Dialog Generation for Voice Browsing

Zan Sun Amanda Stent I.V. Ramakrishnan

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

ABSTRACT

In this paper we present our voice browser system, *HearSay*, which provides efficient access to the World Wide Web to people with visual disabilities. HearSay includes content-based segmentation of Web pages and a speech-driven interface to the resulting content. In our latest version of HearSay, we focus on general-purpose browsing. In this paper we describe HearSay's new dialog interface, which includes several different browsing strategies, gives the user control over the amount of information read out, and contains several different methods for summarizing information in part of a Web page. HearSay selects from its collection of presentation strategies at run time using classifiers trained on human-labeled data.

1. INTRODUCTION

The World Wide Web has become an indispensable aspect of our society, used for education, commerce, medicine and entertainment. However, the primary means of accessing the Web is via browsers designed for visual modes of interaction (*e.g.*, Internet Explorer, Firefox, etc.). This limits access for an entire community of people with visual disabilities. This target population faces particular difficulties in accessing, scanning and summarizing/distilling information on a Web page or group of pages, filling out Web forms, and using Web search facilities.

Creating audio browsable Web content has become the focus of intensive research efforts by industrial enterprises (*e.g.*, IBM) and standardization organizations (*e.g.*, W3C). New markup languages, such as VoiceXML [9], SALT [8] and XHTML+Voice [11], and new voice browser systems, such as IBM's WebSphere Voice Server, have emerged to facilitate the creation, publishing, and exchange of audio browsable Web content. However, adapting to voice browser technology still remains a significant burden for many Web content providers. Furthermore, while current screen readers and voice browsers are useful for reading HTML documents, they impose significant overhead on users. These systems provide almost no filtering of Web page content to eliminate "noise" (*e.g.*, advertisements), and do not provide the user with a semantic view of the pages being browsed. As a result, the user is forced to arrow down or page down through a single columned presentation of all the links on a given page including the navigational links and ads.

In previous work, we developed a voice browser system, HearSay [29]. HearSay provided access to the content of news, commerce and educational Web pages in an efficient and simple way and had a uniquely flexible, controllable interface. However, because HearSay relied on ontologies and hand-built templates for content extraction and presentation, its scope was limited. In this paper we present our revised, general-purpose HearSay system. We focus on HearSay's dialog interface, which includes several different browsing strategies, gives the user control over the amount of information read out, and contains several different methods for summarizing information in part of a Web page. HearSay selects from its collection of presentation strategies at run time using classifiers trained on human-labeled data.

The rest of this paper is organized as follows. In Section 2 we describe the HearSay system. In Section 3 we present HearSay's general purpose browsing strategies. We describe HearSay's content presentation strategies in Section 4. We discuss related work is described in Section 5 and finally, we conclude in Section 6.

2. INTRODUCTION TO HEARSAY

2.1 HearSay Architecture

The architecture of the HearSay voice browser is shown in Figure 1. It includes three basic components: the Browser Object Interface, the Content Analyzer, and the Interface Manager. The *Browser Object Interface*[1] fetches pages from Web servers. Special features include automatic form fill-outs and retrieval of pages pointed to by navigable links that require execution of JavaScript.

The *Content Analyzer partitions* an input Web page into a logical structure of segments containing related content elements by analyzing the page's structure and content. The output of the *Content Analyzer* is a *partition tree* of the content in the input page.

The *Interface Manager* labels each partition in the partition tree using pre-trained classifiers, described in Section 4. These labels are used by the *Dialog Generator* that auto-

[1]http://msdn.microsoft.com/library/default.asp?url=/workshop/browser/prog_browser_node_entry.asp

W4A at WWW2006 23-26 May 2006, Edinburgh, UK

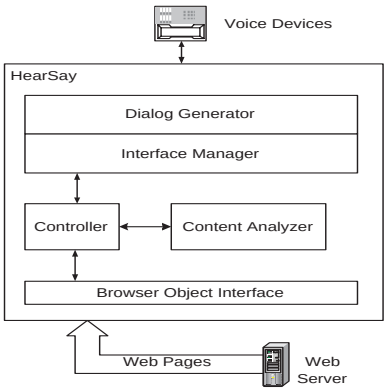


Figure 1: Architecture of HearSay

Figure 2: An example from New York Times.

matically generates a VoiceXML dialog interface to the Web page content. We use our own VoiceXML interpreter, along with freely available text-to-speech synthesizers and speech recognizers, to execute this VoiceXML dialog.

