

Reorganizing Web Sites Based on User Access Patterns

Yongjian Fu Ming-Yi Shih Mario Creado Chunhua Ju¹

Department of Computer Science

University of Missouri-Rolla

{yongjian,mingyi,mcreado,chunhua,}@umr.edu

Abstract

In this paper, an approach for reorganizing Web sites based on user access patterns is proposed. Our goal is to build adaptive Web sites by evolving site structure to facilitate user access. The approach consists of three steps: preprocessing, page classification, and site reorganization. In preprocessing, pages on a Web site are processed to create an internal representation of the site. Page access information of its users is extracted from the Web server log. In page classification, the Web pages on the site are classified into two categories, index pages and content pages, based on the page access information. After the pages are classified, in site reorganization, the Web site is examined to find better ways to organize and arrange the pages on the site. An algorithm for reorganizing Web sites has been developed. Our experiments on a large real data set show that the approach is efficient and practical for adaptive Web sites.

1. Introduction

The use of the World Wide Web as a medium for business, communication, education, and government has increased at an amazing rate over the past few years. The goal of a Web site is to meet the needs of its users. As the interests of its users change over the time, a static Web site that does not change itself will soon become outdated and less useful. A Web site must constantly examine its users and their use of the site, and modify itself accordingly to best serve its users. In other words, Web sites should be adaptive. An adaptive Web site has been defined as a Web site that semi-automatically improves its organization and presentation by learning from visitor access patterns [15].

In this paper, an attempt is made to build adaptive Web sites which improve their navigation based on access patterns of its users. To find information on a Web site, a large number of Web users browse the site first. A Web site should make the navigation as easy as possible so that its users can find the interested information quickly. Moreover, with the interests of its users changing over time, the structure of the Web site should evolve as well. For example, a book on Pearl Harbor may become popular because of the same name movie. It makes sense to promote the book on an on-line bookstore, for example, by adding a link from the home page to the page about the book.

An approach for reorganizing Web sites based on user access patterns is proposed. Our goal is to build adaptive Web sites by evolving site structure to facilitate user access. To be more specific, we aim to build Web sites that provide its users the information they want with less clicks. This minimizes the effort on the user's side. By analyzing the usage of a Web site and the structure of the Web site, modifications to the Web site structure are found to accommodate changes in access patterns of its users. These modifications will be suggested to the Webmaster for consideration and implementation.

¹ Visiting from Hangzhou University of Commerce, China.

The approach consists of three steps: preprocessing, page classification, and site reorganization. In preprocessing, pages on a Web site are processed to create an internal representation of the site. Page access information of its users is extracted from the Web server log. In page classification, the Web pages on the site are classified into two categories, index pages and content pages, based on the page access information. A page classification algorithm has been developed which uses data about a page's type, structure, and usage to determine its category. After the pages are classified, in site reorganization, the Web site is examined to find better ways to organize and arrange the pages on the site. An algorithm for the reorganization of the site has been developed.

The approach has been implemented and tested on a large real data set. Initial experiments show that the approach is efficient and practical for adaptive Web sites. The reorganized Web site requires fewer clicks for users and is thus easier to navigate. Although more experiments are needed, our approach does show promising potentials.

The paper is organized as follows. In Section 2, a brief overview of the background and related work in Web usage mining and adaptive Web sites is given. Section 3 describes the preprocessing of pages on the Web site and the Web server log. The classification of the pages is discussed in Section 4, with an algorithm for Web page classification. Section 5 illustrates various cases and corresponding adjustments in site reorganization and presents an algorithm for site reorganization. Results obtained from various experiments are reported in Section 6. Our study is concluded in Section 7 which also gives possible future work.

2. Background and Related Work

To analyze the usage of the Web, Web mining, especially Web usage mining, has been proposed by many researchers [4,6,10,19]. Web usage mining is the mining of Web usage data [4,19]. In most Web usage mining studies, Web server logs are used as the primary data source, although client and proxy level logs may be used [10]. A Web server log collects a large amount of information about user activities on the Web site by keeping information about the requests of pages on the server.

Most commonly used Web servers such as NCSA's HTTPD and Microsoft's IIS, maintain a log of page requests. For each page request, a record is kept in the log whose format is defined by W3C as Common Log Format (CLF) [9]. CLF specifies the fields in the record, including the IP address of the user, the date and time of the request, the URL of the page, the protocol, the return code of the server, and the size of the page if the request is successful. An extended version of CLF, Extended Log File Format (ELFF), has also been proposed [8], which is more flexible and comprehensive with fields, derivatives, and comments. A few examples of log records in CLF are given in Table 1. The IP addresses are modified for privacy reasons.

Table 1. Excerpt from a Web Server's log file

dan.cs.umr.edu	--	[01/Aug/1997:13:17:45	-0600]	"GET	/~dan/a.html"	200	34
131.39.170.27	--	[01/Aug/1997:13:17:47	-0600]	"GET	/~white/ Home.htm	200	2034
dan.cs.umr.edu	--	[01/Aug/1997:13:17:48	-0600]	"GET	/~dan/b.html	200	8210
131.39.170.27	--	[01/Aug/1997:13:17:50	-0600]	"GET	/~white/cloud.gif	200	4489
131.39.170.27	--	[01/Aug/1997:13:17:51	-0600]	"GET	/~white/hobby.htm	200	890
117.83.344.74	--	[01/Aug/1997:13:17:51	-0600]	"GET	/~katz/arrow.jpg	200	2783

As the primary and most accurate source of Web usage data available, the Web server logs provide a solid basis for obtaining insight into the trends and patterns in user access. Of course, the raw log records must be parsed, cleaned, calculated, and finally grouped into sessions. A session theoretically

represents a single visit of a user to the Web site. Such preprocessing of the server log is explained in Section 3.2.

