Contrasting Stack-Based and Recency-Based Back Buttons on Web Browsers

Saul Greenberg[1], Geoffrey Ho[2] and Shaun Kaasten[1]

[1]Department of Computer Science and [2]Department of Psychology
University of Calgary
Calgary, Alberta, Canada T2N 1N4
+1 403 220 6087
saul@cpsc.ucalgary.ca

Abstract

People frequently use the ubiquitous Back button found in most Web browsers to return to recently visited pages. Because all commercial browsers implement Back as a stack, previously visited branches of the tree are pruned. While this means that people can quickly navigate back up the tree, previously seen pages on alternate child branches are no longer reachable through Back. An alternate method implements Back on a recency model. Here, all visited pages are placed on a recency-ordered list with duplicates removed, which means that all previously seen pages are now reachable via Back. Because advantages and trade-offs exist in both methods, we performed a study that contrasted how people used stack *vs* recency-based Back. Surprising to us, several of our results were contrary to our expectations. First, people have a poor model of *both* stack and recency-Back. Second, people do not predict what pages will appear as they click Back. Rather, they use a 'click until recognize' strategy, where they simply click Back until they recognize the desired page. Third, people show no strong preference of recency *vs.* stack-Back. Consequently, we advocate replacing stack-Back with recency-Back only if other browser design considerations warrant it.

Key words: History system, browser design.

1 Introduction

A person's ability to find and navigate effectively to new information and to new web sites is extremely important, and this has driven many researchers to understand both how people navigate within the Web, and how Web sites should be designed. Equally important is a person's ability to return to pages he or she has already seen: page revisitation is a regular and surprisingly strong navigational occurrence. An early study by Tauscher and Greenberg [6] found that around 60% of all pages an individual visits are to pages they have visited previously. A later replication of this study by others found an even higher revisitation rate of around 80% [1].

Cite as:
Greenberg, S., Ho, G. and Kaasten, S. (2000) Contrasting Stack-Based and Recency-Based Back Buttons on Web Browsers. Report 2000-666-18, Department of Computer Science, University of Calgary, Alberta, Canada, August. Updated January 2002.

Given this revisitation statistic, we believe that Web browsers should go to great lengths to support effective page revisitation. Indeed, most browsers do provide revisitation support through various mechanisms: the Back and Forward buttons, history lists, bookmark facilities, and even site maps that graph the pages that a person has visited [2][6].

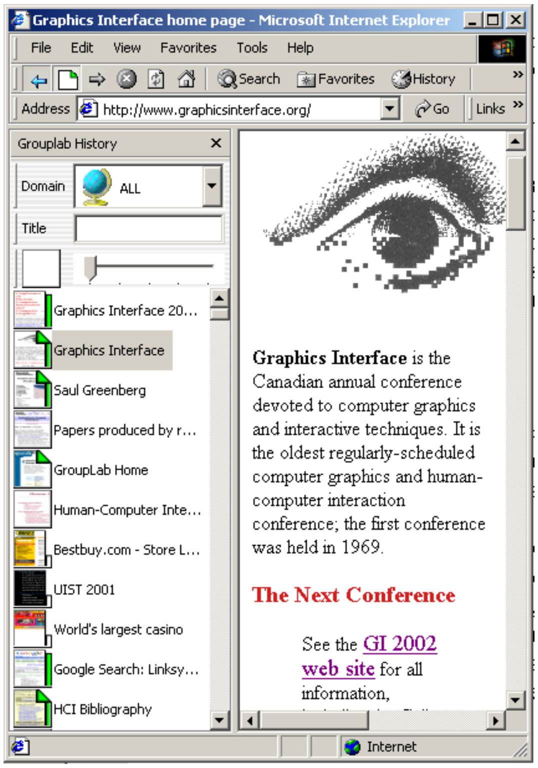
Of these revisitation mechanisms, it is the Back button whose use predominates: Tauscher and Greenberg [6] discovered that pressing the Back button comprised over 30% of all navigational acts. In contrast, other revisitation facilities are used infrequently e.g., <3% for bookmarks, and <1% for history systems.

Greenberg and Cockburn [4] detailed several reasons explaining Back's popularity.

1. Back allows people to rapidly return to very recently visited pages, which comprise the majority of page revisits. This is important, as Tauscher and Greenberg [6] found that there is a 43% chance that the next URL visited will match a member of a set containing the 10 previous visits. Because 60% of all pages are revisits, this means that 43÷60 = 72% of all revisited pages were just seen 1–10 pages ago.
2. Back requires little effort as a person merely clicks on it until the page is reached.
3. People are willing to keep Back on permanent display because it is visually compact.
4. People can use Back successfully even when they have a naïve understanding of the way it works [2].

Back's popularity as a revisitation tool means that it deserves special attention. Somewhat surprising to us is the wide-spread—and unchallenged—acceptance of the stack-based navigation model underlying Back and Forward in virtually all commercial browsers. If we can improve this behaviour even slightly, millions will feel the added benefit.

Consequently, our focus in this paper is to re-examine the usability of the way existing Back and Forward buttons work on a *stack*, and to compare it to an alternative button based on *recency*. We look at recency for several reasons (see Section 2.1). First, previous



Graphics Interface is the Canadian annual conference devoted to computer graphics and interactive techniques. It is the oldest regularly-scheduled computer graphics and human-computer interaction conference; the first conference was held in 1969.

The Next Conference

See the GI 2002 web site for all information,

Figure 1. Recency-based Back, history and bookmarks

research suggests that recency is closer to how people think Back actually works [2]. Second, the stack-Back loses pages, while recency-Back does not.

We also have a personal motivation. We have built a novel revisitation system that integrates Back, History and Bookmarks [5]. It represents pages on a sidebar as thumbnails and titles ordered by recency (Figure 1). Bookmarks are included in this list as specially marked 'dogeared' pages. Using dynamic queries, people can rapidly filter the list to show only frequently-visited pages, dogeared pages, pages within a particular domain, or pages whose title contains a particular string. Of particular relevance to this paper is that the Back and Forward buttons are also based on recency: they simply go up and down the list. While there are obvious advantages of making Back work directly on this visible list, we had no idea if a recency-based Back button would be acceptable to end users.

