THE SEMANTIC NETWORK PROGRAMMING ENVIRONMENT

M. Bennett

1 Who we are

Accuro Information Engineering is a small start-up company providing a range of bespoke software products for business and industry. The Semantic Network programming concept is currently under development as the company's 'next-generation' product base. This paper gives an outline of the key concepts, and looks at some of the opportunities and constraints, in using this concept in the field of business decision-making.

2 Semantics in Cognitive Psychology

Concept

Semantics, in the context of the human mind, is the implementation of meaning. Research in this area has shown that meaningful points in the mind are connected together as a network in which the nodes represent concepts. Relations among these nodes are labelled and directed. An important point here is that information is not moved around in the brain, but resides at unique locations. The experience of meaning arises from the activation of points at those locations. The meaning represented by a node arises as a pattern of relationships between this node and others (1).

This concept derives from research into the approximate response times of subjects to verify statements such as 'A canary is yellow', 'A robin is a bird' etc. This research suggests that knowledge is stored in the mind in an apparently hierarchical manner (2). Very generally, knowledge is represented in the mind in a network of directional interrelations, with weightings.

This model of Semantic Networks affords some intriguing echoes of what we know of neurons, and of the nature of brain, in which mind is implemented. Neurons are known to be highly interconnected. These connections are directional. The strengths of the connections between neurons are weighted, these weights being adjusted according to a Hebbian Learning Function (3).

Scripts and Schemas

The basic network concept outlined above is fine for the representation of discrete nodes of meaning. However, it does not deal with the representation of higher levels of structure.

A number of concepts have been introduced by theorists to account for higher levels of concept representation. These include Schemas, Scripts and Frames (4). Schemas are data structures for representing the generic concepts stored in memory. There are schemas for situations, events, actions etc. Schemas are re-usable structures, which are chosen according to which appears most likely to fit the incoming data.

Scripts and schemas form part of our cultural heritage, and may vary between cultures. They can often be observed, at play as it were, in dreams, where the variables may have quite unlikely points connected to them.

Accuro Information Engineering
177 Kingston Road
LONDON SW19 1LH
email: info@accuro.demon.co.uk
URL: http://www.ibmpcug.co.uk/~accuro

Activation in Semantic Networks

Activation spreads between points in semantic networks. This activation does not represent the movement of information within the network. Information resides at unique locations, by virtue of the fact that each node represents a concept. Activation in the network is not information as we understand it, but information about information (meta-information).

2 Neural Computing

A rigorous account of the state of neural computing is beyond the scope of this paper. However, there are a number of key concepts and architectures which are worth defining, as they are relevant to the Semantic Network model.

For a more detailed account of the concepts of Neural computing, particularly in relation to symbol processing applications, see (5).

Neurons

A neuron can be simply represented as a device which performs a summation of a number of inputs and a number of weights associated with them, and generates an output.

In the case of the McCulloch Pitts model (6), a digital output is given when the weighted summation of the inputs exceeds a predefined threshold. Other neuron models have a variable, or analogue output.

Architectures

There are a number of different neural network architectures. An important distinction is between feed-forward networks and autoassociative or feedback networks.

Learning

There are 4 basic types of learning:

Supervised Training: This may be carried out on a feed-forward network. Supervised learning may use the popular back-propagation algorithm.

Reinforced Training: Global reward and/or punishment is used to give a more 'real-time' learning environment than would be achieved with supervised training.

Unsupervised Training: Unsupervised training typically takes place on feedback network types. Such networks are able to detect patterns in the data presented to them. Stored patterns can be retreived, on presentation of part of the stored data (Content Addressable Memory).

Hebbian Learning: A special case of unsupervised learning is Hebbian learning. In this, the weighting of a connection between two neurons is incremented if the neurons on either side of the connection (synapse) are active simultaneously.

Attractors

The autoassociative or feedback type of network is capable of forming attractors in neural space. Thus a Hopfield network is characterised by having a number of 'local minima', stable states to which the network will converge (7).

The concept of attractors in a meaningful space has been utilised for example in research on dyslexia. In their work on simulating dyslexic behaviour by damaging networks of artificial neurons, (8), Hinton, Plaut and Shallice produced sets of neural networks in which the neural elements represented inputs (letters and phonemes), and arbitrarily defined internal semantic states. These networks were able to converge on points representing words and concepts.

