Computer Science Department Technical Report
Artificial Intelligence Laboratory
University of California
Los Angeles, CA 90024-1596

ENCODING INPUT/OUTPUT REPRESENTATIONS IN CONNECTIONIST COGNITIVE SYSTEMS

Risto Miikkulainen
Michael G. Dyer

December 1988
CSD-880103

Encoding Input/Output Representations in Connectionist Cognitive Systems

Risto Miikkulainen
Michael G. Dyer

December 1988

Technical Report UCLA-AI-88-16

Encoding Input/Output Representations in Connectionist Cognitive Systems [*][†]

Risto Miikkulainen and Michael G. Dyer
Artificial Intelligence Laboratory
Computer Science Department
University of California, Los Angeles, CA 90024
risto@cs.ucla.edu, dyer@cs.ucla.edu

Abstract

Representing input/output distributively provides several desirable properties, e.g. associations and generalizations directly arise from the representations and the system is robust against noise and damage. One way to form distributed representations is to classify each input item along a number of predetermined semantic microfeatures and to represent each microfeature locally. How to determine the appropriate microfeatures remains a problem. Developing representations in the hidden layers of a backpropagation network avoids this issue, but the representations are internal and local to that layer. In the FGREP approach the backpropagation error signal is used to develop representations in an external network (a lexicon). Backpropagation in this approach operates in a reactive training environment, i.e. the required input/output mappings change as the I/O representations change. These representations evolve to improve the system's performance in the processing task and end up reflecting the properties of the input items which are most crucial to the task, facilitating excellent association and generalization. The microfeatures of the resulting representation in general are not identifiable. Different aspects of an input item are distributed over the whole set of units in a holographic fashion, making the system particularly robust against damage. Each representation also carries expectations about its possible contexts.

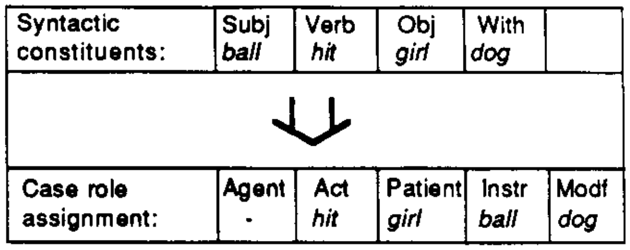

Figure 1: Case role assignment of **The ball hit the girl with the dog**.

1 Introduction

Sentence case role assignment is an example of a cognitive task which is well suited for modelling with connectionist systems. The syntactic structure of the sentence is given and consists of e.g. the subject, verb, object and a with-clause. The task is to decide which constituents play the roles of agent, patient, instrument and patient modifier in the act (figure 1). This requires forming a shallow semantic interpretation of the sentence.

For example, in **The ball hit the girl with the dog**, the subject **ball** is the instrument of the **hit**-act, the object **girl** is the patient, the with-clause, **dog**, is a modifier of the patient, and the agent of the act is unknown. Role assignment is context dependent: in **The ball moved** the same subject **ball** is taken to be the patient. Assignment also depends on the semantic properties of the concept. In **The man ate the pasta with cheese** the with-clause modifies the patient but in **The man ate the pasta with a fork** the with-clause is the instrument. In yet other cases the assignment must remain ambiguous. In **The boy hit the girl with the ball** there is no way of telling whether **ball** is an instrument of **hit** or a modifier of **girl**.

Case role assignment requires taking into account all the positional, contextual and semantic constraints simultaneously, which is what the connectionist systems are particularly good at. An important issue is how the input and output to such systems should be encoded.

In the distributed approach, the input/output concepts are represented as different patterns of activ-

[*]To appear in Touretzky, Hinton and Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, CMU, June 17-26 1988, Morgan Kaufmann Publishers, Inc.

[†]This research was supported in part by an ITA Foundation grant and the JTF program of the DoD, monitored by JPL. The first author was also supported in part by grants from the Academy of Finland, the Finnish Cultural Foundation, The Jenny and Antti Wihuri Foundation and The Thanks to Scandinavia Foundation. The simulations were carried out on equipment donated to UCLA by Hewlett Packard.

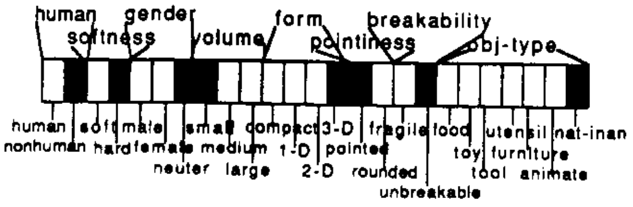

Figure 2: Semantic microfeature encoding of the word **rock** (from [McClelland and Kawamoto, 1986]).

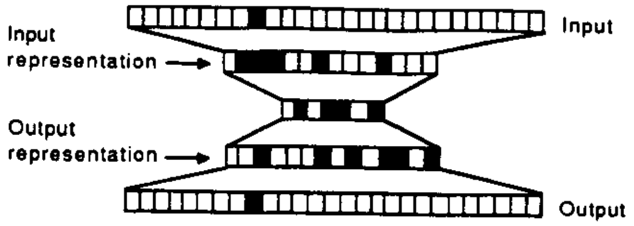

Figure 3: Developing internal representations in hidden layers.

ity over the same set of units. Desirable properties achieved are: (1) it is possible to associate similar concepts and generalize properties by sharing the same activity subpatterns, and (2) the system is robust against noise and damage [Hinton *et al.*, 1986].

One approach for forming distributed representation patterns is **semantic microfeature encoding**, used e.g. by McClelland and Kawamoto in the case role assignment task [McClelland and Kawamoto, 1986] (see also [Hinton, 1981]). Each concept is classified along a predetermined set of dimensions such as human-nonhuman, soft-hard, male-female etc. Each microfeature is assigned a processing unit (or a group of units, e.g. one for each value), and the classification becomes a pattern of activity over the assembly of units (figure 2).

