Access to Relational Knowledge: a Comparison of Two Models

William H. Wilson (billw@cse.unsw.edu.au)
Nadine Marcus (nadinem@cse.unsw.edu.au)
School of Computer Science and Engineering, University of New South Wales, Sydney,
New South Wales, 2052, Australia

Graeme S. Halford (gsh@psy.uq.edu.au)
School of Psychology, University of Queensland, Brisbane, Queensland, 4072, Australia

Abstract

If a person knows that Fred ate a pizza, then they can answer the following questions: Who ate a pizza?, What did Fred eat?, What did Fred do to the pizza? and even Who ate what? This and related properties we are terming *accessibility properties* for the relational fact that Fred ate a pizza. Accessibility in this sense is a significant property of human cognitive performance. Among neural network models, those employing tensor product networks have this accessibility property. While feedforward networks trained by error backpropagation have been widely studied, we have found no attempt to use them to model accessibility using backpropagation trained networks. This paper discusses an architecture for a backprop net that promises to provide some degree of accessibility. However, while limited forms of accessibility are achievable, the nature of the representation and the nature of backprop learning both entail limitations that prevent full accessibility. Studies of the degradation of accessibility with different sets of training data lead us to a rough metric for learning complexity of such data sets.

Introduction

The purpose of this research is to determine whether a backpropagation net can be developed that processes propositions with the flexibility that is characteristic of certain classes of symbolic neural net models. This has arguably been difficult for backpropagation nets in the past. For example, the model of Rumelhart and Todd (1993) represents propositions such as "canary can fly". Given the input "canary, can" it produces the output "fly". However processing is restricted, so it cannot answer the question "what can fly?" ("canary").

There are, however, at least two types of symbolic nets that readily meet this requirement. One type of net model makes roles and fillers oscillate in synchrony (Hummel & Holyoak, 1997; Shastri & Ajjanagadde, 1993) while another is based on operations such as circular convolution (Plate, 2000) or tensor products (Halford, et al., 1994; 1998; Smolensky, 1990). These models appear to have greater flexibility than models based on backpropagation nets, in that they can be queried for any component of a proposition. We will refer to this property of tensor product nets as omni-directional access (cf. Halford, Wilson & Phillips, 1998). Omni-directional access is the ideal form of accessibility.

Another reason for investigating this lies in the work of Halford, Wilson, and Phillips (e.g. 1998) which seeks in part to define a hierarchy of cognitive processes or systems and to draw parallels between this hierarchy and a second hierarchy of types of artificial neural networks. Levels 0 and 1 of this second hierarchy are 2- and 3-layer feedforward nets, and levels 2-5 are tensor product nets of increasing rank. It thus becomes interesting to consider how well feedforward nets can emulate tensor product networks.

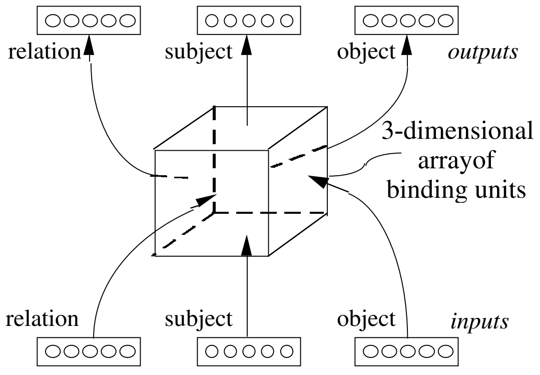

Figure 1 – Tensor product network of rank 3.

As tensor product networks are not as well known as feedforward networks, we shall describe them and their accessibility properties briefly here before proceeding. Tensor product networks are described in more detail, and from our point of view, in Halford et al. (1994). Briefly, a rank k tensor product network consists of a k-dimensional array of "binding units", together with k input/output vectors. For example, a rank 2 tensor product network is a matrix, plus 2 input/output vectors. To teach the network to remember a fact (that is, a k-tuple), the input/output vectors are set to be vectors representing the components of the k-tuple, and a computation is performed that alters the k-dimensional array. Subsequently that fact can be accessed in a variety of ways. It is common to interpret the first component of the *k*-tuple as a predicate symbol, and the remaining components as argument symbols - e.g. for rank 3, the components might be vectors representing the concepts *likes jane pizza* (Jane likes pizza) (see Figure 1). Once this fact has been taught to a rank 3 tensor product network, the following 7 queries can be formulated and answered by a computation involving the tensor product network.

