1 Introduction

Although most retrieval systems are aimed at collections of documents such as research papers or reports, many people handle quite large amounts of information in the course of their working lives, and these collections of personal information can easily be large enough to become unmanageable without some kind of retrieval aid. Personal information retrieval is different from conventional retrieval in a number of ways. One major difference is that the required data (or at least something about it) was once known to the searcher and so the retrieval system will be most effective if it can help and be helped by the searcher's own memory. Another difference is the varied nature of the things to be retrieved. Some may be small, for example a name or phone number. Others may be quite long, for example, the contents of a research report. Some information is naturally available on-line, but other information, for example the identity of a visitor or a verbal communication, needs to be captured or indexed before it can be retrieved.

Personal information that we may want to retrieve falls into two broad classes – events and documents. Events are things that happened in the past and are the information that would be recorded if we kept a careful personal diary. A complete event record would include details of where we were, who we were with and what we were doing at any time in the past. Personal documents are normally text objects (though there are multimedia possibilities) and could be mail messages, research papers, reports and so on. The most natural way to retrieve events is by their time, either absolute or relative to other events. Typical queries might be *Who else was at the meeting last week in Bill's office?* or *Who was the student who came to see me last week after the algorithms lecture?*. In each case time, together with other parameters, is used to identify the event. Documents are conventionally retrieved by their content using Boolean or free-text queries. Although events and documents seem quite different, they are, in practice, closely related – for example, many documents have at least one associated event, which is our first becoming aware of its existence. Although the most obvious way to retrieve a document is by content, we could also find documents by looking at events. For example, I may want to find the mail message from a student that I read after the project meeting last month. Similarly, although events are most naturally retrieved by time or by their temporal relationship to other events, they could also be found by searching on content. A typical query combining content and time might be

> *Who was the visitor I had last week; I can remember reading the mail on . . . just before he arrived.*

There is a lot of psychological evidence[9, 10, 11] that we remember the temporal relationships between events. Hence, though we may not remember when we read a piece of mail, we may remember, for example, that it was on a Friday afternoon just before we left for the weekend. There is also evidence that seeing the context of a past event helps us to recall the event itself. This means that a personal document retrieval system that is designed to work in conjunction with our own memories will be more effective if it also incorporates events and their relationships. This paper desribes an event browser and its use for personal retrieval.

2 Other related projects

There have been two earlier projects aimed at using personal event histories for retrieval – one based at Loughborough university and one at the Rank Zerox Research Centre in Cambridge. These are described below.

2.1 The MEMOIRS project

This interesting project[5], by Lansdale and Edmunds, involved the construction and trial of an event based personal retrieval system. This was done by combining a document retrieval system with a personal diary so that documents could be retrieved, either by attribute, or by remembering contemporary diary events. In their prototype, all the documents were captured as bitmap images and needed indexing before they could be retrieved by attributes (other than time) or content. MEMOIRS also included a graphical event browser which has some features in common with the *hive* browser described in this paper. The main difference between the MEMOIRS browser and *hive* is that MEMOIRS is restricted to a diary and documents whereas *hive* is intended as a browser for general event based information, much of which would be captured automatically in the course of a working life. Another difference is that *hive* is designed to be scaleable and should be able to cope with collections with hundreds of thousands of events.

2.2 The Pepys and Forget-me-not projects

Mik Lamming and his colleagues at the Rank Xerox Research Centre in Cambridge, UK, have carried out a series of projects aimed at providing memory aids by automatically capturing very detailed information about workplace activity. Their main source of event information are *active badges*[3], which are battery powered identification badges which periodically emit unique identification sequences using infra-red. The badges are supported by a system of networked detectors which can receive badge signals and record them in a central database. The use of active badges makes it possible to record which room any individual is in and who they are with at any time; it can be a very rich source of event information.

The Pepys project[2] used active badge data to construct retrospective printed diaries that could be used as memory aids. Pepys used fairly sophisticated artificial intelligence techniques to construct high level events like meetings from low level badge sightings. It also used information about building geography to determine whether badge sightings were significant (for example, a sighting might be ignored if it was on the route from one set of sightings to another).

