SUITOR: An Attentive Information System

Paul P. Maglio Rob Barrett Christopher S. Campbell Ted Selker

IBM Almaden Research Center
650 Harry Road, NWED-B2
San Jose, CA 95120

{pmaglio, barrett, ccampbel, selker}@almaden.ibm.com

ABSTRACT

Attentive systems pay attention to what users do so that they can attend to what users need. Such systems track user behavior, model user interests, and anticipate user desires and actions. Because the general class of attentive systems is broad — ranging from human butlers to web sites that profile users — we have focused specifically on *attentive information systems*, which observe user actions with information resources, model user information states, and suggest information that might be helpful to users. In particular, we describe an implemented system, Simple User Interest Tracker (Suitor), that tracks computer users through multiple channels — gaze, web browsing, application focus — to determine their interests and to satisfy their information needs. By observing behavior and modeling users, Suitor finds and displays potentially relevant information that is both timely and non-disruptive to the users' ongoing activities.

Keywords

Attentive systems, intelligent agents, peripheral information, multimodal input, user modeling, interest tracking.

1. INTRODUCTION

Computer users are routinely bombarded with information from a variety of sources. Running applications vie for screen real estate to display hints, help, status, alerts, and other sorts of information. In attempting to manage competing sources of information, users often configure their screens so that they can attend to what is most important at any time, while still maintaining the ability to monitor and interact with less important information. Nevertheless, monitoring the information at hand can be a full time job, and even then is prone to error, as unwanted and unneeded information can often find its way into the user's environment. Can the user interface do a better job of supporting users' information needs? We think so. In this paper, we explore an *attentive systems* approach to information delivery that is intended to help users manage information by keeping track of what the user is doing and presenting information appropriately.

We define a*ttentive systems* as systems that watch the user, model the user, and anticipate the user. The general class of attentive systems includes human butlers, new commercial devices such as TiVo [28] that automatically record television programs that the user likes or regularly watches, and web sites such as Amazon.com [1] that monitor book-buying and book-browsing behavior to model buyer interests and ultimately suggest additional books. Here we are concerned specifically with systems of this last sort, *attentive information systems* that support users' information needs. In observing user actions, such a system might track what web page the user is browsing or what application currently has focus. It might observe many different sorts of actions or it might observe just a few that are relevant for a specific task. To predict what information might be useful, an attentive system must learn from a user's history of activity to improve the relevance and timeliness of its suggestions, modeling the user and adapting its models over time. In suggesting potentially useful information to the user, an attentive information system should not intrude on the user's ongoing activity, displaying suggestions in the margins or on the periphery of the user's current task.

Attentive information systems are distinguished by three main qualities. First, they gather evidence about user behavior from multiple sources, possibly even across multiple modalities. When only a single source of data about user activities is monitored, there is a high chance of making incorrect inferences of user intentions. Multiple sources help systems disambiguate intentions and build a more accurate model of the user. In particular, multiple inputs disambiguate intentions by helping systems take account of the context of user actions. For example, if the system knows only that the user is visiting the IBM home page, it might be reasonable to assume that the user is interested in IBM's products. However, if the user also has a stock analysis application open and is also reading email about IBM's second quarter earnings, it might be better to infer that the user is interested in the current stock price or other financial news.

The second distinguishing quality of an attentive system is that it models the user at a very fine level of detail. The user model is kept up-to-date by closely tracking user behavior and interests. For instance, if the user is sending email to a friend about dinner plans, it might be appropriate to inform him or her of the hours and fare of local restaurants — but only as long as plans are being made. Once plans to meet have been set, such information is no longer relevant.

The final quality of an attentive information system is that it provide users with information that is relevant but not critical to task performance, and that this information be presented in a non-distracting way. Because it is difficult to guarantee correct and appropriate suggestions that anticipate the user, it would be unreasonable for such systems to issue a key-press command,

interrupt the user with a noisy alert, or replace the contents of a window. We call this sort of information *peripheral information* because it is both peripheral to the task and peripheral to the display. The key to peripheral information is that it is not critical to task performance. Unlike what is generally studied in the literature on monitoring and supervisory control (e.g., [18]), inattention to a peripheral display does not result in catastrophe, such as a nuclear meltdown or a plane crash. However, by providing peripheral information, an attentive system gives the user the opportunity to learn more, to do a better job, or to keep track of less important tasks

We designed the Simple User Interest Tracker (Suitor) to be an attentive information system. It provides an architecture in which simple programs (or agents) monitor user actions, process user actions or world events, and communicate suggestions to users through a variety of means. Suitor pays constant attention to what the user is doing, determines what information will most likely be interesting, and delivers that information on the spot. For instance, we have created agents that can (a) monitor web browsing, (b) monitor a user's eye-gaze to determine where on the screen the user is actually reading, and (c) find additional information about the topic that is being read on the current web page using external web search services. The key is that a user interacts with the computer as usual — reading, typing, clicking — and the system infers user interest based on what it sees the user do.

