Just-In-Time Information Retrieval

by
Bradley James Rhodes

S.B. Computer Science
Massachusettes Institute of Technology
Cambridge, MA, 1992

S.M. Media Arts and Sciences
Massachusetts Institute of Technology
Cambridge, MA, 1996

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in Partial Fulfillment of the requirements of the degree of

DOCTOR OF PHILOSOPHY
at the
Massachusetts Institute of Technology
June, 2000

Signature of Author _____
Program in Media Arts and Sciences
May, 2000

Certified By _____
Professor. Pattie Maes
Associate Professor of Media Arts and Sciences

Accepted By _____
Stephen A. Benton
Chairperson
Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Just-In-Time Information Retrieval

by
Bradley James Rhodes

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning in May 2000,
in Partial Fulfillment of the requirements of the degree of
DOCTOR OF PHILOSOPHY in Media Arts and Sciences

Abstract: This thesis defines Just-In-Time Information Retrieval agents (JITIRs): a class of software agents that proactively present potentially valuable information based on a person's local context in an easily accessible yet non-intrusive manner. The research described experimentally demonstrates that such systems encourage the viewing and use of information that would not otherwise be viewed, by reducing the cognitive effort required to find, evaluate and access information. Experiments and analysis of long-term use provide a deeper understanding of the different ways JITIRs can be valuable: by providing useful or supporting information that is relevant to the current task, by contextualizing the current task in a broader framework, by providing information that is not useful in the current task but leads to the discovery of other information that is useful, and by providing information that is not useful for the current task but is valuable for other reasons. Finally, this research documents heuristics and techniques for the design of JITIRs. These techniques are based on theory and are demonstrated by the field-testing of three complete systems: the Remembrance Agent, Margin Notes, and Jimminy. Specifically, these heuristics are designed to make information accessible with low effort, and yet ignorable should the user wish to concentrate entirely on his primary task.

Thesis Supervisor: Prof. Pattie Maes
Title: Professor of Media Arts and Sciences

This work was supported in part by British Telecom, Merrill Lynch and the Media Lab’s Digital Life consortium.

Just-In-Time Information Retrieval

by
Bradley James Rhodes

The following people served as readers for this thesis:

Reader

Walter Bender
Senior Research Scientist
MIT Media Laboratory

Reader

Dr. Elizabeth Mynatt
Assistant Professor
College of Computing
Georgia Institute of Technology

To my parents

and to all my teachers

both formal and informal

Serendipity is too important to be left to chance.

Acknowledgements

There are so many who have made this work possible that one page can not list them all.

First and foremost I would like to thank my advisor Dr. Pattie Maes, who since my undergraduate thesis almost ten years ago has been my inspiration and role model for how to do good research. She has always given good advice and direction while still allowing me to go my own way, usually only to realize six months later that she was right all along. I am incredibly fortunate to have had her as an advisor all these years.

Next I would like to thank all my teachers through the years, formal and informal. First my family, who have offered loving support and taught me the joy of finding things out. They have always supported me in whatever strange place I find myself, and have given me the confidence to muddle my way out knowing that I am not alone. Also my formal teachers through the years, especially Dr. Dean Karlovitz, Dr. Gus Baird and Don Steel, who gave me a love for math and computer science, and Dr. Jones and Dr. Pat Kelly who gave me a love for philosophy. Special thanks goes to Steve Schneider for starting my on this path early on and form helping me make minor corrections along the way, and to Dr. Marvin Minsky for introducing me to the field of Artificial Intelligence.

I would also like to thank my fabulous thesis and generals committee members Walter Bender, Dr. Beth Mynatt and Dr. Chris Schmandt. They have all provided invaluable feedback and have played a key role in shaping this document. I would also like to thank the DCGS members who have helped in the preparation of this work, most especially Dr. Brian Smith and Dr. Hiroshi Ishii. Many of the insights in this work come from conversations with these faculty.

I have also been blessed with a host of brilliant collaborators who have brought new insights into this work. First I would like to thank Dr. Thad Starner, who wrote the first Remembrance Agent as a class project and who introduced me to the concept of wearable computing. I could not ask for a better collaborator, and a large number of the ideas in this research came from interactions with Thad. Dr. Barry Crabtree of British Telecom has also been directly involved with the research over the years. Barry has taken the role of sponsor to new levels with his involvement, and the results have been spectacular for all involved. Dr. Lenny Foner was not only a source of insight and wisdom but was also a good office-mate and sanity-check when deadline pressures mounted. Nelson Minar is the primary developer of the HIVE system which is now integrated with Jimminy, and has also been good for bouncing around ideas. Dr. Dan Ariely has given good advice about evaluations and the project as a whole. I have also benefited from conversations with Sunil Vemuri, Dr. Henry Lieberman, Dr. Alan Wexelblat, Michael Schrage, Neil Van Dyke, Dr. Ted Selker, Richard DeVaul and Cameron Marlow, and all the students of the Software Agents group over the years.

Those of you who know MIT realize no work is completed without a team of undergraduates from the UROP program. I have been blessed with some incredible UROPs. Bayard Wenzel and Jan Nelson started the project with me and brought the first version of the RA kicking and screaming into the world. Jan also implemented the first version of Margin Notes, which was then brought into beta by Jesse Koontz. Finally Matthew Burnside, Pammie Mukerji, Jesse Rabek, Alex Park and Aneel Nazareth did a stellar job bringing all the systems into the quality code that is now being used by hundreds of people around the world.

A special thanks is reserved for Linda Peterson, who makes everything at the Media Lab run smoothly. She has guided me throughout the ins and outs of MIT requirements, and has seen me safely to shore.

This document was prepared in Frame Maker 5.5 on the Linux platform. Times Roman typeface is used for body text. Pictures and screen captures were as JPG files. Additional diagrams were done with Frame Maker's internal drawing tools.

Table of Contents

List of Figures

CHAPTER 1 *Introduction*

So, to a writer happily engaged on his work and excited by it, there may come a curious extension of his ordinary faculties; he will find portions of knowledge floating back into his brain, available for use, which he had supposed to be thrown away long ago on the rubbish-heap outside the back door of his mind; relevant passages will quote themselves to his mind from books he scarcely remembers to have ever read; and he suddenly sees germane connections where in his ordinary state of mind he would see nothing.
– *C.E. Montague,* A Writer's Notes On His Trade

1.1 What is a Just-In-Time Information Retrieval Agent?

This thesis introduces Just-In-Time Information Retrieval (JITIR) agents: software that proactively retrieves and presents information based on a person's local context in an accessible yet non-intrusive manner. JITIRs (pronounced "jitter") continuously watch a person's environment and present information that may be useful without any explicit action required on the part of the user. The environment that a JITIR monitors is usually computational: email, a web page a person is reading, or a document he is writing. However, it can also be a person's physical environment as sensed by cameras, microphones, Global Positioning Systems (GPS) or other sensors. The information a JITIR provides can come from any number of pre-indexed databases of documents, e.g. email archives, notes files, or documents from commercial databases such as the INSPEC collection of technical paper abstracts.

The three necessary features of a JITIR are proactivity, the presentation of information in an accessible yet non-intrusive manner, and awareness of the user's local context. JITIRs are similar to search engines, alarms and personalized news software, although they differ from each. The similarities and differences are discussed below.

Proactive: Search engines and structured knowledge bases such as Yahoo! are inherently interactive: an information seeker has some query in mind and directly interacts with the system to obtain the desired information. JITIRs, on the other hand, are proactive. The user need not have a query in mind, or even know that information relevant to his situation exists. This proactivity has ramifications for the information-retrieval techniques that can be used, because the "query" used to find useful information is limited to what can be sensed in the environment. Also, the interface must be carefully designed so that unrequested information does not become a distraction from the user's primary task.

Non-intrusive, yet Accessible: When a cell phone rings it provides the information that a person is calling, and may also indicate the identity of the caller by the tone of the ring. A ringing telephone intrudes on one's life: it distracts from whatever task is currently being performed and cannot easily be ignored. If the cell phone is turned off then the caller is often forwarded to voicemail. A silent cell phone is extremely non-intrusive, but without the ringer it is necessary to call the voicemail service directly to

determine whether anyone has called. The information about who called (if anyone) is less accessible than when it was indicated by the ring. JITIRs are designed to operate in between these two extremes.

Alarm systems in the desktop environment are similar. For example, the ICal calendar program for Linux makes a sound and pops up a modal dialogue box a few minutes before an event. The interface is designed to ensure that the user sees and acknowledges the warning. The information is very accessible, but by design it is difficult to ignore an alert. JITIRs will present information in such a way that it can be ignored, but is still easy to access the information should it be desirable. Rather than presuppose whether or not a particular piece of information is important or urgent, JITIRs allow the user to decide whether to view or ignore it depending on his current task and level of cognitive load.

Locally Contextual: Notification systems such as newspaper clipping services and alerts are proactive, but the information they present is based on events outside of the user's local context. For example, an alert might trigger whenever a new piece of email arrives, a stock price goes below a certain threshold, or news that fits a user's personal profile hits the news wire. The notifications are designed to pull a person out of his current context (task) and provide information about a different context that might require his attention. The urgency of a notification can range from the immediacy of a fire alarm to a news briefing that is announced, but intended to be read whenever convenient.

Notification systems present information from a rapidly changing source (e.g. current stock prices), based on relevance to a static or slowly changing user profile. JITIRs are the reverse: they provide information from a source that may or may not be changing (e.g. email archives) based on relevance to a user's rapidly changing local context. Informally, local context is the user's spatially local environment, including his current task. A more formal definition of local context is given in Chapter 3.2.4. Information provided by a JITIR is not meant to pull a person out of his current context, but rather to add additional information that might be useful *within* that context.

In summary, JITIRs are similar to search engines, alarms and notification systems, but none of these systems have all three features necessary for JITIRs: proactivity, a non-intrusive yet accessible interface, and attention to local context.

Note that automatic help systems such as the Microsoft Office Assistant (also known as "that infernal paperclip") fit the definition of a JITIR. However, these help systems are domain specific; they only provide information from a specialized or hand-designed help database, using information-retrieval techniques that are specific for that particular help domain. The purpose of this research is to discover design techniques that can be applied broadly. To this end, the techniques used are designed for generality and extensibility. However, the generality of a system is a trade-off. Clearly no system can support every possible domain and environment, and usually the careful use of domain-specific information can improve performance. For example, a JITIR designed to help someone read email might be able to follow discussion threads, understand mailing lists and how they relate to topics, and know when multiple usernames map to the same person. Such a JITIR would be better suited for the email domain than would a general-purpose system. Such techniques would also not be transportable to other domains.

Note also that the word “agent” has many definitions. JITIRs are *software agents* in that they are long-lived, watch an environment and can take action based on that environment without direct user intervention. They should not be confused with *embodied conversational agents* or *synthetic characters*, which are graphical interactive characters that present a personality and interact with a user in an animistic way **(Cassell 1999)**. They should also not be confused with *distributed agent architectures* or *agent-oriented programming models* **(Martin 1999)**, which are both architectures for software design rather than a class of applications.

(Cassell 1999)
Cassell, J. et al (eds) *Embodied Conversational Agents*, MIT Press, 1999

(Martin 1999)
Martin, D. et al, The Open Agent Architecture: A framework for building distributed software systems. *Applied AI*, 13(1-2), 1999, pp. 91-128

1.2 Contributions

This research makes four main contributions. First, it defines JITIRs: a class of software agents that proactively present potentially valuable information based on a person's local context in an easily accessible yet non-intrusive manner. Second, it experimentally demonstrates that such systems encourage the viewing and use of information that would not otherwise be viewed, by reducing the cognitive effort required to find, evaluate and access information. Third, through experiments and analysis of long-term use it provides a deeper understanding of the different ways JITIRs can be valuable: by providing useful or supporting information that is relevant to the current task, by contextualizing the current task in a broader framework, by providing information that is not useful in the current task but leads to the discovery of other information that is useful, and by providing information that is not useful for the current task but is valuable for other reasons. Fourth, this research documents heuristics and techniques for the design of JITIRs, based on theory and demonstrated by the field-testing of complete systems. Specifically, these heuristics are designed to make information accessible with low effort, and yet able to be ignored should the user wish to concentrate entirely on his primary task.

This section both outlines the contributions of this research and acts as a summary for the chapters that follow. It is followed by a road map for the rest of the document.

1.2.1 Thesis Questions

The three main questions addressed in this dissertation are:

- *How does the use of a JITIR affect the way people seek out and use information?*
- *How can a JITIR automatically find information that would be useful to a person by sensing that person's current local context?*
- *How should a JITIR present potentially useful information?*

This work is interdisciplinary, so the answers to these questions relate to several fields. The first question relates to behavioral psychology and cognitive ethnography. The goal is to determine both quantitative effects, e.g. an increase in the amount of information viewed when using a JITIR, and qualitative effects, e.g. a writer using direct quotes instead of paraphrasing. The second question draws from the fields of information retrieval (IR) and machine perception. Rather than invent yet another information retrieval algorithm, the implementations described in this thesis use a general and extensible IR framework which recognizes a user’s local context and uses plug-in functions that execute different IR algorithms based on that context. The third question is one of interface design and human factors, and also relates to theories of cognitive overload and focus of attention. The three implementations presented have different interfaces, but the design heuristics and techniques described are common to all three and apply broadly to other designs as well.

1.2.2 Overview of Implemented Systems

This research is application-based, and many of the findings come directly from the implementation and use of working systems. Three JITIRs have been implemented and deployed. The first and oldest is the Remembrance Agent (RA), a JITIR that operates within the Emacs text editor. Because Emacs is used for a wide variety of tasks including email, net news and word processing, this version is designed to be especially adaptive to the user's current task environment. The second implementation is Margin Notes, a web-based agent that automatically annotates web pages as they are being loaded into a browser. Margin Notes demonstrates a different task environment and different interface design techniques than those demonstrated by the RA. The third system is Jimminy (as in the cricket), also called the Wearable RA. Jimminy presents information via a head-mounted display attached to a wearable computer. The information provided is based on the wearer's physical environment: where he is, who he is talking to, the time of day, etc. The system demonstrates how a JITIR can be applied "off the desktop."

All three systems use the same information retrieval back end, called Savant, which was designed especially for this research. Savant consists of both a document indexer and a retrieval engine. The indexer uses a template structure to index documents based on file type. For example, it can identify a mail archive file and index individual email documents based on the from field, subject line, date and body of the message. The retrieval engine receives text from the user's environment (email being read or written, web page being read, etc.) and returns a list of documents that are most likely to be useful given that environment. The front ends for the RA, Margin Notes and Jimminy further process this information and determine how it should be displayed. Savant supports what is called *data fusion*: it can parse information from multiple data types including GPS, timestamps and raw text, and combine the retrieval results of multiple algorithms that use a combination of these fields. Savant can be easily extended to include more types of fields and similarity metrics.

1.2.3 Theory

The theory presented in this thesis focuses on the three thesis questions listed above.

(Engelbart 1962)
Engelbart, D., *Augmenting Human Intellect: a conceptual framework*. AFOSR-3233, 1962

(Newell 1989)
Newell, A. et al. Symbolic Architectures for Cognition, in *Foundations of Cognitive Science*, Posner, M. (ed), 1989, pp. 93-131

(Hutchins 1995)
Hutchins, E. *Cognition in the Wild*, 1995

The question "how does the use of a JITIR affect the way people seek out and use information?" is best understood in terms of *intelligence augmentation* **(Engelbart 1962)**; the extension of a person's mental ability beyond natural levels. Intelligence can be viewed in two ways. Classical cognitive science views intelligence as a property of an individual **(Newell 1989)**. Within this framework, questions revolve around how a tool can aid completion of a predefined task. An alternative view is that intelligence is the property of a system that includes an individual, her culture, and physical and cognitive tools available to her **(Hutchins 1995)**. With the latter view the focus shifts to how a tool, as a part of a larger systemic intelligence, affects the behavior of that system and the individual. In particular, it encourages thinking about tools not only in terms of how they affect a given task, but in terms of how they change what tasks are performed and how often. Both perspectives are useful for certain kinds of research. Because the thesis question being examined is about use of JITIRs in real settings, and because practical experience has shown that the environment has a large effect on how JITIRs are used, the view of intelligence as a property of an entire system will be dominant in this research.

(Payne 1993)
Payne, J. et al. *The Adaptive Decision Maker*, 1993

An economic model of human behavior predicts that a person will act to maximize her benefit while minimizing costs. One such model is the effort-accuracy contingent decision framework proposed by Payne, Bettman and Johnson **(Payne 1993)**, which predicts that a person will choose the decision-making strategy that achieves the accuracy needed for the task at hand while minimizing the cognitive effort required. In this framework, a search for information will not be performed if it is too much effort or if

a sufficient increase in accuracy is not expected. Because using the information provided by a JITIR is low effort, it is expected that a JITIR will be used in situations where a search engine would not.

The question “how can a JITIR automatically find information that would be useful to a person by sensing that person's environment?” is best informed by the field of Information Retrieval (IR). Information Retrieval, and particularly the subfield known as Document Retrieval or Text Retrieval, is concerned with developing technology to find documents from a corpus that are relevant to a given query. A JITIR can be thought of as an information retrieval engine where the “query” is automatically generated based on a person’s local context. One possible way to generate such queries is to use domain-specific techniques. For example, the FIXIT system **(Hart 1997)** is a JITIR that is built into an expert system that helps copier-repair technicians. FIXIT looks at the technician’s local context (in this case the symptoms that have been logged in the expert system) and displays pages from the repair manual that might be helpful based on the hierarchical structure of the index for the particular repair manual used. When such domain-specific criteria are not available (i.e. when the database or environment do not have a structure which is known in advance) then general text and information-retrieval techniques can be used to find documents that are in some way similar to the current environment, in the hopes that this similarity is a good predictor of usefulness.

(Hart 1997)
Hart, P. and J. Graham. Query-free Information Retrieval. *IEEE Expert / Intelligent Systems & Their Applications*, 12(5), 1997

Information retrieval techniques were primarily designed for the retrieval of library documents, and more recently web pages. The requirements for query-free retrieval are similar to those in the library domain, but differ in a few important ways. First, search engines often have an implicit *prior* of what kind of data is useful. A prior is domain knowledge that can be used to predict that some pieces of information will be more useful than others. For example, if a person searches the INSPEC database of journal abstracts, this search gives a good hint that he wants a journal reference. With a JITIR there is no explicit choice of database, and therefore fewer priors are available. Another difference is that web searches tend to be short, leading IR researchers to use techniques to expand queries beyond the few words given. JITIRs have the reverse problem: the entire environment is potentially a query, but only some parts of that environment will lead to useful information. Finally, traditional IR systems are evaluated based on how relevant the results are to a given query. Because a JITIR query is automatically generated, the resulting documents must be evaluated based on the more difficult criteria of *utility*. While relevance is still a useful metric to examine, the success of a JITIR must in the end be evaluated based on its usefulness to a particular user in a particular environment.

The third question “how should a JITIR present potentially useful information?” falls in the domain of interface design. The interface for a JITIR must be designed so it does not distract from a person’s primary task. It must be *non-intrusive*. However, it cannot be so non-intrusive as to never be noticed. It must also be *accessible*; it must be easy to switch between the primary task and the information being provided. The first criterion requires that the JITIR does not distract from a user’s primary task when concentration is necessary. Theories of focus-of-attention indicate that it is easier to ignore distracting elements in the environment when they are different from the target stimuli **(Wickens 1992)**. For example, it is easier to drive while listening to the radio than while reading a map, because driving and the radio use different modalities. However, theories of divided attention indicate that it is easier to switch between tasks when the tasks are more *similar* **(Allport 1993)**. In particular, it is easier to time-share between tasks or information sources that share similar mental constructs and schema. These two heuristics combine to form the *proximity compatibility principle* **(Wickens, p. 98)**: which when applied to JITIRs means information provided by a

(Wickens 1992)
Wickens, C. D., *Engineering Psychology and Human Performance*, 1992, pp. 375-382

(Allport 1989)
Allport, A., Visual Attention, in *Foundations of Cognitive Science*, Michael Posner (ed.), 1989

JITIR should be similar in display and structure to parts of the environment to which it is similar mentally, and dissimilar otherwise. For example, suggestions from a JITIR should be near the things they annotate or otherwise describe. This use of spatial proximity makes it so only a small amount of effort is required to switch attention away from the primary task and back again. Suggestions should also be dissimilar to parts of the environment to which they do not relate. For example, if a JITIR displays information related to text written in the word processor it should not have an interface that is similar in look and feel to the web browser. Finally, suggestions should always be separable from aspects of the environment that require a large amount of focused attention. For example, the interface for Jimminy uses the visual modality because it is designed primarily for use in conversations and lectures, where the task requires a large amount of attention in the audio modality.

Three assumptions can be made about suggestions produced by a JITIR. First, they will never be useful one-hundred percent of the time. Even with perfect information retrieval there are times when a user does not want more information, or is already suffering from information overload and cannot be distracted further. Second, the user is in the best position to determine if a particular suggestion will be useful, assuming she is given information about the contents of the suggestion. Third, the act of determining whether a suggestion might be useful is in itself a distraction and creates cognitive load, and this distraction must be minimized if the cost of false positives is to be minimized. These three assumptions lead to the conclusion that suggestions should be displayed with a *ramping interface*: an interface that progressively displays more information about a subject when the user wishes it, while still allowing the user to bail out at any time without further distraction. In a ramping interface information is conveyed in stages. Early stages give information that is most easily processed and that gives a good indication of the contents of information to follow. Later stages contain progressively more information, with the corresponding higher cognitive and perceptual cost to process that information. The idea is that users can quickly learn whether a suggestion will be useful and if not they need not look further. The interaction becomes dialogue between user and JITIR, where the JITIR will proactively offer some information and the user can then ask for more if desired.

1.2.4 Experiments

JITIRs are contextually situated: their use and effectiveness depend greatly on the domain in which they are applied. This sensitivity to the task environment makes JITIRs especially hard to evaluate in the general case. For this reason, several experiments have been performed to cover a range of activity and variations. The first study tests how subjects use JITIRs in a controlled-task experiment involving the writing of an open-ended essay. In particular, it examines how subjects access and use supporting information when using a JITIR versus using a traditional search engine. The results show that users who are given a JITIR as well as a search engine tend to view almost three times as many documents as users who only have access to a search engine. Also, subjects found the Remembrance Agent (the main JITIR used in the experiment) more useful than the search engine for their task. The second studies the differences between the relevance and utility of a suggestion, and examines the effects of the corpus used on the quality of suggestions. The results indicate that the assumptions made in evaluating traditional information retrieval, namely that relevance implies utility and that the choice of database is outside the scope of the IR algorithm, are not valid when applied to JITIRs. The third study examines open-ended long-term usage, where people used JITIRs over the course of many months. The study uncovered many ways in which JITIRs can be valuable to a user, including providing information that changes the user's current task, information that supports the user's current task, information that contextualizes the user's environment, and providing

information that is not useful to the *current* task but is valuable for other reasons, e.g. entertainment. Finally, subjects and users from all three studies were informally interviewed about their experiences.

1.3 Road Map

The roadmap for the remainder of this thesis is as follows:

- This chapter defined Just-In-Time Information Retrieval agents and outlined the contributions of the work.
- Chapter 2 describes the three JITIRs implemented: the Remembrance Agent (word-processor based), Margin Notes (web based), and Jimminy (wearable-computer based). It also describes Savant, the information-retrieval back end that is used by all three systems.
- Chapter 3 presents theoretical background and techniques that should be used in designing JITIRs.
- Chapter 4 describes implementation details about the Remembrance Agent, Margin Notes, Jimminy and Savant.
- Chapter 5 describes experiments and evaluations and discusses their implications.
- Chapter 6 discusses related systems and how they fit into the framework presented in this thesis.
- Chapter 7 draws conclusions and describes areas of future work for this research area.

CHAPTER 2 *Overview of Implemented Systems*

*This isn't just a blue sky outfit you know. People **build** things around here.*
– Andy Lippman

This chapter will describe the three JITIRs implemented in the course of this research: the Remembrance Agent (RA), Margin Notes, and Jimminy (also known as the Wearable Remembrance Agent). It ends with a description of Savant, the back-end information-retrieval engine used by all three systems. The high level designs are described here, but implementation details and design trade-offs are put off until Chapter 4.

2.1 The Remembrance Agent

Emacs is a popular text-editor for the Unix operating system. While it is often used for traditional text-editing tasks such as writing papers or computer code, the editor is powerful enough that it is also used for reading and writing email, net news, and even browsing the web. Emacs supports the use of multiple buffers, with each buffer containing a file, email, net news viewing session, or other information source.

The RA **(Rhodes 1996)** continually presents a list of documents that are related to the current document being written or read. These suggestions appear in order of relevance within a special display buffer at the bottom of the Emacs window. When the user is typing, reading emails, or otherwise changing his environment, the list is updated every few seconds. Suggestions from multiple databases can be listed in the display buffer, each with a certain number of lines. For example, the system can be configured to display suggestions from email archives in the first four lines and suggestions from note files in the next two lines. The display can also show different "scopes" from the same database, e.g. the first few lines can show suggestions related to the past 20 words while the others can show suggestions related to the past 500 words. Suggestions are shown without regard to a user's history, i.e. the RA does not have knowledge of the user's past interests or whether the user has previously seen a particular suggestion.

(Rhodes 1996)
Rhodes, B. and Starner, T. The Remembrance Agent: A continuously running information retrieval system, in *PAAM'96*, 1996, pp. 486-495

FIGURE 1. RA main display

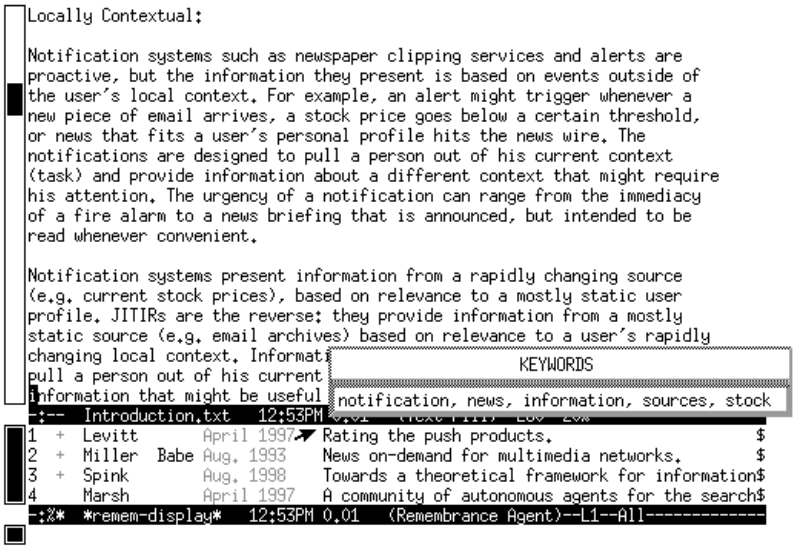


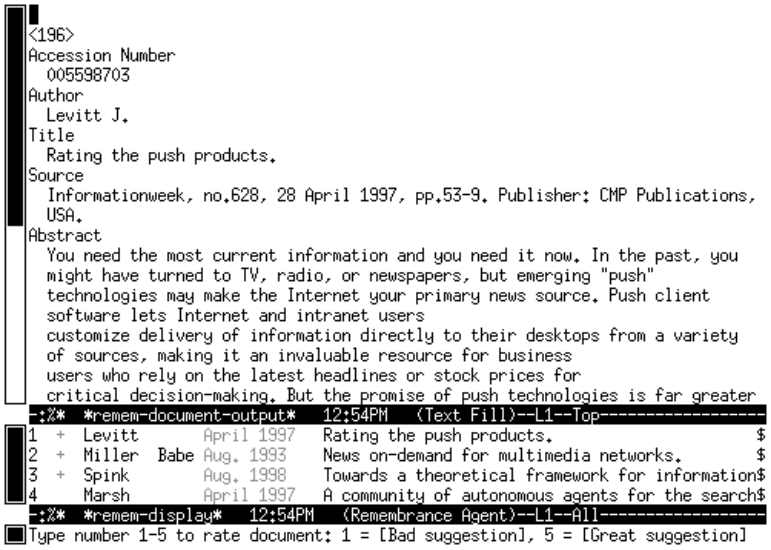
Figure 1 shows a screen-shot of the RA when writing the introduction to this thesis. In this case, the documents being suggested came from a subset of the INSPEC database of conference and journal abstracts and citations (about 150,000 citations). The suggestions are all for papers that might be relevant to the section being written. Suggestions are proposed based on *word co-occurrence*: the occurrence of the same word in both the text being written and the document suggested. For example, both the text being written and the abstract for the "Rating the push products" paper contain the words *notification*, *news*, *information*, *sources*, and *stock* (shown in the pop-up window). The actual display is in color, with alternating columns in different colors to allow the information to be scanned quickly. The full Emacs window is larger in normal operation, which means a larger ratio of editor-lines to RA-suggestion lines.

The summary lines are a combination of fields from the document, and are designed to give the user an indication of the content of the document as quickly as possible. For example, from left to right the first suggestion in Figure 1 contains the line number, a relevance score consisting of zero, one or two plus signs, the author of the citation, the date of publication, and the title of the paper cited in the suggestion. The format of the summary lines can be customized for individual databases. For example, articles from the *Boston Globe* use a longer field to show headlines and show publication date, but don't show the author of the article. Email suggestions, on the other hand, display all the fields used for the INSPEC database, plus the name of the folder in which the email is stored. By default if a suggestion is below a minimum threshold it is not displayed and "No suggestion" is shown instead. It is also possible to configure the system to display below-threshold suggestions, with a minus-sign as the relevance.

Right-clicking on a suggestion causes a small pop-up window to display the top five keywords that led to the abstract being suggested, as seen in Figure 1. These keywords are also displayed to the far right of the suggestion line, although they are only visible when the user has a wide display window. To see the full text being suggested the user types a keyboard shortcut (*control-c r* and the line number to show) or clicks on the desired line number. The full text then replaces the current display buffer, as

shown in Figure 2. By default the RA will not recursively make suggestions based on a suggestion that has been retrieved, though this feature is customizable.

FIGURE 2. RA result screen



After retrieving a document the user is prompted to enter a rating for the suggestion (1-5), although entering a rating is not required. The rating is written to a log file, and is used for evaluation purposes. It is not currently used for any form of machine learning or user-profile creation, although these might be implemented in the future.

Different suggestion databases can be associated with specific buffers or buffer types. For example, the system can be configured to automatically draw suggestions from an email archive database when reading or writing email, and from the INSPEC database whenever writing a paper in the LaTeX formatting mode. These defaults are set by hand in a configuration file. Databases can also be switched manually by typing *control-c r d*. Database changes are sticky: if the database is switched once for a particular buffer then revisiting that buffer will automatically switch to that database thereafter.

The user can also manually perform searches. Clicking on a field in a suggestion causes the RA to perform a search based only on the contents of that field. The user may also type *control-c r f* and enter a query for a particular field, or type *control-c r q* to fill in a query form that includes all fields.

2.2 Interaction With The RA

FIGURE 3. Interaction transitions with the RA

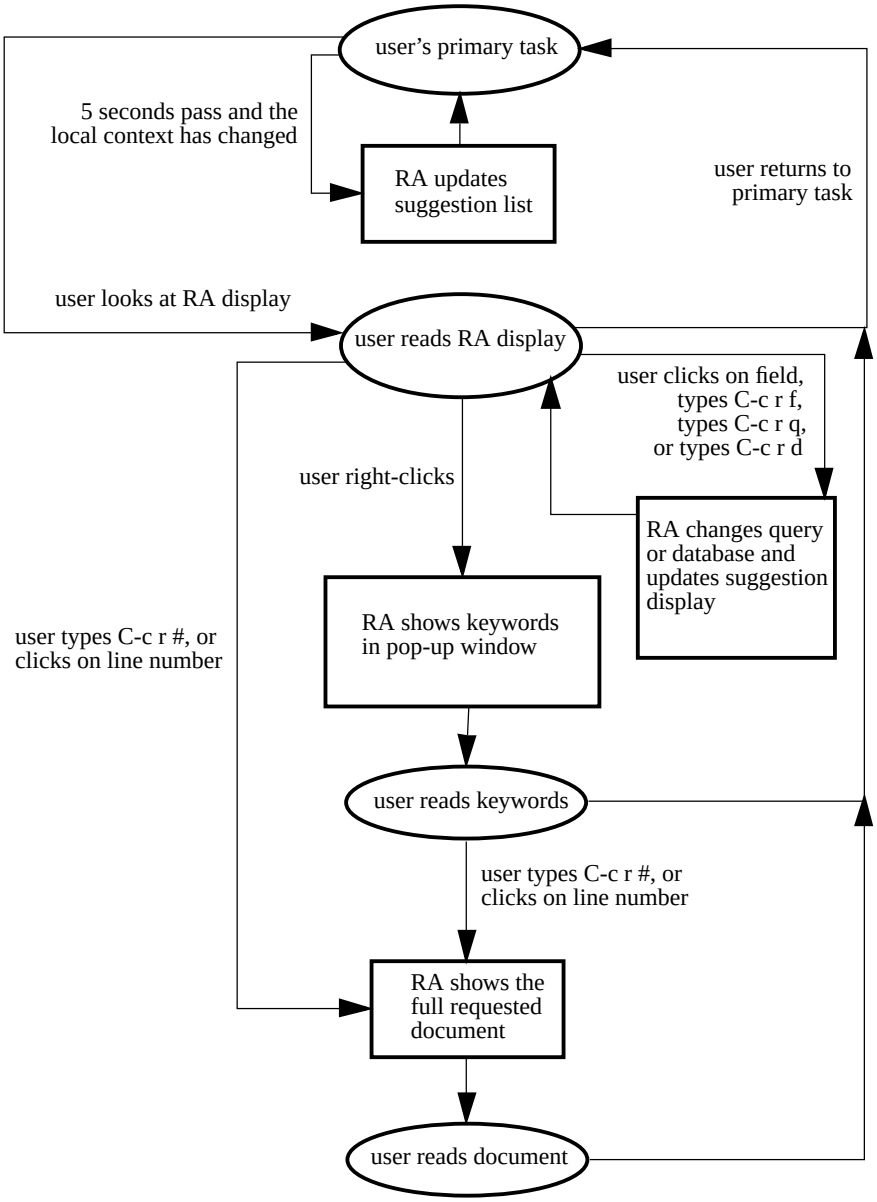


Figure 3 shows the transitions in an interaction with the RA. Rectangles represent actions taken by the RA that change the state of the display. Ovals represent actions taken by the user that change the information she knows. Links are actions taken on the part of the user or the RA, or are events that occur in the world such as five seconds passing since the last update. The overall interaction is a dialogue between user and RA, where sometimes the user initiates and sometimes the RA initiates the dialogue.

Most of the time the user is working on her primary task: writing or reading documents in the main Emacs buffer. Every five seconds (assuming the buffer has changed) the RA updates the suggestion list based on the text in the current buffer and the database(s) being used.

From working on the primary task, the user can decide to look at the RA's display and read one or more suggestion lines. By this action she updates both her knowledge about what information is available and her knowledge in general. She can also take one of several actions that cause the RA to change the suggestions displayed, namely clicking on a field in a suggestion line (which performs a search based on that field's contents), performing a full-text query by typing *C-c r q*, performing a search on a particular field by typing *C-c r f*, the name of the field to search and the text to search for, or changing one or more scopes to a new database by typing *C-c r d* and the new database name.

After reading the display, the user may want to see keywords that are associated with a particular suggestion. By right-clicking on the suggestion line, she causes the RA to display a pop-up window containing the keywords. By reading the keywords she updates her knowledge about the document associated with the suggestion, which in turn may convince her to read the full document or ignore the suggestion and return to her primary task.

If the user wants to read the full document described by a suggestion line, she can do so by typing *C-c r* and the line number of the suggestion or by left-clicking on the line number. This action causes the RA to replace the primary buffer with the requested document. Reading the document, of course, changes the user's knowledge further.

2.3 Margin Notes

(Rhodes 2000)
Rhodes, B. Margin Notes: building a contextually aware associative memory, in *IUI'00*, 2000, pp. 219-224

Margin Notes **(Rhodes 2000)** is a JITIR that automatically rewrites web pages as they are loaded, adding hyperlinks to related documents. As a web page is loaded, Margin Notes adds a black margin strip to the right of the document (the margin strip is always black, regardless of the color of the main page). Like the RA, Margin Notes then compares each section of the document to pre-indexed email archives, notes files, and other text files, based on keyword co-occurrence. If one of these indexed files is found to be relevant to the current section of the web page, a small "annotation" is included in the margin next to the relevant section. The note contains a quick description of the suggested text, a series of circles representing the relevance of the suggestion, and a link to get more information. The annotation consists of a subject, date and author for the suggested text, though the exact make-up of the note is customizable based on the database. For example, annotations for the email database also include the name of the folder in which it was filed, while annotations for the *Boston Globe* database only include headline and date to maximize the amount of headline that can fit.

FIGURE 4. Margin Notes screen-shot

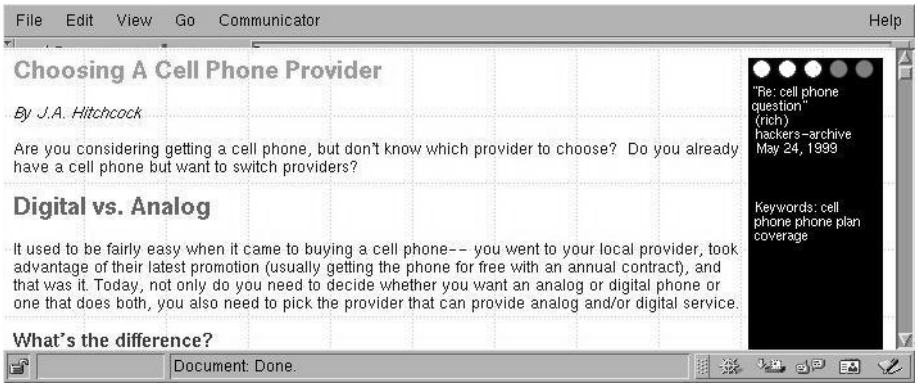


Figure 4 shows a screen-shot of a web page on cell-phone service plans, annotated by Margin Notes. This example is a page and annotation that came up during normal use of Margin Notes, though the relevance of the annotation is better than the typical annotation. The database used for this example is the collection of Media Lab email archives since 1988, a total of over 180,000 messages. The suggested document, listed in the black margin to the right, is email sent to the “hackers” mailing list and gives personal experiences with cellular service in the MIT area. The actual display is in color: the circles at the top of a suggestion are filled in with red to indicate the relevance of the suggestion.

