Revisitation Patterns in World Wide Web Navigation

Linda Tauscher and Saul Greenberg
Department of Computer Science, University of Calgary
Calgary, Alberta CANADA T2N 1N4
Tel: +1-403-220-6087
E-mail: tauscher *or* saul@cpsc.ucalgary.ca

ABSTRACT
In this paper, we report on users' revisitation patterns to World Wide Web (WWW) pages, and use the results to lay an empirical foundation for the design of history mechanisms in Web browsers. Through history, a user can return quickly to a previously visited page, possibly reducing the cognitive and physical overhead required to navigate to it from scratch. We analyzed 6 weeks of detailed usage data collected from 23 users of a commercial web browser. We found that 58% of an individual's pages are revisits, and that users continually add new Web pages into their repertoire of visited pages. People tend to revisit pages just visited, access only a few pages frequently, browse in very small clusters of related pages, and generate only short sequences of repeated URL paths. We compared different history mechanisms, and found that the stack-based prediction method prevalent in commercial browsers is inferior to the simpler approach of showing the last few recently visited URLs with duplicates removed. Other predictive approaches fare even better. These results suggest new approaches to managing history in browsers.

Keywords
History mechanisms, WWW, hypertext, navigation.

1. INTRODUCTION

The World Wide Web (WWW) hypertext system is a large, distributed repository of information. People use graphical browsers to navigate through links and to view pages. These browsers typically provide *history mechanisms* that allow people to select and revisit pages they have viewed previously. If people revisit pages often, such history mechanisms can mitigate three problems people face when navigating the WWW. First, they can help the user navigate through the vast amounts and poor structure of WWW information by providing easy access to information previously visited. Second, they can decrease resource use by supplanting search engines for finding old pages, and by eliminating navigation through intermediate pages en-route to the destination. Third, they can reduce a user's cognitive and physical navigation burdens. Pages can be returned to with little effort, and they can show users where they have been.

Research Report 96-587-07, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada. 1996

However, today's design of history mechanisms tend toward ad-hoc approaches that do not appear to take advantage of previous research into history support within user interfaces e.g., Greenberg (1993) and Lee (1992). In particular, their designs are not based upon actual studies of how people revisit Web pages, and their actual use has been examined only superficially.

Our goal is to place the design of history mechanisms within browsers on an empirical foundation. We had two sub-goals.

1. We wanted to understand people's revisitation patterns when navigating the WWW, yet little empirical data is available. The proportion of Web pages that are revisited by a particular user has not been quantified, and no research has examined patterns of page reuse. In section 3, we will present quantitative results about revisits to Web pages, and examine five possible patterns of reuse.
2. We wanted to evaluate current approaches in today's history systems, validate successful solutions, and suggest better alternatives. Yet today's history mechanisms are rarely evaluated. From research by Catledge and Pitkow (1995), we know that *Back* is heavily used to return to a page, but the history list is not. Cockburn and Jones (1996) performed a usability study that illuminated user's difficulties with the current stack-based history mechanism. However, the goodness of predictions offered by this and other history schemes have not been evaluated, which we will do in Section 4.

2. DATA COLLECTION

XMosaic 2.6 was modified to record a user's browsing activity. Each activity record included time, URL visited, page title, final action, method of invoking the action, user id, and several other items. Volunteer participants then used the browser for approximately six weeks; all were practiced Web users with at least one year of experience. At the end of the study, we analyzed logs from 23 participants. This was followed by hour-long individual interviews, done to gather qualitative data about personal browsing methods and to help use understand why the patterns seen in the analysis arose.

3. RESULTS: HOW PEOPLE REVISIT WEB PAGES

Six analyses pertaining to Web page reuse are reported here. We begin by reporting the rate that Web pages are revisited. The remaining analyses concern five different patterns that may suggest effective approaches to presenting revisited pages for future access. First, we examine how users visit both old and new Web pages over time. Second, we look at the distance (in terms of URLs) between repeated visits to the same URL. Third, we assess the frequency of URL visits. Fourth, we determine the extent to which users repeatedly browse within locality sets (page clusters). Last, we identify repeated sequences of URLs as an estimate of path-following behaviour.

3.1. Recurrence of Web page visits

History systems are only useful if users actually repeat their activities. While Web browsers contain a history mechanism, we do not know how often people revisit their pages. This is important, as it gives us a bound for how useful a history system can be. In other domains, research has quantified this repetition of user actions e.g., telephone numbers dialed (57%), how information is retrieved in a technical manual (50%), and how Unix command lines are entered (75%) (Greenberg, 1993). We analyzed our own data and the Catledge and Pitkow (1995) data to derive the *recurrence rate R*: the probability that any URL visited is a repeat of a previous visit. Each user's data was analyzed independently, and the statistics below represent either averages across all users, or representative individuals.

