Challenges in Building a Mobile SpeechWeb Browser

Randy J. Fortier
School of Computer Science
University of Windsor
Windsor, Canada
rfortier@uwindsor.ca

Richard A. Frost
School of Computer Science
University of Windsor
Windsor, Canada
richard@uwindsor.ca

***Abstract*—Mobile devices support speech interfaces that are inadequate for working with web applications. Providing speech-based interaction with web applications from a mobile device opens up a world of computing to users on the go. We have developed a mobile browser that allows users to easily interact with SpeechWeb applications from their mobile device in a hands-free mode. To overcome the limitations of mobile devices, a new method of parsing speech has been developed. The result is a more accessible web, allowing users to do some of the more interactive tasks, even while traveling, that traditionally required a personal computer.**

Keywords-SpeechWeb; speech web; speech recognition; voice recognition; speech applications; web applications; mobile application; natural language processing.

I. INTRODUCTION

A. *An Introduction to SpeechWeb Applications*

Speech applications have recently been the subject of increasing interest. This is especially true for mobile devices where traditional input mechanisms are limited and output displays are inconveniently small. On mobile phones, which are often used while driving, using a keyboard or touchscreen for input and/or a display for output are both impractical and dangerous. Speech applications are common for dialing phone numbers, obtaining driving directions, sending text messages, and checking E-Mail. However, there are still many applications, such as web applications, that do not have a speech interface on mobile devices.

The web browser grew quickly during the 1990s to become the most popular Internet application. This growth was fostered by the ease of development of web sites, through standardized mechanisms. Even non-programmers could develop a web site to share their ideas. During the following decade, web sites evolved into interactive web applications, due mainly to the fact that anyone could develop those applications on any platform using any programming language. In the current decade, applications and data are migrating to the cloud, and many of these applications are web applications.

A SpeechWeb is a hyperlinked set of web pages with speech interfaces. SpeechWebs allow interrelated pages to connect. The traditional web and a SpeechWeb could be used to represent a semantic web. Speech websites are websites with speech interfaces. Speech websites could be speech interfaces to existing web pages, or specialized speech applications with a web interface. An important consideration for building a semantic web is to remove emphasis on notation and make it easier for people to develop their own sites. A SpeechWeb should facilitate the creation of sites with speech interfaces, with a broad range of complexity, in any programming language. By harnessing the same simplicity of the world wide web in the 1990s, a SpeechWeb can experience similar growth.

As an example, consider a mobile user driving to work. Current technology allows them to be entertained, make phone calls, send text messages, and navigate, all using hands-free interaction. Through SpeechWeb applications, these users could also obtain other information, including data stored in the cloud, and conduct transactions online using their voice. For example, a user could search for and hear restaurant reviews, translate French to English, or learn geographical statistics about a new state or province they have just entered (e.g., What is the population of the state of Michigan?). Google voice search implements a SpeechWeb application specifically for search, and is widely available for mobile devices [8].

The PipeBeach project [6] tried to merge the existing standards of VoiceXML and WML. At the time, WML was becoming popular as a platform for the mobile web. Since then, WML has declined in popularity, due to the increased power of mobile devices. The idea of combining a visual markup language and a speech markup language to create a multi-modal interfaces has also been used within traditional browsers [4]. The PipeBeach project has since been discontinued. Our project attempts to create a SpeechWeb browser that uses voice as its primary input and output mechanism. The w3voice project [7] is another attempt at creating a SpeechWeb infrastructure. This project uses an architecture that results in significant data transfer, and heavy processing loads on the server tier.

B. *Limitations of Speech Interfaces*

Most current speech interfaces to web applications use a screen-scraping approach. They simply read the text on a web page and provide mechanisms for the user to skip paragraphs, follow links, and fill out forms. Finding answers to a question can be very difficult. Speech interfaces to many of the existing web applications are not sufficient.

C. Limitations of Mobile Devices

The limitations of mobile devices are, for the most part, widely known. The screens are small, and current implementations can be difficult to read in bright light. Also due to the small screen, interacting with the touch screen can be error-prone. Keyboards, if available, are inconvenient, error-prone, and not ergonomically designed.

While download rates are catching up to home Internet access speeds, users still want to limit data transfer, due to very high costs on typical mobile plans. Browsing the traditional web results in a very large amount of data flow. Mobile-optimized websites often do not limit data flow, but merely use style to re-structure the information visually.