2.2 Using HearSay

HearSay's main output modality is speech. It accepts input in text or speech, so can be used both on small form-factor devices such as PDAs, and on desktops/laptops.

Suppose Alice is a student who has visual disabilities. She often browses the Web using HearSay. Today she opens the web site of New York Times (`http://www.nytimes.com`) by saying "New York Times". After loading the page (shown in Figure 2), HearSay automatically *partitions* it into segments of related content using structural and semantic information.

HearSay automatically creates a VoiceXML dialog interface to the partitioned content. For the Web page in Figure 2, HearSay might say, "There are two sections. Section one (size: 2 percent) keywords are New York Times, Personalize Your Weather, Updated Friday,.... Navigate it?". From the description Alice decides that section one is the header of the Web page and replies "No". HearSay continues to the next section, saying "Section two (size: 98 percent) keywords are news, international, business, ... Navigate it?". Alice chooses to browse this partition (labeled *2* in Figure 2), which itself contains 3 partitions. After further browsing, Alice may navigate to a headline story linked to in the partition labeled *3* in Figure 2, at which point HearSay will read out the story. Alice can also invoke common navigation commands such as "Go back", "Skip" or "Repeat" at any time.

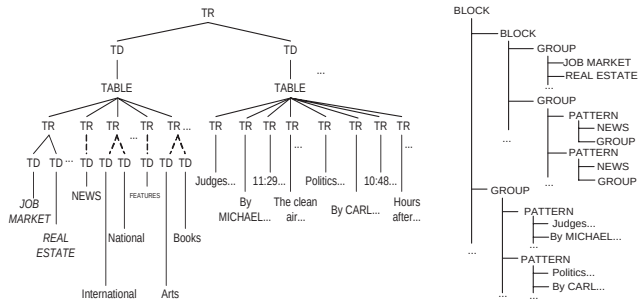
Unlike our first version of HearSay [29], which was based on using domain-specific ontologies and dialog templates, the current HearSay is designed for generality. Consequently, we have modified both our Content Analyzer and our Interface Manager. We briefly describe our Content Analyzer, then look in detail at our Interface Manager.

2.3 Content Analysis

Here we describe the *content analysis* algorithm that HearSay uses to partition a Web page into semantically related segments. It is based on our previous work on structural and semantic analysis of Web content [24, 29, 25, 26]. Content analysis (see [24] for details) is based upon the observation that semantically related items in content-rich Web pages exhibit consistency in presentation style and spatial locality. Exploiting this observation, a pattern mining algorithm working bottom-up on the DOM tree of a Web page aggregates related content in subtrees. Briefly, the algorithm initially assigns *types*, reflecting similarities in structural presentation, to leaf nodes in the DOM tree and subsequently restructures the tree bottom-up using pattern mining on type sequences. The DOM tree fragment for the page in Figure 2(a) is shown in Figure 3(a). The type of a leaf node is the concatenation of HTML tags on the root-to-leaf path and that of an internal node (or partition) is composed from the types of its child nodes. In the restructured tree, known also as the *partition tree*, there are three classes of partition: (i) *group* - which encapsulates repeating patterns in its immediate children type sequence, (ii) *pattern* - which captures each individual occurrence of the repeat, or (iii) *block* - when it is neither group nor pattern. Intuitively the subtree of a group node denotes homogenous content consisting of semantically related items. For example, observe how all the headline news in the central part in Figure 2(a) are rooted under the group node in the partition tree. The leaf nodes of the partition tree correspond to the leaf nodes in the original DOM tree and have content associated with them. The partition tree resulting from structural analysis of the DOM in Figure 3(a) is shown Figure 3(b). The partition tree represents a logical organization of the page's content.

