Discovering User Access Patterns on the World-Wide Web

David W. Cheung, Ben Kao, Joseph Lee
{dcheung | kao | jkwlee}@cs.hku.hk
Department of Computer Science, The University of Hong Kong,
Pokfulam Road, Hong Kong

The World-Wide Web provides its users almost unlimited accesses to the documents on the Internet. We suggest to use intelligent agents to assist users to locate documents related to their interests instead of browsing the Web via primitive search engines. We identify a number of key components in such intelligent systems and propose a system architecture. In particular, we have designed a learning agent and the underlying algorithms for the discovery of areas of interest from the user access logs. The discovered topics can be used to improve the efficiency of information retrieval by prefetching documents for the users and store them in a document database in the system. We have also implemented a prototype system to illustrate the various concepts. Experiments are performed which shows that the areas of interest discovered can in fact be used to improve the efficiency of information retrieval on a distributed information system such as the Internet.

1 Introduction

It has been reported that the Web contains more than 30 million Web pages located on more than 275,600 hosts. While new information are being pumped into the Web everyday, large numbers of the old articles are being updated regularly, some at a very high frequency. It is impossible for a human being to keep track of all this information and changes, and there have been proposals on software tools to help users *retrieve, locate*, and *manage* Web documents. 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 such as Mosaic and Netscape *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.

A level 1 Web tool provides a *user-initiated searching facility* for finding relevant web pages. Internet search engines, such as Alta Vista[1] are examples of level 1 tools. Most of the search engines work by traversing the WWW indexing a large population of the pages. To find out relevant pages on a particular topic, a user supplies keywords to a search engine which describe the concepts. Documents that match the user query will be ranked according to an estimation on the closeness of the documents to the supplied keywords. Information about the matching documents will be sent back to the user.

A level 2 Web tool 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[10]. As an example, SIFT automatically matches new Netnews articles with users' 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.

A level 3 Web tool should have *a learning and deductive component of user profiles.* Example contents of a user profile include topics of interest and browsing patterns. DiffAgent[7] and Letizia[8] are two experimental systems that track a users browsing behavior to infer the users 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. 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.

In this paper, we focus on the techniques in designing a level 4 Web tool. A system design and its architecture is proposed. In particular, we study the problem of constructing user profiles from user access history.

2 Related Works

A number of intelligent Web tools with different functionality have been proposed and implemented. In this section we briefly mention three such systems. Interested readers are referred to the literatures[2,3,5,8] for more details.

WebWatcher. Developed at Carnegie Mellon University, WebWatcher[2] is a tool that assists users by interactively giving them navigation advises. While the user is browsing through Web pages, WebWatcher assesses the hypertext links contained in the pages. It then recommends those links that the system guesses is promising in matching the goal of the session.

Letizia. Letizia[8] is an intelligent agent that works with a conventional Web browser such as Netscape. It tracks the users browsing behavior and tries to infer the users 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.

Besides intelligent systems, another area in which some interesting works have been done is the discovery of useful access patterns from user access logs in distributed information systems like the World-Wide-Web.

A study by Chen et.al.[4] suggests to capture the browsing movement, (forwards and backward), between Web pages in a directed graph called *traversal path graph*. A set of *maximal forward references* which represent different browsing sessions are first extracted from the directed graphs. By using a technique similar to association rule mining in transactional databases, the frequently traversed paths, called *frequent traversal paths*, can be discovered. These frequent paths are the most common traversal patterns of the users.

Another study by Yan et.al.[11] suggests another approach for automatically classifying users of a Web site according to their access patterns. User access logs are examined to discover clusters of users that access similar pages. This technique may discover categories of Web pages that may be useful in application such as designing online catalogues for electronic commerce. Once these page groups are discovered, they may be used to customize the hypertext organization dynamically. The idea is to try and match an active users access pattern with one or more of the categories discovered. Pages in the matched categories that have not been explored by the user and are not adjacent to the users position may serve as a navigational hints.

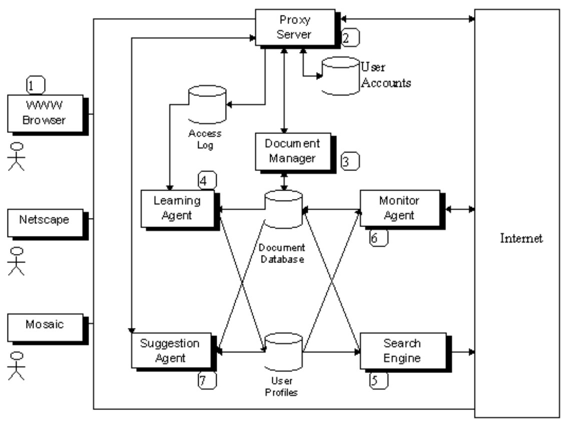
3 An Intelligent Web Tools and Architecture

In this section, we outline the most important requirements of a level 4 intelligent Web tools, and propose an architecture for such a system. 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. The requirements are:

1. The system should be able to discover its users topics of interest automatically. The system should also learn about its users shift of interests over time. This knowledge of users interests is used when the system is navigating the Internet on behalf of the user to discover relevant information.

