Published by
Nature Publishing Group, 2003
The Macmillan Building, 4 Crinan Street, London, N1 9XW, UK

Associated companies and representatives throughout the world

www.nature.com

ISBN: 0-333-792610

Distributed on behalf of the Nature Publishing Group in the United States and Canada by
Grove's Dictionaries, Inc.
345 Park Avenue South,
New York,
NY 10010-1707,
USA

British Library Cataloguing in Publication Data
Encyclopedia of cognitive science
1. Cognitive science – Encyclopedias
I. Nadel, Lynn
153'.03

Library of Congress Cataloging in Publication Data

A catalog for this record is available from the Library of Congress

Typeset by Kolam Information Services Pvt. Ltd., Pondicherry, India
Printed and bound by The Bath Press, England

Activation and Inhibition: Advances in Social Cognition, vol.11, pp. 1–52. Mahwah, NJ: Lawrence Erlbaum.
Crocker J, Major B and Steele C (1998) Social stigma. In: Gilbert DT, Fiske ST and Lindzey G (eds) *Handbook of Social Psychology*, 4th edn. pp. 504–553. New York, NY: McGraw-Hill.
Devine PG and Monteith MJ (1999) Automaticity and control in stereotyping. In: Chaiken S and Trope Y (eds), *Dual-process Theories in Social Psychology*, pp. 339–360. New York, NY: Guilford Press.
Fiske S (1998) Stereotyping, prejudice, and discrimination. In: Gilbert DT, Fiske ST and Lindzey G (eds) *Handbook of Social Psychology*, 4th edn. pp. 357–411. New York, NY: McGraw-Hill.
Steele CM (1997) A threat in the air: how stereotypes shape intellectual identity and performance. *American Psychologist* **52**: 613–629.

Story Understanding

Intermediate article

Erik T Mueller, IBM Thomas J Watson Research Center, Yorktown Heights, New York, USA

CONTENTS

Story understanding researchers have built computer programs that read and answer questions about simple stories. They have proposed a number of knowledge structures and processing mechanisms useful for this task.

INTRODUCTION

The field of story understanding is concerned with building computer programs that can understand stories. It also investigates how people understand stories. Thus far, the field has dealt mainly with short stories a few paragraphs long, rather than full-length novels, which represent a much harder problem. By 'understanding' it is usually meant that the program should be able to answer questions about a story it reads. It should also be able to generate summaries.

STORY GRAMMARS

An early formalism for stories was the story grammar (Rumelhart, 1975). A story grammar consists of a context-free grammar, with semantic rules associated with each rule of the context-free grammar. An example of a simple story grammar is:

Story → Setting + Episode
  (Setting ALLOWS Episode)
Setting → State
Episode → Event + Emotion
  (Event CAUSES Emotion)
Event → Event + Event
  (Event CAUSES Event)
Event → Episode
Event → Action (1)

Consider the story:

1. Althea was in the playroom.
2. She sat on the balloon.
3. It popped.
4. She was pleased. (2)

The story grammar can be used to parse the story as follows:

Story ⇒
Setting ALLOWS Episode ⇒
State ALLOWS Episode ⇒
1 ALLOWS Episode ⇒
1 ALLOWS (Event CAUSES Emotion) ⇒
1 ALLOWS ((Event CAUSES Event) CAUSES Emotion) ⇒
1 ALLOWS ((Action CAUSES Action) CAUSES Emotion) ⇒
1 ALLOWS ((2 CAUSES 3) CAUSES 4) (3)

The complete parse of the story is:

Althea was in the playroom ALLOWS ((She sat on the balloon CAUSES It popped) CAUSES She was pleased) (4)

This parse can then be used to answer questions:

Why did the balloon pop? Because she sat on it. (5)

Why was she pleased? Because she sat on the balloon and it popped. (6)

The problem with story grammars is that they tie together form and content. According to the above story grammar, the form Event1 + Event2 corresponds to the content Event1 CAUSES Event2. Yet it is quite possible that Event1 + Event2 instead corresponds to Event2 CAUSES Event1, as in the text

The balloon popped. She sat on it. (7)

Although story grammars are useful for capturing the structure of certain story forms such as folk tales, they do not account for the content of a story. They fail to address how an understander is able to make sense of a story despite the variety of ways of expressing the story. The ingredient missing from story grammars is the understander's knowledge about the way the world works.

