A Novel Use of VXML to Construct a Speech Browser for a Public-Domain SpeechWeb

L. Su and R.A. Frost[1]

School of Computer Science, University of Windsor,
Windsor, Ontario, Canada N9B 3P4
{su5, Richard}@uwindsor.ca

Abstract. Despite the fact that interpreters for the voice-application markup language VXML have been available for around five years, there is very little evidence of the emergence of a public domain SpeechWeb. This is in contrast to the huge growth of the conventional web only a few years after the introduction of HTML. One reason for this is that architectures for distributed speech applications are not conducive to public involvement in the creation and deployment of speech applications. In earlier work, a new architecture for a public domain SpeechWeb has been proposed. In this paper, it is shown how a speech browser for this new architecture can be readily built through a novel use of VXML. A detailed description of the browser is given, together with a discussion of the advantages of this novel approach.

1 Introduction

A SpeechWeb is a collection of hyperlinked applications that are distributed over the Internet and which are accessible by spoken commands and queries that are input through remote end-user devices. Various architectures and technologies have been developed which are contributing to the development of SpeechWebs:

1. Speech interfaces to conventional web pages. These interfaces run on end-user devices and allow users to scan downloaded web pages and follow hyperlinks through spoken commands [e.g. Hemphill and Thrift 1995]. More sophisticated versions process the downloaded web pages and provide spoken summaries and allow some limited form of content querying.
2. The second architecture involves the use of networks of hyperlinked VXML Pages. VXML [Lucas 2000] is similar to HTML except that it is used to create hyperlinked speech applications. VXML pages, which are executed on VXML browsers, include commands for prompting user speech input, for invoking recognition grammars, for outputting synthesized voice, for iteration through blocks of code, for calling local Java scripts, and for hyperlinking to other remote VXML pages that are downloaded and executed in a manner similar to the linking of HTML pages in the conventional web. Speech recognition is carried out by the

[1] Both authors contributed equally to this paper.

B. Kégl and G. Lapalme (Eds.): AI 2005, LNAI 3501, pp. 401 – 405, 2005.

VXML browser running locally on an end-user device, or at a remote site which is accessed through the telephone.

3. The third architecture is the one used in call centers. End-users communicate with the call center using remote telephones. Speech recognition is carried out at the call center. Often VXML is used to code the call center application.

A comprehensive study of these architectures, and variations of them, can be found in EURESCOM [2000].

A public-domain SpeechWeb is one in which speech browsers run on end-user devices, and end users create speech applications and deploy them on their own web servers. It has been observed [Frost 2004] that the public-domain SpeechWeb is growing at a very slow pace despite the fact that VXML has been around for five years, and that the reason for this is that the three architectures above are not conducive to the involvement of a wide range of end-users. The first suffers from the fact that most conventional web pages are not designed for non-visual browsing, the second architecture requires applications to be written in VXML and also suffers from the fact that these applications have to execute on the end-user device which is not appropriate when large databases or sophisticated natural-language processing is involved. The third architecture requires expensive software to link between telephone access and the call-center application, and also requires end-user voice profiles to be stored at all call centers if high recognition-accuracy is needed.

A new architecture which is conducive to the development of a public-domain SpeechWeb has been proposed in Frost et al [2004]. That architecture is based on Local Recognition and Remote Processing (LRRP). The speech browsers run on local end-user devices, and the hyperlinked applications execute on remote servers. The LRRP architecture is described briefly in section 2 of this paper. The architecture assumes that the end-user speech browsers will be easy to create, easy to deploy, and will make use of the most-recent advances in grammar-based speech-recognition technology. In section 3 of this paper, we show how this can be done. Our approach involves a novel use of VXML. Section 4 contains the URL of a video demonstration of the SpeechWeb browser, and a discussion of current work to extend the capability of the browser to accommodate more advanced speech applications.

2 An Architecture for a Public-Domain SpeechWeb

The LRRP architecture is depicted in Figure 1. The architecture is simple and it is shown later how it can be implemented using readily-available software and commonly-used communications protocols. In the LRRP architecture, speech applications reside on conventional web servers. Each application consists of a recognition grammar and an interpreter. The grammar defines the application's input language. When a speech browser first contacts a remote application, the grammar is downloaded and used to tailor the browser for that application. This is necessary to achieve sufficient recognition-accuracy for non-trivial applications. It has been noted by Knight et al [2001] that grammar-based speech recognition is now the predominant technology used in commercial speech products.