Placing the mouse over an annotation produces a list of the five keywords that were most important in deciding a suggestion's relevance (these are also shown in the figure). Keywords are useful for contextualizing a suggestion and giving a better indication about a suggested document’s contents. Clicking on an annotation creates a new browser window that displays the full text of the email, note-file, or text being suggested. The suggested page also has feedback buttons used for evaluating the system.

Because web pages will often cover many different subjects, Margin Notes segments the web pages it annotates based on HTML tags. A section is defined as text between HTML header tags, horizontal rules, and a few non-standard section delimiters used by HTML editors such as Microsoft Word. Each section receives its own annotation, assuming the section is long enough and the annotation is over a threshold relevance score. The exception is the first annotation on each web page, which is based on the entire page instead of a single section. The inclusion of a page-wide annotation allows both a general annotation as well as specific, focused views. Margin Notes uses the location of the annotation to indicate scope: annotations appear at the top of the section to which they are relevant. This placement is analogous to the use of marginal notes in traditional print. The black margin strip is used to “brand” the annotation as belonging to Margin Notes; all text to the left of the margin is the page being annotated, all text within the margin is placed there by Margin Notes.

2.4 Interaction With Margin Notes

FIGURE 5. Interaction transitions with Margin Notes

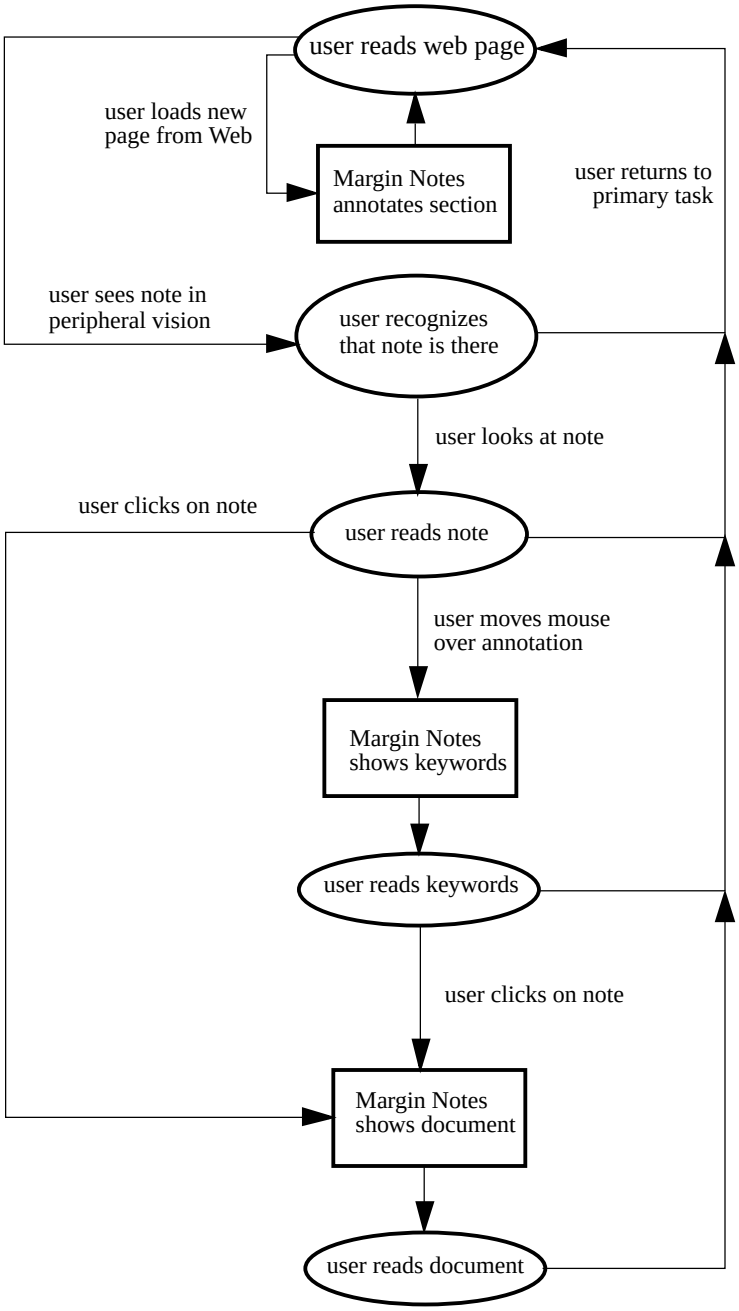


Figure 5 shows the transitions in an interaction with Margin Notes. Rectangles represent actions taken by Margin Notes that change the state of the display. Ovals represent actions taken by the user that change the information she knows. Links are actions taken on the part of the user or Margin Notes.

The interaction starts with the user reading a web page. As a new page is loaded it is annotated by Margin Notes. The black margin is added immediately; section annotations appear over the next few seconds as they are created.

As the user is reading the web page she will occasionally see an annotation in her peripheral vision, letting her know that the note exists. Seeing the note is usually an unconscious act; it is difficult to avoid noticing an annotation when one exists. At this point the user can continue reading the main web page and ignore the annotation, or she can look at the note and read some or all of the summary information. Both noticing the note and reading the summary changes what the user knows about the availability of information, and may provide other useful information in its own right.

After reading the summary the user can go back to reading the main web page, can view keywords associated with a summary by moving the mouse over the annotation, or can click on the suggestion to view the whole document.

2.5 Jimminy

(Rhodes 1997)
Rhodes, B., The Wearable Remembrance Agent: a system for augmented memory, in *Personal Technologies: Special Issue on Wearable Computing*, 1:218-224, 1997

Both the RA and Margin Notes provide potentially useful information based on a person's computational environment: his email, documents, web pages being viewed, etc. Jimminy, also called the Wearable RA, provides information based on a person's physical environment: his location, people in the room, time of day, and subject of the current conversation **(Rhodes 1997)**. Processing is performed on a shoulder-worn "wearable computer," and suggestions are presented on a head-mounted display.

The ultimate goal is that all information about the wearer's physical environment will be available to Jimminy through automatic sensors. However, the focus of this research is not sensor technology but rather what can be done with that technology once it is available. To this end, Jimminy is a general architecture that uses plug-ins for any sensor that can be attached to a wearable computer. Information that is not provided by sensors can be entered into the system by hand. The currently implemented system has been demonstrated with passive sensors that detect a person's physical location and people in the room, and uses the system clock to determine the time of day. The latest revision of Jimminy no longer supports comparison based on time, but previous versions did. The subject of a conversation is entered by hand, as are full-text free-form notes. The wearable computer hardware and sensors are detailed in Chapter 4.4.

The wearable computer serves two main functions. First, it is used as a general note-taking system. In conversations and lectures, notes are usually touch-typed using the one-handed keyboard while maintaining eye contact with the person speaking. The head-mounted display is occasionally viewed to see what has just been typed. Any note written using the wearable can be tagged with people present, subject, location and timestamp using a single key combination. Over the course of four years the wearable has been used by the author to take and annotate over 850 notes. They range from notes on classes and conversations at conferences to notes on dance steps. The second major use of the wearable computer is to retrieve notes and information anytime, anywhere. Reading and understanding information on the head-mounted display is a more attention-demanding task than note-taking. It is also more obvious to observers that the wearable user is distracted: his eyes are clearly focused on the screen and it is difficult to speak and read at the same time. For these reasons information tends to be retrieved during pauses in conversation or in lecture situations.

The interface for Jimminy is based on the Emacs RA, but differs in a few important ways. First, screen real-estate is scarce for a wearable, so suggestions are presented in abbreviated form. Second, the features of the environment that are currently being sensed are listed on the mode line. For example, in Figure 6 the location (room E15-335) and the person in the room (David Mizell) are listed in reverse video on the bottom line. This display is important because sensor data may be incorrect, and the user

also needs a reminder about what environmental information was last entered by hand so it can be updated. Third, keys are defined on the one-handed keyboard to increment or decrement the bias on different field-types. For example, one could set biases so that the person field is twice as important as other fields when finding useful documents. The biases are listed on the mode line as well. Finally, the system will modify the bias for certain features based on recent-modification times. For example, if the wearer of the system enters a new room, the bias for room location is temporarily increased by three. After a minute in the same room the bias is returned to base-line levels.

FIGURE 6. Jimminy screen-shot

Figure 6 shows a screen-shot of Jimminy as it would appear on the head-mounted display. The top 80% of the screen is reserved for notes being entered or read, plus the standard Emacs mode line giving time and information about the file being edited. The next four lines show suggestions based on the wearer's current context. The display is the same as the RA except formatted for the 80-column monochrome display. The bottom mode line shows a condensed view of current biases for location, subject, person and current text being typed, followed by the context (keywords and physical environment) that led to the suggestion. The actual head-mounted display is bright red on black with 720 x 240 pixel resolution, with the number of characters and the aspect ratio the same as shown in the figure.

For the example scenario of Figure 6, imagine the wearer is talking with David Mizell, a fellow wearables researcher from Boeing, and has just entered Media Lab room E15-335 (which is where the automatic embroidery machine is kept). This is a made-up example, although the screen-shot is of the actual system using the real database of notes written on the wearable. The mode line shows the current location and people in the wearer's environment and shows that the bias for location and people present is four and the bias for subject and current text being typed are both one. The periods around the location and person biases indicate they have temporarily been raised because these features changed recently. In one minute they will both go back to the hand-set value of one. Biases can also be set using the one-handed keyboard. The first suggestion "embroidery machine class" is a note that was taken during the training class for the embroidery machine. As can be seen in the keywords section (far

right), the suggestion is based mainly on the room number, but also on some of the words in the notes being typed. The next two suggestions are about David Mizell, the first being notes from a talk on augmented reality that he gave at a conference in 1998, the second being his entry in the wearer's contact Rolodex file. The final suggestion is an email note about the wearable fashion show for which the conductive cloth technology was being developed at the Media Lab, retrieved based on keywords that have been typed into the notes area.

2.6 Interaction With Jimminy

FIGURE 7. Interaction transitions with Jimminy

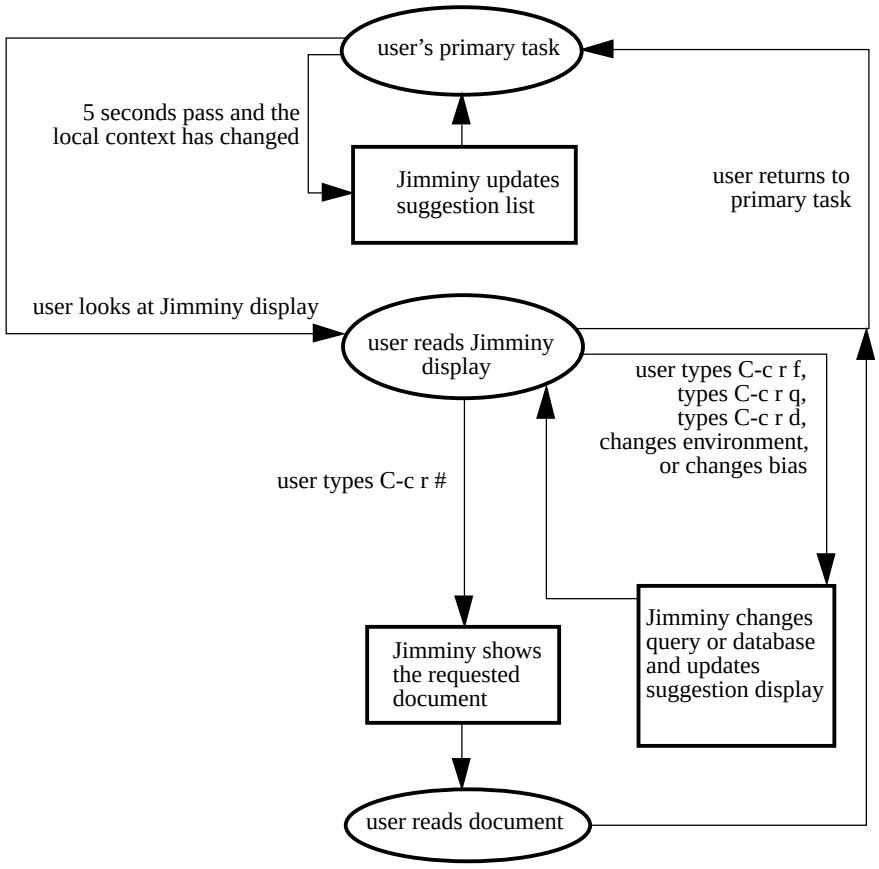


Figure 7 shows the transitions in an interaction with Jimminy. As with the previous interaction diagrams, rectangles represent actions taken by Jimminy that change the state of the display and ovals represent actions taken by the user that change the information she knows. Links are actions taken on the part of the user or Jimminy, or are events that occur in the world such as five seconds passing since the last update or the user walking into a new location.

Most of the time the user is working on her primary task. This primary task may include using Emacs on the wearable to take notes, but it may also include tasks that have nothing to do with the wearable computer, such as talking to a coworker. Every five seconds, Jimminy updates the suggestion list based on the text in the current

buffer, the user's current physical environment, the current bias settings, and the current database(s) being used. If a feature has changed (e.g. she has walked into a new room) then the bias for that feature is temporarily raised by three points and the mode line is updated to reveal that change. After one minute Jimminy automatically lowers the bias back to its original level.

From working on the primary task, the user can decide to look at the Jimminy output on the head-up display and read one or more suggestion lines. As with the RA, she can also perform a full-text query by typing *C-c r q*, perform a field search by typing *C-c r f*, or changing one or more scopes to a new database by typing *C-c r d* and the new database name. She can also change the value of one or more features of the environment as known by Jimminy (e.g. the subject of her conversation or her location), and can increment, decrement or reset the bias for any of the four main features. All of these actions change the internal state of Jimminy and affect the suggestions that are displayed.

If the user wants to read the full document described by a suggestion line, she can do so by typing *C-c r* and the line number of the suggestion. This action causes Jimminy to replace the primary buffer with the requested document.

As should be expected, interaction with Jimminy is similar to interaction with the RA (upon which Jimminy is based). However, with Jimminy the user is often in environments that make it difficult to read large amounts of text. For example, she might be in a conversation where reading more than a few words will cause an uncomfortable pause, or otherwise make her look distracted. For this reason, Jimminy users often will either use the suggestion line itself to jog their memory (but not bring up the full document), or they will wait for natural pauses in the conversation to bring up a full document.

2.7 Savant

All three implemented JITIRs are front ends to the same back-end system, called Savant. The front-end senses the user's local context (that is, the document or email being written, the web page being viewed, or the physical environment of the user of a wearable computer) and sends that information in text form to Savant as a "query." Savant then works as an information retrieval engine: given a query it produces a rank-ordered list of pre-indexed documents that best match the query. Savant consists of two programs: *ra-retrieve* performs information retrieval based on a query, while *ra-index* creates index files so retrieval can be performed quickly. Indexes can be created from any sort of text, including collections of newspaper or journal articles, organization-wide collections such as office memos, or personal sources such as email and notes. Previous versions also allowed pages to be indexed directly from the web. Documents are usually re-indexed nightly to incorporate new changes and additions.

For retrieval, the front-end first sends text from the user's current environment to *ra-retrieve*, which then compares the query to pre-indexed files and returns a list of the most similar documents, ranked in order of similarity. The output includes a one-line summary of the document including subject, author or person associated with the document, date, the name of the file containing the document, a relevance score and a list of the top five keywords or terms from the query that led to this document being suggested. This list is further processed by the front-end to display suggestions.

2.7.1 Template Matching

The power of Savant comes from a strong template-matching system that can recognize types of documents or queries, parse individual fields from each, and index or process them all automatically. For example, if *ra-index* is pointed at a top-level directory of files it will recognize each one as an email archive, HTML file, LaTeX file, collection of notes taken on the wearable computer, set of paper abstracts from the INSPEC database or raw text, and ignore other file formats. It will break multi-document files into individual documents and then break each document into individual fields. For example, archive files from the INSPEC database are broken into individual citations and then further broken into title (*subject*), author (*person*), *date* and *body.* This parsing means indexing can be performed automatically with no need to annotate or label files by hand. When retrieving documents, a user's local context (the query) is also broken into individual fields. For example, an email message being written will be broken into *subject*, *person* (who the email is to), *date* and *body.* Templates are hard-coded into Savant, but are designed to be easily modified or added with a recompilation of the source code.

2.7.2 Data Fusion

Different field-types can have different similarity metrics. For example, the body of a query and the body a document can be compared using a text-retrieval method such as the Term Frequency / inverse Document Frequency (TF/iDF) algorithms discussed in Chapter 4.1.6. The similarity between two dates was, in a previous version of Savant, compared based on the number of days that separate them (the current implementation does not compare dates). New similarity metric plug-ins can be added easily. For example, a collaborator at British Telecom has developed a plug-in that can find documents that are related to a current location based on GPS coordinates **(Crabtree 1998)**. Multiple similarity metrics can also be defined for the same set of data. For example, two different weighting schemes or entirely different algorithms can be defined for comparing the body of a message to text being typed.

(Crabtree 1998)
Crabtree, I.B. et al. Adaptive Personal Agents, in *Personal Technologies*, 2(3) 141-151, 1998

Savant combines the results of multiple similarity metrics for a document in a process known in the information retrieval field as *data fusion* **(Lee 1995).** If a document is similar to a query based on more than one field or similarity metric (e.g. if both the "from" field and body of a message partially match), a linear combination of the similarities are used based on weights defined in the templates. The full algorithm is detailed in Chapter 4.1.5.

(Lee 1995)
Lee, J. Combining Multiple Evidence from Different Properties of Weighting Schemes, in *SIGIR'95*, pp. 180-188

2.7.3 Filtering

Automatically generated queries tend to contain extraneous text that is not useful for retrieval, e.g. signature lines and email headers. Indexed documents will likewise have HTML markup and headers that will dilute the value of important data when selecting documents. To address this problem, each template can associate a filter bank with each field. A filter bank is an ordered list of Perl-style regular expressions that match text that should be removed from the field before parsing. For example, filters associated with the email *body* field recognize and remove email signature lines, headers from included files and common lines such as "Begin forwarded message." Filters associated with the email *person* field remove all information except for the user name, while filters associated with all fields in HTML documents remove hypertext tags and comments.

CHAPTER 3 *Theory and Related Work*

[Do not] put too much confidence in experimental results until they have been confirmed by theory.
– Sir Arthur Eddington

This chapter is divided into three sections, each presenting theory that relates to one of the thesis questions described in Chapter 1.2.1. This research is multidisciplinary, so it is appropriate that the theories described span several fields. The first section draws from cognitive science, psychology and decision science to show how JITIRs are used and affect the way people use information. The second section draws from the field of information retrieval to show how a JITIR can automatically decide what information to show given a person's environment. The last section draws from interface design and human factors to show how a JITIR can present potentially relevant information in a way that does not distract from a person's primary task. Each section will first describe relevant theory and then discuss how that theory can be applied to JITIRs in particular. Chapter 5 presents results of experimental evaluations of some of the theories presented here.

3.1 How JITIRs Affect The Way People Act

This section presents theory from cognitive science, psychology and interface design that is applicable to the following question:

> *How does the use of a JITIR affect the way a person seeks out and uses information?*

The section starts with an abstract discussion of the definition of intelligence and how it applies to the design of tools that augment human intelligence. It becomes more concrete with a discussion of various theories from psychology and decision science that offer a framework for understanding the benefits of using a JITIR. The section concludes with a discussion of how JITIRs fit into this framework.

3.1.1 Intelligence Augmentation

(Engelbart 1962)
Engelbart, D., *Augmenting Human Intellect: a conceptual framework*, AFOSR-3233, 1962

Douglas Engelbart has proposed the concept of *intelligence augmentation*: the extension of a person's mental ability beyond normal levels **(Engelbart 1962)**. In particular, Engelbart was interested in tools and techniques that could increase "the capability of a man to approach a complex problem situation, to gain comprehension to suit his particular needs, and to derive solutions to problems." In his general framework he describes three kinds of "repertoire" that people use to break down and solve complex problems. The first is what he calls *explicit-human* process capabilities, which are executed completely within the human mind. The second are *explicit-artifact* process capabilities, which are capabilities possessed entirely by tools or things and are executed without human intervention. The third are *composite* process capabilities, which are processes that emerge from an interaction between people and artifacts. JITIRs facilitate composite process capabilities. They perform some of the work of retrieving information, but the evaluation and eventual use of that information is still performed by the human.

(Newell 1989)
Newell, A. et al, Symbolic Architectures for Cognition, in *Foundations of Cognitive Science*, Posner, M. (ed), 1989, pp. 93-131

Classical cognitive science is concerned with explicit-human process capabilities. It examines how humans perform tasks, and uses those observations to construct theories about how the mind operates **(Newell 1989)**. From this perspective intelligence is an attribute of an individual. Anything outside of a person's body (e.g. tools, culture, or other people) is by definition not a part of intelligence but is rather a thing that interacts with intelligence. This places explicit-artifact and composite process capabilities outside the area of study of classical cognitive science.

(Hutchins 1995)
Hutchins, E. *Cognition in the Wild*, 1995

In his book *Cognition in the Wild*, Edwin Hutchins argues for a study of cognition as an attribute of an entire system, where the system includes a person, the culture in which he is acting, his physical environment, and both physical and cognitive tools he has at his disposal **(Hutchins 1995)**. Intelligent action stems from the workings of the whole system, not just the individual person.

This viewpoint has two important implications. First, Hutchins argues that ignoring processes that occur "beyond the skin" leads cognitive science to study how people act in the laboratory but to ignore cognitive behavior in the real world. Rather than perform experiments where any influence from the outside world is eliminated, he calls for a *cognitive ethnographic* methodology where behavior and mental action are studied in real-world situations **(Hutchins, p. 354)**:

> *The early researchers in cognitive science placed a bet that the modularity of human cognition would be such that culture, context, and history could be safely ignored at the outset, and then integrated in later. The bet did not pay off. These things are fundamental aspects of human cognition and cannot be comfortably integrated into a perspective that privileges abstract properties of isolated individual minds. Some of what has been done in cognitive science must now be undone so that these things can be brought into the cognitive picture.*

(Norman 1988)
Norman, D. *The Psychology of Everyday Things*, 1988, p. 54-80

Second, because the systemic view of intelligence focuses on the interactions between processes in the head and in the world, it is a good starting point for understanding the composite process capabilities that are available when using a JITIR. For example, Don Norman talks about knowledge in the head versus knowledge in the world, and how one can be substituted for the other **(Norman 1988)**. Say you have agreed to take a neighbor to the airport next Saturday at 3:30pm. Trying to remember is keeping knowledge in the head. Asking your neighbor to call the night before to remind you moves the knowledge to the world, or at least to the head of your neighbor. The idea is that knowledge can be stored in different locations with different advantages and dis-

advantages. For example, someone who only occasionally types might prefer to hunt-and-peck, relying on the letters written on the keys. Someone who types frequently will probably want to learn to touch-type because the added speed is worth the extra effort of moving the knowledge from the world into the head. Well-designed tools will put as much knowledge in the world as possible, both to help a person remember and to constrain his actions such that the right way to use the tool is also the obvious way to use the tool.

How intelligence is viewed will change the kinds of questions that will be asked about a design. If intelligence is viewed in isolation from the rest of the environment then tools will be viewed in terms of how they can aid an individual in his task. This viewpoint encourages questions such as "how quickly can a person using a JITIR find a piece of information" and "how relevant is the information shown?" When intelligence is viewed as a property of a system then the focus shifts to how different elements change the behavior of the system as a whole. In particular, questions include not only how a tool helps the completion of a task, but also how the tool might modify a person's behavior and how it might affect what tasks are performed at all. For example, one might ask whether people quote the work of others more when using a JITIR, or read more information not directly related to their task.

3.1.2 Cost vs. Expected Benefit

Imagine an extremely paranoid executive who wants to be sure he reads every piece of information related to his work. To make sure he is fully informed in his writing, he searches through his old emails, office memos, the New York Times and the Encyclopedia Britannica after every paragraph he writes. This practice would imitate the behavior of a JITIR. In fact, the resulting documents might be more relevant than those returned by a JITIR because he could perform his own searches more precisely than could any automated process.

Of course, not everyone is as meticulous as the executive described. Sometimes a person wants a particular piece of information, and in this case a search engine is the appropriate tool. Other times a person will not bother to perform a search due to lack of time, because the information he already has is "good enough," or because he expects a search will not turn up anything useful.

Such action (or inaction) is in keeping with *Zipf's Principle of Least Effort* **(Zipf, 1949)**:

> *In simple terms, the Principle of Least Effort means, for example, that a person in solving his immediate problems will view these against the background of his probable future problems, as estimated by himself. Moreover he will strive to solve his problems in such a way as to minimize the total work that he must expend in solving both his immediate problems and his probable future problems. That in turn means that the person will strive to minimize the probable average rate of his work-expenditure (over time). And in so doing he will be minimizing his effort, by our definition of effort. Least effort, therefore, is a variant of least work.*

(Zipf 1949)
Zipf, G.K. *Human Behavior and the Principle of Least Effort,* 1949, p. 1

The key point to Zipf's Principle is that a person will try to minimize his total future work, *given his best estimates at the time*. Zipf combines all potentially relevant motivators, including pleasure, into this single variable he calls "work." For example, he discusses the value of having a spouse in terms of efficiency of division of labor

(Zipf, p. 255), rather than as a trade-off between the cost of gaining and keeping a spouse and the rewards it can offer.

(Payne 1993)
Payne, J., et al. *The Adaptive Decision Maker*, 1993

Payne, Bettman and Johnson argue that people choose decision-making strategies based on multiple goals, including goals of accuracy and the conservation of limited cognitive responses **(Payne 1993)**. This framework is based on anticipated accuracy vs. anticipated effort: a decision maker assesses the benefits and costs of available strategies and chooses the strategy with the best cost/benefit trade-off. While their framework emphasizes the accuracy of a decision as the primary benefit and the effort as the primary cost, it acknowledges other motivators. For example, if a decision-maker thinks he will be asked to explain his decision to others he is more likely to pick an easily justifiable strategy. This strategy may not even be the most accurate, but it is still the optimal solution given the social environment **(Payne, p. 254)**.

3.1.3 The Two-Second Rule

Payne, Bettman and Johnson's effort-accuracy framework predicts that increasing the effort involved in performing a task will cause a proportional decrease in the number of times that task is performed. For example, increasing the effort involved in using a search engine should decrease the average number of queries people perform. However, it does not address how large a change in effort is necessary to see changes in usage patterns. Put in terms of time, how much longer must an information tool take to use before people stop using it?

(Miller 1968)
Miller, R., Response time in man-computer conversational transactions, in *AFIPS Conference Proceedings of the Fall Joint Computer Conference*, Vol 33, Part 1, 1968, pp. 267-277

Studies in computer response time indicate that the proper unit of analysis is on the order of seconds or even fractions of seconds rather than minutes or hours. Robert Miller argues that for many tasks more than two-seconds of response delay is unacceptable **(Miller 1968, p. 270)**. In particular, Miller argues that:

> *...it will be easily demonstrated that many inquiries will not be made, and many potentially promising alternatives will not be examined by the human if he does not have conversational speeds – as defined in this report – available to him. But tasks will still be completed as indeed they have been in the past, without conversational interaction, and at least some of them will be completed more poorly by any criterion.*

In other words, even delays of a few seconds will cause people to either not use information tools or to use them less frequently in performing their task. Of course, the exact threshold will depend on the expected benefit of the information being retrieved. Miller also argues that there is not a linear decrease in efficiency as response delay increases; there are response delay thresholds beyond which sudden drops in mental efficiency will occur.

Miller is primarily discussing system response *delays*, that is periods where the user has made an action and is waiting for the system to respond. He places responsibility for these effects on limits of human short-term memory **(Miller, p. 268)**:

> *When I shift from temporarily memorizing the telephone number to dialing it, short-term memory is holding this set of digits and the goal action of completing the dialing. An interruption or delay in achieving a goal usually results in a degree of frustration. The longer a content must be held in short-term memory, the greater the chances of forgetting or error.*

This thesis extends Miller's argument and contends that the two-second rule applies not only to waiting, but also to performing subtasks that distract from a primary task. For example, when performing a search for information about a sub-topic a person needs to use short-term memory to keep his place in the larger framework of the task. The amount of short-term memory required, and thus the amount of effort required in the task, will depend on a number of factors including the complexity of the sub-task, the amount of time required to complete the task and the similarity of the sub-task to the primary task. It will also depend on how much closure there is in the primary task at the moment of interruption; an interruption will cause less effort when performed between two cognitive "clumps" than when performed in the middle of an action.

There are several studies that show minor shifts in the amount of effort required to use information for a given task can have large behavioral effects. For example, Jarvenpaa has shown that the way graphical information is arranged on a display affects both the decision strategy used and the order in which a decision maker acquires information **(Jarvenpaa 1989)**. Another example involves the display of prices in grocery stores. Russo has shown that when a product display includes per-unit prices the sales of lower unit price products go up within a brand **(Russo 1977)**. However, there is little switching between brands when unit-price information is only listed under individual products. This behavior is presumably because different sizes of the same brand are placed next to each other, but between-brand comparison requires comparing prices that are distributed across 15 feet of shopping aisle. When unit prices are listed in order on a single display at the end of each aisle, customers start to switch between brands to find the cheapest overall per-unit price. Finally, David Woods **(Woods 1994)** describes several cases where poor information display has led to errors in airplane piloting and surgery, especially during periods of high tempo of operation and high cognitive load.

(Jarvenpaa 1989)
Jarvenpaa, S. The effect of task demands and graphical format on information processing strategies. *Management Science*, 35:285-303, 1989

(Russo 1977)
Russo, J. The value of unit price information. *Journal of Marketing Research*, 14:193-201,1977

(Woods 1994)
Woods, D. et al. *Behind Human Error: Cognitive Systems, Computers, and Hindsight.*CSERIAC SOAR report 94-01, 1994, pp. 113-119.

3.1.4 Application to JITIRs

There are many ways a person can discover information: she can try to remember, search the web or email archives, talk to a coworker, go to the library, etc. In terms of the theories given above, each of these activities has an associated expected benefit, computed by summing the benefit of all possible outcomes times the probability of that outcome. Each activity will also have expected costs including cognitive and physical effort, social cost, and possibly an economic cost.

When applied to the information search domain, these theories suggest that if the cost of finding and using information is more than the expected utility of the search then the search will not be performed. This decision not to act could occur for several reasons. First, the desired information may not be important enough. For example, the searcher might think she "remembers it well enough," or there might be little reward for accuracy. She might also think a search will not be fruitful, and thus the expected value is low even if an unexpectedly good result would be worthwhile. This expectation might be accurate, or the searcher may be unduly pessimistic. Finally, she could be under time pressure or dealing with a poor search interface, and thus the effort required for a search is too costly.

FIGURE 8. Example effort-accuracy trade-offs for information lookup

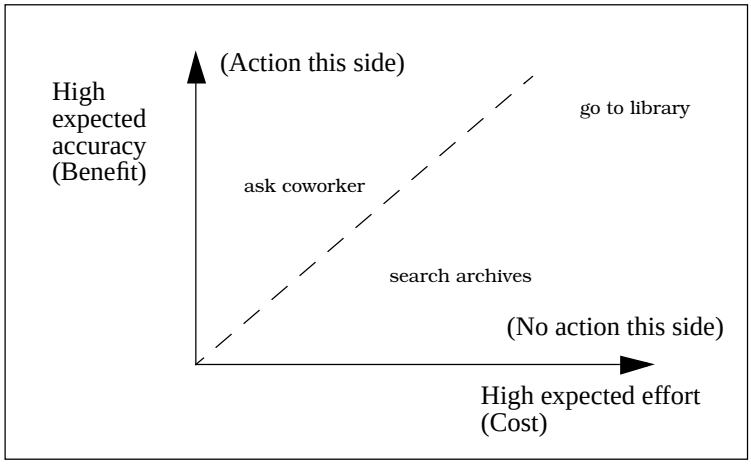


Figure 8, which is based upon a similar figure in **(Payne, p. 93)**, shows one possible cost-benefit trade-off for discovering a particular piece of information. In this example, asking a coworker for information is easy and has a moderate expected benefit, searching email archives has a higher cost and lower benefit, and going to the library has the highest expected cost and also the highest expected benefit. The dotted line is the "line of action." Activities to the left and above the line will be performed, while activities to the right or below the line will not. An individual with the given effort-accuracy trade-offs would ask a coworker for information, but would not bother to search her email archives or go to the library.

The line of action represents the trade-offs between cost and benefit for a particular task, and can shift with changing priorities. For example, an increased need for accuracy would reduce the slope of the line, and thus more kinds of information searches might be performed. Increasing time pressure would cause the reverse. The position of information resources can also shift over time. For example, if the coworker goes home for the night the additional effort of asking him a question would shift that resource to the right.

A few details need to be added to the over-simplified example given above. First, the "line of action" is not necessarily linear at all. As mentioned in the discussion of the Two-second Rule (Section 3.1.3), there can be thresholds beyond which any increased effort will have large effects on whether an action will be taken. Second, the availability of other methods for retrieving information will affect the action that will be taken. For example, a person might be willing to go to the library to get a piece of information, but if she might first ask her office mate and only go to the library if he doesn't know the answer. Third, the evaluation of expected costs and benefits associated with an information source are continuously updated even while in the process of using an information source. For example, a searcher for information may be browsing the web looking for information. Halfway through the web search she comes across a page that mentions a book that has the answer she is looking for. This new information has two effects. First, it changes the expected effort of finding information in the library, because now she has the exact reference for the book she needs. Second, it changes the expected benefit of going to the library because now she knows that the information will be there (assuming the book is not checked out, assuming the information is accurate, etc.). With this new information the searcher might stop her web search in mid-stream and immediately go to the library.

FIGURE 9. User interaction with retrieval of more information

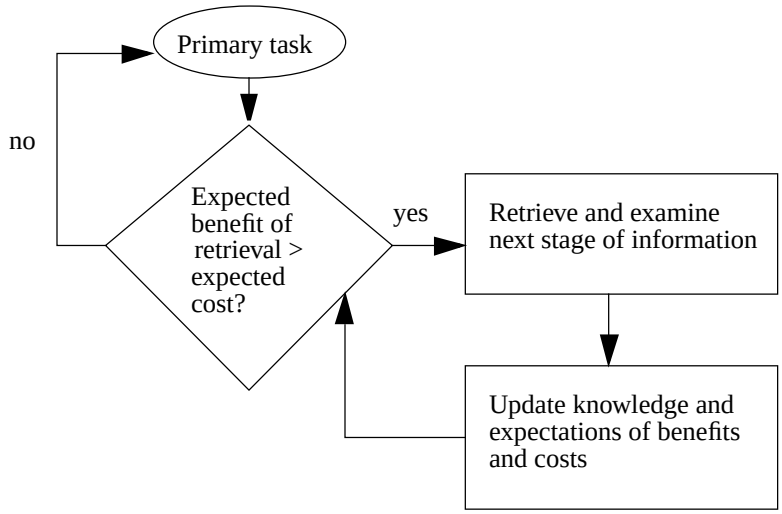


Figure 9 shows the interactions a user will have with a generic JITIR. The process is iterative: at every step the user will consciously or unconsciously reassess the expected benefits and costs of accessing new information, and act accordingly. For example, say a user of the Remembrance Agent is writing a project proposal (his primary task). He may briefly wonder if there is information relevant to the paragraph he just wrote. He does not expect information to exist, but it is worth a quick glance down at the RA's display because the cost of a glance is low. If he does not see any suggestions that are likely to be useful he will go back to his primary task, having spent less than a second in the interaction. On the other hand, if he notices information that might be useful then he will reassess his expected benefits and costs and might decide to look at what keywords were associated with the suggestion, or to bring up and read the full document.

In this way, JITIRs perform two major functions. First, they reduce the cost of searching for information by doing most of the work automatically. In terms of an effort-accuracy trade-off, a JITIR is a resource for retrieving information that is less costly than other information resources (i.e. to the left of other resources on the effort-accuracy graph). Second, by making it easy to get information *about what information is available*, JITIRs can change the expected benefit of searching for information in more depth, and thus encourage the retrieval of more information. This is especially true if a JITIR displays information in stages, as is described in the discussion of *ramping interfaces* in Section 3.3.5. Both these functions increase the likelihood that information is retrieved and used.

It should be noted that while a JITIR reduces the cost of accessing and assessing information, it does not completely eliminate these costs. There is still the small but important cost of looking at a suggestion and evaluating whether or not the result is useful. There can also be a cost for using the given information even after it is supplied. For example, if a JITIR supplies a citation for a technical paper, it may still be necessary to find the actual paper and read it before incorporating that knowledge. Finally, there is the cost in screen real-estate used to display suggestions, and the CPU cost involved in performing searches that may never be used. Many of these issues will be discussed further in Section 3.3: Interface Design.

