Haystack: Per-User Information Environments

David Karger and Lynn Andrea Stein
Laboratory for Computer Science
Massachusetts Institute of Technology
545 Technology Square
Cambridge, MA 02139
Voice: (617) 258-6167 or 2663
Fax: (617) 258-8682
karger@theory.lcs.mit.edu
las@ai.mit.edu

A Innovative Claims.

The Haystack Project aims to create a community of individual but interacting "haystacks": desktop-integrated personal information repositories which archive not only base content but also user-specific meta-information, enabling them to adapt to the particular needs of their users.

- Every information transaction will be brokered by a user's haystack. All information manipulated by a user (files, mail messages, web pages) will become part of his personal collection without user overhead. Information in a user's haystack is accessible to the user at any time through augmented facilities integrated into normal desktop tools (such as an editor or a web browser). A user's haystack will be wholly integrated into his desktop. For example, a haystack user could "find file" in an editor using keyword search on remembered characteristics of the document rather than a precise description of the document's location in the file system.

- Unlike traditional information retrieval systems, Haystack will customize itself to each user. Haystack adapts over time to the behavior and vocabulary of its user, modifying its retrieval algorithms to reflect improved understanding of how its user describes information and information needs, and what kind of documents are best for addressing the user's needs.

- As individuals build up their collections and information retrieval systems, haystack customization will form a lens through which they can view and search for information located in others' collections. This requires bridging multiple views of the information space.

- Different Haystacks will collaborate with each other, learning over time which users have related interests and helping to form "referral systems" that take a user's question to other haystacks that can answer it.

- In addition, each haystack has a pro-active archiver process that actively seeks to keep its collection up-to-date, adding information that might be useful to the user in the future.

- Standard information retrieval evaluation tools are inadequate to assess dynamic interactive systems like the one that we propose to build. We will provide innovative next-generation evaluation tools for use in assessing Haystack and future systems.

By integrating adaptive, collaborative information retrieval into every aspect of a user's interactions with a computer system, Haystack aims to become the quintessential information assistant.

B Deliverables.

This project will result in software for individual inter-linked information spaces ("haystacks") running on a variety of computational platforms and integrated with user environments.

The major deliverables of this project will be as follows:

1. **Per-User Information Environments**

    - A user's haystack will be integrated into his desktop environment and proactively archive information accessed during that user's normal activities.
    - Each haystack will integrate a learning module which will adaptively extrapolate from its user's prior information interactions to provide better subsequent performance.
    - A haystack will also provide lightweight annotation tools integrated into other aspects of information access, lowering the costs of active user customization.
    - Each haystack has an "archivist": a process that is responsible for keeping its information collection up-to-date. The archivist continually scans its own collection as well as the external information space in order to update archived information. For example, when a previously visited web page changes, the new version will be added to the user's haystack and the user optionally notified.
    - Optionally, haystacks will allow archiving and retrieval of multi-modal and/or multi-media information such as the MIT MURI Visual Database project. [Option 1]
    - Optionally, haystacks will be integrated into the SodaBot software agent environment so that they can be deployed as part of agent architectures (e.g., the MIT AI Lab's Intelligent Room). [Option 2]

2. **Facilities for Collaborative Information Access**

    - Haystacks will provide multiple security levels to facilitate both private and shared information collections.
    - Each haystack will use its optimized/customized retrieval patterns to serve as a lens through which its user can view information contained in the public portions of other user's haystacks.
    - The Haystack system will provide tools for exchange of information between heterogeneous haystacks. These tools will mask the fact that the underlying IR systems have different interfaces and capabilities as well as the fact that different users organize their information differently.
    - Haystack will also provide tools for social organization, e.g., identification of colleagues with similar interests or of experts on a particular topic, based on information exposed in a user's public haystack.

3. **Middleware for Integrated Information Retrieval Systems**

    Crucial to the construction of the Haystack system is the specification and implementation of a middleware layer that bridges existing off-the-shelf information retrieval systems and makes them accessible to both off-the-shelf and custom user interfaces.

- The MIRS layer of Haystack integrates multiple off-the-shelf information retrieval back ends with multiple off-the-shelf and custom user interface front ends. Additional IR and UI systems can be added as plug-ins.
- The MIRS supports plugging in of additional information management systems such as text retrieval systems or relational databases.
- The MIRS supports integration of Haystack into additional user applications (such as new mail readers or editors) and custom-designed user interface front ends.
- The MIRS enables interchange of queries and retrieved information among a community of Haystacks, even though different users may have different data formats or personalized semantics.

4. **Next-Generation Evaluation Tools**

   Haystack provides a deeper integration of information retrieval than existing technologies. This project will also produce a set of deeper assessment tools — Corpora of User Interaction Patterns, or CUIPs — by which systems of its kind can be evaluated. These CUIPs include actual user histories of gradually (and abruptly) refined queries as well as traces of successive attempts to retrieve the same or similar information from an information space. No such corpus of extended interaction patterns has ever been collected and made publicly available. We anticipate that such interactive corpora will be crucial to the assessment of the next generation of information access tools.

5. **Analysis of Results and Dissemination**

   Additional results include:

   - Technical reports and papers documenting our progress and assessing the system's effectiveness and feasibility.
   - All systems and source code.
   - Corpora of User Interaction Patterns (CUIPs): publicly available records of usage of the system by large numbers of individuals over time.
   - Periodic project summaries and reviews.
   - Collaboration with other DARPA groups, e.g., the MIT AI Lab's MURI ("Trainable Modular Vision System") and HCI ("Intelligent Room") projects.

6. Additional details on Options 1 and 2 may be found in section D.5.

All work performed at MIT will be free from proprietary rights claims of any outside party and will be bound solely by MIT's policies on intellectual rights, which are consistent with federal guidelines as applied to universities. MIT typically retains ownership of rights to software and data, but grants royalty-free license to the government for use of software and data developed with federal funds.

C Statement of Work.

The proposed research involves the construction of individualized inter-linked information spaces integrated into the user's desktop environment.

Year 1

In the first year of the project, we will focus on the construction of the basic single-user haystack system. In particular:

- A Haystack will be well-integrated into the user's environment. Because our initial test audience is unix-based, we will focus on integrating the most common tools from this environment first. However, our architecture is modular, supporting both back- and front-end plug-ins, and the system is explicitly designed to be portable.
- We will establish a meta-data format and use this to archive text-based documents as well as basic non-document structures (e.g., groups of documents, directories, mailboxes, etc.)
- We will build a basic MIRS component, the middleware layer of our system which bridges between different IR system back ends and end-user application plug-ins.
- We will provide a basic applications programmer interface (API) and plug-ins for several common unix tools.
- We will design a graphical user interface based on off-the-shelf technology back-ended to our MIRS middleware.
- We will design a basic annotation editor to facilitate manual annotation of documents. Use of this component will not be required for normal haystack functionality, but we will exploit it for debugging and testing purposes as well as user studies.
- We will build basic automated annotation systems for the most common kinds of documents to be archived in our haystack, e.g., mail messages, web pages, postscript documents.
- We will support queries against one or more term-based information retrieval systems (plugged into the MIRS).

At the end of the first year, we will demonstrate a stand-alone haystack running on multiple platforms integrated with editing, web browsing, and electronic mail/news reading functions.

Year 2

During the second year, we will focus on the haystack's ability to learn from its user's behavior and to customize itself to better support its user.

- We will augment the haystack with logging facilities, making use of the central role of meta-data to add "query records" as first class objects in our repositories. These will provide the basic data for our learning algorithms as well as the foundation for our Corpora of User Interaction Patterns (CUIPs).

- Haystack will adaptively improve its response to user queries. To facilitate this, Haystack will monitor its user's activity patterns, learning which documents the user chooses to devote attention to. Query records will provide the basis for modified retrieval behavior, so that ultimately only those documents found useful will be initially returned to a common query, and novel queries will be answered with higher precision and recall than earlier in the haystack's history.

- We will build an automated archivist process which will continually work to update information in a user's haystack. The process will also use idle time to attempt to further improve its responsiveness to its individual users' information needs.

- We will integrate security mechanisms into haystack, making it possible for a user to expose portions of his haystack with various levels of protection. This will begin the distinction between public and private information spaces.

- We will provide basic abilities to search against other user's haystacks.

- We will integrate Haystack with additional back ends, including a data base to support structured queries and multiple simultaneous information retrieval systems (e.g. boolean, vector-space, and probabilistic systems). Using the MIRS middleware, we will adaptively improve our response to user queries by trading off the strengths and weaknesses of different back ends.

- We will add additional front ends in the form of new desktop applications, including additional hardware platforms or software environments. We will improve the existing interfaces based on experimentation during the first year.

