Speechnet: A Network of Hyperlinked Speech-Accessible Objects

R. A. Frost, Department of Computer Science, University of Windsor, ONT N9B 3P4 Canada

Abstract

The dominant technology for electronic communication and commerce, the telephone, does not discriminate between sighted and non-sighted users. The web does discriminate.

The web is primarily based on text and visual scanning. Much of its functionality is inaccessible to visually-challenged users, or in applications where visual scanning is inappropriate.

A solution is to augment the web with a network of speech-accessible hyperlinked objects. Each of these objects has an associated grammar which defines the language that the object can respond to. These grammars are downloaded to "speech" browsers which use them to configure their speech-recognizers. This results in highly accurate user-independent continuous-speech interfaces to remote knowledge and functions.

A prototype Speechnet has been constructed using IBM's Via Voice speech technology and a common Internet communication protocol.

1 Introduction

Speechnet is a network of speech-accessible hyperlinked objects that reside on servers on the Internet. These objects, called sihlos, are accessible through remote speech browsers which provide speech-recognition and speech-synthesis functionality. Sihlos contain a grammar and a set of voice properties. When a browser accesses a sihlo, it begins by downloading the grammar and voice properties. It uses the grammar to configure the local speech-recognizer. It also uses the voice properties to configure the local speech synthesizer. The browser is now ready to accept user-independent continuous speech from the user, recognizes the input, coverts it to a character stream, sends the character stream to the remote sihlo, which interprets the input and returns an "answer", which is converted to synthesized speech by the browser.

Speech-recognition accuracy can be very high owing to the fact that the speech recognizer is configured by the grammar to recognize only those expressions which can be processed by the particular sihlo currently "in scope".

Sihlos also contain speech activated hyperlinks to other sihlos. For example, if the user asks "Can I have some information on bicycles", the sihlo might respond with "Yes, I shall get the sporting-goods agent for you", and then send the browser the URL of the sporting-goods sihlo. The browser downloads the new grammar and voice and waits for user speech input which is then directed to the sporting-goods sihlo.

This paper is structured as follows:

1. Introduction
2. An example dialogue with Speechnet
3. A description of the current implementation
4. Potential uses of Speechnet
5. Constructing sihlos
6. Building sihlos as executable grammar objects
7. Designing sihlo grammars
8. Deploying sihlos over the Internet
9. Related work
10. Concluding Comments

2 A sample "dialogue" with Speechnet

The following is a transcript of a "dialogue" with Speechnet. No "voice training" was required and the input was spoken in natural continuous speech. The computer's response is indicated by the prefix `c .`

```
y: Hello.
c: Hello, you have reached ABC Retail
     Incorporated. How can I help you.
y: Do you sell bicycles
c: Yes, I shall get the sporting-goods
   agent for you.
The response from the sihlo now
contains information causing the
browser to switch to the sporting goods
sihlo, which responds with a different
voice
c: Hello, I am the sporting goods
  agent. How can I help you.
y: Do you sell mountain bikes?
c: Yes we do.
The browser switches to the mountain
```

bike sihlo. However, the user is not informed of the switch this time. The switch is made to improve speech-recognition accuracy and to modularize the information "stored" in the sihlo.

```
c: What kind of mountain bike are
you interested in.
y: A mongoose
c: We stock three models of mongoose
etc.
```

The sihlos illustrated in the above are relatively simple in that they do little more than pattern matching on the input strings. More sophisticated sihlos can be built which interpret the input. Such sihlos can be used, for example, to provide pseudo natural-language querying of complex databases. The following is an example dialogue with a sihlo that can answer questions about a limited part of the solar system.

```
y: Hello.
c: Hello, I am solar-man, I know
   about the planets, moons, and
   the people who discovered them.
y: How many moons orbit mars
c: two
y: which planets orbit mars
c: phobos and deimos
y: who discovered phobos
c: Hall
y: did Hall discover every moon
c: no
c: which planet is orbited by phobos
c: mars, you should know that
```

This sihlo is also accessible through the web using a regular html browser. Readers who would like to experiment with this sihlo can access it at:

```
http://www.cs.uwindsor.ca/users/r
      /richard/miranda/wage_demo.html
```

3 The prototype implementation

The prototype implementation of Speechnet has the following features:

