For presentation at the 2010 Fall Symposium Series of the Association for the Advancement of Artificial Intelligence, Complex Adaptive Systems: Resilience, Robustness, and Evolvability, Arlington, Virginia, November 11th – 13th, 2010.

Typicality Effects and Resilience in Evolving Dynamic Associative Networks

Anthony F. Beavers

The University of Evansville
1800 Lincoln Avenue
Evansville, Indiana 47722
afbeavers@gmail.com

Abstract

This paper is part of a larger project to determine how to build agent-based cognitive models capable of initial associative intelligence. Our method here is to take McClelland's 1981 "Jets and Sharks" dataset and rebuild it using a nonlinear dynamic system with an eye toward determining which parameters are necessary to govern the interactivity of agents in a multi-agent cognitive system. A few number of parameters are suggested concerning diffusion and infusion values, which are basically elementary forms of information entropy, and multi-dimensional overlap from properties to objects and then from objects back to the properties that define them. While no agent-based model is presented, the success of the dynamic systems that are presented here suggest strong starting points for further research in building cognitive complex adaptive systems.

Introduction

Traditional connectionist network modeling (with Artificial Neural Networks or ANNs) begins with a fixed network structure (in terms of nodes and connections) and then proceeds to discover a set of weights to transduce signals (here information), transforming them from input to output. The techniques for determining the proper weight set vary, but the general strategy remains the same. It starts with a predefined structure and then sets the weights for connections. Other strategies are possible. One could, for instance, let information content determine the structure of the network, rather than using a predefined structure, adding nodes and connections wherever dictated by the data. McClelland and Rumelhart's early Interactive Activation and Competition (IAC) Models (see McClelland and Rumelhart 1980, and Rumelhart and McClelland 1980) use this approach along with contemporary network structures that are involved in social network analysis, citation networks, co-authorship networks, and so forth. The success of these networks has been sufficient to warrant the claim that they exhibit rudimentary intelligence and qualify as preliminary forms of artificial intelligence, if harnessed to be so.

Dynamic Associative Networks (DANs) use this latter strategy as well. These networks are built up from the data to transform input into output in such a way that the output is predictive, that is, these models learn from experience to form associations based on some sort of statistical procedure that is implicitly determined by the model. (That is, no explicit statistical methods are used.) Developed in the Digital Humanities Laboratory at the University of Evansville over the past four years, DANs have been used in a variety of micro-world experiments, and exhibit cognitive abilities including 1) object identification based on properties and context-sensitivity, 2) comparison of similarities and differences among properties and objects, 3) shape recognition of simple shapes regardless of where they might appear in an artificial visual field, 4) association across simulated sense modalities, 5) primary sequential memory of any seven digit number (inspired by Allen and Lange 1995), 6) network branching from one subnet to another based on the presence of a single stimulus, 7) eight-bit register control that could perform standard, machine-level operations as with standard Turing-style computational devices, and 8) rudimentary natural language processing based on a stimulus/response (i.e. anti-Chomskian) conception of language. The guiding principle of this research has been the search for "initial intelligence," or a basic set of low level skills that might add up to explain larger scale cognitive initiatives.

Unlike ANNs, DANs learn by adding nodes and connections wherever needed. In more recent applications, we have reintroduced both weights and thresholds to try to improve cognitive function. At present, we are not prepared to offer anything close to an explanation of human cognition. Rather, this experimental enterprise has been dedicated solely to trying to determine which network shapes and control parameters *might* account for various cognitive skills. We have learned, for instance, that association of objects taking their proper context into account requires a "fat" network that uses a lot of overlap between "representations," while sequential memory requires a "slim" network with hardly any overlap.

In our latest experiments, we have been working to isolate control parameters to transform an ordinary database into a predictive mechanism in order to create content-addressable memory. Since a database can also be conceived as a network, this project is a continuation of our earlier work and not a departure from it. One hope of this research is to isolate relational parameters between nodes to create networks that can grow in an agent-based environment. At present, we have made no attempt to build this environment, since we have yet to isolate the precise set of parameters to govern interaction between agents. But some progress has been made. Thus, what follows must be regarded as a work-in-progress that may help us to learn to build evolving DANs.

