Combating Information Overload in Non-Visual Web Access Using Context

Jalal Mahmud, *Yevgen Borodin*, and *I.V. Ramakrishnan*
Department of Computer Science, Stony Brook University
Stony Brook, NY 11794, USA
{jmahmud, borodin, ram}@cs.sunysb.edu

Dipanjan Das
School of Computer Science
Carnegie Mellon University
dipanjan@cs.cmu.edu

ABSTRACT

Web sites are designed for graphical mode of interaction. Sighted users can visually segment Web pages and quickly identify relevant information. In contrast, visually-disabled individuals have to use screen-readers to browse the Web. Screen-readers process pages sequentially and read through everything, making Web browsing time-consuming and strenuous. The use of shortcut keys and searching offers some improvements, but the problem still remains. In this paper, we address this problem using the notion of *context*. When a user follows a link, we capture the context of the link, and use it to identify relevant information on the next page. The content of this page is rearranged, so that the relevant information is read out first. We conducted a series of experiments to compare the performance of our prototype system with the state-of-the-art screen-reader, JAWS. Our results show that the use of context can potentially save browsing time as well as improve browsing experience of blind people.

ACM Classification: H5.2[Information Interfaces and Presentation]:User Interfaces. - Natural language, Voice I/O.

General terms: Algorithms, Design, Human Factors, Experimentation.

Keywords: Web navigation, context, screen-reader, CSurf, voice browsing, user interface, information rearrangement.

INTRODUCTION

The Web has become an indispensable source of information. The primary mode of interaction with the Web is via graphical browsers, which are designed for visual interaction. As we browse the Web, we have to filter through a lot of irrelevant data (e.g. banners, commercials, navigation bars). Sighted individuals can quickly segment any Web page and identify the information that is most relevant to them. The task becomes complicated for individuals with visual disabilities. Blind people and people with low vision use screen-readers, such as JAWS and IBM's Home Page Reader [6, 1], to browse the Web. However, screen-readers process Web pages sequentially, and provide little or no content filtering, resulting in *information overload*. To address the problem of information overload in non-visual Web browsing, screen-readers often permit to skip blocks of text in the order they appear on the page. Unfortunately, in many cases, users still have to listen or skip through a substantial part of page content before they get to the information. To help users locate the information quicker, a number of screen-readers allow keyword searching. However, simple searching has two problems: it works only for exact string matching and it disorients users in case of a wrong match. In both cases users have to start from the beginning of the page. The problem of *information overload* in non-visual Web access still remains. Is it possible to do better than that?

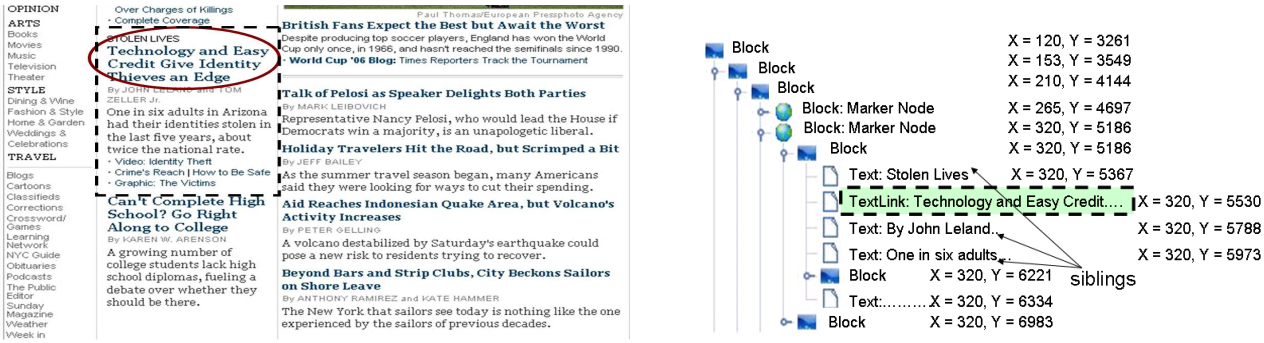
The identification of relevant information on any distinct Web page is subjective. However, as soon as the user follows a link, it is often possible to use the text around it to determine the relevant information on the next page. In this paper we describe a novel approach for combating information overload in non-visual Web access using *context*. We explore the structural and visual organization of Web pages to identify the context. For example, when the user follows the headline news link, encircled in Figure 1(a), the news article in Figure 3(a) should be read out first. To identify the article as most "relevant", all the words contained in the dotted box in Figure 1(a) are selected as *context*. Then, the content of the following page is rearranged to present the article first. The rest of the Web page is read sequentially as before. This approach helps blind and low vision users quickly identify relevant information while surfing the Web, thus, considerably reducing their browsing time.

