A Computational Model of Web Navigation

Craig S. Miller
School of Computer Science
DePaul University
243 S. Wabash Ave.
Chicago, IL 60604
+1 312 362 5085
cmiller@cs.depaul.edu

Roger W. Remington
NASA Ames Research Center
Moffett Field, CA 94035
+1 650 604 6243
rremington@mail.arc.nasa.gov

ABSTRACT
We describe a computational model of Web navigation. This model simulates a user searching a Web site for a specified target item located under one of the site's terminal links. The model's simple, yet plausible, implementation produces results that are consistent with published empirical studies. In particular, under certain conditions, it produces longer search times and a higher failure rate for a three-level site than for comparable two-level sites. Despite its simplicity, it demonstrates complex interactions between site depth and the quality of Web link labels and predicts that, as the quality of link labels diminish, the advantage for flatter Web structures increases.

Keywords
Cognitive model, Web navigation.

INTRODUCTION
The World Wide Web is an integral part of our culture. As a result, Web page usability has become an issue affecting a large number of people, often with significant money at stake. There seems to be no shortage of advice on how to design effective, usable Web pages [2, 6]. Perhaps much of this advice is sound and useful, particularly if it is the result of extensive experience and observation. Yet, it can be difficult to independently evaluate the validity of such advice when its reasons are often not articulated and the supporting evidence is hidden from scientific scrutiny. For example, Spool et al. [6] base their advice on test problems given to a sample of users. However, they do not report important methodological details and it is not certain how their results would generalize to different problems.

Lately, there has been an effort to establish design guidelines based on systematic studies. For example, Byrne, et al. [1] developed a taxonomy of Web actions based on observed user behavior from which they were able to compute the frequency of different classes of behavior. Larson & Czerwinski [4] examined user search times in Web pages of differing hierarchical depth. They found that flat structures produced faster search time than deep hierarchical structures. This is consistent with results from studies of menu depth [3].

While such empirical studies are essential, it can be difficult to generalize from these studies to situations other than those tested. In addition, empirical testing is too expensive and time consuming to address the wide range of content, configurations, and user strategies that characterize the Web. For example, the content of some sites may be extremely difficult to break into clear unambiguous categories, whereas the content of other sites may suggest obvious easy-to-understand categories, perhaps even at multiple levels of abstraction. The same design advice on link structure may not be equally applicable to these diverse sites. Perhaps a flat broad structure is appropriate for the first, a deep narrow structure for the second. Such problems of generalization make it imperative to augment empirical testing with other approaches.

The ability to generalize empirical results comes from an understanding of the underlying process that produced the results. One approach is to develop a cognitive model that simulates the user interacting with the computer interface. By constructing the model so that it is functionally sufficient and uses plausible cognitive mechanisms, we obtain a working, explicit hypothesis of user interaction which can be further tested by comparing its behavior to known empirical results. For example, Peck & John [5] used a computational model to highlight patterns of interactions with a browser.

In this paper, we describe a computational model of a task similar to Peck & John. We simulate a user searching a Web site for a specified target item located under one of the terminal links. Our goal was to test the effectiveness of flat versus hierarchical arrangement of information under conditions in which the link labels differ in their semantic relatedness to the target information. Depth of hierarchy has been shown to affect search times in menus [3] and Web pages [4]. Our factor of relatedness is similar to Scent for Information [4], which is widely held to influence search times. The interaction of these two factors has not before been investigated. Our user model implements a simple, yet plausible, strategy using a set of cognitive resources that approximate those of a human user. We

show how a simple model combined with simulation allows us to investigate the effects of factors that pertain to a wide range of Web site organizations.

REPRESENTING A WEB SITE

Our model interacts with a simplified, abstract representation of a Web browser and a Web site. This abstraction gives us control over the domain's variables and allows us to qualify how our simulation results generalize to a larger class of Web sites. Following the empirical study of Larson and Czerwinski, we restrict our Web sites to structures that are balanced trees. Each site has one root node (i.e. the top page), consisting of a list of labeled links. Each of these links leads to a separate child page. For a shallow, one-level site, these child pages are terminal nodes, one of which contains the target information that the user is seeking. For deeper, multi-level sites, a child page may consist of a list of links, each leading to child pages at the next level. The bottom level of all our sites consists exclusively of terminal pages, one of which is the target page.

