Reconfiguration of speech recognizers through layered-grammar structure to provide ease of navigation and recognition accuracy in speech-web.

Irfan H. Qureshi

University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation

Qureshi, Irfan H., "Reconfiguration of speech recognizers through layered-grammar structure to provide ease of navigation and recognition accuracy in speech-web." (2001). Electronic Theses and Dissertations. 3279.
https://scholar.uwindsor.ca/etd/3279

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000 ext. 3208.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
Reconfiguration of Speech Recognizers Through Layered-Grammar Structure to Provide Ease of Navigation and Recognition Accuracy in Speech-Web

By

Irfan H. Qureshi

A Thesis

Submitted to the Faculty of Graduate Studies and Research through the Department of Computer Science in Partial Fulfilment of Requirements for the Degree of Master of Science in Computer Science

At the University of Windsor.

Windsor, Ontario, Canada

2000

© 2000 Irfan H. Qureshi
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.
ABSTRACT

Developing speech interfaces to large knowledge bases is a new and challenging problem. There is a need for a solution to provide access to large knowledge bases and high recognition accuracy. A partial solution to this problem is to distribute the knowledge base into a network of speech-accessible units of knowledge. But as the number of such units increases the recognition accuracy decreases and navigation among these units becomes difficult. In this thesis, a new technique is investigated. The new technique is based on a layered grammar structure and modification of the unit's input language to provide high recognition accuracy with ease of navigation among units. This technique is a step towards a solution for high recognition accuracy and distribution transparency with ease of navigation for large knowledge bases. A prototype has been implemented to demonstrate the efficiency of the layered grammar based approach.
ACKNOWLEDGMENTS

I would like to express my respect, appreciation and thanks to Dr. Richard A. Frost, his guidance and enthusiasm made this thesis a very enjoyable endeavour for me. I am very thankful to Dr. Walid Saba and Dr. Van Den Hoven for their motivation and constructive comments. I would also like to thank Dr. Richard Frost for chairing my thesis defense seminar.

I would also like to thank the other faculty members and helpful secretaries of the Department of Computer Science Mary, Margaret and Gloria.

Last, I wish to thank my family and the graduate students in the Department of Computer Science.
LIST OF FIGURES

1) Fig A - Structure For Natural-Language Speech Interfaces — 35

2) Fig 1 - Basic Structure For the Existing Speech-Web ——— 56

3) Fig 1a – Structure For the Modified Speech-Web ——— 61

4) Fig 2- Example Fig For A System Without Gramar Layering 62

5) Fig 3 - Example Fig To Illustrate Ease Of Navigation ——— 64

6) Fig 4 - Application----------------------------- 66

7) Fig 5 - Sample Stack Of Grammars------------------ 66
TABLE OF CONTENTS

ABSTRACT---------------------------------------------------III

ACKNOWLEDGEMENTS -----------------------------------------IV

LIST OF FIGURES--------------------------------------------V

CHAPTER

1) INTRODUCTION ------------------------------------------1

1.1) Definitions---------------------------------------------3

1.1.1) Natural Language Speech Interfaces-------------------3

1.1.2) Speech Recognition-----------------------------------3

1.1.3) Speech Synthesis-------------------------------------4

1.1.4) Sihlo-----------------------------------------------4

1.1.5) Speech-Web-------------------------------------------5

1.2) The Problem-------------------------------------------5

1.3) The Solution-------------------------------------------8

1.4) Work Done---------------------------------------------11

1.5) An Example Dialogue Session-----------------------------12

1.6) Summary of Findings-----------------------------------20

1.7) Significance of Work----------------------------------21
1.8) Natural-Language Aspects of The Modified

Speech-Web ........................................22

1.8.1) Language of Each Sihlo is Very Small 22

1.8.2) System Prompts About Other Sihlos 23

1.8.3) Expectation That User Won’t Change Topics Suddenly 24

1.8.4) Expectation That User Will Ask Questions Regarding

   Closely-Related Sihlos ..................................25

1.8.5) Conclusion ........................................25
2) SUMMARY OF SURVEY ON NATURAL-LANGUAGE SPEECH INTERFACES TO DISTRIBUTED DATABASES

2.1) Natural Language Interfaces to Databases 26

2.1.1) Introduction 28

2.1.2) Advantages of NLIDBs 28

2.1.3) Disadvantages of NLIDBs 29

2.1.4) Architectures for NLIDBs 30

(I) Pattern matcher NLIDBs 30

II) Internal Representation Languages 30

III) Syntax-based NLIDBs 31

IV) Semantic grammar based NLIDBs 31
2.1.5) Limitations of NLIDBs

I) Limited Language

II) Difficulty in mapping queries to the database

III) Limited Portability

2.2) Natural-Language Speech Interfaces to Databases

2.2.1) General Structure Of NLSIs

(I) Speech Recognition
    a) Pre-processing Speech
    b) Recognition
        (i) Template matching
        (ii) Acoustic-Phonetic Recognition
        (iii) Stochastic Processing
    c) Communication
2.3) Natural-Language Interfaces To Distributed Databases

2.3.1) General Structure Of NLI To Distributed Databases

(I) Natural Language Processing Component

(II) Data Access Component

(III) File Access Manager

2.3.2) New Techniques In Information Retrieval From Distributed Databases On Web

(I) Multi-Agent Architecture

(II) Ontology- Based Web Site Mapping
2.4) Natural-Language Speech Interfaces To Distributed Databases

2.4.1) General Structure

(I) The Semantic Analyzer

(II) Dialog Manager

(III) Response Generator

2.4.2) Some Current Issues

(I) Distributed Dialogue Manager

(II) Dialogue Modelling

(III) Information Manager

(IV) Spoken Prompts Design

3) OVERVIEW OF THE EXISTING SPEECHWEB

3.1) The Recognizer Object

3.2) The Http Object

3.3) The Synthesizer Object
4) MODIFICATION OF THE SPEECH-WEB BROWSER 59

4.1) Example Database 62

4.2) How to achieve High Recognition Accuracy 65

4.3) How to achieve Ease of Navigation 67

4.4) Details of Modification of Browser Code 68

5) RULES FOR NAVIGATING THROUGH LAYERED MODULES

5.1) Ability to Go Back 93

5.2) Ability to Go Home 93

5.3) Limited number of repetitions 94

5.4) Ability to say “Where Am I?” 95

5.5) Ability to get help anytime 95

6) ANALYSIS OF EFFECTIVENESS 96

7) OVERVIEW OF THE RELATED WORK 98

8) CONCLUSION 101

9) REFERENCES 103
APPENDIX A - PROGRAMS ON THE CLIENT SIDE 115

1) SpeechClient.java .................................................. 115
2) Httpget.java .......................................................... 140
3) Httppost.java .......................................................... 141
4) MC.java ................................................................. 143

APPENDIX B - PROGRAMS ON THE SERVER SIDE 144

1) C program that wraps Miranda code .......................... 144
2) Preferences.h .......................................................... 149
3) Unix Shell Script ...................................................... 150

APPENDIX C - SAMPLE GRAMMAR AND LANGUAGE FILES

A) Grammar Files ....................................................... 151

1) navigation grammar ............................................... 151
2) welcome grammar .................................................. 152
3) sports grammar ...................................................... 153
4) wintersports grammar ............................................ 154
5) icehockey grammar ................................................. 155

B) Language Files ..................................................... 156

1) welcome.m .......................................................... 156
2) sports.m .............................................................. 158
3) wintersports.m ..................................................... 160
4) icehockey.m .......................................................... 163

VITA AUCTORIS ......................................................... 164
1) INTRODUCTION

Speech interfaces are becoming popular as they provide a new means for human interaction with computers. They provide accessibility to computers for visually-challenged users and allow hands-free use of computers. In addition, they provide a hands-free computer interface for data input and output, e.g. airline enquiry system, they provide mobile access to remote applications. Some other present and future applications can be speaker verification for secure access to the web, speech-controlled robots, surfing a database for required information, speech search engines for the web etc.

Some related work has been done by Frost and Chitte [Chitte, Frost 99], who have constructed a speech browser, using speech-activated hyperlinked objects called sikhlos distributed over the web, but there is a need for developing natural language interfaces to sufficiently large knowledge bases with high recognition accuracy and ease of navigation. The technique described in this thesis i.e use of a layered-grammar structure can be used to develop such interfaces which can be employed to develop tools so that people with relatively little knowledge of computing can “speech-enable”
various structured knowledge sources such as books, regular web pages, catalogues, databases, and language interpreters, and easily add them to a "speech-web" deployed over Internet.

This technique will be useful for the creation of a large distributed knowledge base that can be accessed through speech-only interfaces, to blind users and in applications where visual browsing is not appropriate. The "modified speech-web" will be similar to the existing web, but will be accessed by speech-only browsers, which accept spoken natural-language queries and commands.
1.1) Definitions

1.1.1) Natural-Language Speech Interfaces

Speech interfaces are applications which provide users with hands-free access to computers, by using speech. Essential components of a speech interface are a recognizer, an interpreter, and a synthesizer. Natural-language speech interfaces use natural-language syntactic and semantic rules for speech-recognition.

1.1.2) Speech Recognition

Speech recognition is a process by which audio input from speech is converted into text. The steps involved in speech recognition are grammar processing, signal processing, phoneme recognition, word recognition and result generation. In off-the-shelf speech-recognition technology all these processes are automatic and the developer has little control over them. Only the grammar can be created and activated for a recognizer, the grammar defines the words and the patterns in which they may be spoken.
The grammar constrains the recognition process making the recognition faster and more accurate as the recognizer only has to recognize the words and sentences defined by the grammar.

1.1.3) Speech Synthesis

Speech synthesis is text-to-speech conversion. The two major parts for producing spoken language are structure analysis, that processes the text to determine the formatting data used in this stage, and text preprocessing which analyses the sentences for special constructs such as dates, time, acronyms, etc.

1.1.4) Sihlo

These are speech-activated hyperlinked objects distributed over the Internet. [Frost and Chitte, 1999] Each sihlo has an associated grammar, input language, an input language query processor and voice properties. The grammar is downloaded by speech browsers to configure the recognizers. Sihlos are constructed as higly-modular executable-specifications of
attribute grammars. Sihlos contain speech-activated hyperlinks to other sihlos on the network.

1.1.5) Speech-Web

A Speech-Web consists of a collection of sihlos, which are distributed over the network.

1.2) The Problem

The problem we are dealing with is to increase the scope of a knowledge base using speech interfaces without reducing speech-recognition accuracy.

One solution is to reduce the size of the input language by using a limited vocabulary [Moody 98], including only semantically and syntactically correct utterances [Young 1989], [Seneff 1992], using the words and phrases that are easy to distinguish, etc.
Modifying the input language requires reconstruction of the query processor, which itself is a difficult job.

