A Symbolic/Connectionist Script Applier Mechanism

Geunbae Lee, Margot Flowers, Michael G. Dyer
Artificial Intelligence Laboratory
Computer Science Department, UCLA

Abstract

We constructed a Modular Connectionist Architecture which consists of many different types of 3 layer feed-forward PDP network modules (auto-associative recurrent, hetero-associative recurrent, and hetero-associative) in order to do script-based story understanding. Our system, called DYNASTY (DYNAmic script-based STory understanding sYstem) has the following 3 major functions: *(1) DYNASTY can learn distributed representations of concepts and events in everyday scriptal experiences, (2) DYNASTY can do script-based causal chain completion inferences according to the acquired sequential knowledge, and (3) DYNASTY performs script role association and retrieval while performing script application.* Our purpose in constructing this system is to show that the learned internal representations, using simple encoder-type networks, can be used in higher-level modules to develop connectionist architectures for fairly complex cognitive tasks, such as script processing. Unlike other neurally inspired script processing models, DYNASTY can learn its *own similarity-based distributed representations from input script data* using ARPDP (Auto-associative Recurrent PDP) architectures. Moreover DYNASTY's role association network handles both script roles and fillers as *full-fledged concepts*, so that it can learn the generalized associative knowledge between several script roles and fillers.

1 Background and Issues

A script is a knowledge structure of stereotypic action sequences [27]. According to psychological experiments[1], people use scripts to understand and remember narrative texts. But proposed symbolic AI models of script processing (e.g. SAM [2,26]) have many unresolved problems: (1) They are too rigidly defined, so they can not handle script deviations properly. (2) It is difficult to invoke the right script for the input story fragments. Proposed script headers[2] are unnatural and fragile.

A number of neurally inspired connectionist script processing models have been proposed to overcome weaknesses in the symbolic models [3,4,5], but none of them has the semantics needed for representing constituency of concepts and events. Dolan and Dyer [6] are the first to consider micro-feature based underlying representations in connectionist script processing to make their representations have similarity properties: similar concepts have similar representations. But as noted in [7] micro-features are unnatural and akward.

This paper proposes a modular distributed connectionist architecture called DYNASTY (DYNAmic script-based STory understanding sYstem) based on *automatically learned distributed semantic representations.* DYNASTY takes simple coherent groups of sentences as input, e.g.:

> John went to Sizzler. John ate steak and shrimp. John left a tip.

and produces causally completed groups of sentences as output:[1]

> John went to Sizzler. Waiter seated John. John looked at the menu. John ordered steak and shrimp. John ate steak and shrimp. John paid the bill. John left a tip. John left Sizzler for home.

There are three major tasks that DYNASTY must solve in order to handle this example: (1) DYNASTY must learn distributed semantic representations (DSR) for both concepts and events automatically from its input script data. (2) DYNASTY must learn sequential knowledge to do causal-chain completion inference. (3) DYNASTY must learn associations between script roles and their fillers for later retrieval of role bindings.

[1] The events in this output were mentioned by 55 - 75% of the human subjects in a psychological experiment [1].

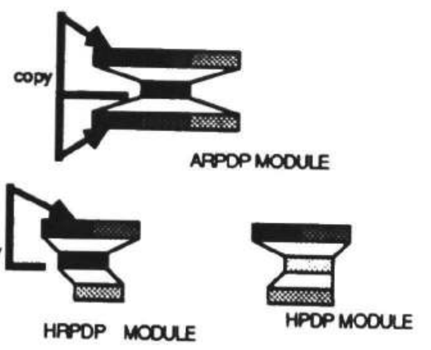

Figure 1: ARPDP, HRPDP, and HPDP modules.