1) Is *likes(jane,pizza)* true?
2) Who likes pizza? This we often write as *likes(X, pizza)*? The response depends on what else has been taught to the tensor product network. If the tensor product network also knows that *likes(fred,pizza)* and *likes(mary,pizza)* then the response will be the sum of the vectors representing Jane, Fred, and Mary - often written jane + fred + mary.
3) What does Jane like? - *likes(jane,X)*? Similar to 2).
4) What relationships hold between Jane and pizza? - *X(jane,pizza)*? Again, similar to 2).

These four are referred to as limited accessibility.

5) Who likes what? - *likes(X,Y)*? The response in this case would be a rank 2 tensor product network storing the pairs (X,Y) for which *likes(X,Y)* is known to the original rank 3 tensor product network. The tensor product network approach solves this by producing a rank 2 tensor product network, which stores the pairs (X,Y). (This output possibility, and corresponding ones for 6) and 7) below, are not shown in Figure 1).
6) Who does what to pizza? - *X(Y,pizza)*? Like 5).
7) Jane does what to what? - *X(jane,Y)*? Like 5).

The full set of 7 forms of access are referred to as full accessibility, or omni-directional access.

A rank 4 tensor product network would have 15 access modes, a rank 5 tensor product network would have 31 access modes, and so on. Provided that an orthonormal set of vectors is used for the set of vectors representing concepts, retrieval is perfect. Facts are learned by a tensor product network one at a time, and do not interfere with each other (given orthonormal representation vectors).

Tensor product networks using orthonormal sets of representation vectors exhibit what has been called full omni-directional access to the facts that have been taught, as noted above. Humans attempting similar tasks may find some types of access easier than others. For example, children who have recently learned sets of multiplication facts such as 9×7=63 are able to use this knowledge to perform division (9×X= 63, what is X?), but may find this more difficult than multiplication (9×7=X, what is X?). We use the term *accessibility* to refer to imperfect or partial versions of omni-directional access. It turns out that some of the nets discussed in this paper also exhibit accessibility rather than full omni-directional access.

Our specific aim in this paper is to experiment with a feedforward net design that appears to have potential to provide at least limited accessibility in a rank 3 situation. When we move to feedforward networks trained by error backpropagation, we hope to preserve the accessibility property that is characteristic of symbolic nets. The model resembles an auto-encoder but has restricted connectivity.

Architecture of the network

The particular backpropagation network we used to test for accessibility consisted of the following components: 15 input units, 15 hidden units and 15 output units. The 15 input units were used to represent 3 items or patterns, each made up of 5 elements. The hidden and output units also each consisted of three groups of 5 units, connected as shown in Figure 2. The input patterns represented relational instances of the form RELATION(SUBJECT, OBJECT). The target output contained the same information: namely, RELATION, SUBJECT and OBJECT.

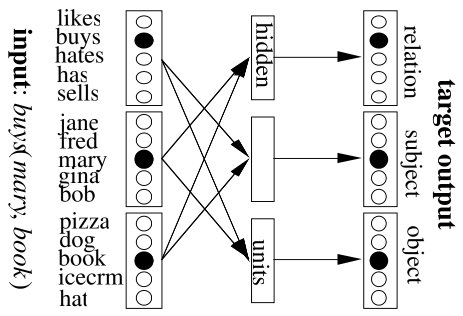

Figure 2 - Connections in our feedforward net architecture.

Notice that this network consists of three functions: one takes as inputs a relation name and a subject, and produces an object as output, the second takes relation name and object and produces subject as output, and the third takes subject and object and produces relation name as output. Thus, while it resembles a traditional auto-association net, note that regular auto-association nets allow connection paths between corresponding input and output neurons, typically allowing total interconnection between input and hidden layers, and between output and hidden layers. In essence, the network architecture can be unraveled into 3 distinct networks that share common inputs. Thus, any weight in the network is influenced by the output errors in only one of the 3 output sets (relation, subject, object). The net makes learning easier by constraining the learning algorithm to look for sets of weights that, for example, ignore predicate input when trying to infer predicate output from argument input. We also conducted some pilot studies with a fully connected network and the network's performance was inferior.

Notice that both the tensor product net architecture and the architecture we are studying here have three groups of input and three groups of output nodes.