This kind of representation is meaningful by itself. It is possible to extract plenty of information just by looking at the representation, without having to have a trained network to interpret it. Several different systems can directly use the same representation patterns and communicate using them.

On the other hand, the patterns must be pre-encoded and they remain fixed. Performance cannot be optimized by adapting the representations to actual task and data. Because all concepts must be classified along the same dimensions, the number of dimensions becomes very large, and many of them are irrelevant to the particular concept (e.g. gender of **rock**). Deciding what dimensions are advantageous to use is a hard problem. There is also a serious epistemological question of *whether the process of deciding what dimensions to use is justifiable or not.* Hand coded representations always have a degree of ad hocness and bias in them. In some cases it is possible to make the task trivial by a clever encoding of the input representations.

Developing internal representations in hidden layers of a backpropagation network avoids these problems (see e.g. the family tree example in [Hinton, 1986]). A network of this type usually consists of input, output and three hidden layers (figure 3). The input and output layers are localist, i.e. exactly one unit is dedicated to each concept. The hidden layers next to the input and output layers contain considerably fewer units, which forces these layers to form compressed distributed activity patterns for the input/output concepts. Developing these patterns occurs as an essential part of learning the processing task, and they end up reflecting the regularities of the task [Hinton, 1986].

This approach does not address the issue of encoding input/output representations. The system does not deal with the representations per se; they develop only as a side effect of modifying the weights to improve the task performance. The patterns are not available outside the system, and they are not used in communication with the system. Moreover, since both penultimate layers develop their activity patterns independently, each concept has two different representations: one as an input and another one as an output concept. These activity patterns are *local, internal processing aids* more than input/output representations which can be used in a larger environment.

This paper describes a third approach, **FGREP** [Miikkulainen and Dyer, 1988], where representations are also developed automatically while the network is learning the processing task, making use of the backpropagation error signal. However, the representations are global input/output to the network and they are stored in an external network (a lexicon), which guarantees unambiguity and makes communication using these representations possible.

2 FGREP: Forming global representations with extended backpropagation

2.1 System architecture

The FGREP system is based on a basic three-layer backward error propagation network (figure 4). The network learns the processing task by adapting the connection weights according to the standard backpropagation equations [Rumelhart *et al.*, 1986, pages 327-329]. At the same time, representations for the input data are developed at the input layer according to the error signal extended to the input layer. Input and output layers are divided into assemblies and several concepts represented and modified simultaneously.

The representations are stored in an external lexicon network. A routing network forms each input pattern

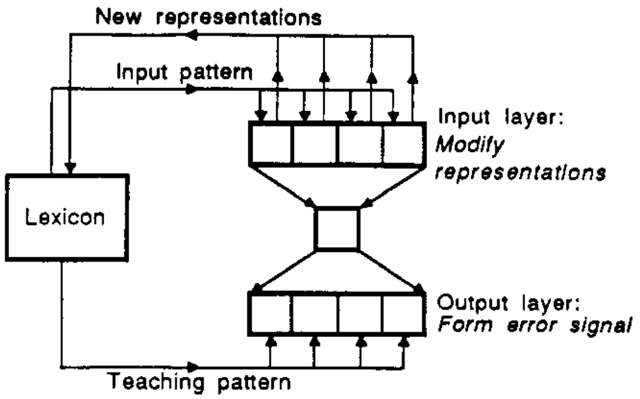

Figure 4: FGREP architecture.

and the corresponding teaching pattern by concatenating the lexicon entries of the input and teaching concepts. Thus *the same representation for each concept is used in different parts of the backpropagation network*, both in the input and in the output.

The process begins with a *random lexicon containing no pre-encoded information.* During the course of learning, the representations adapt to reflect the implicit regularities of the task. It turns out that single units in the resulting representation do not necessarily have a clear interpretation. The representation does not implement a classification of the concept along identifiable features. In the most general case, the representations are simply profiles of continuous activity values over a set of processing units.

The approach is motivated by the eventual goal of using a simple network as a building block in more complex systems. An FGREP network could be used as a module in a larger system, where the different subsystems communicate using the lexicon.

2.2 Extending backpropagation to the representations

Standard backpropagation produces an error signal for each hidden layer unit. By propagating the signal one layer further to the input layer, the representations can be changed *as if they were weights on connections coming in to the input layer.*

In a sense, the representation of a concept serves as input activation to the input layer. The activation of an input unit is identical to the corresponding component in the representation. In this analogy, the activation function of an input unit is the identity function and its derivative is one. The error signal can now be computed for each input unit as a simple case of the general error signal equation [Rumelhart *et al.*, 1986, Eq.14]:

$$\delta_{1i} = \sum_j \delta_{2j} w_{1ij}. \qquad (1)$$

where δ_{xy} stands for the error signal for unit y in layer x, and w_{1ij} is the weight between unit i in the input layer and unit j in the first hidden layer.

Imagine a localist 0:th layer before the input layer, with one unit dedicated to each input concept in each assembly. In this layer at most one unit per assembly is active with value 1 at any time (the one corresponding to the current input), the rest of the units have zero activity. Each localist unit is connected to all units in the input assembly with weights equal to the input representations. Extending back propagation weight change to these weights can be interpreted as *changing the representations themselves*:

$$\Delta r_{ci} = \eta \delta_{1i} r_{ci}, \qquad (2)$$

where r_{ci} is the representation component i of concept c, δ_{1i} is the error signal of the corresponding input layer unit and η is the learning rate. Using this analogy, *representation learning is implemented as an extension of the back propagation algorithm.* While the weight values are unlimited, the representation values must be limited between the maximum and minimum activation values of the units. The new value for the representation component i of concept c is obtained as

$$r_{ci}(t+1) = max[o_l, min[o_u, r_{ci}(t) + \Delta r_{ci}]], \qquad (3)$$

where o_l is the lower limit and o_u is the upper limit for unit activation.

