1 Project Summary

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.

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.

2 Introduction

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. Haystack aims to be a system that looks more like a personal bookshelf, rather than the large library presented by traditional information retrieval systems.

2.1 Motivation

Typical information retrieval systems assume a large corpus which is queried by numerous anonymous 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 probably 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. This makes it different, in crucial ways, from the library. There are important differences in both content and organization, as well as new opportunities for the exchange of information between individuals.

Content

One class of differences has to do with the content of a bookshelf versus a 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.

- Bookshelves exploit the "RE" in information REtrieval. The majority of searches through a bookshelf are for material that we have looked for before. The user 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 new information that might not exist.

- A bookshelf is often more up-to-date (in its own limited area) than a library. Its owner, who is actively seeking information in his area of interest, often finds new information before the library gets around to it.

- A bookshelf contains items that are too "rough" to be placed in a library—memos, drafts, notes, etc. Although not of general value, these items may be of great value to a small number of people.

- Users are well acquainted with the content of their bookshelves. Once they've been pointed at an item, 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 and reactions 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.

In summary, a person's bookshelf contains the bulk of the information that he considers most valuable. It is thus somewhat ironic that the bulk of information retrieval tools are directed towards the secondary collection of the library. For example, nowadays it is much easier to search the web than ones own file system or mail archive.

Organization

A second class of differences has to do with how content is organized and searched:

- 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.

- Individuals exploit their idiosyncratic organization when searching for information: they may look for a blue book, or a book on the bottom shelf, or a book next to another book. At a library, users are limited to searching the standard classification.

- Any item in a person's bookshelf is likely to be one that a person expects to refer to frequently. He thus has a greater interest in actively organizing the information to make it easily accessible "next time" than when looking at a library book that he might never see again.

This contrast between libraries and bookshelves highlights the ways in which bookshelves become our own personal spaces. The bookshelf is changed both passively and actively by the user to make it easier for them to seek information in it. In contrast, the library is unchanged by its interactions with users.

This distinction has an electronic analogue. 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, or by requiring the user to provide more information initially, 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.

Colleagues

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.

- Books in colleagues' 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.

2.2 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. The system should index everything (not just text) that is of interest to its user. We discuss our plans in this are in Section 3.1. The 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 3.2.

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 3.3.

In Section 4, we discuss related work.

3 Our Approach

Our system, Haystack, is an electronic bookshelf in the senses described above. It takes advantage of tight integration with the user's desktop, tracking information as the user encounters it. It customizes and personalizes this information in much the same way that a physical bookshelf does. In addition, this bookshelf has many of the virtues of a traditional IR (library) system. Future versions of this system will learning user profiles and preferences and a collaborative infrastructure among haystacks.

This section describes the data infrastructure of the Haystack system. It is based on an existing prototype written in perl. This version of the system provides most of the essential archive and query functionality, but does not support learning or communication among Haystacks. In addition, there are many additional base features that could be added to the prototype system. The first effort under this proposal would be a reimplementation of the system in Java, integrating the crucial aspects of the perl system and providing enhanced functionality. However, we do not expect the data model described here to change significantly.

3.1 The Haystack Database—Personalized Content

One important way that the Haystack project differs from others is in the *kind* of information it organizes. While typical IR systems focus on the indexing of text (or close variants such as html), Haystack attempts to archive *everything* of interest to its user. This is facilitated by its integration into a user's desktop environment and by its use of a reflective metadata architecture.

An additional distinction between Haystack and existing IR systems comes at the level of the data to be archived. We aim to archive both the *content* and *context* 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. Context includes metadata about the document, such as how and when it was used, that can be exploited to improve retrieval.

Desktop Integration

When information comes into your physical office, you have to decide whether to put it on your bookshelf. In contrast, Haystack aims to archive all information that you encounter. Haystack is integrated with desktop applications. (In the existing prototype, these include a web browser and an editor, but future versions will include a variety of other applications.) This means that it can automatically notice and initiate archival of every piece of information that crosses your desktop.

For example, Haystack can be configured to automatically archive every web page that you visit. (This archiving feature can, of course, be turned off for convenience.) It can archive your mail as you read it or the files in your home directory. Eventually, Haystack could be integrated into the shell (or file system) itself.

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.