Experimental design

The network was given three different sets of relational instances to learn. Each set was made up of five different relational instances. For example, given the relational instance *likes(jane,pizza)*, *likes* is the RELATION, *jane* is the SUBJECT and *pizza* is the OBJECT. The training sets varied in terms of the amount of overlap or the degree of interaction between the elements of each of the five relational instances. Training set 1 was set up to contain little or no overlap between the different relational instances. Training set 3 consisted of five relational instances with a large degree of interaction between the different instances. Training set 2 contained an intermediate degree of overlap.

It was hypothesized that relational instances with the least amount of interaction between the different instances would be the easiest to learn. This is because each instance does not have components that overlap with the other instances. These relations are one-to-one mappings. Accordingly, the network is most likely to achieve success in learning such a set of facts. In contrast, the set of relations with the highest level of overlap is expected to be the most difficult for the network to learn. These relations can be classified as many-to-many mappings, and so cannot be completely learned by a feedforward network. Accessibility may be easier to obtain if each relational instance can be represented in isolation, with little or no reference to the other relations. As the overlap and interaction between the relations and their elements increases, so the degree of accessibility that can be obtained is likely to decrease. This is because information from other related instances is more likely to interfere, when the system is presented with queries.

The software used to run the simulations described in this paper is Tlearn v1.01 (Plunkett & Elman, 1997). Other simulators were also used and similar results were obtained. The settings used included a learning rate of 0.1, momentum set to 0 (the default) and an initial weight range of -0.5 to 0.5.

Training set 1

In this simulation, the network was trained on five relations that have no overlap. Each relational instance consisted of a unique OBJECT, SUBJECT and RELATION. Within each relational instance, each field or argument was represented by a 1-out-of-5 localist encoding. The first five relational instances (and their associated patterns) that were given to the network to learn are shown in Table 1. Figure 3 (see training set 1) contains a graphic representation of the relational instances and their relationships (or in this particular case, their lack of relatedness).

The system was trained for 20 000 epochs. At around 4000 epochs the error curve smoothed close to zero. In other words, the difference between the target and the actual output values was negligible.

Table 1 — The instances and patterns in training set 1.

Relational instance	*Action*	*Subject*	*Object*
likes(jane, pizza)	10000	10000	00100
buys(fred, book)	01000	01000	00010
hates(mary, dog)	00100	00100	00001
has(gina, icecrm)	00010	00010	10000
sells(bob, hat)	00001	00001	01000

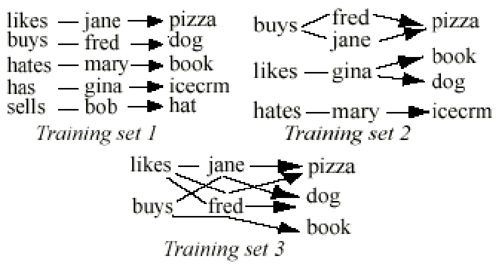

Figure 3 - Graphical representation of 3 training sets.

The system was then presented with a set of test patterns to assess the degree of accessibility that could be obtained (refer to Appendix 1, test pattern set 1). For example, to present the query *likes(jane,X)* the RELATION and SUBJECT input units were set to the patterns for *likes* and *jane* respectively, and the OBJECT input units were set to all zeroes, i.e. the '*X* ' is represented by '00000'. Then the OBJECT outputs were inspected. All of the outputs were checked to see if they matched the target values. If the correct number of output units that should be on is N, then a value of 3/(5N) or greater for an output unit is considered to be on, a value of 2/(5N) or less is considered to be off and any values in the region between 2/(5N) and 3/(5N) can be seen to be "partially on". The output thus falls into one of three categories: 1) Either the output is *correct* and all of the outputs units are correctly on or off (as defined above), or 2) The output is *incorrect* and at least one output unit that should be on is clearly off and vice-versa, or lastly, 3) The output is *uncertain* or partially correct, and output units that should be on are only "partially on". For example, if *likes(fred,pizza)* and *likes(fred,dog)* are facts, then if presented with the query *likes(fred,X)*, the answer would be pizza and dog, i.e. N=2. Therefore, the units representing both dog and pizza need to be on and a value of 0.3 (3/(5N)) or greater for both units is needed for the answer to be accepted as correct. Moreover, output units that need to be off should be less than 2/(5N) or in this case less than 0.2.

With training set 1, correct scores were obtained for all of the test queries. The system was able to handle all of the single and double query test patterns. Therefore, overall an excellent degree of accessibility was achieved with the first training set.

When there is little or no overlap between the elements of the relational instances, the system is able to learn and access elements of the relations with ease. This would correspond with human learning, where the less related information is to other information, the easier it is to understand and learn (Marcus, Cooper & Sweller, 1996; Sweller, 1994).