The goal of the Forget-Me-Not([4] and also [7]) project was the creation of a hand-held memory aid based on automatically collected workplace information. In addition to badge data, they also collected information about phone calls (by monitoring the PABX), electronic mail, printer use and file exchange[1]. A user who wanted to retrieve (or recall) information from the system would do so using a ParcTab[8], a small (128 by 64 pixel) hand-held terminal with a bi-directional infra-red wireless receiver.

3 The UKC Memory Repository project

The University of Kent is currently carrying out an EPSRC funded project to investigate techniques for the use of Memory Repositories (another name for event based personal retrieval systems). The aim of the project is to build on the work done at Xerox but using personal rather than corporate databases

[1] This, of course, involved getting permission from all concerned, and generally a sensitivity about invading people's privacy.

and using palmtop computers and manual input rather than relying entirely on automatically captured information.

3.1 Personal activity recording

One difficulty with the Rank Xerox approach is that it needs a great deal of centrally managed hardware and software – in particular it needs an active badge system with badge sensors in most private offices and public areas. This is expensive to set up unless the active badges are used for more than just aiding memory. A more fundamental problem is that it necessarily involves the central collection of very detailed information about workplace activity. This may be acceptable at Xerox but it is hard to imagine many people trusting their employer not to use the information for other purposes (staff appraisal for example).

The problems with centrally collected information do not arise if each individual collects their own event history. There are several ways to do this – in an academic or research environment, most of us spend a great deal of our working lives at a computer terminal, editing documents and communicating by electronic mail, and this information can easily be captured automatically. Hand-held personal computers (called Personal Digital Assistants, PDAs or Palmtop Computers) provide a practical way to capture information when away from the screen. Most PDAs have a desk diary which, if used conscientiously, will capture information about planned events like meetings or seminars. The PDA can also be used to make time-stamped notes which could later be retrieved by temporal context or by content. With the help of some purpose built software a PDA could easily be used to capture information about unplanned meetings etc. Finally, with the help of a bit of additional electronics, a PDA could automatically capture information. For example, a PDA with an infra-red detector could detect active badge sightings and so record who you have met. Although this would necessitate people wearing active badges, is does avoid the problems of central active badge monitoring. For people whose work involves a lot of travelling, a GPS receiver attached to a PDA could keep a continuous record of where you are. With the help of a microphone and some digital signal processing, a PDA could recognise voices (see [6] for example) and so keep a record of personal encounters, without the need for active badges.

3.2 High level and low level events

Although it is possible to capture detailed information about our working lives, this information is generally at a lower level than the events we remember or may want to retrieve. For example, I may think of myself as working on this paper (which I prepare at a computer terminal), whereas the best any monitoring software will record is that a particular file has been changed. Similarly, I am aware that I went to a meeting this morning but, unless the meeting is timetabled and in my diary, the best an automatic system could detect is that I was in a room with several other people. One approach to this problem is to use artificial-intelligence techniques to try and synthesise high level events from low level events. For example, the system could detect that several people arrived in a room at the same time and so deduce that there was a meeting – this is the approach adopted by Newman, Eldridge and Lamming in the Pepys project [2]. Reliable detection of high-level events is quite difficult to do in practice since the available information is always noisy and incomplete. For example, if the system depends on active badges to detect the presence of individuals then it will only work if people remember to wear their badges. Similarly, if badge detection is done by your PDA then I will only know the arrival times of people who arrived after I did. A potentially more robust approach is to display the raw low-level events (active badge sightings, file changes, etc.) but provide a sophisticated browsing interface that will make it easy for the searcher to spot high level events by seeing patterns in the low level events.

The *hive* event browser, which takes this low level approach, is a first attempt at a graphical event browser and is described in detail in the following sections.

3.3 Types of events collected

An experiment with browsing techniques will be most useful if the event database is as large and varied as possible. On the other hand, even experimental events need to be real since the whole point of using events for retrieval is lost if the events cannot be remembered. Hence, to provide a basis for experiments, I have collected events from my own working life during the past few months. At present, I have the following events.

File changes A log of changes to my own files (file name and time) on the UNIX system where I do all my research and prepare teaching material. This data goes back to November 1993 and currently contains about 75,000 events.

Mail messages Filed mail messages, both sent and received. These contribute about 2900 events.

Active badge sightings This is much less useful than in the Xerox work since we do not have many detectors installed but there a are nearly 1,000 badge sightings during the last six months.