Speech interfaces also have limitations on mobile devices. Processing, memory, and disk space are limited. Voice recognition can test these limits. Even more significant, mobile devices often do not support grammar-based recognition, but only dictation. Dictation, while more flexible, is the less accurate approach, since any word in the target language could be spoken at any moment. Grammar-based recognition can improve recognition accuracy by narrowing the possible spoken words based on what is acceptable according to the grammar, and therefore relevant to the application.

D. Accessing Speech Applications with Mobile Devices

Our efforts have been focused on making SpeechWeb applications easier to create, much as web applications and websites were easy to create in the early days of the world wide web. As a result of our research into these problems, we have produced the following:

- A working prototype of a mobile SpeechWeb application browser.
- A method for improving the recognition accuracy of dictation-based voice recognizers.
- An XML-based language for describing SpeechWeb application interfaces, called SWML.
- A framework that allows programmers and non-programmers alike to create SpeechWeb applications.

E. Outline of this Paper

This paper first provides an overview of our approach. We describe an architecture suitable for a SpeechWeb. We describe an example language, SWML, that could be used for creating SpeechWeb applications. We discuss a method for overcoming the limitations of dictation-based speech recognition, common in mobile devices. Finally, we analyze our approach, and discuss related work.

II. OUR APPROACH

Our research has developed a SpeechWeb browser for mobile devices, using the Android platform. This application is designed to take advantage of the mobility and specialized hardware provided by these devices, and account for their limitations. This browser provides a client for interacting with SpeechWeb applications. SpeechWeb applications can be designed to take advantage of the speech interface, and minimize the effort required for the user to carry out their desired tasks.

As an example, consider a encyclopedic application. There are several such applications on the traditional web. By integrating a speech browser with an application that uses natural language processing, and semantic evaluation, the user can merely ask a question and immediately obtain the answer. Other application tasks, such as filling out registration forms and conducting banking transactions can also be accommodated through a question/answer approach. Simpler applications, analogous to the websites created by non-programmers in the 1990s, are also possible, provided that there are simple, yet flexible, tools to write them. Applications that are more complex are also possible.

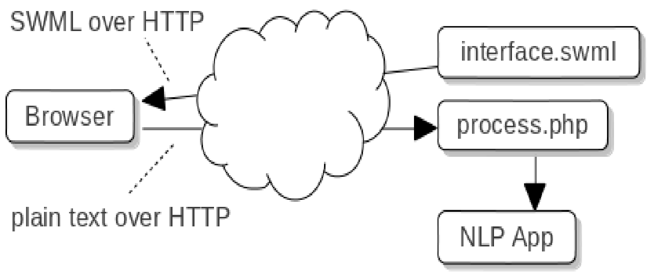

Figure 1: An overview of the SpeechWeb system

With the goal of making SpeechWeb applications easy to develop, an XML-based language for creating the speech interfaces for web applications, SpeechWeb markup language (SWML), has been created. This language allows SpeechWeb applications to be created in a similar manner to how web applications are created.

The SpeechWeb application browser also mitigates the inaccuracy typical of dictation-based voice recognition software by post-recognition filtering using a grammar. While this approach doesn't provide the accuracy of traditional grammar-based recognition, it does improve recognition accuracy significantly.

Our group has also developed a SpeechWeb application infrastructure, which should help users create their own SpeechWeb applications, even if they do not understand programming languages [1,2]. Our work into natural language processing has also produced several interesting applications, which have been integrated into the SpeechWeb.

III. DETAILS

A. SpeechWeb Architecture

Previously, it has been proposed by our research group that SpeechWeb applications use a local recognition, remote processing (LRRP) architecture for transmitting data to a speech application [1]. In this model, all voice recognition occurs on the client side. This was motivated by the fact that the raw audio data would be very large, and yet most user devices are rich clients. On mobile devices, this architecture is even more appropriate due to typical users' desire to limit data flow. Performing the voice recognition on the client-side results in less data flow, since the output of voice recognition is much more compact plain text. A comparison of these speech application architectures is given in Figure 2.

