CSurf: A Context-Driven Non-Visual Web-Browser

Jalal Mahmud Yevgen Borodin I.V. Ramakrishnan

Department of Computer Science
Stony Brook University
Stony Brook, NY 11794, USA
{jmahmud, borodin, ram}@cs.sunysb.edu

ABSTRACT

Web sites are designed for graphical mode of interaction. Sighted users can "cut to the chase" and quickly identify relevant information in Web pages. On the contrary, individuals with visual disabilities have to use screen-readers to browse the Web. As screen-readers process pages sequentially and read through everything, Web browsing can become strenuous and time-consuming. Although, the use of shortcuts and searching offers some improvements, the problem still remains. In this paper, we address the problem of information overload in non-visual Web access using the notion of *context.* Our prototype system, CSurf, embodying our approach, provides the usual features of a screen-reader. However, when a user follows a link, CSurf captures the context of the link using a simple topic-boundary detection technique, and uses it to identify relevant information on the next page with the help of a Support Vector Machine, a statistical machine-learning model. Then, CSurf reads the Web page starting from the most relevant section, identified by the model. We conducted a series experiments to evaluate the performance of CSurf against the state-of-the-art screen-reader, JAWS. Our results show that the use of context can potentially save browsing time and substantially improve browsing experience of visually disabled people.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*natural language, Voice I/O*; H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia—*architectures, navigation*

General Terms

Algorithms, Design, Human Factors, Experimentation

Keywords

Context, Web Accessibility, Screen-Reader, Voice Browser, Non-Visual, CSurf, HearSay, Partitioning, Semantic Blocks

1. INTRODUCTION

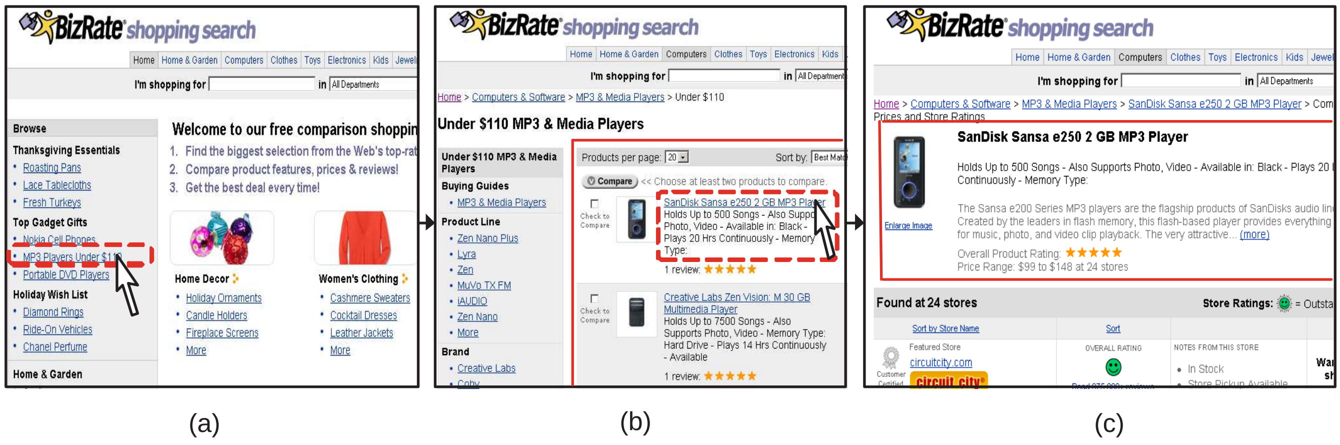
The Web has become an indispensable source of information and we use it more and more in our daily activities. The primary mode of interaction with the Web is via graphical browsers, which are designed for visual interaction. As we browse the Web, we have to filter through a lot of irrelevant data. For example, most Web pages contain banners, ads, navigation bars, and other data distracting us from the information. Sighted individuals can process visual data in no time at all. They can quickly locate the information that is most relevant to them. This task can be time-consuming and difficult for people with visual disabilities. Consider a scenario where graphical interaction is impossible, *e.g.* when users are visually challenged individuals. Typically, such people browse the Web with screen-readers [16, 3].

Many screen-readers process Web pages sequentially, i.e. they first read through the menus, banners, commercials, or anything else that comes before the desired information. This makes browsing time consuming and strenuous. To alleviate this problem, screen-readers provide shortcuts to skip segments of text in the order they appear on the page. Nevertheless, users may still have to listen or skip through substantial page content before they get to the information.

To help users locate the information quicker, most screen-readers allow keyword searching. This assumes that the users know what they are looking for. In some cases searching may help skip directly to the information. 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. This begs the question: Is it possible to devise techniques to get to the relevant information quickly?

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 context of the link to determine the relevant information on the next page and present it to the user first. Consider the example when a blind person needs to find an MP3 player on the BizRate Web site (Figure 1).

Using a standard shortcut-driven voice-browsing interface, the user finds the MP3 category in the product taxonomy in Figure 1(a). When he or she follows the link, shown by the mouse cursor, we use the words of the link to find the segment that contains the desired information, surrounded by the box enclosing the two items in Figure 1(b). The voice browser then starts reading the page from that position. After the MP3 player of interest has been found, and the user follows the SanDisk link in Figure 1(b), the words of the link and its context, enclosed by the dotted rectangle, are used to find detailed description of the MP3 player, surrounded by the solid box in Figure 1(c). In this paper we present a context-driven browsing system, CSurf, that will make possible the use scenario described above.


BizRate shopping search

Home | Home & Garden | Computers | Clothes | Toys | Electronics | Kids | Jewel

I'm shopping for in All Departments

Browse

Thanksgiving Essentials
- Roasting Pans
- Lace Tablecloths
- Fresh Turkeys

Top Gadget Gifts
- Nokia Cell Phones
- MP3 Players Under $110
- Portable DVD Players

Holiday Wish List
- Diamond Rings
- Ride-On Vehicles
- Chanel Perfume

Home & Garden

Welcome to our free comparison shoppin

1. Find the biggest selection from the Web's top-rat
2. Compare product features, prices & reviews!
3. Get the best deal every time!

Home Decor
- Holiday Ornaments
- Candle Holders
- Fireplace Screens
- More

Women's Clothing
- Cashmere Sweaters
- Cocktail Dresses
- Leather Jackets
- More

(a)

BizRate shopping search

Home | Home & Garden | Computers | Clothes | Toys | Electronics | Kids

I'm shopping for in All Department

Home > Computers & Software > MP3 & Media Players > Under $110

Under $110 MP3 & Media Players

Under $110 MP3 & Media Players

Buying Guides
- MP3 & Media Players

Product Line
- Zen Nano Plus
- Lyra
- Zen
- MuVo TX FM
- iAUDIO
- Zen Nano
- More

Brand
- Creative Labs
- Coby

Products per page: 20 Sort by: Best Matc

Compare << Choose at least two products to compare.