3. HEARSAY'S BROWSING STRATEGIES

HearSay's dialog creation component takes as input a partition tree constructed from Web page content. It walks over this tree, constructing a menu-based dialog for browsing the content using speech. In previous versions of HearSay, this dialog was constructed using a set of domain-specific VoiceXML templates. Our focus in this version of HearSay is generality; the system should provide a reasonably efficient interface to any Web page. This involves a) permitting efficient navigation; and b) presenting content efficiently. Because the data structure the system operates over is a partition tree, HearSay's general-purpose navigation strate-



(a) DOM (b) partition tree

Figure 3: Structural Analysis of the Page in Fig 2

gies are breadth-first navigation (BFN) and depth-first navigation (DFN). To permit efficient presentation of content, HearSay lets the user change its *verbosity*, or the amount of information it provides about a partition.

3.1 Breadth-First and Depth-First Navigation

In BFN, all the child partitions in a partition are presented to the user, who then selects one for further browsing. This strategy is straightforward and gives users an overview of the available selections from which they can choose. However, if a partition has many children, it can be hard for a user to listen to and remember all the browsing choices. Consider the category news section of the New York Times shown in Figure 4(a). The partition tree of this particular section is shown in Figure 4(b). There are in fact 20 child partitions of this partition, too many for the user to remember [23].

DFN is used in cases like these. In DFN, each child of a partition is presented individually, with the user given a yes/no choice about whether to navigate into that partition right after it is presented. An alternative to DFN would be to use BFN with barge-in, so that a user could interrupt the system with "navigate" right after hearing about a partition of interest. However, with speech input the use of barge-in leads to more speech recognition errors. In addition, with DFN the user never has to listen to the children of a partition more than once (because the system resumes presenting children at the location where the user last made a choice), whereas with BFN+barge-in, the user would have to listen to the whole list of children of a partition at each return to the root of the partition.

Table 1 shows the BFN and DFN dialogs output from HearSay for the category news presented above.

HearSay2 uses a simple method to choose between BFN and DFN based on the number of children a partition has. However, the user can also switch between BFN and DFN at any time to fit personal preference.

3.2 Verbosity

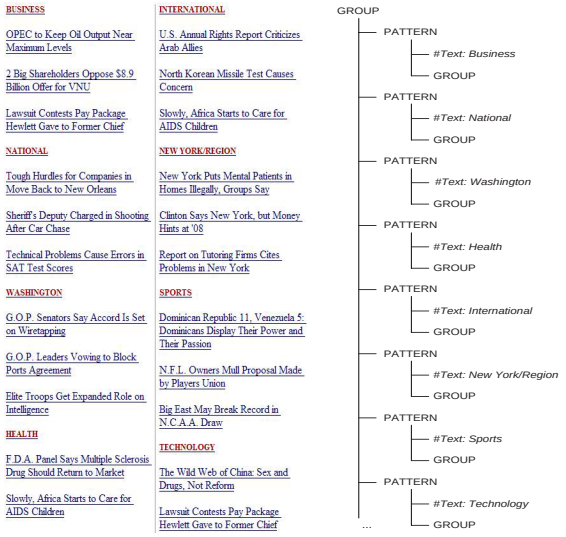
When first browsing a Web page, the user may need lots of information to make navigation choices. However, for expert users this information may be unnecessary and annoying. HearSay now lets users adjust the amount of information it provides about each partition:

Level 1 – non-verbose mode: provide just the depth and type (group, pattern or block) of the partition.

Level 2 – partly-verbose mode: provide all the information from level 1 plus information about the partition's structure *OR* content (see Section 4.2).

Level 3 – full-verbose mode: provide all the information from level 1 plus information about the partition's structure *AND* content.

The user may change the "verbosity level" at any time. Once the level is set, HearSay will use it until it is changed again. A sample dialogue involving changes to the verbosity level is shown in Figure 5.



(a) Category news section (b) The partition tree

Figure 4: Category News from New York Times

HEARSAY2: `You are at level 2. There are 3 sections in this level. Section 1, ...` (presenting the partitions using the default or preset verbose mode.)
USER: (at any time) `Verbose 1.`
HEARSAY2: `Non-verbose mode. Level 2, 3 sections in this level. Section 1, heterogenous content; Section 2, homogenous content, ...`
USER: (at any time) `Verbose 3.`
HEARSAY2: `Full-verbose mode. Level 2, 3 sections in this level. Section 1, heterogenous content (size: 34 percent), first 5 most frequent words are: news, opinions, features, ... It has 12 subsections with labels World News, Sports News, ... The number of words and links are 231 and 34 respectively.`
USER: `...`