In this paper, we describe and contrast stack-based *vs.* recency-based Back button behavior. After summarizing how these behaviors work (Section 2), we introduce our study where we investigate how well people understand and use these two buttons (Section 3). We then present and discuss our results (Section 4), and we close by pointing out implications of our work to the design of web browsers.

2 Stack *vs.* Recency-based Back Buttons

This section summarizes two different behaviors for the Back and Forward buttons: the stack-based behavior found in today's web browsers, and a recency-based behavior proposed and implemented by Greenberg and Cockburn [4]. We will illustrate these two behaviors by showing how people navigate through the small page structure shown in Figure 2.

We use the notation $x \rightarrow y$ where '$\rightarrow$' means that the person has selected or typed a link on page x to go to page y. Similarly, in $y \Leftarrow x$, the '$\Leftarrow$' means backtrack from page y to page x via the Back button. We also define *hub and spoke* navigation as an action where people follow links from one parent (the hub) to two or more children (the spokes). For example, when navigating $b \rightarrow c \Leftarrow b \rightarrow h \Leftarrow b$ in Figure 2, b acts as a hub while c and h are spokes. Of course, c could also act as a hub page if the user navigates a similar pattern to two or more of c's children. This hub and spoke behavior deserves special attention because it is a common navigational act [6] and because it results in page pruning by stack-Back, as described below.

2.1 Stack

Description. The stack algorithm underlying a conventional Back button has three different types of operations.

1. Clicking or typing links adds a page to the top of the stack.
2. Clicking Back and Forward moves the stack pointer down and up the stack respectively, displaying the page at that stack location. The actual stack contents are not altered when navigating with these buttons.
3. When the user is at any position on the stack other than the top and selects a link on a web page, all entries on the stack above the current position are popped (or pruned) off the stack before the new page is added. This is critical, as pages popped off the stack can no longer be revisited using the Back and Forward buttons.

To illustrate these steps and how the stack behavior affects what people see, let us say a person follows the page links in Figure 2 from pages a through d in order, then presses Back twice to return to page b, and then selects a new link on page b to page h. Figure 3a shows the stack after a person navigates $a \rightarrow b \rightarrow c \rightarrow d$, where all pages were pushed onto the stack's top. In Figure 3b, we see that the two clicks of the Back button ($d \Leftarrow c \Leftarrow b$) moves the stack pointer down the stack to b. Navigating from $b \rightarrow h$ pops pages c and d off the stack (Figure 3c), and then adds page h to its top (Figure 3d). Thus pages c and d are no longer reachable through the stack-based Back button.

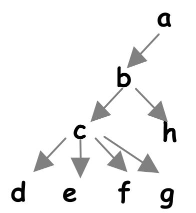


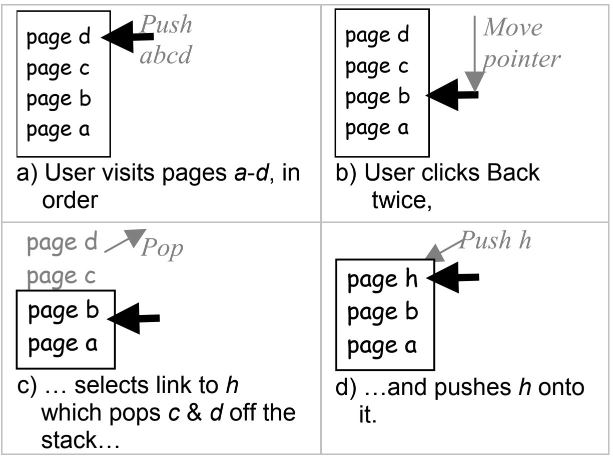
Figure 2. Example page structure.

Advantages and Disadvantages. The consequence of using a stack algorithm is that it automatically prunes navigational branches when people use Back followed by link selection. This approach has some merit: after exploring a branch and selecting a new path of interest the user may no longer need the previous branch of exploration. Because these pages are gone, a sequence of Back clicks will always move one 'up' the page hierarchy, making it easy to return to parent hub pages.

A counter-argument is that there are many cases where people *do* want to return to pages seen on a previously visited branch. For example, in the navigational trace described in Figures 2 and 3, the spoke pages visited on the branches below hub page *b* disappeared as soon as another spoke of *b* (page *h)* was selected. If a person wanted to go back to spoke page *d* from page *h* (perhaps because they needed to review the information on page *d*), they could no longer do it via stack-based Back as page *d* has been pruned.

Still, we could argue that the Back button isn't really required for this case, because the person can first use Back to go from *h*⇐*b*, and then use the normal links on *b* to re-navigate *b*→*c*→*d*. While reasonable for short pages with few links and simple navigational paths, this could be onerous for more complex situations. First, many web pages now override the coloring of previously selected links, which makes them indistinguishable from unvisited ones and thus harder to find. Second, some pages are long and complex: recalling and finding the correct link within the page adds the extra burden of scrolling and searching. Third, finding the correct spot to re-click on image maps may be challenging. Finally, if the person navigated a complex path to a particular page, they may find it difficult to retrace that path later on.

Stack-Back has another problem. Current systems do a poor job of communicating stack's tree-pruning behavior to its users [2], and most people actually believe that Back just returns sequentially to one's previously seen pages (this incorrect view is validated further in Section 4.1). This discord between how Back works and how people think it works is no surprise. The labels Back and Forward imply linearity, rather than of a tree. There are few cues at the interface to help users distinguish between the underlying semantics of page display using link selection (how new pages are added and how the stack is popped) *vs.* the semantics of moving within the stack using the Back and Forward buttons. Consequently, users sometimes wonder why pages are seemingly 'lost' when using Back.



a) User visits pages *a*-*d*, in order

b) User clicks Back twice,

c) … selects link to *h* which pops *c* & *d* off the stack…

d) …and pushes *h* onto it.