Another solution is to use contextual information, i.e. input can be used to guide the recognizer, but this approach is only useful if we have previous information available such as dialogue with the user or utterances from previous dictation. This solution also is not very useful in knowledge-base query applications because analysis of single-user input does not provide much information about the query.

A good solution to this problem is to distribute the knowledge base into a network of units of knowledge, each one of these having query processors, as executable specifications of attribute grammars.

This approach has been demonstrated by the construction of speechWeb [Frost and Chitte 99] which is a collection of speech-accessible hyperlinked objects called sihlos. Each sihlo has its own specific grammar, which is used to configure the speech recognizer of remote speech browsers. Each sihlo also contains the speech-activated hyperlinks to other sihlos. Experiments
have shown that high speech recognition accuracy can be achieved by using Speech-Web. [Frost and Chitte, 1999]

Now as the scope of the knowledge base increases, we need to have more sihlos, and each sihlo must not be too big otherwise the speech recognition accuracy would decrease and it must not be too small otherwise the interface language will be too limited. Now if the sihlos are interlinked together, another problem will be raised. Each sihlo must contain the speech activated hyperlink to new sihlo. As the number of sihlos and the number of hyperlinks increase, the recognition accuracy of the sihlos will decrease.

Another issue is how to design the interface language to provide naturalness of navigation i.e. to increase the expressiveness of language so that many queries of different nature can be supported as well, navigation among these different queries becomes transparent and easier without a significant decrease in speech recognition accuracy.
1.3) The Solution

1) The solution which is investigated in this thesis provides the integration of sihlos without loss of recognition accuracy. The solution uses layered-grammars. The layered-grammars can be “stacked” with the more specific grammars at the top. When an utterance is processed, the most specific grammar is used first and, if the level of certainty of recognition is below a given threshold, the next grammar is used, and so on.

Various researchers have investigated grammar-based speech-recognition systems, but no work has been done on the use of layered grammars to improve recognition accuracy.

Layering of grammars can be very useful to improve speech-recognition accuracy in applications where users can move from more general to more specific domains, e.g. when users home in on more and more specific items for which they want information.

In this solution, sihlos will be integrated in a graph-like structure. Top-level sihlos will have indexed links to a number of other sihlos. These
child sihlos will have indexed links to 'lower-level' grandchild sihlos and so on. The browser will have a stack of grammars from top-level and next-level sihlos, with the child sihlo grammar on the top of the stack and top-level sihlo grammar on the bottom of the stack. Hence, any question about the subsequent parent sihlos can be answered using the stacked grammars; and the recognition accuracy is highly increased. Each of the sihlos at a certain level can be connected to other sibling sihlos by using appropriate grammars and speech-activated hyperlinks, hence browsing among different categories of sihlos will become more natural.

2) **Ease of navigation** can be obtained by:

Designing the language such that dialogue-based speech navigation is achieved with high recognition accuracy with high distribution transparency. This can be achieved by using the network-based approach with a hierarchical graph-like structure. Layered grammars will be incorporated in this approach. The sihlos at each level will be connected to each other and to their child sihlos. Hence, the language for each sihlo can be designed to include questions which allow navigation to other sihlos. In addition, certain questions can be included in the language
which allow the user to go up and down in the network. Hence, navigation from one silo to other silo will become much easier and faster. Also, it will help the system to answer many different kinds of questions.

In summary, the thesis investigated is that:

"The use of layered-grammars in the Speech-Web system facilitates navigation while maintaining high speech-recognition accuracy."
1.4) Work Done

In order to develop the solution to the problem and implement it, the following steps were taken:

a) A detailed survey was performed on Natural-Language Speech Interfaces to Distributed Databases. A summary of this survey has been included in section 3.

b) Rules were developed for navigation amongst speech modules (sihlos) to provide ease of navigation and high speech-recognition accuracy.

c) The existing speech-web browser was modified to implement the new rules.

d) The effectiveness of the modified interface was analyzed by developing an application involving 64 sihlos.
1.5) An Example Dialogue Session

The following is an example of a dialogue with the modified Speech-Web browser using the Department Store application.

C: Hi, Welcome to the Speech Department Store. We deal in Clothing, Sports, Accessories and Electronics. Please say help for assistance.

U: Help

C: You are now in the start silhlo. You have these choices to choose from.

You can go to clothing, sports, accessories or electronics silhlos by saying clothing, sports, acessories or electronics. You can say help again. You can quit the application by saying bye. Please, speak your choice now.

U: sports.

(Hyperlink followed to the sports silhlo, sports grammar will be downloaded on top of the welcome silhlo grammar.)
C: You are now in the sports silo. We sell water sports, winter sports, indoor sports and outdoor sports goods. Please, ask help for assistance.

U: Help

C: This is the sports silo. We deal in water sports, winter sports, indoor sports and outdoor sports goods. You have these choices to chose from.
To go back to start silo, say go back or say home.
To go to water sports silo, please say water sports; to go to winter sports silo, please say winter sports; to go to indoor sports silo, please say water sports; to go to outdoor sports silo, please say outdoor sports; to go to electronics silo, please say electronics; to go to accessories silo, please say accessories; to go to clothing silo, please say clothing. Please, say your selection now.

(Now user can say any of the choices given or he can also go the clothing by saying)

U: clothing.

(In this case since the sports silo language has a speech-activated hyperlink to the clothing silo, its grammar will recognize this question and now the
sports sihlo grammar will be removed and the clothing sihlo grammar will be downloaded on top of the welcome sihlo grammar.)

C: You are now in the clothing sihlo. We deal in clothes for adults and kids. Please say help for assistance.

(User makes a choice about winter sports, by saying)

U: winter sports.

(Now, hyperlink will follow to the winter sports sihlo and its grammar will be downloaded on top of the sports and welcome sihlo grammar.)

C: You are now in winter sports sihlo. We sell skiing, and ice hockey equipment. Please, say help for assistance.

(The user can say help to listen the options. One of the options is to go to the ice hockey sihlo by saying, ice hockey. User says ice hockey)

U: ice hockey.
(Now, a hyperlink will follow to the icehockey sihlo, and its grammar will be downloaded on top of the winter sports, sports and welcome sihlo grammar.)

C: You are now in the ice hockey sihlo. We deal in ice hockey sticks, gloves and skiis. Please, say help for assistance.

(User says help)

U: Help

C: We deal in ice hockey sticks, gloves and skiis. You can go back to the winter sports sihlo, go home, go to the skiing sihlo, or ask for the price for an item by saying what is the price for sticks and so on. Please, speak your selection now.

U: what is the price of gloves.

C: They cost thirty dollars.

(Now if the user wants to know about the electronics, he can say)
U: Electronics.

(Now since icehockey grammar doesn’t understand this question, the user is prompted to say again.)

C: Please, repeat I do not understand or say help for assistance.
U: Electronics.
C: Please, repeat I do not understand or say help for assistance.
U: Electronics.

(After three repetitions, the system will download the navigation grammar and prompt the user to ask for help)

C: I do not understand your question. Please, say help for assistance.
U: Help
C: You have these choices to choose from. You can stay in the ice hockey sihlo, by saying stay; you can go back to winter sports sihlo, by saying go back; you can go to start sihlo, by saying home; you can ask where am I to enquire about the present sihlo; you can say help to get the help; or you can quit the application by saying bye. Please, say your selection now.
(User says go back. The user goes to the winter sports sihlo.)

U: go back

C: You are now in the winter sports sihlo. Please, say help for assistance.

(User says help)

C: Help

(The user listens to the help in the winter sports sihlo. But couldn’t find any option to go to the Electronics sihlo. At this point, he/she can say home to go to the start sihlo or can go back to the sports sihlo. User says go back)

U: go back

(The user goes to the sports sihlo)

C: You are now in the sports sihlo. We deal in sports goods for winter sports, outdoor sports, indoor sports and water sports. Please, say help for assistance.

U: Help

C: This is the sports sihlo. We deal in water sports, winter sports and indoor sports goods. You have these choices to chose from. To go back to the start sihlo, say go back or say home. To go to the water sports sihlo, please say water sports; to go to the winter sports sihlo, please say winter...
sports; to go to the indoor sports sihlo, please say water sports; to go to the outdoor sports sihlo, please say outdoor sports; to go to the electronics sihlo, please say electronics; to go to the accessories sihlo, please say accessories; to go to the clothes sihlo, please say clothes. To ask for help again, please say help. Please, say your selection now.

U: Electronics.

(The system goes to the Electronics sihlo)

C: You are now in the electronics department. We deal in audio and video equipment. Please, say help for assistance.

(Now, the user can ask for help and go to the video or audio equipment. User can also go to the start sihlo, by saying home)

U: home

C: You are now in the start sihlo. Please, say help for assistance.

(In the above session, since ice hockey and winter sports grammars don’t recognize questions regarding the electronics or the clothing sections, they will be removed from the stack. Since, sports grammar can recognize the
question regarding electronics, the hyperlink will follow to the electronics
sihlo and its grammar will be downloaded in place of the sports sihlo, and
further dialogue will continue.)
1.6) Summary Of Findings

The implementation of the new speech browser with layered grammars provided an excellent way to incorporate ease of navigation with high recognition accuracy. The following observations were made:

a) By using the layered-grammar structure, navigation between silos became very easy, without an increase in the size of the grammar or the corresponding silo's language. The user can easily navigate between the silos on the same level and at any time go back to the previous silo or the 'start' silo.

b) The system was found to be user friendly. For example, the user can get detailed help at any point and inform him/her self how to use the system and what to say to the system. The user is allowed to repeat the question three times, before the system prompts him/her to ask for help.

c) The system enabled the integration of silos to incorporate speech access to large knowledge bases with high recognition accuracy.
1.7) Significance of Work

This work is significant in that no one else in the world appears to be investigating real-time navigation through a layered-grammar-based speech recognition system. Some work has been done on web-browsing using voice input (e.g. BrookesTalk [Zajicek 99] or Web98) and navigating in speech only hypermedia (Hypermedia [Arons B 1991]).
1.8) Natural-Language Aspects of the Modified Speech-Web

In this chapter, some of the issues relating the natural-language aspects of the modified Speech-Web will be discussed.

In general, we will discuss how natural is it if the language of each sihlo is very small, how natural will the navigation be if the system has to prompt the user with statements about the existence of other sihlos, how reasonable is it to expect the users not to suddenly change from one topic to another. Also, the idea with the new system is that the users will ask questions in a restricted domain and then move to a closely-related domain. How reasonable is this assumption. What would be the total effect of all of these considerations on the naturalness of Speech-Web interfaces.

1.8.1) Language of Each Sihlo is Very Small

If the language of each sihlo is too small, the user will be restricted to use only the words as defined in the language of the sihlo and will only be able
to ask questions as defined by the grammar of the sihlo. The user must know exactly what kind of questions to ask. This can be provided in the help menu of each sihlo.