When navigating through a site, a user must perceive link labels and gauge their relevance to the sought-after information. While the evaluation is a complex and interesting process in itself, we choose not to model the process per se. Rather, our interest involves the consequences of different levels of perceived relevancy. As a proxy for each link label, we fix a number, which represents the user's immediately perceived likelihood that the target will be found by pursuing this link.

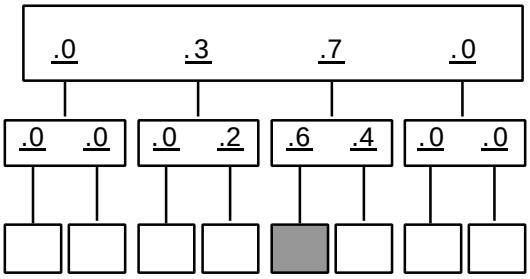

Figure 1

Figure 1 shows an example of a simple two-level Web site. The rectangles represent Web pages. the underlined numbers represent links to child and parent pages. The numerical values associated with a link represent its perceived relevance. The top page in Figure 1 contains four links labeled with numerical likelihood factors of .0, .3, .7 and .0. Like the concept of Scent, these numbers represent the user's belief that the path associated with a given link contains the target information. Each link leads to a child page containing two links. For example, the child page behind the third top-level link (labeled with .7) has two labeled links where the user believes that the first one has the greater likelihood (.6) than the second one (.4). There are eight terminal pages, one of which (filled rectangle) contains the target information. In our terminology, this example site has a 4x2 architecture, where 4 is the number of links at the top-level and 2 is the number of links on each child page.

For the above example, a user strategy that merely follows the most likely links would directly lead to the target. Our method also provides the flexibility for representing sites and user knowledge where the user is partially mistaken (or misled) as to where the target is. For example, Figure 2 shows the same site with a different user for whom the meaning of the labels differ from the user in Figure 1. This user would find the target under what he or she perceives as a less plausible sequence of link selections. In this way it is possible to represent sites that differ widely in strength of association between link label and target information -- from virtual certainty (links of 1.0) to complete uncertainty.

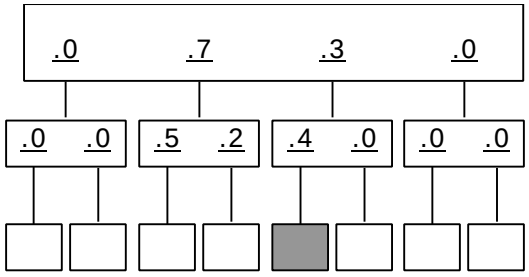

Figure 2

MODELING THE WEB BROWSER AND USER ACTIONS

Byrne, et al. [1] found that selecting a link and pressing the Back button accounted for over 80% of the actions used for going to a new page. Consequently, we focused our modeling on the component actions underlying these behaviors. These include:

- Selecting a link
- Pressing the Back Button
- Attending to and identifying a new page
- Checking a link and evaluating its likelihood

All four are primitive actions that our model performs serially. Fixed times are assigned to each action to account for their duration during a simulation. Our model also simulates changing the color of a link when it is selected so that the modeled user can "perceive" whether the page under this link was previously visited.

A simple strategy

These four primitive actions can be combined to create a simple yet plausible model of a user navigating a Web site attempting to find target information located on one page. With the appearance of a new page, the model first attends to the page, which, if it is a terminal page, includes checking if it contains the target information. If it does not,

the model scans the links on a page selecting in turn those links whose likelihood is equal to or above a fixed threshold. For now, we choose a value of 0.5 with the interpretation that selecting a link of equal or greater value will likely lead to success. When a page appears by selecting a link, the process of checking and scanning the page is repeated.

Once the model detects no unselected links above the threshold value, it returns to the parent page by pressing the Back button. When a previous page reappears by pressing the Back button, the model continues scanning links starting at the last selected link. It does not scan links it has already evaluated. Determining the last link selected places no demands on memory since the last selected link is easily detected by its color, and many browsers return the user to the location of the last selected link.