Figure 3. An example navigational trace and its effect on the stack. Note that pages *c* and *d* are popped off the stack.

2.2 Recency

Description. Perhaps the greatest disadvantage of stack-Back is that it cannot guarantee that previously visited pages are reachable by successive Back clicks. As an alternative, we could provide a complete history of all visited pages by having Back and Forward move a person through a recency-ordered list, where the buttons simply navigate through the pages in reverse order to how they were seen. Surprisingly, the design of recency-Back is not as simple as might be expected. Greenberg and Cockburn [4] explored several models of Back based on variants of a recency-ordered history list: here we describe only the final one that they advocate: recency with duplicates removed and a temporal ordering enhancement.

We begin with a side discussion: the management of duplicate entries in the history list. When a person sees a page more than once, the system would record them as duplicate entries on the recency list. The advantage is that successive Back clicks would go through a literal representation of the order of pages that the user has seen. The disadvantage of retaining duplicates is that the list (and thus the number of Back clicks) could become unnecessarily long and repetitious. Instead, Greenberg and Cockburn [4] suggest pruning duplicate pages by keeping only a single copy of it in its most recent position on the list: this keeps recently revisited pages near the top and thus quickly reachable through Back. Tauscher and Greenberg [6] analyzed this approach, and found that substantially fewer Back presses would be required to return to a desired page when duplicates are pruned.

The algorithm for maintaining a true temporal recency list with duplicates removed is described below.

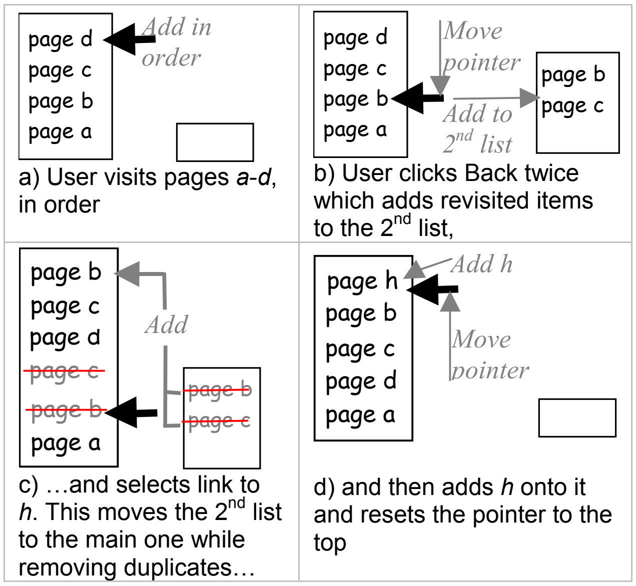
1. As with stack, clicking or typing links displays a page and adds it to the top of a recency list.
2. Similarly, clicking Back and Forward moves a pointer down and up the recency list respectively, where the pointed at page is rendered in the browser. However, the page seen before the button is clicked is also added to the top of a second list.
3. When the user is at any position on the list other than the top and selects a link on a web page, the contents of the second list are moved to the top of the recency list. Any duplicate entries below the current ones are then removed. Finally, the new page is added to the top of the list and rendered in the browser.

The function of the second list introduced in Step 2 is to track the order of pages seen as a person navigates the recency list using Back and Forward. We need this list in Step 3 for reordering the primary list to its true temporal order after a new link is selected. This scheme matches the sequence of page as the user saw them (excluding duplicates), and works over any number and combination of link selections and Back and Forward actions.

Figure 4 illustrates how this algorithm works using the same set of pages and the navigation example of Figures 2+3. As before, visits to the pages *a*→*b*→*c*→*d* produces the main list {*d,c,b,a*} (Figure 4a). Going from *d*⇐*c*⇐*b* creates a second list {b,c} (Figure 4b). As soon as the person selects the new link *b*→*h, b* and then *c* are added to the main list and any duplicates of *b* and *c* further down are removed (Figure 4c). Finally, the new page *h* is added, giving {*h,b,c,d,a*} which is the correct temporal sequence of pages (with duplicates removed) that the user has just seen (Figure 4d).

Advantages and disadvantages. Recency has several potential advantages. First, the list of previously visited pages is complete because no pages are popped off the list. Therefore users are guaranteed to be able to revisit pages already encountered during their browsing session by using the Back button. Second, because the underlying recency list can grow indefinitely, it is feasible for Back to work *between* sessions i.e., successive browser invocations and login sessions. Third, the temporal reordering algorithm means that users always see a temporally correct retracing of their page path using Back, which likely matches how people perceive Back to actually work [2].

Yet one disadvantage of recency-Back arises if a user's goal is to navigate back up the tree to a parent hub rather than to a previously seen spoke page. Superfluous spoke pages are now interposed as recency does not prune those spoke pages visited on a different branch.



a) User visits pages *a-d*, in order

b) User clicks Back twice which adds revisited items to the 2nd list,

c) …and selects link to *h*. This moves the 2nd list to the main one while removing duplicates…

d) and then adds *h* onto it and resets the pointer to the top

Figure 4 The same navigational trace using a recency list.

3 The Study

Is recency-Back a viable replacement for stack-Back? If it proves 'better', then browsers implementers should replace stack-Back with recency-Back. If it proves no better or worse, then implementers have the design option of using recency-Back if they can justify it e.g., as with the integrated revisitation system shown in Figure 1. If it proves worse, then we know that stack-Back is the preferred strategy.

To answer these questions, we designed a study that examined people's mental model of the conventional stack-based Back button. We then explored how well recency *vs.* stack-Back matched peoples' expectations of how it worked by asking people to predict what pages would appear as they navigated via Back. We also asked people which button they preferred. We had several expectations, framed as hypotheses.

Hypothesis 1. Users have a poor mental model of stack-Back. We expect this as it echoes an earlier result [2].

