Intelligent Agents for Matching Information Providers and Consumers on the World-Wide-Web

Joseph K.W. Lee, David W. Cheung, Ben Kao, Jax Law and Thomas Lee

Department of Computer Science
The University of Hong Kong
Pokfulam Road, Hong Kong
Email: {jkwlee,dcheung,kao,ktlaw,ytlee}@cs.hku.hk

Abstract

In this paper, we discuss the various issues in designing intelligent software systems to assist world-wide-web users in locating relevant information. We identify a number of key components in such intelligent systems. These include a web document database management system, a client-based goal-directed search engine, an intelligent learning agent which discovers users' topics of interest by studying their browsing behavior, and an intelligent agent which monitors "hot" web sites. We give examples and suggestions on how these components are designed and implemented. We also describe the architecture of a prototype system that integrates the various components.

1. Introduction

The "Information Superhighway" (Internet) enables a computer user to be connected to virtually endless numbers of sites on the network. Massive amount of information (such as news, stock price quotes) is being pumped into the superhighway at a great rate around the clock. The World-Wide-Web (WWW) uses the Internet to transmit hypermedia documents between computer users located around the world. Due to its extensive coverage and its enormous commercial potential, the WWW has been gaining much attention lately. Large amount of interesting and valuable information has been made available on the Web for retrieval. In order to fully utilize the power of the WWW as a gigantic information source, it is essential to develop intelligent software systems on top of the Web to assist users in retrieving relevant documents. In this paper, we discuss the various issues in designing such intelligent systems. In particular, we identify the key features and components of the intelligent systems, and propose an approach (and the underlying algorithms) for deducing users' topics of interest from their browsing behavior.

It has been reported that the Web contains more than 30 million web pages [1] (not counting USENET news articles) located on more than 275,600 hosts. While new information sources are being added to the Web at a tremendous rate, large numbers of old articles are being updated regularly. For example, a page reporting the score of a basketball game may be updated once every two minutes. Since it is impossible for a human being to keep track of all this information and changes, there have been proposals on software tools to help users *retrieve*, *locate*, and *manage* Web information. We call these software systems *Web tools*.

1.1. Classifying Web Tools

In general, we can classify Web tools into five levels, according to their intelligence and power.

A level 0 Web tool *retrieves documents for a user under straight orders*. The user has to instruct the tool on *where* the documents are located and *how* they are retrieved by supplying the document URLs (which specify both the transfer protocols and the addresses of the documents). Popular Web browsers, such as Mosaic and Netscape fall into this category.

A browser is primarily a user agent for navigating the Web. Its basic functions are to interpret HTML documents, format and present them at a multimedia console, and to navigate their hypertext structures. Typical browsers offer little assistance to users in locating useful information (a user needs to know exactly what he wants, and exactly where the information is located). They also possess very limited functionality in managing the retrieved information. For example, documents fetched by the browsers are *transient* or *browse-once*, meaning that the documents are no longer available without being re-fetched after the browser

1060-3425/97 $10.00 © 1997 IEEE

session is exited (Although most browsers cache documents, some of them do not keep the cache across sessions. Also, the cached documents are not organized for further querying and retrieval by the users). For rarely updated pages, re-fetching them wastes network bandwidth and causes long response time. Even though most browsers allow users to save a document as a file, they do not provide tools for organizing and managing the saved files. Moreover, the structure of a hypermedia document is very often destroyed after begin saved. For example, the images in the saved documents are not retained and the hypertext links are not maintained.

Higher levels Web tools should help tackle the two important problems, namely, information discovery and document management. In this paper we focus on the information discovery problem. (We will, however, mention a simple document management system in Section 4 and discuss how it can be used to help unraveling users' topics of interest. Interested readers are also referred to the WAIS software [9]). We believe that in the era of information overloading, intelligent software systems that can automatically and effectively match or connect users to relevant information play a key role in information technology. We therefore further categorize Web tools according to how much *assistance* they offer to users in pointing them to the interesting information. In general, the less amount of information a user needs to provide to a tool to locate relevant documents, the higher is the tool's ranking.

A level 1 Web tool should provide a *user-initiated searching facility* for finding relevant web pages. Internet search engines, such as Yahoo!, Alta Vista, and Infoseek are examples of level 1 tools. Most of the search engines take a robot-based approach. They usually work by traversing the WWW visiting a large population of the pages. Information about the pages, such as their titles, subjects, word frequency counts, URLs, etc. are stored and indexed. To find out relevant pages on a particular topic, a user supplies keywords to a search engine which describe the concepts. The keywords are then matched against the huge index for relevant information. Documents that match the user query will be ranked according to the degree of consistency it shares with the supplied keywords. Information about the matching documents, such as their URLs and their brief descriptions will then be sent back to the user.

Depending on whether the Web traversal is driven by user queries, search engines can be further classified as either server-based or client-based. A server-based search engine, such as Alta Vista, traverses the Web off-line, and the traversal pattern is not based on any user queries. The idea is to visit and to keep information about as many Web pages as possible for answering future queries. System resource requirements, such as the amount of disk space needed to maintain the large index, are high. A client-based search engine [4], on the other hand, directs the Web traversal according to certain goals, e.g., a set of keywords specifying a particular user's interests. Much of the Web space that are unlikely to contain documents matching the goals are not visited. Web traversals are thus customized to each individual user's goal. Although a client-based search engine is less demanding on system resources than a server-based one, its use risks causing high demand on network bandwidth and overloading the information sources if many Web users employ their own client-based search engines [10].