Note that the backpropagation "sees" the representations simply as an extra layer of weights. By separating the representations from the network and treating them as global, external objects (instead of local, internal weights) we can develop a single, concrete representation for each concept. Since the representations adapt according to the error signal, there is reason to believe that *the resulting representations effectively code properties of the input elements which are most crucial to the task.*

2.3 Reactive training environment

The process differs from ordinary backpropagation in that both the input and the teaching patterns are changing. An input pattern is formed by drawing the current representations of the sentence constituents from the lexicon and loading them into the input assemblies (figure 4). The activity is propagated through the network to the output layer, where the error signal is formed by comparing the output pattern to the teaching pattern, which is also formed by drawing the current representations from the lexicon. The error signal is propagated back to the input layer, changing weights and the input concept representations along the way. Next time the *same input sentence* occurs, the output will be closer to the *same teaching pattern.* The modified representations are now put back to the lexicon, replacing the old ones and thereby *changing the next teaching pattern for the same input.* The shape of the error surface is changing at each step, i.e. backpropagation is shooting at a moving target in a reactive training environment.

It turns out that as long as the changes made in the process are small, the process converges nevertheless. The learning time does not seem to be significantly longer than in the ordinary case. The changes in the error surface are a form of noise (a noisy error signal), which backpropagation in general tolerates very well.

3 Simulations in the case role assignment task

In [McClelland and Kawamoto, 1986] the authors describe a system which learns to assign case roles to sentence constituents. The same task with the same data was used to test FGREP, because it provides a convenient comparison to a system using fixed microfeature encoding. The task was restricted to a small repertoire of sentences studied in the original experiment. These sentences consist of a subject, verb, object and a with-clause. The possible case roles are agent, act, patient, instrument, and modifier-of-patient. The sentence generators are depicted in table 1 and the noun categories in table 2. The generators produce 1553 different sentences, of which 1515 was used for training and 38 was reserved for testing generalization.

The sentence frames and the noun categories are not visible to the system: they are only manifest in the combinations of words that occur in the input sentences. To do the case role assignment properly *the system had to figure out the underlying relations and code them into the representations.*

In this particular task, the teaching input is made up from the input sentence constituents (figure 5). This is by no means necessary for learning the representations. The required output of the network could be anything and the FGREP method would work the same. A "pigeonholing" task is actually harder than a general task because of the reactive training effect.

The training consisted of cycling through the training set 50 times in random order. Initially the components of the representations were uniformly distributed in the interval [0,1] and the connection weights of the network in the interval [-1,1]. The behaviour of the system was fairly insensitive to the learning parameters and system configuration parameters. Several different values of the learning rate η within the range 0.01 - 5.0 were tried and also increasing it in six steps from 0.01 to 0.5 (see [Plaut *et al.*, 1986]). All these cases lead to essentially identical results. The number of units in the representation and the number of hidden units were not crucial either: values as low as 5 and as high as 100 were tested. If more hidden units are used, the task performance, generalization capability and damage resistance improve slightly and the learning in general is faster. Decreasing the number of hidden units on the other hand lays more pressure on the representations, and they become more descriptive. In general, the best results are obtained when the number of hidden units is somewhat less than half the number of units in the input layer. It also turns out that the best generalization and the most descriptive representations are obtained at around the 20th epoch: overtraining tends to make the network reflect the idiosyncronies of the training data, as has been reported e.g. in [Tesauro and Sejnowski, 1988]. Figure 5 shows a snapshot of a real-time display of the simulation running on an HP 9000/350 workstation.

4 Results

4.1 Representations

Figure 6 shows the representations developed by the system, organized according to the noun categories. Starting from random representations, the similarity of the nouns belonging to the same category increases until the changes begin to cancel out. This happens usually within the first 20 epochs. During the remaining epochs, the representations are fairly stable while the task performance still improves. With different initial representations, the final set of representations in general is different. The overall characteristics of the representations and the performance of the system is approximately the same in all cases.

Some concepts belong exactly to the same categories and consequently occur exactly in the same contexts. They are indistinguishable in the data and their representations become identical. [**man, woman, boy, girl**] forms one such group, [**fork, spoon**], [**wolf, lion**], [**plate, window**], [**ball, hatchet, hammer**], [**paperwt, rock**] and [**cheese, pasta, carrot**] others. If there is at least one difference in usage of two nouns, their representations become different. The discriminating input modifies one of the representations while the other one remains the same. Since each noun belongs to several categories its representation can be seen as evolving from the competition between the categories. This is clearest on the part of the ambiguous nouns **chicken** and **bat**, which on the other hand are both **animals**, but **chicken** is also **food** and **bat** is a **hitter**. The representation is a combination of both, weighted by the number of occurrences of each meaning. On the other hand, the fact that there is a common element in two categories tends to make all representations of the two categories more similar. Thus the properties of one concept are generalized, to a degree, to the whole class.

Note that the categorization of a concept in figure 6 is formed outside the system and is independent of the task, other categories and other concepts. The system itself is not attempting categorization, it is forming the most efficient representation of each concept for a particular task. Interestingly, if one runs a merge cluster-

Gn	Sentence Frame	Correct case roles
1	The human ate.	agent
2	The human ate the food.	agent-patient
3	The human ate the food with the food.	agent-patient-modif
4	The human ate the food with the utensil.	agent-patient-instr
5	The animal ate.	agent
6	The predator ate the prey.	agent-patient
7	The human broke the fragileobj.	agent-patient
8	The human broke the fragileobj with breaker	agent-patient-instr
9	The breaker broke the fragileobj.	instr-patient
10	The animal broke the fragileobj.	agent-patient
11	The fragileobj broke.	patient
12	The human hit the thing.	agent-patient
13	The human hit the human with the possession	agent-patient-modif
14	The human hit the thing with a hitter.	agent-patient-instr
15	The hitter hit the thing.	instr-patient
16	The human moved.	agent-patient
17	The human moved the object.	agent-patient
18	The animal moved.	agent-patient
19	The object moved.	patient

Table 1: Sentence generators

Category	Nouns
human	man woman boy girl
animal	bat chicken dog sheep wolf lion
predator	wolf lion
prey	chicken sheep
food	chicken cheese pasta carrot
utensil	fork spoon
fragileobj	plate window vase
hitter	bat ball hatchet hammer vase paperwt rock
breaker	bat ball hatchet hammer paperwt rock
possession	bat ball hatchet hammer vase dog doll
object	bat ball hatchet hammer paperwt rock vase plate window fork spoon pasta cheese chicken carrot desk doll curtain
thing	human animal object

Table 2: Noun categories

The generators are presented as sentence frames, with one to three noun slots. Each slot can be filled with any of the nouns in the specified category, and each slot has a predetermined case role. For instance, **The human ate the food** generates 4 × 4 different sentences, all with the case role assignment **human = agent, food = patient**.