A lot of studies have been conducted in Web usage mining. Some focus on the mining of association rules and navigation patterns in the user access paths [1,4,14,21]. A session is viewed as a transaction in association rule mining and algorithms for association rule mining are employed to find frequent paths that are followed by many users. Others build data cubes from Web server logs for OLAP and data mining [2,22]. The statistics along pages, IP domains, geographical location of users, and access time are calculated from sessions. Some others cluster users based on their access patterns [7,13,18]. There is also research on data preparation [5] and query language [20] for Web usage mining, and Web personalization [11] based on Web usage mining.

Recently, research into adaptive Web site has been proposed by some researchers. An initial definition of the problem was presented in [15]. Clustering of pages based on access patterns has been studied in [16]. Web pages that are not directly linked but are frequently accessed together are clustered and an index page can be synthesized to link these pages together. In [12], pages are clustered based on their occurrences in frequent paths that are found through association rule mining.

In this research, we attempt to use the results from Web usage mining to reorganize the Web site. Page access information of users is coupled with the knowledge of how the Web site is organized and is expected to function, for the purpose of discovering and recommending suitable changes to the site organization. The main difference between our approach and those in [12,16] is that we do not create clusters of pages, rather we let the Web site's structure and organization evolve as the usage evolves.

3. Preprocessing

There are three tasks in preprocessing. The first is Web site preprocessing to obtain the current structure of a Web site, i.e., how the pages are linked. The second is server log preprocessing to organize access records into sessions. The third is to collect access information for the pages from the sessions.

3.1. Web Site Preprocessing

The purpose of this phase is to create an internal data structure to represent the Web site. The Web site is represented as a directed graph in which a page is a node and a link is an arc. Each page of the Web site is parsed sequentially and the links in the page (tags beginning with <A HREF>) are extracted. Each page is assigned a unique page identifier (PID). For each page, PIDs of pages which has a link to it (called its *parents*) and pages which it links to (called its *children*) are stored.

Currently, the Web pages are assumed to be static. Dynamic pages such as those generated by CGI or other server-side scripts are ignored. All non-HTTP references, e.g., "ftp://", "gopher://", "mailto:", etc, are filtered out because they do not represent site structure. In addition, all references to pages on other sites, e.g., a reference to Adobe site for Acrobat reader, are also removed. This is reasonable since these pages are not part of the Web site and cannot be modified, thus should not be included in the reorganization process. Also, multiple links between two pages are treated as one and intra-page links (an intra-page link is a link to the page it is in) are ignored.

3.2. Server Log Preprocessing

Since a lot of irrelevant information for Web usage mining such as background images is also included in the server log, it has to be processed first. A number of preprocessing algorithms and heuristics exist [5]. The steps involved in preprocessing of the server log are as follows.

1. Records about image files (.gif, .jpg, etc) are filtered as well as unsuccessful requests (return code not 200).
2. Requests from the same IP address are grouped into a session. A timeout of 30 minutes is used to decide the end of a session, i.e., if the same IP address does not occur within a time

range of 30 minutes, the current session is closed. Subsequent requests from the same IP address will be treated as a new session.

3. The time spent on a particular page is determined by the time difference between two consecutive requests.

The server log files are transformed into a set of sessions. A session represents a single visit of a user. Each session contains a session ID and a set of (PID, *time*) pairs, where PID is the page identifier and *time* is the time the user spent on the page.

There are some difficulties in accurately identifying sessions and estimating times spent on pages.

- Due to client or proxy caching of pages, the server log may not reliably detect the page requests from users. Some heuristics have been proposed, for example, in [4]. An intrusive method is to install a client-monitoring program. Generally, it is a hard problem.
- The users are identified by the IP addresses used. However this could be prone to errors since IP addresses could be reused or shared. The timeout technique helps to detect different users by setting a limit on idle time, although it is not always precise. It also helps to avoid endless sessions. If necessary, positive means of session identification, such as cookies or embedded sessions IDs, could be used.
- The amount of time a user spent on a page is determined by the time difference between two consecutive requests. This may not reflect the actual viewing time due to network congestions, transmission speed, and interruptions. Besides, the time the user spent on the last page can never be known since it is the last request of the session and there is no more requests after it.

Although the server log is not perfect for Web usage mining, it gives us rough idea about page access. Moreover, it is widely available without client-side programming or other intrusive methods. It provides a comprehensive source of access information with reasonable accuracy.

For example, the Web server log in Table 1 will be organized into sessions as shown in Table 2. It should be noted that session IDs are not IP addresses since they may be reused or shared. Different visits from the same IP address will be identified as different sessions.

Table 2. Sessions from the server log.

Session ID	IP Address	Time/Date	Requested Page
1	dan.cs.umr.edu	01/Aug/1997:13:17:45	/~dan/a.html
		01/Aug/1997:13:17:48	/~dan/b.html
2	131.39.170.27	01/Aug/1997:13:17:47	/~white/Home.htm
		01/Aug/1997:13:17:51	/~white/hobby.htm

From Table 2, it is possible to estimate how much time the user spent on each page by taking the difference in date and time between the current page request and the following page request. For example, in session 1, the user spent 3 seconds on the first page, /~dan/a.html.

3.3. Access Information Collection

In this step, the access statistics for the pages are collected from the sessions. The data obtained will later be used to classify the pages as well as to reorganize the site.

The sessions obtained in server log preprocessing are scanned and the access statistics are computed. The statistics are stored with the graph that represents the site obtained in Web site preprocessing. The obvious problem is what should be done if a page happens to be the last page of a session. Since there is no page requested after that, we really cannot tell the time spent on the page. Therefore, we keep a count for the number of times that the page was the last page in a session.

The following statistics are computed for each page.

- Number of sessions in which the page was accessed;
- Total time spent on the page;

- Number of times the page is the last requested page of a session.

4. Page Classification

In this phase, the pages on the Web site are classified into two categories: index pages and content pages [17]. An index page is a page used by the user for navigation of the Web site. It normally contains little information except links. A content page is a page containing information the user would be interested in. Its content offers something other than links. The classification provides clues for site reorganization.

The page classification algorithm uses the following four heuristics.

1. File type.

An index page must be an HTML file, while a content page may or may not be. If a page is not an HTML file, it must be a content page. Otherwise its category has to be decided by other heuristics.

