Integrated Connectionist Models: Building AI Systems on Subsymbolic Foundations *

Risto Miikkulainen
Department of Computer Sciences
The University of Texas at Austin, Austin, TX 78712-1188
risto@cs.utexas.edu

1 Introduction

Recently there has been a lot of excitement in cognitive science about the subsymbolic (i.e., parallel distributed processing, or distributed connectionist, or distributed neural network) approach. Subsymbolic systems seem to capture a number of intriguing properties of human-like information processing such as learning from examples, context sensitivity, generalization, robustness of behavior, and intuitive reasoning. These properties have been very difficult to model with traditional, symbolic techniques.

Within this new paradigm, the central issues are quite different (even incompatible) from the traditional issues in symbolic cognitive science, and the research has proceeded without much in common with the past. However, the ultimate goal is still the same: to understand how human cognition is put together. Even if cognitive science is being built on a new foundation, as can be argued, many of the results obtained through symbolic research are still valid, and could be used as a guideline for developing subsymbolic models of cognitive processes.

This is the approach taken in building DISCERN (DIstributed SCript processing and Episodic memoRy Network; Miikkulainen, 1993), a distributed neural network model of script-based story understanding. DISCERN is purely a subsymbolic model, but at the high level it consists of modules and information structures similar to those of symbolic systems, such as scripts, lexicon, and episodic memory. At the highest level of cognitive modeling, the symbolic and subsymbolic paradigms have to address the same basic issues. Outlining a parallel distributed approach to those issues is the purpose of DISCERN.

DISCERN is an integrated connectionist architecture. Independent neural network models of the various subtasks are brought together into a single system capable of performing the high-level cognitive task. In this chapter, I present motivation for such integrated connectionist models in general, describe the DISCERN system as an example, and discuss some of the main issues and prospects of the approach.

*To appear in Honavar, V., and Uhr, L. (1994). *Artificial Intelligence and Neural Networks: Steps toward Principled Integration.* New York: Academic Press.

2 Why subsymbolic AI?

Symbolic artificial intelligence is founded on the hypothesis that symbol manipulation is both necessary and sufficient for intelligence (Newell, 1980). Symbolic systems have been quite successful, for example, in modeling in-depth natural language processing (Dyer, 1983; Schank and Abelson, 1977), episodic memory (Kolodner, 1984), and problem solving (Laird *et al.*, 1987; Newell, 1991). However, there are two major issues that the symbolic approach does not address: the statistical (intuitive) nature of certain cognitive processes, and the physical implementation of the cognitive system.

It seems that people have two fundamentally different mechanisms at their disposal for performing cognitive tasks. Following a sequential symbolic strategy is the more obvious of the two. Here, one does not have an immediate answer to the problem, but the answer is sequentially constructed from stored knowledge by a high-level goal-directed process, that is, by reasoning. Another type of cognitive processing occurs through associations immediately, in parallel, and without conscious control, in other words, by intuition. Large amounts of information, which may be incomplete or even conflicting, are simultaneously brought together to produce the most likely answer.

In symbolic systems, knowledge is encoded in terms of explicit symbolic structures, and inferences are based on handcrafted rules that operate on these structures. For cognitive processes based on conscious rule application, such systems are a good approximation. However, intuitive processing cannot be easily implemented. In contrast, neural networks represent knowledge in terms of correlations, coded in the weights of the network. For a given input, the network computes the most likely answer given its past experience. The process is opaque, nonconcatenative, and immediate, and fits very well into modeling intuitive inference (see also Hinton, 1990; Smolensky, 1988; Touretzky, 1991.

The first major motivation for subsymbolic AI, therefore, is to give a better account for high-level cognitive phenomena that are statistical, or intuitive, in nature. For example, the DISCERN system aims at demonstrating how script-based inferences can be learned from experience, how episodic memory organization can be automatically formed based on regularities in the experiences, how word semantics can be learned from examples of word use, and how expectations and defaults automatically emerge from correlations in the data.

Symbolic models are high-level process models, far removed from the physical structures that implement the processes in the brain. As a result, they inherently lack the capability of explaining certain aspects of human performance. In the symbolic framework, it is very difficult to address certain issues, for example: Where do performance errors come from? How can memory become overloaded and why do certain types of memory confusions occur in overload situations? What happens when the system is corrupted with noise, or when parts of it are destroyed?

While artificial neural networks are still only an abstraction of actual neural structures, they are motivated by the fundamental properties of information processing in neural hardware. As a result, they have a good chance of explaining behavioral phenomena that arise from the physical organization of the brain, such as certain performance errors and deficits. This is the second major motivation for subsymbolic AI. For example, the physical organization of the DISCERN system determines what kinds of memory interference, role binding, and lexical errors can occur under noise and damage, and these errors turn out to be quite similar to those observed in human subjects.

3 Why integrated connectionist models?

At the outset, it is not obvious that large-scale artificial intelligence systems can be built from distributed neural networks. Previous research in parallel distributed processing (PDP) has concentrated mostly on isolated, small, low-level tasks, and relied heavily on pre- and postprocessed data (see, e.g., McClelland *et al.*, 1986; Rumelhart *et al.*, 1986c for an overview). In many cases, the problem is reduced to learning a simple mapping. This suits modeling isolated low-level tasks, such as learning past tense forms of verbs (Rumelhart and McClelland, 1986) or pronunciation of words (Sejnowski and Rosenberg, 1987). However, modeling higher-level cognitive tasks with simple pattern transformation networks has been infeasible, for three reasons:

1. High-level tasks are often composites of distinct subtasks. They consist of several interacting subprocesses, such as parsing language, generating language, memory storage, memory retrieval, and reasoning. Complex behavior requires bringing together several different kinds of knowledge sources and processes, something that cannot be done in a single pattern transformation. Such behavior requires structured architectures (Feldman, 1989; Minsky, 1985; Simon, 1969), and combinations of different types of networks.

2. The required network size, the number of training examples, and the training time become intractable as the size of the problem grows (Elman, 1991; Harris and Elman, 1989; St. John and McClelland, 1990; St. John, 1992).

3. It is very difficult to evaluate what the entire system is doing, for example, what knowledge it is acquiring and applying, unless each module processes meaningful internal representations that can be interpreted by an external observer and by other modules in the system.

A logical approach for high-level cognitive modeling, therefore, is to construct the architecture from several interacting modules that work together to produce the high-level behavior. While modularity is practical from the engineering point of view, it is also a plausible cognitive theory. If one believes that the structure of cognitive behavior reflects the structure of its underlying physical implementation, faculties such as episodic memory, lexicon, parsing and generating language, and reasoning should be based on different modules (for related arguments for modularity, see e.g. Fodor, 1983; Minsky, 1985; Shallice, 1988).

The integrated connectionist approach primarily aims at building complete systems that perform well in high-level tasks, and in this sense, the approach is very similar to traditional artificial intelligence. In addition, integrated models try to show how certain parts of human cognition could actually be built. The components are designed as independent cognitive models that, by themselves, account for interesting perception, language-processing, reasoning, and memory phenomena. Combining these models into a single, working system is one way of validating them. In an integrated system, the components are not just models of isolated cognitive phenomena; they are also sufficient constituents for generating complex high-level behavior.

