A Subsymbolic Model of Complex Story Understanding

Peggy Fidelman Risto Miikkulainen
Department of Computer Sciences, The University of Texas at Austin
{peggy,risto}@cs.utexas.edu

Ralph Hoffman
Yale-New Haven Psychiatric Hospital
ralph.hoffman@yale.edu

Abstract

A computational model of story understanding is presented that is able to process stories consisting of multiple scripts. This model is built from subsymbolic neural networks, but unlike previous such models, it can handle stories of variable structure and length. The model can successfully parse and paraphrase script-based stories that share long sequences of common events, with no confusion between the stories. It also exhibits several aspects of human behavior, including robustness to small changes in the sequence of events and emotion priming effects in response to ambiguous cues. It can therefore serve as a foundation for testing theories of normal and impaired story processing in humans.

Introduction

Computational models are a valuable tool for understanding human behavior. They allow investigation of how various theories about cognition may combine to produce observed human function. They can also provide a way of studying impairment in a controlled environment, where the behavioral effects of various types of underlying damage can be investigated systematically by lesioning or otherwise disrupting parts of the model.

In this paper, a computational model of human story understanding is presented that can learn to read and paraphrase script-based stories of arbitrary length. The model is built from subsymbolic neural networks, which mimic many computational properties of the brain such as distributed representations and correlation-based learning and performance. This subsymbolic foundation makes it possible to simulate phenomena that arise from these properties, which is difficult to do with symbolic models of story understanding. Unlike previous subsymbolic models, however, it is not restricted to stories consisting of a rigid, fixed-length structure. This flexibility allows the processing of more realistic stories, consisting of multiple scripts, which in turn allows more meaningful conclusions about human cognition to be drawn.

Using a small corpus of hand-designed representative stories, the model is shown to successfully parse and paraphrase stories that share long sequences of common events, with no confusion between the stories. The model also exhibits several aspects of human behavior, including robustness to small changes in the sequence of events and emotion priming effects in response to ambiguous cues.

The paper is organized as follows. Section 2 describes related work in script-based story processing, including the DISCERN model on which the current model is based. Section 3 details the architecture of the model, and Section 4 examines its behavior under various experimental conditions. Section 5 discusses the results of the experiments and possible directions for future work.

Background and Related Work

Scripts are knowledge structures for stereotypical sequences of events that allow efficient understanding of complex, realistic stories. In this section, psychological evidence for scripts will be reviewed, and computational models based on script theory will be outlined.

Script-Based Story Representations

According to script theory (Schank & Abelson, 1977), people organize knowledge of common routines into stereotypical event sequences. These *scripts* are made up of sequences of events with open slots, as well as requirements about what can fill those slots. Scripts make interaction efficient by providing everyone involved with a set of expectations about what will take place. For example, most people who have traveled by airplane know that first they must get a boarding pass, then wait in a security line, then pass through a metal detector, then wait at the gate, and so on. Without such a script, a person would have to put a lot more intellectual effort into figuring out what was expected of him at each point. If no one had such a script, airports could hardly serve the function they do.

Scripts also serve to make natural language communication efficient. There is no need to recount all the details of an ordinary visit to the dentist, for example; the speaker can just make reference to such a visit, and the listener can fill in the details herself.

The hypothesis that humans use such scripts in cognition and language is well supported by experimental evidence. For example, the degree to which events in stories will be remembered can be predicted by whether those events are part of such a script (Graesser, Gordon & Sawyer, 1979; Graesser, Woll, Kowalski & Smith, 1980). Similarly, the amount of time it takes for a human to understand a sentence can be predicted by whether it fits into a script (Den Uyl & van Oostendorp, 1980). Because scripts are a particularly well-established theory in psychology, they provide a good foundation for a computational model of story processing.

Models of Script-Based Story Processing

Scripts have been used as a basis for several symbolic models of story processing. The first of these was SAM (Script Applier Mechanism) (Cullingford, 1978), able to handle stories with multiple simultaneously active and interacting scripts. FRUMP (Fast Reading Understanding and Memory Program) (DeJong, 1979), on the other hand, skimmed newspaper stories about stereotypical episodes and filled in slots corresponding to the most important parts of the script. Scripts have been used since then in numerous symbolic systems that aim at understanding natural language stories.