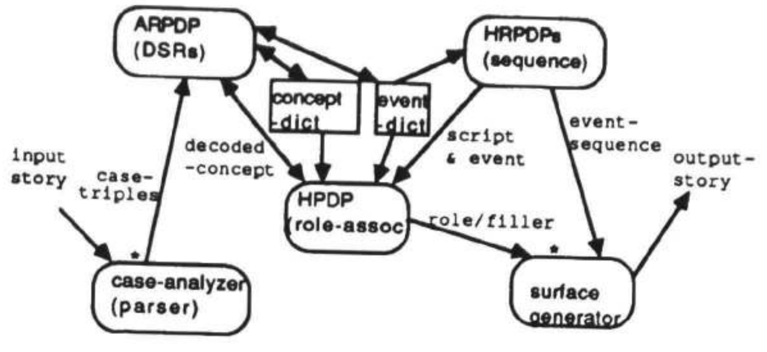

Figure 2: DYNASTY system architecture. The ovals represent PDP modules, while the boxes represent symbolic stores. The lines designate uni-/bi-directional data flow. Modules marked with * are not developed yet.

2 DYNASTY System Architecture

DYNASTY has three different PDP (Parallel Distributed Processing) modules in its system architecture: ARPDP (Auto-associative Recurrent PDP), HRPDP (Hetero-associative Recurrent PDP) and HPDP (Hetero-associative PDP) modules. Figure 1 shows the three different modules in the system. Each module is not an entirely new architecture. For example, Pollack [8] used an ARPDP architecture[2] to generate recursive distributed representations of stacks and parse trees. HRPDP architecture has been used by many researchers, e.g. Elman [9], Allen [10], Hanson [11] and John [12] for several applications: natural language question-answering[10], parsing[11] and sentence comprehension[12]. HPDP architecture is an ordinary three layer PDP architecture. But what is new is that DYNASTY uses all these different PDP sub-architectures as modular components for a coherent system architecture, namely, a system for script application.

Figure 2 shows the overall DYNASTY architecture. DYNASTY modules communicate through a global dictionary [7] which has distributed representations of concepts and events. The ARPDP oval consists of two ARPDP modules, and their functions are to develop distributed semantic representations for concepts and events in an input script-based story. In the same way, the HRPDP oval consists of two HRPDP modules, and their functions are to learn sequential knowledge in the script and produce entire script events sequentially. Finally, the HPDP oval consists of one HPDP module with three symbolic buffers: event-match-list, script-instance-buffer, and script-buffer. Their functions are to learn the script role and filler associations. The internal architectures and their functions will be described in detail.

[2]He used a different name, i.e., RAAM (Recursive Auto Associative Memory).

3 Learning Distributed Semantic Representations

3.1 Criteria for a Distributed Semantic Representation

A distributed representation able to represent conceptual knowledge must have five features:

1.*Automaticity* – The representation must be acquired through some automatic learning procedure, rather than set by hand. For instance, the hand-coded microfeature based representation[15] does not meet this criterion.

2.*Portability* – The representation should be global rather than locally confined to its training environment. That is, the representation learned in one training environment should have structural/semantic invariant properties so that it can be applied in another task environment. For example, the representation in Hinton's family tree example[19] can be said to meet the automaticity criterion, but not the portability criterion, since it cannot be used in any other task.

3.*Structure Encoding* – Feldman[20] has argued that any conceptual representation must support answering questions about structural aspects of the concept. For example, part of the meaning of "irresponsible" is that there was an obligation established to perform an action and the obligation was violated. To answer a question about the meaning of "irresponsible" requires accessing these constituent structures. Any conceptual representation must have structural information in the representation itself about the constituents of the concept and purely holographic representations do not meet this criterion. This structure-encoding criterion implies systematicity, compositionality, and inferential coherence - the three properties that Fodor and Pylyshyn[18] mentioned when criticizing connectionism. The extended back-propagation method, FGREP[7], can be said to meet the first and the second criteria, but the resulting FGREP representation is purely holographic. We can not retrieve any structural information from the representation itself. Thus representations of lexical entries in the FGREP lexicon do not allow us to answer questions about the constituents of any word's conceptual structure. Hand-coded microfeatures are a good representation according to this criterion, since at least one can interpret the semantic content of each microfeature in the representation, but they are arbitrary, lack structure, and create a knowledge engineering bottleneck.