4 Overview of DISCERN

The DISCERN system is a first implementation of the integrated connectionist approach. The high-level task of script processing is broken into hierarchical subtasks, implemented by independent connectionist modules that communicate using distributed representations. The DISCERN

RESTAURANT SCRIPT		
FANCY-RESTAURANT TRACK		
Causal Chain:	**Roles:**	
Entering	Customer	= `John`
Seating	Restaurant	= `MaMaison`
Ordering	Food	= `lobster`
Eating	Taste	= `good`
Paying	Tip	= `big`
Tipping		
Leaving		

Table 1: **Representation of a script-based story as a causal chain and role bindings.** This particular instantiation is a simplified version of the fancy-restaurant script, where only the main events have been listed.

components are cognitive models in their own right; as a whole, DISCERN is a complete AI system that performs approximately at the level of symbolic natural-language processing models.

Before describing the overall architecture and the component models in more detail, I will briefly review the script processing task and outline the issues involved.

4.1 The script processing task

Scripts (Schank and Abelson, 1977; Cullingford, 1978; Dyer *et al.*, 1987) are schemas of often encountered, stereotypical event sequences, such as visiting a restaurant, traveling by airplane, or shopping at a supermarket. Each script divides further into *tracks*, or established minor variations. A script can be represented as a causal chain of events with a number of open roles (table 1). Script-based understanding means reading a script-based story, identifying the proper script and track, and filling its roles with the constituents of the story. Events and role fillers that were not mentioned in the story but are part of the script can then be inferred. Understanding is demonstrated by generating an expanded paraphrase of the original story, and by answering questions about the story.

To see what is involved in the task, let us consider an example of DISCERN input/output behavior. The input stories are based on the fancy-restaurant, plane-travel, and electronics-shopping tracks:

```
   John went to MaMaison.  John asked the waiter for lobster. John left the waiter
a big tip.

   John went to LAX. John checked in for a flight to JFK. The plane landed at JFK.

   John went to Radio-Shack.  John asked the staff questions about CD-players.  John
chose the best CD-player.
```

DISCERN reads the orthographic word symbols sequentially one at a time. An internal representation of each story is formed, where all inferences are made explicit. These representations are stored in an episodic memory. The system then answers questions about the stories:

```
What did John buy at Radio-Shack?
```

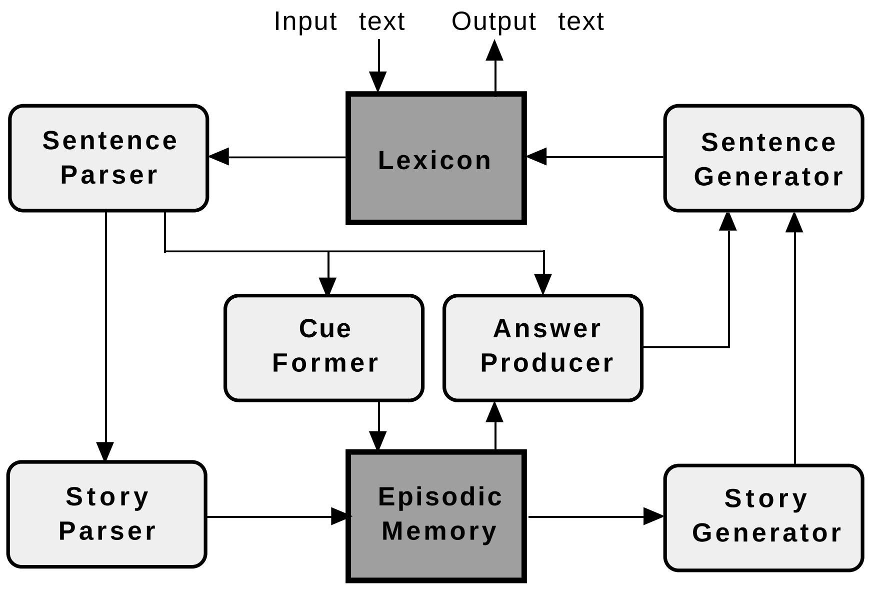


FIGURE 1: **The DISCERN architecture (performance configuration).** The model consists of parsing, generating, question answering, and memory subsystems, two modules each. A dark square indicates a memory module, a light square indicates a processing module.

```
 John bought a CD-player at Radio-Shack.
Where did John fly to?
 John flew to JFK.
What did John eat at MaMaison?
 John ate a good lobster.
```

With the question as a cue, the appropriate story representation is retrieved from the episodic memory, and the answer is generated word by word. DISCERN also generates full paraphrases of the input stories. For example, it generates an expanded version of the restaurant story:

```
John went to MaMaison. The waiter seated John. John asked the waiter for lobster.  John
ate a good lobster.  John paid the waiter.  John left a big tip.  John left MaMaison.
```

A number of issues can be identified from the above examples. Specifically, DISCERN has to (1) make statistical script-based inferences and account for learning them from experience; (2) store items in the episodic memory in a single presentation and retrieve them with a partial cue; (3) develop a meaningful organization for the episodic memory, based on the stories it reads; (4) represent meanings of words, sentences, and stories internally; (5) based on examples of how words are used in the language, organize a lexicon of symbol and concept representations and form a many-to-many mapping between them.

4.2 The DISCERN architecture

DISCERN can be divided into parsing, generating, question answering, and memory subsystems, each with two modules (Figure 1). Each module is trained in its task separately and in parallel. During performance, the modules form a network of networks, each feeding its output to the input of another module.

The sentence parser reads the input words one at a time, and forms a representation of each sentence. The story parser combines the sequence of sentences into an internal representation of the story, which is then stored in the episodic memory. The story generator receives the internal representation and generates the sentences of the paraphrase one at a time. The sentence generator outputs the sequence of words for each sentence.

The cue former receives a question representation, built by the sentence parser, and forms a cue pattern for the episodic memory, which returns the appropriate story representation. The answer producer receives the question and the story and generates an answer representation, which is output word by word by the sentence generator.

5 Lexicon

The input and output of DISCERN consists of distributed representations for orthographic word symbols (also called lexical words below). Internally, DISCERN processes semantic concept representations (semantic words). Both the lexical and semantic words are represented distributively as vectors of gray-scale values between 0.0 and 1.0. The lexical representations are based on the visual patterns of characters that make up the written words. They remain fixed throughout the training and performance of DISCERN. The semantic representations stand for distinct meanings. They are developed automatically by the system while it is learning the processing task.

The lexicon (Miikkulainen, 1990a) stores the lexical and semantic representations and translates between them (Figure 2). It is implemented as two feature maps (Kohonen, 1989, 1990), one lexical and the other semantic. Words whose lexical forms are similar, such as `BALL` and `DOLL`, are represented by nearby units in the lexical map. In the semantic map, words with similar semantic content, such as `predator` and `prey`, are mapped near each other.

The two maps are densely interconnected with associative connections. A localized activity pattern representing a word in one map will cause a localized activity pattern to form in the other map, representing the same word (Figure 2). The output representation is then obtained from the weight vector of the most highly active unit. The lexicon thus transforms a lexical input vector into a semantic output vector, and vice versa. Both maps and the associative connections between them are organized simultaneously, based on examples of co-occurring symbols and meanings.