SYSTEM ARCHITECTURE

The architecture of CSurf, our context-based browsing prototype system, is shown in Figure 2. Users communicate with the system through the **Interface Manager**. The module uses VoiceXML dialogs to interact with the users and present Web page content. The interface allows keyboard and voice input via our own VoiceXML interpreter [2], and provides both basic and extended screen-reader navigation features, such as shortcuts and corresponding voice commands.

Context Analyzer is called twice for each Web page access. When the user follows a link, the module collects the context from the current page. When a new Web page is retrieved, the module executes our algorithm to contextualize the page before it is presented to the user. The **Browser Object** mod-

IUI'07, January 28–31, 2007, Honolulu, Hawaii, USA..



(a) Source Page

(b) Source Frame Tree

Figure 1: Context Identification and Ranking

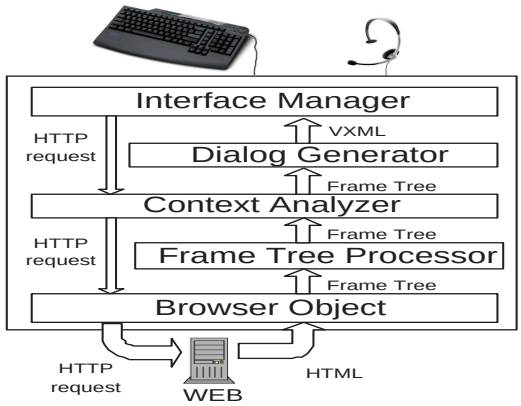


Figure 2: Architecture of CSurf

ule downloads Web content every time the user requests a new page to be retrieved. The module, built on top of the Mozilla Web Browser, is coupled with extended JREX Java API wrapper [7]. **Frame Tree Processor** extracts the *Frame Tree* representation of the Web page. A $Frame\ Tree$, constructed by the Mozilla Engine, is a tree-like data structure that contains Web page content and formatting, specifying how a Web page has to be rendered. This module cleans and reorganizes the frame tree. Subsequently, Context Analyzer reorders the frame tree before passing it to the Dialog Generator. The **Dialog Generator** module uses a collection of dialog templates to convert the frame tree into a Voice-XML dialog. The latter is then delivered to the Interface Manager. Architectural details of dialog generator appears in [10].

CONTEXT ANALYSIS

In this section we formally define the notion of context and describe the phases of the context analysis algorithm.

Context: Given a frame tree of a Web page and a link with its corresponding tree node n_i, *context* is defined as a multiset that includes the text of the link node, along with the text contained in the m sibling nodes $\{n_1, ..., n_{i-1}, n_{i+1}, ..., n_m\}$ of n_i.

Consider an example when the source is the front page of *The New York Times* news Web site, Figure 1(a). The context of the encircled link is the text surrounded by the dotted line. Figure 1(b) shows the corresponding frame tree with the link and its siblings.

1. Context Identification and Ranking: From the frame tree, we extract the text in and around the link and store it in a multiset after removing all function words[1]. We denote this multiset as $S_{context}$. The words in $S_{context}$ are then ranked (weighted) according to their proximity to the link. At this point, the destination Web page is fetched, Figure 3(a).

2. Context Matching: After the context of the source page has been gathered and ranked, the Browser Object module downloads the destination Web page and generates a new frame tree. The algorithm matches the words in the multiset $S_{context}$, to the text in all leaves of the new frame tree, Figure 3(b), which are then assigned respective weights. The steps are enumerated below:

a. Run a depth-first search to identify all leaves $\{Leaf_1, ..., Leaf_n\}$ in the frame tree of the destination page.

b. For each $Leaf_i$:

   i. Set its weight W_i to 0.

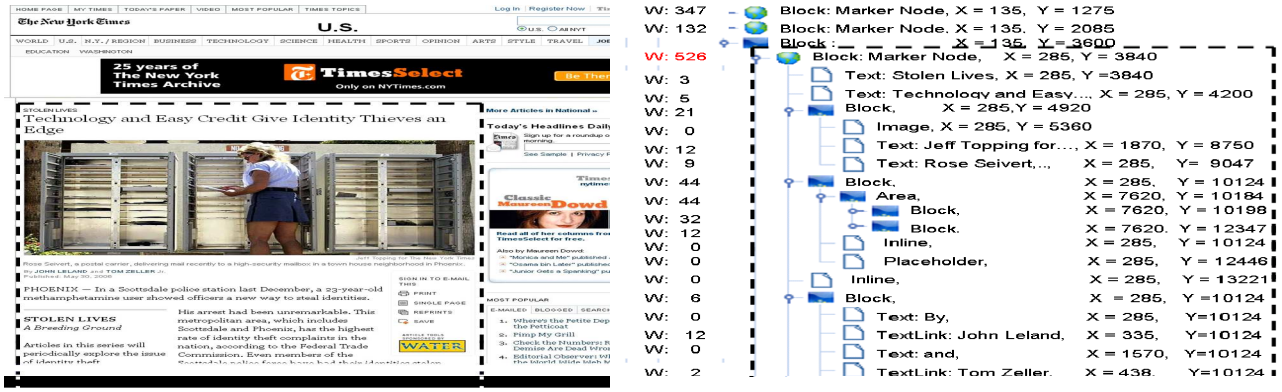
   ii. If $Leaf_i$ contains text, tokenize and store the text in the corresponding $List_i$, after removing the function words.