1. The speech-browser is written in Java, uses IBM's Via Voice speech engine and IBM's implementation of the Java speech APIs. The speech browser can run on any PC which supports Java.
2. In the prototype, sihlos are stored in cgi-bin directories on compute servers. Sihlos are accessed over the Internet through the cgi-bin protocol.
3. Each sihlo is associated with a grammar file which is also stored in the cgi-bin directory.
4. Existing sihlos have been constructed in various ways and in various programming languages. According to the cgi-bin protocol, each sihlo is written so that it accepts user input as a string of characters, and responds on the standard output with a string of characters. The cgi-bin post method is used by the browser to send input and receive the response from a sihlo.
5. Sihlos can be built so that the responses for particular user inputs can include directives to the browser containing the URL and name of another sihlo which the browser then contacts. The speech browser also contains a "go back" function enabling the user to go back to the last sihlo contacted. The is analogous to the back button on an html browser.

As discussed later use of the cgi-bin protocol limits the capabilities of sihlos and an alternative mode of deployment of sihlos over the net is being investigated.

The prototype operates in real time with no noticeable pause when grammar files are downloaded from remote sites. We have not yet tested the system with a huge grammar file such as that which would be required for dictation. It is most probable that the downloading and reconfiguration of the speech-recognizer for such grammars would cause considerable delay in response.

The speech interface requires no training and accepts continuous (natural) speech. The accuracy of speech recognition depends on the 'size' of the input language as discussed later.

4 Potential uses of Speechnet

The prototype implementation can be used to build various applications including the following:

1. Natural-language speech interfaces to databases as illustrated by `solar man`.
2. Collections of speech-accessible knowledge that can be searched by speech commands.
3. Retail information applications such as that given in the example, which guide potential customers to product information through speech dialogue.
4. Speech-accessible calculators which can compute the answer to mathematical questions.
5. Systems which provide up-to-date information accessible through speech requests such as "What is the exchange rate of the Canadian Dollar?"

6. Translators which can translate an utterance into another language. This application requires a sophisticated approach involving careful construction of grammars, and clues to help the speech recognizer correctly recognize the input.

A more sophisticated implementation, for example one based on CORBA as discussed later, would greatly expand the potential applications of Speechnet to include:

1. Applications involving dialogue, where the user responds with speech to requests for information from the sihlo. For example, in completing a tax return from spoken information.
2. Games involving dialogue.
3. Remote speech access to computer–controlled devices. For example, to gather data from remote sensors or to control equipment in hostile environments. Object-oriented programs on remote servers, which are linked to external devices, could be accessed by sihlos.
4. Speech access to applications which perform other tasks in addition to speech responses. For example, it would be possible to build a sihlo which activates a printer in response to commands such as “please run off six copies of the 1999 OCGS report, by noon today”.
5. Speech-driven browsers of conventional html web pages. The “web-page” sihlo could obtain the required web page, process it, and create a grammar for the recognizer on the fly. This grammar would be tailored to the html page in scope and would allow commands such as “go to link ’University of Windsor’” to be processed with good recognition accuracy. IBM has already developed a speech-driven web browser that works like this. The advantage of using Speechnet in conjunction with such products is that remote supercomputers can be used to perform highly-sophisticated analysis of the web page in reasonable time, enabling complete and appropriate answers to spoken questions from remote users. Questions such as “Does this page contain an analysis of the relationship between the consumption of coffee and the productivity of object-oriented programmers?”

5 Constructing sihlos

Simple sihlos can be built by listing all possible inputs and giving the corresponding output. For example, the following program code could be part of a sihlo that responds to simple greetings.

```
answer "hello" = "Hi, how are you"
answer "hi"    = "Hello, how are you"
answer "hello ABC" = "Hello, what
                     can I do for you"
answer "Hi, Sue"
 = "LINK=Hi, I am not sue
        but will get her for you;
    SIHLOURL=www.cs.uwindsor.ca;
    CGI-BIN=cgi-bin/richard;
    SILHO=sue.so;"
etc..
```

The link is used by the browser to switch to the new sihlo `sue.so` at the URl given.

At the same time as constructing the language-processing part of a sihlo, the developer also has to construct a grammar used by the speech recognizer. A grammar suitable for the example above might be as follows:

```
s ::= greeting
    | greeting name
greeting ::= Hi | Hello | Good-morning
name ::= ABC | Sue | Solar-man
```

The developer can also determine which type of voice should be used by the speech-synthesizer when the sihlo responds.