- We will provide improved result browsing mechanisms better integrated with desktop applications. Design of these features will be based on user studies conducted after the first year of the project.

- We will deploy Haystack widely at the Laboratory for Computer Science, enabling a medium-scale user study.

- We will record preliminary sets of CUIPs, enabling a first round of testing of the haystack system.

At the end of the second year, we will demonstrate an adaptive haystack customizing its behavior to better support its user. One haystack will be able to query another. Haystacks will support proactive updating of their archived corpora.

Year 3

In the final year of the project, we will focus on inter-haystack collaboration, system refinement and technology transfer.

- We will improve the adaptive capacity of the haystack, using actual traces of user interactions (our CUIPs) to augment and refine our learning algorithms.

- We will build upon the preliminary sharing mechanisms introduced in year 2 to make haystacks truly cooperative interacting information repositories. We will utilize the learning algorithms developed in year 2 to facilitate characterization of haystacks as more or less valuable to a given individual based on vocabulary, trustworthiness, and particular information need. This includes automated referral among a community of interacting haystacks.
- We will augment existing sharing mechanisms with additional utilization of a haystack's own user customization. This will let a haystack serve as a lens through which to view others' collections.
- We will augment the security facilities of haystack, improving the ability of the system to share information and facilitate collaboration across haystacks.
- We will fully deploy haystack in the Athena academic computing environment at MIT. This will facilitate collection of CUIP data as well as rigorous testing of the system by a large set of naive users.
- We will produce a significant Corpus of User Interaction Data suitable for use by DARPA in its evaluation of next-generation information interaction systems such as Haystack.

At the end of the third year, we will have produced both integrated collaborative customized information environments and a set of testing protocols useful for testing such temporally extended interactive adaptive systems.

Options

Besides the main offerings in this proposal, we include two options that augment the main body of this work. A statement of work on these options, together with their expected results, may be found in section D.5. Budgets for these options are included in section L.

D Results.

We expect the a number of important results from this work, including:

1. The implementation and demonstration of individualized information environments.
2. The development of tools for information sharing at both the user and the system level, and their implementation as a middleware layer.
3. The development and dissemination of next-generation evaluation tools for information access.
4. The transfer of this technology to DARPA and to the commercial sector.

D.1 Integrated Individualized Information Environments.

The driving force behind the Haystack system is the idea of providing the average user with a means to organize the information he accumulates over time. This information could be stored locally on the user's hard drive (local), somewhere on a server (i.e the WWW), or even as a paper copy. Haystack will provide the mechanisms necessary to retrieve this information from wherever it is stored and index it locally within the user's Haystack workspace.

Haystack will be completely integrated into its users' normal desktop activities. Individuals will thus be able to use the information retrieval process to organize their own personal collections of information – including implicit information a user may not be aware of accessing – according to a variety of personal needs. All information manipulated by a user (files, mail messages, web pages) will become part of his personal collection without user overhead. Haystack will also incorporate proactive features that seek to maintain the user's collection of information much as a professional archivist updates and adds to the body of material contained in the archives. Retrieving archived information will be analogous to simply opening a file; the intervening interaction with haystack will simply facilitate recovery of ill- or under-specified file descriptors.

The information archived by Haystack will include not only base document content but user- and system-generated meta-data. Haystack will record all extractable information about a document — where and how it was encountered, in what context, how it was treated (e.g., was it mailed to a colleague?), etc. — as well as authorship, related documents, etc. In addition, the user may exploit lightweight annotation tools at archive or retrieval time to augment these automatic annotations. Further interactions between the user and this document — including, but not limited to, subsequent retrievals through his Haystack — will augment the existing annotations. Haystack will, for example, exploit previous queries to reorganize and optimize later queries and to learn from user preferences how better to locate relevant information in specific contexts.

Among the interactions that Haystack will record and exploit include many that can only be captured by virtue of the system's tight integration with the user's desktop. For example, which pointers did the user actually decide to follow? Did he make a copy? Did it inspire him to send an e-mail message? How are successive queries being refined? Can we get the user to say what he liked? Simply by interacting with information, the user generates archived information about the document, enriching its indexed semantics. This collection of document-specific meta-data will form the basis of both Haystack's contextual retrieval system and its ability to adapt to its individual user's vocabulary, style, and needs. Over time, a user's haystack can be expected to evolve into a more effective system.

D.2 Tools for Information Sharing.

Each haystack provides a personal view of information-space. We will also support sharing of information between and among haystacks. This means that all of the benefits of individual personalized information environments will accrue to one user's search of another's space. A haystack will exploit its knowledge of its own user to view foreign information.

By sharing these views among haystacks, Haystack will also facilitate social organization and automated recognition of "experts", "close colleagues," etc. Since a haystack represents its user's personal perspective on information space, the public portions of that haystack can be used to characterize the user's (or at least the haystack's) ability to answer certain questions and otherwise engage in collaborative abilities.

All of the interchange of public information among haystacks — i.e., with the exception of privacy/security issues — will happen automatically and without requiring explicit intervention of the user. Information from multiple public resources will simply be shared and interpreted by a user's personal haystack as though it were local. As a consequence, the "improvements" made to one haystack by its user will also accrue to all other users of the system.

D.3 New Evaluation Tools.

Current evaluation tools for information retrieval systems emphasize the power of the system to match a single query. Criteria such as precision and recall are applied to the results of a single search by the information retrieval system, and the one scoring highest on these metrics "wins". Often — as in TREC [9] — the queries themselves are designed by highly sophisticated users and are of significant complexity.

Realistically, this is not how most users interact with information retrieval systems. Even relatively generic and non-personalized systems such as web search engines or library catalogs often require multiple successive queries to refine the retrieved document set. Sometimes, these refinements are intended to narrow the set of documents; at others, the user changes his search as information from earlier searches affects his understanding of what he seeks.

Current evaluation metrics do little to advance our understanding of a system's ability to facilitate such interactive searches. We propose instead to develop Corpora of User Interaction Patterns (CUIPs) as an evaluation tool for Haystack. A CUIP documents an entire interaction sequence between a user and his information space. Rather than a static snapshots of a system's performance, a CUIP provides a movie of the system's dynamic behavior. This allows more realistic assessment of information retrieval systems in actual usage contexts.

As we move to a world in which information environments are increasingly personalized and customized, our reliance on tools like CUIPs will inevitably increase. For example, Haystack's adaptive capacity — responding more quickly and accurately to a later query when a similar one has already been handled — cannot be adequately assessed using a static metric. This would be like running an untrained speech recognition system in the CSR competition, or at best like running a system trained to another speaker on a single novel sentence.

We intend to evaluate our system using CUIPs that we have collected. In addition, we will use Haystack with a variety of IR back ends (plugged in through our MIRS middleware layer) to run head-to-head competitions on our CUIPs. Our initial deployment of Haystack — beyond our research group, where it will be in continuous use — will be in the Laboratory for Computer

Science and the Artificial Intelligence Laboratory at MIT. These environments provide a potential community of several hundred active researchers working in a heavily computational environment. Beyond this user community, we intend to make Haystack available on MIT's Athena academic computing environment, which is used by most of the several thousand students at MIT. One of the PIs is also working closely with MIT's Library system, which receives significant on-line usage. Within these environments, we anticipate running both formal and informal evaluations of Haystack as well as a variety of plug-in back ends; this should provide both a control case for evaluating the utility of CUIPs and a significant assessment of Haystack itself.

D.4 Technology Transfer.

As with most academic projects, one of the tangible results of the endeavor will be published papers, which contribute to technology transfer by informing the rest of the research community, and students trained under the project, who contribute to technology transfer by physically moving to industrial or government labs upon graduation.

For a project such as this one, though, it is equally important that the software created be injected into the industrial sector. Because MIT is not a commercial venture, we do not expect direct products to follow from this proposal. At the same time, it is clear that systems of this type are of great interest to both the military and the commercial worlds, and we are eager to see the transfer of our results to actual products and applications.

The software developed for this project will be directly transferable to DARPA sites running appropriate hardware and software systems. Plans will be made together with the cognizant program officers for transfer to alternate platforms. The system will be used by project personnel from the outset and should be directly usable in similar contexts (e.g., DARPA offices). We will work with DARPA personnel to arrange for transfer to and integration into other military venues.

The evaluation tools and techniques that we develop in this project should also be of significant interest to the DARPA community. We plan to make our CUIPs (Corpora of User Interaction Patterns) available to the DARPA community. They should be of utility both for testing systems and for training and testing stand-alone query refinement algorithms. In addition, we will provide the tools necessary for DARPA to construct its own domain- and task-specific Interaction Pattern Corpora for use in any subsequent DARPA evaluations.

