Reading as a Planned Activity

Tamitha Carpenter
tamitha@chaos.cs.brandeis.edu
(617) 736-2718

Richard Alterman
alterman@chaos.cs.brandeis.edu
(617) 736-2703

Brandeis University
Department of Computer Science
415 South Street
Waltham, MA 02254

Abstract

This paper discusses how the goals of a reader lead to reading as a *planned* activity. The external situation provides many of these goals, as well as background, which allow the reader to focus the reading activity. A system is described in which a plan for perusing a given piece of text is chosen with regard to the reading goal(s) and the structure and content of the text. As reading progresses and goals are satisfied or change, the plan is modified using adaptive planning. This model is supported by a protocol study of subjects reading instructions to use a fax machine for the first time. The main sections of this paper will: 1) describe reading in terms of a larger context of activity; 2) introduce the necessary components needed for a reader that plans; and 3) illustrate planned reading in the domain of device instructions.

Introduction

When reading, the reader very often approaches the text with a goal in mind. In the case of fiction, such as folktales, the goal is to understand the motivations of the characters and the moral being conveyed, and perhaps to appreciate the humor. When reading a textbook, the goal is to understand an unfamiliar concept or to complete a task assigned by an instructor. The way the reader approaches a text depends on his goals (cf. van Dijk & Kintsch 1983 p. 53).

This paper will discuss how the goals of the reader lead to reading as a *planned* activity. By choosing a known plan, and adjusting that plan as reading progresses and goals are satisfied, reading becomes more efficient and better suited to the larger context in which it takes place.

This idea of reading as planned activity has rarely been addressed by computational models of reading, even when the goals of the reader are considered. For example, the AQUA system (Ram, 1991) explicitly deals with reading goals, using the goals to control what AQUA considers relevant and interesting. However, the text is still read in the "read-straight-all-the-way-through" nature of traditional readers. Goldman & Saul (1990) deal with non-linear reading strategies in great detail, although these strategies are at the level of sentences and are concerned with when and how rereading occurs. While both systems focus on important characteristics of the reading task, this paper focuses on a more global approach and the role of planning when satisfying a reading goal.

A computational model is under development to use and adapt plans for reading, which we will refer to as SPRITe (using Structure to Plan to Read Instructional TExt). This design is inspired and supported by a protocol study for using a Fax machine conducted in Spring of 1991. SPRITe uses the method of *adaptive planning* (Alterman 1988), a case-based reasoning technique that adapts old plans to new situations. SPRITe's plans are indexed by the type of information needed and the layout of the text, and can be modified as the reading task progresses. As the plans are used to meet the reading goals, they take advantage of several levels of content and structural details (e.g., the basic document structures, common writing tools such as lists).

The main sections of this paper will: 1) describe reading in terms of a larger context of activity; 2) introduce the necessary components needed for a reader that plans; and 3) illustrate planned reading in the domain of device instructions.

Domain

SPRITe is part of a larger system known as FLOABN (Alterman et al., 1991), a simulated agent working with and learning about mechanical and electronic devices. This domain is ideal because the agent approaches the instructions with

a goal, rather than as an isolated reading process. The planner which controls FLOABN's interaction with the device is called Scavenger (Zito-Wolf, *forthcoming*). When Scavenger fails due to a lack of knowledge, the instructions work as a backup knowledge source which SPRITe consults so that FLOABN can proceed.

One of the characteristics of FLOABN is the interaction between activity and reading. While SPRITe, like most traditional models of reading, must establish textual *coherence* (the connection of the meaning of text to the other text around it), SPRITE must also establish coherence between the text and the activity. In the case of instructions, this means that even if the text makes sense internally, if the connection between the text and the device cannot be established, the instructions are useless because they cannot be applied. Currently, coherence is established using a modification of the marker passing techniques described in Norvig (1989). SPRITe's knowledge source is a semantic memory, built semi-automatically, in which basic concepts (e.g., hierarchy for types of money) are combined with representations of approximately 50 devices.

The Protocol Study

The model we describe in this paper has been greatly influenced by a protocol study conducted in Spring 1991. Nine people, graduate and undergraduate students from four different departments in the university, were studied while using a fax machine and its instructions for the first time. This protocol study will be reported in more detail in a future article.

