Remembrance Agent

A continuously running automated information retrieval system

(click for RA distribution page)

Bradley J. Rhodes
MIT Media Lab, E15-305
20 Ames St.
Cambridge, MA 02139
rhodes@media.mit.edu

Thad Starner
MIT Media Lab, E15-394
20 Ames St.
Cambridge, MA 02139
thad@media.mit.edu

Published in *The Proceedings of The First International Conference on The Practical Application Of Intelligent Agents and Multi Agent Technology* (PAAM '96), pp. 487-495.

Abstract

The Remembrance Agent (RA) is a program which augments human memory by displaying a list of documents which might be relevant to the user's current context. Unlike most information retrieval systems, the RA runs continuously without user intervention. Its unobtrusive interface allows a user to pursue or ignore the RA's suggestions as desired.

Introduction

The rise of the Internet has prompted a flurry of systems designed to find, filter, and organize the huge amounts of information now available. Information retrieval systems have been developed for applications ranging from classification and prioritization of email (Maes 1994, Cohen 1996), filter netnews (Lang 1995), answering queries based on Usenet FAQ's (Hamond 1994), and searching the Web (Mauldin and Leavitt, 1994). A few applications have also been developed to organize more user specific information such as notes files, diaries, and calendars (Jones 1986, Lamming & Flynn 1994). However, almost all of these systems have concentrated on query-based information-retrieval-on-demand. For example, they can answer questions such as "when is that conference paper deadline?" or "who's an expert on this particular algorithm?". However, they do not help when the user does not remember enough to even know to ask a question, or what question to ask. This "associative" form of recall is what reminds a user that an important conference exists at all, or that there are references that should be mentioned in a paper which the user might have missed.

With more powerful desktop computers, most of a computer's CPU time is spent waiting for the user to hit the next keystroke, read the next page, or load the next packet off the network. There is no reason it can't be using those otherwise wasted CPU cycles constructively by performing continuous searches for information that might be of use in its user's current situation. For example, while an engineer reads email about a project an agent might remind her of project schedules, status reports, and other resources related to the project in question. When she stops reading email and starts editing a file, it should automatically changes it recommendations accordingly.

The Remembrance Agent described in this paper performs this continuous, associative form of recall by continuously displaying relevant information which might be relevant to an individual user in their current context. It is designed with the philosophy that, as a continuously running and updating program, it should never

distract from the user's primary task, but should instead only augment it. It therefor suggests information sources which may be relevant to the user's current situation in the form of one-line summaries at the bottom of the screen. Here they can be easily monitored, but won't distract from the primary work at hand. The full text of a suggestion can be brought up with a single keystroke.

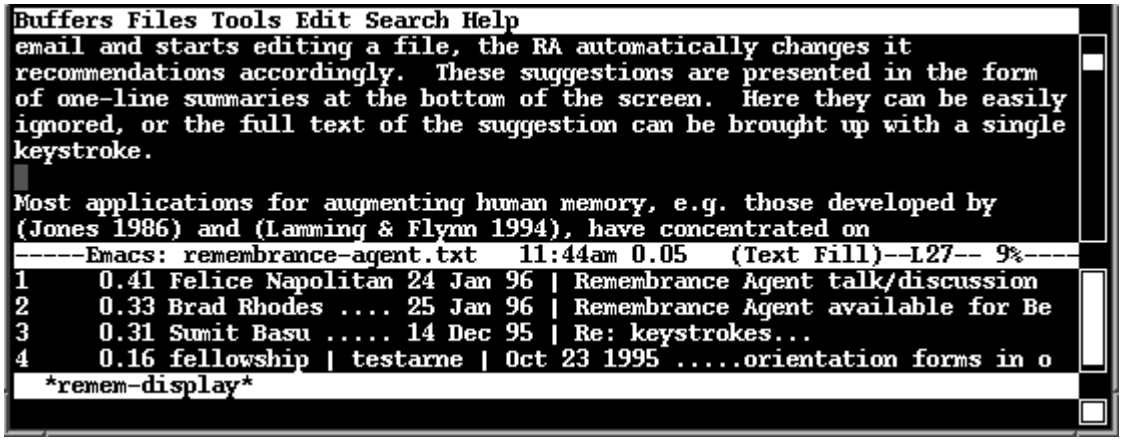

Fig. 1: A (small) screen-shot of the remembrance-agent in action

The Remembrance Agent: current implementation

The Remembrance Agent (RA) is broken into two parts. A front end continuously watches what the user types and reads, and sends this information to the back end. The back end finds old email, notes files, and on-line documents which are somehow relevant to the user's context. This information is then displayed by the front end in a way which doesn't distract from the user's primary task.