To provide data for the experiment discussed here, we went all the way back to McClelland's original "Jets and Sharks" dataset from 1981 (see McClelland 1981) to see if we might accomplish using DANs what McClelland did with his IAC models. In so doing, we believe we have uncovered data implicit in McClelland's dataset that is not made explicit in his original approach. Since a network can also be thought of as an agent-based system, we believe further that this data might prove useful for laying out the parameters to turn our DANs into agent-based models. The purpose of this report is to provide some justification for these beliefs. We start here with a summary of McClelland's IAC model and then a description of our DAN model, which we have named "Dynamic Jets and Sharks." We then turn to discuss the discovered parameters in terms of 1) multi-dimensional overlap and 2) pattern diffusion and infusion (both to be defined subsequently) that together make considerable use of the notion of information entropy articulated in Claude Shannon's information theory (1949), though we will use a different entropy formula for reasons that will become clear. Pursuing this course goes part of the way in isolating the needed parameters to build agent-based DAN models that exhibit collective intelligence. These parameters track "typicality effects," (also to be defined subsequently) that should allow us to create self-corrective and hence resilient, agent-based models.

McClelland's IAC Model

The point of McClelland's IAC "Jets and Sharks" model (1981) is to show that it is possible to retrieve generalities about a dataset from only stored particulars about members of that set. In other words, coding a system with explicit rules about how to generalize is not necessary to arrive at generalities, if a proper network structure that relates those particulars correctly is implemented. The purpose of his 1981 paper is to lay out the details of his network implementation.

To facilitate explanation, it is perhaps easiest to start with the dataset itself, which consists of 27 labels (i.e., individual names) for members of two rival gangs (Jets or Sharks), their ages (20s, 30s or 40s), their highest educational level (Junior High, Senior High or College), their marital status (Single, Married or Divorced), and their occupation (Pusher, Burglar or Bookie) as indicated in the following table:

Name	**Gang**	**Age**	**Edu**	**M Stat**	**Occup**
Art	Jets	40s	J. H.	single	pusher
Al	Jets	30s	J. H.	married	burglar
Sam	Jets	20s	Col.	single	bookie
Clyde	Jets	40s	J. H.	single	bookie
Mike	Jets	30s	J. H.	single	bookie
Jim	Jets	20s	J. H.	divorced	burglar
Greg	Jets	20s	H. S.	married	pusher
John	Jets	20s	J. H.	married	burglar
Doug	Jets	30s	H. S.	single	bookie
Lance	Jets	20s	J. H	married	burglar
George	Jets	20s	J. H.	divorced	burglar
Pete	Jets	20s	H. S.	single	bookie
Fred	Jets	20s	H. S.	single	pusher
Gene	Jets	20s	Col.	single	pusher
Ralph	Jets	30s	J. H.	single	pusher
Phil	Sharks	30s	Col.	married	pusher
Ike	Sharks	30s	J. H.	single	bookie
Nick	Sharks	30s	H. S.	single	pusher
Don	Sharks	30s	Col.	married	burglar
Ned	Sharks	30s	Col.	married	bookie
Karl	Sharks	40s	H. S.	married	bookie
Ken	Sharks	20s	H. S.	single	burglar
Earl	Sharks	40s	H. S.	married	burglar
Rick	Sharks	30s	H. S.	divorced	burglar
Ol	Sharks	30s	Col.	married	pusher
Neal	Sharks	30s	H. S.	single	bookie
Dave	Sharks	30s	H. S.	divorced	pusher

Table 1: McClelland's Original Dataset

Before continuing, a few observations about the data is in order. First, there are three people in the dataset that have duplicate properties with others. Jim and George, John and Lance, and Phil and Ol form duplicate pairs. This will be important later, because eliminating duplicate property sets in the DAN models produce varying results. Second, even a cursory visual glance at the table shows that Jets tend be younger and less educated than Sharks, only four members are divorced, only four are in their 40's, etc. The "magic" of both models will be to find a statistical way through the dataset, taking into account such imbalances, even though neither model actually does statistics understood as a mathematical method for arriving at results. In fact, the power of both models is that they show that networks can march in step with statistical methods, perhaps explaining how something like a brain might be able to make inferences concerning generalities when confronted with incomplete information. Consequently, if the dataset were complete, that is, if every possible property set (i.e., all 162 combinations, leaving the names aside) were represented in the table only once, neither model would be able to generalize, since all results would tie. This observation underscores the probabilistic nature of the two models.