The technology described here is also of significant civilian interest. The PIs both have working relationships with corporate entities such as Xerox PARC, Lotus Development Corporation, and Sun Microsystems. There are also a host of small startups that have spun off of the MIT Laboratory for Computer Science and the Artificial Intelligence Lab. (Indeed, Professor Stein's group has produced one such startup in the last year.) In addition to these potential avenues of transfer, we intend to work with DARPA and/or appropriate governmental or commercial agencies, as well as with our own partner organizations (e.g., the World-Wide Web Consortium) to facilitate the transfer of this technology into the commercial sector.

D.5 Options

Option 1: Integration of Multi-Modal Information Capacity

As an option, we will add facilities to search static or video image databases using either an off-the-shelf package or systems developed under other DARPA programs, such as Lipson's Configural Recognition, Gifford's Computational Video, etc. Under this option, we would integrate both a back end for image querying plus specialized user interface tools to allow full haystack functionality with this radically different type of information. For example, we would integrate an image browsing query front end such as Lipson's (from the MIT AI Lab MURI "Trainable Modular Vision System"). These tools would necessitate a significant extension to our middleware layer, generalizing its information handling capacity beyond the textual. The resulting system would provide a fully integrated (still or video) image retrieval system customized to user preferences and interacting with other interconnected information spaces just as the base proposal would do for textual information.

Option 2: Integration of Software Agents

As an option, we will integrate Haystack with software agents. Under this option, we would augment the basic haystack system with an interface to the SodaBot software agent environment (designed by Prof. Stein's group). This would enable

- Connection of haystack to other heterogeneous networked agent systems. SodaBot provides transparent communication with other software agents.
- Connectivity with alternate communication modalities and information sources. For example, SodaBot supports user agents capable of contacting a user by telephone, fax, electronic mail, etc., as may be most appropriate at any particular time.
- Haystack agents, enhancing Haystack's ability to interact with additional facilities such as electronic commerce applications or fax machines. Individual haystacks equipped with their own agents will encapsulate additional information about users (e.g., phone and fax numbers together with facility to contact users through these media), and otherwise orchestrate non-information-access interactions.
- Integration of Haystack, through these agents, with the MIT AI Lab's Intelligent Room (DARPA HCI) project. The Intelligent Room is a fully-instrumented room capable of sensing, interpreting, and augmenting activity within it. Integration of Haystack would allow "computerless" control of a user's information environment by voice or gesture, storage of non-textual objects (though under this option their retrieval would be limited to textual descriptions), and other multi-modal interactions.

E Disclosure

MIT is not providing any support services to any DARPA technical offices through an active contract or subcontract.

F Milestones.

Y1 Stand-alone Haystack running on multiple platforms integrated with editing, web browsing, and electronic mail/news reading functions. Base: $318,611

Annotation system and annotation editor.

Queries against term-based information retrieval system and structured queries.

Rudimentary result browsing.

Option 1: Design of MIRS plug-ins for visual data query language. $87,805

Option 2: Design of agent API, preliminary integration with SodaBot. $68,482

Y2 Learning and sharing of information spaces. Base: $266,904

Integration of basic learning functionality and adaptive queries.

Security and distinction between public and private information spaces.

Search against other users' haystacks.

Better system integration into user's environment.

Additional platform(s).

Additional underlying storage mechanisms and integrated query abilities.

Improved result browsing.

Preliminary CUIPs.

Cross-comparison study of single back ends.

Option 1: Extension of browsing tool to visual data query language. $67,568

Option 2: Total integration of haystack into agent environment, e.g., Intelligent Room. $70,843

Y3 Learning, refinement and tech transfer. Base: $278,432

Improved learning functionality.

Multiple levels of security and privacy.

Automated referral among a community of interacting haystacks.

Additional searching available.

Discovery of social structure.

Production of CUIPs.

Evaluation of Haystack.

Option 1: Integration of additional modality/media (e.g., video or audio). $69,703

Option 2: Construction and integration of domain-specific haystacks for agent environment, e.g., VCR haystack. Demonstration of hands-off Haystack control. $69,703

G Technical Rationale.

G.1 Motivation

A great deal of research in information retrieval has been detached from the users who could eventually benefit from such research. Typical information retrieval systems assume a large corpus which is queried by numerous clients. While this is clearly an important model, we contend that it is by no means the only one worth studying. The old-fashioned analogue to querying such a system is a trip to the library: a repository of massive amounts of information that clearly contains (somewhere) what we are looking for.

In practice, the library is the last stop on a person's hunt for information. When a person looks for information, he will often start with his own bookshelf. This "personal repository" contains a collection of information, built up over time, that reflects the needs and knowledge of its owner. The owner can find information in it more quickly and can be more confident of understanding what he finds there than in a library. More precisely, the bookshelf exhibits the following differences from the library:

- Everything in the bookshelf was voluntarily placed there by the user. This suggests that the user considers the information in the book useful for answering at least some questions. In contrast, any one individual will consider a large portion of what is present in a library to be completely irrelevant to his needs.

- The user has probably looked most books on his bookshelf before. He is likely to remember with certainty that particular information is present somewhere in the bookshelf, even if he cannot be sure where it is. In contrast, a trip to the library often reflects a search for information that might not exist.

- Users are well acquainted with the content of their books. Once they've been pointed at one, they won't need a lot of time to understand it. Sometimes users even pencil notes into the margins of their books. These notes can remind them of their state of mind when they were last reading the book. In contrast, a library user is often encountering a book for the first time and must therefore expend a significant amount of effort figuring out what it means.

- Any item in a person's bookshelf is likely to be one that a person expects to refer to frequently. They thus have a greater interest in making the information easily accessible "next time" than they do when looking at a library book that they might never see again.

- An individual's bookshelf is organized in an idiosyncratic fashion. While library materials are arranged according to a standardized classification scheme, individuals have been known to arrange their books by topic, chronology, usage pattern, or even size and color. Even users who make no active attempt to organize their books find them structured in some kind of most-recently-used hierarchy.

This contrast between libraries and bookshelves highlights the ways in which bookshelves become our own personal spaces. When we go to a library, we are typically seeking new information, looking for something for the first time. In contrast, bookshelves focus on the "RE" in information REtrieval—the majority of searches through a bookshelf are for material that we have looked for before. After all, we must find an item at least once in order to put it on our bookshelf. Any searches for something already on the bookshelf are for something we've already found once.

If a person's bookshelf fails him, he still has an alternative to the library. The natural next step is to ask his colleagues in neighboring offices. Turning to a colleague offers several advantages over a trip to the library.

- Colleagues have their own personalized collections of information which they can search effectively.
- Colleagues share interests and vocabulary with the original questioner, and are thus likely to be able to understand that person's information needs and effectively communicate anything they might know that can help. A person can describe his problem in a language common to him and his colleague, and his colleague can then use her own knowledge of her collection to find what the original searcher wants.
- At the same time, colleagues know things the individual does not. Access to them amplifies the individual's knowledge.
- Even if colleagues are unavailable, books in their personal collections are more "trustworthy" than random books selected from a library. Their presence in the colleague's collection indicates that someone we trust considers them valuable.

G.1.1 Current IR Systems

Current IR systems, including the widely used "web search sites" such as Alta Vista, Lycos, and Infoseek, operate on a corpus which is dynamic, but use retrieval methods that are static. Regardless of who submits a query, and regardless of how often the query is performed, the information returned will be the same (modulo changes in the underlying corpus). Some systems such as Alta Vista will exhibit a change in *efficiency*, since they perform optimizations such as caching the results for common queries, but they will not exhibit a change in returned results. Even relevance feedback, a scheme that asks users to modify their search in the context of the initial results, is essentially static: the same feedback operations will always yield the same new results.

Such a "one size fits all" approach to query answering is clearly not taking advantage of useful information. A query about "polish" could probably be answered better if we knew whether the person asking was a metalworker or a linguist. A question about "apples" should return different responses to a computer dealer than to a farmer. A technically sophisticated discussion of the latest work on genetics might be the right answer to a question about evolution if the questioner is a skilled researcher, but not if he is an elementary school student. Some of the problems can be "fixed" by further interaction between the user and the system, but it would clearly be more efficient to avoid such extra iterations—especially if they must be repeated for every search.

Another feature that current search systems do not take full advantage of is the reaction of users to the results they are given. If users were able to indicate that a certain document was highly relevant or irrelevant to a given query, then an IR system could raise the subsequent profile of documents frequently flagged as good while lowering the profile of documents flagged as irrelevant. Such a scheme can be made even more effective if the IR system also pays attention to the profile of its user. A document may be highly relevant to one "type" of person while being useless to another.

Thus, the "flagging" of documents of high quality should be dependent on the type of individual performing the query.

