WEBSA: DATABASE SUPPORT FOR EFFICIENT WEB SITE NAVIGATION*

Isabel F. Cruz, Lijun Leo Liu, and Tony Y. Wu
ADVIS Research Group
Computer Science Department
Worcester Polytechnic Institute
USA
ifc@cs.wpi.edu

Abstract WebSA (Web Site Agent) is a web page recommendation system that helps users navigate efficiently within a web site. The system discovers the users' access patterns by mining the raw data from a web log file. The user interface of WebSA presents the weighted recommendations in a separate frame of the web page. Two types of recommendations are given: one based on the user's access patterns and the other based on the overall pattern of all the users of that web site. To support the visual interface and the discovery process, WebSA uses database technology to manage, query, update, and concurrently access the web log data. The bulk of the computations for mining and updating the information are done on a daily basis in advance. This method together with the fine tuning of the database guarantees high performance, even when the raw access log data is in the order of 1 GB.

Keywords: collaborative filtering, visual interface, web, data mining, access pattern, database

1. INTRODUCTION

To aid visitors in navigating within a web site, commonly used techniques include site maps, lists of topics, and general keyword search mechanisms. However, these approaches have some drawbacks. First, there could be several interactions required from visitors, such as loading the pages that host those search services. Visitors are also responsible for locating the target page by providing related keywords, or choosing

*Research supported in part by the National Science Foundation CAREER Award IRI–9896052 and CISE Research Instrumentation Grant 9729878.

categories. Secondly, many search engines are relatively poor in filtering out the "noise". Therefore, a large list of links could be generated. Occasionally, either the keywords are not found or irrelevant results are generated.

Information filtering, whereby information is supplied to the user by means of a static query embodying the user's preferences on a set of dynamic documents [2], and in particular *collaborative filtering* have been used to recommend relevant documents to users. Collaborative filtering makes use of preferences of other people to predict the documents that may be of interest to a particular user [8, 14, 15].

As summarized by Paul Resnick at the Collaborative Filtering Workshop:[1]

> "Guiding people's choices of what to read, what to look at, what to watch, what to listen to (the filtering part); and doing that guidance based on information gathered from some other people (the collaborative part)."

In this paper, we explore a particular collaborative filtering technique as embodied by the Web Site Agent (WebSA) that can be used to aid visitors in navigating within a web site. The WebSA system provides recommendations to web users through the use of collaborative filtering. Our approach has been deployed within the web site of the Computer Science Department of the Worcester Polytechnic Institute (herein abbreviated as CSD-WPI). The approach can be easily transported to any other web site, provided that the access log for that site is made available.

To improve the quality of collaborative filtering services, one has to design an effective data mining strategy to automate information discovery from the raw data and to be able to supply information to the users in a timely fashion. To meet these requirements, we deploy database technology to manage, query, update, and concurrently access the web log data. The bulk of the computations for updating and mining the information are done on a daily basis in advance. This method together with the fine tuning of the database guarantees high performance, even when the raw access log data is in the order of 1 GB.

This paper is organized as follows. In Section 2 we mention related work and compare our approach with those more closely related to it. Section 3 describes the overall architecture of WebSA. Section 4 describes the raw data, the mining algorithm, and the relational schema used. The design of the user interface and of a preliminary usability study is discussed in Section 5. Implementation details encompassing

[1]http://www.sims.berkeley.edu/resources/collab/collab-report.html.

performance issues, tools used, and solutions to specific problems are outlined in Section 6. Finally, we summarize and present directions for future work.

2. COMPARISON WITH RELATED WORK

Software tools sometimes called "agents" are being deployed for speeding up the information retrieval process on the web in general [1, 3, 4, 6, 9, 10, 11, 16, 17] and within individual web sites in particular [7, 12, 13].

Such agents issue recommendations by modeling the user's browsing progress incorporating heuristics to model the user's behavior [9]. In some systems, the user is also asked to give feedback on the retrieved documents [3, 4] in some cases for the purpose of training a learning algorithm [1].

Other agents concentrate on particular web sites, since they require knowledge of the users' actions such as that captured in the log files for that web site [7, 12, 13]. For example, in [7] a user's interest is inferred by the actions of the user on the web pages (such as following links out of that page, mouse and scrolling activity). The approach that is closer to our own (and has provided motivation to WebSA) is by Perkowitz and Etzioni [12, 13] who analyze in detail the links that are followed by the users and propose the "Adaptive Web Sites" approach, the name being derived from web sites whose organization changes to reflect the users' preferences. They have realized that different users have distinct goals and that the original layout of pages in a web server site might hide the most important or frequently used pages in "unlikely" places, making it inconvenient for users to retrieve them. Another realization is that the web site designer's original expectation of usage may be violated. Since a site is used in many ways in practice, it is hard for the designer of a site to cover every aspect in the initial development.

To address these concerns, they propose web sites that automatically improve their organization and presentation by learning from visitor access patterns. The sites may adapt their presentation to satisfy individual users' needs by observing their individual information, which is sometimes called customization or to satisfy overall users' interactions, which is called optimization.

In their paper, they propose to improve users' information retrieval speed in a particular site by creating an index page, i.e., a page containing links to a set of related pages, which are estimated to be the users' most favorable pages. To achieve this, they present a cluster mining algorithm that takes Web server logs as input and outputs the contents of candidate index pages. The web access log is processed into visits, one day counts as one visit. For each visit, the co-occurrence frequencies between pages are computed and represented in a similarity matrix. Next, a graph is generated corresponding to the matrix. By finding cliques, or connected components in the graph, the graph is then divided into sub-graphs, where each sub-graph is considered as a cluster of related pages of a particular topic. For each cluster found, an index page is created.

