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

SCRIPT RECOGNITION WITH HIERARCHICAL FEATURE MAPS

Risto Miikkulainen

October 1989
CSD-890058

SCRIPT RECOGNITION WITH HIERARCHICAL FEATURE MAPS *

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

Abstract

The hierarchical feature map system recognizes an input story as an instance of a particular script by classifying it at three levels: scripts, tracks and role bindings. The recognition taxonomy, i.e. the breakdown of each script into the relevant tracks and roles, is extracted automatically and independently for each script from story examples in an unsupervised learning process. The process resembles human learning in that the differentiation of the most frequently encountered scripts become gradually the most detailed. The resulting structure is a hierachical pyramid of feature maps. The number of input lines and the self-organization time required are considerably reduced compared to ordinary single-level feature mapping. The system is capable of recognizing incomplete stories and recovering the missing events.

1 Introduction

Script theory postulates that people organize the knowledge of everyday routines in the form of scripts [Schank and Abelson, 1977], [Bower *et al.*, 1979]. Scripts are schemas of often-encountered, stereotypical sequences of events. Common knowledge of this kind makes it possible to efficiently perform social tasks such as visiting a restaurant, visiting a doctor, shopping at a supermarket, traveling by airplane, attending a meeting, etc. People have hundreds, even thousands of scripts at their disposal. Each script divides further into different variations, or tracks. For example, there is a fancy-restaurant track, a fast-food track and a coffee-shop track for the restaurant script.

In machine understanding of stories based on scriptal knowledge the script is represented as a causal chain of events with a number of open roles [Schank and Abelson, 1977], [Cullingford, 1978]. Applying this knowledge to a story requires identifying the relevant script and matching its roles with the story. Entering, seating, ordering, eating, paying and leaving form a causal chain for the restaurant script. The roles in this script are customer, restaurant, food, etc. Once the script has been identified and its roles have been filled, the sentences of the story are matched against the events in the script. Events which are not mentioned in the story but are part of the causal chain can be inferred.

*This research was supported in part by an ITA Foundation grant and by grants from the Academy of Finland and the Finnish Science Academy. The simulations were carried out on equipment donated to UCLA by Hewlett Packard. Special thanks go to Michael Dyer for valuable comments on an earlier draft of this paper.

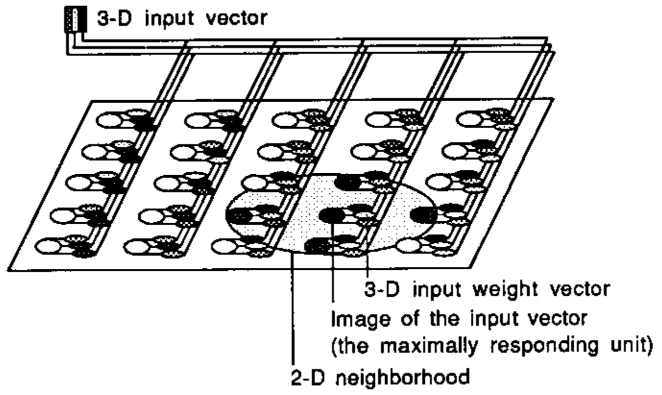

Figure 1: **A self-organizing feature map network.** A mapping is formed from a 3-dimensional input space onto a 2-dimensional network. The values of the input components, weights and the unit output are shown by gray-scale coding.

A topological feature map [Kohonen, 1984] is a large adaptive system which consists of a number of processing units in a laminar organization (figure 1). The map is formed in an unsupervised learning process. The input data is first quantified along a number of features, forming an input space of N-dimensional vectors. Input items are randomly drawn from the input distribution and presented to the network one at a time. All units receive the same input and produce one output, proportional to the similarity of the input vector and the unit's parameter vector (which is also called the input weight vector of the unit). The unit with the maximum response is taken as the image of the input vector on the map. The parameter vector of this unit and each unit in its neighborhood are changed towards the input vector, so that these units will produce an even stronger response to the same input in the future. The parallelism of neighboring vectors is thus increased at each presentation, a process which results in a global order.