Diverse Data, Extensibility

Traditional IR systems archive only text. Haystack archives any information from which sensible text can be extracted, such as postscript or latex files. In addition, Haystack can deal with such pseudo-documents as individual mail messages in a mail archive and with both local files and remote objects (urls). It achieves this by means of plug-in translators and transducers whose operation is controlled by a rule-based system based on the likely type of the incoming object. The operation of the rule-based system is partially user-configurable. Users can also provide additional translators and transducers.

In addition to these non-traditional text types, Haystack also archives non-text objects. In this case, the object itself is not stored in a text-based IR system, but the object's existence and the means by which to recover are archived together with information *about* the object. This is one example of Haystack's use of metadata.

Metadata

Together with the base content archived by Haystack, Haystack records data about the data it stores. This includes, for example, the desktop application that supplied the data to Haystack or the time and date of archiving. It may also include specialized information about the object, such as the sender and recipient of an email message. This is why it makes sense for Haystack to archive non-text data; although the contents of, for example, an executable file, may not make much sense to query, the executable created from a specified source file on a particular date may be a perfectly sensible object to want to retrieve.

Like the translators and transducers described above, Haystack comes with a set of meta-data extractors for standard document types and fields. These currently include author, date, and other types of standard metadata for such document types as email and latex. Again, like the translators and transducers, these field extractors and their accompanying control structures are customizable.

In Haystack, metadata is itself first class. That is, there may be meta-data about the meta-data. Haystack avoids infinite regress through an on-demand meta-data system similar to [Smi82]'s reflective tower. In fact, a typical object (bucket of bits) has no useful data. Rather, we consider the text of the object (typically extracted with a translator or transducer) to be one piece of its metadata.

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.

Context as Well as Content

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 accessed, etc., as well as its text. Much of this information is available only to a system integrated with the user's desktop: As you access information, the context in which you encounter it is recorded together with the information itself. This way we exploit not only the text of the document but also its organization and its relationships to other documents already in the collection. All of these aspects are available to subsequent queries, so that it is possible to ask for "all of the email messages I sent to Chris last Thursday in response to this email message".

Not all of this information need be entered when a document is first encountered. Archived information (including both the text itself and its metadata) can be added to or modified over time to reflect what we learn about the document. In the prototype system, MD5 matching is used as a crude filter for document identity to prevent multiple inadvertent archivings of a single document and to allow merging of information when a document's records are updated.

Active Annotation

In addition to the automatic annotation mentioned above and described in more detail below, the Haystack system allows for explicit user annotation using the same underlying meta-data system. While we never rely on user's hand annotations, the system is fully able to exploit any such efforts that the user may wish to make. In many cases, the user knows better than the system. We give them easy tools for adding to or overriding the system generated metadata. We let the users jot "notes in the margin"—comments on the data.

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 3.3).

While we can never rely on a user's 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 [FBY92, 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.

3.2 Adapting Over Time

So far we have discussed "static" data about objects—information that can be extracted by automatic tools. We have also discussed the fact that a user can choose to add information to their hayloft. Another line of very useful information can be extracted by watching a user manipulate their information. This information can be placed in the haystack to improve retrieval. The benefit of this approach is that it gathers data *without* adding to a user's workload.

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 usage, passive observation of relevance, passive observation of the query process

Passive Observation Of Usage

By watching its user, the Haystack can gather a great deal of information about the data in their collection. This includes information such as what mail got forwarded to who, what documents got edited at the same time as other documents, what web pages were visited, and so on. Some of this information is the kind user would like to actively exploit ("I want the mail I sent to Lynn last week"). Other information can tell the system about connections between objects (if two documents were edited at the same time, they are probably related).

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.

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.

Tools for detecting attention (discussed above) 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.

3.3 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. 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.

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.

4 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 [SAB94]. 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.

4.1 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 [BW97] 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 [App] 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 [RMO+93] 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 [WM96] and the Warwick framework [Lag96] 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. [Met] 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.

4.2 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. [BSY97] 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.

4.3 Collaborative Filtering

Another approach to filtering is *collaborative filtering.* The Information Tapestry of Goldberg, Oki, Nichols and Terry [GONT92] 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 [RIaBR94] 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 [HHWM92] 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. [DST] What's Hot is an internet search engine that places a check-box 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.

5 Results of Prior Support

Lynn Andrea Stein

This section reports results from NSF Young Investigator Award #IRI–9357761, "Embodiment Informs Cognition."