Figure 5: Sample HearSay dialog

4. HEARSAY'S PRESENTATION STRATEGIES

In our current version of HearSay, users can set their own browsing strategy (BFN or DFN) and choose the amount of information about the partition tree they want to hear. However, HearSay should also be able to modify its presentation strategy by itself according to the changing requirements imposed by the content it is presenting. In the

BFN	DFN
HEARSAY2: `You are at level 2. There are 20 sections in this level. Section 1, Business; Section 2, National; Section 3, Washington; Section 4, Health ...`(may takes a while to finish all the 20 sections) `please choose from the above 20 sections.` USER: (If she still remembers the second item) `Section 2.` HEARSAY2: `National.` (continue with the national news) USER: `...`	HEARSAY2: `You are at level 2. There are 20 sections in this level. Section 1, Business. Navigate it?` USER: `No.` HEARSAY2: `Section 2, National. Navigate it?` USER: `Yes.` HEARSAY2: `National.` (continue with the national news) USER: `...`

Table 1: Breadth-first navigation vs. depth-first navigation

sections below, we identify three aspects of content presentation in HearSay. We then describe how we used human-annotated data to automatically train classifiers to determine HearSay's content presentation strategy at run time.

4.1 Navigation: Searching vs. Browsing

It is well-known that user activities over Web pages during navigation consist of two basic types: *searching* and *browsing*. Previously, researchers have looked at how users switch between strategies across sequences of Web pages [12, 33]. Here, we apply these ideas to navigation across partitions (possibly within a single Web page).

In the New York Times homepage shown in Figure 6(a), there are basically two big partitions (labeled 1 and 2). Partition 1 is the header of the page, while partition 2 contains the main content. Partition 2 is further divided into three partitions: a menu on the left-hand side, a set of headline news items in the middle, and a set of other news stories and related content on the right-hand side. A visitor to this page looking for news is probably not interested in listening to partition 1 or the menu in partition 2. Instead, the user will search to partition 3, the headline news items. At this point, her activity will turn from searching to browsing, i.e. listening to the news stories.

As illustrated in this example, each partition in one of HearSay's partition trees can be classified as either a *searching* or a *browsing* partition. For browsing partitions, HearSay can simply read out the partition's contents. We created a VoiceXML dialog template to perform this task. However, for searching partitions, HearSay needs to provide a summary of information about the partition so the user can decide whether to search inside it. The type of summary depends on the structure and content of the partition itself.

4.2 Partition Summaries: Structure vs. Content

We distinguish between two basic types of partition summary for *searching* partitions: *structural*, and *content-based*. In a structural summary, HearSay describes the structure of the partition: its location in the partition tree, its size, etc. In a content-based summary, HearSay presents key words in the content of the partition or gives a short extractive summary based on the text in the partition. For partitions containing heterogeneous content, structural summaries are more informative. For partitions containing semantically related items, content-based summaries are more useful.

We designed a VoiceXML dialog template for each type of summary. However, when constructing a content-based summary for a partition, HearSay must decide which text to include.

4.3 Partition Summaries: Content Selection

Common text summarization techniques use term frequencies in documents to identify important words/phrases/ sentences [22]. These documents are usually fairly large, *e.g.* a news report, or an academic paper, etc. However, our partitions are generally small, so extractive summarization methods work poorly. Our summarization method is based on the observation that, in Web pages, visual hints are commonly used to emphasize important elements (*e.g.*, the titles of news articles are usually in big fonts). Each sentence in a *searching, content-based* partition is labeled as either *important* or *unimportant*. The *important* sentences/phrases are used in the text summary for that partition.

4.4 Evaluation

We used machine learning methods to train classifiers for each of these three binary classification tasks:

- Classify each partition as *browsing* or *searching*
- Classify each *searching* partition as best-suited for a *structural* or *content-based* summary
- Classify each sentence/phrase in a *searching/content-based* partition as either *important* or *unimportant*

4.4.1 Partition Types

We collected about 50 partition trees from different Web pages (mainly in the news domain) and manually labeled a subset of partitions selected at random from each partition tree (the trees are very large, so we could not label all the partitions). Each partition was labeled as *browsing*, *searching/structural* or *searching/content-based*. Separately, we automatically extracted 44 features (shown in Table 2) for each partition in each partition tree. These features represent information about the structure of the partition tree and the content in the original Web page.

