The Wearable Remembrance Agent: A System for Augmented Memory

Bradley J. Rhodes

Software Agents Group, MIT Media Laboratory, Cambridge, MA, USA

Abstract: This paper describes the wearable remembrance agent, a continuously running proactive memory aid that uses the physical context of a wearable computer to provide notes that might be relevant in that context. A currently running prototype is described, along with future directions for research inspired by using the prototype.

Introduction

With computer chips getting smaller and cheaper the day will soon come when the desktop computer processing power will disappear into a vest pocket. Tiny earphones, displays mounted in eyeglasses, and one-handed and no-handed input devices, are likewise becoming cheaper, lighter and less obtrusive, enabling whole computers that are worn like clothing or jewellery rather than held as one would a book or a clipboard. However, without compelling applications to drive a consumer market, wearable computers will never be more than specialised equipment for professionals and expensive novelty toys. These 'killer applications' need to be more than useful. For wearables to compete with desktop, laptop and palmtop computers, applications are needed that uniquely require a computer to be worn rather than carried in a bag or back pocket.

This article will start by describing features available in wearable computers that are not available in current laptops or personal digital assistants (PDAs). It will then describe a class of applications called remembrance agents (RAs) that can exploit these features, and will finally describe the wearable remembrance agent, a wearable just-in-time information system that continually reminds the wearer of potentially relevant information based on the wearer's current physical and virtual context. Finally, it will discuss related work and extensions that are being added to the current prototype system.

Wearable Computers vs. PDAs

The fuzzy definition of a wearable computer is that it is a computer that is always with you, is comfortable and easy to keep and use, and is as unobtrusive as clothing. However, this 'smart clothing' definition is unsatisfactory when pushed in the details. Most importantly, it does not explain how a wearable computer is any different from a very small palmtop. A more specific definition is that wearable computers have many of the following characteristics:

- *Portable while operational:* The most distinguishing feature of a wearable is that it can be used while walking or otherwise moving around. This distinguishes wearables from both desktop and laptop computers.
- *Hands-free use:* Military and industrial applications for wearables especially emphasise their hands-free aspect, and concentrate on speech input and heads-up display or voice output. Other wearables might also use chording keyboards, dials and joysticks to minimise the tying up of a user's hands.
- *Sensors:* In addition to user inputs, a wearable should have sensors for the physical environment. Such sensors might include wireless communications, GPS, cameras or microphones.
- *'Proactive':* A wearable should be able to convey information to its user even when not actively being used. For example, if your computer wants to let you know you have new email and who it is from, it should be able to communicate this information to you immediately.
- *Always on, always running:* By default, a wearable is always on and working, sensing and acting. This is opposed to the normal use of pen-based PDAs, which normally sit in one's pocket and are only woken up when a task needs to be done.

Personal Technologies (1997) 1:218-224

An overall design philosophy for wearables can be implied from these characteristics. Wearable computers are by their nature highly portable, but their main distinguishing feature is they are designed to be usable at any time with the minimum amount of cost or distraction from the wearer's primary task. A wearable computer user's primary task is not using the computer, it is dealing with their environment with the computer in a secondary support role.

The secondary-task nature, proactivity and sensors of wearables make them a natural match for software agent technology. Software agents are programs that continuously run in the background, sensing their environment and occasionally proactively acting on behalf of their user [1]. Augmented memory is an especially good application for agent technology and wearables, since everyone needs help remembering, especially when away from their normal calendars and desktop computers.

Remembrance Agents

The portability of the palmtop computer has driven their primary use as day planners, address books and note-taking devices. The problem with these memory aids is that current computer-based memory aids are written to make life easier for the computer, not for the person using them. For example, the two most available methods for accessing computer data are using file names (forcing the user to recall the name of the file) and browsing (forcing the user to scan a list and recognise the name of the file). Both these methods are easy to program but require the user to do the brunt of the memory task themselves. Hierarchical directories or structured data such as calendar programs help only if the data itself is very structured, and break down whenever a file or a query does not fit into the pre-designed structure. Similarly, keyword searches only work if the user can think of a set of words that uniquely identifies what is being searched for.

Human memory does not operate in a vacuum of query-response pairs. On the contrary, the context of a remembered episode provides lots of cues for recall later. These cues include the physical location of an event, who was there, what was happening at the same time, and what happened immediately before and after [2]. This information both helps us recall events given a partial context, and to associate our current environment with past experiences that might be related.