This award covers research into cognitive architectures and embodied agents. More recently, the award has been used to generalize lessons learned in this specific context to the whole of computer science. In particular, emphasis has been given to the implications of this research agenda for introductory computer science education.

Professor Stein and her students have completed the following work, supported wholly or in part by this grant:

Cognitive Robotics:

- Design and implementation of a navigation plan representation for reactive robots which supports real-time natural language dialog with a human guide.
- Work on synthetic language learning (including context-dependent and compositional languages) and models of multi-agent reinforcement learning.
- Construction of a sophisticated humanoid robotic platform to serve as a substrate for future research.
- Implementation of several active visual systems for prototypes of the humanoid robot, including tracking/pursuit, saccade-to-motion, and contour-following.
- Design and construction of a system for learned hand-eye coordination on the humanoid robot.
- Preliminary implementation of a robust platform for hand-eye coordination experimentation and exploration of cognitive architectures.
- Exploration of alternatives to traditional modular-functionalism in cognitive architectures and computer science.

Software Agents and Information Infrastructure:

- Development of a software agent environment and construction system.
- Integration of the software agent system into the ARPA-Funded Intelligent Room HCI Project.
- Integration of the software agent system and the Intelligent Room with an early prototype of the PI's Haystack Per-User Information Environment and other exploration of information management systems.

Computer Science Education:

- Development of a curriculum outline for a revolutionary introductory computer science course based on embedded interactive computation.
- Further development, classroom testing, and preliminary dissemination of the introductory computer science curriculum.

Some of the most research most relevant to the current proposal is described in greater detail below.

Software Agents

We have previously developed SodaBot, a software agent environment and construction system designed to facilitate the rapid prototyping and deployment of personal and application-based software agents as well as communities of semi-autonomous associate systems. The system is designed to make installation easy across platforms and to reduce the overhead of introducing new agents.

Subsequently, we used SodaBot as the "brain" of the Intelligent Room HCI Project. This project explores advanced human-computer interaction and collaboration technologies. The objective of this project is to create an intelligent room capable of interpreting and augmenting activity occurring within it. To this end, a physically instrumented room has been constructed and vision, speech recognition and production systems, and other I/O systems have been connected.

The architecture of this Room's brain involves a hierarchy of agents which interconnect the Room's sensors and effectors and link to external information sources such as the lab's START system, standard information retrieval systems, and the World-Wide Web. Low-level agents interact directly with Room hardware (e.g. cameras) and software applications, while intermediate agents reconcile and consume the information they gather.

Most recently, we focused on the integration of information resources into with this project. Specifically, we used the software agent infrastructure to build a bridge to an early prototype of our Per User Information Environment, resulting in a multi-modal ubiquitous information management system.

[See especially pubs. [C96], [C97b], [C97a], [KS96], [KS97], [K97], [St97c], [To95b] [To95a],]

David R. Karger

This section reports on results of NSF CAREER Award #CCR–9624239, "Random Sampling in Graph Optimization Problems."

This award covers research into applications of randomization to make algorithms faster and or simpler (preferably both). Over the course of the grant, the following work has been accomplished:

- Design of a near linear-time algorithm for computing network connectivity.
- Design of new algorithms for the maximum flow problem in undirected graphs that shatter long-standing bounds for the problem.
- Implementation of several algorithms for computing network connectivity, for the purpose of determining which work best in practice.
- Implementation and experimental analysis of algorithms for determining the reliability of networks subject to random link failures.
- Development of "consistent hashing,"a technique for extending hashing to unpredictably changing environments, such as the web, and using it for load balancing and caching.
- Application of techniques from combinatorial optimization (the Traveling Salesman Problem) to improving code layout in high-performance compilers.
- Implementation of a prototype version of the Haystack information retrieval system.

Algorithm Design

One focus of our work has been the development of new efficient algorithms for perhaps *the* core problems of graph optimization: finding network connectivity and computing maximum flows. These problems arise frequency in practice and simultaneously have a huge amount of theoretical depth; they have been the subject of intense study for over 40 years. Until recently, despite continuous progress, the field seemed pretty settled: finding a maximum flow took $O(mn)$ time on an m-edge, n-vertex graph, while computing network connectivity required several maximum flow computations and thus took somewhat longer.