4. *Micro-Semantics* – Distributed representations gain much of their power by encoding statistical correlations from the training set, which are used to characterize the environment. These statistical correlations give connectionist models the ability to generalize. To support generalization, distributed representations should exhibit semantic content at the micro level, i.e. similar concepts should end up (by some metric) with similar distributed representations. This criterion provided the original impetus for microfeature-based encodings, since similar concepts are similar because they share similar microfeature values.

5. *Convergence* – A basic operation for any self-organizing (possibly chaotic) representation is convergence to a (possibly chaotic) attractor. At any one time, the representation should have a stable pattern of activation over the ensemble of units in a stable environment, and this pattern should converge to an attractor point in the feature space[14].

3.2 Forming Distributed Semantic Representations (DSRs) of Words

In this section we show how DSRs may be formed and demonstrate their validity for the task of encoding word meanings.

There are two alternate views on the semantic content of words: (1) The *structural view* defines a word meaning only in terms of its relationships to other meanings. (2) The *componential view* defines meaning as a vector of properties (e.g. microfeatures). We take an interim view – that meaning can be defined in terms of a distributed representation of structural/functional relationships, where each relationship is encoded as a proposition. Examples of propositions are verbal descriptions of action-oriented events in everyday experiences.

3.2.1 Representing DSRs

The intuition behind DSRs is that people learn the meanings of words through examples of their relationships to other words. For example, after reading the 4 propositions below, the reader begins to form a hypothesis of what kind of meaning the word "foo" should have.

- Proposition1: The man drinks foo with a straw.
- Proposition2: The company delivers foo in a carton.
- Proposition3: Humans get foo from cows.
- Proposition4: The man eats bread with foo.

The meaning of foo should be something like that of milk. The interesting fact is that the semantics of "foo" is not fixed, rather it is gradually refined as one experiences more propositions in varying environments. To develop DSRs based on propositions, we have to define the structural/functional relationships between concepts with respect to those propositions. For action-oriented events describing propositions, we use thematic case relations, originally developed by Fillmore[13], and extended in several natural language processing systems[21]. We use the following 8 thematic case relations which are similar to the ones defined in Fillmore[13] : agent, object, co-object, instrument, source, goal, location, and time. For example, the DSR of "milk" is now defined as the composition of relationships, e.g. with respect to the 4 propositions above. These are then combined as follows:

milk = F_i (G_c (object, *proposition1*), G_c (object, *proposition2*), G_c (object, *proposition3*), G_c (co-object, *proposition4*),.....)

where *milk* is the meaning representation of "milk"; F_i is some integration function and G_c is some combination function of structural/functional relationships with respect to the corresponding propositions. In the same way, each proposition itself is defined as the composition of the constituent thematic case components *that are themselves combinations of structural/functional relationships* with their corresponding meaning representations of other words:

proposition1 = F_i (G_c (agent, *man*), G_c (verb, *drink*), G_c (object, *milk*), G_c (instrument, *straw*))