McClelland's version consists of a network made of 68 nodes, each of which is "a simple processing device which accumulates excitatory and inhibitory inputs from other nodes continuously and adjusts its (real-valued) output to other nodes continuously in response" (170). Each individual in the set is called an "instance" by McClelland, and there is one instance node for each of the 27 members. The remaining 41 nodes are designated as "property nodes" and are divided into "cohorts" or groups, one of which contains 27 nodes for the names in the set, and others containing 2 nodes for the gangs, 3 nodes for the ages, 3 for the educational levels, etc. Importantly, each of the nodes in a cohort is mutually inhibitory, that is, for instance, if one of the age nodes activates, it sends inhibitory signals to the other two age nodes, shutting them down. The instance nodes also form a mutually inhibitory cohort. (As we will see shortly, the DAN model does not need to use inhibitory connections to shut down signals in the same cohort, since this ability emerges naturally from the arrangement of nodes in the dynamic network.)

According to McClelland, "The system's knowledge of an individual consists simply of an instance node and a set of bi-directional excitatory links between it and the nodes for the properties that individual is known to have" (170). To pick the first example in the list, there is an instance node for Art, which shares mutually excitatory connections to the nodes representing "Jets," "40s," "Junior High," "Single," and "Pusher."

Each node starts with a value of 0. Activation of the network consists of probing a property or a set of properties, which initiates a pattern of spreading activitation that will ultimately settle into a state of equilibrium, after which the resulting values can be read off of each of the nodes. Further mathematical details exceed the scope of the current exposition, though they are presented in McClelland's paper.

As an example of network behavior, McClelland indicates that activating the "Jets" node will, after two hundred cycles or so, reach a state of equilibrium showing the following winning values: .869 for Jets, .663 for 20s, .663 for Junior High, .663 for Single, and .334 for each of the three occupations that tie at that value. Consequently, McCelland notes:

> Though no Jet has all three of these properties, 9 out of 15 of the Jets are in their 20s, 9 have only Junior High educations, and 9 are single. The occupations are divided evenly among the three possibilities. Thus, the model tends to activate the node on each dimension which is most typical of the members of the gang, even though it has never encountered a single instance with all of these properties, and has no explicit representation that the Jets tend to have these properties. (171)

We will notice similar, though not identical, behavior with the DAN model, even though it uses a different architecture. As a hint of things to come, I will go ahead and show the differences for this example now, saving explanations for the discussion section below.

Node	**IAC**	**DAN w/ duplicates**	**DAN w/o duplicates**
Jets	.869	.269	.279
20s	.663	.237	.235
J. H.	.663	.233	.231
Single	.663	.191	.201
Pusher	.334	.152	.175
Burglar	.334	.141	.104
Bookie	.334	.155	.166

Table 2: Initial Comparisons between IAC and DAN Models

Both models pick out the same age, educational level and marital status, but as is clear in the above table, the DANs favor the occupation "Bookie" when duplicates are included and "Pusher" when not over and above the other two possibilities. Why this is so will become clear later, but one should already suspect that the DAN models are tracking something that the IAC model is not.

McClelland offers two other tests of his network's behavior. One involves activating an arbitrary conjunction of the available properties and the other retrieving the properties of an individual by name. Both of these will correlate with the DAN model in interesting, but different, ways.

Dynamic Jets and Sharks

Dynamic Jets and Sharks (DJ&S) is a multi-layer network (or set of networks depending on how one wishes to classify things) that is built up from McClelland's dataset. As with his IAC model, DJ&S does not start with a predefined network structure so much as the network is precisely the data interconnected in a particular way.

For ease of exposition, it might be helpful to think of DJ&S as two networks, a properties network and an objects network. In McClelland's language, the object network here is used to manage "instances." Unlike McClelland, however, DJ&S does not group its property nodes into cohorts and, thus, no inhibitory connections are used to shut down nodes in the same cohort. Rather, DJ&S simply eliminates competing members of the same cohort by itself. Another important difference is that DJ&S does not contain a set of property nodes for the names of individuals in the dataset. Rather, these are used to identify the objects in the object network.