Although there has been a lot of work on subsymbolic processing of sentences in the past two decades (McClelland & Kawamoto, 1986; Jain, 1991; Rohde, 1999; Henderson, 1994; Mayberry, 2004), the approach has been much less successful at the level of stories. Early on, several models were de-

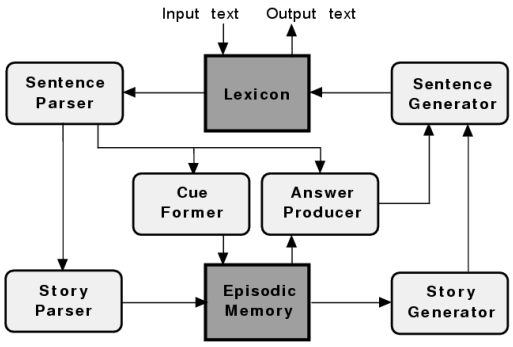

Figure 1: Architecture of the original DISCERN story processing system (Miikkulainen, 1993). All processing modules (light gray) and memory modules (dark gray) are implemented by artificial neural networks. The system could parse, paraphrase, and answer questions about stories that consist of single script instantiations.

veloped that addressed parts of this process, such as script application, sequential inference, and anaphora (Dolan, 1989; Harris & Elman, 1989; St. John, 1992).

In contrast, DISCERN (Miikkulainen, 1993) was an integrated subsymbolic model of script-based story processing, consisting of several neural network modules of the various script-processing subtasks as well as a lexicon and episodic memory (Figure 1). Processing modules in DISCERN were feedforward and simple recurrent networks trained with backpropagation, and the lexicon and episodic memory modules were implemented as self-organizing maps.

In its original form, DISCERN processed stories consisting of a single script. Although this version of DISCERN was trained with three different scripts and three different versions of each of those scripts (Miikkulainen, 1993), stories were nevertheless restricted to one of these nine structures. Except for variation in the role bindings within the script, the system could not process stories that were not simple script instantiations.

Human stories, by contrast, are rarely this simple. Since scripts are a way of encoding routinely occurring sequences of events, a story that is just a single script is by definition not worth telling. Instead, humans use scripts as building blocks in more complex stories.

The model presented in this paper is designed to extend subsymbolic story processing to this next level. Each story is a combination of several scripts in a sequence. Much of the behavior of original DISCERN is retained, but the stories that can be parsed and paraphrased are much more complex and realistic.

Model Architecture

Figure 2 depicts the organization of the new model, presented as an extended version of DISCERN. In this section, the architecture of the model is described in detail, with particular attention given to the parts that are new. More details on the original DISCERN can be found in (Miikkulainen, 1993).

Overview of DISCERN

DISCERN is an integrated natural language processing model built entirely from distributed artificial neural networks. Modularity is a key concept in DISCERN. The different script-processing subtasks such as parsing, paraphrasing, and question answering, as well as the lexical, semantic, and episodic memory components, are implemented in separate

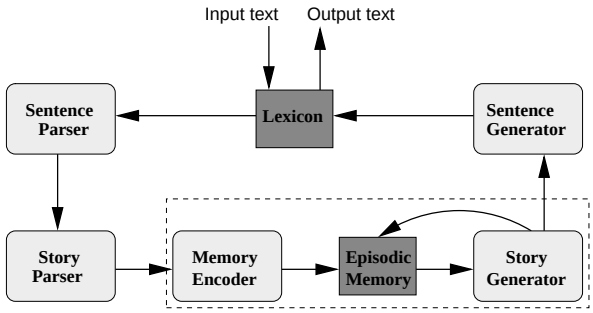

Figure 2: Architecture of the extended DISCERN model (omitting the question answering modules). The modified parts of the model (dashed-line box) make it possible for the model to process stories consisting of multiple sequential scripts.

modules. In the extended model, a memory encoder module is also included to compress the sequence of script representations. Because the experiments reported in this paper focus on paraphrasing, the question answering modules are omitted for simplicity.

