

# Model-directed Web Transactions under Constrained Modalities<sup>1</sup>

Zan Sun

and Jalal Mahmud

and I.V. Ramakrishnan

Department of Computer Science

Stony Brook University

Stony Brook, NY 11794, USA

{zsun, jmahmud, ram}@cs.sunysb.edu

and

Saikat Mukherjee

Siemens Corporate Research

Princeton, NJ 08540

saikat.mukherjee@siemens.com

Online transactions (*e.g.*, buying a book on the Web) typically involve a number of steps spanning several pages. Conducting such transactions under constrained interaction modalities as exemplified by small screen handhelds or interactive speech interfaces - the primary mode of communication for visually impaired individuals - is a strenuous, fatigue-inducing activity. But usually one needs to browse only a small fragment of a Web page to perform a transactional step such as a form fillout, selecting an item from a search results list, *etc.* We exploit this observation to develop an automata-based process model that delivers only the “relevant” page fragments at each transactional step, thereby reducing information overload on such narrow interaction bandwidths. We realize this model by coupling techniques from content analysis of Web documents, automata learning and statistical classification. The process model and associated techniques have been incorporated into Guide-O, a prototype system that facilitates online transactions using speech/keyboard interface (Guide-O-Speech), or with limited-display size handhelds (Guide-O-Mobile). Performance of Guide-O and its user experience are reported.

Categories and Subject Descriptors: I.7.5 [Document and Text Processing]: Document Capture—*Document Analysis*

General Terms: Algorithms, Human Factors

Additional Key Words and Phrases: Web transaction, Content Adaption, Assistive Device

## 1. INTRODUCTION

The World Wide Web has become the dominant medium for doing e-commerce. The volume of goods bought from online stores continues to grow dramatically. A Web *transaction* such as buying a CD player from an online store involves a number of user steps spanning several Web pages. As an illustration let us examine some common steps for buying a CD player from Best Buy (<http://www.bestbuy.com>).

---

<sup>1</sup>This is an extended version of our paper *Model-directed Web Transactions under Constrained Modalities* that appears in the proceedings of International World Wide Web Conference 2006 (WWW'06).

Fig. 1. A Web Transaction Example

To begin the user fills out the search form with “electronics” as the category and “CD Player” as the item. The search result generated in response is shown in Figure 1(b). The user selects the very first item in the result list which leads to the page in Figure 1(c) containing product details. To complete the transaction the user adds this selected item to the cart which leads to the page in Figure 1(d) wherein the user selects checkout. Observe that there are two essential components to a Web transaction: (i) locating the relevant content, such as a search form or the desired item in a Web page, and (ii) performing a sequence of steps, such as filling out a search form, selecting an item from the search list and doing a checkout. For completing a transaction these steps usually span several pages.

Online transactions such as the one described above are usually performed with graphical Web browsers. The primary mode of interaction with graphical browsers is visual, an intrinsically spatial modality. Hence, users can quickly scan through the rich engaging content in Web pages scripted for e-commerce and locate the objects of interest quite easily. Moreover, the spatial organization of content in these pages helps users comprehend the sequence of steps necessary to complete a transaction.

Now consider scenarios where visual interaction is impossible (*e.g.*, when the user is a visually handicapped individual) or the interaction media has small displays (*e.g.*, mobile handhelds). Speech is the primary modality of interaction in the former situation. Speech interfaces and small displays offer narrow interaction bandwidths making it cumbersome and tedious to get to the pertinent content in a page. For instance, state-of-the-art screen readers and audio browsers (*e.g.*, JAWS and IBM’s Home Page Reader) provide almost no form of filtering of the content in a Web page resulting in severe *information overload*. This problem is further exacerbated when such an interaction spans several pages as in an online transaction. In particular the loss of spatially organized content makes it difficult for users to comprehend the sequence of transactional steps. While content summarization can compensate somewhat for this loss, it alone cannot handle the information overload that the user faces.

Thus, there is a need for techniques that facilitate Web transactions using constrained interaction modalities that are far less cumbersome to do than current approaches. This paper addresses this problem and describes our solution.

We capture the two aspects of a transaction, namely its operation sequence and content identification by a *process model* and an *ontology* respectively. The ontology describes the set of *semantic concepts* occurring in Web pages, which are considered essential for conducting Web transactions in a particular domain. The circled elements in Figure 1 are examples of such concepts. The process model is a deterministic finite automata (DFA) that captures the set of transactional sequences. Each state, representing an atomic operation in a transaction, is associated with a set of semantic concepts drawn from the ontology. When the model makes a transition to a state during the course of a transaction, a Web page is provided to the state as an input. If the concepts associated with the state are present in the page, then they alone are identified and presented to the user. For instance,

Fig. 2. Utility Bill Payment using Guide-O-Mobile

if the page shown in Figure 1(b) is given as the input to a state associated with the concepts “Item Taxonomy” and “Search Result” only the two circled items in the figure will be identified and presented to the user. Since transactions are essentially interactive, we associate each concept with a user operation, *e.g.*, the *submit\_searchform* operation with the “Search Form” concept. Each such operation results in a state transition and a *sequence* of operations constitutes a Web transaction. Thus coupling content semantics with model-directed navigation can overcome the information overload problem by delivering relevant content at every step of the transaction.

Our approach to semantic concept identification in Web pages is built upon our previous work [Mukherjee et al. 2005] where we had proposed a technique for partitioning a page’s content into segments constituting concept instances. It uses a combination of structural analysis of the page and machine learning. We adopt it for the problem addressed in this paper and enhance its learning component to produce more robust statistical models of semantic concepts. It is noteworthy to point out that the use of content semantics, as opposed to syntax based techniques for identifying concepts makes it more robust to structural variations in the pages and scalable over Web sources that share similar content semantics.

We also use automata learning techniques (see [Murphy 1996] for a survey) to construct process models from training sequences generated from real Web transactions. The use of process models for online transactions bears similarities to the emerging Web services paradigm for conducting automated transactions. But the fundamental difference is that our technique works on *Web pages* instead of services exposed by a service provider (see Section 6 for detailed comparison).

The rest of this paper is organized as follows: In Section 2, we describe a user scenario and the architecture of Guide-O, a prototype system that we implemented based on the techniques detailed in this paper. Currently, Guide-O can be configured to work with speech (*Guide-O-Speech*) or with small display handhelds (*Guide-O-Mobile*). In Section 3 we formalize the process model and describe its implementation using DFA learning techniques. Content analysis techniques for semantic concept identification appear in Section 4. Quantitative experimental evaluation of the two Guide-O configurations as well as qualitative user experience appear in Section 5. Related work appears in Section 6 and we conclude the paper in Section 7.

## 2. THE GUIDE-O SYSTEM

### 2.1 Use Scenario of Guide-O Speech:

Alice, who is a visually impaired individual, is planning on replacing her broken CD player with a new one from Best Buy. To begin, she speaks Best Buy’s URL to Guide-O. After retrieving the home page Guide-O analyzes this page, extracts the two concepts in it, namely “Item Taxonomy” (the circled item on the left in Figure 1(b)) and “Search Form” (the circled item on the top of Figure 1) and asks Alice to choose one of them. Alice says “Search Form” and in response Guide-O

reads out the drop-down items in the search form pausing briefly after each item. Alice can pick an item at any time by either saying the item name or its number. Alice says “Electronics” and Guide-O prompts her for the electronic item she wishes to search for. Alice responds with “CD Player”. Guide-O submits the search form filled with these two parameters that results in fetching the page containing the search results shown in Figure 1(b). Guide-O extracts the “Search Result” concept and begins reading out the brief description associated with each CD player in this list. Alice says “item 1” causing Guide-O to follow the link associated with the 1st player (CDP-CE375) in the list to the page containing a detailed description of this player (Figure 1(c)). In this page Guide-O extracts three concepts, namely “Search Form”, “Item Detail” and “Add To Cart”. Alice is asked if she wishes to hear the product details. When Alice responds in the affirmative Guide-O reads out the detailed description of the CD player she picked earlier. At the end Alice is asked if she wishes to add this to her shopping cart. Alice responds “yes”. Guide-O follows the link labeled *Add To Cart* in Figure 1(c) to the page shown in Figure 1(d). In this page Guide-O extracts the concepts of “Search Form”, “Shopping Cart”, “Checkout” and “Continue Shopping”. When presented with these choices Alice chooses “Checkout”. To complete the transaction Alice must provide credit card information upon checkout. Its details have been omitted as they are quite similar to the form filling step described in the first step of the scenario. At any point Alice can also say any one of a set of general-purpose navigation commands, such as “Back to top page”, “Start over”, “Repeat” (last item) or “Stop”. Besides, on a laptop or desktop computer Alice could also use a keyboard in addition to speech to interact with Guide-O.

## 2.2 Use Scenario of Guide-O Mobile:

Let us examine how Jane, a Verizon subscriber, will use the Guide-O-Mobile to pay her telephone bill. To begin, she submits Verizon’s URL to Guide-O-Mobile. After retrieving the home page, Guide-O-Mobile analyzes this page, extracts the “My Account” concept segment (Figure 2(a)) in it and displays it to Jane. Jane clicks on the “Sign in” link and in response the “Sign-in” concept segment (Figure 2(b)) is presented and Jane types in the login info. On successful login, the “Billing Summary” segment (Figure 2(c)) is displayed along with two action choices - View-Bill and Pay-Bill. Jane chooses the former and the “Complete Bill” segment (Figure 2(d)) is shown. In this page, Jane is presented with two action choices - Pay-Bill and View-Detail. Jane picks the latter which causes the “Bill Details” concept segment (Figure 2(e)) to be presented. Jane chooses the “Pay-Bill” action here and the “Bill Payment” segment data (Figure 2(f)) is displayed. Jane provides payment information in this page. On receipt of this data a Submit-Form action takes place and the transaction proceeds to the page where the payment data is reviewed and a final sign-out completes the transaction.

