When Should A Cheetah Remind You of a Bat? Reminding in Case-Based Teaching

Daniel C. Edelson
Institute for the Learning Sciences
Northwestern University
Evanston, IL 60208

Abstract

Case-based teaching systems, like good human teachers, tell stories in order to help students learn. A case-based teaching system engages a student in a challenging task and monitors his actions looking for opportunities to tell stories that will assist the learning process. In order to produce stories at the appropriate moment, a case-based teaching system must have a library of stories that are indexed according to how they should be used and a set of reminding strategies to retrieve stories when they are relevant. In this paper, I discuss CreANIMate, a biology tutor that uses stories to help teach elementary school students about animal morphology. In particular, I discuss the reminding strategies and indexing schemes that enable the system to achieve its educational objectives. These reminding strategies are *example remindings*, *similarity-based remindings*, and *expectation violation remindings*.

Introduction

Good teachers are good story tellers. This fact is the inspiration for a new architecture for computer-based educational systems known as case-based teaching[1] (CBT) systems. A case-based teaching system presents stories to help a student learn. Like a good human story teller who, in addition to being a master of delivery, knows the right story to tell at the right moment, an effective case-based teaching system must be able to evaluate a student's situation and identify an appropriate story to help the student learn from that situation. In this paper, I describe the architecture of case-based teaching systems and discuss a system called CreANIMate, which uses stories to help teach animal morphology to elementary school children. In particular, I focus on the reminding strategies of the system.

[1]The term *Case-based Teaching* has been used to describe any type of teaching that makes use of cases (Cognition and Technology Group, 1990; Gragg, 1940). In this paper, CBT refers to a specific class of teaching systems described by Schank (1991a).

The Case-based Teaching Architecture

People learn well when they are engaged in a task that interests and challenges them. A case-based teaching system engages a student in a task that will provide him with rich opportunities to learn. The system capitalizes on these opportunities by presenting stories to the student that help him to learn from his situation. Thus, the student learns both from his interactions with his task and from stories that he encounters as a result of his interactions.

To provide this learning environment, a case-based teaching system consists of two interdependent components, a *task environment* and a *storyteller*. The task environment presents the student with a motivating, challenging task. Typically, a task environment consists of a simulation, a problem-solving environment, or an interactive dialogue. While the student is interacting with the task environment, the storyteller monitors the student's actions looking for opportunities to present stories that will assist his learning. Stories can take the form of advice from experts, narratives describing personal experience, or depictions of actual situations from the domain under study (for more detail, see Schank, 1991a). Multimedia technology makes it possible for a case-based teaching system to present stories in a variety of forms, including video, animation, and text. A case-based teaching system can even improve on the capabilities of a human teacher because of its ability to instantly retrieve stories from very large story-bases on mass-media storage devices.

The primary justification for teaching with stories, besides the observation that good teachers use them (Schank, 1991b), comes from the theory of case-based reasoning (Kolodner et al., 1985; Riesbeck & Schank, 1989). Since people have been observed using case-based reasoning in a variety of situations as diverse as firefighting, medicine, and architecture (Kolodner, 1991), it follows that an important way to support this style of reasoning is to provide students with cases. Uncovering empirical evidence to support this claim is one

aspect of the ongoing research in case-based teaching.

When a person acquires a collection of cases for a domain, the value of this case library is limited by that person's ability to recall useful cases when they are relevant. Thus, a case-based teaching system must provide a student with an appropriate organization for his case library. The way an individual retains a story is influenced by the context in which he hears the story and by his previous knowledge and experience. A case-based teaching system assists the student's interpretation and integration process by presenting stories in context. That is, the system only presents a story to a student when the information in the story is directly relevant to the student's situation. This enables the student to index the story in his memory with respect to the context in which he sees the story.

In building a case-based teacher capable of presenting stories in this way, two important research issues emerge:

Vocabulary. What is the appropriate vocabulary for indexing stories?

Reminding strategies. What are valuable strategies for retrieving stories.