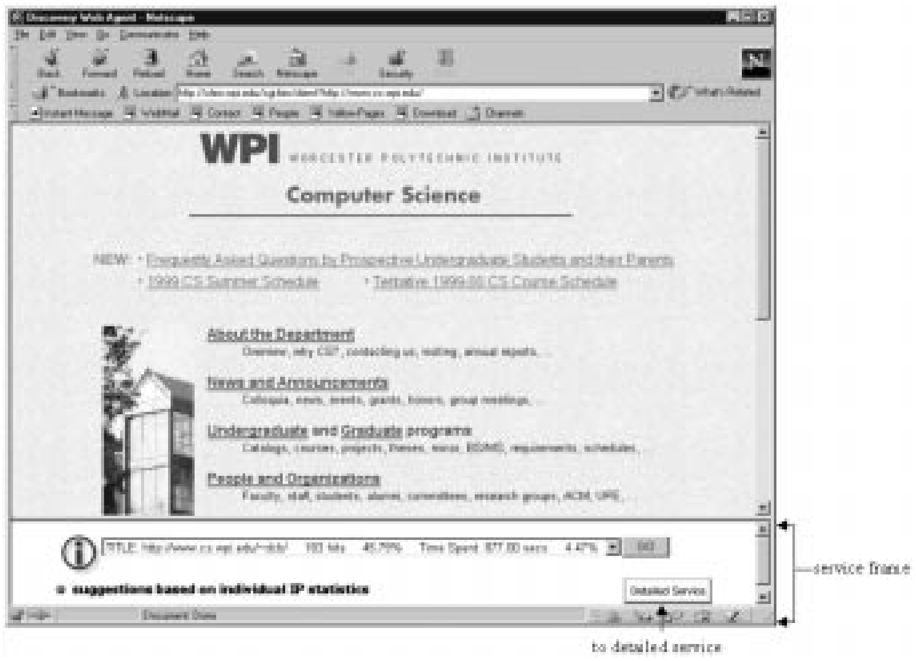
In our paper we mine the access logs in a different way by considering trees that model the users' traversals of the site. New pages are recommended ranked in decreasing order by the probability of those pages having been visited after a page currently being observed by the user was visited. Excluded from the recommended list are the links that appear on the current page. There are two types of recommendations. One of the recommendations is solely based on that particular user past pattern of access to that site. The other recommendation looks at all the users that have visited that site. In this way, new users to a site can follow the most popular traversal paths of that site, hoping to find relevant information fast. Notice that since the interface displays the name of the page, users can perform a simple "content-based filtering" on a small list of links. Returning users to the site, on the other hand, could trace back their previous steps.

Another difference from the work Perkowitz and Etzioni [12] is that we do not change the organization of the site or of each individual page. Both approaches have potential merits and drawbacks. Without extensive user studies, it is difficult to exactly evaluate the two approaches. However, it is well-known that consistency is one of the golden rules of user interface design. The reorganization of a slowly evolving web site could result in unwanted inconsistency. On the other hand, frequently updated web sites, such as those that provide news, sports events, and stock quotes might benefit from constant rearrangement based on overall users' interest.

3. WEBSA OVERVIEW

The user interface that we propose for WebSA is a simple modification of the current web pages of the CSD-WPI web site. WebSA delivers a recommendation service in a separate frame as illustrated in Figure 1.

Our "raw" data to analyze comes from the web server logs, which is basically "semi-structured" data. For the purpose of organizing the raw data in the logs and of increasing the performance of the data mining process, we use an Oracle database for managing the log data. The analysis of the visitors' past accesses can be accomplished in two basic



WPI WORCESTER POLYTECHNIC INSTITUTE

Computer Science

NEW: • Frequently Asked Questions by Prospective Undergraduate Students and their Parents
• 1999 CS Summer Schedule • Tentative 1999-00 CS Course Schedule

About the Department
Overview, why CS?, contacting us, visiting, annual reports, ...

News and Announcements
Colloquia, news, events, grants, honors, group meetings, ...

Undergraduate and Graduate programs
Catalogs, courses, projects, theses, minors, BS/MS, requirements, schedules, ...

People and Organizations
Faculty, staff, students, alumni, committees, research groups, ACM, UPE, ...

TITLE: http://www.cs.wpi.edu/~[illegible] 193 hits 45.79% Time Spent: 877.00 secs 4.47%

suggestions based on individual IP statistics

Figure 1 The WebSA interface, as provided by a separate frame at the bottom of a browser page.

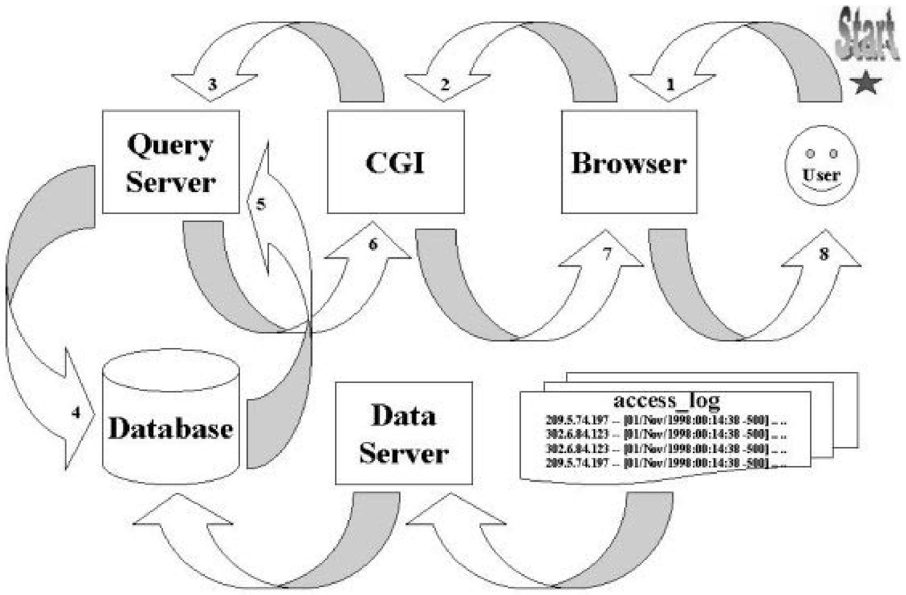
ways: customization and optimization. In the former, the analysis will be done on an individual visitor's past accesses, and its results will be presented only to that user. In the latter, past accesses for all visitors are analyzed.

