Motivation and Introduction

In recent years, we have experienced an explosion in the amount of information available online. Unfortunately, tools which allow users to access this information are still quite rudimentary. Users are forced to express their information needs in boolean query languages. More, results returned are often unnecessarily redundant and poor in quality, partly because the user is unable to specify his needs in terms of a query well enough, and partly because of the stupidity of the software servicing his query. More intelligent systems allow users to pose their information needs in the form of a question [Burke, et al., 1997]. Nonetheless, these kinds of systems still require the user to make his information needs explicit to the system. Thus, while Internet search engines provide a first step at solving this information access problem, most of them not only fail to produce good results reliably, but are also hard to use. Question-answering systems provide a solution to part of this problem, yet remain inconvenient.

In response to the problems posed by the current state of information retrieval systems, we are working on a class of systems we call Personal Information Management Assistants (PIMAs). PIMAs observe user interaction with everyday applications, use these observations to anticipate a user's information needs, and then automatically fulfill these needs by accessing Internet information sources, filtering the results and presenting them to the user. Essentially, they allow everyday applications to serve as interfaces for Internet information systems. For the remainder of this paper, we present our preliminary work on an architecture for this class of systems, and our progress implementing such a system. Finally we discuss our preliminary results and survey directions for future work.

One of the main insights driving our work is that information-seeking behavior, such as posing a query to a search engine, is goal-directed behavior. In this view, posing a query to a search engine is a step in a plan that satisfies the goal of finding information about a
certain topic. Given that finding information is usually in service of some goal, we can construct a library of information-consumption scripts (using "script" in the sense of [Schank and Abelson, 1977]) associated with satisfying a user's goals. Scripts are knowledge structures that house information about highly routine situations. In an appropriate context, they serve the purpose of making strong predictions about situations and sequences of events. For a PIMA, knowledge of a user's information-consumption scripts means the ability to anticipate information-seeking goals and the ability to automatically fulfill them.

**Architectural Overview**

We have built a prototype PIMA that observes user interaction with everyday applications (e.g., Navigator, Explorer, and Word), and, using very preliminary knowledge of information-consumption scripts, is able to anticipate a user's information needs. It then attempts to automatically fulfill them using common Internet information resources. Given the requirements that it must observe several applications and that it must also use multiple Internet information resources, we have adopted a five-tiered architecture (see Figure 1).

The user interacts with the sundry applications shown at the bottom of the diagram, and the information management application in the middle. Through a series of adapters in the green layers, the assistant application communicates with the existing software applications through the operating system's IPC facilities. The assistant then interprets user behavior in these applications, and constructs a query which it sends off to information sources at the top. It collects the results, and applies several heuristics which allow it to present the user a concise, high-quality list of suggestions. These suggestions
are presented in a window for the user to browse. Eventually, we plan to give our PIMA a memory of user interests and expertise, as well as the ability to communicate with other users' assistants, in order to personalize and improve the quality of the results.

**Implementation**

Currently, our PIMA observes user interaction in unmodified versions of Microsoft Internet Explorer and Microsoft Word, as well as a modified version of Mozilla (Netscape's free-source version of Navigator). The PIMA communicates with Microsoft Word and Internet Explorer through COM (PC only), and with Mozilla through BSD-style sockets (UNIX and PC). We designed our architecture with the idea that application interfaces should be the only OS/Software dependent components. We implemented the assistant application in Java, for maximum portability. These design decisions afford us the ability to extend the PIMAs field of observation relatively easily, without having to change the core application code.

**Finding Relevant Pages**

The simplest of the information-consumption scripts we have identified is associated with finding related web pages. The FIND-RELATED-PAGES script is composed of the following basic steps:

1. Summarize the document in terms of a few words.
2. Pose a query to a search engine using these words.
3. Sift through the results, searching for ones that are actually related.