The processing units of the resulting network are sensitive to specific items of the input space (figure 2). Topological relations are retained: two input items which are close in the input space are mapped onto units close in the map. The distribution of the parameter vectors approximates that of the input vectors. This means that the most common areas of the input space are represented to

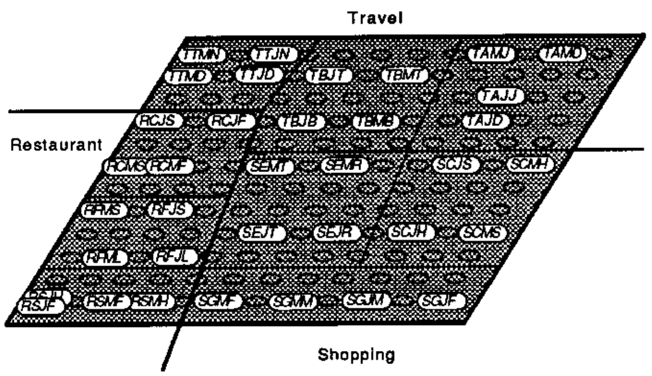

Figure 2: **A single level map of script-based stories.** The first letter of a label indicates the script (`R=restaurant`, `T=travel`, `S=shopping`), the second letter stands for the track (`F=fancy-restaurant`, `C=coffee-shop-restaurant`, `S=fast-food-restaurant` etc), third letter for the main actor (`J=John`, `M=Mary`), and the last for the two different bindings of a track-specific role (see description of the data in section 3.1). Labels indicate the image of each story.

a greater detail, i.e. more units are allocated to represent these inputs. The dimensionality of the map is determined by the definition of the neighborhood, i.e. whether the units are laid out in a line (1-D) or on a plane (2-D) etc. If dimensionality is reduced in the mapping, the dimensions of the map do not necessarily stand for any recognizable features of the input space. The dimensions develop automatically to facilitate the best discrimination between the input items.

Feature maps have several potentially useful properties for script recognition. The classification performed by a feature map is based on a large number of parameters (the input weights), making it very resistant to noise. This suggests that incomplete or somewhat unusual event sequences can be correctly recognized. The map is continuous, and can represent items which are between established categories. In other words, scripts can have soft boundaries. The differences of the most frequent input items are magnified in the mapping, i.e. the variations of the most common event sequences are more finely discriminated. Finally, the process requires no supervision and makes no assumptions of the content of the input stories. The properties of the stories which best distinguish between scripts and their variations are determined automatically, and may be very different for different kinds of scripts.

On the other hand, scripts are hierarchical representations. Each *script class* consists of a number of *tracks*, and each track can be instantiated with different *role bindings.* Hierarchical feature maps can be used to make the hierarchical taxonomy explicit. The hierarchical architecture also cuts down the number of redundant system parameters (input weights) and speeds up the computationally intensive training by effectively dividing the task of forming a script taxonomy into subgoals.

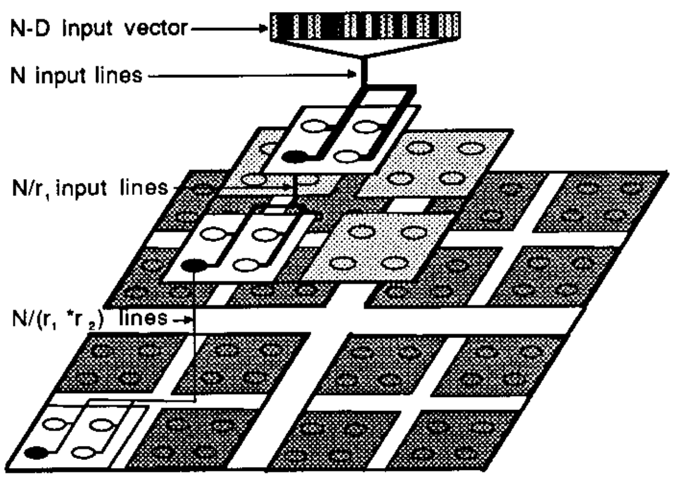

Figure 3: **A hierarchical feature map system.**

2 Hierarchical feature maps

The self-organizing process [Kohonen, 1982b], [Kohonen, 1984] produces a representation of the input space where the input data is spread out spatially on e.g. a 2-dimensional sheet. The map represents most directly input spaces which are continuous and unstructured. If the data is hierarchical, the map reflects this through topological order. For instance, the map of taxonomical data essentially displays the minimal spanning tree of the data items, curved to fill in the whole area of the map [Kohonen, 1982a]. Or, if we form a mapping of script-based stories, different variations of the scripts are mapped near each other (figure 2). Knowing what the hierarchical taxonomy of the data is, it is easy to see that the spatial layout of the map reflects the taxonomy. However, it is hard to *extract* the taxonomy from the map alone.

