ELSEVIER

Knowledge-Based Systems 14 (2001) 103–110

Knowledge-Based SYSTEMS

www.elsevier.com/locate/knosys

An architecture for developing attentive information systems

P.P. Maglio[a,*], C.S. Campbell[a], R. Barrett[a], T. Selker[b]

[a]*IBM Almaden Research Center, 650 Harry Road, NWE-B2, San Jose, CA 95120, USA*
[b]*MIT Media Lab, 77 Massachusetts Avenue, Cambridge, MA 02139, USA*

Abstract

Attentive systems attend 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 Simple User Interest Tracker (Suitor), an architecture for developing attentive information systems that track computer users through multiple channels — eye gaze, web browsing, application use, to determine interests and to try to satisfy information needs. By observing behavior and modeling users, Suitor can be used to find and display potentially relevant information that is both timely and non-disruptive to the user's ongoing activities. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Attentive systems; Intelligent agents; User modeling

1. Introduction

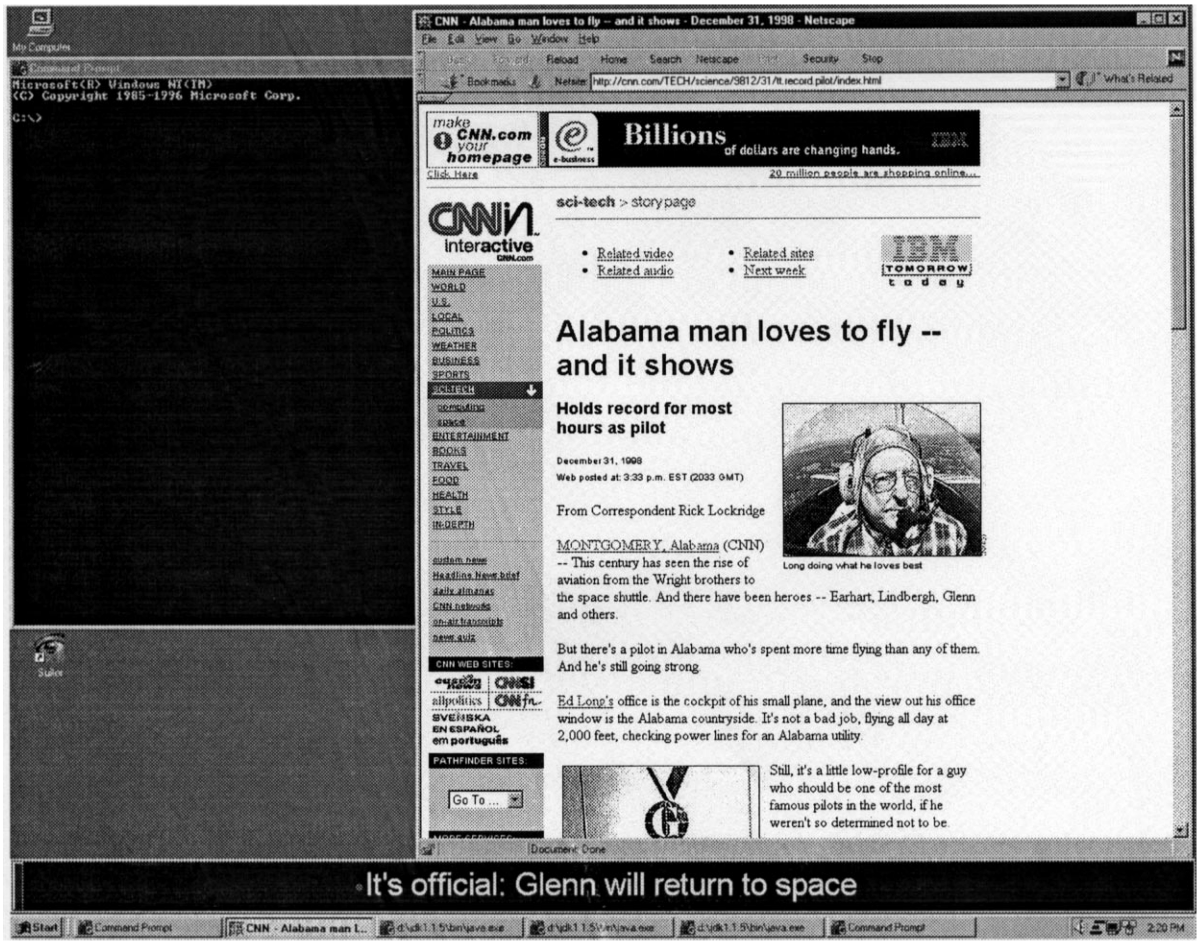
Computer users are routinely bombarded with information from a variety of sources. Running applications vie for screen real estate to display hints, help, status, alerts, and other sorts of information. In attempting to manage competing sources of information, users often configure their screens so that 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 unnecessary information can often find its way into the user's environment. Can the user interface do a better job of supporting user's 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 *attentive systems* as systems that work cooperatively with users, learning user interests and facilitating user needs and goals. This specifically requires watching the user, being aware of the world, modeling the user, anticipating user needs, and efficiently communicating with the user. Attentive systems as a general class includes butlers, personal/executive assistants, and other service professionals. Some new commercial devices also behave like attentive systems, such as TiVo [1], a system that automatically records television programs that the user likes or regularly watches, and web sites such as Amazon.com [2] 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 the last mentioned kind, *attentive information systems* or systems that support user's 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. 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. 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 communicating 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 (see also Ref. [3]).

