HearSay: Enabling Audio Browsing on Hypertext Content

I. V. Ramakrishnan
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400
ram@cs.stonybrook.edu

Amanda Stent
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400
stent@cs.stonybrook.edu

Guizhen Yang
Department of Computer Science and Engineering
University at Buffalo
Buffalo, NY 14260-2000
gzyang@cse.buffalo.edu

ABSTRACT

In this paper we present HearSay, a system for browsing hypertext Web documents via audio. The HearSay system is based on our novel approach to automatically creating audio browsable content from hypertext Web documents. It combines two key technologies: (1) automatic partitioning of Web documents through tightly coupled structural and semantic analysis, which transforms raw HTML documents into semantic structures so as to facilitate audio browsing; and (2) VoiceXML, an already standardized technology which we adopt to represent voice dialogs automatically created from the XML output of partitioning. This paper describes the software components of HearSay and presents an initial system evaluation.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures; H.4.3 [**Information Systems Applications**]: Communications Applications—*information browsers*

General Terms

Algorithms, Design, Experimentation

Keywords

World Wide Web, HTML, VoiceXML, structural analysis, semantic analysis, audio browser, user interface

1. INTRODUCTION

As the World Wide Web has matured, an enormous amount of information, ranging from news to entertainment to scientific discoveries, has been made readily accessible to the general public. Recently Google announced that its search engine has indexed 4.3 billion Web pages. Furthermore, it is estimated that the Web is still constantly growing at a rapid pace, doubling in size every 8 months. However, most Web documents are designed primarily for graphical display and interaction. This means that individuals with visual disabilities and/or highly mobile individuals have reduced access to the many resources the Web provides.

Creating audio browsable Web content has become the focus of intensive research efforts by industrial enterprises (*e.g.*, IBM, AT&T, Microsoft) and standardization organizations (*e.g.*, W3C). New markup languages, such as VoiceXML [4] and SALT [3], and new voice browser systems, such as IBM's WebSphere Voice Server and Lucent Technologies' MiLife VoiceXML Gateway, are emerging to facilitate the creation, publishing, and exchange of audio browsable Web content.

WWW2004, May 17–22, 2004, New York, New York, USA.
ACM 1-58113-844-X/04/0005.

However, adapting to voice browser technology still remains a significant burden for many Web content providers. In order to deliver their content to different browsers with different visual and audio modalities, Web content providers have to design and encode their Web content in different forms using various standard markup languages such as HTML, VoiceXML, and WML [5] — a tremendous development and maintenance overhead. Such diversity in content formats has become an obstacle to the wide adoption of voice browser technology by Web content providers.

Although voice browsers that can read HTML documents exist (*e.g.*, [2]), they do not adequately convey to users the logical structure and semantics of content in Web documents, nor do they provide users with easy ways to select which parts of a document to listen to. Consequently, users who are visually disabled and/or mobile waste a considerable amount of time and attention listening to irrelevant information.

Our goal is to develop intelligent systems that can enable people to interact directly, quickly, and easily with Web content using non-visual modalities, primarily audio. In this paper we present the HearSay system, a prototype audio Web browser we have developed. HearSay is based on a novel technology, *partitioning*, for structural and semantic analysis of HTML documents. HearSay is completely speech driven. It supports interactive *exploratory browsing*, presenting information about an HTML document's logical structure and content and permitting users to select which parts of the document to listen to and when to navigate to a new page. The key ideas underlying this system are:

1. Novel structural analysis techniques that leverage the structural annotations of HTML documents to partition Web content into semantic units, which are organized and then presented to users as a concept hierarchy (or *partition tree*). We have also developed efficient heuristics to identify semantic labels for nodes in the partition tree.

2. Novel semantic analysis techniques that are tightly coupled with structural analysis. We use lexical associations derived from WordNet, combined with ontologies, to more accurately separate semantic units and to assign informative labels to partitions. In addition, we have proposed effective heuristics to propagate semantic associations based on structural cues in HTML documents, further improving the performance of our techniques.

3. Template-based automatic VoiceXML dialog creation using partition trees. This permits on-the-fly generation of dialogs for efficient, user-directed interactive audio browsing. An

initial evaluation of this system has been conducted to demonstrate its utility for real-life browsing tasks and identify areas for further improvement.

The rest of this paper is organized as follows. In Section 2 we discuss related work. A use scenario of the HearSay system is presented in Section 3. Section 4 presents the software architecture of HearSay and gives an overview of our novel partitioning algorithms. In Section 5 we describe an initial system evaluation. We conclude in Section 6.

2. RELATED WORK

2.1 Audio Browsers

The issue of promoting Web accessibility for persons with visual disabilities has become increasingly prominent. In 1997, W3C launched the Web Accessibility Initiative (WAI) [26] to promote the development of browser standards and guidelines (*e.g.*, HTML authoring guidelines and authoring tool 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 [29], IBM's Special Needs Systems program [25], and Sun Microsystem's Java accessibility API [32].

Several studies have highlighted the ineffectiveness of existing screen readers for Web browsing tasks [13, 19]. As a result, several specialized Web audio browsers have been developed [31, 27]. For example, the JAWS [1] system and IBM's Home Page Reader [8, 7, 30] allow navigation via hyperlinks. The BrookesTalk system uses NLP-based text abstracting and summarization techniques to facilitate audio browsing of the Web [38]. Other systems permit VoiceXML browsing of Web pages [2].

HearSay is different from these systems in both scope and approach. In particular, HearSay performs structural and semantic analysis of HTML documents and automatically creates VoiceXML dialogs from the partition trees it discovers, facilitating audio browsing without information overload. HearSay takes a comprehensive, content-based approach to audio Web browsing, giving greater flexibility and usability without sacrificing performance.

Finally, we should point out that HearSay will benefit not only people with visual disabilities but users of small-form devices such as PDAs as well.

2.2 Web Page Analysis

The idea of analyzing Web documents in order to make them amenable for browsing by persons with visual disabilities was explored in [33]. In this work, parts of HTML documents deemed "interesting" are marked by hand. The XPaths to these selected blocks are stored in a repository. When the site server receives a request for a document, the stored XPaths are matched against its corresponding DOM tree to extract the "interesting" blocks. In this approach, if the document organization changes the annotation must be redone. Other researchers have proposed the idea of extracting "interesting" content using semantics [24]. The problem with these approaches is that the identification of "interesting" content is done by hand (and so takes time and is not easily scalable), and is site-specific (and so not generalizable).

