DEVELOPING WEB-BASED SPEECH APPLICATIONS

Charles T. Hemphill and Yeshwant K. Muthusamy

Media Technologies Laboratory
8330 LBJ Freeway, MS 8374
Texas Instruments, Dallas, Texas, USA, 75243
Tel. 972-997-6396, FAX: 972-997-5786, E-Mail: hemphill@csc.ti.com

ABSTRACT

We have developed a speech interface to the Web that allows easy access to information and an approach to intelligent user agents. The mechanisms developed apply to other multimedia applications where speech can serve as an input modality. We describe the benefits of our recognition system to speech-application developers: (1) Developers need not know about speech — in the simplest case, developers simply define HTML links. (2) Developers need not worry about word pronunciations since the system provides these. Developers may specify grammars in a simple BNF syntax and the system automatically converts these for use by the recognizer. (3) Developers with programming skills may use a Web server or the Java programming language to easily produce more sophisticated speech interfaces. (4) Developers reap the benefits of portability through general HTML browsers and languages such as Java. Java also simplifies the development of portable graphical interfaces that couple with speech input.

1. INTRODUCTION

As the popularity of the Web has skyrocketed over the past few years, so has the amount of information available. In addition, the nature of the users has shifted from scientists to the general public. At the same time, the power of workstations and PCs has increased to the point where they can support speaker independent, continuous speech recognition. Finally, increased commercialization and the addition of features such as Java have made the Web more desirable as a place to build speech applications.

We have developed a speech interface to the Web, called Speech-Aware Multimedia (SAM), that allows easy access to information. In the remainder of this paper, we describe the basic features of SAM that make it useful for navigating the Web by voice and for developing Web-based speech applications. We next describe more recent advances of SAM with respect to changes in the Web and new features in browsers. In particular, we describe our Java speech API which creates further opportunities for Web-based applications. We conclude by discussing some application examples, the advantages of our approach to speech interfaces, and a discussion of the underlying recognition technology.

2. SPEECH-AWARE MULTIMEDIA (SAM)

SAM is a speaker-independent, continuous speech, arbitrary vocabulary recognition system that has the following specific features for interacting with the Web:

- customizable **speakable commands** for simple browser control,
- **speakable bookmarks** to retrieve pages by random access using customized phrases,
- **speakable links** to select any hypertext link by simply speaking it, and
- **smart pages** for natural spoken queries specific to pages.

To support these features, SAM has the ability to incorporate new grammars and vocabularies "on the fly". The ability to handle a flexible vocabulary, coupled with the ability to dynamically modify grammars in the recognizer, gives SAM the feel of an unlimited vocabulary system. SAM currently runs under UNIX with Mosaic and under Windows 95 and NT with Netscape.

2.1. Speakable Commands

To control the browser, SAM provides spoken commands to display help pages, scroll up or down, go back or forward, display the speakable commands and bookmarks, add a page to the speakable bookmarks, and edit phrases for the speakable bookmarks. SAM has default phrases for these commands, but the user may change them, if desired, to more convenient ones.

2.2. Speakable Bookmarks

To reach frequently accessed pages, users may add pages to their speakable bookmarks. When adding a page currently displayed in the browser, SAM uses the title of the page to construct a grammar for subsequent access by voice. The initial grammar includes likely alternatives to allow, for example, either "NIST's" or "N.I.S.T's" in a page entitled "NIST's Home Page". The user may then add additional phrases to make access to the information more convenient or easier to remember. The speakable bookmarks remain active at all times giving users instant access to important information.

2.3. Speakable Links

Every time SAM encounters a page on the Web, it parses the HyperText Markup Language (HTML) to determine the links and the Uniform Resource Locators (URLs) associated with them. SAM then transforms the string of words into a grammar that allows likely alternatives as mentioned above. It checks several phonetic dictionaries for pronunciations and uses a text-to-phone mapping if these fail. We currently use a proper name dictionary, and an abbreviation/acronym dictionary, and a 250,000 entry general English dictionary. The text-to-phone mapping proves necessary in many cases, including, for example, pages that include invented words.