Hypothesis 2. When revisiting pages over different navigational paths, users are better predictors of what pages will appear when using recency-Back *vs.* stack-Back. We expect this because people believe Back navigates through the pages as they were seen [2].

Hypothesis 3. After using both a stack-based and a recency-based Back button for a similar set of tasks, people will prefer recency. We expect this because recency better fits peoples' reported mental models of Back [2].

3.1 Participants and expertise

Thirty volunteers participated in this study. All had some level of post-secondary education. Answers to a pre-test questionnaire indicated a mixed but generally web-savvy group. For web usage: 20 participants claimed to use a browser almost every day; 5 stated their use as ranging from once every few days to once a week; while the remaining 5 reported low Web use. Participants described themselves as: 4 being skilled experts, 13 having good (but not expert) skills, 7 having basic skills, and the remaining 6 having beginner-level skills. All participants stated that either Netscape Navigator or Microsoft Internet Explorer was their preferred browser.

3.2 Materials

Participants used Microsoft's Internet Explorer version 5.0 running within Windows 98 on a modern PC with a 1024x768 24-bit color display. All pages used for the study were stored on the local computer. In essence, this configuration meant that navigations and resulting page displays were uniformly rapid.

We added two non-standard software systems to the browser. First was a (visually identical) Back button that used the recency with duplicates removed/temporal ordering enhancement algorithm [4], as described in Section 2.2. Second was an artificial search bar used for one of the tasks: while it resembled a browser's typical search bar result, it actually contained a pre-defined set of static links.

We used several local web sites. We created these by importing/modifying a few popular commercial sites.

3.3 Method

The entire procedure listed below required approximately one hour of participant's time.

Stage 1: Initial mental model. Hypothesis 1 claims that people have a poor mental model of Back. Our strategy was to confront people with how they thought Back worked with how stack-Back actually worked. First, we asked them to articulate their mental model of the (stack-based) Back button they normally use (Figure 5, question 1). Second, we gave them a simple web site—a book table of contents containing links to several chapters—and had them navigate the hub and spoke pattern $a \rightarrow b \Leftarrow a \rightarrow c$. Participants did not see this arcane notation; they were told to go from the table of contents to Chapter 1, then Back to the contents, and then to Chapter 2. Third, we then asked participants how many times they would have to click the Back button from their current page *c* to go back to *b* i.e., from Chapter 2 Back to Chapter 1 (Figure 5, question 2). Fourth, we then told participants to try to return to *b* by actually using the standard stack-Back button. Finally, they were asked questions 3-5 in Figure 5 about how the observed stack-Back behavior matched the answer they previously gave in Question 1.

1. Describe how the back button works, and how the Back button internally manages and stores the pages you visit.

Participants were then asked to navigate through three pages comprising a simple hub and spoke system $a \rightarrow b \Leftarrow a \rightarrow c$.

2. How many times would you have to click Back to return to <page *b*>?

They were then told to try to return to a via Back.

3. Were there any problems?
4. Did this match your model of the back button in question 1?
5. Is the version of the Back button you are using is the same as the one supplied with your normal web browser?

Figure 5. Questionnaire excerpt concerning people's mental model of the Back Button.

Stage 2: Navigation and prediction tasks. Hypothesis 2 suggests that people can better predict what pages they will see when using recency-Back vs. stack-Back. To check this claim, we randomly assigned participants to one of two groups, where each group saw either the stack or recency-Back button first. We gave each participant five different and increasingly complex tasks. Instructions for all tasks are summarized below.

1. We reminded participants to think aloud as they worked.
2. From a home page, we had participants navigate through a series of links to a destination page. Participants had to scan each page they saw in order to find and choose the correct next link.

3a. We asked participants to return to a particular previously visited target page using only the Back button.

3b. Before each and every Back click in step 3a, participants had to predict what page they expected to see. They described the expected page, clicked Back, and then stated if their prediction was correct.

The five tasks stepped through different navigational sequences, corresponding to those illustrated in Figures 6–9 and described below.

Short linear sequence (Figure 6). The participant navigates from page *a* (the home page of the Discover Alberta web site we used) through two intermediate pages *b* and *c* to reach the destination page *d* (a page describing hostels in Edmonton) i.e., $a \rightarrow b \rightarrow c \rightarrow d$; this is illustrated by the straight arrows in the Figure. The participant then uses Back to return to page *a,* making predictions before each click. Correct predictions are denoted in Figure 6 as P1 to P3 (for predictions 1 to 3). Because there are no branches, both recency and stack behave identically i.e., *a* is returned to by $d \Leftarrow c \Leftarrow b \Leftarrow a$ (the curved arrows in the Figure).

Long linear sequence (Figure 7). This task is similar to the one above, except that more intermediary pages are involved. Reaching destination page *i* requires $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow f \rightarrow g \rightarrow h \rightarrow i$. Returning to *a* using both recency and stack Back is by $i \Leftarrow h \Leftarrow g \Leftarrow f \Leftarrow e \Leftarrow d \Leftarrow c \Leftarrow b \Leftarrow a$

Hub and spoke with return to hub (Figure 8). Reaching destination page *h* after visiting all the children of *d* requires $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \Leftarrow d \rightarrow f \Leftarrow d \rightarrow g \Leftarrow d \rightarrow h$. Returning to revisit target hub *a* using the stack Back just goes up the hierarchy by $h \Leftarrow d \Leftarrow c \Leftarrow b \Leftarrow a$ (4 Back clicks). Recency Back takes 7 clicks, as all *d*'s children are seen again $h \Leftarrow d \Leftarrow g \Leftarrow f \Leftarrow e \Leftarrow c \Leftarrow b \Leftarrow a$. These paths and corresponding predictions are denoted in the figure as PR1-7 for recency predictions, and PS1-4 for stack predictions.

Hub and spoke with return to spoke, then hub (Figure 8). This is almost identical to the task above (although using a different set of pages). The only difference is that participants are first asked to revisit the child spoke page *f* of hub *d,* and then the root page *a*. Note that spoke page *f* is not reachable via stack as it is pruned. The revisitation sequence of both recency and stack Back are identical to the hub and spoke task above.