If the model backs up to the root page and finds no remaining unselected pages above the threshold, it then reduces the likelihood threshold to a second value (currently fixed at .1) and starts exploring unselected links. In avoiding previously explored pages, the model only scans links that were not previously selected. Below is a more formal specification of what this "threshold" model does upon the appearance of a page in the browser. The items with an asterisk incur a time cost that adds to the cumulative simulated time.

```
attend to new page*
check for target
if target
   terminate (successful)
else
   for each unvisited link (starting at last selected link)
      determine if link is above threshold (relevant) *
      if link is relevant
         select link * and reset strategy

if no links were selected on this page
   if at root page
      if threshold has already been reduced
         terminate (unsuccessful) task
      else
         reduce threshold to secondary value (i.e. .1)
   else
      press Back button* and reset strategy
```

To illustrate, let us consider how the model would search the pages given the likelihood factors in Figure 2. The model would first attend to the page then scan the topmost links from left to right. The first (.0) is below threshold (.5), but the second (.7) is above, and is selected. When the new page appears the model first determines that it is not the target. It then scans the links on this page, again from left to right. The first link (.5) is evaluated and selected. Since this page neither contains the target nor has any links, the Back button is pressed. The previous page appears and the model continues scanning the next link (.2). Since it is below threshold and there are no more links, it presses Back again. Once the top level is presented it continues scanning the remaining two links (.3 and .0). They are both below threshold with no remaining unexamined links. Since this is the root level, the threshold is reset to .1 and the remaining unexamined links reevaluated. The first (.0) fails, the second (.7) has already been examined and is skipped, but the third (.3) is above the new, lower threshold. It is now selected. With the lower threshold, the first link (.4) is selected and the target found. The detailed trace of this example is specified below. The notes in parentheses are for explanatory purposes and do not incur any time cost.

```
Attend to page
Link (.0) is evaluated, below threshold
Link (.7) is evaluated, above threshold
Link (.7) is selected
(new page appears)
Attend to page
Link (.5) is evaluated, above threshold
Link (.5) is selected
Attend to page -- no target present
Press Back Button
Attend to page
Link (.2) is evaluated, below threshold
Press Back Button (no remaining unexamined links)
Attend to page
Link (.3) is evaluated, below threshold
Link (.0) is evaluated, below threshold
(threshold is reduced to .1)
Attend to page
Link (.0) [first link] is evaluated, below threshold
(link (.7) is skipped; it is already examined)
Link (.3) is evaluated, above threshold (.1)
Link (.3) is selected
(new page appears)
Attend to page
Link (.5) is evaluated, above threshold
Link (.5) is selected
(new page appears)
```

Attend to page -- target is present

We can calculate a simulated search time by assuming plausible time values for attending to a page, evaluating a link, selecting a link, and for pushing the Back button. The above trace shows that attending to a new page occurred 8 times, evaluating a link occurred 9 times, selecting a link 4 times, and pushing the Back button 2 times. For now, we assume that evaluating a link requires a half second and the remaining events require one second. For this example, the total simulated time is 18.5 seconds (8 + .5 * 9 + 4 + 2). Interestingly, had the target been in the terminal Link (.2), under the top-level Link (.7) the strategy would fail to find it. The Link (.2) was examined under the high threshold (.5). When the threshold is later reduced, this link is "hidden" because its parent Link (.7) had been previously selected. We will later see that this situation accounts for a significant portion of search failures as it demonstrates a pitfall of deep-structured architectures.

While the actual practice of Web navigation is certainly varied and complex, the threshold strategy has at least three general qualities that make it a reasonable approximation of actual Web navigation. First, it successfully finds the target whenever it lies behind likely links. Second, by skipping less likely links, it finds the target quickly, again provided that the target is behind likely links. Finally, it places few cognitive demands on the user. Instead of using memory to store previously explored links and pages it uses the browser's record of visited links to indicate what has been explored.

Exploring the effects of site architecture and label ambiguity

Larson and Czerwinski found that human users were faster and more accurate in finding target information with flatter structures, 16x32 and 32x16, than with deeper hierarchies, 8x8x8. There was no significant difference between the performances on the 16x32 and 32x16 architectures. Our initial goal was to test whether our threshold strategy coupled with the user model could replicate the Larson and Czerwinski results using plausible parameters. For our simulations, we created representations of their three architectures: 8x8x8, 16x32, and 32x16. We also created a fourth architecture that had a flat structure of 512 links. For each generated site representation, we randomly placed the target at one of the site's terminal pages.