Attentive information systems are distinguished by three main qualities. First, they gather evidence about user behavior from multiple sources, possibly even across multiple

* Corresponding author.
E-mail addresses: pmaglio@almaden.ibm.com (P.P. Maglio), ccampbel@almaden.ibm.com (C.S. Campbell), barrett@almaden.ibm.com (R. Barrett), selker@media.mit.edu (T. Selker).

0950-7051/01/$ - see front matter © 2001 Elsevier Science B.V. All rights reserved.
PII: S0950-7051(00)00099-X



```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>
```

CNN - Alabama man loves to fly -- and it shows - December 31, 1998 - Netscape

http://cnn.com/TECH/science/9812/31/fl.record.pilot/index.html

Alabama man loves to fly -- and it shows

Holds record for most hours as pilot

December 31, 1998
Web posted at: 3:33 p.m. EST (2033 GMT)

From Correspondent Rick Lockridge

MONTGOMERY, Alabama (CNN) -- This century has seen the rise of aviation from the Wright brothers to the space shuttle. And there have been heroes -- Earhart, Lindbergh, Glenn and others.

But there's a pilot in Alabama who's spent more time flying than any of them. And he's still going strong.

Ed Long's office is the cockpit of his small plane, and the view out his office window is the Alabama countryside. It's not a bad job, flying all day at 2,000 feet, checking power lines for an Alabama utility.

Still, it's a little low-profile for a guy who should be one of the most famous pilots in the world, if he weren't so determined not to be.

Long doing what he loves best

•It's official: Glenn will return to space

Fig. 1. Sample screen showing scrolling display.

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. 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 (see also Ref. [4]). 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. Ref. [5]), 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) as an architecture for developing attentive information systems. Developers can use Suitor to create customized programs (or agents) that monitor user actions, search the world for information, process user actions or world events, and communicate suggestions to users through a variety of means. There are many ways to create individual attentive devices or computers, and many ways to distribute agents across devices in the environment to create attentive spaces. For instance, for a single user working at a computer, 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 on the web about the topic that is being read on the current web page. Applications can be developed that perform specific functions such as task specific help or web navigation assistants, or large-scale attentive systems can be created that monitor the user through multiple modalities and perform complex inference. Additionally, agents from many simple or large applications can be combined into the same system to provide enhanced functionality.

In what follows, first we explore several scenarios to illustrate how developers might use Suitor to create systems that attend to user actions and 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 Fig. 1) — a peripheral display. Suitor currently has agents that allow the system to deliver information through a web browser, by email, or to a personal digital assistant such as a PalmPilot. 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. Ref. [6]). 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 multi-modal information to help it to predict better 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 communicate 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 and the world; (b) process and make inferences about the user; 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. Modules run in the Suitor application and share the Suitor Java/VM. Applications are standalone programs that run independently of Suitor yet contain agents that can communicate with Suitor through Java's remote method invocation (RMI).