Given the framework shown in Figure 9, the information returned by a JITIR can fall into five categories:

1. **False positive (useless)**: The information is not useful. This could be due to a failure of the underlying search technology, low-quality information, or a mismatch between the information provided and the immediate task at hand. As an example of the last case, detailed technical specifications for cellular phone frequencies would be useless to someone who simply wanted to decide what calling plan to purchase.
2. **False positive (already known)**: The information is useful, but is already known and has already been used or discarded. For example, based on a scientific paper being written a JITIR might suggest a journal article that has already been referenced or that the author has already deemed unnecessary.
3. **Decreased Cost**: The existence of the information is known, but it is not worth the effort to search for it through other means. For example, the JITIR may provide an exact quote when otherwise an author would paraphrase. By presenting the information directly, the cost of accessing the information is greatly decreased. In this case, the agent is an information resource to the left of other available search techniques shown in Figure 8.
4. **Increased Expected Benefit**: The information provided by the agent is not itself useful, but it indicates the existence of other information that might be valuable. For example, a JITIR might present a piece of email that is related to a current project proposal, but provides no information that can be directly applied to the task at hand. The email might still indicate that looking at other email in the same thread or talking to the sender could be fruitful. Here the JITIR does not decrease the cost of finding the author and talking to him, but it does increase the expected benefit of doing so. In this case it may be that reading the entire document suggested is below the line of action (too low an expected benefit to use directly), but seeing the summary of the suggestion gives new information about *another resource* that increases the expected benefit of using that other resource.
5. **Decreased Cost and Increased Expected Benefit**: The existence of the information provided was not known and is useful. In this case the JITIR decreases the cost of accessing the full document suggested and also increases the expected benefit of retrieving the document based on the summary. For example, a researcher may not think a reference exists about a particular esoteric field. The expected benefit of performing a search for any information is therefore quite low, and not worth more effort than a quick glance at the JITIR display. If the JITIR suggests a paper describing new work in the field, the expected benefit of digging further for the suggested information is now much higher. In fact, since the existence of the paper is now known the expected benefit is equal to the user's estimated value of the paper, based on the summary. At the same time, the cost of accessing the paper is much lower than it would be without the JITIR.

3.2 Finding Useful Information

This section presents theory from information retrieval that is applicable to the question:

How can a JITIR find information that would be useful to a person by looking at that person's current local context?

The perfect retrieval engine for a JITIR would be able to magically know with certainty whether a piece of information would be useful to a person. The next best thing would be an engine that could know a person's task, what information he already knows, the thoughts he is currently thinking and how he processes new information.

With such information a JITIR could easily deduce whether information would be useful. Unfortunately, JITIRs have neither the ability to prognosticate nor read minds. A JITIR must make due with whatever limited information it can sense automatically from a person's computational or physical environment, plus any heuristics that can be designed into the system.

3.2.1 Overview of Information Retrieval

People have the ability to understand abstract meanings that are conveyed by natural language. This is why reference librarians are useful; they can talk to a library patron about her information needs and then find the documents that are relevant. The challenge of information retrieval is to mimic this interaction, replacing the librarian with an automated system. This task is difficult because the machine understanding of natural language is, in the general case, still an open research problem.

More formally, the field of Information Retrieval (IR) is concerned with the retrieval of information content that is relevant to a user's information needs **(Frakes 1992)**. Information Retrieval is often regarded as synonymous with *document retrieval* and *text retrieval*, though many IR systems also retrieve pictures, audio or other types of non-textual information. The word "document" is used here to include not just text documents, but any clump of information.

(Frakes 1992)
Frakes, W. and R. Baeza-Yates (eds.), *Information Retrieval: Data Structures and Algorithms*, 1992

Document retrieval subsumes two related activities: *indexing* and *searching* **(Sparck Jones 1997)**. Indexing refers to the way documents, i.e. information to be retrieved, and queries, i.e. statements of a user's information needs, are represented for retrieval purposes. Searching refers to the process whereby queries are used to produce a set of documents that are *relevant* to the query. Relevance here means simply that the documents are about the same topic as the query, as would be determined by a human judge. Relevance is an inherently fuzzy concept, and documents can be more or less relevant to a given query. This fuzziness puts IR in opposition to *Data Retrieval*, which uses deductive and boolean logic to find documents that completely match a query **(van Rijsbergen 1979)**.

Sparck Jones 1997)
Sparck Jones, K. and P. Willett (eds.), *Readings in Information Retrieval*, 1993

(van Rijsbergen 1979)
van Rijsbergen, K., *Information Retrieval*, 1979

3.2.2 Evaluation in IR

Information retrieval algorithms are usually evaluated in terms of relevance to a given query, which is an arduous task considering that relevance judgements must be made by a human for each document retrieved. The Text REtrieval Conference (TREC) provides is a forum for pooling resources to evaluate text retrieval algorithms. Document corpora are chosen from naturally occurring collections such as the *Congressional Record* and the *Wall Street Journal*. Queries are created by searching corpora for topics of interest, and then selecting queries that have a decent number of documents relevant to that topic. Queries and corpora are distributed to participants, who use their algorithms to return ranked lists of documents related to the given queries. These documents are then evaluated for relevance by the same person who wrote the query **(Voorhees 1999)**.

(Voorhees 1999)
Voorhees, E. and D. Harman, Overview of TREC7, in N*IST Special Publication 500-242*, 1999

This evaluation method is based on two assumptions. First, it assumes that relevance to a query is the right criterion on which to judge a retrieval system. Other factors such as the quality of the document returned, whether the document was already known, the effort required to find a document, and whether the query actually represented the user's true information needs are not considered. This assumption is controversial in the field. One alternative that has been proposed is to determine the overall *utility* of documents retrieved during normal task **(Cooper 1973)**. Users would be asked how many dollars (or other units of utility) each contact with a document was worth. The answer could be positive, zero, or negative depending on the experience. Utility would therefore be defined as any subjective value a document gives the

(Cooper 1973)
Cooper, W., On selecting a measure of retrieval effectiveness. Part 1. *Journal of the American Society for Information Science*, 24:87-100, 1973

user, regardless of why the document is valuable. The second assumption inherent in the evaluation method used in TREC is that queries tested are representative of queries that will be performed during actual use. This is not necessarily a valid assumption, since queries that are not well represented by documents in the corpus are explicitly removed from consideration. These two assumptions can be summarized as follows: if a retrieval system returns no documents that meet a user's information needs, it is not considered the fault of the system so long the failure is due either to poor query construction or poor documents in the corpus.

3.2.3 Methods for IR

There are many different methods for both indexing and retrieval, and a full description is out of the scope of this thesis. However, a few broad categories will be described to give a feel for the range of methods that exist.

(Salton 1975)
Salton, G. et al, A vector space model for automatic indexing. *CACM*, 18:613-620, 1975

Vector-space model. The vector-space model represents queries and documents as vectors, where indexing terms are regarded as the coordinates of a multidimensional information space **(Salton 1975)**. Terms can be words from the document or query itself or picked from a controlled list of topics. Relevance is represented by the distance of a query vector to a document vector within this information space.

(van Rijsbergen 1979)
van Rijsbergen, K., *Information Retrieval*, 1979

Probabilistic model. The probabilistic model views IR as the attempt to rank documents in order of the probability that, given a query, the document will be useful **(van Rijsbergen 1979)**. These models rely on relevance feedback: a list of documents that have already been annotated by the user as relevant or non-relevant to the query. With this information and the simplifying assumption that terms in a document are independent, an assessment can be made about which terms make a document more or less likely to be useful.

(Rau 1988)
Rau, L., Conceptual information extraction and retrieval from natural language input. In *RAIO 88*, 1988, pp. 424-437

Natural language processing model. Most of the other approaches described are tricks to retrieve relevant documents without requiring the computer to understand the contents of a document in any deep way. Natural Language Processing (NLP) does not shirk this job, and attempts to parse naturally occurring language into representations of abstract meanings. The conceptual models of queries and documents can then be compared directly **(Rau 1988)**.

(Chakrabarti 1999)
Chakrabarti, S. et al. Mining the Web's link structure. *Computer*, 32(8), Aug. 1999, pp. 60-67.

Knowledge-based approaches. Sometimes knowledge about a particular domain can be used to aid retrieval. For example, an expert system might retrieve documents on diseases based on a list of symptoms. Such a system would rely on knowledge from the medical domain to make a diagnosis and retrieve the appropriate documents. Other domains may have additional structure that can be leveraged. For example, links between web pages have been used to identify authorities on a particular topic **(Chakrabarti 1999)**.

(Lee 1995)
Lee, J. Combining Multiple Evidence from Different Properties of Weighting Schemes, in *SIGIR'95*, 1995, pp. 180-188

Data Fusion. Data fusion is a meta-technique whereby several algorithms, indexing methods and search methods are used to produce different sets of relevant documents. The results are then combined in some form of voting to produce an overall best set of documents **(Lee 1995)**. The Savant system described in Chapter 2.7 is an example of a data fusion IR system.

3.2.4 What is Local Context?

The prime difference between IR and Just-In-Time Information Retrieval is that the former is based on a human-generated query and the latter is based on local context. There has been much discussion so far about local context and local environment, but these terms have not yet been formally defined. Context does not mean anything and everything in the environment. If it did then a search engine would qualify as a JITIR, because search engines perform queries based on the text typed into an entry field, which is a part of the environment. In fuzzy terms, context is as opposed to text; it is

the “everything else” of the environment. More precisely, context is the set of features in the environment that are not explicitly intended as input into the system being discussed. Therefore, a context-aware application is a system that can sense the environment beyond the direct input intended to control that system. For example, a smart house that turns the heat down when people leave is a context-aware system, because leaving the house is not (usually) performed just to turn down the heat. On the other hand, a door that opens when a key-card is swiped through a reader is not context-aware even though it senses the environment, because the only thing it senses are actions taken with the sole intent of controlling the door.

This definition of context is dependent on the intent of the user. For example, a smart room that automatically turns the heat down when everyone leaves the house would be using context to control the heat. However, say a person thinks the room was too warm and instead of turning down the thermostat she leaves the room with the sole intent of turning off the heat. In this situation, the act of leaving the room is a direct action, no different in theory than pushing a button or turning a dial to “colder.” Her action of leaving the room is not context, by a strict interpretation of the definition. In practical terms this confusion of intent can be avoided by talking about the normal and intended usage of an application.Thus a smart room is a context-aware application so long as the intended design is to automatically set the temperature without human intervention.

The other part that needs definition is “local.” In Chapter 1, the idea of local context was used to distinguish JITIRs from notification software such as alerts and newspaper clipping services. Clearly local at least implies spatial locality. Effects outside of the immediate vicinity of the user are not used by a JITIR. However, local context could be defined to include what is usually called a user profile. Such a definition would include not only the environment around a user but also his current preferences, list of gifts he needs to buy, books he’d like to read, etc. It could even include his history of actions and interests over time.

The definition of “local context” used here is not that broad. In addition to spacial locality, local context also implies features that shift relatively quickly over time. For example, the keywords that make up a person’s interest profile might change slowly over the course of years. The list of gifts he needs to buy might change over the course of months or weeks. The people he is talking to right now change over the course of minutes or hours. The rapidity of change for a feature places it on a spectrum between u*ser profile* and *local context*, with local context being those features of the environment that change most rapidly. In the middle of this spectrum are grey-area features that change over the course of hours or even days. For example, some systems that are defined as JITIRs in the Related Systems chapter (Chapter 6) base the information retrieved on a profile built up over several hours. Rather than define an arbitrary demarcation between local context and user profile, this thesis accepts the fuzziness of the definition.

3.2.5 Sensing vs. IR

The information that is available and usable by a JITIR depends on the task domain. In a computational environment a JITIR can sense the text a person writes or reads, the applications he has open, the people with whom he communicates, etc. Such information is especially useful for finding documents on related subjects or application-specific information, e.g. information about email corespondents. With physical sensors a JITIR can know people in the room, location, time of day, etc. It might also sense task-specific information such as patient vital signs, reports from mechanical diagnostic equipment, or radiation levels.

In some task domains, the sensing is the hardest part of deciding what information might be useful. For example, a JITIR might take information in from a camera, identify faces from the video stream and present a quick biography of that person. The hard part of this sequence is performing automatic face-recognition from a raw video stream. Once a face is recognized, finding a biography is a simple database lookup. The recognition of such high-level features is often called "perception" to distinguish the results from unprocessed physical sensor data.

In other task domains sensing is trivial, but determining the usefulness of information is hard. For example, it is very easy for a JITIR to "sense" all the words that have been typed into a word processor window. Once these words are known it is quite difficult to know what documents might be relevant. At some level of analysis the distinction between perception and understanding is arbitrary: there is no fundamental difference between associating a person's name with a raw video stream and associating a topic with a raw text stream. In practice, perception tends to find features from physical information such as video and audio streams while information retrieval and data mining tend to look at computational information such as text or interaction history. The techniques used are often similar, though the details differ.

The JITIRs implemented in this research can use many different kinds of information, including physical sensors. However, all three implementations rely most heavily on text retrieval techniques based on the text a person is writing or reading. This focus is mainly due to the generality of text; a wide variety of domains can use a text-based JITIR. Physical sensors are important and are explored with Jimminy, but they are not the primary focus of this research.

3.2.6 Priors

A person chooses information resources based on her needs at the time. If a doctor needs a medical reference she will use the National Institute of Health (NIH) search site. If she wants to know where to play golf she will browse the tourism board database. The choice of information source is one of the best indications of a person's current needs. In terms of probability, the fact that a person chose a particular database gives a strong prior that information in that database will be useful. This is information that is implied by a human-generated query in addition to the query text itself.

Usually JITIRs do not have such a direct indication of what information, if any, might be valuable to a user at a certain time. One way to make up for this lack of priors is to combine all information that might be useful into one database and use traditional information-retrieval techniques on the combined collection. Such a database would include all the medical information from the NIH and all the golf information from the tourism board, plus other information. Unfortunately, such a solution does not scale. Large databases put large memory and computational demands on a retrieval engine. More importantly, a database that covers a large range of topics has only a small percentage of documents that are directly related to a person's *current* information needs, even if the raw number of useful documents is quite large. With such a diluted database, it is likely that the results returned will be useless, even if useful information exists somewhere in the corpus.

Of course, some priors exist even with JITIRs. The kinds of information that can be provided by a JITIR will be implicitly constrained by its interface, the sensors it uses and the people who use it. For example, a JITIR embedded in a word processor can be expected to provide information that might be related to anything typed in that application, but not in other applications. The environment in which a JITIR is deployed provides some natural limits on the kinds of tasks being performed, though usually

not as many as are enjoyed by a specialized search engine. It is also sometimes possible to use features of the task environment to make a good first guess at the kind of information that might be most useful. For example, the RA uses the Emacs editing mode to tell whether a person is reading email, writing code or writing a paper. This information is used to choose between archived email, a code library, or paper abstracts from the INSPEC database.

3.2.7 Multiple Topics

In a manual information system a person usually searches for one piece of information at a time. In contrast, a person's environment will usually relate to several different subjects. For example, this thesis ranges from discussion of information retrieval to interface design to specific implementations of JITIRs. Email will often cover even more disparate topics within the same message.

Sometimes multiple-subject queries can be an advantage. Any document that matches more than one of the subjects (e.g. in this case relating to both information retrieval and interface design) is likely to be a useful document, and documents relating to just one subject represented in a query might still be useful. However, multiple-subject queries can also cause traditional IR techniques to miss documents that are very relevant to one particular subject in favor of documents only somewhat relevant to all subjects represented. One solution to this problem is to break queries into segments and annotate each segment separately. For example, Margin Notes annotates sections that are determined by analyzing the web page's HTML tags. The RA, on the other hand, allows the user to define multiple scopes that look at different sized neighborhoods of words around the cursor. Future versions might also segment plain-text documents into individual topics through linguistic analysis, as is described in **(Hearst 1994)**.

(Hearst 1994)
Hearst, M. Multi-Paragraph Segmentation of Expository Text, in *Proc. of the ACL*, June 1994

Furthermore, parts of the environment might not be useful to a retrieval engine at all. For example, a signature-line at the bottom of an email may not contain any information relevant to a person's current task or interests. This information must therefore be removed or otherwise ignored.

3.2.8 Evaluation of JITIRs

An important question is what metrics should be used to evaluate a JITIR. As discussed in Section 3.2.2, IR algorithms are typically evaluated based on whether the documents returned are relevant to the given query. It is assumed that the query is a good indication of the user's interests. Because JITIRs have no human-specified query, the concept of relevance as the primary evaluation metric is even more problematic than for traditional IR. If a JITIR returns no documents that meet a user's information needs due to poor query construction or poor documents in the corpus, it is the system's fault because the system chose that query and corpus automatically.

It is also possible for a document to be *too* relevant to a person's environment to be useful. For example, early versions of Margin Notes would occasionally suggest email that was highly relevant to the web page being read. These emails were often direct quotes from the web page, and offered no information apart from what was already on the page being annotated.

A better metric for evaluation is the *utility* a JITIR provides. The utility of a JITIR can range from directly helping with the completion of a task to providing entertainment or reassurance without directly aiding with the current task. Utility is a good metric of success for two reasons. First, unlike relevance, utility by definition has intrinsic value to a user. Relevance, on the other hand, is the goal of an IR system only because it is

assumed that in normal situations relevance will imply utility. As will be shown in Chapter 5.3.2 and Chapter 5.4.3, in the context of a JITIR relevance does not necessarily imply utility and vice versa. Second, the use of utility as an evaluation metric assumes that query and corpus choice are the responsibility of the system rather than the responsibility of the user. Third, it takes into account the fact that many kinds of information might be desirable to a user for a variety of reasons.

(Budzik 2000)
Budzik, J. et al., Beyond Similarity. To appear in *Working Notes of the AAAI 2000 Workshop on AI for Web Search*, July 2000

As discussed in Section 3.2.2, relevance is often correlated with usefulness but is not the same thing. In the context of JITIRs, this assertion is experimentally demonstrated both in **(Budzik 2000)** and in Chapter 5.3.2. The difficulty with using utility as an evaluation metric is that utility is dependent on many factors, including the current task being performed, the knowledge the user already has, and the user's current level of distraction. The various factors that can contribute to the value of a particular suggestion are discussed more in Chapter 5.4.3.

3.2.9 Retrieval With a JITIR

In spite of the differences described above, many of the techniques and models used in IR can be readily adapted for use in JITIRs by substituting a representation of the user's environment in place of a human generated query. For example, a vector-space approach to JITIRs **(Salton 1975)** would produce documents that are similar to the information contained in the user's environment as represented in a multidimensional information space. The other methods can likewise be adapted. However, a few differences remain that must be addressed, especially when dealing with retrieval based on text documents and automatically-generated queries.

(Salton 1975)
Salton, G. et al, A vector space model for automatic indexing. *CACM*, 18:613-620., 1975

Queries are longer. In the past five years the IR field has been attempting to produce relevant documents based on shorter queries. This trend has been spurred by the needs of web search engines, where the average query length is less than three words **(Jansen 1998)**. Many of the techniques developed have been ways to perform query expansion, where a short query is automatically augmented with words appearing in the best ranked documents of an initial probe search **(Harman 1995)**. With JITIRs, the environment provides a large amount of data that can potentially be a part of a query, so query expansion is less important.

(Jansen 1998)
Jansen, B., et al. Searchers, the Subjects they Search, and Sufficiency: A Study of a Large Sample of EXCITE Searches. In *Proceedings of WebNet-98*, 1998

(Harman 1995)
Harman, D. Overview of the Third Text REtrieval Conference (TREC-3), in *NIST Special Publication 500-226*, 1995

Both indexed documents and queries are multivariate. Both documents being suggested and queries themselves will often contain people's names, dates, subjects, abstracts, locations, phone numbers, and a host of other information types. While this can be true of manual queries as well, manual queries are often sparser due to the difficulties of entering large amounts of data.

JITIRs need both ranked best and filtering. Search engines normally produce a rank-best list of hits for a given query. The absolute relevance score of a hit is not important as long as the relative ranking is correct. JITIRs require a combination of ranked-best evaluation and filtering. They need to display the top few hits, but also need to display an absolute confidence in the relevance of a suggestion and to potentially suppress the display of low-quality suggestions. This need for filtering is similar to the problem faced by text-filtering systems such as automatic newspaper-clipping services. However, these systems assume that a stream of documents are compared to a long-lasting set of query profiles, such as a list of keywords expressing user interests **(Hull 1998)**. The queries in JITIRs are not long-lasting, so most of the machine-learning techniques used by these text-filtering systems cannot be applied.

(Hull 1998)
Hull, D., *The TREC-7 Filtering Track: Description and Analysis*, in *NIST Special Publication 500-242*, 1998

High precision is required. Information retrieval researchers often talk about the trade-off between precision (making sure all suggested documents are of high relevance) and recall (making sure all relevant documents are suggested). Because JITIRs largely play a supporting rather than a primary task role, they usually should not suggest more than a few documents. More suggestions, even if all of them were relevant, would be too much of a distraction. For this reason, the information retrieval algorithms for JITIRs should tend to favor precision over recall.

3.3 Interface Design

The question addressed in this section is one of interface design and human factors:

> *How should a remembrance agent present potentially useful information?*

The most important design constraint for JITIRs is that reading suggested information should be a secondary task. Unlike users of a search engine, JITIR users are not actively seeking information being suggested. They are less tolerant of distraction from their primary task, and are less willing to dig through suggestions to get useful information. Furthermore, even if a suggested text is highly relevant to a user's current context, he may not be interested in it. The text could already be known, the user may not wish to be distracted, or he may simply not want any new information. For this reason an interface must have a low cost for false positives. It must be a *non-intrusive* interface. However, it cannot be so ignorable as to never be noticed. It must also be *accessable*; it must be easy to switch between the primary task and the information being provided. The first criterion relates to what is called focus of attention, while the second criterion relates to divided attention.

3.3.1 Focus of Attention

Attention is limited. When a person is engaged in a task, it is difficult to pay attention both to that task and to a JITIR simultaneously. Cognitive science has many theories as to why and how mental attention is limited, but these theories do not agree. A full discussion of focus of attention is beyond the scope of this thesis, but a brief synopsis follows.

The dominant theory of attention is that of limited capacity and filtering. The idea is that the brain is a limited computational resource, and therefore focus of attention is required to allocate this limited resource **(Broadbent 1958)**. Information from the eyes, ears and other senses is processed and filtered such that later processing is only performed on the information that was attended.

(Broadbent 1958)
Broadbent, D. E., *Perception and Communication*, 1958

One of the main questions in the filter theory is what information gets filtered at different stages of processing. Broadbent proposed an *early-filter* view in which non-attended information is not processed beyond its basic physical characteristics (e.g. pitch, location, and intensity). This theory explains what is called the *cocktail-party effect*, where a person can concentrate on a single conversation at a party and effectively block out other conversations **(Cherry 1953)**. However, subsequent research has shown that some semantic information can be detected in an otherwise ignored audio channel, including a subject's own name **(Moray 1959)**. This research has led others to conclude that semantic information is at least partially processed, and filtering occurs after such processing. One problem with these late-filtering theories is that they do not explain why a focus of attention is required, given that processing occurs anyway.

(Cherry 1953)
Cherry, E. C., Some experiments on the recognition of speech, with one and two ears. *Journal of the Acoustical Society of America*, 25:975-979, 1953

(Moray 1959)
Moray, N., Attention in dichotic listening: Affective cues and the influence of instructions. *Quarterly Journal of Experimental Psychology*, 11:56-60, 1959

(Duncan 1979)
Duncan, J., Divided attention: the whole is more than the sum of its parts. *Journal of Experimental Psychology: Human Perception and Performance*. Vol. 5, No. 2, 1979, pp. 216-228

(Shaffer 1975)
Shaffer, L. H., Multiple attention in continuous verbal tasks. In P.M.A. Rabbit and S. Dornic, eds. *Attention and Performance V.*, 1975

(Wickens 1992)
Wickens, C. D., *Engineering Psychology and Human Performance*, 1992, pp. 375-382

(Allport 1989)
Allport, A., Visual Attention, in *Foundations of Cognitive Science*, Michael Posner (ed), 1989

Other researchers contend that the question should not be whether filtering occurs early or late, but rather whether the single-resource limited capacity theory is correct at all. For example, Duncan showed that when two hands perform different actions there is some tendency for each hand to perform the action assigned to the other **(Duncan 1979)**. The filter theory does not explain this sort of crosstalk between actions or stimuli. Even more damaging to the single-resource theory is that different task combinations have different amounts of crosstalk, even when they use separate sensory modalities and separate effectors for each pair of tasks. For example, Shaffer showed that copy-typing from a written transcript can be performed while simultaneously vocally shadowing (repeating) continuous speech played into a headphone. However, it is almost impossible to simultaneously read aloud and type from audio dictation, even for a skilled audio-typist **(Shaffer 1975)**.

To account for such crosstalk, Wickens proposes a multiple-resource theory in which processing relies on three independent dichotomous resources: *stages*, *modalities* and *codes* **(Wickens 1992)**. The two stages are encoding (which includes central processing) and responding. Encoding relates to the interpretation and processing of stimuli in the environment, e.g. the interpretation of speech. Responding refers to selection and execution of a task, e.g. the production of speech. Modalities are broken into visual and auditory representations, and codes are broken into spatial and verbal representations. By this theory, if two tasks demand separate resources on any of the three dimensions then time-sharing between the tasks will cause less confusion. For example, driving a car has both encoding and responding stages, and is visual and spatial. Listening to talk radio, on the other hand, is encoding stage, auditory and verbal. The two tasks differ in at least two dimensions, which is why they can be performed together easily. Tuning to a particular radio station, on the other hand, is a largely a response, visual and spatial task, and is therefore harder to perform while driving without distraction.

Alan Allport, once a supporter of the multiple-resource theory, now rejects it and instead explains the need for such crosstalk in terms of a need of a single selection-for-action **(Allport 1989)**. In his framework, there is no cognitive resource that can be used up, thus limiting attention. Instead, limited attention is an evolved trait that is designed to ensure a coherence of behavior when a person is faced with a complex environment with multiple goals and effectors. Without a focus of attention, he argues, a person might start performing one action and shift to another before the end-goal is achieved, undoing what has gone before. A limited attention regulates between the two requirements of insuring attentional engagement with a task over time and still allowing for interruptability when necessary.

Allport summarizes the effects of crosstalk between tasks as follows **(Allport, p. 639)**:

> *The pattern of results in... selective-response tasks can be summarized as follows: (1) When the designated target stimulus provides relatively the most compatible information source available for encode into the representational domain (semantic category, color, name, relative location, and the like) needed for execution of the task, then minimal or zero interference from other, less compatible information sources is observed. (2) In contrast when the to-be-ignored distractor information provides an equally, or even more compatible, specification (information source) for encoding into the required domain of representation than does the (task-designated) target information, interference – that is, delay in response, overt crosstalk error,*

> *or both – is liable to occur. (3) The extent of interference will then depend further on the availability of other (for example, spatial) cues, enabling effective segregation of target and nontarget information.*

The classic example of crosstalk between stimuli is the Stroop test **(Stroop 1935)**, in which subjects are asked to name the color of words written in different colored ink. When the word is a color word that does not match the ink color, e.g. the word "blue" in yellow ink, subjects are slower to react or read the word instead of name the color. By Allport's explanation, the text of the word is a better match for the task of speaking a name than is the color of the ink, and therefore crosstalk occurs. The example of typing from dictation while reading aloud is another example of the distractor representation matching the task better than the primary stimulus. When a subject is asked to shadow words heard in headphones while copy-typing, the words heard are a better fit for the task than the text to be typed. Thus little distraction occurs. When the situation is reversed (reading aloud while dictation typing), the distractor stimulus is a better match for the task than the proper stimulus, so confusion occurs. Making the input stimuli used by the two tasks more distinct can help alleviate confusion. For example, Barr and others have shown that subjects can avoid distraction when listening to a target and one or more distractor audio streams when the distractor streams are spoken in a different voice, in a different ear, or at a different pitch than the target stream **(Barr 1981)**.

(Stroop 1935)
Stroop, J., Studies of inference in serial verbal reactions. *Journal of Experimental Psychology* 18:643-662, 1935

(Barr 1981)
Barr, R. A., How do we focus our attention? *American Journal of Psychology*, 94(4):591-603, 1981

3.3.2 Divided Attention

The above theories describe limits of *focused attention*: the ability to attend to intended stimuli and tasks while ignoring others. The other side of the coin are limits on *divided attention*: the ability to switch between tasks or stimuli. There is a trade-off between focused and divided attention: the same similarity in display that makes it harder to focus on one stimulus and ignore the other makes it easier to switch focus between two stimuli. This leads to the *proximity compatibility principle*, which states that high display proximity (similarity) helps in tasks with similar mental proximity, and where information is related and needs to be treated together **(Wickens, p. 98)**. In other words, information should be similar in display and structure to parts of the environment to which the information is similar mentally, and dissimilar otherwise. For example, military pilots use head-up displays to place annotations visually on top of enemy and friendly aircraft. This use of spatial proximity helps link the annotation to the object, but can make it more difficult to process only one without the other. If aircraft information was instead displayed in a table, confusion and errors could occur if a friendly plane happened to fly close to the display entry for a different (enemy) plane.

Divided attention is also easier when switching between tasks does not require a large shift in the contents of short-term memory. This observation is obviously the case when one of the tasks has low memory requirements. For example, turning on the light while talking on the phone requires little shifting of memory because the location of the light switch can be seen: it is knowledge in the world. However, finding an object or reading email while on the phone requires more swapping of short-term memory between tasks, and the phone conversation will probably suffer. Short-term memory also requires less swapping if the two tasks share similar mental models, or schema. For example, several schema will be shared by tasks that relate to the same general task or subject. This relates back to the proximity compatibility principle. Miyata and Norman describes the situation as follows **(Miyata 1986)**:

(Miyata 1986)
Miyata, Y. and D. Norman, Psychological Issues in Support of Multiple Activities, in *User Centered System Design*, Norman and Draper (eds.), 1986, pp. 268, 270

> *Because of a person's limited processing and memory capacity, one suspends work on current activity at the risk of losing track of the current activity by failing to resume the work where it was interrupted.... It is obviously difficult to maintain a task-driven state in the presence of external events irrelevant to the main task. Task-driven processing continues when the processing is dominated by the schemas relevant to the activity.*

Finally, when dividing attention between several visual information sources people use mental models of probabilities of events to help guide their sampling. For example, airline pilots will glance at rapidly changing displays more often than slowly-changing displays because the probability that the display has moved becomes high more quickly **(Wickens, p. 78)**. Such behavior fits nicely into the accuracy-benefit framework described in Section 3.1.2.

3.3.3 Implications For JITIRs

JITIRs need to allow a person to focus his attention on his primary task, but also to divide his attention between the primary task and the information provided by the JITIR.

To make focused attention easier, a JITIR should use different modalities or channels than are used by a person's primary task. This choice of the lesser-used channel is especially important when the primary task is demanding. For example, Jimminy is designed to give information to a person as he engages in conversation or listens to lectures. In these environments the auditory modality is primary, so Jimminy uses a visual display for output.

It is also important that suggested information is not linked to parts of the environment to which it is not relevant. For example, if a JITIR is giving information related to a text-editor, the display should not be near the web browser, nor should it have a color scheme or layout that associates it with the web browser. Doing so would encourage checking the JITIR output at times when the suggestions are almost guaranteed not to be related to the current task.

On a related note, it is important to be able to distinguish a suggestion from the environment. For example, it must be clear that a suggestion on a web page comes from Margin Notes and is not actually a part of the original web page being viewed. One method is to insure that suggestions appear out-of-band, for example in a separate window or different modality. Another method is to insure that suggestions look nothing like the user's other context. For example, annotations in an augmented reality system are never mistaken for the real world because the resolution of computer graphics is not yet high enough. The third method is branding: insuring that annotations have a unique look and feel that sets them apart and identifies them as output from the JITIR. For example, annotations could use a different font, color, location or modality than the information being annotated.

To make divided attention (task switching) easier, a JITIR should display information in a way that is congruous with the parts of the environment to which it relates. For example, it is likely easier to look at suggestions from an email archive when reading or writing email because the two formats have the same mental model. Similar mappings between suggestion and the context to which it is relevant can be achieved with color and font. Probably the most effective way to link information is to use spatial proximity: put information near what it is about. In the Remembrance Agent, the sug-

gestion display is in a buffer within the editor window rather than a separate window. This links the RA with the particular text being edited and keeps it from being linked with other applications running on the desktop. In Margin Notes, annotations appear to the right of the paragraph or section to which they relate. Moving the scroll bar in the web browser moves the annotations as well, increasing the mental linkage between the two.

The Margin Notes example reveals another use of spatial proximity: it can indicate to which part of a user's context a suggestion is relevant. Even if a person's full context is constrained to a single web page, it should still be apparent whether a suggestion is relevant to a single paragraph, a section, or the entire web page. Spatial proximity is a good way to indicate this information, although when this is not possible the indication can still be by fiat, e.g. by declaring that suggestions are chosen based on relevance to the whole body, or by indicating the scope of relevance in the suggestion some other way.

With proper design a JITIR can allow a person to both focus on his primary task and still divide attention between the JITIR and the primary task when desired. The divided attention is made easier by the fact that the subject of provided information is related to the person's current context, and presumably to his primary task. While attending to a JITIR does require a shift of focus, it does not require a complete change of subject context or the mental schema that are already in working memory. The proper interface will create similarity in information display when those parts of the task and the information suggested have similar mental schema (that is, when they deal with the same subject). For example, a JITIR's display should be collocated with the application to which it relates. Similarly, the information should be easily distinguished from parts of the task environment to which the information does not relate. Finally, information should always be easily separable from the surrounding environment when the task environment is especially demanding on one's attention, as is the case with Jimminy.

Finally, it should be noted that the amount of attention spent visually scanning JITIR suggestions will depend on the expected likelihood that the agent is showing useful information. This probability will depend on previous performance; if in the past the agent rarely showed useful suggestions then it will receive less attention. Of course, usage patterns will also change based on the specific task. For example, if a person is writing about a subject that he knows is in the agent's database he will be more likely to scan the suggestions for something useful. The relative attention to the agent will also depend on the cost of scanning: if the cost for divided attention is high then the agent will be scanned less often.

3.3.4 Knowledge in The Head and in The World

There are two places that knowledge can reside: in the head or in the world. Knowledge in the head is information known by the user. This can be memories of past events, knowledge about how the JITIR operates, or knowledge about a certain topic. Knowledge in the world is information that is in the environment of the primary task, e.g. the text currently being written or read and the way the information is displayed. Knowledge in the world can be subdivided into knowledge that is already in the world (e.g. in the user's primary task environment) and knowledge that is explicitly conveyed by the JITIR.

To understand the contents of a suggestion or document a person uses information from all three sources: the head, the primary task environment and the JITIR. For example, say a person is using Margin Notes to read a web page about South Amer-

ica, using her own email archives as a database. Next to the section on Peru, Margin Notes displays email from a friend of hers with the subject line "My trip report," her friend's user name and the date sent. From this minimal information she remembers that her friend had visited Peru the previous Summer, and recognizes that this is the email describing the trip. All three kinds of knowledge are at work in understanding and contextualizing the suggestion summary. The first and most important in this example is knowledge in the head: her memory about her friend's trip. Because she recognizes her friend's email and remembers that her friend was in Peru around the date indicated, that memory conveys a large amount of information not contained in the suggestion line itself. Second is knowledge in the primary task environment, namely the fact that the suggestion is next to a section of the web page describing Peru. This gives further indication that the suggestion is about Peru and not some other trip. If the suggestion were instead next to a section of a page describing Germany, she might assume the email was a different trip report sent by her well-travelled friend. Finally, the text of the suggestion itself conveys information, namely the person sending the mail, the date, and the fact that it regards the subject "my trip report." If she places the mouse pointer over the suggestion she will also get the keywords contained in both this suggestion and in the section annotated.

To minimize the cognitive and perceptual load required to interpret a suggestion, the interface to a JITIR should use knowledge in the head and knowledge in the world as much as possible. In the example above, the fact that the friend's trip report had already been seen made it much easier to guess at the contents of the suggested document, and presumably to interpret the information in the correct context. If the information in the database has not been seen before, as would be the case with the INSPEC database, knowledge about the topic or database can still be used to interpret a suggestion. For example, if a suggestion from the INSPEC database brought up a paper entitled "Statistical analysis of the population of Lima," the user's knowledge of Peru would be used to understand that Lima is the capital. Furthermore, the user's knowledge that Margin Notes was drawing from the INSPEC database would allow the user to assume that the suggestion is an abstract for a technical paper and not, say, a work of fiction or a tourist guidebook entry. When neither the topic nor the contents of the database is well known by the user, the interface for the JITIR must take on the burden of providing more information in a suggestion summary to allow the user to interpret it properly.

3.3.5 Ramping Interfaces

Three assumptions can be made about suggestions produced by a JITIR. First, they will never be useful one-hundred percent of the time. Even with perfect information retrieval there are times when a user does not want more information, or is already suffering from information overload and cannot be distracted further. Second, the user is in the best position to determine if a particular suggestion will be useful, assuming she is given a information about the contents of the suggestion. Finally, the act of determining whether a suggestion might be useful is in itself a distraction and produces cognitive load. Assuming suggestions are at least occasionally useful this cost is acceptable, but the cost of false positives should still be minimized.

One display technique is what this thesis calls a "ramping interface," where information is conveyed in stages. Each stage of a ramping interface provides a little more information, at the cost of requiring a little more attention to read and understand it. The idea is to present useful information, while at the same time allowing a person to detect bad information and bail out as early as possible. Stages convey two kinds of information: content and information about the content. Early stages will most likely convey more information about the content, including summary lines, keywords that

led to a suggestion being made, and dates related to the document. This information might be useful in its own right, but it is more likely that it will be used to evaluate the benefits of looking at the information conveyed in later stages. Later stages will provide information that is likely to be directly valuable (i.e. the document itself). The only exception to this transition from information useful for setting expectations to information valuable in its own right is when the JITIR is used in a high-cognitive-load environment. For example, Jimminy is often used in conversations where the user can only process a few words from the head-mounted display without seeming distracted. In this situation, it is expected that most of the time the single-line suggestion is the final stage achieved. The main use therefore comes from the suggestion lines reminding the user of an event or piece of information. The full document is only retrieved when more detail is required, or in less environments less demanding than conversations, e.g. lecture situations. Ramping interfaces are similar to the concept of *dynamic scaling* used by the Nomadic Radio system **(Sawhney 1999)**, which is described in Chapter 6.2.2.

(Sawhney 1999)
Sawhney, N. and C. Schmandt, Nomadic Radio: Scalable and Contextual Notification for Wearable Audio Messaging. In *Proc. of CHI'99*, 1999

The ramping interface technique both reduces the cost of false positives and reduces the effort required to retrieve just a piece of the suggested information. As described in Section 3.1.4, introducing information in low-cost stages early on allows the user to quickly check whether something useful is being suggested even when the expected benefit is low. If the information suggested is valuable then expectations change and the user will access more information through later stages.