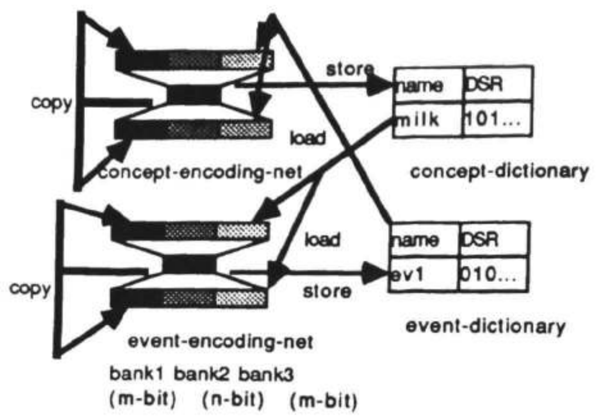

Figure 3: ARPDP Network Architecture for Learning DSRs

3.2.2 Learning DSRs

We have developed ARPDP (auto-associative recurrent PDP) networks for automatically learning DSRs. The basic idea is to *recirculate* the developing internal representation (hidden layer of the network) back out to the environment (input and output layers of the network).[3] Figure 3 shows our ARPDP architecture. The learning portion of the ARPDP architecture contains two symbolic memories (concept dictionary and event dictionary) and two 3-layer ARPDP networks. The input and output layers of each network has 3 banks of units: bank1, bank2, bank3. After each of the 3 banks is properly loaded with the elements of a proposition, the DSR emerges in bank1 by unsupervised auto-associative BEP (Backward Error Propagation)[16].

The DSR learning process consists of two alternating cycles: Concept Encoding and Proposition (Event) Encoding. Below we informally describe each cycle. In each, all concept and proposition representations start with a *don't care* pattern, e.g. 0.5, when the activation value range of each unit in network is 0.0 to 1.0. The structural/functional relationship representation is fixed, using orthogonal bit patterns (for minimizing interference).

Concept Encoding Cycle:

1. Pick one concept to be represented, say CON1.
2. Select all relevant triples for CON1. In the *milk* example, they should be triples like (*milk* object proposition1) (*milk* object proposition2) (*milk* object proposition3), etc.
3. For the first triple, load the initial representation for CON1 into bank1; load the structural/functional relationship into bank2, and load its corresponding proposition to bank3. In the *milk* example, for the first triple, bank1, bank2, and bank3 are loaded with bit patterns for *milk*, object, and proposition1, respectively.
4. Run the auto-associative BEP algorithm, where the input and output layers have the same bit patterns.
5. Recirculate the developed (hidden layer) representation into bank1 of both the input/output layers and perform step3 to step5 for another triple until all triples are encoded.
6. Store the developed DSR into the concept dictionary and select another word concept to be represented.

Proposition (Event) Encoding Cycle: Basically this cycle undergoes the same steps as the Concept Encoding Cycle except that, this time, we load bank1, bank2, and bank3 with (respectively) the proposition (event) to be represented, structural/functional relationship, and its corresponding concept representation (DSR). The result of the encoding is stored into the event dictionary.

Now the overall DSR learning process will be:

1. Perform the entire *concept encoding cycle.*
2. Perform the entire *proposition (event) encoding cycle.*
3. Repeat step1 and step2 until we get stable patterns for all concepts and events.

In this process,the composition function F_i is embodied in the dynamics of the Recursive Auto-Associative Stacking operation[8] and the combination function G_c is just a concatenation of two bit patterns. So what the ARPDP architecture does is form a representation by compressing propositions about a concept into the hidden layer and then use those compressions in the specification of propositions that define *other* concepts, and then recycle the compression formed for *this* concept back into the representation of the original concept (doing this over and over until it stabilizes). Thus each DSR has in it the propositional structure that relates it to other concepts, where each of those are also DSRs. This method produces what may be viewed as generalizations of Hinton's "reduced descriptions" [28].

[3]The idea of recirculation was first developed by Pollack[8] and Miikkulainen and Dyer[7].

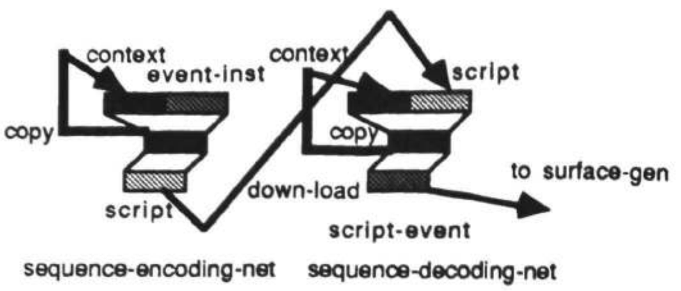

Figure 4: HRPDP architecture for learning the sequentaility of events

The decoding process is the reverse process of encoding: We load the concept representation in the hidden layer of the ARPDP concept encoding network and perform relaxation until we get the desired relationship in bank2 and proposition (event) in bank3 of the output layer. Next, we load the resulting proposition (event) in the hidden layer of the proposition encoding network and get back the constituent relationships and concept representations.