When a user input is recognized by the local browser, it is sent as text to the remote application. The interpreter at the remote web site accepts the input, processes it, and

returns the result as text to the browser on the end-user device. That result is then output as synthesized voice. If the user input is a request to follow a hyperlink to another application, then the interpreter return the URL of the new application as result. In this case the browser recognizes the result as such and contacts the new application for a new recognition grammar, and transfers all future input to the new application until the user requests transfer to another application. The question addressed in this paper is how to implement speech browsers for the LRRP architecture.

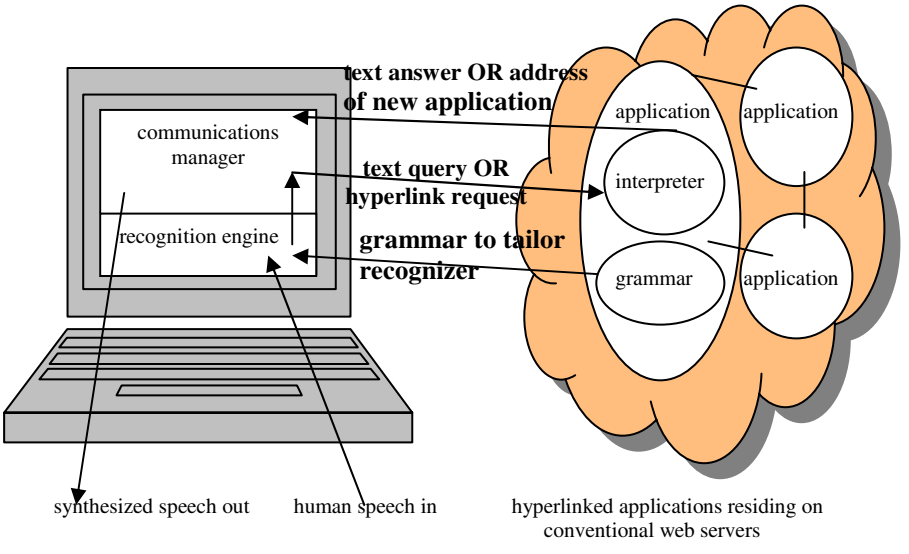

Fig. 1. The LRRP SpeechWeb Architecture

3 A Browser for a Public-Domain SpeechWeb

One approach that can be used to create an LRRP SpeechWeb browser is to use a commercial speech-recognition engine, such as IBM's ViaVoice technology, and to embed it in code which interfaces with the user and the Internet. Our initial browser, which was constructed in this way using JAVA, was satisfactory operationally but had several shortcomings as a vehicle for encouraging involvement in the development of a public-domain SpeechWeb: the code was complicated and consisted of several hundred lines associated with the end-user interface, low-level integration of the recognition engine API, communication with the remote applications, and all of the associated exception-handling. Secondly, the browser could not be deployed in its entirety as it used IBM's proprietary APIs.

Our solution to these shortcomings was to construct the speech browser as a single VXML page. The page can be executed by any VXML interpreter installed on an end-user device. Such interpreters are available from several venders and some can be downloaded for free in their beta versions. Our browser page begins by displaying a

window which can be used to set a default start-up application and/or to direct the browser to a particular starting application for that session only. After that, the browser contacts the application, downloads the recognition grammar and waits for end-user input. When input is recognized it is sent to the application and the result, when received, is output in synthesized voice. The VXML code then loops back to process more user input. When a request to be transferred to another application is sent by the user, and the new URL received by the browser, it accesses the new grammar and begins over again. The only difficulty is that the current version of VXML does not provide a mechanism for a grammar identifier to have its value changed when looping through VXML code. Therefore it was necessary to include a Java object to deal with transfer to a hyperlinked application. The object intercepts the result returned from the remote application. If the result is a new URL, then the object makes a copy of the whole VXML page with a new grammar location, and replaces the original page with the new one. (The original page is maintained in a cache so that the user can "go back" if required). Fig. 2 shows the structure of the new browser.