2.4. Smart Pages

On some occasions, the point-and-click paradigm associated with links falls short. For a more flexible voice-input paradigm, we developed a mechanism called *smart pages*. Smart pages are simply Web pages that contain a link to a grammar appropriate for a page or set of pages. Briefly, they work as follows (using standard Web conventions):

- the page author defines a page-specific grammar, embeds a link to the grammar in the page, and provides a brief English description of the language on the page for the benefit of users
- SAM encounters the smart page and incorporates the associated grammar into the current grammar set
- a user reads the language description and makes a query
- SAM sends the query words back to the page
- the page interprets the words to provide an answer.

The smart page approach offers several advantages:

- Web page authors can define their own page-specific languages and interpret the results in an application specific manner
- SAM recognizes with grammars and sends back words — it does not need to know the semantics of the application
- smart pages go beyond point-and-click interfaces, allowing users to easily fill in several options in one utterance.

3. KEEPING UP WITH THE BROWSERS

With rising popularity of the Web, we have seen the addition of many new features in Web browsers. To increase the usefulness of SAM, we have added voice support for many of these, including:

- **ALT tags** behind images to support more highly graphic pages. We treat these just like speakable links described above. For best results, the text in the ALT tag should match the text in the image. The ALT tags supported include those inside AREA tags of client-side image maps.
- **Framed pages** that include links to pages within separate frames in the browser window.
- **Java programs** referenced from Web pages. The next section describes this in more detail.

We have also added several features to the basic SAM system to allow more complete voice control of the browser:

- Fully automatic mixing of keyboard, mouse, and voice.
- Mouse control by voice for those pages which contain server-side image maps.
- Voice interruption of page retrieval in case the Web becomes too slow.
- Voice selection of buttons from pop-up dialog boxes.
- Voice "Go to sleep" and "Wake up" commands for totally hands-free operation.

Sam itself has become a downloadable program, compressing to under four megabytes. Once downloaded, it guides the user through installation and launches an automatic audio level setting program. It then attaches to an already open browser or launches a new one if needed.

4. SPEECH-ENABLED JAVA

We have extended SAM's smart page idea to the world of Java. Java, developed at Sun Microsystems, is a machine-independent object oriented language that lends itself naturally to applications on the Web. We have developed an Application Program Interface (API) that allows Java authors to speech-enable their Java applets by specifying grammars and actions appropriate for various contexts. We are currently working with Sun Microsystems and others to develop a standard speech API for Java.

Our current Java speech API uses a client-server model to allow the applet or application to talk to the speech recognizer. The recognition server handles the bulk of the processing work, placing a lighter load on the slower interpreted Java program. Also, this provides future flexibility by allowing the client program to execute on a separate, perhaps less powerful, device than the recognition server.

The Java speech API was specifically designed with ease of use in mind so that a Java programmer need only know a handful of methods in order to perform basic recognition in applets. To make an applet speech-aware through the Java speech API, a typical applet will do the following:

- open a connection to the recognizer,
- specify grammars for the recognizer (from a URL, a local file, or from the Java code directly),
- listen for results (synchronously or asynchronously),
- perform actions when it receives recognition results — these actions can include grammar switching and the addition or creation of new grammars,
- ask the recognizer to stop listening (on its behalf),
- close the connection to the recognizer.