Until recently, computers have only had access to a user's current context within a computational task, but not outside of that environment. For example, a word-processor has access to the words currently typed, and perhaps files previously viewed. However, it has no way of knowing where its user is, whether she is alone or with someone, whether she is thinking or talking or reading, etc. Wearable computers give the opportunity to bring new sensors and technology into everyday life, such that these pieces of physical context information can be used by our wearable computers to aid our memory using the same information humans do.

By contrast, the RA is a program that continuously 'watches over the shoulder' of the wearer of a wearable computer and displays one-line summaries of notes – files, old email, papers and other text information that might be relevant to the user's current context. These summaries are listed in the bottom few lines of a heads-up display, so the wearer can read the information with a quick glance. To retrieve the whole text described in a summary line, the wearer hits a quick chord on a chording keyboard.

The desktop RA

An earlier desktop version of the RA is described in [3]. The desktop version runs in the background and watches whatever is typed or read in a word-processor. It then suggests old email, papers or other text documents that are relevant to the current text being displayed, and continuously updates a list of suggestions at the bottom of the screen.

The system has been in daily use for over two years now, and many applications have been found. For example, when a researcher writes a technical paper, the RA suggests email from reviewers, calls for papers for appropriate conferences, relevant class notes, papers written on the same topic, and abstracts from a database of IEEE journal articles that might be good references. With the touch of a key the entirety of the suggested document can be retrieved.

The wearable RA

Even with just the functionality of the desktop version, the RA is more useful on a wearable computer. For example, when taking notes at conference talks, the RA will often suggest documents that lead to questions for the speaker. Because the wearable is taken everywhere, the RA can also offer

suggestions based on notes taken during coffee breaks, where laptop computers cannot normally be used. Another advantage is that because the display is proactive, the wearer does not need to expect a suggestion in order to receive it. One common practice among the wearables users at conferences is to type in the name of every person met while shaking hands. There have been times when the RA has reminded the wearer that the person who's name was entered has actually been met before, and has suggested the notes taken from that previous conversation.

While useful, the system described above does not go far enough in using the physical sensors that could be integrated into a wearable computer. For example, a wearer of the system should not have to type in the name of every person met at a conference. Instead, the wearable should automatically know who is being spoken to, through active badge systems or eventually through automatic face recognition [4]. Similarly, the wearable should know its own physical location through GPS or an indoor location sensor. The Wearable RA system uses this physical context in finding information relevant to a wearer's situation.

Context information is used in two different ways. First, all notes written on the wearable are tagged with physical context information and stored for later indexing. In suggestion mode, the wearer's current physical context is also used to find relevant information. For example, meeting a person will automatically cause the RA to bring up notes taken while in a previous conversation with that person. If sensor data is not available (for example, if no active-badge system is in use) the wearer can still type in additional context information. The current version of the RA uses seven context cues to produce relevant suggestions:

- The information itself (the body of the note), which is turned into a word vector for later keyword analysis. In retrieval, this information comes from whatever the wearer is currently reading or writing on the heads-up display.
- Wearer's physical location. This information can be provided by GPS, an indoor location system, or a location entered explicitly by the user on the chording keyboard.
- People who are currently around. This information can come from an active-badge system, another person's wearable computer, or again can be entered by the wearer.
- Subject field, which can be entered by the wearer as an extra tag. When indexing files the subject can sometimes be extracted from header fields such as the subject line in email.
- Date-stamp, time-stamp and day of the week. These can be stamped onto note files with a single chord on the keyboard, or can be extracted from more structured data. In retrieval, this information comes from the system clock.

An example scenario makes the interaction of these context cues more apparent. Say the wearer of the RA system is a student heading to a history class. When she enters the classroom, note files that had previously been entered in that same classroom at the same time of day will start to appear. These notes will likely be related to the current course. When she starts to take notes on Egyptian hieroglyphics, the text of her notes will trigger suggestions pointing to other readings and note files on Egyptology. These suggestions can be biased to favour hieroglyphics in particular by setting the subject field to 'hieroglyphics'. When she later gets out of class and runs into a fellow student, the identity of the student is either entered explicitly or conveyed through an active-badge system or automatic face recognition. The RA starts to bring up suggestions pointing to notes entered while around this person, including an idea for a project proposal that both students were working on. Finally, the internal clock of the wearable gets close to the time of a calendar entry reminding the wearer of a meeting, and the RA brings up pointers to that entry to remind her that she should be on her way.

