in a connectionist system for natural language understanding, TR CSRI-168, Computer Systems Research Institute, University of Toronto, 1985.
295. B Selman, G Hirst. A rule-based connectionist parsing system. *Proceedings of the Seventh Annual Meeting of the Cognitive Science Society*. Lawrence Erlbaum, 1985.
296. B Selman. Connectionist systems for natural language understanding. *Artif Intell Rev* 3: 23–31, 1989.
297. D Servan-Schreiber, A Cleeremans, J McClelland. Learning sequential structure in simple recurrent networks. In: D Touretzky, eds. *Advances in Neural Information Processing Systems 1*. Morgan Kaufmann, 1989.
298. D Servan-Schreiber, A Cleeremans, J McClelland. Graded state machines: the representation of temporal contingencies in simple recurrent networks. *Mach Learn* 7:161–193, 1991.
299. N Sharkey. Connection science and natural language: an emerging discipline. *Connect Sci* 2: 1990.
300. N Sharkey. Connectionist representation techniques. *Artif Intell Rev* 5: 143–167, 1991.
301. N Sharkey, ed. *Connectionist Natural Language Processing*. Kluwer, 1992.
302. N Sharkey, A Sharkey. Adaptive generalisation. *Artif Intell Rev* 7: 313–328, 1993.

303. N Sharkey, A Sharkey. Separating learning and representations. In: S Wermter, et al., eds. *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*. Springer, 1996.
304. N Sharkey, S Jackson. Three horns of the representational trilemma. In: V Honavar, L Uhr, eds. *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration*, New York Academic Press, 1994.
305. L Shastri. Structured connectionist models. In: M Arbib, ed. *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press, 1995.
306. L Shastri, V Ajjanagadde. From simple associations to systematic reasoning: a connectionist representation of rules, variables, and dynamic bindings using temporal synchrony. *Behav Brain Sci* 16:417–494, 1993.
307. J Shavlik. Learning by symbolic and neural methods. In: M Arbib, ed. *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press, 1995.
308. H Siegelman, E Sontag. On the computational power of neural nets. *J Comput Syst Sci* 50:132–150, 1995.
309. S Small, G Cottrell, L Shastri. Towards connectionist parsing. *Proceedings of the National Conference on Artificial Intelligence*, 1982.
310. P Smolensky. On the proper treatment of connectionism. *Behav Brain Sci* 11:1–74, 1988.
311. P Smolensky. The constituent structure of connectionist mental states: a reply to Fodor and Pylyshyn. In: T Horgan, J Tienson, eds. *Connectionism and the Philosophy of Mind*. Kluwer, 1991.
312. P Smolensky. Connectionism, constituency, and the Language of Thought. In: B Loewer, G Rey, eds. *Meaning and Mind. Fodor and his Critics*. Blackwell, 1991.
313. P Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artif Intell* 46:159–216, 1990.
314. P Smolensky, G Legendre, Y Miyata. Principles for an integrated connectionist/symbolic theory of higher cognition. TR-CU-CS-600-92, Computer Science Department, University of Colorado at Boulder, 1992.
315. P Smolensky. Harmonic grammars for formal languages. In: S Hanson, J Cowan, C Giles, eds. *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann, 1993.
316. P Smolensky, G Legendre, Y Miyata. Integrating connectionist and symbolic computation for the theory of language. In: V Honavar, L Uhr, eds. *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration*. New York: Academic Press, 1994.
317. P Smolensky, M Mozer, D Rumelhart. *Mathematical Perspectives on Neural Networks*. Lawrence Erlbaum, 1996.
318. E Sontag. Automata and neural networks. In: M Arbib, ed. *Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press, 1995.
319. J. Sopena. Verbal description of visual blocks world using neural networks, Tr-10, Departament de Psicologia Basica, University of Barcelona, 1988.
320. J Sopena. ERSP: a distributed connectionist parser that uses embedded sequences to represent structure. TR UB-PB-1-91. University of Barcelona, Department of Psychology, 1991.
321. A Sperduti. On the computational power of recurrent neural networks for structures. *Neural Networks* 10:95–400, 1997.
322. R Srihari. Computational models for integrating linguistic and visual information: a survey. *Artif Intell Rev* 8:349–369, 1994–1995.
323. A Stolcke. Learning feature-based semantics with simple recurrent networks. TR-90-015, International Computer Science Institute, Berkeley, 1990.

324. A Stolcke. Syntactic category formation with vector space grammars. In: Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society. Lawrence Erlbaum, 1991.