2. Number of links.

Generally, an index page has more links than a content page. A threshold is set such that the number of links in a page is compared with the threshold. A page with more links than the threshold is probably an index page. Otherwise, it is probably a content page.

3. End-of-session count.

The end-of-session count of a page is the ratio of the number of time it is the last page of a session to the total number of sessions. Most Web users browse a Web site to look for information and leave when they find it. It can be assumed that users are interested in content pages. The last page of a session is usually the content page that the user is interested in. If a page is the last page in a lot of sessions, it is probably a content page; otherwise, it is probably an index page. It is possible that a specific index page is commonly used as the exit point of a Web site. This should not cause many errors at large.

4. Reference length.

The reference length of a page is the average amount of time the users spent on the page. It is expected that the reference length of an index page is typically small while the reference length of a content page will be large. Based on this assumption, the reference length of a page can hint whether the page should be categorized as an index or content page. More detailed explanation is given in Section 4.1.

A page classification algorithm based on these observations and heuristics is presented in Section 4.2.

4.1. Reference Length Method

The reference length method for page classification [3] is based on the assumption that the amount of time a user spends on a page is a function of the page category. The basic idea is to approximate the distribution of reference lengths of all pages by an exponential distribution. A cut-off point, t , for reference length, can be defined as follows.

$$t = -\ln(1 - \gamma) / \lambda$$

where γ = percentage of index pages,

λ = reciprocal of observed mean reference length of all pages

The definition comes from integrating the formula for an exponential distribution from zero to γ . If a page's reference length is less than t , it is more likely an index page, otherwise, it is more likely a content page.

Therefore, if the percentage of index pages on a site, γ , is known, a reference length can be calculated, which estimates the cut-off between index and content pages. In most cases, such a percentage is unknown and has to be estimated. For a Web site, the percentage of pages that are index pages can be

estimated based on the structure and content of the site or the experience of the data analyst with related server.

Several factors limit the accuracy of the reference length method. First, as mentioned in Section 3.2, the reference length of a page is only an estimate. Network congestions, interruptions such as a phone call or a coffee break, and browser caching, will all cause disparity between the actual viewing time and the time computed from the server log. Second, the method does not consider the sizes of pages when determining the cut-off point. A large index page requires a long reference time and can be mistaken as a content page. Third, the estimation of γ is largely dependent on experiences. Nevertheless, the method provides a simple and robust way of classifying pages.

In the calculation of the cut-off point, the last pages of sessions are omitted, because as mentioned in Section 3.2, the time spent on the last page of a session is unknown.

4.2. Algorithm for Page Classification

An algorithm for page classification is introduced in this section which combines the heuristics mentioned above. To determine the category of a page, its file type is first checked. If it is not HTML, the page is certainly a content page and no other testing will be necessary. Otherwise, its end-of-session count, number of links, and reference length, are examined subsequently.

Two thresholds, *count_threshold* and *link_threshold* are introduced. If a page's end-of-session count is greater than *count_threshold*, it is classified as a content page. If a page's number of links is greater than *link_threshold*, it is tagged as an index page. These thresholds should be selected conservatively so that they positively identify content or index pages. Finally, if necessary, the page's reference length is checked against the cut-off point t . If its reference length is less than t , it is marked as an index page; otherwise it is marked as a content page.

The algorithm for page classification is outlined as follows.

- (1) $\lambda = 1/(\text{mean reference length of all pages})$
- (2) $t = -\ln(1 - \gamma) / \lambda$
- (3) For each page p on the Web site
- (4) If p 's file type is not HTML or
- (5) p 's end-of-session count $>$ *count_threshold*
- (6) Mark p as a content page
- (7) Else If p 's number of links $>$ *link_threshold*
- (8) Mark p as an index page
- (9) Else If p 's reference length $<$ t
- (10) Mark p as an index page
- (11) Else
- (12) Mark p as a content page

5. Site Reorganization

After preprocessing and page classification, we are ready to reorganize the Web site based on the access information. The goal of this phase is to reorganize the Web site such that its users will spend less time searching for the information they desire. The philosophy behind this is that a Web site will provide a better service to its users if it can cut down their navigation time by reorganizing the pages on the site.

More specifically, we want to reorganize the pages so that users can access the information they desire with fewer clicks. Although other factors such as page layout affects navigation, the number of clicks a user has to go through is the dominant factor for navigation since every click requires active rather passive effort from users and often involves a request to and an reply from the server.

The general idea of reorganization is to cut down the number of intermediate index pages a user has to go through. To achieve this, we need to place the frequently accessed pages higher up in the Web

site structure, i.e., closer to the home page, while pages that are accessed infrequently should be placed lower in the structure.

In the meantime, we want to preserve the original site structure whenever possible, since it may bear business or organizational logics. Besides, dramatic changes of the site structure may confuse users.

As a compromise between these two conflicting requirements, we introduce an evolutionary approach to Web site reorganization. The basic idea is to locally adjust the site when a frequently accessed page should be promoted.

In addition, two thresholds are introduced, that is, maximum number of links in an index page (I) and maximum number of links in a content page (C). An index/content page will not have more than I/C links after site reorganization, unless it has more links before reorganization, in which case its links will be intact. These two thresholds are introduced to achieve two objectives. First is to limit the number of links in a page so its layout will be reasonable. This will prevent extreme cases, for example, a flat site structure where all pages are linked from the home page. Second is to somehow contain the changes in the site structure. The selection of these thresholds can be done by the Webmaster or data analyst.

5.1. Cases in Site Reorganization

As mentioned earlier, in site reorganization, frequently accessed pages are put higher up in the site structure. On the contrary, infrequently accessed pages are placed lower in the site structure. In case such reorganization is not possible due to certain threshold such as maximum number of links in a page being exceeded, we will try to merge infrequent pages into a larger page. The mergers will reduce the number of clicks by users due to fewer page requests, thus decrease navigation time. To prevent spurious results, the merging pages must be HTML files and at most one of them can be a content page.