3 The Semantic Network Programming Environment

The SNPE is an attempt to make use of some of the functions and facilities of Neural Networks, while allowing the user to carry out programming by means of symbolic manipulation. Although it is underpinned by neural functions, the SNPE is primarily a symbol processing language.

"Behaviour Orientation"

Let us take a brief look at the programming environment itself: The programmer is faced with an array of elements, each of which relates directly or indirectly to system behaviour. The best way to envisage this "behaviour orientation" concept is to look at a simpler equivalent: the PLC programming environment.

PLCs provide a means of implementing process control (with its high numbers of field inputs and outputs) on conventional linear processors. The CPU addresses a high amount of field I/O, treating these essentially as memory locations.

In order to provide a programming interface which would be familiar to process engineers, a language was devised which could be programmed using relay functions. This is known as Ladder Logic - so called because the lines of code resemble a ladder when they are seen together on screen. In ladder logic, lines known as 'Rungs' contain elements representing relay contacts. At the far end of each rung sits a relay coil.

Other types of programmer interface for PLCs include schematic languages such as the Grafcet standard (9), and the International Standard IEC 1131 Part 3, which supports re-usable Function Blocks (10)

In each of the above, the programmer uses a graphical language in which program elements directly control field I/O or internal states.

Learning

The Hebbian model of learning is utilised in the SNPE. In this process, the weighting parameter for a synapse is incremented as a result of simultaneous excitation of the presynaptic and postsynaptic cells. In the SNPE, this essentially entails training the system at a symbolic level.

The Semantic Network Programming Environment

Programming may be carried out at neural or schematic level.

Neural Level: Before any application can be written, code must be created for it at neural level. The neural level interface comprises a visual display of individual elements, with input addresses and weightings.

Schematic Level: The programmer is provided with an interface in which discrete blocks of code can be manipulated. The neural connections within these blocks are not shown; instead a box is shown with a number of inputs and outputs defined. The programmer is then able to take specific connections within the application as a whole, and connect them to these.

The schematic approach gives us a mechanism for the implementation of scripts and schemas in the application. Any application requirement which can be broken down into schemas may be implemented as schematic blocks, provided that the internal code for the SB can first be written at the neural level.

4 Potential Applications

The full range of possible applications is dependent upon the still largely unexplored relationship between available processing power, and the implementation of the programming concept. Some possible applications may include remote machine control, process plant monitoring, information-gathering 'agents', and artificially intelligent agents within a company environment. Applications in the sphere of business management are realistic.

5 Semantic Networks in Business Management

Given a company or corporate environment in which the bulk of the organisational processes are controlled and represented in a software environment, it should be possible to interface these information structures to a semantic network system. The system can then be programmed to give pre-determined behaviours or recommendations in the light of real-time data, or to form patterns from the data which is presented to it, and act upon these as necessary.

The challenge here is to determine the data interface parameters required by the Semantic Network application(s) in order to operate effectively, and the programming requirements for the network itself.

Symbol Grounding

Since information is not moved around in the network but resides at unique locations, points must be created at the boundary of such a system, which will enable the system as a whole to perform apparently meaningful operations.

We need to look a little more closely at this concept of meaning, with reference to semantic networks.

Our definition of meaning is this:

1 A point on a semantic network can be said to have meaning by virtue of its connections to other points, provided that these points can also be said to have meaning.

2 Points on the network ultimately connect to points which represent actual physical inputs and outputs.

In other words, two independent mechanisms give rise to the phenomenon of meaning: relations to other points, and relations to the outside world.

In an artificial SN system, there is no experience of meaning. What is required instead is the illusion of meaning. That is, regardless of the internal states of the network, or anything that philosophy may ascribe to it, the network has to appear to be performing meaningful operations.

This gives rise to the concept of 'Virtual Meaning'. Virtual meaning is the appearance of meaning generated at the boundary of a SN system.

Example: a friend of mine has a fish, which lives in a tank. She tells me that when the fish wants to be fed, it looks at her through the corner of the tank. Now, you know and I know that the fish cannot see her, because of total internal reflection. What it sees, since it is in the corner of the tank, is a fish. Itself.

Let us assume however that the fish does at least realise that when it wants to be fed, it goes to a certain corner of the tank, and looks at a fish that it finds there.