Search engines have gained popularity among Web users and are indispensable tools in finding information on the Internet. They are, however, passive in nature. Most search engines do not remember a user's query or his goals across sessions. The users are therefore not notified when new or modified information is found available on the Web until the engines are queried explicitly. They thus response to and act for users *on-demand.*

A level 2 Web tools, on the other hand, should maintain users' profiles and have *an active component for notifying users whenever new relevant information is found.* Examples of level 2 Web tools include WebWatcher[2] and SIFT[14]. As an example, with the SIFT system, users supply a number of keywords that describe topics of interest (called *standing orders*). Every day, SIFT automatically matches new Netnews articles with the standing orders, looking for news that are of interest to the users. Summaries of the relevant news articles are sent to the users via emails.

Most of the existing Web tools require the users to explicitly specify a target or a goal of the search, usually in terms of keywords which describe the information interested. Unsophisticated users may find it hard to formalize a query, and in many cases, they do not know the existence of an interesting topic until an example is given to them. A level 3 Web tool, therefore, should have *a learning and deductive component of user profiles.* Example contents of a user profile include topics of interest (e.g., a user may be interested in the U.S. financial markets) and browsing patterns (e.g., he may read the financial summary report at 7pm every weekday). DiffAgent [8] and Letizia [11] are two experimental systems that track a user's browsing behavior to infer the user's topics of interest and can be considered as level 3 tools.

Finally, to better inform the users of the most up-to-date and relevant information, an intelligent Web tool should also understand the behavior of the information sources (e.g., what are the relevant subjects of a page?

How often is a page updated?) We define a level 4 Web tool to be one that has the *capability of learning the behavior of both information users and information sources*. This knowledge enables the system to match the requirements and behavior of the two sides of the information interchange. For example, if a newspaper page is updated at 5:30am every day and that a user always read that page after 6:30am, an intelligent system could pre-fetch the page at say, 6:00am to reduce the access time to the page.

In this paper, we focus on the techniques in designing a level 4 Web tool. In particular, we study the problem of constructing user profiles from user access patterns. We also discuss issues in monitoring information sources.

We remark that a higher level Web tool does not preclude the needs of lower level ones. In fact, Web tools at different levels should work together to achieve effectiveness and efficiency. For example, a level 3 Web tool that learns a user's topics of interest can submit its learning results as standing orders to a level 2 tool, such as SIFT [14]. A level 2 tool could in turn use a level 1 search engine to find out new information matching the standing orders.

The rest of the paper is organized as follows. Section 2 surveys some related studies on the information discovery problem. Section 3 briefly summarizes the desired features of a level 4 Web tool. In Section 4 we describe a prototype of such a tool. The description includes the prototype architecture and its underlying components. In Section 5 we discuss an implementation of our prototype system and the algorithms used in the learning agents (for both users and information sources profiles). In Section 6, we present the experiment result of using the prototype. Finally, we conclude the paper in Section 6.

2. Related Work

There are a number of studies on the design of intelligent Web tools. In this section we briefly mention four such systems. Interested readers are referred to [2, 3, 4, 6, 11] for more details.

Client-based Searching Tools. De Bra and Post modify Mosaic to incorporate a client-based search engine. The system allows a user to specify a searching goal in three different ways: keywords, regular expression and external filters. Relevant documents are than ranked according to the scores.

Web traversal employs the fish search algorithm [16]: a *fish* is spawn when a page is visited. Given a starting web page (a fish), the fish algorithm controls the number of links to follow (and thus the pages to visit) according to three parameters:

width — determines the fraction of the outgoing links to be followed (the number of children a fish produces).

depth — determines how many links are followed without finding relevant information before the direction is given up (how long the fish can keep reproducing without food).

rate — determines which transfer rate is considered acceptable; outgoing links to sites with low transfer rates are avoided.

DiffAgent. The DiffAgent [8] is an intelligent agent developed by IndustryNet, in collaboration with Carnegie Mellon University, for a news-clip service. A DiffAgent monitors many sites on the Internet and notifies its masters (IndustryNet users) if any changes of relevant Web pages are detected. Besides automatic notification, the agent also learns about its user's interest by studying what he reads. The agent scans each article the user read for keywords and phrases. It also obtains feedback from the user on the relevancy of the document (in terms of a score). Such scores and keywords are used by the agent to create a model of the user's interests. The current limitations of the DiffAgent are (1) it handles only news articles but not other forms of information like WWW documents; and (2) it monitors only a predefined set of sites without the ability to explore new information sources.

WebWatcher. Also developed at Carnegie Mellon University, WebWatcher [2] is a tool that assists users by interactively giving them navigation advises. A user starts a WebWatcher session by specifying what information is sought (e.g., publications, personal home-pages, softwares, etc.). While the user is browsing through Web pages, WebWatcher assesses the hypertext links (and the documents referred to by the links) contained in the pages. It then recommends those links that the system guesses is promising in matching the goal of the session. Recommended links are highlighted on the user's display. While WebWatcher waits for the user's action, it pre-fetches any Web pages it has just recommended, and processes these pages to make further recommendations. WebWatcher tracks the user's response to its recommendations, for example, whether the recommendations are taken, and if so, which are taken. User response is logged for further fine tuning of the system's performance.

In the current implementation of WebWatcher, link recommendation is based on a simple function which estimates the probability that a user will select a particular link given the current page and goal. Information about previous browsing paths are not considered in the predictions. Unlike DiffAgent, WebWatcher does not learn the user's searching goals

automatically (which have to be specified at the beginning of a session).