Video snapshots These are taken by a video camera in my office at five minute intervals when monitoring software detects visitors – about 1,000 events altogether.

Desk diary entries There are about 1500 of these going back to November 1994.

Notes These are timestamped notes made of things I thought I might want to remember – 60 at the moment.

Weather reports and news headlines These are captured each day from the Daily Telegraph WWW server.

Calendar This is mechanically generated and includes days of the week and parts (morning, afternoon, evening, etc.) of the day. These artifical event sequences provide very useful context for retrieving other events. For example, I may remember that a particular event occured on a Thursday afternoon.

These events have very different attributes – the only thing they have in common is that each has a time. The file changes are instantaneous and the only other information associated with each event is the name of the file that changed. The video snapshot events are also instantaneous but each has an image that can be viewed. The badge events have a finite duration and a small amount of associated information (location and any other badge wearers present). Mail messages have no duration but have files of text which can be used for retrieval or displayed. Diary entries have a duration and searchable text. A browser needs to be able to display all these events and their relationships in as much or as little detail as is needed.

3.4 Requirements for an event browser

The most important requirement of any browser for a large and complex structure is that it should allow the user to build up a mental picture of the whole system by viewing parts of it and seeing how those parts fit together. To achieve this, the user should be able to see overviews of the whole structure and detailed views of parts of it and should be able to move flexibly between different views.

The following list seems to be a set of basic requirements for an event browser.

1. It must show the event time sequence and show clearly how different events are related in time.
2. It should provide flexible control of the time resolution at which events are displayed. For example, we may want to see all the events within an hour or within a day or a week or a year.
3. It should provide flexible control over which kinds of events are displayed, for example, we may just be interested in meetings or in file changes, or in file changes of a particular type.
4. It should provide fast searching of reasonably large collections of documents (tens of thousands at least).
5. The document searching and the time display should be integrated together so that, for example, I can find all my mail about 'palmtops' during the last month and see how the message times relate to the times of other kinds of event.

4 The hive event browser

A user browses or searches in *hive* by manipulating named sequences of events. On being run, *hive* initially displays just a few primary sequences representing different kinds of basic event (see figure 1) – in our case these represent file changes, mail messages, notes and the other event types listed in section 3.3. A user searches or browses by creating and displaying subsequences of these large basic sequences. Currently, *hive* provides three ways to create a new event sequence.

searching – generate a subsequence of an existing sequence containing those events which match a particular keyword. This is particularly useful with events that have some associated text.

expansion – expand the display to show event sequences based on hierarchical event names. This is only applicable to events that have hierarchical names, for example, file changes.

matching – extract a subsequence of one event sequence consisting of those events which have the same time as an event in another sequence.

A *hive* user who is trying to find a specific event (for example, *Where is the note I made about . . . when . . . ?*) uses the searching, matching and expansion operations to create a sparse subsequence containing the sought event. A user whose search query is more general (*What did I do last Tuesday?*) would browse by expanding and comparing sequences. The *hive* sequence operations are described in detail in the following sections and examples are given showing how they are used to do different kinds of search.

Hive displays each event sequence as a horizontal strip with the name on the left and the event times shown as marks on a time-scale on the right (see figures 1a and 1b). This approach has the advantage that it is intuitively simple and shows clearly when two events are at about the same time or when one event follows closely after another.

One requirement of all browsers for large collections is that they should be able to provide overview and detailed information and should enable the user to relate the two views. *Hive* provides this by allowing the user to continuously change the time resolution simply by holding down a mouse button. This has the effect of making the display appear to zoom in or zoom back in the time dimension. Figure 1b shows the same event sequences as figure 1a but with the time resolution increased to show just the first two days of October, 1996.

4.1 Searching

For events with textual content, one way to extract useful subsequences is by taking all the events that contain a given keyword or match a query statement. This is identical to searching in an ordinary bibliographic retrieval system but with the results of the search displayed graphically as a temporal sequence. *Hive* events can have any number of associated index terms and searching is implemented efficiently (using inverted files) so it is perfectly possible to index mail messages by every word they contain. Hence, I could generate a subsequence of *Mail* events containing the word *palmtop*, say. It is also possible to distinguish fields within messages so I could restrict the sequence to events containing *palmtop* in the subject line.