To decide if a page is frequently accessed, a parameter, minimum frequency (F), is introduced. A page's frequency is defined as the number of sessions it is in divided by the total number of sessions. If a page's frequency is greater than F , it is called a frequent page, otherwise, it is an infrequent page.

In site reorganization, the pages are examined sequentially starting from the home page. For each page, we consider its immediate parents and children, where a parent is any page that has a link to it and a child is any page that it has a link to. Depending on the number of children it has, there are different cases and for each case, different actions may be taken according to the frequency and category of the pages involved.

For each page, we consider three cases depending upon the number of children it has: 1, 2, and 3+. The three cases are illustrated as follows. For the sake of simplicity and also since the processing is done one parent at a time, only one parent is considered in the cases.

(I) CASE I: The current page has one child. In this case, depending on the frequencies and categories of the pages, there are several possible outcomes, as shown in Figures 1, 2, and 3, where page B is the current page.

(a) Page B is an index page.

Obviously, page B is redundant since it only serves as a link to page C. Thus the most obvious solution will be to delete page B and create a direct link from page A to page C, as shown in Figure 1.

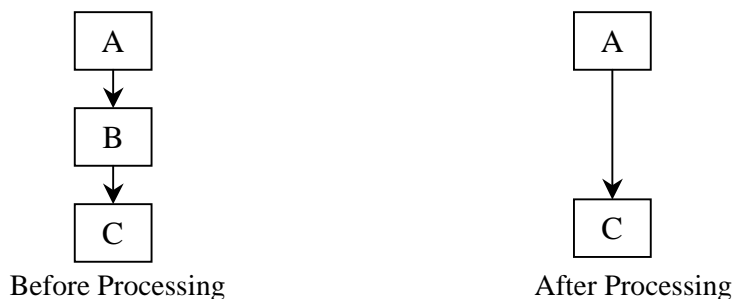
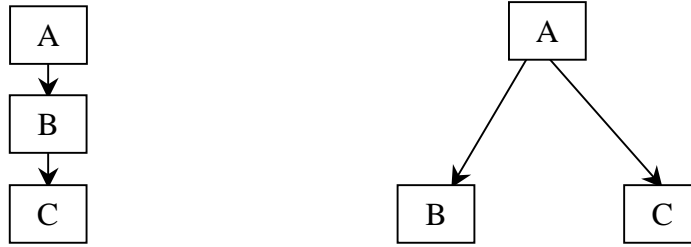


Figure 1. Case I, page B is an index page.

(b) Page B is a content page and page C is frequent.

Since page C is frequent, it should be promoted by adding a direct link from page A to it as shown in Figure 2. This assumes that page A has a free link, i.e., adding a link will not exceed its number of links limit. The maximum number of links in page A is determined by its category (I for index page and C for content page).



Before Processing
After Processing
Figure 2. Case I, page B is a content page and page A has a free link.

If page A has used its links to full capacity, but page C have a free link, it is sometimes worthwhile to demote page B to be a child of page C as shown in Figure 3. This is done if page B is used mostly to fetch page C. This happens when the frequency of page C is more than half the frequency of page B.

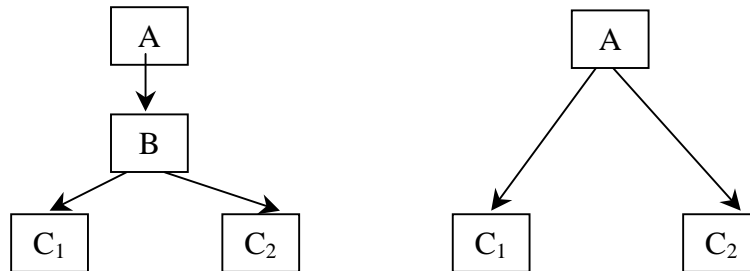


Before Processing
After Processing
Figure 3. Case I, page A has no free links, but C does.

(II)CASE II: The current page has two children. Again, depending on the frequencies and categories of the pages, there are several possible scenarios, as shown in Figures 4, 5, 6, 7, and 8, where page B is the current page and pages C_1 and C_2 are children of B. Without loss of generality, we assume that the frequency of page C_1 is greater than that of page C_2 .

(a) Page B is an index page.

Page B will be removed whenever possible. The easiest scenario is shown in Figure 4 where two direct links from page A to page C_1 and page C_2 are added. However, since two links will be added in page A while only one is deleted, it can only be possible if page A has an extra link to spare.



Before Processing
After Processing
Figure 4. Case II, page A has a free link.

If page A does not have a free link, we will try to merge pages C_1 and C_2 . Two or more pages can be merged if at most one of them is a content page and their total frequency does not exceed F .

Moreover, the merged page will be a content page if a participating page is a content page; otherwise, the merged page is an index page. The limit on the number of links also applies to the merged page. If C_1 and C_2 can be merged, page A will link to the merged page, as shown in Figure 5.

If page A does not have a free link and pages C_1 and C_2 cannot be merged, page A will link to page C_1 which will in turn link to page C_2 , as shown in Figure 6. Of course, this happens only when page C_1 is frequent and has a free link.

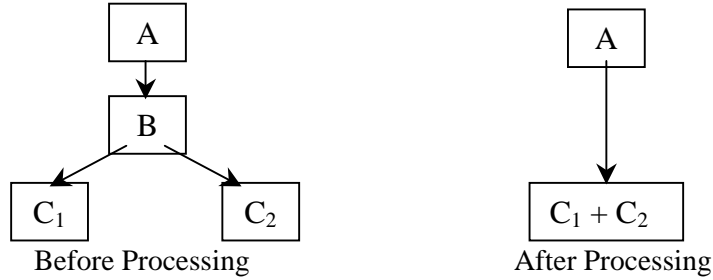


Figure 5. Case II, page A does not have a free link, but C_1 and C_2 can be merged.

