A General Theory of Intelligent, Self Referencing Networks[1]

Joseph Woelfel
University of Buffalo

William Richards
Simon Fraser University

November 15, 1989

[1] This paper was prepared for the Buffalo Conference on Networks at the University of Buffalo, November 8-10, 1989. We are grateful for the advice and guidance of the participants of the conference.

Rah Press

Intelligence as an Emergent Property of Networks:

Approaches to the study of "intelligence" have been diverse, ranging from those which consider intelligence a mysterious quality which belongs to the soul, fundamentally free and not governed by scientific laws and thus not analyzable by scientific means, to rationalistic rules based "artificial intelligence" or "expert systems". Within this diversity, however, one may identify two major theoretical models which underlie at least most Western theories of intelligent action.

The first of these, and by far the most widely accepted, is a model based on Aristotle's dualistic concepts of *intellect* and *will*. The intellect represents the calculating part of intelligence. It is the part which is aware of its surroundings, identifies and names the objects of experience, and projects future states of the organism. The will, on the other hand, "attaches" itself to some of these possible outcomes and "desires" them. It provides a motive force toward achieving the end state. It is the task, then, of the intellect to plan and carry out a course of action which can result in achieving the desired goal state.

The Aristotelian model is not deterministic. Aristotle was aware of the fact that no valid syllogism which could be constructed from a combination of "intellectual" and "willful" premisses could yield an action as a formal logical conclusion. He concluded that human behavior did not have the "certainty" of physical systems, and cautioned his followers to seek only the level of precision and certainty from this class of phenomena which was appropriate to them. Later Christian philosophers, particularly Aquinas, elevated the uncertainty of the Aristotelian dualistic model to the principle of Free Will. By far the largest part of contemporary theorists in this tradition accept this notion of freedom as an inherent characteristic of human behavior.

This rational quest for desired end states or goals is assumed to take place within a system of constraints which includes the actions of natural laws and the goal oriented activities of other intelligences. Thus some of the plans the intellect might derive are impossible and others prohibited or proscribed by potential conflicts with others. These constraints, over time, tend to be more or less loosely codified into explicit and implicit *rules* which specify what kinds of actions are available, permissible and effective for achieving desired goals, and these rules provide a framework within which an intelligent agent must act.

Rules theories take on many forms. Some theorists focus particularly on human activities in social situations, and recommend careful, sensitive and holistic observations of the behaviors of actors in social situations as a basis for uncovering the latent set of rules which governs those behaviors (Cushman & Pearce, 1977). Chomsky's theory of language behavior can be seen as a specific example of a non-deterministic rules-based model: Within Chomsky's model, freedom is central and distinguishes human language from all other species and automata, since the "...normal use of language is not only innovative and potentially infinite in scope, but also free from the control of detectable stimuli, either external or internal" (Chomsky, 1972, p.12). Moreover, any speaker's grammar "...must, then, contain a finite system of rules that generates infinitely many deep and surface structures, appropriately related. It must also contain rules that relate these abstract structures to certain representations of sound and meaning..." (ibid., p. 17).

Perhaps the most rigorous and ambitious use of the rationalistic rules based models occurs in computer based expert systems, which consist of databases of facts, examples and rules relating the facts and examples, and "inference engines" or algorithms which apply explicitly formulated rules for achieving specific goals, such as configuring or repairing a complex system, diagnosing and treating a disease, determining the location of subterranean mineral deposits, or parsing and understanding natural language.

Whatever the specific form of such Aristotelian models, however, most typically adopt Aristotle's judgment about all rational, rule following systems: rational systems are not typically assumed to be deterministic, and even computer based expert systems often include substantial stochastic components. Unlike a "natural law", any rule may be violated, albeit by risking some penalty associated with its violation.

More recently, an alternative model of intelligent behavior has developed from two unrelated research traditions. The first of these is the "symbolic interaction" model. Interactionists particularly, following Mead, have emphasized the "symbolic" nature of human intelligence, and suggest that, through symbolic interaction with other members of a community, people are able to develop an internal representation of the objects of their experience, themselves, and their interrelationships. This symbolic representation system constitutes the "self concept", which is believed to be the foundation of human intelligent action. (Mead, 1934).

Many, perhaps most, interactionists are themselves Aristotelian rules based theorists who incorporate the interactionist concepts of symbolic communication, self concept and particularly situational relativism into the basic rationalistic model. Some, however, advocate a different approach. Within this second model, behaviors are considered to be components of the self which, through direct ("self reflexive") experience or through communication with others, have been defined as the appropriate activity for them under specific circumstances. Thus, if one has learned to define oneself as brave, brave actions will be appropriate under dangerous circumstances, but if one has learned one is a coward, cowardly actions will be seen as appropriate. In any situation one must define the nature of the situation, define oneself, and define a set of potential behaviors which might occur in that situation. The behavior actually enacted will be the one <u>most consistent</u> with the self as it has been defined in that situation.In this model, behaviors are chosen because they are <u>appropriate</u> and not because they lead to a desired end state (Mills, 1940; Foote, 1951; Lemert, 1951; Woelfel and Fink, 1980).

This second model, rather than assuming behavior to be rational and goal oriented, assumes that behavior selection is a "pattern matching" algorithm. Specifically, within this model an individual in a social situation is confronted by a set of "objects" which vary from situation to situation. Among the objects in the situation are a set of potential behaviors or actions which, through previous experience and communications from others, the individual has learned are possible behaviors within that situation. The definition of self *within that situation* is determined by the individual's perception of his/her relationship to the objects in that situation; the *pattern of action* or "behavior" the individual will exhibit will be that which best matches the *pattern of relationships to objects* which defines the self in that situation.[2]

[2] No idea is completely new, and notions of "consistency" or "pattern matching" can be found in Descartes' notion of "appropriateness to the situation", as Chomsky points out (Chomsky, 1972, pp. 12-13).

5 -- Intelligent Groups

Critics of the "pattern matching" model usually indict it specifically for its denial of the role of freedom of action, which they usually associate with the ability to interpret and generate novel patterns. Chomsky, for example, says:

> "...(T)he normal use of language is innovative, in the sense that much of what we say in the course of normal language is entirely new, not a repetition of anything that we have heard before and not even similar in pattern -- in any useful sense of the terms "similar" and "pattern" -- to sentences or discourse that we have heard in the past. (Chomsky, 1972, pp. 11-12).

Although these two views have coexisted for a very long time, research findings from neither group have had much impact on the views of the other. Neither model, moreover, has been able to suggest a physical mechanism which might produce the phenomena under study. In fact, the absence of any conceivable mechanism by which novel responses to novel stimuli might be generated lies very close to the heart of the dispute, as Chomsky makes clear:

> If by experiment we convince ourselves that another organism gives evidence of the normal, creative use of language, we must suppose that it, like us, has a mind and that what it does lies beyond the bounds of mechanical explanation...(Chomsky, 1972, p. 11).

Recently, however, research in another area has shown some potential for revealing a physical mechanism by which a pattern association model of intelligence might be constructed. Workers in what has variously been called "neural network", "Parallel Data Processing" (PDP) and sometimes "connectionist" models have produced suggestive findings which indicate at least some behaviors often considered "intelligent" may be emergent properties of communication networks. Certain kinds of networks can be shown to receive and store patterns of information, "learn" to associate certain patterns of information with other patterns, and solve logical problems. In fact, since parallel data processing networks develop internal symbolic representations of their environment through interaction with the environment, they may be particularly compatible with an interactionist model of human intelligence.

This paper presents a theory which focuses attention on those characteristics of networks which relate to their capacity to ingest, store, process and output patterns of information. Specifically, the paper presents a general theory of networks which communicate with their environment, and through that communication develop representations of the environment, themselves, and their relationship to the environment which serve as a basis for their subsequent actions. Since interactionist theory considers the central object in any individual's reference system to be the self, we also discuss various network architectures which facilitate self referencing. These networks are called here intelligent, self-referencing networks.

The approach taken in this paper is not meant to imply that work in alternative models of intelligence or language behavior is less promising than the approach taken here, but rather we mean only to explore the extent to which communication networks are capable of forming intelligent, self referencing systems. Nor do we mean to consider intelligence solely a property of individual human beings. If intelligence may be a property of networks and not their components, then it is legitimate to examine the extent to which intelligence may be a property of social networks rather than solely of the individual people of which they are composed. We mean to extend our analysis to communication networks in general, and explore in particular the possibility that large scale social networks such as those which exist in groups, organizations and cultures may themselves constitute intelligent, self referencing systems. Within this system, *neural networks* make up a subset of the more general category of *communication networks.*

Basic Components of Information Processing Networks

The foundational concept in the present theory is the concept of *communication,* which refers to the *changing distribution of energy in space as a function of time.* Communication in its most fundamental sense, as we define it here, means flow of energy. These flows are in general time dependent energy fields. There is no concept of intention or purpose implicit in this definition of communication; it is understood simply as a transfer of information or energy by whatever means.