Both the customization and optimization phase aim at discovering the access patterns of the visitors, and based on the discovered patterns, a ranked list of links is produced. Such lists are generated in highest to lowest recommended order. Furthermore, additional information such as the hit frequency, hit probability and time spent on each of the recommendation links are provided as references to visitors. These references help visitors locate their desired sites in less time and view their past access statistics as well.

All the heavy computation of these statistical values and recommendation links were done by a Java program in our server. Instead of processing all the computations at the time that the visits occur, WebSA discovers the access patterns in advance on a nightly basis. In this way, the list of recommendations can be directly retrieved from Oracle upon request, to save processing time. This information can be acquired from the Oracle Database by a Java server program through the Java Database Connectivity (JDBC) facility.

For the front end, there is a Common Gateway Interface (CGI) program, for each visitor using WebSA. This program gathers the necessary identification information from visitors, such as IP addresses and the URL of the visitors' current site. These identification arguments will be delivered to the Java server program for retrieving the recommendations. Furthermore, the CGI program outputs the recommended links and the statistic values to the visitor's current site.

An overview of the implemented architecture is shown in Figure 2.



```
209.5.74.197 -- [01/Nov/1998:00:14:38 -500] ....
302.6.84.123 -- [01/Nov/1998:00:14:38 -500] ....
302.6.84.123 -- [01/Nov/1998:00:14:38 -500] ....
209.5.74.197 -- [01/Nov/1998:00:14:38 -500] ....
```

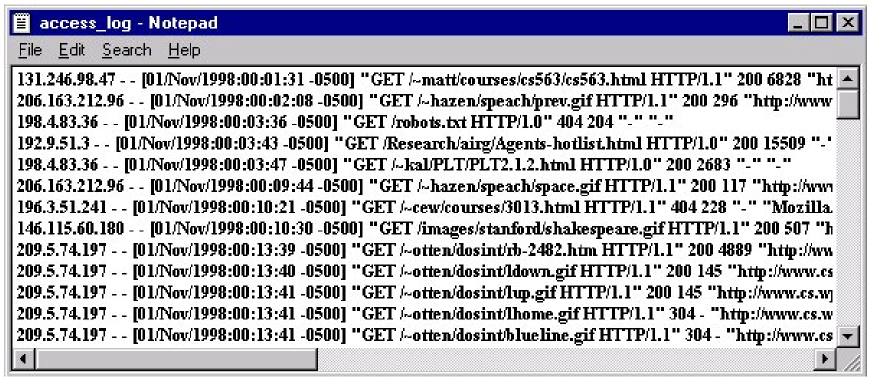
Figure 2 The WebSA Architecture

4. DATA MINING

In this section we describe the data mining process, starting from the analysis of the semi-structured log file, the extraction of the user patterns, and the database modeling of the data that is extracted and therefore made structured.

4.1. LOG FILE DATA

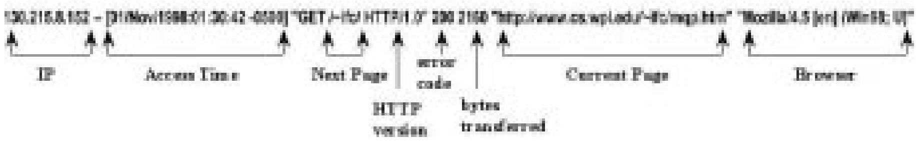
As mentioned previously, a way to provide useful recommendations is to analyze the users' past access behavior. The access patterns of all visitors to a site can be found in the web server's log file. A partial view of the CSD-WPI web server log is shown in Figure 3.



```
131.246.98.47 - - [01/Nov/1998:00:01:31 -0500] "GET /~matt/courses/cs563/cs563.html HTTP/1.1" 200 6828 "ht
206.163.212.96 - - [01/Nov/1998:00:02:08 -0500] "GET /~hazen/speach/prev.gif HTTP/1.1" 200 296 "http://www
198.4.83.36 - - [01/Nov/1998:00:03:36 -0500] "GET /robots.txt HTTP/1.0" 404 204 "-" "-"
192.9.51.3 - - [01/Nov/1998:00:03:43 -0500] "GET /Research/airg/Agents-hotlist.html HTTP/1.0" 200 15509 "-'
198.4.83.36 - - [01/Nov/1998:00:03:47 -0500] "GET /~kal/PLT/PLT2.1.2.html HTTP/1.0" 200 2683 "-" "-"
206.163.212.96 - - [01/Nov/1998:00:09:44 -0500] "GET /~hazen/speach/space.gif HTTP/1.1" 200 117 "http://ww
196.3.51.241 - - [01/Nov/1998:00:10:21 -0500] "GET /~cew/courses/3013.html HTTP/1.1" 404 228 "-" "Mozilla.
146.115.60.180 - - [01/Nov/1998:00:10:30 -0500] "GET /images/stanford/shakespeare.gif HTTP/1.1" 200 507 "h
209.5.74.197 - - [01/Nov/1998:00:13:39 -0500] "GET /~otten/dosint/rb-2482.htm HTTP/1.1" 200 4889 "http://ww
209.5.74.197 - - [01/Nov/1998:00:13:40 -0500] "GET /~otten/dosint/ldown.gif HTTP/1.1" 200 145 "http://www.cs
209.5.74.197 - - [01/Nov/1998:00:13:41 -0500] "GET /~otten/dosint/lup.gif HTTP/1.1" 200 145 "http://www.cs.w
209.5.74.197 - - [01/Nov/1998:00:13:41 -0500] "GET /~otten/dosint/lhome.gif HTTP/1.1" 304 - "http://www.cs.w
209.5.74.197 - - [01/Nov/1998:00:13:41 -0500] "GET /~otten/dosint/blueline.gif HTTP/1.1" 304 - "http://www.cs
```

Figure 3 A fragment of the access_log file of the CSD-WPI web site.