The properties network consists of several layers that might be easiest imagined as rows and columns in a spreadsheet. There is one row for each property in the five groups: Jets, Sharks, 20s, 30s, 40s, Junior High, Senior High, College, Divorced, Married, Single, Pusher, Burglar, and Bookie. As with the McClelland network, in the initial condition all values are set to 0. Activating the network is done by "probing" a single property or any arbitrary group of properties, which assigns a value of 1 to any activated property. Activation then passes through the object network (details to come) which dynamically determines a set of weights that will then be passed back to the properties network and summed to arrive at the system's output. The winners are then determined by inspection of the network's output layer.

As with the properties network, the object network is also multi-layered and might also best be thought of as rows and columns in a spreadsheet. Here, there is one row for each of the members in the dataset starting with Art and ending with Dave as indicated in Table 1 above.

Determining the Weights in the Object Network

In the case of both networks, the various cells represent nodes in the network and the summation equations represent connections. Activation passes directly from the property inputs to the first layer of the object network by hard coding each property with its associated member. Thus, in the first instance, Jets, 40s, Junior High, Single, and Pusher are connected to Art; in the second instance, Jets, 30s, Junior High, Married, and Burglar are connected to Al, and so forth. Additionally, each member in the object network is connected back to its appropriate set of individual properties. (Before continuing, it is worth mentioning that the minimal details of the networks thus described are sufficient to produce the same set of winners as in McClelland's original model, but we are hoping here to do more.)

Subsequently, we use an information entropy formula to weight the value that activation of a property node passes to its corresponding object nodes. Using the entropy formula suggested by Shannon and Weaver (1949) caused the system to produce erratic and unintuitive results. However, testing showed that a simpler entropy formula based solely on the reciprocal of the count (1/number of output nodes) functioned quite well. To avoid confusion and any suggestion of correlation between information and the laws of thermodynamics that figures so deeply in Shannon's work, we call our entropy values "diffusion" values. In straightforward terms, if a particular property is linked to 9 objects, then the weight that that property node passes to each of the objects to which it connects is 1/9 or .1111. The complete list of diffusion values for the 14 properties is thus:

Property	Count	Diffusion
Jets	15	.0667
Sharks	12	.0833
20s	10	.1000
30s	13	.0769
40s	4	.2500
Junior High	10	.1000
Senior High	11	.0909
College	6	.1667
Divorced	4	.2500
Married	10	.1000
Single	13	.0769
Pusher	9	.1111
Burglar	9	.1111
Bookie	9	.1111

Table 3: Diffusion Values for the 14 Properties

Diffusion values make intuitive sense, when we consider that the more a node passes its activation to other nodes in the network, the more its impact decreases. Diffusion thus is a measure of the uniqueness of a property as distributed in the dataset: the more unique a property is the higher its impact. This fact explains in part why Pusher, Burglar, and Bookie do not tie in the DAN models represented in Table 2 above.

To see this play out in the concrete, we return then to the case of Art mentioned above, who had the properties of being a Jet in his 40s, who attended Junior High, is single and a pusher. The total excitation on the Art node is thus, according to Table 3 above:

$$.0667 + .2500 + .1000 + .0769 + .1111 \text{ or } .6047$$

Excitation values are simultaneously calculated for each of the other members in the dataset. Al for instance shows an excitation value of .4547, Sam shows .5214, etc.

The goal of calculating values for the members of the object network is to arrive at a dynamically-determined set of weights for each member that can be passed back to the properties network. Empirical testing of the model showed that excitation values by themselves produced meaningless results. Thus, the network was designed to do some normalizing work on the excitation values to arrive at the needed weights. The process is fairly straightforward.

Operating on a similar principle of diffusion, we first use an infusion formula that is again the reciprocal of the count, except here the count represents not the number of output nodes, but the number of input nodes summed to arrive at the excitation value. Hence, the formula is simply 1/number of input nodes. Since each of the members in the object network has precisely five properties, the infusion values for each of them is the same: 1/5 or .2000.