In what follows, first we explore several scenarios to illustrate how Suitor attends to user actions and attends to user information needs. Then we detail Suitor's architecture and implementation, discussing issues of user observation, user modeling, and peripheral information. Finally, we discuss related work, and conclude with some thoughts on future directions.

2. SCENARIOS

In one mode, Suitor delivers information to its user through a scrolling one-line text display located at the bottom of the screen (see Figure 1) — a peripheral display. Suitor can also deliver information through a web browser, by email, or to a personal digital assistant such as a PalmPilot.

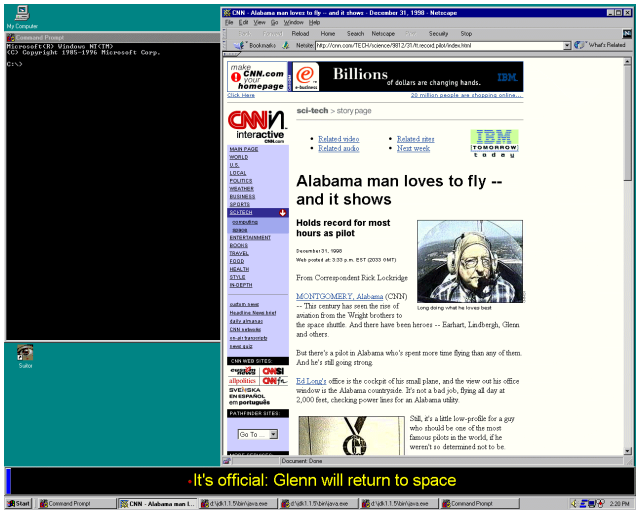

Figure 1: Sample screen showing scrolling display.

To see Suitor in action, consider how it might affect the information environment of a computer user named George. The following scenarios are fully implemented, except as noted in the text.

While debugging a program, George notices a headline in the scrolling display about terrorism threats in Europe. Because he flies to Europe twice a month on business, George clicks on the headline and the full story appears in a browser window. Throughout the day, George selects stories concerning the same topic from the scrolling display and even goes to the web to search for additional information. Soon he notices new stories about terrorism in Europe, air safety, and security at airports appearing in greater numbers in the scrolling display. In fact, there are now more stories about world news in general.

Like many information *push* systems, Suitor polls a variety of news categories — including world news, local news, politics, sports, and weather — to obtain headlines to display. Unlike standard push, the user is not modeled statically, for instance, as a set of check boxes for selecting broad topics of interest (e.g., [21]). Rather, Suitor infers what categories of news are of interest by attending to ongoing user activities. Likewise, Suitor is not constrained to any one type of information but can display what might be relevant to the user at the time.

Suppose George begins using Microsoft Word to edit a manuscript. After working for a few minutes, Word tips begin scrolling across the display, interspersed with headlines and stock quotes. From one of these scrolling tips, George notices that Ctrl-f is the keyboard shortcut for the menu navigation Edit—Find. In this way, Suitor provides information that is likely to be more relevant to the user's ongoing activity than an arbitrary news headline. When George stops using Word, Suitor stops displaying Word tips.

Suitor contains a number of agents to attend to web browsing and eye gaze. Browsing activity that can be monitored includes current URL, entered URL, web page text, and entered search terms. For example, if George opens a browser and goes to the IBM home page. A few seconds later the current IBM stock price and news about IBM appears in the scrolling display. On seeing the stock prices, George remembers that he wants to invest his bonus in hard-drive storage technology, so he clicks on the IBM stock symbol in the scrolling display, which opens a new browser window with details on the day's activity of the stock. Next, George checks the stock of a competing hard-drive storage company, Seagate technologies. Later, George discovers that he is getting quotes for both IBM and Seagate updated every half hour, as well as business news about each company.

Using eye-gaze information, Suitor can determine what application George is looking at or what information George is reading in the scrolling display. Suitor can use this as positive relevance feedback to adjust its model of George and thus to provide more timely information. Suppose George continues to be interested in investing in hard-drive storage technology, always reading the stock quotes for IBM and Seagate but ignoring quotes for other companies such as General Electric. After a while, stock quotes for IBM and Seagate continue appearing regularly but quotes for General Electric appear far less frequently. Suitor does not require the user to maintain a list of stock symbols but rather it infers from ordinary behavior what stock prices to display.