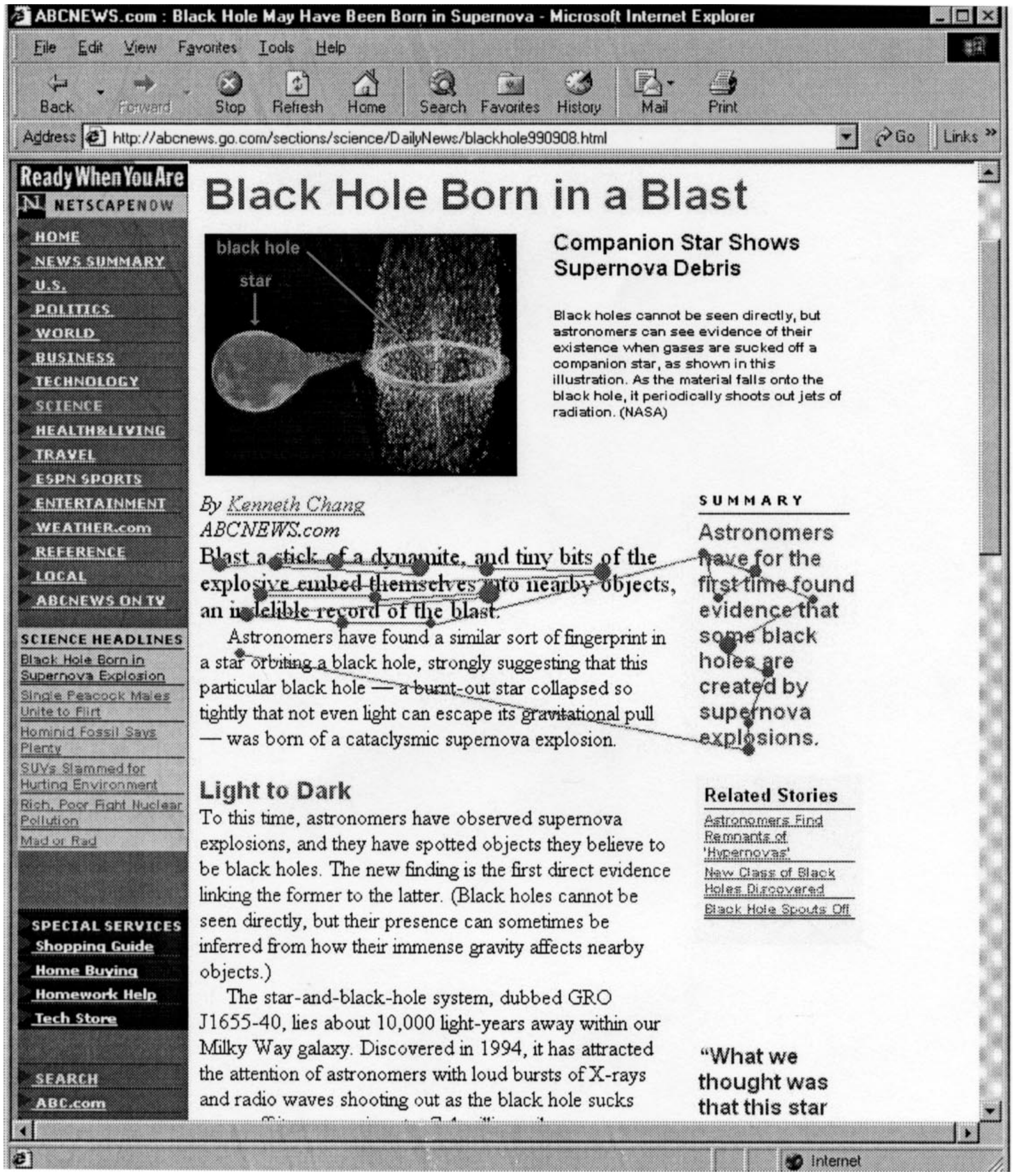
Agents communicate with one another through a common currency of *facts*. These might represent a notification or the text of a news story. All facts know which agent submitted them, what time they were submitted, when they should expire, and what other facts they depend on. All agents registered to 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. The Suitor architecture allows developers to create a wide range of input sensors

[1] See Ref. [7] for details of our initial implementation, which was called deFacto.