Search bar (Figure 9). This complex task simulates a user navigating through several sites by using the results of a search presented in a search bar. The order of navigation by the participant (where *sb* denotes the search bar) is: $sb \rightarrow b \rightarrow c \Leftarrow b \rightarrow d$; $sb \rightarrow e$; $sb \rightarrow f \rightarrow g \rightarrow h \rightarrow i$; $sb \rightarrow j \rightarrow k$. The participant is then asked to return to target page *f*, and then to *c*. The path of stack Back is $k \Leftarrow j \Leftarrow i \Leftarrow h \Leftarrow g \Leftarrow f \Leftarrow e \Leftarrow d \Leftarrow b$ (8 Back clicks). Note that because stack Back pruned spoke page *c* when the participant went back to *b*, target *c* is not reachable. Recency Back takes 9 clicks, and includes target page *c* $k \Leftarrow j \Leftarrow i \Leftarrow h \Leftarrow g \Leftarrow f \Leftarrow e \Leftarrow d \Leftarrow b \Leftarrow c$.

Stage 3: Subjective preferences. Hypothesis 3 claims that people would prefer recency-Back. To check this, we had participants redo the entire set of tasks performed with the first type of Back button with the other type of button. Through a post-test questionnaire, we then asked them to comment on each Back button type and which they preferred.

4 Results

We first describe our participants' mental model of stack-Back. We then report both the prediction and preference data of only the first 15 participants for reasons that will become apparent in the subsequent discussion. Afterwards, we present the preferences of the remaining 15 participants.

4.1 Mental Model of the Stack-based Back button.

When asked to describe how their conventional stack-Back button worked (Figure 5 question 1), the description of all but two participants indicated an incorrect or incomplete mental model. Most participants simply said that Back just returns to all previously viewed pages. Some were more explicit (but still incorrect), where they said that pages are stored and displayed as a list of all pages in the order seen. A few other answers hinted that participants were aware of the stack-Back pruning behavior, but even then they had an incorrect view as to when and why this happened. To quote several participants, Back:

- goes to previous page, but sometimes you can't…I think it goes back to [a] different user;
- takes you back to previous pages in your navigation path…does seem to fail;
- takes you back to last few pages you visited but after a few clicks it takes you to the main pages (only).

We then confronted participants with how the stack-based Back button actually worked. As mentioned previously, after they navigated a simple hub and spoke pattern, they were asked to predict how many Back clicks were required to return from the second spoke to the first spoke. The correct answer is that it is not possible, as stack-Back will have pruned it off the list. However, only 2 of the 30 participants correctly answered this question (the same two that knew about the stack); the 28 others incorrectly predicted two Back button clicks.

We then asked people to try and navigate to that page via Back, only to find that they could not. Most, but not all, admitted that it did not match their mental model as stated when replying to Question 1 (Figure 5). A few said that in hindsight it did, but not for the correct reason. For example, one person said "there are often errors…it often doesn't work at all". Two others said that "some sites just do this". Also interesting is that about half of all participants thought that the Back button they just used did *not* behave the same as the one they normally used, even though it did!

What is clear is that people have a poor mental model of stack-Back i.e., Hypothesis 1 is supported. Most people were not able to articulate how stack-Back really worked. They could not predict its pruning behavior in even a very simple hub-and-spoke example. They were either surprised when the system did not work as predicted, or they had an arcane rationale for why it