Training set 2

The next training set the system was given to learn had a higher degree of interaction between the relational instances and their elements than training set 1. In particular, both Fred and Jane like pizza and Jane buys both dogs and books. For the first two instances the same OBJECT is liked by two SUBJECTS and can be characterized as a many-to-one mapping, and for the last two instances the same SUBJECT buys two different OBJECTS, a one-to-many mapping. In contrast, *hates(mary,icecream)* is the only instance that does not overlap with the other four, and is a one-to-one mapping. The OBJECT and SUBJECT in this instance are unique and are not contained in any of the other instances. The five relational instances contained in training set 2 are shown in Table 2. The overlap or interrelations between the elements of the relational instances can be seen graphically in Figure 3 (see training set 2).

The system was trained for 20 000 epochs. At around 2000 epochs the error stabilized at around 0.6 in terms of Tlearn's error measure. The trained net transformed the training patterns as follows:

input		*output*
likes(jane, pizza)	→	likes(fred+jane, pizza)
likes(fred, pizza)	→	likes(fred+jane, pizza)
buys(gina, book)	→	buys(gina, book+dog)
buys(gina, dog)	→	buys(gina, book+dog)
hates(mary, icecrm)	→	hates(mary, icecrm).

It can be seen that the four (related) assertions have now been combined into two assertions. What has been learned is intelligible - *likes(gina,book+dog)* is easy to interpret as signifying that gina likes both books and dogs.

A set of test patterns (see Appendix 1, test pattern set 2) was then given to the system to assess performance on the accessibility property. All output units were then inspected to see if they matched the target units. Using the scoring criterion described above, output patterns were either considered to be 1) correct, 2) incorrect or 3) uncertain. All of the queries with a single unknown element were correctly answered by the system and some of the queries with two unknown elements were correct. None of the queries were considered incorrect, however, four queries obtained uncertain or partially correct scores. These queries were *likes(X,Y), buys(X,Y), X(gina,Y),* and *X(Y,pizza)*. It is interesting to note that all of the responses of questionable correctness need to access information that is contained in more than one relational instance, i.e. information from the many-to-one and one-to-many mappings. For instance, the answer to *likes(X,Y)* is that both Fred and Jane like pizza. This can be clearly expressed in the representation available, but the trained system does not do so. Thus although, accessibility is still relatively good, the net struggles with the queries that access information that has to be integrated from two relational instances.

Table 2 — The instances and patterns in training set 2.

Relational instance	*Action*	*Subject*	*Object*
likes (jane, pizza)	10000	10000	00100
likes (fred, pizza)	10000	01000	00100
hates(mary,icecrm)	00100	00100	10000
buys (gina, book)	01000	00010	00010
buys (gina, dog)	01000	00010	00001

We also tried training the network with the 3 patterns: *likes(fred+jane,pizza)*, *buys(gina,book+dog)*, and *hates(mary,icecream)*, that is, the 3 outputs the net just discussed (call it net 2A) produced in response to the training patterns. The network rapidly learned these patterns, not surprisingly. We tested this network (net 2B) on the queries shown in Appendix 1, test pattern set 2, and found that it had inferior accessibility performance compared with net 2A.

The greater number of uncertain or partially correct scores obtained during testing, for training set 2 (net 2A) reflects the fact that these five assertions may be considered harder to learn. These findings suggest that as the degree of overlap between the relational instances and their elements increases and as the amount of related information that needs to be considered at once increases, so the level of accessibility that the system can cope with, decreases. This corresponds with our understanding of difficulty associated with learning for people. The more interactivity there is between different learning elements, the harder information is to learn (Sweller & Chandler, 1994). The more difficult it is to learn information, the harder it is to transform and use that information. It thus appears, that as the information becomes more complex and so more difficult to learn, the backpropagation system struggles to achieve a reasonable level of accessibility. The next training set supports this hypothesis.

Training set 3

The last training set has the highest degree of interaction between the relational instances and their elements. The relational instance *likes(fred,pizza)* overlaps with two other instances. The SUBJECT *fred* performs the RELATION *likes* on both the OBJECTS *dog* and *pizza*, a one-to-many mapping. Also, both SUBJECTS *fred* and *jane* perform the RELATION *likes* on the same OBJECT *pizza*, a many-to-one mapping. The five relational instances contained in training set 3 are shown in Table 3. Figure 3 (see training set 3) contains a graphic representation of the relational instances and their interrelatedness.