The lexicon architecture facilitates interesting behavior. Localized damage to the semantic map results in category-specific lexical deficits similar to human aphasia (see, e.g., Caramazza, 1988; McCarthy and Warrington, 1990). Dyslexic performance errors can also be modeled. If the performance is degraded, for example, by adding noise to the connections, this will result in two types of parsing errors and two types of generation errors. In parsing (1) a lexical input pattern may be mapped incorrectly onto a nearby unit in the lexical map, which corresponds to reading the word incorrectly; (2) the activity in the lexical map may propagate incorrectly to a nearby unit in the semantic map. For example, `CHICKEN` is understood semantically as `livebat`. Analogously, in generation (1) a semantic input pattern could be recognized incorrectly, and a word with a similar but incorrect meaning produced; (2) activity may propagate to an incorrect unit in the lexical map, and a word with a similar orthographic form but different meaning output. These types of errors are common in human deep dyslexia as well (Coltheart *et al.*, 1988).

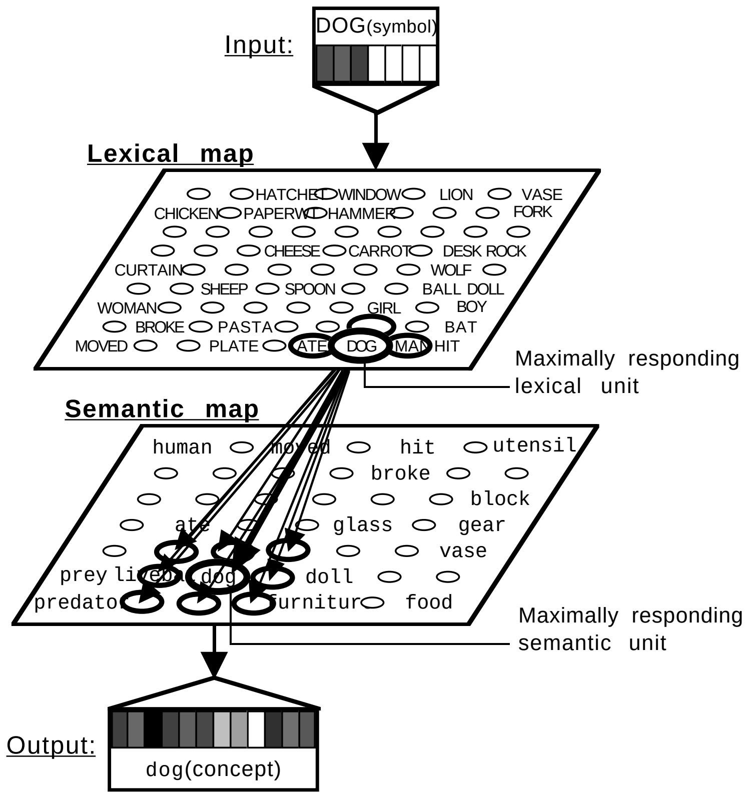


FIGURE 2: **The lexicon.** The lexical input symbol `DOG` is translated into the semantic representation of the concept `dog`. The representations are vectors of gray-scale values between 0.0 and 1.0, stored in the weights of the units. The size of the unit on the map indicates how strongly it responds. Only a few strongest associative connections of the lexical unit `DOG` are shown.

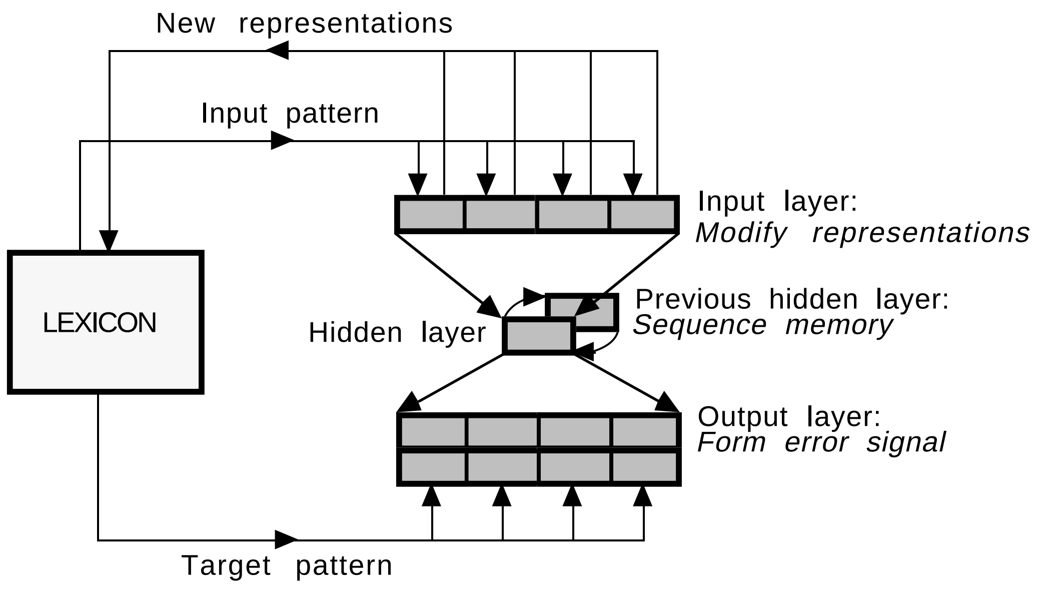


FIGURE 3: **The FGREP-module.** In the case of sequential input or output, the hidden layer pattern is saved after each step in the sequence, and used as input to the hidden layer during the next step, together with the actual input.

6 FGREP processing modules

Processing in DISCERN is carried out by hierarchically organized FGREP modules. Each module performs a specific subtask, such as parsing a sentence or generating an answer to a question. All these modules have the same basic architecture.

The FGREP mechanism (Forming Global Representations with Extended backPropagation) (Miikkulainen and Dyer, 1991) is based on a basic three-layer backward error propagation network, with the I/O representation patterns stored in an external lexicon[1] (Figure 3). The input and output layers of the network are divided into assemblies. A routing network forms each input pattern and the corresponding teaching pattern by concatenating the semantic lexicon entries of the input and teaching items.

The network learns the processing task by adapting the connection weights according to the standard backpropagation procedure (Rumelhart *et al.*, 1986b, pp. 327–329). At the end of each cycle, the current input representations are modified at the input layer based on the error signal. The modified representations are put back to the lexicon, replacing the old ones and thereby changing the next teaching pattern for the same input. In other words, backpropagation is shooting at a moving target in a reactive training environment.

The representations that result from this process have a number of interesting properties. Since they adapt to the error signal, the representations end up coding properties most crucial to the task. Representations for words that are used in similar ways in the examples become similar. Thus, these profiles of continuous activity values can be claimed to code the meanings of the words as well. Interestingly, single representation components do not usually stand for identifiable semantic

[1]Technically, the FGREP lexicon consists of only the semantic component of the lexicon model, that is, it is a storage for semantic representations.

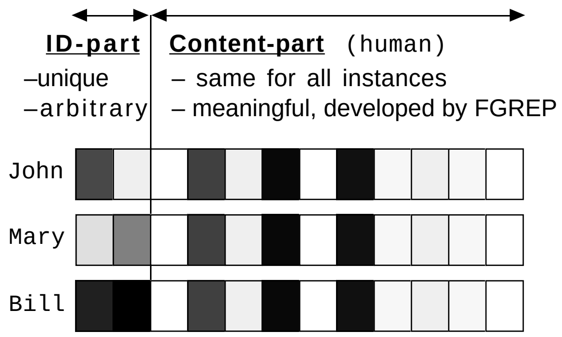


FIGURE 4: **Cloning word instances.** Instances `John`, `Mary`, and `Bill` are created from the prototype word `human`.

features. Instead, the representation is holographic. Word categories can often be recovered from the values of single components, making the system very robust against damage. Performance degrades approximately linearly as representation components become defective.