Now suppose George is working in Word and glances down to read a headline that looks interesting. A browser window opens and loads the story of the headline he has just read. After glancing briefly at the story, George decides that it is not interesting and continues working in Word. A few minutes later, George reads another headline that seems interesting. The story for it loads in the open browser window and he spends time reading it. Headlines of related stories begin appearing in the scrolling display. By reading or clicking any of the related headlines, the associated story can be obtained. Because George takes the time to read the stories behind some of the headlines, Suitor infers that he *might* be interested in reading related stories as well. Because these stories appear in a new browser window, it does not interfere with current browsing or with any other current task. Suitor does not take control of any task or force the user to perform any specific action.

Suitor can take advantage of multimodal information to help it better predict what is likely to be of interest to the user and to create an accurate user model. Suppose George begins editing his resume and then starts to search the web for employment opportunities. He goes to Sun Microsystems's web site to see current job listings. Two openings at other companies in System Administration — George's field — appear in the scrolling display. He clicks on both listings and two browser windows appear with details of the positions. One of the positions offers more money than Sun but George continues to look for more options. He has heard that Hewlett Packard is a good place to work so he goes to their main web site. In the scrolling display appears a shortcut URL for going directly to HP's job listings. Clicking on this URL, George investigates employment at HP.

Suitor can also use eye-gaze information to help disambiguate user interests. Suppose George is reading a magazine in his web browser but skips over most of the articles until he comes to one on the design of the new US currency. Tracking George's gaze, Suitor knows that this is the only topic he has actually read. In the scrolling display, there is a TV listing for a PBS special about the new currency. By clicking on the listing, George gets the times and days he can catch the show. In the future, Suitor could set a VCR to record relevant programs so that they can be viewed at a convenient time.

Suitor also contains agents that interface with the PalmPilot to upload text. The PalmPilot can act as a peripheral display or as a mobile information delivery device. Suppose George has been busy all day and has not had time to explore any of Suitor's suggestions. Suitor can upload news stories, stock quotes, employment listings, movie schedules, and TV listings directly to his PalmPilot before he leaves. Riding home on the bus George can catch up on the news and plan his evening. A program in the PalmPilot keeps track of the information read so that it can repeat back to Suitor what has been viewed on this device.

To see how Suitor can implement these scenarios, we now turn to details of its architecture and implementation.

3. ARCHITECTURE AND IMPLEMENTATION

Suitor provides an architecture for creating simple programs — agents — to investigate user activity, reflect on that activity, gather information from the user's computer or the outside world, and report relevant information to the user. More precisely, Suitor implements an interprocess communication mechanism that enables separate programs to work together to (a) gather information about the user, (b) gather information about the state of the world, and (c) report information to the user. This mechanism amounts to a shared blackboard and a corresponding scheme for dispatching information posted on the blackboard to interested agents. In addition to a communication scheme, Suitor also provides the infrastructure for developing and deploying agents, for remembering what the user is interested in, and for reporting to the user through a variety of means, such as a scrolling headline display, email, or PalmPilot.[1]

Implemented in Java, Suitor programmers can create modules and applications that track user behavior, infer user interest, find related information, and display that information to the user. Modules are groups of agents that perform a specific function; for instance, we have implemented a gaze module that monitors where the user is looking, what application is being looked at, and what text is being read. Applications are standalone programs that run independently of Suitor yet contain agents that can communicate with Suitor through Java's remote method invocation (RMI).

Agents communicate with one another through a common currency of *facts*. These might represent a notification or the text of a news story. All facts know which agent submitted them, what time they were submitted, when they should expire, and what other facts they depend on. All agents registered to listen for a certain type of fact are invoked when that type of fact is posted to the blackboard. Facts remain on the blackboard until they either expire or are explicitly retracted. Suitor periodically scans through all facts and removes those that have expired. An agent can also remove facts by asking Suitor to retract all facts it has submitted. This is useful in trying to keep up with the user's ever-changing interests.

There are four main functions a Suitor module might perform: watching the user (user input), remembering user actions (user model), getting information from local and remote databases (information gathering), and making suggestions (peripheral display). To create the scenarios described previously, modules were created that (a) monitor what application is being used, (b) monitor what web page is being viewed, (c) track where the user is looking on the screen, (d) record text input from the keyboard, (e) maintain a user model of descriptive keywords, (f) find stock prices and news, and (g) download information to a PalmPilot. A separate scrolling display application (a news ticker) was created to output information unobtrusively at the bottom of the screen.