The use scenario above illustrates how a user conducts Web transactions with Guide-O-Mobile using mobile handheld devices where graphical interaction with a standard browser is not feasible because of their small screen size. Instead of displaying the entire Web page to the user, Guide-O-Mobile only displays the extracted concepts of a page. Since usually these are few in number with small content, they can be browsed more easily even under small screen size constraint.

Fig. 3. Guide-O System Architecture

Concept	Operation
Shopping Cart	view_shoppingcart
Add To Cart	add_to_cart
Edit Cart	update_cart
Continue Shopping	continue_shopping
Checkout	check_out
Search Form	submit_searchform
Search Result	item_select
Item List	item_select
Item Taxonomy	select_item_category
Item Detail	show_item_detail

Table I. Concepts in Ontology for Online Shopping.

### 2.3 Guide-O Architecture

The architecture of Guide-O is shown in Figure 3. It can be configured to operate with speech/keyboard inputs (Guide-O-Speech) or with mobile handheld devices (Guide-O-Mobile). In the latter configuration the *Interface Manager* automatically generates a DHTML page to display the extracted concepts while for the former configuration it automatically generates a VoiceXML<sup>2</sup> dialog interface. We use freely available text-to-speech synthesizers and speech recognizers in Guide-O-Speech. We have developed a VoiceXML interpreter to handle both speech as well as keyboard inputs.

The *Content Analyzer* partitions an input Web page into segments containing semantically related content elements (such as “Item Taxonomy”, “Search Result” in Figure 1) . Using an underlying shallow ontology of concepts present in Web pages it classifies these segments to the concepts in the ontology and labels them with their names. (Section 4 describes the classification technique.) Table I shows concept names in our shallow ontology for online shopping transaction domain (e.g. books, consumer electronics, office supplies). Table II lists some concept names in a shallow ontology for the utility bill payment transaction domain. Associated with each concept is an operation as shown in the tables. When the user selects a concept in a state, the corresponding operation is invoked. The ontology also includes information for classification of page segments to concepts.

The *Browser Object Interface* fetches pages from Web servers. Special features include automatic form fillouts and retrieval of pages pointed to by navigable links, which requires execution of javascript.

The *Process Model* orchestrates all the actions that take place in Guide-O. In a state, it takes a URL as the input, calls the Browser Object Interface to fetch the page and the Content Analyzer to extract from the page the concepts that it expects. The extracted concepts are organized as a concept tree that is dispatched by the Process Model to the Interface Manager for presentation to the user. When

<sup>2</sup><http://www.w3.org/TR/voicexml20/>

Concept	Operation
My Account	Sign-In
Sign In	Submit-Form
Billing Summary	Pay-Bill, View-Bill
Complete Bill	Pay-Bill, View-Detail
Bill Payment	Submit-Form
Bill Details	Pay-Bill

Table II. Concepts in Ontology for Utility Bill Payment.

Fig. 4. A Process Model for Online Shopping

Fig. 5. A Process Model for Utility Bill Payment

the user selects a concept, it is sent to the Process Model as an operation on the concept. A state transition based on the operation is made and the cycle repeats.

The architecture described is in essence the runtime engine that drives the Guide-O system. The Process Model and the Concept Extractor in the engine are learned a priori in the learning component.

### 3. PROCESS MODEL

Formally, our process model is defined as follows: Let  $C = \{c_0, c_1, \dots\}$  be a set of concepts, and  $I(c)$  denote the set  $c$  of concept instances. Let  $Q = \{q_0, q_1, \dots\}$  be a set of states. With every state  $q_i$  we associate a set  $S_i \subseteq C$  of concepts. Let  $O = \{o_0, o_1, \dots\}$  be a set of operations. An operation  $o_i$  can take parameters. A transition  $\delta$  is a function  $Q \times O \rightarrow Q$ , and a concept operation  $\rho$  is also a function  $C \rightarrow O$ . Operations label transitions, *i.e.*, if  $\delta(q_i, o) = q_j$  then  $o$  is the label on this transition. An operation  $o = \rho(c)$  is enabled in state  $q_i$  whenever the user selects an instance of  $c \in S_i$  and when it is enabled a transition is made from  $q_i$  to state  $q_j = \delta(q_i, o)$ .

Technically a concept instance is the occurrence of a concept in a Web page. For example, the circled items in Figure 1 are all concept instances. But for brevity we choose not to make this distinction explicitly and use concepts and concept instances interchangeably when referring to content segments in Web pages.

Figure 4 illustrates a process model. The concepts associated with the starting state  $q_1$  are “Item Taxonomy”, “Item List”, and “Search Form”. This means that if these concept instances are present in the Web page given to  $q_1$  as its input, they will be extracted and presented to the user. User can select any of these concepts. When the user selects the “Search Form” concept he is required to supply the form input upon which the *submit\_searchform* operation is invoked. This amounts to submitting the form with the user-supplied form input. A Web page consisting of the search results is generated and a transition is made to  $q_3$ . As discussed in the use scenario in Section 2 we have omitted the payment steps following checkout.

Hence  $q_6$  which is entered upon a *check\_out* operation is deemed as the final state.

Figure 5 is an example of a process model for utility bill payment corresponding to the use scenario for Guide-O-Mobile described in Section 2.

(a) (b)

Fig. 6. (a) A Prefix Automata. (b) Learned DFA

### Process Model Learning

A process model can be built manually or learned from training data. For simple tasks, users can create their own process models. However, when we don't have sufficient "knowledge" on how to accomplish the task in a step-by-step process or there are variations in how the same task is carried out in different sites, manual construction of the process model becomes cumbersome. This is more evident when the resultant model is large (i.e. has large number of states and transitions). Here we propose a learning based approach of process model construction, which is generic and scalable. Specifically, we built the process model using DFA learning techniques, a thoroughly researched topic (see Section 6 for related work). In the DFA learning problem the training set consists of two sets of example strings, one labeled positive and the other negative. Only strings in the positive set are in the DFA's language. The objective is to construct a DFA that is *consistent* with respect to these two sets, *i.e.*, it should accept strings in the positive set while rejecting those in the negative set. We adapted the heuristic in [Oncina and Garc 1991] for learning our process model, the choice being mainly dictated by its simplicity and low complexity.

The training sequences we used for learning the process model consist of strings whose elements are operations on concepts. The sequence  $\langle submit\_searchform, item\_select, add\_to\_cart, check\_out \rangle$  is one such example. These training sequences are (manually) labeled "completed" and "not completed". The positive example set ( $S+$ ) consists of sequences labeled "completed" while the negative example set ( $S-$ ) consists of those labeled "not completed".

We first construct a prefix tree automata as shown in Figure 6(a) using only the examples in  $S+$ . In Figure 6(a), the sequence of operations along each root-to-leaf path constitutes a string in  $S+$ . For this example  $S-$  consists of the strings:  $\{\langle check\_out \rangle, \langle submit\_searchform, add\_to\_cart \rangle, \langle submit\_searchform, check\_out \rangle, \langle select\_item\_category, add\_to\_cart, check\_out \rangle\}$ . The prefix of every string in  $S+$  is associated with a unique state in the prefix tree. The prefixes are ordered and each state in the prefix tree automata is numbered by the position of its corresponding prefix string in this lexicographic order. Next we generalize the prefix tree automata by state merging. We choose state pairs  $(i, j)$ ,  $i < j$  as candidates for merging. The candidate pair  $(i, j)$  is merged if it results in a consistent automata. For example, merging the pair (1,2) is consistent whereas (3,4) is not merged as the resulting automata will accept the string  $\langle submit\_searchform, check\_out \rangle$  in  $S-$ . The DFA that results upon termination of this merging process on the above example set is shown in Figure 6(b).

Since we do at most  $Q^2$  state mergings, where  $Q$  is the cardinality of  $S+$ , the time complexity is polynomial. In Section 5 experimental evaluation of the performance of the learned process model is presented.

(a) DOM (b) partition tree

Fig. 7. Structural Analysis of the Page in Fig 1(a)

#### 4. CONTENT ANALYSIS

Herein we describe the *content analyzer* module that extracts the relevant concepts. In a nutshell this is achieved by *partitioning* a Web page into segments of semantically related items and classifying them against concepts in the ontology. Below we provide the detail of our approach.

##### The Approach

It is based on our previous work on learning-based semantic analysis of Web content [Mukherjee et al. 2005]. Briefly, the technique rests on three key steps: (i) inferring the logical structure of a Web page via structural analysis of its content, (ii) learning statistical models of semantic concepts using light-weight features extracted from both the content as well as its logical structure in a set of training Web pages, and (iii) applying these models on the logical structures of new Web pages to automatically identify concept instances.

##### 4.1 Structural Analysis:

The basic idea of structural analysis comes from our previous work (see [Mukherjee et al. 2005] for details), which is based upon the observation that semantically related items in Web pages typically exhibit consistency in presentation style and spatial locality. For example, in the search result page of BestBuy (Figure 1(b)), all the search result items which are semantically related, located in the middle of the page, have similar presentation: an image followed by the product name and the description of the product.

**4.1.1 Type System.** Exploiting the above observation, we have redesigned our simple but effective type system in the previous work to encode the presentation style and structural information. Instead of treating the HTML tags equally for the primary type (which is the sequence containing all the HTML tags in the root-to-leaf path in the DOM tree, see [Mukherjee et al. 2005] for more details), we divide the HTML tags into three categories: *style tags*, *layout tags* and *functional tags* (see Table III). Style tags in a HTML page are about the presentation style of visible elements such as `font` and `style`; functional tags are those that enable HTML document browsers to handle events and functions such as `script` and `meta`; layout tags are all the tags other than the above two categories, and are used to arrange the position of the visible elements. For the purpose of partitioning the content, the functional tags are omitted since they don't contribute to the presentation style of the content.