Currently, the front-end runs in elisp under Emacs-19, a UNIX based text editor which can also be used for applications such as email, netnews, and web access. It displays one-line suggestions at the bottom of the emacs display buffer, along with a numeric rating indicating how relevant it thinks the document is. These suggestions contain just enough info to relate the contents of the full document being suggested. For example, the suggestion line for a piece of email would contain who the email was from, when it was sent, and its subject-line. A notes file would contain the file-name, date last modified, the first few words of the file, and perhaps the owner of the file. With a simple key combination, the user can bring up the full text of a suggested document. The frequency with which the front end provides new suggestions, the number of suggestions, and whether to look at text notes files, old email, or other document sources is customizable for each user. The user can also customize how much of their current document the RA looks at when creating a suggestion. For example, the top display line may look at the last 500 words when making a suggestion, the two lines at the last 50 words, and the bottom line at the last 10 words. In this way, the RA can be relevant both to the user's overarching context and to any tangent they happen to be on.

The current implementation uses the SMART information retrieval program as a back end, which decides document similarity based on the frequency of words common to the query and reference documents (Salton & Lesk 1971). SMART indexes all the document sources nightly, and also supplies relevant documents based on the "query" text supplied by the front end. While SMART is not as sophisticated as many more modern information-retrieval systems, it has the advantage that it requires no human pre-processing of the documents being indexed.

Evaluation

One difficulty in evaluating a system such as the RA is that there is a large difference between relevant suggestions (those that have a strong relation to the user's current context), and useful suggestions. For example, if the user receives email about ballroom dancing and the RA suggests an announcement for a ballroom dance

event, that suggestion is highly relevant. However, if the event being announced occurred six months ago, that suggestion may be completely useless. Furthermore, a suggestion which might be useful in some situations might be more of an annoyance if the user is trying to concentrate on their primary task. For this reason, traditional methods of evaluation compare computer-generated relevancy scores with a human's opinion of relevancy (Harman, 1995) don't work well. Since the usefulness of the RA depends on the user's current task, what they already remember, how focused they are on their task, and a large number of other factors it can only be evaluated properly in the context of actual use.

The RA has been in alpha testing for about three months on a small number of platforms, so usability testing up to this point has been limited to a few users / developers. The beta-test version is now being released to a much larger user-base, so usability test will be starting shortly based both upon user surveys and logs of how often users bring up the full text of a suggestion. However, despite the small numbers of users in the alpha-testing, many lessons have been learned regarding both how the RA should be designed and to what areas it should be applied. These lessons are discussed in the next section.

Design Issues

There are three different ways of handling the timing of an automated task. The first is to perform a task only when specifically requested. Spell-checking a file and performing a web-search fall into this category. A second way is for an automated task to always lurk in the background, but to only act when a specific "trigger" occurs. Such programs include calendar programs that automatically tell you when you're going to be late for a meeting, and programs that alert you when you have new email waiting. Finally, there are tasks which are performed continuously, like a clock program or CPU-load meter. The RA falls into this third category.

There are several reasons for this design decision. As stated earlier, systems which only provide information "on request" can't help with associative recall. The user has to know that there is knowledge to be had in a particular situation. The RA therefore needs at least some ability to be proactive in its suggestions. However, unlike the calendar program that warns of upcoming meetings, it is impossible to create a back end which can reliably know when a document is useful for a user. Thus the RA will always suggest many false positives. It is also far less important that a user see an individual suggestion than it is for a calendar program. If the RA only displayed a suggestion when the relevance passed a certain threshold, the users' attention would be drawn away from their primary task whenever a new suggestion appeared. With the high ratio of false-positives, this would rapidly become a prohibitive distraction.