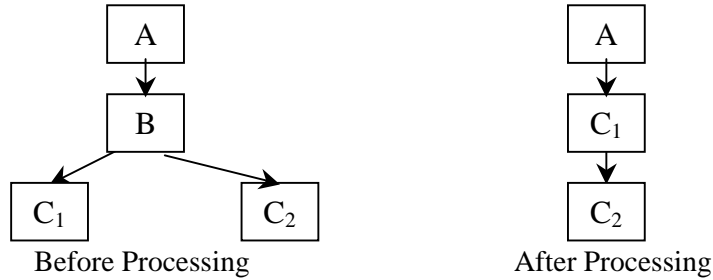


Figure 6. Case II, page A does not have a free link and C_1 and C_2 cannot be merged

(b) Page B is a content page.

Since page B is a content page, it cannot be deleted. However, if C_1 is a frequent page, it should be promoted higher in the structure. If page A has a free link, a link from page A to page C_1 is added, and the link from page B to page C_1 is removed, as shown in Figure 7.

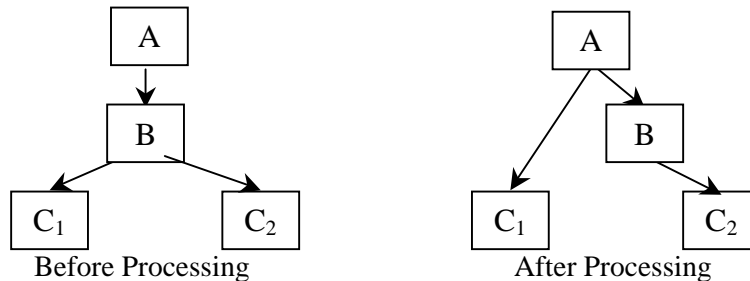
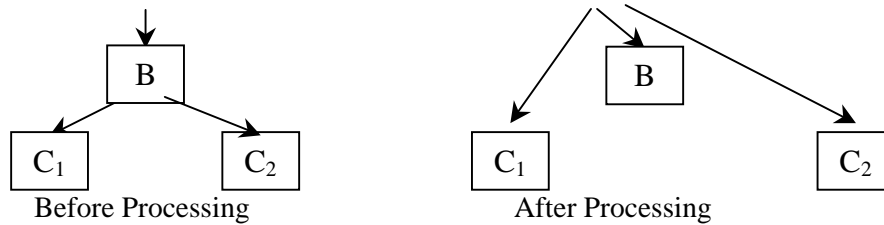


Figure 7. Case II, page B is a content page and page A has a free link.

If both pages C_1 and C_2 are frequent, they should be promoted higher in the structure. If page A has two free links, links from page A to pages C_1 and C_2 are added, and the links from page B to pages C_1 and C_2 are removed, as shown in Figure 8.

When C_1 is not frequent, no change on the structure is proposed. Note in this situation, C_2 will not be frequent either. If page A does not have enough links, the structure remains intact too.

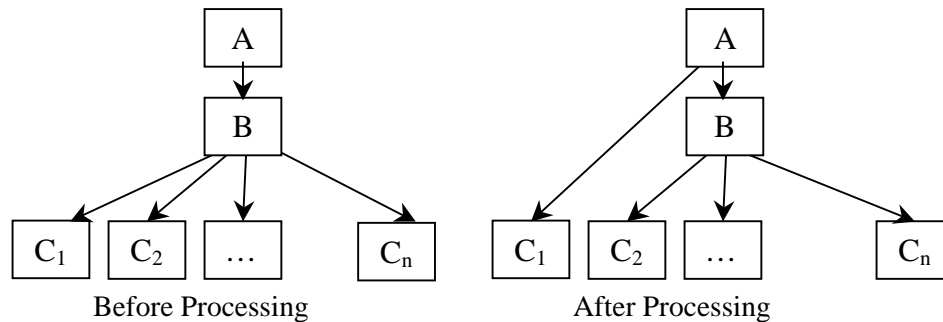




(III)CASE III: The current page has three or more children. There are several possible situations, as shown in Figures 9, 10, and 11, where B is the current page and C_1, \dots, C_n are children of B. Without loss of generality, we assume that the child pages are ordered in decreasing order of frequency. That is, frequency of page C_1 is greater than that of page C_2 and so on until C_n . Since there are many possible combinations of the frequency and category of pages, we focus on page C_1 . If page C_1 is a frequent page and is significantly more frequent than other children, i.e., its frequency is greater than or equal to the sum of the frequencies of C_2, \dots, C_n , C_1 should be promoted. Besides, we will try to merge infrequent pages.

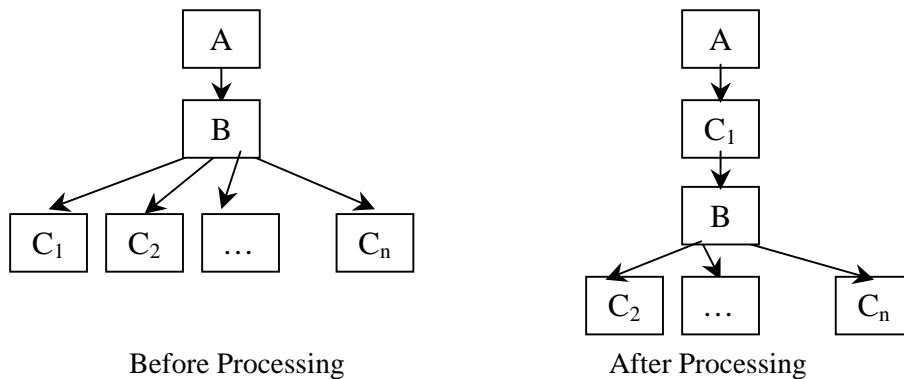
(a) Page C_1 is significant and A has a free link.

A link is added from page A to page C_1 , as shown in Figure 9.



(b) Page C_1 is significant and A does not have a free link, but C_1 does.

Since a significant number of requests is for page C_1 , but they have to go through page B, if page B is mostly traversed to get its children, it may be worthwhile to insert page C_1 between page A and page B, as shown in Figure 10. This is done when the frequency of page C_1 is more half the frequency of page B.



(c) Merge of infrequent pages.