KNOWLEDGE-DRIVEN STORY UNDERSTANDING

Stories do not specify everything down to the last detail. Rather, to understand a story one must 'fill in the blanks' and make inferences. Given the text

Althea shook the piggy bank. Three dimes fell out. (8)

a reader easily infers that the dimes fell out because Althea shook the piggy bank, even though this was not explicitly stated. This inference can be made because the reader knows that coins are often stored in piggy banks, that piggy banks have a slot through which coins and other small objects can pass, that shaking helps those objects pass through that slot, and that unsupported objects fall. Such information, known to the reader but not contained in the text, is variously referred to as 'world knowledge', 'general knowledge', 'commonsense knowledge', or simply 'knowledge'.

Much research has focused on identifying the types of knowledge required for story understanding, representing that knowledge within a computer program, and building programs that make use of the knowledge.

Demons

Early work on story understanding (Charniak, 1972) used a single mechanism, called 'demons', for representing and applying knowledge. A demon consists of a test and an action. The test specifies a condition to await. The action specifies an action to perform when the condition becomes true. The following demon allows a program to make the correct inference regarding the piggy bank:

Test: Person P shakes piggy bank B and money M comes out of B.
Action: Assert that M comes out of B because P shakes B. (9)

That is, this demon generates the inference:

Three dimes fell out BECAUSE Althea shook the piggy bank. (10)

Using a large number of demons, a story understanding program will be able to make a large number of inferences. However, demons can go off on a tangent, generating inferences of doubtful relevance to a story:

The dimes were in the piggy bank BEFORE they fell out. (11)

The dimes were somewhere else BEFORE they were in the piggy bank. (12)

The dimes were minted BEFORE they were somewhere else. (13)

Althea picked up the piggy bank BEFORE she shook the piggy bank. (14)

Althea was somewhere else BEFORE she picked up the piggy bank. (15)

and so on. A type of knowledge structure, called 'scripts', was therefore proposed (Schank and Abelson, 1977) for capturing the relevant inferences in a typical situation.

Scripts, Plans, and Goals

Scripts are bundles of information about situations or activities that are common in a given culture, such as eating at a restaurant, attending a birthday party, or taking the subway. Scripts consist of roles, props, settings, entry conditions, results, and scenes. Here is a short version of the restaurant script:

Roles: customer C, waiter W
Props: table T, menu M, food F, bill B, money D
Settings: restaurant R
Entry conditions: C is hungry, C has D

Results: C is satiated
Scenes:
1. Entering: *C* goes to *R*, *C* sits at *T*
2. Ordering: *C* reads *M*, *C* chooses *F*, *W* comes to *T*, *C* requests *F* from *W*
3. Eating: *W* brings *F* to *C*, *C* eats *F*
4. Exiting: *W* brings *B* to *C*, *C* gives *D* to *W*, *C* leaves *R*

This script may be used to fill in missing information. Told that someone went to a restaurant, ordered lobster, paid the bill, and left, the listener infers (unless told otherwise) that the person ate the lobster.

Of course, the above is not the only possible sequence for eating in a restaurant. Certain types of variations can be accommodated by scripts. A script may have several 'tracks'. The restaurant script has a fast food track, a cafeteria track, and a fancy restaurant track. In the fast food track, the customer pays for the food before eating and may eat the food inside or outside the restaurant. A script may contain alternative paths. If the service is poor, the customer leaves a smaller tip.

However, suppose the story begins:

Suzy was hungry. She went to the Zagat website. (16)

Though the restaurant script mentions hunger, it does not mention visiting a particular website. Knowledge structures for plans and goals were therefore proposed (Schank and Abelson, 1977) to deal with story events that do not follow an existing script. A person has a goal to reduce hunger, and one plan for achieving this goal is toeat at a restaurant. Another plan is to eat at home. A subgoal of eating at a restaurant is to go to the restaurant. A subgoal of going to the restaurant is to know the address of the restaurant. One plan for knowing the address of a restaurant is to read a restaurant guide. Another plan is to ask a friend. One plan for reading a restaurant guide is to read it online. Zagat is a restaurant guide. There are many other plans and subgoals for many other goals. (*See* **Natural Language Processing: Models of Roger Schank and his Students**)

Knowledge of these plans and goals may be used to explain why Suzy went to the Zagat website, namely, in order to read the Zagat guide, so that she could know the address of a restaurant, so that she could go to the restaurant, so that she could eat, so that she could satisfy her hunger.

Themes

Scripts, plans, and goals allow a reader to connect up elements of a story locally. But stories usually have some overall point, moral, or theme. That is, stories are coherent globally. Researchers have proposed a number of related knowledge structures for capturing the point of a story: thematic organization packets, story points, plot units, thematic abstraction units, and planning advice themes.