The recognition accuracy will be very high but the naturalness of the language will decrease. Assuming that user knows that he/she can only ask questions related to the current sihlo or ask the system to transfer to another sihlo, the sihlo language should still be sufficiently large, so that the user can ask the same query in different ways which would provide more naturalness.

1.8.2) System Prompts About Other Sihlos

If the user asks questions within a restricted domain about any closely related sihlo, it would be natural to provide the user with information about the existence of another sihlo, e.g. in the department-store application, if the user is in the clothing sihlo and asks about adult clothing, the system can prompt the user that adult clothing department has the required item. Also, the information about the sibling sihlos can be provided. It would increase the naturalness of interface. On the other hand, if the user asks a question about the ice-hockey sihlo in the clothing sihlo, the system can still prompt
the user about the existence of ice-hockey sihlo, but it would cause the language of the sihlo to become quite large and recognition accuracy would decrease.

Similarly, if the user asks something about weather in Windsor, when in the department-store application, the user can go to the index sihlo and ask the question about the weather, and then the system can prompt that this information can be found in weather sihlo. Hence, a certain level of naturalness can be provided without losing the recognition accuracy.

1.8.3) Expectation That User Will Not Change Topics Suddenly

The expectation that the user won’t change topics suddenly is not a correct assumption. As an example, the user can ask the price of an item in the department-store application and then ask a question about the cost of return ticket to Miami. In such cases, the user will be expected to go to the index sihlo and then ask the question regarding the new domain. The index sihlo can then prompt the user about the existence of the requested-information domain and can also transfer him/her to that domain.
1.8.4) Expectation That User Will Ask Questions Regarding Closely-Related Sihlos

The expectation that user will ask questions regarding closely-related sihlos is a fair assumption, as the user may ask for more specific details about a topic but it is also natural that the user may ask question about something else. In this case, if we provide links to other sihlos, in each sihlo language, the recognition accuracy would become very poor. Hence, certain links can be provided in the index sihlo about different domains. Also, each sihlo can have prompts to guide the user about the existence of sibling sihlos and child sihlos. Hence, a certain level of naturalness of language can be supported.

1.8.5) Conclusion

The overall effect of all of these assumptions on the naturalness of the system would be that a certain level of naturalness of language can be provided to the modified system, while expecting the users to understand the restrictions on the system. A good solution would be to provide the user training about the use of the system.
2) SUMMARY OF SURVEY ON NATURAL LANGUAGE INTERFACES TO DISTRIBUTED DATABASES

As part of the background work for this thesis, a survey on Natural-Language Interfaces to Distributed Databases was performed. The survey was divided into four main sections. A summary of each of the sections is included in the thesis report.

Natural-Language Speech Interfaces to Distributed Databases are gaining much popularity, as they provide fast, efficient and easier access to distributed information sources for visual as well as non-visual users.

Earlier research consisted of systems which provided non-speech natural-language interfaces to different databases. With the advent of speech technology, and dialogue-modeling techniques, speech interfaces to databases came into existence. Some of these systems are described in the survey [Qureshi,2000] which is summarized in this chapter.
Recent enhancement in web technology is a major component in the development of Natural-Language Interfaces to Distributed Databases. These systems are also discussed in the survey.

Current research in dialogue modeling, spoken-language systems, and distributed communication has played an important role in the designing and implementation of Natural-Language Speech Interfaces to Distributed Databases. The survey summarized in this section discusses in detail the architecture and the issues related to these systems.
2.1) Natural-Language Interfaces To Databases

2.1.1) Introduction

A Natural-Language Interface to a DB is a system which gives the user the capability of interacting in a natural language such as English with the database. Most of the NLIDB systems use partial Natural language Interfaces as only the query part uses natural language input, while some of them allow the user to update the database in natural-language too. [Salvester 86].

2.1.2) Advantages of NLIDBs

[Hutchison, 95] The main advantage of NLIDBs is that they allow people to interact with databases without learning the syntax of a formal language such as SQL. Also, the user is not required to know the structure of the database. Also, there is no need for advanced hardware technology in NLIDB as compared to other database query interfaces such as graphical queries.
2.1.3) Disadvantages of NLIDBs

Some of the disadvantages are:

1) The user may have to enter long questions, which can have multiple meanings, hence requiring the system to figure out the actual meanings to perform the query.

2) NLIDBs usually requires that the database be well-designed. Also, a lot of initial configuration is required.

3) The users think that the computer is more intelligent than it actually is. It may cause misunderstanding of answers to queries.
2.1.4) Architectures For NLIDBs

There are four types of NLIDBs: pattern matchers, internal-language based, syntax-based and semantic-grammar based.

I) Pattern Matchers NLIDBs:

These systems work on a) patterns to which queries are matched and b) on the actions which are taken in the case of pattern matching. These are easier to implement but can give incorrect results. An example is LADDER by G.G.Hendrix [Hendrix 78] etc.

II) Internal Representation Language

This is the most common architecture for NLIDBs. In this case the query is translated into some form meaning intermediate representation which is then converted to a database query.

The analyser is used to do syntax and semantic processing. It gives modularity to such systems. For example: the separation of the user’s
understanding of query from the production of a database query allows for an off-the-shelf linguistic front end to be used.

An early system is described in [Burger 71]: Converse which is a natural-language compiler that accepts sentences in a user-entered English query and produces a concepts network to construct semantic interpretations as computable procedures. These procedures are further evaluated by a DBMS that updates, modifies, and searches the databases according to the semantic interpretation.

III) Syntax-based NLIDBs

These systems produce a parse tree from the user's query which is then transformed to a database query by using some mapping rules e.g. Lunar [Woods et al 72].

IV) Semantic-grammar based NLIDBs

The grammar in these systems is closely tied to the database and is not very portable which gives the advantage that if a users' query is parsed, then a
suitable database query can be constructed. These systems are extensions to syntaxed-based systems and the grammar is related to the domain and task. One such front end is described in [Sparck 1983]. It uses general, i.e. domain-independent, semantic information for question interpretation.

2.1.5) Limitations Of NLIDBs

The main limitations are I) limited query language, II) difficulty in mapping queries to the databases and III) lack of portability of the system.

(I) Limited Language

The language used by NLIDB is always a subset of English. As a result the user has to formulate the queries to make them suitable for the NLIDB.

[Androutsopoulos 94] has provided two solutions to this problem:
One is to expand the linguistic range and the other is to make users aware of the limits. Restricting the general form of the queries allowed can make the users aware of the restrictions. The other solution is to allow the user to
build queries by choosing options from menus as in the Kaleidoscope database management system [Cha & Weiderhud 91].

(II) Difficulty in mapping queries to the database

Most NLIDBs have been designed with interfaces to a relational database and use SQL as the query language. As the range of the natural language increases, the ability to map the user’s query to SQL query decreases.

(III) Limited Portability

Most of the earlier systems were closely tied to their applications. Newer systems are more modular, and components can be re-used.
2.2) Natural-Language Speech Interfaces To Databases

This section gives an overview of natural-language speech interfaces to databases. We discuss some recent approaches and issues in speech interfaces to databases and an overview of some example systems is also given.

2.2.1) General Structure Of NLSIs

The main parts of a natural-language speech interfaces are a speech recognizer, a natural language processing system, a dialog management unit in case of a dialog system, and a speech synthesizer. In some of the systems, known as spoken-language systems, the speech recognizer and the NLP system are integrated using some technique.[Fig A]

Most of the development in this area has been done in the 90’s and many applications were produced in the areas of retrieval of text, product information, travel information, weather information, stock quotes, banking information, and defense applications.
Fig A [Bennacef 94]
1) Speech Recognition

Speech recognizers are generally used as human-computer interfaces for other software. They perform three main tasks [Judith 96]:

- **Pre-processing**  Converts the spoken input into a form the recognizer can process
- **Recognition**  Identifies what has been said
- **Communication**  Sends the recognized input to the software/hardware system that needs it.

(a) Pre-processing Speech

Speech is an analog waveform. Digital-signal-processing techniques are used to convert all captured speech data, noise, silences etc. into a digital format. Speech-recognition systems often include hardware to perform this conversion; others include sound cards to accomplish this.
(b) Recognition

This is the process of identifying what the user has said. It can be done in one of the following ways:

i) Template matching, ii) Acoustic-phonetic recognition, and iii) Stochastic processing

(i) Template matching

Template matching is a form of pattern recognition. Each word in the application is stored as a separate feature vector (template). Spoken input by the users is stored as templates and then compared with the application templates to find the correct match.
(ii) Acoustic-Phonetic Recognition

It basically involves three steps:

1) Feature Extraction, 2) Segmentation and labelling, and 3) Word-level recognition

During feature extraction the system examines the input for certain spectral patterns, such as frequent frequencies, which are needed to distinguish phonemes from each other. The collection of extracted features is interpreted using acoustic-phonetic rules. These rules identify phonemes (labelling) and determine where one phoneme ends and the next begins (segmentation).

The system then searches through the application vocabulary for the words matching the phoneme hypothesis.

(iii) Stochastic Processing

Stochastic processing requires the creation and storage of models of each of the items that will be recognized. It doesn’t involve any direct matching between stored models and input. It is based upon complex statistical and probabilistic analyses which are observed by the network-like structure in
which those statistics are stored, called Hidden Markov model (HMM). The recognition system processes the input by comparing it with stored models. These comparisons produce a probability score indicating the likelihood that a particular HMM is the best match for the input.

(iii) Communication

After the speech input has been identified, the application has to pass the information to other modules to use it. For this purpose, speech-enabled applications are developed.

(II) Natural-Language Processing Systems (NLPs)

An NLP is a general-purpose language processor, which builds a formal representation of the meaning of English input. It may be augmented with the DB application program i.e. an algorithm which builds a query in an augmented relational algebra from the output of the NLP. The major parts of an NLP system [Conlon, 94] are
a) The lexicon, which contains information such as, correct word spelling, a word's part of speech, syntactic behavior and semantic roles.

b) The parser, which analyzes sentences and converts them into parse trees that are easier for the computer to process.

c) The language-understanding system, which analyzes the parse trees to find the meaning of the related sentences.

d) The text generation system, which generates either computer language or natural language text output.

(III) Speech Synthesis

A speech synthesizer converts text to speech, the two major parts for producing spoken language are structure analysis, which processes text to determine the formatting data used in this stage, and text pre-processing which analyses the sentences for special constructs such as dates, time, acronyms, etc.

The other steps are a) text-to-phoneme conversion, b) prosody analysis and the final stage is c) waveform production. The important factors while judging the quality of the spoken text is naturalness and understand-ability.
Naturalness is defined by the degree to which the synthesized output sounds like human speech. Understand-ability is a measure of how well typical users can understand what is being output by the synthesizer.