An important advantage of the case-based teaching architecture is that it enables a system to present appropriate stories to the student without requiring that the system understand the contents of the stories fully. In the same way that a case-based reasoner is able to retrieve cases before it understands exactly how it will use them, a case-based teaching system is able to present stories without understanding them as completely as a student will. Both case-based architectures are able to perform this way because they have cases that are indexed according to situations in which they are useful, as opposed to being indexed according to their content. Traditionally intelligent tutoring systems (ITS's) have needed to understand all of the information that they want to impart to the student. This complete knowledge enables systems to identify "buggy" behavior, missing information, or misconceptions on the part of the student, e.g., MENO-II (Soloway et al. 1983) WEST (Burton & Brown 1976;), GUIDON (Clancy 1987). A case-based teaching system sacrifices some of this ability in favor of ease of scale-up and a more expressive knowledge communication[2] strategy.

To summarize, a case-based teaching system uses a task environment which engages and challenges the student to allow the student to encounter situations that are rich in opportunities for learning. At the same time, the storyteller component of the system monitors the student's situation looking for opportunities to present stories that will assist the student's learning process. The context in which the student sees the stories helps that student to interpret and integrate those stories in a way that will assist him to reason from cases in the future.

[2]This term is from Wenger (1987)

The CreANIMate System

The CreANIMate program is a system designed to teach elementary school children about animals, their physical features, and how they survive in the wild. It helps them to understand the important connections between the way that an animal looks, the way it behaves, and how it survives. To engage a student, CreANIMate invites him to create a new animal by taking an existing animal and modifying it in some way. For example, a student might request a gerbil with large claws. The task of creating a new animal was selected because of its inherent appeal, because it encourages creativity, and because it provides many opportunities for learning. A student has the opportunity to learn about an animal by modifying it in much the same way that a scientist studies a system by perturbing it from its natural state and observing the ramifications of that perturbation.

Once the student proposes an animal, the program initiates a dialogue in which the student considers the viability of his animal. For example, the program might conduct a discussion of why it might be helpful for a gerbil to have large claws, or how gerbils use the paws that they currently have. The discussion is accompanied by video clips of actual animals in the wild that illustrate relevant principles. The program might show clips illustrating how different animals use their claws or how certain types of claws are especially well suited for specific purposes. The underlying lesson of the interaction is the relationship between the physical features of animals and the ways in which the animals use those features to help them to survive in their environment. Which examples of these basic principles a student sees is determined entirely by the particular student's interests.

The CreANIMate task environment consists of a question-and-answer dialogue in which the system helps the student to explore the ramifications of changing some aspect of an animal. Each CreANIMate dialogue is based around a question that is fundamental to understanding the ways in which animals survive in the wild. We call these questions *explanation questions*, in the terminology of Schank (1986). Explanation questions are the questions in any domain that a knowledgeable individual asks to construct an explanation for a phenomenon in that domain. Some explanation questions in animal morphology are, "Why is it useful for this animal to perform a particular action?" and "How does this animal use a particular feature to help it survive?" We emphasize explanation questions because these questions provide the framework for representing a domain. The questions that underlie a domain provide the structure that is critical for indexing cases as well as for explaining new observations. When a student sees a story in the context of

an explanation question, that question helps to emphasize the features that the student can use to index that story in his own memory. The goal of the CreANIMate task environment is to conduct dialogues that cover as many explanation questions as possible. The centrality of questions in CreANIMate follows in the tradition of GUIDON and WHY (Stevens & Collins 1977). Edelson (1991) discusses in greater detail the use of questions in an instructional dialogue. The following short transcript of the program's operation gives a sense of the dialogues it conducts:

```
Suppose you could create a new animal by
taking an existing animal and changing it
some way... What would you make?
STUDENT> a butterfly that can fight
Here is your animal before we change
it...[Shows a picture]
OK, but first we need a reason for your
butterfly to fight. Is there a reason you
want your butterfly to fight?
STUDENT> so it can protect itself
That's a good idea. Kangaroo rats kick dirt
to help them defend themselves against
predators. Would you like to see that?
STUDENT> yes
[VIDEO: Kangaroo Rat Kicks Dirt at Snake]
I know of some other animals that defend
themselves against predators, but they don't
fight. For example, some schools of fish
use the help of others so they can defend
themselves against predators.
Would you like to see that?
STUDENT> yes
[VIDEO: Shark Protects School of Fish]
Kangaroo rats are not the only animals that
fight in order to defend themselves. For
example, bees fight to defend themselves.
Would you like to see that?
STUDENT> no
So, one reason a butterfly might fight is to
defend itself. Why would you like your
butterfly to fight?
STUDENT> Show me more reasons.
```