325. M St John, J McClelland. Learning and applying contextual constraints in sentence comprehension. *Artif Intell* 46:217–257, 1990.
326. M St John, J McClelland. Parallel constraint satisfaction as a comprehension mechanism. In: R Reilly, N Sharkey, eds. *Connectionist Approaches to Natural Language Processing*. Hillsdale, NJ: Lawrence Erlbaum, 1992.
327. G Sun, C Giles, H Chen, Y Lee. The neural network pushdown automaton: model, stack, and learning simulations, UMIACS-TR-93-77, Institute for Advanced Computer Studies, University of Maryland, 1993.
328. R Sun. Beyond associative memories: logics and variables in connectionist networks. *Inform Sci* 70, 1992.
329. R Sun. Logics and variables in connectionist models: a brief overview. In: V Honavar, L Uhr, eds. *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration*. New York: Academic Press, 1994.
330. R Sun. A two-level hybrid architecture for structuring knowledge for commonsense reasoning. In: R Sun, L Bookman, eds. *Computational Architectures Integrating Neural and Symbolic Processes*. Kluwer, 1995.
331. R Sun. Connectionist models of reasoning. In: O Omidvar, C Wilson, eds. *Progress in Neural Networks*. Vol 5. Ablex Publishing, 1997.
332. R Sun, F Alexandre. Connectionist-Symbolic Integration. From Unified to Hybrid Approaches. Hillsboro, NJ: Lawrence Erlbaum, 1997.
333. R Sun, L Bookman. *Computational Architectures Integrating Neural and Symbolic Processes*. Kluwer, 1995.
334. W Tabor, C Juliano, M Tanenhaus. Parsing in a dynamical system: an attractor-based account of the interaction of lexical and structural constraints in sentence processing. *Lang Cognit Processes* 12:211–271, 1997.
335. J Taylor. The historical background. In: E Fiesler, R Beal, eds. *Handbook of Neural Computation*. Oxford University Press, 1997.
336. S Thorpe. Localized versus distributed representations. In: M Arbib, ed. *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press, 1995.
337. S Thrun. *Explanation-Based Neural Network Learning: A Lifelong Learning Approach*. Dordrecht: Kluwer Academic, 1996.
338. S Thrun. Learning one more thing. TR-CMU-CS-94-184, Carnegie Mellon University, 1994.
339. S Thrun. Lifelong learning: a case study. TR-CMU-CS-95-208, Carnegie Mellon University, 1995.
340. S Thrun, L Pratt. *Learning To Learn*. Dordrecht: Kluwer Academic, 1997.
341. S Thrun. Extracting rules from artificial neural networks with distributed representations. In: G Tesauro, D Touretzky, T Leen, eds. *Advances in Neural Information Processing Systems 7*. Morgan Kaufmann, 1995.
342. P Tino, B Horne, C Giles. Finite state machines and recurrent neural networks—automata and dynamical systems approaches. UMIACS-TR-95-1, Institute for Advanced Computer Studies, University of Maryland, 1995.
343. D Touretzky. Connectionism and compositional semantics. In: J Barnden, J Pollack, eds. *Advances in Connectionist and Neural Computation Theory 1: High-Level Connectionist Models*. Ablex, 1991.
344. D Touretzky, D Pomerleau. Reconstructing physical symbol systems. *Cogn Sci* 18:345–353, 1994.

345. G Towell, J Shavlik. Using knowledge-based neural networks to refine roughly-correct information. In: S Hanson, G Drastal, R Rivest. *Computational Learning Theory and Natural Learning Systems*. Cambridge, MA: MIT Press, 1994.
346. F Tsung, G Cottrell. Phase-space learning in recurrent networks. Technical Report CS93-285, Dept. of Computer Sciences and Engineering, University of California, San Diego, 1993.
347. T van Gelder. Classical questions, radical answers: connectionism and the structure of mental representations. In: T Horgan, J Tieson, eds. *Connection and the Philosophy of Mind*. Dordrecht: Kluwer, 1991.
348. T van Gelder. Compositionality: a connectionist variation on a classical theme. *Cogn Sci* 14:355–384, 1990.
349. T van Gelder. Defining “distributed representation”. *Connect Sci* 4:175–191, 1992.
350. T van Gelder. Why distributed representation is inherently non-symbolic. In: G Dorffner, ed. *Konnektionismus in Artificial Intelligence und Kognitionsforschung*. Springer, 1990.