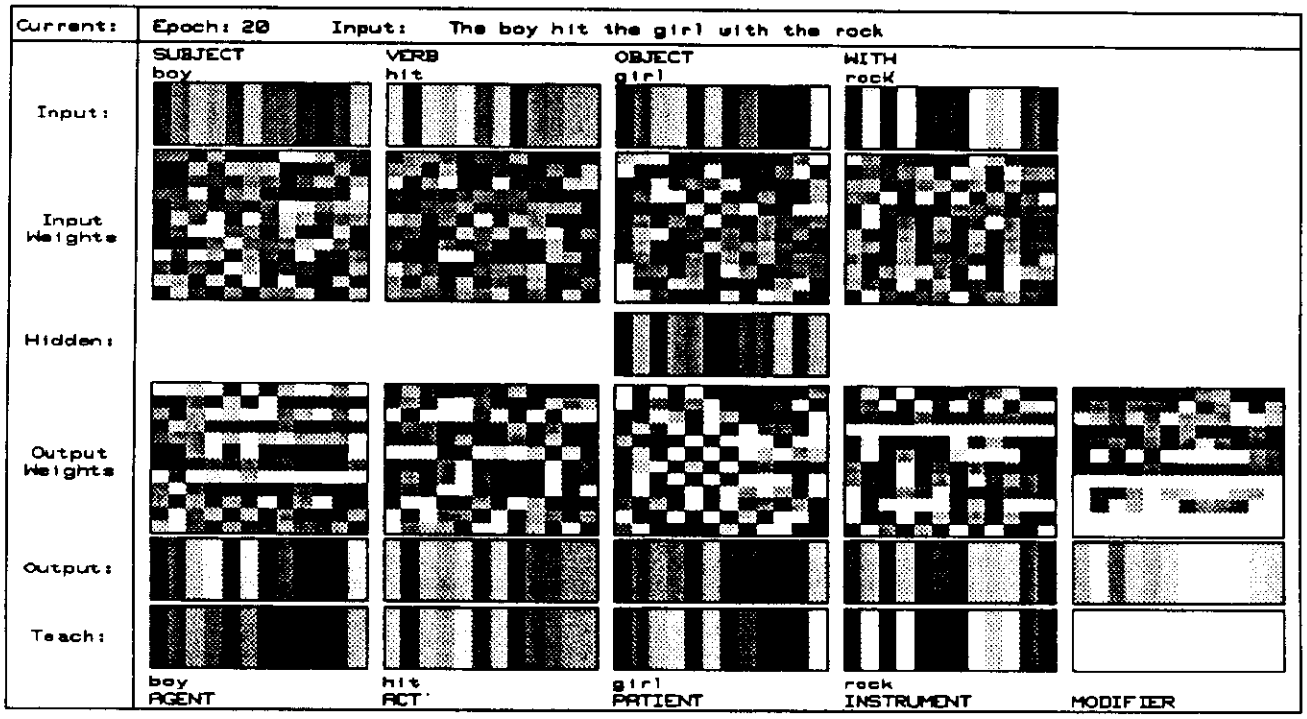

Figure 5: Snapshot of the simulation.

The header line displays the current input sentence. The input and output layers of the network are divided into assemblies, each of which holds one concept representation at a time. Each unit in an input assembly is set to the activity value determined by the corresponding component in the lexicon entry. The weights on the connections are displayed as square matrices between the layers. After the network has successfully learned the task, each output assembly produces an activity pattern which is identical to the lexicon representation of the concept that fills that role. The correct role assignment is shown at the bottom row of the display. This pattern forms the teaching input to the network. Gray scale values from white to black are used in the figure to code the unit activities in the range [0, 1] and the weight values within the interval [-1, 1].