Our type system is based on the Document Object Model (DOM) of the HTML document. A segment of the DOM presentation of the HTML page is shown in Figure 7(a), where the internal nodes of the tree are labeled with HTML tags and the displayable content are at the leaf nodes. Formally:

*Definition 4.1.* every node in DOM tree has an associated type  $T$  which is de-

Category	HTML tags
Style tags	font, color, style, strong, i, b, a, big, em, img.
Layout tags	table, tr, td, p, br, etc. (All the tags not belong to the other two categories.)
Functional tags	head, script, meta, var, object, applet.

Table III. Categories of the HTML tags.

defined as following,

- A *primary type*  $T$  of a node  $n$  is a 2-tuple  $\langle \mathcal{D}, \mathcal{L} \rangle$ , in which  $\mathcal{D}$  is the style of  $n$  indicated by collecting the style tags in the path from root to  $n$  and/or extracting the `class` attributes from *Cascade Style Sheets* (CSS)<sup>3</sup>;  $\mathcal{L}$  is the sequence of tags  $(t_1, t_2, \dots)$ , in which  $t_i$  is the  $i_{th}$  HTML layout tag in the path from the root node of DOM tree to  $n$ .
- A *compound type*  $T$  is a sequence of types  $(T_1, T_2, \dots)$ , in which  $T_i$  is also a *type*.
- A *type*  $T$  is either a primary type or a compound type.

Intuitively, the primary type encodes the presentation style (including location and visual cues such as font type and size) of a piece of content (text or image) that corresponds to a leaf node in a DOM tree. For example, the leaf nodes of the product names (“*SONY 5-Disc...*”, “*SONY CD-R...*”, etc.) in Figure 7(a) have the same primary type  $T_1 = \langle \mathcal{D}_1, \mathcal{L}_1 \rangle$ , where  $\mathcal{D}_1 = \{\text{bold}=\text{true}, \text{italic}=\text{true}, \text{link}=\text{true}, \text{size}=11\text{px}, \text{face}=\text{serif}, \text{color}=\#000000, \text{image}=\text{false}\}$  and  $\mathcal{L}_1 = \dots \text{FORM.TABLE.TR.TD}$ . Similarly, all the descriptions of items (“*Variable line...*”, “*Internal storage...*”, etc.) share the same primary type  $T_2 = \langle \mathcal{D}_2, \mathcal{L}_2 \rangle$ , where  $\mathcal{D}_2 = \{\text{bold}=\text{false}, \text{italic}=\text{false}, \text{link}=\text{false}, \text{size}=9\text{px}, \text{face}=\text{serif}, \text{color}=\#000000, \text{image}=\text{false}\}$  and  $\mathcal{L}_2 = \dots \text{FORM.TABLE.TR.TD}$ .

A compound type essentially summarizes the structural recurrence information of a subtree rooted at an internal node. For example, in Figure 7(a) the subtree rooted at the circled TABLE node are the search results. The property of spatial locality combined with consistency in presentation style reveals structural recurrence information about semantically related items. For example, the TR nodes under the circled TABLE node have the same compound type  $T_{TR} = T_1T_2T_3T_4T_5$  where each  $T_i$  is the primary type of the  $i_{th}$  leaf node under them. By concatenating all the types of the circled TABLE node’s children, we get a type sequence  $T_{TR}T_{TR}T_{TR}\dots$  where the sequential pattern,  $T_{TR}$ , exactly captures the structural recurrence information of each semantic unit (i.e., the items in the search results). In our type system, such a repeating type sequence  $T_{TR}$  results in a compound type  $T_{TABLE} = T_1T_2T_3T_4T_5$  for the circled TABLE node.

**4.1.2 Partition Creation.** We now formalize the notion of a *pattern* used in partitioning a Web page into semantically related items:

*Definition 4.2.* A pattern  $p$  is a substring of type sequence  $T$  with the length

<sup>3</sup>There are many attributes defined in CSS styles. However in our work, we only use a small subset of the CSS style attributes.

$|p| \geq 2$ , such that there are  $n_p (\geq 2)$  occurrences of  $p$  in  $T$  and for any two occurrences  $p_i, p_j$  of  $p$ ,  $p_i \cap p_j = \emptyset$ .

It is quite common that a sequence contains more than one pattern, and only one pattern can be used to identify the related items. In our previous work, we defined a *maximal* pattern to be the only candidate based on the frequency of its appearance. A more robust definition is as follows,

*Definition 4.3.* A pattern is *maximal* if and only if  $\forall$  pattern  $p' \neq p$ , either  $n_{p'} < n_p$  or  $p \not\subset p'$ .

For example, in the compound type  $T_1T_2T_3T_1T_2T_3T_1T_2$ , the maximal patterns are  $T_1T_2$ ,  $T_1T_2T_3$ ,  $T_2T_3T_1$  and  $T_3T_1T_2$ .  $T_2T_3$  is also a pattern but not maximal, as the pattern  $T_1T_2T_3$  repeats the same number of times as  $T_2T_3$  and  $T_2T_3 \subset T_1T_2T_3$ .  $T_1T_2T_3T_1$  has two occurrences in the sequence but it is not a pattern, since the two occurrences overlap with each other.

---

**Algorithm 1** PartitionTree(n)
 

---

**Require:**  $n$ : a node in a DOM tree

- 1: **if**  $n$  is a leaf node **then**
- 2:  $n.type =$  the primary type defined above.
- 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 all** child node  $x$  of  $n$  **do**
- 8: PartitionTree( $x$ )
- 9: **end for**
- 10: Analyze( $n$ )
- 11: **end if**

---

Our top-level structural analysis algorithm, namely *PartitionTree* shown above, partitions each internal node according to the maximal pattern found in the type sequence of its children. *PartitionTree* traverses the DOM tree top-down and then re-structures it bottom-up. When a leaf node is processed, we collect its primary type in Line 2. For the internal node with only one child, we process its only child; delete the node and move its child up to the node's parent, as shown in Line 4-5. However, for an internal node with multiple children, *PartitionTree* is invoked on its children for type computation (Line 8) and then, pattern discovery is performed on the type sequence of its children (Line 10).

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. As mentioned above, such similarity is discovered by finding the maximal pattern in the type sequence  $S$  of its children. The main body of algorithm *analyze* is an iterative process. At Line 6 the set of maximal patterns in the type sequence  $S$  is generated. Each such maximal pattern is the basis of a *partition scheme* (Line 8), and we assign a "goodness" score to the schemes (Line 9). The scheme with the highest score will be used to actually partition the sequence  $S$  (Line 15). We will show the

---

**Algorithm 2** Analyze( $n$ )

---

**Require:**  $n$ : an internal node in a DOM tree

- 1:  $finalpattern = null$
- 2:  $S =$  the type sequence of all the child nodes of  $n$
- 3: **repeat**
- 4:    $max = 0$ ;  $maxpattern = null$ ;  $maxscheme = null$
- 5:   collapse adjacent nodes in  $S$  which share the same type
- 6:    $\Phi = \text{MaximalPatterns}(S)$
- 7:   **for all** each maximal pattern  $p$  in  $\Phi$  **do**
- 8:      $scheme = \text{GenerateScheme}(p, S)$
- 9:      $score = \text{EvaluateScheme}(scheme)$
- 10:    **if**  $score > max$  **then**
- 11:      $max = score$ ;  $maxpattern = p$ ;  $finalpattern = p$ ;  $maxscheme =$   
        $scheme$
- 12:    **end if**
- 13:   **end for**
- 14:   **if**  $maxpattern \neq null$  **then**
- 15:     partition and update  $S$  with scheme  $maxscheme$
- 16:   **end if**
- 17: **until**  $maxpattern = null$
- 18: **if**  $finalpattern \neq null$  **then**
- 19:   set the type of  $n$  to  $maxpattern$
- 20: **else**
- 21:   set the type of  $n$  to  $S$
- 22:   flatten all the grandchild nodes of  $n$  (each child of  $n$  will be a 1-level subtree)
- 23: **end if**

---

details shortly. The process repeats until no more patterns can be found. Finally, the type of  $n$  will be set to the type sequence of the last found pattern (if it exists), or to the type sequence  $S$  of its children (otherwise). If no pattern is found, its immediate children are deleted and their child nodes are attached to the node  $n$  itself (Line 22). In our algorithm, we use *suffix trees* to find the repeating substrings in  $S$  (see [Gusfield 1997] for the details). We find the maximal patterns amongst these substrings.

Let us illustrate the algorithm step by step using an example. Suppose algorithm *PartitionTree* is invoked on the tree shown in Figure 8(a). The nodes are labeled as  $n_i, 1 \leq i \leq 19$ . Each leaf node has its primary type displayed on the left. The rectangle between  $n_{11}$  and  $n_{12}$  is a separator<sup>4</sup> detected in the HTML document. *PartitionTree* first traverses to the leaf nodes  $n_{14}$  and  $n_{15}$  (other leaf nodes are processed similarly), assigns the primary types to them and calls the *Analyze* algorithm. It first runs on the node  $n_6$ , and is unable to find any pattern. Since all the child nodes of  $n_6$  are leaf nodes, *Analyze* skips propagating the grandchildren and just assigns  $T_2T_3$  as the compound type of  $n_6$ . *Analyze* continues on  $n_2$  where

<sup>4</sup>A *separator* denotes a horizontal or vertical blank region in a Web page. Its purpose is to indicate the boundary between visible content. Examples of separators include stand-alone BR tags, blank table cells and images used as spacers.

still no pattern can be found. At this time all grandchildren of  $n_2$  are propagated up. Node  $n_6$  gets deleted;  $n_{14}$  and  $n_{15}$  become the immediate children of  $n_2$ . The nodes  $n_3$  and  $n_4$  are processed similarly. This leads to the intermediate tree in Figure 8(b), where  $n_2$ ,  $n_3$  and  $n_4$  are assigned the compound types shown on their left.