We applied random sampling to challenge these bounds. Like statisticians, we use a random sample of the given problem (more precisely, a random subproblem generated by choosing a small random subset of the input graph edges) to learn about the structure of the whole. The fact that the sample is small means that studying it and extrapolating from it can be done quickly. This led to an algorithm for finding the connectivity of a network in close to $O(m)$ time on an m-edge graph, an improvement by a factor of n. Similar progress has been made in improving bounds for the maximum flow problem.

Resulting publications: [BK96], [Kar96], [Kar97], [Kar98b], [Kar98a], [KL98].

Algorithm Implementation

Advances in algorithm design can benefit a larger group if they are coupled with algorithm implementations. Over the course of this grant we have carried out two algorithm implementation experiments. In the first, we implemented four recently developed algorithms for determining the connectivity of a network. We studied the behavior of the programs on real-world inputs and also developed "hard" inputs for the algorithms in an attempt to identify their weaknesses, and then added heuristic modifications to deal with these weaknesses and improve the performance of the algorithms in practice. Our final implementations were dramatically faster (often by a factor of 100) than previous implementations (run on the same computer for fairness).

In a more modest undertaking, we also implemented and tested an algorithm recently developed by Karger for estimating the reliability (probability of remaining connected) of a network suffering random edge failures.

Resulting Publications: [CGK+97], [KT97].

Applied Algorithms

One of our interests is in finding applications of algorithmic ideas to problems arising in other areas of computer science.

One domain where we have pursued this approach is compilers. Modern compilers attempt to lay out code blocks to minimize the pipeline delays induced by conditional branches in the code. We showed that this problem could be formulated as a Traveling Salesman Problem; carried out this formulation for real code and solved the resulting Traveling Salesman Problems. This let us provide better code layouts than had bound previously. Equally important, it demonstrated the limitations of the layout process: since it gave optimal layouts, we know that no layout, no matter how found, can be better.

In conversation with the World Wide Web Consortium, we identified another problem ripe for a theoretical attack. Users of the web often experiences long delays in accessing information. These delays are caused by congested links and overloaded servers. The web consortium asked us to develop methods that could reduce this congestion and balance load among servers. We developed a new theoretical tool, consistent hashing, that lets us bring the load balancing benefits of traditional hashing to the ever-changing and unpredictable domain of the internet. We developed algorithms which guarantee balanced load among servers. We are presently in the process of implementing these algorithms to study their performance in practice.

Resulting Publications: [YJKS97], [KLL+97].

6 Biographical Sketches

6.1 David R. Karger

David R. Karger (A.B. *summa cum laude* in Computer Science, 1989, Harvard College; Certificate of Advanced Study in Mathematics 1990, Churchill College, Cambridge University; Ph.D., 1994, in Computer Science, Stanford University) is an Associate Professor of Computer Science and a member of the Laboratory for Computer Science. He was the recipient of a David and Lucille Packard Foundation Fellowship (1997–2002), an Alfred P. Sloane Foundation Fellowship (1996–1998), the 1997 Mathematical Programming Society Tucker Prize, the 1995 ACM Doctoral Dissertation award, and a 1995 NSF CAREER grant. 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 progress on several core problems. He is the author of more than 20 publications and has served on the program committees of some of the most important theoretical computer science conferences. He is a member of the executive committee of SIGACT, ACM's theoretical computer science group.

Karger's 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.

Professor Karger's World Wide Web home page can be found at the following address:

`http://theory.lcs.mit.edu/~karger/`

Two publications relevant to proposed work:

1. D. Cutting, D. R. Karger, J. Pedersen, and J. 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.

2. D. Cutting, D. R. Karger, and J. 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.

Five Other Significant Publications:

1. D. R. Karger, P. N. Klein, and R. E. Tarjan. "A Randomized Linear-Time Algorithm to Find Minimum Spanning Trees." *Journal of the ACM*, 42(2):321–328, 1995.

2. D. R. Karger and C. Stein. "A New Approach to the Minimum Cut Problem." *Journal of the ACM*, 43(4):601–640, July 1996. Preliminary portions appeared in SODA 1992 and STOC 1993.

3. D. R. Karger, R. Motwani, and M. Sudan. "Approximate Graph Coloring by Semidefinite Programming." *Journal of the ACM*, to appear. Preliminary version in *Proceedings of the* 35^{th} *Annual Symposium on the Foundations of Computer Science*, pages 2–13. IEEE, IEEE Computer Society Press, November 1994.