As explained in the beginning of this section, infrequent pages are merged if possible. Assume pages C_i, C_{i+1}, \dots, C_n are infrequent. They are added into a merged page in the ascending order of frequency, i.e., from C_n to C_i . When no more pages can be added into the merged page because it will become frequent, or its number of links will exceed its limit, or a second content page is being added, a new merged page starts. The remaining pages are added into the new merged page in the similar way, until all pages are done. An example is shown in Figure 11.

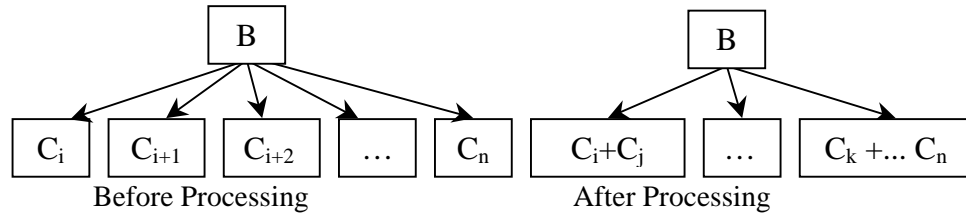


Figure 11. Merging infrequent pages.

5.2. Algorithm for Site Reorganization

Based on the cases discussed in the previous section, the algorithm for site reorganization is outline as follows.

- (1) Initialize a queue Q
- (2) Put children of the home page in Q
- (3) Mark the home page
- (4) While Q not empty
- (5) $current_page = pop(Q)$
- (6) Mark $current_page$
- (7) For each parent p of $current_page$
- (8) local adjustment according to the cases in Section 5.1
- (9) Push children (maybe merged) of $current_page$ into Q if they are not marked

6. Experiments

The algorithms have been implemented using C++ on a Sun Microsystems Ultra 10 machine with 256MB of memory running Solaris 2.6. The approach has been tested on the Hyperreal Web site (<http://www.hyperreal.org>). The server log used is for September and October 1997 (available at <http://www.cs.washington.edu/homes/map/adaptive/download.html>).

The server log is composed of 61 files, each representing one-day's requests, for every day in September and October 1997. The requests in the log files have been organized into sessions. For privacy reasons, the original IP addresses of the visitors have been replaced with unique session IDs. The log files are approximately 79MB in size containing about 78,000 sessions spanning around 700,000 page requests.

6.1. Experiments with the Page Classification Algorithm

To evaluate the page classification algorithm, we study the *precision* of classification for both index pages and content pages. For each page category, its *precision* is calculated as the percentage of pages in the category which are correctly classified. The recall of each category is not calculated since it is implied in the other category's precision. The precision is calculated by randomly selecting 20 pages of a category and counting how many are correctly classified.

The parameters used in the page classification algorithm are listed in Table 4. Various values of *link_threshold*, γ , and *count_threshold* are tested. Our experiments show that *count_threshold* does not affect the results very much. Due to space limit, experiments on *count_threshold* are not reported in the

paper. The value of *count_threshold* is fixed as 100 sessions (about 0.1%) for all experiments. The results for *link_threshold* and γ are reported in this paper.

Table 4. Parameters used in page classification.

Parameter	Definition
γ	estimated percentage of index pages.
<i>link_threshold</i>	maximum number of links a content page can have.
<i>count_threshold</i>	maximum end-of-session count for an index page.

Table 6 shows the number of pages and the precision of both categories for various values of *link_threshold*. The value of γ is set to 60%.

Table 6. Effects of *link_threshold* on precision.

<i>link_threshold</i>	Number of index pages	Precision of index pages	Number of content pages	Precision of content pages
0	891	100%	341	80%
5	887	100%	355	85%
10	885	100%	357	85%
15	838	95%	404	85%
20	744	95%	498	90%
25	720	90%	522	90%
30	710	90%	532	90%
35	707	90%	535	90%
40	702	90%	540	90%

As shown in Table 6, when *link_threshold* increases, the number of pages classified as index pages decreases because a page needs to have more links to be classified as an index page. Understandably, the precision for index pages decreases when *link_threshold* increases. For the content pages, it is just the opposite. The best overall precision is achieved when *link_threshold* is around 20.

Table 7 shows the cut-off point, *t*, and the number of pages and the precision of both categories for various values of γ . The value of *link_threshold* is set to 20.

Table 7. Effects of γ on precision.

γ	Cut-off point <i>t</i> (in seconds)	Number of index pages	Precision of index pages	Number of content pages	Precision of content pages
10%	4	258	65%	984	95%
20%	9	391	80%	851	95%
30%	14	502	85%	740	95%
40%	21	591	90%	651	95%
50%	28	669	90%	573	95%
60%	38	744	95%	498	90%
70%	49	803	95%	439	90%
80%	66	880	100%	362	85%
90%	95	952	100%	290	85%

As shown in Table 7, when γ increases, the cut-off point *t* increases which means more pages will be classified as index pages and less pages as content pages. This is obvious considering the definition of γ , which estimates the percentage of index pages.

For small values of γ ($<30\%$), the algorithm may classify some index pages as content pages. As γ increases, the number of pages classified as index pages increases, and so does the precision of index pages. However, a large value of γ ($>80\%$) will cause some content pages to be classified as index pages, thus reduces the precision of content pages. For our test data set and the Web site, the best values of γ ranges from 40% to 70%. Depending on the nature of Web sites, the value of γ should be estimated to best reflect the actual percentage.

To summarize, the page classification algorithm correctly classifies the majority of pages. The parameters, γ and *link_threshold*, need to be tuned according to the characteristics of Web sites.

6.2. Experiments with the Site Reorganization Algorithm

To evaluate the effectiveness of the site reorganization algorithm, we examine the number of pages, the number of links, as well as the average number of clicks in a session, before and after the reorganization.

The parameters used in the site reorganization algorithm are listed in Table 5. The effects of various values of F are reported in this paper. The values of I and C are fixed to 30 and 10, respectively. Results from most other values of I and C show only minor differences. Since the total number of sessions is fixed at 77629, we use absolute values instead of percentages for F for illustrative purpose.