Check to Compare
SanDisk Sansa e250 2 GB MP3 Player
Holds Up to 500 Songs - Also Suppo Photo, Video - Available in: Black - Plays 20 Hrs Continuously - Memory Type:
1 review: ★★★★★

Check to Compare
Creative Labs Zen Vision: M 30 GB Multimedia Player
Holds Up to 7500 Songs - Also Supports Photo, Video - Memory Type: Hard Drive - Plays 14 Hrs Continuously - Available
1 review: ★★★★★

(b)

BizRate shopping search

Home | Home & Garden | Computers | Clothes | Toys | Electronics | Kids | Jewel

I'm shopping for in All Departments

Home > Computers & Software > MP3 & Media Players > SanDisk Sansa e250 2 GB MP3 Player > Com
Prices and Store Ratings

SanDisk Sansa e250 2 GB MP3 Player

Holds Up to 500 Songs - Also Supports Photo, Video - Available in: Black - Plays 20 Continuously - Memory Type:

The Sansa e200 Series MP3 players are the flagship products of SanDisks audio lin Created by the leaders in flash memory, this flash-based player provides everything for music, photo, and video clip playback. The very attractive... (more)

Enlarge Image

Overall Product Rating: ★★★★★
Price Range: $99 to $148 at 24 stores

Found at 24 stores | Store Ratings: = Outsta

Sort by Store Name | Sort

Featured Store circuitcity.com | OVERALL RATING | NOTES FROM THIS STORE
Customer Certified | | • In Stock • Store Pickup Available | Wa sh

(c)

Figure 1: Product Search Example

CSurf brings together Content Analysis, Natural Language Processing (NLP), and Machine Learning algorithms to help blind users quickly identify relevant information on following a link, thus, considerably reducing their browsing time.

The rest of the paper is organized as follows: in Section 3.1, we describe a technique for partitioning Web pages, based on their structural and visual organization, exploiting the observation that semantically-related blocks of information are often spatially aligned (e.g. see the alignment of product items in Figure 1(b)). In Section 3.2, we explain our context-collection algorithm, based on the cosine-similarity topic detection method; and in Section 3.3 we describe relevant information identification algorithm, based on the statistical model learned by a Support Vector Machine (SVM). A thorough performance testing and a preliminary user evaluation is presented in Section 4. Related work appears in Section 5, followed by concluding remarks in Section 6.

2. SYSTEM ARCHITECTURE

CSurf, our context-based browsing system, has its roots in our previous work on HearSay [27] audio browser. CSurf extends HearSay with a context analysis module, upgrading or replacing most of the old modules. CSurf is composed of the following modules: Interface Manager, Context Analyzer, Browser Object, Frame Tree Processor, and Dialog Generator, see Figure 2.

Users interact with CSurf Web Browser through the **Interface Manager**, which is an extended VoiceXML interpreter, VXMLSurfer [5], that we have developed. The module uses VoiceXML[1] dialogs to communicate with its users, process user input, and present Web page content. The Interface Manager provides both basic and extended screen-reader navigation features, such as shortcuts, voice controls, etc. The system allows both keyboard and voice inputs, and can process commands along with keyboard shortcuts. Text-to-speech and speech recognition engines are accessed through the Java Speech API (JSAPI) [18], providing a flexible interface capable of supporting different speech engines. In its current configuration, the Interface Manager uses freely available engines: FreeTTS [9] and Sphinx [32].

[1]*VoiceXML* (VXML) is the W3C's standard XML format for specifying interactive voice dialogues between a human and a computer (www.w3.org/TR/voicexml20).

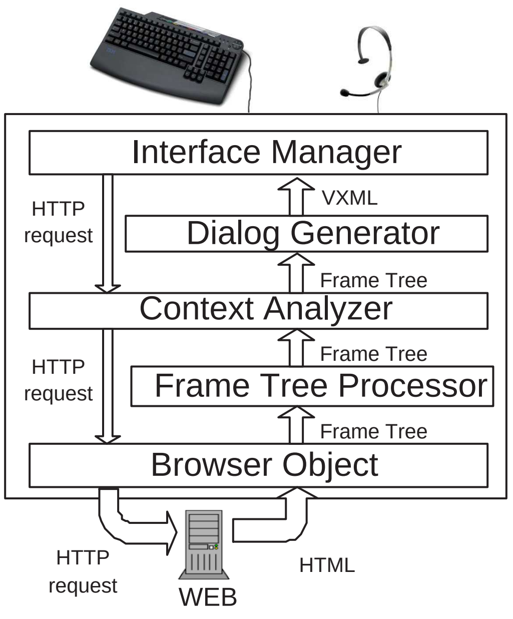


Figure 2: Architecture of CSurf

Context Analyzer is called twice for each Web page access. When the user follows a link, e.g. indicated by the arrow in Figure 1(b), the module collects the context of the link, enclosed by the dotted rectangle. When a new page is retrieved, the module executes our SVM-based algorithm to locate the content segment estimated to be most relevant with respect to the context of the followed link. The context processing algorithms are described in detail in Section 3.

The **Browser Object** module downloads Web content every time the user requests a new page to be retrieved. The module is built on top of the Mozilla Web Browser [20] coupled with JREX [17] Java API wrapper. Mozilla engine takes care of all the standard browser functionalities such as support for cookies, secure connection, history, pop-up blocking, etc. We have extended JREX to extract a *Frame Tree*, Mozilla's internal representation of a Web page, *after* the Web page has been rendered on the screen. This way, Mozilla takes care of any dynamic content, cascading stylesheets, malformed HTML, and other rendering problems. This relieves CSurf browser from having to deal with heavy DOM-tree objects, while giving it even more information about the content and the style.

Frame Tree Processor uses JREX API to extract a Frame Tree representation of Web pages (Figures 3, 4) from the Browser Object. We define a *Frame Tree* as a tree-like data structure that contains Web page content, along with its 2-D coordinates and formatting information, that specifies how the Web page has to be rendered on the screen. Frame coordinates refer to the *upper-left corners* of the corresponding Web page segments displayed on the screen.

A frame tree is composed of nested *frames*[2], so that the entire page is a root frame, containing other nested frames down to the smallest individual objects on the page. For example, Figure 3(a), section 1, shows a snapshot of the New York Times front page, with rounded rectangles illustrating the frames. In Figure 3(b), the corresponding frame-tree is partially expanded to demonstrate the types of frames. We distinguish between the following classes of frames: text, links, images, image-links, and non-leaf frames. We will continue referring to any node of a frame tree as a *frame*.

Frame Tree Processor uses a number of heuristic algorithms to clean, reorganize, and partition the frame tree. The module also detects *blocks* representing semantic clusters of information in a Web page. Figure 3(a) shows the New York Times Web page split into four sections (clusters). The frame tree nodes corresponding to these sections are marked with 3-D block icons in Figure 3(b). Block 1 contains a banner, block 2 contains a search bar, block 3 has a taxonomy, and block 4 - the news headlines. Subsequently, the Context Analyzer identifies the most relevant block, before passing the frame tree to the Dialog Generator. Section 3 describes the corresponding algorithms in detail.

The **Dialog Generator** module uses a collection of VoiceXML dialog templates to convert the frame tree into VoiceXML dialogs. The latter are then delivered to the Interface Manager. A number of sub-dialogs are also used to present history, help, lists of (un-)visited links in the page, etc. The Dialog Generator module currently supports only basic screen-reading dialogs. More research is needed to determine the optimal structure and representation for the zooming, customizable, and domain specific dialogs, which can be also expressed using VoiceXML and processed by the Interface Manager.

3. CONTEXTUAL BROWSING

This section presents the core of CSurf's Context Analyzer module, that drives *contextual browsing*. The two main algorithms enabling contextual browsing are *Context Identification* and *Relevant Block Identification*. Both of these algorithms utilize a *Geometrical Clustering* algorithm used by the Frame Tree Processor module to partition Web pages into segments, containing semantically related content.