Letizia. Letizia [11] is an intelligent agent that works with a conventional Web browser such as Mosaic or Netscape. It tracks the user's browsing behavior and tries to infer the user's goals. While the user is reading a page, Letizia conducts a resource-limited search based on the goals it deduced. Relevant pages found during the search will be recommended to the user upon request. The goal of Letizia is to automatically perform some of the Web exploration on behalf of the user to anticipate future page accesses.

3. System Features

Before we discuss the design and implementation issues of our prototype level 4 Web tool, let us state some of the desired features for which such a tool should possess. To reiterate, the goal of the intelligent Web tool is to discover the most updated and the most relevant information to its user with the least amount of user effort and system resources required. Here, we list some desirable features of such tools:

The system should learn its user's topics of interest automatically. Several systems (e.g., [3, 8, 11, 15] have already shown that such interests can be discovered by examining the user's browsing behavior and the contents of the documents he has read. The system should also learn about its user's shift of interests over time. This knowledge of user's interests is used when the system is navigating the Internet on behalf of the user to discover relevant information

The system should learn its user's access patterns and information sources' update patterns. A user may access certain documents on a regular basis. For example, he may read a newspaper front page at 9am and a financial report at 6pm, every day. The system should learn about when the documents are updated at the sources, and retrieve in advance the latest version before the user requests them. For pages that are updated at irregular intervals (such as the price of a certain stock), the system should intelligently monitor the source and notify the user for new updates.

The system should make efficient use of network resources. A system that performs an exhaustive search of the whole Web, for example, may encounter many interesting documents. However, it also creates excessive traffics on the Internet and on the system's network connection, crippling the system. If Web search is required, a goal-directed search is a better alternative than exhaustive search in saving network bandwidth. Searching goals of multiple users should be clusterized. Similar goals should be batched together to reduce the number of Web searches conducted.

The system should maintain a database and a full-text index on retrieved documents. The database serves three purposes: (1) it helps the user to organize and retrieve previously read documents; (2) data mining techniques [5] can be applied to the saved documents to discover the user's interest; and (3) the system can use the set of saved documents as starting point for automatic navigation and exploration.

The system should be compatible with most WWW browsers. There should be no extra requirements for a browser to communicate with the system in addition to the standard HTTP protocol.

4. System Architecture

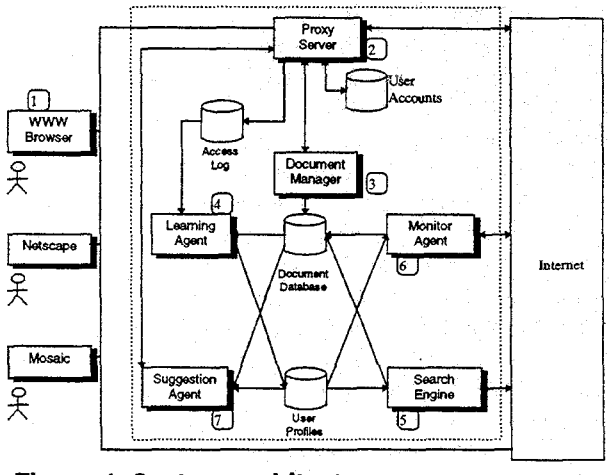

Figure 1. System architecture

In this section we give an overview of our prototype system. Figure 1 shows the system architecture and components. Let us take a guided tour of the system following the flow of information (indicated by the numbers shown beside the major components).

WWW Browsers (1). A user accesses the Internet through a conventional Web browser such as Netscape or Mosaic. If the user chooses to use our system, he simply instructs the browser to connect to a Proxy Server maintained by the system. Since almost all modern browsers are able to use a proxy with simple configuration, users of the system are free to pick their favorite browsers as the interface. To start a session, a user fills in a special HTTP form and send it to the Proxy identifying himself. The browser window will then split into two frames (see Figure 2). The upper frame is the *System Frame* which consists of a number of buttons. These buttons are hypertext links referring to various functions provided by the system. For example, the first

button, when followed, instructs the system to make suggestions on new information sources and interesting Web pages; the second button invokes the search function from the Document Manger (to search the Document Database for documents containing certain keywords). The lower frame displays the normal documents requested by the users. During a session, all HTTP requests go through the Proxy. This mechanism allows the system to track every single document read by the user. We remark that the user can maintain his privacy by quitting the session at any time. The Proxy will be bypassed and further HTTP requests will not be logged after that.

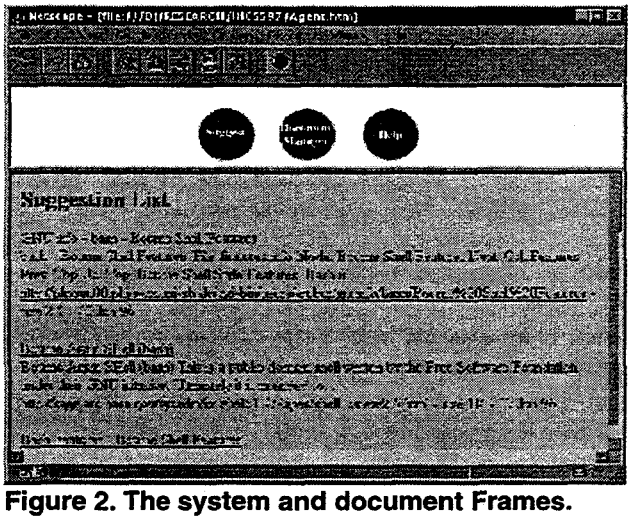

Figure 2. The system and document Frames.