The language processor, grammar and voice properties are packaged together as a sihlo and placed in an appropriate location on a server that is accessible through the Internet. In our prototype implementation, sihlos reside in cgi-bin directories.

Developing the structure of a large collection of sihlos, and deciding on how they should be hyperlinked is analogous to the design of a large html web site. However, no guidelines have yet been developed to facilitate this process. The work of Arons [1] may be relevant to this subject.

In many cases, the construction of sihlos is considerably more complicated than described above. In the following, we discuss techniques that can be used to construct complex sihlos, design input languages and grammars which result in high speech-recognition accuracy, and deploy sihlos more effectively over the Internet.

6 Building sihlos as executable specifications of the input languages.

All sihlos are language processors in some form or other. As discussed above, the construction of simple sihlos is straightforward. The sihlo can use pattern matching on the input string to determine the response. More complex sihlos, such as solar-man, require more sophisticated language-processing techniques to be used.

Parser generators such as YACC [10] or Cup [4] can be used to construct the language-processing part of a sihlos automatically by "compiling" the grammar defining the language to be processed. Parser generators are very useful tools for many applications. However their use in the construction of sihlos is limited for the following reasons:

1. They requires a fairly high level of programming skill.
2. The resulting language processors are not modular and their interface with the rest of the program is non-trivial.
3. They usually do not accommodate ambiguous languages and therefore cannot be used to build natural-language interfaces.

An alternative approach is to construct the sihlo as an executable specification of the input language. Not only does this facilitate the construction of the language processor, it has also been shown that this facilitates the concurrent design of the processor and the grammar for the speech recognizer [7].

Executable specifications of language processors have been used in the logic [17] and functional-programming [2 and 8] communities for many years. The basic idea is that constructs are added to the programming language which enable a language-processing program to be built so that its structure has a one to one correspondence with the grammar of the language to be processed. For example, consider the grammar G given in section 3, which defines a language that includes `"Hello"`, `, "Hi Sue'`, `"Good-morning ABC"`, `etc`

```
s ::= greeting
    | greeting name
greeting ::= Hi | Hello | Good-morning
name ::= ABC | Sue | Solar-man
```

Using the parser-combinators (higher-order functions) `$then`, `$orelse`, and `term`, from the functional-programming paradigm, one can construct a recognizer for the language defined by this grammar as follows:

```
s =  greeting
     $orelse
       (greeting $then name)
greeting = hi $orelse hello
             $orelse good_morning
name = abc $orelse sue
           $orelse solar-man
hi = term "Hi"
hello = term "Hello"
etc.
```

The correspondence between the program and the grammar is one-to-one.

The `solar-man` sihlo is constructed entirely as an 800–line executable specification of an attribute grammar. It can answer tens of thousands of questions. An attribute grammar consists of semantic rules as well as syntactic rules. Making an attribute grammar executable allows one to specify and implement the processes by which results from compound expressions are computed from their parts. A complete listing of the `solar-man` code is accessible through the web page given earlier.

It has been shown [5] that the functional parser-combinator method can be adapted for use in object-oriented software development through the definition of new constructs called *Executable Grammar Objects.* The basic idea is that a class of language processors is defined and extended to the subclasses `term`, `then` and `orelse`. Instances of `term` are created through a constructor which takes a token such as "Hi" as input. Instances of `orelse` are constructed and their "alternatives" set by passing alternative processors to a `setAlternatives` method. Instances of `then` are created in a similar manner. Although the approach requires the introduction of more syntax to the grammar in order to convert it to an object-oriented program, the process is nearly clerical.