The processing modules (including the memory encoder) are either feedforward or simple recurrent networks, trained with backpropagation. The lexicon module is implemented as a set of two self-organizing maps, organized in an unsupervised learning process. In the original model, the episodic memory was also a self-organizing map. In order to keep memory effects separate from paraphrasing performance, in the current implementation it is replaced with a simple array indexed by the memory encoder. (In future implementations, a map-based memory may be used as well.) All modules are trained separately and simultaneously. They learn to process each other's output, and when connected, filter out noise and errors so that the system performance is stable.

The modules communicate using distributed representations for semantic concepts, stored in a central lexicon. The representations are developed automatically by all processing networks (including the memory encoder module) while they are learning their processing tasks. With backward error propagation extended to the input layer, the representations are modified as if they were an extra layer of weights, while at the same time making them publicly available in the lexicon. This mechanism, called *FGREP*, creates a reactive training environment where the required input $\rightarrow$ output mappings change as the I/O representations change.

Single units in the resulting representations do not stand for clearly identifiable semantic features or label distinct categories. All aspects of an input item are distributed over the whole set of units in a holographic fashion, making the system robust against noise and damage. Each representation also carries expectations about its possible contexts. The emerging representations improve the system's performance in the processing task and therefore efficiently code the underlying relations relevant to the task. This coding results in good generalization capabilities, superior to parallel distributed systems with semantic feature–encoded representations.

The lexicon can be extended by cloning new instances of the items, that is, by generating a number of items with the same semantics but with distinct identities. This goal is accomplished by combining the semantic representation with a unique ID representation. This *ID+content technique* is motivated by sensory grounding of words, and forms a basis for symbolic processing in subsymbolic systems. It is possible

to approximate a large number of items by dividing them into equivalence classes, resulting in combinatorial processing capabilities with linear cost.

Representing input and output as *sequences* overcomes the combinatorial explosion in communicating structurally complex data. Internal representation can be made more general by using *data-specific assemblies*, that is, by letting part of the representation determine how the rest of the assemblies should be interpreted. These techniques are implemented in recurrent FGREP networks, of which there are two kinds. A sequential-input network reads a sequence of input items into a stationary output representation, displaying expectations about the full context of each item. A sequential-output network produces a sequence of output items from stationary input.

The FGREP modules, together with a central lexicon, are used as the basic pattern transformation building blocks in DISCERN. Processing is carried out by a hierarchical organization of four FGREP modules, trained to paraphrase stories based on a sequence of scripts using natural language input and output. The complexity of this task is reduced by effectively dividing it into subgoals. Each module is trained separately and in parallel, each developing the same lexicon.

The properties reviewed above also apply to the extended model. In addition, there are several differences, outlined below.

Extension to Multi-Script Stories

As noted in Figure 2, the major changes to the original DISCERN occur in two of the modules: the memory encoder and the story generator. The episodic memory has also been modified from its original version in order to make the behavior resulting from these changes clear. In addition, an assembly representing the emotional valence of a story was added to make it possible to characterize the behavior of the model in more varied conditions.

Emotional Valence Emotion is represented by a single assembly (12 units) in the output of the story parser. In the experiments reported in this paper, the three valences "positive," "neutral," and "negative" are used, and their representations are learned with FGREP the same way as representations for story words. Each story is associated with a single emotional valence, sustained throughout the parsing and paraphrasing process. The valence does not express any specific propositional aspect of a story; it simply represents the emotional context for the story in the model's memory. Emotion can therefore be used to prime recall and processing of the stories in the extended model.

The story parser learns to produce each story's emotion from the sequence of sentences it receives as input, just as it learns to produce the rest of the slot-filler representation for each script. It is worth noting that the script does not uniquely determine the emotion in all cases. This point is illustrated by the two example stories in Table 1.

To the story parser, the emotional valence is in many ways just like the other slots in a story representation, since its possible bindings all reside in the lexicon. In the memory modules and the story generator, emotion plays an important role, affecting the system's choice between alternative continuations of the story.