To collect the context, a topic-detection algorithm is applied to the information surrounding the followed link. We gather the text that shares a common topic [1] with the link, and use this context to identify the relevant information on the destination Web page. Then, a support vector machine [34] is used to compute the relevance score of these sections with respect to the context. Subsequently, the Web page is presented to the user starting with the highest ranking section. If the relevant section was not identified correctly, the user can always skip to the beginning of the page. The following subsections discuss the algorithms in detail.

[2]Note, this is different from HTML frames.

3.1 Geometric Clustering

As described in Section 2, instead of implementing its own segmentation algorithm, CSurf utilizes a data structure created by the Mozilla's rendering engine. While rendering a page on the screen, Mozilla creates a tree-like structure of nested *frames* that holds the content of the Web page: the leaf frames of the tree contain the smallest individual elements, e.g. a link, an image, etc.; non-leaf frames "enclose" one or more leaf frames and/or other non-leaf frames; and the root frame "contains" the entire page. We refer to this data structure as a **frame tree**. Figure 3(b) shows an example of graphical representation of the frame tree.

We use an observation that semantically related information exhibits spatial locality [23, 22] and often shares the same alignment on a Web page. Since a frame tree represents the layout of a Web page, we infer that geometrical alignment of frames may imply semantic relationship between their respective content. If all descendants of a frame are consistently aligned either along X or Y axes, we call such a frame **consistent**.

A *Maximal Semantic Block*, or simply **block**, is the largest of the consistent frames on the path from a leaf to the root of a frame tree. Thus, it is likely to be the largest possible cluster containing semantically related items of information. For example, Figure 3(a) shows how the alignment information is used to cluster the New York Times Web page into maximal semantic blocks: banner labeled as 1, search - 2, taxonomy - 3, and news - 4.

The *FindBlocks* algorithm is used to find the blocks in a frame tree. The algorithm runs a depth-first search over the frame tree and recursively determines whether the frames are consistent, ignoring the alignment of leaf-frames. A frame is *consistently X-aligned* if all of its non-leaf descendants are X-aligned. Similarly, a frame is *consistently Y-aligned* if all of its non-leaf descendants are Y-aligned. Otherwise, the frame is not considered to be consistent. In such case, all of its children are marked as *blocks*.

```
Algorithm FindBlocks
Input: $Frame$: node of a frame tree
Output: $Blocks$: set of maximal semantic blocks
1.  Identify all children $C_1, C_2, \ldots, C_m$ of $Frame$
2.  $Frame.IsConsistent \leftarrow$ true
3.  for $j \leftarrow 1$ to $m$
4.      do if $C_j.IsLeaf$ = false
5.          then FindBlocks($C_j$)
6.              if $C_j.Alignment$ = NONE
7.                  then $Frame.IsConsistent \leftarrow$ false
8.  if $Frame.IsConsistent$ = false
9.    then for $j \leftarrow 1$ to $m$
10.           do if $C_j.Alignment \neq$ NONE
11.               then $Blocks \leftarrow Blocks \cup \{C_j\}$
12.   else $Frame.Alignment \leftarrow$ GetAlignment($Frame$)
13.       if $Frame.Alignment$ = NONE
14.         then for $j \leftarrow 1$ to $m$
15.               do if $C_j.Alignment \neq$ NONE
16.                   then $Blocks \leftarrow Blocks \cup \{C_j\}$
17. return $Blocks$
```

The *FindBlocks* algorithm uses the *GetAlignment* algorithm to check whether the children of a frame have maching alignment. That is, the *GetAlignment* algorithm determines that a frame is *X-aligned* if all of its children are aligned on the left, right, or center of the X-axis. Y-alignment of a frame is computed in a similar fashion.

```
Algorithm GetAlignment
Input: $Frame$: node of a frame tree
Output: $Alignment$ : alignment of $Frame$'s descendants
1.  Identify all children $C_1, C_2, \ldots, C_m$ of $Frame$
2.  $XFirst \leftarrow C_1.X$
3.  $YFirst \leftarrow C_1.Y$
4.  $XAlignedDescendants \leftarrow$ true
5.  $YAlignedDescendants \leftarrow$ true
6.  $Alignment \leftarrow$ NONE
7.  for $j \leftarrow 2$ to $m$
8.      do if $C_j.IsLeaf$ = false
9.            then $XCord \leftarrow C_j.X$
10.                $YCord \leftarrow C_j.Y$
11.                if $XCord \neq XFirst$
12.                  then $XAlignedDescendants \leftarrow$ false
13.                if $YCord \neq YFirst$
14.                  then $YAlignedDescendants \leftarrow$ false
15.                if $C_j.Alignment \neq XAlign$
16.                  then $XAlignedDescendants \leftarrow$ false
17.                if $C_j.Alignment \neq YAlign$
18.                  then $YAlignedDescendants \leftarrow$ false
19. if $XAlignedDescendants$ = true
20.   then $Alignment \leftarrow XAlign$
21. if $YAlignedDescendants$ = true
22.   then $Alignment \leftarrow YAlign$
23. return $Alignment$
```

The maximal semantic blocks, obtained by the Geometric Clustering algorithm, are further used by the Context Identification and Relevant Block Identification algorithms. The Dialog Generator module also makes use of the blocks when structuring its VoiceXML dialogs.

3.2 Context Identification

Once the Geometric Clustering algorithm has segmented the Web page into maximal semantic blocks, and the user selected a link to be followed, the Context Identification algorithm collects the context of the link. Before we proceed to describe our algorithm in greater detail, we formally define the notion of context as:

Context of a link is the content around the link that maintain the same *topic* as the link.

Consider Figure 3, showing the front page of The New York Times Web site and the corresponding frame tree. The context of the link, indicated by an arrow, is the text surrounded by the dotted line. Notice how the topic changes from one headline to another.

A block, produced by the Geometric Clustering algorithm, ideally represents a segment of text on the same subject, but may have several topics within it. Therefore, we limit topic boundary detection and context collection to the block containing the link. Context collection begins from the link and expands around the link until the topic of the text changes. A simple cosine similarity technique is used to detect the boundaries of the topic, see equation (1).

The *FindContext* algorithm initializes the *Context* multiset with the words and word combinations (bigram and trigram), excluding the function words[3], from the link and its non-link siblings; the text in the link siblings is ignored because links tend to be semantically independent of each other, i.e. have different topics. It then collects all text pertaining to the same topic around the link, adding the words to the *Context* multiset.

In the NYTimes example, Figure 3(a), the user follows the link "Top General Warns Against Iraq Timetable", indicated by the mouse pointer. We initialize the multiset with the text collected from the link node of the frame tree, indicated by a mouse pointer in Figure 3(b), as well as from the non-leaf sibling which follows the link node. The multiset now contains single words (e.g. "general", "david", "stout", "gen" "john", etc.), their bigrams (e.g. "david stout", "gen john"), and trigrams (e.g. "gen john abizaid").

After the initialization stage, we collect the context of the link, starting from the parent frame of the link node, by expanding the context to include the frame's siblings. We divide the siblings into the *PredList* and *SuccList*, containing the predecessor and successor siblings respectively, to expand the context window in both directions. Next, we calculate the geometric distances[4] between the initial frame and its siblings and sort the siblings accordingly.