The region at which two or more flows of energy intersect is defined as a *node*. Within this theory, the *state* of any node is a function of the flows which define it. If the energy fields which intersect to define a node are one dimensional (as the flow of electricity through an ideal one dimensional wire), then the node resulting from the intersection will be zero dimensional, or a point. If the energy flows are dichotomous, that is either on or off, then the node will take on only discrete values. If the energy fields are continuously variable, then the node can take on any positive real value; if the fields may vary in sign, the node may take on any real value positive or negative. If the fields are *n*-dimensional, then the node will be a diffuse *n*-dimensional region whose value will be a function of its coordinates in *n*-space.

In general, a set of energy fields may intersect to generate multiple nodes of various configurations, each of which will be a time-dependent energy field. The set of these intersecting energy fields at any moment will define a *network*, and the set of nodes resulting from the interactions will represent the "pattern" which the network represents at that moment.

This paper restricts itself to the case of one-dimensional energy fields and their resulting "point-nodes". The simplest node can take on only two values along a single dimension, which may be described for convenience as "off" and "on." The value taken by a node at any point in time is called its "activation value." The set of values taken by any set of nodes at a given moment can be defined as a "pattern". "Communication" in this restricted model may be defined as the transfer of all or part of the activation value of any node(s) to any other node(s).

Like any system, a network may be partitioned arbitrarily so that a subset of the original network is defined as the "environment" relative to the other remaining part. This partitioning may be wholly heuristic, and done solely for the purpose of ignoring the internal properties of the portion of the network defined as the environment. This concept of arbitrary partitioning is particularly important in the case of social networks, where each individual person may be considered a node in an organization and each organization may itself be considered a node in a larger social network. The individual himself/herself may be partitioned into a set of neural networks.

Often the level of communication among an arbitrary set of neurons within a single individual may be small or zero while the communication between neurons in one individual and another (albeit mediated by electromagnetic forms of transmission other than typical neural mechanisms) may be substantial. In this (quite common) case, the communication network does not reside wholly within a single individual, but rather may exists across a set of individuals. This at least gives rise to the possibility that the intelligence of such a network may not reside solely in each of the individuals, but rather might be considered a property of the interpersonal network taken as a whole.

The network (considered at whatever level of aggregation) may communicate with its environment through weights or links from the environment to nodes within the network. Nodes which receive information from links to the environment are defined as "input nodes", and nodes which pass information through links to the environment are called "output nodes." Nodes which have no direct connection to the environment are typically called "hidden nodes." In neural networks, the function of a node as input, hidden or output is usually fixed by biological or programming factors, but in social networks, individual nodes may play each of these roles under different circumstances.

Input nodes receive information from the environment in the form of signals which alter their level of activation. In the general case, such signals can take on a wide variety of forms ranging from "simple, signed numbers of limited precision" to "...arbitrary symbolic messages to be passed among...units" (Rumelhart & McClelland, 1987, p. 132), but they all represent communication as defined above, that is, transfer of information or energy from one node to another.

The function by which the activation value of a node is related to an incoming signal is called the "activation function". For a binary node, this function may be as simple as a binary threshold, so that the value of the node is set "on" if the input signals exceed a given threshold level, and off otherwise. For nodes whose activation values may be multivalued, activation functions may be more complicated, particularly when the activation values may also be multidimensional, but the binary representation provides a sound starting point for initial understanding.

A single binary node can encode a pattern consisting of one bit of information. As the number of nodes in a set increases, the amount of information which can be encoded increases. For a network whose input nodes are binary, information received from the environment may be represented as a pattern of ones and zeros displayed over the input nodes. Thus, when a network receives information from the environment, it does so by encountering a signal at each input node at each point in time. Those nodes whose input signals exceed the threshold value will be activated, while others will remain off. The pattern of nodes which are activated constitutes a pattern which represents the pattern of signals at that point in time. The changing pattern of activations over time represents processes in the environment of which the network is "aware."

The number, arrangement and character of the input nodes, along with the character of the activation function, determines what kinds of pattern the input system will be able to represent. Most literature on neural networks and parallel data processing models considers only one dimensional (vector) arrays of one dimensional binary or continuous nodes, since the underlying model for the node in these areas is generally the neuron or the switch. In the more general case we consider here, nodes may themselves be networks whose activations may be highly multidimensional.

A one dimensional (vector) array of binary input nodes can record the presence or absence of a set of features. Figure 1 shows a vector of nodes, each of which represents a letter of the alphabet.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
X	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0

Neurons represented as a vector. Neurons marked
with "X" are activated, all others are off.
This pattern could represent "ACT", CAT", "TAC",
"TCA", "CTA", or "ATC", since no sequence information
is encoded in this pattern. Multiple occurrences of
the same letter cannot be encoded within this scheme.

Figure 1 A One Dimensional Locally Encoded Network

The nodes marked "A", "C" and "T" are on, which indicates that the network recognizes the presence of those letters (features) in the environment. The one dimensional array of nodes, however, cannot encode the sequence of those features, so the pattern encoded in Figure 1 might represent "CAT", "ACT", or any of four other sequences of letters.

A two dimensional array of binary input nodes can keep track of not only the presence or absence of features, but also their sequence. Figure 2 shows a two dimensional (matrix) array of input nodes. As in Figure 1, each column represents a letter of the alphabet, but each row represents an ordinal position in a time sequence.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	0	0	0	0	0	0

Input neurons represented as a matrix. Columns represent letters, rows represent ordinal position in a sequence. This network of neurons encodes the phrase "HELLO SPOT".

Figure 2 A Two Dimensional Locally Encoded Network

The pattern of activations shown in Figure 2 represents the English sentence "HELLO, SPOT". Higher dimensional arrays can represent correspondingly more complicated patterns[3].

[3] As suggested earlier, in the more general case of multidimensional energy flows, it is possible to find arrays in which the nodes are themselves multidimensional. A multidimensional node can represent more than one value simultaneously. This is quite a common case particularly in social networks, where nodes are typically themselves networks on another level of analysis. Like unidimensional nodes, multidimensional nodes might take on binary, multivalued or continuous values,or a combination of these. In general, the higher the number of nodes in a network, the higher the number of dimensions each node can encode, and the higher the number of values per dimension the node can take on, the more complex the patterns the network can represent.

Distributed encoding:

Both the models in Figure 1 and Figure 2 represent examples of "local encoding", in which each node represents one feature. A model which encodes a single feature as a pattern of activations among several nodes embodies what is called is called "distributed encoding", and can store considerably more information in a given number of input nodes. A set of 7 binary nodes is sufficient to encode any of the ASCII characters; a 50X7 matrix of binary nodes can encode the English sentence "The quick red fox jumped over the lazy brown dog," -- or any other string of fifty ASCII characters -- including capitalization and punctuation.

Communication Processes and Network Structure:

The model presented up until now has considered only sets of nodes each of which communicates with the environment, and none of which communicates with each other. Theater marquees and television screens are examples of this class of network. But while the patterns they can encode can be very elaborate, they are passive copies of the environmental input and exhibit essentially no internal processing. Nodes may, of course, communicate with each other at various levels. The channels through which nodes communicate have been called variously "links", "connections", "weights" and other terms, and those terms will be used here as synonyms. These weights may in general take on any real value[4], and are meant here to represent the proportion of the activation level of any node that will be transmitted to another node to which it is connected by that channel. Thus the weight $w_{i,j}$ represents the proportion of the activation value of the i_{th} node that will be communicated to the j_{th} node.

[4] Sometimes, particularly in the case of social networks, precise data about the actual weights or connection strengths is not available to investigators, and so a considerable literature exists in which the connections between nodes are discussed and analyzed as if they were binary. Whatever measurement difficulties might be encountered in any empirical situation, however, this practice is clearly inadequate for the investigation of intelligent, self referencing networks, since these are sensitive to very small variations in weights. In the case of the SPOT and ROVER programs discussed below, rounding the weights at the third decimal place results in serious deterioration of performance.

How a node will respond to the inputs it receives from those nodes which communicate with it is determined by its "activation function." The activation function determines how a node will combine the various signals it receives from all those nodes connected to it. The actual array of potential activation functions is infinite, but they may be described in general from simpler to more complicated functions.

The first is the simple linear function, in which all inputs to a given node are summed, and that node then outputs a signal which is the sum of all its inputs. Simple linear networks can have substantial information storage and retrieval capacities, but cannot produce internal representations of environmental patterns that differ from those in the environment, nor can they perform complex inferences, such as the "exclusive or" relation. Included within the class of linear networks is the *perceptron*, which was studied extensively by Rosenblatt (1962) and Minsky and Papert (1969) who first demonstrated the limitations of inference inherent to the linear two layer network.

A second common activation function is a simple step function, in which a node outputs a given value if the inputs to it sum to more than a given threshold[5]. Even such a simple rule as this introduces important nonlinearity into a network which makes it capable of generating internal representations of external patterns which are not simple linear combinations of external signals, and thus substantially increases its inferential capabilities. Non linear networks can solve problems like the "exclusive or" relation (Rumelhart, et. al., 1986, pp. 318-362, McClelland & Rumelhart, 1988, Chapter 2). The step function, however, is not everywhere continuous, which causes mathematical difficulties for some learning algorithms.