Memory Encoder Module In the original DISCERN, the entire story could be represented by one slot-filler representation. These fixed-length representations could be easily handled by both the episodic memory module and the story generator. However, in the extension there is no limit on the number of scripts – and thus the number of slot-filler representations – involved in a story. It is therefore necessary to design a way of compressing stories of arbitrary length into a fixed-size representation, while still retaining enough information for the story generator.

STORY 1	**STORY 2**
Emotional valence: negative	Emotional valence: positive
$airline Bob _ express jet Chicago long coach	**$airline John _ express jet Chicago long coach**
Bob goes to express check-in .	John goes to express check-in .
Bob gets boarding-pass to coach seat .	John gets boarding-pass to coach seat .
Bob goes through airport security .	John goes through airport security .
Bob gets on jet plane to Chicago .	John gets on jet plane to Chicago .
Flight is long .	Flight is long .
Plane arrives at Chicago airport .	Plane arrives at Chicago airport .
Bob gets off plane .	John gets off plane .
$outlaw Bob terrorist-cell plants-bomb _ airport _ van	**$relationship John Mary trusts girlfriend _ _ good**
Bob belongs to terrorist-cell .	John meets Mary .
Bob is bad .	Mary is girlfriend of John .
Bob drives a van .	Mary cares about John .
Bob plants-bomb at airport .	Mary is good to John .
	John trusts Mary .

Table 1: Two example stories with the same starting scene but different emotional valence. Each sentence of a story is parsed from a sequence of words, such as "Bob goes to express check-in," into a case-role representation such as "Bob goes _ express _ _ check-in," which is a concatenation of FGREP representations for words that fill the case roles agent, act, indirect object, attribute, direct object, origin, and destination, with "_" indicating that the role is not filled (i.e., a blank representation). Similarly, the sequence of sentence case-role representations is parsed into a slot-filler representation of the script, such as "$airline Bob _ express jet Chicago long coach," which is a concatenation of FGREP representations for words that fill the slots script, agent, patient, primary attribute, secondary attribute, place, duration, and tertiary attribute. The same script can be associated with different emotional valences in different stories.

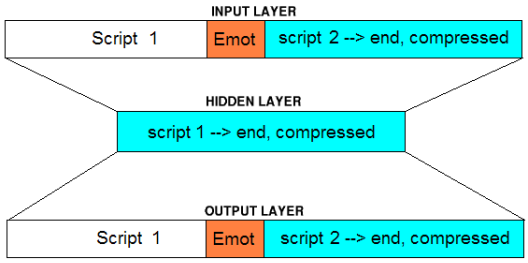

Figure 3: The memory encoder module. During training, the network learns to replicate the input activations as well as possible in the output layer. As a result, the hidden layer activations become a compressed version of the information contained at the input layer.

Recursive Auto-Associative Memory, or RAAM (Pollack, 1990), is an architecture that forms compact distributed representations of recursive data structures such as lists. Because the stories handled by the model are essentially lists of scripts, RAAM provides a way of compressing them. Figure 3 shows the structure of the memory encoder module. The top and bottom rows represent the input and output layers, and the middle row represents the hidden layer, which is smaller than the input and output layers. During training, the network learns to replicate the input activations as well as possible in the output layer. As a result, the hidden layer activations become a compressed version of the information contained at the input layer. A list is compressed by starting at the end of the list and building up a compression for it one item at a time. The representation for that item (in this case, a slot-filler representation for one script) makes up the first part of the input to the network, and the second part of the input consists of the compression of all the items that follow, which was created by the network on the previous time step.

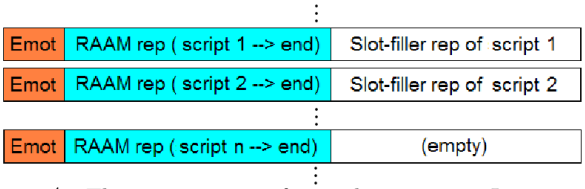

Figure 4: The organization of episodic memory. Items in the memory are created by the memory encoder module. Cues to the memory will be compared against the part of the entries shown here in color.