According to the evaluation and experiments reported elsewhere[17], the resulting DSRs meet all the 5 criteria: automaticity, portability, structure-encoding, micro-semantics, and convergence.

4 Learning Sequentiality of Events

Event sequences are encoded in two HRPDP networks, namely, a sequence encoding network and a sequence decoding network. Figure 4 shows this portion of the system architecture. The sequence encoding network has 2 banks in the input layer, namely, a context bank and an event bank, and has 1 script bank in the output layer. Similarly, the sequence decoding network has 2 banks: a context bank and a script bank in its input layer and 1 event bank in its output layer.

During the training phase, the system repeats the sequence encoding and decoding procedures for all the scripts defined in the system. For one script, the seqence encoding procedure is:

1. Select all relevant script instances (specific and incomplete event sequences with script roles already filled) and choose one instance.
2. Load *script bank* with the fixed orthogonal script representation.
3. Load *context bank* with *don't care* patterns and *event bank* with the first event representation in the chosen script instance from the event dictionary.
4. Do hetero-associative BEP.
5. Copy the developed hidden layer into the *context bank* and load the *event bank* with the next event representation.
6. Repeat step 4 to step 5 with all the event representations in the chosen script instance.
7. Choose another script instance and repeat step 2 to step 6 until all the selected script instances are encoded.

In this procedure, the weight vectors along with the context bank learn the correct encoding of the sequences for each script instance. For the same script, the sequence decoding procedure is:

1. Load the *context bank* with *don't care* patterns and load the *script bank* with the fixed orthogonal script representation.
2. Load the *event bank* with the first event representations of the chosen script. In this case, the generic event with the script roles unfilled is used.
3. Do hetero-associative BEP.
4. Copy the developed hidden layer into the *context bank* and load the *event bank* with the next event representation.
5. Repeat step 3 to step 4 with all the event representations in the script.

In the same way, the weight vectors along with the context bank learn the correct decoding of the sequences for the script.

In the performance phase, the system can do the causal chain completion inferences by using learned sequential knowledge. In this phase, the context bank is loaded with the don't care patterns and the event bank is loaded with the event representations from the input story. After a series of relaxations and copy actions, the script representation emerges in the script bank of the sequence encoding network. This is similar to the script recognition process in symbolic AI models. But here we don't need to worry about the script header problems: *All the events in the input story cooperate to invoke one script representation.* We load this script pattern into the script bank of the sequence decoding network and the same series of relaxations and copy actions make the completed event sequence emerge in the event bank. Since these

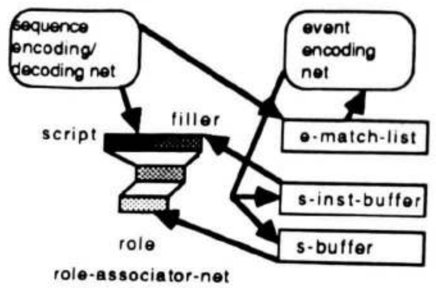

Figure 5: HPDP architecture for script role association.

events are role stripped, we need to fill the roles with fillers using a role associator network(Figure 5). After roles are filled, the output event sequences form the causal chain, i.e. the completed story output. This role binding operation will be addressed in the next section.

5 Role Association and Retrieval in DYNASTY

Role binding is not easy in a system using distributed representations since it is impossible to have context-free role variables. With similarity-based distributed representations, a solution using binding units, like in [22], will not scale up. In Smolensky's tensor approach [23], the superposition of several (schema role filler) triples in one cube makes *unorthogonal patterns* hard to be retrieved correctly.