A third commonly used activation function is the logistic function, sometimes referred to as a "sigmoid" function, because its shape when plotted resembles an integral sign:

$$a_{pj} = 1/\left(1+e^{-net_{pj}}\right) \tag{1}$$

[5] The concept of a threshold function is appropriate particularly for social networks, where a communication from one or more nodes may activate another; e.g., "Please call me if anyone calls", or "If there are too many complaints, contact Quality Control".

where:

a_{pj} = the activation of the j_{th} node for the p_{th} pattern, and

net_{pj} = the net input to the j_{th} node for the p_{th} pattern from all input nodes.

The logistic function is particularly useful since it provides the nonlinearity and increased inferential capacity of a step function, but is a continuous differentiable function. This is particularly important in supervised learning or "back propagation" models, since these require that the differences between the pattern output by a network and the desired or "target" pattern be expressed as a continuously differentiable function of the weights so that the weights may be changed to produce the correct output (Rumelhard, et. al., 1987,pp. 318-362).

Each of these activation functions establishes the activation value of the node solely as a function of the inputs from other nodes, but more complicated models can take into account the present absolute or relative activation value of the node. These considerations produce another family of activation functions such as "competitive learning", in which nodes already highly activated are more likely to be further activated for a given level of input than those not so highly activated (Grossberg, 1976), or "resonance", in which sets of interconnected nodes, once activated, will tend to maintain each other's activation levels (Grossberg,1978).

Activation functions can take into account variables other than the set of inputs from other nodes and the activation value of the node itself. *Time* is perhaps the most common such variable, and is usually included to model a decay function such that the node loses a proportion of its activation as a function of time. This decay functions as a "restoring force" which tends to return nodes to their "resting activation levels" as a function of time (Grossberg, 1978; McClelland & Rumelhart, 1988, pp. 12-15).

Activation functions need not be deterministic. Several important models, such as the Harmony Model (Smolensky, 1987, pp. 194-281) and the Boltzman Machine (Hinton & Sejnowski, 1987, pp. 282-317) employ stochastic activation functions, in which the *likelihood* that a node will be activated is a function of the inputs to that node. Stochastic models may well be better representative of actual neural functioning, but are almost certainly more representative of the way inputs function to activate or fail to activate nodes in social networks than deterministic models, at least insofar as the great complexity of input patterns in social networks usually precludes complete measurement of the total net input to any node.

Information Processing and Network Structure:

The weights, along with the activation functions for each node, make up the structure of the network and determine the patterns of flow of information through the network. These flows in turn determine the process by which a network receives information from the environment, constructs an internal representation of that information, and outputs a response.

Figure 3 shows a hypothetical network consisting of six nodes representing the words "Cat", "Dog", "Barks", "Howls", "Meows", and "Purrs".

Input = "Meows"

		Cat	Dog	Barks	Howls	Meows	Purrs
	Cat		-.8	-.9	.2	.8	.9
	Dog	-.8		.9	.3	-.8	-.7
	Barks	-.9	.9		.5	-.3	-.9
	Howls	.2	.3	.5		-.2	-.1
+1	Meows	.8	-.8	-.3	-.2		.8
	Purrs	.9	-.7	-.9	-.1	.8	
		on	off	off	off	on	on

Figure 3 Spreading Activation Network 1

Each of the nodes may take on the value "0" (off), or "1" (on). The nodes are connected to each other by weights which represent their relative "closeness" in the network.[6] They communicate with each other by a simple threshold rule: the signal sent from any node *i* to any node *j* equals the product of the activation value of *i* and strength of the connection between *i* and *j*. Thus the total signal received by any node *j* will be the sum of the signals received from all the other nodes, or

$$anet_i = \sum_{j=1}^{N} w_{ij} a_j \qquad \mathbf{2}$$

The way a node responds to the set of signals it receives is determined by its activation function; in this case we adopt the rule that the node will be activated if the sum of its input signals is positive; otherwise it will be turned off, or

$+1 \text{ if } x > 0$

[6] In the present example, the weights are essentially the correlations between frequencies of occurrence of the various words. Thus "Meows" and "Cat" tend to "go together", with a weight of .8, while "Meow" and "Dog" have a negative coefficient of --.8.

a_i = unchanged if $x = 0$

-1 if $x < 0$

Following this rule, we assume the network receives the input "Meows" from its environment (i.e., the node which represents "Meows" has been activated.) This sets the activation value of "Meows" at +1, and the activation values of the other nodes at 0. Multiplying the weights in each column by the activation values of the corresponding rows, then summing within each column shows that the activation of the node "Meows" will "spread" to the nodes "Cat" and "Purrs", setting their activations to 1, but will leave the nodes "Dog", "Barks" and "Howls" off.

Figure 4 shows that activating the node "Howls", will also activate the nodes "Cat",

Input = "Howls"

		Cat	Dog	Barks	Howls	Meows	Purrs
	Cat		-.8	-.9	.2	.8	.9
	Dog	-.8		.9	.3	-.8	-.7
	Barks	-.9	.9		.5	-.3	-.9
+1	Howls	.2	.3	.5		-.2	-.1
	Meows	.8	-.8	-.3	-.2		.8
	Purrs	.9	-.7	-.9	-.1	.8	
		on	on	on	on	off	off

Figure 4 Spreading Activation Network 2

"Dog" and "Barks"; Figure 5 shows that activating both the nodes "Barks" and "Howls" will also activate "Dog", but will leave "Cat", "Meows" and "Purrs" off.[7]

Input = "Howls" and "Barks"

		Cat	Dog	Barks	Howls	Meows	Purrs
	Cat		-.8	-.9	.2	.8	.9
	Dog	-.8		.9	.3	-.8	-.7
+1	Barks	-.9	.9		.5	-.3	-.9
+1	Howls	.2	.3	.5		-.2	-.1
	Meows	.8	-.8	-.3	-.2		.8
	Purrs	.9	-.7	-.9	-.1	.8	
		off	on	on	on	off	off

Figure 5 Spreading Activation Network 3

This example shows clearly that communication among the nodes of the network produces an apparently qualitative change in the pattern recognition and storage capabilities of the network. When the nodes do not communicate, the network can represent a pattern of virtually any complexity when activated directly by the environment, but the complete input is required to produce the complete pattern. When the nodes communicate, however, the complete pattern can be produced with only a partial input. When a sufficient subset of the nodes in a stored pattern is activated, the activation of those nodes will "spread" through the links and in turn activate the rest of the nodes in the pattern.

[7] A more thorough example would examine the results of the communication after more than the first step or "cycle" of the network. This exercise can produce a surprising amount of complexity very rapidly, particularly in real cases where finite speeds of communication determine the order in which nodes are turned on or off. The activation of the node "Howls", for example, turns on both "Barks" and "Cat". But since "Dog" and "Cat" are so strongly negatively connected, each turns the other off. If "Howls" communicates its activation to "Dog" *before* "Cat", "Cat" will not be activated. If it communicates its activation to "Cat" before "Dog", "Dog" will not be activated. This phenomenon is referred to as "hysteresis" (McClelland & Rumelhart, 1988, pp. 16-17).

It is worth emphasizing the fundamental role communication as it has been defined here plays in this process. A pattern is stored by "connecting" its elements together. Things that "go together" are "close". Nodes or elements in turn *communicate* their activation values to other nodes in proportion to their closeness in the communication network. If a node is "on", it will tend to transmit that "on-ness" to other nodes through the links between them, so that the "on-ness" will spread to other nodes which represent the other elements in the pattern. Similarly, if a node is "off", it will tend to communicate its "off-ness" to other nodes through the links between them. *The entire pattern is encoded in the pattern of communication among the nodes as connections or weights, and can be recovered by the activation of any suitable subset of nodes.*

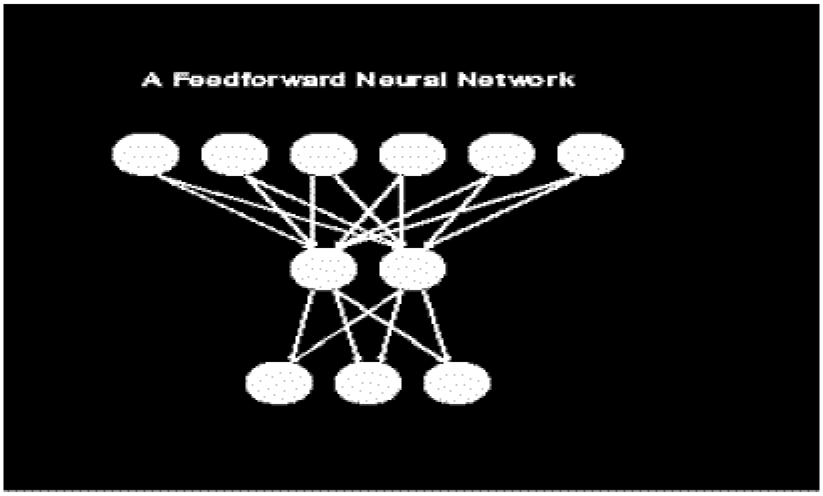
Conversational Networks:

The main characteristic of communication networks as we have discussed them here is their ability to represent patterns and to associate one pattern with another[8]. In the most general sense, conversations may be construed as sequences of patterns, with each utterance considered a pattern of sounds, words, or even letters. With this in mind, it is possible to construct communication networks whose structures are optimized for the recognition and association of linguistic patterns. The process of constructing a communication network consists essentially of defining the pattern of communication links which are allowed among the nodes.