With hierarchical feature maps the taxonomy can be made explicit. The highest level of the hierarchy is first laid out on a single map by the ordinary self-organizing process (figure 3). A small map size forces the process to make only a gross, high-level classification. The units in this map stand for the highest level categories.

For each unit in the top-level map, there is another feature map beneath it, and similarly for each unit in the second-level map. The system of maps thus forms a pyramid-like structure. A lower map in the structure receives as its input only those input items which belong to the category represented by its parent unit. In other words, the unit which "wins" the input item passes this item down to its submap. The lower map forms a subcategorization of these input items, mapping the differences within the category. The complete hierarchical classification of an input item is indicated by the maximally responding units at each level of the hierarchy.

Input representations in a cognitive system often have some discrete structure, such as e.g. role specific assemblies. If the data is hierarchical, the items belonging to the same category have a number of componets in com-

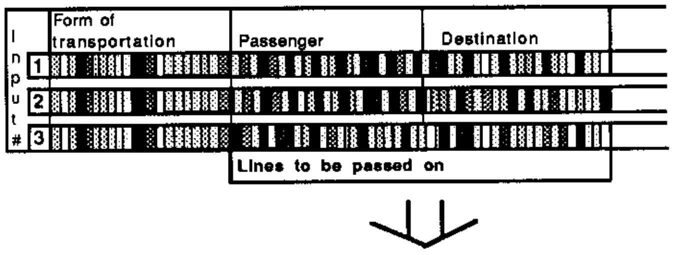

Figure 4: **Compression of the input lines.**

mon. These components can be dropped from the input to the next-level map. The higher-level map acts as a filter, choosing the relevant items for each submap and compressing the representation of these items to the most relevant components before passing them on for a more detailed mapping.

Each unit has to determine independently which of its input lines are the most relevant in distinguishing between items of the category. It has to find the *lines with the most variation* between the items it wins. These lines are different for different units. For example, a unit which stands for the train track of the travel script, wins only items which have **train** as the form of transportation (figure 4). The values on the input lines which code the form of transportation do not change from one input to another. On the other hand, the passenger and destination may be different in different stories, and lines representing these roles have a lot of variation. These lines alone are sufficient to determine the role bindings of the track, and they are therefore passed on to the next map.

In self-organizing a hierarchical feature map system, it is most efficient to start the process at the highest level, and include a lower level only when the higher level has become ordered. This way the set of inputs to the lower level stays constant, and no work is wasted on inputs which will eventually be classified into another category. Each unit determines its own compressed set of input lines during the last epoch before the lower level is included. In our implementation, the lines of each unit were simply ordered with respect to the variance over the inputs won during that epoch, and a fixed fraction r_i was chosen at each map level i. This means that usually only a few lines of the representation for e.g. passenger are passed on to the next level (the ones with the highest variance), not the whole representation. Instead of a fixed fraction, a variance threshold could also be used, in which case the extent of the compression would adapt to the input data.

Self-organizing a hierarchy of small maps instead of a single large one means dividing the classification task into subgoals, which is an efficient way to reduce complexity. A story representation is first classified as an instance of a script, then as a track within the script etc. Within the script (or track), *only a subset of the input representation needs to be examined to refine the classification further.* The relative differences between the items are greater, making the self-organization easier.

In a single-level map all units need to receive the complete representations of all input items (figure 1). Modifying the weights is costly, because the initial neighborhood must cover almost the entire map [Miikkulainen, 1987]. In a hierarchical map system, the maps at lower levels receive only small subsets of the original input lines, and the neighborhoods are always small because the maps are small at all levels (figure 3). This reduces the training time considerably.