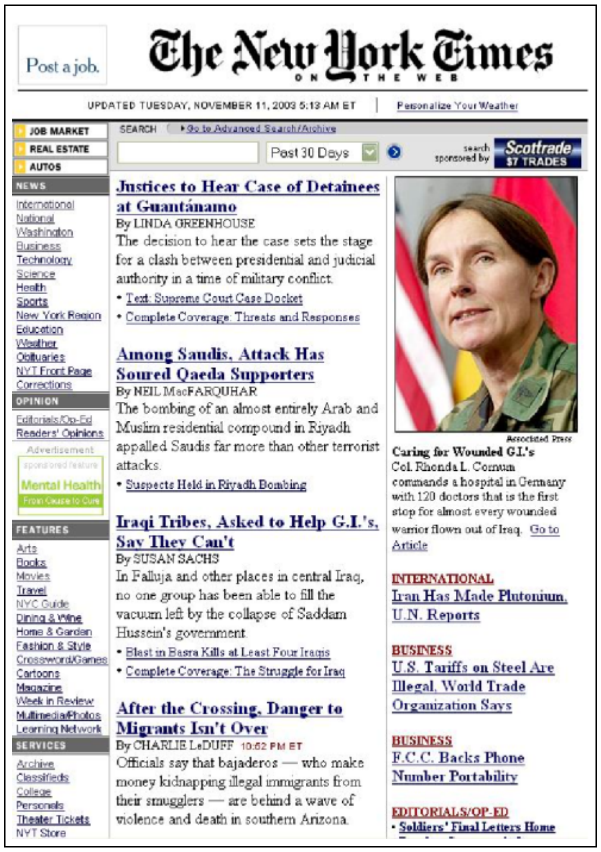
A significant amount of research has been done recently on the topic of segmenting Web documents [15, 16, 14, 10, 11, 36, 37]. Our technology differs from these techniques in three important ways. First, these techniques are either domain-specific [15, 14] or depend on *ad hoc* interpretation of a fixed set of HTML markups [16, 11, 36, 37, 10]. Our partitioning techniques can be applied to *any* domain. Second, for these techniques to be robust, a substantial amount of human effort has to be spent on tuning system parameters (manually-coded ontologies [15, 14] or hard-wired threshold values [16, 11, 36]) for specific domains. Our structural analysis techniques are fully automated and domain-independent, while the domain-dependent part — semantic analysis, and in particular, labeling of partitions — can be automated by training classifiers and mining ontologies from the Web (see Section 6 for future work). Finally, the problem of assigning semantic labels to partitions was not addressed in [36, 37].

Some interesting recent research has focused on enriching Web documents with semantic labels [23, 21, 22, 12]. In [23, 21, 22] powerful ontology management systems form the backbone of systems that support *interactive* annotation of HTML documents. This is in contrast to our approach where annotation is *automatic*. Another automatic approach to Web document annotation is described in [12]. However, the techniques proposed in [12] do not make use of the structural markups of Web documents. Consequently, their partitioning algorithm may fail on content-rich HTML documents containing multiple concepts and multiple concept instances for each concept.

In summary, our approach to partitioning and labeling of Web pages uses tightly coupled, automatic structural and semantic analysis techniques to obtain concept hierarchies (or *partition trees*) from HTML documents.

3. A USE SCENARIO

In this section we will illustrate the intended use of HearSay. Suppose Alice is a blind student who is taking a class on current affairs. She often browses newspaper Web sites during her commute



Justices to Hear Case of Detainees at Guantánamo
By LINDA GREENHOUSE
The decision to hear the case sets the stage for a clash between presidential and judicial authority in a time of military conflict.
• Text: Supreme Court Case Docket
• Complete Coverage: Threats and Responses

Among Saudis, Attack Has Soured Qaeda Supporters
By NEIL MacFARQUHAR
The bombing of an almost entirely Arab and Muslim residential compound in Riyadh appalled Saudis far more than other terrorist attacks.
• Suspects Held in Riyadh Bombing

Iraqi Tribes, Asked to Help G.I.'s, Say They Can't
By SUSAN SACHS
In Falluja and other places in central Iraq, no one group has been able to fill the vacuum left by the collapse of Saddam Hussein's government.
• Blast in Basra Kills at Least Four Iraqis
• Complete Coverage: The Struggle for Iraq

After the Crossing, Danger to Migrants Isn't Over
By CHARLIE LeDUFF 10:52 PM ET
Officials say that bajaderos — who make money kidnapping illegal immigrants from their smugglers — are behind a wave of violence and death in southern Arizona.

Caring for Wounded G.I.'s
Col. Rhonda L. Cornum commands a hospital in Germany with 120 doctors that is the first stop for almost every wounded warrior flown out of Iraq. Go to Article

INTERNATIONAL
Iran Has Made Plutonium, U.N. Reports

BUSINESS
U.S. Tariffs on Steel Are Illegal, World Trade Organization Says

BUSINESS
F.C.C. Backs Phone Number Portability

EDITORIALS/OP-ED
• Soldiers' Final Letters Home

Figure 1: New York Times Front Page

```
<form id="home">
  <field name="choice">
    <prompt>
      Alice, please choose one of these:
      Headline News. News. Opinion. Exit.
    </prompt>
    <grammar> Headline News | News | Opinion | ... | Exit </grammar>
    <filled>
      <if cond="choice=='Headline News'"><goto next="#headline_news"/>
      <elseif cond="choice=='News'"/><goto next="#news"/>
      <elseif cond="choice=='Opinion'"/><goto next="#opinion"/>
      ...
      <elseif cond="choice=='Exit'"/><exit/>
      </if>
    </filled>
  </field>
</form>
```

Figure 2: VoiceXML Dialog for New York Times Front Page

in order to prepare for in-class discussions. Today she browses the Web site of the New York Times (http://nytimes.com) using her cell phone. Alice chooses to use HearSay's voice interface. First she speaks the URL to HearSay. After retrieving the New York Times front page (see Figure 1), HearSay performs structural and semantic analysis on this Web page to determine semantically related units in it. The result is a semantic partition tree as shown in Figure 3.

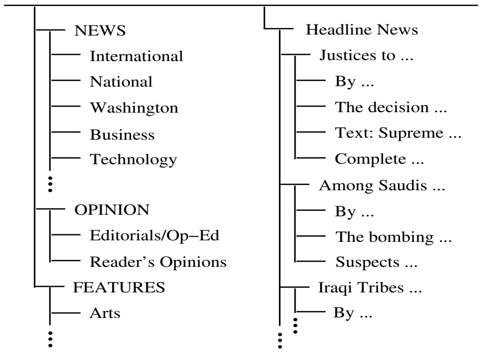
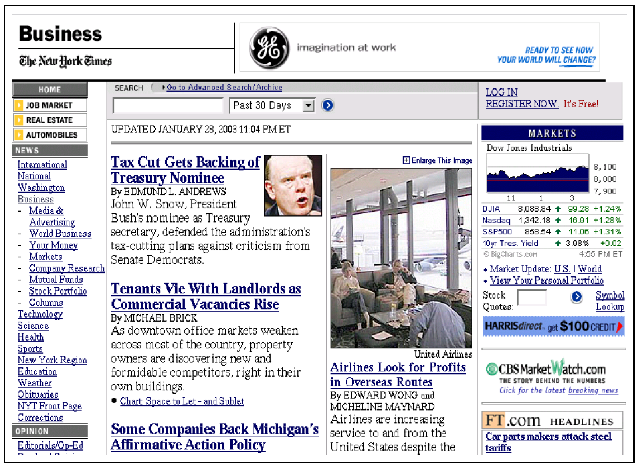


Figure 3: Semantic Partition Tree for Figure 1

Each item in a semantic partition has a label and a type (*e.g.*, navigation link, text, form element). For instance, all the items under the partition labeled "NEWS" in Figure 3 are those links under the "NEWS" category in the news taxonomy of New York Times (upper left corner in Figure 1). Similarly, all the items in the partition labeled "Headline News" are the headline news items in the New York Times front page (center portion of Figure 1). Examples of such items are shown in Figure 3.