Consider a story about a man who loses his job, runs out of money, and then happens to save a wealthy person who gives him a large reward. The essence of this story is captured by the story point called 'fortuitous solution' (Wilensky, 1982):

Person *P* is in a negative state.
An incidental event *E* occurs.
E results in a positive state for *P*. (17)

Plot units (Lehnert, 1982) also capture the essence of stories. Plot units are graphs consisting of linked positive events, negative events, and mental states. For example, the 'retaliation' plot unit describes any story in which person *A* causes a negative event for person *B*, which motivates *B* to cause a negative event for *A* (see Figure 1).

Space and Time

Recent approaches to story understanding (Duchan *et al.*, 1995) stress the importance of the reader's immersion in the story world. Readers imagine they are inside stories and vicariously experience them. As a story unfolds, the reader keeps track of the shifting 'where', 'when', 'who', and 'what' of the story.

Two-dimensional grids have been proposed (Mueller, 1998) for representing typical locations such as a grocery store, theatre, or hotel room, and for keeping track of the 'where' in a story understanding program. Given the text

Jim was in his hotel room. (18)

an imagined view from above is represented as a square grid in which certain cells are assigned to the various elements of the room: bed, door, minibar, wall, phone, table, and Jim.

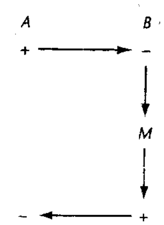

Figure 1. The 'retaliation' plot unit. Person *A* causes a negative event for person *B*, which motivates *B* (via mental state *M*) to cause a negative event for *A*.

Grids allow a story understanding program to make inferences regarding the distance between objects, the relative position of objects (left, right, front, back), whether story characters can see or hear each other, and whether there is a path from one location to another. The program answers questions by consulting the grid:

Was Jim near a bed? Yes. (19)

The 'when' of a story can be represented using absolute timestamps (e.g. '11 a.m. GMT on 12 January 1997') or relations on time intervals (e.g. '*A* happened before *B*', '*A* happened during *B*').

One possible organizational structure for the 'where' and 'when' is the scenario participant map (Dyer, 1983). This is a graph consisting of settings (such as the elevator or hotel room where the story action takes place) connected by transitions (such as walking through the hallway to get to the hotel room).

USING SCRIPTS, PLANS, GOALS, STORY POINTS, AND PLOT UNITS

A story understanding program builds representations of knowledge structures such as scripts, plans and goals while reading a story. Those representations can then be used for question answering, paraphrasing, and summarization. Table 1 lists some of the story understanding programs that have been built over the years. Some of these programs can be downloaded from the internet (see 'Further Reading').

Script Application

The 'script applier mechanism' or SAM program (Cullingford, 1978) uses scripts to understand stories as follows. Suppose the first sentence of a story is

Fred went to a restaurant. (20)

SAM must activate the restaurant script. Script headers are attached to scripts to assist in script activation. In the above case, a locale header activates the restaurant script because the text mentions that a story character (Fred) went to the setting of the script (restaurant). A precondition header activates a script when the text mentions the main entry condition of a script, as in *Fred was hungry*. A direct header activates a script when the text simply states that the script occurred, as in *Fred ate at a restaurant*. A script is also activated when the text mentions an event of the script.

Alternatively, script activation may be viewed as a text categorization problem and handled using statistical natural language processing techniques. The task is to assign a segment of text to one of many scripts.

When the restaurant script is activated, 'Fred' is assigned to the customer role and 'restaurant' is assigned to the restaurant setting. SAM's representation of the story after reading the first sentence is:

restaurant script, $C =$ 'Fred', $R =$ 'restaurant'
last matched event = 'C goes to R' (21)

As further sentences of the story are read, they are matched to events of the script and additional assignments are made as necessary. For example, given *he ordered lobster*, the representation is updated to:

restaurant script, $C =$ 'Fred', $R =$ 'restaurant', $F =$ 'lobster'
last matched event = 'C requests *F* from *W*' (22)

In order to answer a question about the story, the program tries to locate an event of an active script

Table 1. Representative story understanding programs

Program	*Year*	*Knowledge structures and mechanisms*
Ms Malaprop	1977	frames
SAM	1978	scripts
PAM	1978	plans, goals
BORIS	1982	scripts, plans, goals, emotions, themes, integration
AQUA	1989	asking and answering questions while reading, explanation
DISCERN	1991	scripts, architecture of subsymbolic networks
TACITUS	1993	axioms, weighted abduction
AbMaL	1994	emotions
SNePS/Cassie	1995	propositional semantic networks, beliefs
ThoughtTreasure	1998	plans, goals, emotions, grids, simulation

that both matches the question and occurs in the script at or before the last matched event. (Later events have not yet happened in the story.) The script event 'C chooses F' matches the question *What did Fred choose?* and occurs before the last matched event. The variables C and F in the event are replaced by their values, resulting in the answer *Fred chose lobster*.