Two results of the study that will be used in this paper are that 1) the subjects interleaved interaction with the device and reading of the instructions; and 2) the subjects approached the instructions in non-linear, although still systematic ways. In general, this meant that a subject would examine and possibly try to use the fax machine, and then would turn to the instructions with a goal to satisfy ("what do these buttons do?" or "I've dialed the number. What next?"). The goal influenced the way the subject navigated the instructions, which indicated that the subject had a plan for satisfying that goal, although the plan was not necessarily complete. In addition, as the subjects became more familiar with the structure of the text, the methods they used changed. This showed that the subjects were building representations of the instructions, and their familiarity with the text changed the plans they were using.

Plans for Reading

The process of using instructions to aid in the use of devices has several stages. First, the device is encountered and identified to some extent. Second, the problem area is determined through interaction with the device -- e.g., an attempted step failed for some reason. Third, the instructions are located and identified. In the final stage, the instructions are studied and, if successful, a solution to the problem is read and understood.

The first two stages of this process are executed by Scavenger. For the extent of this paper, the third stage will be overridden by the assumption that the instructions are readily available and obviously applicable to the current device. This assumption can be defended by the fact that, very often, the instructions are either printed on the device or are stored with the device. (A technique for choosing from a set of possible texts will be explored in the future.)

This section will describe the final stage in detail. The process of studying the text will be accomplished with adaptive planning. The plans take advantage of standard structures (e.g., enumerated lists) common to instructional texts (i.e., *superstructures*: van Dijk & Kintsch, 1983) and other formatting methods (e.g., boldface) as extra clues to the textual content, to navigate the text, and as features for indexing the cases. The technique for establishing coherence, as described in the *domain* section, is the most basic step of each plan.

To begin, the reading planner must have a core of plans and sub-plans. Table 1 catalogues several methods, culled primarily from the protocol study described in section 2, which provide building blocks for reading plans. The entries under **Level** indicate at what level the method operates, either the global level of document or sections (*superstructure*) or the more local level of phrases, sentences, and paragraphs (*structure*). Possible future levels might be *selection* for choosing appropriate documents, and *eye-movement* for emulating Goldman & Saul's (1990) findings on the movements routinely used by the human eye when reading.

For human readers, the structure of a piece of text is, to a large extent, visually identified. For the purposes of building an automated system for reading and interpreting instructions, we have chosen to represent the structure of the input text using the Latex (Lamport 1986) formatting language. We have chosen this representation for two main reasons: 1) many easily available documents are already in Latex form; and 2) it is

Table 1: A few general methods for different levels of reading. The "Time" column is marked if the method may be used when time is short. *structure* is a list used to store the elements of the text's structure. Words beginning with a '\' are latex commands.

Reading Goal	Level	Method	Time
Determine structure of text	super-structure	for each *x* in the top-level of *text* if *x* is a latex command push *x* onto *structure*	X
Use text structure to locate relevant text	super-structure	if \tableofcontents is in *structure* then **examine-toc** "file.toc" else if \index is in *structure* then **examine-index** "file.idx" else if \section is in *structure* then find most relevant section :	
Skim a section of text	structure	Recursively search section for \figure, \subsection, or list structures. If found, **skim** as appropriate.	X
Skim a sentence	structure	Look at primary subject-verb or verb-object pair in *sentence*. If sentence is compound, **skim** each sub-sentence.	
Skim a list	structure	If initial text does not begin with \item (i.e., heading of list) then **skim-sentence**. If heading isn't coherent with goals, abandon list.	X
		Determine if list-type fits goal (e.g., number list fits for finding a plan step). If so, **skim** each item.	X
Remember structure	super-structure	When especially relevant text is found (e.g., description of device parts), remember location for later use.	X

Table 2: Example abstract plans especially suited to reading instructions.