The representation of a word is determined by all the contexts in which that word has been encountered, and consequently, it is also a representation of all those contexts. Expectations emerge automatically and cumulatively from the input word representations. Also, the system never has to process very novel input patterns, because generalization has already been done in the representations.

Three types of FGREP modules are used in the system: nonrecurrent (the cue former and the answer producer), sequential input (the parsers), and sequential output modules (the generators). In the recurrent modules the previous hidden layer serves as sequence memory, remembering at what point in the sequence the system currently is and what has occurred before (Elman, 1990; Figure 3). In a sequential input network, the input changes at each time step, while the teaching pattern stays the same. The network learns to form a stationary representation of the sequence. In a sequential output network, the input is stationary, but the teaching pattern changes at each step. The network learns to produce a sequential interpretation of its input.

It is possible to extend the FGREP lexicon by creating a number of distinct word instances from the same semantic word, such as the tokens `John, Mary, Bill` from the type `human` (Figure 4). The representation now consists of two parts: the content part, which was developed in the FGREP process and encodes the meaning of the word, and the ID part, which is unique to each instance of the same word. The ID part has no intrinsic meaning in the system, only distinguishing one instance of a word from all other instances of the same word. The technique can be thought of as an approximation of sensory grounding, where the ID part stands for the sensory referent of the word.

The ID+content technique can be applied to any word in the training data, and in principle, the number of instances per word is unlimited. This allows us to approximate a large vocabulary with only a small number of semantically distinct representations at our disposal. Word discrimination degrades approximately linearly as a function of the number of instances. This is remarkable since the number of different input/output patterns grows polynomially.

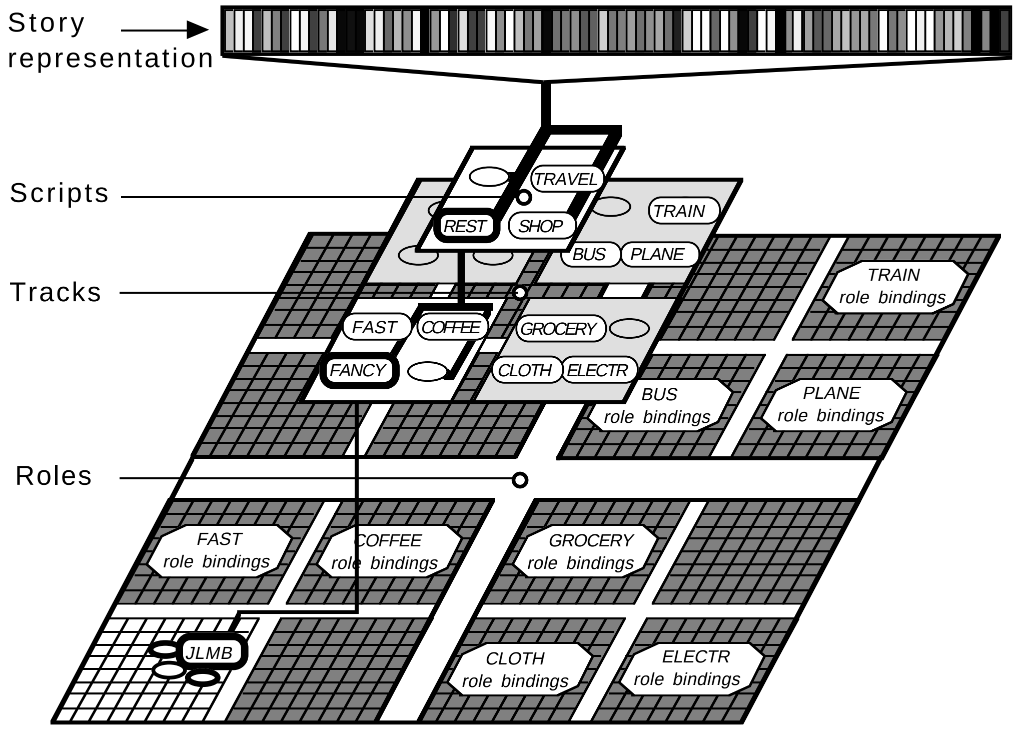


FIGURE 5: **The hierarchical feature map classification of script-based stories.** Labels indicate the maximally responding unit for the different scripts and tracks. This particular input story representation is classified as an instance of the restaurant script (top level) and fancy-restaurant track (middle level), with role bindings customer=`John`, food=`lobster`, restaurant=`MaMaison`, tip=`big` (i.e., JLMB, bottom level).

7 Episodic memory

The episodic memory in DISCERN is a hierarchical feature map system (Miikkulainen, 1990c) combined with the trace feature map mechanism (Miikkulainen, 1992). The map hierarchy provides the organization for the memory, and the trace feature map technique implements storage and retrieval of memory traces.

7.1 Map hierarchy

The feature map hierarchy is a pyramid organized according to the hierarchical taxonomy of script-based stories (Figure 5). The highest level of the hierarchy is a single feature map that lays out the different script classes. Beneath each unit of this map there is another feature map that lays out the tracks within the script. The different role bindings within each track are separated at the bottom level. The map hierarchy receives a story representation as its input and classifies it as an instance of a particular script, track, and role binding. In other words, the map hierarchy provides a unique memory representation for each script-based story.

Let us follow the classification of the story about John's visit to MaMaison. The top-level map receives the complete story representation vector and maps it onto the unit labeled REST, for restaurant script (Figure 5). This unit compresses the vector by removing components whose values are the same in all restaurant stories. The representation now consists of information that best distinguishes between the different restaurant stories. The REST-unit passes the compressed representation down to its submap, which classifies it as an instance of the fancy-restaurant track. Again, the FANCY-unit removes the components common to all fancy-restaurant stories, and passes the highly compressed vector to its submap. The representation is now limited to information about the role bindings, and it is mapped onto the unit representing customer=`John`, food=`lobster`, restaurant=`MaMaison`, tip=`big`.

A higher-level map in the hierarchy acts as a filter: (1) it chooses the relevant input items for each lower-level map, and (2) compresses the representation of these items to the most relevant components. Maps lower in the hierarchy form increasingly finer distinctions between the stories.

The hierarchical script taxonomy is extracted from examples of story representations. The pyramid structure itself is predetermined and fixed, but the maps are self-organized one level at a time from top to bottom. Each unit independently determines how to compress its input vectors by finding components with least variance.

Hierarchical feature maps have a number of properties that make them useful for memory organization: (1) The organization reflects the properties of the data, the hierarchy representing its taxonomy, and maps laying out the topology of each level. (2) The most salient components of the input data are singled out, and more resources are allocated for representing them accurately. (3) The classification is very robust, and usually correct even if the input vector is noisy or incomplete. (4) The organization is formed in an unsupervised manner, extracting it from input examples. (5) Self-organizing a hierarchy of small maps instead of a single large one means dividing the classification task into hierarchical subgoals, which is an efficient way to reduce complexity.

7.2 Trace feature maps

An ordinary feature map is a classifier, mapping an input vector onto a location in the map. A trace feature map, in addition, creates a memory trace at that location. The map remembers that at some point previously it had received an input item which was classified at the particular location. The traces can be stored one at a time, as stories are read in, and retrieved with a partial cue.