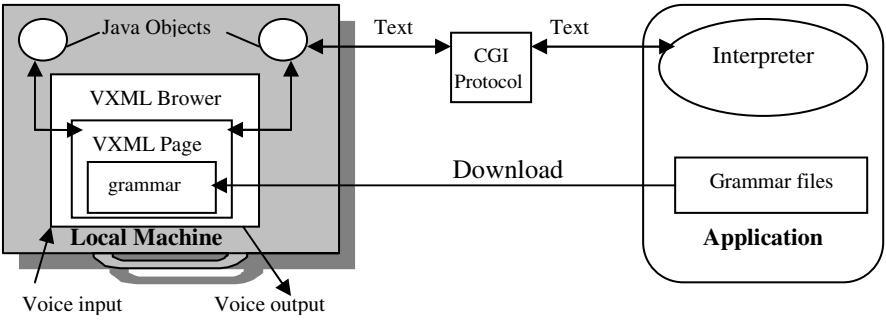

Fig. 2. A novel use of VXML to build a SpeechWeb browser

The following is an example session with a small SpeechWeb:

Application: Hi, I am solarman. I know about the planets and their moons.
User: Which moons orbit mars?
Application: Phobos and Deimos.
….
User: Can I talk to Monty?

The browser follows the hyperlink returned by Solarman to the new application Monty.

New Application: Hi, I am Monty. What can I do for you?

User: Tell me a joke.
….

This use of VXML is distinct from the intended use of VXML. Rather than create a SpeechWeb as a set of hyperlinked applications written as VXML pages, which are downloaded and executed on the end-user device when accessed, we have a single

VXML page which acts only as the speech interface to the remote hyperlinked applications which execute on the remote web servers. We claim several advantages for this approach. The browser consists of a few lines of easily-maintainable VXML code, together with two small Java objects. VXML handles exception handling, etc. at a high level. The full power of the available VXML interpreters can be taken advantage of. Our browser will benefit from all future improvements to VXML interpreters. It will be possible to tailor the browser to end-users when VXML interpreters become available that make use of voice profiles. Applications can be written in any language supported on the remote servers. All communication between the browser and the remote applications is through text. The applications can be deployed on regular web servers using any protocol (such as CGI) which supports http get and post operations.

4 Conclusion

The SpeechWeb browser has been successfully tested, and a demonstration of it can be viewed at the following, using QuickTime Player: http://davinci.newcs.uwindsor.ca/~speechweb/movie.mov

The browser has one significant shortcoming. It does not support dialogue. The CGI protocol which it uses accepts text from the browser, passes it through the standard input to the application on the web server, causes that application to execute, returns the standard output from the application as text to the browser, and then closes the application's execution. We are currently investigating various techniques to overcome this "single shot" limitation in order to support dialogue that is necessary for more complex and useful applications as discussed in [McTear 2002].

References

1. EURESCOM – The European Institute for Research and Strategic Studies in Telecommunications. Report EDIN 0010-0923 of Project P923-PF, MultiLingual WEB sites: Best practice, guidelines and architectures. (2000).
2. Frost, R. A., N. Abdullah, K. Bhatia, S. Chitte, F Hanna, M. Roy, Y. Shi and L. Su, LRRP SpeechWebs, IEEE CNSR Conference (2004) 91-98.
3. Frost, R. A. A Call for a Public-Domain SpeechWeb. Accepted in July 2004 for publication in the CACM.
4. Hemphill, C.T. and Thrift, P. R. Surfing the Web by Voice. Proceedings of the third ACM International Multimedia Conference (San Francisco 1995) 215 – 222
5. Knight, S. et al. Comparing Grammar-Based and Robust Approaches to Speech Understanding: A Case Study. In Eurospeech 2001, the 7th European Conference of Speech Communication and Technology (Aalborg Denmark 2001).
6. Lucas, B. VoiceXML for Web-based Distributed Conversational Applications. Communications of the ACM 43 (9) (2000) 53-57.
7. McTear, M. F. Spoken dialogue technology: enabling the conversational user interface. ACM Computing Surveys 34 (2002) 90-169.