In a ramping interface the user should always be able to get more information by going to the next stage, and the action required to get to that stage should be proportional to the amount of information provided in the current stage. It should only require a simple action such as reading a a few words for a user to go to early stages. Later stages might require the user to click on an icon, trading off simplicity for increased control of what information is displayed. Note that stages are not defined by display actions taken by the JITIR, but rather are defined as pieces of information that can be individually processed by a user. For example, a display that shows a large bold-faced title followed by supporting material has two stages, because a reader can easily process each chunk of information separately and use that information to decide whether to read further. Even a fully rendered web page might contain several stages of a ramping interface (title, headers, etc.), assuming each stage is always displayed in the same location and format so the user can know where to look for the relevant information. In this way, a ramping interface is similar to the inverted pyramid used in journalism, where a newspaper article is always conveyed in the stages of headline, lead sentence, and main paragraph with the details following in the main body.

As an illustration, the ramping interface in Margin Notes works as follows:

TABLE 1. Stages of the Margin Notes ramping interface

No action	Peripheral	Histogram	Read note	Mouse-over	click, read
(agent decides)	(note exists)	(relevance)	(topic)	(keywords)	(document)

The first stage could be referred to as the no-action, no-information stage. In this stage it is Margin Notes, not the user, that decides whether information is available. If the system decides that the information that might be suggested is not worth the potential distraction then it does not annotate, leaving the margin blank. There are several reasons a section might not be annotated. First, the most relevant document may still be below the relevance threshold required. The section may also be too small to annotate,

or the document as a whole might be too small. At this first stage, the input is simply a passive sensor watching the user's actions. No user action or attention is required to show the JITIR what to do. The output at this stage is potentially nothing: the JITIR simply passes the HTML to the browser and continues.

If an annotation is displayed, the user will see it in her peripheral vision. Noticing a suggestion is a small perceptual and cognitive load, and gives a small amount of information (namely, the fact that an annotation exists). The small load is in keeping with the philosophy that the effort required to jump to the next stage should be commensurate with the amount of work or attention required by the current stage. At this point the user may ignore the suggestion entirely, and the interaction has cost nothing more than some screen real-estate and a very minor cognitive load.

If she wishes to go to the next stage, she can quickly view a graphical line of filled-in circles indicating the relevance value for that suggestion. To do this the user must actively move her focus of vision and process the graphics (a perceptual process). In earlier versions a two-digit number was used to indicate relevance. While the same information was available, it was more difficult to find and process quickly. The relevance bar, on the other hand, only requires perceptual processing to determine the relevance of a suggestion.

The next stage is reached by reading the suggestion description, which requires further attention on the part of the user. Information in the annotation is as concise as possible to allow rapid scanning for content. The box also contains many different kinds of information to try to contextualize the suggestion. For example, when email is displayed the box contains subject, author, date and archive filename in the hope that at least one of these will be a good indication of what the suggestion is about.

The user can get to the fifth stage by moving the mouse over the annotation, which causes the keywords associated with the note to appear in a pull-down window. Going to this stage requires physical action by the user (a mouse-over). While keyword information could have been included in the original suggestion (reducing the system to a five-stage ramping interface) it was decided that this information made the annotation too cluttered.

To jump to the final stage, the user clicks on the link and gets the fully suggested text. At this point she is completely committed to seeing the text, and has been distracted from her primary task. Hopefully if the user gets to this point the suggested text is worth the attention spent.

Note that the actions required to go through the stages of the Margin Notes ramping interface in order are also the natural actions to get more information in the context of web-browsing. The user sees a link, reads it, moves the mouse to the link and clicks. This allows the user to jump to the full suggestion quickly, while still seeing all stages. It is also possible to skip stages in the ramp. For example, the user could read a suggestion and immediately click to see the full document without ever reading the keywords. It is also possible to leave a stage halfway through. For example, the user might read a suggestion’s title or even just part of the title and never read the author or date fields.

When designing a ramping interface it is also helpful to consider at what stage a user is likely to receive the information he needs. In the Margin Notes system, it is assumed that most of the time users will receive information from the full suggested text, but that occasionally they will be reminded by the suggestion note itself and

never need to follow the link. On the other hand, a JITIR designed for a car or other attention-demanding environment might be designed such that users normally need not go past the first stage of a suggestion, except in special circumstances.

CHAPTER 4 *Implementation Details*

God is in the details.
– Ludwig Mies van der Rohe

This chapter will describe the design and implementation details of Savant (the information retrieval back-end), the Remembrance Agent, Margin Notes, and Jimminy. It goes into enough detail to act as both a user guide and documentation for modifying or reimplementing the systems. For high level descriptions, see Chapter 2.

4.1 Savant: The Information-Retrieval Back End

All three implemented JITIRs use the same back-end system, called Savant. The front-end senses the environment (that is, the document or email being written, the web page being viewed, or the physical environment of the wearer of a wearable computer) and sends that information in text form to Savant as a "query." Savant then works as an information retrieval engine: given a query it produces a rank-ordered list of pre-indexed documents that best match the query. Savant consists of two programs: *ra-retrieve* performs information retrieval based on a query, while *ra-index* creates index files so retrieval can be performed quickly. Indexes can be created from generally useful sources such as a collection of newspaper or journal articles, organization-wide collections such as office memos, or from personal sources such as email and notes. Previous versions also allowed pages to be indexed directly from the web. Documents are usually re-indexed nightly to incorporate new changes and additions.

4.1.1 Template Matching

The power of Savant comes from a strong template-matching system that can recognize documents, parse out fields, and automatically index the documents based on their component fields. For example, if pointed at a top-level directory of heterogeneous files it will recognize and parse email archives, HTML files, LaTeX files, notes taken on the wearable computer, paper abstracts from the INSPEC database, and raw text while ignoring other file formats. It will also break email archives into individual messages. This parsing means indexing can be performed completely automatically with no hand annotation or labelling of documents necessary. Different fields from documents are identified and indexed separately. For example, the from field of email archives are indexed as *people* while the title fields of HTML documents and the subject fields of email messages are indexed as *subjects*. The template system is also used for queries, so fields can be parsed out of email as it is being written or web pages as they are being read.

As of the most recent version, templates are defined in Savant's source code rather than a configuration file, but are designed to be easily modified or added with a recompilation. For example, Figure 10 shows the template for parsing an RMAIL type email archive:

FIGURE 10. Example template code for Savant

```
                    /*******************************/
                    /*    RMAIL archive template   */
                    /*******************************/
                    current_template = create_template(
(1: Name)                          "RMAIL",
(2: ID Regexp)                      "BABYL OPTIONS",
(3: Delimiter)                      "^_^L",
(4: Action)                         ACCEPT_ACTION,
(5: Type)                           INDEX_TYPE);
                     atfn(
(6: Field Name)                     "SUBJECT",
(7: Field ID Regexp)               "Subject:\\s*(.*?\n)\\S",
(8: Parentheses Chosen)             1,
(9: Filter Bank)                    email_subject_filter,
(10: Field Weight)                  1,
(11: Field Title Length)            90);

                     atfn("BODY", "\n\n(.*)$",1,
                           email_body_filter,4, 0);

                     atfn("PERSON",
                          "From:\\s*(.*?\n)\\S", 1,
                           email_from_filter, 1, 50);

                     atfn("DATE",
                           "Date:\\s*(.*?\n)\\S", 1,
                           NULL, 1, 40);
```

In this example, the first five lines define the template structure itself:

1. The name of the template is "RMAIL."
2. An RMAIL file can be identified because it contains the words "BABYL OPTIONS" in the header. This string can be any Perl-style regular expression **(Siever 1999)** that matches within the first 500 characters of a file of the desired type.
3. RMAIL files can contain many individual emails; each are separated by the delimiter control-underscore, control-L.
4. Individual files can be rejected based on their type using the *action* parameter. For example, both Postscript and PDF file types are rejected, while RMAIL files are accepted.

(Siever et al 1999)
Siever, E., et al., *Perl in a Nutshell*, 1999, p. 63-70

5. Templates can be either for indexing or for queries, as determined by the *type* parameter. Index templates are used by *ra-index* to identify files and break them into fiends to save to disk. Query templates are used by *ra-retrieve* to parse a person's current environment into fields for comparison against pre-indexed documents.

The next four commands define fields that exist within the RMAIL template. Fields are individual data types, each with its own data representation, indexing method and similarity metric. Fields will be discussed further in Section 4.1.3.

6. Each field is added by name with the "add template field name" macro. Possible field types (e.g. *body* or *subject*) are defined earlier in the template system.
7. The field name is followed by a Perl-style regular expression that is used to match the text contained in a particular field. For example, the subject field is identified by the keyword "Subject" followed by a colon and zero or more spaces, followed by text. The end of the field is a carriage return followed by non-whitespace. This definition describes the standard format for subject lines in email.
8. The number following the regular expression indicates which set of parentheses in the regular expression contains the desired text. In this case, the first (and only) set of parentheses matches the subject line itself without the "Subject" keyword.
9. The next argument points to a set of filters that will be applied to the field data. This set of filters is defined earlier in the template system, and will be described shortly.
10. Next is the bias (weight) for this field in the retrieval process. In the case of RMAIL, the body of the message gets a bias of four while the other fields get a bias of one. Biases associated with query-type templates can be modified by the front-end, biases associated with an index-type template can not.
11. The final number identifies the maximum number of characters from this field that get saved to disk and sent to the front-end (90 in the case of the RMAIL Subject field).

The template structure can currently parse fields of types *body*, *location*, *date*, *time*, *day*, *subject* and *person*. The following templates are currently defined, with the following associated fields. Default biases are listed in parenthesis next to the field name.

RMAIL: RMAIL email archive format

- *person* (1): who the email is from
- *subject* (1): subject line for this email
- *date* (1): when this email was sent
- *body* (4): main text of the email

Unix email archive: Standard Unix-style format for email archives

- *person* (1): who the email is from
- *subject* (1): subject line for this email
- *date* (1): when this email was sent
- *body* (4): main text of the email

Athena email archive: Email format used by the MIT Athena system

- *person* (1): who the email is from
- *subject* (1): subject line for this email
- *date* (1): when this email was sent
- *body* (4): main text of the email

***Boston Globe*:** Format for the *Boston Globe* archive used at the Media Lab

- *person* (1): author of the article
- *subject* (2): title of article
- *date* (1): when article was written
- *body* (2): main text of article

***MIT Tech* newspaper:** Format for articles of the *MIT Tech*

- *subject* (1): title of article
- *date* (1): when article was written
- *body* (1): main text of article

HTML: HTML format

- *person* (1): email address(es) in any “mailto” fields
- *subject* (1): title of web page
- *body* (1): main text of web page

Jimminy: The format used by Jimminy for notes taken on the wearable

- *location* (1): physical location in which the note was taken
- *person* (1): people who were nearby when note was written
- *subject* (1): subject of note
- *date* (1): date note was taken (based on timestamp)
- *time* (1): time the note was taken (based on timestamp)
- *day* (1): day of the week the note was taken (based on timestamp)
- *body* (1): main text of note

Edupage archive: Format used by the *Edupage* news clipping archives

- *subject* (1): headline for the news summary
- *date* (1): dateline for the news summary
- *body* (1): main text for news summary

LaTeX format: Format for writing page layout and technical papers

- *subject* (1): title of the paper
- *body* (1): main text for the paper

INSPEC: Format for compilations of citations from the INSPEC database

- *person* (1): author(s) of the paper cited
- *subject* (1): title of paper cited
- *date* (1): date of publication for paper cited
- *body* (4): abstract for paper cited

ACM: Format for citations in the *ACM Electronic Guide to Computing Literature*

- *person* (1): author(s) of the paper cited
- *subject* (1): title of paper
- *date* (1): date of publication for paper
- *body* (1): title and keywords for paper (this database does not include abstracts)

Plain text: Normal, unrecognized text (the default)

- *body* (1): the full text of the document

The following file types are rejected and not indexed:

Postscript: The Postscript document formatting language

Framemaker: Files for FrameMaker desktop publishing

PDF: Adobe Acrobat's PDF format

HQX: The Macintosh HQX compressed-file format

RCS-control: Control files for RCS version control system

Binary: Any binary (non-text) file is automatically rejected

The previous templates are all used to recognize and parse files being *indexed*. The following templates are defined for recognizing and parsing a user's local context during *retrieval*:

RMAIL: RMAIL being read, as formatted by the Emacs rmail-mode reader

- *person* (1): who the email is from
- *subject* (1): subject line for this email
- *date* (1): when this email was sent
- *body* (4): main text of the email

Mail: Email being written, as formatted by the Emacs mail-mode

- *person* (1): who the email is from
- *subject* (1): subject line for this email
- *date* (1): when this email was sent
- *body* (1): main text of the email

GNUS: Net news being written or read, as output by the Emacs GNUS news reader

- *person* (1): who a news posting is from
- *subject* (1): subject line for this news posting
- *date* (1): when this message was posted
- *body* (1): main text for this news posting

Margin Notes: Single sections of HTML pages, as returned by Margin Notes

- *person* (1): email address(es) in any “mailto” fields
- *subject* (1): title of web page, or the text in a main header (H1) or sub-header (H2)
- *body* (5): main text of web page section

HTML: HTML being written, as returned by the Emacs html-mode

- *subject* (1): title of web page being written
- *body* (1): main text of web page being written

LaTeX: LaTeX format file being written, as returned by the Emacs latex-mode

- *subject* (1): title, section or subsection text of the paper being written
- *body* (1): full text of the document

Jimminy context format: Format Jimminy uses to express the user’s local context

- *location* (1): user’s current physical location
- *person* (1): people currently nearby
- *subject* (1): current subject of conversation (as entered by user)
- *date* (1): current date (based on system clock)
- *time* (1): current time of day (based on system clock)
- *day* (1): current day of the week (based on system clock)
- *body* (1): text of any note currently being written

RA manual query: Format used by the RA to express a manual query (C-c r q)

- *location* (1): location field for query
- *person* (1): person field for query
- *subject* (1): subject field for query
- *date* (1): timestamp for query
- *time* (1): time of day for query (based on timestamp)
- *day* (1): day of the week for query (based on timestamp)
- *body* (1): body field for query

RA field query: Format used by RA to express a field query (C-c r f)

- *location* (1): location field for query
- *person* (1): person field for query
- *subject* (1): subject field for query
- *date* (1): timestamp for query

- *time* (1): time of day for query (based on timestamp)
- *day* (1): day of the week for query (based on timestamp)
- *body* (1): body field for query

Plain text: Any text currently in the user's local context (default)

- *body* (1): text being written

4.1.2 Filters

As mentioned in Section 3.2.7, JITIR queries tend to contain extraneous pieces of text such as signature lines and email headers that are not useful for retrieval. Indexed documents will likewise have HTML markup and headers that will dilute the value of important data when selecting documents. To address this problem, each template can associate a filter bank with each field. A filter bank is an ordered list of Perl-style regular expressions that match text that should be removed from the field before parsing. For example, filters associated with the email *body* field recognize and remove email signature lines, headers from included files and common lines such as "Begin forwarded message." Filters associated with the email *person* field remove all information except for the user name, while filters associated with all fields in the HTML template remove hypertext tags and comments.

4.1.3 Fields

Fields are abstract data types that span across different kinds of documents. For example, HTML documents and email can both have a subject associated with them, even though the subject is the "Subject" line in email and the "Title" field in HTML. Currently defined fields are *body*, *location*, *date*, *time*, *day*, *subject* and *person*. Earlier versions of Savant used *date*, *time*, and *day* in indexing and retrieval, but the latest version does not. The latest version does still support using the date field as a part of the data returned and displayed when showing a suggestion line or annotation.

This field structure allows Savant to take advantage of multivariate queries and documents. For example, in email sometimes a document associated with the sender is useful, other times a document related to the body of the email is useful. Savant can produce documents related to either or both fields.

Each field is associated with six methods: parser, deparser, index-store, next-word, update-sims, and free-parsed.[1] These methods decompose the indexing and information retrieval process into abstract steps. New data types can be added by implementing these six methods for the particular type. For example, a collaborator at British Telecom has implemented a field type that indexes Global Positioning System (GPS) coordinates. The decomposition of indexing and retrieval also makes code reuse simple. For example, it is possible to change the text-retrieval algorithm and weighting scheme by writing a new update-sims module, but reusing all other modules.

FIGURE 11. Parser

```
void *parser (char *fielddata, Field *self,int docnum)
```

Parse a field from an indexed document or from a query. Turn it into machine readable format.

1. In an object-oriented language these would be implemented as true methods. Because Savant is written in C, they are implemented using function pointers.

fielddata: a string of data for a field as it comes from the document (after filtering). E.g. the raw text from the body of an email message.

self: the field itself. This can be ignored, or can be used to encode different fields differently. E.g. in text the typenum encodes the field type so BODY text is never compared with SUBJECT text.

docnum: the document number. This is a serial number, one per doc.

RETURNS: a pointer to some machine readable structure used by deparser, index_store, nextword, and cleanup_parsed. The format of this structure depends on the particular kind of field.

FIGURE 12. Deparser

```
GBuffer *deparser (void *parsedata, Field *self)
```

Take field data output from the parser and turn it into printable text. This is used for debugging and user feedback.

parsedata: a pointer to parsed data. This is of the type returned by parser.

self: the field itself.

RETURNS: a GBuffer (growable string type) containing human-readable text representing this parsed field data. This is mainly used for debugging strings (printed with the -d option).

FIGURE 13. Index-store

```
void index_store (void *parsedata, char *dbdir,
                  int last_write_p)
```

Take field data output from the parser and store it to disk.

parsedata: a pointer to parsed data. This is of the type returned by parser, and contains the info to be indexed.

dbdir: a string containing the fully expanded directory name of the index files. This is the directory where all the files should be written. Temporary scratch files can be written here as well, though they should be deleted after the final write.

last_write_p: this is set to one if this is the final write. This lets the procedure know it is time to close files, merge checkpoint files, delete scratch files, and perform other cleanup.

RETURNS: nothing. However, this function should write this field info to disk in whatever format is most suited for later fast retrieval.

FIGURE 14. Nextword

```
void *nextword (void *parsedata, int reset_p)
```

An iterator: Take field data output from the parser and return the next "word" from the parsed output. Word is loosely defined as a single element, e.g. a single word, GPS location, date, person's name, etc.Nextword is only called on a query field during retrieval, not on indexed document fields.

parsedata: a pointer to parsed data. This is of the type returned by parser.

reset_p: if reset_p == 1, start with the first word of this parsed data. Otherwise, return the next one. Nextword is responsible for keeping it's own place (via static memory, presumably).Yes, it's icky, but it works.

RETURNS: The word type includes any information that might be needed by the retrieval system, e.g. word weight, machine readable version of the word, normalization info, etc. The word might also contain a human-readable version of the word, for filling into the top contributors keyword list by update_sims_word. The return value is used by update_sims_word during retrieval. Return NULL when there are no more "words."

FIGURE 15. Update-sims-word

```
void update_sims_word (void *word,
                       Remem_Hash_Table *all_sims,
                       Field *field,
                       Retrieval_Database_Info *rdi)
```

A procedure that takes a word at a time and updates (adds to) an array of document similarities. This procedure can be any algorithm so long as it can handle updating a document similarity one word at a time, without seeing all the other words. (This is in the interest of speed. If an algorithm actually needs global knowledge, like say the length of a query or document for normalization, the information can be squirrelled away either in the index files format for this type or in the value returned by nextword.) Called during query retrieval, not indexing. Update_sims_word also needs to update the Top_Contributors list, which contains the similarity and printname of each word that contributed the most to choosing each document.

word: the single word that is potentially adding weight to a document. Of the type returned by nextword.

all_sims: an array of similarities, indexed by docnum.Similarities include an array of "top contributors" to a document being chosen (used for user feedback of why this document was chosen).

field: This is the field of this data (and can be ignored if not needed).This might be useful to grab other function pointers.

rdi: Document info that might be useful to an information-retrieval algorithm. This can include information such as total number of documents and the expanded path for the index files so they can be opened.

FIGURE 16. Cleanup-parsed

```
void cleanup_parsed (void *parseddata)
```

Free all the memory allocated by the parser routine.This is necessary because C doesn't garbage collect.

parsedata: a pointer to parsed data. This is of the type returned by parser.

RETURNS: nothing.

The currently defined fields use the following versions of the six procedures listed above:

Body* and *Subject:

- *parse-text:* a parser that breaks text into words, throws out "stop" words from a list of common words, and stems those words using the Porter stemmer described in Section 4.1.6.
- *deparse-text:* a deparser that turns the machine-readable term into a printable (though stemmed) word.
- *index-store-text:* an indexer that stores words contained in documents in an inverted file structure
- *nextword-text:* an iterator that returns the next word from a parsed word vector
- *update-sims-word-text-okapi:* a similarity update procedure that uses the Okapi weighting scheme described in Section 4.1.6.
- *free-parsed-text:* a procedure that frees the memory allocated in *parse-text*

Location* and *Person:

- *parse-text-nostopstem:* parse text into individual words, but don't remove stop words or stem the words. Stemming is not performed because locations are often textual descriptions of room numbers such as "E15-305d" that are not conducive to being stemmed by algorithms that expect normal English words.
- *deparse-text*
- *index-store-text*
- *nextword-text*
- *update-sims-word-text-tfidf:* a similarity update procedure that uses TF/iDF with the alternative (non-Okapi) weighting scheme described in Section 4.1.6.
- *free-parsed-text*

Date, Time and Day: These fields are not indexed in the latest version of Savant. Previous versions used a date-specific parser, deparser, and update-sims-word, but reused *index-store-text*, *nextword-text*, and *free-parsed-text* because dates were stored in the same inverted file structure used for text.

4.1.4 Index Algorithm

Indexing is performed by *ra-index* with the following syntax:

```
ra-index [--version] [-v] [-d] [-s] <base-dir> <sources>
         [-e <excludees>]
```

The *-v* option gives verbose descriptions, *-d* is debug mode, *-s* indicates to follow symbolic links. The <*base-dir*> is the directory where index files are put. This is followed by <*sources*>, a list of top-level directories that are to be scanned for indexable files. Subdirectories are recursively scanned. Dot-files and Emacs backup-files are ignored. The optional <*excludees*> argument lists directories and files that should not be indexed. The index algorithm is as follows:

Identify files for indexing. The directories listed in <*sources*> are scanned recursively and a list of all applicable files is created.

For each file:

> **Recognize file type.** If the file is binary it is discarded (files with more than 5% unprintable characters in the first 2K are considered binary). Otherwise it is compared to each defined template type, in order, until a match is found. The final template "plain text" matches all documents. If the action type for the matching template is "REJECT" it is discarded. Note that a file is recognized based entirely on its content, not by file-name extension or location.
>
> *For each document in the file:*
>
> > **Find the next document within the file.** Some files (e.g. HTML) have only one document. Others, such as email archive files, have many separate documents per file. Individual documents are found using the delimiter parameter of the file's template, which marks the point between documents.
> >
> > **Find fields.** Find the contents of the fields that are defined in this file's template, using the template's regular expressions.
> >
> > **Filter fields.** Run each field through its associated bank of filters. Continue to apply filters, in order, until either the text does not change or there is no text remaining.
> >
> > **Parse fields.** Parse each field using its field-specific parser, and store the machine-readable output.
> >
> > **Write document information to disk.** This stores two kinds of information. First, it stores filename and offset information to a document-locations file. Second, it stores title information that is used as output when a suggested document is sent to a JITIR front-end.
> >
> > **Store parsed field data.** Store each parsed field using the field-specific *index-store* method. Often the storing procedures will use scratch files to avoid running out of RAM while indexing large corpora.
> >
> > **Clean up parsed data.** Free memory allocated by the *parse-fields* step using the field-specific clean-up method.

Finalize writes to document files. This step finalizes writes to the document-location and title files, and closes the file.

For each field defined:

Finalize field writes. Call the index-store method for each defined field, with the *last-write-p* variable set to indicate this is the clean-up phase. This allows field-specific procedures to close their files and possible to merge scratch files that have been written to disk.

4.1.5 Retrieval Algorithm

Retrieval is performed by *ra-retrieve*. The function can be executed manually (usually in verbose mode), but usually the program is called and controlled by the JITIR front-end. The program is called with the following syntax:

```
ra-retrieve [--version] [-v] [-d] <base-dir>
            [--docnum <docnum>]
```

The *-v* option gives verbose descriptions, *-d* is debug mode. The <*base-dir*> is the directory where index files are kept. Calling with the *--docnum* option makes ra-retrieve return the document specified and exit. Less memory is used by this option, causing a faster retrieval.

Unless the *--docnum* option is used, ra-retrieve enters an interactive mode where the following commands can be accepted:

query [n]: Find the n most relevant documents to the query text that follows. Default is five. After entering this command the query-text is entered, followed by a control-d or by a carriage-return, control-e, carriage-return.

retrieve n: Retrieve and print the document with document number n.

loc-retrieve n: Retrieve and print the document location (full file name and character offset) for document number n.

info: Display version number and number of documents in the current database.

quit: Quit

help or ?: Display help information.

print-biases: Print the list of hand-set biases. Also print whether hand-set or template biases are being used.

use-handset-biases: Use the query biases in the hand-set list, rather than using the biases indicated by the query's template. The default is to use template biases, but Jimminy uses hand-set biases.

use-template-bases: Use the query biases indicated by the query's template.

set-bias <field-name> <bias>: Set the hand-set query bias for *field-name* to *bias*.

Suggestion ranking can be based on many different fields. For example, an email message can be relevant to a query based on similarity in the from field, date, subject line or body of the message. Each field can use a different algorithm to determine relevance, so a date in a query will add relevance based on how many days difference there is between the query and a potential document, while the relevance of a body of an email is based on the Okapi version of the Term Frequency / inverse Document Frequency algorithm (TF/iDF). Some versions of Savant also handle GPS coordinates, and the entire system is designed so that adding new data types is straightforward.

When a query is entered, the retrieval algorithm works as follows:

Recognize the query based on template type. The query is compared with each template of type QUERY_TYPE until a match is found. The last query template is a default that matches any text. If the template's action is REJECT_ACTION, the query is ignored and no suggestions are returned.

Find fields. Find the contents of the fields that are defined in the query's template, using the template's regular expressions.

Filter fields. Run each field through its associated bank of filters. Continue to apply filters, in order, until either the text does not change or there is no text remaining.

Parse fields. Parse each field using its field-specific parser, and store the machine-readable output in RAM.

For each field:

> **Find next word.** Find the next word in the field using the *nextword* function defined for this field type. Word is loosely defined as any individual term or element, and could be a word in a text, a single location in a list of locations, or other atomic unit.
>
> *For each word:*
>
> > **Update similarities list.** Update the running list of similarities of each document to this field of the query by calling the *update_sims_word* procedure defined for this field. The procedure will examine the index files written by *ra-index*. The procedure also updates the *Top_Contributors* list, which contains the similarity and printable name of each word that contributed most to the choosing of a particular document. This is used to produce the keywords list that is displayed in the interfaces. Similarities returned for a field are guaranteed to be between zero and one. If an *update_sims_word* function does return a value out of this range, the value is reset to zero or one.
>
> **Bias similarities for this field.** Apply biases to the similarity returned for each document based on this field. The bias process produces a total similarity based on a linear combination of index biases and either the hand-set or template-based query biases. For a given set of query-biases and index-biases, the combination is computed in the following way:
>
> $$\textit{query biases} = q_1, q_2, \ldots, q_{\text{num-fields}}$$
> $$\textit{index biases} = i_1, i_2, \ldots, i_{\text{num-fields}}$$

$$\textit{non-normalized biases} = q_1 i_1, q_2 i_2, \ldots$$

$$M = \textit{combined biases sum} = q_1 i_1 + q_2 i_2 + \ldots$$

$$\textit{normalized bias} = \frac{q_1 i_1}{M}, \frac{q_2 i_2}{M}, \ldots$$

$$\textit{biased similarity} = (\textit{similarity})(\textit{normalized-bias})$$

Compute total similarities for documents. Total similarities is equal to the sum of all field similarities for a document:

$$\textit{total-similarity} = \sum_{i=1}^{\text{num-fields}} \textit{biased-similarity}_i$$

Given that individual similarities are between zero and one, this value is guaranteed to be between zero and one as well.

Sort documents in rank order. Sort documents by total similarity. To improve speed, the algorithm only guarantees sorting of the top n documents, where n is the number of suggestions to return.

Print top documents. Print the suggestion information. Printed information is a line number, relevance score from zero to one, and document number, followed by each defined field up to the maximum number, followed by keywords that lead to the suggestion, followed by the amount by which each field contributed to the total similarity (after biasing). Each piece of information is separated by the pipe symbol ("|").

Free memory. Run the *cleanup_parsed* procedure to free allocated memory.

4.1.6 Text Retrieval Similarity Metrics

Even though Savant is a general information retrieval architecture, the most common uses rely heavily on text retrieval techniques. Unfortunately, there is no clear winner in text retrieval algorithms. At best, various heuristics and techniques are known that tend to work well in some cases, and these techniques are often tweaked or modified depending on factors such as document size, query size, and limited vocabulary **(Salton 1988)**. This research is not setting out to create yet another text-retrieval engine, especially since it is clear no algorithm will be well suited to every task domain. Instead, two known algorithms have been implemented within Savant, with enough variation to demonstrate the ability to mix and match different procedural components. These components will be described below. The particular algorithms that are available in Savant are well-known in the literature, but no attempt is made here to show that they are the best of all possible algorithms.

(Salton 1988)
Salton, G. and C. Buckley, Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24:513-523, 1988

All of Savant's text retrieval plug-in procedures are based on the Term Frequency / inverse Document Frequency method (TF/iDF), which represents text in terms of vectors **(Salton 1975)**. The simple *parser* procedure converts text into a frequency vector with each term representing the number of times a particular word is used in the document or query. This is currently used for the *person* and *location* fields. Free-form English text fields such as the *body* field use a more complex *parser* procedure that removes stop-words and stems the remaining words. Stop-words are from a predefined list of common words such as "and," "a," and "the." Such words do not con-

(Salton 1975)
Salton, G. et al, A vector space model for automatic indexing. *CACM*, 18:613-620, 1975

vey meaning on their own, and are removed both to reduce the size of index files and to improve the retrieval quality. Stemming is the conflation of words that probably mean the same but have different endings. For example, the word “stemming” and “stemmed” would both be converted to the root word “stem.” The particular algorithm used is the Porter stemmer **(Porter 1980)**. One can perform either *strong stemming* or *weak stemming*, weak stemming is defined as only step 1 of the Porter stemmer (thus stemming fewer variations). Walker and Jones **(Walker 1987)** showed that weak stemming significantly increases recall and does not significantly decrease precision. However, strong stemming decreases precision in favor of more recall. Because JITIRs should favor precision over recall, the current system uses weak stemming. This was a recent modification, so the evaluations described in Chapter 5 use the (probably inferior) strong stemming.

(Porter 1980)
Porter, M., An Algorithm For Suffix Stripping, *Program* 14(3), July 1980, pp. 130-137

(Walker 1987)
Walker, S. and R. Jones, Improving Subject Retrieval in Online Catalogues. British Library Research Paper no. 24, vol. 1, 1987

The three main components of a TF/iDF similarity metric are **(Harman 1992)**:

(Harman 1992)
Harman, D., Ranking Algorithms, in *Information Retrieval: Data Structures and Algorithms*, W. Frakes and R. Baewa-Yates (eds), 1992

- **NTF:** The normalized term frequency for the word within a particular document. The idea is that commonly used words (high term frequency) are more important in a particular document and therefore should be given higher weight.
- **iDF:** An inverse document frequency weighting, which favors (gives higher weight to) rare words. The assumption here is that rare words convey more meaning and distinguish between documents more than common words.
- a method for combining *NTF* and *iDF* to form a single similarity metric for a document given a query.

The algorithms used here add a fourth component as well:

- a normalization factor for the total field similarity, in addition to the normalization for term frequency listed above. At the very least this normalization factor forces similarities between the values of zero and one, which is required by the bias algorithm. The normalization may also normalize for the length of the query in much the same way most traditional text retrieval algorithms normalize for document length. Usually traditional algorithms leave out the query-length normalization because it only applies a constant multiplier to all document similarities *for a given query* **(Salton 1988)**, but because the similarity score is also used to remove (not show) low-similarity suggestions the score should be normalized across multiple queries.

(Salton 1988)
Salton, G. and C. Buckley, Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24:513-523, 1988

The first TF/iDF algorithm that is available in Savant uses the following settings:

$$IDF_T = \log\left(\frac{(N - n_T)}{n_T}\right)$$ **(Croft 1979)**

(Croft 1979)
Croft, W. and D. Harper, Using Probabilistic Models of Document Retrieval Without Relevance Information, *Documentation*, 35(4), 285-95.

$$NTF_T = \frac{\log(freq_{dT} + 1)}{\log(length_d)}$$ **(Harman 1986)**

(Harman 1986)
Harman, D., An Experimental Study of Factors Important in Document Ranking, in *ACM Conference on Research and Development in Information Retrieval*, 1986

$$query\ normalization = \frac{1}{C}$$

$$similarity = \sum_{T \in Q} (NTF \cdot IDF_T \cdot C)$$

where

Q is a query, containing the term (word) T
N = the number of documents in the indexed collection

n_T = the total number of occurrences of term *T* in the collection

$freq_{dT}$ = the frequency of term *T* in document *d*

$length_d$ = the number of unique terms in document *d*

C = a constant (100 by default)

(Robertson 1995)
Robertson, S. et al, Okapi at TREC-3, in *NIST Special Publication 500-226, 1995*

The second TF/iDF method used is the Okapi weighting scheme **(Robertson 1995)**, slightly simplified because no relevance feedback information is available to Savant. The weighting scheme is given by the following similarity:

$$similarity = \sum_{T \in Q} \frac{W \cdot freq_d \cdot freq_q (k_1 + 1)(k_3 + 1)}{(K + freq_d)(k_3 + freq_q)}$$

where

Q is a query, containing term *T*

W is the inverse document frequency = $\log(N - n_T + 0.5) - \log(n_T + 0.5)$

N = total number of documents

n_T = number of documents containing this word

$K = k_1 \cdot \left((1 - b) + \frac{b \cdot dl}{avdl}\right)$

dl = document length (number of words)

avdl = average document length (number of words, rounded off)

$freq_{dT}$ = frequency of term *T* in the document

$freq_{qT}$ = frequency of term *T* in the query

k_1 = a tuning parameter. High value means $freq_d$ is more important.

k_3 = a tuning parameter. High value means $freq_q$ is more important.

b = a tuning parameter. High value means penalize big documents more.

(Walker 1998)
Walker, S. et al, Okapi at TREC-6, in *NIST Special Publication 500-240, 1998*

Defaults for k_1, k_3, and *b* are 1.2, 100 and 0.75 respectively, which are the values used in **(Walker 1998)**.

Two versions of the Okapi algorithm are available. The first normalizes similarities by a constant factor, as is the case for the first TF/iDF algorithm given above. The second normalizes by the query length:

$$normalization\ factor = \frac{C}{length_q}$$

where

C = a constant for giving the right range between zero and one (3.0 by default)

$length_q$ = number of unique terms in the query

4.1.7 Design Decisions

The design decisions and trade-offs for Savant are as follows:

Templates and plug-ins. The template structure and the ability to specify plug-in functions for different fields makes Savant extremely flexible. Domain-dependent similarity metrics (e.g. based on fields in an email message) can be used when available in the domain, but more general similarity metrics such as plain text similarity

can be used as a fall-back. The disadvantage of the template system is that recognition and parsing is based only on the contents of a file; they can not use other information such as the file's name or user-defined hints. This feature could conceivably be added in later versions.

Heterogeneous databases. Databases can contain multiple types of files, e.g. a combination of email, notes and HTML files. This feature allows databases to be compiled by topic rather than type. However, while databases can contain heterogeneous documents, individual indexed files cannot. For example, an archive of multiple RMAIL-format email files must contain only RMAIL files; it cannot shift halfway through to Unix-format email files.

Max document length. Savant automatically stops indexing a file if it finds a single document that is over ten megabytes (10M) in size. If the limit is reached, the indexer automatically goes to the next file. The limit is to avoid situations in which an archive file is incorrectly identified as a type of file with only one document. However, the limit sometimes causes especially long documents to be skipped that could legitimately be indexed.

Breaking up long fields. An earlier version of Savant would break up the *body* field of a document into 50-line chunks. Retrieval of a document would actually be based on similarity of a query to individual chunks, and retrieving the document would automatically jump to the point of similarity. The disadvantage is that document similarity is therefore limited to those 50 lines when more information may improve retrieval performance. The new version does not support the splitting of long documents, but the feature may be added back in later versions.

Fields only comparable if same named type. Different fields are only comparable if they are of the same named type. For example, the subject of an email and title of an INSPEC paper citation can be compared as long as they are both defined as the same type *subject*. Fields of the same type can have different templates and filters depending on file type, but the fields are guaranteed to be stored in the same type of index files. However, fields of different named types cannot be compared, *even if they are of the same underlying index type*. For example, if an INSPEC file had a field called *abstract*, that field could not be compared to an email *body* field even if they were both stored using the same TF/iDF vector-based storage mechanism. This is to simplify the template structure, and make it easy to add additional templates without requiring a rewrite of older templates. If instead any field could be compared to any other field so long as the underlying index types matched it would be necessary to specify which of those many possible field combinations should actually be compared between document types.

Template in source rather than configuration file. A previous version of Savant used a separate, user definable configuration file to define templates. In the latest version this was replaced by a template definition file integrated into the source code, requiring a recompile if the template is modified. This modification was made for two reasons of practicality. First, the change allowed the system to use the C parser to handle the flexibility required for more complex templates. Second, it makes issuing new updates much easier. When the old version was updated there would often be version skew between old configuration files and the new binaries, and users invariably installed one but not the other, causing bugs and confusion.

Bias settings. The biases for both index and query file types are set to defaults defined in the template files. These biases are somewhat arbitrary, set based on beliefs about the importance and expected range of various pieces of data. For example, the *body* field of an email document is given a bias of four times the other field values. This is based both on the assumption that the body of an email message is most important for determining the value of the message to a user, and the assumption that a body field that matches will still have a low similarity score while a match of a user name is more likely to have a similarity score near one because it can either match or not, but nothing in between. *The Boston Globe* template defines the *body* field bias as only two times the other field values, because it is assumed that the title of an article (the *subject* field) is still relatively important in determining the meaning of an article. Because of the arbitrariness of these hand-set biases, one of the goals for future versions is to use machine learning to adjust biases based on usage patterns.