Now we take a closer look at the partition scheme generation and its evaluation. In the sequel, for simplicity, we will refer to a node by its type unless otherwise stated. Therefore, the type sequence can also be regarded as the sequence of nodes in the DOM tree. Next we generate the partition scheme of  $S$  for a pattern  $p$  using *GenerateScheme* below.

---

**Algorithm 3** GenerateScheme( $p, S$ )

---

**Require:**  $p$ : a maximal pattern.  $S$ : a sequence of nodes.

- 1: Partition  $S$  into  $\beta_0\gamma\beta_1\gamma\dots\gamma\beta_m$  where  $\gamma = p$  and  $\beta_i$  represents the node sequence between each pair of patterns. Set the initial partition  $\rho_i$  to the  $i_{th}$   $\gamma$ .
  - 2: For each  $\beta_i$ , split it to  $\beta_{i,1}$  and  $\beta_{i,2}$  if there is a *separator* between  $\beta_{i,1}$  and  $\beta_{i,2}$ .
  - 3: For each  $\beta_i$ , if the nodes in  $\beta_i$  have different parent nodes, split it to  $\beta_{i,1}$  and  $\beta_{i,2}$  such that the nodes in each of them come from one parent node.
  - 4: For any sequence  $\rho_i\beta_i\rho_{i+1}$ , if after all the above steps, it becomes  $\rho_i\beta_{i,1}\beta_{i,2}\dots\beta_{i,k}\rho_{i+1}$ , we merge  $\beta_{i,1}$  into  $\rho_i$  and  $\beta_{i,k}$  into  $\rho_{i+1}$ .
  - 4: The final scheme is  $\{\rho_1, \rho_2, \dots, \rho_m\}$ .
- 

After a partition scheme is created under node  $n$ , we assign a “goodness” score to it. This is done as follows: suppose the scheme  $\lambda$  has  $n_\lambda$  partitions under node  $n$ . Let us denote each partition as  $\rho_i^\lambda, 1 \leq i \leq n_\lambda$ . Then the score for  $\lambda$  is:

$$\frac{2 \times |\lambda| \times n_\lambda}{|\lambda| + n_\lambda} - \sum_{1 \leq i \leq n_\lambda} (|\rho_i^\lambda| - \overline{|\rho^\lambda|})^2 \quad (1)$$

where  $|\rho_i|$  denotes the size of the partition  $\rho_i$ .

The first term in the score is the harmonic mean of coverage and number of partitions, and the second term is the standard square deviation of the sizes of all the partitions. The combination is based on the observation that the semantically related items usually have similar size of contents (low standard square deviation), and we prefer the scheme whose coverage and the number of partitions are both large (high harmonic mean).

We pick the partition scheme with the highest score and do the partitioning based on the selected scheme. The partitioning step first removes all the substructure in the child nodes, resulting a 1-level tree rooted at  $n$ . It then aggregates all the child nodes in partition  $\rho_i$  under a newly created node  $n_{\rho_i}$ , namely *pattern node*, whose type is set to the pattern  $p$ .  $n_{\rho_i}$  is then inserted as the child of  $n$ .

Let us continue the example in Figure 8. When *Analyze* is invoked on  $n_1$ , the algorithm assigns  $n_1$  with the type sequence of its children  $T_1T_2T_3T_4T_5T_2T_3T_6T_7T_1T_2T_3$ . The maximal patterns discovered are  $p_1 = T_2T_3$  and  $p_2 = T_1T_2T_3$ . Algorithm

(a)

(b)

(c)

Fig. 8. An example for the partitioning algorithm.

*GenerateScheme* is invoked on each of the maximal patterns. For  $p_1$ , the initial partition scheme is  $[T_1](T_2T_3)[T_4T_5](T_2T_3)[T_6T_7T_1](T_2T_3)$  (we use parenthesis to indicate the boundaries of  $\rho_i$ , and bracket to indicate the boundaries of  $\beta_i$ ). Since there is a separator between  $n_{11}$  and  $n_4$ ,  $\beta_2 = T_6T_7T_1$  is split into  $[T_6T_7][T_1]$ . Also since  $n_4$  and  $n_5$  have different parent nodes,  $\beta_1 = T_4T_5$  is split into  $[T_4][T_5]$ . The same holds for  $\beta_{2,1} = T_6T_7$ . Accordingly the type sequence is now  $[T_1](T_2T_3)[T_4][T_5](T_2T_3)[T_6][T_7][T_1](T_2T_3)$ . *GenerateScheme* modifies the scheme to  $(T_1T_2T_3T_4)(T_5T_2T_3T_6)[T_7](T_1T_2T_3)$ . This scheme is assigned the score 4.05 ( $|\lambda| = 11, n_\lambda = 3, \rho_1 = \rho_2 = 4, \rho_3 = 3$ ) by the goodness scoring formula given in (1).

Similarly the maximal pattern  $p_2$  results in the partition scheme  $(T_1T_2T_3T_4)[T_5T_2T_3T_6][T_7](T_1T_2T_3)$ , with the score of 2.93 ( $|\lambda| = 7, n_\lambda = 2, \rho_1 = 4, \rho_2 = 3$ ). Therefore the partition scheme based on  $p_1$  has the highest score and is selected to actually partition the sequence. The partitioning step removes  $n_2, n_3$  and  $n_4$ . We insert the new *pattern nodes*  $P$ 's (see Figure 8(c)) according to the scheme. The newly inserted pattern nodes are assigned the compound type  $T_p = T_2T_3$ . The type sequence of  $n_1$  becomes  $T_pT_pT_7T_p$ . Since no more patterns can be found in the new type sequence, the loop ends and  $n_1$  is assigned the type  $T_p$ . The final result of *Analyze* on  $n_1$  is shown in Figure 8(c), where the nodes labeled with  $P$  are the newly inserted pattern nodes.

Finally in the restructured tree, known also as the *partition tree*, there are three classes of internal nodes: (i) *group* - which encapsulates repeating patterns in its immediate children type sequence, (ii) *pattern* - which captures each individual occurrence of the repeat pattern, 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 items in the search results list in Figure 1(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 7(a) is shown Figure 7(b). The partition tree represents the logical organization of the page's content.

## 4.2 Concept Identification

After a partition tree is generated, our task is to associate *an instance of a concept* to a particular node in the partition tree. The subtree rooted at such a node represents the smallest *segment* of the Web page covering the concept instance. Towards this, we have designed a feature space for partition tree nodes. For each

node in the partition tree, its features are collected as a vector in the feature space. The features are then used to build a statistical concept model. The model is trained using pre-labeled nodes. For a new partition tree, each node is ranked by the concept model and the one with the highest rank is selected as the instance of the concept. Below we give a description of the feature space.

4.2.1 *Feature space.* Each node in the partition tree can be represented by a vector of values representing its attributes, namely *features*. The set of feature vectors forms a multi-dimensional *feature space*. In our concept identification problem, given a partition tree node  $p$ , the value of feature  $f_i$ , denoted as  $n_{f_i,p}$ , is the frequency of occurrence of feature  $f_i$  in  $p$ . The features of  $p$  are categorized into three types, namely *word*, *p-gram* and *t-gram*.

(i) *Word features* - These are features drawn from the text encapsulated within a partition tree node. Intuitively, an instance of a certain concept usually contains a set of particular words occurring more often than others. For example, given the **Search Form** concept, an instance usually contains words like “search” and “find”. Such concept-specific words are of great importance in identifying the instance. Word features are the most widely used features in text-related machine learning tasks such as text categorization.

For a leaf node in the partition tree, its word features are drawn from its own text while for an internal partition tree node, the words present in all the leaves within the subtree rooted at it are aggregated. All the words are case-insensitive, and stop words (e.g., “the”, “in” and “please”) are ignored in both cases.  $n_{f_i,p}$  is the number of times the word  $f_i$  occurs in the text of  $p$ .

(ii) *p-gram features* - These are features representing the presentation of content. We have observed that in Web sites sharing similar content semantics, the presentation of a semantic concept in those sites exhibits similarity. For instance, in Figure 1(b), each item is presented as a link with the item name, followed by a short text description, and ending with miscellaneous text information. Similar visual presentation can also be found on other sites. A p-gram feature captures these presentation similarities. The basic p-gram features are *link*, *text*, and *image* found in the leaf nodes of a partition tree. Recall that during structural analysis, pattern nodes aggregate every individual repeat in a type sequence. Since repeats are typically associated with similar visual presentation, complex p-gram features are constructed only at pattern nodes by concatenating the p-gram features of their immediate children nodes. Internal nodes aggregate basic and possibly complex p-grams from the subtrees rooted at them. Let  $n_{f_i,p}$  be the number of times  $f_i$  occurs in the subtree rooted at  $p$ . For instance, in the left tree of Figure 9, the p-gram feature at the pattern node labeled “P” is  $\langle \text{text} \cdot \text{link} \rangle$ , while in the right tree of Figure 9, the node labeled “B” at the same position has the p-gram features  $\{\langle \text{text} \rangle, \langle \text{link} \rangle\}$ .

(iii) *t-gram features* - While p-gram features capture the visual presentation, t-gram features represent the structure of the partition tree. Recall that internal partition tree nodes can be either *group*, *pattern*, or *block* while link, text, and image are the different classes of leaf nodes. The structural arrangement of these classes of nodes also characterizes a concept and this is what is captured by t-gram features.

Fig. 9. t-gram Features

Given a partition tree node with  $N$  nodes in its subtree, the complete structural arrangement within the node can be described in terms of a set of subtrees of  $k$  ( $2 \leq k \leq N$ ) nodes where each subtree is an arrangement of group, pattern, block, link, text, or image type nodes. Since enumerating all these subtrees has exponential complexity, we restrict our analysis to subtrees of 2 nodes. When  $k = 2$  the t-gram is essentially a parent-child feature. For instance, in Figure 9, when  $k = 2$  the t-gram feature space of the left tree is  $\{ \langle G, P \rangle, \langle P, Text \rangle, \langle P, Link \rangle \}$ , and the right tree is  $\{ \langle B, B \rangle, \langle B, Text \rangle, \langle B, Link \rangle \}$ , where  $G$  and  $B$  are labels of group and block nodes respectively.