Main Goal	Plan to achieve goal
Begin reading	If this is a new episode on a new device then Choose document and Determine its structure else Remember previous episode(s) and their relevant document and structure Repeat Choose reading plan Use reading plan Until reading goal is satisfied Return results to Scavenger Remember successful steps as a plan case **or** Adapt plan using this reading experience
Find steps - long text	If document has table of contents scan table of contents for related section Find the section. Look at top level for lists, especially enumerated lists. If a list is found, **skim**. otherwise, **skim** entire section.
Find steps - short text	If document is 2 pages or less, look *structure* for lists, especially enumerated lists. If a list is found, **skim**. otherwise, **skim** entire document.
Identify device part	Check remembered text for descriptions of device parts. If found, go to location and **skim**. When a matching part is found, return name. Otherwise, recursively **determine structure** of text **Skim** figures or lists, especially bullet lists.

fairly straight-forward to represent and to interpret the most common text structures. In Table 1, the Latex commands all begin with a '\' character.

Table 2 gives a few abstracted examples of how some of the methods in table 1 are used. Other methods, like **remember structure** work more like daemons, waiting in the background until certain conditions (e.g., "especially relevant text is found") occur, then executing and retreating once again to the background.

The first abstract plan shown in table 2 is always executed when SPRITe begins. This plan initializes the text and then chooses another plan to actually use the text. When the reading plan is finished, the first plan informs Scavenger of the result and then executes some "clerical" work.

This clerical work consists of using the reading episode that just occurred to create a case or, if a previous case was used in lieu of a plan, to adapt the case. A "case" for this system is the series of successful reading steps that were used. For example, one plan might be "use the section headings to locate some relevant text, then skim the text". When this is executed, the heading itself might be sufficient to satisfy the goal. The resulting case would consist of "read section headings". This new case would then be indexed in the same way as the plan from which it was developed, plus or minus any detectable features, including the device and the instruction set being used, which distinguish the new case.

Adapting a case can happen in two ways. If a case is chosen instead of a plan, but fails, some standard adaptation techniques are tried. If this fails as well, another plan or case is tried. Then the successful execution steps are remembered and compared to the first case. If the differences are minor, the case is adjusted and stored with the same name and indices. Otherwise, a new case is developed and the indices of the original case are adjusted.

Example

This section is meant to illustrate how SPRITe works given a set of plans and a problem to solve. The device chosen for this example is the Airfone®, a telephone found on many airplanes. The instructions for the Airfone® are somewhat less than a page of text, and are shown in figure 1 in latex form with parentheses illustrating the hierarchical structure of the latex commands.

When execution begins, Scavenger has attempted to use the Airfone® using its plan for a standard pay telephone. When this fails and adaptation yields no progress, Scavenger asks for help from the instructions. The resulting interpretation process is shown in figure 2.

Since the Airfone® instructions are short and contain no table of contents, SPRITe chooses the second plan (from table 2) for finding a step which Scavenger can use. Upon examining the top level (that is, the first element of each top-level latex command in the instructions shown in Figure 1), one list structure is found. If no lists had been found, SPRITe would have tried another plan, or as a last resort, would have skimmed the entire document until relevant text was found.

SPRITe notes the location of the list in memory so that, if these instructions are used in the future, the step of scanning for lists will be unnecessary. Next, SPRITe tries to identify the contents of the list, but since it has no heading, SPRITe proceeds to the next step -- skim the elements in the list. By skimming the first element in the list, SPRITe finds that "insert credit card" is coherent with one of Scavenger's failed steps - "insert coin".

```
((\title ([AIRFONE SYSTEM OPERATING INSTRUCTIONS]))
 (\it([The Airfone system accepts these major credit cards.]))
 (Cain Travel Card, American Express, Carte Blanche, Diners Club
        International, Discover, enRoute, MasterCard, Visa.)
 (\begin([enumerate])
  (\item(Insert credit card as shown.) (Face up with card name to the
        right.))
  (\item(Lower door handle over card.) (Note: door will remain locked
        until handset is replaced.))
  (\item(Observe lighted display above for instructions.)
     (When instructed \(approx. 10 sec.\) remove phone by firmly
        grasping top of handset and pulling out.)
     (Return to seat to place calls.))
  (\item(Press green DIAL TONE button and wait for dial tone.) (To
        place call, dial AREA CODE and NUMBER.))
  (\item(To end call, press red HANG UP button.) (To place additional
        calls repeat Step 4.))
  (\item(Return handset to wall unit from which it was taken.) (Insert
        heel first as shown, then push top in firmly.))
  (\item(Observe lighted display above for instructions.) (Door handle
        will raise automatically in 10-15 seconds.)
        (Then remove card.))
  (\end([enumerate])))
 ([\center ([\large ([\bf (How To Place A Call Using The Airfone
        System.)])])])))
```