4.2 The Remembrance Agent

The RA front end is written in Emacs LISP, and both communicates the user's current context to Savant and displays the resulting suggestions. When the RA is operating, every five or ten seconds it will snapshot a region of text around the current cursor position and send this information to Savant. Suggested documents returned by Savant are used to update the display.

4.2.1 Customization

Many parts of the RA are user customizable. In Emacs, programs are customized by setting variables in configuration files. The user-accessable variables for the RA are:

remem-prog-dir: The directory where the Savant binaries are located.

remem-database-dir: The base directory for indexes. This is a directory of directories. For example, the base directory *RA-indexes* might contain the subdirectories *notes*, *mail*, and *inspec-files*. Each of these subdirectories would contain index files produced by Savant.

remem-scopes-list: A list of scopes to be displayed, where a scope is of the form *(directory-name, number-lines, update-time, query-range)*. Through this variable the user can configure the RA to simultaneously display suggestions from multiple databases, or from the same database with different amounts of text used for the query. For example, the RA could be configured to use the first three lines to show suggestions from the *mail* database based on the 500 words nearest to the cursor, and the next two lines to show suggestions from the *notes* database based on the past 20 words nearest to the cursor. Update-time is the number of seconds between queries. Each scope spawns a separate Savant process.

remem-load-original-suggestion: If set to true, when viewing a suggestion the RA will actually load the entire file containing the suggested document. If set to false the RA will only displaying a copy of the single document retrieved. This feature is useful if the document viewed is a part of a larger archive, e.g. a single email in a mail folder, because other documents in the file will be loaded into Emacs for context. However, large archive files can also take a few seconds to load.

remem-log-p: If set to true, enable logging.

remem-logfile: File where usage information gets logged. This information is not used by the system, but is useful for debugging and long-term evaluation.

remem-use-major-mode-templates: Each buffer in Emacs has a *major-mode* which is used to determine display and edit customizations. For example, mail is read in RMAIL-mode and written in mail-mode and HTML is written in html-mode. Emacs automatically determines the major-mode from the file's headers or filename extension. When this variable is set to true, the major-mode information is passed on to Savant, which uses the information for template-based recognition of the query type. Queries are then parsed into fields for individual filtering and retrieval, as described in Section 4.1.5.

remem-print-even-bad-relevance-p: Normally low-relevance suggestions are not displayed, and the text "No suggestion" is shown along with a minus sign where relevance score would be. Setting this variable to true indicates that suggestions should be displayed regardless of relevance.

remem-mode-aware-changing: The RA can change databases based on major mode. For example, it can be configured to load the INSPEC database whenever a buffer is visited that is in the LaTeX paper-editing major-mode. Setting this variable to true turns on this functionality.

remem-mode-db-alist: A parameter associating scope information with major modes. Databases and major-modes must be associated by hand, since it is unknown what particular databases a user might have.

remem-buffname-db-alist: The RA can also change databases based on buffer name. For example it could load the *personal-notes* database whenever a file named "diary" is loaded. These associations are set here. These associations are also automatically set when the user changes databases for a given buffer using the "database change" command.

color customizations: All the colors and fonts in the display can be modified to suit a user's environment. Reasonable defaults are defined for both light and dark backgrounds.

formatting customizations: Individual databases can be assigned to specific field formatting. For example, a suggestion from an email database displays its archive filename. The filename is not useful for suggestions from the INSPEC database, so it is not shown for INSPEC. The subject field is also given more space in the INSPEC database because paper titles tend to be longer and more informative than email subjects.

4.2.2 Commands

Commands for the RA are prefixed with *control-c r* (C-c r). The available commands are:

TABLE 2. RA command list

C-c r t (Control-c r t)	Toggle RA on and off
C-c r v (Control-c r v)	View updated results (execute new query now, bypassing the timer)
C-c r # (Control-c r <number>)	Show suggested document
C-c r r # (Control-c r r <number>)	Rate this document
C-c r f (Control-c r f)	Field search
C-c r q (Control-c r q)	Full-page query. This creates a form where field information can be entered, thus using the RA as a normal search engine
C-c r d (Control-c r d)	Change one or more scopes to a different database
Left mouse-click on line number	Show suggested document
Left mouse-click on a field	Perform search on this field. For example, clicking on the PERSON field of a suggestion will search for any other documents associated with that person.
Middle or right mouse-click	View keywords popup window
Resize display window	Automatically shows more or fewer suggestions to fill window. If there are multiple scopes, the ratio of lines allocated to each scope remains the same.

4.2.3 Design Decisions

Several design trade-offs have been made in the Remembrance Agent:

Scope-specific configuration. The RA allows configuration of the display format on a per-scope basis, where each scope contains a database. Format configurations include the length, order, and color of each field which is displayed. For example, the INSPEC corpus requires a longer title (subject) line than does the email corpus because technical paper titles are usually longer than email subject lines. Email databases, on the other hand, show the file name from which the mail message was drawn because this usually represents the folder to which it was filed. The INSPEC database does not show file information because the filename is a serial number and does not carry any meaning. The upside to specifying a single format for each scope is that the fields for all documents within a scope form a single column, making it easier to scan down an entire list of documents for a particular field value. The downside is that if a database for a particular scope is heterogeneous in document type, all documents within the scope must still be displayed with a single format. For example, all documents within a database that contains INSPEC citations, emails and notes files about a particular subject would be displayed with a single set of format configurations.

Loading whole document vs. a copy. When a suggested document is retrieved, the RA can either load the original file containing the document or it can display a copy of the document. The advantage of loading the original file is that the context of the file is loaded as well. For example, retrieving the original email archive file containing a suggested email automatically loads the file with email-display formatting and the ability to see other email messages within the same folder or thread. However, for large archive files this can also cause a delay in viewing a document while the entire

file is loaded. For this reason, by default only a copy of a suggested document is retrieved.

Use of color. By default, the RA's display uses colors to distinguish between neighboring fields. This makes it easier to parse the field data, but bad color choice can also draw they eye away from the user's primary task. For this reason, colors are customizable. Two color pallets are defined as defaults, one for light backgrounds and one for dark backgrounds.

Attempt to avoid duplicate documents. The RA can display the same database in multiple scopes, varying only the number of words used to create the "query." This often means that documents are suggested in multiple scopes, taking up display space with redundant information. To avoid this wasted space, the RA only displays the first occurrence of a document, based on the unique document number in the index. It is also possible that a database will contain duplicates of the same information. For example, the same email message might be saved in multiple folders. However, it is not as easy to detect whether these messages are "functionally equivalent" since it is hard to determine in advance whether differences such as the folder an email is stored in is an important difference. Currently no attempt is made to remove duplicates based on content.

Integration with Emacs Frame. The RA display is a single buffer within the Emacs window. This integration within Emacs creates a mental connection between suggestions and the information the suggestions regard (see Chapter 3.3.2 for more discussion). It also insures that suggestions are near the user's primary focus of attention, so it requires a smaller amount of eye motion to scan the RA's display. Finally, the RA display is automatically iconified, obscured or revealed along with the entire Emacs display. On the downside, the user is not given the ability to move the RA display independent of the Emacs window. For example, it is not possible to extend the Emacs text editing buffer the entire vertical length of the screen and place the RA display to the side of the Emacs window.

Keywords display. The keywords associated with a suggestion are useful for contextualizing a suggestion, but are usually not as useful as the subject, date, or person associated with a document. Due to the large amount of information that may be desirable, keywords can be displayed both by right-clicking on the suggestion line or by making the Emacs window wide enough that the keywords are on the line to the far right. This gives the user control over keyword display simply by resizing the Emacs window.

Feedback not required. When a document is retrieved using the RA, the user is prompted to rate the document returned on a scale from one through five. Even though rating requires only a single keystroke, it was decided that requiring the rating places extra burden on the user. Since the point of the RA is to reduce the effort of accessing information, the feedback is optional.

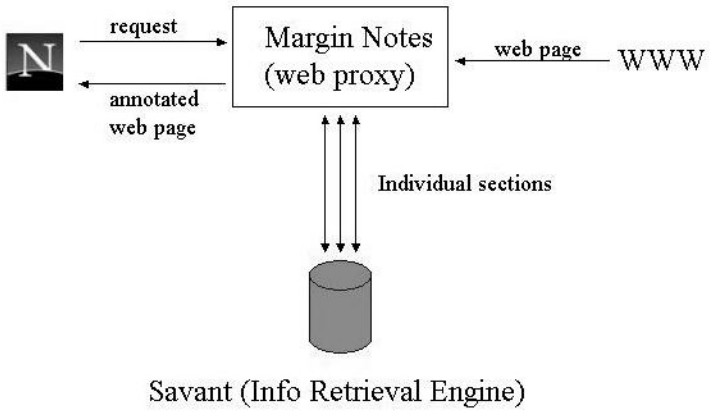
4.3 Margin Notes

Margin Notes is implemented using the Apache-mod-perl web server. The user's web browser is configured to pass all page requests through Margin Notes as a proxy server. The basic idea is simple: Margin Notes sits between the browser and the Web and rewrites the HTML to add annotations to pages as they are sent to the browser. The details are more complex because Savant can take time to retrieve documents, and it is important to get the requested page (the user's primary task) back to the browser without a significant delay. The user experience is that web pages come back

immediately with full main text and a black margin, and suggestions appear in the margin a few seconds afterwards. A status bar at the bottom of the page indicates how close the page is to being fully annotated.

Requests go from the browser to Margin Notes, which retrieves the requested page. If the page is short it is returned to the browser with no annotation. Otherwise the HTML is rewritten to add a black margin along the left-hand side. The margin contains transparent images that will be replaced by annotations after queries are completed. Each section of the web page receives an image, where a section is defined as text between the HTML header tag, horizontal rule tag, and a few non-standard section-break formats used by the MS Word HTML editor. Margin Notes then passes the modified HTML to the browser so the requested page (the user's primary task) can be rendered without additional delay. Figure 17 shows the Margin Notes architecture.

FIGURE 17. Margin Notes architecture



After sending the page to the browser, Margin Notes sends the text of each section to a Savant process, which returns the top documents that match the section. First, duplicates of previous annotations shown on this web page are removed. Then if the highest ranked document is above the relevance threshold an image is created on-the-fly that contains the annotation information. Javascript placed at the bottom of the web page retrieves the new image and replaces the blank annotation that was previously displayed. A status bar at the bottom of the black margin indicates how many of the potential annotations have already been displayed.

4.3.1 Design Decisions

The following are trade-offs made in the Margin Notes design:

Locally annotated section. Unlike the RA, which constantly updates a list of several annotations based on the current cursor location, Margin Notes places annotations to the right of each section being annotated. The advantage to this technique is that the scope of a suggestion is clear; an annotation is about the section it is next to and no other suggestion. Annotations also remain stationary. Unlike suggestions in the RA, after a Margin Notes annotation appears it does not change, which can help eliminate distraction. However, there are two main disadvantages to this technique. First, there is only space for one annotation per section of a web page. The RA, on the other hand,

can display several suggestions about the current section a person is reading by changing them over time. Second, it is not clear what a note is annotating when a web page has multiple columns of information. This problem is especially noticeable with news sites and portal sites that have side bars, navigation links and advertisements surrounding a small piece of main text.

General vs. specific annotations. Another problem with locally annotated sections is that it can lead to a case of "missing the forest for the trees," where annotations are related to sections but no annotation relates to the general topic of the web page. To avoid this problem, the first annotation on a page uses the entire web page as a query, with twice the weight to the first section as the other sections of the page.

Preference for the first line when truncating. Sometimes the information to be displayed in a Margin Note annotation takes more than the seven lines reserved for an annotation. In this case, each field is reduced to one line until all fields fit, starting with the last field displayed. This algorithm insures that fields at the start of the annotation, which include the subject and other more important fields, are truncated last. Less important fields such as the date field are reduced first.

Integration with the browser window. Like the RA, Margin Notes places annotations within the same window as the text being annotated. This integration helps associate annotations with the information in the browser itself (as opposed to other applications) and makes it easier to scan an annotation without being completely distracted from the web page being viewed. However, the same disadvantages described for integration of the RA and Emacs also apply here. Furthermore, the current implementation of Margin Notes achieves integration with the browser by rewriting the incoming HTML code on-the-fly. This rewriting can be difficult for some pages, given that much HTML on the web is not compliant with standards and is not robust to even simple modification.

Length minimums for sections and documents. If a section is too short (less than 100 words by default) then it is not annotated. This minimum length exists for two reasons. First, fewer words in a query tends to produce lower quality suggestions because the IR algorithms have less information to process. Second and more importantly, if a browser window is wide then small sections will take up fewer lines in the display than are covered by a full annotation. Annotating these small sections would require that extra spaces be added to the section, thus violating the idea that the main window text not be modified in ways that are not clearly associated with the JITIR interface. Documents below a certain length (less than 200 words by default) are completely skipped by Margin Notes and are not even annotated with a black margin. This limitation exists because short documents often do not have enough text to establish a primary topic for the IR algorithm, so suggestions are often of poor quality.

Lower bound on similarity. Suggestions below a certain relevance threshold (by default 0.1) are not shown at all for a section, leaving just the black margin. The advantage is that the user is not distracted by suggestions that are not likely to be valuable. The disadvantage is that valuable suggestions might occasionally not be shown. It is also difficult to set these thresholds properly, and the values may be dependent on the particular task and corpus used.

Upper limit on similarity. If a suggestion in Margin Notes is of an extremely high relevance (by default, > 0.65) then the suggestion is ignored and the next most relevant suggestion is displayed instead. This seemingly paradoxical heuristic is because such high relevances are often due to exact copies of the web page being viewed

being suggested, thus producing no new information. A slightly less similar document contains more information that is not already in the web page, and will hopefully still be relevant to the current environment. The downside to this heuristic is that sometimes extremely relevant or valuable documents will not be displayed. The same difficulties in finding a good value for a lower bound on similarity apply to the upper bound as well.

Black margin strip vs. contrasting background. The margin strip that contains annotations is always black, regardless of the color of the page being annotated. The advantage of the constant color is that it gives Margin Notes a stabile look and feel that is the same for all annotated pages. The disadvantage is that the black background may blend in if the annotated page has a black background as well.

4.4 Jimminy

Jimminy is based on the implementation for the Remembrance Agent, and runs within Emacs on a wearable computer. The primary differences between Jimminy and the RA are that Jimminy's display is more compact and suggestions are based not only on notes being typed in an Emacs buffer but also the data returned by physical sensors attached to the wearable.

The hardware for the system is the "Lizzy" wearable computer designed by Thad Starner **(Starner 1997a)**, which consists of a 100 MHz 486 computer running Linux, a head-mounted display and one-handed chording keyboard. The entire system fits in a small shoulder bag. The head-mounted display is the "Private Eye" made by Reflection Technology. Display resolution is 720x280 pixels, monochrome red with one level of dimness. This gives a crisp 80 column by 25 row display with a virtual image seeming to float in front of the wearer. The keyboard is the "Twiddler" one-handed keyboard made by HandeyKey Corporation, which uses a 17-button system where multiple keys can be struck at once to access all the symbols possible on a normal keyboard, plus extra combinations for macros. Average typing speed is about 35 words per minute using the twiddler, though Starner has been clocked at up to 60 words per minute.

(Starner 1997a)
Starner, T., *Lizzy: MIT's wearable computer design 2.0.5*. http://www.media.mit.edu/wearables/lizzy/

The wearable also includes several ways to sense the outside world. When outdoors, a GPS (Global Positioning System) is used to detect the wearer's location. When indoors, a 418 MHz AM radio receiver detects unique radio beacons that have been placed around the Media Lab **(Rhodes 1999)**. An alternate system uses IR (infrared light) instead of radio, which gives a finer control over where a beacon will be detected **(Starner 1997b)**. Beacon numbers are converted to room number via a static lookup. By putting these beacons into name badges, the wearable can also detect who is currently in the same room. This method is essentially identical to the active badge system described in **(Want 1992)**. However, because people do not generally wear these active badges, the people sensor is only used for demonstration purposes. The wearable also has a 1.2 Mbit wireless network connection that is used for communications and receiving information from sensors not directly attached to the computer.

(Rhodes 1999)
Rhodes, B., et al. Wearable Computing Meets Ubiquitous Computing: reaping the best of both worlds, in *ISWC'99*, 1999, pp. 141-149

(Starner 1997b)
Starner, T. et al, The Locust Swarm: An environmentally-powered, networkless location and messaging system, in *ISWC'97*, 1997, pp. 169-170

(Want 1992)
Want, R. et al, The Active Badge Location System, in *ACM Trans. on Info. Sys.*, 10(1):91-102, January 1992

Jimminy communicates with the sensors through Hive, a distributed agent architecture developed by Nelson Minar **(Minar 1999)**. Hive provides an easily extended platform for networking sensors and processes both on and off the wearable. A Hive cell runs on the wearable computer and polls the location beacon and active badge sensor. It coverts this information into room numbers and people, and sends this information to the main Jimminy program. One advantage of Hive is its extensibility. For

(Minar 1999)
Minar, N. et al, Hive: Distributed agents for networking things, in *Proc. of ASA/MA'99, 1999*

example, the system can trivially be extended to communicate with off-body sensors through the wireless connection.

With recent improvements in sensor technology and in the processor speed of wearable computers, it is expected that other types of sensing technology will soon become available. One such technique is ASR (Automatic Speech Recognition), which is now accurate enough that information retrieval performed on a database of raw audio news stories can be performed with over 55% precision **(Johnson 2000)**. This is close to the level of performance that would be needed to automatically generate queries for Jimminy by transcribing a person's natural conversational speech. Another technique is vision-based automatic face recognition **(Pentland 1994)**, which could be used instead of active badges to let the wearable know what other people are in the room.

(Johnson 2000)
Johnson, S.E., et al, *Spoken Document Retrieval for TREC-8 at Cambridge University*, 2000

(Pentland 1994)
Pentland, A., et al, View-based and modular eigenspaces for face recognition, in *Proc. of Computer Vision and Pattern Recognition*. 1994, pp. 84-91

4.4.1 Design Decisions

There are four main trade-offs in the Jimminy design, in addition to the design trade-offs for the RA upon which it is based:

Increase and decay of biases. When a feature of the environment changes, Jimminy automatically increases the bias for that environmental feature by a factor of three. Thus when the wearer of a Jimminy system enters a new room, notes taken in that particular room are more likely to be shown than are notes related to other features that have not changed recently. After one minute the bias is automatically decreased back to the level to which it had been previously set. This automatic decay insures that a feature will not continue to dominate all suggestions made. After a minute it is assumed that the recently changed feature is not necessarily the center of the wearer's attention anymore, and that a more balanced display of suggestions will be more useful. The value of a minute is somewhat arbitrary. In environments where the user is not likely to be able to even glance at the display for more than a minute after the environment changes the period of increased biases should be longer.

Display of environmental information. Sensors are error-prone, so it is important that the wearer of a Jimminy system be able to verify that sensor data is correct. If no sensors are used and environmental features are entered by hand, it is still useful to have a reminder of the last data entered. For these reasons Jimminy displays the wearer's current location, people in the room, and subject fields. The information is displayed in the mode-line of the Jimminy display, which usually contains information that is duplicated in other mode lines anyway. Field biases are also displayed in the mode line for similar reasons.

Design for the expert user. Jimminy is designed for the expert user. Besides the obvious expertise required to take notes and control the system using a chording keyboard, the Jimminy display also incorporates three design features for aiding the expert user:

- Key combinations (chords) are defined to raise and lower the steady-state levels for different field biases. These key combinations allow the user to quickly change biases, but the chords must be memorized.
- In the original design, biases were displayed next to their associated field in the mode line. This positioning made it easy to tell the correlation between fields and biases, but made it hard to quickly see the biases for all fields. In the current version biases are displayed next to each other on the left-hand side of the mode line

rather than next to the field they annotate. This new position makes it easier for the expert user to determine the relative values of field biases, but requires memorization of the order in which biases are displayed.

- Similarly, fields are displayed on the mode line without labeling. The order in which fields are displayed must therefore also be memorized, although it is often possible to distinguish different field types by their contents.

Design for small screen. As mentioned above, fields in Jimminy are displayed using less space fields in the RA. This is due to the small amount of screen real-estate available on the head-mounted display. Fields that extend beyond the limited space are truncated.

CHAPTER 5 *Evaluation*

Certainly, there are many things that can be learned only in closely controlled experiments. But little is known about the relationships of cognition in the captivity of the laboratory to cognition in other kinds of culturally constituted settings. The first part of the job is, therefore, a descriptive enterprise.
– *Edwin Hutchins,* Cognition in the Wild

JITIRs are situated applications: their design and effectiveness depend on the task environment in which they are applied. In terms of how JITIRs are used, the utility of a suggestion depends on the user's knowledge, immediate needs, and current levels of distraction. In terms of information retrieval, the techniques that can be applied depend on the nature of the environmental features that are available to create a query and on the structure of the corpus of information being presented. Finally, the interface must be customized to the user's task environment so attention can easily be switched between a JITIR and the user's primary task. All this integration with the task environment makes evaluation difficult, because observations in one task domain may not generalize to others.

This chapter describes three user studies. How far the results generalize to other domains depends on many environmental factors. More important than the results for a particular environment is *why* the results occur. The stories behind the data are useful for understanding JITIRs and how they can be applied to other task environments. For this reason, all three studies were concluded with informal interviews asking users for anecdotes and insights about their experiences.

The chapter starts with a description of the corpora (databases) that were used in the various user studies, followed by descriptions and discussion for each study. The first study is a controlled-task experiment that compares the usefulness of JITIRs to a traditional search engine. The second study evaluates the information retrieval used by the implemented JITIRs. It also examines how the database used by an RA affects the quality of suggestion. Third is a long-term user study that looks at use of a JITIR over the course of several months, and in some cases years.

5.1 Corpora Used

Several corpora were used by the JITIRs tested in these experiments:

INSPEC corpus: This is a subset of the INSPEC database[1] of conference proceedings and journal article citations and abstracts, pruned to be especially rich in citations that are related to research performed at the MIT Media Lab. To create the corpus, the Ovid INSPEC database was queried for all citations from January 1st 1998 through August 14st 1999 that had a Media Lab researcher as the primary author. In all, 93 citations were found. The subject headers for each citation were listed. These subject headers are assigned by the IEEE, chosen from a standard INSPEC thesaurus of subject headers.The following subject headers had been used to label at least five of the 93 articles: user interfaces, software agents, human factors, portable computers, virtual reality, interactive systems, Hidden Markov Models, image sequences, motion estimation, computer aided instruction, computer animation, computer vision, graphical user interfaces, groupware, image recognition, interactive devices, multimedia systems, social aspects of automation, and statistical analysis. The INSPEC database was then queried for all citations that had at least one of the previously mentioned "Media Lab five-star" subject headers. This produced a set of 152,860 paper citations that were especially relevant to Media Lab research. A local copy of the corpus was cached and used as the INSPEC corpus.

Media Lab email corpus: Many lab-wide email mailing lists at the MIT Media Lab are automatically archived, dating from 1988 to the present. The subset of these archives that are publicly accessable to the Media Lab community were used as the basis for the Media Lab email corpus. The database is re-indexed nightly, and as of this writing contains 209,327 email messages from 2,467 email archives. For privacy reasons, this database was only made available to the Media Lab community, and Lab members were given the opportunity to exclude any mailing lists from the database.

Agents Group email corpus: The Software Agents group is one of the working groups within the Media Lab. A special corpus was made by combining the group mailing list archive with archives from any publicly archived Media Lab mailing list that was subscribed to by more than two members of the Agents group. This database is updated nightly, and as of this writing contains 15,083 messages from 41 archives. This database was only made available to members of the Software Agents Group.

MIT Tech corpus: The Tech is MIT's oldest student newspaper, and has online archives from 1985-1986 and 1989-present. The corpus is updated as new weekly issues come out, and as of this writing contains 16,240 news articles, features and picture descriptions.

Jimminy corpus: A wearable computer has been used by the author to take over 850 notes in the past four years. All these notes are annotated with timestamp, location where the note was written, people who were in the vicinity and the subject of the note. The subjects range from project ideas and class notes to notes from conversations to dance steps.

1. http://www.ovid.com

Personalized corpora: Individual users were encouraged to create their own personalized databases from their own email archives, note files, papers or even by indexing their entire home directory. The size, quality of information and specificity of these databases varied dramatically.

5.2 Controlled Task Evaluation

The point of this first experiment was to answer the following question:

> *How does using a JITIR affect how a person seeks out and uses information?*

The cost vs. expected benefit framework described in Chapter 3.1.2 predicts that a JITIR will be used to retrieve information that would not otherwise be retrieved, as long as (1) the expected benefit of retrieving the information is lower than the expected cost and (2) the effort required to retrieve that information by other means would not be worth the expected benefit. The two-second rule described in Chapter 3.1.3 states that the level of effort beyond which a person will not bother to take action is often very low. This rule predicts that, at the low-effort end, even if a JITIR only decreases the effort required to find information by a few seconds it is likely to be used more often. In other words, reducing a ten-second search process to two seconds should have a much larger impact on usage than would a reduction from a one minute process to fifty-two seconds.

This experiment compares the information retrieval patterns of experimental group subjects given the RA, Margin Notes and a search engine versus control group subjects only given access to a search engine. All subjects were given a search engine for two reasons. First, it is important that subjects in both groups have access to the same information. Comparing a task performed with a JITIR to a task performed with no information tool is unfair, since experimental group subjects would have access to information that is completely inaccessible to the control group. Second, it is useful to compare the use of JITIRs and the search engine within the experimental group to see which is preferred and whether search engine use is decreased when a JITIR is introduced.

5.2.1 Method

Twenty-seven subjects were recruited from the MIT community of undergraduates, graduate students and alumni via email lists, posters and direct request. They were told only that they were participating in an experiment in which they would write about MIT housing. Subjects were divided randomly into experimental and control groups.

After signing the consent form subjects were given a pre-task survey asking baseline information about their knowledge on MIT housing, how much they cared about such issues, and how often they read the MIT Tech newspaper. Surveys, task description and other supporting material for this experiment are listed in Appendix A.

Experimental group subjects were provided with and given brief training on the use of Emacs, the Remembrance Agent, Margin Notes and the MIT Tech search engine web page.[2] Control group subjects were given Emacs and the MIT Tech search page only. The MIT Tech search page uses ht://Dig[3], a freely available search engine that provides the ability to search for web pages that contain all keywords listed, contain any

2. http://www-tech.mit.edu/search.html
3. http://www.htdig.org

keyword listed, or match a boolean search. The RA, Margin Notes and the Tech search page all pulled from the same MIT Tech corpus of news articles. Subjects rated themselves with an average Emacs expertise between "occasional user" and "regular user." Only one subject was more than an occasional user of the RA, and none had used Margin Notes more than occasionally.

Subjects were then given the following task:

> *Pretend you are a guest contributor for the Tech and write an editorial or news article about MIT housing. The article could cover the Freshmen On Campus decision, graduate housing, new dormitories, or any other aspect of MIT housing and how it affects student life. The article should be around a page (about 600-700 words).*
>
> *You will have up to 45 minutes to complete your article. This is not meant to rush you, but rather to put an upper bound on the duration of the experiment. If you complete the article before the 45 minutes, get the experimenter and he will continue to the next phase. If you wish, at the end of the experiment your article will be emailed to and/or printed out so you can have a copy. Your article will be compared to articles written by others in this experiment based on a number of criteria.*
>
> *You should have already received a quick tutorial with the Tech search page, Emacs RA and Margin Notes. Feel free to use these tools as much or as little as you wish in writing your article. If you have questions now or during the experiment please ask the experimenter.*

The control group task description was identical except it did not mention the RA or Margin Notes. The RA, Margin Notes and the web browser were instrumented to log the articles read and the search terms entered with the search engine.

Emacs was placed in one "virtual desktop" and the web browser was placed in another, and subjects were required to hit a function key to change between applications. Requiring users to switch screens (and thus lose visual context) is an extra burden that probably affected the usage of the search engine. However, the Tech search page takes more than half a screen to display. If both applications were present on the same virtual desktop subjects would need to either iconify or tile the applications to switch between them. Rather than leave the application-switching strategy up to individual subjects, the separate virtual desktops were chosen to force conformity. Given these limitations on screen real-estate (even on the 21" display that was used), the requirement to switch virtual desktops to access the search engine is well within the bounds of the normal usage patterns of a search engine. Note also that bringing up a full document in the RA replaces the current buffer, so the RA has an associated loss of visual context as well.

The topic of MIT housing was chosen for two reasons. First, housing is a topic most people in the MIT community know about and care about. In the past few years the MIT administration has made several controversial decisions about housing, prompting several demonstrations, sit-ins, and letters to the editor in school newspapers. Second, it is a topic often discussed in the MIT Tech. Of the 16,240 articles in the Tech corpus 1,197 (7%) contain the word "housing."

After writing the essay, subjects were given a post-task survey that asked them to rate the usefulness of the different tools used. The tools used were rated based on how distracting they were, usefulness in the task, how much the subject would want the tool when performing a similar task, how much the subject paid attention to the tool, and how often the suggestions / results displayed were useful even when the subject did not follow the result to see the full article. Subjects were also asked to rate their own expertise with the tools provided and to rate the difficulty of the task.

The essays produced in the task were coded along three criteria: overall quality, number of references to facts and number of references to Tech news articles. Coding was blinded: two coders were trained with an example and then coded all essays without knowing whether they were control or experimental group. Overall quality was defined as "good logical flow."

Fourteen control group and thirteen experimental group subjects were tested. Of these, two outliers from the control group viewed a number of articles more than 1.5 times the inner-quartile distance from the third quartile, and were eliminated. One of these two reported that he had finished early and browsed the Tech about other issues for the remaining time, the other reported she had little knowledge about housing and so wrote about "what the Tech says about housing."

It should be noted that two-thirds of the experimental subjects had never used the RA or Margin Notes before the experiment and only one had used the RA more than occasionally. A similar task with expert users would be interesting for comparison.

5.2.2 Usage Results

As can be seen from Table 3, subjects from the experimental group viewed around three times as many different Tech articles as did those in the control group. Within the experimental group, subjects viewed around two-and-a-half times as many articles using the RA as they did using the search engine. The difference between total pages viewed in the two groups and the differences between search engine and RA use within the experimental group are significant to the 0.01 level. Less than one-third of the experimental subjects viewed any documents that were suggested by Margin Notes, and even those did not view many. This is not surprising, since subjects would not even see a suggestion from Margin Notes unless they first used the search engine. For all practical purposes, the number of times a subject used the search engine was an upper bound on the number of times Margin Notes was used.

TABLE 3. Number of unique Tech articles viewed (n=12, n=13)

	Total	Search Engine	Emacs RA	Margin Notes
Control Mean	2.8±1.7	2.8 ± 1.7		
Control Median	2	2		
Experimental Mean	7.3±2.4	1.9 ± 1.0	4.9 ± 1.9	0.5 ± 0.5
Experimental Median	8	2	6	0

The number of unique articles viewed had a significant positive correlation with the subject's expertise in Emacs ($r = 0.56$, $p = 0.05$), accounting for 32% of the variance. Not enough subjects had expertise in using the RA to notice any statistically significant correlation.

5.2.3 Preference Results

Subjects who were given all three tools showed a consistent preference for the RA over the search engine and the search engine over Margin Notes. As can be seen in Table 4, seventy-seven percent of experimental subjects ranked the RA as their number one choice. (One subject ranked both Margin Notes and the RA as number one and two subjects did not use or rank the search engine or Margin Notes at all, so percentages do not add up to 100%.) Rankings are significant to the p=0.05 level.

TABLE 4. Experimental group reported ranking (n=13)

	Experimental Search Engine	Emacs RA	Margin Notes
% Subjects Who Ranked #1	23%	77%	8%
% Subjects Who Ranked #2	54%	15%	8%
% Subjects Who Ranked #3	8%	8%	69%

As can be seen in Table 5, the RA was also rated a point higher than the search engine (out of seven) for both usefulness and desirability, given the following two questions:

- How useful did you find the [tool] (1-7)?
- If you were to perform a similar task, how much would you want to have the [tool] running and available (1-7)?

The differences between usefulness of the search engine and the RA are statistically significant (p=0.05). The differences between the search engine and Margin Notes and the differences in whether the subject would want to use the system again are not significant. Errors are listed to the p=0.05 level.

TABLE 5. Ratings (n=12 control, 13 experimental)

	Control Search Engine	Experimental Search Engine	Emacs RA	Margin Notes
Found Useful (1-7)	3.17 ± 1.1	3.8 ± 1.3	5.1 ± 0.7	2.8 ± 1.1
Would Want Again (1-7)	5.5 ± 1.1	4.8 ± 1.1	6.0 ± 0.8	3.9 ± 1.2

5.2.4 Level of Distraction

It was expected that because the RA and Margin Notes are proactive they would distract the user. However, the survey results indicate that this was not the case, and in fact the search engine was considered more distracting than the RA or Margin Notes, though not significantly so. This statistic may be off due to confusion with the wording of the question. For example, some of the subjects responded that “the search engine wasn’t distracting at all, I did not use it.” Others responded that “the search engine was extremely distracting, and therefore I did not use it.”.

TABLE 6. Reported level of distraction for the different tools (1-7)

Search Engine (control)	Search Engine (experimental)	Emacs RA	Margin Notes
2.8 ± 1.1	2.6 ± 1.3	1.5 ± 1.0	1.8 ± 0.5

5.2.5 Time Taken and Article Length

Subjects were given a maximum of forty-five minutes to complete their essay, and were given a five-minute warning before the time limit. Almost all users took up till the five-minute warning or the forty-five minute mark, giving a mean time of 44 minutes to write the essay for the control group and 40 minutes for the experimental group. However, only 28% of the subjects (four control group, three experimental group) stated that they did not have enough time to adequately complete the essay, indicating that subjects were filling the time allotted even though it was not strictly necessary. There was no significant difference between the time taken by subjects in the control and experimental group.

There was also no significant difference between the length of essays produced in the control and experimental groups, and both groups had high variance. Control group essays had a mean length of 537 words (standard deviation = 204 words) and experimental group essays had a mean length of 527 (standard deviation = 253 words).

5.2.6 Differences in Essays

Finally, the essays themselves were examined and coded to see if a difference could be found between the control and experimental groups. Articles were blinded and coded for number of explicit facts mentioned, number of references to the Tech and "overall quality of logical flow." The scores given by the two coders had correlations coefficients of r=0.8, r=0.75, and r=0.45 respectively. However, individual variance among individual subjects was high and no statistically significant difference was found between the two groups.

5.2.7 Discussion

These results support both the cost vs. expected benefit framework and the two-second-rule.

Users retrieved between two and three times as many documents with the RA as with the search engine, even though search engine use did not significantly diminish when the RA was added. This result supports the cost vs. expected benefit prediction that a JITIR encourages retrieving and examining new information that would not otherwise be examined. The subjective preference, usefulness and distraction ratings further support the idea that the documents that were read were actually useful in some way.

This interpretation is also consistent with subject comments in post-task interviews. For example, two subjects in the experimental group independently commented that they "would be writing opinions, and the RA would bring up the facts to support those opinions automatically." This is in contrast to one control group subject who commented that as a matter of pride he went to the search engine to check his facts, but it was a large effort and he almost did not bother. The trade-off was best summarized by another experimental subject who commented that "the hits [from the RA] weren't as effective or relevant as the ones from the search engine, but I would never bother using the search engine." For this user, the search engine produced more focused results because he could explicitly control the query that was used, but this additional accuracy was not enough to warrant the additional effort required.

It is also interesting that such a large number of extra documents were retrieved when subjects were provided with a JITIR. The cost vs. expected benefit framework predicts these extra documents are followed either because the effort is lower with a JITIR, because the JITIR proactively provides information that was not expected to exist (increasing expected benefit), or both. Assuming at least some number of these extra documents that were followed were because of lowered effort, this result implies that there is a large advantage to decreasing tool-use time from the roughly ten sec-

onds needed to use the search engine to the one or two seconds needed to use the Remembrance Agent. In other words, it supports the two-second rule.

This interpretation is also supported by subject interviews. For example, two subjects commented that the extra effort of switching between screens (a single function key, plus a loss of context on the screen) was a barrier to using the search engine. Clearly if such small levels of effort are a barrier, then minor reductions of such barriers should have large pay-offs. On the other side, one experimental group subject never used the RA at all. He commented that he had tried to use the mouse to click on a number, but had missed (using the mouse to bring up suggested documents requires clicking on a single numeral with the mouse). He was not very familiar with Emacs, and found that typing the four characters "control-c r <number>" was too difficult to bother bringing up a document compared with the ease of typing a single function key to change screens and reach the interface with which he was familiar.

Another interesting result was in comparing user knowledge to search engine use. On average the experimental group used the search engine slightly less than did the control group, but the difference was not significant. However, the two groups differed greatly in how search-engine use correlates with knowledge about the topic of MIT housing. In the control group there was an insignificant correlation between use of the search engine and topic knowledge ($r = 0.19$). In the experimental group there was a large and significant negative correlation between number of articles retrieved with the search engine and topic knowledge, accounting for 61% of the variance ($r=-0.78$, $p=0.002$). There was no significant correlation between knowledge of the topic and use of the RA. This difference in the groups can be explained as follows.

Information use can either be data-driven or idea-driven. In a data-driven task the information that is available drives how the task is accomplished. For example, a subject might search for anything related to "housing" and write about what he finds. In idea-driven tasks information is retrieved to support a topic and logical argument that is already formed. For example, a subject might be writing about high rents in Boston and want some numbers to back up that claim.

The RA gives information that is related to what is already in a person's environment. Sometimes the relationships will be unexpected, but suggestions will rarely be completely unrelated to the environment. This dependence on local context means JITIRs are more useful for idea-driven tasks than for data-driven tasks. Given a blank page the RA cannot offer suggestions: it needs a context. Search engines, on the other hand, can provide information on topics not yet in the environment as well as support for already existing arguments. Because it is less effort to access information from the RA than to perform a query using the search engine, it is expected that the RA will compete favorably for support-type information tasks while leaving the retrieval of information on new topics to the search engine.