Proxy (2). Users communicate with the system via a WWW proxy server [12]. The Proxy is a special HTTP server that lies between a browser and the Internet. When a user issues a HTTP request, the request is forwarded to the Proxy, which fetches the desired Web document on behave of the user. The retrieved page is then deposited into the Document Database through the Document Manager (to be discussed shortly).

The Proxy serves two important functions to the system. First, it caches the documents retrieved by the users into the Document Database (It also caches documents that are retrieved by the system during automatic exploration of the Web). This helps reducing network traffics when the same document is requested repeatedly by the When submitted a HTTP request, the Proxy will first check with the Document Manager and see if the desired document is already cached at the Document Database. If so, the local copy is returned to the user; else, the Internet is accessed. Second, the Proxy knows every document read by every user. This allows high-level logging of user information be performed. In the prototype each user request generates a log record, which consists of the user's ID, the URL requested, the time of the request, and the document retrieved. All log records are stored in the Access Log database. Information kept in the log is used by the Learning Agent to reconstruct the browsing histories of the users, such as the paths of page accesses, time-related access patterns, statistical summaries of the pages read, and the keywords searched by the users against the Document Database. We will study how the Learning Agent analyzes this information to deduce the kinds of information sought by the users in Section 5.

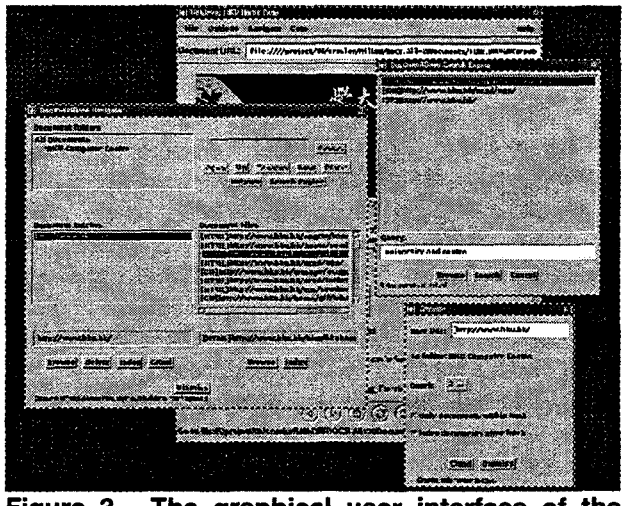

Figure 3. The graphical user interface of the Document Manager.

Document Manager (3). The Document Manager is the interface for accessing the Document Database. It is responsible for storing and retrieving documents deposited by the system (from both user-initiated and system-initiated HTTP requests). The hypertext structures of the saved documents are preserved by reconstructing some of the links. For example, if a page A contains a hypertext link to another page B. When the pages A and B are retrieved and stored as local copies A' and B' respectively in the database, the link in the local page A' will be modified to point to the local copy B'. Links that refer to non-local pages are not changed. Besides providing a persistent storage for the documents, the database also maintains a full-text index on the documents.

Figure 3 shows the graphical user interface of the Document Manager. Three windows on the foreground, each designed to handle a unique function of the Document Manager, are shown. The Document-base Navigator window allows a user to manipulate and navigate the Document Database. For example, a user can organize collections of documents into folders. He can also browse the documents by simply clicking on their

URLs. When the "Crawl" button is depressed, the Crawler window (lower right one) pops up. This window provides an interface for the user to retrieve a batch of documents and the hypertext links among them. This function requires the user to specify a few parameter values such as the URL of a starting page and the *depth* of the traversal. In the example shown in Figure 3, all pages that are within 2 hypertext links away from the Hong Kong University Home Page (http://www.hku.hk/) are retrieved. Finally, the Document Manager provides a searching facility (upper right window in the figure). A user can conduct a keyword search on the Document Database. Both conjunctive and disjunctive queries are allowed.

The Document Manager provides the users with two spaces of hypertext navigation: the Document Database and the World Wide Web. The Document Database provides a relatively small set of documents that are of special interest to the users. The WWW, on the other hand, is a jungle of exciting and useful information waiting for a hunter. When a user discovers interesting documents from WWW explorations, he can move them to the Document Database for future references without having to re-locate them when they are needed.

The Document Manager is written in the Java language [7] except that indexing and keyword searches are implemented using the Wide Area Information Servers (WAIS) software [9].

Learning Agent (4). The Learning Agent discovers user access patterns and topics of interest by analyzing the access log created by the Proxy. It generates a user profile for each user. A profile consists of two types of information:

Information seeking goals. A goal is a set of weighted keywords and phrases that describes a topic that is interesting to the user. For example, the goal: {"Basketball, 0.7", "Chicago-Bulls, 0.4"} shows that the user is interested in information about "basketball" in general (with a weight of 0.7), and the term "Chicago-Bulls" in particular (with a weight of 0.4). These keywords are used to drive the Search Engine for information discovery.

Time-related access patterns. Some documents are requested by one or more users on a regular basis, e.g., newspaper, stock price quotes. A time-related access pattern records the period and location of a periodically accessed document. This information is used by the Monitor Agent for pre-fetching documents.

We will discuss in details the algorithms used in generating the user profiles in Section 5.

Search Engine (5). The Search Engine performs robot-based traversal about the Web. Interesting documents encountered during Web exploration is stored and indexed in the Document Database. In the prototype, the Search Engine is implemented as a goal-directed crawler, employing an algorithm similar to the fish-search algorithm. User goals, generated by the Learning Agent are used to drive the crawler, which tries to avoid visiting Web sites and their pages that are irrelevant to the goals. It could also search the Document Database with the user goals to get a set of starting pages for further exploration.