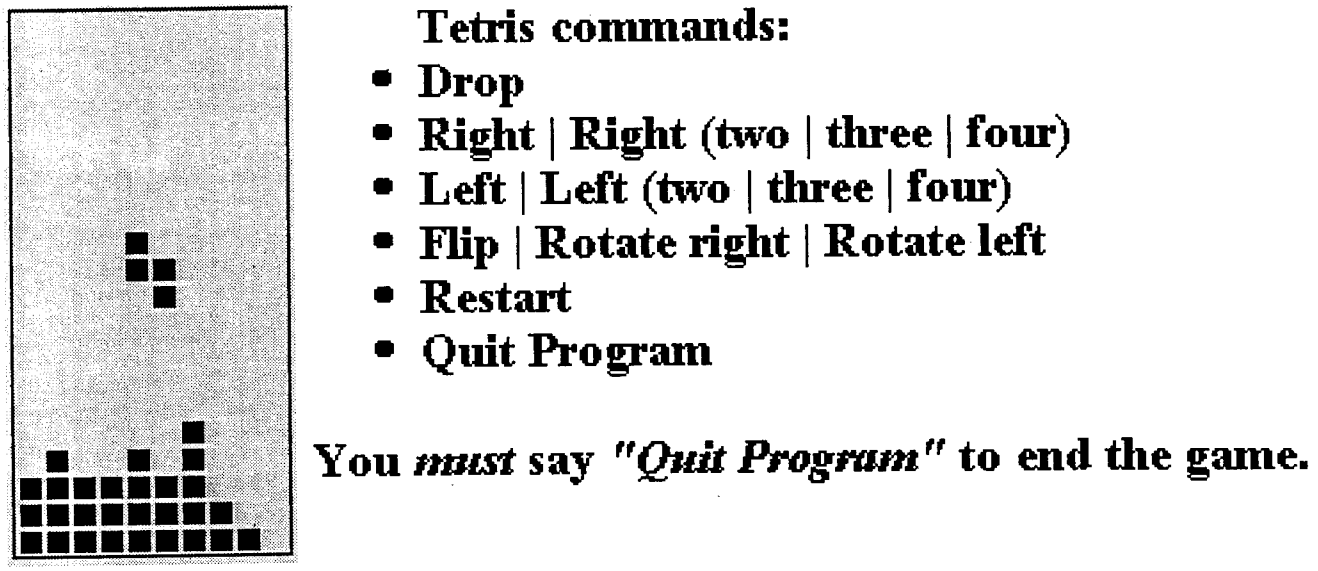

Figure 1. A Simple Speech-Aware Java Applet within a Web page.

The API supports multiple Java Applets simultaneously, sending the recognized result to the appropriate applet based on the recognized grammar. We have implemented the API internally within SAM and as a stand-alone application that can be used in conjunction with an applet viewer. When operating within SAM, an applet may request the speech focus to behave as if it were operating in a stand-alone application.

Figure 1 illustrates a shareware applet that we downloaded from the Web to test our Java Speech API. Making the original 600 line applet speech aware required only an additional 20 lines of code, plus a result handling function. These 20 lines remain the same for virtually all applets — the only differences lie in the complexity of the grammars desired and the handling of the returned results. Like our smart pages, this applet also explains the expected language to the user via a Web page. However, the Java applet has the advantage that it not only uses a dynamic grammar, but it also runs as dynamic code on the client side to more effectively and efficiently map dialog contexts to the appropriate action.

5. CREATING APPLICATIONS

SAM and the Java Speech API easily lend themselves to a number of applications beyond those of general Web surfing by voice [2] and game playing as described above. Given the SAM system, users may create meaningful speech applications by using any one of several GUI-based Web page creation tools to create pages containing hypertext links. More sophisticated Web page authors may use the standard CGI bin script mechanism to create smart pages or the Java speech API to create more interactive speech interfaces. In the remainder of this section, we discuss three representative applications that benefit from our approach: voice-powered multimedia presentations, voice access of information through kiosks, and voice interaction with multimedia repair, inspection, and training information.

A multimedia presentation that includes hypertext links allows the presenter to easily choose among several paths dynamically based on factors such as audience reaction and time available. Many tools exist to create such presentations, including standard presentation creation tools that now support the definition of links and export of the presentation as HTML. SAM can enhance a multimedia presentation by allowing the user to both talk to the audience and unobtrusively control the presentation at the same time. For example, a presenter might say "and now let's talk about '*SAM applications*' " where the audience hears the whole phrase and SAM hears the last part after an well-timed flip of a microphone switch. With a wireless microphone, presenters can wander untethered during the presentation. With speech controlled Java, we can create more effective presentations that synchronize music with animations or enter appropriate parameters to illustrate "what if" options, all by voice.

Because voice is natural and engaging to people, voice controlled information kiosks can attract and retain customers interested in product-specific information. The use of active noise cancelling microphones or microphone arrays makes this application feasible in noisy environments. Voice input also avoids the problems of missing mice or dirty touch screens. The information available in the kiosk can come from a live Web connection or a periodically updated CD-ROM. In either case, SAM can accommodate new links and information without updates of the recognition system via its speakable link feature.

Voice has always been a natural solution for hands-busy systems. With the development of wearable computers, multimedia repair manuals are now a reality. SAM fits well into this paradigm, where the user can use voice to interact with the information in a hypertext fashion or enter parameter values to give the system an intelligent agent flavor. Other applications in this genre include inspection systems and training systems. In these scenarios, the user can both access up to date information and update a central database.