User Interface

Hardware

The wearable RA is currently running under Linux on a wearable 100 MHz 486 based on the 'Tin Lizzy' design developed by Thad Starner [5]. The input device is a one-handed chording keyboard called the Twiddler (made by Handykey), with which one can reach typing speeds of 30–50+ words per minute on a full-function keyboard. The output device is a Private Eye heads-up display, a 720×280 monochrome display, which produces a crisp 80×25 character screen. The display is currently worn as a 'hat top computer', with the view screen pointing down from the top right corner of the wearer's field of view (see Fig. 1). This mount position gives the wearer the ability to make full eye contact while still allowing access

Fig. 1. The heads-up display for the wearable platform.

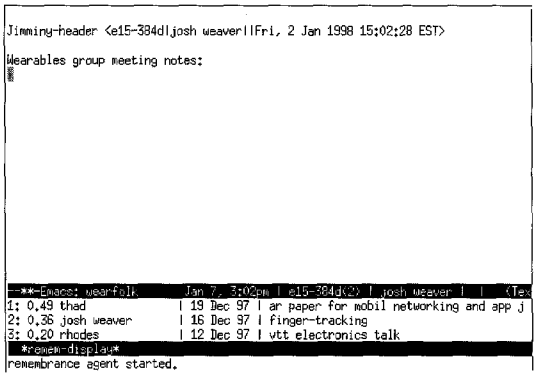

Fig. 2. A screen-shot of the wearable RA. The main screen comprises notes just starting to be taken during a wearables group meeting. The mode line shows the current context, namely the date, the room number and the personspeaking, with a bias to the location. The bottom lines show the RA's suggestions. The top suggestion was from another weekly group meeting, and so matches on room location, time of day and day of week.

to a full screen of information with a single glance. Others in the wearables group at the Media Lab have experimented with eyeglass mounts that provide an overlay effect of text or graphics on the real world [4].

The physical sensors on the wearable include a 'locust' infra-red receiver that communicates with location beacons placed around the MediaLab [6]. The computer's internal clock is used to determine date, time and day of the week. Subject and person fields are still entered by hand, though the latter could easily be automated with active badges.

Software

The software is running within Emacs, a commonly used text editor for Unix. RA suggestions are displayed in the bottom three lines of the screen, with a one-line summary for each suggestion. A summary line consists of the line number, relevance score for this suggestion, and the person, date and subject associated with this suggestion. Above the summary lines, the Emacs mode line shows the current date, time, location, person list and subject information that are considered to be the current context. Query biases, which change how different context features are weighted, are displayed in parentheses next to the features if set.

When context is changing, the suggestion list updates every 5 seconds. The top three suggestions are always shown, regardless of their relevance score. Sometimes a suggestion summary line can be enough to jog a memory, with no further look up necessary. However, often it is desirable to look up the complete reference being summarised. In these cases, a single chord on the Twiddler can bring a suggested reference in the main buffer. If the suggested file is large, the RA will automatically jump to the most relevant point in the file before displaying it. Other chords are defined to manually set context information not provided by a physical sensor, and to raise or lower the bias of a particular context feature. Because the Twiddler has over 2000 possible chords, it is simple to add quick-keys for all the common functions.

While the RA is designed to be a proactive reminder system, it also provides the functionality of a standard fuzzy-query database engine. By entering a query at an Emacs prompt, the wearer can override all current context cues with the new input.

Internals

Indexing documents

When a note is written, the location, person, subject and date tags are automatically attached to the note if they are available. If one of the four context tags are not available (for example, if no active badge is being used), then the writer of the note can enter the field by hand using the chording keyboard. The fields can also be left blank. New tags can be added easily, and it it hoped that as new sensors become available they can be

integrated into the existing system and replace the by-hand entry.

Sometimes these context fields can also be determined from a source other than physical sensors. For example, subject and person tags can automatically be extracted from email header information. In this case, the person field is the person who sent the email, not the person who happens to be with you when you read the email. If an indexed file is a structured source such as email or HTML, this is detected and the RA automatically picks out any information it can from the file. Multiple notes or documents can also be detected and broken out of a single file, as would be the case for an email archive file.