   iii. Search for the context keywords from set $S_{context}$ in $List_i$.

   iv. For each successful context keyword match, increment W_i by the weight of the keyword.

Continuing our example in Figure 3, the frame tree of the destination page will be searched for all words occurring in the context of the link from the source page. By the end of this step, the leaf nodes, marked with clear-box icons in Figure 3(b), will have accumulated their weights. The weight "$W : n$", in the leftmost column, express the relevance of

[1] $Function\ Words$ or grammatical words are words that have little lexical meaning or have ambiguous meaning, but instead serve to express grammatical relationships with other words within a sentence, or specify the attitude or mood of the speaker (Wikipedia.org)



(a) Destination Page

(b) Destination Frame Tree

Figure 3: Most Relevant Block Identification

each leaf-node with respect to the context gathered from the source Web page. Higher weight implies greater relevancy.

3. Block Ranking and Rearrangement: We propagate the weights from the leaves of the frame tree up to a certain node (block), which we call a *Marker Node*. This is a node of a block tree, whose children are organized contiguously on the Web page and have the same geometric alignment, i.e. have matching X or Y coordinates. To identify marker nodes, we initiate a depth first search from the root of the frame tree and recursively merge the geometrically aligned nodes. Web pages are usually organized in such a way that semantically related information is grouped together and has the same geometric alignment. Thus, the geometric alignment can help identify semantically related blocks.

For example, in Figure 1(a), all Web page objects in the area surrounded by the dotted line have the same geometrical alignment and are semantically related. The parent node will be identified as a marker node in frame tree. In Figure 1(b) marker nodes are shown by circular icons. The following steps describe the Block Ranking procedure:

a. Starting with one level above the leaf nodes, propagate the weights of the leaves up in the frame tree.

b. For block node m (with n children) calculate its weight:

$$W_m = \sum_{i=1}^{n} (n - i + 1) * W_{child_i}$$

where i = 1, ..., n, and W_{child_i} denotes the weight of the ith child of the block node m.

c. Do not propagate the weights beyond the marker nodes.

d. Store all marker nodes in a list for further processing.

We observed that the first node in any block is usually more important than the subsequent nodes. We use this observation and multiply the weight W_{child_i} of each child node in block m by $(n - i + 1)$. Thus, with each new child node, we reduce its contribution to the total weight of the block m. Then, the frame tree is reorganized based on the weights of the marker nodes, so that the most relevant block of information is placed first.

The marker node, expanded in Figure 3(b), happens to have the most weight, which makes it the most relevant block. The section of the Web page, corresponding to the selected marker node, is shown in Figure 3(a). The frame tree is rearranged in such a way, that the Interface Manager will first read the most relevant block, which, in our case, is the article about the identity theft. Having finished with the article, the Interface Manager will continue reading the rest of the Web page content.

PERFORMANCE

We experimentally compared our Web browser prototype with the state-of-the-art JAWS screen-reader. In our performance evaluation, we used over a dozen Web sites spanning 4 content domains: *news*, *books*, *consumer electronics*, and *office supplies*. Our sighted performance testers did 5 navigations on each Web site and measured the total time[2] taken to reach the relevant information using CSurf. For baseline comparison, they were also asked to do the same experiments with JAWS screen-reader using the same set of Web pages. We allowed the use of shortcuts to skip blocks of text and accelerate browsing. The results of the performance comparison are shown in Figure 4. We also calculated how well CSurf identified relevant block in each Web site. Our algorithm showed a reasonable accuracy of over 80% in all four content domains (Figure 5). We believe that a more sophisticated algorithm could substantially improve the accuracy.

We have also demonstrated CSurf to our low-vision consultant, who is an instructor teaching his students to use JAWS at Helen Keller Services for the Blind (HKSB) [5], Hempstead, NY. Our consultant noted that context-based browsing has the potential to substantially reduce information overload and improve browsing experience, compared to Web browsing with regular screen-readers.

[2]Here *total time* represents the period that starts when the user follows a link and ends when our system begins to read the relevant information on the next Web page.