ABCNEWS.com : Black Hole May Have Been Born in Supernova - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print
Address http://abcnews.go.com/sections/science/DailyNews/blackhole990908.html
Go Links
Ready When You Are
NETSCAPENOW
HOME
NEWS SUMMARY
U.S.
POLITICS
WORLD
BUSINESS
TECHNOLOGY
SCIENCE
HEALTH&LIVING
TRAVEL
ESPN SPORTS
ENTERTAINMENT
WEATHER.com
REFERENCE
LOCAL
ABCNEWS ON TV
SCIENCE HEADLINES
Black Hole Born in Supernova Explosion
Single Peacock Males Unite to Flirt
Hominid Fossil Says Plenty
SUVs Slammed for Hurting Environment
Rich, Poor Fight Nuclear Pollution
Mad or Rad
SPECIAL SERVICES
Shopping Guide
Home Buying
Homework Help
Tech Store
SEARCH
ABC.com
black hole
star
Internet

Black Hole Born in a Blast

Companion Star Shows Supernova Debris

Black holes cannot be seen directly, but astronomers can see evidence of their existence when gases are sucked off a companion star, as shown in this illustration. As the material falls onto the black hole, it periodically shoots out jets of radiation. (NASA)

By Kenneth Chang
ABCNEWS.com

Blast a stick of a dynamite, and tiny bits of the explosive embed themselves into nearby objects, an indelible record of the blast.

Astronomers have found a similar sort of fingerprint in a star orbiting a black hole, strongly suggesting that this particular black hole — a burnt-out star collapsed so tightly that not even light can escape its gravitational pull — was born of a cataclysmic supernova explosion.

SUMMARY

Astronomers have for the first time found evidence that some black holes are created by supernova explosions.

Light to Dark

To this time, astronomers have observed supernova explosions, and they have spotted objects they believe to be black holes. The new finding is the first direct evidence linking the former to the latter. (Black holes cannot be seen directly, but their presence can sometimes be inferred from how their immense gravity affects nearby objects.)

The star-and-black-hole system, dubbed GRO J1655-40, lies about 10,000 light-years away within our Milky Way galaxy. Discovered in 1994, it has attracted the attention of astronomers with loud bursts of X-rays and radio waves shooting out as the black hole sucks

Related Stories

Astronomers Find Remnants of 'Hypernovas'
New Class of Black Holes Discovered
Black Hole Spouts Off

"What we thought was that this star

Fig. 2. Example pattern of eye movements during reading overlaid on web page text.

and agents. Sensors can monitor users and the world through multiple modalities, including vision, sound, and touch. Investigator agents can be created to gather any type of information the developer wants including user interactions with the operating system, user identity, and information from networked databases on the Internet. So far, we have created investigators that monitor running applications, applications with focus, keyboard input, mouse movements, web browsing, web searching, news information on the Web, stock quotes, and user eye gaze.

One focus of our work has been on eye-gaze, as it can be a powerful source of evidence on user information interests. The user's eye-gaze is monitored by the gaze module that calculates the coordinates of gaze direction at 30 frames per sec. 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 Refs. [8,9] for information on our gaze-tracking system). Within the gaze module, there are a set of investigators and reflectors to process incoming gaze data and determine if the user is viewing information in the ticker display, looking at a certain application, or looking at news stories in the browser. Recently, Suitor's sensing abilities have been expanded with a new investigator that determines whether the user is reading text on the screen. This investigator watches the pattern of eye-gaze data and tries to detect and track when the user is reading.

The reading investigator works by first stabilizing the raw gaze data, classifying each eye movement into one of seven categories, pooling category evidence, and switching from detection to read-tracking mode based on the pooled evidence. Stabilization is done by quantizing the raw gaze data, that is, averaging the data over non-overlapping 100 ms intervals. Stabilization helps reduce the influence of micro-saccades and gaze measurement errors (see Fig. 2). Eye movements are then classified into one of seven categories: read forward, skim forward, scan jump, skim jump, regression, anticipatory saccade, or reset jump. Each category either adds points to the evidence counter, takes points away, or does not change the counter. Eye movements such as read forward serves as evidence for reading, and thus add points to the evidence pool, whereas movements such as regressions server as evidence against reading and thus take points away from those accumulated in the evidence pool. When the pool reaches a threshold value, then reading is detected and the mode changes to read tracking. In read-tracking mode, evidence is no longer accumulated and the only way to break out of this mode is to detect a scan jump eye movement. Preliminary empirical tests have shown this reading-detection algorithm provides both robust and efficient reading detection.

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.

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 [10,11]. 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, Refs. [12,13]).

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.

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. 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 [14]. 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. 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. In any event, 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, and we have found that in fact these sorts of displays can be used effectively to provide peripheral information (see Ref. [4]).

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 gazes down and looks at one. Because the user reads a particular headline (indicating interest in the topic), the news story associated with it is displayed in a browser window.