One may fairly wonder at this point why they need to be used at all, if the infusion values are the same for each member of the object network. The answer is that while they do not make a quantitative difference in the output of the object network (explanation to follow momentarily), they allow us to use a parallel strategy in determining the output values in the property network. There, as we shall see, the diffusion values will all be the same, while the infusion values differ. The point of keeping the two networks parallel where the calculation of diffusion and infusion values are concerned is that we are seeking the simplest number of parameters to implement the DAN models in an agent-based system. Infusion is not necessary in the object network as diffusion is not necessary in the property network. But in neither case does their inclusion interfere with the system's performance. So, if no harm comes by keeping the two networks parallel, then parsimony would suggest using them anyway.

To arrive at the normalized weight for each member of the object network, we, therefore, multiply the excitation value by the infusion value (here .2000) to arrive at an activation value. Then to arrive at a normalized weight for each member in the set, we divide the activation value by the peak activation for the member, where peak activation is defined as the activation value that is shown when all and only the properties for the member in question are activated on the first level of the properties network. The result is a weight that represents the percentage of activation for each member in the object network. It should be clear, of course, that when the properties of Art are activated, the weight for Art will be 1.0000, and similarly for each of the others.

Importantly, the weights for each of the members in the object network vary dynamically depending on which property nodes, whether singly or in combination, are activated. This is demonstrated for activation on the Jets node and then the Sharks node in the following table. (For clarity, the Jets and Sharks nodes were *not* activated together.)

Name	**Property**	**Weight**
Art	Jets	.1100
Al	Jets	.1463
Sam	Jets	.1275
Clyde	Jets	.1100
Mike	Jets	.1541
Jim	Jets	.1059
Greg	Jets	.1415
John	Jets	.1391
Doug	Jets	.1574
Lance	Jets	.1391
George	Jets	.1059
Pete	Jets	.1493
Fred	Jets	.1439
Gene	Jets	.1275
Ralph	Jets	.1541
Phil	Sharks	.1552
Ike	Sharks	.1864
Nick	Sharks	.1902
Don	Sharks	.1552
Ned	Sharks	.1552
Karl	Sharks	.1314
Ken	Sharks	.1807
Earl	Sharks	.1314
Rick	Sharks	.1364
Ol	Sharks	.1552
Neal	Sharks	.1902
Dave	Sharks	.1364

Table 4: Object Weights Derived from the Activation of a Single Property

The fact that the weights vary in the third column of the table, even though the input value was 1 in all cases, illustrates the important point that uniqueness is being taken into account by the network. The higher the weight, the more the property can be said to define the member in question. Thus, being a Jet is more central to Doug's identity than it is to Jim's. Indeed, 15.74% of Doug's identity is accounted for by his membership as a Jet, while only 10.59% of Jim's identity is accounted for by his gang membership.

Obviously, the numbers go up when entire property sets are activated for the individual members. A few samples will prove illustrative:

Name	**Art**	**Doug**	**Ralph**	**Dave**
Art	**1.0000**	0.2374	0.5864	0.1836
Al	0.3663	0.3157	0.5358	0.1694
Sam	0.2752	0.4880	0.2752	0.0000
Clyde	0.8164	0.4210	0.4028	0.0000
Mike	0.5643	0.7683	0.7428	0.1784
Jim	0.2651	0.1059	0.2651	0.3981
Greg	0.3799	0.3362	0.3799	0.4312
John	0.3483	0.1391	0.3483	0.0000
Doug	0.3396	**1.0000**	0.5219	0.3976
Lance	0.3483	0.1391	0.3483	0.0000
George	0.2651	0.1059	0.2651	0.3981
Pete	0.3221	0.7755	0.3221	0.2043
Fred	0.5713	0.5264	0.5713	0.4534
Gene	0.4880	0.2752	0.4880	0.2128
Ralph	0.8216	0.5110	**1.0000**	0.4357
Phil	0.2063	0.1431	0.3494	0.5046
Ike	0.3951	0.5915	0.5670	0.3571
Nick	0.4282	0.5569	0.6036	0.8246
Don	0.0000	0.1431	0.1431	0.2974
Ned	0.0000	0.3494	0.1431	0.2974
Karl	0.3934	0.3179	0.0000	0.2738
Ken	0.1667	0.3636	0.1667	0.3766
Earl	0.3934	0.1432	0.0000	0.2738
Rick	0.0000	0.2745	0.1258	0.8186
Ol	0.2063	0.1431	0.3494	0.5046
Neal	0.1754	0.8109	0.3508	0.5718
Dave	0.1814	0.2745	0.3072	**1.0000**