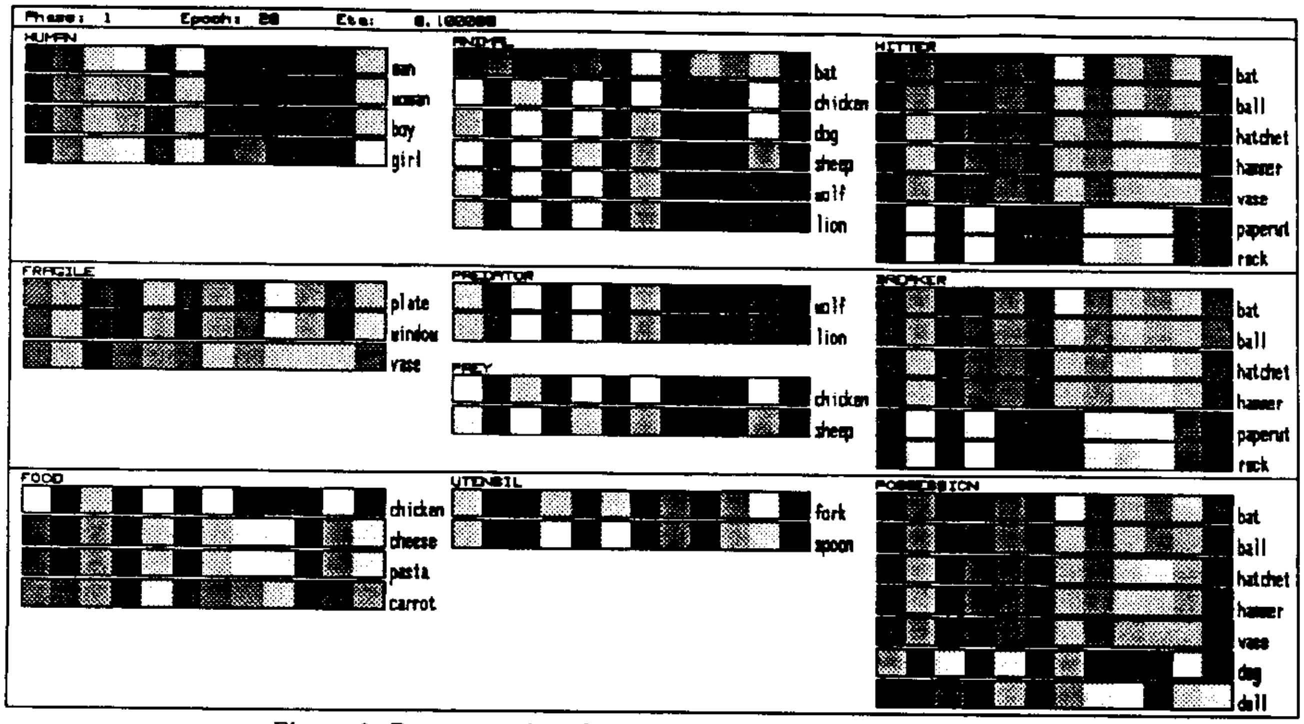

Figure 6: Representations have converged to reflect the categories.

ing algorithm [Kohonen, 1982] on the representations, the optimal clusters turn out to be quite similar to the noun categories (figure 7).

To look at the spatial relations of the representations, we first have to map the 12-dimensional representation vectors into a 2-dimensional space. One way to do this is Kohonen's self-organizing feature mapping [Kohonen, 1984]. This method is known to map clusters in the input space to clusters in the output space. The map is topological, i.e. the distances in the map are not comparable (more dense regions are magnified), but the topological relations of the input space are preserved.

The feature map shows the same clusters that were used in generating the input (figure 8). Note that the ambiguous nouns **chicken** and **bat** are mapped between their two possible categories **animal** and **food** and **animal** and **hitter**.

Inspection of the representations in figure 6 suggests that a single unit does not play a crucial role in the classification of concepts. The fact that a concept belongs to a certain category is indicated by the activity profile as a whole, instead of particular units being on or off. The representations are also extremely *holographic*. The whole categorization is clearly visible even in the values of a single unit (figure 9), and even more so in the space spanned by two units (figure 10).

It is very hard to point out what features the in-

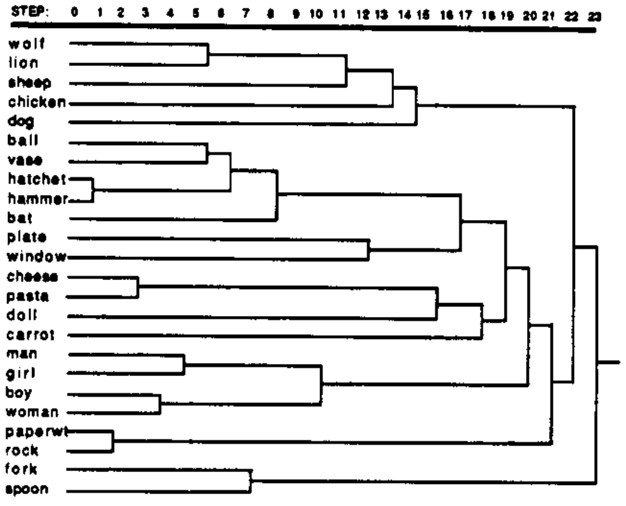

Figure 7: Merge clustering the representations.
At each time step, the clusters with the shortest euclidian single linkage distance are merged.

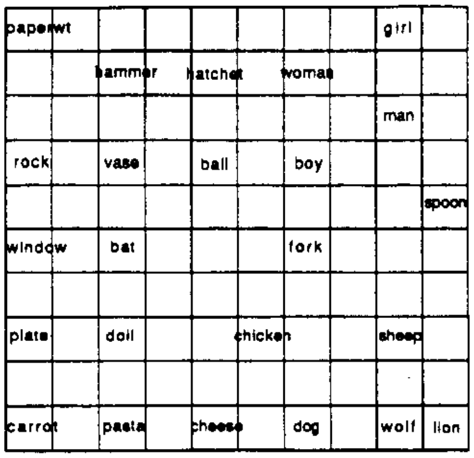

Figure 8: 2-D Kohonen-map of the representations. Labels indicate the maximally responding unit in the 10 × 10 feature map network for each representation vector.

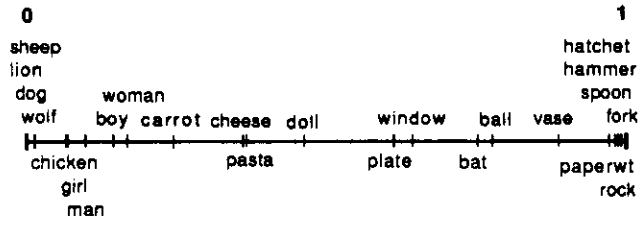

Figure 9: Categorization by unit 2.