To assign likelihood factors to the links, we first assigned a one to links that led to the target and a zero to those that did not. We then perturbed these values with noise according to the formula below. For each link, this algorithm was repeatedly invoked until it generated a number in the range from zero through one.

$$g * n + v$$

where g is a number chosen randomly from a standard normal distribution (mean=0, stdev=1); n is the noise factor multiplier (equivalent to increasing the variance of the normal distribution); and v is the original likelihood value (0 or 1).

Figure 3 shows a 4x2 architecture n = .3 that was used in our simulation. This procedure also generated atypical representations in which the target was not associated with the highest valued link.

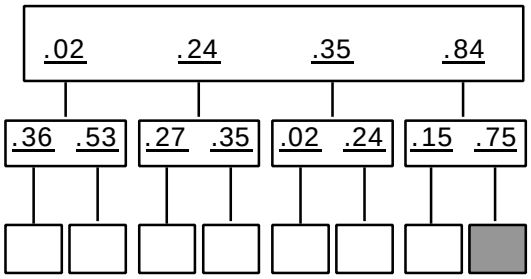

Figure 3

Our simulations paired each of the four architectures (8x8x8, 32x16, 16x32, and a flat 512) with a range of noise parameters (0.0 through 0.5). For each pairing, 10,000 times trials were run, each of which used a newly generated site representation.

As in the example trace, computations of search time assumed that evaluating a link required a half second and the remaining events (selecting a link, pressing the back button, and attending to a new page) required one second. We chose these times since they lie within a range of plausible values. For example, very short link labels could each be examined more quickly than a half second whereas longer labels may require as much as several seconds. Likewise, the duration of one second for browser events assumes a reasonably responsive browser and internet connection. Certainly these times could be slower. While we will not further explore the consequences of varying time costs here, future experimentation would show what interactions do occur, if any. Following the Larson and Czerwinski study, any trial lasting longer than 5 minutes (300 seconds) of simulated time was terminated and labeled as a failed search.

SIMULATION RESULTS

So that we may compare our results with those reported by Larson and Czerwinski, we encoded failed attempts as taking 300 seconds (5 minutes) when calculating mean times. All mean times had a 95 percent confidence interval that is plus or minus 2 seconds or smaller. Because the performances on the 32x16 and 16x32 architectures were nearly identical, we only display the 16x32 here. Figure 4 shows the calculated mean times from the simulation conditions. The most salient trend is that link noise has a profound detrimental effect on search time. The flat 512 architecture clearly produced the slowest times. The deep 8x8x8 architecture and the broader 16x32 architecture produced nearly identical search times. However, an analysis of the failure percentages, shown in Figure 5,

indicates that for the noisier configurations, the 8x8x8 architecture had a significantly higher failure rate than the 16x32. Histograms plotting the time distribution at a noise factor of 0.3 further confirm the performance differences between all three architectures. The flat architecture produces the most uniform distribution whereas the deepest architecture (8x8x8) produces a bimodal distribution with the two modes lying at the extremes. The 16x32 architecture produces less extreme modes, both in terms of placement and magnitude.