After being tagged, these notes, email and other information sources are indexed by the back end. First, each file in the indexed directories is identified as email, HTML, plain text, annotated note file, etc. Large files are broken up into smaller overlapping 'windows' so the topic of a single window will tend to remain focused. This window information is used later when bringing up the full text of a suggestion, so the RA can jump to the most relevant part. Next, common words in the body of a document are thrown out, and the remaining words are stemmed and converted into a word vector where the number of occurrences of each word in the indexed document is an element in the vector. Other features such as location, person, subject, date, time stamp and day of the week are also tokenised and labelled by type, and are stored in their own vectors. These vectors are stored in a database, organised by token for fast retrieval.

Determining suggestions

When running in its normal suggestion mode, the RA will convert the current screenful of text being looked at into a word vector, in the same way vectors are created during indexing. The other six content-information vectors are similarly created, using context variables typed in by the wearer or from physical sensors. Location, person, subject and day of the week are all represented as strings of text (e.g. 'Room E15-305D'). Dates and time stamps are represented as number of seconds since 1 January 1970.

For each of the pre-indexed documents, the RA will compare each query vector with each of the respective indexed vectors. The similarity between body, location, person, subject and day of week vectors is found using a text retrieval technique called Term Frequency inverse Document Frequency (TFiDF). The technique assumes that co-occurrence of terms in a document vector and the query vector indicates similarity. Words that are especially common in a document (high term frequency) are assumed to carry more weight, but words that are common in the entire indexed corpus carry less weight because they are less distinguishing (inverse Document Frequency). More rigorously, the similarity between two vectors of the same type is found by multiplying the document term weight for each term (word or feature) by the query term weight for that term, and summing these products. In notation, relevance = Σ_{vector} DTW · QTW, where

$$\text{DTW} = \frac{\text{tf} \cdot \log \frac{N}{n}}{\sqrt{\sum \text{vector} \left(\text{tf}_i \cdot \log \frac{N}{ni}\right)^2}}$$

and

$$\text{QTW} = \frac{\left(0.5 + \frac{0.5\text{tf}}{\max \text{tf}}\right) \cdot \log \frac{N}{n}}{\sqrt{\sum \text{query} \left(\left(0.5 + \frac{0.5\text{tf}}{\max \text{tf}}\right) \cdot \log \frac{N}{n}\right)^2}}$$

Term frequency (tf) is the number of times a term appears in a document, N is the total number of documents in the collection, and n is the number of documents to which a term is assigned. The method used is identical to the *tfc* · *nfx* TFiDF method used by Salton [7], with the addition of a normalised query vector to keep the final relevance score between zero and one. There is also an exception to Salton's algorithm: if the document term weight is higher than the query term weight then the document term weight is lowered to match the query term weight. This helps keep pathologically short documents from being unduly favoured.

Date and time stamp vectors are treated almost the same, except a fuzzy matching is used. The closer two times or dates are to each other, the higher the relevance score, following a logarithmic scale. These fuzzy similarities are then multiplied by each *DTW* · *QTW* product above.

This process leaves each document with seven relevance factors, one for each type of contextual information. These are merged together with biases based both on the kind of document (index biases) and the current query settings (query biases). Index biases are set in the template file, and represent the relative weighting of different kinds of information for that document type. For example, one

might decide that the subject line and body text are very important in email, but for an address database the person field is the most important feature. During indexing, each document has these biases saved to a file for later retrieval. Query biases are currently set by hand. If a piece of contextual information exists (i.e. if the sensor is available or the user enters the data), then the query bias defaults to one for that feature, otherwise it defaults to zero. However, with the touch of a chord these biases can be turned up or down. Given a query bias and an index bias for a specific context feature, the total bias for that feature is found by

$$\text{TotalBias}_i = \frac{\text{QueryBias}_i \cdot \text{IndexBias}_i}{\sum_i \text{QueryBias}_i \cdot \text{IndexBias}_i}$$

The total relevance score for a given document is calculated by $R = \Sigma_i \text{TotalBias}_i \cdot \text{relevance}_i$. The summary lines of the most relevant suggestions are displayed continuously on the bottom few lines of the heads-up display.