We found an overall recurrence rate of R=58% (σ = 9%) for our 23 subjects, and 61% (σ = 9%) for 55 subjects from the Catledge and Pitkow study. These numbers clearly show that users revisit Web pages heavily, although it also means that ~40% of all page navigations are to new pages. These recurrence rates qualify Web browsing activity as a *recurrent system*. Greenberg (1993) coined this term to characterize systems where users predominately repeat activities they had invoked before, while still selecting new actions from the many that are possible.

In post-study interviews, people gave us their major reasons for visiting new pages and revisiting old ones. They revisit pages because: the information contained by them changes; they wish to explore the page further; the page has a special purpose (e.g. search engine, home page); they are authoring a page; or the page is on a path to another revisited page. People visit new pages because: their information needs change; they wish to explore a particular site; a page is recommended by a colleague; or they notice an interesting page while browsing for another item.

Since web browsing is a recurrent system, we believe that browser interfaces should support page revisits by minimizing a user's effort to return to a page when compared to the effort of navigating there from scratch. The key is to give preferential treatment to the large number of repeated actions. This involves identifying patterns in history use, as discussed in subsequent sections.

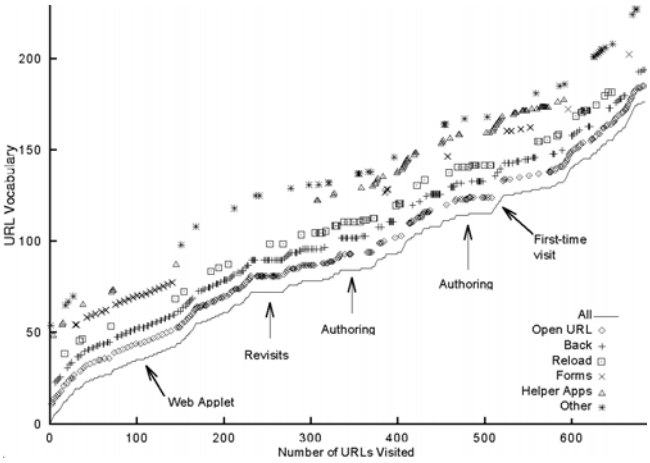

Figure 1. URL Vocabulary for participant 15

3.2. Growth of URL vocabulary

The first pattern considered shows the distribution of old and new page visits over time. We generated vocabulary graphs for each subject, where the URL *vocabulary*—the number of unique URLs seen so far—is plotted over the total number of URLs visited. These plots illustrate how users extend their vocabularies, and how recurrences are distributed over time.

For example, Figure 1 shows the plot for participant 15. The curve *All* represents the overall URL vocabulary size at any moment in time. Major navigation actions are also plotted as separate curves shifted above the vocabulary line by a constant amount: *Open URL*, *Back*, *Reload*, *Forms*, and *Helper Applications*. The *Other* category includes all remaining navigation actions. These curves show when the most common navigation actions were invoked and, taken together, comprise the *All* curve.

URL vocabulary growth graphs for all subjects exhibit a linear and positive slope, typified by the line in Figure 1. Both data and interviews indicate that users incorporate new URLs into their repertoire at a regular rate, and that revisits are fairly evenly distributed.

These plots are only roughly linear, and many local variations to the slope are also evident. We noticed that the nature and extent of these vary amongst individuals. We analyzed these variations and their corresponding navigation actions, and asked participants questions about them during interviews. Consequently, we identified seven browsing patterns, four of which are illustrated in Figure 1.

1. *First-time visits* to a cluster of pages is evident at the steeply sloped area between URLs 510 to 525.
2. *Revisits to pages*. Plateau areas show revisits to pages. For example, the long plateau in combination with the Back or Open URL actions between URLs 240 to 280

occurred when this participant reviewed a set of online course notes.

3. A*uthoring of pages.* These manifest themselves as plateaus, where the subject used Reload extensively to view the modified page e.g., between URLs 480 to 510.
4. Regular use of *web-based applications.* Between URLs 50 to 150 (x-axis), there is a moderately sloped area with a combination of Open URL, Back and Forms activity. This was caused by the subject's revisits to a knowledge elicitation tool packaged as a Web application.

Three other browsing patterns were seen in other subjects.

5. *Hub-and-spoke.* People visit a central page (hub) and navigate the many links to a new page (spoke) and back again. This is akin to a breadth-first search.
6. *Guided tour.* Some page sets include structured links (e.g., *next page),* and people can choose to follow these.
7. *Depth-first search*. People follow links deeply before returning to a central page, if at all.