Again, in our example, the parent frame of the link in the frame tree is the node labeled as *"a"* in Figure 3(b). The node does not have any predecessor siblings. Its successor siblings, labeled as *"b"* and *"c"*, are respectively 2795 and 3675 pixels away from frame *"a"*. Hence, we start with the sibling "b", construct multiset *SText* from the sibling's text, and compare its content to the content of the *Context* multiset. The comparison is done using cosine similarity of the multisets. More formally, for any two multisets M_1 and M_2, their cosine similarity is defined as:

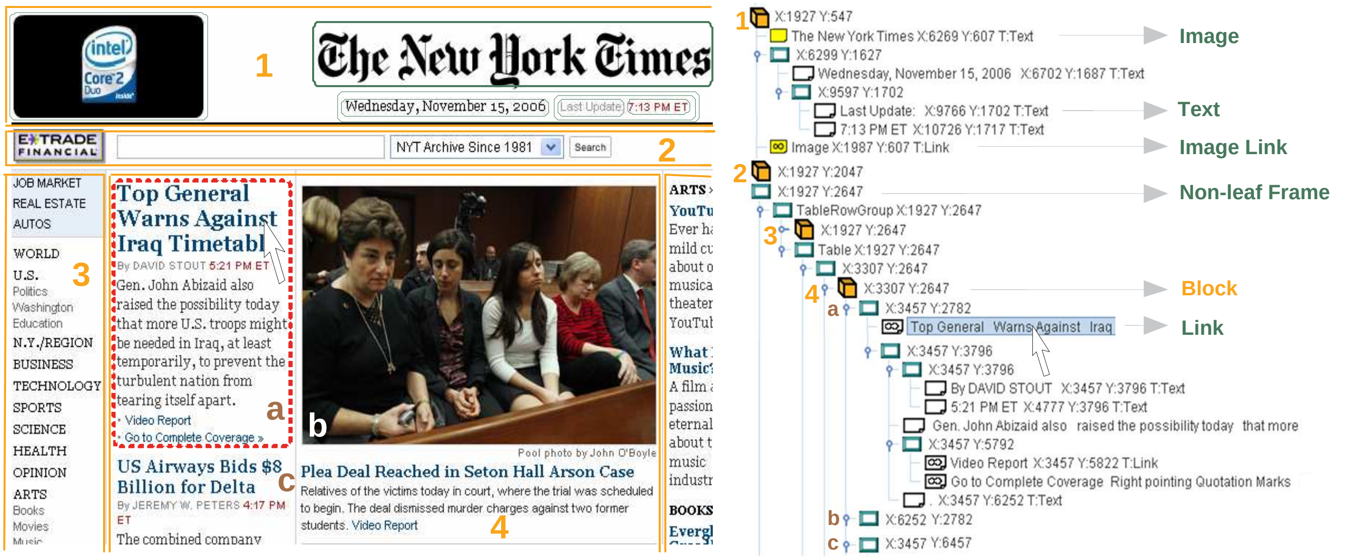
$$Cos(M_1, M_2) = \frac{M_1 \cap M_2}{\sqrt{|M_1|}\sqrt{|M_2|}} \qquad (1)$$

In the above formula, each multiset, created from a passage of text, is considered to be a vector. The cosine of the angle between the vectors is equal to 1 if the passages are identical, and 0 if they are dissimilar. We consider two multisets to be similar if their cosine similarity is above a threshold. We have statistically computed the threshold (See section 4.2 for details) that best determines whether a topic changes between the *Context* and the *SText* multisets.

If the cosine similarity between the multisets is above the threshold, i.e. topic boundary is not detected, the multisets are merged. Otherwise, we stop expanding the context window in that direction. The process continues until the *Block* boundary is reached or when there is no direction to expand. At that point, the algorithm returns the *Context* multiset as the context of the link.

[3]*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)

[4]Geometric distance between two frames is the Euclidean distance between their upper-left corners on the screen.



1
2
3
4
a
b
c
The New York Times
Wednesday, November 15, 2006
Last Update 7:13 PM ET
NYT Archive Since 1981
Search
Top General Warns Against Iraq Timetabl
By DAVID STOUT 5:21 PM ET
Gen. John Abizaid also raised the possibility today that more U.S. troops might be needed in Iraq, at least temporarily, to prevent the turbulent nation from tearing itself apart.
Video Report
Go to Complete Coverage »
US Airways Bids $8 Billion for Delta
Plea Deal Reached in Seton Hall Arson Case
Image
Text
Image Link
Non-leaf Frame
Block
Link
X:1927 Y:547
The New York Times X:6269 Y:607 T:Text
X:6299 Y:1627
Wednesday, November 15, 2006 X:6702 Y:1687 T:Text
X:9597 Y:1702
Last Update: X:9766 Y:1702 T:Text
7:13 PM ET X:10726 Y:1717 T:Text
Image X:1987 Y:607 T:Link
X:1927 Y:2047
X:1927 Y:2647
TableRowGroup X:1927 Y:2647
X:1927 Y:2647
Table X:1927 Y:2647
X:3307 Y:2647
X:3307 Y:2647
X:3457 Y:2782
Top General Warns Against Iraq
X:3457 Y:3796
X:3457 Y:3796
By DAVID STOUT X:3457 Y:3796 T:Text
5:21 PM ET X:4777 Y:3796 T:Text
Gen. John Abizaid also raised the possibility today that more
X:3457 Y:5792
Video Report X:3457 Y:5822 T:Link
Go to Complete Coverage Right pointing Quotation Marks
X:3457 Y:6252 T:Text
X:6252 Y:2782
X:3457 Y:6457

(a) Source Page (b) Source Frame Tree

Figure 3: Context Identification

```
Algorithm FindContext
Input: $LinkNode$: leaf-frame containing the link
Output: $Context$: multiset with collected context
1.  $Context \leftarrow$non-function words, their bigrams and trigrams from $LinkNode$ and its non-link siblings
2.  Let $ancesBlock$ be the ancestor $Block$ of $LinkNode$
3.  if $ancesBlock \neq LinkNode.Parent$
4.    then $Node \leftarrow LinkNode.Parent$
5.          $Expand \leftarrow$true
6.          repeat
7.              $ChildList \leftarrow Node.Parent.Children$
8.              Let $PredList$ and $SuccList$ be the lists of predecessors and successors of $Node$ in $ChildList$, sorted by their geometric distance from $Node$
9.              $StopExpand \leftarrow$false
10.             repeat
11.                 $Sibling \leftarrow PredList.Next$
12.                 $SText \leftarrow$non-function words, their bigrams, trigrams from $Sibling$
13.                 $Similarity \leftarrow Cos(Context, SText)$
14.                 if $Similarity > Threshold$
15.                   then $Context \leftarrow Context \cup \{SText\}$
                      else $StopExpand \leftarrow$true
16.                        $Expand \leftarrow$false
17.             until $PredList.IsLast$ or $StopExpand$
18.             Repeat line 9 to 17 for $SuccList$
19.             $Node \leftarrow Node.Parent$
20.         until $Node = ancesBlock$ or $Expand =$ false
21. return $Context$
```

Continuing with our example in Figure 3, we collect the text from the closest sibling frame *"b"*, corresponding to the news item "Plea Deal in Selton Hall Arson Case". The multiset *SText*, constructed for this frame, now contains {"plea", "deal", "selton", ..., "plea deal", ..., "plea deal selton", ...}. We compute the cosine similarity of the *Context* and a *SText* multisets, which turn out to be below our threshold. The algorithm detects a topic boundary between the content of the multisets and, therefore, stops expanding the context window and returns the *Context* multiset. The context of the followed link, Figure 3(a), is enclosed by the dotted line.