2. The system should learn its users access patterns and information sources update patterns. It should also learn about when the documents are updated at the sources, and retrieve in advance the latest version before the user requests them.

Figure 1: System architecture



3. The system should make efficient use of network resources. Exhaustive searches that can generate excessive traffics should be avioded. Searching goals of multiple users should be clusterized. Similar goals should be batched together to reduce the number of Web searches conducted.

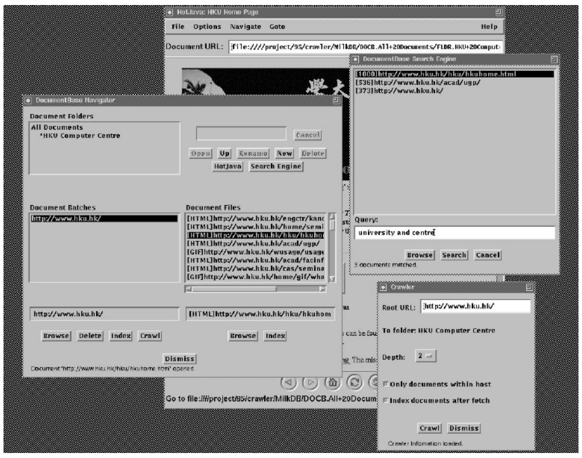
4. The system should maintain a database and a full-text index on retrieved documents. Data mining techniques [4] can be applied to the saved documents to discover the various types of useful and interesting access patterns.

5. 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.

Here we present an overview of a system design of a level 4 intelligent Web tool which is the model used in our prototype system. Figure 1 shows the system architecture and its 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. During a session, all HTTP requests go through the Proxy. This mechanism allows the system to maintain a user access log.

Proxy (2). Users communicate with the system via a WWW proxy server. When a user issues a HTTP request, the request is forwarded to the Proxy,



Document URL: file:////project/95/crawler/MilkDB/DOCB.All+20Documents/FLDR.HKU+20Comput

Document Folders
All Documents
*HKU Computer Centre

Document Batches
http://www.hku.hk/

Document Files
[HTML]http://www.hku.hk/engctr/kanc
[HTML]http://www.hku.hk/home/semi
[HTML]http://www.hku.hk/hku/hkuho
[HTML]http://www.hku.hk/acad/ugp/
[GIF]http://www.hku.hk/wusage/usage
[HTML]http://www.hku.hk/acad/facinf
[HTML]http://www.hku.hk/cas/semina
[GIF]http://www.hku.hk/home/gif/wha

http://www.hku.hk/
[HTML]http://www.hku.hk/hku/hkuhom

Document "http://www.hku.hk/hku/hkuhome.html" opened.

Go to file:///project/95/crawler/MilkDB/DOCB.All+20Docum

[1000]http://www.hku.hk/hku/hkuhome.html
[536]http://www.hku.hk/acad/ugp/
[373]http://www.hku.hk/

Query: university and centre

3 documents matched.

Root URL: http://www.hku.hk/

To folder: HKU Computer Centre

Depth: 2

Only documents within host

Index documents after fetch

Crawler Information loaded.

Figure 2: The graphical user interface of the Document Manager.

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). 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 logging of user information be performed. In the prototype each user request generates a log record, which consists of the users ID, the URL requested, the time of the request, and the document retrieved. Information kept in the log is used by the Learning Agent to reconstruct the access patterns of the users. We will study how the Learning Agent analyzes this information to deduce the kinds of information sought by the users in Section 4.

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. Besides providing a persistent storage for the documents, the database also maintains a full-text index on the documents. Figure 2 shows the graphical user interface of the Document Manager.

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

1. Topics of interest. A topic is a set of weighted keywords and phrases that is of interest to the user. For example, the topic: <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 can be used to drive the Search Engine for information discovery.

2. 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 prefetching documents.