A trace feature map is a single ordered feature map with modifiable lateral connections between the units (Figure 6). Initially, the lateral connections are all inhibitory. When an input vector is presented to this map, a localized activity pattern forms as a response. A trace is created by modifying the lateral connections within this response. A connection to a unit with higher activity is made excitatory, while a connection to a unit with lower activity is made inhibitory, both proportionally to the activity level of the source unit. The units within the response now "point" towards the unit with highest activity (Figure 6).

A stored vector is retrieved by giving the map an approximation of the vector as a cue. The initial response is again a localized activity pattern, and because the map is topological, it is likely to be located somewhere near the stored trace. If the cue is close enough, the lateral connections pull the activity to the center of the trace, and the external input weights of the most highly active unit give the stored vector. If the cue is too far away, the initial response does not reach the "basin" of the trace, and the activity oscillates between nonactivity (caused by the inhibitory lateral connections) and the initial response. In other words, the trace feature map can complete a partial cue, and indicate when there is no appropriate trace in the memory.

The trace feature map exhibits interesting memory effects that result from interactions between traces. Later traces capture units from earlier ones, making later traces more likely to be retrieved

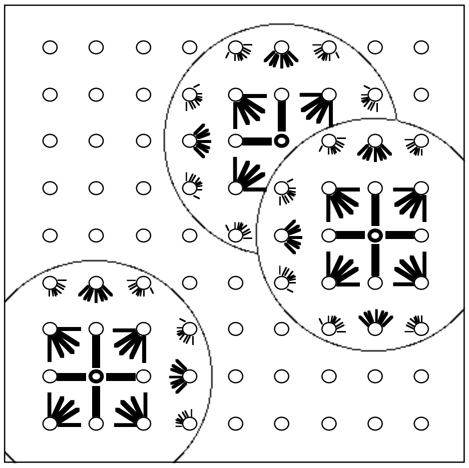


FIGURE 6: **A trace feature map.** Three traces (delineated by the large circles) are stored in the map. Line segments indicate excitatory lateral connections originating from each unit, with length and width proportional to the magnitude of the weight. Inhibitory connections are not shown. The trace at right has partially obscured an earlier trace.

(Figure 6). The extent of the basins determines the memory capacity. The smaller the basins, the more traces will fit in the map, but more accurate cues are required to retrieve them. If the memory capacity is exceeded, older traces will be selectively replaced by newer ones. Traces that are unique, that is, located in a sparse area of the map, are not affected, no matter how old they are. Similar effects are common in human long-term memory (Baddeley, 1976; Postman, 1971).

7.3 Storage and retrieval

A story is represented in the episodic memory by the maximally responding units at the script, track, and role-binding levels. However, each unit on a role-binding map stands for a unique story, and a trace needs to be created only at the bottom level. The script and the track level are ordinary feature maps, while the role-binding level consists of trace feature maps.

When a representation is stored in the episodic memory, the map hierarchy determines the appropriate role-binding map and the location on that map. The trace feature map mechanism then creates a memory trace at that location.

A story is retrieved from the memory by giving it a partial story representation as a cue. Unless the cue is highly deficient, the map hierarchy is able to recognize it as an instance of the correct script and track, and form a partial cue to the role-binding map. The trace feature map mechanism then completes the role binding. The complete story representation is retrieved from the weight vectors of the maximally responding units at the script, track, and role-binding levels.

8 Connecting the modules in DISCERN

8.1 Performance phase

Let us follow DISCERN (Figure 1) as it processes the story about John's visit to MaMaison. The lexical representations of each word are presented to the lexical map of the lexicon, which produces the corresponding semantic representation as its output (Figure 2). These are fed one at a time to the sentence parser, which gradually forms a stationary case-role representation of each sentence at its output layer (Figure 7). After a period is input, ending the sentence, the final case-role pattern is fed to the input of the story parser.

In a similar manner, the story parser receives a sequence of sentence case-role representations as its input, and forms a stationary slot-filler representation of the whole story at its output layer (Figure 8). This is a representation of the story in terms of its role bindings, and constitutes the final result of the parse. The story representation is fed to the episodic memory, which classifies it as an instance of a particular script, track, and role binding, and creates a trace in the appropriate role-binding map (Figure 5).

The generator subsystem reverses the parsing process. The story generator network receives the story representation as its input and generates a sequence of sentence case-role representations. Each of these is fed to the sentence generator, which outputs the semantic representations of the output words one at a time. Finally, the lexicon transforms these into lexical words.

The sentence parser and the sentence generator are also trained to process question sentences and answer sentences. The cue former receives the case-role representation of the question (Figure 9) produced by the sentence parser, and generates an approximate story representation as its output (Figure 10). This pattern is fed to the episodic memory, which classifies it as an instance of a script, track, and role binding. The trace feature map settles to a previously stored memory trace, and the complete story representation (Figure 8) is retrieved from the weights of the maximally responding units.

The answer producer receives the complete story representation, together with the case-role representation of the question, and generates a case-role representation of the answer sentence (Figure 11), which is then output word by word by the sentence generator.

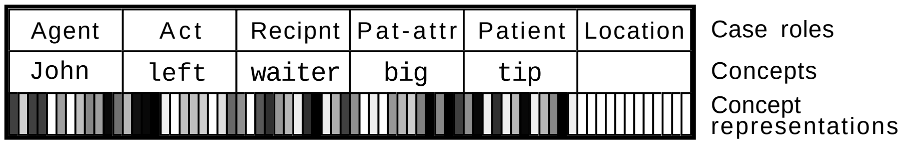
8.2 Training phase

A good advantage of the modular architecture can be made in training the system (Figure 12). The tasks of the six processing modules are separable, and they can be trained separately as long as compatible I/O material is used. The modules must be trained simultaneously to ensure that they will develop and learn to use the same semantic representations. The hierarchical organization of the episodic memory can be developed at the same time.

The lexicon ties the separated tasks together. Each FGREP network tries to modify the representations in such a manner that its performance in its own task would improve. The requirements of the different tasks are combined, and the representations are never exactly optimal to any individual network. The networks compensate by adapting their weights, so that, in the end, the representations and weights of all networks are in harmony: the output patterns produced by one network are exactly what the next network had learned to process as its input, and the final representations reflect the combined use of the words in the six tasks.

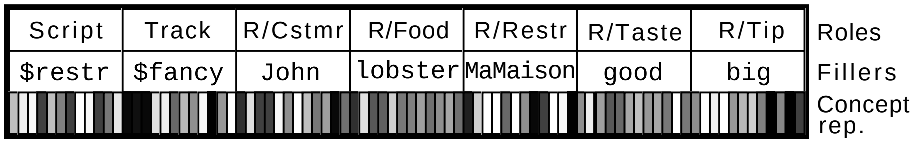
9 Discussion of DISCERN

The complete DISCERN system performs very well in the script processing task. Missing events and role fillers are inferred whenever possible, and at the output, about 98% of the words are correct (tested with 96 three-sentence stories, instantiated from three scripts, each with three



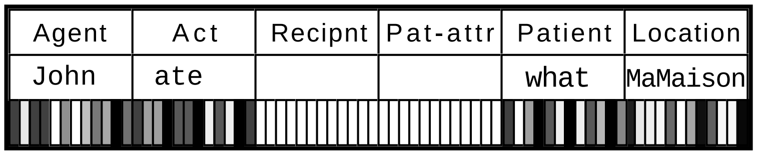
Agent	Act	Recipnt	Pat-attr	Patient	Location
John	left	waiter	big	tip	

FIGURE 7: **Case-role representation of the sentence** `John left the waiter a big tip.` The concept representations in each case-role correspond to the concept representations in the lexicon.