3.1 Observing Users and the World

Investigator agents gather information from the world outside of Suitor. They can monitor user actions, watch a web site for new information, or scan through local and remote databases. Investigators can submit facts about the user or the world when some event occurs or when some data has been collected. Investigators do not operate on facts; they merely monitor state.

[1] See [29] for details of our initial implementation, which was called deFacto.

3.1.1 Gathering Information from Outside

Observing users in particular often requires connecting to the operating system (e.g., to observe running applications, keyboard input) or to the outside world (watch user's eye-gaze). In either case, native code must be written that sends data to Java-based Suitor. For example, the module that watches what application the user is currently working with relies on native C++ functions to determine the window with focus and the name of its executable.

Similarly, the user's eye-gaze is monitored by a module written in C++ to calculate the coordinates of gaze direction at 30 frames per second. The camera uses an array of LED's to project infrared light to the user's eye, resulting in a reflectance point on the cornea and the illumination of the pupil. With the reflectance point and the center point of the pupil along with a short calibration session, the location of eye-gaze can be easily calculated to within half an inch (see also [9,31] for information on our gaze-tracking system).

Watching a user's web browsing in Suitor is done by an independent application written with the Web Intermediaries (WBI) development kit [2,3,10]. This WBI application sits between the browser and the web, watching the flow of HTTP traffic between the two. Several types of information can be gathered this way about user activities, including the current URL, the text contained on the current page, and query data in submitted web forms.

One way of finding information that might be relevant to the user is to observe news sources on the web, such as CNN [4] or Yahoo! News [30]. That is, a module for gathering news can be constructed out of investigator agents that periodically poll web resources, submitting facts on new headlines, stories, or other information available there. Which headlines and stories are relevant to the user depends on what the user is interested in, as determined by Suitor's current user model.

3.2 Modeling Users

Reflector agents can submit facts and operate on facts. That is, in producing their own facts, reflectors think about (reflect on) the facts that are submitted by other agents. Reflectors essentially decide what to do about the information discovered by investigators and other reflectors. They can be used, for instance, to construct a model of the user's interests.

3.2.1 Modeling Users as Text

Suitor's user model is simple. Text gathered by investigators monitoring user interactions with the computer are combined and analyzed to produce a small list of key words. Of course, Suitor can use other sorts of user models, including statistical or probabilistic models [7,8]. In the case we have implemented, text is gathered from user keyboard input, from user email, from web pages read, and from files visited in Emacs. Key words are derived from these text sources by determining the frequency of the words in the pooled text at a given time relative to the frequency of the words overall. The keywords are those words whose frequency is high in the current set relative to their overall frequency (following, for instance, [15,23]).

The user's current interest is represented as a list of words that distinguish the sorts of text being written and read at any given time. Facts about what the user is typing and what the user is viewing constantly flow into Suitor's blackboard from investigator agents. As these facts arrive, reflector agents determine the word frequencies and update the current list of key words, that is, the current model of the user. As the user's interests change over time — as the user's activities shift from one task to another — the key words that represent the user's interests change. Thus, our user model contains both the user's current interest — current list of key words — and the user's history of interests — old lists of key words.

3.3 Displaying Suggestions

Actor agents are essentially the inverse of investigators; they act on facts that have been submitted to Suitor but they cannot submit facts themselves. Actors process facts from reflectors and perform some action (side effect) on the outside world, such as displaying information to the user. For instance, our scrolling ticker displays headlines and other facts to the user. Before actor agents select facts from the blackboard for display on the screen, reflector agents prioritize the news and other facts that investigator agents have gathered by comparing them with the user model. More precisely, before information is displayed, it is rated according to how much it overlaps the current and long-term model of user interests. Only facts that have some overlap with the user's interest are selected for display, and then they are ordered according to how much they overlap.

3.3.1 Displaying Suggestions Peripherally

An attentive system like Suitor ought to present suggestions to the user that are not distracting, yet are timely and relevant to the task at hand. Constant monitoring of user actions and the concomitant modeling of user interests are meant to ensure that suggestions are timely and relevant. For its display, Suitor provides a one-line scrolling ticker at the bottom of the screen to show the user its suggestions. This sort of scrolling display is intended to be both informative and unobtrusive. To try to ensure that Suitor's suggestions are not too distracting, we have begun to experimentally test the relative informativeness and distractibility of a variety of scrolling ticker displays.