**4.2.2 Concept Model.** Our concept identification task is to assign a score  $s_{n,c}$  to every node  $p$  in the partition tree given a concept  $c$ . The node covering only the instance of  $c$  gets the highest score. Specifically, given  $\mathcal{N}$  denotes the domain of nodes,  $\mathcal{C}$  is a domain of concepts and  $\mathcal{R}$  is the domain of real numbers, the task is to approximate the *target function*  $\phi : \mathcal{N} \times \mathcal{C} \rightarrow \mathcal{R}$  with  $\tilde{\phi} : \mathcal{N} \times \mathcal{C} \rightarrow \mathcal{R}$  called *concept model*, such that for any node  $n$  assigned highest score by  $\phi$ ,  $\tilde{\phi}$  will perform similarly (i.e., very likely to assign the highest score also).

We use *Bayesian learning method* for the concept identification. Specifically we are looking for the node  $p$  that has the maximum  $\phi(p, c) = P(c|p)$  (i.e., given a node  $p$ , the probability that it is an instance of a concept  $c$ ) among all the nodes in the partition tree. By Bayes theorem, we have

$$\arg \max_{p \in \mathcal{N}} P(c|p) = \arg \max_{p \in \mathcal{N}} \frac{P(p|c)P(c)}{P(p)} = \arg \max_{p \in \mathcal{N}} P(p|c)$$

where  $P(c)$  is a constant and  $P(p)$  is uniformly distributed. Therefore, our *concept model*  $\tilde{\phi}$  is defined as  $P(p|c)$ , which consists of two components: (i) a probability distribution on the frequency of occurrence of the word, p-gram, and t-gram features of  $p$ , and (ii) a probability distribution on the number of nodes present in the entire subtree of  $p$ . A collection of partition trees whose nodes are (manually) labeled as concept instances serve as the training set for learning the parameters of these distributions.

A maximum likelihood approach is used to model the distribution of a feature in a concept. Given a training set of  $L$  partition tree nodes identified as instances of concept  $c_j$ , we are interested in the features occurring in  $L$ . The probability of occurrence of a feature  $f_i$  in  $c_j$  is defined using Laplace smoothing as:

$$P(f_i|c_j) = \frac{\sum_{p \in L} n_{f_i,p} + 1}{\sum_{i=1}^{|F|} \sum_{p \in L} n_{f_i,p} + |F| + 1}$$

where  $n_{f_i,p}$  denotes the number of occurrences of  $f_i$  in partition node  $p$  and  $|F|$  is the total number of unique feature including word, p-grams, and t-grams. Features that do not appear in the training set have the fixed probability of

$$\frac{1}{\sum_{i=1}^{|F|} \sum_{p \in L} n_{f_i,p} + |F| + 1}$$

Fig. 10. Extracted Concept Tree

The number of nodes within the subtree of a partition tree node for a concept  $c_j$  is modeled as a Gaussian distribution with parameters mean  $\mu_{c_j}$  and variance  $\sigma_{c_j}$  and is defined as:

$$\mu_{c_j} = \frac{\sum_{p \in L} |p|}{|L|}, \sigma_{c_j} = \sqrt{\frac{\sum_{p \in L} (|p| - \mu_{c_j})^2}{|L| - 1}}$$

where  $|p|$  is the number of nodes within the subtree of  $p$ , and  $|L|$  is the number of training examples.

For a node  $p$  of any new partition tree, we use a modified multinomial distribution to model the likelihood  $P(p|c_j)$ :

$$P(p|c_j) = \left( \frac{\bar{N}!}{N_{f_1,p}! \cdots N_{f_{|F|},p}!} \right) \times \prod_{i=1}^{i=|F|} P(f_i|c_j)^{N_{f_i,p}}$$

where  $\bar{N} = K \times e^{(|p| - \mu_{c_j})^2 / (2\sigma_{c_j}^2)}$ , with  $K$  being a normalized total feature frequency count,  $|p|$  being the total number of partition tree nodes within the subtree rooted at  $p$ , and  $N_{f_i,p}$  is a scaled value of  $n_{f_i,p}$  such that  $\sum_i N_{f_i,p} = \bar{N}$ . The normalization of the multinomial coefficient ensures that the subtree of  $p$  with the size closer to  $\mu_{c_j}$  will have greater likelihood to  $c_j$ .

Note that the above formulation of the likelihood takes into consideration both the *number of nodes* within  $p$  as well as the frequencies of the various features in the content encapsulated within  $p$ . This results in a tight coupling between content analysis and document structure during concept identification. The partition tree node with the maximum likelihood value is identified as the concept instance. The concept tree created with the identified concept instances is shown in Figure 10. Observe that the irrelevant content in the original DOM tree has been filtered.

## 5. EVALUATION

Herein we report on the experimental evaluation of the learned process model, concept extraction and the integrated Guide-O system. Thirty Web sites spanning the three content domains of *books*, *consumer electronics* and *office supplies* were used in the evaluation.

To create the concepts in the ontology we did an analysis of the transactional activities done by users on the Web in each of the three domains. Based on the analysis we came up with an “inclusive” set of concepts in Table I.

### 5.1 Process Model

We collected 200 example transactional sequences from 30 Web sites. These were labeled sequences whose elements are concept operations as illustrated in the sequences used in Figure 6. A number of CS graduate students were enlisted for this purpose. Specifically each student was told to do around 5 to 6 transactions with a Web browser and the sequences were generated by monitoring their browsing activities. We used 120 of these sequences spanning 15 Web sites (averaging 7 to 9 sequences per site) as the training set for learning the process model. The

(a) Recall/Precision (b) Failure Analysis

Fig. 11. Evaluation of the Learned Process Model.

Fig. 12. Guide-O-Speech Performance.

Fig. 13. Recall for Concept Extraction.

remaining 80 were used for testing its performance. The learned model is shown in Figure 4.

The first metric that we measured was its recall/precision<sup>5</sup>. Figure 11(a) shows the recall/precision values for model learning in each of the three domains. Recall/precision values were 90% and 96% for the books domain, 86% and 88% for the consumer electronics domain and 84% and 92% for the office supplies domain.

The second metric we measured was the number of transitions that remained to be completed when a true trace (completed transaction) in the test set failed to reach the final state (check\_out state). Figure 11(b) shows the failure curve. Analysis of the failure curve indicates that almost half of the failure traces end one hop away from the final state of the model and only 10% of the failures end three or more hops away from the final state. This means that fast error recovery techniques can be designed with such a process model.

## 5.2 Concept Extraction

We built a statistical concept model for each of the concepts in Table I. Recall that the five concepts in the upper half of the table are generic for all the three domains whereas those in the lower half are domain specific. For instance the feature set of a list of books differs from that of consumer electronic items. So we built one model for each concept in the upper half of the table and three - one per domain - for each concept in the lower half.

The concept model were built using the techniques in Section 4. To build the model for each of the five generic concepts we collected 90 pages from 15 out of the 30 Web sites. For each of the domain specific concept we collected 30 Web pages from five Web sites that catered to that domain.

Note that pages containing more than one concept were shared during the building of the respective concept models. These models drive the concept extractor at runtime. We measured the recall<sup>6</sup> of the concept extractor for each concept in the ontology. Roughly 150 Web pages collected from all of 30 Web sites was used as the test data. Figure 13 shows the recall values for all of the 10 concepts in each of the three domains.

An examination of the Web Pages used in the testing revealed that the high

<sup>5</sup>Recall for a process model is the ratio of the number of completed transactions accepted by the model over the total number of completed transactions. For Precision, this denominator becomes the total number of accepted transactions (completed and not completed).

<sup>6</sup>Recall value for a concept is the ratio of the number of correctly labeled concept instances in Web pages over the actual number of concept instances present in them.

recall rates (above 80% for “Item Taxonomy”, “Search Form”, “Add To Cart”, “Edit Cart”, “Continue Shopping” and “Checkout”) is due to the high degree of consistency of the presentation styles of these concepts across all these Web sites. The low recall figures for the “Item Detail” (about 65% averaged over the three domains) and “Shopping Cart” (about 70%) is mainly due to a high degree of variation in their features across different Web sites. A straightforward way to improve the recall of such concepts is to use more training data. However even this may not help for concepts such as “Add To Cart” that rely on keywords as the predominant feature. Quite often these are embedded in a image precluding textual analysis. It appears that in such cases local context surrounding the concept can be utilized as a feature to improve recall.

We also observed that Web Pages with sufficient textual content relevant to the concept was identified as an instance of that concept with higher accuracy. Furthermore, concept extraction performed better for schematic Web pages (i.e. Web pages generated from a template), because structural feature extraction is dependent on the presentation style and organization of different elements of the page.

### 5.3 Integrated System

We conducted quantitative and qualitative evaluation of the integrated system; the former measured time taken to conduct Web transactions and the latter surveyed user experience. The evaluation was conducted separately for each domain using the corresponding domain-specific concept extractors (see Section 5.2).

**Experimental Setup** We used a 1.2 GHz desktop machine with 256 MB RAM as the computing platform for running the core Guide-O system (shown within the outermost box in Figure 3). We also used it to run Guide-O-Speech. To do that we installed our own VoiceXML interpreter along with off-the-shelf speech SDK components. Guide-O-Mobile was architected to run in a client/server mode with the machine running the core Guide-O system as the server and a 400MHz iPaq with 64MB RAM as the client. Thirty CS graduate students were used as evaluators. Prior to evaluation they were trained on how to use the two systems. We chose 18 Web sites (6 for each domain) to evaluate Guide-O-Mobile and 9 (3 for each domain) to evaluate Guide-O-Speech. We conducted roughly 5 to 6 transactions on each of them and calculated mean ( $\mu$ ) and standard deviation ( $\sigma$ ) for all the measured metrics. Over 93.33% of those transactions could be completed. Evaluation of the integrated system discussed below is based on these completed transactions.