As the transcript shows, the student begins the dialogue by selecting a modification for an animal, and the program responds by asking an important question about that animal, e.g. "Is there a reason you want your butterfly to fight?" In the ensuing dialogue, the program presents videos that illustrate potential answers to that question, finally giving the student the opportunity to commit to an answer to the question for his animal. The actual program has an appealing graphical interface in which the student responds to questions by selecting options with a pointing device such as a mouse and by typing in partial sentences. The current prototype, which contains more than 200 animals, 600 animal attributes, and 130 video clips, is being evaluated in use by fourth through seventh graders.

Reminding in CreANIMate

When a person thinks of a story to tell, we say that he or she is reminded of a story. We usually think of reminding as a passive process, something that happens to someone, not something a person does. Upon examination of the process, one finds a set of processes and representations devoted to actively extracting features from the world and using those features to index useful cases or stories (Schank et al., 1990). Since teaching is an expertise, teachers have a particular set of heuristics that enable them to observe their students and retrieve appropriate examples, explanations, and stories (three forms of instructional cases). In the CreANIMate system, we employ several reminding heuristics that enable the system to retrieve stories to achieve specific pedagogical objectives. Each reminding heuristic places specific demands on the information available in the indices of stories. In the remainder of this section, I describe three reminding strategies. For each strategy, I give an example from the operation of the system, a description of the indexing information that enables this type of reminding, and a description of the reminding algorithm itself.

Example Remindings

The bread and butter reminding for the CreANIMate system, just as it is for any teacher, is the example. For instance, in a dialogue in which the student asked for a tortoise that could run fast, the system introduced the explanation question, "What features could help a tortoise to run fast?" When the student asked for suggestions, the program responded:

```
Cheetahs run fast. Do you know what
cheetahs have to help them run fast? (I have
an awesome video about that.)
STUDENT> They have long legs.
That's right. Cheetahs have long, muscular
legs to help them to run fast. Would you
like to see that?
STUDENT> Yes.
```

In response to the question asking what features could help a tortoise to run fast, the system retrieved a story that shows how the long, muscular legs of cheetahs enable them to run fast. The system suggests answers to questions in the form of example video clips. The strategy for identifying explanation questions and retrieving examples resembles the *issues and examples* strategy of Burton & Brown (1976). This dialogue demonstrates an advantage of the case-based architecture over a system that contained the same type of knowledge but no cases to use as examples, e.g. WHY, GUIDON[3]. Such a system could only respond by saying, "Long, muscular legs can be used to run fast."

[3]GUIDON uses cases, but not as examples.

To accomplish these remindings, the system relies on specific information being available in an index. An index must detail the explanation questions that the story can exemplify. The relationship between physical features and the functions they support is one of the central lessons of the system. Therefore, one part of an index may indicate the use of a physical feature, e.g., long, muscular legs, for a function, to run fast, in the story. For every feature/function pair that appears in a video clip, there is a corresponding entry in the index indicating the presence of that pair in the story. The same is true for actions that are used to achieve a survival behavior, for instance, *to run fast to pursue prey*. Thus, part of the index for the cheetah story is[4]:

```
[INDEX CHEETAH-PURSUING-PREY-CHEETAH
 :ANIMALS ([ANIMAL CHEETAH])
 :FEAFUNS
  ([FEAFUN :FEATURE [FEATURE LONG-
                     MUSCULAR-LEGS]
    :FUNKSHUN [FUNKSHUN RUN-FAST]]...)
 :PLANS
   ([PLAN :FUNKSHUN [FUNKSHUN RUN-FAST]
     :BEHAVIOR [BEHAVIOR PURSUE-PREY]])...]
```