More precisely, the gaze module has an investigator agent that periodically gets eye-gaze locations from a sensing device (camera). A reflector agent, triggered when gaze information is posted to the blackboard, determines whether or not the user is reading. Another reflector agent, also triggered when gaze information is posted to the blackboard, determines if the gaze was in a browser window. Yet another reflector agent checks if both reading is detected and gaze was in a browser window and sends a request to have related information sent to the peripheral display.

Tracking eye gaze in particular seems a very powerful means for gathering evidence about user interest [15]. 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 [16]. 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 [12,17] 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. Rhode's Margin Notes system is similar [3].

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 [13]. 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. The RA has also been implemented on wearable computer systems and collects more evidence about the user, including location (through GPS), people nearby, and timestamp [18]. 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 [11,19] or text from keyboard input [20]. COACH (Cognitive Adaptive Computer Help) [20] 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.

5. Conclusions

Attentive systems attend to the user, gather information about the world, model user interests, and communicate with the user in a non-distracting way through peripheral displays. Attentive information systems work cooperatively with users to learn their informational interests and to facilitate their needs and goals. Suitor is a framework for developing attentive information systems. 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.

Within the Suitor architecture, a wide variety of attentive information systems can be created, from large adaptive applications to specific software tools. Suitor's modularity allows incremental development, adding agents and modules as new sensors, devices, and displays become available. As agents incorporating new technologies are developed and deployed, we believe that such systems will show the benefits that come from truly multi-modal, interactive, and attentive interfaces, namely naturalness, robustness, and efficiency.

Acknowledgements

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

References

[1] TiVo Inc, Welcome to TiVo, Available at http://www.tivo.com/.
[2] Amazon.com Inc, Amazon.com — Earth's Biggest selection, Available at http://www.amazon.com/.
[3] B.J. Rhodes, Margin notes: building contextually aware associative memory, Proceedings of the Conference on Intelligent User Interfaces (IUI 2000), ACM Press, New York, 2000.
[4] P.P. Maglio, C.S. Campbell, Tradeoffs in the display of peripheral information, Proceedings of the Conference on Human Factors in Computing Systems (CHI 2000), ACM Press, New York, 2000.
[5] N. Moray, Monitoring behavior and supervisory control, in: K.R. Boff, L. Kaufman, J.P. Thomas (Eds.), Handbook of Perception and Human Performance, vol. II, Wiley, New York, 1986.
[6] PointCast Inc, Welcome to PointCast, Available at http://www.pointcast.com/.
[7] G. Underwood, P.P. Maglio, R. Barrett, User centered push for timely information delivery, Computer Networks and ISDN Systems (1998) 30.
[8] IBM Research, Blueeyes, Available at http://www.almaden.ibm.com/cs/blueeyes/.
[9] S. Zhai, C. Morimoto, S. Ihde, Manual input cascaded (MAGIC) pointing, Proceedings of the Conference on Human Factors in Computing Systems CHI '99, ACM Press, New York, 1999.
[10] D. Heckerman, E. Horvitz, Inferring informational goals from free-text queries: a bayesian approach, Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, 1998, pp. 230–237.
[11] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, K. Rommelse, The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users, Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, 1998, pp. 256–265.
[12] H. Lieberman, Letizia: an agent that assists web browsing, International Joint Conference on Artificial Intelligence, AAAI Press, 1995, pp. 924–929.
[13] B.J. Rhodes, T. Starner, The remembrance agent: a continuously running information retrieval system, Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multiagent Technology, 1996, pp. 487–495.
[14] D.A. Norman, Things that make us smart, Addison-Wesley, Reading, MA, 1993.
[15] I. Starker, R.A. Bolt, A gaze-responsive self-disclosing display, Proceedings of the Conference on Human Factors in Computing Systems CHI '90), ACM Press, New York, 1990.
[16] R.J.K. Jacob, 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, pp. 151–190.
[17] H. Lieberman, Autonomous interface agents, Proceedings of the Conference on Human Factors in Computing Systems CHI '97, 1997, pp. 67–74.
[18] B.J. Rhodes, The wearable remembrance agent: a system for augmenting memory, Personal Technologies 1 (1997) 218–224.
[19] F. Linton, D. Joy, H. Schaefer, Building user and expert models by long-term observation of application usage, Proceedings of the Seventh International Conference on User Modeling, Springer, London, 1999, pp. 129–138.
[20] T. Selker, COACH: a teaching agent that learns, Communications of the ACM (1994) 37.