In the access log file, each line represents a transaction of transferring a file, which can be a HTML file, a picture image, a TXT format file, an executable file, or many others types of files. When a user accesses a web page at the CSD-WPI web site, one or more access lines are added to the end of the access log file, depending on how many files need to be loaded for forming that target page. Due to the nature of recording new accesses to the end of access log, the access log is already ordered sequentially by access time. Furthermore, each access line in the log is uniformly recorded. A decomposition of the fixed structure of each line is illustrated in Figure 4. There are cases for which some of the fields are unavailable: users may explicitly configure their web browser not to send out header information to the web server for privacy reasons, or sometimes instead of following links, users type in a URL to reach a page, or sometimes users reload a page. For each of the unavailable fields, its contents will be indicated with a "-" character in the access log.



```
130.215.8.152 - [31/Nov/1998:01:30:42 -0500] "GET /~lfc/ HTTP/1.0" 200 2160 "http://www.cs.wpi.edu/~lfc/mpg.htm" "Mozilla/4.5 [en] (Win98; U)"
```

Figure 4 The format of each access line in the CSD-WPI's web server log file.

By carefully studying the access log's properties, as described above, we consider only the helpful fields in each access line for our further analysis, including IP, access time, current page, next page and browser fields.

The IP field can be used to identify a user. We assume that each user will have a fixed IP address. This implies that a bias might occur in reflecting ones' individual past access patterns, if they are sharing the same proxy server, computer, or dial in connection. Note that for dial-in accesses through an Internet Service Provider, a random IP is generated each time. We are targeting users of fixed computers with broadband connections. Alternatives to our approach, with their own drawbacks, are: storing a cookie in the users browser, in which case further analysis is precluded by users refusing cookies, or asking users to sign up, an extra step (which often involves the use of cookies) that makes the recommendation service less transparent and may therefore discourage certain users.

The access time stamp field will allow us to calculate the duration a user has spent on a particular page. Since the access lines in the log file are ordered by time, we can simply compute the time difference between the entering and leaving time stamp of a page to figure out the duration of access to that page. From the current and next page fields in each access line, we can find out the hit frequency on each page in the past and their direct links to other pages. Furthermore, the hit probability of each page can be computed from the hit frequency and the paths users followed to get to that page by tracing the direct links from previously visited pages using a breadth-first search method.

4.2. ACCESS PATTERN EXTRACTION

In this project, a tree modeling method is deployed for extracting patterns in the otherwise seemingly unorganized collection of log data. For each IP address and each page, a tree is constructed to represent the possible paths from that page. An example of such a tree model is illustrated in Figure 5.

The page being currently examined by the user is the root node of the tree. Each internal node in the tree represents a page that is at most five links away from the current page.

The edges indicate direct links between two pages. For each internal node, the hit frequency, hit probability, time spent, and percentage of time spent are calculated. In the CSD-WPI web site, the average page contains more than ten links. Therefore, the tree may grow exponentially down each level. For storage space and retrieval performance concerns, only the top five levels of the tree are considered. Only pages at levels 3, 4, and 5 are considered as recommendations to the users, since pages

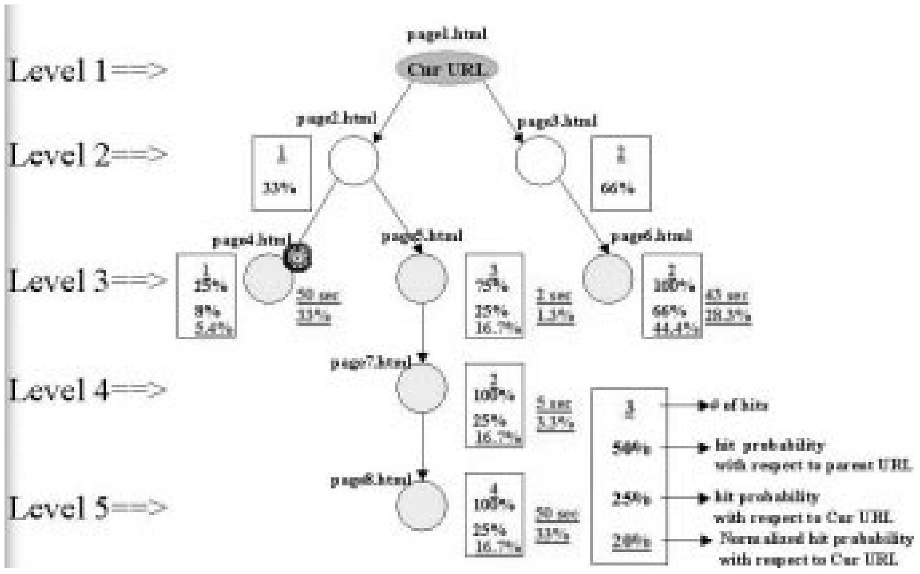


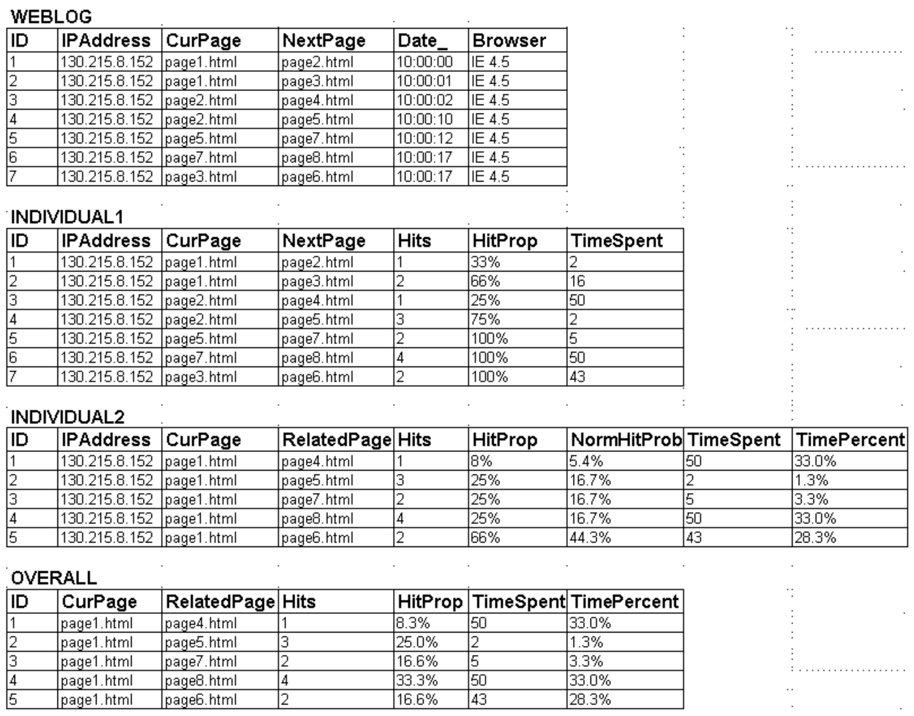
Figure 5 A tree model of all possible paths from page1.html, corresponding to the INDIVIDUAL2 table of Figure 6.