G.1.2 Our goal in a nutshell

The Haystack project is an attempt to bring some of the benefits of personalization to electronic information retrieval. We want to give each individual the tools needed to maintain their own "haystack"—a corpus of information that is personalized both in content and in organization. Such a system should adapt to the behavioral patterns of its user, learning over time what its user means by a particular query and what corpus objects the user will find most satisfying. Part of this involves recording explicit preferences of the user. Much more involves behind-the-scenes work: metaphorically looking over the user's shoulder, collecting information on the user's behalf and watching what he does in order to adapt future haystack behavior. We give some details of these ideas in Section G.2.1.

Once we have a system that helps individuals to organize their information, we aim to leverage this organization by exposing as much data as possible from each individual's haystack to other haystacks. Any improvement in information quality or organization achieved by one individual can then be used to advantage by all other participants. Recalling our bookshelf metaphor, we aim to provide tool that allow for the identification and interrogation of "kindred spirits" who collect and organize information an individual will find useful. Our plans in this area are discussed in Section G.2.2.

In order to explore our ideas, it is crucial that our system actually be used. Especially to explore the benefits of collaboration, we need to establish a large user base. We therefore aim to design a tool that is sufficiently robust and helpful that people will actually use it. Since we expect that even a minimal system will be of use to people aiming to organize their mail, file system, and favorite web pages, we expect to attract a moderate-size community of users at MIT. Our first deployment target will be to the several hundred researchers of the MIT Laboratory for computer science; we then home to entice involvement by the several thousand MIT undergraduates who are a part of MIT's Athena project of networked workstations.

Besides providing a testbed to explore our ideas about collaborative and personal information management, we expect the "trace" of our deployed system's activity to provide a corpus of useful information about how people actually use IR systems. We will develop data gathering tools that (with permission) keep track of the kinds of queries people perform, the kinds of answers they get, and what they do in response. Such a query corpus has not been readily available in the past, and we expect that its existence will create an opportunity for new kinds of user studies. This is discussed further in Section G.2.4.

G.2 Our Approach

We plan to design a system that takes advantage of the opportunities inherent in running on a person's desktop. These include close contact with its user, close contact with its user's important information, and a dedicated computational resource. We believe that such a system will let us address several questions:

- How can individuals use an information retrieval system to organize their own personal collections of information? How might an information retrieval system learn from its users and evolve over time into a more effective system?

- As individuals build up their own collections and information retrieval systems, how can they search for information that might be located in others' collections, especially when such information is organized by information retrieval systems that may differ greatly from their own?
- How can we take advantage of the extra computational resources a distributed information system has over a centralized one?
- What kinds of assessment tools can be used to evaluate the performance of such an integrated adaptive information environment?

G.2.1 Personalization

A major advantage of a personal system is that it lets the user archive data he cares about. It is ironic that it is presently easier to search the entire space of web documents than it is to find a message in one's own mail archive. Providing retrieval tools for a user's own content will provide the maximum incentive for making use of them. This in turn will maximize the amount of information going into the system, which will maximize its effectiveness to the user (and, we argue later, to everyone else). Here we describe three ways to exploit the kind of tight integration available in a personal IR system: passive observation of relevance, passive observation of the query process, and active annotation of documents.

Passive Observation of Relevance

An important benefit of running an IR system locally is in the amount of information it can gather locally *without* a user's active participation. Haystack can use such information to "learn" from user activity and improve its retrieval performance over time.

Perhaps the most obvious bit of information is that a given document that was suggested by the IR system as a potential query result was actually retrieved. If the retrieval is by means of a point-and-click on a web page, then even an internet search engine could take note of it (though none currently do). However, a local system can also notice other informative facts. For example, upon seeing a set of search results, a user might:

- load one of the documents into his web browser, scroll through it, and follow links from it
- edit one of the documents
- mail one of the retrieved documents (or its URL) to a friend
- send mail to someone mentioned in one of the retrieved documents

In fact, a given result set might even motivate the user (by reminding them of something) to retrieve a document that is *not* in the result set. These actions are good indications that the chosen document is relevant to the given query. Such activities would be very hard for an internet IR system to detect.

A basic user activity that Haystack will want to detect is "attention" to a particular document. The actions described above all indicate that the user is giving a certain document his attention.

Detecting attention is not a trivial task. For example, if a document sits for a long time in a browser window, is it because the user is carefully reading it, or because he has gone for coffee? The more tightly Haystack is integrated into the system, the better it will be able to distinguish true from apparent attention. For example, even Alta Vista Could detect whether a user follows a link to a particular page. A local web browser could detect whether the user adds the page to his bookmark file. It could conceivably also detect how much time elapses until the user loads another page. With even tighter integration, it could notice that the user is scrolling through the document, a stronger indicator of attention than if the document is sitting passively in the browser. It could also detect whether the user has switched to a different application, which would indicate that the user is no longer paying attention to the document. This motivates fully integrating the system into the desktop instead of just building a nice but independent user interface.

Tools for detecting attention can be used to determine document relevance. If a user enters a query and then pays attention to a particular document, this is a good indicator that the document is very relevant (in the user's mind) to that query. It is also an indicator that documents which were highly ranked in the retrieval but were not given attention are bad matches to that query. For similar reasons, a document that receives only brief attention and is then replaced by another document is probably not as relevant as the new document.

Several tools from the learning community can be applied effectively to this problem. There has already been a great deal of study of how learning algorithms can be applied to information retrieval systems. In particular, learning algorithms have often been used to train information filtering systems that select "interesting" documents from a stream of newly arriving documents. While this ignores the issue of queries seeking documents on a particular topic, we believe such tools can be adapted to train a system on how to answer queries based on observed reactions to previous answers to similar queries.

Passive Observation of the Query Process

Another kind of observation which will be helpful is the detection of query evolution. A typical interaction between a user and an IR system involves multiple iterations: the user types a query, receives a response, decides that the query needs to be modified, receives a new response, and so on. If we can "track" the evolution of such a query, we have important information for next time. The original query is the one that, in the user's opinion, describes his information need. The final query that (hopefully) retrieves relevant documents is the query that the IR system opines to describe the relevant documents. This indicates a difference of opinion over language that can be usefully recorded for the future. If the user ever types his original query, or one similar to it, again, Haystack can use the information it has to modify that query into one that the IR system will properly interpret to retrieve the documents the user actually wants. The system might annotate a document with terms that do not appear in it but that the user types when he expects to find that document. Or, it might add a "query record" for the original query that describes the evolution of that query into the one the user wants.

We plan to develop tools that track this kind of query refinement process. An important problem is to detect when a user gives up on one query and starts on another. A natural heuristic is to notice whether the new query contains words overlapping with the old. A more sophisticated one is to notice whether there are overlaps in the document sets the queries retrieve. Chronology of queries is also relevant.

Active Annotation of Documents and Queries

A user archiving his own information expects to have to find much of that information later. He would therefore have an interest in making it easier to find next time. I, for example, would be willing to annotate some of my own documents with keywords or mnemonic phrases according to their subject. I would be much less willing to take the time to annotate the results of an Alta Vista search, since I doubt that I will need to perform that particular search again (if I did expect to need that information again, I would probably download relevant parts into my own personal collection). Although this is a process motivated by self-interest, it will also benefit others, since we plan to allow one user's annotations to be exported from his haystack to the benefit of other users (see Section G.2.2).

While we can never rely on a user being willing to take them time to do this, we can develop tools that make it as easy as possible so that it is done often. One approach is the following. A new query will cause the generation of a "query record" that initially lists the documents the IR system believes relevant to the query. The user might then delete certain documents from the query record and add others. This record can be used by Haystack to adjust results for that or similar queries: items that match but which were actively excluded from the query record should have their relevance downgraded. Items that were actually examined for a long period of time should have their relevance ranking raised. The query record can also be used as a pre-defined set of documents for implementing the "relevance feedback" scheme popular in IR systems [7, Chapter 11]: the terms in the documents of the query record are likely to define a query that will be good for finding other, new documents that are also relevant to the query. This also impacts on the problem of information filtering, which we will touch upon later.

G.2.2 Collaboration

We can amplify all of the benefits described above if we allow individual haystacks to interact. The annotations and retrieval decisions that one user makes can be of great value to other users (particularly colleagues with similar tastes and vocabularies). The mere fact that a given document is in many people's haystacks, even unannotated, suggests that it is a document of high intrinsic value—we might prefer it over documents that are apparently better matches to the query but are less popular. We propose to develop a system that maximizes the export of such information. We aim to export information about which documents are in which user's haystacks, what annotations users have made to the documents, and what documents users have selected as being relevant to particular queries. Consider that the typical way to search for a paper book is to ask one's office-neighbor for it. Analogously, we would like to let individuals search for information in other people's haystacks. Both to limit the costs of a search and to improve the filtering of what is returned, it is important for the system to learn over time which other individuals are most likely to have information that a given user finds relevant—these haystack "neighbors" are the systems that should be queried first and whose results should be most trusted.