Table 3 — The instances and patterns in training set 3.

Relational instance	*Action*	*Subject*	*Object*
likes (jane, pizza)	10000	10000	00100
likes (fred, pizza)	10000	01000	00100
likes (fred, dog)	10000	01000	00001
buys (fred, book)	01000	01000	00010
buys (jane, dog)	01000	10000	00001

The system was trained for 20 000 epochs. At around 3000 epochs the error stabilized at around 0.7 in terms of Tlearn's error measure. It should be noted that *buys(fred,book)* and *buys(jane,dog)* each have at most one attribute in common with the other instances. The trained net transformed the three more overlapping instances as follows:

input		*output*
likes(jane, pizza)	$\rightarrow$	likes(fred+jane, pizza)
likes(fred, pizza)	$\rightarrow$	likes(fred+jane, pizza+dog)
likes(fred, dog)	$\rightarrow$	likes(fred, pizza+dog).

Notice that from this output, there is no way to interpret these instances without inferring that Jane also likes dogs. The whole is not truly equivalent to the sum of the parts, which in this case are the three (and not four) given relational instances. Thus the pattern *likes(fred+jane,pizza+dog)* even if it were valid, would be unintelligible (in contrast to *likes(gina,book+dog)* in training set 2).

The test patterns shown in Appendix 1 (test pattern set 3) were used to test the trained net for accessibility properties. As before, output units were inspected to see if they matched the target units. Using the scoring criterion described above output patterns were either considered to be correct, incorrect or uncertain. All of the queries with a single unknown element were correctly answered by the system. However, only two of the queries with two unknown elements were correct. The two correct queries were *X(Y,dog)* and *X(Y,book).* The rest of the queries with two unknown elements were incorrect. These queries all access information from more than one relational instance. For example, the query *X(jane,Y)* should have a response of *likes+buys, pizza+dog*. However, the system's response to this query is only *buys, pizza+dog*. As with all the other incorrect queries, some of the relevant information has been lost. It appears that as the information becomes more and more overlapping, the network finds it harder and harder to handle queries that access related elements of information. This type of network appears to be more suited to dealing with one-to-one relations, rather than many-to-many mappings.

Training set 4

A fourth training set was given to the system to learn. It consisted of the relational instances *likes(jane,pizza), likes(fred,pizza), likes(fred,dog), buys(fred,dog),* and *buys(jane,book)*. The amount of overlap between these instances, and the test results, fall somewhere between training sets 2 and 3. All the single unknown element queries were answered correctly. Three of the two unknown element queries were answered correctly, two were uncertain and two were incorrect.

Conclusion

As the degree of overlap between the arguments and predicates of the relational instances in the training set increases, the degree of accessibility provided by the nets simulated decreases. It is well-known that when trained on data that corresponds to a one-to-many mapping, the activations of the output units corresponding to the "many" will be reduced in comparison to a one-to-one mapping. To us, the interesting thing is the effect of argument and predicate overlap on accessibility, and the fact that beyond some critical level of overlap, the trained net starts to produce "generalizations" which, seen from the relational-instance point of view, mean that the net has learnt false propositions e.g. *likes(jane,dog)*.

By way of contrast, tensor product networks (Halford et al., 1998) provide full accessibility for arbitrary sets of relational instances, and do not lose critical information when tested.

Backpropagation nets can handle propositional information that is in the form of distinct functions. For example, the model of Rumelhart and Todd (1993) handles propositions such as "canary can fly" in the sense that, given an input "canary can" it produces the output "fly". However, it was not tested for the accessibility property. Our backpropagation net was tested for accessibility, but succeeded in only a limited sense. It could only handle queries to data sets that are relatively simple, in terms of the overlap and relatedness of information. As the relational instances in the data set become more and more related, so accessibility deteriorates. Consequently the net could not model propositional knowledge adequately. In contrast, a tensor product net can process more complex data sets and still have full access to all the elements of the relational instances.

In a sense, it is not surprising that a backprop-trained net does not do as well at this task - backprop tends to do well at perceptual tasks where generalization of an interpolative type is useful, whereas the data used in this is discrete. Since their introduction, backprop nets and variants have been used in cognitive modeling tasks including those concerned with discrete relational knowledge (Hinton, 1986; Rumelhart & Todd, 1993). This paper has attempted to explore the boundaries of applicability of such models.