HearSay also automatically generates a speech dialog interface to the semantic partition tree. Part of a sample VoiceXML dialog for the partition tree in Figure 3 is shown in Figure 2. Alice can now browse this site using audio. HearSay reads out the labels of top-level partitions ("NEWS", "OPINION", ..., "Headline News", ..., etc.), pausing briefly after each item unless Alice chooses to speed the system up. Alice can pick an item at any time, by saying the label or the partition number. If Alice says "News" (or "Item 1"), HearSay reads out the label and type of each item in the "NEWS" partition, including the fact that all the items it contains are links, and the name of each link. If Alice says "Business", HearSay will follow this link to the business section of the New York Times (Figure 4) and partition the resulting document as shown in Figure 5 where Alice can select an article to listen to. At any point Alice can also say any one of a set of browsing commands, such as "Start over", "Repeat" or "Stop".


Business
The New York Times
imagination at work
READY TO SEE HOW YOUR WORLD WILL CHANGE?
HOME
JOB MARKET
REAL ESTATE
AUTOMOBILES
NEWS
International
National
Washington
Business
- Media & Advertising
- World Business
- Your Money
- Markets
- Company Research
- Mutual Funds
- Stock Portfolio
- Columns
Technology
Science
Health
Sports
New York Region
Education
Weather
Obituaries
NYT Front Page
Corrections
OPINION
Editorials/Op-Ed

SEARCH Go to Advanced Search/Archive
Past 30 Days
UPDATED JANUARY 28, 2003 11:04 PM ET

Tax Cut Gets Backing of Treasury Nominee
By EDMUND L. ANDREWS
John W. Snow, President Bush's nominee as Treasury secretary, defended the administration's tax-cutting plans against criticism from Senate Democrats.

Tenants Vie With Landlords as Commercial Vacancies Rise
By MICHAEL BRICK
As downtown office markets weaken across most of the country, property owners are discovering new and formidable competitors, right in their own buildings.
• Chart: Space to Let - and Sublet

Some Companies Back Michigan's Affirmative Action Policy

Enlarge This Image
United Airlines
Airlines Look for Profits in Overseas Routes
By EDWARD WONG and MICHELINE MAYNARD
Airlines are increasing service to and from the United States despite the

LOG IN
REGISTER NOW. It's Free!
MARKETS
Dow Jones Industrials
8,100
8,000
7,900
11 1 3

DJIA	8,088.84 ⬆	99.28	+1.24%
Nasdaq	1,342.18 ⬆	16.91	+1.28%
S&P500	858.54 ⬆	11.06	+1.31%
10yr Tres. Yield	⬆	3.98%	+0.02

© BigCharts.com 4:55 PM ET
• Market Update: U.S. | World
• View Your Personal Portfolio
Stock Quotes: Symbol Lookup
HARRISdirect. get $100 CREDIT
CBS MarketWatch.com THE STORY BEHIND THE NUMBERS
Click for the latest breaking news
FT.com HEADLINES
Car parts makers attack steel tariffs

Figure 4: New York Times Business Page

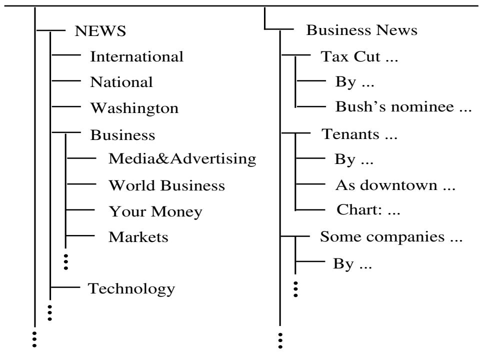


Figure 5: Semantic Partition Tree for Figure 4

4. ENABLING AUDIO BROWSING

4.1 Voice Browser Architecture

The software architecture of HearSay is depicted in Figure 6. The main function of the *partition generator* module is to partition an input HTML document into semantically related segments by performing structural and semantic analysis. The semantic analysis component uses knowledge resources, such as lexicons (*e.g.*, WordNet [6]), and domain ontologies that encode the salient characteristics of commonly accepted concepts in the particular domain. (Example domains are news, consumer electronics, etc.)

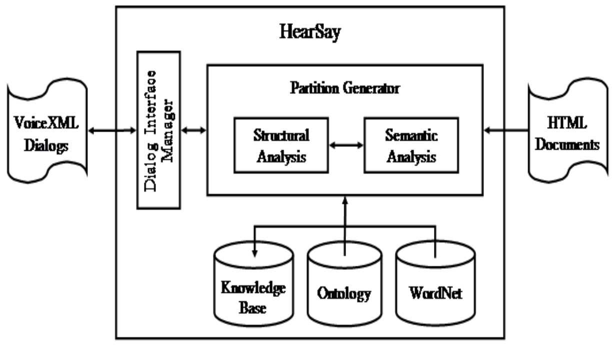


Figure 6: Architecture of HearSay

Users interact with HearSay through the *dialog interface manager*. We do not explicitly include in the HearSay architecture modules for text-to-speech conversion and speech recognition, as this functionality is provided by a number of Web browsers that support the VoiceXML standard. Our current implementation is built on top of IBM's WebSphere toolkit which includes text-to-speech and speech recognition components.

4.2 Partitioning

The partition generator takes an HTML document as input. It performs structural and semantic analysis on the DOM tree of this HTML document and outputs a partition tree (represented in XML), which captures the logical structure of the original input HTML document.

The key observation used in the partition generator is that *semantically related items in an HTML document normally exhibit consistency in presentation style and spatial locality*. This is particularly true of content-rich Web sites that change frequently such as news portals and e-commerce sites, because these sites are typically maintained using content management software that creates HTML documents by populating templates from backend databases.

For example, in the New York Times front page (shown in Figure 1), there is a fixed news taxonomy on the upper left corner. There is also an implicit template for major headline news items. Each of these items begins with a hyperlink labeled with a news headline (*e.g.*, "Among Saudis ...") followed by the news source (*e.g.*, "By NEIL ..."), followed by an optional timestamp, text summary of the article (*e.g.*, "The bombing ..."), and some pointers to related news (*e.g.*, "Suspects Held ...").

There are also presentation similarities in the items in the news taxonomy of the New York Times front page (Figure 1). The main taxonomic items, "NEWS", "OPINION", "FEATURES", ..., etc., are all presented in bold font. All the subtaxonomic items (*e.g.*, "International", "National", "Washington", ..., etc.) under a main taxonomic item (*e.g.*, "NEWS") are hyperlinks. This kind of consistency in presentation style has a very strong manifestation in the DOM tree of an HTML document. For example, Figure 7 depicts a fragment of the DOM tree for the New York Times front page shown in Figure 1. The root-to-leaf sequences of HTML tags for the nodes "NEWS" and "FEATURES" are exactly the same, so are the sequences of HTML tags for the nodes "International", "Arts", ..., etc. (*font* tags with different attributes, *e.g.*, size, are distinguished using different subscripts in Figure 7).

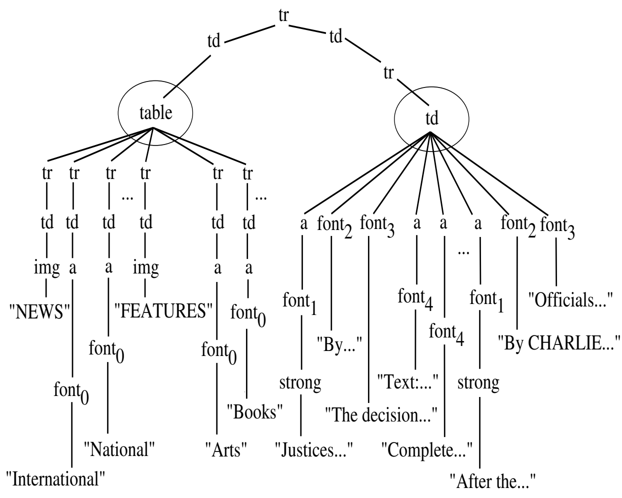


Figure 7: DOM Tree Fragment for Figure 1

Together with consistency in presentation style, spatial locality can also be commonly observed in a Web page and its corresponding DOM tree. For example, when rendered in a browser (see Figure 1), all the taxonomic items for the New York Times are placed in close vicinity occupying the upper left portion of the page. In the DOM tree (see Figure 7) corresponding to this HTML document, all these taxonomic items are grouped together under one *single* subtree rooted at the *table* node (shown in circle). Similarly, all the major headline news items are clustered under a different subtree rooted at the *td* node (shown in circle in Figure 7).