Table 5. Parameters used in site reorganization.

Parameter	Definition
F	minimum frequency.
I	maximum number of links in an index page.
C	maximum number of links in a content page.

Figure 12 shows the total number of pages on the reorganized Web site for various values of F . The original Web site is shown as $F = 0$. It turned out that for the Web site we tested, pages were not deleted, only merged. When $F = 0$, pages cannot merge because every page is a frequent page. However, this is apparently not always true for other Web sites.

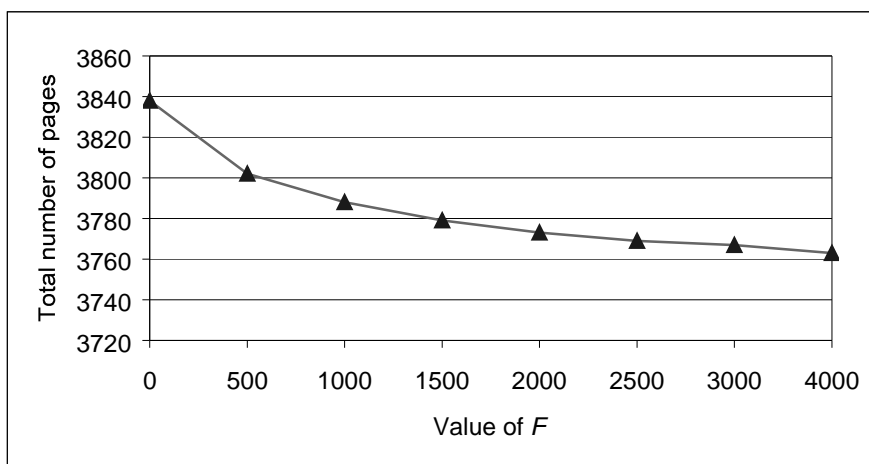


Figure 12. Effects of F on total number of pages.

As shown in Figure 12, when F increases, the total number of pages on the reorganized site declines. This is because when F increases, more and more pages will be counted as infrequent and more pages will be merged into a single page. It should be pointed out that only 1,242 out of 3,838 pages on the Web site are accessed in the log files. The majority of the pages on the site are untouched. This probably explains why the reorganization does not cause drastic changes in the total number of pages.

Figure 13 shows the total number of links in the pages on the reorganized site for various values of F . Again, the original Web site is shown as $F = 0$. Like the total number of pages and for the same reason, the total number of links in the pages on the reorganized site declines when F increases.

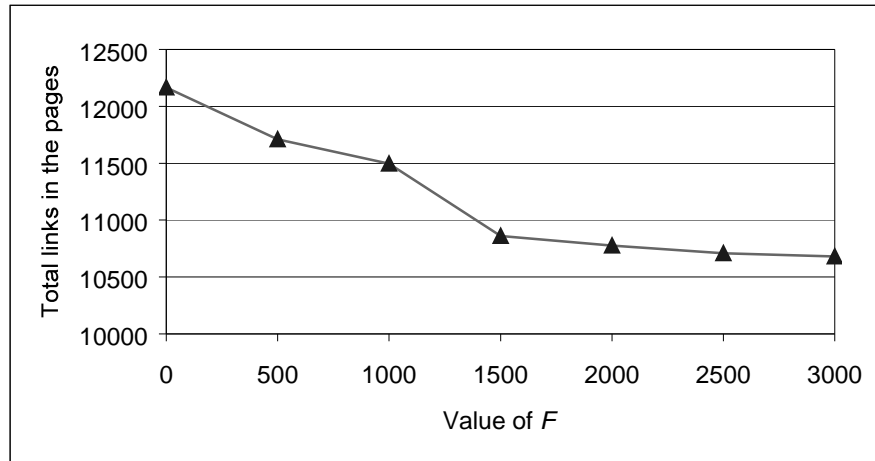


Figure 13. Effects of F on total number of links in the pages.

Figure 14 shows the percentage of decrease in the average number of clicks in a session for the reorganized Web site for various values of F . The original Web site is shown as $F = 0$. The average number of clicks in a session is always less in the reorganized site. This shows that the users on average will click less and navigate easier with the reorganized Web site. The percentage of decrease in the average number of clicks increases with F .

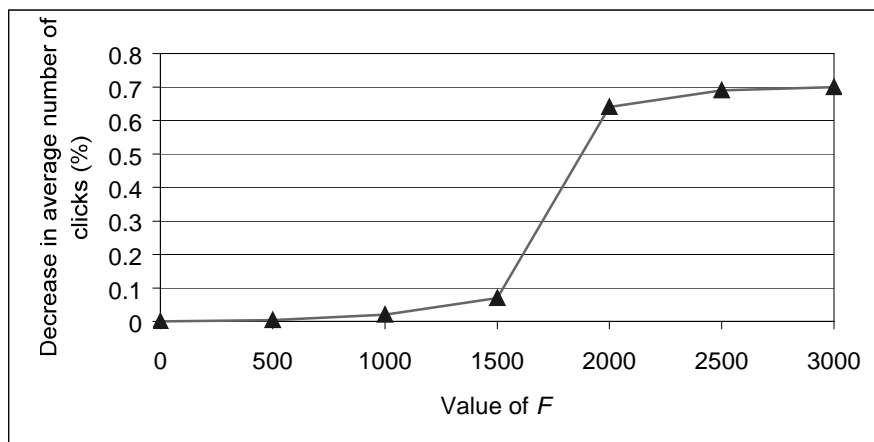


Figure 14. Effects of F on average number of clicks.

However if we look at the percentage of decrease, it is not significant. This can be attributed to the fact that the reorganization algorithm processes primarily the infrequent pages as explained above. Since these pages are not accessed by many users, the average number of clicks in a session does not change a lot. Additionally, since the Web site we tested is a commercial Web site, it is already quite well organized and hence the changes made during reorganization will not be many.