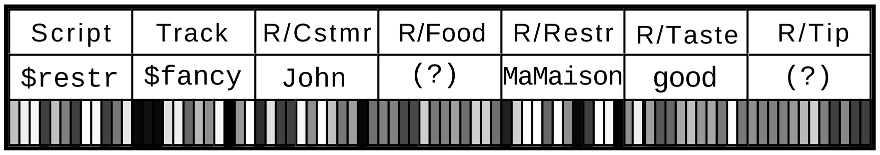
Script	Track	R/Cstmr	R/Food	R/Restr	R/Taste	R/Tip
$restr	$fancy	John	lobster	MaMaison	good	big

FIGURE 8: **Representation of the story by its role bindings.** The assemblies are data-specific: their interpretation depends on the pattern in the script slot. The role names R/... are specific for the restaurant script.



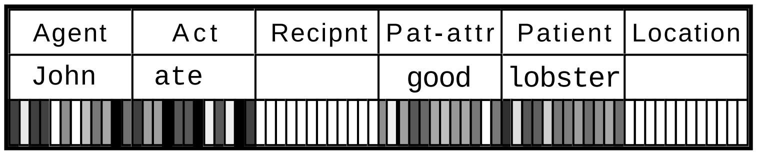
Agent	Act	Recipnt	Pat-attr	Patient	Location
John	ate			what	MaMaison

FIGURE 9: **Case-role representation of the question** `What did John eat at MaMaison?` Questions are represented as sentences, but processed through a different pathway.



Script	Track	R/Cstmr	R/Food	R/Restr	R/Taste	R/Tip
$restr	$fancy	John	(?)	MaMaison	good	(?)

FIGURE 10: **Memory cue.** Most of the story representation is complete, but the patterns in Food and Tip slots indicate averages of all possible alternatives.



Agent	Act	Recipnt	Pat-attr	Patient	Location
John	ate		good	lobster	

FIGURE 11: **Case-role representation of the answer** `John ate a good lobster.`

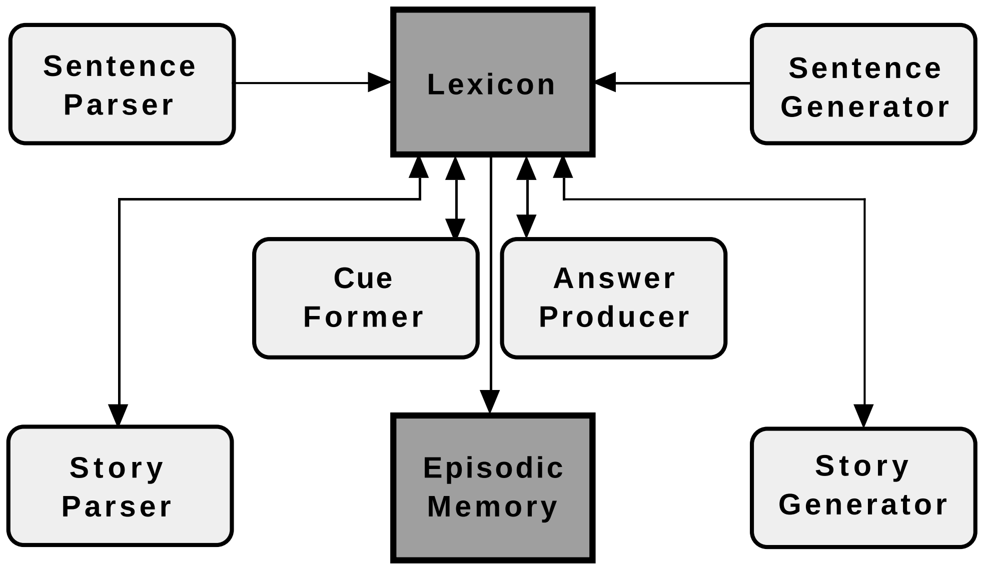


FIGURE 12: **Training configuration.** Each module is trained separately and simultaneously with compatible I/O data (originating from the same example stories). There is no propagation between modules, but they simultaneously perform different parts of the same story processing tasks. All processing modules also modify the same representation set in the lexicon.

tracks, each with three open roles, and two instances cloned for each filler word). If there is not enough information to fill a role, the most likely filler is selected and maintained throughout paraphrase generation. Thus, DISCERN performs plausible role bindings—an essential task in high-level inferencing and one that has been postulated to be very difficult for parallel distributed processing systems to achieve (Dyer, 1991).

The system extracts the appropriate inferences automatically, based on statistical correlations in the input examples. This differs from the symbolic models of script processing (Schank and Abelson, 1977; Cullingford, 1978; Dyer, 1983), where the inferences are based on handcrafted rules and representations of the script. Similarly, the episodic memory is organized automatically according to similarities between example stories, as opposed to hand-coded memory structures and organization rules in comparable symbolic systems, such as those of Lebowitz (1980) and Kolodner (1984). On the other hand, DISCERN currently does not have mechanisms for representing multiple simultaneously active scripts, or stories that consist of sequential activation and deactivation of multiple scripts.

A modular PDP system can only operate if it is stable, that is, if small deviations from the normal flow of information are automatically corrected. DISCERN has several automatic safeguards against minor inaccuracies and noise. The semantic representations are distributed and redundant, and inaccuracies in the output of one module are cleaned up by the module that uses the output. The memory modules clean up by categorical processing: a noisy input is recognized as a representative of an established class and replaced by the correct representation of that class. As a result, small deviations do not throw the system off course, but rather the system filters out the errors and returns to the normal course of processing.

The subsymbolic approach is effective in script processing because scripts are regular event sequences, and their structure can easily be extracted by a neural network system. Script inferences are intuitive, immediate, and occur without conscious control, matching the statistical nature of processing in distributed networks. However, DISCERN cannot handle deviations from scripts very well. For example, in the training data the customer never leaves a tip in a fast-food restaurant. If later DISCERN encounters a fast-food-restaurant story where customer actually does leave a tip, DISCERN will simply override this information. Tip=`none` has become part of the established fast-food-restaurant track, and DISCERN has no mechanism to represent a deviation from the regular course of events. Another major limitation of DISCERN as a cognitive model is that the slot-filler representations for each script must be designed by hand. DISCERN does not discover new scripts by itself.

10 Further issues in building integrated connectionist models

Although processing simple script instantiations is a start, there is a long way to go before integrated connectionist models will rival the best symbolic AI systems. For example, in story understanding, symbolic systems have been developed that analyze realistic stories in-depth, based on higher-level knowledge structures such as goals, plans, themes, affects, beliefs, argument structures, plots, and morals (e.g. Alvarado *et al.*, 1990; Dyer, 1983; Reeves, 1991; Schank and Abelson, 1977). Subsymbolic systems are not yet capable of modeling cognitive processes at this level. Subsymbolic systems are very good at dealing with regularities and combining large amounts of simple pieces of evidence, but they do not easily lend themselves to processing complex knowledge structures and unusual and novel situations. In designing subsymbolic models that would do that, we are faced with two major problems: (1) how should control of complex processing strategies be implemented, and (2) how should abstractions be represented and learned.