Browsers and history mechanisms should support the many browsing patterns users exhibit. For example, stack-based history mechanisms and the Back button found in most browsers supports both hub-and-spoke and depth-first search patterns. The Reload button is very convenient for authoring. Guided tours contain hyperlinks that encourage a linear pattern of navigation. Yet improvements in history designs are possible. Perhaps the excessive backtracking that results from depth-first navigation styles and the hub-and-spoke could be reduced by a graphical overview map, or by retaining the index page within the browser window, as often done with Netscape Navigator 2.0's *frames* feature.

In this section, we saw that both old and new Web pages are visited regularly over time. The huge number of pages in the vocabulary implies that we cannot simply offer all previously visited pages within a visual history system, as this would be unmanageable. Instead, we must offer the user only a handful of candidate pages that have a high probability of being revisited. Researchers in other domains have noticed that recently used actions are the ones most likely to be repeated. (Greenberg 1993). This warrants an investigation into recency effects by considering URL visits as a function of distance.

3.3. URL visit frequency as a function of distance

For any URL visited, the probability that it has already been seen by the user is quite high (R=58%). But how do particular URLs contribute to this probability? Do all URLs visited have a uniform probability of recurring, or do the most recently visited URLs skew the distribution? If a graphical history mechanism displayed the previous p entries as a list, what is the probability that this includes the next entry (Greenberg, 1993)?

The recurrence distribution as a measure of *distance* was calculated for each subject. Distance is determined by counting the number of items between the current URL being visited from its first match on the history list. For example, a distance of 1 occurs when the user reloads the current page, or successfully interrupts the transmission of a page. A distance of two occurs when the current page is a revisit to the one seen two pages back. Figure 2 plots this data up to a distance of 50, averaged across all subjects. The horizontal axis shows the distance of the repeated URL on the history list relative to the one being entered. The vertical axis represents the rate of URL recurrence at a particular distance, denoted as R_d. According to Figure 2, there is a R_{d1} = 10% probability that the current URL is a repeat of the previous URL (distance = 1), R_{d2} = 19% for a distance of 2, R_{d3} = 2%, R_{d4} = 5%, and so on. The spikes at distances of 2, 4, 6 and 8 arise from users' navigating back to previous pages by the 'Back' navigation action.

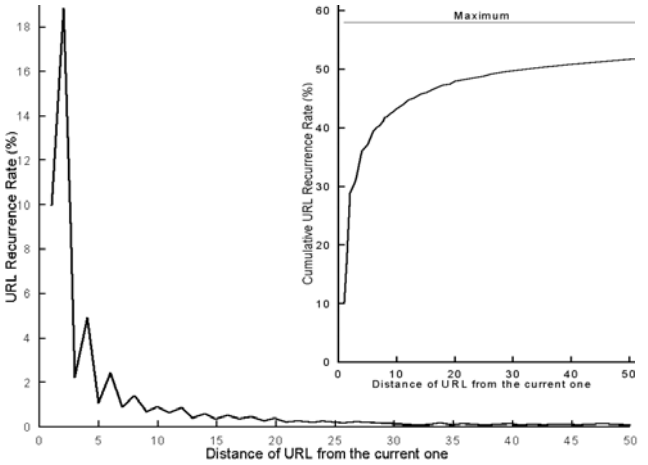

Figure 2. URL recurrence rate as a function of distance (all participants); inset plots recurrence rate as a running sum

The most striking feature of this data is the extreme recency of the distribution. The 6 or so URLs just visited contain the majority of pages visited next, although the probability values of R_d rapidly decrease after the second item. Beyond a distance of 8, the low values (< 1% each) and low rate of decrease make items equivalent for practical purposes.

This is illustrated further in the inset of Figure 2, which reports the same data for all subjects as a running sum of the probability, denoted as R_D. The horizontal line at the top of the graph is the maximum mean recurrence rate for all subjects of 58%. The most recently visited URLs are responsible for most of the cumulative probabilities. For example, there is an R_{D6}=39% chance that the next URL visited will match a member of a set containing the 6 previous submissions. In comparison, all further contributions are slight (though their sum total is not).

3.4. Frequency of URL accesses

Frequency is a popular method for ranking items of interest. We examined this pattern in two ways. First, we generated a frequency graph for each subject. Second, we developed a taxonomy of conceptual page types for frequently visited pages from our post-study interviews.

All subjects produced a similar frequency distribution, where only a small number of URLs are highly visited, and a very large number of URLs have very low usage frequencies. Over all subjects, 60% of pages were only visited once, 19% were visited twice, 8% were visited 3 times, and 4% were visited 4 times. The few pages that were frequently accessed tend to fall into certain categories that also explained their popularity: personal pages, start-up documents, indices, search engines, individual and organization home pages, Web applications, navigation pages, and authored pages.