IV) Natural-Language Spoken Dialog Systems

These systems provide the answer to the user’s query by using spoken dialogues [Lamel, 99], [Dybkjar, 94]. The main components of a spoken-language dialog system are a) the speech recognizer, b) the semantic analyzer which extracts the meaning of the spoken query, and c) the dialog manager, which controls the information retrieval component including database access and response creation.

Semantic interpretation is done in two steps, first a literal understanding of the query is provided, and then it is reinterpreted using the ongoing dialog. The dialog manager provides communication between the user and the DBMS. It maintains both the dialog and generation histories. It outputs a response based on the dialog state, the caller’s query, and the information obtained from the database.
2.3) Natural-Language Interfaces To Distributed Databases

In this section, we will discuss the general structure and aspects of natural-language interfaces to databases distributed over the computer network. In addition, some of the sample new techniques developed in this area will also be described.

2.3.1) General Structure Of NLI To Distributed Databases

The general structure of a Natural-Language Interface to a Distributed Databases (DDB) consists of a natural-language processing component which converts the user’s input into a subsequent query or sequence of queries to the DDB as a whole, a data-access component to create a sequence of queries against individual files, and a file access manager, which locates the required files, connects to the remote computers and constructs the answer to the user’s query.
I) Natural-Language Processing Component

This component receives the user’s question in natural language and produces a query or set of queries to the DDB. These queries, refer to specific fields, but make no assumptions about how the information in the database is stored in different files.

II) Data-Access Component

This component uses the model of the structure of the DDB to break down a query against the entire DDB into a sequence of queries against individual files. It also saves the linkages among the retrieved records, which helps in creating the answers to the overall query. This component produces the file queries in the language of the remote DBMSs but these queries refer to generic files, i.e. files containing the specific record, not the specific files in specific directories on specific machines.
III) File-Access Manager

This component finds the location of the generic files and manages the access to them. It relies on a locally-stored model showing where the files are located throughout the distributed database. When it receives the query, it searches the model for the primary location of the file (or files) mentioned in the query. It then connects the appropriate computers through the network, logs into them and opens the files that are to be accessed. If the remote computer crashes, the file becomes inaccessible, or if the network connection fails, this component has the ability to recover, and if it can find a backup file, it connects to the back-up site and sends the query again.
2.3.2) New Techniques In Information Retrieval From Distributed Databases On The Web

Some of the current techniques developed for the information retrieval from distributed databases are introduced in this section.

I) Multi-Agent Architecture

Multi-agent architecture is used for information retrieval using the Web and has proved to be quite efficient. The agents are varied depending on the nature of their work (e.g., mediator, collector, user interface etc). They cooperate and communicate with each other to support intelligent information retrieval. Some of these systems are described in [Osmar 97] and [Das 97].

II) Ontology-Based Web-Site Mapping

Ontology-based web-site mapping is used to produce conceptual meta-information which is the information obtained by merging the search results from the local collection of Web pages of useful sites. One such system is described in [Zhu, 99].
2.4) Natural-Language Speech Interfaces To Distributed Databases

The number of on-line information sources and services available is rapidly increasing, these reside on the Internet, local area networks or commercial networks such as Compuserve, and use different information-access protocols. To access these information resources, different interfaces have been created including speech interfaces. This section describes the general structure of the speech interfaces to distributed databases and then discusses some of the key issues in this area. We also, discuss some of the current systems in this area.

2.4.1) General Structure

The general structure of speech interfaces to distributed databases consists of a) a speech recognizer, b) a natural-language processing system, c) a dialog-management unit in case of spoken-language dialog systems, d) some communication protocol between different units, e) distributed-information resources, and f) a speech synthesizer.
These parts are already described to some extent in section 2.2. However, a
detailed discussion on dialog design and management will be provided in
this section.

(I) The semantic analyzer

The output of a speech recognizer is passed to the semantic analyzer.
The semantic analyzer often uses a case-frame approach to extract the
meaning of the sentence. A case is a relationship between a predicate
(usually a verb) and one of its arguments. A case frame of a predicate is a set
of cases, which is related to that predicate. Case frame grammars are used to
construct the semantic frame [Hayes, 86], [Matrouf, 90]

The case-frame parser parses an input sentence using the whole set of case
frames. It generates a semantic frame representing the meaning of the
sentences. Sometimes the sentence may contain multiple queries, which will
result in the generation of multiple semantic frames. Sentence parsing is
done by first selecting the relative case frame using keywords and then
building a semantic frame representation of the meaning of the sentence by
instantiating its slots. The parser is recursively applied on the sub-case frames until there are no suitable words left to fill in the slots.

(II) Dialog Manager

[Bennacef 96] The output of the semantic analyzer is passed to the dialog manager. Based on this input and the different knowledge sources available to it, the dialog manager generates an appropriate response to the user. These knowledge sources include the system-user dialog exchange history, the long-term dialog history and a task model. Using these sources a dialog state is formed.

A dialog can have several different phases such as: acquisition, negotiation, navigation, post-acceptance, and meta-communication. In the acquisition phase, the information needed to complete the current task is obtained. The negotiation phase is the phase when the user modifies the request, after getting the information from the system. The navigation occurs when the user changes from one task to another. Post-acceptance phase occurs if the user accepts the solution given by the system. The system then proposes other tasks and if the user does not enter into any task, the system closes the
dialog. Meta-communication deals with the detection and treatment of errors.

Each user’s input goes through three interpretation steps:

a) Contextual Understanding

It is the understanding of the semantic frame from the semantic analyzer taking into account the ongoing dialog.

b) Switching

The dialog manager makes this decision if there is a change in task (navigation) or an error (i.e. conflict between the contextual understanding and the dialog histories)

c) (i) Continuation

If there is no change of task or error detection, the ongoing task is continued, i.e. item acquisition or database access.
(ii) Change of task

The current task is ended and the new task is started using the task model. Also, information about the previous task is stored so that the system can go back to it.

(iii) Error recovery

In case of an error, the error-recovery is based on dialog phase (acquisition or post-acquisition), the current task, the dialog state and histories.

(III) Response Generator

The response generator creates the database query, accesses the database, and presents the result to the user. In many systems a SQL request is created using the semantic frame employing specific rules, where each rule forms a part of the SQL request. These rules can be easily modified. The generated SQL request is used to access the database and retrieve information. This
retrieved information is converted into a natural-language response and converted to speech using a speech synthesizer.

2.4.2) Some of the Current Issues

Some of the issues regarding spoken-language distributed systems will be discussed in this section:

I) Distributed Dialogue Manager

[Scott, 92] has described a distributed structure for the dialogue manager. It consists of five modules. The linguistic interface module provides the interface between the dialogue manager and the parser. The dialogue module is responsible for overall dialogue flow. The belief module requires context information from the dialogue module in order to guide the interpretation process. The belief module also co-operates with the message-planning module to provide semantic descriptions of concepts referenced in the plan of system utterances. And the task module is responsible for consulting databases and informing the dialogue module about the current state of the task.
(II) Dialogue Modeling

[Jones, 93] has described a theory for dialogue modeling which can be used for the analysis and generation of the dialogue. According to this theory, different types of dialogues (lecture, interviews etc) can be categorised by Dialogue Structure Models DSM. The dialogue elements DEs are the factors, which control the dialogue structure. Each constraint associated with DE is a possible action, which the system may take. Once the system has selected a constraint, this information is used by the generation module to create a response. This theory has been implemented as a part of a large NLP system named LOLITA.

III) The Information Manager

[Whittaker, 97] In information retrieval using spoken-language dialogue systems, some of the inputs are required to match information stored in the database, however complexity can arise as such databases are irregular. In addition, many forms of information such as names and addresses can be variable – for example, multiple valid spellings of surnames, abbreviated first names, etc. A distinct approach, incorporating an Information Manager
is used to allow the vocabulary of the application to be mapped to the
vocabulary of the dialogue and it also takes care of the various ways in
which the information may be presented by users, and should be presented to
them.

The information manager uses a data model, which contains a number of
vocabulary models, each associated with one vocabulary within the
application. These models distinguish between the items of vocabulary as
represented in the database and the alternative ways in which they may be
spoken and spelled by the caller.

**IV) Spoken Prompts Design**

[Brian, 97] has presented a set of heuristics for spoken-language dialogues
for choosing appropriate prompt styles and showed that a set of dimensions
can be formed by using these heuristics. A point in the space formed by
these dimensions is a style for the prompts. A toolkit has also been
developed to automatically generate prompts in a variety of styles.
3) OVERVIEW OF THE EXISTING SPEECH-WEB

The work described in this thesis report involves an extension to an existing speech interface to distributed data. This interface is called Speech-Web [Frost, Chitte, 1999].

Speech-Web consists of sihlos on the internet. Each sihlo resides on a server and consists of a grammar associated with its input language, a speech processor and certain voice characteristics. The speech-browser contacts a main sihlo, and downloads the grammar and voice properties to configure its speech-recognizer and voice synthesizer. The user can ask questions related to the current sihlo. The browser converts the recognized input and sends the text to the sihlo. The sihlo finds the answer to the user’s query and sends it to the browser. The browser converts the message to spoken output. If the user’s query involves another sihlo, the hyperlink of the other sihlo with some message is sent to the browser. The browser converts the message to spoken output and then contacts the new sihlo. Then, the grammar from the new sihlo is downloaded and the browser is configured with this grammar. The user can ask questions according to the grammar and the questions are answered by corresponding sihlo. The advantage of downloading the
grammar files from the remote locations is that the user who is creating the
silios can create the grammar files associated with each particular siliol thus
reducing the size of the input language which increases the recognition
accuracy.
3.1) Basic Structure

The basic structure of the existing Speech-Web is shown in Fig 1.
Fig B shows the major components of the Speech Browser: the Synthesizer object, the Recognizer object, Sihlos, and the Miranda W/AGE program wrapped in C.

The speech interface was built using Java and the C wrapper for Miranda was modified to suit the needs of this application. The recognizer and synthesizer engines of IBM via-voice were used.

3.1.1) The Recognizer Object

This object initializes the via-voice recognizer engine and recognizes the input as constrained by the Grammar. The grammar-file is a text file with ".gram" extension and is based on Java Speech Grammar Format.

The Grammar file is loaded either from the local machine or from a remote sihlo through the web. This information is obtained during the start up. The web address of the remote sihlo is also obtained at the start up.
3.1.2) The Http Object (in the browser program)

The Http Object is responsible for fetching the grammar files, and the results of the queries. There are two Http objects **HttpGet** and **HttpPost**.

The **HttpGet** object is used to download the grammar file. The **HttpGet** object executes CGI get request on a shell script, passing the grammar file name as the parameter. The result is stored in a string object and returned to “load grammar” method of the recognizer object.

The **HttpPost** object is used to query the sihlo. The recognized text is used to construct a **HttpPost** query on the respective sihlo. The result obtained is stored in a string object. This string is then passed to speak method of the synthesizer. Also this string is appended to the display so that sighted users can read it.