Although it is possible in principle to conceive of every node in a network communicating with every other node, in concrete situations such networks seldom occur. In the human brain, for example, conventional estimates set the number of neurons at about 10^{11}, but each neuron is generally thought to connect to perhaps between 10^{3} and 10^{4} other neurons.

[8] Patterns, like networks, may be arbitrarily partitioned. It may be convenient for some purposes, for example, to consider the phrase "How are you?" to be a single pattern, and to consider the phrase "I'm well, thank you" to be another. Or it may, for other purposes, be useful to consider both phrases part of a single pattern. Depending on the arbitrary terminology employed, a network might be considered a "heteroassociator", which associates one pattern with another or one part of a pattern with another part, or an "autoassociator", which associates any part of a pattern with the entire pattern.

Similarly, although there are about 5^9 people on the earth, each person communicates with only a very small subset of them, and communicates substantially with only a few.



In fact, constraints on which communication links may not be made -- that is, which weights must always be set to zero -- determine the overall structure or architecture of a network. Figure 6 shows a simple yet interesting information processing network: a three layer feed-forward network. The row of nodes at the top of the figure represent input nodes; they are connected to a row of hidden nodes, which in turn are connected to a row of output nodes. Nodes within a row are not connected to each other, nor are any of the input nodes connected to any of the output nodes except through the hidden layer, and these connections are themselves only one way paths. Input nodes may communicate to the hidden nodes, and hidden nodes to the output nodes, but the reverse processes -- hidden to input and output to hidden -- are prohibited.

A network of this configuration can receive inputs from its environment, form an internal representation of the input patterns in the hidden layer, and output a pattern corresponding to the input pattern. The pattern of weights between the input and hidden layer and the hidden and output layer will determine the relationship of the output pattern to the input pattern.

A network of this general configuration is implemented in the FORTRAN program SPOT. Its the input and output layers are each in the form of the 50X7 matrix described earlier, with a 1 X 115 vector of hidden nodes between them. Each of the 7 nodes in each row of the input and output matrices are required for the distributed encoding of each character in the

ASCII set, and each row of the input and output layer is thus able to represent one such character. This network can learn to associate a number of utterances or "strings" of up to 50 characters with any other arbitrary set of strings of up to 50 characters.[9] The "memory" of what output strings "go with" what input strings is contained entirely in the pattern of weights among the layers.

The network works as follows: input strings of letters are "encoded" into their ASCII representations which consists of a seven digit array of 1's and 0's; and each node corresponding to a 1 in the appropriate row of the input matrix is turned "on", (that is,its activation value is set to 1), and all others are turned "off", or set to zero. Each letter in the string of letters thus corresponds to one row of the 50X7 matrix of input nodes.

The activation values of the input nodes (either 1 or 0) are multiplied by the weights which represent the communication strengths between the input nodes and the hidden nodes. The activation values of the hidden nodes are then calculated from the activation function given in equation (1) above -- that is, their values are set as the logistic of the sum of the activations of the input nodes multiplied by the weights from input to hidden nodes. The pattern of activations of the hidden nodes represents an internal symbolic representation of the input pattern, and serve an intermediary role between the input pattern and the output pattern. The mapping of the input pattern as it is encoded on the input neurons onto the hidden neurons represents an internal restructuring, and, (in the case of the SPOT and ROVER algorithms presented here) more parsimonious form of the input pattern, but it is fair to say that a complete understanding of the way in which hidden nodes functions awaits a good deal more research. It is known, however, that the hidden layer makes possible solutions to problems that cannot be solved by linear models (Minsky & Papert, 1969; Rumelhart, et. al., 1986, pp. 318-362, McClelland &

[9] How many such patterns the network could learn would be determined by the number of connections possible between the layers, which in turn is determined by the number of nodes in the layers. For input and output layers of fixed size, as in the present example, increasing the number of nodes in the hidden layer will increase the number of patterns the network can learn.

Rumelhart, 1988, Chapter 2). The level of complexity of the internal symbolic structures that can be formed, and the level of complexity of the functional relations between input patterns and output patterns that can be learned by a network are related to the number of hidden neurons[10].

The activations of the hidden nodes are then propagated to the output nodes by exactly the same process. (The values of the weights determines completely what output nodes will be activated for any given pattern of input node activations, and thus determines uniquely what the network will output or "say" for any given input. How the weights are actually set will be discussed below.) These output nodes are then 'thresholded"; that is, if their calculated values exceed an arbitrary threshold value, the are set to 1, otherwise they are set to zero. The resulting 50X7 binary matrix of output nodes can then be "decoded" into the appropriate ASCII characters.

A network like SPOT can be taught to associate any input phrase with any output phrase by setting the communication weights appropriately. Thus, for example, it is possible to choose a set of weights such that the pattern of activations of input nodes corresponding to a phrase such as "How are you, SPOT?" turns on the set of output nodes which correspond to the phrase "I'm fine, thank you," while the pattern of input activations corresponding to another phrase will activate a pattern of output nodes corresponding to still another phrase. How many such pairs of input and output patterns the network can learn is a function of the number of nodes and the degree of similarity among the patterns.

[10] Increasing the number of hidden nodes in an otherwise unchanged network increases not only the number and complexity of associations the network can learn, but also increases the speed with which it can learn them. This fact is somewhat confounded when the parallel architecture is simulated on a Von Neumann machine, since the number of operations the algorithm must perform increases as a function of the number of connections in the network. The SPOT algorithm,for example, learned a simple training set in 1,005 seconds with 55 hidden nodes, but took 27 trials to do so. Increasing the number of hidden nodes to 75 cut the time to 757 seconds and reduced the number of trials to 15. Increasing the number of hidden nodes to 100 reduced the number of trials to 14, but *increased* the time to 960 seconds. Increasing the number of hidden nodes to 115 reduced the number of trials to 12 and the time to 935 seconds. If the network were able to carry out its activities in a fully parallel fashion, increasing the number of nodes would result in a monotonic reduction in both number of trials and overall time.

Forming and Changing the Network Structure:

There are two fundamental types of patterns implicit in the discussion up until now. The first are the patterns composed of the activation values of the nodes at a given moment, or the changing pattern of activations across an interval of time. These patterns, however complex they may be, are transitory and represent the image a network is displaying at a historical time. In a neural network, they represent the "thoughts" the network is exhibiting, while in a social network they represent the "currents of opinion" (Durkheim, in Simpson, 1963, pp.27-27) flowing through a culture at any moment.

The second type of pattern is represented by the much deeper, more stable patterns represented in the set of weights in the network. These weights represent the relatively stable basic structure of the network and the set of relationships among the other patterns. They represent not what a network is "thinking" at a given moment, but what it has learned since its inception. While the activation values of the nodes at any given moment determine the pattern a network is displaying at that moment, the memory of all patterns known to the network is contained in the pattern of weights or communication patterns among the nodes. The processes by which these weights are formed and changed, therefore, are central to whatever intelligence a network might exhibit.

The first general type of processes by which the architecture of a network can be changed might be considered *causal* or perhaps *accidental* processes, that is, processes in which forces from the environment impose themselves on the network and its components without any deliberate goal in view.[11] Perhaps the most common such process is that suggested in broad outline by Hebb (1949): Weights among nodes which are simultaneously activated are strengthened.[12] The logic behind the Hebbian rule is quite straightforward: when a network is

[11] These forces include genetic forces which determine the structure of neural networks, geographic and climatic forces which influence social networks, as well as accidental forces which may alter or disrupt the structure of any network.

[12] Hebb did not provide an exact mathematical form for this rule, and minor variations in form are abundant. The weights in the examples presented in Figures 3, 4 and 5 are given as correlations among the

displaying a given pattern as a set of activated nodes, it can "remember" or store that pattern by connecting together the nodes which represent the pattern so that they tend to be activated as a unit. If the rule which "ties together" nodes which are simultaneously active is always automatically enforced, then such a network will tend to store any pattern it displays automatically. If the rule is repeatedly enforced, then each time a pattern is displayed by a network the connections among its constituent nodes will be strengthened and the network's memory of the pattern will be reinforced. If the initial activation values of the nodes are set by inputs from the environment, then the Hebbian rule guarantee that such a network will develop a memory of the major features of its environment, and will selectively favor the recollection of those patterns most frequently presented to it by the environment.[13]

A typical example of a Hebbian learning network is CATPAC. CATPAC is a computer program which reads ASCII text which has been divided into "episodes", which are arbitrary delineations supplied by the analyst.[14] CATPAC then parses the text into words, and counts the number of times each word occurs in each episode. This *words* by *episodes* matrix **E** is then postmultiplied by its transpose to calculate the *words* by *words* matrix **W** which represents the *frequency of cooccurrence* of all possible pairs of words within each episode. These cooccurrence frequencies may be thought of as connection strengths or weights between pairs of words. Those that "go together" most frequently are most tightly connected. Suitably normalized, CATPAC will produce networks of the kind shown in Figures 3 through 5 (Woelfel, 1987).