We take a different approach to script role binding problems, namely, we consider both script roles and fillers as *full-fledged concepts.* So script roles are *associated* with their fillers rather than bound in the symbolic sense. This approach is in the same spirit as Wilensky's [24] frame/slot (or node/link) distinctions in his CRT (Cognitive Representation Theory), and as Touretzky and Geva [25], who used diffuse patterns for both slot names and fillers in their DUCS (Dynamically Updatable Concept Structures) architecture.

Figure 5 shows the role association/retrieval architecture in DYNASTY. While the HRPDP architecture is doing sequence encoding and decoding, the corresponding events in the input and output story are kept in the event-match-list. The event pairs are decoded using the ARPDP proposition (event) encoding network (see Figure 3) and stored into the script-instance-buffer and script-buffer respectively. The decoded results for the events in the input story (from the script-instance-buffer) are loaded into the filler bank, while the results for the events in the ouput story (from the script-buffer) are loaded into the role bank in the role associator network. The script bank in the same network is loaded with the patterns in the script bank in the sequence encoder network. After BEP training, the weight vectors learn the generalized features for script role and filler associations with the corresponding script representations. Then this role associator network is used to retrieve correct roles when the script and fillers are given. This is a new approach to the script role binding problem: By accumulating the associative knowledge between several roles and fillers while processing several scripts, the role associator network learns *the general role-filler associative features, not individual role-filler bindings.*

6 Experiment Results

We selected 4 scripts: *going to a restaurant, attending a lecture, grocery shopping, and visiting a doctor* from [1] and made 8 variations of each script. From the resulting 32 scripts, we extracted 122 events (propositions) to train our ARPDP modules. Figure 6 shows parts of our learned DSRs for the concepts and events.

In concept representations (CON-NAME in Figure 6), the first group designates script role concepts, while the second group designates their filler concepts. The filler concepts (e.g. John, Jack) for the same role (e.g. customer) develop similar representations. The third group designates some of the verb representations. Some of the concepts developed exactly the same representations, which is due to the limited number of propositions provided. The more propositions used in the training, the more refined are the representations.[4]

In event representations (EVENT-NAME in Figure 6), the first group designates events in the *restaurant* script, while the second group designates the same events in a specific instance (with script roles filled by proper filler concepts). The corresponding events in the second group also develop similar representations.

Next, we made up input stories (not causally completed ones) and fed them to our HRPDP and HPDP modules to get the causally completed stories with the correct role associations.

Figure 7 shows our results for the *restaurant* script when the system is fed with the input story *"John went to Sizzler. John ate steak-and-shrimp. John left a tip."* In each representation, the first row designates system output, and the second row shows the correct values for the comparison.

[4]122 propositions are obviously insufficient in number to learn 70 concepts. We postulate that a child must experience a great number of propositions to learn a single concept correctly.

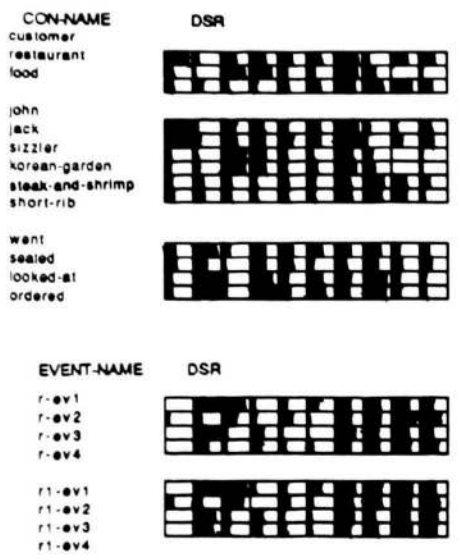

Figure 6: Learned DSRs of concepts (Nouns/Verbs) and events. The experiment is done using momentum accelerated back-propagation. Learning rate = 0.07, momentum factor = 0.5, 30 epochs for each concept and event; one epoch = 100 cycles of auto-associative backprop. The value range is 0.0 - 1.0 continuous which is shown by the degree of box fill-up.