For each feature/function pair or function/behavior pair in an index, the system is able to use it as an example for two different explanation questions. For example, the feature/function pair, long, muscular legs to run fast, can be used as an example in either a dialogue about "What features must an animal have in order to run fast?" or "What can long, muscular legs be used for?" The algorithm for example reminding is relatively straightforward. Example reminding is always triggered by a request from the student. Suppose the student had requested a tortoise with long legs. First, the system recognizes that the student's request gives rise to the explanation question, "Why would it be useful for a tortoise to have long legs?" This initiates the example reminding process, which searches for ways animals use long legs. Since all of the objects in the system are linked in abstraction hierarchies, the system is able to recognize that a story about a subtype is also a story about its supertype. The system knows that [FEATURE LONG-MUSCULAR-LEGS] is a subtype of [FEATURE LONG-LEGS] so it is able to recognize that the cheetah story above is an example of a reason that animals have long legs. It could then show the cheetah story as an example of one reason animals have long legs–to run fast. To summarize, the example reminding algorithm simply consists of 1) identifying candidate explanation questions relevant to the student's request and 2) searching down abstraction hierarchies starting from the student's requested modification looking for indices that illustrate the explanation question.

[4]This is a concise printed representation of a complex data structure. The first word in a [...] form is the type of object being represented. Feature means physical feature, Funkshun refers to the action that a physical feature is used for. (It is spelled *funkshun* because *function* already names a Common Lisp data structure.) I also refer to funkshuns as *actions* in the text. Feafun refers to a feature/function pair. A Behavior is a high-level survival behavior and a Plan is a function/behavior pair. Each one of these objects is implemented in the system as a frame in a highly interconnected semantic network.

Similarity-based Remindings

One of the risks of teaching with examples is that the student may draw overly specific conclusions from the examples that they see. Therefore, one objective of a case-based teacher is to assist the student in drawing inferences at the right level of abstraction. The strategy that CreANIMate uses to help the student form appropriate generalizations is called similarity-based reminding. In similarity-based reminding, the system retrieves a story that illustrates the same basic principle as a previous example, but is sufficiently different to allow the student to form a generalization at an appropriate level of abstraction. The following example of a similarity-based reminding was initiated by a student's request for a tortoise that could run fast:

```
Cheetahs run fast. Do you know why
cheetahs run fast?
(I have an impressive video about that. )
STUDENT> So they can catch other animals.
That is right. Would you like to see that?
STUDENT> yes
[VIDEO: Cheetah Pursuing Prey]
That reminds me of a cool video. Fishing
bats also move fast in order to get food.
Only, instead of running fast to pursue
their prey, they fly to pounce on their
prey. Would you like to see that?
STUDENT> yes
[VIDEO: Fishing Bat]
```

In this example, the program presented a video of a cheetah that runs fast to pursue its prey. This story was produced as an example of a reason that animals run fast. However, to ward off the possibility of the student drawing an overly-specific conclusion, e.g., "Animals always pursue their prey by running fast," the program presents a similar story about an animal that moves fast to get its food, but instead of running fast, it flies. In order to perform similarity-based reminding, the system must be able to identify stories that are similar, but not identical, to the given example. This is done by adding abstraction information to every plan or feafun in an index. The following is a portion of the index for the cheetah story in the example above:

```
[INDEX CHEETAH-PURSUING-PREY-CHEETAH
  :PLANS
   ([PLAN :FUNKSHUN [FUNKSHUN RUN-FAST]
      :BEHAVIOR [BEHAVIOR PURSUE-PREY]
      :ABSTRACTION
        [PLAN :FUNKSHUN [FUNKSHUN MOVE-FAST]
         :BEHAVIOR [BEHAVIOR HUNT]]]..)...]
```

The plan, *run fast to pursue prey*, is annotated with the abstraction *move fast to hunt*. The system uses this information to help it identify stories that are suitable similarity-based remindings for the cheetah story. The system understands the abstraction *move fast to hunt* to mean that if another story contains a plan that falls under the abstraction *move fast to hunt*, then that story is an appropriate similarity-based reminding[5]. The story about the fishing bat which flies to pounce on its prey is retrieved because *move fast to hunt* is an abstraction of *fly to pounce on prey*.

In the current example, the explanation question is "Why might it be useful for a tortoise to run fast?" The cheetah's running fast to pursue prey is presented as a possible answer for that question. Other possible answers will follow. Since the student is at risk for concluding that the only way to pursue prey is to run fast, the similarity-based reminding tries to find things that are similar to *running fast* that will help the student form an appropriate generalization. To do so, the similarity-based reminding algorithm performs an ever-widening search starting with running fast looking for actions that are like *running fast* in support of behaviors like *pursue prey*. The search is restricted by the abstraction in the index so that it will only accept an index that is encompassed by the abstraction *move fast to hunt*.