The combination of personalization and collaboration creates a useful alignment between individual's supplies of and demands for information. Past work on collaborative filtering (see below) has expected individuals to behave altruistically, spending time ranking documents after they have found them, when they will no longer accrue benefit from doing so. Haystack, in contrast, asks people to perform only the information organization activities that will help *them*, and then exports the outcome of those activities to others. Haystack fits the modern economic model of individuals behaving selfishly and this contributing to the general good.

We also feel that our haystack approach will encourage even more dissemination of information than is presently undertaken. Right now, there is overhead in publishing one's information: the publisher must take active steps to present it in such a way that it is accessible. This requires that the publisher properly organize the information to allow others to find it. With Haystack available to take care of the organization, publishing a document will be as easy as deciding what to make public.

Another opportunity that the linking of haystacks creates is in connecting individuals to other people who can address their information need. The information I have stored in my haystack is likely a good indicator of my knowledge and interests. A question that matches a lot of material in my haystack is likely to be a question I can usefully answer. The haystack system can therefore serve as an "information brokerage" connecting questioners to experts.

Among the issues which must obviously be addressed if such interaction is to take place is that of privacy. We believe that a lot can be accomplished simply by creating a distinction between "private" data available only to its owner and "public" data available without restriction. Even this simple scheme will provide significant opportunities: the speed with which many people have published information on the web indicates that there is a great deal of information that people are happy to make available without restriction. Certainly the academic research community and advertisers are eager for their information to be disseminated as widely as possible.

In the long term we plan to develop a more general protection scheme that allows one to specify which users may specify who can access various data elements. A great deal of work has been done in this area and we plan to use "off-the-shelf" protection schemes rather than developing our own. However, one particular problem that must be dealt with is our desire to make haystack as transparent as possible to the user. This means that our protection scheme will have to do a great deal of automatic decision making instead of requiring that a user individually specify protections for every document and annotation.

Another issue that must be addressed is the heterogeneity of many different haystacks. Different individuals will use different IR systems, will use different vocabulary in their annotations, and will have different opinions about the quality of certain objects. Our system will therefore have to learn (or initially, be told) certain facts like:

- Which users are trusted by the user to have "good opinions" about what documents are of high quality.
- What other users mean when they say (annotate) certain things about documents (for example, one person might add an annotation about "writer" when another says "author").
- What other users mean when they perform certain queries. If I perform a query on "finance," my haystack should take note of a colleague's query on "banks" if that colleague studies money, but not if they study rivers.

To facilitate such collaboration, we will develop middleware IR system (MIRS) layer with facilities for integrating multiple user (front end) and back-end systems (see section G.3. The MIRS will be responsible for sharing information in such a way that the queried system can understand the query and the querying system can understand the results. We will investigate the use of information interchange protocols, such as Harvest, SOIF, or KIF, that may be appropriate for this task; We anticipate that we will have to augment these protocols to convey information about relevance judgements and quality measures.

G.2.3 A Distributed System

The system we propose will produce a collection of interacting haystacks running on numerous desktops. It is a cliche that such a distributed system has more power than any current centralized system, provided its raw advantage in resources can be properly harnessed. The tool we propose to harness this power is the *archivist,* a component of each haystack that is continuously working to improve the information in it.

At the most prosaic level, the archivist can simply be a maintainer of links. Modern web-search engines rely on crawlers that gather in all the information on the internet. When as piece of information changes, moves, or disappears, the web-search engine doesn't find out until the crawler revisits the information. Thus we are all familiar with searches that turn up dangling internet pointers. The scale of the internet forces even the most powerful centralized crawlers to revisit web sites relatively infrequently.

An individual has a much more modest collection of information. It is therefor much cheaper to keep it up to date. It will be inexpensive for the archivist to revisit each archived document relatively frequently, detecting changes or moves and alerting the user if they so desire.

At a more sophisticated level, the archivist can attempt to proactively sniff out information in the expectation that its user might eventually need it. The body of information in a user's collection gives a great deal of insight regarding his interests. The archivist can search for additional information within the same domain—new papers by archived authors; new advertisements for archived hobbies; responses to the author's own activities. Such activity resembles the kind of work on information filtering discussed in Section H, but with an important difference. While all those filtering methods rely on the user to rate results, we can use all of the passive observation machinery discussed in the previous sections to learn about an author without his active participation. This kind of per-person monitoring of information space is clearly a computation intensive undertaking, one that it seems unlikely that a centralized system can scale up to meet.

G.2.4 A Corpus of Individual Usage Patterns

The emphasis that the Haystack system places upon the logging of user retrieval mechanisms makes it ideal as a tool for gathering information about how people actually use IR systems. We believe that such information will help us and others tune IR systems to effectively do what users want, instead of what we think they want.

The bulk of current IR research corpora, such as the well know Medlars and Cranfield collections, are, like the the IR systems that test on them, static. They contain a corpus and a list of queries together with a list of which documents are relevant to which queries. Systems attempt to optimize their performance at matching queries to the documents judged relevant. These collections are small and involve a small set (at most a few hundred) of queries. They do not contain any record of how the original user responded when faced with the results of their query: were they satisfied? Did they attempt to expand the query, or switch to a completely different query?

Even the recent *Text Retrieval Evaluation Corpus* put together by NIST has the same model: a set of documents, and a set of a few hundred queries laboriously put together by professional information searchers. The objective, as above, is to build a system that returns the maximum number of documents judged relevant to the query by professional readers.

The kind of information we plan to log is complementary to that gathered by these collections.

Instead of looking at queries and relevant documents in a vacuum, our log will record the long term interaction of a user with a corpus. We will see how the user's queries change over time, as they become more familiar with their material. We will log the strategies they use to refine queries, possibly discovering information about individual preferences that will impact the design of future systems. For example, what kinds of queries do users make? Are they usually boolean in nature? Single words? Do they tend to be over-precise and find no documents, or do they overgeneralize and get swamped with useless results? Are users typically satisfied to get a few documents relevant to a query, or are they unhappy until they are sure they have gotten the very best few? Do they prefer high precision and low recall, or the converse? What query refinement strategies do they use? How much difference is there in practice in the way different people describe the same information need? Each haystack will provide a user-specific set of answers to these questions.

A quote from Karen Sparck Jones [11] regarding current test corpora demonstrates a perceived need for the kind of data we aim to collect:

> Request sets remain a problem because they are typically still small, and often not genuine samples from real users or do not have the same characteristics as real ones. There is a further problem where request sets illustrate professional habits, but not those of the 'ordinary end-user'....
>
> Large request sets are needed...to ensure that they are genuinely representative of user populations, especially in frequency distributions, for such different 'direct' properties as:
>
> - linguistic expressiveness (eg many/few terms);
> - topic particularity (eg broad/narrow terms);
> - collection fittedness (eg central/marginal topics);
> - as well as in vital 'indirect' property:
>   relevance requirement (eg highly/broadly, most/some documents, etc etc). [sic]

Our logging goals fit comfortably with our collaboration and learning goals. We have already discussed plans to expose as much information as possible (subject to privacy concerns) about what users have in their haystacks. This can easily be extended to similarly expose what users have looked for in the past (for example, we might expose the "query records" discussed in Subsection G.2.1). This might have other beneficial consequences. For example, we can envision a future in which certain individuals, companies, or advertisers search for queries that they think they can answer. If the query was made relatively recently, and the results the user has do not seem to satisfy it, then these query prospectors may actually contact the user with an offer to provide them with additional information.

G.3 Architecture

We are interested in building on top of standard information retrieval systems to create a community of individual but interacting "haystacks": personal information repositories that archive not only base content but also user-specific meta-information, enabling them to adapt to the particular needs of their users.

We have identified three design decisions which will be necessary to accomplish this task.

- We will use a heterogeneous collection of off-the-shelf components operating as plug-ins to our own middleware system. This increases the flexibility and portability of our system and allows us to leverage recent advances in commercially available subsystems.
- We will fully integrate our system into the user's desktop. This lowers the burdens of usage, increases the system's utility to the user, and enables us to collect realistic data.
- The information contained in our haystacks will include both the traditionally-archived content of documents and as much manually and automatically extractable meta-data as is available. This changes the nature and potential usability of the archive itself.

G.3.1 Heterogeneous Off-the-shelf Components

The Haystack project is focused on the individual customization of information retrieval systems, rather than the creation of improved information retrieval engines. Our architecture envisions the use of one or multiple IR systems as plug-ins for the back end—black boxes of the classic type that statically take a query and return the documents they consider most relevant to it. Haystack will, over time, modify these documents so they are returned only when the user actually wants them. Instead of trying to modify the IR system to improve performance, we will modify the corpus.