7. Conclusions and Future Work

A study on building adaptive Web sites is reported. An approach to reorganize Web sites based on user access patterns has been proposed. This approach aims to build Web sites that provide its users the information they want with less clicks. By analyzing the usage of a Web site and the structure of the Web site, modifications to the Web site structure are found to improve the structure of the Web site.

In this approach, the Web site and its server log are first processed to acquire its structure and access information. The pages on the site are then classified into index or content pages based on access information. The Web site is finally examined to find better ways to organize the pages. Two algorithms, one for page classification and the other for site reorganization, have been developed.

The proposed approach has been implemented and tested on a real data set from an actual Web site. The results demonstrate a high accuracy in page classification and a decrease in the number of clicks the user must perform to get interested information. Judging from the results obtained so far, it can be concluded that the approach is promising for adaptive Web sites.

We are currently working on more experiments on real and artificial data. They will help us to gain more insight on parameters selection and fine-tuning of the algorithms. Additionally, it will be interesting to see how the approach can be improved if other sources of data about the users are available besides the server log. An example is to extend the proposed algorithms for e-commerce sites where more accurate user data such as transactions are available. In such applications, the performance can be more effectively evaluated, for example based on revenues.

The current approach assumes each pageview contains a single page, i.e., a click will result a single file to be returned from server or cache. This is not true for Web sites with frames. An extension of our approach to deal with pageviews instead of pages will be interesting.

Alternatively, the current reorganization algorithm bases its decisions purely on the user accesses. It would be very interesting to study the possibility of incorporating knowledge of the Web pages by Web content mining to increase the effectiveness and reliability of the reorganization algorithm.

References

1. J. Borges and M. Levene, *Mining Association Rules in Hypertext databases*, Proc. 1998 Int'l Conf. On Data Mining and Knowledge Discovery (KDD'98), 149-153, 1998.
2. A. Buchner and M. Mulvenna, *Discovering Internet Marketing Intelligence through Online Analytical Web Usage Mining*, SIGMOD Record, 27, 1998.
3. R. Cooley, *Web Usage Mining: Discovery and Application of Interesting Patterns from Web Data*, Ph.D. dissertation, Univ. of Minnesota, 2000.
4. R. Cooley, B. Mobasher, and J. Srivastava, *Web Mining: Information and Pattern Discovery on the World Wide Web*, Proc. Int'l Conf. On Tools with Artificial Intelligence, 558-567, Newport Beach, CA, 1997.
5. R. Cooley, B. Mobasher, and J. Srivastava, *Data Preparation for Mining World Wide Web Browsing Patterns*, Journal of Knowledge and Information Systems, 1, 1999.
6. O. Etzioni, *The World Wide Web: Quagmire or Gold Mine*, Communications of the ACM, volume 36, number 11 (November), pp. 65-68, 1996.
7. Y. Fu, K. Sandhu, and M. Shih, *Clustering of Web Users Based on Access Patterns*, International Workshop on Web Usage Analysis and User Profiling (WEBKDD'99), San Diego, CA, 1999.
8. P. M. Hallam-Baker and B. Behlendorf, *Extended Log File Format*, <http://www.w3.org/pub/WWW/TR/WD-logfile.html>
9. A. Luotonen, *The Common Log File Format*, 1995, <http://www.w3.org/pub/WWW/Daemon/User/Config/Logging.html> .
10. S. Madria, S. Bhowmick, W. K. Ng, E. P. Lim, *Research Issues in Web Data Mining*, DAWAK'99, Florance, Italy, Sept. 99.

11. B. Mobasher, H. Dai, T. Luo, M. Nakagawa, Y. Sun, and J. Wiltshire, *Discovery of Aggregate Usage Profiles for Web Personalization*, Proceedings of the Web Mining for E-Commerce Workshop (WebKDD'2000), Boston, August 2000.
12. B. Mobasher, R. Cooley, and J. Srivastava, *Creating Adaptive Web Sites Through Usage-Based Clustering of URLs*, Proceedings of the 1999 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX'99), November 1999.
13. G. Paliouras, C. Papatheodorou, V. Karkaletsis, and C. D. Spyropoulos, *Clustering the Users of Large Web Sites into Communities*, Proceedings Intern. Conf. on Machine Learning (ICML), pp. 719-726, Stanford, California, 2000.
14. J. Pei, J. Han, B. Mortazavi-Asl, and H. Zhu, *Mining Access Pattern Efficiently from Web Logs*, Proc. 2000 Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'00), Kyoto, Japan, April 2000.
15. M. Perkowitz and O. Etzioni, *Adaptive Web sites: an AI Challenge*, Proceedings of Int'l Joint Conf. on Artificial Intelligence (IJCAI), 16-23, 1997.
16. M. Perkowitz and O. Etzioni, *Adaptive Web sites: automatically synthesizing Web pages*, Proceedings of Fifteenth National Conference on Artificial Intelligence, Madison, WI, 1998.
17. A. Scime and L. Kerschberg, *Websifter: an Ontology-based Personalizable Search Agent for the Web*, Proc. Int'l Conf. on Digital Libraries, 439-336, Kyoto, Japan, 2000.
18. C. Shahabi, A. Zarkesh, J. Adibi, V. Shah, *Knowledge Discovery from Users Web-Page Navigation*, In Proceedings of the IEEE RIDE97 Workshop, April 1997.
19. M. Spiliopoulou, *The laborious way from data mining to web mining*, Int. Journal of Comp. Sys., Sci. & Eng., Special Issue on "Semantics of the Web", 14:113-126, Mar. 1999.
20. M. Spiliopoulou and L. C. Faulstich, *WUM: A Tool for Web Utilization Analysis*, EDBT Workshop WebDB'98, Valencia, Spain, Mar. 1998. Springer Verlag
21. M. Spiliopoulou, L. Faulstich, and K. Winkler, *A Data Miner analyzing the Navigational Behaviour of Web Users*. Workshop on Machine Learning in User Modeling of the ACAI'99 Int. Conf., Creta, Greece, July 1999.
22. O. R. Zaiane, X. Xin, and J. Han, *Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining technology on Web Logs*, Proc. Advances in Digital Libraries, 19-29, 1998.