Monitor Agent (6). The Monitor Agent monitors specific sites and Web pages that are known to contain interesting documents. It serves two functions. First, users can indicate that certain documents should be kept up-to-date. For this type of documents, the Monitor Agent periodically accesses them and learn about the sources' update patterns (like how often is a page updated, and when). With this knowledge, the Monitor Agent schedules future retrieval of the pages and keeps them fresh in the Document Database. The refresh schedule of a document is determined by two factors:

1.The maximum *staleness* (i.e., how out of date a page can be) that is acceptable by the user.
2.Any regularity in the source's update pattern.

As an example, if a source updates a particular page once every hour, and that the user requires that the page be not older than the latest version by 30 minutes. The Agent can refresh the page by the hour. On the other hand, if the page is being updated irregularly, the Agent will need to refresh the page once every 30 minutes.

The second function of the Monitor Agent is to schedule pre-fetches. As we have mentioned, part of the user profiles created by the Learning Agent contains information about pages that are regularly accessed by the users. The Monitor Agent tries to predict when a user will access a particular page and pre-fetches it for the user. Successful pre-fetches can greatly reduce the response time of retrieving the documents. As incorrect predictions imply useless retrievals and wasted network bandwidth, only those pages whose access patterns are highly predictable are pre-fetched.

Suggestion Agent (7). Useful and new documents discovered by the Search Engine and the Monitor Agent are stored in the Document Database and for which the Suggestion Agent is notified. The Suggestion Agent ranks the newly found documents and assigns scores to them according to the extent that they match the user's profiles. A user, when looking for new information, can submit a form to the Suggestion Agent through the Proxy requesting document recommendation. The form asks the user for keywords that describe a topic. The Suggestion Agent then composes a list describing a set of relevant new pages found by the Search Engine and the Monitor Agent to the user. Documents in the list are ranked according to the scores assigned to them by the

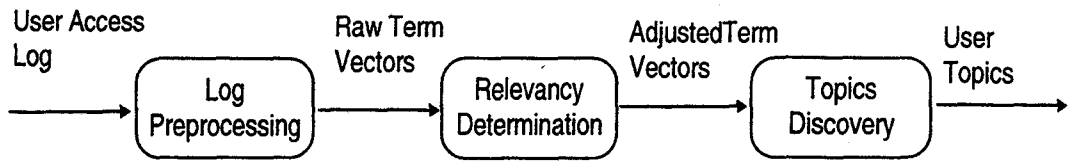

Figure 4. Transforming user access log to topics of interest.

Suggestion Agent, and the degree that they match the user supplied keywords. The description includes their URLs, titles, and the first 10 lines of the pages. Documents recommended can be subsequently retrieved from the Document Database. The Agent will also learn from the user's feedback (such as, which recommendations are taken) to improve the quality of future recommendations.

5. Learning and monitoring

The Learning Agent and the Monitor Agent are two important components of a level 4 Web tool. On the client side, the Learning Agent is capable of identifying the access patterns from the users' access logs, while on the server side, it can discover the update patterns of the web pages collected. The Monitor Agent serves the clients by monitor adaptively the updates on the web pages that users are interested in. The information sources with documents matching the access patterns discovered by the Learning Agent form a domain of monitoring for the Monitor Agent. In this section, we will discuss the approach and the algorithms adopted in our prototype system in the implementation of the Learning and Monitor Agents.

5.1 Discovery of Topics of Interests

We first discuss the simple case of processing the access log of a single user. Subsequently, we will explain how the topics from multiple users can be discovered.

The mechanism to discover the topics of interest is a 3-phase process (Figure 4). The input to the learning agent of the discovery process is the user's access log which consists mainly of a sequence of URLs. We use the CERN httpd as the proxy server to capture the log entries. In the first phase, the document of every URL entry in the log is retrieved and processed to produce a term vector of (keyword, weight) pairs. Entries in the log which do not refer to textual documents (e.g., images, movies, Java applets) are removed before the generation of the term vectors. For example, a term vector including pairs such as (NBA, 0.5), and (basketball, 0.3) may be extracted from a sport document. A modified version of TFIDF [10] is used to compute the weight of the keywords.

The term vectors produced in phase-one could be used to discover the user's topics directly. However, there could be a significant amount of noise in the vectors. Firstly, some Web pages may not be supplying any information to the user, but are visited simply because they provide linkages to other documents. The learning agent has to determine the relevancy of the documents and some particular keywords. Some heuristics have been designed for this purpose. For example, if a Web page has a large number of URL's, it is very likely a directory page, and its relevancy to the discovery should be decreased. An access graph which captures the forward and backward browsing relationship between Web pages is compiled from the access log entries to support the identification of access paths. An access path starts from the user's home page and terminates at a backward browsing movement. Once the access paths are determined, the documents near the end of an access path can be determined and they have a much higher chance of containing some relevant information. Also, the URL's that lead to the highly relevant documents may contain keywords that are very close to the information domain that the user wishes to browse. Weight enhancement are done on the relevant documents and keywords, weight decrement is performed on the irrelevant Web pages, and directory pages which have a large number of URL's exceeding a threshold are removed. The output of the second phase is a sequence of term vectors with adjusted weights.

The last phase of the discovery is to produce the topics of interests. The clustering technique is being used to form the topics. The output are a small number of topic vectors truncated to a small predefined length, such as (NBA, basketball, stadium, arena). Note that this set of keywords represent an area of interest of a user. In fact, these keywords can be input into a goal-directed search engine to retrieve documents that have a high chance of matching to the user's browsing habit.