For these reasons the RA's suggestions are kept unobtrusive. There should be no (or at least minimal) cost to the user to see a suggestion and ignore it if deemed not useful at the time. Suggestions are therefore kept to a single line each, and are always printed at the bottom of the text-editor window. The full display area is also limited to a maximum of 9 lines, though it defaults to operating with only three or four. Finally, no color cues or highlights were used in the suggestion-display area, and the suggestions are only changed every 10 seconds or so to further avoid distraction from the primary task. However, suggestions must also pack enough information to allow users to judge their usefulness quickly. Especially important is a date field, as the age of a document is often the best indicator of whether it is useful in a given situation. In some cases, the description line supplies desired information directly without ever needing the full document. For example, it might provide the spelling of the last name of someone who recently sent email.

While suggestions are unobtrusive, it must be trivial both to access the entirety of a suggested document and to return to the primary document once the new one has been viewed. Anecdotal evidence suggests that any more than a second or two delay in bringing up a document is a serious barrier to regular use. In the current implementation, "control-c" and the display line-number brings up the suggested document. It is similarly easy to get back to the primary document.

Using the system has shown that suggestions are much more useful when the document being suggested only contains one "nugget" of information, and when that nugget is clearly displayed on its one-line description. This "less is more" approach solves several problems. First, it allows the user to tell what a suggestion contains from

its description without having to peruse the entire document. Second, a document with only one primary point is more likely to be a good hit. Documents which address several issues will rarely match the user's situation exactly, but will often partially match. Finally, if a suggested document is read, the shorter it is the quicker the user can get on with their primary work. Email and short notes files seem to be a very good length for RA suggestions. In future versions of the RA, longer documents will be broken up into pieces during indexing so that a suggestion can relate to a particular piece.

Personal email and notes files are a good source of documents for other reasons as well. Most importantly, they are both pre-personalized to the individual user and automatically change as the user's interest changes. This is one of the more important features of the RA, because it means that through proper choice of data to index the RA automatically adapts to the individual user. This guarantees that the pool of suggestions is much richer in potentially interesting information than, for example, the Web, and also helps solve many synonym problems. For example, when an AI researcher mentions "agents" in a paper, that person's RA will suggest work on autonomous agents. A chemist's RA, on the other hand, will bring up lists of various solvents and other chemical agents.

Some of the future applications discussed in the next section require using non-personalized indexes, which is analogous to using another person's memory. However, there are disadvantages beyond those discussed above. For example, it is harder to judge the content of a suggestion from its one-line summary when the user is less familiar with it. The user might also not be able to properly judge the validity of an unfamiliar suggestion. For example, they might not know that a set of instructions were made in jest, or were in a subsequent message shown to be incorrect. Similarly, documentation or lectures geared towards one kind of audience may be inappropriate for another, so it helps when the original context is well understood. Finally, there are privacy issues when using someone else's memories. Even if one accesses someone's email files with express permission, the people who sent that person mail might not have approved.

Future Applications

Wearable computing

When running on a desktop computer, an RA can only guess the user's context based on the document they are reading or editing. However, the advent of wearable computing (Starner et al. 1995) will allow RAs to work with much more information and many more situations. Global Positioning Systems (GPS) will let the RA know where the user is, while camera and face recognition will let it know who they're talking to. With this extra information, a (greatly enhanced) RA could know that it is around lunch time, that according to camera input the user is with her lunch date, according to her appointment book she has an appointment with her boss in an hour, and according to her GPS she's downtown. From this information it could recommend several good restaurants known for their fast service that are within a few blocks of downtown, which would be agreeable to her lunch date as well.

Reference advisor for technical papers

Another application currently being pursued is to have the RA suggest technical reports on a given subject. When it recommends other researchers' papers, these could be referenced or tagged for later reading. When it recommends one's own old papers, these can be scanned for similar material which might be used in the new paper. Such a system could also recommend conferences where the call-for-papers is similar in content to one's own paper.

Knowledge transfer

One of the large difficulties facing industry is bringing new members of an existing work-group up to speed quickly. If a work-group created its own knowledge-base, new members could access the group Remembrance Agent. This would not only give the employee access to the group knowledge itself (as would any database), but

also to the meta-knowledge of when particular information is relevant or valuable. Similar applications would exist wherever just-in-time training is required. While this application uses a knowledge base not familiar to the user at first, they will not suffer too greatly from the lack-of-context problem discussed above because the knowledge is focused on their current situation, namely their new position within the group. Any context they do not yet know, they need to learn anyway, and the RA will help them learn it.