This scenario gives rise to three experiences of meaning:

> To the fish, there is a mystery fish in the corner of the tank, which it can go to when it wants to be fed. Going to the corner-fish means that it will be fed.
>
> To you and me, the fish looks at its own reflection.
>
> To my friend, the fish looks at her to 'tell' her that it is hungry.

At the semantic boundary a set of points is shared between different networks (brain/machine; human/fish) which may mean different things on either side of the boundary. If these points are interpreted on one side as symbols, these symbols can appear to have been manipulated in useful ways.

Data Types at System Boundary

Information is not stored or moved around on the Semantic Network. Points at the boundary, which have meaning to the user, must be encoded in terms of pure boolean true/false states. Analogue variables, where required, must be broken down into piecewise linear approximations.

The system must be capable of presenting boundary information in an apparently meaningful logistical domain. The data being represented may originate from databases or spreadsheets. The bulk of the data which might be usefully read and acted on by a semantic network system would take the form of arrays of outputs derived from spreadsheets or database applications, organised specifically for the purpose - just as one might output data to a graph.

Another boundary option would be electronic management utilities like electronic forms, such as those available from Accuro. These may be provided with 'keys' to set bits in a semantic network input array. There are already products designed specifically to output financial markets data to a format readable by neural nets.

Another key area to be addressed is the organisational processes themselves. A business or organisation may be viewed as a network of recurrent conversations (11). These conversations should have been taken into account and realistically modelled in the course of computerising the business.

Assuming that the organisational processes have been adequately modelled and reflected in existing computer systems, the Semantic Network application should be able to take data generated by these systems and perform apparently meaningful operations. The organisational 'conversations' are essentially schemas, with generic inputs and outputs, and so should be amenable to being implemented as Schematic Blocks in the SN application.

6 Other Possibilities

Semantic Networks provides an environment in which it should be possible to implement Genetic Algorithms. Genetic algorithms provide a powerful means for making systems sensitive to changes in their environmental parameters. Since every element of semantic code represents something in terms of the system's behaviour, rearranging these by means of GAs should not cause areas of code to become meaningless, but instead alter their fitness in terms of the behaviour of the system.

Fuzzy logic is inherent in the options available to the programmer: by setting the weightings and numbers of connections between points, fuzzy relations can be created.

7 Conclusions

The use of the Semantic Network Programming Environment, while not the most powerful means of getting a machine to crunch data, provides a powerful and realistic way forward in modelling and controlling organisational parameters, many of which could previously only be manipulated by human minds.

References:

(1) Rumelhart, D.E. and Norman, D.A. (1983) Representation in Memory. CHIP Technical Report (no 116)). San Diego: Center for Human Information Processing, University of California: (C) John Wiley and Sons, Inc 1985.

(2) Quillian, M.R. (1968) Semantic Memory. In M. Minsky (Ed.), Semantic Information Processing. Cambridge, Mass: MIT Press.

(3) Hebb, D.O. (1949) The Organisation of Behaviour. New York: Wiley.

(4) Rumelhart, D.E. and Ortony, A. (1977) The representation of knowledge in memory. In R.C. Anderon, R.J Spiro, & W. E Montague (Eds.), Schooling and the acquisition of knowledge. Hillsdale, N.J.: Lawrence Erlbaum Assoc.

(5) Aleksander, I. and Morton H.B. (1993) Neurons And Symbols, The Stuff That Mind Is Made Of. London: Chapman Hall.

(6) McCulloch, W.S. and Pitts, W.H. (1943), A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5, 115-133.

(7) Hopfield, J.J. (1982) Neural networks and physical systems with emergent collective computational abilities. Proc. National Academy of Science, 79, 2554-8.

(8) Hinton, G.E., Plaut, D.C. and Shallice, T. (1993) Simulating brain damage. Scientific American, Oct 1993 58-65.

(9) Function Chart "Grafcet" for the description of Logic Control Systems Juin 1982, Norme Francaise Enregistree NF C03-190.

(10) International Standard IEC 1131 Part 3, First Edition, Programmable Controllers, Programming Languages 1993 March, International Electrotechnical Commission, Geneva.

(11) Winograd, T and Flores, F (1986) Understanding Computers and Cognition. Norwood NJ: Ablex.