The main advantage of the use of executable grammar objects is that complex sihlos can be constructed as highly modular executable specifications which are easy to construct, modify, reuse and deploy in a distributed- computing environment using the most advanced Internet communication protocols. A more comprehensive account is given in [7]

7 Designing sihlo grammars

Designing powerful and accurate speech interfaces is a difficult task. Increasing the functionality of an interface often involves increasing the expressiveness of the input language, and this often results in a decrease in speech-recognition accuracy. Speechnet overcomes this problem

to some extent by allowing the functionality, and the input language, to be divided into modules (the sihlos). However, in some cases it is not possible to subdivide the functionality of a sihlo, and the resulting input language may need to be carefully designed in order to achieve acceptable recognition accuracy. The following are some techniques that can be used to modify input languages:

1. The vocabulary can be restricted. It has been shown that humans can readily adapt to relatively severe restrictions [12].
2. Replace problem words with equivalent words that are more easy to distinguish from other candidate words.
3. Replace a phrase by a word. For example, the phrase `person who discovered something` could be replaced by the word `discoverer`. In some cases, accuracy may be improved if the single word is used.
4. Replace a word or a phrase by an equivalent phrase. For example, the word `who` could be replaced by the phrase `which person`. This might overcome recognition problems when phonetically 'low-key' words such as `who` are used.
5. Restrict the input language so that only those utterances that are semantically (as well as syntactically) correct are considered as candidates. For example, `discovered by Hall` would be considered as a candidate but `discovered by mars` would not. In syntax-directed speech recognizers this can be achieved by coding semantic constraints as syntactic rules in the grammar which is used to direct the recognition process. Young et. al [19] have shown how this approach can be used to build robust interfaces by having layers of grammars. If an utterance is failed to be recognized at one level of constraints, the constraints are relaxed and a more general grammar is used. Use of grammars augmented with semantic constraints is also discussed in [18].
6. Modify the input language so that the recognition search space at problematic points in the utterance is reduced. For example, recognition accuracy can be improved for syntax-directed speech recognizers if proper names are preceded by qualifying phrases which limit the search space at the problem point. For example, by replacing `hall` by `the person called hall` the search space is reduced from the total number of proper-names in the system to the number of names of people.

There are trade-offs which have to be investigated when such techniques are used: The modified language will often be less natural to use, recognition time may increase, or significant changes may have to be made to the interpreter that uses the output from the speech recognizer. A case study of applying these techniques is given in [7].

8 Deploying sihlos over the Internet

The current implementation of Speechnet has a number of shortcomings. The most important is that the cgi-bin protocol only supports "sessionless" communication. It is not easy to construct sihlos which have a dialogue with users.

An alternative to the cgi-bin protocol is to use the Common Object Resource Broker Architecture CORBA [3].

CORBA provides a framework which allows distributed processing involving objects executing on different computers in a heterogenous computing network. CORBA is based on the OMG Object Model which provides a means by which the external behavior of objects can be specified in a way that is independent of the language in which the object is written. These specifications form an interface through which clients can request services from objects that may reside on remote compute servers running different operating systems. There are several implementations of CORBA. Some of these implementations, eg Orbix [9], support distribution of objects on Windows and Unix systems.

There are many advantages of using CORBA. In particular, it would enable the creation of more sophisticated sihlos as discussed in section 4.

9 Related work

A substantial amount of work has been done in building non-visual web browsers [21]. That work addresses a different but related problem to that addressed by the work described here.

Techniques for presenting 'speech as data', allowing a user to navigate by voice through a database of recorded speech without any visual clues have been investigated by Arons [1]. That work has some relevance to the design of hyperlinked collections of sihlos.

A non-visual browser has been developed by Morley et al. [14] but uses techniques other than speech to navigate the knowledge base.

Techniques related to executable grammar objects have been presented by other researchers. For example: Moreira and Clark [13] have developed a method which integrates formal description techniques with standard object-oriented analysis methods. In their approach, the specifications are executable and prototyping can be used to validate the specification against the requirements, thereby en-

abling the early detection of inconsistencies, omissions and ambiguities in the requirements definition. Peake and Salzman [16] have extended the functional approach to modular parsing by adding the object-oriented constructs of class inheritance and dynamic method dispatch. Their approach differs from the use of executable grammar objects as it extends the functional approach rather than adapting it for use in an object-oriented language.

10 Concluding Comments

The work described in this paper appears to be the first to propose the development of a network of speech-accessible hyperlinked objects whose grammars can be downloaded by remote speech browsers in order to achieve high recognition accuracy. The prototype implementation has demonstrated the viability of the approach. Some techniques are being developed to facilitate the construction of language processors and the design of sihlo grammars

If adequate tools can be built to simplify the construction of sihlos, Speechnet (or similar networks) have the potential to significantly improve access to knowledge and electronic functions, for visually-challenged users, and in applications where visual browsing is not appropriate.