We built two support vector machines [14] using these features. One classifies partitions as *searching* or *browsing*; the second classifies *searching* partitions as *structural* or *content-based*. We used libSVM [13] to train our classifiers. We used a sequence of binary classifiers rather than a single one-versus-all SVM because: (a) binary classifiers usually have better performance than multi-class classifiers; and more importantly, (b) the two problems do not necessarily follow the same feature-space mapping and distribution.

We used five-fold cross-validation to evaluate the performance of our classifiers for this task. We experimented with different kernels; our best results, obtained using the Radial

(a) (b)

Figure 6: An example from the New York Times

Basis Function kernel, are 86% correct for navigation type (*browsing* or *searching*) and 88% correct for summary type (*content-based* or *structural*).

In HearSay's Dialog Generator, a VoiceXML dialog template is applied to each partition based on its classification and the user-selected verbosity level. When the user selects verbosity level 2, the *content-based* or *structural* template is used for *searching* partitions. Both structural and content-based information are presented for *searching* partitions at verbosity level 3; no description is provided for *searching* partitions at verbosity level 1. The *browsing* template is used for *browsing* partitions at all verbosity levels.

4.4.2 Partition Summaries

We manually labeled 700 leaf nodes from our 50 partition trees. Each node was labeled as *important* or *unimportant*. Separately, we automatically extracted the the 13 features given in Table 3 for each node. These features include information about the position of the leaf node as well as the formatting information.

We trained decision tree classifiers for this task using Weka [32] implementations of the decision tree algorithms (*i.e.*, J48[2], ADTree[16], NBTree[19] and LMT[20]). As a baseline, we took the first sentence/phrase in a partition as the summary of that partition.

For testing, we used 10 partition trees from 10 new news Web sites. Our results are shown in Table 4. The best decision tree (*i.e.*, J48) works significantly better than the baseline for partitions where there are clear visual hints. However, we found that performance on this task is highly dependent on the performance of the underlying partitioning algorithm. For example, sometimes boundaries of par-

[2]Weka's version of C4.5[28].

No.	Name	Description
1-5	NODE_T	The type of the partition.
6-11	PAR_T	The type of the parent.
12-17	PREV_T	The type of the left sibling.
18-22	NEXT_T	The type of the right sibling.
23	NUM_W	Number of words.
24	NUM_LK	Number of links.
25	NUM_L	Number of leaf nodes.
26	NUM_CH	Number of direct children.
27	NUM_LS	Number of left siblings.
28	NUM_RS	Number of right siblings.
29	LEVEL	Number of levels to the root.
30	LONG_LV	Number of levels to the deepest leaf node.
31	SHORT_LV	Number of levels to the nearest leaf node.
32-36	NUM_CT	Number of children with the same type.
37	TOTAL_W	Total number of words in the tree.
38	PAR_W	Total number of words in the parent.
39	TOTAL_L	Total number of leaf nodes in the tree.
40	PAR_L	Total number of leaf nodes in the parent.
41	MAX_LV	Level of the deepest branch in the tree.
42	PAR_MAX	Level of the deepest branch in the parent.
43	MIN_LV	Level of the most shallow branch in the tree.
44	PAR_MIN	Level of the most shallow branch in the parent.

Table 2: Features for SVM.

No.	Description
1-3	The node type of its parent and siblings.
4	Number of words in the current node.
5	Number of words in the parent node.
6-7	Number of previous and next siblings.
8	Number of levels to the root.
9	Max number of levels to the root from siblings.
10-13	Font style.

Table 3: Features used in labeling problem.

Web site	Baseline	J48	ADTree	NBTree	LMT
NYTimes	84%	100%	97%	92%	81%
CNN	71%	80%	74%	80%	69%
GoogleNews	78%	100%	100%	97%	97%
LATimes	85%	88%	88%	88%	78%
MSNBC	81%	81%	81%	81%	81%
Average	79.8%	89.8%	89.2%	85.6%	81.2%

Table 4: Primary evaluation for labeling problem.

titions were incorrect, so that *important* sentences were in the wrong partition. Example rules learned by the classifiers are,

- *if the element is among first three elements in this partition, the number of levels to the root of this partition is less than 3, and its font size is bigger than 2, then it is important.*
- *if the element is a link under a pattern node, it is the first element, or it is the second element but the font size is bigger than 1, then it is important.*

In Table 5, we show part of a dialog generated from the Web page of Figure 6(a). Switches from one dialog strategy to another are annotated in the dialog.