3.5. Locality

While recency characterizes recurrences in terms of distance, *locality* considers recurrences in terms of periods of time where repeated references are made solely to a small and related group of items, called a locality set (Lee, 1992). We applied the locality detection algorithm (Madison and Batson, 1976) to the WWW data to determine whether users generate locality sets, that is, whether they browse within clusters of pages.

While locality sets were found (Tauscher 1996), they do not appear to offer much value in terms of predicting the user's next activity within Web browsing. There are several reasons for this claim. First, most locality sets were very small, consisting of only one or two unique URLs. Second, these sets lasted for only a short time, usually 2.5 to 4.5 URLs. Third, few locality sets were repeated; those that were repeated tended to be of size one or two. Fourth, only 15% of URLs visited were part of a locality set.

3.6. Longest Repeated Sequences

The concept of *paths*, an ordered traversal of hypertext links, has been associated with hypertext ever since Vannevar Bush envisioned hypertext in 1945. If paths exist, it may be useful to capture and offer them via history, thus simplifying people's efforts to retrace a path. Also, if users follow paths solely as a route to a destination, shortcuts could allow a user to go directly there.

We applied the Pattern Detection Module (PDM) algorithm (Crow and Smith, 1992) to the WWW browsing data in an attempt to identify longest repeated sequences (LRSs) of page visitations. As with locality, we discovered that LRSs are not particularly useful for predicting Web browsing for a variety of reasons. We found that though LRSs do exist, they tend to be short (Tauscher 1996). The few longer LRSs usually reference only one or two pages. In terms of repetition, the average frequency for LRSs of all lengths hovered around two which is the minimum requirement to be considered a LRS. Also, there is a strong recency effect: repeats of LRSs occur within a short distance of each other.

4. CONDITIONING THE DISTRIBUTION

The recurrence distributions were derived by considering all page visits for a user as one long input stream with no barriers placed between sessions. We have seen in Section 3.3 that although a small set of recently visited URLs accounts for a high proportion of revisits, others lie outside. Consider a set of the 10 previous URLs on the history list. From the inset in Figure 2, there is a 42% chance that the next URL has not appeared before, an R_{D10}=43% chance that it has occurred within the last 10 visits, and a 15% chance that it appeared at distances greater than 10. This section explores the possibility that the distribution can be conditioned, first to increase the recurrence probabilities over a set of a given size, and second to evaluate methods that are currently in use. Eight conditioning methods are presented within four major categories: recency, frequency, stack, and hierarchically structured. A later results section will consider how effective each method is.

We will illustrate these methods by using the small sample trace in Table 1, which shows the last 16 pages visited by a user. Pages are numbered by order of visit, with #16 being the most recently visited page. The user's action to navigate to those pages are shown on the right. Italicized pages are revisits. Each conditioning method is then applied to this trace, and the ordering of items that will be shown to the user in the conditioned history list is given in Table 2.

4.1. Recency ordered history lists

Three types of recency ordered history lists were evaluated. The first is *sequential ordering*, the time-ordered list of all URLs visited by the user, including revisits to the same URL. Thus the history list as shown in Table 2a is a literal trace, and is an exact match to the trace in Table 1.

The problem is that repeated items have multiple entries, which occupy valuable space on a history list of a limited length. Hence, two strategies for pruning redundant URLs were applied: saving the URL only in its *original position* on the history list, and saving the URL only in its *latest position*. Tables 2b and 2c provide an example of both approaches to pruning duplicates. Note that there are fewer URLs on these lists as compared to the strict sequential version which retains all URLs. Also, note that the user's StartUp document (a heavily accessed page) occupies the bottom position on the list (#1) when URLs are saved in their original position, while it is propagated up the list when they are saved in the latest position (#10). Thus we expect the 'latest position' approach to perform better, because just revisited URLs will stay at the top of the list (whereas they migrate to the bottom on 'original position'), and because local context is maintained (Greenberg, 1993).

Greenberg (1993) claims two benefits of recency-based history. First, the URLs presented are the ones the user has just visited. Thus, the user will remember and can effectively predict which URLs will appear on the list. Second, recency does not suffer from the initial startup instability that other methods do when there are only a few URLs available to present to the user.