11 Acknowledgments

The concepts underlying Speechnet, speech-accessible hyperlinked objects and executable grammar objects have resulted from a long–term NSERC-funded research project involving many people at the University of Windsor. In particular, the author wishes to acknowledge Sanjay Chitte who wrote the Java code for the Speechnet browser, Tarek Haddad, who developed a speech interface to web pages, Barbara Szydlowski who wrote a text-based web-interface to the natural-language processors, Ono Tjandra who patiently explained the advantages and disadvantages of Corba, and Walid Mnaymneh, Maunzer Batal, and Stephen Karamatos who provided technical help through the project. The author also wishes to thank Peter Best and Zdenek Vopat, two remarkable students, who volunteered to act as test-drivers surfing Speechnet.

12 References

[1] Arons, B. (1991) Hyperspeech: Navigating in Speech-Only Hypermedia. In Proceedings of Hypertext '91 (San Antonio, TX, Dec. 15-18), ACM, New York, 1991, pp. 133-146.

[2] Augusteijn, L. (1993) *Functional Programming, Program Transformations and Compiler Construction*. Phillips Research Laboratories. ISBN 90–74445–04–7.

[3] CORBA: `http://www.egr.msu.edu/~thakkarv/corba.html`

[4] Cup:`http://www.cs.princeton.edu/~appel/modern/java/CUP/CUPman.html`

[5] Frost, R. A. (1999) Improving the Efficiency of Executable Grammar Objects. University of Windsor Computer Science Technical Report CS0099–1.

[6] Frost, R. A., and Chitte, S. (1999) Sihlos: Speech-accessible hyperlinked objects. University of Windsor Computer Science Technical Report CS0099–3.

[7] Frost, R. A.. (1995) Use of executable specifications in the construction of speech interfaces, Proc. *IJCAI Workshop on Developing AI Applications for the Disabled.* University of Montreal, 1995.

[8] Frost, R. A. (1992) Constructing programs as executable attribute grammars. *The Computer Journal* 35 (4) 376 — 389.

[9] Iona: `www.iona.com`

[10] Johnson, S. C. (1975) YACC – Yet Another Compiler Compiler, CS Technical Report #32, *Bell Telephone Laboratories*, Murray Hill, NJ.

[11] Leermakers, R. (1993) *The Functional Treatment of Parsing*. Kluwer Academic Publishers, ISBN 0–7923–9376–7.

[12] Moody, T. S. (1988) The effects of restricted vocabulary size on voice discourse structures. *PhD Thesis*, North Carolina State University.

[13] Moreira, A, and Clark, R (1996) Adding rigor to object-oriented analysis. *IEE Software Engineering Journal* 11 (5) 270–280.

[14] Morley, S., Petrie, H., O'Neill, A., McNally, P. (1998) *Auditory Navigation in Hyperspace: Design and Evaluation of a Non-Visual Hypermedia System for Blind Users*. The Third International ACM SIGCAPH Conference on Assistive Technologies ASSETS '98, April 15-17, 1998, Marina del Rey, CA USA

[15] Paakki, J. (1995) Attribute grammar paradigms — a high-level methodology in language implementation, *ACM Computing Surveys* 27(2) 196–255.

[16] Peake. I. and Salzman, Eric (1997) Support for modular parsing in software reengineering. *Proc. International Workshop on Software Technology and Engineering Practice, STEP* Jul 14–18 1997, 58–66.

[17] Pereira, F. and Warren, D. H. D. (1978) Definite Clause Grammars compared with Augmented Transition Networks. *Technical Report*, Department of Artificial Intelligence, University of Edinburgh.

[18] Seneff, S. (1992) TINA: A natural language system for spoken language applications. *Computational Linguistics* March 1992 61–86.

[19] Young, S. R., Hauptmann, A. G., Ward, W. H., Smith, E. T. and Werner, P. (1989) High level knowledge sources in usable speech recognition systems. *CACM* 32 (2) 183–194

[20] Zumer, V., Korbar, N. and Mernik, M. (1997) Automatic implementation of programming languages using object oriented approach, *Journal of Systems Architecture* 43 (1) 203–210.

[21] Zajicek M., Powell C., Reeves C., (1999), 'Web search and orientation with BrookesTalk,' California State University Northridge, CSUN '99, Technology and Persons with Disabilities, Los Angles.