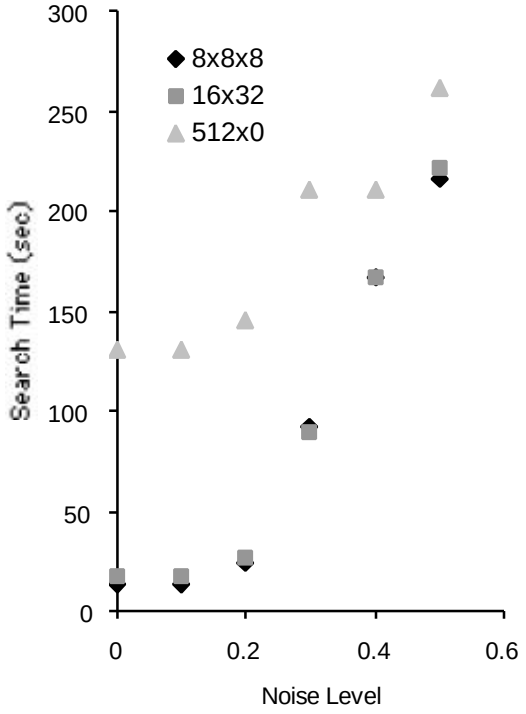

Figure 4

Like Larson and Czerwinski we found that the 16x32 and 32x16 architectures produced nearly identical mean times and failure rates. For sufficiently noisy sites, the 16x32 and the 32x16 architectures yielded a lower failure rate than the 8x8x8 architecture. However, whereas Larson and Czerwinski reported slower mean times for the 8x8x8, our simulations found no differences in mean times between the 8x8x8 architecture and the two two-level architectures.

In considering this last discrepancy between the mean times of the model and of the actual users, we discovered that the reported mean times led to a misleading comparison. Because the 8x8x8 had a higher failure rate, we determined that the 5 minute cut-off strongly favored the 8x8x8 architecture and thus artificially lowered its mean time. Also, at a middle level of noise (i.e. 0.3), our simulated times were more than twice as slow as the results reported by Larson and Czerwinski. To address the former and to compensate for the latter, we halved the costs of all simulated actions and ran the simulations again. This time, we found that for sufficiently noisy sites (0.3 and larger), the 2-level architectures produced faster mean times than the 8x8x8 architecture. For example, at noise level of 0.3, the 16x32 architecture produced a mean time of 65.9 seconds and the 8x8x8 architecture produced a mean time of 75.4 seconds. In general, for this last round of simulations, our model fully replicated the qualitative aspects of the Larson and Czerwinksi results at noise factors of 0.3 and greater.

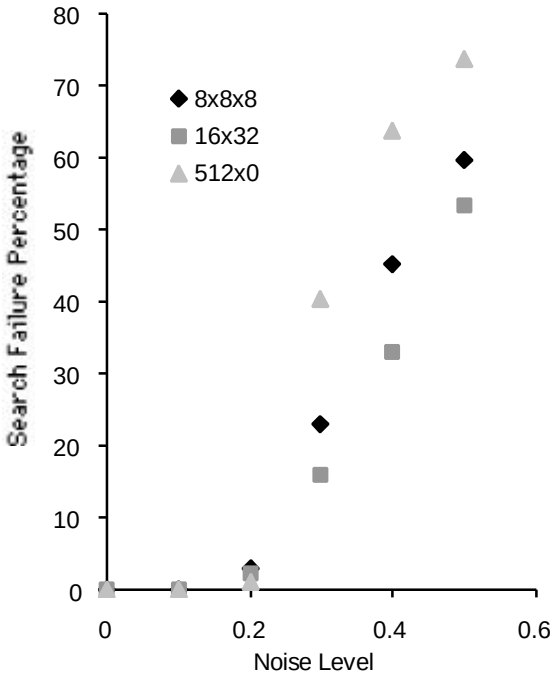

Figure 5

DISCUSSION

We have shown here that a simple model of Web page navigation can account for the search times observed in empirical studies. Our simulations support the empirical observation that a deep hierarchy produces slower search times. Our results go further in suggesting that the perceived relevance between the link label and the target information played the key role. We used gaussian noise to vary label relevance, where high noise is associated with ambiguous relevance. As noise increased the advantage for flatter structures increased. Moreover, we noted qualitative aspects of performance that were revealing. Deep structures (8x8x8) produced a bimodal distribution. There were extremely fast times at the cost of increased failures. These results obtain largely because when the model readjusts its threshold at the top level it does not explore the children of