Several scripts can be active in SAM at a time, and the program can also handle certain deviations from a script, such as being served a burnt hamburger.

Tracking Plans and Goals

Since human behavior is to a large extent goal-directed, tracking the plans and goals of story characters is essential to understanding. If one is told that someone is making marinara sauce, one assumes the person has the goal of eating and is in the middle of preparing a meal.

Given observations of the behavior of story characters, the process of plan recognition (Kautz, 1991) produces explanation graphs such as that shown in Figure 2.

Since two recipes known to the understander use marinara sauce, the explanation graph contains these alternatives. The top-level goal is to satisfy hunger. In order to satisfy this goal, a subgoal of eating at home is activated. This in turn activates a subgoal to prepare a meal, which activates a subgoal to prepare spaghetti marinara or chicken marinara (the understander does not know which),

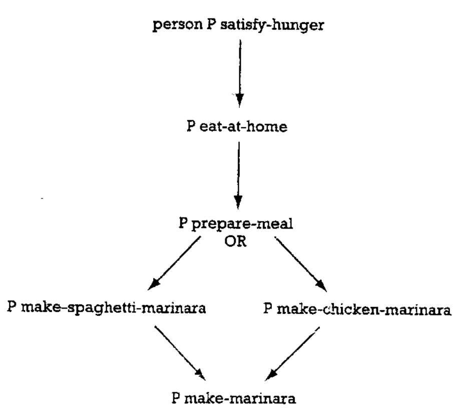

Figure 2. An explanation graph for the information that someone is making marinara sauce.

which activates a subgoal to make marinara sauce. Producing such graphs requires a library of common plans.

Stories often mention states leading to goals. The program thus creates links from states to goals activated by those states:

`P hungry → P satisfy-hunger` (23)

Emotions are intertwined with goals, and stories often mention the emotional reactions of story characters (Dyer, 1983). Goal successes result in positive emotions, while goal failures result in negative emotions. The program therefore tracks goal outcomes, and creates links from goal outcomes to their resulting emotions:

`P obtain-employment → P happy (succeeded)` (24)

The program answers questions about a story by consulting the explanation graphs:

Why was Joan making marinara sauce?
She was hungry and wanted to eat. (25)

Why was Jim happy? Because he was hired for a job. (26)

Using Story Points and Plot Units

Story points and plot units are derived from explanation graphs, extended with further goal situations. Goal conflict is the situation in which several goals of a single character interfere with each other. Goal competition is the situation in which the goals of several characters interfere. Goal concord is the situation in which the goals of several characters are compatible. Goal subsumption is the situation in which one goal continually enables satisfaction of another goal.

Plot unit graphs map to explanation graphs as follows. Positive events correspond to goal successes or positive mental states, negative events correspond to goal failures or negative mental states, and mental states correspond to active goals. Named plot units are recognized by building plot unit graphs and matching those graphs to a library of named plot units such as retaliation or fortuitous problem resolution.

Themes such as story points and plot units are useful for predicting what might come next in a story. If a sufficiently large portion of a theme is recognized, the program anticipates the events predicted by the remainder of that theme.

Themes are also useful for producing remindings. If a story is recognized as being an instance of a given theme, the understander may be reminded of another story with that theme.

Finally, themes are useful for summarization. The retaliation plot unit leads to the summarization template:

Because *A* caused a negative event for *B*,
B later caused a negative event for *A*. (27)

For example, a summary that might be produced from the above template is:

Because Harrison turned Debra down for a date, she later turned him down when he changed his mind. (28)

INTEGRATING SENTENCE ANALYSIS WITH STORY UNDERSTANDING

As a story understanding program reads a text, it must incorporate new information into its existing representation of the story. There are two ways this might be performed. In batch interpretation, the program updates its representation after reading each sentence. In incremental interpretation, the program updates its representation after reading each word. The incremental approach is valid from a cognitive standpoint since people appear to be able to interpret words in real time as they are read (Just and Carpenter, 1980).