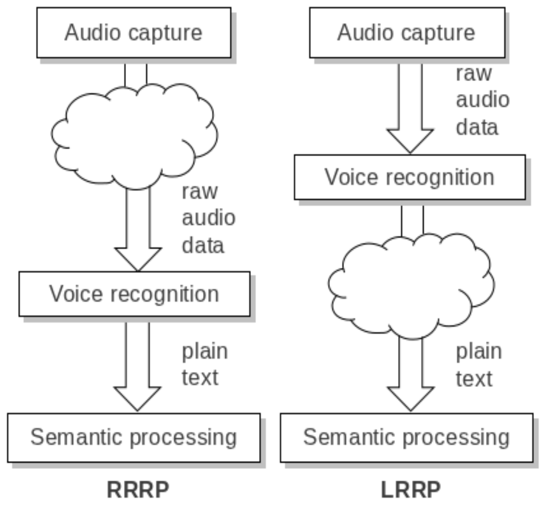

Figure 2: A comparison of RRRP and LRRP architectures

In an LRRP web application, the voice recognition occurs on the client device. This process converts the raw audio data into plain text. This plain text is transmitted to the server side of the web application, in a similar fashion to how data is sent to the server side in an web 2.0 web application via AJAX. This approach has the following significant benefits:

- Data flow is typically reduced by a far greater amount than would be possible through compression alone
- Server-side processing requirements for a server side SpeechWeb application under heavy load are significantly reduced

This approach is in contrast to that used by speech-based calling centres, which use voice recognition to allow users to navigate through a complex telephone system. These calling centres use remote recognition, remote processing (RRRP), increasing the processing requirements on the server side. For SpeechWeb applications, where data would be labeled as data by the mobile provider, the cost of transferring the large amount of audio data would also be a significant problem.

B. *SpeechWeb Markup Language (SWML)*

SWML is an XML-based language, not unlike HTML for traditional web applications. The purpose is to convey information to users, and provide means of collecting information from them. The XML syntax is familiar to many existing web developers. As it can also be easily parsed by software, tools can be developed to make development even easier for non-technical users.

For a file format for SpeechWeb applications, there are three primary requirements. First, there should be away of welcoming the user and giving them instructions on how to use the SpeechWeb application. Second, there needs to be a way to specify what are the valid expressions of the application's language, often expressed as a grammar. The main purpose of the grammar on the client is to improve the accuracy of speech recognition. Third, there needs to be a way to determine how the SpeechWeb application should respond to inputs. For example, when the application has collected sufficient data, it should send that data to the server for processing.

VoiceXML [3] is a similar file format to SWML. Both formats are largely interchangeable, but the main advantage of SWML over VoiceXML is its simplicity. A simple file format, similar to HTML in the 1990s, should make it easier for non-programmers to develop SpeechWeb applications. However, for more interactive applications, VoiceXML would be preferred. One advantage of VoiceXML over SWML is that VoiceXML documents can specify, using JavaScript, how to respond to inputs using client-side code. This capability allows AJAX-like interaction in a SpeechWeb application.

The existing format for SWML is a simple, proof-of-concept file format, which allows developers of a SpeechWeb application to specify:

- An initial prompt, which gives the user an idea of what they can do with the application
- The conditions that must be satisfied in order for data to be submitted to the server side of the SpeechWeb application
- The grammar rules, which describe the sort of sentences the user can say

Shown in Figure 3 is a simple SWML document, which could be used by a non-technical user for a simple question/answer SpeechWeb application. The grammar can simply describe the possible questions that are valid.

The prompt is used to introduce the user to the SpeechWeb application. It can be used simply to greet the user, or it could be used to provide instructions on how to use it. Existing prototype applications use relatively short prompts, in keeping with the minimized data flow policy.

The submit condition describes when the browser has collected the right amount and type of data to send to the application. In the example in Figure 4, a sentence is the unit of data transfer. Once the browser gathers a complete sentence, defined in the grammar itself, it sends that data onto the server to the URL specified. This server side of the application can answer the question itself, or act as a web portal to a non-web application. A submit condition can also include boolean expressions, which is useful for surveys where multiple questions need to be answered before the results are submitted to the server tier.

The grammar substructure describes a complete context-free grammar. Our browser uses this grammar to improve the accuracy of its voice recognition. In addition, the browser also supports local parsing and optional client-side transformation. Terminal symbols are described using a traditional rule, containing a name (category) and its definition (rhs). The definition is often a single word, but can be phrases or sentences in more trivial applications. If there is more than one terminal rule for the same category, each of the rules are treated as *or conditions*.