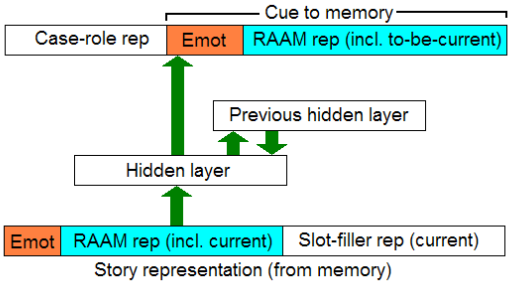

Figure 5: The story generator module in the extended model. This module produces a sequence of case-role sentence representations that constitute the full paraphrase of the story. In addition, it produces a cue to the episodic memory that will determine the story generator's own next input.

Episodic Memory Module The task of the episodic memory is to store the slot-filler representations of all scripts that together make up each story. The episodic memory was implemented as a simple array of items created by the memory encoder module, as depicted in Figure 4. Cues to the memory consist of an emotion plus a compression of the remaining story, and the retrieved script will be the one whose emotion and compression most closely match the cue (in terms of Euclidean distance). In this way, the successive scripts' slot-filler representations can be retrieved from the memory one after another, and given to the story generator as input.

Story Generator Module The input to the story generator consists of the sequence of script slot-filler representations retrieved from the episodic memory module. As in the original DISCERN, this module produces a sequence of case-role sentence representations as its output that constitute the full paraphrase of the story. However, in the extension to DISCERN, the story generator has an additional task: at each step, in addition to outputting the sentence, it has to produce a cue to the episodic memory that will determine the story generator's own next input. Figure 5 shows the modified architecture of the story generator module.

Processing Multi-Script Stories

After the modules have been trained, they are connected together in a chain, as depicted in Figure 2. Stories in the corpus are presented to the model one word at a time. The sentence parser builds a case-role representation of each sentence as it comes in, and at the end of each sentence that representation is passed as input to the story parser. With this sequence of sentence case-role representations as input, the story parser builds a slot-filler representation of the current script. At the same time, it builds a representation of the emotional valence of the current story. At the end of each script, the slot-filler representation is put aside for later use by the memory encoder.

Once the entire story has been read and the slot-filler representations for all its scripts (as well as a representation of overall emotional valence) have been formed by the story parser, they are presented one at a time to the memory encoder network, starting with the last script and working backwards as described at the end of the explanation of the memory encoder module above. After a script has been presented to the memory encoder, the activation in this network's hidden layer is combined with the slot-filler representation of that script and the emotion of the whole story to form an entry in the episodic memory. Then the second part of the memory encoder network's input is set to match its hidden layer activations, the first part is set to the slot-filler representation of the *previous* script in the story and the emotion of the entire story, and the process is repeated to form an episodic memory entry for this previous script. In this way the episodic memory is populated with the scripts of one story. This entire process is repeated for all remaining stories in the corpus.

Once the model has read all stories, the paraphrasing phase begins. The first entry in the episodic memory, which corresponds to the beginning of a story, is retrieved, and the story generator inputs are set to this representation. It consists of the slot-filler representation of the first script, the emotional valence of the story, and the RAAM compression of the entire story. The story generator then produces a case-role sentence representation corresponding to the first sentence of the first script; at the same time, it produces a cue to the episodic memory. The memory retrieved based on this cue becomes the next input to the story generator; at this point, it is the same representation as in the previous step. After all sentences of the script have been output, the cue changes, the representation of the next script is retrieved, and the story generator continues by generating the sentences of the second script. In this manner, the story generator is able to step through several successive scripts, while at the same time maintaining a memory (in its hidden layer) of the entire story.

The sentence generator uses the case-role representation to output a sequence of words which tell the first sentence of the first script of the story. This process continues until the end of this story is reached, and then it is repeated for all remaining stories in the episodic memory.