There are two ways the story understanding program might be structured: as a series of modules with distinct responsibilities, or as one large process. The trend is towards modular processing, because it is easier to build and understand modular programs, and some self-contained modules for natural language tasks such as part-of-speech tagging now exist.

Modular processing has often been associated with batch interpretation, though this need not be the case. For example, syntactic and semantic parsing modules may cooperate to produce an interpretation incrementally (Mahesh *et al.*, 1999).

Let us adopt a modular, batch approach in order to sketch out a complete story understanding program. The first module of a story understanding program is the sentence boundary detector, which segments an input text into sentences. The next module is the entity recognizer. This module segments a sentence into words, phrases, and other entities such as places, numbers, dates, and times. The next module is the tagger, which assigns a part of speech to each entity. The next module is the syntactic parser, which takes a stream of tagged entities and produces syntactic parses such as:

```
[S
  [NP [Name Jim]]
  [VP
    [V set]
    [NP [Det the] [N milk]]
    [PP [Prep on] [NP [Det the]
    [N table]]]]]
```

The next module is the semantic parser, which takes syntactic parse trees and converts them into logical formulae such as:

`set(Jim,milk,on(table))` (29)

Ambiguities are recognized by each module and passed along to the next module. For example, a word may have several possible parts of speech, and a sentence may have several possible syntactic and semantic parses. (*See* **Natural Language Processing; Natural Language Processing, Disambiguation in; Parsing; Parsing: Overview**)

The understanding modules include a script applier, a plan recognizer, and a theme recognizer. Logical formulas from the semantic parser are fed to the understanding modules, which then update their representation or understanding of the story. The understanding modules must agree among themselves how to resolve the ambiguities that were introduced by previous modules, as well as any newly encountered ambiguities.

The question answerer and summarizer take questions from the semantic parser, and examine representations produced by the understanding modules to produce answers and summaries. They use a generator to convert representations into natural language.

The capabilities that have been proposed by various researchers as necessary for story understanding include: to extract themes and morals; to go back and reread; to look for hidden messages; to model naive physics; to model physical objects, devices, and settings; to model the minds of story characters (theory of mind); to pose questions during reading and answer them; to read according to some goal for reading; to read creatively and invent new explanatory frames; to recognize a typical situation (scripts); to reconcile conflicting interpretations; to revise an interpretation; to track emotions of story characters; to track plans and goals of story characters; to track temporal relationships; to track the shifting 'where', 'when', 'who', and 'what'; to use discourse markers; to use imagery or visual representations; to use past experiences to guide understanding; to zoom in on detail; and to zoom out from detail. This list does not include capabilities normally assumed in natural language processing, such as syntactic parsing and anaphora resolution.

The processing mechanisms used in various story understanding programs include: abduction (inference to the best explanation); backward chaining; constraint satisfaction; demons; discrimination nets; finite-state automata; forward chaining; indexing; logic; neural networks; pattern matching; plan recognition; production systems; simulation; society of agents; spreading activation or marker passing; and working memory. These vary in scope. For example, it has been claimed that all levels of natural language processing, including syntax, semantics, and pragmatics, can be handled using abduction.

SPREADING ACTIVATION AND CONNECTIONIST APPROACHES

Dissatisfaction with strictly symbolic mechanisms has led some researchers to experiment with mechanisms inspired by the physiology of the brain and experimental results in psychology such as those from priming experiments. (*See* **Syntax and Semantics, Neural Basis of; Connectionism; Priming**)

Spreading Activation

Consider the text

Suzy wanted to eat dinner. She got in her car. (30)

One way to relate these two sentences is to conduct two simultaneous searches through the plan library. One search starts from `eat-dinner` while the other starts from `enter-car`. When one search meets the other, a path has been found from `eat-dinner` to `enter-car`, as shown in Figure 3.

This method is known as 'spreading activation' or 'marker passing' (Charniak, 1986). Once a path is found, it must be verified for correctness. It must be checked, for instance, that the person who wants to eat dinner is the same person who got in the car. Marker passing with verification is one method for doing plan recognition to produce an explanation graph.