Our hope is that scrolling displays are at least a little like the automobile's speedometer. The automobile driver's main task is driving, but speed information displayed in the periphery is not overly distracting and at the same time informative. The speedometer is designed perfectly to convey non-essential but useful information [20]. It embodies a kind of peripheral information — information is not central to the current task, but that might be helpful to it or that might be informative in other ways.[2] Such an interface is peripheral because the normal mechanism for accomplishing the task is still available. The interface simply seeks to make the task easier and richer.

---

[2] The term *ambient information* has been used to refer to subtle environmental cues when designed into systems to convey information such as network traffic peripherally [11]. Specific environmental changes, such as the frequency of background noise or amount of background lighting, are associated with specific changes in system status. Of course, the design problem here is to make the mapping from system state to environmental state as obvious as possible [19].

3.3.2 Testing Ticker Displays

We conducted a pilot study to assess how distracting and how effective three single-line text displays are in conveying information under dual-task demands — that is, when the user is working on a primary task such as text editing, but has a secondary task of keeping track of scrolling information. We tested three kinds of tickers: (a) continuous scrolling text (CS), (b) discrete scrolling text (DS), and (c) serial presentation (SP). In the CS case, text scrolls at a constant rate horizontally from right to left. In the DS case, text scrolls quickly to the center of display vertically, where it stops for some period before scrolling off the display. In the SP case, text does not scroll at all; rather, it is shown in a constant position in the center of the display, each update replacing existing text with new text.

To our knowledge, only two classes of ticker-like displays have been systematically explored, CS and SP — and then only to compare these and other display schemes in terms of informativeness rather than in terms of dual-task demands. Studies directly comparing CS and SP found no difference in comprehension for the display reading task [14]. Other studies comparing CS to static displays found that text is read more slowly [25] and is less comprehensible [6] when scrolling than when displayed on a static page, and this effect does not depend on the number of words on the screen or on window size [5]. Studies comparing SP and static pages of text show that comprehension performance is about the same [13], and that reading latency is about the same as well [22]. Overall, these studies fail to show strong differences in informativeness among CS, SP, and static displays.

Our study used a dual-task procedure that is similar to real task demands faced by computer users. The first task required editing the text of a document of moderate reading difficulty and the second task involved reading fictional news headlines from one of the tickers. Participants were instructed to give both equal priority (text editing and headline reading). Distraction was measured as the change in performance for text editing alone (i.e., no scrolling ticker) versus text editing while concurrently reading the ticker. Our results showed that the discretely scrolling ticker had the least impact on editing performance. For CS, the number of corrections decreased by 32% from the no ticker condition; for SP, the number of corrections decreased by about 20%; and for DS, corrections decreased by about 10%. All pairwise differences were significant.

Informativeness was defined as how well the headlines were remembered, as measured by a post-experiment multiple-choice test. No difference was found among the displays, and this was not the result of floor or ceiling effects, as scores spanned a normal range from 30% to 100% correct, with means of 70% for CS, 67% for DS, and 76% for SP.

Overall, our results show a difference for distraction but not for informativeness. Thus, the different displays serve different functions in a dual-task situation. Peripheral information should be both informative, presenting users with previously unknown facts, and non-distracting, enabling users to remain focused on their primary tasks. Because the discrete ticker was highly informative and only slightly distracting, we believe that this is exactly the sort of display that can be used effectively to peripherally inform computer users. This is why we have chosen it as Suitor's main means for displaying suggestions.

3.4 Putting It All Together

Having described the various sorts of agents and functions available in Suitor, we can now show an example of Suitor in action. In this scenario, news headlines are scrolling by in the ticker display, and the user looks down and reads one. Because the user's gaze dwells on a particular headline longer than some threshold period (indicating interest in the topic), the news story associated with it is displayed in a browser window.

More precisely, as shown in Figure 2, the gaze module has an investigator agent that periodically posts facts to the blackboard indicating where the user is looking. A reflector agent, triggered when gaze information is posted to the blackboard, determines whether the user is gazing at the scrolling ticker display. If the user has been looking at the display long enough, a reflector agent inside the ticker application determines which headline in particular the user is looking at — this is done inside the ticker itself, as the ticker knows what headlines are being displayed. Finally, the reporter module looks up the story and posts a fact to trigger an actor agent that communicates directly with the browser.

Tracking eye gaze in particular seems a very powerful means for gathering evidence about user interest [27]. If the user pays attention to certain displayed information, the system can take that as positive relevance feedback, effectively suggesting that it display more of the same sort of information. Conversely, if the user does not pay attention to certain information, the system can take that as negative feedback, suggesting that it not display similar information again. In these cases, gaze is not used to control the system explicitly, such as for directly selecting what to display; rather, gaze is at least one step removed, figuring in the calculation of user interest, which in turn figures in what is displayed.