A network whose weights are determined solely by a variant of the Hebb rule or other causal mechanisms becomes "like" it's environment in a wholly passive way. A second set of processes by which network structure can be changed might be called *purposive* or *supervised* processes. One such alternative, which is itself a variant of the Hebb rule, has been suggested

occurrences of the items, or in the terminology developed here, correlations among the frequencies of activations of the nodes in the patterns.

[13] The Hebbian rule has certain limitations which are well understood in the Parallel Data Processing community. Among the most important is the tendency to confuse similar patterns.Since similar patterns will be encoded into similar weights, presentations of partial information about highly covariant patterns will result in frequent confusions among the patterns (McLelland, et. al., 1987, p.38).

by Rumelhart, et al, (1987, pp 318-362).The essential feature of this "back propagation" model is the existence of a "target" pattern, that is, a pattern which is, for any arbitrary reason, considered to be the "correct" output pattern for a particular input pattern. In the conversational networks described above, for example, the output pattern "I'm fine, thank you." is the "correct" pattern the network is expected to output when receiving the input pattern "How are you, Spot?"

The target pattern represents a pattern of activation values of the output nodes of a network corresponding to the desired output. The difference between the pattern desired and the pattern actually output by the network can easily be defined as the difference between the activation values of the nodes observed and those expected by the pattern. These differences may be considered the *errors* produced by the network. These errors can of course be described as a function of the activations of the nodes, which can in turn be expressed as a function of the weights connecting the nodes. It is possible, then, to express the errors as a function of the weights. If the activation functions of the nodes are continuous (as is the logistic function typically used in back propagation networks), then the derivative of this function is defined everywhere on the function, and it is easily possible to modify the weights (usually by a quasi steepest descent algorithm) until the error is minimized[15]. Such a network can learn to produce a desired output pattern for a given input pattern.

Both the SPOT and ROVER algorithms described in this paper are back propagation models; they are supplied with a set of input phrases along with the set of desired output phrases associated with those inputs. Connections between input and hidden nodes and hidden

[14] An "episode" can represent any arbitrary delineation of text. Each episode might represent a single interview, or the answer to a single question, or a single page or a single line of text, or any other arbitrary boundary,

[15] Notice that this process occurs "backwards" through the network, beginning with the errors of the output nodes, then moving to the weights from hidden to output, then to the activations of the hidden nodes and then to the weights from input nodes to hidden nodes. This backwards sequence is the basis for the name "back propagation."

and output nodes are initially randomized, so that when the network receives an input pattern, the response it outputs is simply a random activation of the output nodes. The errors are then calculated as the differences between the actual values of the output nodes and the values associated with the correct pattern. By a quasi steepest descent algorithm, the weights of the connections among the nodes are then modified until all the correct response patterns are associated with the appropriate input patterns.

These networks are "trained" by presenting them with lists of paired patterns. The first pattern in each pair of patterns is an "input pattern", and represents a given pattern of activation of the input nodes of the network. The second pattern in each pair represents the pattern of activation of the output nodes which is meant to be associated with that input pattern.

When the input pattern is initially displayed, the (initially random) connections between input and hidden nodes and hidden and output nodes causes a random pattern of activation of the output nodes. The values of this output pattern are subtracted from the values in the "target pattern" and the differences represent error. These errors can then be expressed as functions of the activations which in turn are expressible as a function of the weights. The derivative of this function is then calculated and the weights are modified, the input pattern is presented again and the process is iterated until the errors fall below a specified tolerance. Because the activation function of the nodes in SPOT, ROVER and ROVER II are nonlinear (logistic) functions, this procedure is essentially an iterative non-linear multiple regression model which finds a set of weights which maps the pattern of input activation values onto the desired pattern of output activations.

Figure 7 shows SPOT learning the "correct" responses for two phrases: to the phrase "Hello, Spot!", it is expected to say "Hello.", and to the phrase "How are you, Spot," it is expected to reply "I'm well, thank you." As implemented in SPOT, the network requires 8 tries to get both responses correct, although the network "overlearns" for two more iterations (not shown) until the error is within the prespecified tolerance.

The network begins by producing a random response to the initial inputs, but quickly learns the correct response through a series of successive approximations, each time closer to the pattern than the last. Figure 8 shows errors for each iteration, along with the elapsed time for each trial.

```
Hello, Spot!
   _!/ 5!A!/' /!' ?
How are you, Spot?
Hello.
Hello, Spot!
Hmmlo. ,( 0`  ` _ p(
How are you, Spot?
Jello.   0       (
Hello, Spot!
Hmmlw.$,( 0` (  ``t,
How are you, Spot?
Jen w. (  p`    /_d,
Hello, Spot!
Hmllo.
How are you, Spot?
J_m w&`($ t`b$` wof,
Hello, Spot!
Hello.
How are you, Spot?
Igm wed(, tha$d you.
Hello, Spot!
Hello.
How are you, Spot?
I'm well, thafk you.
Hello, Spot!
Hello.
How are you, Spot?
I'm well, thafk you.
Hello, Spot!
Hello.
How are you, Spot?
I'm well, thank you.
```

Figure 6 Three Layer Feedforward Network SPOT Learning Two Phrases (Dialogue)

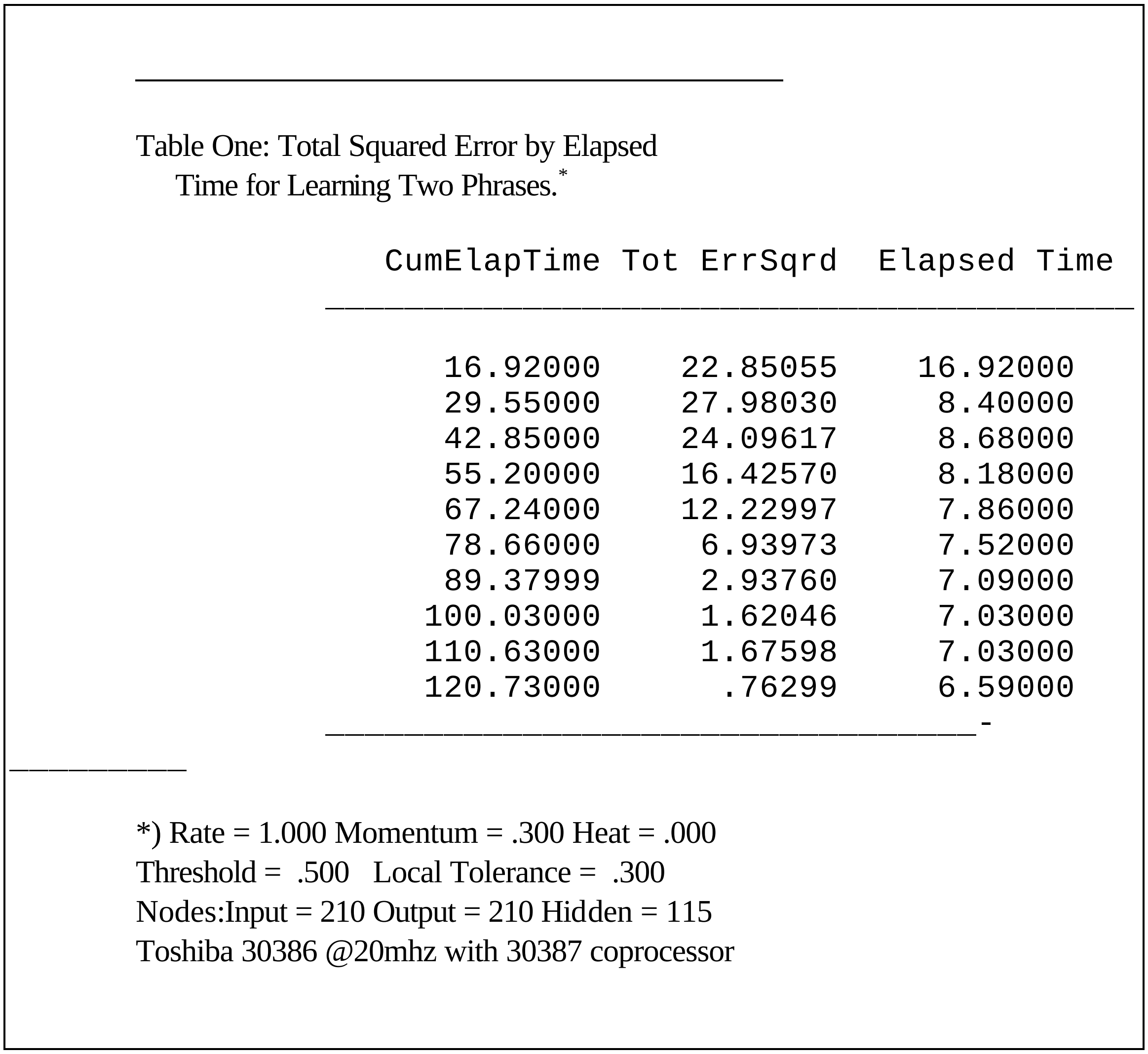


Table One: Total Squared Error by Elapsed
Time for Learning Two Phrases.[*]