Experiments

The model was trained and tested with a corpus of 9 stories, 8 of which consisted of two scripts and one of which consisted of three. Each of the 9 scripts from which the stories were composed consisted of 4–7 sentences. Many of these sentences were common to several of the scripts. In some cases the same sequences of sentences occurred in several scripts; for example, the sequence "<PERSON> is bad. <PERSON> drives a <CARTYPE>." occurred in 3 of the 9 scripts. The ID+content technique, described briefly in Section 3a, was used to create 3 distinct characters from the semantic representation "PERSON." In the discussion that follows, the term "instances" refers to these 3 words.

The modules were trained separately and simultaneously for 30,000 epochs, by which point the error had plateaued. The modules were then connected together in a chain, as shown in Figure 2, and the performance of the entire system was analyzed. First, performance under normal conditions was evaluated, when the stories simply need to be paraphrased. Then the behavior of the system was tested under various special conditions such as errors and ambigu-

Module	All words	Instances	$E_i < 0.15$	E_{avg}
Sent. pars.	100.0	100.0	98.9	0.011
Story pars.	99.4	100.0	97.6	0.014
Story gen.	99.7	100.0	97.5	0.045
Sent. gen.	99.6	98.2	98.2	0.027

Table 2: Paraphrasing performance of the model under normal conditions. The first column indicates the percentage of words correctly output by each module. The second column shows performance on words created with the ID+content technique. The third column represents the percentage of output units whose error was less than 0.15, and the fourth column shows the average error over all output units for each module.

ity, leading to predictions about human behavior.

Behavior under Normal Conditions

The overall accuracy of the model with all modules connected in a chain is shown in Table 2. Note that all modules produce nearly 100% of words correctly, meaning that the system is performing its task nearly perfectly at all steps.

For a qualitative characterization of this performance, consider the stories in Table 3. These two stories have the same emotional valence and include the $lawchase script with all of the same role bindings, but the model is able to generate both correctly. This indicates that the story compressions produced in the memory encoder module are rich enough that the story generator can use these compressions alone to distinguish between stories.

STORY 1	**STORY 2**
Emotional valence: negative	Emotional valence: negative
$outlaw Bob terrorist-cell plants-bomb _ airport _ van	**$lawchase Bob police plants-bomb _ airport _ van**
Bob belongs to terrorist-cell .	Police go to airport .
Bob is bad .	Witness talks to police .
Bob drives a van .	Police find evidence of Bob in plants-bomb .
Bob plants-bomb at airport .	Bob is bad .
$lawchase Bob police plants-bomb _ airport _ van	Bob drives a van .
Police go to airport .	Police try to catch Bob .
Witness talks to police .	**$airline Bob _ express jet Chicago long coach**
Police find evidence of Bob in plants-bomb .	Bob goes to express check-in .
Bob is bad .	Bob gets boarding-pass to coach seat .
Bob drives a van .	Bob goes through airport security .
Police try to catch Bob .	Bob gets on jet plane to Chicago .
$lawcatch Bob police plants-bomb is is is praised	Flight is long .
Bob is bad .	Plane arrives at Chicago airport .
Bob is caught by police .	Bob gets off plane .
Bob is charged with plants-bomb .	
Bob is jailed .	
Police are praised .	

Table 3: Distinguishing between similar stories. Even though these two stories share the same emotional valence and include the $lawchase script with all the same role bindings, the model can distinguish between them and is able to generate both of them correctly.

Behavior under Special Conditions

A model's response to errors and ambiguity can give valuable insight into its validity as an explanation of human behavior. Such experiments with the model can also lead to predictions about human behavior.

Inherited Behaviors of Original DISCERN The original version of DISCERN displays several desirable error-correcting behaviors. Many of these behaviors are automatically inherited by the extended model.

For example, when isolated words appear in inappropriate contexts in the input, they are automatically corrected in the memory trace and paraphrase. This property follows from the distributed nature of all the representations used in both the original DISCERN and the extended model.

Module	All words	Instances	$E_i < 0.15$	E_{avg}
Story gen.	98.5	96.4	95.8	0.060
Sent. gen.	97.2	93.7	95.8	0.039

Table 4: Performance of the paraphrasing modules with the system's emotion frozen at "positive." The system generates all stories correctly, including those with an emotional valence other than "positive."