This theory explains the strong negative correlation found in the experimental group between search engine use and knowledge about the topic. People who knew little about MIT housing used the search engine to find information that could form the core of their essay. The RA does not help this task as well as a search engine, which is why there is only a small and stastistically insignificant drop in search engine use between the control and experimental groups. People who knew a large amount about MIT housing often had a line of reasoning for their essay before ever looking for new information; they only needed support for their existing arguments. In the control group subjects used the search engine to find support for the arguments. Subjects

using the RA, on the other hand, would often have information supporting their arguments brought up automatically, removing the need to use the search engine.

5.3 Information Retrieval Evaluation

The second experiment examines how traditional information retrieval relates to JITIRs. As described in Chapter 3.2, information retrieval algorithms are usually evaluated in terms of relevance. Queries, and thus the topics for which relevance is judged, are chosen such that the database contains information related to the given topic **(Voorhees 1999)**. It is argued in Chapter 3.2.8 that relevance is not the appropriate evaluation metric for JITIRs because even though a suggestion is relevant it may not be valuable to a user in his current situation. Instead, the evaluation metric proposed is one of utility, where utility is broadly defined as some value within a given task or environment. It is also argued (in the discussion of *priors* in Chapter 3.2.6) that JITIRs cannot rely on the database used being a good match for the user's particular task, and that the choice of database can make a large difference in the utility received from a JITIR.

(Voorhees 1999)
Voorhees, E. and D. Harman, Overview of TREC7, *NIST Special Publication 500-242*, 1999

The primary goal of this experiment is to show the following:

> *The relevance and utility of a suggestion are correlated, but only loosely.*
>
> *The match between the corpus, the task environment, and the algorithms used affects the utility of suggestions.*

The experiment evaluates the relevance and usefulness of information retrieved and suggested by Margin Notes within the context of a particular task environment (writing or re-reading research papers) and corpus (the INSPEC corpus and the Media Lab email corpus). The relevance and usefulness scores give indications of the quality of suggestions produced by a JITIR, at least within this task environment with this corpus. However, the primary goal of the experiment is to show that the main mode of evaluation for traditional information retrieval (i.e. relevance given queries that are hand-chosen for a particular corpus) are not good enough to predict the utility of a JITIR.

It should be noted that this set of experiments are evaluating relevance and utility within the confines of a single task, namely how suggestions might be useful to a researcher who is writing or re-reading a paper she recently wrote. Other tasks could be chosen, such as reading or reviewing a paper written by someone else or writing email. Other corpora and IR algorithms could also be used instead of the ones tested here. The goal is not to show that the results obtained here are generally applicable, but rather to show that the methodology used to evaluate traditional IR cannot be used reliably for evaluating JITIRs. For more generalizable results, Section 5.4 describes the long-term user experiments that have been performed.

5.3.1 Method

Media Lab researchers were asked to submit conference papers and articles they had converted to HTML and placed on the web. Two copies of each paper were annotated by Margin Notes, one using the INSPEC corpus and one using the Media Lab email corpus. These two corpora are close to the same size: at the time of the experiment the INSPEC corpus was 152,860 documents and the Media Lab email corpus was 183,125 documents. Both are also customized for a Media-Lab audience. However, the INSPEC database is a better match for annotating research papers. The documents in the INSPEC database are also of higher quality (provide more trustworthy or useful information) on average than the email archives. The two corpora were compared in

this experiment to gain insight into how much the quality of database affects the overall performance of a JITIR.

For this experiment, all sections were annotated regardless of whether the relevance passed a minimum threshold. A printout of both copies of the annotated paper were given to the author. In the interactive version the keywords for a section are only shown on mouse-over, but in the printout all keywords were included. Also, relevance scores for each section were blanked out so authors would not be unduly biased. With each annotated paper a packet was included that contained printouts of each citation or email suggested by Margin Notes, along with the following questions for each annotation:

1. How relevant was this citation to your paper in general (1-5)?
2. How relevant was this citation to the specific section it annotates (1-5)?
3. How useful would this citation be if you saw it while writing or re-reading your paper (1-5)?
4. For what reasons might this citation not be useful (circle any/all that apply)?
    - The cited work is not relevant enough
    - The cited work is low-quality
    - I already knew about the cited work
    - I don't need any more references for this section
    - Other:
5. How well do the title, authors and date part of the margin note indicate the usefulness of the citation (1-5)?
6. How well does the entire margin note (title, authors, and date + keywords) indicate the usefulness of the citation (1-5)?
7. Other comments on this citation / annotation?

For the email packet, the word "citation" was replaced with "email message."

(Budzik 1999)
Budzik, J., and K. Hammond. Watson: Anticipating and Contextualizing Information Needs. In *Proc. of the 62nd Annual Meeting of ASIS*, 1999.

Nine researchers were asked to evaluate the annotations from the INSPEC database on their papers, for a total of 112 specific annotations. Seven of those researchers also turned in evaluations from the Email database, for a total of 76 annotations. (The email survey was added in the middle of the experimental run, and two of the researchers had already left the Media Lab and could not be reached). This experimental protocol is similar to (and influenced by) the protocol used by Budzik and Hammond for evaluating the Watson system **(Budzik 1999)**.

Note that this methodology can produce two kinds of sampling errors. First, only Media Lab researchers and papers were selected, which may not be representative of a larger population. Second, because several sections were taken from the same papers the individual sections are not completely independent.

5.3.2 INSPEC Relevance vs. Usefulness Results

In general the INSPEC annotations were rated highly for relevance. As can be seen in Table 7, the average score was 3.3 out of 5 for relevance to the paper in general and 3.4 out of 5 for relevance to a specific section. More importantly, around half the

annotations received a score of four or five for relevance to the annotated section and the paper in general. All errors are shown at the p=0.05 level.

TABLE 7. INSPEC annotation rating breakdown (5 = best)

	General Relevance	Section Relevance	Usefulness
Score = 1	16%	21%	32%
Score = 2	16%	9%	18%
Score = 3	21%	16%	15%
Score = 4	17%	19%	18%
Score = 5	30%	35%	17%
Average Score	3.3 ± 0.3	3.4 ± 0.3	2.7 ± 0.3
%Score = 4 or 5	47 ± 9 %	54 ± 9 %	35 ± 9 %

The results in Table 7 are for all annotations, regardless of whether they received a high enough relevance score from Margin Notes that they would be displayed in the real system. If annotations are ranked according to the relevance score generated by Savant and the top 20% of annotations are compiled, the result is a much better averages for relevance, but not for usefulness. These results are shown in Table 8.

TABLE 8. INSPEC "best 20%" annotation rating breakdown (5 = best)

	General Relevance	Section Relevance	Usefulness
Score = 1	14%	19%	38%
Score = 2	10%	0%	14%
Score = 3	5%	10%	10%
Score = 4	19%	5%	14%
Score = 5	52%	67%	24%
Average Score	3.9 ± 0.7	4.0 ± 0.7	2.7 ± 0.7
%Score = 4 or 5	82 ± 16 %	77 ± 18 %	36 ± 21 %

In both the top 20% of annotations and the full set, usefulness scores were not as good as relevance scores. The average usefulness score was only 2.7, with only a third of the annotations receiving a 4 or 5. As can be seen in Table 9, relevance was almost always rated greater than or equal to usefulness. This one-sided dependence is a clear indication that, at least for this task, relevance was a necessary but not sufficient condition for usefulness.

TABLE 9. Difference between Relevance and Usefulness

Difference	% Citations with this Gen. Rel. minus Usefulness	% Citations with this Sec. Rel. minus Usefulness
-4	0	0
-3	0	0
-2	4	5
-1	13	13
0	46	36
1	20	27
2	4	6
3	4	3
4	11	11

The reasons given for why a citation might not be relevant are listed in Table 10. Note that more than one answer might be given for a particular citation.

TABLE 10. Reasons citations might not be useful

Reason	% Citations for which reason was given
Not relevant enough	42
Already knew about citation	29
Low quality	12
Citation is own paper	10
Don’t need more references	7
Other	4

Readers were also asked whether the annotation description was a good indication of whether an annotation would be useful. As mentioned in the discussion of Ramping Interfaces in Chapter 3.3.5, the hope is that bad annotations can be recognized and ignored with minimum distraction while still alerting the reader to useful annotations.

As can be seen in Table 11, the information contained in the initial annotation was usually enough to determine whether a suggestion would be useful or not. The addition of keywords improved those results.

TABLE 11. INSPEC suggestion helpfulness (5 = best)

	Description Only	Description + Keywords
Score = 1	2%	1%
Score = 2	10%	9%
Score = 3	21%	14%
Score = 4	33%	26%
Score = 5	35%	50%
Average Score	3.9 ± 0.2	4.2 ± 0.2

5.3.3 INSPEC Results Discussion

As indicated in Table 9, relevance was necessary but not sufficient for an annotation to be useful. This result was partially due to the constraints of the task environment and the phrasing of the question: "How useful would this citation be if you saw it while writing or re-reading your paper?" The discussion on the long-term user study will show broader criteria for usefulness. However, it is still reasonable to expect that suggestions not relevant at all to the current environment will be no more useful than any other randomly selected document from the corpus.

Of particular interest is the reasons given in Table 10 for the 14% of citations that were rated low on usefulness (1 or 2) but high on general relevance (4 or 5). Of these citations, 100% were noted as being not useful because the citation was already known by the person doing the rating. Moreover, 68% were not only known, but were in fact written by the person doing the rating. One might say these documents were *too* relevant to be useful.

These highly relevant but not useful documents came in two categories. Most were not useful because they were already known. While it would be useful to eliminate these suggestions through some form of user feedback or perhaps a user profile, they are not bad suggestions on their face. In particular, they might be quite useful to a user who has less knowledge about the paper being annotated, i.e. someone who does not know about the suggested citation already. The second category of documents were those that were so similar that no new information was presented. For example, one suggestion was for the citation to the very paper being annotated. This sort of error occurs more frequently outside of this particular task domain. For example, if a user is browsing old emails it is almost certain that the first suggestion is for the particular mail being viewed. No new information is provided by this kind of suggestion. To avoid this kind of problem Margin Notes attempts to detect and not show documents that are almost identical to the current environment, but the detection cannot be perfect because documents can still differ from the current environment in only trivial ways.

Finally, it should be noted that just because a document is known does not mean it is useless. Of the thirty-three documents that were listed as possibly not being useful because they were already known, 20% were still given a usefulness score of four or five. One subject mentioned that, even though she knew about the citations, they were still useful reminders.

There are also other reasons a suggestion might not be useful aside from relevance or already being known. For example, 12% of the suggestions were labeled “low quality.” Quality, of course, is a subjective judgement. For example, one citation was labeled as low quality because the subject matter was a topic that had been explicitly discounted by the author: it discussed formalisms for his field when his paper explicitly ignored formal approaches. Finally, it is possible that no suggestion will be useful simply because the user does not need or want any new information in the current task environment.

5.3.4 Corpus and Environment Differences: Media-Lab Email Corpus Results

The INSPEC corpus is in many ways the perfect match for annotating research papers. It is a good fit for the task, it is of narrow focus and citations have all been through a peer-review process to insure quality. To gain some insight into the importance of a good database, the same experiment was conducted with the Media Lab Email corpus. While still personalized for the Media Lab research community, the email corpus is not as close a match for research papers, its focus is quite broad and the quality of email content varies widely.

As can be seen in Table 12, relevance and usefulness scores were on average 0.75 (out of five) lower than for the INSPEC database. Several readers also commented that Email suggestions were far less useful than the INSPEC suggestions.

TABLE 12. Media Lab Email annotation rating breakdown (5 = best)

	General Relevance	Section Relevance	Usefulness
Score = 1	42%	33%	58%
Score = 2	16%	12%	21%
Score = 3	11%	17%	16%
Score = 4	8%	14%	1%
Score = 5	24%	24%	4%
Average Score	2.6 ± 0.4	2.8 ± 0.4	1.7 ± 0.2
INSPEC Avg (from Table 7)	3.3 ± 0.3	3.4 ± 0.3	2.7 ± 0.3

Differences between the two databases are all significant at least to the p=0.05 level.

As can be seen in Table 13, the reasons given for why email might not be useful are similar to the reasons given for the INSPEC database, except reasons were given for a larger percentage of documents.

TABLE 13. Reasons email might not be useful vs. INSPEC

Reason	% Email reason given	% INSPEC Citations (from Table 10)
Not relevant enough	55	42
Already knew information	36	29
Low quality	18	12
Email is from author	4	10
Don't need more references	4	7
Other	4	4

Differences were less pronounced between email and INSPEC for how well the annotations indicated whether the document would be useful. The difference between descriptions only are significant to p=0.02, differences between description and keyword together are not significant.

TABLE 14. Media Lab Email suggestion helpfulness (5 = best)

	Description Only	Description + Keywords
Score = 1	12%	0%
Score = 2	13%	9%
Score = 3	16%	13%
Score = 4	35%	44%
Score = 5	24%	33%
Average Score	3.4 ± 0.3	4.0 ± 0.2
INSPEC Avg (from Table 11)	3.9 ± 0.2	4.2 ± 0.2

5.3.5 Discussion on Corpus Differences

There are several reasons the Email corpus produces lower quality results than the INSPEC corpus. Most important is the suggestions on average are lower relevance. Lower relevance can be due to two problems. First, the INSPEC database is focused specifically on research topics. The email corpus has a much broader range of topics, and therefore the percentage of documents that might be relevant to a particular paper is probably lower. Second, the emails tend to have a wide range in terms of document length and focus of subject. These can both cause difficulties for the particular text retrieval algorithm being used. For example, one particularly long (fifty page) email was given as a suggestion for several papers, and was always given a low usefulness and relevancy rating. The paper was incorrectly chosen because the particular text similarity metric used gives too much weight to long documents. The similarity metric can be fixed by applying better normalization techniques such as those described in **(Singhal 1996)**, but the larger point is that one corpus (the INSPEC corpus, with relatively fixed-length documents) is a better match for the particular algorithm used than another corpus. The problem emphasizes the need for using an adaptable framework such as Savant, where algorithms can be picked on the fly depending on the environment.

(Singhal 1996)
Singhal, A. et al, Pivoted Document Length Normalization, in *Proceedings of SIGIR'96*, 1996, pp. 21-29

Another comment was that it is harder to judge whether an email is high quality. Most INSPEC citations have gone through a peer-review process, insuring a minimum level of quality. Furthermore, the conference or journal in which a citation is published is usually a good indication of quality. Email lacks both these indicators, leading at least one subject to comment that it was difficult to tell whether the information in an email was accurate.

Finally, INSPEC citations include an abstract that is intended to contextualize a paper. The abstract makes citations easy to read in isolation and still understand what they are about. Email documents are often written in the context of a larger conversational thread that is missing when a single email is read in isolation.

5.4 Long-Term User Studies

The third and final experiment looks at the use of the Remembrance Agent, Margin Notes, and Jimminy "in the wild," that is in unconstrained natural use over the course of many months.The systems logged usage patterns, but more important and generalizable are the stories told during informal interviews. The primary goals of this experiment are to at least partially answer the following questions:

What kind of value do people get from a JITIR?

What issues are important for designing a JITIR?

5.4.1 Method

Various versions of the Remembrance Agent has been available for free download on the Web since June 1996, and Margin Notes has been available for use within the Media Lab since July 1997. Since that time Margin Notes has undergone one complete rewrite, the RA has undergone two. In the first four months of 2000, five hundred and four separate machines downloaded the Remembrance Agent for an average of four unique downloads per day. Three hundred forty-five people are currently on the announce list for future releases.

Users were interviewed over email and in person. There was no standard set of questions; they were asked simply to tell stories about their experiences good and bad. Unsolicited stories have also come up in casual conversation with users. The log files for six users of the RA were also analyzed.

5.4.2 Long-term Log-file Results

The log files analyzed here are from six users of the RA, three from the Media Lab and three from the Internet at large. All were using their own combination of personal databases. The logs spanned from three to seven months of usage data. Note that users in this study were self-selecting, in that only logs from people who had used the RA for a long period of time were used. Furthermore, because the sample size is small these log-file results are not statistically significant and should be taken as anecdotal.

The total amount of time logged for all six users was 740 calendar days, 312 days in which the RA was actively being used. In this time period 186,480 suggestions were displayed (including duplicates), 197 (0.1%) of which were followed after being automatically suggested based on the user's local context. Based on this data, users looked at approximately one suggestion every four calendar days, or about two suggestions per week. Note that this time span includes weekends, vacation, and potentially even full months where the RA is never turned on. Counting only days where the user was using the RA at all, a user viewed a document every 1.5 days, translating to between four and five times a week.

When users follow a suggestion they are asked to rate the value of the document suggested from one through five, five being the best. However, rating is not mandatory, as forcing users to rate documents would add an extra effort to a process which is already highly effort-sensitive. As expected, this led to only 25% of the suggestions that were actually followed being rated (49 out of 197), so there may be a significant sample-bias. Average rating was 3.1 out of five, with an almost flat distribution across all scores.

5.4.3 Interview Results

In all, 104 stories have been collected from 34 separate users of the RA, Margin Notes and Jimminy. Stories come from 23 long-term users of one or more of the systems (eight from the MIT community, fourteen who downloaded the system from the Internet, plus the author and one other collaborator), seven subjects from the first (essay) experiment and three subjects from the second (IR) experiment. Three of the interviewees plus the author are wearable computer users who have used either the RA or Jimminy on the wearable computer. These wearable users include Thad Starner, the developer of the *Lizzy* wearable computer and a collaborator on the early versions of the RA. Some of these stories come from the author's research diary; these stories will be indicated as such.

The stories and comments have been divided into four main areas: the *value* of using a JITIR, issues regarding the *design* of JITIRs, *other issues* including privacy and the use of a JITIR in a larger community setting, and *future directions and desired features*. The first three areas are discussed below; future directions are discussed in Chapter 7. While not all the stories collected are discussed, the sampling shown gives the flavor and range of user experiences.

5.4.3.1 Value of a JITIR

As discussed in Chapter 3.2.8, there are many different ways in which the information provided by a JITIR can be of value to a person. These values can be organized in different ways, but one split is according to whether the information supplied is used to start a new action, to support the task already being performed, to contextualize the task already being performed, or to entertain. The distinctions between these four groupings are not exact and some cases may fall into more than one category, but they give an indication of the range of possible uses a JITIR can have.

Information that prompts a shift in the task being performed. Suggestions in this category are useful because they provide information that changes the task a user is performing in some major way. One example comes from a collaborator who was writing a proposal while using Radar **(Crabtree 1998)**, which is the Microsoft Word version of the RA developed in conjunction with British Telecom:

> *[The proposal] was more or less finished and I was editing it. Almost wherever I went in this proposal Radar kept suggesting an email a colleague had sent me that I kept ignoring. I didn't associate this guy with the work that was being proposed, but as it kept bringing it up I thought I had better humour it (imagine humouring a piece of code for God's sake!!!) It was a good thing I looked, because he had attached a similar proposal from someone else in [the company] that he thought I might be interested in. If I hadn't looked at it and referenced this other proposal it would have made mine look pretty silly!*

(Crabtree 1998)
Crabtree, I.B. et al. Adaptive Personal Agents, in *Personal Technologies*, 2(3) 141-151, 1998

The author's research diary logs a similar experience using the RA with email:

> *When I was the MIT Ballroom Dance Club publicity chair, someone wrote me asking for our fall-term schedule. I was busy and would have put him off, but it turns out this was the second time he had written to ask. He didn't say so and I didn't remember, but his first request from two weeks previous came up. So I wrote back immediately, saying "Sorry I didn't get back to you sooner...."*

In another example from the research diary, Margin Notes running on the INSPEC database produced the following result:

> *I was looking at the "Shopper's Eye" project web page from Anderson Consulting and Margin Notes came up with a citation for it. Everyone in the group had thought the work wasn't published. So I sent the citation to the group mailing list. While writing the email, the RA (running with my personal email archives) brought up an email that said one of the group members had been at the conference where the work had been published. So I also asked in the email if he had heard their talk.*

In all three cases there was no reason to expect that useful information was available, and in fact the user did not even know he had an information need. By providing unexpectedly useful information, the JITIR both increased the expected benefit of reading the full text of the information provided and reduced the effort of doing so.

The RA can also provide information where there is a known information need but the user does not know that information is available. For example, several users have reported that they would be in the process of asking a question in email, only to have the RA answer their question for them. For example, one of the Media Lab users sent this mail to the software agents group mailing list:

> *Anyone know offhand who anonymizer.com's competitors are? I also seem to remember that someone (Bell Labs?) was running an experimental Nym service where you'd have one ID on their site and they'd create nyms for you to use on other Web sites... Ah, the RA tells me it was Lucent, the URL was http://lpwa.com:8000/ which now maps to Proxymate.com. Any other leads?*

In the preceding example, the sender of the email found the answer while writing the question, then changed the email to ask a follow-up question. This pattern is also evident in other emails, e.g. this email from the author:

> *I think I asked this before, but where's the twiddler X-driver... oh, wait – my RA just told me. Thanks anyway ;-). So when's it going to be pointed to from the wearables page?*

There have also been several reported cases where someone has asked an RA user a question via email and the RA brought up the answer. In some cases the user of the RA would have answered the question regardless, but the RA reduced the effort required to find the answer. For example, this entry from the research diary:

> *My roommate sent me email saying: "Hi Brad, Joanie mentioned some party she's having tonight, and said I could get the info from you. Are you going?" The first hit from the RA was Joanie's invitation, which I would have had to dig up for him otherwise.*

Other times the question was sent to a wide mailing list rather than to an individual. In these cases there is less social pressure to respond to the question, and thus the benefit of finding an answer is lower than it would be for answering a direct question. For example, at one point a user sent email to the Media Lab main mailing list asking if anyone could recommend a good dentist in the area. In response to the email, the author's RA brought up a message sent to a social mailing list a year and a half earlier, recommending a particular dentist a few blocks away from MIT. Because the question was not directed to anyone in particular and because the author did not remember that recommendations for a dentist were in his personal email archives he would not have bothered to search the archives. However, the RA both increased the expected benefit of retrieving the full document and reduced the cost of retrieving the information. He therefore forwarded the recommendation to the Media Lab list.

It is also possible to deliberately use the RA as a brainstorming tool. One of the subjects in the first experiment described his experience writing his essay as follows:

> *It's almost an unfair advantage to have the RA. I just started writing down words to brainstorm for the essay, and read the things it brought up.*

Similarly, one of the Media Lab users described his use of the RA as a brainstorming tool like so:

> *...it is also interesting to write poetry with the RA on. I get a few interesting hits that way too. Usually mail pertaining to my girlfriend or my ex-girlfriends...*

In the last two examples a specific relevance to the task at hand is less important, because the task itself is less defined. In such cases a search engine will not necessarily be useful because it is not even clear what the topic of a search should be.

Supporting material. As discussed in the controlled-task experiment in Section 5.2, often a JITIR is used to bring up supporting material for arguments that are currently being made. For example, when people were writing about MIT housing the RA would bring up Tech articles that backed up the claims being made. This class of suggestion does not drastically change the task being performed, but does augment the way the task is being performed in some way. For example, the research diary logs the following experience when using the RA on personal email:

> *While writing email about MP3 piracy and the lawsuits against Napster, I was arguing that a lot of lawsuits were trying to stop legal activity. The RA popped up the 1996 story about C2Net getting sued by the Software Publishers Association for not signing their "code of conduct." It was a great example, and I had forgotten all about it. After reading the email, I was wondering what happened to the*

> *lawsuit. The third hit down was an ACLU notice from two months later that said the suit had been dropped after the negative press it produced, but the SPA reserved the right to refile it. I included the example to help support my argument.*

In this case the example was used to add to an existing task, namely to argue a particular point. The RA can also provide supporting material that would be retrieved by other means regardless. Another example from the same discussion on copyright demonstrates this point:

> *I was writing [to a mailing list] about copyright and I mentioned Pamela Samuelson's article about how copyright law is being used for censorship. I wanted to include a URL to the article, so I watched the display as I typed until I had finished the sentence, then her article popped up. From there I got the URL.*

In this example the ultimate outcome is not changed (the URL would have been found regardless), but the RA saved time and effort.

In another form of support, several users run the RA over Linux HOWTO files, technical manuals and reference pages. Often they find the particular information they need is one of the suggestions being displayed. In these cases the existence of supporting material is often known, but it isn't worth a large amount of effort to retrieve that information. The JITIR lowers the cost of retrieving the information to acceptable levels.

Information that contextualizes the current environment. Even if the information provided does not directly affect how the current task is performed it can still help contextualize the current environment. For example, one user discussed how he used the RA while writing a class paper. The RA suggested email that was sent to the class list in previous years. While he did not use any of the suggested information directly in his paper, he said it was "reassuring to see that people were using technical terms in the same way I was." In other words, the RA helped contextualize his paper in the broader framework of students who had already taken the class.

Another example of contextualization is when large projects have many pieces that interconnect. In these cases, the RA can help organize the different pieces of information. For example, Thad Starner uses the RA on his wearable to order his notes files. When writing a note, if the RA suggests a related file he will combine the current and related file so that related notes are kept together. Other users have commented that they use the RA for writing hypertext documents, and often will link suggestions into the document as contextualizing or supporting material. A user who downloaded the RA from the Internet had a similar usage pattern:

> *I'm a sociology Ph.D. student doing lots of interviewing and the RA is very useful for suggesting links between different interview transcripts. Also, I use it to suggest other relevant texts I have written, theoretical papers I have written, research notes, and so forth.*

Another example came from a user of Radar (the Microsoft Word version of the RA built in collaboration with British Telecom):

> *I was updating my CV [and] most of the emails Radar came back with were job advert postings. That was pretty spooky!*

In this last case the user was not actively looking for a job, but the suggestions still put his CV into the larger context of jobs that were available.

As a final example, often people doing technical support via email will get repeat customers; clients who report a bug or a problem and then six months later write back again with a new problem. Several users have reported that the RA has brought up previous email from clients writing with problems. Sometimes these emails give useful information like the kind of hardware or configuration the client has. Other times the new problem is unrelated to the old, but the suggestion still gives the contextual information that this is not a brand new client needing assistance but is a repeat visitor. The established relationship has implications for the tone of response, or perhaps a follow-up on the old problem will be added to the end of the new response.

Value for other tasks and entertainment value. There have been many cases where the suggestion provided by the RA or Margin Notes is not useful or even relevant to a user's *current* task, but is valuable for other reasons. This class of suggestion is illustrated by this recent example from the research diary:

> *I was writing to my mom and dad about my thesis defense, and mentioned that Marti Hearst showed up. As I was describing who she was, a suggestion [from the RA] came up for an announcement of a talk she gave at Stanford in 1997 (I'm still on some Stanford mailing lists). The information wasn't useful for writing the email to Mom and Dad because they don't need that kind of detail, but the announcement was useful to me because it talked about some of Hearst's research that I didn't know about, but should have.*

In this example, the suggestion was not useful for the current task, but was useful in a completely different task (namely, as background for this thesis). Such relevance to a different task is not completely by chance, because events in a person's life are not completely disconnected from each other. In this case, the email being written was about a person who attended the author's thesis defense, so it is reasonable to expect that suggestions might be related to the research that was being defended.

There area also many examples where a suggestion is not useful for the current task, but is valuable because it is *entertaining*. For example, one user excitedly reported:

> *Your Ph.D. thesis told me to send a list of "Top 10 Things About Thanksgiving That Sound Dirty But Aren't" to my friend. So I did.*

While writing a friendly letter the RA had suggested a humor piece that he had received in email some time before. The subject was not directly related to the letter he was writing. However, the tone of the humour piece fit the tone of the email he was currently writing, so he included it in his message. In this case the suggestion was not necessarily useful to solving a task, and it is not even clear what kind of information

could be considered “useful” when writing a friendly letter. But it was definitely entertaining, and he was glad it had made the suggestion.

In a related case, the author was browsing the web looking for a distraction from work. He came across a joke page, which was annotated by Margin Notes with several more joke emails. While not exactly related to the page except in basic style, the jokes still fit the tone of what he was trying to do (find distractions) and were useful in the context of this larger goal.

Another user described the RA as entertaining as follows:

> *I write a lot of email to my girlfriend (obviously). However I find it immensely entertaining that the RA brings up emails from my ex-girlfriends... and I do go through the emails every once in a while just to see what my ex was talking about or what I was talking about to my ex.*

Again, the suggestions are not useful to any particular task being performed, but they are entertaining. Finally, the research diary mentions the following experience:

> *I was writing email to Alan, and one of the lower-down suggestions had the subject “quote of the day.” I could tell it had nothing to do with my email because of the keywords – it was only suggested because it was email from Alan and it had the word “quote” in it (I was asking if I could quote him in my thesis). But I thought it might be cute to read so I did. It was a funny quote and I was glad I saw it. A random-number generator could probably have done just as good a suggestion.*

In this last case, the suggestion was completely irrelevant to the task and it was clear from the keywords that it was irrelevant. And yet the resulting document was judged to be valuable. This example also shows how a good interface can make up for mistakes in the information retrieval process, even to the point of occasional irrelevant suggestions still being valuable.

5.4.3.2 Design Decisions

The need for filtering. Early testing of the RA and Margin Notes showed the need for domain-specific filtering of non-useful information. In particular, the keywords associated with emails would often match based on words like “forwarded from” and other header information. Also, when reading email the RA would often suggest documents based on the email’s signature line rather than based on the body of the message. HTML documents had similar difficulties with different kinds of tags, as seen in this research journal entry:

> *So here’s a bizarre hit – I was looking at some old ragtime dance moves on the wearable, and the RA was convinced that it was similar to a couple of web pages from my Discourse class. Why, you ask? Well, the dance moves file has all sorts of things like “grape-vine left, release left hand, man turns 180, etc.” The discourse file is HTML, and the relevant part for the RA was things like “<td align=”left”>Oct. 24<td align=”left”>Intonation and Speech Synthesis” etc.*

From these two examples and other like them it became clear that some parts of documents were not useful, but that the detection of those useless components varied depending on the document type. This revelation led to the document-type and query-type dependent filtering that is now in the template component of Savant.

Accessibility of fields in display. The displays for the RA, Margin Notes and Jimminy are all designed to make it simple to scan one or several fields without necessarily reading the whole display. Some of the trade-offs discussed in Chapter 4 include the display of many fields of information (even at the cost of truncating some of the longer fields) and displaying information in fixed-width fields in the RA and Jimminy so the values for a particular field type form a single easily scannable column. The importance of this scanability is shown in this research journal entry:

> *Someone asked [on a mailing list] if anyone knows of a good technology for printing road maps. I'm busy so I'm not even reading whole emails like this; I'm just scanning them. I wonder to myself if I have something in my files about his question, and peak down at the top RA hit. The subject [of the suggestion] is "Mapblast" but the date is from 1996 so I figure it's probably not all that relevant to today's technology. I don't even bother looking at the other hits in the list or other fields; I just file his email without answering it and go on.*

In this example, the *subject* field indicates what might be a useful suggestion, but the *date* field offers more information that reduces the probability that the suggestion is useful. Because both fields were easily scannable the user was able to quickly make an assessment about whether to look at the suggestion further, and in this case he did not. The total elapsed time was no more than a couple of seconds.

Users have also commented that certain fields are more important than other fields for determining the value of a suggestion, but that the details of this importance are dependent on the kind of information being displayed. For example, one of the subjects in the information-retrieval experiment commented that when Margin Notes truncated the title of a paper or journal article it was difficult to tell what a citation was about without retrieving the entire document. Truncation was not as much of a problem for the email database because subject lines were shorter. This observation led to the database-specific formatting of field-length.

Two-second rule. JITIRs are designed with the notion that evaluating and accessing information should be as trivial as possible, and that when an information task takes a small amount of effort already (on the order of a few seconds), small additional reductions in effort will have large effects. JITIRs provide information that is not useful enough to retrieve by other means, or at least are not expected to be useful enough to retrieve.

Several examples support this notion. First, Thad Starner has commented that he wears the *Twiddler* (the wearable computer's one-handed keyboard) on his hand much of the time, even when not actively taking notes:

> *The extra overhead of putting down your coke can to get the Twiddler is too much – you won't do it.*

One of the subjects in the essay experiment made a similar comment about using the RA versus using a normal search engine:

> *Hits weren't as effective or relevant as when using the search engine, but I'd never use the search engine.*

In both these cases the extra effort being discussed is quite small, on the order a few seconds. And yet that small difference tends to have a large effect on usage patterns. In the reverse direction, one of the subjects from the first experiment commented that he did not use the RA at all and only used the search engine, because he found it too difficult to click on the line number (a single-character wide target) and typing *control-c r* and the number was "too many keystrokes compared to just using the search engine." Here again the issue of only a few keystrokes (plus a less well-known interface) has a large effect on his usage pattern.

Finally, from long-term use of Jimminy it has become apparent that it is quite difficult to bring up and read or even skim an entire document while in a conversation without appearing distracted. Four main techniques have been developed by the wearables users for these situations:

- Don't bring up full documents at all, but use the suggestion itself to jog normal memory.
- Don't bring up full documents until attention is on another participant in the conversation. This technique usually entails noticing a suggestion and then waiting for an appropriate pause in the conversation to actually retrieve and scan the document.
- Start a sentence with filler, for example "the answer to the question is..." and while that is being said bring up and quickly scan the rest of the document for the answer. This is a somewhat risky move. Starner describes one occasion where he started to answer a question in class ("what is deixis?") while looking up the answer, but mistyped a key during retrieval. He wound up saying "Deixis is when a person... uh, just a second...." The class laughed, but also realized that Starner had been using the wearable other times in the past when he *had* found the answer by the end of the sentence.
- Make a comment that lets other participants know why you are distracted, such as "just a second, let me look that up."

Interestingly, these techniques are not as important when in a conversation with people who already have a good mental model of what a wearable computer does and how the wearer is using it. In these cases, participants can notice non-verbal cues when a wearable user types on the chording keyboard or moves his eyes to the display. The participant then usually pauses while the wearable user retrieves or writes a note, then continues as if nothing had happened. Such effects are quite normal in other social situations where the mental model of everyone in the group is well known. For example, professors will frequently pause during a lecture to allow students to write down notes. Such pauses are a normal part of the non-verbal discourse.

Choosing the right database for the task. Several users have commented that the database they use has a large effect on whether the JITIR is valuable. For example, one user of the RA commented:

> *[The] RA is very promising technology, and I suspect that I would find it more useful if I had a more homogenous database, i.e., I was usually writing papers, and I had a database of papers indexed. However, when I was using*

> *[the] RA, I was doing student journalism, doing some web development, writing papers for classes, and composing/ reading e-mails. The student journalism material had little relevance to the other categories of data, for example. I suppose I may have just needed a larger database.*

The problem described here is that the class of documents indexed only sometimes matches the user's task. The user suggests a larger database might help, and to some extent it will because then at least documents relevant to the task will be somewhere within the corpus. However, as described in Chapter 3.2.6, this solution both causes a corpus to be diluted and adds additional CPU and memory requirements to the retrieval engine. A better solution is for the system to automatically recognize at some high level of abstraction what kind of task a user is performing and choose the appropriate database accordingly. Based on the user comment above and others like it, the RA now can automatically select different databases based on the user's major-mode (e.g. mail mode, LaTeX mode, or net-news mode). Emacs can automatically set the user's major-mode based on the file-name extension or the header of the file being edited, or the user can set his major-mode by hand.

Note that this feature of the RA does not solve the much harder problem of completely understanding a user's context and choosing the appropriate database. For example, the RA still cannot detect that a user is writing an article for the school newspaper unless he has defined a specific mode for this task. However, the choosing of a database based on the kind of editing the user is performing does allow the user to define some basic heuristics that can help the RA choose the proper database to display.

Even when a JITIR cannot itself determine what database will be useful, the user can change to a different database manually. Like the rest of the design heuristics associated with JITIRs, the easier it is to change databases the more likely it is that the user will do so. This example comes from the author's research diary, from a time when he was running the RA on his personal email database:

> *I was writing email to Thad asking about stories [of his use of the RA]. I realized I might have something in my Jimminy-notes database, so I switched databases to see. Sure enough, I followed two suggestions for notes taken while talking with Thad about the ease-of-access aspects of wearables. One of them was something I could use for the thesis.*

The RA makes it easy to change databases, and also allows the user to display multiple databases simultaneously. This ease of switching encourages checking suggestions from multiple databases. Margin Notes requires changing the proxy server port number to change databases, which in Netscape requires clicking through five layers of menus and changing a number, plus reloading of the page being annotated. This extra effort means users of Margin Notes tend to change databases less often.

Long-term use of Jimminy has shown that the importance of different features for a particular database will also vary depending on the user's task and environment. For example, when going to conferences the *person* field of a notation is often useful, because it brings up information about people who have not been seen for a year. However, in the author's day-to-day use of Jimminy neither the *person* nor the *location* field are particularly useful, because the author is a graduate student who never leaves the lab. This means that the same group of people and locations are constantly

reoccurring in many situations, and therefore there is not a strong mapping between the location of an event or the people who are around and the particular topic of the event. The text notes of notes contained in the *body* field, on the other hand, are still a good indicator of the topic of a particular note, and tends to produce more useful results. Savant uses the inverse document frequency of the *person* and *location* fields to help alleviate this problem; locations and people that are associated with a large number of notes are automatically given less weight.

5.4.3.3 Other Issues

Several other issues have been raised in interviews, including privacy issues, the use of JITIRs in a community environment, the attribution of a personality to a JITIR, and the fact that the secondary task nature of a JITIR can lead to user's forgetting to start the program.

