# Web Page Analysis for Voice Browsing

Michael K Brown
Avaya Labs, Basking Ridge, NJ
Stephen C. Glinski
Lucent Technologies, Murray Hill, NJ
Brian C. Schmult
Somerset Geographics, Doylestown, PA

## Abstract

Voice User Interfaces (VUI's) are rapidly evolving to include Web based content. The primary motivations are: to provide a widely available means for creating new interactive voice applications; addressing needs for mobility; and addressing issues in accessibility. Much of this activity has focused on creating new markup languages like VoiceXML, now being standardized by the W3C Voice Browser Working Group. However, today there is a very large investment in legacy HTML content that can be automatically converted to produce a voice dialog interface to content not originally intended for voice access. We will describe two approaches, telephone browsing and transcoding, focusing mostly on the former since that work is more mature.

## Introduction

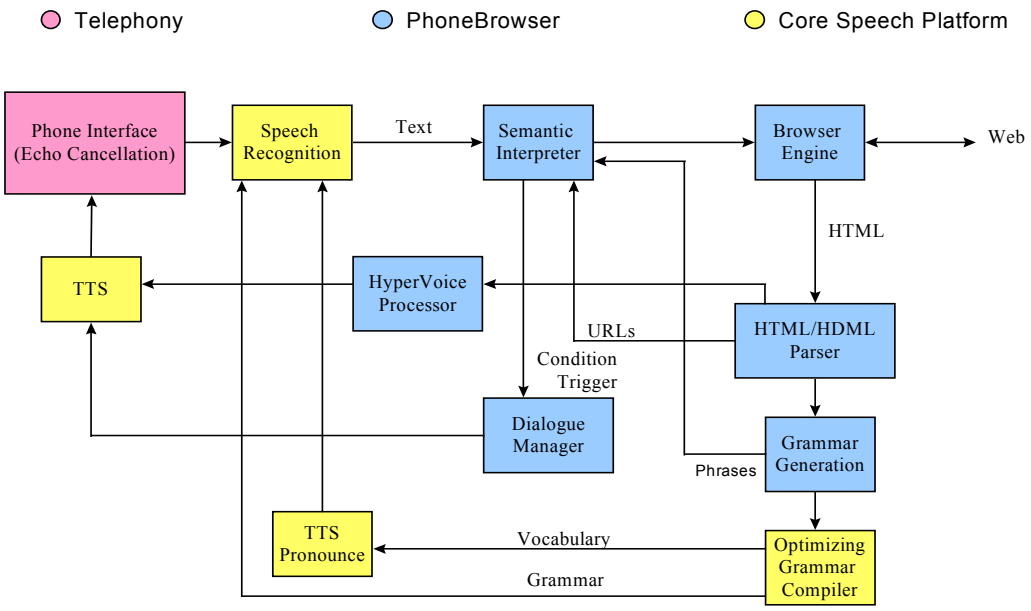
Voice browsing Web documents has moved from the laboratory into viable commercial applications only within the last two years. Prior to this most networked voice applications used proprietary scripting languages to define the user dialog interaction. Starting in October 1998 the W3C (World Wide Web Consortium) formed a new Working Group to address the issues of creating new voice applications on the Web [1]. VoiceXML is a dialog markup language adopted by the W3C as the basis for the standard [2]. This new markup language allows a Webmaster with a voice enabled server platform to write new voice application on a Web server.

In spite of this widely recognized solution for voice applications on the Web there are disadvantages to this solution for many existing Web sites and applications already based on HTML. It takes substantial new investment to rewrite these sites in a new markup language and takes further effort to continue to maintain the parallel representations of the content. Therefore it is useful to use existing HTML content as the basis for some voice applications.

We are taking two approaches to solving the problem of using existing or new HTML for voice applications. We will primarily address the first method, a system called PhoneBrowser [3], and describe how HTML documents are converted for voice interaction. A second method in early stages of development is called transcoding. Transcoding involves converting (re-coding) HTML during transmission over the network, often performed within a proxy Web server. We currently use a Document Object Model based transcoder in cooperation with the PhoneBrowser to summarize and modify HTML content to make it more conversational. We have devised new transcoding methods that provide greater tolerance to structure and format changes to the incoming content. We will briefly revisit transcoding in closing remarks.

PhoneBrowser is speaker-independent voice enabled proxy server that provides automatic voice description of Web documents over any telephone and automatically activates the hyperlink titles by voice command. It is intended to do more than just read the document so it will describe what it sees on the Web page. Ultimately the goal is to provide a conversational system that acts like an agent looking at a Web site (not necessarily one page at a time, presumably designed for visual interface) and carrying on a conversation about the contents of the site.

**Figure 1. PhoneBrowser Architecture**




The first scalable platform was built in 1997 and has since gone through several revisions. We now have about 100 PhoneBrowser telephone ports online. In the next section we describe the latest version of PhoneBrowser and how it processes Web documents to automatically create a voice interface.