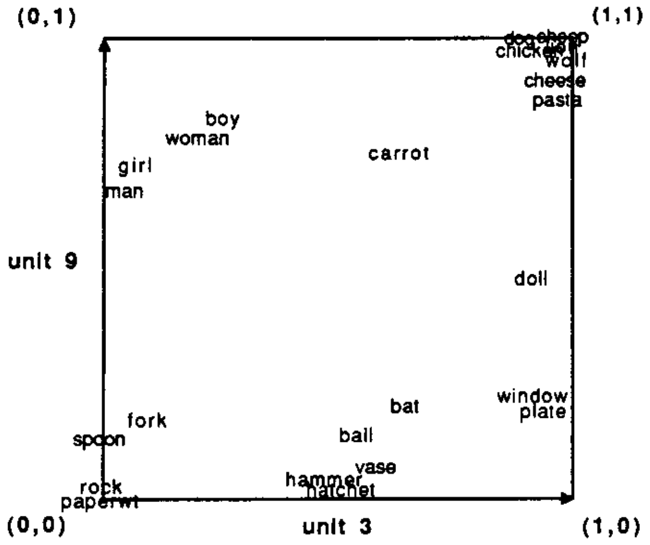

Figure 10: Categorization by units 3 and 9.

dividual units are actually coding. In [Hinton, 1986] Hinton was able to give interpretation for some of the units in the hidden layer representation, although he points out that the system develops its own microfeatures, which may or may not correspond to ones that humans would use to characterize data. Our results suggest that *the microfeatures in the resulting representation in general are not identifiable.*

4.2 Performance with the learned representations

The performance in the role assignment task after 50 epochs was tested with two test sets: the first one consisted of two randomly chosen sentences from each generator which had been used in the training, and the second one consisted of the test sentences which the network had not seen before. The robustness of the representations was tested by removing the last $n\%$ of the units from each input assembly. Tables 3 and 4 present results for each sentence.

The system learned the correct assignment of most sentences. Difficulties arise in ambiguous cases, where the system develops an intermediate output between the two possible interpretations, indicating a degree of confidence in the choices. One such case is presented in the snapshot of figure 5. **Rock** can be either the instrument of **hit** or a possession of **girl**. In most similar occasions it is the instrument, and the network develops a much stronger representation in the instrument assembly than in the modifier assembly. The system also performs poorly with the **animal** meaning of **bat**. Because a vast majority of the occurrences of **bat** are **hitters**, its pattern becomes more representative of the **hitters** than **animals**.

Strikingly, there seems to be very little difference in performance between the familiar and unfamiliar test sentences. *The generalization capabilities of the system are excellent.* This is a result of the system's tendency to develop similar representations to similarly behaving concepts. In a precoded, fixed microfeature -based system, such as McClelland and Kawamoto's, even though two concepts are equivalent in the input set, their representations remain different. With the FGREP approach, the representations become more similar, which means that generalization is necessarily stronger. *Damage resistance is also very good.* Even with half the input units removed, the system gets 65% of the output within 15% of the correct value. This is partly due to the general robustness of hidden-layer networks but also partly due to the fact that the representation is not coded into specific microfeature-units, but is distributed over all units in a holographic fashion.

Quantitative comparison of performance to McClelland and Kawamoto's system is hard because of the different architectures and goals. Percentage of correct units is not a very good performance measure

GEN	INPUT			DAMAGE%:	0.0	25.0	50.0
1	man	ate			81.7	66.7	65.0
1	girl	ate			81.7	66.7	65.0
2	woman	ate	cheese		96.7	70.0	60.0
2	woman	ate	pasta		98.3	73.3	60.0
3	woman	ate	chicken	pasta	95.0	60.0	53.3
3	man	ate	pasta	chicken	85.0	70.0	48.3
4	girl	ate	pasta	spoon	93.3	71.7	65.0
4	boy	ate	chicken	fork	93.3	75.0	53.3
5	dog	ate			68.3	60.0	60.0
5	sheep	ate			73.3	63.3	58.3
6	lion	ate	chicken		81.7	61.7	55.0
6	lion	ate	sheep		83.3	65.0	58.3
7	woman	broke	window		90.0	71.7	58.3
7	boy	broke	plate		90.0	78.3	56.7
8	man	broke	window	bat	91.7	66.7	68.3
8	boy	broke	plate	hatchet	88.3	68.3	65.0
9	paper	broke	vase		88.3	83.3	78.3
9	rock	broke	window		88.3	78.3	63.3
?10	bat	broke	window		71.7	66.7	51.7
10	wolf	broke	vase		78.3	65.0	55.0
11	vase	broke			86.7	80.0	70.0
11	window	broke			80.0	75.0	75.0
12	man	hit	pasta		100.0	88.3	70.0
12	girl	hit	boy		98.3	90.0	70.0
?13	man	hit	girl	hatchet	83.3	70.0	51.7
13	woman	hit	man	doll	90.0	76.7	56.7
14	woman	hit	bat	hammer	100.0	85.0	71.7
?14	girl	hit	woman	ball	80.0	80.0	70.0
15	hatchet	hit	pasta		98.3	90.0	86.7
15	hammer	hit	vase		100.0	95.0	86.7
16	man	moved			88.3	73.3	75.0
16	woman	moved			85.0	73.3	71.7
17	woman	moved	plate		98.3	73.3	70.0
17	girl	moved	pasta		98.3	83.3	70.0
?18	chicken	moved			71.7	61.7	71.7
18	lion	moved			75.0	60.0	70.0
19	doll	moved			90.0	80.0	71.7
19	desk	moved			95.0	81.7	80.0
AVERAGE%:					87.8	73.6	65.4