Visit #	URL	Action
16	acsl.cs.uicu.edu/kaplan/applets	Open URL
15	acsl.cs.uicu.edu/kaplan/worlds-environ	Open URL
14	acsl.cs.uicu.edu/kaplan/worlds	Open Hotlist
13	www.acm.org/sigchi/chi96/forms/Proc	Open URL
12	www.acm.org/sigchi/chi96/call/index	Open URL
11	*www.acm.org/sigchi/chi96/*	Open URL
10	*www.acm.org/sigchi/homepage*	StartUp Doc.
9	*www.acm.org/sigchi/homepage*	Back
8	*www.acm.org/sigchi/cscw96/*	Back
7	www.acm.org/sigchi/cscw96/dates	Open URL
6	www.acm.org/sigchi/cscw96/	Open URL
5	*www.acm.org/sigchi/homepage*	Back
4	*www.acm.org/sigchi/chi96/*	Back
3	www.acm.org/sigchi/chi96/Deadlines	Open URL
2	www.acm.org/sigchi/chi96/	Open URL
1	www.acm.org/sigchi/homepage	StartUp Doc.

Table 1. A trace of the last 16 pages visited, and the user actions to get them. The top page (16) was just visited.

4.2. Frequency ordered history lists

Frequency ordering, where the most revisited page appears at the top of list and the least visited page appears at the bottom, is an obvious way of ranking URLs. Frequency ordering could be problematic. While user's needs and interests change quickly, the newer URLs need to be revisited frequently before they can migrate to the top of the list. Similarly, older frequently used items that are no longer of interest remain near the top. Still, there are certain types of pages that users tend to frequent regularly, and perhaps we can expect useful offerings of frequency ordering after periods of extended browsing (which stabilizes the frequency distribution).

An issue associated with frequency ordering is how to break ties with URLs that have the same frequency. Greenberg (1993) evaluated two schemes for *secondary sorting* within frequency ordered lists: recency and reverse-recency. Recency was found to perform better so that is the method of secondary sorting that we have applied. Table 2d shows the effect of frequency ordering with secondary sorting by recency upon the navigation session in Table 1.

4.3. Stack-based approaches

Current Web browsers maintain a history list that operates as a stack; the most recently visited page is usually pushed onto the top of the stack, so older pages appear underneath. Unlike recency, pages can be popped off the stack and lost. The way browsers push and pop pages from the stack depends upon three techniques the user employs for displaying the page: loading, recalling, and revisiting (Cockburn and Jones, 1996). In their terminology, *loading* a page causes it to be added to the top of the stack, possibly resulting in all pages above the current position to be lost. *Recalling* a page changes the pointer to the currently displayed page in the stack. *Revisiting* a page occurs when the user reloads the page, and has no effect upon the stack.

a) Sequential ordering by recency
16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

b) Recency, duplicates in latest position
16, 15, 14, 13, 12, 11, 10, 8, 7, 3

c) Recency, duplicates in original position
16, 15, 14, 13, 12, 7, 6, 3, 2, 1

d) Frequency, second key recency
10, 11, 8, 16, 15, 14, 13, 12, 7, 3

e) Stack, non-persistent
16, 15, 14, 13, 12, 11, 10

f) Stack, persistent
16, 15, 14, 13, 12, 11, 10, 1

g) Context-sensitive Web subspace
14 (16, 15)
10 (13, 12, 11, 8, 7, 3)

h) Hyperlink sublist (session 1 only)

16	12 (13)	7
15 (16)	11 (12, 3)	3
14 (15)	10 (11, 6)	
13	8 (7)	

Table 2. Examples of history lists conditioned by different methods. Numbers represent the URLs in Table 1.

We expect that the stack method will perform reasonably well for very short recurrence distances, as it will appear similar to recency. It will lag at modest distances because some recent URLs are popped off the stack when the user loads a page while at some point other than the stack's top. It will do poorly for long distances because current browsers clear the stack between sessions. Table 2e shows the history list based on this *sessional stack* at the end of the example navigation session. Note that the trace in Table 1 shows that a new session started on page #10 (indicated by the startup document action), so earlier pages are lost.

Because we may want to revisit pages from previous sessions, we constructed a *persistent stack* that retains the stack from the prior session. While the persistent stack will perform similar to the sessional stack for short distances, it should do better for long distances because some URLs are retained between sessions. Still we do not expect this method to perform better than recency ordering with no duplicates due to the absence of some URLs. Also, the persistent stack will be longer than necessary due to the presence of duplicates. Table 2f shows this persistent stack; it now includes a pointer to a page from the first session.

4.4. Hierarchically structured history lists

Two methods that employ hierarchical structuring were examined: *recency ordered hyperlink sublists*, and *context-sensitive Web subspace* history lists.