For non-terminals, nearly the same format is used. The category names the non-terminal, and the rhs describes a sequence of terminals and non-terminals, separated by a space. This sequence describes a single rule for that non-terminal. If there are multiple rules for the same category, each is treated as an *or condition*. An optional transform can be added, which describes how the expression should be modified before being submitted to the server tier. This eliminates the need, in some applications, where the server tier requires the expressions to be transformed in some way, such as parenthesization. This is optional, since some applications have no need for such transformations, and other applications perform the transformations on the server tier.

```
<speechweb>
 <prompt>
   Welcome to the solar system encyclopedia.
 </prompt>

 <submit-condition url="/simple.php">
   <any-of options="question" />
 </submit-condition>

 <grammar>
   <terminal category="sentence"
          rhs="what is your name" />
   <terminal category="sentence"
          rhs="what is your favourite colour" />
   <terminal category="sentence"
          rhs="what is your favourite food" />
   <terminal category="sentence"
          rhs="how old are you" />
   <non-terminal category="question"
          rhs="sentence" />
 </grammar>
</speechweb>
```

Figure 3: A Simple SWML Document

The submit-condition can be extended to accommodate less trivial conditions. For example, if the SpeechWeb application asked a question, the user could answer the question directly, or perhaps ask a question of their own. A SpeechWeb application could be designed to conduct an oral survey with the user, requiring that some or all questions have been answered. The example in Figure 4 illustrates a simple SpeechWeb application that requires the user to make statements of a litigious nature. The submit-condition can contain arbitrarily nested and and or conditions to facilitate both of these scenarios.

```
<speechweb>
 <prompt>
   Welcome to the solar system encyclopedia.
 </prompt>

 <submit-condition url="/submit.php">
   <all-of options="s1,s2" />
 </submit-condition>

 <grammar>
   <terminal category="s1" rhs="I hereby
acknowledge that the work being submitted is my
own work" />
   <terminal category="s2" rhs="I hereby deny
that I have submitted all or part of this work
for an assignment in another course" />
 </grammar>
</speechweb>
```

Figure 4: An SWML Document with a Non-trivial Submit Condition

Shown in Figure 5 is another sample SWML document. This example illustrates a context-free grammar for a very small subset of English. In this case, there are a set of terminal and non-terminal rules describing the valid sentences of the language. In this example, the grammar is a context-free grammar, used for a natural language processing application. For simpler applications, the grammar could be as simple as a list of acceptable phrases or questions.

In the example in Figure 5, the grammar has also been augmented with rules for transforming the English sentences into sentences marked up with parse information (e.g., planet → COMMON_NOUN(planet)). In this example speech application, to which we're sending the user's utterance, some words have been defined as semantic functions, so the user's phrase has been modified to have the expected syntax. In this example, the vp (verb phrase) function returns a list of matching objects (for example, a list of objects that rotate). The np (noun phrase) function filters this list according to its noun (for example, earth might limit the results to include – at most – earth).

```
<speechweb>
 ...omitted for brevity...
 <grammar>
   <terminal category="pnoun" rhs="luna" />
   <terminal category="pnoun" rhs="earth" />
   <terminal category="tverb" rhs="orbits" />
   <terminal category="iverb" rhs="rotates" />

   <non-terminal category="np" rhs="pnoun" />
   <non-terminal category="vp" rhs="iverb" />
   <non-terminal category="vp" rhs="tverb np"
         transform="^tverb ( ^np )" />
   <non-terminal category="s"  rhs="np vp"
         transform="^np ( ^vp )" />
 </grammar>
</speechweb>
```

Figure 5: An SWML Document with Transformation Rules

C. *Improving Recognition Accuracy*

Grammar-based voice recognition is more accurate than dictation-based voice recognition for one simple reason: The grammar is used to limit the possible words that can be accepted. Consider the grammar in Figures 5 and 6. If the user says "mars," it matches an np (noun phrase). If parsing an s (sentence), we next expect a vp (verb phrase). In this simple grammar, only two words are allowed next (orbits and rotates). As this is a simple example, such results should not be expected in the general case, but the reduction is still significant. With fewer choices available, the voice recognizer has a better chance of being right about what was uttered.

Unfortunately, many mobile devices do not support grammar-based voice recognition. To get past this limitation, we can apply the grammar after recognition for dictation-based voice recognizers that support the return of a list of possible phrases (often including probabilities for each phrase). Any phrases that do not follow the syntax established by the grammar are eliminated as options, and the remaining phrase with the highest probability is chosen. This process is illustrated in Figure 7.