10.1 Connectionist control of cognitive processes

Integrated connectionist models such as DISCERN demonstrate how far it is possible to go in modeling high-level behavior by combining simple low-level processes. DISCERN is not "conscious" of what it is doing, that is, it does not have representations concerning the nature of its own representations and processes. As a result, it cannot employ high-level strategies to control its own processes; its behavior is limited to a series of reflex responses.

With a high-level monitor and control mechanism, it would be possible to build much more powerful subsymbolic models:

1. Current systems try to process every input in exactly the same way, regardless of whether or not the input makes sense. A high-level control system could monitor the feasibility of the task and the quality of output, and initiate exception processing when the usual mechanisms fail. For example, unusual events and deviations from a script could be detected and then processed by special mechanisms.

2. Retrieval from episodic memory could be controlled intelligently. The monitor could decide how to search for traces and how to respond when multiple traces are found, or when nothing is found.

3. The monitor could keep the system on a stable path by detecting and correcting internal inaccuracies that cannot be caught by the automatic low-level filters (such as confusions between IDs in DISCERN).

4. Sequential high-level procedures and reasoning mechanisms could be implemented, such as comparing several items in the memory and applying high-level rules to conclude new information.

It might be possible to implement the necessary monitoring, modulation, and decision-making tasks with trainable control networks. These modules would receive input from several pathways in the system, thus monitoring its state, and their output would gate the system pathways through multiplicative connections (such as those of Pollack, 1987; Rumelhart *et al.*, 1986a). Equipped with such mechanisms, subsymbolic models would no longer be limited to automatic reflex behavior, and would be able to perform much more robustly in the real world.

10.2 Representing and learning abstractions

Representing structure, processing exceptions, and making dynamic inferences are often cited as the three main challenges for subsymbolic artificial intelligence. I will argue that these problems are closely related, and stem from the fundamental limitation of the current architectures in representing and learning abstractions of data.

Let us briefly review each problem. First, assembly-based representations, as they are commonly used in subsymbolic systems, must be designed in advance and remain fixed. Units cannot be created or moved around in the network, but can only function in the same exact configuration they were trained in. It is very difficult to represent multiple fillers (e.g., `the man and the boy`), additional constituents, and recursive structures (see Dolan, 1989; Hinton, 1990; Pollack, 1990; Smolensky, 1990 for possible approaches).

Second, processing knowledge in PDP models is based on statistical regularities; PDP models cannot handle deviations from regularities very well (Miikkulainen and Dyer, 1989; St. John, 1992). Exceptions are simply overridden. The network has no representation for all-or-none role bindings, and as a result it cannot process truly novel inputs according to a symbolic-like higher-level rule.

The third problem, that of dynamic inferencing (Touretzky, 1991), is evident in such areas as processing relative clauses. A sentence-processing network may be able to generalize to different versions of the same sentence structure, but not to new recursive clause structures (Miikkulainen, 1990b). It cannot make inferences by dynamically combining processing knowledge it has previously seen only in separate situations.

All these three problems originate from the use of distributed neural networks as statistical pattern transformers. The networks are trained to compute smooth functions between patterns, and as a result they are only able to interpolate between the patterns they were trained on. They cannot represent different alternatives distinctly, and they cannot extrapolate to new patterns. In order to process novel constituents and novel structures, the networks must be supplied with meta-level information that describes the structure of the data, such as schemas, rules, and abstractions. A novel input needs to be recognized in terms of meta-level categories, and processed according to abstract knowledge in that category. In terms of the above problems:

1. A flexible representation must contain information telling what the components are and how they are related.

2. Processing exceptions requires explicit representation of the general rules and variable bindings, so that the network is able to choose to use them to override the statistical regularities.

3. Dynamic inferencing is only possible if the meta-structure of the input is explicitly represented. For example, if the sentence-processing system is able to represent the general structure of a relative clause, it would be able to apply that structure to novel clause constructs.

Abstractions are regularities that best describe the structure of the data. It might be possible to devise a self-organizing process that is sensitive to the internal structure of the training examples. The network would learn the processing task, and at the same time develop a layout of rules, schemas, and other abstract structures that best describe the data. Further input would then be interpreted and represented in terms of this layout (i.e., in terms of the internal structure of the input). Such a capacity would be a major step toward extending subsymbolic AI beyond the limitations of current models.

11 Conclusion

Above all, DISCERN serves to show that subsymbolic high-level AI is feasible. DISCERN constitutes a first implementation of the integrated connectionist approach, demonstrating that it is possible to build complete models from independently designed connectionist components. The scale-up properties of the approach seem quite good. Hierarchical modular structure with sequential communication efficiently reduces the complexity of the high-level task. The modules filter out each other's errors, and system performance is stable. With meaningful intermediate representations, more modules could be added to the existing system. The system develops its own representations for intercommunication between modules, and these representations are optimized for the overall task.

DISCERN also grounds several high-level phenomena in subsymbolic (statistical) processes. Learning word meanings, script processing, and episodic memory organization are based on self-organization and gradient-descent in error. Script-based inferences, expectations, and defaults automatically result from generalization and graceful degradation in distributed networks. At the level of maps, pathways, and networks, DISCERN is also a plausible physical model. Several types of performance errors in role binding, episodic memory, and lexical access are explained by the physical organization of the system, and many aphasic impairments can be modeled by local damage to the model.

The two main liabilities of DISCERN are that processing is based on a series of reflex responses, and that many of the internal knowledge structures are fixed and specified in advance. Developing methods for connectionist control of high-level behavior and for learning more of the necessary knowledge structures automatically are the two main areas for future research. Progress in these areas should lead eventually to substantially more powerful subsymbolic AI systems.

Acknowledgement

This research was supported in part by an Initial Teaching Alphabet (ITA) Foundation grant and by fellowships from the Academy of Finland, the Emil Aaltonen Foundation, the Foundation for the Advancement of Technology, and the Alfred Kordelin Foundation (Finland) when the author was at UCLA. The simulations were carried out in part on the CRAY Y-MP8/864 at the San Diego Supercomputer Center.

References

Alvarado, S., Dyer, M. G., and Flowers, M. 1990. Argument representation for editorial text. *Knowledge-Based Systems* **3**, 87–107.

Baddeley, A. D. 1976. *The psychology of memory.* Basic Books, New York.

Caramazza, A. 1988. Some aspects of language processing revealed through the analysis of acquired aphasia: The lexical system. *Annual Review of Neuroscience* **11**, 395–421.

Coltheart, M., Patterson, K., and Marshall, J. C. (eds.) 1988. *Deep dyslexia*, second edition. Routledge and Kegan Paul, London; Boston.

Cullingford, R. E. 1978. *Script application: Computer understanding of newspaper stories.* Ph.D thesis, Department of Computer Science, Yale University, New Haven, Connecticut. (Technical Report 116).

Dolan, C. P. 1989. *Tensor manipulation networks: Connectionist and symbolic approaches to comprehension, learning and planning.* Ph.D thesis, Computer Science Department, University of California, Los Angeles. (Technical Report UCLA-AI-89-06).

Dyer, M. G. 1983. *In-depth understanding: A computer model of integrated processing for narrative comprehension.* MIT Press, Cambridge, Massachusetts.