In this way, Suitor provides a *non-command interface*, as it relies on pooled evidence to respond to user actions. That is, Suitor relies on natural eye movements and other ordinary user actions as control signals rather than on explicit user commands [12]. The user is not forced to check a box for a category of news and then click a button for delivery of news. Rather, Suitor works behinds the scenes inferring from normal user actions what information to display. The option is available, however, for the user to interact with Suitor in a more direct manner. For example, if Suitor displays a headline that the user wants additional information on, the user can click the headline and the full story will appear in the current browser window. Suitor can then infer from this action that the user is interested in the specific topic of the headline and to a lesser extent, the general category of the headline (i.e., sports, world news, and politics). Thus, positive relevance feedback can be obtained by watching what the user does when interacting with the peripheral information display.

4. RELATED WORK

We are not the first build attentive systems. In the domain of web browsing, for instance, Lieberman's Letizia [15,16] is attentive, as it monitor's web use and scouts the web ahead of the user, determining the potential relevance of links on each page viewed. Letizia observes browsing, models user interest as a set of key words, and displays suggestions in a browser window placed off to the side.

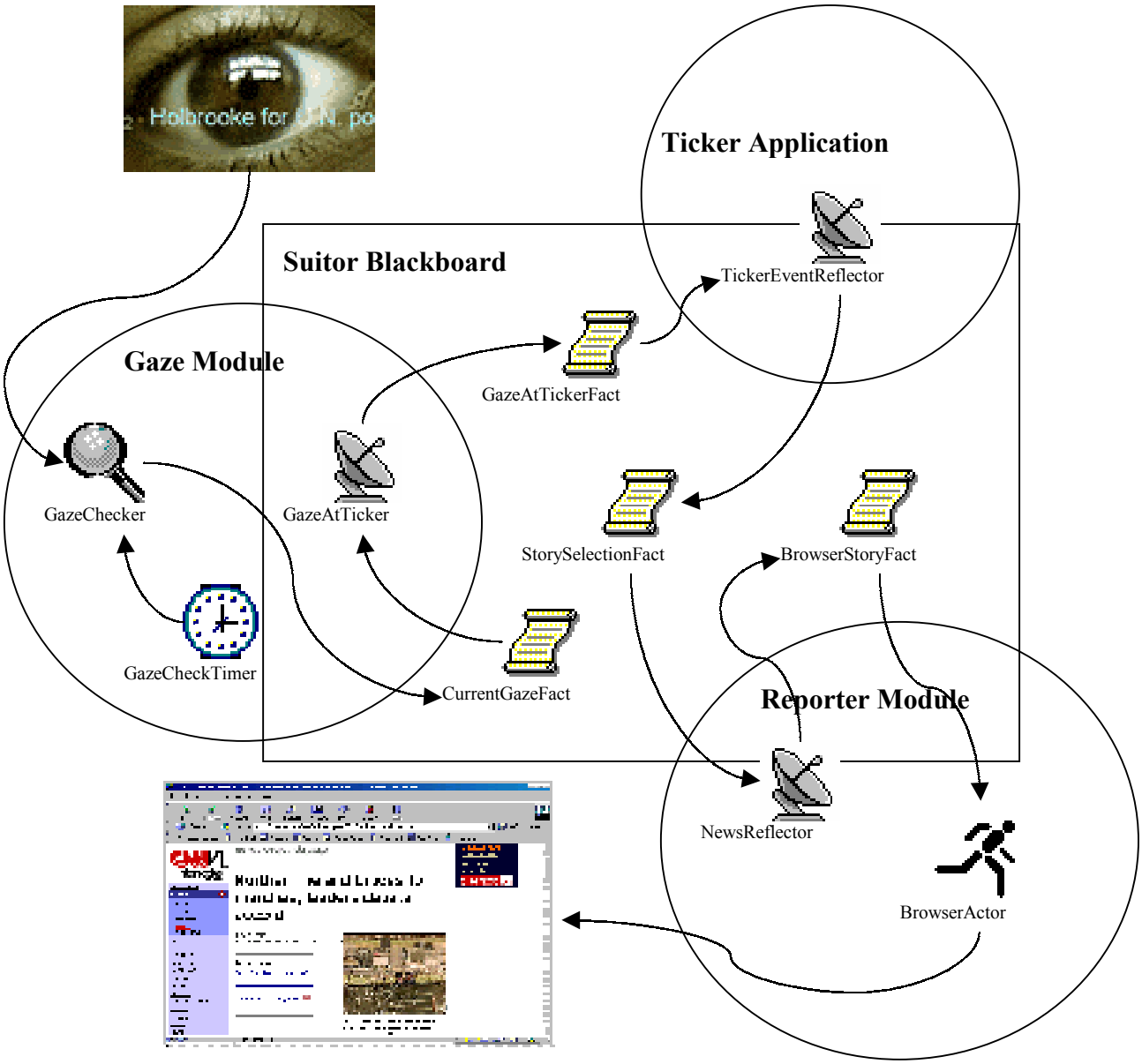

Figure 2: When the user reads a headline in the scrolling ticker display, agents determine that the user has read a headline, which headline was read, and ultimately display the story in a browser window.