#### **Quantitative Evaluation of:**

(i) **Guide-O-Speech:** Evaluators were asked to measure the total time taken to complete the transactions with Guide-O-Speech. The screen was disabled and evaluators had to interact with the system using a headphone and keyboard. For baseline comparison evaluators were also asked to conduct another experiment with the JAWS screen reader on the same set of transactions. For every page used in a transaction sequence they were asked to record the time it took JAWS to read from the beginning of the page until the end of the selected concept’s content. The sum of these times over all the pages associated with the transaction denotes the time taken to merely listen to the content of the selected concepts with a screen

Web Sites	Voice Interactions		Pages Explored		Time Taken (using)			
					Guide-O-Speech		JAWS Screen Reader	
	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$
Amazon	8	0	5	0	306.1	13.73	1300	120.2
BN	9	0.82	5	0	351.5	40.78	1130	176.78
AbeBooks	10.8	0.96	6	0.82	384.9	31.35	700	176.78
Amazon	9.8	3.1	5.2	1.26	386.6	60.15	1413	130.11
CompUSA	9	1.15	6	0.82	360.5	13.34	931.5	211.42
tigerdirect	9.4	1.29	6	0.82	357.7	30.07	1293	212.13
OfficeMAX	10	0.82	5.8	0.96	341.6	23.69	686	61.52
OfficeDepot	10	2.16	6	1.41	309.9	45.87	604	55.23
QuillCorp	9.6	1.73	5.2	0.96	382.6	49.61	625	51.84

\*All times are in seconds

Table IV. Guide-O-Speech Performance.

C1	Did you find the concepts used in doing the transaction informative?	93.33%
C2	Do they capture all the useful information in the Web pages?	76.67%
C3	Did they help in accomplishing the transaction?	86.67%
S1	How often were you able to find the desired item?	96.67%
S2	How often were you able to complete the transaction for the item found?	93.33%
S3	Do you feel that the system restricted your navigation?	80%
S4	Did you find the system useful for conducting transactions?	96.67%

Table V. Questionnaire with Response

reader. Table IV lists the metrics measured for each Web site.

From 5 to 6 transactions on each Web sites (3 Web sites for each content domain), we calculated mean ( $\mu$ ) and standard deviation ( $\sigma$ ) of all the metrics that are listed in Table IV. Figure 12 shows comparative performance of guide-o-speech and JAWS.

Observe that Guide-O-Speech compares quite favorably with the best-of-breed screen readers and hence can serve as a practical assistive device for doing online transactions.

**(ii) Guide-O-Mobile:** Evaluators first conducted transactions with Guide-O-Mobile. Next they did the same set of transactions with the original pages loaded onto the iPaq, *i.e.* the page fetched from the Web was dispatched “as is” to the iPaq. Figure 14(a) lists the metrics measured. The “page load time” column contains the times taken to load the original pages into the iPaq. The “analysis time” is the time taken by the Guide-O server to fetch pages from the Web do content analysis, generate the DHTML page of the extracted concepts and upload them onto the handheld. Interaction time is measured from the moment a page is loaded into the iPaq until the user takes a navigable action like a form-fillout, clicking

a link, etc. Figure 14(b) and (c) presents graphically the overall and interaction times respectively. Observe the significant decrease in interaction time to complete a Web transaction using Guide-O-Mobile. This reduction is mainly due to the filtering away of irrelevant content by the process model. Consequently the user avoids needless scrolling steps. Furthermore since the transaction is goal directed by the process model the user is able to complete the transaction in fewer steps. Another advantage of filtering away irrelevant content is the relatively small page load times. These times have been absorbed in the “analysis time”.

### Qualitative Evaluation:

To gauge user experience we prepared a questionnaire for the evaluators (see Table V). They were required to answer them upon completing the quantitative evaluation. The questions were organized into two broad categories – system (S1 to S4) and concepts (C1 to C3) – the former to assess the overall functionality and usability of the Guide-O system and the latter to determine the effectiveness of the semantic concepts in doing Web transactions. The questions were quite generic and applicable to both Guide-O-Mobile and Guide-O-Speech. All of the concept questions (C1 to C3) and questions S3 and S4 required a yes/no response. From the responses to S1 and S2 we computed the mean percentage shown in the table.

### Results:

A large percentage of evaluators felt that the concepts presented were self explanatory and contained enough information based on which they could take the right steps to make progress on their transaction (Response to Question C1). Some evaluators felt that notification of promotional offers, coupons, etc. was important and that such concepts ought to be presented (Response to Question C2).

Most were able to find the items they were looking for (Response to Question S1). However at times they were unable to complete the transaction (The “no” response to Questions C3 and the unfinished transactions in S2). Analysis of such transactions revealed that in many cases the problem arose because: (a) the expected concepts in a state were not extracted; (b) the extracted concepts were mislabeled; (c) the model could not make the correct transition. The last two problems could be addressed by training the concept identifier and the process model with more examples.

Quite a few evaluators felt that they expected more flexibility on how they can complete the transaction (Response to Question S3). Observe that the number of possible paths to complete a transaction is limited by the training data and hence this criticism can be addressed with more training. Overall they all felt that the system was adequate to do their tasks (Response to Question S4).

Evaluators also had general comments. In particular they all felt that the system requires help utilities to assist users to become familiar with the use and effect of each concept.

## 6. RELATED WORK

The work described in this paper has broad connections to research in Web services, semantic understanding of Web content, automata learning, non-visual Web access and browsing with mobile devices.

**Web Services:** This is an emerging paradigm that focuses on technologies that

Web Sites	Original Web Page in Hand-held				Guide-O-Mobile			
	page load time		interaction time		analysis time		interaction time	
	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$
Amazon	236.2	11.78	236	15.48	74.4	5.98	127	9.77
BN	293.2	31.59	194.6	34.79	73.2	7.95	101	14.23
Khazana	285.2	4.66	204.2	7.4	72.4	5.5	96.6	11.78
Blackwell	337.2	39.96	176	35.94	73.6	3.13	107.2	5.63
Angusrobertson	208.6	12.97	170.6	14.01	67.6	4.67	88.8	23.36
AbeBooks	195.6	23.04	179.8	9.83	72.6	4.93	102.6	13.09
Buy	384.2	46.82	311.6	21.17	87.4	8.47	160.4	13.24
Amazon	237.6	6.19	232.6	8.26	75.2	5.12	111.8	19.28
Bestbuy	283	24.1	251.2	16.63	83.8	5.32	130.2	22.31
CompUSA	272	11.22	227.4	12.2	84.6	5.68	115.6	7.3
eCost	224.6	26.54	287.4	42.65	78.2	11.3	142.6	14.89
outpost	326.4	45.8	179.6	17.69	65.6	6.54	134	20.19
tigerdirect	239	8.12	193.6	32.97	70	7.04	116.6	16.56
OfficeMAX	179.2	42.61	220.8	25.59	77.8	8.44	107.2	20.22
OfficeDepot	189.4	20.01	195	9.51	73.6	7.23	127.6	7.5
Walmart	300.6	54.16	220.2	23.92	95.4	12.66	123.2	17.71
Shop	383.6	48.15	227.4	10.76	89.4	11.61	99.4	16.83
QuillCorp	328.4	41.36	191.4	22.32	71.8	8.41	79.8	6.53

\*All times are in seconds.

(a)

(b) Overall time (c) Interaction time

Fig. 14. Guide-O-Mobile Performance.

let service providers to export their functionalities on the Web so as to facilitate automated e-commerce. It has given rise to standardization efforts resulting in languages such as WSDL for describing services, SOAP for accessing services, UDDI for service discovery and integration, BPEL4WS for business process specification, and OWL-S as an ontology for semantic description of service metadata. Service providers are beginning to utilize them for exposing their services (see for example <http://www.amazon.com/webservices>). The complimentary task of annotating service descriptions with semantic metadata has been addressed in [Hess et al. 2004; Patil et al. 2004; Sabou et al. 2005]. In contrast to these works we address a different kind of annotation problem, namely automatic annotation of different kinds of concepts that can occur in a Web page. Web services expose very basic functionalities which by themselves are not sufficient to conduct complex transactions. For instance, Amazon's Web service exposes basic tasks such as searching for a product, adding a product into the shopping cart, etc. One has to compose these primitive services in order to perform complex transactions. This problem has been receiving attention lately [Agarwal et al. 2003; Chen et al. 2003; Rao et al. 2004; Traverso and Pistore 2004; Wombacher et al. 2004]. All these works typically use

process definitions and an ontology to create the composite service with varying degrees of automation. Note that our technique is based on composing operations over Web pages instead of services. A vast majority of transactions on the Web are still conducted over HTML pages. This focus on Web pages is what sets our work apart from those in Web services. However using our approach, users can not understand the goal of the step because this has been split in several Web pages. Thus users lose the global vision of the transaction. We are currently working on an implementation, where the local step of the user can be visually shown in the context of the global task – the process model and the semantics.

Using standard web services constrains the user to a set of pre-defined tasks (e.g. searching for a product). However, our approach does not confine a user to whatever service is exposed by the provider. Using our transactional approach, users are able to create services (e.g. service for buying airline ticket, service for job search) which can accomplish his personal goal. Thus we are providing a scalable technique for creating services.