6. UNDER THE HOOD

To make the SAM interface and Java speech API possible, we have developed a speech engine that dynamically incorporates new grammars and vocabulary without restarting the system. Unlike most speech engines that accommodate a single regular grammar or bigram grammar, ours incorporates a Directed Acyclic Graph (DAG) of regular grammars (RGs). This structure allows us to incorporate new grammars quickly and concisely without an expensive expansion process. Additionally, we use context-dependent phonetic models that lend themselves to vocabulary independence through the use of phonetic features [1]. These models, coupled with our large dictionary and text-to-phone rules, help free developers from efforts below the syntax level.

We have developed an API for our speech engine that supports a large number of applications, including its use with newer Web and multimedia browsers as they become available. The engine is written in ANSI C and runs on most Unix and Win32 environments. Using the engine API consists of the following basic steps:

- Open a connection to the engine.
- Ask the engine how many bytes of speech data to read for each frame (based on the models and sample rate).
- Read speech data and ask the engine to process each speech frame. The frame processing function uses energy-based endpoint detection to decide when to start recognizing and a user-supplied callback function to determine a set of grammar start symbols with which to begin recognition search.
- Close the connection to the engine.

The API also includes calls to add or replace grammars in the current set or to enable or disable existing grammars. The engine automatically derives a DAG of RGs (RGDAG) from the current set of grammars. For SAM, grammars get created from the speakable commands and bookmarks, by parsing and tokenizing links and ALT tags, by reading smart page grammars, or by incorporating Java applet grammars. Other functions are available to automatically convert Context-Free Grammars (CFGs) (without embedded recursion) to RGDAGs, minimize these grammars, segregate for sex-dependent models, and find pronunciations from the phonetic dictionary or create them from text-to-phone rules as needed.

We currently use a set of context-dependent phoneme models trained in triphone contexts over a large amount of speech data, clustered in a decision tree at the frame level according to linguistic feature questions and acoustic match, and finally reclustered at the phone level into a decision tree using linguistic questions concerning the phone and its neighbors. We use phone-level clustering so that we can load a fixed phone inventory in advance and yet accommodate phones unseen during training via the decision tree. We divide the models according to sex for increased accuracy, but this is transparent to the user. We have trained both microphone and telephone speech models in this manner. The phone models themselves are simply probabilistic RGs and become part of the RGDAG.

We have also developed a Japanese version of SAM [3] and the phonetic models required for a Spanish version [4]. To build a SAM system in a new language requires the construction of phone models trained in the appropriate language, the development of a dictionary and text-to-phone rules for pronunciations, and the development of a tokenization module for speakable links. In the Japanese version, the tokenization module proved most challenging given the variety of character sets available and the general lack of word delimiting characters.

7. CONCLUSION

We have described our latest advances in speech recognition systems that support a wide variety of speech application interfaces. Underlying all of these advances is our ability to dynamically incorporate new grammars and vocabularies "on the fly". We have applied our system to the dynamic worlds of Web pages and Java systems, creating conventions and APIs that in turn support a large number of easy to develop speech applications. We anticipate that the portability of our mechanisms and the increasing number of machines able to support recognizers will lead to many successful speech applications.

ACKNOWLEDGMENTS

We would like to thank Yu-Hung Kao who devised and trained our current phonetic model set. We also thank Jack Godfrey for providing his linguistic expertise in many phases of this effort. Tom Staples and Doug Mahlum provided excellent support for various interfaces in the PC environment. This work was partially funded by DARPA. We gratefully acknowledge DARPA's support through contract DAAA15-94-C-0009.

REFERENCES

[1] Y.H. Kao, C.T. Hemphill, B.J. Wheatley, P.K. Rajasekaran, "Toward Vocabulary Independent Telephone Speech Recognition," *Proc. of ICASSP, 1994.*

[2] C.T. Hemphill and P.R. Thrift, "Surfing the Web by Voice," *Proc. of Multimedia'95*, Nov. 5-9, San Francisco, CA.

[3] K. Kondo and C. Hemphill, "Surfin' the World Wide Web with Japanese," *Proc. of ICASSP, April 1997.*

[4] Y.K. Muthusamy and J.J. Godfrey, "Vocabulary-independent Recognition of American Spanish Phrases and Digit Strings," *Proc. Eurospeech, 1997, this proceedings.*