As can be seen, the system is excellent at causal completion inference and script role retrieval.

7 Future Directions and Conclusion

A modular connectionist architecture with recursive, compositional distributed representations (the DSRs in DYNASTY) opens a new way to building practical connectionist systems that can do fairly high-level cognitive tasks. This type of neurally inspired cognitive architecture can bridge the gap between symbolic AI and the more numerical (statistical) neural network field. Usually symbolic AI systems lack in expandibility since they are brittle and break easily with large practical data. But our DYNASTY exhibits the reverse property: *The more data the system is fed, the more robust and refined its performance.* The next step is to extend this type of architecture from the prototype level to the practical level including parsing, generation and question-answering modules.

We have designed DYNASTY, a modular connectionist architecture for script processing. DYNASTY

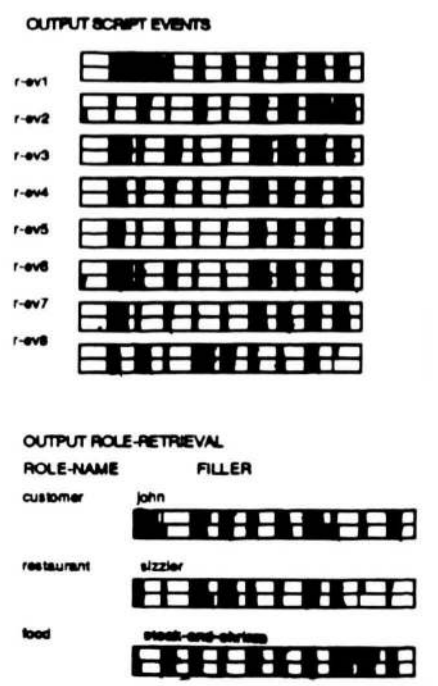

Figure 7: Causally completed output story with correct role-fillers retrieved.

can (1) automatically form distributed representations of the concepts (words) and events in the domain of script-based story understanding, (2) generate completed script event sequences from fragmentary input, and (3) successfully bind the roles in the script for the unstated events in the input. Moreover the representations formed contain constituent structure that can be extracted and events, roles, concepts with similar semantics end up with similar representations, i.e., they satisfy the 5 criteria for a DSR[17].

References

[1] Bower, G. H., Black J. B. and Turner, T. J. Scripts in memory for text. *Cognitive psychology*. 11, 177-220. 1979.

[2] Cullingford, R. E. SAM, in Schank, R. C. and Riesbeck, C. K. (Eds.) *Inside computer understanding: Five programs plus miniatures*. Lawrence Erlbaum Associates. 1981.

[3] Golden, R. M. Representing causal schemata in connectionist systems. *Proceedings of the eight annual conference of the cognitive science society*. Amherst, MA, 1986.

[4] Chun, Hon Wai and Alejandro Mimo. A model of schema selection using marker passing and connectionist spreading activation. *Proceedings of the ninth annual conference of the cognitive science society*. Seattle, WA. 1987.

[5] Rumelhart, D. E., Smolensky, P., McClelland, J. L. and Hinton, G. E. Schemata and sequential thought processes in PDP models. In Rumelhart and McClelland (Eds.) *Parallel Distributed Processing.* Vol. 2, Bradford Book/MIT Press, 1986.

[6] Dolan, C. P. and Dyer, M. G. Symbolic schemata, role binding, and the evolution of structure in connectionist memories. *Proceedings of the first international conference on neural network.* San Diego, CA, Volume II, 287-298. 1987.

[7] Miikkulainen, R. and Dyer, M. G. Forming global representations with extended back-propagation. *Proceedings of the IEEE second annual international conference on neural nets.* San Diego, CA. 1988.