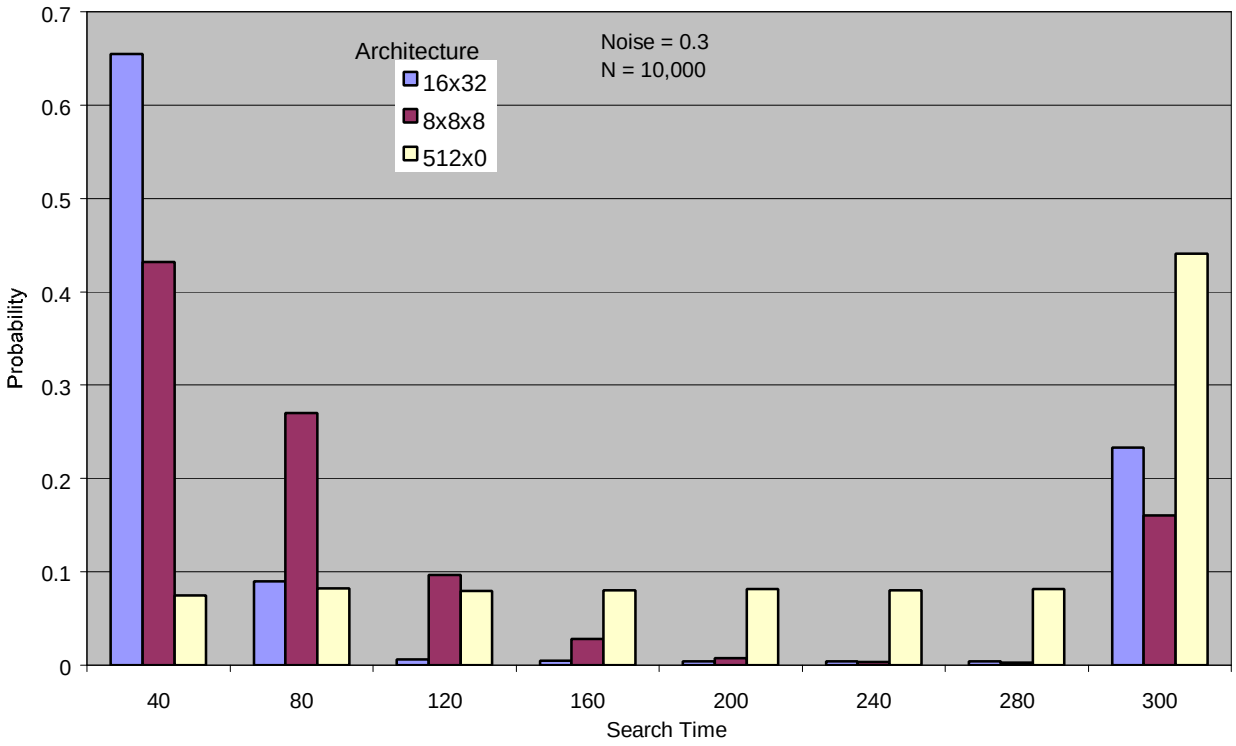

Figure 6

links it has already examined (at the higher threshold). It remains to be determined whether subjects show the same behavior pattern.

Our simulations revealed a complicated pattern of interactions between the factors underlying search in Web pages. This suggests that general design principles may be too simplistic to be applied across the board. Instead, we argue that continued model-based simulation should be focused at more complex architectures that better characterize the link structures in Web pages that more accurately capture Web interaction. These and future simulations should also serve in directing further user studies.

REFERENCES

1. Byrne, M.D., John, B.E., Wehrle, N.S., & Crow, D.C. (1999). The tangled web we wove: A taskonomy of WWW use. In Proceedings of CHI'99 Human Factors in Computing Systems, ACM press.
2. Forsythe, C., Grose, E., & Ratner, J. (1998). Human Factors and Web Developmentt. London: Lawrence Erlbaum Associates.
3. Landauer, T. K., & Nachbar, D. W. (1985). Selection from alphabetic and numeric menu trees using a touch screen: Breadth, depth, and width. In Proceedings of CHI'85 Human Factors in Computing Systems, ACM press, pp. 73-78.
4. Larson, K. & Czerwinski, M. (1998). Web page design: Implications of memory, structure, and scent for information retrieval. In Proceedings of CHI'98 Human Factors in Computing Systems, ACM press.
5. Peck. V. A. & John, B. E. (1992). Browser Soar: A computational model of a highly interactive task. In Proceedings of CHI'92 Human Factors in Computing Systems, ACM press, pp. 165-172.
6. Spool, J. M., Scanlon, T., Schroeder, W., Snyder, C. & DeAngelo, T. (1999). Web Site Usability: A Designer's Guide. San Francisco: Morgan Kaufmann, Inc.