4.2.1 Structural Analysis

Our structural analysis of an HTML document is based on the key observations mentioned above. First, the observation about consistency in presentation style leads to the idea of associating a *type* (explained below) with each leaf node in a DOM tree. Second, the observation about spatial locality gives rise to the idea of discovering structural recurrence patterns for semantically related items and propagating types bottom-up in the DOM tree.

In HearSay, we exploit a simple but effective type system to encode information about presentation styles and structural recurrences. Types in our system are either primitive types or compound types in the form of $seq(T_1 \ldots T_n)$ where each T_i is a type. Each leaf node in a DOM tree is associated with a primitive type, which concatenates all the HTML tags on the root-to-leaf path to this node. Intuitively, a primitive type encodes the presentation style (including location and visual cues such as font type and size) of a piece of text that corresponds to a leaf node in a DOM tree.

For example, in the DOM tree fragment of Figure 7 (which corresponds to the Web page shown in Figure 1) all the leaf nodes corresponding to the main taxonomic items, "NEWS", "OPINION", "FEATURES", ..., etc., have the same primitive type, T_1: $tr \cdot td \cdot table \cdot tr \cdot td \cdot img$. All the subtaxonomic items under each main taxonomic item, such as "International", "National", ..., etc., under the "NEWS" item, also share a primitive type, T_2: $tr \cdot td \cdot table \cdot tr \cdot td \cdot a \cdot font_0$.

A compound type essentially summarizes the structural recurrence information of a subtree rooted at an internal node. Note that in Figure 7 the subtree rooted at the $table$ node (circled) groups together several main taxonomic items each of which is followed by a number of subtaxonomic items, *i.e.*, the entire taxonomy is clustered under this *single* DOM subtree. This property of spatial locality combined with consistency in presentation style reveals structural recurrence information about semantically related items, which can be denoted by $T_1T_2T_2 \ldots T_1T_2T_2 \ldots$. In this string the *sequential pattern*, $T_1T_2^*$ (here $*$ denotes Kleene closure), exactly captures the structural recurrence information of each semantic unit (*i.e.*, a main taxonomic item followed by a number of subtaxonomic items). In our type system, the sequential pattern $T_1T_2^*$ is *generalized* as the compound type $seq(T_1T_2)$, which is assigned as the type of the $table$ node (circled) in Figure 7.

As illustrated by the example above, the idea underlying our approach to structural analysis is to gather sequential patterns in the type sequence of nodes in a DOM tree in a bottom-up fashion. Given any two types as defined above, their equivalence is defined straightforwardly: two types are equivalent if and only if they are syntactically the same. Our top-level partitioning algorithm is outlined in algorithm *PartitionTree*.

```
**Algorithm** PartitionTree($n$)
**input**
    $n$ : a node in a DOM tree
**begin**
1.  **if** $n$ is a leaf node **then**
2.      $n.type$ = the sequence of HTML tags from the root to $n$
3.  **else if** $n$ has only one child node $c$ **then**
4.      PartitionTree($c$)
5.      Replace $n$ with $c$ and remove $n$ from the DOM tree.
6.  **else**
7.      **for** each child node $x$ of $n$ **do** PartitionTree($x$) **endfor**
8.      Analyze($n$)
9.  **endif**
**end**
```

To transform the DOM tree of an HTML document into a semantic partition tree, algorithm *PartitionTree* is invoked on the root of the DOM tree. The algorithm first traverses the DOM tree top-down and then restructures it bottom-up. We use the notation $n.type$ to refer to the type attribute of a node n. In algorithm *PartitionTree*, Line 2 assigns primitive types to all leaf nodes. Internal nodes with only one child are handled in Lines 4–5. In such a case, the type of this single child node is computed and then simply propagated up the tree. However, for an internal node with multiple children, algorithm *PartitionTree* is first invoked on each of its children to compute their type information (Line 6). Then algorithm *Analyze* performs a pattern discovery on the sequence of types of its child nodes (Line 7).

Algorithm *Analyze* takes an internal node, n, as input. Its main function is to partition the child nodes of n by examining their structural similarity.There are two main stages in algorithm *Analyze*. The first stage is an *iterative* process for discovering structural similarities among the types of child nodes, where each iteration involves collapsing consecutive, identical types into one (to group repeated items) and then mining the so called *maximal repeating substrings* (to find structural recurrences). Essentially, a maximal repeating substring is a *repeating* substring that covers a *majority* of elements in a sequence.[1] In addition, its coverage should be maximized and its length minimized (under the prerequisite that its coverage be maximized). In our algorithms we use suffix trees [20] to efficiently mine maximal repeating substrings.

In the second stage of algorithm *Analyze*, the last pattern discovered during the first stage is used to partition the remaining sequence of nodes further. Here we use a simple heuristic to handle variations in document structures (*e.g.*, missing data items): we always try to associate the nodes between partitions with the left partition. (Later we will show how this simple heuristic can be enhanced by semantic analysis.)

[1]Normally the support threshold value is set to 30%-50% in our system. It is used to factor in structural variations.

```
**Algorithm** Analyze($n$)
**input**
    $n$ : an internal node in a DOM tree
**begin**
1.  $S$ = the sequence of all the child nodes of $n$
2.  **for** each node $c$ in $S$ **do**
3.      **if** $c.flatten = true$ **then**
4.          Replace $c$ with the sequence of all the child nodes of $c$.
5.      **endif**
6.  **endfor**
7.  $\tau = \varepsilon$
8.  **do**
9.      Collapse adjacent nodes in $S$ which share the same type.
10.     $\alpha$ = MaximalRepeatingSubstring(TypeStr($S$))
11.     **if** $\alpha \neq \varepsilon$ **then** $\tau = \alpha$ **endif**
12.     **if** $|\alpha| > 1$ **then**
13.         **for** each substring $\rho$ in $S$ such that TypeStr($\rho$) = $\alpha$ **do**
14.             Replace $\rho$ with NewNode($\rho$,$seq(\alpha)$).
15.         **endfor**
16.     **endif**
17. **while** $|\alpha| > 1$
18. **if** $\tau = \varepsilon$ **then**
19.     $n.flatten = true$
20. **else**
21.     Partition $S$ into $\beta_0\gamma\beta_1 \ldots \gamma\beta_m$, where TypeStr($\gamma$) = $\tau$.
22.     **for** each $\gamma\beta_i$ **do** Replace $\gamma\beta_i$ with NewNode($\gamma\beta_i$, NewType($\tau$)). **endfor**
23.     $n.type$ = NewType($\tau$)
24. **endif**
25. Make the nodes in $S$ the new children of $n$.
**end**
```

The process for discovering structural similarity may not always succeed, since our algorithm traverses a DOM tree bottom-up and multiple semantically related items can be dispersed in several subtrees. Therefore, the "true" sequential pattern may not be evident until our algorithm is invoked on an internal node that is close enough to the root. If structural similarity is not found at a node, then the type information of all its child nodes is simply propagated up the tree.