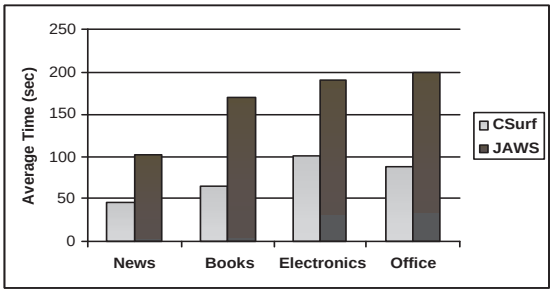


Figure 4: CSurf vs. JAWS

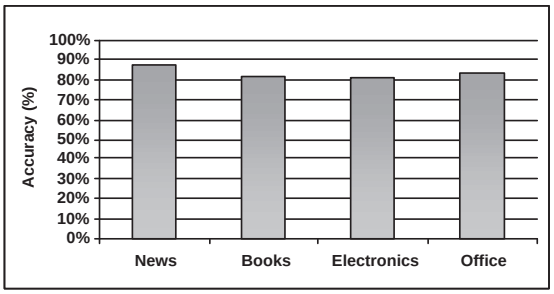


Figure 5: Accuracy of Relevant Block Identification

RELATED WORK

The work in this paper is related to research in non-visual Web access, information rearrangement in Web pages, and contextual analysis. Blind users access the Web using screen-readers, such as JAWS [6] and IBM's Home Page Reader [1]. BrookesTalk [13] summarizes Web pages to address the *information overload* problem in *non-visual Web access*. The work described in [4] generates "gist" summary of a Web page to help visually impaired users reduce the information they need to read on that page. However, summarization of the entire page does not help find the relevant information in that page on following a link. Another approach is described in [12] where users are presented with the logical structure of the Web page to select which part they want to listen. CSurf goes beyond these systems in scope and approach; it helps find relevant information quickly on following a link. *Information Rearrangement in Web Pages* mostly relies either on rules [11] or logical structures [12]. Our system uses visual layout of Web pages to automatically capture contextual information and re-arrange the content.

Contextual Analysis for non-visual Web browsing has not been previously explored. The system in [3] uses the context of a link to get the preview of the next Web page before following the link. This idea is also used in AcceSS system [9], to get the preview of the entire page. However, presenting a preview does not guarantee the reduction of browsing time. In contrast to all of these works, we aim to help blind users quickly identify relevant information *after* following a link. A poster describing our initial ideas on context-based browsing [8] appears in ASSETS'06. Here, we describe our algorithm for context-based browsing and its preliminary experimental results.

CONCLUSION AND FUTURE WORK

In this paper, we described an algorithm for context-based non-visual Web browsing. The preliminary experimental results of our algorithms show that context-based browsing can reduce information overload in non-visual Web access. We identify several potentially useful areas for further research. If search keywords are treated as context, our algorithm can be easily extended to allow smart searching within a Web page. NLP techniques can be employed to enhance context processing and searching. We are currently researching summarization techniques to further save the browsing time. We are also investigating the feasibility of applying machine learning algorithms to context identification.

REFERENCES

1. C. Asakawa and T. Itoh. User interface of a home page reader. In *ACM Intl. Conf. on Assistive Technologies (ASSETS)*, 1998.

2. Y. Borodin. A flexible vxml interpreter for non-visual web access. In *ACM Conf. on Assistive Technologies (ASSETS)*, 2006.

3. S. Harper, C. Goble, R. Stevens, and Y. Yesilada. Middleware to expand context and preview in hypertext. In *Assets '04: Proceedings of the 6th international ACM SIGACCESS conference on Computers and accessibility*, 2004.

4. S. Harper and N. Patel. Gist summaries for visually impaired surfers. In *Assets '05: Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility*, pages 90–97, 2005.

5. `http://www.hellenkeller.org`.

6. `http://www.freedomscientific.com`.

7. `http://jrex.mozdev.org`.

8. J. Mahmud, Y. Borodin, D. Das, and I. Ramakrishnan. Improving non-visual web access using context. In *ASSETS*, 2006.

9. B. Parmanto, R. Ferrydiansyah, A. Saptono, L. Song, I. W. Sugiantara, and S. Hackett. Access: accessibility through simplification & summarization. In *W4A '05: Proceedings of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A)*, pages 18–25, 2005.

10. I. Ramakrishnan, A. Stent, and G. Yang. Hearsay: Enabling audio browsing on hypertext content. In *Intl. World Wide Web Conf. (WWW)*, 2004.

11. T. Raman. Audio system for technical readings. *PhD Thesis, Cornell University*, 1994.

12. H. Takagi and C. Asakawa. Transcoding proxy for non-visual web access. In *ACM Intl. Conf. on Assistive Technologies (ASSETS)*, 2000.

13. M. Zajicek, C. Powell, and C. Reeves. Web search and orientation with brookestalk. In *Proceedings of Tech. and Persons with Disabilities Conf.*, 1999.