in level 2 are direct links from the current page, which are just one click away. Therefore level 2 is excluded to avoid repetitive links within the original page.

4.3. DATABASE MODELING

Data is stored into the Oracle database after the mining operation from the access log. Figure 6 shows the tables in the Oracle database, which contain all the intermediate and final recommendation data. For simplicity, only the data for a single IP address is displayed (while the actual tables hold data of all IPs). The WEBLOG table stores all the useful fields from the access log, each tuple corresponds to one access line. Tuples in the WEBLOG table are sorted by time in ascending order, like the access lines in the access log file. The difference is that the WEBLOG table contains only accesses after filtering out the "noise" that results from requesting image files, direct access from outside the CSD domain and the CGI pages.

The INDIVIDUAL1 table combines all tuples with same IP, CurPage and NextPage in the WEBLOG table. The number of the occurrences of each of these tuples is stored in the Hits attribute. The HitProb attribute, which stands for hit probability, is computed by dividing the hit number of the CurPage to the NextPage by the total hit number from



WEBLOG

ID	IPAddress	CurPage	NextPage	Date_	Browser
1	130.215.8.152	page1.html	page2.html	10:00:00	IE 4.5
2	130.215.8.152	page1.html	page3.html	10:00:01	IE 4.5
3	130.215.8.152	page2.html	page4.html	10:00:02	IE 4.5
4	130.215.8.152	page2.html	page5.html	10:00:10	IE 4.5
5	130.215.8.152	page5.html	page7.html	10:00:12	IE 4.5
6	130.215.8.152	page7.html	page8.html	10:00:17	IE 4.5
7	130.215.8.152	page3.html	page6.html	10:00:17	IE 4.5

INDIVIDUAL1

ID	IPAddress	CurPage	NextPage	Hits	HitProp	TimeSpent
1	130.215.8.152	page1.html	page2.html	1	33%	2
2	130.215.8.152	page1.html	page3.html	2	66%	16
3	130.215.8.152	page2.html	page4.html	1	25%	50
4	130.215.8.152	page2.html	page5.html	3	75%	2
5	130.215.8.152	page5.html	page7.html	2	100%	5
6	130.215.8.152	page7.html	page8.html	4	100%	50
7	130.215.8.152	page3.html	page6.html	2	100%	43

INDIVIDUAL2

ID	IPAddress	CurPage	RelatedPage	Hits	HitProp	NormHitProb	TimeSpent	TimePercent
1	130.215.8.152	page1.html	page4.html	1	8%	5.4%	50	33.0%
2	130.215.8.152	page1.html	page5.html	3	25%	16.7%	2	1.3%
3	130.215.8.152	page1.html	page7.html	2	25%	16.7%	5	3.3%
4	130.215.8.152	page1.html	page8.html	4	25%	16.7%	50	33.0%
5	130.215.8.152	page1.html	page6.html	2	66%	44.3%	43	28.3%

OVERALL

ID	CurPage	RelatedPage	Hits	HitProp	TimeSpent	TimePercent
1	page1.html	page4.html	1	8.3%	50	33.0%
2	page1.html	page5.html	3	25.0%	2	1.3%
3	page1.html	page7.html	2	16.6%	5	3.3%
4	page1.html	page8.html	4	33.3%	50	33.0%
5	page1.html	page6.html	2	16.6%	43	28.3%

Figure 6 Database tables of processed access logs, which contain all the intermediate and final recommendation data.

CurPage to all other direct next pages. The values for the TimeSpent attribute are calculated by evaluating the time difference between time stamps for entering and leaving the NextPage in the WEBLOG table. If more than one access to that page is found, the all access times are combined. We have chosen to give a greater weight to the most recent access. Older access patterns will, in this way, be phased out more quickly.

Both the WEBLOG and INDIVIDUAL1 tables are used to store intermediate values for computing the final results in INDIVIDUAL2 and OVERALL tables. INDIVIDUAL2 table contains the recommendation trees for each page an individual user has visited. An illustration of the tree model is shown in Figure 5. The contents in the RelatedPage attribute are the candidates of recommendation pages, as in the levels 3, 4 and 5 in the tree model.

In the INDIVIDUAL2 table, the Hits attribute can be directly obtained from INDIVIDUAL1, where the HitProb attribute is calculated by multiplying each node's HitProb value with its parent nodes' Hit-

Prob value from INDIVIDUAL1. This yields the hit probability on each internal node with respect to the current page, or root page of the tree. This algorithm is intended to avoid the possible bias produced by overlapping trees, in which case the hit number is higher but does not reflect the users' past choices starting from the current page. To make the sum of the hit probabilities from all candidate pages be 100%, the HitProb column is normalized. The TimeSpent column is fetched directly from the INDIVIDUAL1 table by querying for the same IP and NextPage. The TimePercent, which stands for the percentage of time spent on a page, is calculated by dividing each TimeSpent value by the total TimeSpent of all the candidate pages in the tree in INDIVIDUAL2.