Table 3: Performance, familiar sentences

GEN	INPUT			DAMAGE%:	0.0	25.0	50.0
1	boy	ate			85.0	66.7	63.3
1	woman	ate			83.3	66.7	63.3
2	woman	ate	chicken		93.3	65.0	58.3
2	man	ate	chicken		95.0	66.7	58.3
3	woman	ate	chicken	carrot	91.7	66.7	55.0
3	boy	ate	carrot	pasta	91.7	66.7	50.0
4	man	ate	chicken	fork	93.3	73.3	51.7
4	woman	ate	carrot	fork	91.7	73.3	58.3
5	bat	ate			71.7	56.7	56.7
5	chicken	ate			75.0	65.0	61.7
6	wolf	ate	chicken		78.3	66.7	56.7
6	wolf	ate	sheep		80.0	65.0	61.7
7	girl	broke	plate		90.0	71.7	58.3
7	woman	broke	plate		88.3	70.0	56.7
8	man	broke	vase	ball	90.0	73.3	66.7
8	girl	broke	vase	hatchet	95.0	76.7	75.0
9	hammer	broke	vase		93.3	85.0	71.7
9	ball	broke	vase		95.0	86.7	76.7
?10	bat	broke	vase		75.0	68.3	61.7
10	dog	broke	plate		73.3	68.3	53.3
11	plate	broke			81.7	80.0	73.3
11	plate	broke			81.7	80.0	73.3
12	boy	hit	girl		95.0	91.7	68.3
12	girl	hit	carrot		100.0	86.7	70.0
?13	man	hit	boy	hammer	80.0	71.7	46.7
13	boy	hit	woman	doll	91.7	76.7	56.7
14	girl	hit	curtain	ball	100.0	85.0	68.3
14	girl	hit	spoon	rock	98.3	78.3	63.3
15	paperwt	hit	chicken		86.7	85.0	81.7
15	rock	hit	plate		95.0	81.7	70.0
16	boy	moved			85.0	73.3	73.3
16	girl	moved			85.0	73.3	73.3
17	man	moved	window		100.0	75.0	70.0
17	girl	moved	hammer		100.0	81.7	71.7
18	wolf	moved			71.7	61.7	76.7
18	sheep	moved			65.0	61.7	71.7
19	paperwt	moved			75.0	68.3	58.3
19	hatchet	moved			96.7	83.3	78.3
AVERAGE%:					87.3	73.5	64.7

Table 4: Performance, unfamiliar sentences

The leftmost entry in each row identifies the generator which produced the sentence (referring to table 1). The degrees of damage were 0, 25 and 50 percent, meaning that 0, 3 and 6 units were removed from each input assembly. The figures indicate the percentage of output units whose values were within 15 percent of the correct output value. Ambiguous sentences are indicated with "?".

for their system since in all cases most of the output units should be off. Also, to learn the representations properly the training set should be as large as possible, whereas if the representations are predetermined, smaller training sets can be used. Generalization capabilities of the FGREP system as a function of training set size have not been tested.

4.3 Coding expectations about possible contexts

Representation is determined by all the contexts where the concept has been encountered. Consequently, *it is also a representation of all these contexts.* The more frequent the context, the stronger is its trace in the representation. When a concept is encountered and its representation activated, a large number of expectations about the context are immediately active with different degrees of confidence (figure 11). As more concepts are input, the expectations are amalgamated and the set of possible interpretations narrows down. The expectations emerge automatically and continuously from the input concept representations, which should turn out to be useful in building larger language understanding systems. These distributed expectations could replace the symbolic expectations traditionally used in natural language conceptual analyzers, e.g. [Dyer, 1983].

The FGREP approach is an implementation of the philosophy that *concepts are defined by the way they are used.* Learning a language is learning the usage of the language elements: *language is a skill.* The meaning of a concept is encoded in its representation. This is defined by all the contexts where the concept has been encountered, and it determines how the concept behaves in different contexts. The representation as well as the meaning evolves continuously as more experience is gained.

5 Future work

The prime direction of further work is to use the FGREP system as a building block in more complex cognitive systems. A major disadvantage is that the representations are task and network specific. Expectations and other knowledge embedded in them is directly available (without interpreters) only in the net-

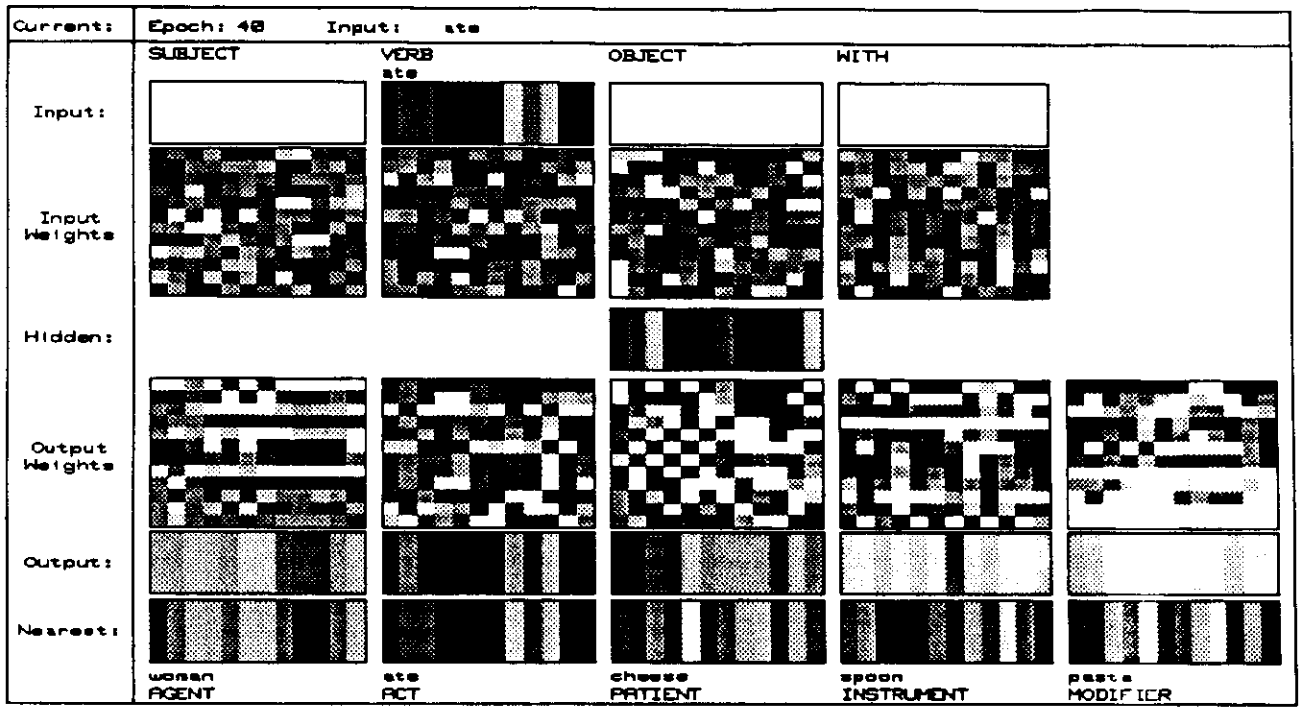

Figure 11: Expectations embedded in the word **ate**.