What has come out in the wash is evidence from the model's performance of a new dimension of task difficulty. This dimension measures component overlap in a set of facts to be learned. This type of difficulty seems to correlate with model performance at the boundary between rank 1 and rank 2 tasks (in the sense of Halford et al., 1998).

It is clear that humans do have accessibility with respect to their relational knowledge. What might be interesting to investigate is whether they have greater difficulty learning sets of facts like those in training set 3 than those in training set 1, and whether accessibility also takes longer to develop (see Sweller & Chandler, 1994 for a discussion of element interactivity and its effects on learning).

Acknowledgments

This work was supported by a grant from the Australian Research Council. We wish to acknowledge helpful discussions with Steve Phillips, and helpful comments made by a reviewer of a previous version of the paper.

References

Halford, G. S., Wilson, W. H., Guo, J., Gayler, R. W., Wiles, J., Steward, J. E. M. (1994). Connectionist implications for processing capacity limitations in analogies. In K. J. Holyoak & J. Barnden (Eds.), *Advances in connectionist and neural computation theory, vol. 2: Analogical connections.* Norwood, NJ: Ablex.

Halford, G. S., Wilson, W. H., & Phillips, S. (1998). Processing capacity defined by relational complexity: Implications for comparative, developmental, and cognitive psychology. *Brain and Behavioural Sciences,* 21, 803-864.

Hinton, G. E. (1986). Learning distributed representations of concepts. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society, 1-12,* Hillsdale, NJ: Lawrence Erlbaum Associates.

Hummel, J. E., & Holyoak, K .J. (1997). Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review,* 104, 427-466.

Marcus, N., Cooper, M., & Sweller, J. (1996). Understanding Instructions. *Journal of Educational Psychology,* 88(1), 49-63.

Plunkett, K., & Elman, J. L. (1997). *Exercises in Rethinking Innateness: A Handbook for Connectionist Simulations.* Cambridge, Mass: MIT Press.

Plate, T. A. (2000). Analogy retrieval and processing with distributed vector representations. *Expert Systems: The International Journal of Knowledge Engineering & Neural Networks,* 17(1), 29-40.

Rumelhart, D. E., & Todd, P. M. (1993). Learning and connectionist representations. In D.E. Meyer & S. Korhnblum (Eds*), Attention and Performance XIV* (figure 1.9 p15, top paragraph p16). Cambridge, Mass: MIT Press.

Shastri, L. & Ajjanagadde, V. (1993). From simple associations to systematic reasoning: A connectionist representation of rules, variables, and dynamic bindings using temporal synchrony. *Behavioural and Brain Sciences,* 16(3), 417-494.

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

Sweller, J. (1994). Cognitive load theory, learning difficulty and instructional design. *Learning and Instruction,* 4, 295-312.

Sweller, J., & Chandler, P. (1994). Why some material is difficult to learn. *Cognition and Instruction,* 12, 185-233.

Appendix 1

Test pattern set 1
likes(jane,X), buys(fred,X), hates(mary,X), has(gina,X), sells(bob,X), likes(X,pizza), buys(X,fred), hates(X,dog), has(X,icecrm), sells(X,hat), X(jane,pizza), X(fred,book), X(mary, dog), X(gina, icecrm), X(bob,hat), likes(X,Y), buys(X,Y), hates(X,Y), has(X,Y), sells(X,Y), X(jane,Y), X(fred,Y), X(mary,Y), X(gina,Y), X(bob,Y), X(Y,pizza), X(Y,book), X(Y,dog), X(Y,icecrm), X(Y,hat).

Test pattern set 2
likes(jane,X), likes(fred,X), buys(gina,X), hates(mary,X), likes(X,pizza), buys(X,book), buys(X,dog), hates(X,icecrm), X(jane,pizza), X(fred,pizza), X(gina,book), X(gina,dog), X(mary,icecrm), likes(X,Y), buys(X,Y), hates(X,Y), X(jane,Y), X(fred,Y), X(gina,Y), X(mary,Y), X(Y, pizza), X(Y,book), X(Y,dog), X(Y,icecrm).

Test pattern set 3
likes(jane,X), likes(fred,X), buys(fred,X), buys(jane,X), likes(X,pizza), likes(X,dog), buys(X,book), buys(X,dog), X(jane,pizza), X(fred,pizza), X(fred,dog), X(fred,book), X(jane, dog), likes(X,Y), buys(X,Y), X(jane,Y), X(fred,Y), X(Y,pizza), X(Y,dog), X(Y,book).