In the domain of text editing, the Remembrance Agent (RA) is attentive because it monitors user input from several sources and displays relevant documents in a non-distracting manner [23]. The RA watches input from the keyboard and text information in Emacs, suggesting related information culled from text files located on the user's computer. Some of the text information scanned includes old e-mail, papers, files of notes, as well as other text documents. The RA determines the similarity of the text documents and the current text by the relative frequency of words common to both. If relevant documents are found, the first line of each is displayed in a window at the bottom of the screen.

More recently, the RA has been implemented on wearable computer systems and collects more evidence about the user, including location (through GPS), people nearby, and timestamp [24]. This information is more about the context than the user's actions. Like the desktop version, the wearable RA presents relevant information in an unobtrusive window at the bottom of a heads-up-display.

Help systems can also be attentive, but these typically monitor fewer input sources (often only one), have pre-existing user models (expert models), and focus on user performance in a single task. Software help agents most often watch command sequences for a specific application [8,17] or text from keyboard input [25]. COACH (Cognitive Adaptive Computer Help) [25] is a good example of an attentive interface because it continuously updates its user model based on keyboard input, offers help when the user is having a problem, and displays help peripherally.

Suitor is different from the Letizia, the Remembrance Agent, and help systems such as COACH in that it collects multiple sources of user behavior as well as context related directly to user actions. In Suitor, a wide range of user actions can be monitored, including keyboard input, mouse gestures, the current URL, text of the current web page, and the application with focus. We have also built agents to track eye gaze so that Suitor can determine what application the user is looking at, what suggested information the user is reading, and what web page the user is reading, as well as specific text the user is reading.

5. CONCLUSION AND FUTURE WORK

Suitor passively tracks computer users to determine their interests, and then delivers them relevant and timely information. In the cases we have implemented, there is a simple mapping from detection of user interest to method for finding relevant information. For application help, once it is found that the user is interested in changing the font size (by watching what keystrokes are taken or what menu items are selected), for example, the system can look in its database for documentation on this. For news stories relevant to the current web page, once it is determined what the page is about (by keying on the URL's server), well known web sites can be polled for stock prices or related news. For information relevant to what is typed or what is read, once words of interest are determined, related text can be retrieved from local files or from distant databases. Thus, Suitor does not do very deep or detailed reasoning to develop its suggestions. Although this may be appropriate for some of the cases we have explored here, one clear direction for future work is to create more elaborate chains of reasoning so that individual user actions do not completely determine the information or suggestions Suitor derives.

A related direction for future work is to create more elaborate user models that integrate evidence from a variety of sources, for instance, by combining user actions with keyboard input and gaze. Our simple text-based user model already combines text written and text read to determine a set of currently relevant words. However, it does not combine text with any understanding the user's behavior in terms of the user's goals. By considering evidence of user behavior derived from a variety of sources more generally, we hope to be able to build more accurate models of user interest and user need than we can without an understanding of user goals.

In summary, Suitor is a framework for developing attentive information systems — systems that monitor user behavior, model user interest, and make helpful suggestions. We have implemented methods for observing user behavior that ranges from spying on application usage and text typed to tracking eye gaze and web browsing. We have implemented a simple user model from the words typed and the words read that describes the user's interest at any time. We have implemented and tested schemes for displaying suggestions peripherally — so that they are not too distracting. But in the end, we have only just begun to explore ways of tracking users and of finding and displaying related information.

6. ACKNOWLEDGMENTS

Gentry Underwood contributed greatly to Suitor. Carlos Morimoto and Myron Flickner developed our gaze tracking system [9]. Denis Lalanne implemented our user model. Teenie Matlock provided helpful comments on a draft of this paper.