Connectionist Approaches

So far we have discussed symbolic representations of knowledge structures such as scripts, plans, and goals. An alternative kind of representation is a subsymbolic, or connectionist, one, in which

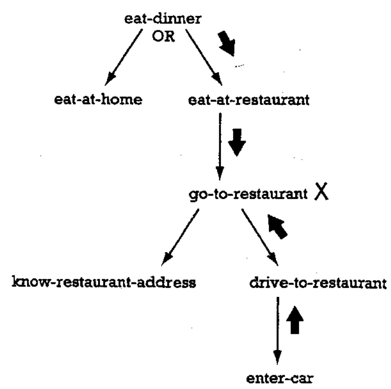

Figure 3. The statements 'Suzy wanted to eat dinner' and 'she got in her car' can be related by conducting two simultaneous searches through the plan library, one from `eat-dinner` and the other from `enter-car`. These are shown by the downward- and upward-facing stubby arrows, respectively. The point of intersection is marked by the 'X'.

concepts are represented not as discrete entities but by the pattern of activity in a neural network.

The connectionist approach has several advantages. Firstly, neural networks are neurally inspired, so a story understanding program implemented in neural networks is more likely to model how understanding is actually implemented in the brain. Secondly, neural networks can be trained on a set of examples in order to learn representations and make generalizations automatically, reducing the need for a programmer to specify knowledge structures. Thirdly, the performance of neural networks generally degrades in a steady fashion.

On the other hand, the connectionist approach has several disadvantages. Firstly, neural networks have difficulty with novel inputs, since they need to be trained on a large number of examples of their inputs. Secondly, it is difficult to implement in neural networks certain operations such as role assignment and composition that are easy to implement in symbolic programs.

Miikkulainen (1993) used the connectionist approach to build a complete program called DISCERN that reads and answers questions about script-based stories. The program is built from independent connectionist modules that communicate using distributed representations. The modules are: lexicon, episodic memory, sentence parser, sentence generator, story parser, story generator, cue former, and answer producer.

The episodic memory stores and generalizes script-based stories. It is organized by the programmer into a fixed three-level architecture. The top level represents the script class, the middle level represents the script track, and the bottom level represents the script roles. The neural networks at each level are self-organized by training on an artificially generated set of stories involving the restaurant, shopping, and travel scripts. In testing, the distributed representation of an input story, which may contain missing role bindings, is fed to the episodic memory. The episodic memory is able to fill in any missing role bindings using the generalizations it made during training. The cue former retrieves answers to questions about input stories from the episodic memory.

DISCERN can only handle one script per story and is unable to handle deviations from a script. Thus it is not as sophisticated as SAM, one of the early symbolic story understanding programs.

ASSESSMENT

Since the 1970s, we have learned much about story understanding. Yet it is still not known how to scale up a story understanding program so that it can understand more than 'toy' stories. The problem does not appear to be what symbolic or subsymbolic mechanism is used for processing – probably a number of mechanisms will do the job – but how to get a story understanding program to work at the human level at all.

It is hardly surprising that story understanding is a difficult problem. It is a task that calls the entire mind into play. All of the explanatory frames, skills, and mechanisms used to deal with everyday life can be invoked in story understanding. A story can be about almost anything, from picking up and holding an apple (motor skills) to observing the subjective redness of the apple (consciousness). (*See* **Language Comprehension, Methodologies for Studying; Discourse Processing**)

Programmers have difficulty managing the complexity of building and debugging story understanding programs. Even if a library of common-sense knowledge is available (Cyc and ThoughtTreasure are attempts at building such libraries), it is still difficult to build processing mechanisms that apply the library in understanding. It appears that the knowledge library needs to be developed with the understanding program in mind, yet it is not clear how this can be done. Story understanding might take a cue from statistical language processing and information extraction. Success has been achieved in these fields by building modules that address well-defined subproblems, such as part-of-speech tagging or filling templates about terrorism news stories. By putting together many modules that address parts of the story understanding problem, it may be possible to reach a complete solution. (*See* **Natural Language Processing, Statistical Approaches to**)

It is easy to forget how ambiguous natural language is. A sentence has many possible interpretations. Many of those possibilities are implausible, but the program does not always know that. Furthermore, the possible interpretations of each sentence must be considered in light of the possible interpretations of previous sentences. So if there are two interpretations of the first sentence, there might be four after reading the second, eight after reading the third, and so on. Knowledge structures such as scripts were designed to prevent such problems, but in practice they do not always work.

It is a mystery how people are able to avoid this proliferation of possible interpretations and understand stories. Cognitive psychologists have conducted experiments investigating how inferences are made during narrative comprehension (Graesser *et al.*, 1994). Cognitive neuroscientists are beginning to address discourse comprehension (Beeman and Chiarello, 1998). Someday it may be possible to use a brain scan with high spatial and temporal resolution to trace a behavior such as a verbal response to a question back to its causes during reading.