The clustering is based on the similarity between the term vectors. The similarity $s(v_1, v_2)$ between two term vectors v_1 and v_2 is given by the inner product of v_1

and v_2. The distance s(v_1 , C_2) between a term vector v_1 and a cluster C_2 is given by s(v_1 , c_2), where c_2 is the centroid of C_2 . The centroid of a cluster is a term vector which is the average of all the vectors in the cluster. Adjusted term vectors from phase-two are clustered with respect to a similarity threshold. All term vectors in a cluster have a similarity from its centroid greater than a threshold. If the number of clusters found are too large, the clustering will be repeated with a smaller similarity threshold until the number is small enough. The last step in this phase is to convert the clusters to topics. Since the term vectors in a cluster are very close to its centroid, it is reasonable to use the centroid to represent all the term vectors in a cluster in deriving the topic. However, a centroid in general may have too many keywords and many of them may have relatively small weights. A further selection of keywords from those in a centroid is performed on every cluster. The centroids can be truncated with respect to a predefined length threshold or the keywords in it can be filtered against a weight percentage threshold. At the end, the output will be a small number of topic vectors such as (NBA, basketball, Chicago-Bulls), (NHL, hockey, Canada, Maple-Leaf).

In the following, an example is presented to illustrate the above algorithm in details. Figure 5 is a document recorded in our access log. In processing this document to extra the term vector, not every keyword in the document are regarded to have the same importance. The HTML tag are used to modify the weight of the keywords. For example, those in the title with the tag <TITLE> will have their weights increased, while those in the header under the tag <H1> will received a smaller adjustment, all other keywords in different sections of the document are modified in a similar fashion. Once the keywords are extracted, their weights are calculated by using the TFIDF method. The resulted term vector of this document is ((shell, 0.2994), (bash, 0.2425), (bourne, 0.2379), ...).

```
<HEAD>
<TITLE>Bourne Again SHell (bash)</TITLE>
</HEAD>
<BODY>
<H1>Bourne Again SHell (<SAMP>bash</SAMP>)</H1>

<P>This is a public domain shell written by the
Free
Software Foundation under their GNU initiative.
Ultimately it is intended to be a full
implementation of the IEEE Posix Shell and Tools
specification. This shell is widely used within
the academic commnity.</P>

<P>bash provides all the interactive features of
the C shell
(csh) and the Korn shell (ksh). Its programming
language is compatible with the Bourne shell
(sh).</P>

<P>If you use the Bourne shell (sh) for
<A       HREF="/UNIXhelp/scrpt/index.html">shell
programming</A> consider using bash as your
complete shell environment.</P>

<UL>
<LI><A
HREF="/UNIXhelp/shell/oview1.1.html">Summary of
features</A></LI>
</UL>

</BODY>
</HTML>
```

Figure 5. A sample document. Underlined words are those appear in the term vector.

The above term vector, together with other term vectors extracted from the documents in the access log become a domain for the discovery of areas of interest. These vectors are input into the learning algorithm to produce clusters. The following is a cluster generated by the learning algorithm. The similarity threshold used in this example is 0.1. We present the URLs of the documents in the cluster instead of their term vectors.

- http://theory.uwinnipeg.ca/unixfiles/Unixhelp/shell_oview2.5.html
- http://www.fnal.gov/cd/UNIX/unixhelp/environment_exmp_bash.html
- http://www.chernikeeff.co.uk/data/doc/lightstr/r2_1/hp_os/gnubash.htm
- http://www.lib.ox.ac.uk/internet/news/faq/by_category.unix_faq.shell.html
- http://www.ilap.com/UNIXhelp1.3/Pages/shell/ovew2.2.html

The centroid of the above cluster calculated from its term vector is ((bash, 0.2881), (shell, 0.2792), (bourne, 0.2312), (unix, 0.1138), (csh, 0.1024), (z-shell, 0.08673), (tcsh, 0.03126), ...).

Lastly keywords in the centroid of the cluster with small weight are truncated to produce the topic vector ((bash, 0.2881), (shell, 0.2792), (bourne, 0.2312)).

We have described the algorithm to discover the topic vectors from the log of a single user. The same algorithm can be applied to a log recording the access behavior of multiple users. The topic vectors discovered would be the common areas of interests. If these topic vectors are used by a goal-directed search engine, the retrieved documents would match the common interests of these users. This approach could save a significant percentage of the communication bandwidth required if the search are performed independently by individual users with their own topic vectors.

5.2 Discovery and Monitoring of Hot Pages

Besides discovering the areas of interests, the prototype system also support the discovery of "hot"

pages. Many users have their favorite Web pages and these documents usually are updated frequently. The hot pages can be discovered from the users' access log by checking their occurrences against a threshold. Similar to the areas of interest discovered, the hot pages are also announced to the Monitor Agent so that they can be retrieved in advance from the information sources whenever they are updated.

In order to retrieve the hot pages in a timely fashion, we propose to use an adaptive monitoring algorithm in the prototype system. Initially, the Monitor Agent polls a hot page with a fixed time period to request its last-updated-date in order to determine whether the page has been updated and hence should be retrieved again. When the page's update frequency starts to change, the agent will adjust its polling frequency according to its update frequency. In the prototype, the simple strategy of using the period between the last two updates to determine the time of the next polling is adopted.

The Learning Agent can provide the system with the users' areas of interest and the hot pages. The Monitor Agent can adaptively determine when these "useful" and "hot" documents should be checked for updates. Together, these two agents provide an integrated and intelligent service to bring to the customers the documents and information that they need.