3.3 Relevant Block Identification

After the context of the link has been gathered on the source page, the Browser Object module downloads the destination Web page and generates a new frame tree. Again, we use our Geometric Clustering algorithm to segment the page into maximal semantic blocks: 1, 2, 3 in Figure 4. Then, the Relevant Block Identification algorithm matches the context against every block in the frame tree and computes the relevance of each block with respect to the collected *context*.

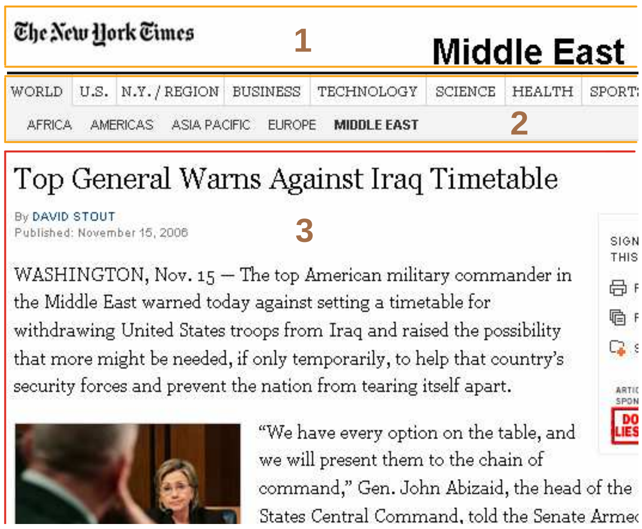
Intuitively, a relevant block identification algorithm should use a block ranking function to weigh the blocks and, then, pick the top-scoring block as the most relevant one. Formally, block ranking is a function which takes a vector $\vec{f}$ of block feature values $f_1, f_2, \ldots, f_n$ and a vector $\vec{w}$ with feature contributions $w_1, w_2, \ldots, w_n$, and returns the weight W of the block:

$$\mathbf{F} : (f_1, f_2, \ldots, f_n) \times (w_1, w_2, \ldots, w_n) \rightarrow W \qquad (2)$$

A naive approach is to manually design such function and fix the individual weights for each feature vector. For example, a function can be based on addition of feature values:

$$\mathbf{F}(\vec{f} \times \vec{w}) = w_1 \cdot f_1 + w_2 \cdot f_2 + \ldots + w_n \cdot f_n \qquad (3)$$

However, manually designing such function is not practical, justifiable, or scalable over a multitude of features. Therefore, we learn a block ranking function using a statistical learning method: we define "block ranking" as a learning problem and use a support vector machine (SVM) [34, 7], a well-known statistical model used in classification and


The New York Times 1 Middle East

WORLD U.S. N.Y./REGION BUSINESS TECHNOLOGY SCIENCE HEALTH SPORT

AFRICA AMERICAS ASIA PACIFIC EUROPE MIDDLE EAST 2

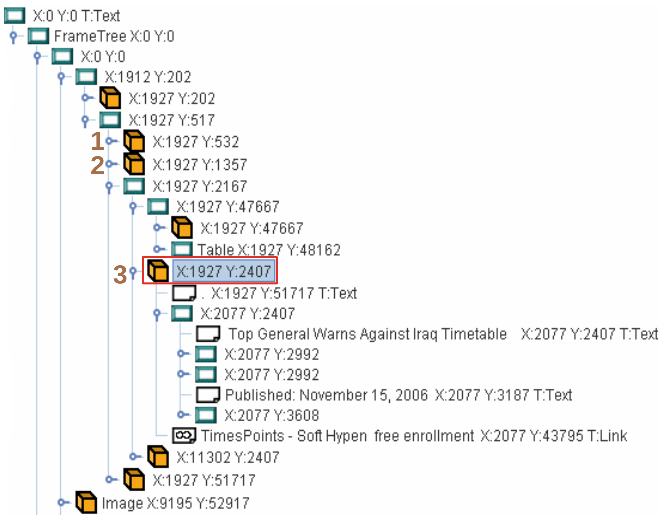
Top General Warns Against Iraq Timetable

By DAVID STOUT
Published: November 15, 2006 3

WASHINGTON, Nov. 15 — The top American military commander in the Middle East warned today against setting a timetable for withdrawing United States troops from Iraq and raised the possibility that more might be needed, if only temporarily, to help that country's security forces and prevent the nation from tearing itself apart.

"We have every option on the table, and we will present them to the chain of command," Gen. John Abizaid, the head of the States Central Command, told the Senate Armed

(a) Destination Page


```
X:0 Y:0 T:Text
FrameTree X:0 Y:0
  X:0 Y:0
    X:1912 Y:202
      X:1927 Y:202
      X:1927 Y:517
     1  X:1927 Y:532
     2  X:1927 Y:1357
        X:1927 Y:2167
          X:1927 Y:47667
            X:1927 Y:47667
            Table X:1927 Y:48162
       3  X:1927 Y:2407
            X:1927 Y:51717 T:Text
            X:2077 Y:2407
              Top General Warns Against Iraq Timetable   X:2077 Y:2407 T:Text
              X:2077 Y:2992
              X:2077 Y:2992
              Published: November 15, 2006 X:2077 Y:3187 T:Text
              X:2077 Y:3608
            TimesPoints - Soft Hypen free enrollment X:2077 Y:43795 T:Link
          X:11302 Y:2407
        X:1927 Y:51717
    Image X:9195 Y:52917
```

(b) Destination Frame Tree

Figure 4: Most Relevant Block Identification

Table 1: Description of Block Features

Feature	Description
$f_{unigram}$	exact match of context words
f_{bigram}	exact bigram match (pairs of words)
$f_{trigram}$	exact trigram match (triples of words)
$f_{stemUnigram}$	match of word stems*
$f_{stemBigram}$	match of stemmed bigrams
$f_{stemTrigram}$	match of stemmed trigrams

regression analysis, to learn a block relevance model. Then, we use the machine-learned model to determine the relevance of blocks with respect to a given context. The blocks are ranked according to their relevance to the context.

As many other machine learning tools, SVM takes a feature vector as input and produces its classification (when SVM is used in classification problem). We define two classes for our block relevance model: relevant and not relevant, and describe each block of the destination page with a set of feature values, which we compute by trying to match single words, bigrams, trigrams, and their stemmed[5] counterparts, contained in context, to the text in the blocks. The features are listed in Table 1. On each successful match the corresponding feature value is incremented.

We used a freely available SVM package, libsvm [7], to learn the block relevance model. To train the SVM model, we collected and labeled training examples, where each block is represented by a tuple $(\vec{f}, l)$, where $\vec{f}$ is a feature vector for that block and l is its relevance label. To simplify the labeling, a block can be either relevant with respect to some context: $l = 1$, or not relevant: $l = 0$.

To compile labeled training data, we manually collected Web logs from about 1000 Web pages using our data-collection tool described in Section 4.1. Each Web log is a tuple of (*Source Page*, *Destination Page*), with the context and the link selected on the source page, and the most relevant block selected on the destination page. We identified the set of blocks $B_1, B_2, \ldots, B_m$ on the destination Web page, computed their feature values, and labeled the blocks as relevant (*1*) or not relevant (*0*). The training examples were, then, used to train the SVM and learn the SVM model for relevant block identification.