5. RELATED WORK

The issue of promoting Web accessibility for persons with visual disabilities has become increasingly prominent. As early as 1997, W3C launched the Web Accessibility Initiative (WAI) [10] to promote the development of browser standards and guidelines (*e.g.*, HTML authoring guidelines and user agent guidelines) to make the Web more accessible to individuals with visual disabilities. Similar initiatives have been developed by industry: examples include Microsoft's accessibility initiative [1], IBM's Special Needs Systems program [4], and Sun Microsystem's Java accessibility API [5].

Several studies have highlighted the ineffectiveness of existing screen readers for Web browsing tasks [15, 17]. As a result, several specialized Web voice browsers have been developed to adapt HTML-based contents. For example, the JAWS system [6] and IBM's Home Page Reader [3] allow navigation via hyperlinks. The BrookesTalk system [2] uses NLP-base text abstracting and summarization techniques to facilitate voice browsing of the Web. The voice feature in the Opera browser [7] provides a set of voice-based commands.

A key difference between HearSay and these systems is that HearSay performs extensive analysis of the content of HTML documents, while other systems do minimal processing of Web page content. Content-based analysis enables segmentation of Web page content into related blocks and facilitates efficient speech-driven browsing.

As accessibility becomes a more mature research area, the design of specialized voice-driven interfaces for is receiving more attention (e.g. [31, 18]). For example, in [30] the authors examined the accessibility issues relating to the Web and proposed solutions in the context of a screen reader system. More guidelines relating to improving the accessibililty of search engines were proposed in [21]. However, in most research relating to accessibility the proposed solution is for content authors to add additional tags to Web page content, or for the engineers of the browser to provide specialized ontologies and rules to facilitate content presentation [29, 27]. By contrast, we take the markup of content on the Web as is, and use automatic analysis to make the content accessible. Our approach is to provide efficient access to as much of the Web as possible; we are willing to sacrifice some elegance for coverage.

6. CONCLUSION AND FUTURE WORK

We have described our new HearSay Web browser for people with visual disabilities. HearSay is designed for efficient, broad-coverage voice-driven Web browsing. In this paper, we focused on the general-purpose browsing and content presentation strategies employed in HearSay. In future work, we plan to conduct a complete evaluation of HearSay and refine our browsing and presentation strategies accordingly.

7. ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under grant number IIS-0534419 and by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010. The authors would like to thank the anonymous reviewers for their very helpful feedback.

8. REFERENCES

[1] Accessibility at Microsoft. http://www.microsoft.com/enable/.
[2] BrookesTalk. http://cms.brookes.ac.uk/computing/speech/.
[3] IBM Home Page Reader. http://www-306.ibm.com/able/.
[4] IBM human ability and accessibility center. http://www-306.ibm.com/able/.
[5] Java accessibility techonologies. http://java.sun.com/products/jfc/accessibility/index.jsp.
[6] JAWS Screen Reader. http://www.freedomscientific.com.
[7] OPERA browser with voice. http://www.opera.com/voice/.
[8] Speech Application Language Tools. http://www.saltforum.org/.
[9] VoiceXML 2.0. http://www.w3.org/TR/voicexml20/.
[10] Web Accessibility Initiative(WAI). http://www.w3.org/WAI.
[11] XHTML+Voice profile 1.2. http://www.voicexml.org/specs/multimodal/x+v/12/.
[12] L. D. Catledge and J. E. Pitkow. Characterizing browsing strategies in the World Wide Web. *Computer Networks and ISDN Systems*, 27(6):1065–1073, 1995.
[13] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines, 2001. http://www.csie.ntu.edu.tw/ cjlin/libsvm.

(Browsing strategy: BFN. Verbosity level 2.)

(Partition type: Searching. Structural summary.)
HEARSAY: Level 2, 3 sections.
Section 1 (size: 13 percent) has 169 words and 51 links. Frequent words are News, Opinion, Features, Services, Member.
Section 2 (size: 63 percent) has 819 words and 133 links. Frequent words are International, National, Washington, Business, Editor, Complete Coverage.
Section 3 (size: 24 percent) has 312 words and 39 links. Frequent words are Market, Complete News, Forum, World, Reviews.
Please choose from the three sections.