7. REFERENCES

1. Amazon.com, Inc. *Amazon.com – Earth's Biggest selection*. Available at http://www.amazon.com/.
2. Barrett, R., Maglio, P. P., & Kellem, D. C. How to personalize the web. In *Proceedings of CHI '97,* 1997.
3. Barrett, R. & Maglio, P. P. Intermediaries: New places for producing and manipulating web content. *Computer Networks and ISDN Systems, 30,* 1998, 509--518.
4. Cable News Network, *CNN Interactive*. Available at http://www.cnn.com/.
5. Duchnicky, R. L. & Kolers, P. A. Readability of text scrolled on visual display terminals as a function of window size. *Human Factors, 25*, 1983, 683-692.
6. Granaas, M. M., McKay, T. D., Laham, R. D., Hurt, L. D. & Juola, J. F. Reading moving text on a CRT screen. *Human Factors, 26*, 1984, 97-104.
7. Heckerman, D. & Horvitz, E. Inferring informational goals from free-text queries: A bayesian approach, in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 1998, 230-237.
8. Horvitz, E. Breese, J., Heckerman, D., Hovel, D., &. Rommelse, K. The Lumiere project: Bayesian user modeling for inferring the goals and needs of software

users, in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 1998, 256-265.

9. IBM Research. *BlueEyes*. Available at http://www.almaden.ibm.com/cs/blueeyes/.
10. IBM Research. *Web Intermediaries – WBI*. Available at http://www.almaden.ibm.com/cs/wbi/.
11. Ishii, H. & Ullmer, B. Tangible bits: Towards seamless interfaces between people, bits, and atoms. In *Proceedings of CHI '97*, ACM Press, 1997, 234-241.
12. Jacob, R. J. K. Eye movement-based human computer interaction techniques: Toward non-command interfaces, in R. Hartson & D. Hix (Eds.), *Advances in Human Computer Interaction, Vol. 4*. Ablex, Norwood NJ, 1993, 151-190.
13. Juola, J. F., Ward, N. J., & McNamara, T. Visual search and reading of rapid serial presentations of letter strings, words, and text. *Journal of Experimental Psychology: General, 111*, 1982, 208-227.
14. Kang, T. J. & Muter, P. Reading dynamically displayed text. *Behaviour & Information Technology, 8*, 1989, 33-42.
15. Lieberman, H. Letizia: An agent that assists web browsing, in *International Joint Conference on Artificial Intelligence*, 1995, 924-929.
16. Lieberman, H. Autonomous interface agents, in *Proceedings of CHI '97*, 1997, 67-74.
17. Linton, F., Joy, D., & Schaefer, H., Building user and expert models by long-term observation of application usage, in *Proceedings of the Seventh International Conference on User Modeling*, 1999, 129-138.
18. Moray, N. Monitoring behavior and supervisory control, in K. R. Boff, L. Kaufman, & J.P. Thomas (Eds.), *Handbook of Perception and Human Performance: v. II,* Wiley, New York, 1986.
19. Norman, D. A. Cognitive engineering, in D. A. Norman & S. W. Draper (Eds.), *User centered system design*, Erlbaum, 1986.
20. Norman, D. A. *Things that make us smart*. Addison-Wesley, Reading MA, 1993.
21. PointCast, Inc. *Welcome to PointCast*. Available at http://www.pointcast.com/.
22. Potter, M. C., Kroll, J. F. & Harris, C. Comprehension and memory in rapid sequential reading, in R. Nickerson (Ed.), *Attention and Performance VIII*. LEA, Hillsdale NJ, 1980.
23. Rhodes, B. J., & Starner, T. The remembrance agent: A continuously running information retrieval system, in *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multiagent Technology,* 1996, 487-495.
24. Rhodes, B. J. The wearable remembrance agent: A system for augmenting memory, *Personal Technologies, 1*, 1997, 218-224.
25. Sekey, A. & Tietz, J. Text display by 'saccadic scrolling'. *Visible Language, 16*, 1982, 62-76.
26. Selker, T., COACH: A teaching agent that learns, *Communications of the ACM, 37(1),* 1994, 92-99.
27. Starker, I. & Bolt, R. A. A gaze-responsive self-disclosing display, in *Proceedings of the Conference on Human Factors in Computing Systems, CHI '90*, 1990, 3-9.
28. TiVo Inc. *Welcome to TiVo*. Available at http://www.tivo.com/.
29. Underwood, G., Maglio, P. P., & Barrett, R. User centered push for timely information delivery. *Computer Networks and ISDN Systems, 30*, 1998.
30. Yahoo! Inc, *Yahoo! News*. Available at http://dailynews.yahoo.com/.
31. Zhai, S., Morimoto, C., & Ihde, S. Manual input cascaded (MAGIC) pointing, in *Proceedings of CHI '99*, 1999..