The machine-learned SVM model is now used to predict the relevance of a given block with respect to some context. Given a set of blocks $B_1, B_2, \ldots, B_m$, we compute the feature values for each block by matching the context against the text in the block. Next, we use the learned SVM model to label the blocks as either relevant or not-relevant, and get the associated probability values. Then, we pick the highest ranking block, in terms of the probability values, as the most relevant one. Block 3, expanded in Figure 4(b), was chosen by the SVM as the most probable candidate for contextual relevancy. Subsequently, the CSurf will read the page starting from section 3 of the Web page in Figure 4(a).

4. EVALUATION AND EXPERIMENTS

We developed the system, CSurf, by significantly expanding a naive implementation, we had previously described in a short paper [19]. Unlike CSurf, the naive prototype, which we will call CSurf-Simple, did not use topic detection, SVM-based learning, word-stemming or other NLP techniques. Furthermore, we had not previously done a comprehensive evaluation of the CSurf-Simple system.

In this section we present an extensive performance evaluation of CSurf; we also compare the quantitative performance of CSurf and CSurf-Simple and show that the former has considerably higher performance in terms of accuracy of context identification, relevant block identification, and browsing efficiency. However, we leave the detailed analysis of the individual contributions of the afore-mentioned techniques to future work.

[5]Word stemming is done using Porter's stemmer [26]

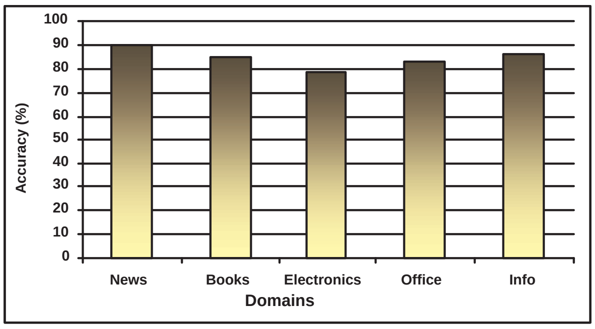


Figure 5: Accuracy of Context Identification

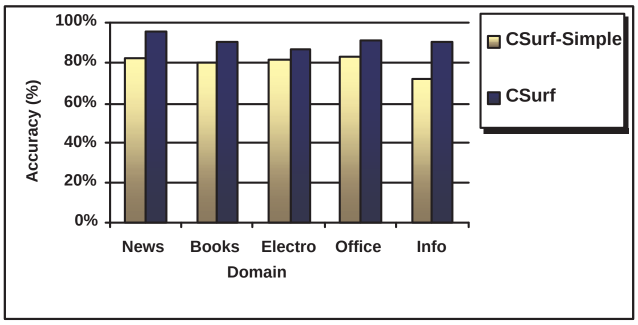


Figure 6: Accuracy of Relevant Block Identification

4.1 Experimental Setup

In our experiments and evaluation we have used both blind and sighted users. Our system was evaluated by thirty graduate students and three blind people. It was impractical to get quantitative measurements of the accuracy of our algorithms with blind users. Therefore, we used sighted students to obtain those metrics, while blind evaluators helped get qualitative feedback.

Prior to the experiments, all evaluators were trained to use the CSurf Web browser. The evaluators interacted with the system using keyboard and headphones. Blind evaluators were allowed to browse the Web sites, which they were familiar with. We used twenty five Web sites for the student evaluation and data collection, 5 Web sites in each of 5 content domains: *news*, *books*, *consumer electronics*, *office supplies*, and *informational*. The informational category included various Web sites, such as LIRR[6] and Medicaid[7].

To have an efficient infrastructure for experiments, we designed a visual tool for viewing frame trees, as well as collecting data. We also embedded a Web browser to aid the data collection. We manually collected around *1000* (source-destination) pairs of Web pages to calculate a threshold for topic identification and to train the SVM model. During the data collection stage, the participants were asked to select any link and the context around it on the source pages. Then, they were told to follow the link and select one block containing what, they thought, was the most relevant information with respect to the link they had chosen. The frame trees, corresponding to the source and the destination pages, were automatically saved together with user selections.

4.2 Accuracy of Context Identification

We used the collected source pages to statistically compute the accuracy and the threshold for our topic boundary detection algorithm, as described in Section 3.2. We used 50% of the page samples to estimate the threshold value and the remaining 50% were used to calculate the accuracy of topic identification algorithm.

We defined the *accuracy* of context identification as cosine similarity between the human- and computer-selected context, with the cosine similarity value ranging between 0 and 1, where 1 signified a 100% match. We used this measure for threshold estimation, as well as quantitative evaluation of the context-identification algorithm.

[6]http://www.mta.nyc.ny.us/lirr
[7]www.cms.hhs.gov/home/medicaid.asp

We designed a greedy algorithm that started with an unrealistically high threshold (*1*), that would only accept identical passages of text; we used our context identification algorithm to find the context of the selected links in 50% of the sample Web pages, compared the results with the human-selected context, and, then, adjusted the threshold value iteratively until it converged to the accuracy that locally could not be improved any further.

Specifically, we set the threshold $T_1 = 1$, $n = 1$, and $\delta = 0.1$; we compared the accuracies A_n and A_{n+1} while adjusting the threshold $T_{n+1} = T_n - \delta$ iteratively, as long as $A_n < A_{n+1}$. Then, we used a binary-search approach to converge to the optimal threshold T_{opt} between T_n and T_{n+1}, where the accuracy A_{opt} was the local maximum. Finally, we used the remaining 500 collected Web pages to calculate the average accuracy of context identification in each of the 5 domains. The accuracy is summarized in Figure 5.

CSurf scored in the range of *80%* to *90%* accuracy compared with human-selected context. CSurf scored the highest with the "News" and "Informational" Web sites; higher accuracy of context identification in these two domains can be explained by the fact that they are better organized and have more textual content. Context identification accuracy received the worst score (80%) in the "Electronics" domain, and average scores in "Books" and "Office Supplies" Web sites. This is, most likely, because e-commerce Web sites crowd their pages with more diverse information, preferring to use more images than text. While CSurf handles *ALT* tagged images, many online stores disregard Web accessibility guidelines, making it difficult to use images as context.

4.3 Accuracy of Relevant Block Identification

The collected data was also used for SVM training and the evaluation of the algorithm's accuracy. Using the relevant block identification algorithm on each pair of the collected Web pages, we computed a feature vector for each block of the destination Web page, as described in (Section 3.3). The human-selected most-relevant blocks were labeled with 1's (i.e. relevant), while the rest were labeled with 0's. We divided the training data in two sets: training (90%) and cross-validation (10%).

Once the SVM model was learned, we tested it by using 100 Web page pairs of the cross-validation set to predict block labels. The labels were then compared with the human-selected ones, and the accuracy of the prediction was calculated based on the hit-miss approach, i.e. the number of correctly identified blocks over the total number of blocks.

The learned model showed an average of 91% accuracy in its prediction of the most relevant blocks, summarized in Figure 6. The same chart shows the prediction-accuracy comparison of CSurf over CSurf-Simple with 14% improvement. To do a proper comparison of the relevant block identification algorithms, we updated CSurf-Simple with the topic-detection-based context identification algorithm. This allowed us to have a uniform reference model for comparing the two relevant-block-identification algorithms.