References

Beeman M and Chiarello C (eds) (1998) *Right Hemisphere Language Comprehension*. Mahwah, NJ: Erlbaum.

Charniak E (1972) *Toward a Model of Children's Story Comprehension*. AI Laboratory Technical Report 266, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Charniak E (1986) A neat theory of marker passing. In: *Proceedings of the Fifth National Conference on Artificial Intelligence*, pp. 584–588. Menlo Park, CA: AAAI Press.

Cullingford RE (1978) *Script Application: Computer Understanding of Newspaper Stories*. Research Report 116, Computer Science Department, Yale University.

Duchan JF, Bruder GA and Hewitt LE (eds) (1995) *Deixis in Narrative*. Hillsdale, NJ: Erlbaum.

Dyer MG (1983) *In-Depth Understanding*. Cambridge, MA: MIT Press.

Graesser AC, Singer M and Trabasso T (1994) Constructing inferences during narrative text comprehension. *Psychological Review* 101(3): 371–395.

Just MA and Carpenter PA (1980) A theory of reading: from eye fixations to comprehension. *Psychological Review* 87(4): 329–354.

Kautz HA (1991) A formal theory of plan recognition and its implementation. In: Allen JF, Kautz HA, Pelavin RN and Tenenberg JD (eds) *Reasoning About Plans*, pp. 69–125. San Mateo, CA: Morgan Kaufmann.
Lehnert WG (1982) Plot units: a narrative summarization strategy. In: Lehnert WG and Ringle MH (eds) *Strategies for Natural Language Processing*, pp. 375–412. Hillsdale, NJ: Erlbaum.
Mahesh K, Eiselt KP and Holbrook JK (1999) Sentence processing in understanding: interaction and integration of knowledge sources. In: Ram A and Moorman K (eds) *Understanding Language Understanding*, pp. 27–72. Cambridge, MA: MIT Press.
Miikkulainen R (1993) *Subsymbolic Natural Language Processing*. Cambridge, MA: MIT Press.
Mueller ET (1998) *Natural Language Processing With ThoughtTreasure*. New York, NY: Signiform. [Available from: http://www.signiform.com/tt/book/]
Rumelhart DE (1975) Notes on a schema for stories. In: Bobrow DG and Collins AM (eds) *Representation and Understanding: Studies in Cognitive Science*, pp. 211–236. New York, NY: Academic Press.
Schank RC and Abelson RP (1977) *Scripts, Plans, Goals, and Understanding*. Hillsdale, NJ: Erlbaum.
Wilensky R (1982) Points: a theory of the structure of stories in memory. In: Lehnert WG and Ringle MH (eds) *Strategies for Natural Language Processing*, pp. 345–374. Hillsdale, NJ: Erlbaum.

Further Reading

Bartlett FC (1932) *Remembering*. Cambridge, UK: Cambridge University Press.
DISCERN. [ftp://ftp.cs.utexas.edu/pub/neural-nets/software/discern.2.1.1.tar.Z.] [The *DISCERN* program developed by Miikkulainen].
Hobbs JR, Stickel ME, Appelt DE and Martin P (1993) Interpretation as abduction. In: Pereira FCN and Grosz BJ (eds) *Natural Language Processing*, pp. 69–142. Cambridge, MA: MIT Press.
ICU. [http://www-2.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/nlp/nlu/icu/0.html.] [Miniature version of the SAM model developed by Cullingford.]
Lenat DB (1995) Cyc: a large-scale investment in knowledge infrastructure. *Communications of the ACM* **38**(11): 33–48.
McDypar. [http://www-2.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/nlp/nlu/mcdypar/0.html.] [Miniature version of the parser used in the BORIS program developed by Dyer.]
McKoon G and Ratcliff R (1992) Inference during reading. *Psychological Review* **99**(3): 440–466.
ter Meulen AGB (1995) *Representing Time in Natural Language*. Cambridge, MA: MIT Press.
O'Rorke P and Ortony A (1994) Explaining emotions. *Cognitive Science* **18**: 283–323.
Ram A and Moorman K (eds) (1999) *Understanding Language Understanding*. Cambridge, MA: MIT Press.
Schank RC and Riesbeck CK (1981) *Inside Computer Understanding*. Hillsdale, NJ: Erlbaum.
SNePS. [ftp://ftp.cse.buffalo.edu/pub/sneps.] [The SNePS program.]
Thought Treasure. [http://www.signiform.com/tt/htm/tt.htm.] [The Thought Treasure program and database developed by Mueller.]