Now we illustrate the working steps of algorithm *Analyze* using an example. For simplicity, we will just show how it manipulates a sequence of types and omit other details. Suppose algorithm *Analyze* is invoked on a node S with the sequence of types of its child nodes set to $T_1T_2T_3T_2T_3T_4T_1T_2T_3T_5$ and the system support threshold value set to 50%. During the first stage of algorithm *Analyze*, the first iteration of its pattern mining process will identify T_2T_3 as a maximal repeating substring. Let us use a new type T_6 to denote $seq(T_2T_3)$. Then the type sequence becomes $T_1T_6T_6T_4T_1T_6T_5$. In the second iteration the first two occurrences of T_6 are collapsed into one, resulting in $T_1T_6T_4T_1T_6T_5$, in which T_1T_6 is a maximal repeating substring. Again, we use a new type T_7 to represent $seq(T_1T_6)$. So after the second iteration the type sequence becomes $T_7T_4T_7T_5$. Now the pattern mining process can be terminated since nothing new can be found. During the second stage of algorithm *Analyze*, the sequence $T_7T_4T_7T_5$ is partitioned into T_7T_4 and T_7T_5 using our heuristics and T_7 is assigned as the type of node S.

4.2.2 Semantic Analysis

There are two main problems with structural analysis. First, it may not *always* yield correct partitions corresponding to concept instances, especially in the presence of structural variation. In particular, the analysis based on maximal repeating substrings alone does not guarantee complete partitions. For example, in Figure 1 the fourth major headline news item starting with "After the Crossing ..." does not have any pointer to related news while all the others do. Invoking algorithm *Analyze* on the td node in Figure 7 (shown circled which contains all major headline news items) gives sequence S: $\gamma \cdot T_4 \cdot \gamma \cdot T_4 \cdot \gamma \cdot T_4 \cdot \gamma$, where $\gamma = seq(T_1 T_2 T_3)$; T_1, T_2, T_3, T_4 corresponds to news title, source, text summary, and pointers to related news, respectively. The correct partitions corresponding to the four major headline news items should be $P_1 = \gamma \cdot T_4$, $P_2 = \gamma \cdot T_4$, $P_3 = \gamma \cdot T_4$, $P_4 = \gamma$, such that $S = P_1 \cdot P_2 \cdot P_3 \cdot P_4$.

The second problem with structural analysis has to do with assigning semantic labels to partitions. Usually the labels of (small) partitions deep in a partition tree are provided by Web site designers in the document itself (*e.g.*, "INTERNATIONAL", "BUSINESS", ..., etc. appearing in the third column in Figure 1). When such a label is present in a document, it is usually the first text item in the partition. Adopting this simple heuristic, we can extract labels for many partitions. On the other hand, we also have to deal with labeling concept instances using labels not seen in the document (*e.g.*, "Headline News" in Figure 3). Such situations occur when smaller partitions are aggregated into bigger partitions.

To address the problems above, we use semantic analysis techniques, which discover lexical and concept associations in a subtree and propagate the associations discovered around the DOM tree. We now outline these semantic analysis techniques.

Lexical Association. Lexical association seeks to relate two *consecutive* pieces of raw text by examining whether they share common words (after dropping "stop" words such as "the") either directly or via synonym relationships. This is a light-weight linguistic processing technique for identifying small segments of related text. It is implemented in our system using WordNet [6].

Recall the example at the beginning of this section, where structural analysis produces the sequence $S = \gamma \cdot T_4 \cdot \gamma \cdot T_4 \cdot \gamma \cdot T_4 \cdot \gamma$. With only structural information, there is no way to identify the correct semantic units, $P_1 = \gamma \cdot T_4$, $P_2 = \gamma \cdot T_4$, $P_3 = \gamma \cdot T_4$, $P_4 = \gamma$, corresponding to the four major headline news items in the second column of Figure 1. However, observe that in partition P_2, the texts associated with γ and the following T_4 share the common word "Riyadh". Similarly, in partition P_3, the texts associated with γ and the following T_4 share the common word "Iraq". Therefore, by lexical association γ and T_4 can be semantically related and hence merged into one partition with high confidence.

Ontology Concept	Mapping Function
Headline News	Rule: Function of Title, Source, Content
National	Keyword: National, U.S.
International	Keyword: International, World
Science & Technology	Keyword: Science, Technology
Arts & Entertainment	Keyword: Arts, Movies, Music, Entertainment, Books, Travel
Business	Keyword: Business, Finance
Sports	Keyword: Sports, Baseball, Basketball
Health	Keyword: Health, Fitness
Detailed News	Rule: Function of Title, Content
News Taxonomy	Keyword: NEWS, OPINION

Table 1: Ontology Concepts and Their Mapping Functions

Concept Association. Concept association maps a partition to a semantic concept that succinctly summarizes the meaning of its content. The concept becomes the label of the partition. To make concept associations we leverage domain knowledge encoded in a domain-specific *ontology*. Informally an ontology describes concepts and their relationships, their features and attributes in a domain of interest. Part of an ontology for the news domain is shown in Table 1. To assign labels that are not present in an HTML document to partitions, we need to invoke rules in the ontology to classify the content of a partition. For instance, Table 1 shows the concept association rules for our news ontology. To determine if a partition can be classified as a headline news item the ontology uses a rule which is a function of the main features associated with it, namely, title (hyperlink), keywords for recognizing news sources (such as AP, Reuters, etc.), and features associated with news summaries such as constraints on the text length.[2]

Propagating Lexical and Concept Associations. In principle, the light-weight semantic analysis techniques introduced above are "incomplete" and hence not all concept instances can be identified. However, recall the key observation that semantically related items exhibit both consistency in presentation style and spatial locality. We can exploit this observation to *propagate* lexical and concept associations around the DOM tree.

For instance, recall the example illustrating lexical association above. In that example we determined via lexical association that γ and T_4 in P_2 and P_3 are semantically related. Such an association between γ and T_4 can be propagated to other nearby (γ, T_4) pairs to form the partition P_1 (and so the remaining γ becomes P_4).

Structural Types vs. Semantic Types. Like lexical associations, concept associations discovered can also be propagated to structurally similar items to assign semantic labels to partitions. In contrast to *structural* types that summarize the structural recurrence information about semantically related items, semantic labels can be viewed as *semantic* types that directly capture the semantics of partitions. Because semantic types factor out structural disparities, weaving structural and semantic types together enables our sequential pattern analysis process to uncover higher level concepts.

Now each node in a DOM tree is associated with two types: a structural type and a semantic type. Once a semantic type is assigned to the root node of a partition, it will replace its structural type (if it exists) in the structural analysis process. However, its structural type is still retained in order to propagate concept associations. This will enable structural and semantic analysis to work in tandem. Specifically, immediately prior to invoking structural analysis on the root of a DOM subtree, the semantic type of a child node is propagated to all its siblings having the same structural type.

4.3 Dialog Interface Manager

Our dialog interface manager is written in Java. It uses a collection of VoiceXML *templates* to create VoiceXML dialogs on the fly from the XML output of the partitioning system. Nodes or subtrees in the partition tree are matched to particular templates, variables in the templates are filled in with labels or node counts from the partition tree, and the resulting pieces of VoiceXML are strung together to form a complete VoiceXML dialog.