CumElapTime	Tot ErrSqrd	Elapsed Time
16.92000	22.85055	16.92000
29.55000	27.98030	8.40000
42.85000	24.09617	8.68000
55.20000	16.42570	8.18000
67.24000	12.22997	7.86000
78.66000	6.93973	7.52000
89.37999	2.93760	7.09000
100.03000	1.62046	7.03000
110.63000	1.67598	7.03000
120.73000	.76299	6.59000

*) Rate = 1.000 Momentum = .300 Heat = .000
Threshold = .500 Local Tolerance = .300
Nodes:Input = 210 Output = 210 Hidden = 115
Toshiba 30386 @20mhz with 30387 coprocessor

Figure 7 Three Layer Feedforward Network SPOT Learning Two Phrases

The values of the weights represent the learned pathways of communication between the input nodes and the output nodes. They represent the network's "memory" of what string of ASCII characters it should output for any given input string of ASCII characters.

```
Conversationwith Galileo*SPOT
 O: = Other; S: = SPOT

 O: How are you, Spot?
 S: I'm well, thank you.

 O: How are you, Sopt?
 S: I'm well, thank you.

 O: How are you?
 S: I'm well, thank you.

 O: Hou art yee, Spet?
 S: I'm well, thank you.

 O: How are
 S: A'm well, thank you.

 O: How you doin', Buddy?
 S: Agm weld, t _nk )>v,

 O: Howdy do?
 S: A'm"weld, phanc yov(
```

Figure 8 Conversation with Three Layer Feed-forward Neural Network SPOT

Such a network can be taught to carry on a rudimentary conversation. ("Teaching" the network consists exclusively of setting the weights or communication strengths between input and hidden nodes and hidden and output nodes.) For an input phrase such as "How are you, Spot?", it might be taught to respond "I'm fine, thank you." What's more, as shown earlier, because the activation values of the output nodes are determined by the entire pattern of activations in the layers preceding them and the entire pattern of connections among those layers, the complete output pattern can be activated by an input of only part of the input pattern; minor misspellings of the input string, or even leaving parts of the input blank will still result in the output of the entire pattern. Thus, for the input string "How are you, Sotp?", or just "How are you?", the network would respond "I'm fine, thank you." As the input pattern deviated further from the pattern the network had learned to associate with the output, the output pattern would degrade fairly gracefully, but would retain the main features of the correct output pattern even with considerable distortion or deletion from the input pattern.

Figure 9 shows a brief conversation with a three layer feed forward neural network (SPOT) in which the input pattern (marked "O:" in the figure) is gradually changed from the pattern the network has learned to recognize. The network is able to output the correct response even after substantial changes in the input pattern, but eventually degrades as the input pattern departs further from the learned pattern.

Innovative Language:

As the previous example shows, the thresholding function guarantees that a network can continue to produce without error an output pattern it has learned to associate with a given input pattern, even when the input pattern differs to some extent from the pattern originally learned. One way, then, that a network deals with an innovative input pattern, that is, one it has not previously encountered, is to output the pattern that corresponds to an input pattern which is similar to the novel input pattern. As the example also shows, however, as the input pattern deviates still further from the original form, the network outputs a pattern which also differs from the one learned. When these deviations are arbitrary, as they are in the example, the network can produce an output pattern that is itself arbitrarily degraded or noisy. The network will, in other words, produce a novel output when receiving a novel input, although in this situation the output will typically be meaningless, as in Figure 7. Such productions of novel but meaningless output patterns in response to novel input patterns does not meet Chomsky's criterion that the novel output be meaningful.

It is possible, however, for novel inputs to a network to be related in systematic ways to input patterns the network has already learned. When this happens, it is possible for the network to output a novel pattern which it has not previously encountered, but which is still a meaningful pattern.

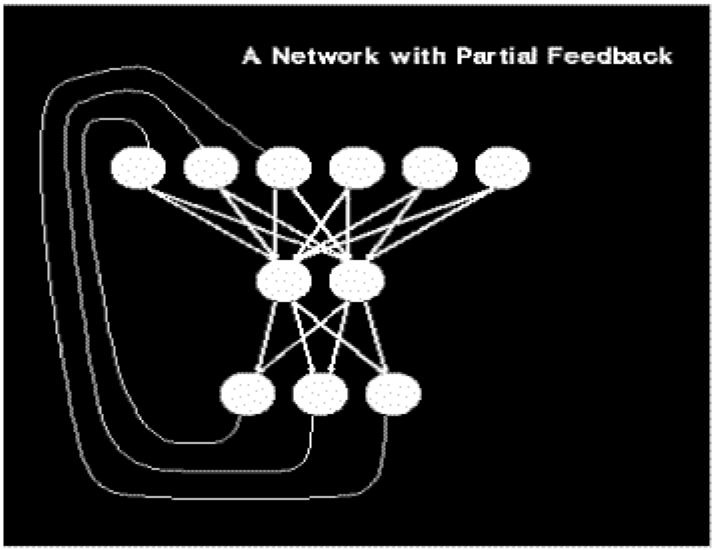
A network identical in structure to the network shown in the previous example, although employing local rather than distributed encoding[16], was taught to associate the input

[16] When a network of this type encounters two or more previously learned input patterns simultaneously, it activates the combination of communication channels or connections appropriate to that combined input set. This produces an output pattern which is a combination of the output patterns associated with each of the input patterns separately. In the particular distributed encoding scheme employed by the SPOT algorithm, this results in an output pattern which represents a set of ASCII characters which, while a proper combination and in fact a valid inference, nevertheless requires additional

pattern "GREET" with the output pattern "HELLO", and also to associate the input pattern "MY FRIEND" with the output pattern "BOB". When the pattern "GREET MY FRIEND" is input to the network, it responds "HELLO BOB". Neither the input pattern "GREET MY FRIEND" nor the output pattern "HELLO BOB" has ever been encountered by the network before, but the network is able nonetheless to generate an appropriate English sentence which is a "correct" novel response to the novel input pattern.

To be sure, this is a very limited example of innovation, but, in principle, it responds to Chomsky's (1972) argument that the number of possible English sentences is simply too large to have been learned and remembered, but must instead be generated from a set of internal rules. (Chomsky, 1972, pp 11-12). The sentence "HELLO BOB" was "generated" by the network in response to a novel input not previously encountered, but the network was not following any rules in so doing. Nor was the novel response in any meaningful sense programmed into the network, but rather was exclusively the result of its training.

Self Referencing Networks:



interpretation to be understood. In the locally encoded network used in this example, each node represents a single letter; thus when the network outputs a combination of these letters, they can be understood easily.

The network shown in Figure 7 has several interesting conversational properties: it can associate appropriate linguistic outputs with arbitrary language inputs, it can recognize a known input pattern even if it differs fairly substantially from the exact form in which it was learned, and it can produce meaningful and novel output utterances in response to novel inputs. It will, however, always respond in exactly the same way to the same input pattern regardless of the context in which it occurred. The network illustrated in Figure 10, on the other hand, is somewhat more sophisticated. This network resembles the previous network except for feedback loops from the output nodes to half of the input nodes. This means that the input pattern which is associated with a given output pattern includes not only the pattern from the environment, but also the pattern previously output by the network. This network need not respond in exactly the same way twice to any given input pattern from the environment. The network in Figure 10 is *self referential* in that it takes its immediate past behavior as part of the pattern to which it must respond.

Conversation with Three Layer
Network with Feedback
O: = Other; S; = Spot

O: How are you, Spot?
S: I'm fine, thank you.

O: How are you, Spot?
S: Still fine, thanks.

O: Oh, I'm sorry.
S: That's O.K.

Figure 10 Conversation with a Three Layer Network with Output Feedback

In ROVER, the computer program which implements this design, the feedback from output nodes to input nodes is done after thresholding the output units. It is more appropriate to think of this network as monitoring its *behavior* rather than its *"thinking"*. If the feedback loop were implemented before thresholding, the input nodes would be aware of what the network was *thinking* just before it "spoke", but would not be aware of what it actually said. A more sophisticated network (like the one implemented in ROVER II, below) could, of course, be aware of both by taking feedback from both places.

Figure 11 shows a conversation between a network of this type and a person (other). Note that the network responds differently to exactly the same input string depending on what it has previously said. The network has taken its past behavior into account in determining its response to the input from its conversation partner.[17]

While the network shown in Figure 10 is self-referential in an important sense, the network shown in Figure 12 is even more so. The networks described so far associate input patterns with output patterns through weighted communication connections from input nodes to output nodes through hidden layers of nodes. When the network has learned an association, the activation of the nodes associated with the input pattern will be channelled through the weighted communication channels to the nodes associated with the proper output pattern. It is also possible, as shown earlier, for a novel input pattern to be related in a systematic way to patterns which a network has previously learned, so that the network "knows" a correct response for even these novel input patterns. But when an input pattern that the network has not learned to associate with any particular output pattern is input to the network, it will output an arbitrary nonsensical pattern. The network does not know whether it "knows" what it is about to say, and will produce babbling for unlearned input patterns.