Second, if the input story contains a sentence that is out of order, the story parser is often still able to generate the correct slot-filler representation, which results in the events appearing in normal order in the paraphrase. This behavior is more dependable the closer the sentence is to its correct position, and the closer it is to the end of the story. These effects are also seen in humans (Abelson, 1981).

Effect of parsing errors In the original DISCERN, if a plausible but incorrect role binding arises during the parsing phase for a given story and is not corrected prior to memory formation, that story will consistently be generated with the incorrect role binding in the future. This behavior is not automatically inherited by the extended model, since the memory encoding and retrieving processes have been substantially modified. However, the extended model also exhibits this behavior. Even though the information flow is more complex, it is still likely to be consistent.

To our knowledge, there is not yet data about what can cause humans to make consistent errors when paraphrasing a story. The model predicts that one cause of such errors may be parsing errors that are not corrected before the memory is formed.

Effect of Emotion Priming Setting the emotional valence in the input of the story generator to a particular emotion during story generation creates an effect of emotion priming.

When the story generator is presented with an unambiguous input pattern, the model will always generate the correct story, even in cases where the emotion of the story does not match the emotion of the model. For example, the model is able to generate the first story in Table 1 correctly even when the story generator input emotion is set to "positive," as long as the rest of the input pattern is an accurate representation of that story. The overall performance of the paraphrasing portion of the model when its emotion was frozen at "positive" for all stories is given in Table 4. Note that performance is disrupted very little: the vast majority of words are still output correctly.

If the input to the story generator is ambiguous, the system will still output one of the stories consistent with the cue, but among the alternatives it favors the story that matches its current emotional state. For example, when the story generator emotion is set to "positive" and the rest of the cue is equally consistent with each of the stories in Table 1, the model will generate the second story.

This behavior is consistent with human data. Human recall is biased toward memories that are associated with a person's present emotional state, either because the memory was formed during a similar mood or because its content is affectively valenced in a similar way (Blaney, 1986).

Effect of Imperfect Memory Retrieval To investigate the effects of imperfect memory retrieval, random noise was added to the story generator input at every memory retrieval step. The noise consisted of uniformly distributed random numbers in the range $[-\frac{k}{2}, \frac{k}{2}]$, where k is a parameter in the

range $[0, 1]$.

The model proves to be quite robust to this noise. When $k \leq 0.2$, almost no effect is observed. As the amount of noise increases, the model exhibits graceful degradation, with errors appearing first in the instance words (i.e., people). Only much later, when k is in the range of $0.3 - 0.4$, do significant errors begin to appear in the other words.

Interestingly, the errors are not evenly distributed in the story generator output: some sentences consistently exhibit more error than others. This behavior can be seen as a possible explanation for human data, which indicate that certain events in a script are more central than others and will be recalled more reliably (Abelson, 1981; Graesser et al., 1979; Graesser et al., 1980). The model suggests that the other sentences may be recalled at will, but because their representation is full of errors, they will be rejected by the cognitive system monitoring the output, and therefore they will not be included in the paraphrase.

Discussion and Future Work

The results presented in this paper show how subsymbolic story processing can be extended to multi-script stories. The behavior of the model matches normal human behavior in several ways. It is robust to script events being slightly out of order, and less robust if an event is far from its correct place. It corrects words that appear in inappropriate contexts. Given an unambiguous cue, the model can tell the correct story regardless of its emotional state, and in the case of an ambiguous cue, it will choose to tell the alternative most consistent with its emotional state. When memory retrieval is noisy, the model has more trouble with some sentences of a given script than others. It also makes predictions about human behavior. When humans make consistent errors in the retelling of a given story, it may be because the story was originally parsed in a wrong but plausible way and was encoded in memory with these mistakes.