It is usually applied when the user is interested in augmenting their knowledge of the subject of the document at hand. We make the assumption that the user is usually interested in documents relevant to the one he is reading or composing. For the two Web browsers, our PIMA recognizes when a user navigates to a new web site, either by clicking on a link, or by explicitly opening a URL. In Microsoft Word, it recognizes when a use has opened a document or changed it significantly. The PIMA responds to these user behaviors by anticipating the user will want to know about Web sites like the document he is looking at. There are essentially two processes associated with retrieving relevant documents: query construction and information filtering.

**Query Construction**

In order to retrieve relevant sites, the PIMA constructs a query based on the contents of the current web page, and sends it off to AltaVista [AltaVista, 1998] (see Figure 2).
In order to construct a query, the PIMA uses three techniques to decide on which words should be included: a standard stop list and two heuristics for rating the importance of a word. The first heuristic is that words at the top of the page tend to be more important than those at the bottom. The second is words that occur with high frequency (that are not in the stop list) are usually representative of the document. The specifics of our term-weighting function are still under development, and as such, beyond the scope of this paper. The terms with the top 20 weights are sent to AltaVista.

Information Filtering

Because the results returned from AltaVista are often redundant, containing copies of the same page or similar pages from the same server, the PIMA must filter the results so as not to add to a user's feeling of information overload [Maes, 1994]. If these similarities are not accounted for, some of the more interesting pages returned by AltaVista may be missed. Moreover, we constantly face the risk of annoying the user instead of helping him. As a result, we actively attempt to reduce the amount of spurious information presented, and in doing so address some of the problems associated with a constantly-updating interfaces (like [Lieberman, 1997]). To this end, we have designed our
prototype to collect search engine results and cluster similar pages, displaying a single representative from each cluster for the user to browse.

For this task, we currently use three pieces of information AltaVista returns for each document: the document's title, its URL, and the date on which it was last modified. For each of these pieces we have developed a heuristic to numerically describe the degree of similarity of two pages. The similarity of two titles is represented numerically by the percentage of words they share in common; two URLs are compared by examining their host, port and directory structure; and two dates are judged by the number of days separating them. The combination of these three heuristic similarity metrics is sufficient to determine the uniqueness of the documents returned, allowing us to provide more intelligent output to the user.

Table 1 shows a typical response from AltaVista generated by a query posed in response to a page on Java Standardization (we have deleted several long ones for brevity). Notice there are a number of mirror sites, as well as logical duplicates (they may have different URLs, but they are the same file). Table 2 shows these URLs after clustering. Instead of presenting the user with 20 sites, we present him with 10, effectively removing the duplicates and mirrors.

| Informal XML API Standardization for Java | http://xml.datachannel.com/xml/dev/XAPIJ1p0.html |
| Java Standardization | http://java.sun.com/aboutJava/standardization/index.html |
| Java Standardization | http://www.javasoft.com/aboutJava/standardization/index.html |
| Informal XML API Standardization for Java | http://www.datachannel.com/xml/dev/Commonality.html |
| Java Standardization | http://www.intel.se/design/news/javastand.htm |
| The impact of Java | http://www.idg.net/ |
Table 1: Output of a query (Titles and URLs) generated from a page on Java Standardization

<table>
<thead>
<tr>
<th>Title</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Standardization</td>
<td><a href="http://aidu.cs.nthu.edu.tw/java/JavaSoft/">http://aidu.cs.nthu.edu.tw/java/JavaSoft/</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://www.javasoft.com/aboutJava/standardization/">www.javasoft.com/aboutJava/standardization/</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://www.javasoft.com/aboutJava/standardization/javastd.html">www.javasoft.com/aboutJava/standardization/javastd.html</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://java.sun.com/aboutJava/standardization/index.html">http://java.sun.com/aboutJava/standardization/index.html</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://www.javasoft.com/aboutJava/standardization/index.html">http://www.javasoft.com/aboutJava/standardization/index.html</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://www.intel.se/design/news/javastand.htm">http://www.intel.se/design/news/javastand.htm</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://www.idg.net/new_docids/find/java/suns/standardization/submitter/approval/affects/new_docid_9-48305.html">http://www.idg.net/new_docids/find/java/suns/standardization/submitter/approval/affects/new_docid_9-48305.html</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://java.sun.com/aboutJava/standardization/javastd.html">http://java.sun.com/aboutJava/standardization/javastd.html</a></td>
</tr>
<tr>
<td>Informal XML API Standardization for Java</td>
<td><a href="http://xml.datachannel.com/xml/dev/XAPI1p0.html">http://xml.datachannel.com/xml/dev/XAPI1p0.html</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://www.datachannel.com/xml/dev/Commonality.html">http://www.datachannel.com/xml/dev/Commonality.html</a></td>
</tr>
<tr>
<td>change nothing - JavaWorld - October 19</td>
<td><a href="http://www.javaworld.com/javaworld/jw-10-1997/jw-10-iso.html">http://www.javaworld.com/javaworld/jw-10-1997/jw-10-iso.html</a>,</td>
</tr>
</tbody>
</table>