Our algorithm showed reasonable performance in all five content domains. It is notable that our algorithm again achieved the best result of over *95%* in the news domain. The relevant information identification algorithm depends on the geometric organization of Web pages and the performance of the Geometric Clustering Algorithm; the latter performs the best on well-structured Web sites. The structural organization is often much better in News Web sites than in other domains, which explains CSurf's high performance in that domain. The "Informaional" category was not among the high performers, because informational Web sites tend to have more homogenous content and less structure, compared to the news Web sites. The relevant block identification algorithm averaged about *90%* accuracy in the e-commerce (Books, Electronics, Office Supplies) categories. The decrease of accuracy, compared to the News Web sites, was due to the ambiguity introduced by the high similarity among the different items occurring within the page. Another contributing factor was the presence of user reviews, having more word matches and, thus, scoring higher than product descriptions.

4.4 Browsing Efficiency with CSurf

We performed preliminary evaluation of the advantages introduced by context-directed browsing vs. simple screen reading. Since our goal was to provide faster access to relevant information, we measured the time taken to reach the desired information after following a link using CSurf, CSurf-Simple, and a state-of-the-art screen-reader, JAWS. The tree experiments were conducted in parallel.

We did, on average, 5 navigation steps on each of the 25 Web sites. The CSurf evaluators were not allowed to move back on the page when the actual relevant information preceded the CSurf's choice of the "most" relevant block. In such case, the users had to "wrap" around and start from the beginning of the page before they could get to the information. This is a better approximation of the behavior of blind users, because they would feel lost on an unfamiliar page. On the other hand, depending on personal preferences and the familiarity with a Web page structure, blind users could prefer to skip to the beginning of the page as soon as they became aware of the CSurf's mistake, thus, increasing the actual time gain than reported in this section.

Familiarity with a Web page after the first reading is another predicament in getting valid evaluation results. Therefore, the experiments were performed by sighted users with the goal of getting a rough estimate of the potential improvements of the browsing time. Rigorous structured between- or within-the-group evaluations by blind people are required to determine the actual time gains, which may be different for blind users. However, our preliminary results show promise that context-directed browsing can substantially improve browsing efficiency for blind people. Qualitative evaluation by blind people is described in Section 4.5.

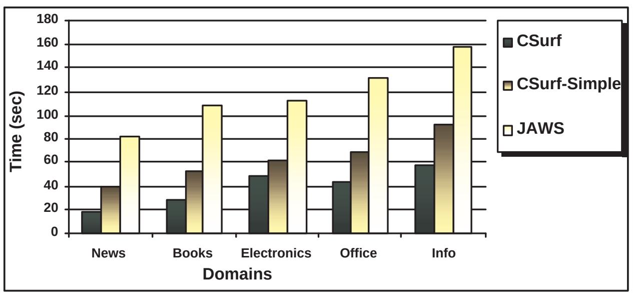


Figure 7: CSurf vs. CSurf-Simple vs. JAWS

We observed that for all domains our system was able to present the relevant information faster than JAWS. Since JAWS only follows the layout of the page content, it takes it much longer to get to the relevant information. On the other hand, CSurf takes its users to the information directly, thus, reducing their browsing time. In Figure 7, we summarize the results of the experiments by giving time comparisons of CSurf, CSurf-Simple, and JAWS in all 5 content domains.

The results showed a speedup of *38%* over CSurf-Simple, and *66%* over the state-of-the-art screen-reader JAWS. The variations in the average time taken between different domains was due to the small sample sizes. The news domain showed the higher time gain than other domains, clearly, due to the higher accuracy of relevant block identification.

4.5 End-User Experience

Following the experiments, we also conducted preliminary qualitative evaluation of CSurf with 3 blind and low-vision users at Helen Keller Services for the Blind (HKSB), Hempstead, NY [12]. The evaluators, who were experienced computer users proficient with JAWS, were quickly trained to use CSurf and were, then, asked to browse familiar-to-them Web sites. In the end of the two-hour evaluation session, they were asked to give their opinions on CSurf's performance. The evaluators noted that context-directed browsing, although not always accurate, is a substantial improvement over regular browsing with screen-readers, because screen-readers always start reading from the beginning of the page. It is worth mentioning that our research was spurred by the request of HKSB's blind instructors to find a way "not to start reading from the beginning of the page" while browsing the Web.

Blind users also liked several other features of CSurf, such as various shortcuts, e.g. the list of links in a page; advanced voice controls to change voices and voice properties; and a magnified input window for partially sighted users. Among the shortcomings of the system, the blind evaluators found some deficiencies with our navigation controls, which made it difficult to move within a page. In our current VoiceXML dialogs, users are allowed to skip between blocks in any direction, but they cannot move back within a block. We are in the process of redesigning the dialog structure to accommodate this functionality. Also, because CSurf processes and analyzes Web pages statically, some dynamic content, such as Flash and JavaScript menus, may not be accessible. We are improving our system to handle the above-mentioned issues and have already scheduled a comprehensive evaluation of the improved CSurf at HKSB.

5. RELATED WORK

The work described in this paper has broad connections to research in non-visual Web access, Web content analysis, and contextual analysis.

Non-visual Web Access. Several research projects aiming to facilitate non-visual Web access include work on browser-level support [16, 3, 33], content adaptation and summarization [37, 30, 11], organization and annotation of Web pages for effective audio rendition [28, 14, 13], etc.

Some of the most popular screen-readers are JAWS [16] and IBM's Home Page Reader [3, 33]. An example of a VoiceXML browsing system (which presents information sequentially) is described in [24]. All of these applications do not perform content analysis of Web pages. BrookesTalk [37] facilitates non-visual Web access by providing summaries of Web pages to give its users an audio overview of Web page content. The work described in [11] generates a "gist" summary of a Web page to alleviate information overload for blind users. However, summarization of the entire page does not help find the relevant information within it. CSurf, having roots in our earlier HearSay audio browser [27], goes beyond these systems in scope and approach: it analyzes the content of Web pages and helps find relevant information in them while navigating from one page to another.

The works describing organization and annotation of Web pages for better audio rendition typically rely on rules [28] or logical structures [13]. The ASTER system [28] permits visually challenged individuals to manually define their own document-reading rules. Other researchers propose the idea of extracting content using semantics [14]. They describe a framework for manual annotation of the content w.r.t. a schema, representing the task a user wishes to accomplish. These annotation rules are also site specific, and, hence, not scalable over content domains.

The essential difference between our work and all of the above-mentioned research is that we do not require any domain knowledge in terms of rules. CSurf dynamically captures the contextual information and uses it to facilitate non-visual Web access.

Web Content Analysis. A critical piece of our context analysis algorithm is in partitioning Web pages into geometric segments (blocks). Substantial research has been done on segmenting Web documents [33, 8, 35]. These techniques are either domain-specific [8], site-specific [33], or depend on fixed sets of HTML markups [35]. Semantic partitioning of Web pages has been described in [21, 22]. These systems require semantic information (e.g. ontologies). In contrast to all of these works, our geometric clustering method does not depend on rules, domain knowledge or ontologies.

Web page partitioning techniques have been used for content adaptation [6, 4] and content caching [29]. VIPS [36] algorithm uses visual cues to partition a Web page into geometric segments. This algorithm is used in [31], where the segments are described by a set of features (e.g. spatial features, number of images, sizes, links, etc.). The feature values are then fed into an SVM, which labels the segments according to their importance.

CSurf also uses an SVM to rank the blocks on the destination Web page w.r.t. the context of the link in the source page. In contrast to [31], where the SVM model was learned using features only from the content of Web page segment, our SVM model uses the feature set (See Table 1), computed from both the context of the link and the content of the block.