Recency ordered hyperlink sublists is similar to the recency ordered history list with duplicates saved only in their latest accessed position. The difference is that for each URL on the normal list, a secondary recency-based history list can be raised containing only those pages that the user had

visited by selecting a hyperlink from the current page. The user first scans down *i* entries in the normal list for an exact match that terminates the search, or for an entry that contains the desired hyperlink. In the latter case, the sublist of hyperlinks is displayed (perhaps as a cascading menu) and the search continues until an exact match is found *j* entries later. The distance of a matching recurrence is simply *i* + *j*. Table 1h shows the hyperlink sublist. The hierarchy in this example is not very full because it is generated from a short trace; we expect that longer traces would fill the slots in the secondary lists.

We expect recency ordered sublists to perform better than recency with duplicates saved in their latest position. First, more URLs are accessible through a hierarchy. Second, if the user needed to visit a page only because it contained a hyperlink to the desired page, that page can now be selected directly from a sublist. Third, because sublists contain only URLs that the user has already accessed from a particular page, the user may find it easier to find a specific URL on the sublist, especially if the jump-off page is long and contains many irrelevant links.

The *context-sensitive Web subspace* is based upon a graphical history display designed by Cockburn and Jones (1996). The display creates a new Web 'subspace' each time the user *directly accesses* a page. This page is added as an item in a *Webs* menu. Any pages accessed until the next directly-accessed page (the subspace) are added to a secondary menu that cascades from the original *Webs* menu entry. For our analysis, we considered the following actions as a *direct access* to a URL: typing a URL, selecting a Hotlist item, cloning or opening a new window, and accessing a URL via client-dependent hard-wired buttons or menus. Within the main and secondary menu, we sort the URLs based on recency, and remove duplicates, saving according to the latest position. A URL can thus occur only once in the main menu or a particular secondary menu, but it may be found within several subspaces if the user navigated to it in different ways. This is appropriate since subspaces seem to be a reasonable method for inferring a user's context when browsing. That is, when the user follows a series of hyperlinks, many of the pages visited will tend to be related in some way.

Table 1g shows the history list at the end of the navigation session. There are 2 URLs in the main menu indicating that 2 different URLs were direct accessed (the *StartUp Document* and the *Open Hotlist*). The last subspace the user browsed is located at the top of the list. The sublists show the contents of the web subspace sorted in recency order.

4.5. Evaluation Methods

We evaluated all methods described above by implementing them as algorithms, and using our subjects' traces to simulate their performance in practice. Our evaluation accounts for the following factors.

1. *Theoretical performance.* We know that a perfect predictor of revisits could not better, on average, the recurrence rate R=58%, which is reached only if the user reuses previous URL visits at every opportunity. Thus we can see how close our algorithms are to this upper bound.
2. *Presentation.* We assume predictions will be displayed to users as some kind of visible list, perhaps with a hierarchical sub-list (e.g., cascading menu). Calculations are made on data structures that mimic these lists.
3. *Number of items.* While showing a history list of the complete trace would give optimum performance, this is not realistic given the large number of items that users would have to scan through. Pragmatically, each list should contain only a small set of previous submissions and offer them to the user as predictions. We consider how well a method performs with different list sizes by calculating predictability as a running sum over distance, where a chosen distance now indicates the maximum number of items scanned in the history list. Figure 3 was created this way. We have already introduced R_D to indicate the cumulative recurrence rate R at a particular distance D. We will also use R_{D10} (R at a distance of 10) as a comparative benchmark. This is reasonable: as seen in Figure 3, the first few items (D<10) of all methods contribute the most to the running sum, and there are diminishing returns for showing lists with D>10.
4. This analysis does *not* account for the conceptual model that the history list presents. Yet this is important, for a person's ability to predict whether the item they seek is present on the list is generated from their conceptual model. Consequently, we will discuss (but not measure) the simplicity of the method's conceptual model.

Our analysis does not compare a user's cognitive and physical effort involved in choosing items from each history method. For example, selecting items from a hierarchical sublist would be more effort than choosing from a sequential list, because a user must decide what branches to follow, raise cascading menus, and so on. There is probably a tradeoff between predictability (especially with complex methods) and effort, and future research should measure this as well.

4.6. Results

A strict sequential list of URLs ordered by recency performs reasonably well when a small set of URLs are listed e.g. R_{D10} = 43%. A benefit of this method is that its conceptual model is simple and familiar. That is, a person knows what they have just done and can thus predict if an item will be on the history list. We will use this value as a benchmark to contrast other methods.