USER: Section 2.

(Browsing strategy: DFN.)

HEARSAY2: Level 3, 6 sections.
Section 1 (size: 18 percent) has 147 words and 8 links. Frequent words are court, prosecutors, sunni, Iraq, centers. Navigate it?

USER: Yes.

(Browsing strategy: BFN.)
(Partition type: searching. Content-based summary.)

HEARSAY2: Level 4, 4 sections.
Section 1 is about Plant Operators Must Install Pollution Controls, Court Rules.
Section 2 is about Judge Gives Prosecutors New Chance in Terror Case.
Section 3 is about Sunnis Denounce Plan by U.S. and Iran to Hold Talks on Iraq.
Section 4 is about New Yorkers Will Pay to Sweat at 6 Previously Free Centers.
Please choose from the four sections.

USER: Section 3.

(Partition type: browsing.)

HEARSAY2: Level 5. Link: Sunnis Denounce Plan by U.S. and Iran to Hold Talks on Iraq. By KIRK SEMPLE, 7:11 PM ET. Sunni leaders today said the talks would amount to "unjustified interference" by foreign nations in Iraq's domestic affairs. Link: Kurds Destroy Shrine, Navigate it?

USER: No.

HEARSAY2:(continue) Link: Photos, Navigate it?

USER: No.

HEARSAY2:(continue) Link: Complete Coverage: The Reach of War, Navigate it?

USER: No.

HEARSAY2:(continue) You've finished browsing this section. Now going back to upper level.
Level 4, 4 sections.
Section 1 is about Plant Operators Must Install Pollution Controls, Court Rules.
(repeating the description of block 3)

...

Table 5: Sample HearSay dialog based on the Web page in Figure6(a)

[14] N. Cristianini and J. Shawe-Taylor. *An Introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[15] C. Earl and J. Leventhal. A survey of windows screen reader users: Recent improvements in accessibility. *Journal of Visual Impairment and Blindness*, 93(3), 1999.

[16] Y. Freund and L. Mason. The alternating decision tree learning algorithm,. In *Proc. 16th International Conf. on Machine Learning*, pages 124–133. Morgan Kaufmann, San Francisco, CA, 1999.

[17] J. Gunderson and R. Mendelson. Usability of world wide web browsers by persons with visual impairments. In *Proceedings of the RESNA Annual Conf.*, 1997.

[18] P. Karampiperis and D. Sampson. Designing learning systems to provide accessible services. In *W4A '05: Proceedings of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A)*, pages 72–80, New York, NY, USA, 2005. ACM Press.

[19] R. Kohavi. Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 202–207, 1996.

[20] N. Landwehr, M. Hall, and E. Frank. Logistic model trees, 2003.

[21] B. Leporini, P. Andronico, and M. Buzzi. Designing search engine user interfaces for the visually impaired. *SIGCAPH Comput. Phys. Handicap.*, (76):17–18, 2003.

[22] I. Mani and M. T. Maybury. *Advances in Automatic Text Summarization*. MIT Press, 1999.

[23] G. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psycological Review*, 63:81–97, 1956.

[24] S. Mukherjee, I. Ramakrishnan, and A. Singh. Bootstrapping semantic annotation for content-rich HTML documents. In *Intl. Conf. on Data Engineering (ICDE)*, 2005.

[25] 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.

[26] 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.

[27] 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, New York, NY, USA, 2005. ACM Press.

[28] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

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

[30] M. F. Theofanos and J. G. Redish. Bridging the gap: between accessibility and usability. *Interactions*, 10(6):36–51, 2003.

[31] Z. Trabelsi, S.-H. Cha, D. Desai, and C. Tappert. A voice and ink XML multimodal architecture for mobile e-commerce systems. In *WMC '02: Proceedings of the 2nd international workshop on Mobile commerce*, pages 100–104, New York, NY, USA, 2002. ACM Press.

[32] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition edition, 2005.

[33] G. Xu, A. Cockburn, and B. McKenzie. Lost on the Web: An introduction to Web navigation research. In *The 4th New Zealand Computer Science Research Students Conference*, 2001.