6. Experimental Results

The prototype system is implemented on an AIX system on an RS/6000 workstation of model 410. The proxy server is a CERN httpd. We have collected the user access logs from 9 volunteers for the duration of one month. The logs are then input to the Learning Agent to compute the areas of interest. Some users in the test period were quite active, while some of them were not. Table 1 shows the number of unique term vectors generated for each users.

User	# of term vectors	User	# of term vectors
0	364	5	151
1	299	6	213
2	365	7	25
3	642	8	511
4	1044		

Table 1. Number of URLs accessed by each user.

One of the major step in the discovery of the users' topics of interest is to identify the clusters. The choice of the similarity threshold has a decisive effect in the formation of the clusters. A smaller threshold would facilitate the formation of larger clusters, but they may not reflect a very focusing topic. A large threshold would produce clusters which are more focusing, but also decrease the cluster sizes significantly. Figure 6 plots the number of clusters discovered against different threshold values for all the nine users in our experiment. When the threshold equals to 0, any two term vectors who have some overlapping keywords would be treated as similar. Therefore, the least number of clusters will be generated in that case. We have observed that the number of clusters increases faster when the threshold is at the end of our test range.

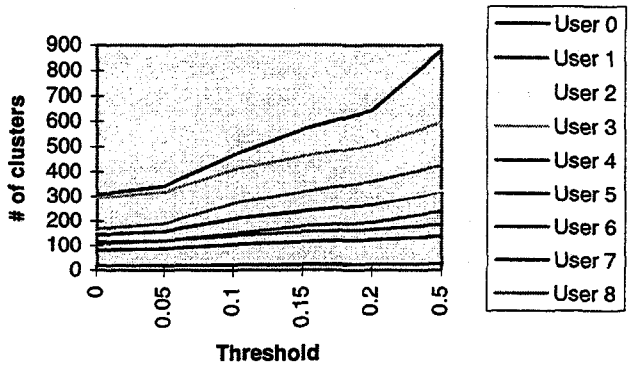

Figure 6. Number of clusters against different threshold values.

In Table 2, we have recorded the clustering results of all the users with respect to different thresholds (T is the threshold value). For each user, the first row (Max) is the size of the largest cluster with respect to a particular threshold. The second row (Mean) is the average size of all the clusters. The third row is the number of clusters generated.

Table 2. Various statistics against the choice of threshold value.

It is also informative to investigate the distribution of the clusters for different thresholds. Figures 7, 8 and 9 are the distributions of the clusters against their sizes and similarity thresholds. User 7 has the least number of clusters, its distribution is in Figure 7. The clustering can only be detected when the threshold is below 0.1. The distribution of user 2 is in Figure 8. This user has the largest number of documents, the clustering effect can be seen on all the threshold values. However, the effect is more visible when the threshold is around 0.1 or less. User 3 is an average user is our experiment, the number of documents in his log is about the average. The distribution of the clusters of this user is in Figure 8. In fact, among all the users in our experiment, it can be seen that the cluster distributions are more visible when the threshold is less than or equal to 0.1.

		T= 0.0	T= 0.05	T= 0.10	T= 0.15	T= 0.20	T= 0.50
User 0	*Max*	19	17	14	12	9	9
	Mean	2.53	2.30	1.73	1.51	1.36	1.14
	# cluster	144	158	210	241	267	320
User 1	*Max*	25	24	22	21	21	12
	Mean	2.68	2.43	1.95	1.61	1.49	1.20
	# cluster	108	119	148	180	194	240
User 2	*Max*	19	17	13	11	10	5
	Mean	2.34	2.13	1.65	1.44	1.32	1.14
	# cluster	156	171	221	253	276	321
User 3	*Max*	24	22	12	8	7	4
	Mean	2.18	2.02	1.57	1.39	1.27	1.08
	# cluster	294	318	408	461	504	597
User 4	*Max*	34	28	24	24	23	21
	Mean	3.40	3.05	2.24	1.84	1.62	1.19
	# cluster	307	342	467	568	643	879
User 5	*Max*	15	11	7	5	5	4
	Mean	1.86	1.74	1.45	1.30	1.22	1.09
	# cluster	81	87	104	116	124	138
User 6	*Max*	12	12	12	12	12	12
	Mean	1.85	1.79	1.52	1.37	1.30	1.15
	# cluster	115	119	140	155	164	186
User 7	*Max*	6	4	3	2	2	1
	Mean	1.32	1.32	1.19	1.14	1.09	1.00
	# cluster	19	19	21	22	23	25
User 8	*Max*	30	19	19	18	15	8
	Mean	3.02	2.69	1.88	1.61	1.42	1.20
	# cluster	169	190	272	318	360	426

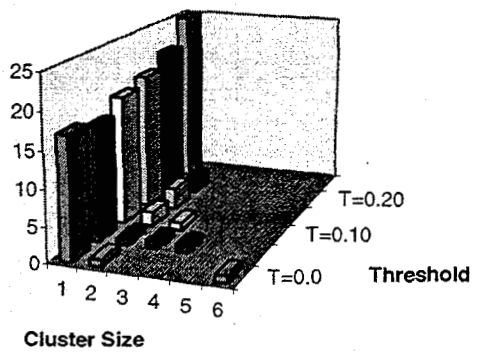

Figure 7. Cluster size distribution of user 7.

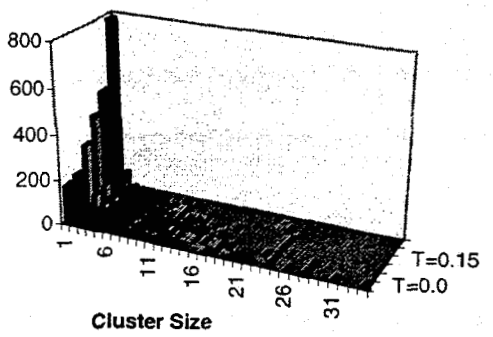

Figure 8. Cluster size distribution of user 4.