To make the inclusion of diverse information retrieval systems possible, we will build a middleware system, the MIRS, which masks the details of particular information retrieval systems from the user. The MIRS serves to detach the information retrieval "engine" from the user interface and annotation system, specifying only that the engine should accept a query and return documents that "match" under whatever criteria it uses. To add a new back end to the MIRS involves implementing a few simple wrapper functions; the MIRS interacts with its back ends solely through these. This plug-in architecture makes Haystack relatively portable: the high-overhead functionality of the IR back end can be platform dependent while the user interaction system is lightweight and system independent.

Making the IR system a plug-in has several advantages but also raises challenges. On the positive side, such an approach makes haystack an excellent tool for comparing IR systems and for using them painlessly. Current "hardcore" IR research often focuses on the internals of the system without paying a great deal of attention to the user interface. Many prototype systems make do with a primitive TTY interface. And each, of course, has its own query language and idiosyncrasies. Plugging such systems into haystack immediately provides the benefits of an integrated modern user interface. This of course improves ease of use. It also means that an IR system can be replaced by a better one without forcing the user to learn a new query language.

A corresponding challenge is to design an interface that is simple but simultaneously rich enough to take advantage of the varying capabilities of IR systems. Some IR systems take boolean queries, while other have a fuzzy model. Some stem their terms, while others use exact matches. Some IR systems have dynamically updatable indices, while others require a (slow) full rebuild when new documents are added. We will develop interfaces that hide these details from the user. A greater challenge arises if we wish to integrate graphical and non-traditional information retrieval methods, such as Scatter/Gather [5], that do use the standard type-a-query-and-get-some-documents model.

We can address such challenges at the individual level, but also expect them to arise at the collaborative level. When different people using different IR systems want to share haystacks, we want to pass information around in such a way that the queried system can understand the query and the querying system can understand the results. Information interchange protocols such as Harvest or SOIF may be appropriate for this task. But we will probably have to augment them to convey information about relevance judgements and quality measures.

Another opportunity inherent in this plug-in model is simultaneous use of multiple IR systems. Once the user enters their query, it is trivial to forward it to multiple IR systems. A greater challenge is to integrate the results that come back, in particular, to combine the relevance judgements of the various systems. As before, we believe that learning over time will indicate how useful each IR system is to the particular individual making use of it.

Haystack's plug-in architecture facilitates the cross-comparison of different IR systems, and we intend to run head-to-head experiments in the intermediate stages of this project. In the later stages of the project, we will support access to more than one IR system for a single query, allowing a haystack to exploit the relative merits of each system. In addition to off-the-shelf information retrieval systems, we anticipate integrating at least one database system to handle structured queries. Optionally, we will extend the architecture to handle multi-media or multi-modal plug-in back ends; in particular, under option 1 we will investigate integration of an image search tool such as the one developed by Lipson under the MIT AI Lab's MURI ("Trainable Modular Vision System") project. [Option 1]

G.3.2 Tight Integration with User's Desktop

Unlike massive-corpus search systems, a personal IR system can be fully integrated into a user's environment. Instead of forcing a user to turn to a web browser to access a search engine, we aim to let them access their haystack from within whatever application they are currently running. Each application is able to provide certain information retrieval functionalities which we can use. For example, a web browser might be interfaced to Haystack to allow functionalities such as "archive what I am currently looking at." Integrating haystack into a text editor will allow the user to archive the document currently being edited, edit additional information about any document, or use search tools to pull a document into the editor.

Such easy integration relies on the fact that the front ends—the user interfaces—for haystack are interfaced through our middleware layer. This enhances our ability to add other existing and newly available desktop tools and affords us with increased cross-platform portability. Since our initial test audience is unix-based, we anticipate beginning with the most common utilities in that environment: an editor, a mail reader, a web browser, etc. The user should be able to access his haystack through any of these tools in as natural a manner as finding any other piece of information. For example, in the editor, rather than opening a file by giving its location, the user should be able to "retrieve" a file by specifying a query.

Integration into the user's desktop decreases the overhead of using haystack to the point that we believe most users will interact with a significant portion of their data through our system. A user can use haystack's search tools to load a partially remembered file into his editor. Similarly, a user will be more willing to annotate a document if he can do so in a few keystrokes, without stopping what he is doing. Even something as simple as specifying a file name can be thought of as performing a particularly restricted kind of search. Minimizing the psychological and operational barrier to using the IR system will maximize the willingness of the user to invest time in "prepping" it, and will therefore maximize the benefit the user sees from it.

Integration also creates better opportunities for passive observation. Unlike a remote web server that is only notified each time a user clicks on a link, we can track details such as which desktop windows are active, whether the user is scrolling to a particular place in a document, and so on. This extra information will tell us more about what a user considers "relevant" to their current task, and thus give us more opportunity to adapt for the future.

In addition to the tight integration with existing desktop applications, Haystack provides two other interfaces. One is an applications programmer interface, intended for use by other programs. (Optionally, this interface will be used to connect Haystack with SodaBot, the software agent environment and construction system developed in one of the PI's groups — Option 2.) The second is intended for users accessing Haystack directly, without performing one of the functions of existing desktop applications. For example, a user may wish to retrieve a document for printing or annotation. Following our principle of using off-the-shelf components whenever possible, we will use a web browser for this interface; however, the functionality is not tied to web browsing and can be used separately from it.

G.3.3 Archiving MetaData

A final distinction between Haystack and existing IR systems comes at the level of the data to be archived. Existing IR systems generally archive the text of documents. We aim instead to archive the *content* and *meaning* of the document. While the text of a document is a good first approximation to its content, we aim to evolve this approximation towards a more accurate representation.

In Haystack, we archive all available information about a document. This includes its location, its history, its authorship, its relevance to past queries, when it was last accedded, etc., as well as its text. All of these (including the text) can modified over time to reflect what we learn about the document. The idea here is that we are exploiting not only the text of the document but also its organization and its relationships to other documents already in the collection. In Haystack, this meta-data is first class: it is itself information to be archived and retrieved.

Archiving meta-data means that we can also archive things that do not make sense to traditional IR systems. We can, for example, archive binary files whose contents are meaningless to a text-based IR system. (The "man page", or manual, for the binary executable makes an excellent annotation to aid in appropriate retrieval.) We can also archive groups of documents. This includes directories and folders, mail files and newsgroups. It also includes queries. By archiving a query, we provide a "place" to remember details about the user's interaction with that query: what refinements it underwent, which documents satisfied it, and which did not.

The idea of archiving meta-data (including the "content" which is the traditional domain of IR) is a bit like semantic information retrieval. It exploits information about what the document is about. Semantic information retrieval runs into difficulty extracting content — semantics — from the syntax of the document. Traditionally, purely syntactic methods have outperformed semantic methods simply because the semantic methods were too slow to extract useful information. Our approach does not suffer from these drawbacks because the semantic information that we use is all readily available: in the document format and superficial properties, in the interactions between the user and the information, and in the optional annotations that a user may provide.

To represent this meta data, we will use the LORE (Lightweight Object Repository) system developed at Stanford. Lore is generic database for storing and updating *objects* that are linked by labeled *relations*. Unlike traditional database systems, LORE does not require that the database schema be fixed in advance; it allows for the lightweight addition of new relations and new objects.

H Comparison with other ongoing research.

Our project grows out of two long-established research thrusts, in information retrieval and information filtering.

Information retrieval focuses on the question of guiding users to information in a large, fixed corpus. Classic applications include library catalogs. Research is embodied in such systems as SMART [16]. More recent applications include the large web search engines (Alta Vista, Excite, Lycos). In all these projects, the typical assumption is that the corpus is a large, non-personalized collection of information that is queried by numerous clients. Nearly all systems involve a keyword-query interface and return lists of relevant documents. The overriding question is which documents are most relevant to any particular set of keywords. It is important to notice that the identity of the client plays no role in deciding relevance. In particular, there is no recognition of the fact that the same query can mean different things to different people.

Information filtering, also know as routing, attempts to deal with individual users' needs. Information filtering has an on-line model, in which new information is continuously arriving, and the goal is to decide which information to bring to the attention of any particular user. A user often "trains" the system by pointing out several "interesting" documents; the system then attempts to decide, based on this information, whether a given new piece of information will be interesting. Unlike information retrieval systems, information filtering systems will make different decisions based on the identity of their clients. However, although they are good at deciding what new information a client might be glad to see, they are not designed to help users solve a newly-arisen information need. Such a new information need can be modeled as a change in what the user considers interesting, but to make this work the user would have to take the time to train the system on documents that are relevant to the new query—a time consuming and labor intensive task.