[17] While the network implemented in ROVER takes into account only the last utterance the network has made along with the new input from its conversation partner, there is no reason in principle, nor any particular technical difficulty in extending the model back for as many stages as desired; a network can easily be programmed which will take into account the last two or four or eight or any number of previous exchanges in determining what it should output. It is also easily possible to weight earlier episodes differentially, giving them successively less weight as they recede into the past. How many stages (or how long in real time) a network ought to take into account in determining its response in order to be an interesting conversation partner remains an unanswered empirical question, but presents no special programming difficulties. Such networks could not be accused of "linear, sequential" thinking, since they might well revise their understanding of a previous utterance given a later one.

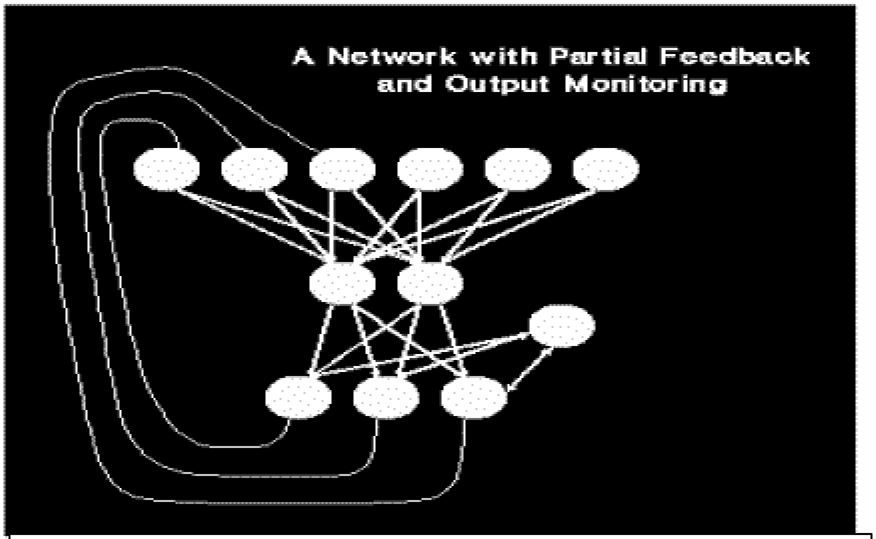


Figure 11: Network with Feedback and Output Monitoring

It is quite important that a self-referential network like ROVER not babble, since such a network will necessarily take into account the immediate history of a conversation as the pattern to which it must respond. If that history contains a sequence of random or arbitrary utterances, there will likely never be a consistent pattern for the network to learn, which would seem to present a formidable barrier to developing conversational competence.

The network in Figure 12 has an additional node which monitors the other output nodes to determine whether they are patterned or not. In order to understand how this monitor node operates, it is useful to recall that the network represents a pattern by turning some of its output nodes "on" and turning the rest "off". When the network is representing a pattern it has learned, therefore, its output values all be either nearly 1.0 or 0.0. (Since the activation function for this network is the logistic, actual values range closer to .9 and .1.) When the network is representing arbitrary or random nonsense, on the other hand, the values of the output nodes will take on the full range of values between 0.0 and 1.0, with a mean value of about .5. Thus a network which is representing a learned pattern will have output activation values that are maximally different from the mean activation level.

Input to the monitor node, then, consists of the (squared) differences between the actual values of each output node and .5, the mean value expected for an arbitrary nonsense output.

Once appropriately normalized, these values are summed and entered into the activation function for the monitor node; if its actual activation exceeds a preset[18] threshold, the monitor node "senses" a learned, patterned output, and activates the network's output. If, on the other hand, the activation value of the monitor node falls below its critical threshold, it is quite likely that the pattern represented by the output nodes is simply an arbitrary, unlearned nonsense pattern. In this case, the network's output is set to "blank". It is important to note that this node does not determine whether the output pattern is "correct" or "sensible", but simply can detect the difference (in most cases) between a systematic, patterned output and gibberish.

While it would be wrong to attribute too much sophistication to the model implemented in ROVER, the monitor neuron goes beyond simple self reference, and adds a minor but nonetheless important self *evaluative* dimension to the network. While the model implemented in ROVER is "aware" of its past behavior and takes it into account in determining its subsequent behavior, the model in ROVER II is aware of both its past behavior and certain characteristics of its present "mental state" or "thinking", and it "evaluates" that state before implementing the action implied therein.

While the networks implemented in the SPOT and ROVER algorithms show in principle that one may construct conversational, self referencing systems of communication networks, they are in fact very simple, small and limited networks. The largest (ROVER II) consists of only 601 neurons, and 39,725 possible communication links[19]. Compared to a single human brain, with perhaps 10^{11} neurons, these networks are minuscule. Further, while the most

[18] While in the ROVER II implementation this threshold is hardwired, it would be a straightforward modification to make its value depend on inputs to the network, so that, for some kinds of input, the network would be very careful not to babble; that is, to make very sure it "knew" what it was about to say before responding, while, in response to other inputs, it might be more willing to guess at a response even though there was a high likelihood it was nonsensical.

[19] There are 350 input neurons, 75 hidden neurons, 175 output neurons and one monitor neuron. There are thus 350 X 75 = 26,250 possible pathways from the input layer to the hidden layer, another 13,125 pathways from the hidden layer to the output layer, 175 pathways from the output layer to the monitor neuron, and 25 feedback pathways from output to input.

sophisticated of the networks described here, the architecture of ROVER II is severely limited compared to that of a single human individual. ROVER II has only one input "sense": its input is restricted to 50 ASCII characters from a file or keyboard, while a human individual can receive information from multiple senses. The simultaneous activation of nodes connected to visual, auditory, taste, olfactory and tactile senses, coupled with a simple Hebbian learning rule which enhanced the connection among those nodes simultaneously activated, make possible the formation of complex internal patterns which can be activated by partial inputs, so that a picture of food, for example, could produce the same pattern as the taste or smell of the same food. This is in principle possible for an artificial network like ROVER II, although the technical difficulties of simulating such massive parallelism on Von Neumann architecture machines are for the moment quite formidable. (Although the ROVER II architecture is completely parallel, its implementation is simulated on a Von Neumann serial machine. This means that it cannot actually do any two things simultaneously, and must take in information in "batches" and operate in discrete "jumps" or "cycles".)

ROVER II is thus substantially handicapped when taking in information needed to define its social situation; it may well be more appropriate to compare it to a person who received all his or her information about the world from a teletype which could deliver only 50 ASCII characters at a time. In spite of these limitations, however, ROVER II provides a useful basis for understanding the way in which the basic structure of a network functions in the processing of information which can be useful to an analysis of social networks and their information processing capabilities.

Information processing in social networks

In a social network, each individual is him/herself a network of substantial size, and can thus be thought of as a multidimensional real valued node. Social networks, which can consist of many individuals, thus have a total number of nodes many times this figure.

Networks consisting of arrays of multidimensional real-valued nodes can store very complicated patterns. Humans and organizations represent nodes which can produce highly multidimensional real-valued outputs. As an illustration of the pattern storage and recognition capacity of networks of this type, a group of students heard the following paragraph read aloud:

> I have a very small bedroom with a window overlooking the heath. There is a single bed against the wall and opposite it is a gas fire with a gas ring for boiling a kettle. The room is so small that I sit on the bed to cook. The only other furniture in the room is a bookcase on one side of the gas fire next to the window -- its got all my books on it and my portable radio -- and a wardrobe. It stands against the wall just near to the door, which opens almost directly onto the head of my bed. (Johnson-Laird, 1983, pp.)

Afterward, each of them was asked to report their estimates of the distance between each of the 13 objects mentioned in the paragraph and each of the others, which represents a highly multidimensional output of 78 real valued numbers per person (node). The numbers for each distance were entered into the Galileo Version 5.4 computer program (Woelfel & Fink, 1980), which averaged[20] the distances over nodes (respondents) and generated the coordinates of the 13 objects in space. The picture generated from this exercise is consistent with the room in the text read to the students. Random splits of the data show the same room, as should be expected. Thus the network of individuals recorded the pattern of the entire paragraph within a single aloud reading -- and could reproduce it accurately -- even though none of the individuals reported being able to picture or draw the room.