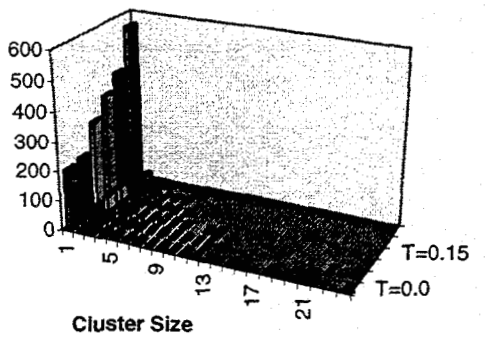

Figure 9. Cluster size distribution of user 3.

7. Discussion and conclusion

The impact of information superhighway to information retrieval is not simply the introduction of just another new technology. It has brought in a revolutionary change in the distribution and collection of information. Very soon, Internet will be connecting people in a large percentage of the households in the world. Everyone would like to be able to access every piece of information in the Internet easily and efficiently. However, this will bring in an unimaginably high bandwidth requirement and a very challenging management problem. These problems can be solved by building intelligent agents to provide sophisticated and smart services. The Web tools proposed in this paper is an attempt in this direction. In this study, we have provided a framework to classify intelligent Web tools into 5 levels (levels 0 - 4). The architecture of a system which has integrated a number of level 4 Web tools has been described. The Web tools can discover the users' accessing behavior and bring in documents that users are anticipating. More importantly, resources for the discovery and retrieval can be shared by multiple users and be performed only when it is necessary

as suggested by the Monitor Agent. It also provides a Suggestion Agent to advice the users on the documents that they may need without browsing through the Internet blindly and hence draining resource. We also have built a Document Database in the system to store those documents that the users have a high chance to access, and these documents are being updated in an efficient way with the help of the Monitor Agent.

To prove its feasibility, we have built a prototype system to demonstrate the ideas proposed in this study. An algorithm for mining the users' topic vectors from their access logs has been implemented, the result shows that the areas of interests discovered are valid and meaningful. An adaptive algorithm for the Monitor Agent to monitor the update frequency of the hot pages have also been proposed.

In the future, more learning capability will be introduced into the Learning Agent. It is very likely that there are "hot page groups" and "hot pages access sequences", and the Learning Agent can be enhanced to support these types of discovery. Another interesting area for future study is the structure of the Document Database. Initially, it is primary a hypermedia structure. It may not be the best structure for the users to find their required information. A relational type structure may provide a more effective structure to handle users' queries. Furthermore, the indices into the documents could be adjusted dynamically according to the information collected. For example, the indices to collections of videos and text documents could be different.

7. References

[1] Alta Vista. http://altavista.digital.com.

[2] Armstrong et al. WebWatcher: A Learning Apprentice for the World Wide Web. *Working Notes of the AAAI Spring Symp.: Information Gathering from Heterogeneous, Distributed Environments.* AAAI Press, 1995, pp.6-12.

[3] M. Balabanovic and Y. Shoham. Learning Information Retrieval Agents: Experiments with Automated Web Browsing. *Proceedings of the AAAI Spring Symposium on Information Gathering from Heterogenous, Distributed Resources,* Stanford, CA, March 1995.

[4] De Bra and R.D.J. Post. Information retrieval in the World-Wide Web: making client-based searching feasible. *Proceedings of the First International World-Wide Web Conference*, R. Cailliau, O. Nierstrasz, M. Ruggier (eds.), Geneva, 1994.

[5] M. S. Chen, J.S. Park and P.S. Yu. Data Mining for Path Traversal Patterns in a Web Environment. *Proceedings of the 16th International Conference on Distributed Computing Systems,* Hong Kong, May, 1996, pp385-392.

[6] O. Etzioni and D.S. Weld. Intelligent Agents on the Internet: Fact, Fiction, and Forecast. *IEEE Expert*, Vol. 10, No. 4, August, 1995, pp.44-49.

[7] *The Java Language Overview.* Available via http://java.sun.com.

[8] D.H. Jones and D. Navin-Chandra. IndustryNet: A Model for Commerce on the World Wide Web. *IEEE Expert*, October 1995, pp.54-59.

[9] B. Kahle and A. Medlar. An Information system for corporate users: Wide Area Information Servers. *Connexions - The Interoperability Report, 5(11): 2-9,* 1991.
M. Koster. Robots in the Web: threat or treat? *ConneXions*, Volume 9, No. 4, April 1995.

[10] H. Lieberman. Letizia: An Agent that Assists Web Browsing. *International Joint Conference on Artificial Intelligence*, 1995.

[11] A. Luotonen and K. Altis. World-Wide Web Proxies. *Proceedings of the First International World-Wide Web Conference*, R. Cailliau, O. Nierstrasz, M. Ruggier (eds.), Geneva, 1994.

[12] G. Salton. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley, Reading, MA. 1989.

[13] T.W. Yan and H.Garcia-Molina. SIFT-A Tool for Wide-Area Information Dissemination. *Proceedings of the 1995 USENIX Technical Conference*, 1995, pp.177-186.

[14] T.W. Yan et. al. From User Access Patterns to Dynamic Hypertext Linking. *Proceedings of the Fifth International World-Wide Web Conference,* Paris, France, May 1996.

[15] http://www.win.tue.nl/2L670/dynamic/fishsearch.html.