More recently, a variant of information filtering known as collaborative filtering has been explored. Instead of having the system decide whether a document will be interesting to a user by comparing it to a user's profile, systems such as Tapestry (Goldberg, Nichols, et al, Xerox PARC) let other users provide the quality information. As users read documents, they mark them as worth reading; large numbers of these interest marks cause a document to be brought to others' attention. Like standard information filtering, these projects typically seem focused on a model of continuously arriving documents from which interesting ones need to be selected, and have less to say about how these ideas can be applied to queries.

Haystack is an attempt to draw together some of the threads that have been explored separately in information retrieval, learning based information filtering, and collaborative information filtering. Integrating the information retrieval system into each individual's desktop instead of restricting it to large, communal corpora gives us the opportunity to personalize the information retrieval process, adapting it to individual attitudes about what is interesting and how it is described. Two different haystacks might give completely different answers to two different individuals' queries, and be right because the two users mean different things. The fact that the system is explicitly designed to deal with queries creates an alternative to filtering systems' model of a "notification" service, instead allowing useful processing of a users specific information need at the time it becomes apparent.

H.1 Information Retrieval and MetaData

Several systems are currently being developed to mediate between a user and the numerous search engines that are currently available. Systems such as Metacrawler provide a single interface that ships user queries to multiple search engines with different query languages and gathers the results into one uniform display for the user. Metacrawler gives a user relatively little control over its meta-interface. Other systems such as Sensemaker [3] allow the user a great deal more control over how returned information is displayed. However, this class of tools focuses on mediating between a user and multiple remote corpora. They do not allow users to search through their own personal corpora. While they allow users to customize their view of results, they do not let users customize their corpus.

A number of commercial vendors, such as Alta Vista, are offering "personal editions" of their internet search software. However, these systems are mired in the same "static" paradigm as their large-scale brethren. Their model is the return of documents matching a given query, and their behavior does not change over time. They do not provide passive or active tools for annotating documents to allow for improved future searches.

Apple's Cyberdog system [1] exhibits some of the desktop integration properties we aim to put into our system. Cyberdog allows uniform manipulation of different types of "documents" (web pages, mail files, multimedia) to be manipulated and bundled into "folders" by dragging and dropping. Cyberdog also provides a limited amount of search capability. A different undertaking at Apple, the "find by content" system, promises someday to transparently index all of a user's local files for searching. However, there is no support for annotation, system adaptation, or interconnection of personal collections. Yet another interface metaphor being explored [15] is a "Piles of Paper" user interface metaphor which allows users to bundle groups of related documents together on the desktop.

There are many efforts underway to develop metadata syntax that will allow for the annotation of information. Systems such as the Dublin Core [17] and the Warwick framework [12] represent a growing consensus among librarians and computer scientists regarding a proper syntax and semantics for metadata information. This scheme allows for multiple metadata representations (making it a metametadata system). Commercial vendors also exploring metadata representations—see, for example, Apple Computer's MetaContent Format. [13] All the systems just mentioned attach semantics to the syntactic elements. We recognize the value of semantics, but have not yet committed to attaching any semantics to metadata. In particular, when dealing with a single user, we feel that we can leave it to the individual to maintain the semantics of the system, allowing the system to get by with purely syntactic methods. When we extend our system to allow interaction between multiple haystacks, some semantic information might make it easier to translate the data in multiple haystacks into the form expected by our user.

H.2 Learning and Filtering

Some of the the type of technology we propose to deploy has already been shown to be an effective tool for the task of *filtering.*—that is, identifying new "interesting" items in a continuously growing collection that should be brought to the attention of a particular user. Both collaborative and learning based methods have been proposed for this kind of task.

Balabanovic et al. [2] have demonstrated a system that learns about user preferences regarding web pages; the model is to train by presenting users with candidate web pages and asking them which are interesting. This lies somewhere between our two plans regarding search results. As a scheme which requires user interaction, it will be less-used than our planned passive observation which could detect preferences without user interaction. On the other hand, although it involves the user, it limits their responses to yes or no, rather than letting them provide annotation regarding the page content.

H.3 Collaborative Filtering

Another approach to filtering is *collaborative filtering.* The Information Tapestry of Goldberg, Oki, Nichols and Terry [8] was the first to propose this model. Tapestry allowed users to rate email messages sent to them from mailing lists; the ratings were exposed to other users receiving messages from the same mailing lists. Messages ranked very low by numerous colleagues could be ignored. People were also encouraged to annotate documents with information that would be helpful to their colleagues. However, since the benefit of annotation accrues to everyone *but* the annotator (he has already read the mail), very few items ever get annotated. GroupLens [14] is a newer but similar project dedicated to filtering NetNews. Unlike Tapestry, which was designed as a monolithic system catering to people at a single site, GroupLens addresses that added complexities that arise when individuals form a distributed system. Like Tapestry, GroupLens relies on active participation by readers who rate what they see. It therefore suffers from the same problem: by the time someone can rate an article, they receive no benefit for doing so.

In a contrasting approach, systems such as Hill and Hollan's Read Wear and Edit Wear [10] use passive observation of their users to gather information about such questions as which parts of a document people spend the most time reading or what parts of a spreadsheet they most modify. This is the kind of information we aim to gather for our applications. However, while Hill and Hollan directly display the usage history information to other users and allow them to draw conclusions from it, we hope to extract from the mass of usage data the relevance information that will make future searches more effective.

These collaborative filtering approaches exhibit many of the elements we plan to include in our system. However, they all differ in one crucial respect. In filtering, the goal is to bring "interesting" material to the attention of the user. The user plays a passive role here, waiting for such interesting material to arrive. Instead, we aim to address the search problem, in which a user suddenly discovers an information need and sets to work trying to satisfy it. An arbitrary "interesting" document will not satisfy him; instead, he wants to see documents relevant to his query. This adds an extra dimension of complexity beyond that faced by the filterers; instead of using rankings to decide how good a document is, we have to decide how good it is *for a particular query.*

A system more similar in spirit to what we propose is What'sHot, developed at Australia's Distributed Systems Technology Center. [6] What's Hot is an internet search engine that places a checkbox by each returned item, allowing a searcher to choose items to mark the resources that they like. This information is used to adjust the ranking of documents, so that more popular documents are given more visibility. However, What'sHot suffers from the standard problem of asking people to rank web documents *after* they have read them, when there is no longer any benefit to their doing so. By operating passively, Haystack will gather information even from lazy users. By managing an individual's own information, Haystack will increase his motivation to annotate results.

I Personnel.

The proposers' web pages can be found at:

http://theory.lcs.mit.edu/˜karger

and

http://www.ai.mit.edu/people/las

David Karger (A.B. Summa cum laude in Computer Science, 1989, Harvard University; Certificate of Advanced Study in Mathematics, Cambridge University, 1990; Ph.D., 1994, in Computer Science, Stanford University) is an Assistant Professor of Computer Science and a member of the Laboratory for Computer Science at the Massachusetts Institute of Technology. He is a recipient of the ACM 1994 Doctoral Dissertation Award and a 1996 National Science Foundation CAREER award. He has just been awarded a 1997 Alfred P. Sloane Foundation Fellowship. He held a National Science Foundation graduate fellowship from 1990-93 and a Hertz Foundation graduate fellowship from 1993-94.

Professor Karger's research interests include algorithms and information retrieval. His work in algorithms has focused on applications of randomization to graph optimization problems and led to significant breakthroughs on several core problems. His work on information retrieval includes the co-development of the Scatter/Gather information retrieval system at Xerox PARC, which suggested several novel approaches for efficiently retrieving information from massive corpora and presenting it effectively to users. He has received two patents related to his work on this project. He is the author of more than 30 publications and has served on the program committees of some of the most important theoretical computer science conferences.

Professor Karger's academic year salary is fully covered by MIT. He expects to spend roughly 35% of his research time on this project during the academic year and during the summer.

Lynn Andrea Stein (A.B. cum laude in Computer Science, 1986, Harvard and Radcliffe Colleges; Sc.M., 1987, and Ph.D., 1990, in Computer Science, Brown University) is an Associate Professor of Computer Science and a member of the Artificial Intelligence Laboratory and the Laboratory for Computer Science at the Massachusetts Institute of Technology. She was a recipient of the 1995 Ruth and Joel Spira Teaching Award and the 1993 National Science Foundation Young Investigator Award as well as the 1992 General Electric Foundation Faculty for the Future Award, the 1989 Sigma Xi Graduate Student Award, and an I.B.M. Graduate Fellowship from 1987 until 1990. She is the former holder of the Class of 1957 Career Development Chair and a founding Fellow of the Kiss Institute for Practical Robotics.