The hierarchical feature map technique actually gives us a continuum of classification systems. At one extreme there is the single-level feature map, which produces a topological layout of the input space. At the other extreme there is a hierarchy of two-unit maps, which forms a statistically balanced divisive clustering tree of the input items: the inputs are first divided into two equally large classes, each class is further divided into two parts etc. If enough is known about the structure of the input space, *a hierarchical feature map architecture can be chosen, which self-organizes to directly represent the hierarchical semantics of the input space.*

3 Simulations

3.1 Input data

The script recognition system was trained with simple versions of the restaurant, shopping and travel scripts, with a total of nine tracks, listed below. Two fillers (`John, Mary`) were used to fill the `PERSON` role in all tracks. In addition, the restaurant tracks had two fillers each for the `FOOD` roles, shopping tracks had two fillers each for the `ITEM` roles and travel tracks two fillers each for the `DESTINATION` roles. The fillers are listed next to the track name below. The actual input stories were generated by replacing the role names with different combinations of fillers. The system had to extract the hierarchical taxonomy from the repeated presentation of the resulting 36 stories.

Fancy-restaurant track : **lobster,steak**
```
PERSON went to a fancy-restaurant.
Waiter seated PERSON at a table.
PERSON asked the waiter for FANCY-FOOD.
PERSON waited for the FANCY-FOOD.
PERSON ate the FANCY-FOOD.
PERSON paid the waiter.
```

Coffee-shop-restaurant track : **spaghetti,fish**
```
PERSON went to a coffee-shop-restaurant.
PERSON sat down at a table.
PERSON asked the waiter for COFFEE-FOOD.
The waiter brought PERSON the COFFEE-FOOD.
PERSON ate the COFFEE-FOOD.
PERSON paid at the cashier.
```

Fast-food-restaurant track : **hamburger,fries**
```
PERSON went to a fast-food-restaurant.
PERSON waited in line for the cashier.
PERSON asked the cashier for FAST-FOOD.
PERSON paid the cashier.
PERSON ate the FAST-FOOD.
PERSON threw away the trash.
```

Clothing-shopping track : **shoes,hat**

```
PERSON went to a clothing-store.
PERSON looked for a CLOTHING-ITEM.
PERSON compared CLOTHING-ITEM prices.
PERSON tried a number of CLOTHING-ITEMs.
PERSON chose the best CLOTHING-ITEM.
PERSON paid at the cashier.
```

Electrical-shopping track : **TV,radio**

```
PERSON went to an electrical-store.
PERSON asked the cashier for an ELECTRICAL-ITEM.
PERSON asked questions about the ELECTRICAL-ITEM.
PERSON compared ELECTRICAL-ITEM prices.
PERSON chose the best ELECTRICAL-ITEM.
PERSON paid the cashier.
```

Grocery-shopping track : **fruit,meat**

```
PERSON went to a grocery-store.
PERSON chose a shopping-cart.
PERSON chose a number of GROCERY-ITEMs.
PERSON compared GROCERY-ITEM prices.
PERSON waited in line for the cashier.
PERSON paid the cashier.
```

Airplane-travel track : **JFK,DFW**

```
PERSON checked-in at the airport.
PERSON waited for the boarding.
PERSON got-on the plane.
The plane took-off from the airport.
The plane arrived at the PLANE-DESTINATION.
PERSON got-off the plane.
```

Train-travel track : **NYC,DC**

```
PERSON bought a ticket at the railway-station.
PERSON waited for the train.
PERSON got-on the train.
The conductor punched the ticket.
The train arrived at the TRAIN-DESTINATION.
PERSON got-off the train.
```

Bus-travel track : **town,beach**

```
PERSON went to the bus-stop.
PERSON waited for the bus.
PERSON got-on the bus.
PERSON paid the driver.
The bus arrived at the BUS-DESTINATION.
PERSON got-off the bus.
```

Each story in the training set is represented by the concatenation of the case-role representations of its sentences (figure 5). A case-role representation consists of assemblies of components, with distributed patterns in the assemblies indicating the fillers of the surface-semantic roles of the sentence. The input for the feature map system thus consists of the *surface semantics of the sentences of the story.*

The case-role representations were formed by a separate front end, the backpropagation-based word-parser network from the DISPAR system [Miikkulainen and Dyer, 1989]. This network reads in the input sentences sequentially word by word, developes distributed representations for the words, and produces a case-role representation as its output. The resulting word representations reflect the regularities in the use of the words [Miikkulainen and Dyer, 1988].

The two different instances of the filler words (John, Mary etc.) were formed from a common stem. The first component of the corresponding role word (PERSON etc.), developed by the word-parser, was replaced by 0.0 for one and 1.0 for the other. The purpose was to give these words approximately the same surface semantics (which is coded in the representation) while keeping the representations unique [Miikkulainen and Dyer, 1989].