## PhoneBrowser Processing

The PhoneBrowser architecture is shown in Figure 1. Not shown in detail is the telephone call and resource management since this is incidental to the document for VUI processing. The process starts with a telephone call placed by a user (alternatively the system can initiate a session). The user is prompted for a logon pass phrase. The users pass phrase establishes the first connection to the Web site associated with that phrase and loads the first Web page. The HTML is parsed, as described later, separating text from other media types, isolating URL from HTML anchors and isolating the associated anchor titles (including ALT fields) for grammar generation.

Grammar generation computes combinations of the words in titles to produce a wide range of alternative ways to say subsets of the title phrase. In this process simple function words (i.e., "and," "or," "the," etc.) are not allowed to occur in isolation where they would be meaningless. Browser control commands are mixed in to control typical browser operations like "go back" and "go home" (similar to the typical browser button commands). Further automatic expansion of the language models for link titles can include thesaurus substitution of words, etc. soon to be implemented. The current processing allows the user to say any keyword phrase from the title with deletions allowed to activate the associated link.

Grammar processing continues with compilation and optimization into a finite-state network where redundancies have been eliminated. The word vocabulary associated with the grammar is further processed by a Text-To-Speech (TTS) pronunciation module that generates phonetic transcriptions for each word of the grammar. Since the TTS engine uses pronunciation rules it is not limited to dictionary words. The grammar and vocabulary are then loaded into the speech recognizer. This process typically takes about a second.

At the same time, the Web document is described to the user, as discussed later. The user may then speak a navigation or browser command phrase to control browsing. Each user navigation command takes the user to a new Web page. If the command is ambiguous the dialog manager collects the possible interpretations

into a description list and asks the user to choose one. Recognizing that a collection of Web pages is inherently a finite-state network, it is easy to see that this mechanism provides the basis for a finite-state dialog and Web application control system.

## Document Processing

Document analysis is performed in the HTML parser, grammar generator, and HyperVoice processor modules. The typical HTML Web page is first parsed into a list of elements based mostly on the HTML tags structure. Some elements are aggregations (tables, for instance) but the element list is not a full parse tree, which we found was not needed and in some cases actually complicates processing. Images, tables, forms and most text structure elements like paragraphs are recognized and processed according to their recognized type.

Much of the effort in building a robust HTML processor is dealing with malformed HTML expressions such as unclosed tag scope, overlapping tag scopes, etc. Unfortunately space does not allow for fully addressing this issue here. Commercial browsers currently handle these issues in differing ways. Briefly, the handling of HTML errors by PhoneBrowser mostly follows the style of Netscape Communicator.

Images often have ALT attribute tags that are used to derive the voice navigation commands for these items. The location of each image is announced along with any associated caption. This feature can be disabled on a site-by-site basis when the user does not want to hear about images.

Tables are first classified according to purpose, either layout or content. Most tables are actually used for page layout which can be recognized by the variety and types of data contained in the table cells. Data tables are processed by a parser according to one of a set of table model formats that PhoneBrowser recognizes. This provides primarily a simple way of reading the table contents row by row, which is often not very satisfying. Alternatively a transcoder can be used to reconstruct the table in sentential format. An example of this is a PhoneBrowser stock quote service where the transcoder extracts data from the Web site table and builds a new Web page containing sentences describing the company name, ticker symbol, last trade price, change, percent change and volume rather than simply reading the numbers. The user can also ask for related news reports.

Forms with pop-up menus and radio buttons are handled by creating voice command grammars from the choices described in the HTML. Each menu or button choice is spoken to the user who can repeat that phrase or a phrase subset to activate that choice. Open dialog forms (e.g. search engines) present a larger challenge. Since there is nothing to define a grammar, the implication is a full language model. While large vocabulary dictation speech systems are available, most require speaker training to achieve sufficiently high accuracy for most applications. PhoneBrowser is intended to be immediately usable without training so dictation is not yet supported. This also implies that creating arbitrary text for messaging is also not yet supported.

One additional type of form input is an extension to HTML. A GSL (Grammar Specification Language) or JSGF (Java Speech Grammar Format) specification can be inserted into an HTML anchor using an attribute tag (currently LSPSGSL). Using this method an application can specify an elaborate input grammar allowing many possible sentences to address the associated hyperlink and construct a GET type form response where the QUERY_STRING element is constructed by inserting the speech recognition text results. Grammar specifications written this way may represent many thousands of possible sentence inputs giving the end user great speaking flexibility.

## Concluding Remarks

Transcoding to a standard representation like VoiceXML is an alternative to direct interpretation of existing Web content. Because VoiceXML can specify multiple dialog states per Web page while HTML typically represents single states per page, transcoding between HTML and VoiceXML can be challenging. The PhoneBrowser method has the advantage of being available now whereas VoiceXML 2.0 will not be ready for general availability until late 2001.

## References

[1] http://www.w3.org/Voice/
[2] http://www.voicexml.org/
[3] http://www.w3/org/Voice/1998/Workshop/Michael-Brown.html