Dyer, M. G. 1991. Symbolic neuroengineering for natural language processing: A multilevel research approach. In: Barnden, J. A., and Pollack, J. B. (eds.), *High-level connectionist models*, Ablex, Norwood, New Jersey, pp. 32–86.

Dyer, M. G., Cullingford, R. E., and Alvarado, S. 1987. Scripts. In: Shapiro, S. C. (ed.), *Encyclopedia of artificial intelligence*, Wiley, New York, pp. 980–994.

Elman, J. L. 1990. Finding structure in time. *Cognitive Science* **14**, 179–211.

Elman, J. L. 1991. Incremental learning, or The importance of starting small. In: *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, Erlbaum, Hillsdale, New Jersey, pp. 443–448.

Feldman, J. A. 1989. Neural representation of conceptual knowledge. In: Nadel, L., Cooper, L. A., Culicover, P., and Harnish, R. M. (eds.), *Neural connections, mental computation*, MIT Press, Cambridge, Massachusetts, pp. 68–103.

Fodor, J. A. 1983. *Modularity of mind: An essay on faculty psychology.* MIT Press, Cambridge, Massachusetts.

Harris, C. L., and Elman, J. L. 1989. Representing variable information with simple recurrent networks. In: *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, Erlbaum, Hillsdale, New Jersey, pp. 635–642.

Hinton, G. E. 1990. Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence* **46**, 47–75.

Kohonen, T. 1989. *Self-organization and associative memory*, third edition. Springer, Berlin; Heidelberg; New York.

Kohonen, T. 1990. The self-organizing map. *Proceedings of the IEEE* **78**, 1464–1480.

Kolodner, J. L. 1984. *Retrieval and organizational strategies in conceptual memory: A computer model.* Erlbaum, Hillsdale, New Jersey.

LAIRD, J. E., NEWELL, A., and ROSENBLOOM, P. S. 1987. SOAR: An architecture for general intelligence. *Artificial Intelligence* **33**, 1–64.

LEBOWITZ, M. 1980. *Generalization and memory in an integrated understanding system.* Ph.D thesis, Department of Computer Science, Yale University, New Haven, Connecticut. Research Report 186.

MCCARTHY, R. A., and WARRINGTON, E. K. 1990. *Cognitive neuropsychology: A clinical introduction.* Academic Press, New York.

MCCLELLAND, J. L., RUMELHART, D. E., and THE PDP RESEARCH GROUP 1986. *Parallel distributed processing: Explorations in the microstructure of cognition, Vol. 2: Psychological and biological models.* MIT Press, Cambridge, Massachusetts.

MIIKKULAINEN, R. 1990a. A distributed feature map model of the lexicon. In: *Proceedings of the 12th Annual Conference of the Cognitive Science Society*, Erlbaum, Hillsdale, New Jersey, pp. 447–454.

MIIKKULAINEN, R. 1990b. A PDP architecture for processing sentences with relative clauses. In: KARLGREN, H. (ed.), *Proceedings of the 13th International Conference on Computational Linguistics*, Yliopistopaino, Helsinki, Finland, pp. 201–206.

MIIKKULAINEN, R. 1990c. Script recognition with hierarchical feature maps. *Connection Science* **2**, 83–101.

MIIKKULAINEN, R. 1992. Trace feature map: A model of episodic associative memory. *Biological Cybernetics* **67**, 273–282.

MIIKKULAINEN, R. 1993. *Subsymbolic natural language processing: An integrated model of scripts, lexicon, and memory.* MIT Press, Cambridge, Massachusetts.

MIIKKULAINEN, R., and DYER, M. G. 1989. A modular neural network architecture for sequential paraphrasing of script-based stories. In: *Proceedings of the International Joint Conference on Neural Networks* (Washington, DC), Vol. II, IEEE, Piscataway, New Jersey, pp. 49–56.

MIIKKULAINEN, R., and DYER, M. G. 1991. Natural language processing with modular neural networks and distributed lexicon. *Cognitive Science* **15**, 343–399.

MINSKY, M. 1985. *Society of mind.* Simon & Schuster, New York.

NEWELL, A. 1980. Physical symbol systems. *Cognitive Science* **4**, 135–183.

NEWELL, A. 1991. *Unified theories of cognition.* Harvard University Press, Cambridge, Massachusetts.

POLLACK, J. B. 1987. Cascaded back-propagation on dynamic connectionist networks. In: *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, Erlbaum, Hillsdale, New Jersey, pp. 391–404.

POLLACK, J. B. 1990. Recursive distributed representations. *Artificial Intelligence* **46**, 77–105.

POSTMAN, L. 1971. Transfer, interference and forgetting. In: KLING, J. W., and RIGGS, L. A. (eds.), *Woodworth and Schlosberg's experimental psychology*, third edition, Holt, Rinehart and Winston, New York, pp. 1019–1132.

REEVES, J. F. 1991. *Computational morality: A process model of belief conflict and resolution for story understanding.* Ph.D thesis, Computer Science Department, University of California, Los Angeles. (Technical Report UCLA-AI-91-05).

Rumelhart, D. E., Hinton, G. E., and McClelland, J. L. 1986a. A general framework for parallel distributed processing. In: Rumelhart, D. E., and McClelland, J. L. (eds.), *Parallel distributed processing: Explorations in the microstructure of cognition, Vol. 1: Foundations*, MIT Press, Cambridge, Massachusetts, pp. 45–76.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. 1986b. Learning internal representations by error propagation. In: Rumelhart, D. E., and McClelland, J. L. (eds.), *Parallel distributed processing: Explorations in the microstructure of cognition, Vol. 1: Foundations*, MIT Press, Cambridge, Massachusetts, pp. 318–362.

Rumelhart, D. E., and McClelland, J. L. 1986. On learning past tenses of English verbs. In: Rumelhart, D. E., and McClelland, J. L. (eds.), *Parallel distributed processing: explorations in the microstructure of cognition, Vol. 2: Psychological and biological models*, MIT Press, Cambridge, Massachusetts, pp. 216–271.

Rumelhart, D. E., McClelland, J. L., and the PDP Research Group 1986c. *Parallel distributed processing: Explorations in the microstructure of cognition, Vol. 1: Foundations.* MIT Press, Cambridge, Massachusetts.

St. John, M. F. 1992. The story gestalt: A model of knowledge-intensive processes in text comprehension. *Cognitive Science* **16**, 271–306.

St. John, M. F., and McClelland, J. L. 1990. Learning and applying contextual constraints in sentence comprehension. *Artificial Intelligence* **46**, 217–258.

Schank, R. C., and Abelson, R. P. 1977. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures.* Erlbaum, Hillsdale, New Jersey.

Sejnowski, T. J., and Rosenberg, C. R. 1987. Parallel networks that learn to pronounce English text. *Complex Systems* **1**, 145–168.

Shallice, T. 1988. *From neuropsychology to mental structure.* Cambridge University Press, Cambridge, United Kingdom.

Simon, H. A. 1969. *The sciences of the artificial.* MIT Press, Cambridge, Massachusetts.

Smolensky, P. 1988. On the proper treatment of connectionism. *Behavioral and Brain Sciences* **11**, 1–74.

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

Touretzky, D. S. 1991. Connectionism and compositional semantics. In: Barnden, J. A., and Pollack, J. B. (eds.), *High-level connectionist models*, Ablex, Norwood, New Jersey, pp. 17–31.