The OVERALL table combines all IPs' trees with the same current/root page. The Hits column is computed by summing all the Hits with the same CurPage and RelatedPage in INDIVIDUAL2. In the OVERALL table, the hit probability, the HitProb attribute, is based on the hit number instead of the percentage approach for INDIVIDUAL2's HitProb column. For the OVERALL table, the HitProb column is calculated by dividing each node's hit number by the total hit number of all candidate nodes in the tree. The TimeSpent for each page can be directly fetched from the INDIVIDUAL2 table, where as the time spent percentage can be obtained by dividing each node's TimeSpent by the total time spent of all candidate nodes in the tree.

5. USER INTERFACE AND PRELIMINARY USABILITY STUDY

The WebSA's user interface (see Figure 1) is provided by a CGI application. It takes care of the overall layout, page design and the interaction between the user and WebSA.

The first interface design rationale considered was to avoid layout modifications of the original web page design. Therefore, the display of the original page is not modified. The services of WebSA are provided in a separated frame of the browser window, which will keep the original structure and design intact, yet the services will always be presented to the users in the bottom frame.

The second rationale of interface design is to minimize the space of the recommendation service in the browser, as browser space is limited. Thus, instead of itemizing and listing all the recommendation, the CGI application of WebSA provides the recommendations in a pull-down menu.

The third rationale of our interface design is to attract the user's attention to the services provided by WebSA. This can be accomplished with simple means, such as, background color choice, color scheme of the service page and also the icons and logo design.

The last rationale and also the golden rule of interface design is the consistency of the interface. The frame layout is set by WebSA, where the height of the bottom frame is fixed and occupies the least possible space. The design of both service pages adopts the least variation, which will increase the users' familiarity to the user interface of WebSA.

Figure 7 shows the WebSA frame for the detailed service where: (1) both individual- and overall-based recommendations are provided; (2) for each of them, ten ranked links are given instead of the five links for the default service shown in Figure 1.

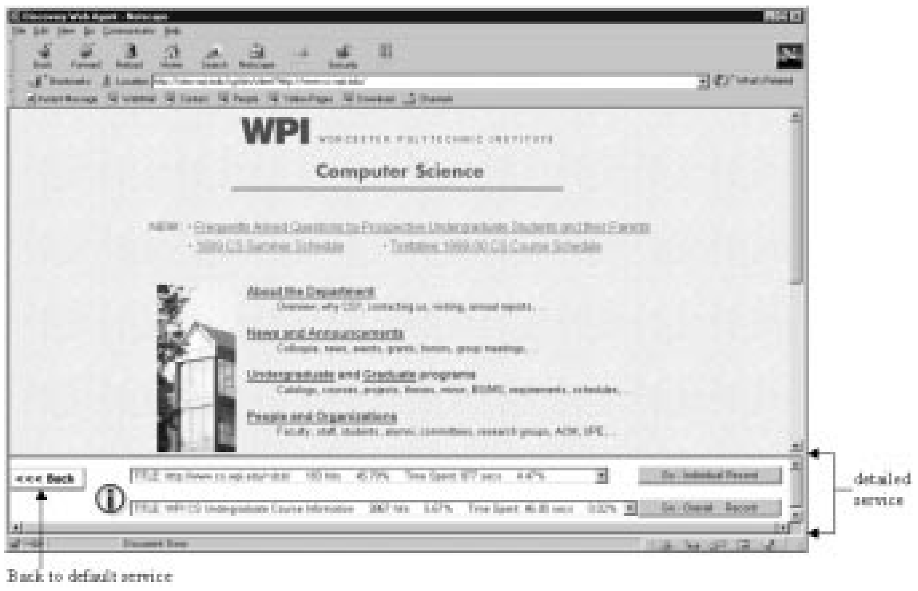


Figure 7 The WebSA detailed interface, as provided by a separate frame at the bottom of a browser page.

We have conducted a preliminary user survey where ten users from CSD-WPI participated. 80% of the people surveyed were using fixed IP addresses accessing the Internet. Therefore most people could take full advantage of the personal recommendation service (based on their IP address). All of the people surveyed were accessing the CSD-WPI web pages as least 0.5-hour everyday so they were in a good position to compare the effectiveness of the service with regular browsing without the service. From all the data we collected, 69% of the users agree that WebSA is helpful or somewhat helpful for browsing the CSD-WPI web sites, 78% feel the interface design of WebSA is intuitive, and 80% are satisfied with the performance of WebSA.

6. IMPLEMENTATION

6.1. PERFORMANCE ISSUES

Performance was a major concern in designing WebSA. If users had to wait significantly for the services, they will probably prefer not to use them. Another objective was that the backend computations for updating the service data should not take too long as not to unduly overload our Oracle server or local network. To eliminate the delay in delivering services to the users, we choose to pre-calculate service data on a daily basis. In this way, no major computations are performed while recommendations are provided to the users. To offer the recommendations, the CGI application simply queries the pre-computed results from the database and then displays them to users. The query process was made efficient, since appropriate indices are used that match the queries.

The background process for preparing the service needs to be optimized too. The web log's size is increasing at the rate of 2.6 megabytes per day. After a couple of months, the log file would be in the magnitude of hundreds of megabytes. Data mining for useful recommendations requires the repetitive retrieval of certain patterns from the large log file. These operations can be very expensive, since loading the log file will take up most of the memory in the operating system and will yield tremendous disk access, which is 1000 times slower than memory access on average. We used an Oracle database to store the web log data. Furthermore, we conveniently used the built-in utilities provided by the database, including indices for faster data retrieval and the concurrent retrieval synchronization feature.

When doing database computations using Java, some special techniques were deployed. These techniques includes using prepared statements, sending a set of SQL commands in one transition to avoid overhead costs, and dropping indices before heavy modification on database tables and adding indices after finishing updating database tables for optimal retrieval services.

The detailed architecture of WebSA is shown in Figure 8.