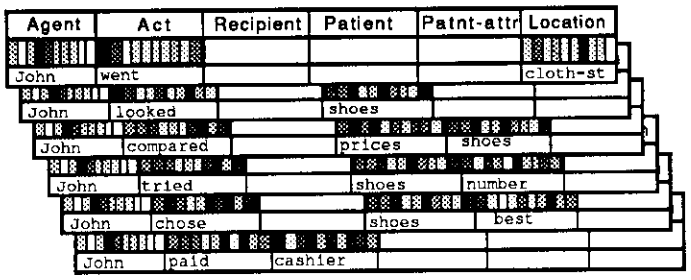

Figure 5: **Story representation.** A story is represented by a 360-dimensional vector, which consists of 6 sentence case role representations, each $6 \times 10 = 60$ -dimensional. This particular story is an instance of Clothing-shopping track, with PERSON = John, CLOTHING-ITEM = shoes.

The word representation consisted of 10 units and the case-role representation of 6 assemblies (**agent**, **act**, **recipient**, **patient**, **patient-attribute** and **location**) (figure 5). Each story contained 6 events, making the input vectors $6 \times 6 \times 10 = 360$ -dimensional.

3.2 The resulting map hierarchy

A three-level pyramid of feature maps was used to form the recognition taxonomy (figure 6). The highest level consisted of a 2×2 map, the next level of four 2×2 maps, and the lowest level of sixteen 3×3 maps. Of the 360 input lines to the first level, each unit independently determined the 54 lines with the highest variance (15%), and passed them on to the second level. Of these 54 lines, each second-level unit passed on 10 (20%) to the lowest level.

The top-level developed into a map of the different script classes. In a 2×2 map, the restaurant, shopping and travel stories were mapped onto different corners. The second level distinguishes between the different tracks of each script. The compressed input vectors to this level consist mostly of the differences in the order of events, and of the most unique case-role assignments. At the bottom level, the different role-binding combinations are separated. The bottom-level feature maps display topological ordering: the PERSON role is differentiated along one axis, while the other axis is used to separate the bindings of the other open role of the track. Note that *these dimensions were discovered by the mapping itself, and they are different for different scripts.* The compressed input vectors to the role-binding level consist mainly of the first components of the filler words (either 0.0 or 1.0), which provide for the best differentiation between fillers.

The outcome of the self-organizing process is somewhat sensitive to the system parameters. To get a clean three-level classification into scripts, tracks and role bindings the maps must have approximately the right size and each input must be approximately as frequent in the input data.

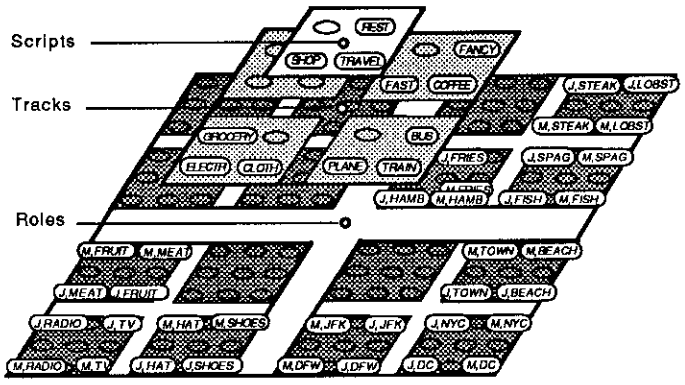

Figure 6: **The resulting map hierarchy.**

If, for instance, the highest-level map is too large, or one of the script classes is far more frequent than the others, the different tracks may get separated already at the highest-level map.

However, even if the parameter settings are not ideal, the script recognition works the same. *The system uses whatever architecture is given to establish the best classification of the input data.* The configuration parameters may need to be experimentally adjusted to achieve the desired semantics for the levels of the hierarchy, but the function of the recognition system is fairly robust.

The script taxonomy can be used to recognize a story as an instance of a particular script, track and role binding. The images of the input story at each level of the hierarchy give this classification. The system can correctly classify a story even if a number of events are missing (left blank in the representation). This is a consequence of the general redundancy of the system: the incomplete input is still closest to the correct script, and is classified as such.