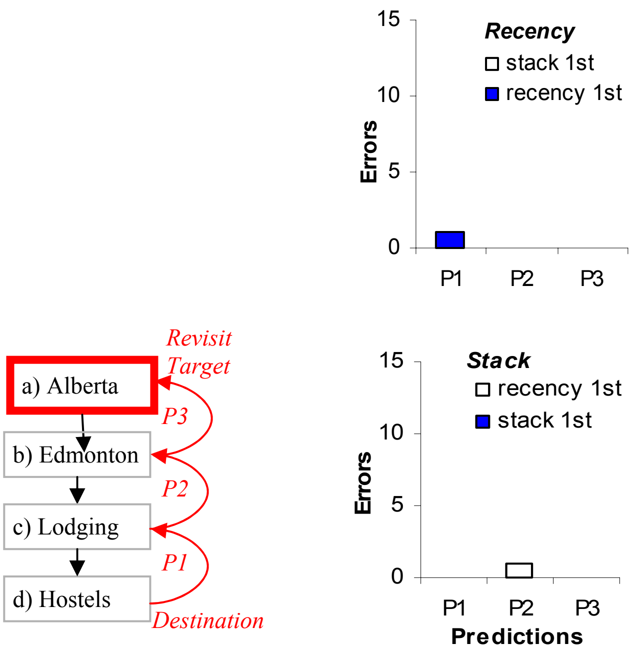


Figure 5: Short Linear

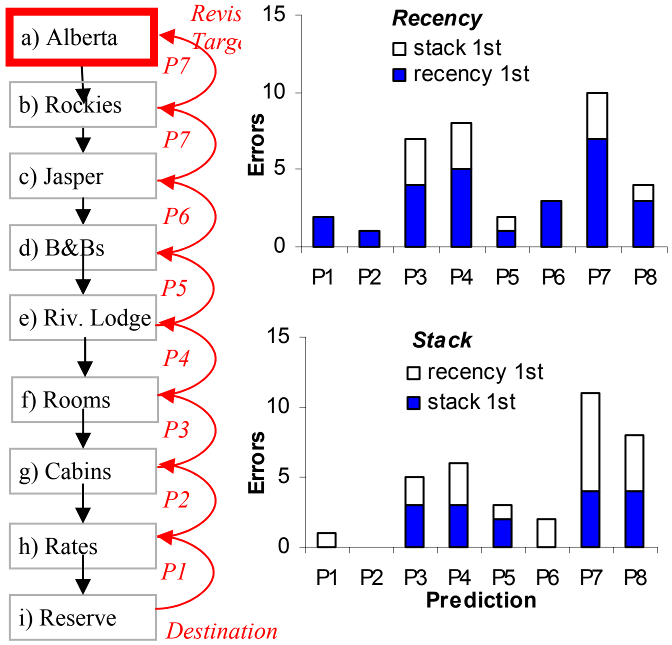
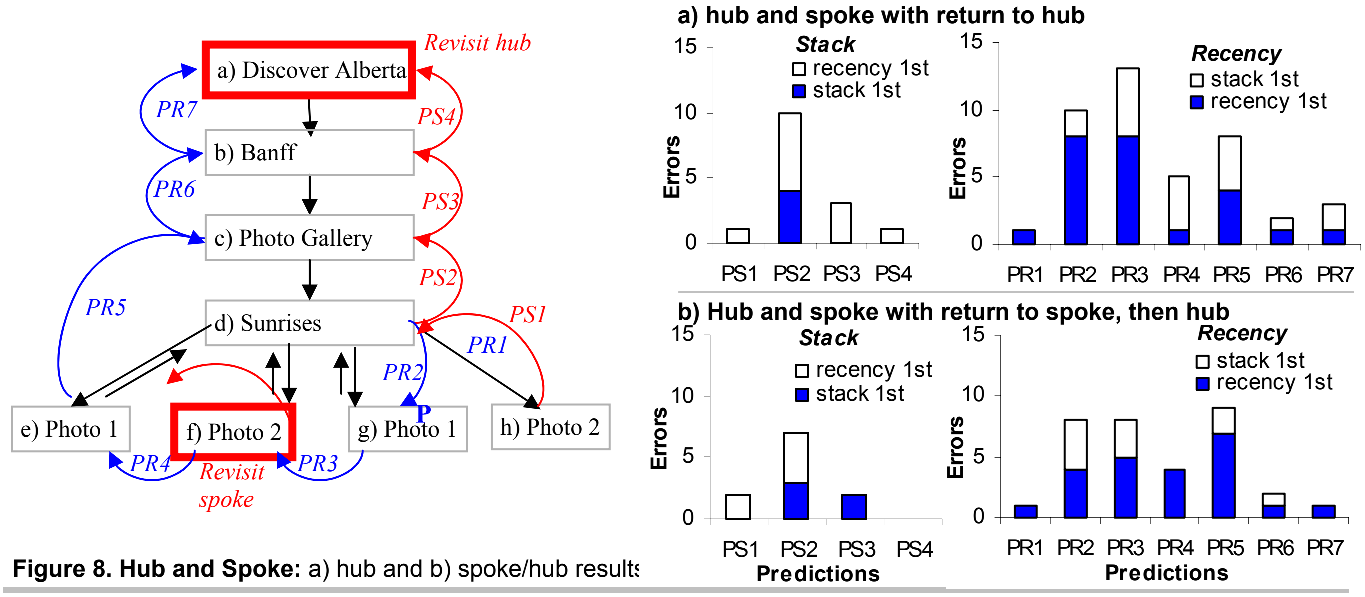


Figure 6. Long Linear



a) hub and spoke with return to hub

b) Hub and spoke with return to spoke, then hub

Figure 8. Hub and Spoke: a) hub and b) spoke/hub results

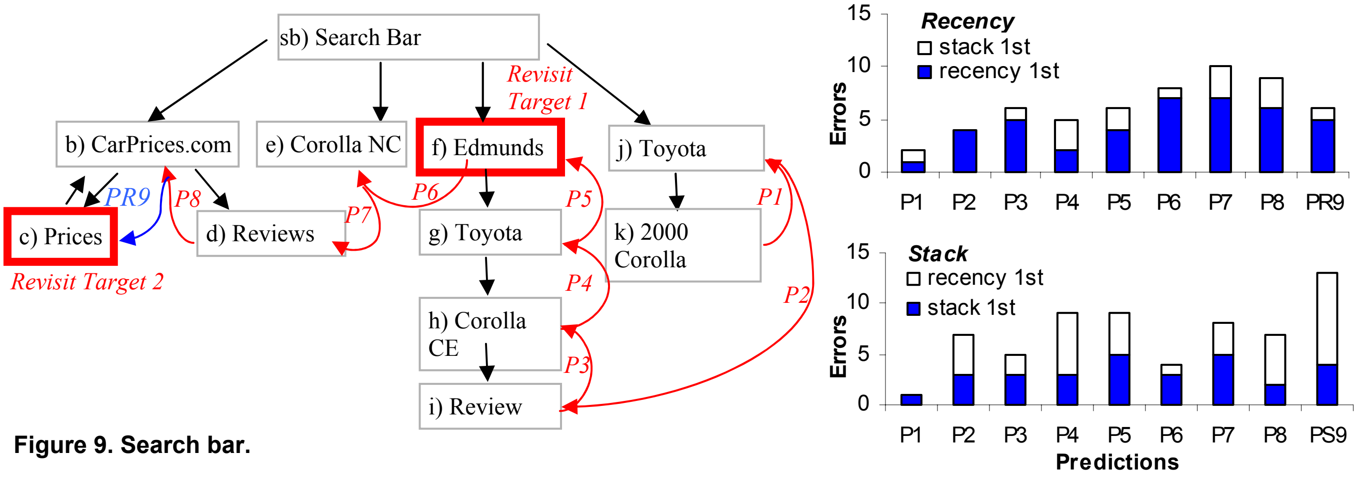


Figure 9. Search bar.

behaved the way it did. This result accords with Cockburn and Jones' [2] study involving ten computer scientists. As their study was done in the fairly early days of web browsers, we could have argued that today's web users are more browser-literate. This replication of their findings clearly demonstrates this is not the case: users find the exact behavior of the Back button as inscrutable now as it was then.

4.2 Predicting pages returned to by Back.

We asked the first 15 participants to predict what page they would see before they pressed Back as they tried to return to the revisit target. We will explain shortly why we did not ask the remaining 15 participants to do this prediction task. Eight participants began with recency-Back, while the remaining seven began with stack-Back. The results of these 15 are summarized below and plotted in Figures 5-8.