Thus, similarity-based reminding allows the program to present stories that help a student to generalize a principle up to an appropriate level of abstraction. In order to do so, stories are not indexed just according to the specific principle the story illustrates, but to an appropriate abstraction of that principle.

Expectation Violation Remindings

The greatest opportunity for learning takes place when an expectation that you have is violated by experience. This is what Schank (1982) calls failure-driven learning. In this case, failure refers to the failure of an expectation to explain an observation, not the failure of an individual to achieve a goal. The failure of an expectation to explain an observation triggers learning. A case-based teaching system attempts to capitalize on expectations that a student might have in order to promote failure-driven learning.

Expectation failures do not just promote learning they provoke interest. In general, stories are interesting to the extent that they challenge our expectations rather than confirm them. An experience that conforms exactly to expectation is boring. Thus, while CreANIMate is unable to judge what the student's actual expectations are, it is still assured of capitalizing on the student's interests with expectation violation remindings.

In order to perform expectation-violation remindings, the system must contain information about what expectations a student might have. An indexer adds this information when entering stories into the system. These expectations are entered as one of three types of rule: 1) **Only-rules** e.g., "Only mammals have hair." 2) **No-rules**, e.g., "No mammals lay eggs." 3) **All-rules**, e.g., "All birds fly." For representational purposes, we categorize expectations as either exclusive or inclusive expectations. Exclusive expectations are based on the exclusion of some animal from some category and inclusive expectations are predicated on inclusion in the category. Only-rules and no-rules lead to exclusion violations and all-rules lead to inclusion violations. Because of space limitations I am only able to show the inclusive reminding strategy here. While the rest of the information in the system's knowledge base is considered to be true under all circumstances, the information expressed in these rules is treated simply as likely student expectations.

Inclusive Expectation Remindings Inclusive expectation remindings are triggered by all-rules. In the following example, the reminding was triggered by the expectation that all birds flee predators by flying.

```
Quail fly to flee predators.  I have a
dramatic video about that.  Would you like
to see that?
STUDENT> yes
[VIDEO: Hawk chases quail]
Did you know that not all birds fly to flee
predators?  Do you know how ostriches flee
predators?
(I have an awesome video about that.)
STUDENT> They run.
Yes, that is right.  Ostriches run fast to
flee predators.  Would you like to see that?
STUDENT> yes
[VIDEO:  Ostrich runs fast]
```

This default expectation is entered in the system as the following all-rule:

```
[ALL-RULE :ANIMALS [ANIMAL BIRD]
   :VALUE [PLAN :FUNKSHUN
                [FUNKSHUN FLY]
                :BEHAVIOR
                [BEHAVIOR FLEE-PREDATOR]]
   :EXPECT-VIOL [FUNKSHUN FLY]
   :INDEX [INDEX OSTRICH-RUNS-FAST-OSTRICH]]
```

This rule reads, "Expect that all birds fly to flee predators. In the ostrich story, this expectation is violated by the substitution of something else for *to fly*." The reminding strategy is triggered by a story which exemplifies the expectation–a story of a quail flying to flee a hawk. When the system presents the first story, the inclusion reminding algorithm initiates a search for any inclusive expectations about flying and fleeing predators. In this case, it finds the above all-rule associated with the action *to fly* and finds that the animal in the initial story, the quail, fits the requirement of inclusion in the category *bird*. The storyteller

[5]The idea for using abstractions this way originated with Richard Osgood (1990)

then takes advantage of the opportunity to present the reminding.

To summarize, remindings from expectation violations are designed to capitalize on students' default expectations. They are triggered by rules. When the system presents a story, it determines whether there is a relevant rule by searching the hierarchy for rules that correspond to the elements of the current story. If there is such a rule, and the animal in the current story corresponds to the category specified by the rule, then the storyteller presents the story.