6.2. SOFTWARE TOOLS

For the database implementation we used an Oracle database server installed on an NT server. Configuration settings were adjusted in order to increase the performance, e.g., cache memory size, rollback buffer,

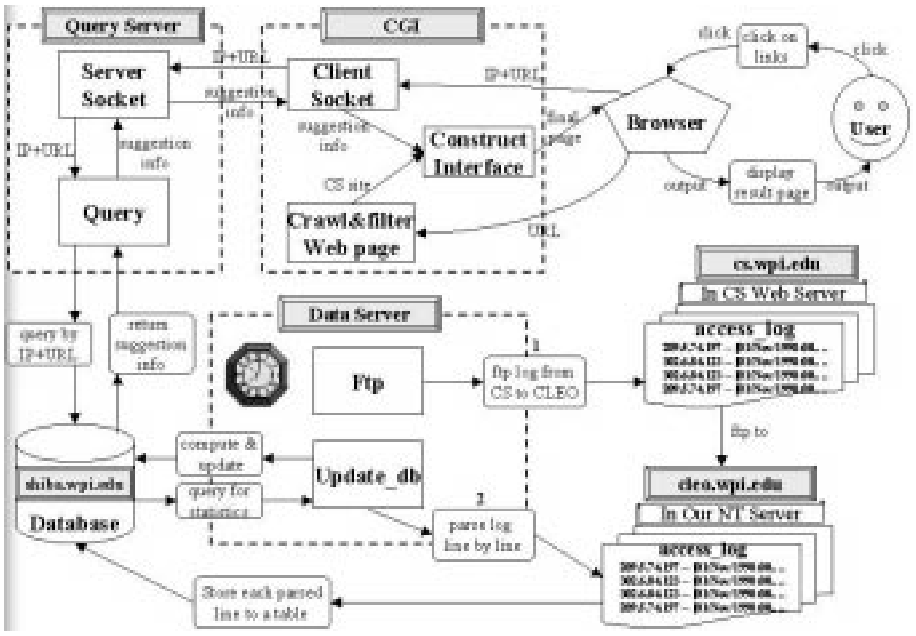


Figure 8 Functional model of WebSA

etc. For the purposes of monitoring and administering the database, we also installed an Oracle 8 SQL client application on the development machine.

The major development languages in our project are Java and Visual C++. Java is a very handy development language because it provides the developer with rich built-in libraries. Visual C++ was also used because it provides a very easy handle to sockets in the WIN32 environment. The communications to and from the Oracle database server are relying on the connection built with JDBC. We used the Xitami web server, which is lightweight and supports CGI.

6.3. PROBLEMS AND SOLUTIONS

In this section, we discuss some of the problems that arose during the implementation and the chosen solutions.

The connection between the Query Server and CGI application is a TCP socket. However, these two applications are implemented using different development languages. Due to the easy connection to the Oracle database with JDBC, the Query Server is written in Java, while the CGI application is written in C++ as we consider C++ a better CGI programming language than Java. The problem arises when the sockets are created using different programming languages. The Java server socket and the WIN32 socket created by C++ do not work properly together. There were always some data-transfer errors in the communication. In order to guarantee the correctness of data transfer between these two sockets, some extra user defined control signals are added into the communication between client and server sockets. There is a parser sitting on both sides to check for the messages received, that request recovery if the corruption of the message is detected. And the start and end transfer signals are added to the message exchanges. The user-defined controls guarantee the socket communication on the TCP connection without noticing the difference of the internal mechanism. In retrospect (and for a future implementation), accessing the database directly with a CGI application would have been simpler.

Another problem arose when the interface of WebSA was expanded to take arguments from different HTML formats. The arguments passed to the CGI application were input from direct argument parsing, i.e., filtered links, and form submission, i.e., recommendations from the pull-down menu of WebSA. This gave the CGI application a certain degree of difficulty to cope with the arguments. The direct argument parsing and form submission should be dealt differently, otherwise it will crash the CGI application. The solution for this problem is to set an extra argument checking in the CGI application. Once the CGI application performs the checking, there is a direct argument, which will be taken as the user request. If the direct argument is absent, the CGI application will then obtain the argument from the form submission through CGI environment variables.

An interesting problem arises when users access the CSD-WPI web site through WebSA. In this case, the information of the IP address of the user's local machine was lost. Also, due to the missing HTTP header information, which usually will not happen with browsers like Internet Explorer and Netscape Navigator, the user's current page is not recorded into the CS access_log file. Examining the lines recorded into the access_log file, we can clearly identify the information loss when using WebSA:

```
130.215.8.113 - - [17/Mar/1999:16:03:42 -0500] "GET / HTTP/1.0"
200 5653 "-" "-"
```

The IP address field is always recorded as the IP address of our host (`http://cleo.wpi.edu`), which is the NT server machine of WebSA. The field of the currently browsed page and the browser information are therefore missed. This would, in the long term, affect the analysis performed by WebSA, especially if it became heavily used. The solution we used to resolve this problem was to submit additional HTTP header information while requesting the document from the web server. The additional HTTP header information is included in the current page and the browser information. We also noticed that the CGI application has no way to keep track of the current web page's URL. Therefore a new tag, <REFERRER>, is introduced for the specific use of WebSA. This tag is used to record the browsing page's URL, for the reference of the CGI application. This tag avoids setting cookies on the user's local machine. After performing such changes, the corresponding line in the access_log of the above example is changed to

```
130.215.8.113 - - [17/Mar/1999:16:03:42 -0500] "GET / HTTP/1.0"
200 5653 "http://www.cs.wpi.edu/" "Mozilla/4.5 [en] (WinNT; I)
&&130.215.8.113"
```

The Data Server of WebSA was also adjusted in order to work with the extra piece of information appended at the end of the browser field of the access_log file.

7. CONCLUSIONS

The goal of the Web Site Agent (WebSA) is to provide a recommendation service to help web users better browse sites according to the analysis of the data available from the access log file. The analysis takes into account the access probability for the pages of the site, with the page currently under examination as a starting point. The WebSA is dynamic in that it incorporates pattern changes on a daily basis.

The WebSA interface, differs from the Adaptive Web Site interface [12, 13] in that the layout of the web site pages does not change according to the recommendations. From a human-computer interaction perspective, where consistency of the interface is one of the principal directives, our approach seems more adequate, although studies would have to be conducted to confirm this, otherwise, "universal principle" in the context at hand.