HearSay's templates include both generic templates (*e.g.*, for lists of links, lists of strings) and domain-specific ones (such as the one for newspaper articles shown in Figure 8). They include prosodic markup as well as structural markup and variables for elements of the XML partition tree. For example, the template in Figure 8 includes a list of prompts, some variables (marked with "@") for labels from nodes in the partition subtree matching this template, and pauses that encourage the user to provide input. We

[2]More sophisticated classifiers (*e.g.*, Bayes) can also be used to make concept associations.

News Portal	Major Headline News		Minor Headline News		Category News		Detailed News		News Taxonomy	
	Rec. %	Prec. %	Rec. %	Prec. %	Rec. %	Prec. %	Rec. %	Prec. %	Rec. %	Prec. %
New York Times	100	100	50	100	100	100	-	-	100	100
	-	-	25	100	-	-	100	100	-	-
CNN	100	100	-	-	100	100	-	-	100	100
	-	-	-	-	0	0	100	100	100	100
Yahoo News	-	-	-	-	81.8	100	-	-	100	50
	-	-	66.7	100	-	-	100	100	100	100
Google News	76.9	100	100	100	-	-	-	-	100	100
	-	-	-	-	-	-	100	100	100	100
ZdNet	93.3	100	66.7	100	-	-	-	-	-	-
	-	-	78.9	93.3	-	-	100	100	-	-
CNet	100	100	87.5	100	100	100	-	-	100	100
	-	-	0	0	-	-	100	100	100	100
Bloomberg News	100	100	-	-	-	-	-	-	100	100
	-	-	-	-	-	-	100	100	100	100
Recorder News	90	90	-	-	-	-	-	-	100	100
	-	-	-	-	-	-	100	100	-	-

Table 2: Recall and Precision Measures of Partitioning Results on 50 News Web Pages

use prosodic markup to mark places where the user should provide input, to improve "hearability" by providing variation in speaking style, and to reduce the impact of the sequential information presentation problem [34] by grouping a long list into shorter sublists. The VoiceXML dialogs output by the dialog creator permit input by item index (*e.g.*, "Item 1") or by name (*e.g.*, "World News", "Article", "Summary").

```
<field name="@FIELDNAME@">
  <prompt timeout="500ms">
    @DESCRIPTION@
    <break msecs="500"/>
    To hear a summary choose Summary.
    To hear the article choose Article.
    To go back to the menu choose Go Back.
    To exit choose Exit.
    <break msecs="5000"/>
  </prompt>

  <grammar>
    Summary | Article | Go Back | Exit
  </grammar>

  <filled>
    <prompt>
      <break msecs="500"/>
      <value expr="@FIELDNAME@"/>
    </prompt>
    @CONDITION@
  </filled>
</field>
```

Figure 8: VoiceXML Template for a News Article

The VoiceXML dialog creator specifies only how to apply the VoiceXML dialog templates to a semantic partition tree to generate VoiceXML dialogs. The structure of the dialogs themselves is in the dialog templates. This permits some system flexibility. The dialog creator produces VoiceXML dialogs from input partition trees in real time; the process is very efficient (see Section 5). However, the coherence of output dialogs is only as good as the quality of the input partition trees.

5. EVALUATION

We have developed a preliminary prototype of HearSay and have conducted two feasibility evaluations of this technology. First, we tested our partitioning algorithms on a collection of real HTML documents and measured the quality of partitions generated, which is an important indicator of the overall system performance. Second, we conducted a feasibility evaluation of the dialog interface to identify performance issues with the prototype; for this evaluation, we used 14 evaluators who are sighted. We also solicited comments on HearSay from 5 evaluators who are blind.

5.1 Effects of Partitioning

To evaluate the effectiveness of our structural and semantic analysis techniques for partitioning, we conducted an experiment on 50 HTML documents collected from 8 different news portals[3]. The ontology used in our experiment is shown in Table 1 (manually coded). Out of these 50 HTML documents, 15 are "front" pages (with multiple concepts and multiple instances for each concept) and the remaining 35 are "detailed" pages (mainly a long text description of some news story).

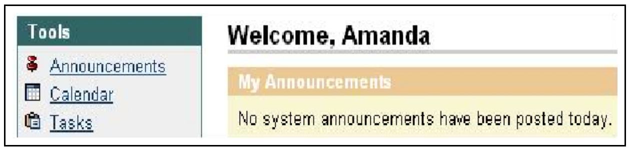
First we manually identified the ontology concepts and their instances for the HTML documents used in our experiment. We then ran our partitioning algorithms on these documents and computed their performance using two metrics, *recall* and *precision*, for each concept. Recall is a measure of "yield"; it is the ratio of the number of partitions correctly identified as instances of a concept over the actual number of concept instances. Precision is an indication of "accuracy"; it is the fraction of the number of partitions correctly labeled as instances of a concept over the total number of partitions (correctly and erroneously) labeled as instances of the concept by our system. The numbers shown in Table 2 are aggregated for each concept over all the HTML documents collected from the same news portal. Furthermore, each news portal is divided into front and detailed news pages and their corresponding recall and precision metrics are shown in the top and bottoms rows, respectively. Note that in Table 2 an entry with the symbol "-" means absence of the corresponding concept in the HTML documents.

The recall and precision numbers in Table 2 illustrate the effectiveness of our partitioning algorithms in practice. On detailed news pages which typically contain few concepts and concept instances our algorithms show high recall and precision. Even for front pages which normally include multiple concepts and multiple instances for each concept, our algorithms show little degradation. It is noteworthy pointing out that our algorithm almost always achieves 100% recall and precision on taxonomic news items, which typically show high structural homogeneity. Note that the recall for major headline news in Google's front page is considerably

[3]Google News indexes many other news portals.

lower than that of CNet's front page, due to the large number of concept instances in Google. Also observe that in ZdNet's front page, the recall value is low. This is due to the "incompleteness" of our semantic analysis techniques.

We also measured the amount of time our system took to process an HTML document. On a Pentium III machine with an 800MHz processor and 256MB memory, the running times ranged from 60 milliseconds to 6289 milliseconds for documents with 96 nodes to 1709 nodes in their DOM trees.



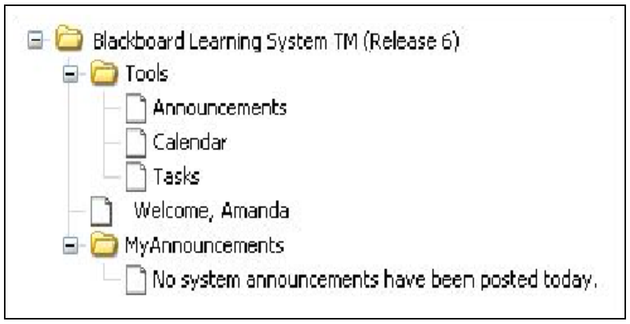

Blackboard Learning System TM (Release 6)
Tools
Announcements
Calendar
Tasks
Welcome, Amanda
MyAnnouncements
No system announcements have been posted today.

Figure 9: Partitioning Results on a University Education Blackboard Web Page

Finally, to demonstrate the applicability of our techniques to Web documents in different domains, we tested our partitioning algorithms on several other (structured) Web sites such as online office supplies offerings and electronic education bulletin boards, which are normally machine generated from templates. Figure 9 shows the results of partitioning a university course Web page; the top part is a fragment of the Web page while the lower one shows the partitioning results. Note that here structural analysis alone suffices to generate the partition tree shown in Figure 9.