351. T van Gelder. The dynamical hypothesis in cognitive science. *Behav Brain Sci* 21, 1998, 1–14.
352. T van Gelder, R Port. Beyond symbolic: toward a Kama-Sutra of compositionality. In: V Honavar, L Uhr, eds. *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration*, New York: Academic Press, 1994.
353. J van Leeuwen. *Handbook of Theoretical Computer Science*. Vol A: Algorithms and Complexity. Elsevier, 1990.
354. J van Leeuwen. *Handbook of Theoretical Computer Science*. Vol. B: Formal Models and Semantics. Elsevier, 1990.
355. V Vapnik. Learning and generalization: theoretical bounds. In: M Arbib, ed. *The Handbook of Brain Theory and Neural Networks*, Cambridge, MA: MIT Press, 1995.
356. P Verschure. Taking connectionism seriously: the vague promise of subsymbolism and an alternative. *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, 1992.
357. P Verschure. Connectionist explanation: taking positions in the mind-brain dilemma. In: G Dorffner, ed. *Neural Networks and New Artificial Intelligence*. Thomson Computer Press, 1997.
358. G Wahba. Generalization and regularization in nonlinear learning systems. In: M Arbib, ed. *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press, 1995.
359. D Waltz, J Pollack. Massively parallel parsing: a strongly interactive model of natural language interpretation. *Cogn Sci* 9:51–74, 1985.
360. D Wang, B Yuwono. Incremental learning of complex temporal patterns. *IEEE Trans Neural Networks* 7:1465–1481, 1996.
361. X Wang, E Blum. Dynamics and bifurcation of neural networks. In: M Arbib, ed. *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press, 1995.
362. R Watrous, G Kuhn. Induction of finite-state languages using second-order recurrent networks. *Neural Comput* 4:406–414.
363. J Weckerly, J Elman. A PDP approach to processing center-embedded sentences. *Proceedings of the Fourteenth Annual Meeting of the Cognitive Science Society*, Lawrence Erlbaum, 1992.
364. A Weigend, D Rumelhart. Weight elimination and effective network size. In: S Hansan, et al., eds. *Computational Learning Theory and Natural Language Systems*, MIT Press, 1994.
365. A Wells. Turing’s analysis of computation and theories of cognitive architecture. *Cogn Sci* 22:269–294, 1998.

366. S Wermter. Learning semantic relationships in compound nouns with connectionist networks. Proceedings of the Eleventh Annual Conference of the Cognitive Science Society, 1989.
367. S Wermter. A hybrid symbolic/connectionist model for noun phrase understanding. In: N Sharkey, ed. Connectionist Natural Language Processing. Kluwer, 1992.
368. S Wermter. Hybrid Connectionist Natural Language Processing. Chapman & Hall, 1995.
369. S Wermter, R Hannuschka. A connectionist model for the interpretation of metaphors. In: G Dorffner, ed. Neural Networks and a New Artificial Intelligence. Thomson Computer Press, 1997.
370. S Wermter, W Lehnert. Noun phrase analysis with connectionist networks. In: R Reilly, N Sharkey. Connectionist Approaches to Natural Language Processing. Hillside, NJ: Lawrence Erlbaum, 1992. (In: Neural Networks and Artificial Intelligence. Thomson computer Press, 1997.)
371. S Wermter, E Riloff, G Scheler. Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing. Springer, 1996.
372. H White, K Hornik, M Stinchcombe. Multilayer feedforward networks are universal approximators. Neural Networks 2:359-366, 1989.
373. H White, M Stinchcombe. Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. Proceedings of the International Joint Conference on Neural Networks, Washington DC. IEEE Press, 1989.
374. H White. Artificial Neural Networks: Approximation and Learning Theory. Boston: Blackwell, 1992.
375. H White. Learning in artificial neural networks: a statistical perspective. Neural Comput 1:425-464, 1989.
376. R Williams, D Zipser. A learning algorithm for continually-running fully recurrent neural networks. Neural Comput 1:270-280, 1989.
377. W Winiwarter, E Schweighofer, D Merkl. Knowledge acquisition in concept and document spaces by using self-organizing neural networks. In: S Wermter, E Riloff, G Scheler, eds. Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing. Springer, 1996.
378. P Wyard, C Nightingale. A single layer higher order neural net and its application to context free grammar recognition. Connect Sci 2:347-370, 1990.
379. Z Zeng, R Goodman, P Smyth. Learning finite state machines with self-clustering recurrent networks. Neural Comput 5:976-990, 1993.
380. Z Zeng, R Goodman, P Smyth. Discrete recurrent neural networks for grammatical inference. IEEE Trans Neural Networks 5:320-330, 1994.