Privacy. When a JITIR is used to retrieve personal information (e.g. personal email archives) or published information (e.g. citations from INSPEC or articles from *The Boston Globe*) there are few privacy concerns. However, several users expressed concern when databases were drawn from private or semi-public corpora. Clearly there is a possibility for violations of privacy if personal corpora such as email are revealed to the world. For this reason, databases were only created from corpora where all users with access to the RA already had access to the documents in the corpus. For example, only Media Lab users were allowed access to the databases drawn from the Media Lab mailing list archives.[4]

However, even with these safeguards in place several privacy issues were raised. The largest concern was that archives, while public to the Media Lab community, are still expected to be read only by people who are actively searching for information. One Media Lab user described the problem in these terms:

> *I don't like the centralized permanent public archiving of group lists. My experience with the RA highlights one reason: I often get recommendations of emails that were clearly not intended to be seen beyond the original mail alias audience. For example there are flames by [other students] from years ago (occasionally complaining about other parties in the Lab, who were not addressees, but who will likely see the flames if they run the RA or some other tool, or even grep or read the archives). I also get similarly sensitive emails from other groups. The RA is not the problem – don't slay the messenger. The problem is the fact that we have a central archive of sometimes-sensitive emails that is accessible to too large and varied a set of people outside the intended audience.*

(Bennahum 1999)
Bennahum, D. Daemon Seed: Old email never dies, *Wired*, 7.05, May 1999

As is mentioned in the above quote, the problem of archives being available to an unintended audience is not unique to the RA, and in recent years this issue has led to many corporations instituting email deletion policies **(Bennahum 1999)**. However, technologies like the RA reuse archives in ways that were not anticipated when the email was first written and posted. For example, posters to a class mailing list might

4. Mailing lists are by default not archived, but many list administrators turn on archiving. These archives range from general technical lists (e.g. the *wearables* list or the *linux-users* list) to group and class-specific mailing list archives. All are protected with Unix group file permissions, though most are accessible to anyone in the Media Lab. Only those archives that are available to anyone within the lab were indexed, and members of the lab were asked to name any lists they did not want to be indexed.

expect class members to look back through the archives, or even for people taking the class in subsequent years to look through the archives. However, posters will probably not expect their emails to appear, unsolicited, on other people's displays. This is sometimes called the "Dejanews effect," after the concerns raised when the company *Dejanews* started archiving and making searchable old net news articles.

Even when files have been examined and private documents have been removed, private information can unintentionally be released. For example, Thad Starner and the author would often share notes taken on each other's wearables. All the notes files that were exchanged were examined by their respective owners before-hand. However, a few days after "exchanging brains" the author asked Starner about a particular meeting he was going to attend. Starner was surprised, as the meeting had not been publicly announced and he had deliberately removed mention of the meeting from his notes before releasing them. However, Jimminy had combined two different pieces of information that, taken together, indicated the existence of the meeting. This example shows the difficulty in removing private information, even when the recipient of the database is a known and trusted person. It becomes even more difficult when distributing a corpus to a larger user base.

Group minds and the need for context. The previous example points out one of the more interesting applications of JITIRs: the creation of a "group mind" where information is shared among members of a community in an associative manner. The big advantage of a group mind is that information is likely to be relevant to a person's life if it is information known by other people in her own community.

As was discussed in Chapter 3.3.4, it is easier for a person to process and understand information that she has previously seen. A suggestion of this kind jogs a user's memory, but much of the knowledge required to interpret the suggestion is already in the user's head. Second easiest to process or understand is a suggestion that has not been seen before but where the user knows the suggestion's context. For example, the user may not have seen a particular suggested email but may know the sender and the topic being discussed. Many of the examples discussed above fit into this kind of group mind. Suggested information that is written for an audience that does not include the user are the hardest to interpret, contextualize and understand.

One example of a misinterpretation of an RA suggestion came from a subject in the information retrieval experiment. One of the annotations from the Media Lab email archives was an email from a student in another group at the Media Lab that jokingly said "look what I've been doing" and included a citation discussing some abstract physics research. In fact, the citation he sent was from another researcher at an entirely different university that happened to have the same first and last names. All the original recipients of the email knew the sender and know it must be a joke. However, the subject who saw the email did not know the original sender well and commented "wow, I had no idea he was working on that sort of thing."

As another example, another subject in the IR experiment commented that she read a suggested email discussing the topic of her paper, but she did not know if the person writing the email was competent. This difficulty stemmed from two sources. First, she had not read the particular email list to which the mail was sent, so she had not seen the sender of the email before. Second, email does not include a reputation mechanism upon which she could find more information about the person sending the email. The INSPEC database, on the other hand, included a built-in reputation system, namely the reputation of the peer-reviewed journal or conference proceedings in which the citation is published.

In a final example, when Starner and the author exchanged notes files from their respective wearables it was often hard to interpret subject lines from the other person's notes. For example, Jimminy would occasionally display a suggestion with the subject "linux" from Starner's notes files. Starner knew what information was contained in his *linux* file, but the author did not.

Social expectations when using a JITIR. Once it becomes known that a person is using a JITIR, other people start changing their expectations about the user. Several users have commented that their friends now send them questions for their RA to answer. For example, one user received mail from a friend asking:

> *You have a neat email program, I gather. Could you find, say a time I said to you "you're just a regular Lloyd Dobler, aren't you?" (or something in that vein) – let me know.*

Using the RA he found it quickly, but the important point is that people realize that such questions are less of a burden to a user of a JITIR and therefore feel more free to ask them. Once several people in a group start using a JITIR, especially when everyone is known to use a similar database, then other assumptions are made. For example, one RA user sent the following email to the agents mailing list after loading the printer (Godzilla) with thesis-bond paper:

> *Godzilla is about to be filled with thesis paper. If you are not familiar with the implications of this, please look at your RA window to see many treatises by Lenny on the topic. :)*

The implication of his email was that people should not print while he was printing his thesis, a topic of much flamage a year previous when Lenny was printing *his* thesis.

(Reeves 1996)
Reeves, B. and C. Nass, *The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places*, 1996

Attribution of personality. In a series of experiments, Reeves and Nass have shown that people will personify technology **(Reeves 1996)**. True to their results, users of JITIRs will often describe them in anthropomorphic terms. Several of the examples listed above contain anthropomorphisms such as "I thought I had better humour it" and "your Ph.D. thesis told me to...." Another long-term user from the Media Lab commented that when Margin Notes would make a bad suggestion he would think to himself "Oh, silly Margin Notes. You think this is relevant because of these words, but it isn't."

Forgetting to turn it on. One of the more paradoxical (and frustrating) aspects of the users of JITIRs is that they often forget to turn on the system. One user from the Media Lab commented that the RA was quite useful when he was running it but he never remembered to start it. Another user who downloaded the system from the Internet had a similar experience. The RA had a bug (now fixed) that caused it to be turned off whenever the GNUS news-reading program was run. This user commented:

> *RA would be more useful to me if I could turn it on and leave it on. I find that when using GNUS (pgnus) RA only lasts for reading one news group and must be re-invoked. I most often just forget to turn it back on.... My laziness gets worse: I think my use of RA would be a lot more effective if I remembered to recatalog my notes. Mostly, I have been*

> *just taking [the program] and letting it do whatever it does from the collection I first indexed back with my first version.*

In spite of this so-called laziness, these users and others like them claim the RA and Margin Notes are useful and desirable even though they forget to start them. When asked about this seeming contradiction users did not have an explanation, but it fits within the effort-accuracy framework. Unlike directed-use tools such as search engines, there is usually never a point when a user knows she needs to use a JITIR. The whole point is that the JITIR surprises the user with useful information. Because the JITIR is usually in the background for a user, it is easy to forget to start the system if it is not currently operating. For this reason, many users configure the RA to automatically turn on when reading email or when starting Emacs.

CHAPTER 6 *Related Systems*

In the sciences, we are now uniquely privileged to sit side by side with the giants on whose shoulders we stand.
– Gerald Holton

This chapter describes other JITIRs as well as systems that are related but do not qualify as JITIRs under the strict definition given in Chapter 1.1. The list provided here is not intended to be a complete annotated bibliography, but rather to show the range of techniques being pursued in this field. For theoretical work related to this research, see Chapter 3.

6.1 Other JITIRs

The systems described in this section have all three features required to be a JITIR: proactivity, an interface that is accessible but non-intrusive, and a dependence on local context to produce suggested information. These systems can be described in terms of how they select the information they provide and how that information is displayed. In other words, they can be described in terms of information retrieval and interface design. The information retrieval methods used can be further broken down into the amount of history that is considered to be local context, how structured the local context that is sensed needs to be for the system to work, and how structured the information provided by the system needs to be. The interface design can be described in terms of how much unrequested information the system provides versus how much it requires the user to dig for information, i.e. how the system trades off accessibility for non-intrusiveness of the interface. Neither end of any of these four spectra is better than the other; these are all trade-offs that must be made based on the environment for which a system is being designed. Each of these spectra are described in more detail below.

Amount of history used. Chapter 3.2.4 describes *user profiles* and *local context* as two ends of a continuum of information sources a proactive information system can use to provide a user with useful information. At one end of the spectrum, a user profile includes information about a person that changes slowly over time, e.g. her history of actions taken, her interests, her job title, or her mailing address. At the other end, local context includes information about a person that changes quickly, e.g. the room she is in, the person she is talking to, or her current topic of conversation. In between these two ends is information that changes over the course of hours, days or weeks, e.g. her main project for the week, the destination of a trip she will be taking in a few days, or the Birthday present she wants to buy sometime within the month.

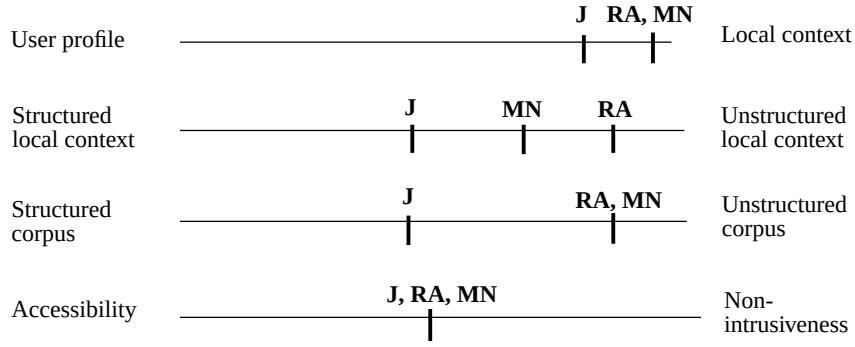
By definition, a JITIR relies on local context to determine what information is valuable to a user, but systems that qualify as JITIRs can still vary with regard to exactly where on the continuum they fall as long as they tend towards local context.

Reliance on the structure of the local context. Some JITIRs rely on the domain-specific structure of the local context to determine what information to provide. This reliance on domain-specific local context can improve the performance of a JITIR. For example, Margin Notes relies on knowledge about the structure of HTML documents to parse web pages into sections. However, such reliance also makes it harder to apply the techniques used by the JITIR to other domains.

Reliance on the structure of the corpus. JITIRs can also rely on the domain-specific structure of the corpus of information the JITIR provides. For example, Jimminy relies on the fact that the database of notes taken on the wearable are all annotated with information about the user's physical environment when the note was taken. While Jimminy can still fall back on pure text-retrieval when using a non-annotated database, much of its functionality is lost. As with the reliance on structure in the local context, reliance on structure in the corpus can provide additional performance at the cost of generality.

Accessibility versus intrusiveness. As discussed in Chapter 3.3.3, there is often a trade-off between accessibility and non-intrusiveness in a JITIR interface. Different systems will fall at different points along the continuum between presenting information early (maximizing accessibility) and providing almost no information until requested by the user (maximizing non-intrusiveness).

FIGURE 18. Dimensions of the RA, Margin Notes and Jimminy



J = Jimminy
MN = Margin Notes
RA = Remembrance Agent

Figure 18 shows the placement of the RA, Margin Notes and Jimminy along the four dimensions described above. Such placement is subjective, but gives a rough idea of where the systems stand in relation to one another. All three systems rely on the user's current context more than aspects of her history. Jimminy is positioned slightly more towards the user profile because it does rely on which features have changed within the last minute, in order to set the feature biases. The RA can improve performance by relying on the structure of the local context or corpus (using the template system), but

can also operate easily when the local context or corpus is raw unstructured text. Margin Notes uses the same corpora as the RA but relies on the fact that the local context is HTML, while Jimminy requires an annotated corpus of notes and a local context that includes information from the physical world in order to work as intended. All three systems are designed to equally trade off accessibility and non-intrusiveness.

6.1.1 Watson

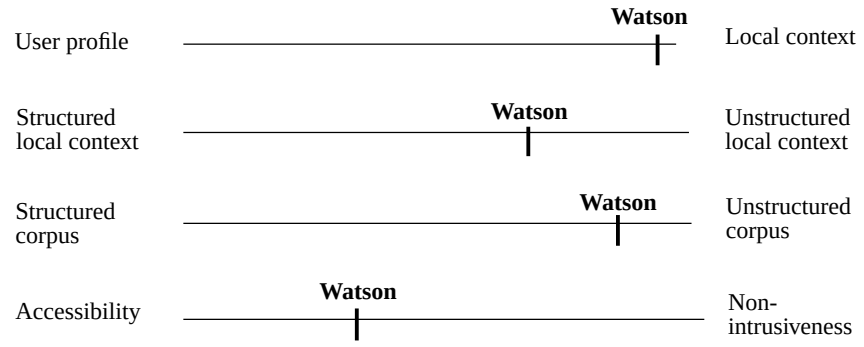
Watson **(Budzik 1999)** is a JITIR that proactively suggests information from the Web based on a user's current web page or Microsoft Word document. Results are clustered and displayed in a separate window, with the title of each web page appearing as a one-line suggestion. Clicking on the suggestion line brings up the suggested page in the web browser.

[**Budzik 1999)** Budzik, J. and K. Hammond. Watson: Anticipating and Contextualizing Information Needs. In *Proc. of the 62nd Annual Meeting of the ASIS*, 1999

To produce suggestions, Watson analyzes the content of the document the user is manipulating, identifies the most important words in the document, and sends those words to a third-party search engine such as AltaVista.[1] Watson can also use more domain-specific third-party search engines. For example, whenever a Microsoft Word user creates an image caption, Watson will retrieve images from the ArribaVista image search engine[2] that might match that caption.

Watson also has a semi-proactive mode the designers call *query in context*. In this mode, a user specifies a query in an entry field in the Watson window, just as she would in a normal search engine. Watson then sends the query to AltaVista, augmented with terms from the user's current local context. The web pages returned are related to the user's query as well as the context of her current task.

FIGURE 19. Dimensions for Watson



Like the RA, Watson relies almost entirely on the user's current local context to find useful information, rather than relying on a user profile or historical information. While Watson uses general databases such as AltaVista, it also uses domain-specific heuristics where possible. For example, the words that are chosen for use in the query sent to the search engine are selected based on the part of the document they are from (title, section, top or bottom of document), the font used for the word, whether the word appears in a list, and domain-specific heuristics such as whether a word appears

1. http://www.altavista.com
2. http://www.arribavista.com

in the navigation bar of a web page. As mentioned earlier, Watson also uses domain-knowledge to find images that match picture captions. The corpus, on the other hand, is relatively unstructured because Watson uses third-party search engines. While Watson may occasionally use domain knowledge about a database (e.g. the fact that the database is an image server), in normal use Watson only relies on a document having an associated URL. Finally, Watson displays around ten full web-page titles in a separate window. For this reason, the interface for Watson takes up more room and attention than does the interface for the RA, in exchange for making more suggestions easily accessible.

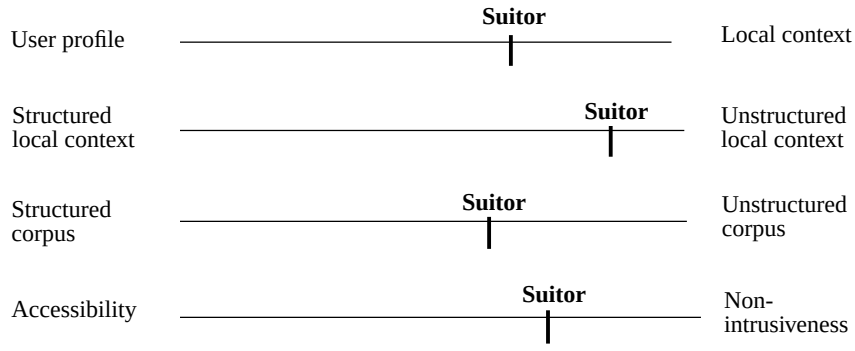
6.1.2 Suitor

(Maglio 2000)
Maglio, P., et al. SUITOR: An Attentive Information System. In *The Proc. of IUI 2000*, January 9-12, 2000, pp. 169-176

The Simple User Interest Tracker (Suitor) **(Maglio 2000)** watches a person's web browser, word processor, and other applications and provides news and stock information in a scrolling display directly above the task bar at the bottom of the screen. Suitor builds a model of a user's interests by watching what application is being used, what web page is being viewed, what text is being typed, and where the user is looking on the screen based on an eye-tracker. The choice of what news and stock information to display is then based on this short-term user profile that is built up over the course of several hours.

Suitor is based on a blackboard architecture with multiple agents, each looking for domain-specific information. For example, if a person is looking at the IBM web page and also looking at financial pages, one agent within Suitor will tell the system to display IBM stock quotes at regular intervals. If he starts using Microsoft Word, another agent will tell Suitor to display tips on using MS word.

FIGURE 20. Dimensions for Suitor



Suitor bases its suggestions on the interests a user has shown over the course of several hours. For example, if the user views a web page about Europe then for the next few hours the scrolling suggestion headlines will tend to show news headlines about Europe and travelling. The advantage to using a short-term profile is that a more complete model of the user's interests can be built by watching multiple actions. The disadvantage is that suggestions may no longer be related to the user's current interests. Suitor does not rely much on the structure of the user's local context, although it does monitor what application is being used and makes assumptions about the user's interests based on that information. The corpora used are currently structured, e.g. stock information is shown whenever a person shows interest in a particular company.

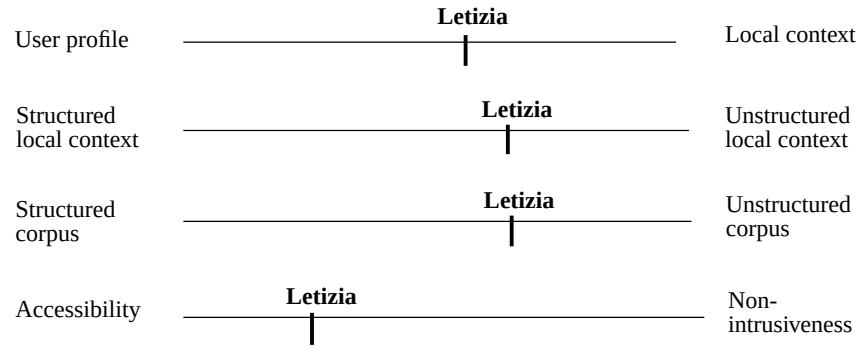
Finally, Suitor initially shows only a single headline or stock quote. Only when the user clicks on the scrolling display will he receive other information.

6.1.3 Letizia

[**Lieberman 1997)** Lieberman, H. Autonomous Interface Agents. In *Proc. of CHI'97*, March 1997, pp. 67-74

Letizia **(Lieberman 1997)** automatically recommends web pages a person might want to visit, based on a short-term user profile and the web page he is currently viewing. Letizia automatically creates a user profile by compiling keywords contained in previous pages viewed by the user. It then acts as an "advance scout," following links from the user's current web page and bringing up those pages in the "local neighborhood" that match the user's profile. In the RA, Margin Notes and Jimminy the source of suggested information is static (though slowly updated) and the user's current context is used to retrieve this information. Letizia is just the opposite: the source of information is the user's current local context (the pages linked to from his current web page), and documents are suggested based on the user's profile that changes over the course of hours or days. Letizia shows the entirety of a recommended Web page in a separate window rather than presenting a summary of potential hits and letting the user pick which suggestions to view.

FIGURE 21. Dimensions for Letizia



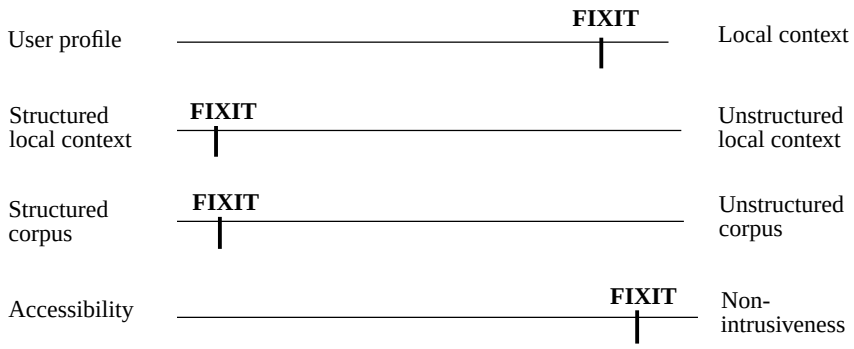
As mentioned above, Letizia bases its suggestions on a user profile that can be updated over the course of days, although the corpus is based on the user's local context. Letizia relies on the fact that the user is in a web-browsing environment for both the creation of the user profile and for suggesting documents, although it uses standard text-retrieval techniques (TF/iDF) to do the final matching of a suggestion to a user's profile. Finally, Letizia shows the entirety of a suggested web page within a large window. This interface makes it easy to read and access the suggested page, but is more intrusive and takes more screen real-estate than would a ramping interface.

6.1.4 FIXIT

(Hart 1997) Hart, P. and J. Graham. Query-free Information Retrieval. *IEEE Expert / Intelligent Systems & Their Applications*, 12(5), September / October 1997.

FIXIT **(Hart 1997)** is a JITIR embedded within an expert system for copy machine repair. Based on the user's interaction with the expert system, FIXIT automatically brings up copier-repair manual pages relating to the fault being diagnosed. It uses the table of contents for the repair manual to find useful pages, based on the diagnosis produced by the expert system. While the techniques used are domain-dependent, the corpus (manual pages) is a legacy system and was not hand-annotated for use with the system.

FIGURE 22. Dimensions for FIXIT



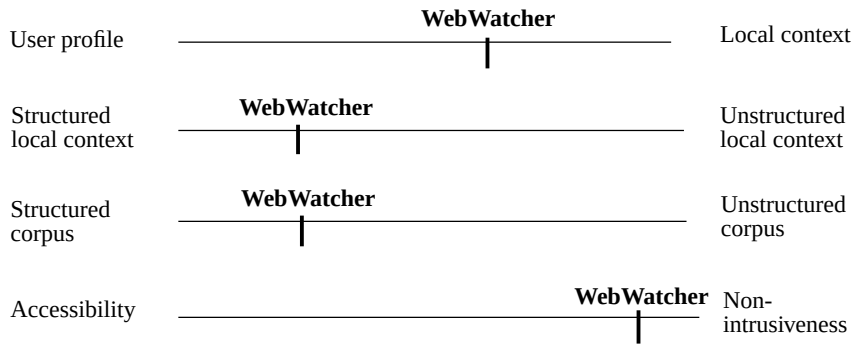
FIXIT uses a user's current state in a diagnosis procedure to find relevant manual pages, although the user's previous results in tracking down a particular fault are also used. The primary feature of FIXIT is that its information retrieval algorithms rely very heavily on the particular structure of the local context and the structure of the manual pages being suggested. In this case, the local context is the potential faults being proposed by the expert system (based on a Bayesian network), which map to topic headings from the table of contents for the manual that forms the corpus. This reliance on known structure makes FIXIT well suited for the repair of a particular set of copiers, but also makes it difficult to transfer the techniques used or the system itself to a different domain. Finally, by default FIXIT only displays a small icon next to a line of output from the expert system, indicating that a manual entry is available. Clicking on the icon brings up a separate window containing several suggested manual pages that may be useful in the repair technician's current task. This interface is especially non-intrusive, at the cost of making the information harder to access.

6.1.5 WebWatcher

(Joachims 1997)
Joachims, T. et al. WebWatcher: A Tour Guide for the World Wide Web. In *IJCAI*'97, 1997

WebWatcher **(Joachims 1997)** is similar to Letizia, in that it recommends links from the current web page being browsed, based on a short-term user profile. However, WebWatcher cannot suggest links beyond the narrow set of pages for which it is installed. For example, WebWatcher was installed to act as a "tour guide" on the Carnegie Mellon University computer science web pages, but could not work outside of that set of pages. Instead of automatically learning a user profile based on previous user actions (as Letizia does), WebWatcher explicitly asks a user for her current goal or interest when she goes to WebWatcher's front page. The user's stated interest is used to determine which links from a page should be recommended, based on link/interest pairings that have been learned from previous users of the system.

FIGURE 23. Dimensions for WebWatcher



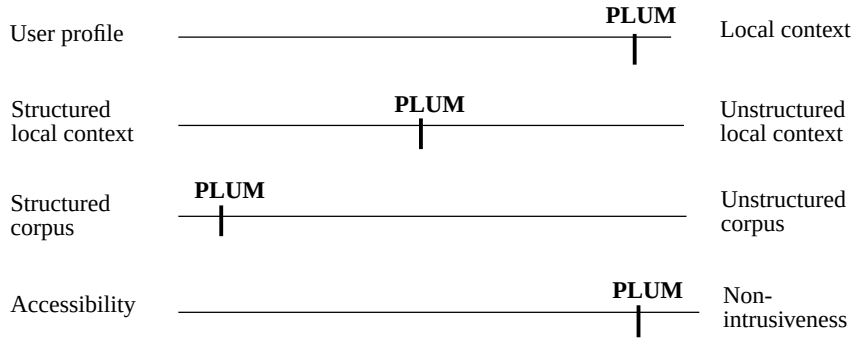
Like Letizia, WebWatcher bases its suggestions on a short-term user profile, with the pool of possible suggestions drawn from the links off of the user's current web page. WebWatcher also relies heavily on the link structure of the web to learn what link to suggest. Links are suggested by comparing the user's stated interest to the link's description text plus the interests of all users who have followed the link. Comparisons are based on a standard TF/iDF document retrieval algorithm. Links are suggested by adding a pair of small icons around the link itself. This interface is very non-intrusive, but does not give any information about why a link is being suggested. However, the user should still be able to guess why a link is being suggested, given that all suggestions are based on the keywords stated by the user at the start.

6.1.6 PLUM

(Elo 1995)
Elo. S., *PLUM: Contextualizing News for Communities Through Augmentation.* Master's thesis, MIT Media Arts and Sciences, 1995

The Peace, Love, and Understanding Machine (PLUM) system **(Elo 1995)** will automatically add hyperlinks to disaster news stories to better give a reader a personalized understanding of the news. For example, a report that 220 people were killed in an Earthquake in a small town in Japan would contain hyperlinks from the number "220" to a description of what percentage of the population were killed and how many people that percentage would be in the reader's own home town.

FIGURE 24. Dimensions for PLUM



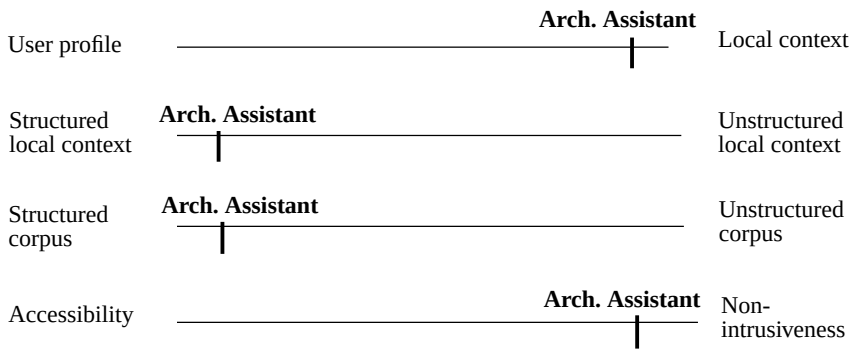
PLUM actually uses both the user's local context (the disaster story she is reading on the web) and a long-term, static user profile that contains the user's hometown. Based on the user's hometown, information in the article such as population sizes and geographic regions are contextualized with information the user can understand. PLUM relies on the fact that disaster stories tend to fit a known structure, and it parses the natural language text to extract this information. The corpus of information provided, on the other hand, is designed specifically for the system. PLUM is very non-intrusive; words are turned into hypertext links but no other information is added. However, like with WebWatcher the user can usually guess what kind of information is behind a link based on knowledge in the user's head (her knowledge about how PLUM works) and knowledge in the world (the words that makes up the link).

6.1.7 Context-Aware Archeological Assistant

(Ryan 1998)
Ryan, N., et al. Enhanced Reality Fieldwork: the Context-aware Archaeological Assistant, in *Computer. Applications in Archaeology* 1997, 1998

The Context-aware Archeological Assistant **(Ryan 1998)** is a mobile system built for taking archeology notes in the field. The system runs on either the Palm Pilot or Newton hand-held computers, connected to a Global Positioning System (GPS). Notes written using the system are automatically tagged with the user's current location. As the user moves around, notes that were taken in the area appear as icons on a small map displayed on the hand-held. The Archeological Assistant is very similar to Jimminy, as both retrieve automatically annotated notes based on the user's current physical context.

FIGURE 25. Dimensions for Archeological Assistant



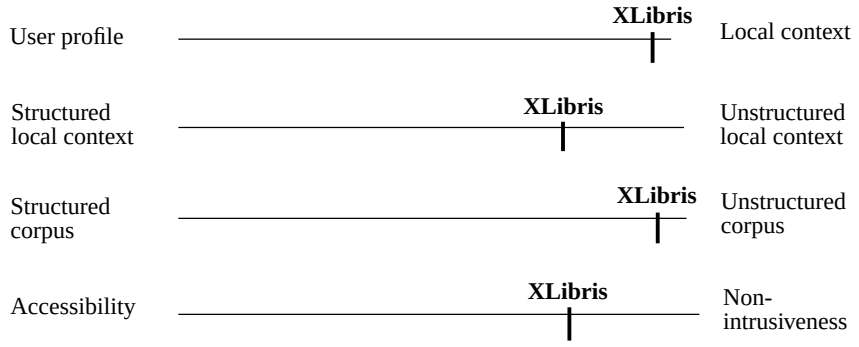
The Archeological Assistant only uses the user's local context (his location) to suggest documents. However, unlike Jimminy, suggestions are only displayed based on location; there is no fall-back to other content. This makes the Assistant highly reliant on the particular format of the data and local context being sensed. The interface for suggesting information is quite subtle and non-intrusive: individual notes appear as small boxes overlaid on top of a map of the trail the user has walked. Clicking on a square brings up the entire note.

6.1.8 XLibris

(Price 1998)
Price, M. et al. Linking by Inking: Trailblazing in a Paper-like Hypertext. In *The Proc. of Hyper-Text'98*, 1998, pp. 30-39

XLibris **(Price 1998)** is a JITIR that provides information based on a document being read in large pen-based computer. Documents are suggested based on a query constructed from the words that are highlighted or otherwise near annotations the reader has made using the pen. Suggestions then appear in the margin of the document, represented by a thumbnail image of a document. The developers note that thumbnails are useful for recognizing previously viewed documents based on the shape and flow of the words on the page, but are not good for recognizing new information because the words are too small to read.

FIGURE 26. Dimensions for XLibris



XLibris uses only the text of a document that has been highlighted or annotated by the user, so it uses highly localized context (not even spanning a single page). The system relies upon user annotations to identify important regions of the text, but beyond this any text can form a query. The corpus can be any form of text, and no structure is expected or used. Finally, the interface is designed to be non-intrusive. Like Margin Notes, the interface uses the placement of the suggestion to indicate what information a suggestion regards. However, unlike Margin Notes, XLibris only uses thumbnail icons of the suggested document, trading off accessibility of the information for non-intrusiveness of the interface.

6.2 Other Related Systems

The systems described in this section are related to JITIRs, but do not meet all the features required by the definition.

6.2.1 Forget-Me-Not

(Lamming 1994)
Lamming, M. et al. The design of a human memory prosthesis. *The Computer Journal*, 37(3):153-163, 1994

Forget-Me-Not **(Lamming 1994)** is a portable computer system that automatically detects where a person is, who he is talking to, and other information and stores it in a diary. Forget-Me-Not is similar to Jimminy in that information is annotated and stored, but Forget-Me-Not continuously and automatically stores information to its diary file while Jimminy only logs a person's physical context when he is actively taking notes. However, Forget-Me-Not is not a JITIR because it does not proactively provide information: it can only be used to answer explicit queries. For example, the diary can be queried to answer questions such as "who was it that I ran into after I got off the phone with Joe?"

6.2.2 Nomadic Radio

(Sawhney 1999)
N. Sawhney and C. Schmandt. Nomadic Radio: Scalable and Contextual Notification for Wearable Audio Messaging. In *Proc. of CHI'99*, 1999

Nomadic Radio **(Sawhney 1999)** is a wearable system that delivers news, voice mail, and email via audio. Information is played into two shoulder-worn speakers, starting with low ambient sounds and then scaling through levels of intrusiveness from a subtle auditory cue to full foreground presentation of the message. This system, which Sawhney calls *Dynamic Scaling*, is similar to a ramping interface but with a few important differences. Ramping interfaces are designed to make it simple for the user to decide what level of information she wishes to receive. If she decides information is not valuable she can simply not look further. A dynamic scaling system, on the other hand, follows a pre-determined sequence of increasing intrusiveness, and it requires action on the part of the user to *not* hear more. This difference stems from the fact that Nomadic radio is audio while the ramping interfaces described in this thesis are visual. It is easy to avert one's gaze away from a suggestion and thus not see more. It is much more difficult to "avert one's ears" and not hear another stage.

To keep the interface from being distracting, Nomadic Radio uses the user's history, his local context and the importance of the message being played to decide how intrusive a message should be. If the user has not recently used the system, if she is in the middle of a conversation (as detected via the microphone) or if a message is unimportant then the system will follow a relatively non-intrusive ramp for outputting information. For example, the system might play a quiet sound of water running that slowly increases in volume (thus getting the user's attention), followed by an auditory cue and a short summary. The full body of the message would only be played if the user requests it. On the other hand, if the system expects that the user is not busy or if a message is judged to be important then a faster ambient sound and perhaps a full preview will be played. The system also maintains a model of how interruptible the user is at the moment and uses machine learning techniques to change that model based on how often the user overrides the default level of dynamic scaling with which a message is played.

Nomadic Radio proactively provides information in an accessible yet non-intrusive manner, but it does not choose the information to provide based on a user's local context, so it does not count as a JITIR.

6.2.3 Audio Aura

(Mynatt 1998)
Mynatt, E. et al., Designing Audio Aura. In *Proceedings of CHI'98*, 1998, pp. 566-573

Audio Aura **(Mynatt 1998)** is another audio-based wearable system that uses ambient sound to automatically indicate email, group activity and information delivered based on the user's location. The goal of Audio Aura is to present serendipitous information via background audio cues. The information provided is often tied to the user's physical actions in the workplace, e.g. when the user passes by an office she hears the name of the person who works there. Like Nomadic Radio, one main research goal of the system is to provide audio alerts in an accessible yet non-intrusive manner. This is accomplished by carefully choosing sounds without sharp attacks, high volume levels, or substantial frequency content in the same general range as human voice.

6.3 Other Interface Techniques

Several techniques have been proposed for the presentation of information in an accessible yet non-intrusive manner. These techniques include augmented reality, ubiquitous computing and ambient interfaces, and various interfaces for presenting hypertext information. Each of these kinds of techniques will be briefly described below.

6.3.1 Augmented Reality

Augmented Reality (AR) is the overlay of a computer-generated graphics on top of a person's vision of the real world. For example, Columbia's KARMA system **(Feiner 1993)** used a head-up display to overlay graphical instructions and diagrams on top of real-world devices to aid in tasks such as copy machine repair. **(Rekimoto 1998)** describes similar system where a person wears a see-through head-up display, which displays graphical "post-it notes" that appear to float on top of real objects. By tracking a person's head position via a CCD camera mounted on the display, the computer system can move the graphical overlay as if it were a part of the physical world.

(Feiner 1993)
Feiner, S. et al. Knowledge-based augmented reality. *CACM*, 36(7):52-62, July 19**93**

(Rekimoto 1998)
Rekimoto, J. et al. Augment-able Reality: Situated Communications Through Physical and Digital Spaces. In *ISWC'98*, 1998

While augmented reality is usually associated only with graphical overlay, AR can also be thought of as what might be called a *deictic interface*: an interface that uses the real world to contextualize the information being provided by linking the interface to components of the real world. For example, in Rekimoto's system a virtual post-it note might be "attached" to a VCR and read "don't use this, it's broken." The interface itself is using the real-world object that is physically collocated with the post-it note to let the user know the object to which "it" refers. Audio Aura uses a similar technique when a name is spoken next to a person's office. The location in which the name is spoken (a feature of the physical world) is associated with the virtual information being provided (the name). Such a system might be thought of as a form of *audio augmented reality.* Taking the metaphor one step further, the interface of Margin Notes could be considered a form of augmented reality (though most AR researchers would probably disagree). Margin Notes uses a part of the real world (the user's web browser, his primary task) to indicate what section of the web page a particular annotation regards.

6.3.2 Ubiquitous Computing and Ambient Interfaces

While AR integrates computer-generated graphical information into the real world, the goal of *ubiquitous computing* is to integrate computational elements themselves into the real world **(Weiser 1993)**. For example, small amounts of computation can be integrated into walls, name tags, and pieces of furniture. Ubiquitous computing shares many of the same promises as does augmented reality: the location of information can be used to help convey information in a way that is easily processed and contexualized. In the above example, instead of a virtual post-it note appearing on a head-up display announcing that the VCR is broken, a small computer display in the room or on the VCR itself would announce the fact.

(Weiser 1993)
Weiser, M. Some Computer Science Issues in Ubiquitous Computing. *CACM*, 36(7):75-84, July 1993

Ambient interfaces **(Ishii 1998)** are another method for providing information in a non-intrusive manner. The idea is that humans can process many kinds of information "in the background" while still concentrating on a primary task. For example, a person can tell if there is a lot of traffic outside of her office without being distracted, because the traffic sounds have become a part of the background. People who design ambient interfaces hope to capitalize on this ability by conveying information in the form of abstract light spots on the wall, the sound of water or rain drops falling, or the level of light in a room. The advantage to such interfaces is they tend not to be a distraction. However, it is also difficult to convey a large amount of information through such interfaces.