Figure 1: The Airfone instructions as loaded into memory.

```
Goal -- Use an Airfone
Related knowledge -- Scavenger chose "pay-telephone" action plan.
                     Scavenger tried to execute "lift receiver" -- failed
                     Scavenger tried to execute "insert coin" -- failed due to "no coin slot found"
Reading goal -- Find first step
Determine structure of document "Airfone-instructions" --
***structure consists of: emphasized text, plain text, enumerated list, and very emphasized text.
Choose reading plan -- reading goal matches "Find steps"
        -- Document has no table of contents & is less than 2 pages in length.
***Plan chosen is "Find steps - short text"
Checking previous experience for list structures -- none found.
Scanning top level for list structures.
***Found one list structure "\begin{enumerate}" -- Remembering location of this structure for future use
Skimming enumerated lists --
  Look for list heading -- no heading found.
Skimming items --
***Primary verb-object pair of first item is "Insert credit card"
***Coherent relationship between "Insert credit card" and "Insert coin" at payment.
***Tell Scavenger to replace "insert coin" with "insert credit card" and reorder steps with "insert credit card" before "lift receiver"
***Remember as "Find steps 001" successful steps "Scan top level for list structures;
                    Skim first list item"
```

Figure 2: Execution of Airfone example.

```
Scavenger executed "insert credit card"
Scavenger tried to execute "lift receiver" - failed.
Reading goal - Find next step.
Choose reading plan -- reading goal matches "Find steps 001"
        -- Document has no table of contents & is less than 2
           pages in length.
***Plan chosen is "Find steps 001"
Checking previous experience for list structures.
***Found one list structure "\begin{enumerate}"
Skimming first list item ---
***Primary verb-object pair of first item is "Insert credit card"
   Match with executed step -- no new information.
Plan failed.
Choose new reading plan -- reading goal matches "Find steps"
        -- Document has no table of contents & Document is
           less than 2 pages in length.
***Plan chosen is "Find steps - short text"
Checking previous experience for list structures.
***Found one list structure "\begin{enumerate}"
Skimming enumerated lists --
  Look for list heading -- no heading found.
Skimming items --
***Primary verb-object pair of first item is "Insert credit card"
   Match with executed step -- no new information.
***Primary verb-object pair of second item is "Lower door handle"
   No match -- may have new information, because of following
           matching executed step.
Read second item --
***Second item is "Lower door handle over card."
   Relationship with executed step "Insert credit card" -- indicates
           continuation with prop.
***Tell Scavenger to insert "lower door handle" after "insert credit
           card" with permanent order relationship
***Remember as "Find steps 002" successful steps "Scan top level
           for list structures;
                Skim list items until no match to
           executed steps
                     Read list item"
      differentiated from "Find steps 001" by indexing as
           "continuation of episode".
```

Figure 3: Continuation of Airfone example.

```
Scavenger executed "insert credit card", "lower door handle", and
           "lift receiver".
Scavenger tried to executed "listen for dial tone" -- failed
Reading goal - Find next step.
Choose reading plan -- reading goal matches "Find steps 002"
        -- Document has no table of contents &
           Document is less than 2 pages in length.
           -- This is a continued episode
***Plan chosen is "Find steps 002"
Checking previous experience for list structures.
***Found one list structure "\begin{enumerate}"
Skimming items --
***Primary verb-object pair of first item is "Insert credit card"
   Match with executed step -- no new information.
***Primary verb-object pair of second item is "Lower door handle"
   Match with executed step -- no new information.
***Primary verb-object pair of third item is "Observe lighted
           display"
   No match to executed step.
Read third item --
***Third item is "Observe lighted display above for instructions."
   Reading instruction found. Try executing.
Reading lighted display -- no text on display.
Return to original instructions.
Continue skimming third item --
***Next primary verb-object pair of third item is "Remove phone"
   Match with executed step -- no new information.
***Fourth item is compound -- verb-object pairs of fourth item are
           "Press green DIAL TONE button"
           "Wait for dial tone"
   Match found between "Wait for dial tone" and failed step
           "Listen for dial tone"
***Tell Scavenger to insert "Press green DIAL TONE button"
           before "Listen for dial tone" with permanent
           order relationship
***Note repeated steps "Scan top level for list structure" and
           "skim 'insert credit card"
***Adjust "Find steps 002" to be "Scan top level for list
           structures;
          Skim list items until no match to executed steps
                or until match with failed execution
             If no-match then Read list item"

Scavenger execution completed successfully.
```