3.1.3) The Synthesizer Object (in the browser program)

The synthesizer object is initialized during the initialization of the interface. The resultant string obtained is passed to the speak method of the synthesizer object. The synthesizer object uses the sound card of the computer to speak out the result.
4) MODIFICATION TO THE SPEECH-WEB BROWSER

The work described in this thesis involved modifying the Speech-Web software. These modifications were carried out by the author. Fig 1 a.

The modified Speech-Web browser acts as follows:

The main sihlo sends a certain message to the speech browser, including the hyperlink of the sihlo associated with the choice made by the user. The message is converted to synthesized voice output and the browser may contact the new sihlo. Then, it downloads the grammar from the new sihlo and this grammar will be stacked on top of the main sihlo's grammar. This stacking of grammars is the major change to the Speech-Web browser. The new sihlo sends a certain text to the browser including the choices to select from the subsequent child sihlos or sibling sihlos, which will be converted into the synthesized voice output. If the user selects the sibling sihlo, then the grammar for the current sihlo will be removed and the grammar for the sibling sihlo, will be stacked on top of the main sihlo grammar. On the other hand, if the user selects the child sihlo, then the grammar for the child sihlo will be downloaded and stacked on top of the current stack. At each level, a
certain dialogue will be supported with the help of downloaded grammars. For example, if the user input is recognized by top-level grammar, then the top level shilo will be contacted for the query response and so forth. The languages associated with each shilo will be designed to provide certain dialogue-based tasks.
The basic structure of the modified Speech-Web is shown in Fig 1 a.
4.1) An Example Database:

To explain our new approach, an example database is used. Let us suppose this is the database for a department store, dealing with clothing, accessories, electronics and sports sections while each of these sections are subdivided into different sub-categories.

Let us suppose that we have 512 sihlos altogether. If we use a single layered structure, the input language of the index sihlo used to connect all the sub-categories and 512 sihlos would be quite large, as a result of which recognition accuracy would be subsequently decreased. If the user makes a certain choice, the index-sihlo has to go through all possible options i.e. 512 to find the correct choice, and then connect to that sihlo (Fig 2)
Instead, if we use the layered structure, a much smaller language is required at each level and, hence, recognition accuracy improves, as selection has to be made from fewer choices. For example, if we divide 512 silos into a tree-like structure with 8 child silos connected to each silo, to get to the bottom level silo, the top-level silo has to find the correct silo from 8 choices, the second-level silo, has to select from 8 choices again, and the third level silo has to select from 8 choices, again, hence in total only 24 choices are needed to get to the correct silo, instead of 512 in the single layered structure.

(Fig 3)
FIGURE 3
4.2) How To Achieve High Recognition Accuracy

In the department store database example, the top-level or welcome sihlo will contain speech-activated hyperlinks to a sports sihlo, a clothing sihlo, an accessories sihlo and an electronics sihlo. At the second level the sports sihlo, clothing sihlo, accessories and the electronics sihlo, will be connected to each other by speech-activated hyperlinks incorporated in their languages, and their grammars will support any question regarding navigation to each other. Any query at this level will be supported by a stack of grammars containing a second-level grammar stacked above the grammar of the welcome sihlo.

Let’s suppose the sports section consists of outdoor sports, winter sports, and indoor sports. Similarly outdoor sports consists of running, cycling etc., winter sports consists of snowboards, skiing, ice hockey etc., and indoor sports consists of basketball, volleyball etc. In this case, outdoor sports, winter sports and indoor sports sihlos will be connected to each other. As a result of a query to any of the sihlo at this level, each of these sihlos, will have its own grammar stacked on top of the sports sihlo grammar and
welcome sihlo grammar. (FIG 4) Now, the ice hockey sihlo under winter sports category, will be inter-connected to skiing and snowboarding sihlos, and it will have its own grammar stacked above the winter sports sihlo grammar, sports sihlo grammar and the welcome sihlo grammar. (FIG 5)
4.3) How To Achieve Ease Of Navigation

In the previous example of the department store, if the user asks about snowboards, the snowboard silo will be contacted for the user’s enquiry. However, if the user wants to ask about ice hockey, the snowboard silo can’t reply to this question, and its grammar will be removed. The snowboard grammar will however recognize the user’s request to go to ice hockey grammar and it will tell the browser the hyperlink for the ice hockey silo. This is incorporated because, snowboard language is designed to answer the question to go to ice hockey as well as skiing silos. The snowboard silo language contain the hyperlink to ice hockey silo and ice hockey silo contain the hyperlink for snowboard silo. In this case the ice hockey silo can send the hyperlink of snowboard to the browser, without going to the winter sports silo and downloading the winter sports silo grammar. Hence, much ease of navigation can be obtained by using speech-activated hyperlinks to the silos in a network-type design. Hence, dialogue-based speech navigation is possible with high recognition accuracy.
4.4) Details Of The Modification In Browser Code

The details of the modification made to the existing browser code are included in the Appendix I.

The SpeechClient class is the main class of the system. The Http protocol is used for communicating with the remote executable sihlos. The two methods HttpGet and HttpPost are implemented by the two classes HttpGet and HttpPost respectively. The DataInputFrame object is used to obtain information from the remote grammar file. SpeechClient has all of the necessary methods and object references to support the functionality of the interface.

When the system is started, the welcome grammar is loaded from the remote URL to the recognizer. It is stored on the local machine at the bottom of stack. The recognizer is loaded with the grammar.

The resultSpeech method is used to access the spoken text and send it to the welcome sihlo by doing an httpPost request. The result from the sihlo is checked for any grammar change by checkGrammarChange method. This
method checks if the result string starts with the token “STACKREMOVE=” or “STACK=”.

If it starts with “STACKREMOVE=”, the next token contains the stack level from which the current file has to be removed. The further tokens contains the name and web address of the file which has to be downloaded and stored on the stack at the current level. The readRemoteGrammar method is used to store the grammar file name, sihlo name and web address of the sihlo in the corresponding arrays. The grammar file is then loaded on the recognizer.

If the result string starts with “STACK=”, the next token contains the stack level of the file to be loaded to recognizer from the local machine. The grammar file name, cgi-bin filename, and web address are also loaded from the corresponding arrays.

The code for the checkGrammarChange method is as follows:
public String checkGrammarChange( String result ){

    /*** tokenize the string based on "=" */

    StringBuffer result1 = new StringBuffer();
    StringBuffer vType = new StringBuffer();

    if ( result.startsWith("STACKREMOVE=")) {

        String name = "";

        /*** tokenise the string and the appropriate variables */

        StringTokenizer tokens = new StringTokenizer(result,";");

        while(tokens.hasMoreTokens()) {

            result1.append(" you are now in ");

            filename = stringValue( tokens.nextToken() , delimiterValue ) ;

            webAddress = stringValue( tokens.nextToken() , delimiterValue ) ;

            cgibinFileName= stringValue( tokens.nextToken() , delimiterValue ) + "/" +
            (grammarFileName=valueString(tokens.nextToken(),delimiterValue))="/"+".so;

            name = grammarFileName;

            shilnones[Integer.parseInt(filename) - 1] = name;

        }

    }

}
grammarFileName = grammarFileName+".gram";

vType.append( valueString( tokens.nextToken() , delimiterValue ));

/** load new grammar, check if the recogniser has a grammar attached to it */
/* yes delete the grammar and add the new grammar make a cgi-bin call */

try {

    RuleGrammar[] delGrammar = recog.listRuleGrammars();
    for (int j= 0; j < delGrammar.length;j++){
        recog.deleteRuleGrammar( delGrammar[j]);
        recog.commitChanges();
    }

    readRemoteGrammar(recog);
    recog.commitChanges();
    resultList.append(name);
    resultList.append("sihlo.");
}

} catch( Exception delGram) {

    System.err.println( "Exception deleteing Grammar");
    delGram.printStackTrace();

}
if (vType.toString().equalsIgnoreCase("female")){
    flagFemaleVoice = true;
    flagMaleVoice = false;
}
else{
    flagFemaleVoice = false;
    flagMaleVoice = true;
}

return result1.toString();
}
else

if ( result.startsWith("STACK="))
{
    StringTokenizer tokens = new StringTokenizer(result,";");
    while(tokens.hasMoreTokens()) {
        result1.append("You are now in ");
        filename = valueString( tokens.nextToken() , delimiterValue ) ;
        grammarFileName = filename +".gram";
        vType.append("female");
    }
}
/** load new grammar, check if the recogniser has a grammar attached to it
* yes delete the grammar and add the new grammar.  */

try{
    RuleGrammar[] delGrammar = recog.listRuleGrammars();
    for (int j = 0; j < delGrammar.length;j++){
        recog.deleteRuleGrammar( delGrammar[j]);
        recog.commitChanges();
    }
    readLocalGrammar(recog);
    recog.commitChanges();
    String name = sihlonames[Integer.parseInt(filename) - 1];
    result1.append(name);
    result1.append(" sihlo.");
    }catch( Exception delGram){
        System.err.println("Exception deleteing Grammar");
        delGram.printStackTrace();
    }
    return result1.toString();
}

return result;
The code for `readRemoteGrammar` method is as follows:

```java
void readRemoteGrammar(Recognizer recog) {
    try {
        HTTPget get = new HTTPget("www.cs.uwindsor.ca", 80);
        get.submit("/cgi-bin/iqures1/h.sh", grammarFileName);
        System.out.println(grammarFileName);
        System.out.println(get.result());
        StringReader strReaderGrammar = new StringReader(get.result());
        rulegram = recog.loadJSGF(strReaderGrammar);
        if (filename == "1")
            speak("Welcome to main sihlo. Please say help for assistance");
        filenames[Integer.parseInt(filename) - 1] = grammarFileName;
        webaddresses[Integer.parseInt(filename) - 1] = webAddress;
        cgibinfilenames[Integer.parseInt(filename) - 1] = cgibinFileName;
        String file = filename + ".gram";
```
FileWriter fw = new FileWriter(new File(file));
fw.write( get.result());
fw.close();
rulegram.setEnabled(true);
recog.commitChanges();
/** create info object and add to history */
LocInfo loc=new LocInfo(webAddress, cgibinFileNamé, grammarFileNamé);
history.addElement(loc);
}catch(Exception gram){
System.err.println("Grammar file exception unable to get from URL");
gram.printStackTrace();
}

The code for the readLocalGrammar method is as follows:
void readLocalGrammar( Recognizer recog) {
    try {
        Reader reader = new FileReader(grammarFileName);
        File inputFile = new File(grammarFileName);
        FileReader in = new FileReader(inputFile);
        rulegram = recog.loadJSGF(reader);
        rulegram.setEnabled(true);

        /** create info object and add to history */
        webAddress = webaddresses[Integer.parseInt(filename) - 1];
        cgibinFileName = cgibinfilenames[Integer.parseInt(filename) - 1];
        String grammarFile = filenames[Integer.parseInt(filename) - 1];
        grammarFileName = grammarFile;
        System.out.println(grammarFileName + " loaded");
        LocInfo loc=new LocInfo(webAddress,cgibinFileName,grammarFileName);
        history.addElement(loc);
        reader.close();
    } catch (Exception readlocal) {
    
}
readlocal.printStackTrace();

System.err.println("Exception in read local grammar");

To allow the repetition for up to three times, a modification has been made to the `resultRejectedSpeech` method. This method uses a counter, if the counter is less than 3, the user is prompted to speak again, and if the count exceeds 3, a navigation grammar is loaded from the remote site to the recognizer. This navigation grammar guides the user what to do.