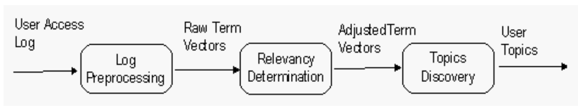
We will discuss in details the algorithms used in generating the user profiles in Section 4.

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 profiles, in particular, the topics of interest, 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.

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 a page is updated, and when). With this knowledge, the Monitor Agent schedules future retrieval of the pages and keeps them fresh in the Document Database. The second function is to schedule prefectches of pages that are regularly accessed by the users. The agent learns this from the user profiles created by the Learning agent.

Suggestion Agent (7). The Suggestion Agent ranks the newly found documents and assigns scores according to the extent that they match the users profiles. It composes a list of relevant new pages found on users' requests. The Agent also learns from the users feedback to improve the quality of future recommendation.

Figure 3: Discovery of topics of interest from user access log.



4 Discovery of Access Patterns

The Learning Agent and the Monitor Agent are two important components in our intelligent Web tool. On the client side, the Learning Agent is capable of identifying access patterns from the user access logs, in particular, the topics of interest of the users, while on the server side, it can discover the update patterns of the web pages collected. The Monitor Agent serves the clients by monitoring adaptively the updates on the web pages that the users are interested in.

The documents refereed in the user access logs contain a lot of unpolished information. However, with respect to the discovery of topics of interest, the information are mostly of the lower levels and may not be useful. In order to discover the most interesting topics, we need to identify and extract the most relevant kewords. In this section, we discuss the approach of keyword extraction and describe the algorithms implemented in our prototype system for this discovery task.

4.1 Discovery of Topics of Interests

The mechanism for discovering topics of interest is a 3-phase process (see Figure 3). The input to the process is the user access log. Initially, assume the log records the access history of a single user. We will extend it to cover the multi-users case later. We describe the three phases in detail in the following.

Phase 1. The Learning Agent processes each textual document as recorded in the user access log and produce a term vector of (keyword, weight) pairs. For example, a term vector including the pairs (NBA, 50) and (basketball, 35) may be extracted from a sport document. A modified formula for TFIDF[9] can be used to compute the weight of the keywords.

Phase 2. The term vectors produced in phase-one could be used to discover the users topics directly. However, there is a significant amount of noise in the vectors. For example, some Web pages may not be supplying any information to the user, but are visited simply because they have a large number of reference hyperlinks. Therefore, the learning agent has to determine the relevancy of every document and subsequently adjust the weights of the keywords according to the relevancy of the document from which they are extracted. The following heuristics are very useful for this purpose.

1. A Web page which has a large number of URLs is very likely a directory page of reference links. A document in which the number of hyperlinks exceeds a threshold will be regarded as a reference page and its relevancy is adjusted downward. The weights of the keywords extracted are thus lowered.

2. The amount of time that a user has spent in a document can be estimated from the records in the access logs. A page which is visited for a very short time is very likely a reference page or a pass-by page for browsing purpose. A time threshold is used to identify these pages and their relevancy are adjusted downward to reduce their significance.

3. The documents accessed at the end of a session are mostly visited because of their contents. In order to identify these content-rich documents, a *browse movement graph* which captures the forward and backward browsing relationship between Web pages is constructed from the access log. This graph is similar to the traversal path graph described by Chen[4]. Here, we use it for a different purpose. Every document in the access log is a node in the browse movement graph and an edge is created on the graph if the user has traversed from one document to another. In the browse movement graph, traversal paths can be identified. A traversal path starts from the home of the user and ends when a backward movement is recorded. (A backward movement is a go-back operation in a browser such as Netscape.) All traversal paths can be identified by determining all the backward browsing movement in the access log. A traversal path can be regarded as a browse session and the documents near the end of a session have a high chance of being a content-rich document. To refine this strategy, the time that a user has spent in such a document is also considered. To become a content-rich document, it must be near the end of a traversal path and the user must have stayed on it for a significant amount of time. A document identified as content-rich will have its relevancy adjusted upward to increase its significance.

4. The nodes in a browse movement graph described in point 3 above may have different fan-out. A node that has a high fan-out number represents a document from which many browsing-direction-changes are originated. Therefore, its functionality in the browsing is more like a reference page than a content provider. Hence, its relevancy should be adjusted downward and the weight of the keywords from it should be reduced.

5. Keywords extracted from a document with high relevancy are adjusted further according to their functional role in a document. For example, those in a title will have their weights increased the most, following by those extracted from higher level headers. In a HTML document, this adjustment can be performed by searching for the HTML tags such as the title tag <TITLE> and the top level header tag <H1>.