All graphs work as follows. First, graphs come in pairs, where one plots data from the stack-Back and the other from the recency-Back conditions. Second, the X-axis is the prediction number, which corresponds to those in the navigational sequence included in each figure. The Y-axis is the number of people who made errors on that prediction. Third, we divide the error data into two categories: if a graph plots a condition, we plot errors made by those who did that condition first as the bottom dark part of each bar. Those who used that condition second are on the upper light area. For example, the dark parts of the bars on a graph plotting a stack-Back condition indicate the errors made only by participants who did stack-Back first, while the light area are errors made by those who did the recency-Back first. This categorization lets us visually separate the data for learning effects. While included for illustration, our results will discuss only on the bottom dark area of each bar; we know there should be no learning effects in this data because these people did that condition first.

Short linear sequence. Each participant made three predictions P1, P2 and P3 for this sequence. The two graphs in Figure 6—the bottom one for the stack-Back and the top one for the recency-Back—plots where they made errors. Errors are rare; people are good at predicting short linear sequences in either condition.

Long linear sequence. Each participant made eight predictions P1 through P8 as they navigated back to the revisit target (Figure 7). As seen on the graphs, people made many errors in both the stack and recency-Back conditions after the first two predictions, although there are relatively fewer errors in the stack-Back condition.

Hub and spoke with return to hub. In this task, the target page was a hub up the hierarchy (the root page *a*). Four of the seven participants who went first using stack-Back erred only in their second prediction PS2 (Figure 8a, graphs at top). That is, there seemed to be some confusion as to where Back would take them after reaching the hub page *d.* In contrast, the eight participants using recency-Back made many prediction errors. As with stack, they were uncertain of what would happen after first reaching hub page *d* (PR2) and then again after seeing the second child (PR3)—most thought it would return to hub page *d* again. Only one person made an error on PR4, likely because they now realized they would see all children in order. However, many then expected to see hub page *d* again on PR5 rather than the hub's parent *c*.

Hub and spoke with return to spoke, then hub. The only difference between this and the previous task is that participants were asked to revisit the spoke page *f* before revisiting hub page *a.* We wanted to see if predictions were better or worse if they were looking for a child spoke page instead of a page up the hierarchical path. Comparing errors with the previous hub and spoke navigation with stack-based Back (where target page *f* is unreachable), we see two additional errors on PS3 (Figure 8b, graphs at bottom). We surmise that participants thought they would see the other spoke pages at these points. Recency-Back seems to have somewhat fewer errors going through the first few children (PS2 and PS3) when compared to the return to hub data, although the error rate is somewhat higher afterwards. As before, many expected to see hub page *d* after each spoke was revisited.

Search bar. The error rate for predictions on this complex revisitation task was very high for both conditions (Figure 9).

Preferences. After completing all tasks with one Back button type, participants repeated them with the other button. While the web site used was different, the navigational structure was identical. We then asked them which button they preferred. Nine favoring stack, four recency, and two were undecided. Comments by participants suggested that predictions were easier to make with the stack model as it went directly up the tree, thereby skipping sub-pages.

Think-aloud. During all tasks, we observed participants as they formed their predictions and thought-aloud about how they were making them. What was immediately obvious after running just a handful of participants was that predicting the next page often required a great deal of cognitive work as well as time. We saw participants try to mentally reconstruct where they had been: they would search the current page for clues as to what its parent could be while trying to recall what they had seen. They seemed to fare better on pages that had a logical hierarchical or path structure, and less so on pages whose structure was somewhat more arbitrary.

4.3 Discussion Part 1

On the surface (and without looking at statistical significance) Hypotheses 2 and 3 are rejected: stack-Back seems better than recency. Participants' error rate for predictions appears lower, and a majority of participants (9:4, 2 neutral) preferred stack to recency. This poor overall performance also supports Hypothesis 1 i.e., that people have a poor mental model of how stack-Back works. Surprising to us is that people's bad performance with recency hints that they have a poor mental model of how a recency-Back button works.

Yet something is wrong with this story. In the think-aloud observations for *both* conditions, we saw participants expend a great deal of time and effort when making predictions. That is, making predictions is hard work and does not accord with how we see people using Back in everyday use: navigating with Back is done without much apparent thought, and people backtrack through successive pages very quickly and successfully.

This discordance led us to rethink our rationale for Hypothesis 2. We initially thought that people could anticipate or predict from their mental model what specific pages would appear when clicking Back, and they somehow did this in their everyday use of browsers. From our observations, this is clearly incorrect. Instead, our findings suggest an alternate Hypothesis 2:

Hypothesis 2 (alternative). When revisiting pages over different navigational paths, users are poor predictors of what pages will appear when using either recency-Back or stack-Back. Instead, people use a 'click until recognize' strategy, where they have a vague expectation that the searched-for page will appear sometimes, and they simply click the button until they recognize the desired page.

This alternative Hypothesis 2 also raises doubts about our previous rejection of Hypothesis 3, for the prediction task does not reflect how people actually used Back. Forcing them to make predictions almost certainly interfered with their goal of returning to the target page. This may have led to some preference bias for stack-Back, which is likely easier to predict because one just has to reconstruct how one moves up the hierarchy.

Participants' difficulties for page prediction were so striking that we felt no need to statistically analyze our data further, or to have our remaining 15 participants suffer through the prediction task. Rather, we altered the study on the fly to re-evaluate Hypothesis 3 as described below.

4.4 Stack *vs* Recency-Back without Predictions

We re-examined Hypothesis 3 by seeing how participants would rate their preferences to the two Back buttons if they did not have to make predictions. We continued the study with the next fifteen participants exactly as before, except that we omitted step 3b in stage 2 of the method outlined in Section 3.3.