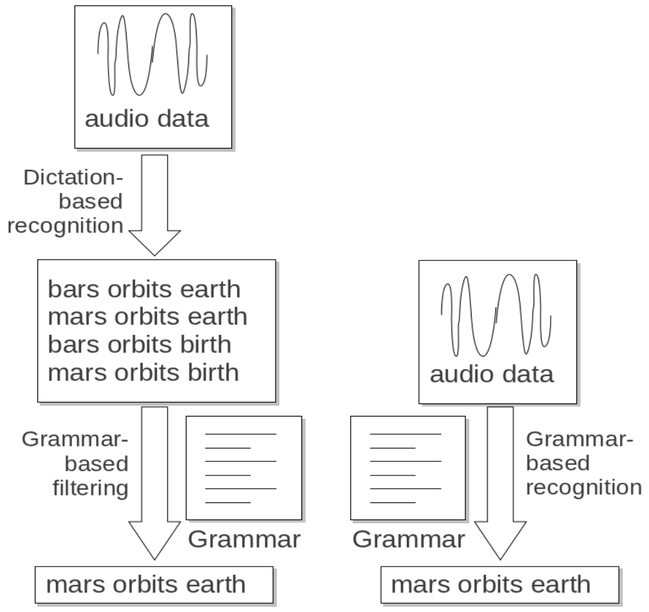

Figure 7: A Comparison of Data Flows in Grammar-based Recognition and Grammar-based Filtering

Dictation-based voice recognizers generally allow the user to utter any sequence of English words, grammatical or not. This flexibility is useful when writing E-Mails or word processing documents, but results in frequent mistakes. When a sequence of words is uttered, it does not limit the words that follow; although it may affect the probability assigned to the results in some context-capable recognizers.

The main difference between grammar-based recognition and applying grammar-based filtering to dictation-based recognition results is the granularity, to which the grammar is applied. In grammar-based recognition, each word is subjected to the constraints of the grammar before it is accepted. In grammar-based filtering, a number of phrases have already been recognized and any phrases with non-grammatical words or sequences of words are eliminated.

D. Benefits to Our Approach

The mobile SpeechWeb application browser gives mobile users the ability to use speech interaction for applications that currently do not support speech input. This browser differs from a typical speech browser, such as those used by the visually impaired, in that it is not intended to simply dictate the text in a traditional web application but provide a browser for a whole new web built upon speech interfaces.

In the cloud computing era, desktop applications and their data are migrating onto the web. Speech applications provide a convenient way to access data. Mobile applications allow people to access this data while on the go. Combining the convenience of a speech interface with the mobile devices is a natural progression. There are examples of speech applications on mobile devices. However, speech applications that provide access to locally-stored data, obtained through traditional approaches (e.g., using IMAP to download E-Mail messages), are not sufficient for most users needs. These users are increasingly using web applications in their daily life. These applications compute and store data on the web. When web applications provide access to data in the cloud, a speech interface to those web applications allows users to access to that data in a non-traditional way. The advantage is that speech interfaces allow mobile device users to access their data in a hands-free manner. For example, users could access cloud data while driving, if those applications had a speech interface.

The browser limits data flow in two significant ways. First, it uses the LRRP architecture that sends only the post-recognition plaintext across the network. Second, the SpeechWeb applications are specially designed with efficient access to narrow information in mind. Rather than download a complete web page with pages of text, and using a cumbersome speech interface to navigate through that text, the speech applications should be designed to give exactly the information the user needs. For example, a encyclopedic SpeechWeb application might answer queries about the data they contain, unlike web-based encyclopedias that return all known data on a specified topic. This strategy is more appropriate for mobile users, since – as their name implies – they are on the move and may not be able to navigate through data as easily as a user at a desktop application or traditional web application.

IV. ANALYSIS

The post-recognition parsing takes a list of possible spoken phrases, in the order of their probability and removes any phrases that do not match the grammar. The order of probability is retained. Assuming the grammar is correct, any phrase not matched by the grammar is not correct. Therefore, we would expect the recognition accuracy to be as good or better than dictation-based recognition.

To demonstrate typical improvement, a user study was conducted. In this study, users are shown 28 valid sentences, containing transitive verbs and noun phrases, from a sample English-based grammar and are asked to speak these phrases aloud. The dictation-based recognition proceeds to generate up to 25 possible spoken phrases based on the user's speech input. The position of the correct phrase in this list, if any, is recorded. This same position is determined again, after the list of possible spoken phrases is filtered by post-recognition parsing. These two statistics are used, along with the percentage of first-rank results, to compare the accuracy of each approach. The results of this study have been included in Figure 8. In this table, the results for post-recognition parsing are found under the name grammar filtering.