[8] Pollack, J. Recursive auto-associative memory: devising compositional distributed reprsentations. *Proceedings of the tenth annual conference of the cognitive science society.* Montreal. 1988.

[9] Elman, J. L. Finding structure in time. Technical report 8801. Center for research in language, UCSD, San Diego. 1988.

[10] Allen, R. B. Sequential connectionist networks for answering simple questions about a micro-world. *Proceedings of the tenth annual conference of the cognitive science society.* Montreal. 1988.

[11] Hanson, Stephen J. and Kegl, Judy. PARSNIP: A connectionist network that learns natural language grammer from exposure to natural language sentences. *Proceedings of the ninth annual conference of the cognitive science society.* Seattle, WA. 1987.

[12] John, St. M. F. and McClelland, J. L. Applying contextual constraints in sentence comprehension. *Proceedings of the tenth annual conference of the cognitive science society.* Montreal. 1988.

[13] Fillmore, C. The case for case. In Bach, E. and Harms, R. (Eds.) *Universals in linguistic theory,* New York: Holt, Rinehart and Winston, 1968.

[14] Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of national academy of science,* Vol. 79, pp 2554-2558, 1982.

[15] McClelland, J. L. and Kawamoto, A. H. Mechanisms of sentence processing: assigning roles to constituents of sentences. In McClelland and Rumelhart (Eds.) *Parallel Distributed Processing.* Vol. 2. Bradford Book/MIT Press, 1986.

[16] Rumelhart, D. E., Hinton, G. E. and Williams, R. Learning internal representations by error propagation. In Rumelhart and McClelland (Eds.) *Parallel Distributed Processing.* Vol. 1, Bradford Book/MIT Press, 1986.

[17] Lee, G., Flowers, M. and Dyer, M. G. Learning distributed representations of conceptual knowledge. Research Report, Artificial Intelligence Lab, Dept. of Computer Science, Univ. of California at LA, 1989.

[18] Fodor, J. and Pylyshyn, Z. Connectionism and cognitive architecture: A critical analysis. *Cognition,* 28, 3-71, 1988.

[19] Hinton, G. E. Learning distributed representation of concepts. *Proceedings of the eighth annual conference of the cognitive science society,* Amherst, MA, 1986.

[20] Feldman, J. A. Neural representation of conceptual knowledge. Technical report, TR 189, Dept. of CS., Univ. of Rochester, New York, 1986.

[21] Schank, R. C. and Riesbeck, C. K. *Inside computer understanding: Five programs plus miniatures.* Lawrence Erlbaum Associates. 1981.

[22] Touretzky, D. and Hinton, G. E. A distributed connectionist production system. Technical report, CMU-CS-86-172. Computer Science Department, Carnegie Mellon Univ., Pittsburgh, 1986.

[23] Smolensky, P. A method for connectionist variable binding. Technical report, CU-CS-356-87. Dept. of Computer Science, Univ. of Colorado, Boulder, 1987.

[24] Wilensky, R. Some problems and proposals for knowledge representation. Technical report, UCB/CDS 86/294, Computer Science Division, Univ. of California at Berkeley, 1986.

[25] Touretzky, D. and Geva, S. A distributed connectionist representation for concept structures. *Proceedings of the tenth annual conference of the cognitive science society.* Montreal, 1988.

[26] Schank, R. and Abelson, R. *Scripts, plans, goals, and understanding.* LEA Press, Hillsdale, NJ. 1977.

[27] Dyer, M. G., Cullingford, R. and Alvarado, S. Scripts, in Shapiro (Eds.) *Encyclopedia of artificial intelligence.* John Wiley and Sons, Inc. 980-994, 1977.

[28] Hinton, G. Representing part-whole hierarchies in connectionist networks. *Proceedings of the tenth annual conference of the cognitive science society,* Montreal, 1988.