3.3 Reproducing stories

It is possible to reproduce the story from its representation in the feature maps. In the self-organizing process the input weight vectors have become approximations of the input vectors [Kohonen, 1984]. The weight vector at each unit is thus a representation of the average story in the category the unit stands for. The weights at different levels of the hierarchy represent *different levels of abstraction of the story.*

The weights of the image unit at the top level represent the skeleton of the script, where the different tracks and role bindings have been averaged out. The weights at the input lines which were passed on to the next level are averages over all input items. When these weights are replaced with the corresponding weights of the image unit at the next level, the track becomes established. At this point the story is otherwise complete but the role bindings are still unspecified. For example, in the place of the actor name (`John` or `Mary`) there is the representation of the general actor (`PERSON`), which is an average of the two. The bindings are finally established when the lowest-level weights are combined with the representation.

This technique can be used to produce a full paraphrase of an incomplete story, i.e. *to fill in the missing events.* The story is first classified as an instance of one of the stories known to the system. The weights for this classification are then read out, forming the full paraphrase. For example, the story

```
John bought a ticket at the railway-station.
John got on the train.  The train arrived at
NYC.
```

is classified as an instance of the travel script, train track, with `PERSON` = `John`, `TRAIN-DESTINATION` = `NYC`. Reading out the weights gives the full instantiation of the script:

```
John bought a ticket at the railway-station.
John waited for the train.  John got on the
train.  The conductor punched the ticked.  The
train arrived at NYC. John got off the train.
```

Note that the maps contain several units which are not images of any particular input item. These extra units are necessary for the self-organizing process to develop a meaningful topological organization of the map. The weights on these units end up representing stories which are between the actual inputs that the system has seen. In some cases they may be meaningful abstractions or generalizations, extracting the similarities of the items around them in the map. Such a unit between two script units, for example, can represent uncertainty about the script class the incomplete input story belongs to. Certain events can be safely filled in, and they are specified in the weights of this unit. Others are uncertain, and the corresponding weights of the unit do not represent anything in particular. In most cases though, the uncommitted units do not represent anything useful. A combination of e.g. the airplane-travel script and the cake-baking script is not anything that occurs in real life.

4 Discussion

In an earlier paper we have described a connectionist script reader system DISPAR (DIStributed PARaphraser, [Miikkulainen and Dyer, 1989]), which is based on backpropagation learning of a slot-filler representation of the story. An external supervisor decides on a fixed set of slots (such as customer, food, who-to-pay etc.), and tells the system what the correct fillers should be for each story. A disadvantage of this approach is that the number of different scripts the system can represent is limited. All the variations of all stories 1) must be known in advance and 2) must be represented in the fixed number of slots.

With hierarchical feature maps, it is possible to recognize and represent a large number of different scripts, and the system determines automatically what is relevant for the taxonomy. The stories are classified into script classes, tracks, and different role bindings based on the statistical similarities in the input representations. The breakdown of each script is independent of the others. Tracks and roles are developed which facilitate the best discrimination between the stories of the script, and these are usually different for different scripts.

On the other hand, the classification is purely statistical. There is no way to include semantic information to guide the process, as in the supervised learning approach. All relevant information must be included in the input vectors. Also, since the relevant roles are determined separately for each script, the common roles end up being represented multiple times. Each script in our data has a role for the main actor, but this role has to be mapped separately on every map at the lowest level. Half of each role map is dedicated to stories with `John` as the main actor, the other half with `Mary` (figure 6).

The self-organizing process resembles a human learner in building the script taxonomy. The process begins by establishing a gross ordering of the input data, dividing the input into a few large categories [Miikkulainen, 1987], i.e. the most prominent and regular event sequences are recognized as the first rudimentary scripts. For example, the restaurant and the shopping stories are first grouped together, separate from the travel stories. These categories become gradually more refined, as more attention is paid on the details. *The more frequently a certain kind of input occurs in the input data, the more of its details become significant.* The restaurant and shopping stories together are twice as numerous as the travel stories, and what used to be different variations of the shopping-restaurant script eventually become scripts on their own right.