4.2 Sequence expansion

Some kinds of event have a natural hierarchical structure. This is particularly true of file changes events which may inherit their structure from the directory hierarchy. In our case, another class of events with a natural hierarchical structure are Mail messages, which are divided into folders. Similarly the calendar events are divided into days of the week which are subdivided into periods. For event types like these it is useful to be able to expand a sequence to show hierarchical subsequences. Hence, for example, I could expand the Mail sequence to show sequences of mail in each individual mail folder. Similarly I could expand the *Files* sequence to show changes to top level files or directories within my home directory.

4.3 Time matching

Although searching allows us to find events by content, and expansion lets us find events in a hierarchy, an event based retrieval system also needs to provide some way to locate events by their temporal relationship to other events. For example, I may want to find the note I made shortly after attending a seminar, or on a very wet day, or on a Thursday afternoon or on some combination of these. To do this, I really need to find the subsequence of known events (in this case, notes) that have some temporal relationship with another sequence of known events. *Hive* allows the user to combine sequences of events by extracting a subsequence consisting of those events in one sequence that are contemporary with an event from another sequence. This alone is not enough (I want to find the note I made after rather than during the seminar) and so *hive* also allows tolerance margins to be set, effectively widening each event in the second sequence, so that events will be allowed if they are sufficiently close to an event in the matching sequence.

4.4 Viewing individual events

Having used searching and browsing to find likely looking events, the user will need to be able to look at any documents associated with the events. The events are very different and so it is not feasible to build event viewing into the browser. Instead, *hive* allows each event to have an associated viewing command, which is stored as part of the database. Any event's command can be run by selecting the event (which is highlighted in red) and then selecting the **View** button from the *hive* menu. The command usually creates a separate window in which to display the event document. Although this approach involves storing a separate viewing command for each viewable event, this is a small price to pay for making the viewing completely independent of *hive* itself. *Hive* has no built in assumptions about the nature of the events it displays.

5 Examples

Here are some examples of the way that searching and browsing can be done by manipulating event sequences using *hive*.

Query 1 *I would like to find a note I made about this time last year about possible changes to the algorithms course.*

Search Create a subsequence of the *Notes* event sequence by searching on *algorithms*, then examine event in the sequence about a year ago.

Query 2 *What on earth did I do during the first week in May this year?*

Search Zoom in on the time display to show just the first week in May. Expand the *Files* sequence to see which top level files and directories where changed in that week. Selectively examine changes within directories. Look at any notes, mail or video snapshot events during that week.

Query 3 *After one of our Friday morning project meetings someone mentioned an interesting sounding book on the psychology of memory. I can't remember anything more about it but I am sure I made a note after the meeting.*

Search (Assume that just searching the *Notes* sequence on *memory* and *psychology* does not find it – *memory* turns up too many notes and *psychology* was not in the note.)

Search the *Diary* sequence to get a list of meetings. Expand the the Calendar to get a sequence of Friday mornings. Extract a *contemporary subsequence* of Friday mornings with a meeting and then use that to extract a subsequence of *Notes* made on such mornings. If necessary, this sequence could be searched in turn to get a subsequence of notes that mention memory, and were made on a Friday morning that also had a project meeting.

This search is illustrated in figures 2a and 2b. In figure 2, sequence (1) is the diary entries that mention meetings. Sequence (2) contains the Friday morning pseudo events, obtained by expanding the *Calendar* to give days of the week and then Fridays to give parts of the day (all the Calendar sequences except Friday mornings were then deleted from the display). Sequence (3) is obtained by matching and contains all the Friday mornings that coincide with a meeting. Finally, sequence (4) contains all the notes that where made during an event in sequence (3). Figure 2b shows the same sequences as figure 2b but has been zoomed-in to show the detailed events around the note made at 1100 on May 5th 1995.

Query 4 *I would like to retrieve some mail concerned with lectures which arrived while I was working on this paper.*

Search This example is illustrated in figures 3a and 3b. The directory containing the paper draft is in `texdocs/papers/hive` (sequence (1)). Although this directory could be expanded in turn to show changes to individual files, that would not be helpful in this case. I now want to extract a subsequence of *Mail* events that coincide with *hive* activity. Using time matching to extract a contemporary subsequence directly will only find mail messages that arrived at the exact moment a file changed but if I first set the **Before** search margin to 8 Hours, I can obtain the sequence of mail messages that arrived up to eight hours before changing a file (sequence (2)) . This should catch all the relevant ones since I do not usually edit for more than eight hours without saving what I have done. A simple keyword search can then be done to generate the sequence of mail messages that mention lectures (3).