Pruning duplicates is a simple way of improving the performance of a recency-ordered list when duplicates are saved only in their latest position (R_{D10} = 47% vs. 43% for strict recency). While this type of list does not show the exact sequence of URLs visited by the user, it still presents

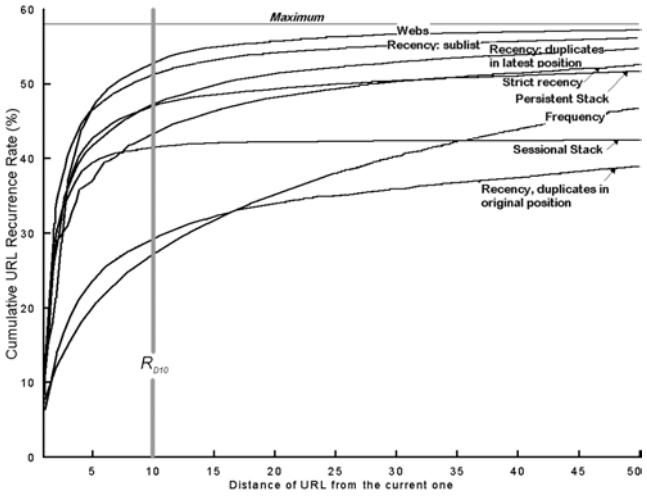

Figure 3. Cumulative probabilities of recurrences over distances up to 50

a clear conceptual model, and we expect that the user can easily understand how the lists contents are ordered. Saving duplicates in only their original position is very poor (R_{D10} = 29%), as frequently used items remain at the bottom. The striking differences between these three recency orders are illustrated by the plots in Figure 3.

Frequency ordering is the worst predictor of the 8 evaluated for short lists, with R_{D10} = 27% vs. 43%. While it does improve as distance is increased, it does not catch up to strict recency (Figure 3). Frequency has other problems. Users may find it more difficult to predict which pages would appear on a frequency-ordered list beyond the two or three that they visit the most. As well, frequency ordering suffers instability when few items are on the history list, and excessive inertia when the list is long. Still, frequency could be applied to a few key URLs, possibly used as an auxiliary method in conjunction with another history mechanism that gives better overall performance.

The sessional stack method found in most Web browsers is slightly better than strict recency at very short distances (R_{D5} = 40% vs. 37%) and slightly worse at R_{D10} (42% vs. 43%). As seen in Figure 3, it is much worse as the list gets long. The persistent stack is an improvement over the stack and strict recency methods in terms of its recurrence probabilities over distance (R_{D10} = 47% vs. 43%). Both approaches suffer problems as users typically form an incorrect conceptual model with this method—Cockburn and Jones (1996) discovered in their usability study that users were often surprised at the behaviour of their history list, and they could not predict how it worked. Because other methods are equal or better than even the persistent stack in terms of predictiveness (such as recency with duplicates removed), they are better choices.

Recency ordered hyperlink sublists have the highest recurrence probability over all methods for very short distances (2–4), and are second best at modest distances (R_{D10} = 51% vs. 43%). The catch is that this result is optimistic, since a person requires greater cognitive and physical effort to select items from the hyperlink sublists e.g., to make an accurate selection from a hyperlink sublist, the user must recall which main list item contains the desired URL. Also, note that hyperlink sublists can provide access up to 55 URLs for a R_{D10} (the main list of 10 items + 9 items on sublist one + 8 items on sublist two + 7 items on sublist three, etc.).

The best method we evaluated—context-sensitive web subspaces—showed that 53% of all URL selections can be successfully predicted with a set of 10 items. Given that R = 58% on average, which is the best a perfect reuse facility could achieve, this method is ~91% effective. The caveat is similar to hyperlink sublists, as users of context-sensitive web subspaces require greater physical effort to select a sublist item, and greater cognitive effort to recall which sublist might contain the URL. In addition, users may have more difficulty understanding how this method works, as they need to know what a 'direct access URL' is to grasp the way the history list is organized.

In conclusion, our analysis of conditioning methods shows that several methods improve upon the effectiveness of current stack-based history mechanisms. As seen in Figure 3, recency is a simple yet reasonable predictor, especially when duplicates are saved only in their latest position. A further appeal of recency is that it is conceptually easy for a user to understand. While the two hierarchical methods are better predictors, we suspect that they may not work as well in practice due to the extra physical and cognitive overheads mentioned earlier. Further research is required to evaluate how useful these methods are in practice.

4.7. Actual use of Web-based history mechanisms

We collected usage data on three history mechanisms found in Mosaic: backtracking through the 'Back' button, stack-based history lists, and personalized hotlists of URLs.

1. Backtracking was frequently performed, and 30% of all logged navigation actions involved the use of the Back action (plotted as a '+' in Figure 1). As mentioned previously, the spikes at even distances in Figure 2's URL distance plot is largely an artifact of consecutive invocations of Back.
2. Mosaic's hotlist and history facilities were used infrequently by our subjects (3% and <1% respectively).