Table 2: Results of clustering search engine responses.

**Exploiting Structural Context: An Example**

To demonstrate the power of this paradigm of interaction, we have programmed our PIMA to recognize when a user inserts a caption. The script associated with this situation suggests a different class of information-consumption behaviors we can anticipate: those that are dependent on the structural context of the active document. The FIND-RELATED-IMAGES script is applied when the user has inserted a caption with no image.
to fill it (and probably others---we make no claims about being exhaustive, here). It contains the following steps:

1. Summarize the desired picture in terms of a few words.
2. Send these words off to an image search engine.
3. Sift through the results and choose the best one.

In response to the above situation, the PIMA applies this script and predicts the user will require an image. It then sends off a query to Arriba Vista [Arriba Vista, 1998], a commercial implementation of WebSeer [Frankel, et al., 1997], an image search engine. The PIMA constructs the query using a piece of knowledge about the structure of documents, in general: that the caption of an image is a verbal summary of that image. Hence the query it constructs is simply the conjunction of the stop-listed terms in the caption. The results are presented in a web browser window, from which the user can drag-and-drop the images into Microsoft Word (see Figure 3).

Discussion and Directions for Future Research

Our initial observations suggest that the combination of our heuristics for query generation and for response clustering produce very high-quality results. Our hypothesis is that this is due to the fact that the query generation algorithm we apply to documents roughly mirrors the process of document indexing, and that the clustering heuristics are effective. While our initial results are promising, the system has much room for improvement. Most obviously, our library of scripts is very sparse. Augmenting it so it understands more user/application interactions (and thus is able to anticipate more kinds
of information needs) will be of primary concern. Moreover, the query construction
process ignores the structure of the documents it uses to produce them. Applying
heuristics to improve the query construction algorithm based on document structure will
be fairly straightforward to implement. Moreover, queries frequently include terms that
are of little information value to vector-space retrieval systems like AltaVista.
Composing a table of term frequencies from a random sample of web documents and
using this table to negatively weight terms with very high frequencies will increase the
number of "quality" query terms AltaVista receives. As a further improvement, we plan
on adding support for more information resources and developing a vocabulary for
expressing the kind of information available, as well as a means by which the assistant
can be made aware of new information resources, similar to [Doorenbos, 1997]. Finally
our prototype is reactive in the strictest sense---it has no memory, and knows nothing
about what the user prefers. Giving our PIMA the ability to learn user preferences and
leverage this knowledge as it attempts to anticipate information needs and select
appropriate sources is sure to improve the quality of suggestions dramatically. Clearly
there is much more to be done.

Conclusion

In summary, we have outlined several major problems associated with contemporary
information access paradigms. In response, we presented an architecture for a class of
systems we call Personal Information Management Systems. These systems observe user
interactions with everyday applications, anticipate information needs, and automatically
fulfill them using Internet information resources, essentially turning everyday
applications into interfaces for information sources. We presented our initial work on a
prototype of this kind of system, and closed with directions for future research.

References


Schoenberg, S. 1997. Question Answering from Frequently-Asked Question Files:
Experiences with the FAQ Finder System. Technical Report TR-97-05, The University of
Chicago, Department of Computer Science.


Chicago, Department of Computer Science.