Automatic Hypertexting

In the past year several on-line magazines have appeared, such as HotWired (HotWired), many of which have paper counterparts. One of the values added by the on-line versions of these magazines are hyper-linked text, where a reader can click on a word and get more information on that subject. A future RA could conceivably automatically turn normal email, netnews, or papers into hyper-linked documents, automatically linking hot-words to relevant background information.

Background checker

Another Web-based RA could perform background checks on people sending mail, referenced in papers, or recognized by a wearable-computer-mounted camera. Such an RA could automatically provide information on a person's employer, job title, phone numbers, and profiles of their interests based on newsgroups they frequent. At the click of a button, the user could access that person's home-page for even more information.

Other future work

The current RA is only personalized through using personal data as the suggestion pool, and by reindexing that data every night. However, it cannot learn which of these suggested documents are actually useful. To solve this problem, the back-end is being provided with an algorithm which learns based on which suggestions the user asks to have displayed. This way useless suggestions can be culled out over time and new ones tried.

Related work

While most information retrieval efforts have focused on query-response tasks (Harman, 1995), a few systems exist which continuously present the user with suggested web-documents based on the user's interests and the web-document currently being viewed. WebWatcher (Armstrong, Freitag, Joachims, and Mitchell, 1995) is one such system, though it requires explicit interaction to indicate interest in certain topics. Another continuously proactive web-browsing system is Letizia (Lieberman, 1995), which bases it's suggestions entirely on the user's past browsing activity and links from their current page. Letizia suggests pages within a few links of the current page, and brings them up in a separate window for the user to view or ignore as required.

References

R. Armstrong, D. Freitag, T. Joachims, and T. Mitchell, 1995. WebWatcher: A Learning Apprentice for the World Wide Web, in AAAI Spring Symposium on Information Gathering, Stanford, CA, March 1995.

W. Cohen, 1996. Learning Rules that Classify E-mail, in AAAI Spring Symposium on Machine Learning in Information Access, Stanford, CA, March 1996.

D. Harman (Ed.). (1995). The Third Text REtrieval Conference (TREC-3). National Institute of Standards and Technology Special Publication 500-225, Gaithersburg, Md. 20899.

K. Hammond, R. Burke, and K. Schmitt, 1994. A Case-Based Approach to Knowledge Navigation, in Working Papers of the AAAI Workshop on multi-media and artificial intelligence, July 1994.

HotWired. http://www.hotwired.com/

W. Jones, 1986. On the Applied Use of Human Memory Models: the Memory Extender Personal Filing System. The International Journal of Man-Machine Studies 25, 191-228

M. Lamming and M. Flynn, 1994. " Forget-me-not:" Intimate Computing in Support of Human Memory. In Proceedings of FRIEND21, '94 International Symposium on Next Generation Human Interface, Meguro Gajoen, Japan.

K. Lang, 1995. NewsWeeder: Learning to filter netnews, in Machine Learning: Proceedings of the Twelfth International Conference, Lake Taho, CA, 1995.

H. Lieberman, Letizia: An Agent That Assists Web Browsing, International Joint Conference on Artificial Intelligence, Montreal, August 1995.

P. Maes, 1994. Agents that Reduce Work and Information Overload. Communications of the ACM 37(7):30-40.

M. Mauldin and J. Leavitt, 1994. Web Agent Related Research at the Center for Machine Translation. Presented at the SIGNIDR meeting, August 4, 1994, McLean, Virginia. http://www.lycos.com/

G. Salton, ed. 1971. The SMART Retrieval System -- Experiments in Automatic Document Processing. Englewood Cliffs, NJ: Prentice-Hall, Inc.

T. Starner, S. Mann, B. Rhodes, J. Healey, K. Russell, J. Levine, and A. Pentland, 1995. Wearable Computing and Augmented Reality, Technical Report, Media Lab Vision and Modeling Group RT-355, MIT

C. Wickens, 1992. Engineering Psychology and Human Performance. Scott Foresman Little Brown.