Providing WebSA with database support is also a contribution of our work. In this way, recommendations are answers to declarative queries that could be changed or extended easily. Moreover, database technology provides us with fast performance and concurrent access to the data, which can deliver fast performance even when the accumulated raw data is in the order of 1GB.

8. FUTURE WORK

After collecting suggestions from our surveys and from colleagues, extensions to the current work have become apparent, which can be in some cases directly implemented given the data already collected.

Furthermore, the user interface can be extended to be more flexible and graphical. We group the directions for future work as follows:

1 Some people's access patterns might be totally different from others. Therefore, one could attempt to categorize users into communities by examining the similarity between users' access tree and provide recommendations based on the community the user "falls" in. The similarity of access trees can be evaluated, for example, by observing the number of nodes that are common to different users' access trees.

2 Instead of providing the recommended links in a flat listing, one could use a Java applet to produce a graphical representation of the access tree. The graphical access tree can be expanded further levels down by clicking on it, similarly to the hierarchy for the folders in the Windows Explorer application (which is available in Swing). Different user interface styles could be customizable to the specifications of the user [5].

3 WebSA does not support multi-framed pages. Multi-framed pages are currently not common in the CSD-WPI web site, but they might become common in the near future. One can extend our agent's ability in this direction.

4 Another possible direction for future work is to deploy WebSA in a larger, more complex, and less organized web site. This would be beneficial for further testing its robustness and usefulness.

Acknowledgments We are thankful to Andreas Koeller for his assistance with the Oracle server and to Daniel Fox for initial discussions on the prototype implementation.

References

[1] R. Armstrong, D. Freitag, T. Joachims, and T. Mitchell. Web-Watcher: A Learning Apprentice for the World Wide Web. In *Working Notes of the AAAI Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments*, pages 6–12, Stanford University, 1995. AAAI Press.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley Publishing Company, Reading, Mass., 1999.

[3] M. Balabanovic. An Adaptive Wep Page Recommendation Service. In *Proc. of the First International Conference on Autonomous Agents.* AAAI Press, 1997.

[4] M. Balabanovic and Y. Shoham. Learning Information Retrieval Agents: Experiments with Automated Web Browsing. In *Working Notes of the AAAI Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments*, Stanford University, 1995. AAAI Press.

[5] I. F. Cruz and G. T. McGuire. Publication and Customization of Electronic Documents Using PANDA. In *Proc. ACM SIGDOC Conference*, pages 58–64, 1999.

[6] O. Etzioni. Moving Up the Information Food Chain: Deploying Softbots on the World Wide Web. In *Proc. of AAAI '96*, 1996.

[7] J. Goecks and J. Shavlik. Automatically Labeling Web Pages Based on Normal User Actions. In *Proc. IJCAI Workshop on Machine Learning for Information Filtering*, 1999.

[8] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM*, 35(12):61–70, December 1992.

[9] H. Lieberman. Letizia: An Agent that Assists Web Browsing. In *Proc. IJCAI '95*, pages 924–929, 1995.

[10] S. Luke and J. Hendler. Web Agents that Work. In *Proc. IEEE Multimedia 97*, pages 76–80, 1997.

[11] S. Luke, L. Spector, D. Rager, and J. Hendler. Ontology-based Web Agents. In *Proc. of the First International Conference on Autonomous Agents.* AAAI Press, 1997.

[12] M. Perkowitz and O. Etzioni. Adaptive Web sites: an AI Challenge. In *Proc. IJCAI '97*, 1997.

[13] M. Perkowitz and O. Etzioni. Adaptive Web Sites: Automatically Learning from User Access Patterns. In *Proc. of the Sixth International WWW Conference*, Santa Clara, CA, 1997.

[14] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proc. of the Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.

[15] P. Resnick and H. R. Varian. Recommender Systems. *Communications of the ACM*, 40(3):56–58, March 1997.

[16] J. Rucker and M. J. Polanco. Siteseer: Personalized Navigation for the Web. *Communications of the ACM*, 40(3):73–75, March 1997.

[17] A. Wexelblat and P. Maes. Footprints: History-Rich Web Browsing. In *Proc. Conf. Computer-Assisted Information Retrieval (RIAO)*, pages 75–84, 1997.

Biographies

Isabel F. Cruz is a faculty member in the Computer Science Department of the Worcester Polytechnic Institute in Massachusetts, where she heads the Information Systems Research Group (ADVIS). She received her Ph.D. in Computer Science from the University of Toronto in 1994 and was a postdoctoral fellow in the Department of Computer Science at Brown University.

She has been invited to give more than 30 talks worldwide and has published more than 40 research articles in Databases, Visual Languages, Graph Drawing, Multimedia, and Information Retrieval. Her editorial activities include being the founding editor of the ACM SIGMOD Digital Symposium Collection since 1998 and an associate editor of the Journal of Visual Languages and Computing (Academic Press) since 1995. She serves regularly on the program committees of the main conferences in her field including ACM SIGMOD, ACM Multimedia (once as associate program chair), VLDB, IEEE Visual Languages, and the IEEE International Conference on Data Engineering. She has also organized and co-chaired several international conferences and workshops.

She has received numerous awards including a CAREER Award from the National Science Foundation and a Government of Canada Award. In addition, her research has been funded by grants from the National Science Foundation, DARPA, NATO, and the CRA.

Lijun Leo Liu is a design engineer in the Corporate Internet Technology Department at the EMC Corporation in Massachusetts. He received his B.S. in Computer Science from the Worcester Polytechnic Institute in 1999 and is the recipient of a Computer Science Outstanding Senior Award. He worked on this paper while he was a member of the ADVIS Research Group at the Worcester Polytechnic Institute. His research was partially supported by an REU Award from the National Science Foundation.

Tony Y. Wu is a technical staff member in the Communication Software Management System Department at Lucent Technologies in New Jersey. He received his B.S. in Computer Science with High Distinction from the Worcester Polytechnic Institute in 1999. He worked on this paper while he was a member of the ADVIS Research Group at the Worcester Polytechnic Institute. His research was partially supported by an REU Award from the National Science Foundation.