Method	Average Rank	Percentage of First-Rank
Dictation-based	8.16	19.64%
Grammar filtering	4.96	40.54%

Figure 8: A Comparison of the Accuracy of Grammar-based Recognition and Grammar-based Filtering

As shown in Figure 8, the recognition accuracy in the user study was improved by 20.9%. Recognition in this case has very poor accuracy, as is typical for dictation-based recognition when recognizing complete sentences. As expected, the average rank of the correct sentence in the list of possible utterances has improved, since non-grammatical sentences in this list are removed in the process.

V. Related Work

A. Related Projects

The simple XML-based document format, SWML, was created for simplicity; to facilitate non-technical users creating SpeechWeb applications. VoiceXML [3], X+V [4], and Salt [5] are other file formats that contain similar data. However, these formats are complex, which can be a deterrent for non-technical users. Users can begin creating SpeechWeb applications quickly, and for simple question/answer applications the SWML document can be auto-generated. For more interactive SpeechWeb applications, such as those requiring scripted behaviour, the flexibility of VoiceXML would outweigh its complexity. None of these speech application formats have widespread support on mobile platforms.

The discontinued PipeBeach project [6] provided a speech interface to the traditional web for mobile devices. This project builds upon the VoiceXML standard, and as such is dependent upon VoiceXML application support. At the time, mobile devices with speech recognition capability were not yet widely available, and thus no VoiceXML browsers could be executed on the devices. One of the project's goals was to combine the WML and VoiceXML standards. One approach is server-side translation of WML into VoiceXML. The popularity of WML was limited, due to rapid changes in mobile device capabilities.

The w3voice project [7] is a Japanese-language SpeechWeb initiative. The project uses an RRRP architecture, sending raw audio data to the server side for processing by a third-party speech recognizer. RRRP architectures require large data transfers, since the raw audio data must be transferred to the server-tier for recognition. RRRP architectures also require significant resources on the server tier for recognition, since all clients' audio must be recognized on the same site. LRRP SpeechWeb architectures, on the other hand, send only plain text to the server tier, and use a decentralized recognition model.

Our research group has also developed a desktop SpeechWeb browser, based on X+V [9]. Support for X+V is not provided by any known open source applications for mobile devices, and thus a simple port of this browser was insufficient. The mobile SpeechWeb browser uses a custom file format, and post-recognition parsing to improve accuracy.

VI. Conclusions and Future Work

Our group has created a speech browser for mobile devices, that improves recognition where the device uses dictation-based speech recognition. The browser performs voice recognition on the device itself, to reduce data transfer requirements and server processing requirements.

This SpeechWeb application browser has been integrated into the larger SpeechWeb project, the goal of which is to create useful speech-based web applications and encourage others to do the same. We have developed speech applications for query encyclopedic databases, such as information about the solar system, as well as simple applications, such as one that tells jokes. Applications that are used to conduct speech-based surveys are being developed to service people in areas where mobile phones are common, but traditional computing devices are rare.

The SpeechWeb application infrastructure is being used as a test platform for natural language syntax and semantic processing research, allowing additional semantic evaluation constructs to be demonstrated, and providing useful SpeechWeb applications to users.

References

[1] R. A. Frost, “A Call for a Public-Domain SpeechWeb,” *Communications of the ACM*, vol. 48, iss. 11, pp. 45-49, November 2005.

[2] R. A. Frost, A. Karaki, D. Dufour, J. Greig, R. Hafiz, Y.Shi, S. Daichendt, S. Chandon, J. Barolak, and R. Fortier, “MySpeechWeb: Software to Facilitate the Construction and Deployment of Speech Applications on the Web,” *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility*, pp. 249-250, October 2008.

[3] http://www.w3.org/TR/voicexml21/ [retrieved: October 17, 2011]

[4] http://www.w3.org/TR/xhtml+voice/ [retrieved: October 17, 2011]

[5] http://msdn.microsoft.com/en-us/library/ms994629.aspx [retrieved: October 17, 2011]

[6] http://www.w3.org/2000/09/Papers/Pipebeach.html [retrieved: October 17, 2011]

[7] http://w3voice.jp [retrieved: September 1, 2011]

[8] http://www.google.com/mobile/voice-search/ [retrieved: October 17, 2011]

[9] Richard A. Frost, Xiaoli Ma, Yue Shi, “A browser for a public-domain SpeechWeb.” *Proceedings of the Sixteenth International World Wide Web Conference*, pp. 1307-1308, May 2007