Table 5: Object Weights Derived from the Activation of Whole Property Sets

Further comment here is necessary. The activation of all the properties for Doug, for instance, produces the scaled weight set that appears under his name above. The clear winner in that set is, of course, Doug with a value of 1.0000. But Doug shares four properties in common with Mike, Pete, and Neal, who have activation values of .7683, .7755, and .8109, respectively. Neal rises over Mike and Pete because a greater percentage of his pattern is completed at 81.09% taking the uniqueness of his properties and his peak activation value into account. In this case, Neal is more unique (less typical) than the other two. Hence, he stands out a little more.

Playing with the parameters could, in most cases, invert the stack and have the more typical members rise to the top rather than the less typical. This possibility and the fruits that it will bear will be studied in further research. However, it is important to point out that we are dealing here with a probabilistic and nonlinear model, thus making inferences of this sort not intuitively obvious, and further examination will be necessary before determining the precise manner in which uniqueness is taken into account by the network. For instance, in the case of Ralph, we find that he shares four properties with Art and Mike, each of whom provide weights of .5864 and .7824, respectively. Yet, Nick, with whom Ralph shares only three properties, provides a weight of .6036, falling in between the weights of Art and Mike when Ralph's pattern is fully activated.

Whatever the verdict on the uniqueness issue may be, we have an explanation here of how the object network's output weights are dynamically-determined for each member in the object network. Several methods were tried in the process of arriving at this procedure, and the sole criterion for choosing this one over the others is that it works where the others did not, though I will have to say later how this measure of success was determined. In any case, the next step in the process is to feed these weights back into the properties network.

Dynamically Weighting the Properties Network

Once the weights are determined for each member of the object network, they are then passed back to the properties network following a casual version of Hebb's principle. In the concrete, the properties that were activated to produce Art's complete pattern (and hence his output weight) now become the recipients of Art's output weight in another layer of the properties network. Thus, whatever Art's output weight may turn out to be during a particular activation cycle is distributed to nodes representing Jets, 40s, Junior High, Single, and Pusher, and similarly for each of the other members in the object network. In this way, the properties that define a member of the object network are themselves redefined downstream by the very objects they defined earlier in the process. Thus, the property to object relation becomes, in a sense, a clustering procedure that activates each of the pertinent properties that make up an object in exactly the same way and at the same time.

As before, a diffusion value is used, but since each object here is defined by five properties, the value is in all instances the same, that is, .2000. (In the case of the properties network, the infusion values will differ, where the diffusion values remain the same, and hence, the latter do not affect network performance, but again, for reasons of parsimony rather than function, we use them anyway.) The output weight from the object network is then multiplied by the diffusion value to determine the input weights that are then pushed back into the properties network. The table below shows a partial representation of the weights used to derive the output shown in Table 2 above for the DAN model with duplicates.

Name	**Jets**	**20s**	**J.H.**	**Sing**	**BK**
Art	.0220		.0220	.0220	
Al	.0293		.0293		
Sam	.0255	.0255		.0255	.0255
Clyde	.0220		.0220	.0220	.0220
Mike	.0308		.0308	.0308	.0308
Jim	.0212	.0212	.0212		
Greg	.0284	.0284			
John	.0278	.0278	.0278		
Doug	.0315			.0315	.0315
Lance	.0278	.0278	.0278		
George	.0212	.0212	.0212		
Pete	.0299	.0299		.0299	.0299
Fred	.0299	.0299		.0299	
Gene	.0255	.0255		.0255	
Ralph	.0308		.0308	.0308	
Excitation	.4035	.2371	.2329	.2478	.1397
Infusion	.0667	.1000	.1000	.0796	.1111
Output	.0269	.0237	.0233	.0191	.0155

Table 6: Partial Calculations for the Properties Network based on the Activation of the Jets Node Alone

The infusion values used here are the same as the diffusion values represented in Table 3 above. This stands to reason, since we are here putting back together what the object network took apart. In this way, we might think of the relationship from properties to objects and then from objects to properties as a procedure that unifies and diversifies properties based on their association with objects.