The bottom layer shows the lexicon entry which is closest to the output generated by the network (euclidian distance of normalized vectors). The representations and the network have captured the fact that a likely agent for **ate** is human, patient is food, instrument is a utensil, and that food can be eaten with other food.

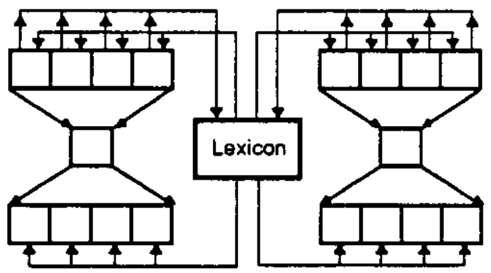

Figure 12: Two networks developing the same representations.

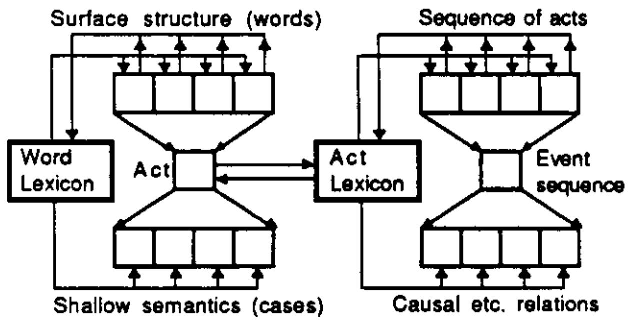

Figure 13: Hierarchical interaction through reduced descriptions.

work where the representations were developed. However, the representations do reflect the similarities in usage of the concepts, which is often useful information.

An FGREP network can serve as an input filter, self-organizing the input representations. Other systems can use these representations and add their own information on top of them. Alternatively, different subnetworks can develop the representations in the same lexicon simultaneously, as an extension of reactive training environment (figure 12). The final representations reflect the regularities in all subtasks and can be directly used by each subnetwork.

Hierarchical interaction of FGREP networks is possible through reduced descriptions (figure 13). A lower level network develops a reduced description of the whole input layer pattern in its hidden layer. These representations form the lexicon of another FGREP network which is learning higher level relations between the input patterns. Modifications by the higher level network to its input representations are propagated back to the lower level network by forming an error signal at the hidden layer, and changing the input weights and representations accordingly.

6 Conclusion

The FGREP method provides an alternative to semantic microfeature encoding and hidden layer learning of input/output representations. With backward error propagation extended to the input layer, meaningful global representations are developed automatically while the system is learning a processing task. Backpropagation operates in a reactive training environment, i.e. the required input/output mappings change as the I/O representations change.

There are no identifiable microfeatures nor discrete categories in the resulting representations. All aspects of an input item are distributed over the whole set of units in a holographic fashion, making the system particularly robust against damage. Each representation also carries expectations about its possible contexts.

The representations evolve to improve the system's performance in the processing task and therefore efficiently code the underlying relations relevant to the task. This results in very good association and generalization capabilities. Using the learned representations in the case role assignment task, the system was able to assign case roles correctly, indicate degree of confidence when the sentence was ambiguous, and generalize correctly for unfamiliar sentences.

References

[Dyer, 1983] Michael G. Dyer. *In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension.* MIT Press, Cambridge, MA, 1983.

[Hinton, 1981] Geoffrey E. Hinton. Implementing semantic networks in parallel hardware. In Geoffrey E. Hinton and James A. Anderson, editors, *Parallel Models for Associative Memory*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1981.

[Hinton, 1986] Geoffrey E. Hinton. Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Cognitive Science Society Conference*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.

[Hinton *et al.*, 1986] Geoffrey E. Hinton, James L. McClelland, and David E. Rumelhart. Distributed representations. In David E. Rumelhart and James L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume I: Foundations*, MIT Press, Cambridge, MA, 1986.

[Kohonen, 1982] Teuvo Kohonen. Clustering, taxonomy, and topological maps of patterns. In *Proceedings of the Sixth International Conference on Pattern Recognition*, IEEE Computer Society Press, 1982.

[Kohonen, 1984] Teuvo Kohonen. *Self-Organization and Associative Memory*, chapter 5. Springer-Verlag, Berlin; New York, 1984.

[McClelland and Kawamoto, 1986] James L. McClelland and Alan H. Kawamoto. Mechanisms of sentence processing: assigning roles to constituents. In James L. McClelland and David E. Rumelhart, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume II: Psychological and Biological Models*, MIT Press, Cambridge, MA, 1986.

[Miikkulainen and Dyer, 1988] Risto Miikkulainen and Michael G. Dyer. Forming global representations with extended backpropagation. In *Proceedings of the IEEE Second Annual International Conference on Neural Networks*, IEEE, 1988.

[Plaut *et al.*, 1986] David Plaut, Steven Nowlan, and Geoffrey E. Hinton. *Experiments on Back-Propagation.* Technical Report CMU-CS-86-126, Computer Science Department, Carnegie-Mellon University, 1986.

[Rumelhart *et al.*, 1986] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In David E. Rumelhart and James L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume I: Foundations*, MIT Press, Cambridge, MA, 1986.

[Tesauro and Sejnowski, 1988] Gerald Tesauro and Terrence J. Sejnowski. *A Parallel Network that Learns to Play Backgammon.* Technical Report CCSR-88-2, Center for Complex Systems Research, University of Illinois, Urbana-Champaign, 1988.