The role binding at the lowest level of the mapping takes place in a somewhat limited sense. Once the story has been recognized, the bindings are consistent and plausible (in the weight representation), making high-level inferencing possible [Dyer, 1988]. However, all the role-filler combinations must be mapped out in advance. While this is not a problem for certain roles (representing a closed set of variations of the script), roles like `PERSON` should be open for any person name. The current system cannot bind arbitrary representations into roles, it *must have seen the bindings before in the training data.*

The hierarchical feature mapping technique makes most sense when the input data is strongly hierarchical, and the architecture of the system matches this hierarchy. In the extreme case of a uniform distribution of independent continuous variables there is nothing to be gained by using hierarchies. If enough is known about the structure of the input data, it should be possible to choose an architecture from the continuum between the single-level map and binary clustering, which efficiently reflects the structure of the input space.

If the input data has the enough structure, the reduction of the required input connections and the speed-up of the self-organization can be quite dramatic. In the script recognition system, there were $4 \times 360 = 1,440$ connections to the highest level, $16 \times 54 = 864$ to the second, and $144 \times 10 = 1,440$ to the bottom level, a total of 3,744 input connections. A comparable single-level mapping (figure 2) with 144 units for the same data required 51,840 input connections. Self-organizing the single-level mapping took about 9.5 hours on an HP 900/350 workstation, while the hierarchical mapping was complete in three minutes, a difference of two orders of magnitude.

5 Future work

It would be desirable to develop a more general representation for the input stories. A large number of units in the current representation are always blank, and the representation is extremely sensitive to the order and number of events. Instead of simple concatenations of a fixed number of event representations, the system should be able to read time sequences of event representations of undetermined length. One possible approach is to use a feature map of events as a front end. The events would be presented to this map sequentially, and would form a trajectory on the map. This trajectory (i.e. the composite response pattern on the front end map) would form the input to the script recognition system.

It might be possible to develop a mechanism for automatically adjusting the sizes of the maps or the depth of the hierarchy according to the input data. Two different inputs should always be mapped onto different units at the lowest level. This constraint could be used to self-organize the system architecture, in addition to the current self-organization of the individual feature maps.

The primary function of the system is to build a taxonomy which allows recognizing a story as an instance of a particular script, track and role binding. Higher-level systems such as a paraphraser or a question answerer could be built based on this representation. It might be possible to extract the information the self-organized system has found about the relevant tracks and roles, and use this to automatically generate training data for supervised-learning based higher-level systems, thus combining the advantages of both approaches.

6 Conclusion

The hierarchical feature map system recognizes an input story as an instance of a particular script by classifying it at three levels: scripts, tracks and role bindings. The recognition taxonomy, i.e. the breakdown of each script into the relevant tracks and roles, is extracted automatically and independently for each script from story examples in an unsupervised learning process. The process resembles human learning in that the differentiation of the most frequently encountered scripts become gradually the most detailed. The resulting structure is a hierachical pyramid of feature maps. The number of input lines and the self-organization time required are considerably reduced compared to ordinary single-level feature mapping. The system is capable of recognizing incomplete stories and recovering the missing events.

References

[Bower *et al.*, 1979] Gordon H. Bower, John B. Black, and Terrence J. Turner. Scripts in memory for text. *Cognitive Psychology*, (11):177–220, 1979.

[Cullingford, 1978] R. E. Cullingford. *Script Application: Computer Understanding of Newspaper Stories.* Technical Report 116, Yale University, Department of Computer Science, 1978. Ph.D. dissertation.

[Dyer, 1988] Michael G. Dyer. Symbolic NeuroEngineering for natural language processing: a multilevel research approach. In J. Barnden and Jordan Pollack, editors, *Advances in Connectionist and Neural Computation Theory*, Ablex Publ., 1988. (in press).

[Kohonen, 1982a] 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, 1982b] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, (43):59–69, 1982.

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

[Miikkulainen, 1987] Risto Miikkulainen. *Self-Organizing Process Based on Lateral Inhibition and Weight Redistribution.* Technical Report UCLA-AI-87-16, Artificial Intelligence Laboratory, Computer Science Department, University of California, Los Angeles, 1987.

[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.

[Miikkulainen and Dyer, 1989] Risto Miikkulainen and Michael G. Dyer. A modular neural network architecture for sequential paraphrasing of script-based stories. In *Proceedings of the International Joint Conference on Neural Networks*, IEEE, 1989.

[Schank and Abelson, 1977] Roger Schank and Robert Abelson. *Scripts, Plans, Goals, and Understanding - An Inquiry into Human Knowledge Structures. The Artificial Intelligence Series*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1977.