Figure 4: Conclusion of Airfone example.

SPRITe then suggests that Scavenger should replace the old step and with "insert credit card". In addition, the reading steps that executed successfully are remembered and stored as a case of the plan "Find steps - short text". If Scavenger has no further problems, the reading task is terminated. Otherwise, SPRITe can use the knowledge of the text it has already found to solve future difficulties.

Figure 3 shows a continuation of the Airfone example. Scavenger successfully uses the plan information found in the instructions, but is unable to continue execution. SPRITe, using the document and text structure determined at the beginning of the episode, chooses to use the case developed in the initial reading. Unfortunately, the case was too limited and is unable to find the needed information, so a new plan is chosen.

The new plan chosen is again "Find steps - short text". The execution of the plan is similar to the previous execution, except that two list items are skimmed. The second list item doesn't yield new information when skimmed, but since it follows a successfully executed step in an enumerated list, it is likely that this item contains the next step. When SPRITe reads the sentence in full, it finds a reference to the credit card from the previous step, which indicates that "lower door handle over card" is a continuation of the previous step and, therefore, *must* be executed immediately following the previous step. (This reasoning uses knowledge about the enumerated list structure in instructional text. This knowledge consists of recognizable *instruction patterns* that this structure creates.) SPRITe returns the step information to Scavenger and again remembers the successful reading steps as a new case.

Once again, Scavenger executes the new instructions, but is unable to continue with its plan. Figure 4 shows the conclusion of the Airfone example. This final episode with the Airfone instructions proceeds much like the previous, except that after executing the case, SPRITe notices that the new information was not found in the way that the case indicated, so the case is adapted to encompass the current execution. The case is then re-stored and execution is completed.

Conclusion

This paper has shown how reading can be approached as a planned activity. By using plans for reading, the model described is able to take advantage of two pieces of information traditionally ignored by computational reading systems: 1) the context in which reading takes place (i.e., the non-reading activity of the agent); and 2) the characteristic structure of the class of text (e.g., instructional text) and the formatting of the specific text being read. Furthermore, our protocol study illustrates that this planning model for reading in a larger context reflects the behavior of human readers more accurately than the sequential model.

The implementation of this model, SPRITe, develops plans to read a given set of instructions in the context of engagement with a particular device. By planning, adapting, and storing new cases, SPRITe builds a library of several concrete methods for reading which over time improve its proficiency both with the specific texts from which the methods were developed, and with the class of text in which the specific texts belong.

References

Alterman, R., Zito-Wolf, R., & Carpenter, T. (1991). Interaction, Comprehension, and Instruction Usage. *The Journal of Learning Sciences*, 1(3&4):273-318.

Alterman, R. (1988). Adaptive Planning. *Cognitive Science*, 12:393-421.

Goldman, S. & Saul, E. (1990). Flexibility in text processing: A strategy competition model. *Learning and Individual Differences*, 2(2):181-219.

Lamport, L. (1986). *Latex: A Document Preparation System*. Addison-Wesley.

Norvig, P. (1989). Marker passing as a weak method for text inferencing. *Cognitive Science*, 13(4):569-620.

Ram, A. (1991). A Theory of Questions and Question Asking. *The Journal of Learning Sciences*, 1(3&4):273-318.

van Dijk, T. & Kintsch, W. (1983). *Strategies of discourse comprehension*. Academic Press, New York, NY.

Zito-Wolf, R. (forthcoming). *Case-Based Representations for Procedural Knowledge*. PhD thesis, Brandeis University.