Unlike the previous 15 participants, we saw people quickly and effortlessly returned to the target page (if it was reachable) using both Back buttons. Where previous participants favored stack, this new set of participants showed no strong leaning for one button type over another. Eight preferred recency, six preferred stack, and one was undecided. People who preferred recency commented:

- go through the actual order more than not;
- pages come back sequentially as they should;
- more predictable: goes through the actual order;
- doesn't feel like more clicking;
- stack missed a whole bunch of pages;
- more intuitive… liked [that it had] no duplicates.

People who preferred stack commented:

- more used to it;
- recency produced extra clicks;
- doesn't take you back to sub-pages.

4.5 Discussion Part 2

Hypothesis 3 is still rejected, as people did not prefer recency over stack-Back. However, our new results suggest the following alternate hypothesis:

Hypothesis 3 (alternative). After using both a stack-based and a recency-based Back button for a similar set of tasks, people will not prefer one type of button over the other.

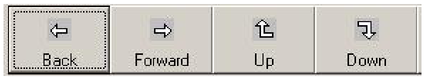
This change in preference as well as the ease which participants returned to target pages also re-enforces our conviction stated in the alternative Hypothesis 2. That is, people do not have an exact model of what pages they expect to see as they use Back, and that they use a 'click until recognize' strategy instead.

One point deserves further elaboration. A recurring comment made in regards to the recency model was the inability to recover 'hub' pages: participants expected to see the hub page after each visit to a child. Our recency with duplicates removed algorithm showed the hub page once, but the perception of the users was that it did not. Instead, they expected a pure sequential model. Does this suggest that hub pages should be duplicated rather than only shown once? We think not. With more extended use of recency Back, users may realize that the hub pages are accessible and may find the duplication of pages unnecessary. We also believe that seeing hub pages several times will introduce a different type of confusion i.e. that Back is merely cycling through the same sequence of pages. Still, more testing is needed before drawing a final recommendation of how duplicates are handled.

5 Implications to Browser Designers

At first glance, there is no compelling reason to change the current stack-based Back button to a recency-based one. People seem comfortable with stack-Back, even though they have a poor model of it. This is because their 'click until recognize' strategy does not require an accurate model of its behavior. As well, people seem somewhat unconcerned about the mysterious disappearance of pruned pages, blaming it on the vagrancies of computers. Importantly, there is no overwhelming preference by our participants of recency-Back over stack. While we could still argue that recency is better than stack because no pages are lost, we cannot make a compelling argument that the familiar stack-Back idiom should be replaced in conventional browsers.

However, the ambivalence between recency *vs.* stack means that new designs *can* include recency with no penalty. One possibility we prototyped includes both the stack and recency-based buttons. We relabel stack-Back and Forward as Up and Down, as this more accurately reflects the semantics of moving up and down the navigational hierarchy that is a side product of stack-pruning. Back and Forward are now recency-based, as they reflect the semantics of moving backwards and forward on the recency-ordered history list. This is not new, as similar buttons are found on several non-web browser products e.g., Microsoft's file explorer, and the MSDN document browser. Nonetheless, we recommend caution. Because users have a fuzzy notion of how stack and recency behave, the differences between these buttons may be unclear to them. As well, it adds complexity: yet another decision must be made as to which revisitation method should be chosen.



Perhaps a more compelling reason for using a recency-based Back button is to remove the differences between Back and the other revisitation systems available on web browsers. As previously described and as illustrated in Figure 1, our new revisitation system integrates Back, history, and bookmarks by unifying them to operate over a single recency-based list [4]. Back and Forward simply become shortcuts for navigating the history / bookmark list item by item. If the history list is visible, then items are highlighted as the user selects Back e.g., we see in Figure 1 that the 2nd item is marked, which means the person has just pressed Back once. This visually exposes and re-enforces how Back works. Our study suggests that this replacement of stack-Back with recency-Back can be done with no penalty.

6 Summary

Even though Back is probably the most highly-used interface widget in existence today, there are (to our knowledge) no other published studies that scrutinize alternatives to its widely-deployed stack algorithm. In this paper, we studied an alternative Back algorithm using recency with duplicates removed.

While we began the experiment with particular expectations (framed as hypotheses), some of them proved incorrect. From our results, we now claim that people have a poor model of both stack and recency-Back. We also claim that in everyday use, people do not mentally predict what pages will appear as they click Back. Rather, we suggest that people employ a very simple 'click until recognize' strategy, where they simply click Back until they recognize the desired page. We also claim that people have no strong preference of recency or stack.

For browser designers, we advocate the replacement of stack-based Back with recency only if other design considerations warrant them. We feel that good design opportunities do exist, especially for a recency-based Back to be integrated with a recency-based history list to produce a single model of how pages can be revisited.

There is no question that the high usage rate of Back warrants further research: millions will be affected by even a small improvement in its design.

Acknowledgements. Microsoft and NSERC funded us. Kent Sullivan, Robert Graf, Linda Tauscher and Andy Cockburn contributed intellectually.

References

[1] Cockburn, A. & McKenzie, B. What do web users do? An empirical analysis of web use. *Int J Human Computer Studies* **54**, 903-922, 2001.

[2] Cockburn, A. & Jones, S. Which way now? Analysing and easing inadequacies in WWW

navigation. *Int J Human-Computer Studies* **45**(1), 105-129, 1996.

[3] Cockburn, A. & Jones, S. Design issues for World Wide Web navigation visualisation tools. *Proc RIAO'97: The 5th Conference on Computer-Assisted Research of Information*. McGill University, Canada, 55-74, 1997.

[4] Greenberg, S. & Cockburn, A. Getting back to Back: Alternate behaviors for a web browser's Back button. *Proc 5th Annual Human Factors and the Web Conference*, NIST, Gaithersburg, USA, 1999.

[5] Kaasten, S. and Greenberg, S. (2001) Integrating Back, History and Bookmarks in Web Browsers. *Extended Abstracts of ACM CHI'01*, 379-380.

[6] Tauscher, L. & Greenberg, S. How people revisit web pages: Empirical findings and implications for the design of history systems. *Int J Human Computer Studies*, **47**(1), 97-138, 1997.