Stress

Advanced article

Harry van der Hulst, University of Connecticut, Storrs, Connecticut, USA

CONTENTS

Stress is a phonetic manifestation of accent, which marks the phonological head of the word.

INTRODUCTION

In this article, I will discuss the phenomenon of linguistic stress as it applies to words. Units that are larger than words (such as phrases and sentences) can be said to have stress too, but I will not touch on these larger units here. Right away, in the very next section, I propose to shift our attention to the notion of accent, which I define as more fundamental than stress. Stress, as we shall see, can be seen as a phonetic manifestation of accent. I will provide a typology of the various ways in which accent is manifested besides through stress.

MANIFESTATIONS OF ACCENT

It seems prudent first to make an attempt to define what stress is, or, at least, how I will use the term in this article. Starting with what most people who are able to read this article know (i.e. people who know English), let us consider the following pairs of English words:

convíct	cónvict	
protést	prótest	
pervért	pérvert	(1)

If one pronounces these words, pairwise, one will notice a difference that seems to involve the (relative) prominence of the syllables that the words are composed of. Let us capitalize the prominent syllables:

conVÍCT	CÓNvict	
proTÉST	PRÓtest	
perVÉRT	PÉRvert	(2)

Stress, as I will define it, is (relative) syllable prominence. It is now fair to ask what is meant by 'prominence'. This brings us into the realm of phonetics, i.e. the study of the way speech is produced and perceived. Relative prominence corresponds, on the one hand, with greater articulatory effort in production and, on the other, with greater salience, or audibility, on the perceptual side. Stressed syllables, then, stand out and are easier to perceive than the unstressed, or lesser stressed syllables. Greater articulatory force can be the cause of several effects that can be measured by investigating the details of production, or the physical properties of the produced acoustic signal, e.g.:

Phonetic properties of stressed syllables
- The stressed syllable has greater duration
- The stressed syllable is louder (greater amplitude)
- The stressed syllable is pronounced at a higher pitch (higher fundamental frequency)
- The segments are pronounced with greater precision (3)

This list is not meant to be finite, nor is it couched in the latest language of the trained phonetician. Also, some or all of the phonetic properties may be exclusively or primarily manifested in only a part of the syllable such as its vowel or its rhyme. Whatever the details, a stressed syllable will differ from unstressed syllables in having 'more' of whatever 'stretchable' property any syllable may have (such as duration, pitch, loudness, manner of articulation).

Following researchers such as Hyman (1977), I propose to reserve the term 'stress' for prominence as signaled by the above collection of cues. Then, I will also follow these researchers in saying that stress, in the sense just defined, is a phonetic manifestation or exponent of an abstract property, accent.

Before we address the question of how accent is to be formally understood, let us include another language in the discussion, namely Safwa (Bantu). Consider the following words or word combinations:

a'mi-ino	'teeth'	
ga'mi-ino	'the very teeth'	
mi-ino'	'it is teeth'	
inko'ombe i'im-bisi	'uncooked beans'	
inko'ombe m-bisi'	'the beans are uncooked'	(4)

Again, I have provided certain vowels with what is often (and appropriately) called an 'accent mark'. As in the case of English, speakers of Safwa perceive the syllables that contain these accented vowels as more prominent than the surrounding syllables. When we now look at the articulatory and acoustic properties of the vowels in question, it turns out that what distinguishes them from other vowels in the word is just (or mainly) their relative higher pitch. Thus, the relevant vowels are singled out by only one of the properties that cue the presence of accent in English. But if 'stress' is the collection of all the properties in (3), we cannot say that Safwa has stress. So what *do* we say? The obvious answer may be that Safwa has pitch. We can now capture the difference between English and Safwa terminologically by referring to the former as a stress-accent language and the latter as a pitch-accent language, as proposed in Hyman (1977).

Before we discuss the matter of accent locations, it will also be important to see that the accents can be cued by phonological properties instead of, or in addition to, nondistinctive phonetic cues, although the line between what is called 'phonetic' and what is called 'phonological' lies in different places for different researchers. One important way in which word accents can reveal themselves is by function as anchoring points for some of the tones that make up the intonation melody. (Because these tones, being pitch events, link to word accents, researchers often refer to them as 'pitch-accents', not to be confused with Hyman's notion of pitch-accent introduced earlier.) An intonation melody in a language like English consists of one or more 'pitch-accents' (which can be high tone, low tone,