Professor Stein's research spans the fields of software agents, human-computer interaction and collaboration, object-oriented programming, cognitive robotics, and common sense reasoning. Her research group has produced the SodaBot software agent environment and programming language. She is currently a principal investigator of the MIT Artificial Intelligence Laboratory's Intelligent Room project. She is author or co-author of more than 30 publications and has served on the program committees of numerous national and international conferences in these fields. She is co-founder and organizer (1993) of the Robot Building Event at the national conference of the American

Association for Artificial Intelligence (AAAI). From 1991 to 1996 she served as Associate Chair and then Chair of the AAAI's Symposium Committee; she is currently an Executive Councillor of the AAAI.

Professor Stein's academic year salary is fully covered by MIT except during the 1997-1998 academic year, when she will be on sabbatical in residence and working primarily on her research activities. Professor Stein would expect to spend approximately 20% of her research time during the academic year on this project. During the summer, Professor Stein would devote approximately 50% of her time to this project. Professor Stein is currently supported in part by the MIT AI Lab's HCI project (scheduled to end in FY97) and in part by the ONR/MURI project. She also holds an NSF Young Investigator Award (until FY 98).

J Prior Accomplishments.

The Laboratory for Computer Science and the Artificial Intelligence Laboratory at MIT have extensive history in related fields, including support for the White House Email system and the National Performance Review, the Library 2000 project, Content Routing and Semantic File Systems, and the World-Wide Web Consortium. The PIs have been involved with a number of related projects, including the current Intelligent Room DARPA HCI project. Two specific accomplishments bear special attention here.

J.1 Scatter-Gather

Professor Karger was one of the principal developers of the Scatter/Gather information retrieval system [5, 4] at Xerox PARC. Scatter/Gather explored the use of document clustering as a user interface tool, and demonstrated that it provided an effective complement to keyword based searching. While keyword searching is analogous to checking a book index, Scatter/Gather used document clustering to implement a table-of-contents metaphor. This system allowed users completely unfamiliar with a collection to effectively search for information within it, homing on on suitable documents simply be pointing and clicking.

Scatter/Gather provides a browsing tool which complements and extends traditional search techniques. This tool is fully automated, requiring no human preprocessing to assign keywords or subject labels to the documents. Instead, it is the computer which determines the topics; the user need only determine which topics appear relevant to his information need. Keyword searches require that the user come up with the right question; with Scatter/Gather it is the computer which asks the questions, while the user has the much easier task of answering them.

Scatter/Gather also serves as an effective companion to keyword search. It can be used before searching to help formulate a good query, and it can be used after keyword search on those many occasions when a query retrieves too many documents to examine individually.

The interface was efficiently implemented to allow real-time (sub-second) interaction on arbitrarily large corpora, after small amounts of fully automatic preprocessing. Publications describing this work appear in the 1992 and 1993 proceedings of the conference of ACM's SIGIR (special interest group on information retrieval).

J.2 SodaBot

SodaBot is a general-purpose software agent user-environment and construction system. It is designed to simplify the specification of agent interactions. SodaBot consists of both an agent programming language, SodatBotL, and a runtime environment to support these agents. It is intended to facilitate construction and use of systems of software agents. SodaBot is designed for easy and modular installation on a variety of platforms; it is written in PERL, which runs under UNIX and on PC and Macintosh systems. The SodaBot system also supports a high-level programming language, SodaBotL, which is designed specifically for the specification of software agents. SodaBot and SodaBotL were developed by Michael Coen and others in Lynn Andrea Stein's AP Group.

In SodaBot, each application or user is given an individual basic software agent (BSA). The BSA is an *agent operating system*: a generic (in the sense of universal) computational framework for implementing and running specific agent applications. A BSA simultaneously (via a time-sharing algorithm) runs multiple SodaBotL programs provided both by its owner and by other people. The BSA architecture *disconnects* agent programs from the specific computational environment in which they run. They no longer need to be "hard-coded" with specific parameters for particular activities.

The SodaBot agent programming language (SodaBotL) offers high-level primitives and control-structures designed around human-level descriptions of agent activity. SodaBotL abstracts out the low-level details of agent implementation. In a typical Unix environment, for example, agent creators are freed from the bother of dealing with system calls, mail servers, sockets, and X-windows. This abstraction also provides portability; only the BSA must be ported to a new platform.

One of SodaBot's major strengths is the ease with which it can programmatically make use of new resources made available to it. For example, SodaBot provides an expect mechanism which allows SodaBot agents to interact in a scripted manner with any application which provides a textual interface. In developing SodaBot based applications, one often provides access to a given service by encapsulating the service within an agent. Other agents in the system can then make use of the service without knowing the specific details of its interface. In a sense, SodaBot allows quick and simple deployment of networked services for applications which themselves are not inherently network-based. SodaBot abstracts away the distributed networked nature of these services and allows agents to concentrate on the content of their actions, not the execution.

The SodaBot system consists of a set of core basic services such as networked data connections, the expect mechanism, access to the Web, access to other agents, etc. Additional features of SodaBot include *automatic distribution of user-created agents* and a graphical user-interface.

SodaBot is currently in use as the "brain" of the MIT AI Lab's Intelligent Room (DARPA HCI) project.

K Facilities.

This proposal would make use of the existing facilities at the MIT Laboratory for Computer Science.

The research described here will be performed using the facilities of the MIT Laboratory for Computer Science (and, under options 1 and 2, of the Artificial Intelligence Laboratory). The Facilities of the MIT Lab for Computer Science include existing office and laboratory space and extensive state-of-the-art networked heterogeneous computational facilities. Additional dedicated workstations will be acquired under this proposal to support the significant computational burdens of some of the IR back ends that we wish to explore as well as the amassing of large Corpora of User Interaction Patterns. Detailed of these planned acquisitions are given in section L.

Options 1 and 2 would in addition use facilities at the MIT Artificial Intelligence Laboratory, including the extensive vision laboratory (option 1) and the existing fully-instrumented "Intelligent Room" (option 2).

L Cost Breakdown.

III. Attachments

References

[1] Apple Computer, Inc. Cyberdog. `http://www.cyberdog.apple.com/overview`.

[2] Marko Balabanovic, Yoav Shoham, and Yeogirl Yun. An adaptive agent for automated web browsing. Technical Report CS-TN-97-52, Stanford University, 1997.

[3] Michelle Q Wang Baldonado and Terry Winograd. Sensemaker: An information-exploration interface supporting the contextual evolution of a user's interests. Stanford Digital Library Project SIDL-WP-1996-0048, Stanford University, 1997. also available at `http://www-diglib.stanford.edu/cgi-bin/WP/get/SIDL-WP-1996-0048`.

[4] Douglas Cutting, David R. Karger, and Jan Pedersen. Constant interaction-time scatter/gather browsing of very large document collections. In *Proceedings of the* 16^{th} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, July 1993. Pittsburgh, PA.

[5] Douglas Cutting, David R. Karger, Jan Pedersen, and John W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the* 15^{th} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, 1992.

[6] DSTC. What'shot information broker. `http://www.dstc.edu.au/BDU/APAP/WhatsHot/WhatsHot.html`.

[7] William B. Frakes and Ricardo Baeza-Yates, editors. *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[8] David Goldberg, Brian Oki, David Nichols, and Douglas B. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, December 1992.

[9] Donna Harman, editor. *The Fourth Text REtrieval Conference*. National Institute of Standards and Technology, 1995. NIST SP 500-236. Web access at `http://www-nlpir.nist.gov/TREC/`.

[10] W.C. Hill, J.D. Hollan, D. Wroblewski, and T. McCandless. Edit wear and read wear. In *Proceedings of CHI 92 Conference on Human Factors in Computing Systems*, pages 3–9, New York, 1992. ACM.

[11] Karen Sparck Jones. Report on building and using test collections panel, sigir 1996. *SIGIR Forum*, 30(3):5–6, 1996.

[12] Carl Lagoze. The warwick metadata workshop. *DLIB Magazine*, August 1996. `http://www.dlib.org/dlib/july96/lagoze/07lagoze.html`.

[13] Metacontent format. `http://mcf.research.apple.com/`.

[14] Paul Resnick, Neophytos Iacovou, Mitesh Suchak andPeter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, NC, 1994. server at `http://www.cs.umn.edu/Research/GroupLens/`.

[15] Daniel E. Rose, Richer Mander, Tim Oren, Dulce B. Ponceleón, Gitta Salomon, and Yin Yin Wong. Content awareness in a file system interface: Implementing the 'pile' metaphor for organizing information. In *Proceedings of the* 16^{th} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 260–269, June 1993.

[16] Gerard Salton, James Allan, and Chris Buckley. Automatic structuring and retrieval of large text files. *Communications of the ACM*, 37(2):97–108, February 1994.

[17] Stuart L. Weivel and Eric J. Miller. Dublin core metadata element set. `http://www.oclc.org:5046/research/dublin_core/`, May 1996.