5.2 HearSay System Usability

In this section we report an initial usability evaluation of the HearSay system. In our experiments, we selected three news sites, the New York Times, CNN, and Google News (Google News stories come from many different news sites). We created nine tasks for evaluators to perform. The assignment of task to news site was varied between evaluators. No two successive tasks referred to the same news site. The first three tasks tested basic HearSay functionality: the usefulness of HearSay for browsing to a Web page (*e.g.*, "Write down the headline of the first article in the 'World' or 'International' section of Google News"). The second three explored the user experience when listening to a part of a Web site (*e.g.*, "Write down the headline of an article about the presidential race in the 'US', 'National' or 'Politics' sections of the New York Times, and the name of a location mentioned in that article"). The final three tested the usefulness of HearSay when synthesizing information across multiple Web pages (*e.g.*, "Write down the name of a country mentioned in both the 'Business' and 'World'/'International' sections of CNN, and the headlines of the articles about that country"). All interactions with HearSay were logged; comments were solicited from all evaluators.

For this evaluation, evaluators were seated at a desktop computer in a lab, with ordinary lab activities going on around them (*i.e.*, a noisy environment). Evaluators wore a Koss headset with close-talking noise-canceling microphone. Each evaluator was given a sheet of paper with the nine questions on it, and was given verbal instructions similar to the following: *You may refer to things by name, e.g., "World News", or by number, e.g., "Item 1". You may also use commands such as "Go back" and "Repeat"*. Evaluators were not told anything else about HearSay, but could ask questions.

Results for this evaluation are shown in Table 3. The first task was removed for all evaluators as it was their first interaction with HearSay. All dialogs that had been "restarted" were also removed. This leaves 100 dialogs for analysis. Results are reported separately for each task type. The average time to follow links per dialog was 34 seconds per dialog.

Separately, HearSay was used by five evaluators at Arizona State University who are blind. These evaluators performed a subset of the tasks used in our feasibility evaluation. They also used HearSay for guided exploration (no specified task). They then made comments on HearSay, comparing it to the JAWS screen reader.

Most evaluators were very positive about their experience with HearSay. They liked the fact that the system organizes content into menus and items that permit the user to skip unwanted parts of a page. Native speakers of American English also liked the speech recognition in the interface, despite a high rate of incorrect barge-in detection. Evaluator suggestions can be divided into three categories: suggestions for improving the content extraction and organization; comments about desired additional functionality; and comments about the dialog interface.

Content Extraction and Organization. Evaluators thought that the extraction and organization of links and subsections in Web pages worked well. However, they wanted more control over their interaction with the content inside fairly unstructured text sections of Web pages (*e.g.*, newspaper article text). For example, they wanted to be able to skip to a particular paragraph or topic, repeat a particular part of the text and otherwise maintain pointers inside the text. Evaluators also wanted HearSay to be clearer about the types of menu items (links, groups, text, etc.), and wanted access to summary information for menus (*e.g.*, length).

Desired Additional Functionality. Several evaluators commented that the current browsing pattern used in HearSay is too limited. Proposed additions/alternatives include a skip forward feature, keyword enabled browsing and a context feature. A skip forward feature would permit an expert HearSay user to skip several menu levels up or down using commands such as "Skip 3 levels" or "Goto the national news menu". Keyword enabled browsing would permit a user to avoid traversing the whole page if she or he is searching for a particular type of information. For example, a user might go to her class Website and say "Find assignments". The system would find the part of the Website that contains the key words the user provided. Evaluators particularly wanted a reserved keyword for "Top" (the top page of the current Website). Finally, a context feature would permit a user to ask "Where am I?" and get the sequence of menu items or voice commands that led to the current browsing context.

Dialog Interface. The speech aspects of the dialog interface were a particular issue in these evaluations. First, the system is too repetitive. Because it permits barge-in (a feature our evaluators used frequently), it picks up noise in the environment as well as non-verbal sounds from the user (*e.g.*, breaths); when this happens, it repeats

Dialog Type	Dialog Length (sec.)	Total User Turns / SR Errors	Unique System Turns / Total System Turns	Page-Internal Links Followed / Page-External Links Followed	Time to Follow Links (sec.)
Browsing	232	17 / 10	16 / 26	4 / 2	16
Listening	435	28 / 16	22 / 43	7 / 4	31
Synthesizing	603	40 / 21	27 / 67	13 / 5	51
No answer	577	38 / 20	26 / 63	12 / 6	
Answer	390	26 / 15	20 / 41	7 / 3	
Total	437	29 / 16	22 / 47	8 / 4	34

Table 3: User Evaluation Results

itself. The most immediate way to improve the audio browsing experience would be to turn off this behavior, so that the system does not repeat itself unless asked by the user.

Second, there is a high proportion of speech recognition errors (1 out of every 2 utterances). The evaluators were all native English speakers but were not all native speakers of American English. The environment was also noisy, leading to many "speech recognition" errors when in fact the user did not speak (an average of 60% of the total number of speech recognition errors included no input words, and the majority of these are from environmental noise or non-verbal sounds from the user, *e.g.*, the user touching the microphone). However, with use the speech recognition error rate decreases. Two other ways to improve this aspect of the interface would be to use speaker-dependent speech recognition, and to support alternative input modalities (*e.g.*, keyboard, stylus)[4]. If these two sources of dialog error are removed, the time required for audio browsing will be almost halved.

Evaluators were told that they could refer to items by name (*e.g.*, "National News") or by number (*e.g.*, "Item 1"). We were interested in how often evaluators chose each type of reference. Of utterances that were recognized correctly, 51% were "by item" references and 30% were control utterances ("Go Back", "Repeat" or "Exit"), So fewer than 20% were by name references. Of utterances that were classified as speech recognition errors, 60% were not understood at all (most probably from environmental noise or non-verbals from the user), 13% were by item references, 10% were control utterances and 17% were "by name" references. It is perhaps not surprising that novice users struggling with speech recognition errors would choose by item references over by name references, but we plan to continue to permit by name references for more expert users who may know where they want to browse to without listening to a list of options.

Evaluators made three main comments about the dialog interface. First, both evaluators who are sighted and those who are visually disabled complained about the text-to-speech of HearSay. They wanted speech synthesis that is less monotonous. They also wanted explicit control over aspects of the speech synthesis, particularly speed. Second, evaluators commented that the system should provide more feedback. Finally, evaluators wanted access to alternative input modalities.

6. CONCLUSIONS AND FUTURE WORK

We have described the design and implementation of HearSay, an audio Web browser system. We have presented experimental results showing that HearSay can be used for "hands-free" audio browsing, although improvements in speed and accuracy are needed. This system has great potential for improving access to hypertext information for users with visual disabilities. We have also identified several potentially useful areas for continued research.

[4]This is not possible within the VoiceXML framework; it requires XHTML+Voice.

Partitioning. In our current implementation of HearSay, domain-specific ontologies as well as classifiers for identifying concepts in partitions were manually crafted. It will be interesting to automate these steps along the lines proposed in recent works using shallow natural language processing techniques (*e.g.*, [17]). Another important direction is to incorporate the co-training framework to learn classifiers from positive and unlabeled data [9], since labeling even a relatively small amount of data by hand still takes a lot of time. While most of these works address ontology mining from text, our problem is different since we deal with semistructured HTML documents. Consistency in presentation styles of semantically related elements in such documents can be exploited to mine ontological concepts as well as their features for automatically building classifiers. Finally, we plan to improve learning of probability distributions of important words for a domain concept (Bayes approaches to classification) using our recently developed techniques for multi-attribute data extraction with high precision and recall [35].