Although the model is based on DISCERN, it represents a significant step beyond DISCERN's original capabilities. Being able to process stories of multiple scenes is important if the model is to be used to understand human behavior. One particularly promising direction for future work is to use the model to test hypotheses about the possible causes of schizophrenia. The underlying pathology of this disease is unknown; it is typically diagnosed on the basis of its most recognizable symptoms, which are language-related. As might be expected, current theories about the causes of schizophrenia focus on anomalies at the level of systems of neurons. A symbolic model of language processing would be of little use in investigating such causes. On the other hand, what differentiates schizophrenic language from normal language are often chaotic transitions from one script to the next, or an intermingling of emotionally charged personal and impersonal scripts that result in delusions. Previous subsymbolic models, which could process only stories consisting of a single script, consequently would be inadequate as well. The model presented in this paper, however, is both built on subsymbolic principles and able to process multi-script stories with retrieval biases driven by emotion priming. As such, it forms a promising foundation for investigating the high-level effects of the low-level pathologies hypothesized to underlie schizophrenia.

Conclusion

A cognitive model of story understanding was presented that is able to process stories consisting of multiple scripts. The model was also shown to replicate several aspects of human behavior. Because it is built on neural networks, which mimic many of the low-level computational aspects of the brain, it constitutes a promising basis for future studies on normal and impaired human story processing.

Acknowledgments

This work was supported by NIMH grant R01MH066228.

References

Abelson, R. P. (1981). Psychological status of the script concept. *The American Psychologist*, 36:715–729.

Blaney, P. H. (1986). Affect and memory. *Psychological Bulletin*, 99:229–246.

Cullingford, R. E. (1978). *Script Application: Computer Understanding of Newspaper Stories*. PhD thesis, Department of Computer Science, Yale University. Technical Report 116.

DeJong, G. F. (1979). *Skimming Stories in Real Time: An Experiment in Integrated Understanding*. PhD thesis, Department of Computer Science, Yale University. Technical Report 158.

Den Uyl, M., and van Oostendorp, H. (1980). The use of scripts in text comprehension. *Poetics*, 9:275–294.

Dolan, C. P. (1989). *Tensor Manipulation Networks: Connectionist and Symbolic Approaches to Comprehension, Learning and Planning*. PhD thesis, Department of Computer Science, University of California Los Angeles. Technical Report UCLA-AI-89-06.

Graesser, A. C., Gordon, S. E., and Sawyer, J. D. (1979). Recognition memory for typical and atypical actions in scripted activities. *Journal of Verbal Learning and Verbal Behavior*, 18:319–332.

Graesser, A. C., Woll, S. B., Kowalski, D. J., and Smith, D. A. (1980). Memory for typical and atypical actions in scripted activities. *Journal of Experimental Psychology: Human Learning and Memory*, 6:503–515.

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*, 635–642. Hillsdale, NJ: Erlbaum.

Henderson, J. (1994). Connectionist syntactic parsing using temporal variable binding. *Journal of Psycholinguistic Research*, 23:353–379.

Jain, A. N. (1991). *PARSEC: A Connectionist Learning Architecture for Parsing Spoken Language*. PhD thesis, Computer Science Department, Carnegie Mellon University. Technical Report CMU-CS-91-208.

Mayberry, III, M. R. (2004). *Incremental Nonmonotonic Parsing Through Semantic Self-Organization*. PhD thesis, Department of Computer Sciences, The University of Texas at Austin. Technical Report AI-TR-04-310.

McClelland, J. L., and Kawamoto, A. H. (1986). Mechanisms of sentence processing: Assigning roles to constituents. In *Parallel Distributed Processing, Volume 2*, 272–325. Cambridge, MA: MIT Press.

Miikkulainen, R. (1993). *Subsymbolic Natural Language Processing: An Integrated Model of Scripts, Lexicon, and Memory*. Cambridge, MA: MIT Press.

Pollack, J. B. (1990). Recursive distributed representations. *Artificial Intelligence*, 46:77–105.

Rohde, D. L. (1999). A connectionist model of sentence comprehension and production. Dissertation Proposal, School of Computer Science, Carnegie Mellon University.

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

Schank, R., and Abelson, R. (1977). *Scripts, Plans, Goals, and Understanding: An Inquiry into Human Knowledge Structures*. Hillsdale, NJ: Erlbaum.