Related Work

Probably the closest system to this work is the Forget-me-not system developed at the Rank Xerox Research Centre [8]. The Forget-me-not is a PDA system that records where its user is, who they are with, who they phone, and other such autobiographical information, and stores it in a database for later query. It differs from the RA in that the RA looks at and retrieves specific textual information (rather than just a diary of events), and the RA has the capacity to be proactive in its suggestions as well as answer queries.

Several systems also exist to provide contextual cues for managing information on a traditional desktop system. For example, the Life streams project provides a complete file management system based on timestamp [9]. It also provides the ability to tag future events, such as meeting times, that trigger alarms shortly before they occur. Finally, several systems exist to recommend Webpages based on the pages a user is currently browsing [10,11].

Design Issues and Future Work

The physical-based tags are a recent extension to the RA, but the base system has been up and running on the wearable platform for over a year, and several design issues are already apparent from using this prototype. These issues will help drive the next set of revisions.

The biggest design trade-off with the RA is between making continuous suggestions versus only occasionally flashing suggestions in a more obtrusive way. The continuous display was designed to be as tolerant of false positives as possible, and to distract the wearer from the real world as little as possible. The continuous display also allows the wearer to receive a new suggestion literally in the blink of an eye rather than having to fumble with a keyboard or button. However, because suggestions are displayed even when no especially relevant suggestions are available, the wearer has a tendency to distrust the display, and after a few weeks of use our limited experience suggests that the wearer tends to ignore the display except when they are looking at the screen anyway, or when they already realise that a suggestion might be available. The next version of the RA will cull low-relevancy hits entirely from the display, leaving a variable-length display with more trustworthy suggestions.

Furthermore, notifications that are judged to be too important to miss (for example, notification that a scheduled event is about to happen) will be accompanied by a 'visual bell' that flashes the screen several times. This flashing is already being used in a wearable communications system on the current heads-up mounted display, and has been satisfactory in getting the wearer's attention in most cases. Another lesson learned from the interface for this communications system is that the screen should radically change when an important message is available. This way the wearer need not read any text to see if there is an important alert. Currently, the communications system prints a large reverse-video line across the lower half of the screen, which is used to quickly determine if a message has arrived.

Another trade-off has been made between showing lots of text on the screen versus showing only the most important text in large fonts. The current design shows an entire 80 column by 25 row screen, but this often produces too much text for a wearer to scan while still trying to carry on a conversation. Future versions will experiment with variable font size and animated typography [12].

Acknowledgements

I would like to thank Jan Nelson, who coded most of the remembrance agent back-end, and my sponsors at British Telecom.

References

1. Maes P. Agents that reduce work and information overload. Commun ACM 1994; 37(7).
2. Tulving E. Elements of episodic memory. Clarendon Press, Oxford, 1983.
3. Rhodes B and Starner T. Remembrance agent: a continuously running automated information retreival system. In: Proceedings of practical applications of intelligent agents and multi-agent technology. (PAAM), London, 1996.
4. Starner T, Mann S, Rhodes B, Levine J, Healey J, Kirsch D, Picard RW and Pentland A. Augmented reality through wearable computing. Presence 1997; 6(4).
5. Starner T, Mann S and Rhodes B. The MIT wearable computing web page. http://wearables.www.media.mit.edu/projects/wearables, 1995.
6. Starner T, Kircsh D and Assefa S. The locust swarm: an enviromentally-powered, networkless location and messaging system. In: First international symposium on wearable computers, Cambridge, MA 1997, 169–170.
7. Salton G. Automatic text processing: the transformation, analysis and retrieval of information by computer. Addison-Wesley, Reading, MA, 1988.
8. Lamming M and Flynn M. Forget-me-not: intimate computing in support of human memory. In: FRIEND21: International symposium on next generation human interface, Meguro Gajoen, Japan, 1994, 125–128.
9. Freeman E and Gelernter D. Lifestreams: a storage model for personal data. ACM SIGMOD Bull 1996.
10. Lieberman H. Autonomous interface agents. In: Proceedings of CHI-97, Atlanta, GA. ACM Press, New York, 1997.
11. Joachims T, Freitag D and Mitchell T. Webwatcher: a tour guide for the world wide web. In: International Joint Conference on Artificial Intelligence, August 1997.
12. Small D, Ishizaki S and Cooper M. Typographic space. In: CHI '94 Companion, 1994.

Correspondence and offprint requests to: Bradley J. Rhodes, MIT Media Lab, Cambridge, MA 02139, USA. Email: rhodes@media.mit.edu