The success of 'Back' is in line with our observation that extreme recency is a good predictor of what pages will be revisited. The poor use of other history facilities is likely due to interface issues. For example, hotlists require considerable effort to manage: users may not bother to add an URL to the list, may forget that it is there, or only record URLs that are convenient starting points. The 'Window History' in Mosaic 2.6 is not visible and requires several actions to access. Because it is based on the stack model,

the desired URL may have been popped off the list even though it was entered a short time ago.

5. DISCUSSION AND CONCLUSIONS

Based on these results we formulate some empirically-based generalizations of how users revisit pages using features found in a typical WWW browser such as Mosaic.

1. Users revisit a considerable number of Web pages. Our analysis of recurrence rate shows that there is a 58% probability that the next page visited was previously seen. This qualifies Web browsing as a recurrent system.
2. While many pages are revisited, users continually incorporate new pages into their repertoire at a regular rate. There are also local variations in the vocabulary growth rate and use of navigation activities across users, and across their browsing timeline. These variations indicate the presence of different browsing activities.
3. Users visit very few Web pages frequently. Consequently, many Web pages are only visited once (60%) or twice (19%). The few frequently accessed pages tend to fall into certain definable categories.
4. Users exhibit considerable recency of revisits. The major contributions to the recurrence distribution are provided by the last few pages visited, which also explains why 'Back' is frequently used (30% of all navigation actions).
5. Users revisit pages that have not been accessed recently. For example, 15% of recurrences are not covered by a list of the last 10 URLs visited. Still, doubling or tripling the size of the list does not increase its coverage much.
6. Users do not have strongly repeatable linear patterns when browsing selected clusters of pages. Both locality sets and longest repeated sequences are small, rarely repeated, and also exhibit recency.
7. Methods to present a history list of previously visited pages are available that are more predictive and usable than the current stack-based approach. Presenting the last 10 or so recent URLs, with duplicates saved only in the latest position, surpasses current stack based approaches and are likely much more usable. Other methods fare even better, although their usability must be determined.
8. The Back button is very effective and simple to use. Perhaps it could be improved further by basing it on a recency rather than a stack model.

Guidelines for graphical Web browser history mechanism design have been developed (Tauscher 1996) from the principles above and from Greenberg's (1993) general principles on reuse. Our guidelines address: access to pages the user has visited; reducing the cognitive and physical effort of using a history mechanism; providing a reasonable set of candidates for reuse; improving the conditioning method; supporting alternative strategies; and allowing end-user customization of the history data.

This paper provided empirical data that justifies the need for suitable history mechanisms in graphical Web browsers. Our analysis of different designs proves that the predictiveness of the current stack-based model can be improved upon. Using the methodology and principles herein, designers can refine current history mechanisms and investigate new approaches.

There are still many unanswered questions. We have not evaluated the physical and cognitive effort for reviewing a particular conditioned set of history list predictions. Also, we have not assessed the impact of different browser and HTML artifacts upon reuse such as *frames*, although we suspect that the numbers reported here would not change dramatically.

Acknowledgments. Many thanks to L. Catledge and J. Pitkow, who graciously provided us with their data logs for comparative purposes, and who gave us clues on how to instrument Mosaic by providing us with their version of it. The 28 people who volunteered to participate in the study made this research possible. This research was partially funded by Canada's National Sciences and Engineering Research Council.

REFERENCES

Bush, V. (1945). As we may think. *Atlantic Monthly*, 176(1), pp. 101-108, June.

Catledge, L. and Pitkow, J. (1995). Characterizing browsing strategies in the World-Wide Web. In *Proceedings of the Third International World Wide Web Conference*, Darmstadt, Germany. http://www.igd.fhg.de/www/www95/papers/

Cockburn, A. and Jones, S. (1996). Which way now? Analysing and easing inadequacies in WWW navigation. *Int. J. Man-Machine Studies*, in press.

Crow, D. and Smith, B. (1992) DB_Habits: Comparing minimal knowledge and knowledge-based approaches to pattern recognition in the domain of user-computer interactions. In Beale and Finlay (Eds.), *Neural Networks and Pattern Recognition in Human-Computer Interaction*, 39-61, Ellis Horwood.

Greenberg, S. (1993). *The computer user as toolsmith: The use, reuse, and organization of computer-based tools*. Cambridge University Press.

Lee, A. (1992). *Investigations into history tools for user support*. Ph.D. Thesis, Department of Computer Science, University of Toronto, Ontario, Canada.

Madison, A. and Batson, A. (1976). Characteristics of program localities. *Comm. ACM*, 19(5), 285-294.

Tauscher, L. (1996). *Evaluating history mechanisms: An empirical study of reuse patterns in World Wide Web navigation*. MSc Thesis, Department of Computer Science, University of Calgary, Alberta, Canada. May. http://www.cpsc.ucalgary.ca/grouplab/papers/