4. D. R. Karger. "Minimum Cuts in Near-Linear Time." In *Proceedings of the* 28^{th} *ACM Symposium on Theory of Computing*, pages 56–63. ACM, ACM Press, May 1996.

5. D. R. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. "Relieving Hot Spots on the World Wide Web." In *Proceedings of the* 29^{th} *ACM Symposium on Theory of Computing*, ACM Press, May 1997.

Students supervised:

Matt Levine
Daniel Lewin
Sudipta Sengupta

Raj Iyer
Eytan Adar
Mark Asdoorian
Mehul Shah
Jack Fu
Jaime Coffin

Other Collaborators within the last 48 months:

Andras Benczur
Michaelangelo Grigni
Philip Klein
Cliff Stein
Lynn Stein
Joel Wein

6.2 Lynn Andrea Stein

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 a Fellow of the Kiss Institute for Practical Robotics.

Professor Stein's research spans the fields of cognitive robotics, common-sense reasoning, software agents, human-computer interaction and collaboration, and object-oriented programming. She is author or co-author of more than 50 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. 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 World Wide Web home page can be found at the following address:

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

Five publications relevant to proposed work:

- Stein, L. A., "Intelligence and Reason: A Response to Etzioni," Letter to the Editor, AI Magazine *15 (2)*, 11–12, Summer 1994.
- Stein, L. A., "Science and Engineering in Knowledge Representation and Reasoning," AI Magazine *17(4)*, 77–83, Winter 1996.
- Stein, L. A., "Eye of the Beholder," *IEEE Expert*, *12 (3)*:5,4, May-June 1997.
- Stein, L. A., "Beyond Objects," Educator's Symposium, *Conference on Object Oriented Programming Systems, Languages, and Applications*, Atlanta, Georgia, October 1997.
- Stein, L. A., "Philosophy as Engineering," Invited Commentary, Computational Intelligence *10 (2)*, 99–102, February 1994.

Five other significant publications:

- Stein, L. A., "Post-Modular Systems: Architectural Principles for Cognitive Robotics," Cybernetics and Systems *28(6)*:471–487, September 1997.
- Stein, L. A., "Rethinking CS101: Or, How Robots Revolutionize Introductory Computer Programming," to appear in *Computer Science Education*.

- Stein, L. A., "Imagination and Situated Cognition," Journal of Experimental and Theoretical Artificial Intelligence, *6*, 393–407, 1994. Reprinted in *Android Epistemology*, K. M. Ford, C. Glymour, and P. J. Hayes, eds., AAAI Press/The MIT Press, 1995, pp. 167–182. Earlier version appears as MIT AI Lab Memo *1277*, February 1991.
- Yanco, H., and L. A. Stein, "An Adaptive Communication Protocol for Cooperating Mobile Robots," *¿From Animals to Animats: Proceedings of the Second International Conference on the Simulation of Adaptive Behavior*, J.-A. Meyer, H. L. Roitblat and S. W. Wilson, eds., The MIT Press/Bradford Books, 1993, pp. 478-485. Also appears as MIT AI Lab Memo *1379*, November 1992.
- Brooks, R. A., and L. A. Stein, "Building Brains for Bodies," Autonomous Robotics *1 (1)*, 7–25, 1994.

Students supervised:

Ian Douglas Horswill
Ellen R. Spertus
Antonio Mercado Jr.
Michael H. Coen
Mark Torrance
Holly Yanco
Mike Wessler
Josh Kramer
Eytan Adar
Mark Asdoorian
Craig Henderson

Other Collaborators within the last 48 months:

Robert Duvall
James Hendler
David Karger
Rodney Brooks

References

[ACM97a] ACM. *Proceedings of the 29^{th} ACM Symposium on Theory of Computing.* ACM Press, May 1997.

[ACM97] ACM-SIAM. *Proceedings of the 8^{th} Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1997.

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

[BK96] András A. Benczúr and David R. Karger. Approximate s–t min-cuts in $\tilde{O}(n^2)$ time. In Miller [Mil96], pages 47–55.

[BSY97] Marko Balabanovic, Yoav Shoham, and Yeogirl Yun. An adaptive agent for automated web browsing. Technical Report CS-TN-97-52, Stanford University, 1997.

[BW97] 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`.

[BS94a] Brooks, R. A., and L. A. Stein, "Building Brains for Bodies," *AIAA/NASA Conference on Intelligent Robots for Field, Factory, Service, and Space*, Houston, Texas, March 1994.