To match the output above with the figures in Table 2 above, excitation is multiplied by infusion to produce the output. (For convenience sake, in the last moment, we multiplied the result by ten just to shift values one place to the left. Doing so does not in any way affect the performance of the network.)

Discussion

Human cognition is glitchy, and, as a consequence, it is difficult to know when a cognitive model is successful or not. Even in cases where models succeed and fail at the same tasks as human beings, it is difficult to know whether this is due to a proper understanding of the computational and biological processes involved or to imperfections in our biological wetware (or unrealistic perfections in our computer models!). That a human being might require several exposures to a given stimulus where its counterpart model does not tells us very little, that is, unless we look under the hood, as it were, to see what the brain is actually doing. Currently, we still lack the fine-grained imaging technologies to do this, and so extrapolating from computer models to biological cognition will have to wait. Knowing what constitutes a successful biological model will also have to wait. But within limited parameters, we can gauge whether a network's response to a given input or set of inputs makes sense. Can the network respond with some semblance of intelligence? This criterion then is what we use as a mark of success. We can also compare the output from the DAN models to that of McClelland's IAC model. In the following tables, we look at the other two examples provided by McClelland in his original paper.

The first case considers what happens when the network is probed by activating two property nodes simultaneously, in this case the nodes for "20s" and "Junior High":

Node	**IAC**	**DAN w/ duplicates**	**DAN w/o duplicates**
Lance	.127	.4184	.4202
John	.127	.4184	Duplicate
Jim	.094	.3185	.3113
George	.094	.3185	Duplicate
Jets	.732	.4710	.4660
20s	.855	.5470	.5330
J. H.	.862	.5420	.5330
Married	.589	.2540	.2190
Divorced	.389	.3190	.2080
Burglar	.721	.4250	.3410

Table 7: Comparison of the IAC Model with Two DAN Models for Simultaneous Activation of Two Properties

Generally, the values of the various models are commensurable with the following exceptions: as opposed to the IAC model, the DAN model with duplicates prefers Divorced over Married. Second, unlike the IAC model, the DAN model without duplicates prefers Single (not shown) with a value of .3590. In the former case, Single is almost preferred with an output value of .3180 (also not shown). Given that Lance and John are married and Jim and George are divorced, the mismatch between the two cases might seem understandable. However, both DAN models lean in the direction of Single with Divorced barely topping Single in the case with duplicates. Inspection of activation patterns in the properties network, however, shows that activating both "20s" and "Junior High" activates most of the Jets and hardly any of the Sharks, and since Jets tend to be Single the output pattern is pulled in that direction. (9 Jets are single, 2 are divorced and 4 are married.) Once again, however, the nonlinear nature of the DAN models makes tracking the precise explanation for the differences difficult without mathematical analysis which we hope to do in future research.

McClelland's second case involves severing the link between Lance and his occupation as Burglar to show that the network can fill in missing information. Here, the DAN with duplicates performs equivalently where picking out the winners is concerned, but the Dan without duplicates is slightly off by comparison.

Node	**IAC**	**DAN w/ duplicates**	**DAN w/o duplicates**
Lance	.779	.9592	.9496
Jets	.710	.9110	.9130
20s	.667	.9970	.9850
J. H.	.704	.9890	.9770
Married	.552	.8360	.8370
Divorced	.347	.4240	.2710
Burglar	.641	.6900	.5360

Table 8: Output Results with the Connection between Lance and Burglar Severed

In the case of the DAN without duplicates the winner is not the Burglar, but the Pusher at .5670, which is close but still not correct. Again, it is difficult to assess precisely what this means, since removing the duplicates does not, in effect, have us working with McClelland's dataset. Further mathematical analysis will be necessary to determine precisely what is going on here.

In this light, it is worth pointing out that while both models are probabilistic, McClelland's model is linear whereas the DAN models are nonlinear. This difference is significant when it comes to trying to determine the parameters for an agent-based complex adaptive system where nonlinear behavior is a requirement.