This illustrates two important characteristics of information processing networks: first, a network is capable of encoding very complicated patterns of information very quickly and of retrieving it accurately. Second, it shows clearly that the information encoded is not a property

[20] Averaging the values is a very simple but common function for numeric outputs which can be viewed as analogous to thresholding. More complicated functions, such as log transforms, trimming, and the like, are often used, as are other measures of central tendency, but the concept of an aggregate pattern which has meaning for a collection of nodes while essentially uninterpretable based on outputs from only a single node remains the same.

of the individual nodes, since, in the example given, none of the individual nodes (individuals) reports being able to picture or recall the overall geometric structure of the room. Rather the pattern exists only in the network of nodes considered as a whole. Moreover,in the present example as well, it is possible to see that the information is stored in such a distributed and redundant way that the pattern can be retrieved from a reasonably sized random subset of the nodes. This characteristic makes it clear how extensive networks such as organizations, nation states and cultures can retain complex information patterns such as attitudes, beliefs and values over generations even when many or even all individual nodes are lost to the system due to immigration, death and other factors. Most importantly, however, this example makes it clear that *the network as a whole exhibits emergent pattern storage and retrieval capabilities that go beyond those of the component nodes.*

Self Referencing Social Networks

While the capacity of the social network to store complicated patterns of information quickly and to recover them even from subsets of the original network is very substantial, the sophistication of the network illustrated in the previous example is, in certain ways, less than that illustrated in ROVER II. Like ROVER, this social network can record a pattern and output that pattern or a pattern related to it, but, unlike ROVER II, it, as a single entity independent of its constituent members, is itself unaware that it knows the pattern or not. In fact, if exactly the same questionnaire is administered to a collection of people who have not heard the paragraph read, they will for the most part output arbitrary numbers and produce a "room" which bears only accidental relations to the room described in the text.[21] Moreover, as a direct consequence

[21] Of course, each individual person in the sample will believe that he or she does not have a pattern in mind which he/she is being asked to describe, but that belief also characterizes those who did hear one reading of the description. In fact, no individual but the most exceptional does have an internal representation of the room after a single hearing. The issue here is the larger sociological or cultural issue: in the case where no one has heard the reading, neither the group as a whole nor any individual member of the group has a sense of the overall pattern of the room. But in the case where the group has heard a reading, it is still the case that no single individual has a clear grasp of the room as a pattern, but the group as a whole

of this lack of self awareness, it is not possible for this social network to correct whatever errors might exist in its representation.

The inability of the social network in the example to be self aware and self correcting is a direct consequence of the simple architecture of the network, which consists of (in the present case) about 40 multidimensional real-valued nodes, each one of which serves both as an input node and an output node. There are no hidden nodes, nor do any of the nodes communicate with any other. There is no "monitor node", which functions to compile the activation values of the other nodes, and so the network, regardless of its information storage capacity, has essentially no cognitive capacity whatever. It is essentially nothing more than an elaborate memory. The very simple network embodied in ROVER II, therefore, while quite limited compared to a human individual, can exhibit more complicated cognitive activity within the limits of its very limited memory than the set of 40 human individuals with approximately 4 $X10^{12}$ neurons, *as long as no internal communication structure is allowed to develop among those individuals.* The reason for this is that the size and complexity of any single pattern which can be encoded by a network is a direct function of the number of nodes, the dimensionality of the nodes, and the number of values each node may take on for each dimension, but the cognitive capacity of a network -- that is, the number of such patterns the network might store, its capacity to associate them with other patterns, and its capacity to monitor its own activities -- is a function of the pattern of communication among the nodes. *It is possible, therefore, to construct an intelligent, self referencing network from a set of nodes which are themselves not intelligent, and it is possible to construct a network of intelligent, self referencing nodes which collectively is neither intelligent nor self-referencing.*

Conclusions and Implications:

does have such a pattern embedded in it. Like ROVER, however, the collective group has no idea that it "knows" the pattern, even though it does know it.

While none of the simple networks presented in this paper may be claimed to exhibit anything more than the most rudimentary intelligence or self awareness, they illustrate certain factors that are essential to the development of an intelligent, self referencing, goal directed network:

First, there must exist a set of input nodes which receive information from the environment, a set of hidden nodes, which allow the network to form an internal representation of the input information, and a set of output nodes which the network communicates information to its environment. Second, there must be a pattern of communication channels from the input nodes to the hidden nodes and from the hidden nodes to the output nodes. Third, there must be a set of communication links from the output nodes to the input nodes so that the network can receive information about its own behavior. Fourth, there must exist a node or set of nodes which monitor the output activations of the network to determine whether those values represent previously learned patterned information, and which can activate a "training mode" if the output is not patterned so that the network can learn a response to the new pattern. Fifth, there must be a set of nodes which encode a *pattern* or *goal state* which is associated with each input pattern, which is intended to serve as the appropriate output for that input. Sixth, there must be a defined *error function* which makes it possible to calculate the extent to which the pattern displayed by the output nodes differs from the goal state encoded in the pattern nodes. Seventh, the error function must be able to express the error as a function of the activation values of the nodes. Eight, the activation values of the nodes must in turn be expressible as functions of the weights or communication channels among the nodes. Ninth, there must be some active algorithm by which the network is able to modify its internal pattern of weights to reduce the errors. Tenth, the overall functional relations from input through hidden to output nodes must in general be non linear.

When these conditions are met, it will be possible for a network to receive information from its environment, form internal symbolic representations of that information, act (produce outputs), monitor its actions, and modify its actions if they are inappropriate. Networks which meet these conditions can not only learn about their environment in a passive way, but can actively modify their own configuration to produce desired outputs for given inputs.

To be sure, the implementation of sophisticated, intelligent, self referencing networks with sufficient capacity to exhibit interesting behaviors involves no minor technical accomplishment, particularly when simulated on Von Neumann architecture machinery. Nor are the difficulties solely technical. The monitor function described in the ROVER II model is quite rudimentary, and can only determine whether a proposed output is patterned or not. Much more sophistication is required from a model capable of interesting behavior. Certainly a more sophisticated monitor would take into account the appropriateness of the output for the circumstances under which it was proposed. An interactionist model would require as well that the network take into account the reaction of others in the communication situation. Cooley's (1903) "looking glass self" model requires that the reaction of others to one's behavior can result in "...pride, mortification or shame," which are emotions well in advance of ROVER II's crude capabilities.

Nonetheless, however crude the level of implementation, the fundamental architecture employed in ROVER II represents a useful first step. Indeed, while the limitations of ROVER II are severe and obvious, it does exhibit cognitive abilities that some have claimed distinguish human intelligence from machine intelligence: First, it can recognize limited language patterns and associate them with appropriate responses. Secondly, it is self-reflexive, and can observe its own activity and take that activity into account when determining its response. Third, it is recursive, and can revise a past judgment based on new information; that is, it need not always give the same response to the same input, but evaluates each input in light of its previous activity. Fourth, it is robust enough to provide the "correct" response even when the input is partially garbled or incomplete. Fifth, it can monitor its own internal cognitive state in a limited way, evaluate its potential activity and modify that activity based on that evaluation. And sixth, it can learn to associate new patterns through interaction with others. Since ROVER II can do all these things, yet is clearly not remotely "human", these characteristics can not be the essential characteristics which distinguish human intelligence from machine intelligence in a qualitative way. While it is impossible to rule out the possibility that there is a qualitative difference between what algorithms like ROVER II achieve and the actions of intelligent organic systems, it is important to point out that at least a major component of the difference in cognitive capacity between ROVER II and a simple organic intelligence is attributable to the sheer size differences between these systems.

Equally interesting is the potential impact such models might have on future developments of social network theory, particularly in the area of self referential organizations which may be structured in such a way as to maximize the development of a collective self awareness and collective self concept. Theorists particularly since Cooley (1903) have discussed the "we" as it develops in small, primary groups, but modern media and polling technologies give promise of developing such "primary" and self referential architectures for much larger sets of people. While far beyond the scope of the present paper, these issues promise real interest for future research.

REFERENCES

Cooley, C.H., Human Nature and the Social Order, New York, Charles Scribner's Sons, 1902.

Hebb, D.O.,, The Organization of Behavior, New York, Wiley, 1949.

Hinton, G.E., and T.J. Sejnowski, "Learning and relearning in Boltzmann Machines",in Rumelhart, D.E., and J. L. McClelland, (Eds.). Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Cambridge, MA, The MIT Press, 1986. pp. 282-317.

Johnson-Laird, P.N., Mental Models: Towards a Cognitive Science of Language. Inference and Consciousness, Harvard University Press, Cambridge, MA, 1983

McClelland, J.L., and D.E. Rumelhart, Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises, Cambridge, MA., The MIT Press, 1988.

McClelland, J.L., D.E. Rumelhart and G.E. Hinton, "The appeal of parallel distributed processing",in Rumelhart, D.E., and J. L. McClelland, (Eds.). Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Cambridge, MA, The MIT Press, 1986. pp. 3-44.

Mead, G. H., Mind, Self and Society from the Standpoint of a Social Behaviorist, C. Morris, (Ed.). University of Chicago Press, 1934.

Minsky, M., and Papert, S., Perceptron, Cambridge, MA, MIT Press, 1969.

Rosenblatt, F., Principles of Neurodynamics, New York, Spartan, 1962.

Rumelhart, D.E., G.E. Hinton, and R.J. Williams, "Learning internal representations by error propagation", in Rumelhart, D.E., and J. L. McClelland, (Eds.). Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Cambridge, MA, The MIT Press, 1986. pp. 318-362.

Rumelhart, D.E., and J. L. McClelland, (Eds.). Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Cambridge, MA, The MIT Press, 1986.

Simpson, G., Emile Durkheim: Selections from his work, New York, Crowell, 1963.

Smolensky, P., "Information Processing in Dynamical Systems: Foundations of Harmony Theory", in Rumelhart, D.E., and J. L. McClelland, (Eds.). Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Cambridge, MA, The MIT Press, 1986. pp. 194-281.

Woelfel, J.D.,"The Galileo System: A Theory and Method for Analyzing Cognitive Processes", in J.C. Mancuso and M.L.G. Shaw, (Eds.), Cognition and Personal Structure, New York, Praeger, 1987.pp. 169-193.

Woelfel, J.D., and E.L. Fink, The Galileo System: Theory and Method, NY Academic Press, 1980.