[BS94b] Brooks, R. A., and L. A. Stein, "Building Brains for Bodies," Autonomous Robotics *1 (1)*, 7–25, 1994.

[CGK+97] Chandra C. Chekuri, Andrew V. Goldberg, David R. Karger, Matthew S. Levine, and Cliff Stein. Experimental study of minimum cut algorithms. In *Proceedings of the 8^{th} Annual ACM-SIAM Symposium on Discrete Algorithms* [ACM97], pages 324–333.

[C94a] Coen, M. H., Letter to the Editor, AI Magazine *15 (2)*, 9–10, Summer 1994.

[C94b] Coen, M. H., "SodaBot: A Software Agent Environment and Construction System," student abstract in *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, Washington, August 1994.

[C94c] Coen, M. H., "SodaBot: A Software Agent Environment and Construction System," MIT AI Lab Technical Report 1493, November 1994.

[C95] Coen, M. H., "SodaBot: A Software Agent Environment and Construction System (Extended Abstract)" in Proceedings CIKM '94 — Intelligent Information Agents Workshop, November, 1995.

[C96] Coen, M. H., "Intelligent Information Retrieval," Area Examination Paper, 1996.

[C97a] Coen, M. "Towards Interactive Environments: The Intelligent Room." Proceedings of HCI-97.

[C97b] Coen, M. H., "Building Brains for Rooms: Designing Distributed Software Agents." Proceedings of Innovative Applications of Artificial Intelligence (IAAI) '97.

[CKPT92] 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, Copenhagen, Denmark, 1992.

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

[FBY92] William B. Frakes and Ricardo Baeza-Yates, editors. *Information Retrieval: Data Structures and Algorithms.* Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[GONT92] 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.

[HHWM92] 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.

[HS94] Horswill, I. D., and L. A. Stein, "Life after Planning and Reaction," *AAAI Fall Symposium on the Control of Intelligent Systems*, New Orleans, Louisiana, November 1994.

[Kar96] David R. Karger. Minimum cuts in near-linear time. In Miller [Mil96], pages 56–63.

[Kar97] David R. Karger. Using random sampling to find maximum flows in uncapacitated undirected graphs. In *Proceedings of the* 29^{th} *ACM Symposium on Theory of Computing*, pages 240–249. ACM, ACM Press, May 1997.

[Kar98a] David R. Karger. Better random sampling algorithms for flows in undirected graphs. In *Proceedings of the* 9^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 490–499. ACM-SIAM, January 1998.

[Kar98b] David R. Karger. Random sampling in cut, flow, and network design problems. *Mathematics of Operations Research*, 1998. To appear. A preliminary version appeared in STOC 1994.

[KL98] David R. Karger and Matthew Levine. Finding maximum flows in simple undirected graphs faster than bipartite matching. In *Proceedings of the* 29^{th} *ACM Symposium on Theory of Computing*. ACM, ACM Press, May 1998.

[KLL+97] David R. Karger, Eric Lehman, Tom Leighton, Matt Levine, Daniel Lewin, and Rina Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *Proceedings of the* 29^{th} *ACM Symposium on Theory of Computing* [ACM97a], pages 654–663.

[KS96] Karger, D., and L. A. Stein, "Haystack," June 1996.
http://www.ai.mit.edu/projects/haystack/karger-stein-9606.html

[KS97] Karger, D., and L. A. Stein, "Haystack: Per-User Information Environments," February 1997.
http://www.ai.mit.edu/projects/haystack/karger-stein-9702.html

[KT97] David R. Karger and Ray P. Tai. Implementing a fully polynomial time approximation scheme for all terminal network reliability. In *Proceedings of the* 8^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms* [ACM97], pages 334–343.

[K97] Kramer, J. D., *Agent-Based Information Retrieval*. M.Eng. Thesis. MIT Department of Electrical Engineering and Computer Science. May 1997.

[Lag96] Carl Lagoze. The warwick metadata workshop. *DLIB Magazine*, August 1996. `http://www.dlib.org/dlib/july96/lagoze/07lagoze.html`.

[MSW96] Marjanović, M., Scassellati, B., Williamson, M., "Self-Taught Visually Guided Pointing for a Humanoid Robot," to appear in *From Animals to Animats IV: The Fourth International Conference on the Simulation of Adaptive Behavior*.