The code for `resultRejectedSpeech` method is as follows:

```java
void resultRejectedSpeech(ResultEvent e) {
    if(counter < 3) {
        String Result = "Please repeat I do not understand.";
        speak(Result);
        counter++;
    }
```
else {

    counter = 0;

    grammarFileName = "navigation.gram";

    flagFemaleVoice = true;

    String Result3 = "I am sorry I could not understand. Please say, help for assistance.";

    speak(Result3);

    /** load new grammar, check if the recogniser has a grammar attached to it
    /* yes delete the grammar and add the new grammar. */
    
    try{
        
        RuleGrammar[] delGrammar = recog.listRuleGrammars();

        for (int j = 0; j < delGrammar.length;j++){
            
            recog.deleteRuleGrammar( delGrammar[j]);

            recog.commitChanges();
        }
    }

    try{

        HTTPget get = new HTTPget ( "www.cs.uwindsor.ca", 80 );

        get.submit("/cgi-bin/iqiues1/h.sh", "navigation.gram");

        System.out.println(grammarFileName);

        System.out.println ( get.result() );

    }
StringReader strReaderGrammar = new StringReader(get.result());

rulegram = recog.loadJSGF(strReaderGrammar);

rulegram.setEnabled(true);

recog.commitChanges();

} catch(Exception gram) {

System.err.println("Grammar file exception unable to get from URL");

gram.printStackTrace();

}

recog.commitChanges();

} catch( Exception delGram) {

System.err.println( "Exception deleting Grammar");

delGram.printStackTrace();

}

}
The `resultSpeech` method checks if the user’s spoken text matches with any of the tags in the navigation grammar. If the user says “help”, it provides the user with certain options. The user can go back, go home, quit the program, stay in the same sikhlo, ask for help again or ask where he is now. The code for `resultSpeech` method is as follows:

```java
void resultSpeech(ResultEvent e) {
    FinalRuleResult resultSentence = (FinalRuleResult)( e.getSource() );
    String tags[] = resultSentence.getTags();
    ResultToken tokens[] = resultSentence.getBestTokens();
    StringBuffer textSpoken = new StringBuffer();
    if (tags[0].equals("sentence")) {
        for ( int i = 0; i < tokens.length ; i ++ ) {
            textSpoken.append( tokens[i].getSpokenText() + " ");
            System.out.print(tokens[i].getSpokenText()+" ");
        }
        textFieldQuery.setText("\n");
        textFieldQuery.setText(textSpoken.toString());
        lastTextSpoken = textSpoken.toString();
    }
}
```
/** this is for echo command */

String question = "question=" + textFieldQuery.getText().trim();

HTTPRequest = new HTTPPost(webAddress, cgibinFileName, question);

String result = postRequest.getResults();

System.out.println(result);

/** pass the result to string processing check for any change in grammar file */

String result1 = checkGrammarChange(result);

textAreaDisplay.appendText(result1);

textAreaDisplay.appendText("\n");

speak(result1);

/** change the voice if the flag is set true */

if (flagFemaleVoice){
    synthProp = synth.getSynthesizerProperties();

    try{
        voice[1] = new Voice();
        voice[1].setGender(Voice.GENDER_FEMALE);
        voice[1].setAge(Voice.AGE_DONT_CARE);
        synthProp.setVoice(voice[1]);
}catch ( Exception femVoice){
    System.err.println ( "Set Female voice failed ");
}

}else if(flagMaleVoice){
    try{
        Voice maleVoice = new Voice();
        maleVoice.setGender(Voice.GENDER_MALE);
        maleVoice.setAge(Voice.AGE_MIDDLE_ADULT);
        synthProp.setVoice(maleVoice);
    }catch(Exception femVoice){
        System.err.println ( "Set Female voice failed ");
    }
}

else if (tags[0].equals("abort")) { 
    try {
        synth.deallocate();
        synth.allocate();
    }catch ( Exception dealloc)
{  
    dealloc.printStackTrace();
}

} else if (tags[0].equals("pause")) {
    synth.pause();

} else if (tags[0].equals("resume")) {

    try {
        synth.resume();
    } catch (Exception resume) {
        resume.printStackTrace();
    }

} else if (tags[0].equals("bye")) {

    String s = "Thanks for coming to the SpeechWeb. Please come again. Have a nice day. " ;

    /** set output to text area */
    textAreaDisplay.appendText(s);
/** set output to the speaker */

speak(s);

try{
    synth.waitForEngineState(Synthesizer.QUEUE_EMPTY);
    synth.deallocate();
}
catch (Exception ealldealloc){
    System.err.println ( " ERR IN DEALLOCATING" );
}

System.exit(0);

} else if (tags[0].equals("back"))
{
    counter = 0;
    if(Integer.parseInt(filename) > 1)
    {
        filename = String.valueOf(Integer.parseInt(filename) - 1);
        webAddress = webaddresses[Integer.parseInt(filename)-1];
cgibinFileName = cgibinfilenames[Integer.parseInt(filename) - 1];

grammarFileName = filename + ".gram";

flagFemaleVoice = true;

/** load new grammar, check if the recogniser has a grammar attached to it
 * yes delete the grammar and add the new grammar. */

try{

    RuleGrammar[] delGrammar = recog.listRuleGrammars();

    for (int j = 0; j < delGrammar.length;j++){

        recog.deleteRuleGrammar( delGrammar[j]);

        recog.commitChanges();

    }

    readLocalGrammar(recog);

    recog.commitChanges();

    grammarFileName = filenames[Integer.parseInt(filename) - 1];

    String name = sihlonames[Integer.parseInt(filename) - 1];

    String Result = "You are now in the "+ name + "sihlo. Please say

    help for assistance";

    speak(Result);

}catch( Exception delGram){

    System.err.println( "Exception deleting Grammar");

}
delGram.printStackTrace();

}

}

if(Integer.parseInt(filename) == 1)
{
    filename = "1";
    String rejResult = "Already at home can not go back. Please say help ";
    speak(rejResult);
}

else if (tags[0].equals("home"))
{
    counter = 0;
    filename = "1";
    webAddress = webaddresses[Integer.parseInt(filename)-1];
    cgibinFileName = cgibinfilenames[Integer.parseInt(filename) - 1];
    grammarFileName = filename + ".gram";
    flagFemaleVoice = true;
/** load new grammar, check if the recogniser has a grammar attached to it */

try{
    RuleGrammar[] delGrammar = recog.listRuleGrammars();
    for (int j = 0; j < delGrammar.length;j++){
        recog.deleteRuleGrammar( delGrammar[j]);
        recog.commitChanges();
    }
    readLocalGrammar(recog);
    recog.commitChanges();
    String Result = "You are now in the start silho.
                    Please say help for assistance";

    speak(Result);
}

}catch( Exception delGram){
    System.err.println( "Exception deleteing Grammar");
    delGram.printStackTrace();
}
else if (tags[0].equals("stay"))
{
    counter = 0;
    webAddress = webaddresses[Integer.parseInt(filename)-1];
    cgibinFileName = cgibinfilenames[Integer.parseInt(filename) - 1];
    grammarFileName = filename + ".gram";
    flagFemaleVoice = true;

    /** load new grammar, check if the recogniser has a grammar attached to it
     * yes delete the grammar and add the new grammar.  */
    try{
        RuleGrammar[] delGrammar = recog.listRuleGrammars();
        for (int j= 0; j < delGrammar.length;j++){
            recog.deleteRuleGrammar( delGrammar[j]);
            recog.commitChanges();
        }
        readLocalGrammar(recog);
        recog.commitChanges();
    }
grammarFileName = filenames[Integer.parseInt(filename) - 1];

String name = sihlonames[Integer.parseInt(filename) - 1];

String Result2 = "You are now at the " + name + " sihlo. Please say help for assistance.";

speak(Result2);

} catch( Exception delGram){

    System.err.println( "Exception deleting Grammar");
    delGram.printStackTrace();

}

else if (tags[0].equals("where"))
{

    counter = 0;
    flagFemaleVoice = true;

    grammarFileName = filenames[Integer.parseInt(filename) - 1];
    String name = sihlonames[Integer.parseInt(filename) - 1];
    String Result2 = "You are now at the " + name + " sihlo. Please say help for assistance.";
speak(Result2);

}  
else if (tags[0].equals("help"))
{
    counter = 0;
    if((Integer.parseInt(filename) - 1) > 0 )
    {
        String Result1= "You are now at the " + 
        sihlonames[Integer.parseInt(filename) - 1] + 
        "sihlo.";
        Result1 += "You have these options. If you want to stay in this 
        sihlo, please say, stay.";
        Result1 += "If you want to go to the " + 
        sihlonames[Integer.parseInt(filename) - 2] + 
        " sihlo, please say, go back.";
        Result1 += " If you want to go home, please say, home. ";
        Result1 += " If you want to ask where you are, say, where am i";
        Result1 += " If you want to end the application, please say, bye.";
        speak(Result1);
else
{
    filename = "1";
    webAddress = webaddresses[Integer.parseInt(filename)-1];
    cgibenFileName = cgibenfilenames[Integer.parseInt(filename) - 1];
    grammarFileName = filename + ".gram";
    flagFemaleVoice = true;

    /** load new grammar, check if the recogniser has a grammar attached to it
     * yes delete the grammar and add the new grammar. */
    try{
        RuleGrammar[] delGrammar = recog.listRuleGrammars();
        for (int j = 0; j < delGrammar.length;j++){
            recog.deleteRuleGrammar( delGrammar[j]);
            recog.commitChanges();
        }
        readLocalGrammar(recog);
        recog.commitChanges();
}

91
String Result = "You are now in the start sihlo. Please say help for assistance";

speak(Result);

} catch (Exception delGram){
    System.err.println( "Exception deleting Grammar");
    delGram.printStackTrace();

}
5) RULES FOR NAVIGATING THROUGH LAYERED MODULES

To implement efficient navigation among silos, the following rules were designed and implemented.

5.1) Ability to Go Back

The user can go back to the previous silo at any point. It is achieved by using the "go back" link to the previous silo in each silo language.