**Semantic analysis of Web content:** The essence of the technique underlying our content analysis module is to partition a page into segments containing “semantically” related items and classify them against concepts in the ontology. Web page partitioning techniques have been proposed for adapting content on small screen devices [Buyukkoten et al. 2001; Buyukkoten et al. 2000; Chen et al. 2003; Yin and Lee 2004], content caching [Ramaswamy et al. 2004], data cleaning [Song et al. 2004; Yi and Liu 2003], and search [Yu et al. 2003]. The idea of using content similarities for semantic analysis was also recently explored in [Zhang et al. 2004] in the context of Web forms. The fundamental difference between our technique and all the above works is the integration of inferring the logical structure of a page (see the partition tree in Figure 7(b)) with feature learning. This allows us to define and learn features, such as p-grams and t-grams, using the partition tree.

Concept identification in Web pages is related to the body of research on semantic understanding of Web content. Powerful ontology management systems and knowledge bases have been used for interactive annotation of Web pages [Heflin et al. 2003; Kahan et al. 2001]. More automated approaches combine them with linguistic analysis [Popov et al. 2003], segmentation heuristics [Dill et al. 2003], and machine learning techniques [Cimiano et al. 2004; Hammond et al. 2002]. Our semantic analysis technique is an extension of our previous work [Mukherjee et al. 2005] and, in contrast to all the above, does not depend on rich domain information. Instead, our approach relies on light-weight features in a machine learning setting for concept identification. This allows users to define *personalized* semantic concepts thereby lending more flexibility to modeling Web transactions. It should also be noted that the extensive work on wrapper learning [Laender et al. 2002] is related to concept identification. However, wrappers are syntax-based solutions and are neither scalable nor robust when compared to semantics-based techniques.

**Process Model Learning:** Our work on learning process models from user activity logs is related to research in mining workflow process models (see [van der Aalst and Weijters 2004] for a survey). However, our current definition of a process is simpler than traditional notions of workflows. For instance, we do not use sophisticated synchronization primitives. Hence we are able to model our processes

as DFAs instead of workflows and learn them from example sequences. Learning DFAs is a thoroughly researched topic (see [Murphy 1996] for a comprehensive survey). A classical result is that learning the smallest size DFA that is consistent with respect to a set of positive and negative training examples is NP-hard [Angluin 1978; Gold 1978]. This spurred a number of papers describing efficient heuristics for DFA learning (*e.g.* [Murphy 1996; Oncina and Garc 1991]). We have not proposed any new DFA learning algorithm for our work. Instead we adapted the simple yet effective heuristic with low time complexity described in [Oncina and Garc 1991]. Navigating to relevant pages in a site using the notion of "information scent" has been explored in [Chi et al. 2001]. This notion is modeled using keywords extracted from pages specific to that site. In contrast our process model is domain specific and using it a user can do online transactions on sites that share similar content semantics.

**Content Adaptation:** Adapting Web content for browsing with handhelds and speech interfaces is an ongoing research activity. Initial efforts at adapting Web content for small-screen devices relied on WML (Wireless Markup Language) and WAP (Wireless Application Protocol) for designing and displaying Web pages [Kaasinen et al. 2000]. These approaches imposed additional burden on Web designers to create separate WML content. In contrast, we do not require any additional effort on the part of content providers - Guide-O takes the original Web pages, processes them, and presents them to the users.

Subsequent research [Buyukkoten et al. 2001; Buyukkoten et al. 2000; Chen et al. 2003; Yang and Wang 2003; Yin and Lee 2004] dealt with adapting HTML content onto these devices by organizing the Web page into tree-like structures and summarizing the content within these structures for efficient browsing. They were effective for ad-hoc exploratory browsing. However, summary structures often cause needless navigational steps when a user is interested in some specific content. In our work we need to first filter away the content based on the transactional context represented by the states of the process model. Summarization can be used to present the filtered content succinctly.

Page-splitting techniques used in content adaptation for small-screen devices are described in [Chen et al. 2003; Xiangye Xiao and Fu 2005]. A page-analysis technique is proposed in [Chen et al. 2003] to analyze the structure of a Web page and split it into small logically-related units that fit on the screen of a mobile device. In the case when a Web page is not suitable for splitting, an auto-positioning method or scrolling-by-block is used to facilitate browsing without splitting the original Web page. In [Xiangye Xiao and Fu 2005], a Web page that does not fit on a small screen is transformed into a set of pages, each of which fits into the screen. Page-splitting techniques could be somewhat disorienting for the users familiar with the original page structure, however, browsing full pages on mobiles is far more disorienting because most of the mobile browsers distort the pages while rendering. Guide-O also uses a partitioning algorithm to split Web pages into partitions, and then, identify the concepts. However, Guide-O goes beyond the above-mentioned systems, because it presents the users with the concepts relevant to the transactional step.

A number of research projects have tried to condense Web pages by displaying thumbnails and summarizing Web content [Baluja 2006; Milic-Frayling and Som-

merer 2002]. For example, SmartView [Milic-Frayling and Sommerer 2002] uses a page-splitting technique to group the elements of a Web page and present them together while allowing a user to zoom into the individual elements. The work described in [Baluja 2006] segments Web pages into regions, presents each region using a thumbnail, and also allows a user to zoom into a region by pressing a single key. In the latter work, the segmentation task is formulated as a machine learning problem based on entropy reduction and decision tree learning. Zooming and thumbnail presentation styles can facilitate the identification of relevant sections in Web pages, but they do not pinpoint the concept instance in a page. These features, however, can further enhance Guide-O-Mobile’s usability.

Recently, desktop-class applications for browsing on mobile devices appeared on the market. Apple iPhone [iPhone ], for instance, provides access to desktop-class applications and various software, (e.g. rich HTML email, full-featured Web browsing, widgets, etc.). The ThunderHawk [Thunderhawk ] mobile Web browser offers desktop level browsing experience to mobile users. It provides rapid operation, zooming capabilities, different viewing modes (e.g. full screen and split screen), etc. Users can adjust a Web page’s resolution according to their preferences and view the page with minimal scrolling. Desktop-class browsers are the future of mobile browsing. However, they also require user interaction to scroll and zoom in on the information. Therefore, they could also benefit from concept instance identification and presentation to further improve user experience.

Technologies for making the Web accessible to visually handicapped individuals via non-visual interfaces analogous to [Raman 1998], include work on browser level support (e.g. [Asakawa and Itoh 1998]), content adaptation and summarization (e.g. [Richards and Hanson 2004; Zajicek et al. 1999; Harper and Patel 2005]), ontology-directed exploratory browsing as in our HearSay audiobrowser [Ramakrishnan et al. 2004], and organization and annotation of Web pages for effective audio rendition [Huang and Sundaresan 2000; Pontelli et al. 2002; Takagi et al. 2002].

Some of the most popular screen-readers are JAWS [jaw ] and IBM’s Home Page Reader [Asakawa and Itoh 1998; Takagi et al. 2002]. An example of a VoiceXML browsing system (which presents information sequentially) is described in [NetECHO[tm] ]. All of these applications do not perform content analysis of Web pages. BrookesTalk [Zajicek et al. 1999] facilitates non-visual Web access by providing summaries of Web pages to give its users an audio overview of Web page content. The work described in [Harper and Patel 2005] generates a “gist” summary of a Web page to alleviate information overload for blind users.

How we differ with these works is similar to the differences highlighted above vis-a-vis content adaptation research for handhelds. Specifically we need a more global view of the content in a set of pages in order to determine what should be filtered in a state. An interesting outcome of this generality is that by collapsing all the states in the process model into a single state, our speech-based HearSay system for exploratory browsing becomes a special case of Guide-O-Speech.

## 7. CONCLUSION

The use of complete Web pages for doing online transactions under constrained interaction modalities can cause severe information overload on the end user. Model-directed Web transaction can substantially reduce if not eliminate this problem. Our preliminary experimentation with Guide-O seems to suggest that it is possible to achieve such reductions in practice. Presently we are conducting preliminary usability studies of Guide-O's speech at Helen Keller Services for the Blind (<http://www.helenkeller.org>) to get feedback from the visually handicapped community.

There are several avenues of future research. In this paper we have presented a client side Web transactional system, where process model resides in client side. In future we will also develop server side Web transactional system, where content providers will annotate the Web pages such that concept instances will be labeled with the concept name. Content providers can easily use XHTML to label concept instances in their Web sites and describe process models specific to their sites. Most online stores generate their Web pages automatically from templates. So we expect that labeling the content of templates will not require extensive effort on the part of content providers. At the same time, the use of XHTML for semantic labeling will impose no limitation, allowing content providers to define their own states and actions as they deem necessary.

We build separate ontology and process model for each type of Web transaction (e.g. online shopping, utility bill payment, flight ticket booking). An interesting problem is to develop a generic ontology and process model to support different types of Web transactions. Currently, we do not consider the security mechanisms defined in the Web sites to handle sensitive information. In future, we will investigate the possibility of handling security mechanisms that are already present in the Web site.

The Guide-O system works with the assumption that Web transaction is performed correctly. We plan to extend our system to incorporate failure scenarios. To handle failure scenarios, we will maintain a history of transactional progress. On failure (e.g. when the system does not identify any concept on the current page), the model will go back to the previous state. Similarly, when a user chooses to go back, the model will go back to the previous state.

In our current framework, Guide-O-Mobile is set up to run as a client/server application with the server doing the bulk of the processing. In the future it should be possible to port all the server steps to the handheld when improvements in handheld hardware and software technologies will provide more memory and a richer set of browser operations than what are currently available in handheld.

In addition, displaying concepts compactly in handhelds along the lines suggested in [Florins and Vanderdonck 2004] will further improve the user's ease of interaction.

Finally, integration of our framework with Web services standards is an interesting problem. Success here will let us specify the process model in BPEL4WS which in turn will enable inter-operation with sites exposing Web pages as well as those exposing Web services.

**Acknowledgements:** This work was supported in part by NSF grants CCR-

0311512 and IIS-0534419.