6 Architecture

It is pointless having a personal search and browse tool unless it is quickly available when needed and quick to use. Hence, *hive* is written in C and uses the usual retrieval machinary of inverted files and

B-trees to provide searching. The database of event names and times is created when the database is built. It is represented as a multiway tree and is memory-mapped in the search program so that startup is almost instantaneous.

The database building program takes about ten minutes to run on an SGI Indy UNIX workstation but it can easily be run automatically each night. Input to the building program is a sequence of simple text records with one record per event. The only compulsory fields in an input record are the (possibly hierarchical) name and a date and time. Other optional fields are:

duration – in minutes for events that are not instantanious;

command – that can be run from within *hive* to display any object associated with the event;

search terms – there may be any number of these and they are used to extract subsequences by searching.

7 Conclusions

It is rather too early to say how effective event based personal retrieval really is. Certainly, the *hive* approach of combining and manipulating event sequences seems to provide an effective access to event based data and may be useful for other kinds of historical retrieval. Using the existing *hive* tools it is quite easy to construct and examine sequences of events satisfying quite complex criteria.

On the other hand, it is less easy to say how successfully the *hive* approach solves the problems of personal retrieval. Some functions provided by hive have (not surprisingly) found plenty of use – in particular the ability to do content searching on notes and mail messages and see the results located in a graphical time sequence. What needs more research is the proposition that the temporal relationships between past events can be an effective aid to retrieval. My personal experience in the short time it has been available has been that temporal relationships are less useful for retrieval than the combination of content and absolute time. That said, my own experience is only an informal sample of one – it would be useful to do experiments with a group of real users.

References

[1] Robert W Scheifler *and* Jim Gettis. The X Window System. *ACM Transactions on Graphics*. vol. 5, no. 2, April 1986, pp. 79-109.

[2] W M Newman, M Eldridge, M G Lamming. PEPYS: Generating autobiographies by automatic tracking. In *Proceedings of the second European Conference on Computer-Supported Cooperative Work* (L Bannon, M Robinson and K Schmidt (eds)). Kluwer Academic Publishers, Amsterdam, 1991.

[3] R Want, A Hopper, V Falcao *and* J J Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems* vol. 10, no. 1, 1992, pp. 91-102.

[4] M G Lamming, P J Brown, K Carter, M Eldridge, G Flynn, G Louie, P Robinson, A Sellen. The design of a human memory prosthesis. *Computer Journal*. vol 37, no. 3, 1994, pp. 153-163.

[5] M Lansdale and E Edmunds. Using memory for events in the design of a personal filing system. *International Journal of Man-Machine Studies*. vol 46, 1992, pp. 97-126.

[6] Herbert Gish and Michael Schmidt. Text-Independent Speaker Identification. *IEEE Signal Processing Magazine* vol ??, 1994, pp. 18-32.

[7] W M Newman and M G Lamming, *Interactive System Design* Addison-Wesley, 1995.

[8] N Adams, R Gold, W N Schilit, M M Tso and R Want. The PARC Tab mobile computing system. *Xerox PARC Technical Report* 1993.

[9] E Loftus and W Marburger Since the eruption of Mount St Helens, has anyone beaten you up? Improving the accuracy of retrospective reports with landmark events. *Memory and Cognition* vol 11, 1983, 114-120

[10] E Tulving. *Elements of Episodic Memory* Clarendon Press, 1983.

[11] My Memory: A study of autobiographical memory over six years. *Cognitive Psychology* vol 18, 1986, 225-252.

8 Figure captions

Figure 1a: The *hive* event browser showing event sequences over several months

Figure 1b: The top-level event sequences during the first two days of October 1995

Figure 2a: An example of event searching (see example 3)

Figure 2b: As figure 2a but showing a short time interval

Figure 3a: Another example of event searching (see example 4)

Figure 3b: As figure 3a but showing a period of three months