6. URLs that lead to documents which have high relevancy may contain keywords that reflect the users browsinginterest. Thus the keywords found in these URLs should be adjusted with increased weights to reflect their importance.

In summary, the task of the second phase in the discovery process is to use the above heuristics to adjust the relevancy of the raw term vectors extracted in the first phase. The output of the second phase is a sequence of term vectors with their weights adjusted.

Phase 3. The last phase of the discovery process is to produce the topics of interests from adjusted term vectors. Clustering techniques are being used to form the topics. The output is a small number of *topic vectors* truncated to a small predefined length, such as (NBA, basketball, stadium, arena). Note that this topic vector represents an area of interest of a user. In fact, the keywords in this topic vector can be input into a goal-directed search engine to retrieve documents that have a high chance of matching the users browsing habit.

In our prototype, the *reallocation method*[6] is applied on the set of adjusted term vectors in order to generate clusters of similar vectors in the second phase. The distance between any two term vectors is measured by their similarity. The higher the similarity is, the smaller the distance would be. The similarity $s(v_1, v_2)$ between two term vectors v_1 and v_2 is given by the normalized inner product of v_1 and v_2. When a new term vector is added to a pool of clusterized vectors, its distance from the centroid of all the clusters formed so far will be measured. The new vector will be absorbed by the closest cluster unless their distance is longer than a certain threshold, in which case the new vectors form a new cluster by itself. The centroid of a cluster is a term vector which is the mean of all the vectors in the cluster. If too many clusters are formed with a threshold, clustering will be repeated with a smaller similarity threshold, (a larger distance threshold), until the number of clusters formed is small enough.

The last step of this phase is to convert the clusters into topics. Since the term vectors in a cluster are very close to the centroid, it is reasonable to use the centroid to represent all the term vectors in deriving the topic from a cluster. However, a centroid in general may have too many keywords and many of them may carry relatively small weights. A further selection of keywords

Figure 4: A sample document. Underlined words are those appear in the term vector.

```
<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 community.</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 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/ovie1.1.html">Summary of features</A></LI></UL>
</BODY></HTML>
```

within a centroid is performed. 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 (CFL, football, Tigers) and (NHL, hockey, Canada, Oilers).

4.2 An Example of Discovering Topics of Interest

In this section, an example is presented to illustrate the above algorithm. Figure 4 is a document recorded in an access log. In the first phase, raw term vectors from this document are extracted. Once the keywords are identified, their weights are calculated by using the TFIDF method. In the second phase, the heuristics proposed are used to adjust the weight of the term vectors. For example, the HTML tag are used to modify the weight of the keywords. Those in the title with the tag <TITLE> will have their weights increased, while those in the header under the tag <H1> will receive a smaller increment, all other keywords in different sections of the document are modified in a similar fashion. Also, this document is identified in the browse movement graph as a content-rich document, and the keywords weight are further increased. The resulted term vector of this document is ((shell, 0.2994), (bash, 0.2425), (bourne, 0.2379), ...).

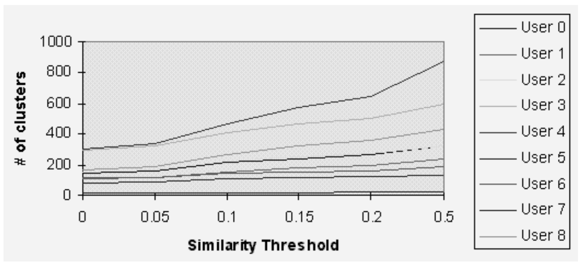
In the last phase, the term vectors with adjusted weights are input into the clusterization process for the discovery of areas of interest. The similarity threshold used in this example is 0.1. The following URLs represent a cluster generated

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

Table 1: Number of term vectors from each users

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

Figure 5: Number of clusters vs similarity thresholds.



The centroid of the above cluster calculated from its term vectors is the vector ((bash,0.2881), (shell,0.2792), (bourne,0.2312), (unix,0.1138), (csh,0.1024), (z-shell,0.08673), (tcsh,0.03126),...). Finally, the keywords in the centroid of this 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 for discovering topic vectors from the log of a single user. The same algorithm can be applied to a log recording the access history of multiple users. The topic vectors discovered would be the common areas of interest among them. If these topic vectors are used by a goal-directed search engine, the retrieved documents would match the common interest of these users.

The Learning Agent can provide the system with the topic vectors which reflect the users areas of interest. Moreover, the Monitor Agent described in Section 3 can adaptively determine when these useful and interesting 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.

5 Experimental Results

Our prototype system has been implemented on AIX on an IBM 410 RS/6000 workstation. The proxy is a CERN httpd server. Nine volunteers have been

Table 2: Size and number of clusters generated vs threshold values.