## REFERENCES

- JAWS Screen Reader. <http://www.freedomscientific.com>.
- AGARWAL, S., HANDSCHUH, S., AND STAAB, S. 2003. Surfing the service web. In *Intl. Semantic Web Conf. (ISWC)*. 211–216.
- ANGLUIN, D. 1978. On the complexity of minimum inference of regular sets. *Information and Control* 39, 3, 337–350.
- ASAKAWA, C. AND ITOH, T. 1998. User interface of a home page reader. In *ACM Intl. Conf. on Assistive Technologies (ASSETS)*. ACM Press, 149–156.
- BALUJA, S. 2006. Browsing on small screens: recasting web-page segmentation into an efficient machine learning framework. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*. ACM Press, 33–42.
- BUYUKKOTEN, O., GARCIA-MOLINA, H., AND PAEPCKE, A. 2001. Seeing the whole in parts: Text summarization for web browsing on handheld devices. In *Intl. World Wide Web Conf. (WWW)*. ACM Press, 652–662.
- BUYUKKOTEN, O., GARCIA-MOLINA, H., PAEPCKE, A., AND WINOGRAD, T. 2000. Power browser: Efficient web browsing for pdas. In *ACM Conf. on Human Factors in Computing Systems (CHI)*. ACM Press, 430–437.
- CHEN, L., SHADBOLT, N., GOBLE, C. A., TAO, F., COX, S. J., PULESTON, C., AND SMART, P. R. 2003. Towards a knowledge-based approach to semantic service composition. In *Intl. Semantic Web Conf. (ISWC)*. 319–334.
- CHEN, Y., MA, W.-Y., AND ZHANG, H.-J. 2003. Detecting web page structure for adaptive viewing on small form factor devices. In *Intl. World Wide Web Conf. (WWW)*. ACM Press, 225–233.
- CHI, E., PIROLI, P., CHEN, K., AND PITKOW, J. 2001. Using information scent to model user information needs and actions on the web. In *ACM Conf. on Human Factors in Computing Systems (CHI)*. ACM Press, 490–497.
- CIMIANO, P., HANDSCHUH, S., AND STAAB, S. 2004. Towards the self-annotating web. In *Intl. World Wide Web Conf. (WWW)*. ACM Press, 462–471.
- DILL, S., EIRON, N., GIBSON, D., GRUHL, D., GUHA, R., JHINGRAN, A., KANUNGO, T., RAJAGOPALAN, S., TOMKINS, A., TOMLIN, J., AND YIEN, J. 2003. SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation. In *Intl. World Wide Web Conf. (WWW)*. 831–840.
- FLORINS, M. AND VANDERDONCKT, J. 2004. Graceful degradation of user interfaces as a design method for multiplatform systems. In *IUI '04: Proceedings of the 9th international conference on Intelligent user interface*. ACM Press, New York, NY, USA, 140–147.
- GOLD, E. M. 1978. Complexity of automaton identification from given data. *Information and Control* 37, 3, 302–320.
- GUSFIELD, D. 1997. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press.
- HAMMOND, B., SHETH, A., AND KOCHUT, K. 2002. Semantic enhancement engine: A modular document enhancement platform for semantic applications over heterogenous content. In *Real World Semantic Applications*, V. Kashyap and L. Shklar, Eds. IOS Press.
- HARPER, S. AND PATEL, N. 2005. Gist summaries for visually impaired surfers. In *ACM Intl. Conf. on Assistive Technologies (ASSETS)*. ACM Press, 90–97.
- HEFLIN, J., HENDLER, J. A., AND LUKE, S. 2003. SHOE: A blueprint for the semantic web. In *Spinning the Semantic Web*, D. Fensel, J. A. Hendler, H. Lieberman, and W. Wahlster, Eds. MIT Press, 29–63.
- HESS, A., JOHNSTON, E., AND KUSHMERICK, N. 2004. Assam: A tool for semi-automatically annotating semantic web services. In *Intl. Semantic Web Conf. (ISWC)*. 320–334.
- HUANG, A. AND SUNDARESAN, N. 2000. A semantic transcoding system to adapt web services for users with disabilities. In *ACM Intl. Conf. on Assistive Technologies (ASSETS)*. ACM Press, 156–163.

- IPhone. <http://www.apple.com/iphone/>.
- KAASINEN, E., AALTONEN, M., KOLARI, J., MELAKOSKI, S., AND LAAKKO, T. 2000. Two approaches to bringing internet services to wap devices. In *Intl. World Wide Web Conf. (WWW)*.
- KAHAN, J., KOIVUNEN, M., PRUD'HOMMEAUX, E., AND SWICK, R. 2001. Annotea: An open rdf infrastructure for shared web annotations. In *Intl. World Wide Web Conf. (WWW)*. ACM Press, 623–632.
- LAENDER, A., RIBEIRO-NETO, B., DA SILVA, A., AND TEIXEIRA, J. 2002. A brief survey of web data extraction tools. *SIGMOD Record* 31, 2.
- MILIC-FRAYLING, N. AND SOMMERER, R. 2002. Smartview: Flexible viewing of web page contents. In *Intl. World Wide Web Conf. (WWW)*. ACM Press, 172.
- MUKHERJEE, S., RAMAKRISHNAN, I., AND SINGH, A. 2005. Bootstrapping semantic annotation for content-rich html documents. In *Intl. Conf. on Data Engineering (ICDE)*. ACM Press.
- MURPHY, K. 1996. *Passively learning finite automata*.
- NetECHO[tm]. <http://www.internetspeech.com>.
- ONCINA, J. AND GARC, P. 1991. *Inferring regular languages in polynomial update time*. World Scientific Publishing. Pattern Recognition and Image Analysis.
- PATIL, A., OUNDHAKAR, S., SHETH, A., AND VERMA, K. 2004. Meteor-s web service annotation framework. In *Intl. World Wide Web Conf. (WWW)*. 553–562.
- PONTELLI, E., XIONG, W., GILLIAN, D., SAAD, E., GUPTA, G., AND KARSHMER, A. 2002. Navigation of html tables, frames, and xml fragments. In *ACM Intl. Conf. on Assistive Technologies (ASSETS)*. ACM Press, 25–32.
- POPOV, B., KIRYAKOV, A., KIRILOV, A., MANOV, D., OGNANOFF, D., AND GORANOV, M. 2003. Kim - semantic annotation platform. In *Intl. Semantic Web Conf. (ISWC)*.
- RAMAKRISHNAN, I., STENT, A., AND YANG, G. 2004. Hearsay: Enabling audio browsing on hypertext content. In *Intl. World Wide Web Conf. (WWW)*. ACM Press, 80–89.
- RAMAN, T. V. 1998. Audio system for technical readings. *1410*, xvi + 121.
- RAMASWAMY, L., IYENGAR, A., LIU, L., AND DOUGLIS, F. 2004. Automatic detection of fragments in dynamically generated web pages. In *Intl. World Wide Web Conf. (WWW)*. ACM Press, 443–454.
- RAO, J., KÜNGAS, P., AND MATSKIN, M. 2004. Logic-based web services composition: From service description to process model. In *Intl. Conference on Web Services (ICWS)*.
- RICHARDS, J. T. AND HANSON, V. L. 2004. Web accessibility: a broader view. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*. ACM Press, New York, NY, USA, 72–79.
- SABOU, M., WROE, C., GOBLE, C., AND MISHNE, G. 2005. Learning domain ontologies for web service descriptions: An experiment in bioinformatics. In *Intl. World Wide Web Conf. (WWW)*. 190–198.
- SONG, R., LIU, H., WEN, J.-R., AND MA, W.-Y. 2004. Learning block importance models for web pages. In *Intl. World Wide Web Conf. (WWW)*. ACM Press, 203–211.
- TAKAGI, H., ASAKAWA, C., FUKUDA, K., AND MAEDA, J. 2002. Site-wide annotation: Reconstructing existing pages to be accessible. In *ACM Intl. Conf. on Assistive Technologies (ASSETS)*. ACM Press.
- Thunderhawk. <http://www.bitstream.com/wireless/index.html>.
- TRAVERSO, P. AND PISTORE, M. 2004. Automated composition of semantic web services into executable processes. In *Intl. Semantic Web Conf. (ISWC)*.
- VAN DER AALST, W. AND WEIJTERS, A. 2004. Process mining: A research agenda. *Computers and Industry* 53, 231–244.
- WOMBACHER, A., FANKHAUSER, P., AND NEUHOLD, E. J. 2004. Transforming bpel into annotated deterministic finite state automata for service discovery. In *Intl. Conference on Web Services (ICWS)*.
- XIANGYE XIAO, QIONG LUO, D. H. AND FU, H. 2005. Slicing\* tree based web page transformation for small displays. In *CIKM*. ACM Press, 303–304.

- YANG, C. AND WANG, F. L. 2003. Fractal summarization for mobile devices to access large documents on the web. In *Intl. World Wide Web Conf. (WWW)*. ACM Press, 215–224.
- YI, L. AND LIU, B. 2003. Eliminating noisy information in web pages for data mining. In *ACM Conf. on Knowledge Discovery and Data Mining (SIGKDD)*. ACM Press, 117–118.
- YIN, X. AND LEE, W. S. 2004. Using link analysis to improve layout on mobile devices. In *Intl. World Wide Web Conf. (WWW)*. ACM Press, 338–344.
- YU, S., CAI, D., WEN, J.-R., AND MA, W.-Y. 2003. Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *Intl. World Wide Web Conf. (WWW)*.
- ZAJICEK, M., POWELL, C., AND REEVES, C. 1999. Web search and orientation with brookestalk. In *Proceedings of Tech. and Persons with Disabilities Conf.*
- ZHANG, Z., HE, B., AND CHANG, K. C.-C. 2004. Understanding web query interfaces: Best-effort parsing with hidden syntax. In *ACM Conf. on Management of Data (SIGMOD)*. ACM Press, 107–118.