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

[Mil96] Gary Miller, editor. *Proceedings of the* 28^{th} *ACM Symposium on Theory of Computing*. ACM, ACM Press, May 1996.

[RIaBR94] Paul Resnick, Neophytos Iacovou, Mitesh Suchak and Peter 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/`.

[RS97] Ringrose, R., and L. A. Stein, "Learning Hand-Eye Coordination by Associating Local Linear Models," Workshop on the Trainable Modular Vision Project, American Academy of Arts and Sciences, Cambridge, Massachusetts, January 1997.

[RMO+93] 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.

[SAB94] 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.

[Sc95] Scassellati, Brian, "High-level Perceptual Contours from a Variety of Low-level Physical Features," M. Eng. Thesis, Department of Electrical Engineering and Computer Science, May 1995.

[Sc96] Scassellati, B., "Mechanisms of Shared Attention for a Humanoid Robot," *AAAI Fall Symposium on Embodied Cognition and Action*, Cambridge, MA, November 1996.

[Smi82] Brian Cantwell Smith. Reflection and semantics in a procedural language. Technical Report 272, Massachusetts Institute of Technology Laboratory for Computer Science, Cambridge, Massachusetts, January 1982.

[St94] Stein, L. A., "Intelligence and Reason: A Response to Etzioni," Letter to the Editor, AI Magazine *15 (2)*, 11–12, Summer 1994.

[St9a] Stein, L., "Rethinking CS101: Or, How Robots Revolutionize Introductory Computer Programming," to appear in *Computer Science Education* Special Issue on Robotics in Computer Science and Engineering Education.

[St96b] Stein, L., "Science and Engineering in Knowledge Representation and Reasoning," *AI Magazine* Winter 1996.

[St96c] Stein, L. A., "Architectures for Cognitive Robotics," November 1996.
http://www.ai.mit.edu/projects/cognitive-robotics/nmod-prop.ps

[St96d] Stein, L. A., "Science and Engineering in Knowledge Representation and Reasoning," *AI Magazine* 17 (4):77-83, Winter 1996.

[St97a] Stein, L. A., "Post-Modular Systems: Architectural Principles for Cognitive Robotics," to appear in *Cybernetics and Systems*, 28(6), September 1997.

[St97b] Stein, L. A., "Interactive Programming: Revolutionizing Introductory Computer Science," *Computing Surveys*, 28A (4). Earlier version appears in the working notes of the ACM Strategic Directions in Computing Research Working Group on Computer Science Education.
http://www.ai.mit.edu/projects/cs101/sdcr.html)

[St97c] Stein, L. A., "Eye of the Beholder," IEEE Expert, 1997.

[St97d] Stein, L. A., "Beyond Objects," Educator's Symposium, Conference on Object Oriented Programming Systems, Languages, and Applications, Atlanta, Georgia, October 1997. Earlier version appears as http://www.ai.mit.edu/projects/cs101/beyond-objects.ps.

[St97e] Stein, L. A., "Non-Algorithmic Computation: A Research Agenda."

[To94a] Torrance, M. C., "Natural Communication with Mobile Robots," Sc.M. Thesis, January 1994.

[To94b] Torrance, M. C., "Two Case Studies in Active Language Use," *AAAI Spring Symposium on Active Natural Language Processing*, Stanford, CA, March 1994.

[To95a] Torrance, M. C., "Advances in Human-Computer Interaction: The Intelligent Room," in *CHI '95 Research Symposium*, Denver, CO, May 1995.

[To95b] Torrance, M. C., "Active Notebook: A Personal and Group Productivity Tool for Managing Information," *AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval*, Cambridge, MA, November 1995. (Copies to be sent under separate cover.)

[We94] Wessler, M., "A Modular Visual Tracking System," student abstract in *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, Washington, August 1994.

[WM96] Stuart L. Weivel and Eric J. Miller. Dublin core metadata element set. `http://www.oclc.org:5046/research/dublin_core/`, May 1996.

[Ya94a] Yanco, Holly A., "Robot Communication: Issues and Implementation," Sc.M. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology. May 1994.

[Ya94b] Yanco, H., "Synthetic Robot Language Development," student abstract in *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, Washington, August 1994.

[YJKS97] Cliff Young, David S. Johnson, David R. Karger, and Michael D. Smith. Near-optimal interprocedural branch alignment. In *ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 183–193, Las Vegas, NV, 1997. ACM.