5.2) Ability to Go Home

The user can go to the start silo at any point by saying "home". This is achieved by having "home" link to the start silo in each silo language.
5.3) Limited number of repetitions

The system allows a maximum of three repetitions per query. After that the user is prompted to ask for help. This is achieved by using a navigation grammar being downloaded in case maximum repetitions are exceeded. This navigation grammar prompts the user to say “help” and as a result of which the user is provided with the options to go back, go home, stay in the same silo, or quit the program. The user can also inquire for the name of the current silo by saying “where am I”. Hence, the system is quite user friendly.

5.4) Ability to say “Where Am I?”

If the user gets confused at any point, he/she can say “where am I”. The system will give him/her the name of the silo, he/she is in right now. Then the user can ask the appropriate questions, to get the guidance through the system.
5.5) Ability to get help anytime

The user can get detailed help at any point. Each sihlo has its own detailed help, informing the user what to say to get the information, go back to which sihlo, go home, what type of questions to ask, etc. Hence, the user can easily navigate through the system by listening to the help prompts.
6) ANALYSIS OF EFFECTIVENESS

A Speech-Web Department Store application has been created to test the effectiveness of the layered-grammar approach to navigating through a set of silos. The user's confusion at any point is reduced by providing detailed help in each module. This help tells the user where he is, what to say in order to go back to previous level, go home or go to any other department, inform about the items in the current department and what kind of questions to ask to get the price of an item. Hence, the system is very user friendly.

Recognition accuracy is improved by limiting the number of repetitions to three, as the user can ask for help if he/she is not sure what to say. If the number of repetitions is less than three, the help hard-coded in the language of the particular silo will be used to assist the user about the current department. Otherwise, if the repetitions exceed three, a navigation grammar will be downloaded. This navigation grammar tells the user where he/she is, how to go back to previous level, how to stay in the current department, or how to quit the application.

Hence, the user is assisted quickly in case of any confusion.
Further improvement to recognition accuracy could be made by further modifying the system to use a recording of the user speech and test it against all the grammars on the stack. In this case the grammar which matches the user’s query would be able to provide the answer to the user’s query. Hence, the number of repetitions would be greatly reduced. However, this modification is beyond the scope of this project and could be included in future modifications to the system.

Several users have used the system using normal rate of speech, and the recognizer was able to recognize their speech, thus the application is an example of continuous speech and speaker-independent recognition. In most of the cases, the users were able to control the system within a few minutes of use.
7) OVERVIEW OF THE RELATED WORK

Some of the related work and recent projects in the speech-recognition field are discussed. But their design and application is quite different from the approach described here. Not much work has been done on the speech-access to remote applications using the Internet.

Sativa Srinivasan, [1999] designed a speech-recognition application using the object-oriented approach which was a part of the Medspeak/Radiology product with John Vergo at IBM T.J Watson Research Centre. This application helps radiologists to produce diagnostic reports by speaking. Medspeak is only designed to be used by radiologists.

Web 98 [http://econointl.com/sw/] is a speech-activated web browser that helps visually challenged users in navigating the complex web pages. It can be used as a stand alone application providing a “Talking Web Browser”. It uses a software-based speech synthesizer and Internet Explorer as the engine.
Haddad, T. 1997 [Frost, Haddad 1998]. This speech-in/speech-out web-browser was developed by Mr. Tariq Haddad at the University of Windsor.

This browser recognizes continuous speech commands and uses a text-to-speech synthesizer to reply back to the user. It uses a PE-500 speech recognizer from SSI technology and a Dectalk text-to-speech synthesizer. The grammar is a part of the implementation code.

Josh Bers, 1997 [Bers 98]. Voicelog prototype system. This application is a multimodal interface to applications running on a web server. It uses a Java-based interface and speech-recognition is achieved by using a client-server approach. It uses both pen and speech input from the online user using a web-browser. By using this system, the user can get the vehicle models and then order selected parts by the help of these models.

BrookesTalk is a web browser, which can be used by blind, visually impaired as well as sighted user and it allows the user to search the web for information as well as scan the Web pages. [http://www.brookes.ac.uk/schools/cms/resarch/speech/btalk.htm]
Nicole Yankelovich, 1995 [Yankelovich 95]. Speech Acts is used for spoken natural-language applications and conversational speech applications can be built using SpeechActs. It was developed at Sun labs 1993-1997. The telephone is used to provide access to remote applications.

Arons B 1991. Hyperspeech. [Arons 91] Navigating in Speech-Only Hypermedia. This application is developed to allow the user to navigate through a recorded speech database. No visual clues are needed for browsing. This project could be extended to applications such as the use of recorded speech, for personal memory aid. It can help the user to create, organize and filter audio notes.
8) CONCLUSION

The layered grammar approach to integrate silos seems to be very effective in terms of recognition accuracy and navigation among silos. Hence, large knowledge bases could be accessed efficiently and reliably using this approach.

The application implemented is a prototype system, however, further conversational dialogs can be included to make it more user-friendly. The system is a low-cost, user-independent continuous-speech interface to remote applications and a number of different applications can be created in the same way, such as airline reservation systems, business applications, telephone directory assistance, etc.

The system could be modified by using recorded speech. The user’s query could be recorded as a wave file and then compared with each of the grammar on the stack. The silo, whose grammar corresponds to the user’s query at first, would be able to answer it. Hence, the number of repetitions would be kept to minimum, therefore, increasing the recognition accuracy.
However, this work is beyond the scope of this thesis and could be implemented in future.

In this implementation when the user switches to a different silo using the speech link, the recognizer is reconfigured for the grammar associated with the new silo. The recognizer now recognizes queries for the new silo. However, the recognizer can be configured for multiple grammars and then the query recognized is sent to the appropriate silo.

Future work could also include a detailed investigation of different aspects of human-computer interaction. For example, how users react to different situations such as misrecognition, different synthesized voices, rate of spoken text and providing feedback to the designer so that more robust conversational interfaces can be constructed.
REFERENCES


[This paper describes a hyperspeech system, which is a speech-only hypermedia application. The different issues of navigation and system architecture in an audio environment without a visual display are discussed. Speech recognition is used to navigate a database of digitally recorded speech segments and synthetic speech is used for control information and user feedback.]

[This paper describes the development in dialog management and natural language generation in the LIMSI RAILTEL system for access to rail travel information. Each dialog is analysed to determine the source of any errors (speech recognition, understanding, information retrieval, processing, or dialog management]


• 6) [Burger 71] “The CONVERSE natural language data management system: current status and plans”

Proceedings of the ACM Symposium on Information Storage and Retrieval, University of Maryland.

[This paper describes a natural-language compiler that accepts sentence in a user-extendable English subset, produces surface and deep-structure syntactic analyses, and uses a network of concepts to construct semantic interpretations formalized as computable procedures. The procedures are evaluated by a data management system that updates, modifies, and searches databases. The system has been designed to handle large vocabularies and large collections of facts efficiently.]


P 27 – 35

• 9) [Dybkjaer, 94] Dybkjaer, H; Dybkjaer, L; Bernsen, N.O, "Database access via spoken language interfaces", Journal: Datalogiske, Skrifter, no 58, p 69 – 79. 1994


[ This paper describes an speech interface to the web whose input are user-independent continuous-speech browser commands in English, and
response is in the form of synthesized speech. This project clearly emphasizes that speech interfaces to the web can be constructed with relatively simple hardware and software tools.]


- **13) [Hendrix78]** “Developing a natural language interface to complex data” Handrix, G. G; Sacerdoti, E. D; Sagalowicz,D; and Slocum,J. ACM TODS, 3, 2, June 1978, 105 – 147.

[This paper describes the aspects of an intelligent interface that provides natural language access to a large data distributed over a computer network. The system architecture consists of natural language processing component, data access component and file access manager. These components acts in tandem to convert the natural language queries into calls to DMBSs at remote sites. To increase the system usability, special language features are also discussed, such as spelling correction, processing of incomplete inputs, and run-time system personalization. A
comparison with other related work is and system’s limitations are also discussed.]

• 14) [Hutchison 95] Malcolm Hutchison (MSC HCI), Natural Language Interfaces to Databases, 1995.


[This paper describes TINA, a natural language system for spoken language applications. In this system, concepts from context free grammars, augmented transition networks, and the unification concepts are combined. To improve recognition performance, a highly constraining probabilistic language model is used. This system provides an automatic sentence generation capability which helps in specifying the overgeneralization problems and also in generating a word-pair language model for a recognizer. Two application domains are created in which the parser is integrated with MIT's SUMMIT recognizer. The parser screening recognizer outputs either at the sentential level or to filter theories during the active search process.]

[This paper describes a front end for natural language access to databases making use of domain-independent semantic information for question interpretation. The approach supports portability. Different modules of the front end are described, and the system’s performance is illustrated by different examples.]


[This is a question-answering system to enable a linear geologist to conveniently access, compare, and evaluate the chemical analysis data on lunar rock and soil composition that is accumulating as a result of the Apollo moon missions. This system is sufficiently natural and complete, hence the task of selecting a wording for a complex request becomes a negligible effort for the geologist user. This goal is successfully achieved in this system and hence, it preceded all the previous systems. This system is the result of 2 years of joint effort by Belt, Berenek, and Norman, and the Language Research Foundation, Cambridge, Mass. ]


- **28)[Yankelovich 95]** Nicole Yankelovich, Gina-Anne Levow, Matt Marx.”Designing SpeechActs: Issues in Speech User Interfaces” Sun Mircosystems Laboratories. Two Elizabeth Drive, Chelmsford, MA,
USA 01824, MIT AI Laboratory, 545 Technology Square, Cambridge, MA 02139, 1995.

[This paper describes SpeechActs, which is an experimental conversational speech system. The paper also examines some of the challenging issues facing speech interface designers and describes approaches to address some of these challenges.]