Threshold		0.00	0.05	0.10	0.15	0.20	0.50
User 3	Max	24	22	12	8	7	4
	Mean	2.18	2.02	1.57	1.39	1.27	1.08
	# clusters	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
	# clusters	307	342	467	568	643	879

accessing the WWW via the proxy for a month. We have collected the user access logs from these volunteers. The logs are then submitted to the Learning Agent in the system to discover the topic vectors and subsequently the areas of interest. Table 1 shows the number of unique term vectors generated for each users.

One of the major steps in the discovery of the users topics of interest is to identify the clusters. The similarity threshold is an important parameter in the determination of the clusters. A smaller threshold would generate less but larger clusters, but they may not reflect a very focusing topic. A larger threshold would produce more clusters in total and each one of them would be better focused, but smaller in size. Figure 5 is the number of clusters discovered against different similarity thresholds for all the nine users in our experiment. When the threshold is set to 0, any two term vectors having an overlapping keyword 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 near the end of our test range.

In Table 2, we have recorded the clustering results of three selected users for different thresholds. For each user, the first row (Max) is the size of the largest cluster. The second row (Mean) is the average cluster size. The third row is the number of clusters generated.

It is also informative to investigate the distribution of the clusters for different thresholds. Figures 6 and 7 are the distributions of the clusters against their sizes and similarity thresholds. The distribution of user 4 is shown in Figure 6. 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 close to the average. The distribution of the clusters of these user is shown in Figure 7. Among all the users in our experiment, it can be observed that the cluster distributions are more visible when the threshold is less than or equal to 0.1.

Figure 6: Cluster size distribution of user 4.

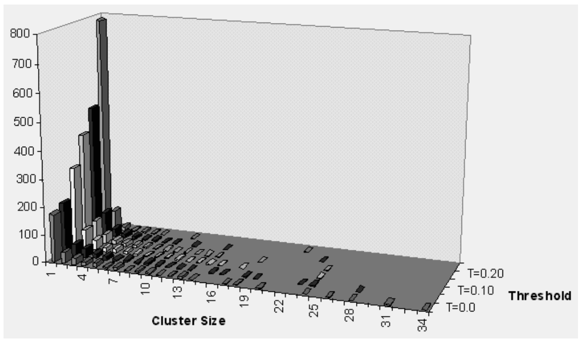
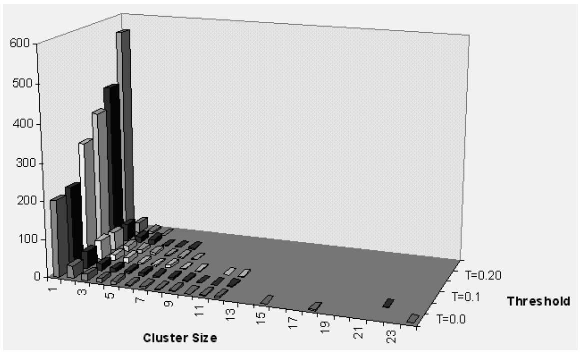


Figure 7: Cluster size distribution of user 3.



6 Discussion and Conclusion

In this study, the architecture of a level 4 Web tool has been described. The Web tools can discover the users access patterns and bring in documents that users are anticipating. An algorithm for the discovery of topic vectors and areas of interest from user access logs has been proposed. The key in the discovery process is to identify the relevancy of the documents and their keywords in order to select those that have higher influence in the determination of relevant topics. Several important techniques have been adopted in the adjustment of document relevancy. Clustering technique is used to find out topic vectors from term vectors whose weights are adjusted. To prove its feasibility, we have built a prototype system to demonstrate the ideas proposed in this study. The algorithm for mining the users topic vectors from their access logs has been implemented in the prototype and the result shows that the areas of interest discovered are relevant and meaningful. 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 page access sequences, and the Learning Agent can be enhanced to discover these types of access patterns.

References

1. 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. 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.
4. 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.
5. 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.
6. W.B. Frakes, R. Baeza-Yates. *Information Retrieval: Data Structure and Algorithms.* Pentice-Hall, 1992.
7. D.H. Jones and D. Navin-Chandra. IndustryNet: A Model for Commerce on the World Wide Web. *IEEE Expert*, October 1995, pp.54-59.
8. H. Lieberman. Letizia: An Agent that Assists Web Browsing. *International Joint Conference on Artificial Intelligence*, 1995.
9. G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, Reading, MA. 1989.
10. 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.
11. 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.