(Ishii 1998)
Ishii, H. et al. ambientROOM: Integrating Ambient Media With Architectural Space, in *Proceedings of CHI'98*, 1998, pp. 173-174

CHAPTER 7 *Future Work and Conclusions*

A work of art is never completed, only abandoned.
– Paul Valéry

7.1 Future Work

There are several directions in which this research can proceed. These directions can be categorized into five broad areas: improved personalization, machine learning and maintaining state, improved user control, communityware applications and extending the toolkit. Each of these areas will be discussed below.

7.1.1 Improved Personalization

In the systems described the only personalization is through choosing a personalized database and the setting of a few minor customization variables. This is because the research focuses on presenting information based on knowledge of the world (local context) rather than knowledge of the user (personal profile). However, hybrid systems could be quite effective.

One way to add personalization is to combine local context with a personal profile so suggestions are related to both. For example, such a system might give extra weight to suggestions from email that are associated with people the user knows, or to suggested INSPEC citations from conferences the user has attended. The simplest combination would be to add the user's profile as another "sensor" in the current infrastructure. This would allow users to increase or decrease the weight given to the environment or personal profile in terms of what features are shown. However, this could also be overdone: giving too much weight to a profile would cause the same documents to be suggested regardless of the current environment. It is important that if a profile is used as a feature in the overall document similarity metric that it only be used to refine the ordering of documents and not become the dominant feature.

Another use of a profile would be to remove certain documents from consideration. For example, a profile could be used to avoid showing INSPEC citations that were written by the author. Users might also want to specify features that always give extra value to a document regardless of the local context. For example, one might specify that email sent to a certain list or articles from certain journals are more valuable than others.

Users have also requested the ability to personalize the display of information. For example, one subject in the IR experiment said she wished Margin Notes would indi-

cate in the annotation if a paper was written by a researcher who was local. She would be more likely to read these citations because she would know the author was a potential collaborator.

7.1.2 Machine Learning and Maintaining State

The current JITIR implementations are stateless: no information is saved between queries with the exception of Jimminy's one-minute increase in bias when a feature changes. One use of state would be to enable users to say "never show this document again." Similarly, many users have requested a configuration where, after showing a suggestion once, it does not show it again for a while. This was especially requested for Margin Notes, where people often go back to the same set of pages. However, users also commented that they would occasionally go to a Web page not to get information on the page, but rather to look at an annotation that they remembered had been suggested for that page. This would not be possible if the system only showed new annotations, so it should be a configurable option.

Currently the biases for different fields are set by hand in Savant. These weights are heuristics based on the designer's expectations, for example it is expected that the body of an email message will be four times as important as the person sending the email. However, such guesses can be wrong, and can change depending on the particular database and user. Much better would be to use machine learning techniques to adjust these weights automatically based on user feedback. Machine learning could also be used to help determine which database is appropriate for a given context. For example, the system might learn that a database of office memos is useful whenever email is being read from one's boss.

7.1.3 Continuum of Control

This research is concerned with the proactive display of information. However, there is a large continuum between control by the agent and control by the user. For example, in the housing essay experiment several subjects requested the ability to limit articles returned to a specific time period, or to a certain section of the document. Some user control is already available in the RA by clicking on the field of a suggestions, manual field-searches and full-page queries, but the addition of other manual controls and hints to the system would be a natural extension. It would also be useful to attach these JITIRs into a full knowledge-management system, such as Lotus Notes, that supports complex manual browsing and searching of large amounts of data. This combination would allow the user to easily find out much more about a suggestion's subject or context after the JITIR has brought it to her attention.

7.1.4 Community-ware Extensions

One of the uses of all three implemented JITIRs has been to create a form of "group mind" or group memory. For example, the RA using the Media Lab email corpus can bring up past discussions related to current issues, even when all the original participants have since left the lab. It is also possible to forward email to a database, where it is automatically archived and indexed. These community-ware applications could be further extended. For example, the system could use what documents were followed and rated highly by other members of a community to help set weights and choose databases for new users. This helps alleviate the lack of training data for machine learning, discussed above. The system could also offer simple support for forwarding annotations to other users.

Privacy is an important issue with these communityware extensions. First, it must be easy to screen one's information before it becomes available to a larger community. It should always be obvious who can and cannot access personal files, and this should

be easily controllable. Even with these safeguards in place privacy is still an issue because JITIRs present information outside of its original context. For example, a student might take lecture notes and make them available to her research group. The lecturer makes assumptions about the audience to whom he is speaking and will leave certain assumptions unstated depending on the audience's sophistication. The student makes further assumptions when writing notes. If a JITIR then shows these notes to someone who was not the intended audience that person could misinterpret the original statements because they are being read in a context that was not originally intended. This could be viewed as a violation of the lecturers privacy, or at least his right to control the breadth of publication for his words.

A similar problem arises with the reuse of mailing lists, as is discussed in Chapter 5.4.3.3. When sending mail there is the expectation that the mail will be read by those on the list. There might also be the expectation that the mail is archived by individuals on the list or automatically, and that people not on the list might search the archives if they are interested in the list. However, there is usually not the expectation that someone not on a mailing list might have your email automatically shown to him even when he is not browsing the archives or does not know about the existence of the mailing list. These issues all need to be addressed, probably by a combination of technical and social solutions.

7.1.5 More Sensors for Jimminy

As mentioned in Chapter 2.5, the end-goal is for Jimminy to passively sense all aspects of a person's environment and use this sensor data to provide useful information. The system has been shown with passive sensors for location and people in the room (via active badges), but the subject of conversation is still entered by hand. The addition of automatic speech recognition would go a long way towards the goal of being entirely sensor-based, as would the addition of automatic face-recognition instead of relying on people wearing active badges.

7.2 Conclusion

This thesis has defined Just-In-Time Information Retrieval agents (JITIRs): a class of software agents that proactively present potentially valuable information based on a person's local context in an easily accessible yet non-intrusive manner. Four main conclusions can be drawn from this research. First, JITIRs encourage the retrieval and examination of more information than would be retrieved without the use of a JITIR. Second, the use of *relevance* as an evaluation metric is more problematic for JITIRs than it is for traditional information retrieval, and does not capture the true value of a JITIR to a user. Third, there are many ways in which a JITIR can be valuable to a user, including providing information that changes the task being performed, information that supports a current task, information that contextualizes the current environment, and entertainment. Finally, the design of a JITIR needs to be strongly integrated with the task and environment in which it will be used. This integration should be at all levels of design, including the information retrieval algorithms used, the kind of information to display for a given task, and the interface for displaying information.

7.2.1 Encouraging Use of More Information

In terms of an economic model of human behavior, people will retrieve information if the expected benefit is larger than the expected cost of retrieval. JITIRs increase information retrieval by lowering the effort required to retrieve a document, and by giving the user an indication of the contents of a suggestion early in the interface, thus increasing the expected benefit of retrieval. JITIRs are not a replacement for direct information retrieval systems such as search engines, but they are partial substitutes. Like search engines, JITIRs can provide background or supporting material related to

a current task. JITIRs are also useful for free-form brainstorming, where a search engine is not as useful because there is not a specific topic that can be used as a query. Search engines are more appropriate than JITIRs when the answer to a specific question is desired.

7.2.2 Utility Versus Relevance

Traditional information retrieval algorithms are usually evaluated based on relevance to queries that have been picked based on the database from which information is drawn. This evaluation metric is based on the assumption that a query is a good representation of a user's actual needs, and that in normal use the corpus from which information is drawn contains documents that will be useful to a user in his current task. Neither of these assumptions are true for JITIRs. First, JITIRs automatically create the query used to retrieve information, so there is no guarantee that it represents the user's needs. Second, users do not explicitly state and may not even know their information needs, and therefore cannot be relied upon to specify the corpus from which a JITIR should draw information. This means there is no guarantee that the corpus contains any useful information at all.

JITIRs should be evaluated based on utility in a given environment rather than relevance. Relevance and utility may be correlated, but the correlation is not necessarily a strong one. In particular, relevant information might not be useful if it is already known, if it is low quality, or if the user simply does not need any information at the time it was given.

7.2.3 Value of a JITIR

The experiments performed for this thesis show that there are many ways a JITIR can be valuable to a user.

First, a JITIR can change the nature of the task being performed. For example, it might produce information that lets the user know that someone has already solved the problem currently being discussed. JITIRs can also provide answers to questions being written in email while they are being asked, and provide answers to questions that would not otherwise be answered because the effort required to find the answer would not be worth the benefit.

Second, a JITIR can provide information that supports a task being performed. For example, the information may support arguments being made, or provide a user with a quote from a primary source where she would otherwise have paraphrased from memory.

Third, a JITIR can help contextualize the current environment. For example, a newspaper article being read can be contextualized in terms of historical events related to the story. Such contextualization may not directly help with the current task, but it still gives the user a broader understanding of the environment.

Finally, JITIRs can provide information that is valuable outside of the current task, for example by providing entertainment. In the obvious case, a JITIR might recommend jokes or otherwise humourous documents when the user is already receptive to a distraction from work. Furthermore, the associations made by a JITIR can be entertaining in their own right because they reveal similarities that were not previously apparent. For example, a user might be entertained to know that email sent by a current girlfriend is similar to email sent by a previous girlfriend.

7.2.4 Need For Integration

JITIRs need to be integrated with the task environment at all levels of design. In particular, the task environment should influence the choice of features from the local context that are used, the corpus from which information is drawn and the interface used to display information.

A JITIR's information retrieval algorithm needs to use features from a user's local context that are good indicators of the current situation. For example, the people associated with an event may be very important for a salesman who has occasional meetings with a large number of clients. It is a less useful feature for an office worker who works with the same people every day, because then the people who attend an event are less indicative of what the event is actually *about*. Individual features also tend to contain noise, which must be removed. For example, email that is being read often contains headers and signature lines that have nothing to do with who the mail is from or what it is about. Similarly, web pages contain HTML markup tags that don't convey the overall meaning of a page. What constitutes noise in a feature depends on the task environment in which the JITIR is deployed, and thus the filtering algorithm used must be customized for the environment.

The effectiveness of a JITIR also depends on how well the database matches with the user's task and environment. For example, if a user is trying to decide what cell-phone service plan to use, no citation from the INSPEC database will be useful. Pages from personal email, the web, or product reviews from *Consumer Reports* are much more likely to contain useful information. On the other hand, the INSPEC database may be very appropriate for an electrical engineer who is designing a new kind of cellular phone system. To some extent the database can be chosen based on the user of the system. For example, if a researcher only uses her word processor to write technical papers, then a JITIR embedded in that word processor could always use the INSPEC database. However, if she also used the word processor for writing personal email then the JITIR would need to either determine which database to use at any particular time, or at least allow the user to easily change databases.

Finally, the interface for a JITIR must be designed with the user's task and typical environment in mind. First, the modality used by the JITIR should be chosen so it does not interfere with the user's primary task. For example, a JITIR used by a student during lectures should not use speech, because audio is already being used by the primary task. On the other hand, a JITIR intended for someone driving a car might use speech because the visual modality is primary in that environment. The design of a JITIR's interface also should take into account the cognitive load and social constraints of a typical user. For example, Jimminy tends to be used during conversations, where the user is under heavy cognitive load and there is a social cost to taking time to read a suggestion. In this case the interface needs to present information such that value can be gained by reading just a few words. Users of the RA and Margin Notes are in less demanding environments both cognitively and socially, so they can take more time to engage the system directly.

APPENDIX A *Controlled Experiment Material*

This appendix contains the experimental material for the controlled-task (MIT housing essay) experiment. Identical consent forms, pre-task surveys and debriefings were given to both groups. The first set of task description and post-task survey was given to the control group, the second to the experimental group.

Consent Form

Your participation in the following experiment is completely voluntary. You are free to withdraw this consent at any time, for any reason, and to request that any data collected be destroyed. If at any time you feel uncomfortable, or unsure that you wish your results to be part of the experiment, you may discontinue your participation with no repercussions.

In a few minutes, you will be asked to compose an essay using the computer. You will be provided with one or more information tools that you may use or not, at your discretion. Also, you will be asked to fill in one questionnaire before writing the essay and one after. You are free to decline to answer any or all questions. The entire experiment should take about an hour. You will be paid $10 as compensation for your participation.

If at any time you are uncomfortable with what you are being asked to do, you are free to ask that the experiment be suspended. All information collected during your participation will be destroyed and your payment will be prorated based on the time you have already spent.

Any responses that are collected during the experiment will be completely anonymous. From this point forward, only the ID number that appears on the upper right corner of this packet will be used to refer to you.

If you have any questions at any point during the experiment, the experimenter will gladly answer them.

Please read the following and sign on the lines below:

“I, the undersigned, have read and understood the explanations of the following research project and voluntarily consent to my participation in it. I understand that my responses will remain confidential and that I may terminate my participation at any time.

In the unlikely event of physical injury resulting from participation in this research, I understand that medical treatment will be available from the MIT Medical Department, including first aid emergency treatment and follow-up care as needed, and that my insurance carrier may be billed for the cost of such treatment. However, no compensation can be provided for medical care apart from the foregoing. I further understand that making such medical treatment available; or providing it, does not imply that such injury is the Investigator's fault. I also understand that by my participation in this study I am not waiving any of my legal rights.

I understand that I may also contact the Chairman of the Committee on the Use of Humans of Experimental Subjects, MIT 253-6787, if I feel I have been treated unfairly as a subject.”

Name:_____

Date:_____

Location: MIT Media Lab

Pre-task Survey

1. Gender: **Male** **Female**

2. Age _____

3. Is English your native language? **Yes** **No**

4. How much do you know about recent changes and controversy regarding MIT housing (1-7)?
   Nothing at all 1 2 3 4 5 6 7 **A large amount**

5. How much do you care about the issue of MIT housing (1-7)?
   None at all 1 2 3 4 5 6 7 **A large amount**

6. How much experience do you have in writing essays or news articles (1-7)?
   No experience 1 2 3 4 5 6 7 **Lots of experience**

7. How often do you read the MIT Tech?
   Never 1 2 3 4 5 6 7 **Every issue**

Task description

Pretend you are a guest contributor for the Tech and write an editorial or news article about MIT housing. The article could cover the Freshmen On Campus decision, graduate housing, new dormitories, or any other aspect of MIT housing and how it affects student life. The article should be around a page (about 600-700 words).

You will have up to 45 minutes to complete your article. This is not meant to rush you, but rather to put an upper bound on the duration of the experiment. If you complete the article before the 45 minutes, get the experimenter and he will continue to the next phase. If you wish, at the end of the experiment your article will be emailed to and/or printed out so you can have a copy. Your article will be compared to articles written by others in this experiment based on a number of criteria.

You should have already received a quick tutorial with the Tech search page. Feel free to use this tool as much or as little as you wish in writing your article. If you have questions now or during the experiment please ask the experimenter.

Post-task Survey

1. How difficult did you find the given article-writing task?
   Very Easy 1 2 3 4 5 6 7 **Quite difficult**

2. How much did you learn about the subject of the article during the experiment?
   None at all 1 2 3 4 5 6 7 **A large amount**

3. Did you have enough time to adequately complete the task?
   Yes No

4. How would you characterize your level of expertise with web-based search engines (circle one)?

   Never used one before today
   Occasional user
   Regular user
   Expert

5. How would you characterize your level of expertise with the Emacs text-editor (circle one)?

   Never used it before today
   Occasional user
   Regular user
   Expert

6. How distracting to your task did you find the search engine (1-7)?
   Not distracting 1 2 3 4 5 6 7 **Very Distracting**

7. How useful did you find the search-engine provided (1-7)?
   Not useful 1 2 3 4 5 6 7 **Very Useful**

8. If you were to perform a similar task, how much would you want to have the search-engine running and available (1-7)?
   Not at all 1 2 3 4 5 6 7 **I would definitely want it**

9. How often did you find the hits given by the search engine were useful in their own right, even when you didn't follow them to see the full article (1-7)?
   Never 1 2 3 4 5 6 7 **Quite often**

10. To what extent did you pay attention to the search engine (1-7)?
    Almost no attention 1 2 3 4 5 6 7 **Lots of attention**

11. When you read an article suggested by the search engine, how often was it useful to you (1-7)?
    Never 1 2 3 4 5 6 7 **Quite often**

Task description

Pretend you are a guest contributor for the Tech and write an editorial or news article about MIT housing. The article could cover the Freshmen On Campus decision, graduate housing, new dormitories, or any other aspect of MIT housing and how it affects student life. The article should be around a page (about 600-700 words).

You will have up to 45 minutes to complete your article. This is not meant to rush you, but rather to put an upper bound on the duration of the experiment. If you complete the article before the 45 minutes, get the experimenter and he will continue to the next phase. If you wish, at the end of the experiment your article will be emailed to and/or printed out so you can have a copy. Your article will be compared to articles written by others in this experiment based on a number of criteria.

You should have already received a quick tutorial with the Tech search page, Emacs RA and Margin Notes. Feel free to use these tools as much or as little as you wish in writing your article. If you have questions now or during the experiment please ask the experimenter.

Post-task Survey

1. How difficult did you find the given article-writing task (1-7)?
   Very Easy 1 2 3 4 5 6 7 **Quite difficult**

2. How much did you learn about the subject of the article during the experiment (1-7)?
   None at all 1 2 3 4 5 6 7 **A large amount**

3. Did you have enough time to adequately complete the task?
   Yes **No**

4. Please rank the usefulness of the three tools you used, in order from 1-3 (1 being most useful):

   Emacs RA: _____
   Margin Notes: _____
   Search Engine: _____

5. How would you characterize your level of expertise with web-based search engines (circle one)?

   Never used one before today
   Occasional user
   Regular user
   Expert

6. How would you characterize your level of expertise with the Emacs text-editor (circle one)?

   Never used it before today
   Occasional user
   Regular user
   Expert

7. How would you characterize your level of expertise with the Emacs RA (circle one)?

   Never used it before today
   Occasional user
   Regular user
   Expert

8. How would you characterize your level of expertise with Margin Notes (circle one)?

   Never used it before today
   Occasional user
   Regular user
   Expert

9. How distracting to your task did you find the search engine (1-7)?
   Not distracting 1 2 3 4 5 6 7 **Very Distracting**

10. How distracting to your task did you find the Emacs RA (1-7)?
   Not distracting 1 2 3 4 5 6 7 **Very Distracting**

11. How distracting to your task did you find Margin Notes (1-7)?
   Not distracting 1 2 3 4 5 6 7 **Very Distracting**

12. How useful did you find the search-engine (1-7)?
   Not useful 1 2 3 4 5 6 7 **Very Useful**

13. How useful did you find the Emacs RA (1-7)?
   Not useful 1 2 3 4 5 6 7 **Very Useful**

14. How useful did you find Margin Notes (1-7)?
   Not useful 1 2 3 4 5 6 7 **Very Useful**

15. If you were to perform a similar task, how much would you want to have the search-engine running and available (1-7)?
   Not at all 1 2 3 4 5 6 7 **I would definitely want it**

16. If you were to perform a similar task, how much would you want to have the Emacs RA running and available (1-7)?
   Not at all 1 2 3 4 5 6 7 **I would definitely want it**

17. If you were to perform a similar task, how much would you want to have Margin Notes running and available (1-7)?
   Not at all 1 2 3 4 5 6 7 **I would definitely want it**

18. How often did you find the hits given by the search engine were useful in their own right, even when you didn't follow them to see the full article (1-7)?
   Never 1 2 3 4 5 6 7 **Quite often**

19. How often did you find the suggestions given by the Emacs RA were useful in their own right, even when you didn't follow them to see the full article (1-7)?
   Never 1 2 3 4 5 6 7 **Quite often**

20. How often did you find the suggestions given by Margin Notes were useful in their own right, even when you didn't follow them to see the full article (1-7)?
   Never 1 2 3 4 5 6 7 **Quite often**

21. To what extent did you pay attention to the search engine (1-7)?
   Almost no attention 1 2 3 4 5 6 7 **Lots of attention**

22. To what extent did you pay attention to the suggestions given by the Emacs RA (1-7)?
Almost no attention 1 2 3 4 5 6 7 **Lots of attention**

23. To what extent did you pay attention to the suggestions given by Margin Notes (1-7)?
Almost no attention 1 2 3 4 5 6 7 **Lots of attention**

24. When you read an article suggested by the search engine, how often was it useful to you (1-7)?
Never 1 2 3 4 5 6 7 **Quite often**

25. When you read an article suggested by the Emacs RA, how often was it useful to you (1-7)?
Never 1 2 3 4 5 6 7 **Quite often**

26. When you read an article suggested by Margin Notes, how often was it useful to you (1-7)?
Never 1 2 3 4 5 6 7 **Quite often**

Debriefing Statement

The experiment you just participated in was designed to test the Remembrance Agent (RA), a query-free information system that automatically suggests documents that might be relevant to text a person is writing using a word processor or reading on the web. Half the subjects in this experiment were provided with a remembrance agent as well as a search engine while writing their essay. The other half were only provided with the search engine.

We are trying to discover whether using the RA encourages searching for more information and the use of more detailed information than standard search engines. We will be examining how many specific facts are mentioned or referenced in the essays from both groups, and will also be looking at how many newspaper articles are retrieved and read using the search engine and using the RA. A log has been kept of search terms used, time-stamps of queries, and particular news articles that were read using both systems. This information will be used to determine how remembrance agents might be designed and used in future applications, and to evaluate the concept of query-free information retrieval.

If at any time, now or later, you experience any ill effects (either mental or physical) as a result of your participation in this experiment, please do not hesitate to tell the experimenter, or call 253-9601 and ask for Brad.

Feel free to ask any questions about the experiment at this time.

Your help has been greatly appreciated, and will aid the Media Lab understanding how remembrance agents might be designed and applied in different environments.

If you would like to use the RA it is available from:

```
http://www.media.mit.edu/~rhodes/RA/
```

References

1.	Allport, Alan (1989) “Visual Attention,” in *Foundations of Cognitive Science*, Michael Posner (ed.), MIT Press, pp. 631-682	Page 21, 52, 52
2.	Barr, Robin (1981) “How do we focus our attention?” *American Journal of Psychology*, 94(4):591-603	Page 53
3.	Bennahum, David (1999) “Daemon Seed: Old email never dies,” *Wired*, 7.05, May 1999	Page 112
4.	Broadbent, Donald (1958) *Perception and Communication*, Pergamon Press	Page 51
5.	Budzik, Jay and Kristian Hammond (1999) “Watson: Anticipating and Contextualizing Information Needs,” in *Proceedings of the 62nd Annual Meeting of the American Society for Information Science*, Information Today, Inc.	Page 96, 119
6.	Budzik, Jay, Kristian Hammond, Larry Birnbaum, and Marko Krema (2000) “Beyond Similarity,” to appear in *Working notes of the AAAI 2000 Workshop on AI for Web Search*, Austin, TX, July 2000, AAAI Press	Page 50
7.	Cassell, Justine, Joseph Sullivan, Scott Prevost, and Elizabeth Churchill, eds. (1999) *Embodied Conversational Agents*, MIT Press.	Page 19
8.	Chakrabarti, Soumen, Byron Dom, S. Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, Andrew Tomkins, David Gibson, and Jon Kleinberg (1999) “Mining the Web’s Link Structure. *Computer*, 32(8):60-67, IEEE Computer Society	Page 46
9.	Cherry, E. Colin (1953) “Some experiments on the recognition of speech, with one and two ears” *Journal of the Acoustical Society of America*, 25:975-979	Page 51
10.	Cooper, William (1973) “On Selecting a Measure of Retrieval Effectiveness. Part 1,” in *Journal of the American Society for Information Science*, 24:87-100. Also in Sparck Jones, Karen and Peter Willett, eds. (1997) *Readings in Information Retrieval*, Morgan Kaufmann Publishers Inc, pp. 191-204	Page 45
11.	Crabtree, Barry, Stuart Soltysiak, and Marcus Thint (1998) “Adaptive Personal Agents,” in *Personal Technologies*, 2(3):141-151, Springer-Verlag London Ltd	Page 36, 103

Page 75

12. Croft, W. Bruce and David Harper (1979) “Using Probabilistic Models of Document Retrieval Without Relevance Information” Journal of *Documentation*, 35(4), 285-95. Also in Peter Willett (ed.) *Document Retrieval Systems*, Taylor Graham, London, 1988

Page 52

13. Duncan, John (1979) “Divided attention: the whole is more than the sum of its parts” *Journal of Experimental Psychology: Human Perception and Performance*. Vol. 5, No. 2:216-228

Page 123

14. Elo, Sara (1995) *PLUM: Contextualizing News for Communities Through Augmentation*, Master’s thesis, MIT Media Arts and Sciences, 1995

Page 20, 38

15. Engelbart, Douglas (1962) “Augmenting Human Intellect: a conceptual framework,” *Summary Report*, Stanford Research Institute, on Contract AF 49(638)-1024, October 1962

Page 127

16. Feiner, Steven, Blair MacIntyre, and Dorée Seligmann (1993) “Knowledge-based augmented reality,” *Communications of the ACM*, 36(7):52-62, ACM Press

Page 45

17. Frakes, William and Ricardo Baeza-Yates, eds. (1992), *Information Retrieval: Data Structures and Algorithms*, Prentice-Hall

Page 75

18. Harman, Donna (1986) “An Experimental Study of Factors Important in Document Ranking,” in *ACM Conference on Research and Development in Information Retrieval*, ACM Press, pp. 186-193

Page 75

19. Harman, Donna (1992) “Ranking Algorithms,” in *Information Retrieval: Data Structures and Algorithms*, William Frakes and Ricardo Baewa-Yates, eds. Prentice-Hall, pp. 363-392

Page 50

20. Harman, Donna (1995) “Overview of the Third Text REtrieval Conference (TREC-3),” in Donna Harman (ed.) *NIST Special Publication 500-226: Overview of the Third Text REtrieval Conference (TREC 3)*, National Institute of Standards and Technology, pp. 1-20

Page 21, 121

21. Hart, Peter and Jamey Graham (1997) “Query-free Information Retrieval,” *IEEE Expert / Intelligent Systems & Their Applications*, 12(5):32-7, IEEE Computer Society

Page 49

22. Hearst, Marti (1994) “Multi-Paragraph Segmentation of Expository Text,” in *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL ‘94)*, Las Cruces, NM, June 1994

Page 50

23. Hull, David (1998) “The TREC-7 Filtering Track: Description and Analysis” in Ellen Voorhees and Donna Harman (eds.) *NIST Special Publication 500-242: The Seventh Text REtrieval Conference (TREC 7)*, National Institute of Standards and Technology, pp. 33-56

Page 20, 38

24. Hutchins, Edwin (1995) *Cognition in the Wild*, MIT Press

Page 127

25. Ishii, Hiroshi, Craig Wisneski, Scott Brave, Andrew Dahley, Matt Gorbet, Brygg Ullmer, and Paul Yarin (1998) “ambientROOM: Integrating Ambient Media With Architectural Space,” in *Conference Proceedings on Human Factors in Computing Systems (SIGCHI’98)*, ACM Press, pp. 173-174

Page 50

26. Jansen, B., Spink, A., and Bateman, J. (1998) “Searchers, the Subjects they Search, and Sufficiency: A Study of a Large Sample of EXCITE Searches,” in *Proceedings of WebNet-98, World Conference on the WWW, Internet and Intranet*, AACE Press

27. Jarvenpaa, Sirkka (1989) “The effect of task demands and graphical format on information processing strategies,” *Management Science*, 35:285-303 Page 41

28. Joachims, Thorsten, Dayne Freitag, and Tom Mitchell (1997) “WebWatcher: A Tour Guide for the World Wide Web,” *Fifteenth International Joint Conference on Artificial Intelligence (IJCAI*’97), Vol. 1, Morgan Kaufmann Page 122

29. Johnson, S.E., P. Jourlin, K. Sparck Jones, and P.C. Woodland (2000) *Spoken Document Retrieval for TREC-8 at Cambridge University*, to appear in Ellen Voorhees and Donna Harman (eds.) *NIST Special Publication XXX-XXX: The Eighth Text REtrieval Conference (TREC 8)*, National Institute of Standards and Technology Page 85

30. Lamming, Mik, Peter Brown, Kathleen Carter, Margery Eldridge, Mike Flynn, Gifford Louie, Peter Robinson, and Abigail Sellen (1994) “The design of a human memory prosthesis” *The Computer Journal*, 37(3):153-163 Page 125

31. Lee, Joon Ho (1995) “Combining Multiple Evidence from Different Properties of Weighting Schemes,” in *ACM Special Interest Group on Information Retrieval (SIGIR’95)*, ACM Press, pp. 180-188 Page 36, 46

32. Lieberman, Henry (1997) “Autonomous Interface Agents,” in *Conference Proceedings on Human Factors in Computing Systems (SIGCHI’97)*, ACM Press, pp. 67-74 Page 121

33. Maglio, Paul, Rob Barrett, Christopher Campbell and Ted Selker (2000) “SUITOR: An Attentive Information System,” in *Proceedings of the 2000 International Conference on Intelligent User Interfaces (IUI 2000)*, ACM Press, pp. 169-176 Page 120

34. Martin, D., Adam Cheyer, and Douglas Moran (1999) “The Open Agent Architecture: A framework for building distributed software systems,” *Taylor & Francis. Applied Artificial Intelligence*, 13(1-2):91-128 Page 19

35. Miller, Robert (1968) “Response time in man-computer conversational transactions,” in *American Federation of Information Processing Societies Conference Proceedings of the Fall Joint Computer Conference*, Vol 33, Part 1, pp. 267-277 Page 40

36. Minar, Nelson, Matthew Gray, Oliver Roup, Raffi Krikorian, and Pattie Maes (1999) “Hive: Distributed agents for networking things,” in *Proceedings of ASA/MA’99, the First International Symposium on Agent Systems and Applications and Third International Symposium on Mobile Agents*, IEEE Computer Society, pp. 118-29 Page 84

37. Miyata, Yoshiro and Donald Norman (1986) “Psychological Issues in Support of Multiple Activities,” in *User Centered System Design*, Donald Norman and Stephen Draper, eds. Lawrence Erlbaum Assoc., pp. 268-284 Page 54

38. Montague, Charles E. (1930) *A Writer’s Notes on His Trade*, Chatto & Windus, 1930 Page 17

39. Moray, N. (1959) “Attention in dichotic listening: Affective cues and the influence of instructions” *Quarterly Journal of Experimental Psychology*, 11:56-60 Page 51

40. Mynatt, Elizabeth, Maribeth Back, Roy Want, Michael Baer, and Jason Ellis (1998) “Designing Audio Aura,” in *Conference Proceedings on Human Factors in Computing Systems (SIGCHI’98)*, ACM Press, pp. 566-573 Page 126

41. Newell, Allen, Paul Rosenbloom, and John Laird (1989) “Symbolic Architectures for Cognition,” in *Foundations of Cognitive Science*, Michael Posner (ed.), MIT Press, pp. 93-131 Page 20, 38

Page 38

42. Norman, Donald (1988) *The Psychology of Everyday Things*, Basic Books

Page 20, 40, 42

43. Payne, John, James Bettman, and Eric Johnson (1993) *The Adaptive Decision Maker*, Cambridge University Press

Page 85

44. Pentland, Alex, Baback Moghaddam, and Thad Starner (1994) "View-based and modular eigenspaces for face recognition," in *Proceedings 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, pp. 84-91

Page 75

45. Porter, Martin (1980) "An Algorithm For Suffix Stripping" *Program* 14(3), July 1980, pp. 130-137. Also in Sparck Jones, Karen and Peter Willett, eds. (1997) *Readings in Information Retrieval*, Morgan Kaufmann Publishers Inc, pp. 313-316.

Page 125

46. Price, Morgan, Gene Golovchinsky and Bill Schilit (1998) "Linking by Inking: Trailblazing in a Paper-like Hypertext," in *Proceedings of the Ninth ACM Conference on HyperText and Hypermedia: links, objects, time and space--structure in hypermedia systems*, ACM Press, pp. 30-39

Page 46

47. Rau, Lisa (1988) "Conceptual information extraction and retrieval from natural language input," in *Proceedings of Actes: Recherche d'Informations Assisteepar Ordinateur (RAIO '88)*, pp. 424-437. Also in Sparck Jones, Karen and Peter Willett, eds. (1997) *Readings in Information Retrieval*, Morgan Kaufmann Publishers Inc, pp. 527-533

Page 114

48. Reeves, Byron and Clifford Nass (1996) *The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places*. Cambridge University Press

Page 127

49. Rekimoto, Jun, Yuji Ayatsuka, and Kazuteru Hayashi (1998) "Augment-able Reality: Situated Communications Through Physical and Digital Spaces," in *Digest of Papers: Second International Symposium on Wearable Computers (ISWC'98)*, IEEE Computer Society, pp. 68-75

Page 25

50. Rhodes, Bradley and Thad Starner (1996) "The Remembrance Agent: A continuously running information retrieval system," in *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'96*), Practical Application Company, pp. 486-495

Page 32

51. Rhodes, Bradley (1997) "The Wearable Remembrance Agent: a system for augmented memory," in *Personal Technologies: Special Issue on Wearable Computing*, 1:218-224, Springer-Verlag London Ltd.

Page 84

52. Rhodes, Bradley, Nelson Minar, and Josh Weaver (1999) "Wearable Computing Meets Ubiquitous Computing: reaping the best of both worlds," in *The Third International Symposium on Wearable Computers (ISWC'99)*, IEEE Computer Society, pp. 141-149

Page 29

53. Rhodes, Bradley (2000) "Margin Notes: building a contextually aware associative memory," in *Proceedings of the International Conference on Intelligent User Interfaces (IUI'00)*, ACM Press, pp. 219-224

Page 45, 46

54. van Rijsbergen, Keith (1979) *Information Retrieval*, Butterworths London

Page 76

55. Robertson, S. E., S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford (1995) "Okapi at TREC-3" in Donna Harman (ed.) *NIST Special Publication 500-226: Overview of the Third Text REtrieval Conference (TREC-3)*, National Institute of Standards and Technology, pp. 109-127

56. Russo, Jay (1977) “The value of unit price information,” *Journal of Marketing Research*, 14:193-201 Page 41

57. Ryan, Nick, Jason Pascoe and David Morse (1998) “Enhanced Reality Fieldwork: the Context-aware Archaeological Assistant,” in V. Gaffney, M. van Leusen and S. Exxon, eds, *Computer Applications in Archaeology 1997*, British Archaeological Reports, Oxford 1998, Tempus Repartum. Page 124

58. Salton, Gerard, A. Wong and C.S. Yang (1975) “A Vector Space Model for Automatic Indexing,” *Communications of the ACM*, 18:613-620. Also in Sparck Jones, Karen and Peter Willett, eds. (1997) *Readings in Information Retrieval*, Morgan Kaufmann Publishers Inc, pp. 273-280 Page 46, 50, 74

59. Salton, Gerard and Christopher Buckley (1988) “Term-weighting approaches in automatic text retrieval” *Information Processing and Management*, 24, 513-523. Also in Sparck Jones, Karen and Peter Willett, eds. (1997) *Readings in Information Retrieval*, Morgan Kaufmann Publishers Inc, pp. 323-328 Page 74, 75

60. Sawhney, Nitin and Christopher Schmandt (1999) “Nomadic Radio: Scalable and Contextual Notification for Wearable Audio Messaging” in *Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems (SIGCHI’99)*, ACM Press Page 57, 126

61. Shaffer, L. H. (1975) “Multiple attention in continuous verbal tasks,” in P.M.A. Rabbit and S. Dornic, eds. *Attention and Performance V.* Academic Press, New York Page 52

62. Siever, Ellen, Stephen Spainhour, and Nathan Patwardhan (1999) *Perl in a Nutshell*, O’Reilly & Associates Page 62

63. Singhal, Amit, Chris Buckley, and Mandar Mitra (1996) “Pivoted Document Length Normalization,” in *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 21-29 Page 101

64. Sparck Jones, Karen and Peter Willett, eds. (1997) *Readings in Information Retrieval*, Morgan Kaufmann Publishers Inc. Page 45

65. Starner, Thad (1997) *Lizzy: MIT’s wearable computer design 2.0.5*. http://www.media.mit.edu/wearables/lizzy/ Page 84

66. Starner, Thad, Dana Kirsch, and Solomon Assefa (1997) “The Locust Swarm: An environmentally-powered, networkless location and messaging system,” in *Digest of Papers: The First International Symposium on Wearable Computers (ISWC’97)*, IEEE Computer Society, pp. 169-170 Page 84

67. Stroop, J. (1935) “Studies of inference in serial verbal reactions” *Journal of Experimental Psychology* 18:643-662 Page 53

68. Voorhees, Ellen and Donna Harman (1999) “Overview of the Seventh Text REtrieval Conference (TREC-7),” in Ellen Voorhees and Donna Harman (eds.) *NIST Special Publication 500-242: The Seventh Text REtrieval Conference (TREC 7)*, National Institute of Standards and Technology, pp. 1-24 Page 45, 95

69. Walker, Steven and Richard Jones (1987) “Improving Subject Retrieval in Online Catalogues.” British Library Research Paper no. 24, vol. 1. Page 75

Page 76

70. Walker, Steven, S.E. Robertson, M. Boughanem, G.J.F. Jones, and K. Sparck Jones (1998) “Okapi at TREC-6: Automatic ad hoc, VLC, routing, filtering and QSDR” in Donna Harman (ed.) *NIST Special Publication 500-240: The Sixth Text REtrieval Conference (TREC-6)*, National Institute of Standards and Technology, pp. 109-127

Page 84

71. Want, Roy, Andy Hopper, Veronica Falcao, and Jonathon Gibbons (1992) “The Active Badge Location System,” *ACM Transactions on Information Systems*, 10(1):91-102

Page 127

72. Weiser, Mark (1993) “Some Computer Science Issues in Ubiquitous Computing.” *Communications of the ACM*, 36(7):75-84, ACM Press

Page 21, 52, 53, 54

73. Wickens, Christopher (1992) *Engineering Psychology and Human Performance, Second Edition*, HarperCollins Publishers

Page 41

74. Woods, David, Leila Johannesen, Richard Cook, and Nadine Sarter (1994) *Behind Human Error: Cognitive Systems, Computers, and Hindsight.* CSERIAC SOAR report 94-01

Page 39, 40

75. Zipf, George Kingsley (1949) *Human Behavior and the Principle of Least Effort; an introduction to human ecology,* Addison-Wesley Press

Index