The main difference between our research and the above-mentioned techniques is that we exploit geometrical and logical structure of Web pages both to collect context and to identify relevant information on the next Web page.

Contextual Analysis. The notion of context has been used in different areas of Computer Science research. For example, [15] defines context of a Web page as a collection of text, gathered around the links in other pages pointing to that Web page. The context is then used to obtain a summary of the page. Summarization using context is also explored by the InCommonSense system [2], where search engine results are summarized to generate text snippets.

The use of contextual information for non-visual Web access is not a well-studied problem. A technique resulting from early efforts at context analysis for non-visual Web access is described in [10], where context of a link is used to get the preview of the next Web page, so that visually disabled individuals could choose whether or not they should follow the link. This idea is used in AcceSS system [25], to get the preview of the entire page. However, presenting a preview does not guarantee the reduction of browsing time.

All of these works define the context of the link as an ad-hoc collection of words around it. In contrast, our notion of context is based on topic similarity of text around the link. We use a principled approach for context analysis with a simple topic boundary detection method [1], confined to geometric clusters that have semantically related content.

CSurf is fundamentally different from all of these works in its application. Specifically, it aims to help visually disabled users quickly identify relevant information on following a link, thus, potentially reducing their browsing time.

Our initial ideas on context-directed browsing will appear as a short paper [19]. We have substantially extended this naive preliminary work with a geometric clustering technique to identify maximal semantic segments; a topic detection [1] algorithm for context collection; an SVM block-relevance model to identify relevant information in Web pages for context-directed browsing; as well as word-stemming and other NLP techniques. We developed a stable and usable CSurf Web-browser supporting many of JAWS shortcuts. We have also conducted performance evaluations to justify our techniques, and preliminary usability testing by blind users at HKSB.

6. CONCLUSION AND FUTURE WORK

In this paper, we described the design and implementation of CSurf, our context-directed non-visual Web browser. The system uses Web page partitioning and techniques from NLP and Machine Learning. We demonstrated the effectiveness of our algorithms by showing substantial performance improvements over our base-line prototype, CSurf-Simple. We leave the detailed analysis of the individual contributions of SVM, word-stemming, and topic detection to future work. Using our system, visually impaired individuals can potentially imitate the browsing behavior of sighted users, saving their time on not listening to irrelevant information. Thus, CSurf goes beyond traditional screen-readers in its ability to combat information overload and "cut to the chase".

In the future, we will explore the use of other features for enhancing the performance of context-browsing. For example, by introducing more features that distinguish between parts of speech and text formatting, the accuracy of the SVM model could be improved. Bringing other NLP techniques (e.g. summarization) to enhance browsing experience is another interesting direction for research. Contextual browsing also has implications for handheld devices with small screens. It may be possible to adapt our algorithms to identify and display the most relevant sections of Web pages on small screens effectively, making navigation with handhelds more efficient.

Finally, we would like to mention that the instructors at HKSB are ready to switch over to CSurf, as soon as we have a stable version implementing all major shortcuts of JAWS screen-reader. HKSB have also agreed to provide their power users with visual disabilities for pre-release beta testing of CSurf. Following the beta-testing, we are planning to release the first version of the CSurf non-visual context-based voice browser.

7. REFERENCES

[1] J. Allen. Topic detection and tracking: Event-based information organization. Kluwer Academic Publishers, 2002.

[2] E. Amitay and C. Paris. Automatically summarising web sites - is there a way around it? In *Proceedings of ACM 9th CIKM*, pages 173–179, 2000.

[3] C. Asakawa and T. Itoh. User interface of a home page reader. In *ASSETS*, 1998.

[4] S. Baluja. Browsing on small screens: recasting web-page segmentation into an efficient machine learning framework. In *WWW*, pages 33–42, 2006.

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

[6] O. Buyukkoten, H. Garcia-Molina, and A. Paepcke. Seeing the whole in parts: Text summarization for web browsing on handheld devices. In *WWW*, 2001.

[7] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. `http://www.csie.ntu.edu.tw/∼cjlin/libsvm`.

[8] D. Embley and L. Xu. Record location and reconfiguration in unstructured multiple-record web documents. In *WebDB*, 2000.

[9] `http://freetts.sourceforge.net`.

[10] 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.

[11] 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.

[12] `http://www.hellenkeller.org`.

[13] M. Hori, G. Kondoh, K. Ono, S. ichi Hirose, and S. Singhal. Annotation-based web content transcoding. In *WWW*, 2000.

[14] A. Huang and N. Sundaresan. A semantic transcoding system to adapt web services for users with disabilities. In *ASSETS*, 2000.

[15] B. B.-M. J.-Y. Delort and M. R. Enhanced. Enhanced web document summarization using hyperlinks. In *HYPERTEXT'03: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 208–215, 2003.

[16] `http://www.freedomscientific.com/`.

[17] `http://jrex.mozdev.org/`.

[18] `http://java.sun.com/products/java-media/speech`.

[19] J. Mahmud, Y. Borodin, D. Das, and I. Ramakrishnan. Combating information overload in non-visual web access using context. In *IUI*, 2007. Short paper.

[20] `http://www.mozilla.com/firefox/`.

[21] S. Mukherjee, I. Ramakrishnan, and A. Singh. Bootstrapping semantic annotation for content-rich html documents. In *ICDE*, 2005.

[22] S. Mukherjee, G. Yang, and I. Ramakrishnan. Automatic annotation of content-rich html documents: Structural and semantic analysis. In *Intl. Semantic Web Conf. (ISWC)*, 2003.

[23] S. Mukherjee, G. Yang, W. Tan, and I. Ramakrishnan. Automatic discovery of semantic structures in html documents. In *Intl. Conf. on Document Analysis and Recognition*, 2003.

[24] `http://www.internetspeech.com`.

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

[26] M. Porter. An algorithm for suffix stripping. In *Program*, pages 130–137, 1980.

[27] I. Ramakrishnan, A. Stent, and G. Yang. Hearsay: Enabling audio browsing on hypertext content. In *WWW*, 2004.

[28] T. Raman. Audio system for technical readings. *PhD Thesis, Cornell University*, 1994.

[29] L. Ramaswamy, A. Iyengar, L. Liu, and F. Douglis. Automatic detection of fragments in dynamically generated web pages. In *WWW*, 2004.

[30] J. T. Richards and V. L. Hanson. Web accessibility: a broader view. In *WWW*, pages 72–79, 2004.

[31] R. Song, H. Liu, J.-R. Wen, and W.-Y. Ma. Learning block importance models for web pages. In *WWW*, pages 203–211, 2004.

[32] `http://cmusphinx.sourceforge.net`.

[33] H. Takagi, C. Asakawa, K. Fukuda, and J. Maeda. Site-wide annotation: Reconstructing existing pages to be accessible. In *ASSETS*, 2002.

[34] V. Vapnik. Principles of risk minimization for learning theory. In *D. S. Lippman, J. E. Moody, and D. S. Touretzky, editors, Advances in Neural Information Processing Systems 3*, pages 831–838. Morgan Kaufmann, 1992.

[35] Y. Yang and H. Zhang. HTML page analysis based on visual cues. In *Intl. Conf. on Document Analysis and Recognition (ICDAR)*, 2001.

[36] S. Yu, D. Cai, J.-R. Wen, and W.-Y. Ma. Improving pseudo-relevance feedback in web information retrieval using web page segnmentation. In *WWW*, 2003.

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