VoiceXML Dialogs. In addition to improving HearSay's dialog interface based on feedback from our evaluators (see Section 5), we also plan to work on creating summaries for Web pages or partitions of Web pages. In our use scenario, all the items in the "News" partition on the front page of the New York Times are links. If this information could be summarized at the start of the dialog, it would not have to be repeated for each item. There are other situations in which summarization is helpful. For example, while presenting the results of a search for a product on a shopping site (*e.g.*, http://shopping.yahoo.com) it may be more helpful to summarize the product information (*e.g.*, "Nike shoes, size six, black") than to just read the product name (*e.g.*, "Nike trackers"). When presenting a Web page that consists mainly of large chunks of text, it may be helpful to summarize the entire text of the page or the text in each chunk (similar to the abstracts the BrookesTalk system can produce [38]).

We plan to incorporate standard text summarization techniques (*e.g.*, those used by [18, 28]) into HearSay. We will also explore category-specific summarization techniques that we can incorporate into the ontologies used in HearSay. These summarization techniques rely on an ontology; a summary for an entity must include information about the major features associated with the class of that entity in the ontology. This requires identifying those features in the Web page content, and grouping them appropriately for spoken output.

New Applications and Domains. We have been exploring two additional applications for our partitioning algorithm. The first application, commerce, involves comparing partitions across Web pages to identify structurally or semantically similar units. The second application, education, involves looking at course Web sites (which are much more variable than news Web sites) to see how partitioning can be applied to improve access to educational materials.

7. ACKNOWLEDGMENTS

We would like to acknowledge the work on HearSay done by Amarjeet Singh, Joon Im, Seong Shin, Zarana Shah, Shailesh Kamat, and Rajeev Kasthuri. We thank Srinivas Vadrevu, Terri Hedgpeth, and Hasan Davulcu for managing the deployment and evaluation of HearSay at Arizona State University. We are also grateful to the evaluators for their time and patience.

8. REFERENCES

[1] http://www.freedomscientific.com/.

[2] netECHO[tm]. http://www.internetspeech.com.

[3] Speech Application Language Tags. http://saltforum.org/.

[4] VoiceXML. http://www.voicexml.org/.

[5] Wireless Markup Language Specification. http://www.wapforum.org/what/technical.htm.

[6] WordNet. http://www.cogsci.princeton.edu/˜wn/.

[7] C. Asakawa and T. Itoh. User interface of a home page reader. In *ACM International Conference on Assistive Technologies (ASSETS)*, 1998.

[8] C. Asakawa and C. Laws. Home Page Reader: IBM's talking Web browser. Technical report, IBM, 1998.

[9] A. Blum and T. M. Mitchell. Combining labeled and unlabeled data with co-training. In *Computational Learning Theory (COLT)*, 1998.

[10] O. Buyukkoten, H. Garcia-Molina, and A. Paepcke. Focussed Web searching with PDAs. In *International World Wide Web Conference (WWW)*, 2000.

[11] C. Y. Chung, M. Gertz, and N. Sundaresan. Reverse engineering for Web data: From visual to semantic structures. In *International Conference on Data Engineering (ICDE)*, 2002.

[12] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. Tomlin, and J. Yien. SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation. In *International World Wide Web Conference (WWW)*, 2003.

[13] 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.

[14] D. Embley and L. Xu. Record location and reconfiguration in unstructured multiple-record Web documents. In *ACM International Workshop on the Web and Databases (WebDB)*, 2000.

[15] D. W. Embley, D. M. Campbell, R. D. Smith, and S. W. Liddle. Ontology-based extraction and structuring of information from data-rich unstructured documents. In *International Conference on Information and Knowledge Management (CIKM)*, 1998.

[16] D. W. Embley, Y. Jiang, and Y.-K. Ng. Record-boundary discovery in Web documents. In *ACM International Conference on Management of Data (SIGMOD)*, 1999.

[17] J. Franke, G. Nakhaeizadeh, and I. Renz, editors. *Text Mining: Theoretical Aspects and Applications*. Springer-Verlag, 2003.

[18] J. Goldstein, V. O. Mittal, J. G. Carbonell, and J. P. Callan. Creating and evaluating multi-document sentence extract summaries. In *International Conference on Information and Knowledge Management (CIKM)*, 2000.

[19] J. Gunderson and R. Mendelson. Usability of World Wide Web browsers by persons with visual impairments. In *RESNA Annual Conference*, 1997.

[20] D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

[21] S. Handschuh and S. Staab. Authoring and annotation of Web pages in CREAM. In *International World Wide Web Conference (WWW)*, 2002.

[22] S. Handschuh, S. Staab, and R. Volz. On deep annotation. In *International World Wide Web Conference (WWW)*, 2003.

[23] J. Heflin, J. A. Hendler, and S. Luke. SHOE: A blueprint for the semantic web. In D. Fensel, J. A. Hendler, H. Lieberman, and W. Wahlster, editors, *Spinning the Semantic Web*. MIT Press, 2003.

[24] A. Huang and N. Sundaresan. A semantic transcoding system to adapt web services for users with disabilities. In *ACM International Conference on Assistive Technologies (ASSETS)*, 2000.

[25] IBM. IBM special needs systems. http://www.ibm.com/sns, 1998.

[26] H. Kochocki, S. Townsend, N. Mitchell, and A. Lloyd. W3C launches internation Web accessibility initiative. Technical report, W3C, 1997.

[27] H. Lieberman. Letizia: An agent that assists Web browsing. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.

[28] C. Lin and E. Hovy. From single to multi-document summarization: A prototype system and its evaluation. In *Meeting of the Association for Computational Linguistics (ACL)*, 2002.

[29] Microsoft Corporation. Microsoft accessibility technology for everyone. http://www.microsoft.com/enable/, 1998.

[30] T. Oogane and C. Asakawa. An interactive method for accessing tables in HTML. In *ACM International Conference on Assistive Technologies (ASSETS)*, 1998.

[31] C. Schmandt. Audio Hallway: A virtual acoustic environment for browsing. In *ACM Symposium on User Interface Software and Technology (UIST)*, 1998.

[32] SUN Microsystems. Accessibility support for the Java platform. 1998.

[33] H. Takagi, C. Asakawa, K. Fukuda, and J. Maeda. Side-wide annotation: Reconstructing existing pages to be accessible. In *ACM International Conference on Assistive Technologies (ASSETS)*, 2002.

[34] M. A. Walker, R. Passonneau, and J. E. Boland. Quantitative and qualitative evaluation of DARPA Communicator spoken dialogue systems. In *Meeting of the Association of Computational Lingustics (ACL)*, 2001.

[35] G. Yang, S. Mukherjee, and I. V. Ramakrishnan. On precision and recall of multi-attribute data extraction from semistructured sources. In *IEEE International Conference on Data Mining (ICDM)*, 2003.

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

[37] S. Yu, D. Cai, J.-R. Wen, and W.-Y. Ma. Improving pseudo-relevance feedback in Web information retrieval using Web page segnmentation. In *International World Wide Web Conference (WWW)*, 2003.

[38] M. Zajicek, C. Powell, and C. Reeves. Web search and orientation with BrookesTalk. In *Technology and Persons with Disabilities Conference*, 1999.