Conclusion

The case-based teaching architecture is designed to take advantage of the way people naturally learn and reason from stories. A case-based teacher presents a student with an engaging task that provides rich opportunities for learning. Like a good human teacher, a case-based teaching system is able to capitalize on a student's situation in order to present appropriate stories to further the student's learning. Unlike a human, a computer-based system is able to draw instantly from an extremely large database of stories, recorded in a variety of media from graphics to video.

To be effective, however, such a system must be able to retrieve the right story at the right time. In the course of developing CreANIMate, we have developed a knowledge representation and a collection of reminding strategies that enables the system to retrieve stories to achieve specific educational objectives. Currently, these strategies include: 1) example reminding, to provide examples; 2) similarity-based reminding, to assist in generalization; 3) and expectation violation, to challenge students' expectations. Each of these reminding strategies is activated by a particular context in order to provide the student with a story that is directly relevant to his situation. However, these reminding types are just the beginning. Future research will explore reminding strategies that include counterexamples, extremes, opposites and others.

Acknowledgements

This work is supported in part by a grant from the IBM Program for Innovative Uses of Information Technology in K-12 education, and by the Defense Advanced Research Projects Agency monitored by the Office of Naval Research under contract N00014-91-J4092. The Institute for the Learning Sciences was established in 1989 with the support of Andersen Consulting, part of the Arthur Andersen Worldwide Organization.

The CreANIMate team includes, in addition to the author, the following indexer, programmers, and grad students: Bob Kaeding, Ken Greenlee, Riad Mohammed, Diane Schwartz, John Cleave, and Will Fitzgerald. I would like to thank Andy Fano, Vivian Choy Edelson, and the anonymous reviewers for helpful comments on earlier drafts of this paper.

References

Burton, R.R., & Brown, J.S. 1976. A Tutoring and Student Modeling Paradigm for Gaming Environments. In Colman, R., and Lorton, P. Jr. (Eds.) *Computer Science and Education. ACM SIGCSE Bulletin,* 8(1), 236-246.

Clancey, W. J. 1987. *Knowledge-Based Tutoring: The GUIDON Program.* Cambridge, MA: MIT press.

Cognition and Technology Group at Vanderbilt, The. 1990. Anchored Instruction and Its Relationship to Situated Cognition. *Educational Researcher* 19(6), 2-10.

Edelson, D. C. 1991. Why do Cheetahs Run Fast? Responsive Questioning in a Case-Based Teaching System. In Proceedings of the International Conference on the Learning Sciences, 138-144. Charlottesville, VA: Association for the Advancement of Computing in Education.

Gragg, C. I. 1940. Because Wisdom Can't Be Told. Harvard Alumni Bulletin, 78-84.

Kolodner, J. L. 1991. Improving Human Decision Making through Case-Based Decision Aiding. *AI Magazine* 12(2): 52-67.

Kolodner, J. L., Simpson, R. L. and Sycara-Cyranski, K. 1985. A Process Model of Case-Based Reasoning in Problem-Solving. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence.

Osgood, R. 1990. Personal Communication.

Riesbeck, C.K., & Schank, R.C. 1989. Inside Case-Based Reasoning. Hillsdale, NJ: Lawrence Erlbaum.

Schank, R.C. 1982. *Dynamic Memory.* Cambridge: Cambridge University Press.

Schank, R.C. 1986. *Explanation Patterns: Understanding Mechanically and Creatively.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Schank, R.C., Osgood, R. et al. 1990. A Content Theory of Memory Indexing. Technical Report #2, Institute for the Learning Sciences, Northwestern University.

Schank, R. C. 1991a. Case-Based Teaching: Four Experiences in Educational Software Design. Institute for the Learning Sciences Technical Report #7, Northwestern University.

Schank, R.C. 1991b. Tell me a story : A new look at real and artificial intelligence. New York: Simon and Schuster.

Solowary, E.M., Rubin, E., Woolf, B.P., Bonar, J., and Johnson, W.L. 1983. MENO-II: An AI-based Programming Tutor. *Journal of Computer-based Instruction* 10(1): 20-34.

Stevens, A.L., & Collins, A. 1977. The Goal STructure of a Socratic Tutor. *Proceedings of the National ACM Conference,* Seattle, Washington, 256-263. New York: Association for Computing Machinery.

Wenger, E. 1987. *Artificial Intelligence and Tutoring Systems.* Los Altos, CA: Morgan Kaufmann Publishers.