Still, when both DAN models are probed using the complete property sets for each member of the object network, the output is correct 100% of the time. Additionally, the networks are predictive in the sense that their output instructs the user on which nodes to activate next. Thus, in the DAN with duplicates, if the user starts by activating the node for Jets, the network instructs the user to then select 20s, following which it asks for Junior High, then Burglar, then Divorced, after which it presents Jim and George as the winners, each indicating that they both share all five of the selected properties, which they do. Similar patterns emerge when starting with different properties as the network tries to find that property set that is most distributed across the object network based on the initial conditions. Clearly something is going on here that is worth further investigation.

Another effect worth mentioning is that where McClelland's network required that individual nodes of the same cohort needed to be inhibited upon activation of another node of the same cohort, nothing of the sort is necessary here, and only rarely does the network request activation of members of the same cohort, though it does on certain occasions. This effect must be regarded as something of a cognitive success, since teaching a child that Rover is a dog does not involve teaching her that Rover is not a cat, not an elephant, not a sheep, etc.

Finally, it is worth pointing out that McClelland's network goes through several cycles before settling into a state of equilibrium, around 200 in the first example reported above. All of the results reported for the DANs above have been based on a single pass through the network, and it would be interesting to see what would happen if the output results were continually recycled through the network. Will the network become chaotic or will it land in a stable state? Again, the nonlinear nature of the models makes it difficult to say short of trying, offering us another opportunity for further research.

Conclusions

There is something quite provocative about the DAN models regarding their simplicity and the fact that they do not need to be trained. Weights are dynamically-determined based on diffusion and infusion values which are elegantly derived on the basis of the reciprocal of the count for the number of nodes responding, in the case of diffusion on the number of nodes to which a given node connects and in the case of infusion on the number of nodes that connect to a given node. Thus adding or removing objects and properties does not involve retraining the network, as is often the case with connectionist models.

Second, there is also the elegant simplicity of allowing objects, as collections of properties, to determine the clustering of those same properties. Indeed, this is what was meant by the multi-dimensional overlap indicated at the start of this paper. Basically it amounts to defining objects in terms of their properties and then redefining those properties in terms of the objects that they earlier defined.

The entire behavior of the DAN models can be comprehended in these terms alone: diffusion and infusion values and the multi-dimensional overlap between properties and objects. Whether these interactive parameters will prove effective in building agent-based cognitive models that grow based on their experience can only be determined on the basis of further testing.

Acknowledgements

I would like to thank Michael Zlatkovsky and Derek Burrows for their help in implementing dynamic associative networks on various platforms. Matthias Scheutz and Randy Beer were both helpful in isolating mathematical equivalences between DAN models in general and their connectionist counterparts. Selmer Bringsjord, Mark Valenzuela and Talitha Washington were instrumental in helping me understand the characteristics of linear and nonlinear models and their relationships with probability. Ted Charmichael and Colin Allen were both worthy interlocutors along the way. Finally, Dixrek Powell was helpful in the preparation of the final stages of this manuscript.

References

Allen, C. and Lange, T. 1995. Primary Sequential Memory: An Activation-Based Connectionist Model. *Neurocomputing* 11(2-4): 227-243.

Beavers, A. 2010. More Fun with Jets and Sharks. The North American Computing and Philosophy Conference, Carnegie Mellon Univ., Pittsburgh, Penn., July 24-26.

McClelland, J. and Rumelhart, D. 1980. An Interactive Activation Model of the Effect of Context in Perception, Part I. Chip Report 91, Center for Human Information Processing, Univ. of California, San Diego.

McClelland, J. 1981. Retrieving General and Specific Information form Stored Knowledge of Specifics. In Proceedings of the Third Annual Conference of the Cognitive Science Society. Berkeley, Calif.

Rumelhart, D. and McClelland, J. 1980. An Interactive Activation Model of the Effect of Context in Perception, Part II. Chip Report 95, Center for Human Information Processing, Univ. of California, San Diego.

Shannon, C. and Weaver, W. 1949. *The Mathematical Theory of Communication*. Urbana, Illinois: Univ. of Illinois Press.