• 31) [Zhu 94] James Glass, Giovanni Flammia, David Goodine, Michael Phillips, Joseph Polifroni, Shinsuke Sakai, Stephanie Seneff, Victor Zue, “Multilingual spoken-language understanding in the MIT Voyager

[This paper describes the multilingual MIT Voyager spoken language system, which can engage in verbal dialogues with the users about a geographical region within Cambridge, MA in the USA. The system can provide information about distances, travel times or directions, as well as addresses, telephone numbers or location. Voyager has been ported to Japanese, Italian, French and German.]
APPENDIX A: PROGRAMS ON THE CLIENT SIDE

1) File SpeechClient.java

import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.util.*;
import javax.speech.*;
import javax.speech.recognition.*;
import javax.speech.synthesis.*;
import java.util.Locale;
import HTTPget;
import HTTPpost;
import LocInfo;

public class SpeechClient extends Frame {

    /** class variables */
    Synthesizer synth;
    Recognizer recog;
    RuleGrammar rulegram;
    SynthesizerProperties synthProp;
    Voice[] voice = new Voice[2];
    public String webAddress;
    public String cgbinFileName;
    public String grammarFileName;
    public String lastTextSpoken;
    public String delimiterValue = "=";
    public String filename = "1";
    public int counter = 0;
    public String[] filenames = new String[10];
    public String[] webaddresses = new String[10];
    public String[] cgbinfilenames = new String[10];
    public String[] sihlonames = new String[10];
    public boolean flagLocal = false;
public boolean flagFemaleVoice = false;
public boolean flagMaleVoice = false;

/** History object Loc Info Objects are stored */
Vector history = new Vector();
/** GUI variables */
GridBagLayout gridBagLayout = new GridBagLayout();
GridBagConstraints gbCon = new GridBagConstraints();
GridBagConstraints gbConBtnPanel = new GridBagConstraints();
BorderLayout layoutDisplay = new BorderLayout();
/** Layout for button Panel */
GridBagLayout layoutButtonPanel = new GridBagLayout();
/** Text components */
TextField textFieldQuery = new TextField();
TextArea textAreaDisplay = new TextArea("", 5,70,
TextArea.SCROLLBARS_VERTICAL_ONLY);
TextField textFieldStatus = new TextField(15);

/** Label for the frame */
Label title = new Label("Speech Interface");
/** Panels used */
Panel panelStatus = new Panel();
Panel panelButton = new Panel();
Panel panelDisplay = new Panel();
/** Buttons used */
Button buttonConnect = new Button();
Button buttonStop = new Button();
Button buttonPause = new Button();
Button buttonResume = new Button();

/** constructor for speech client */
public SpeechClient(){
try{
/** initialise GUI */
initGUI();
/** create synthesizer */
createSynth();
}catch (Exception e){
e.printStackTrace();
System.err.println("Error in the constructor");}
for (int i = 0; i <= 9; i++)
{
  filenames[i] = new String();
  webaddresses[i] = new String();
  cgibinfilenames[i] = new String();
  sihlonames[i] = new String();
}

/** methodinitializes the GUI */
private void initGUI() throws Exception{
  /** add the layout */
  this.setSize(new Dimension(550,450));
  this.setBackground(new Color(190, 190, 190));
  this.setForeground(new Color(0, 0, 0));
  this.setLayout(gridBagLayout);
  /** add layout for the panels */
  panelDisplay.setLayout(layoutDisplay);
  panelButton.setLayout(layoutButtonPanel);
  textFieldQuery.addActionListener(new SpeechClient_textFieldQuery_actionAdapter(this));

  buttonConnect.setLabel( "connect" );
  buttonConnect.addActionListener(new SpeechClient_buttonConnect_actionAdapter(this));

  buttonStop.setLabel ( "Stop Speech" );
  buttonStop.addActionListener(new SpeechClient_buttonStop_actionAdapter(this));

  buttonPause.setLabel ( "Pause Speech" );
  buttonPause.addActionListener(new SpeechClient_buttonPause_actionAdapter(this));

  buttonResume.setLabel ( "Resume Speech" );
  buttonResume.addActionListener(new SpeechClient_buttonResume_actionAdapter(this));

  /** constraints for display panel */
gbCon.fill = GridBagConstraints.HORIZONTAL;
gbCon.insets = new Insets(2, 2, 2, 2);
gbCon.gridx = 0; gbCon.gridy = 0; gbCon.gridwidth = 0; gbCon.gridheight = 1;
gbCon.weightx = gbCon.weighty = 0.0;
this.add(title, gbCon);
gbCon.fill = GridBagConstraints.BOTH;
gbCon.gridx = 0; gbCon.gridy = 1; gbCon.gridwidth = 2; gbCon.gridheight = 4;
gbCon.weightx = gbCon.weighty = 1.0;
this.add(panelDisplay, gbCon);
gbCon.fill = GridBagConstraints.VERTICAL;
gbCon.gridx = 3; gbCon.gridy = 1; gbCon.gridwidth = 2; gbCon.gridheight = 4;
gbCon.weightx = gbCon.weighty = 0.0;
this.add(panelButton);
gbCon.fill = GridBagConstraints.HORIZONTAL;
gbCon.gridx = 0; gbCon.gridy = 6; gbCon.gridwidth = 0; gbCon.gridheight = 1;
gbCon.weightx = gbCon.weighty = 0.0;
this.add(panelStatus, gbCon);
panelStatus.add(textFieldStatus);

/** buttonpanel */
gbConBtnPanel.fill = GridBagConstraints.BOTH;
gbConBtnPanel.insets = new Insets(1, 1, 1, 1);
gbConBtnPanel.gridx = 0; gbConBtnPanel.gridy = 2;
gbConBtnPanel.gridwidth = 2; gbConBtnPanel.gridheight = 1;
gbConBtnPanel.weightx = gbConBtnPanel.weighty = 0.0;
panelButton.add(buttonConnect, gbConBtnPanel);

gbConBtnPanel.gridx = 0; gbConBtnPanel.gridy = 3;
gbConBtnPanel.gridwidth = 2; gbConBtnPanel.gridheight = 1;
gbConBtnPanel.weightx = gbConBtnPanel.weighty = 0.0;
panelButton.add(buttonStop, gbConBtnPanel);

gbConBtnPanel.gridx = 0; gbConBtnPanel.gridy = 4;
gbConBtnPanel.gridwidth = 2; gbConBtnPanel.gridheight = 1;
gbConBtnPanel.weightx = gbConBtnPanel.weighty = 0.0;
panelButton.add(buttonPause, gbConBtnPanel);

gbConBtnPanel.gridx = 0; gbConBtnPanel.gridy = 5;
gbConBtnPanel.gridwidth = 2; gbConBtnPanel.gridheight = 1;
gbConBtnPanel.weightx = gbConBtnPanel.weighty = 0.0;
panelButton.add( buttonResume , gbConBtnPanel);

    /*** add the text area and the text field */
panelDisplay.add(textAreaDisplay);
panelDisplay.add(textFieldQuery,"South");
}

/** create synthesizer method */
void createSynth(){
    try{
        /** create voice object for male and female voice types */
         voice[0] = new Voice();
         voice[0].setGender(Voice.GENDER_MALE);
         voice[0].setAge(Voice.AGE_MIDDLE_ADULT);
SynthesizerModeDesc synthMode = new SynthesizerModeDesc();
synthMode.setLocale(Locale.ENGLISH);
/** create synth for english */
synth = Central.createSynthesizer( synthMode );
synthProp = synth.getSynthesizerProperties();
synthProp.setVoice(voice[0]);
/** make it ready to speak */
synth.allocate();
synth.resume();
}catch (Exception e){
    e.printStackTrace();
}
}

/** method to get the intial information */
void getintMessage(){
    /** get the initial introduction */
    HTTPget get = new HTTPget ("www.cs.uwindsor.ca", 80);
    get.submit("/cgi-bin/iqures1/h.sh", "welcome.m");
String resultintmess = get.result();
System.out.println (resultintmess );
try {
void startString()
{
    // Call this method when ever a grammar file is loaded
    String question = "question=" + "startstring".trim();
    System.out.println(question);
    HTTPpost postRequest = new HTTPpost(webAddress, cgibinFile, question);
    String result = postRequest.getResults();
    System.out.println(result);
    textAreaDisplay.appendText(result);
    textAreaDisplay.appendText("\n");
    speak(result);
}

/* Method to read local grammar */
void readLocalGrammar(Recognizer recog)
{
    try
    {
        Reader reader = new FileReader(grammarFile);
        File inputFile = new File(grammarFile);
        FileReader in = new FileReader(inputFile);
        rulegram = recog.loadJSGF(reader);
        rulegram.setEnabled(true);

        // Create info object and add to history
        webAddress = webaddresses[Integer.parseInt(filename) - 1];
        cgibinFile = cgibinfilenames[Integer.parseInt(filename) - 1];
        String grammarFile = filenames[Integer.parseInt(filename) - 1];
        grammarFile = grammarFile;
        System.out.println(grammarFile + " loaded");
        LocInfo loc = new LocInfo(webAddress, cgibinFile, grammarFile);
        history.addElement(loc);
        reader.close();
    } catch (Exception readlocal) {
        readlocal.printStackTrace();
        System.err.println("Exception in read local grammar");
    }
}

    synth.speak(result, null);
    } catch (Exception reinit) {
        reinit.printStackTrace();
    }
}
/** method to read remote grammar */
void readRemoteGrammar( Recognizer recog) {
    try {
        HTTPget get = new HTTPget( "www.cs.uwindsor.ca", 80 );
        get.submit("/cgi-bin/iquresl/h.sh", grammarFileName);
        System.out.println(grammarFileName);
        System.out.println( get.result() );
        StringReader strReaderGrammar = new StringReader( get.result() );
        rulegram = recog.loadJSGF( strReaderGrammar );

        if (filename == "1")
        {
            speak("Welcome to Speech Web start sihlo. Please say help for assistance");
        }
        filenames[Integer.parseInt(filename) - 1 ] = grammarFileName;
        webaddresses[Integer.parseInt(filename) - 1 ] = webAddress;
        cgibinfilenames[Integer.parseInt(filename) - 1 ] = cgibinFileName;
        String file = filename + ".gram";
        FileWriter fw = new FileWriter(new File(file));
        fw.write( get.result() );
        fw.close();
        rulegram.setEnabled(true);
        recog.commitChanges();
        /** create info object and add to history */
        LocInfo loc = new LocInfo (webAddress, cgibinFileName,grammarFileName);
        history.addElement(loc);
    } catch(Exception gram) {
        System.err.println("Grammarer file exception unable to get from URL");
        gram.printStackTrace();
    }
}

/** method to initialise recogniser */
void initRecogniser() {
    try {
        /** create a recognizer */
recog = Central.createRecognizer(
    new EngineModeDesc(Locale.ENGLISH)
);
recog.allocate();

/** load the default grammar */
    if (flagLocal)
        readLocalGrammar(recog);
    else
        readRemoteGrammar(recog);
    }
    recog.addResultListener( new SpeechListener( this));
    RecognizerAudioListener audioListn = new AudioRecoListener(this);
    recog.getAudioManager().addAudioListener( audioListn );
    recog.commitChanges();
    recog.requestFocus();
    recog.resume();
}catch(Exception initRecog){
    initRecog.printStackTrace();
}
}

/** generic method to get the response */
void answerQuestion ( ResultToken tokens[] ,StringBuffer textSpoken ){
    for ( int i = 0; i< tokens.length ; i ++ ){
        textSpoken.append( tokens[i].getSpokenText() + " ");
        System.out.print(tokens[i].getSpokenText()+" ");
    }

textFieldQuery.setText("'");
textFieldQuery.setText(textSpoken.toString());
lastTextSpoken = textSpoken.toString();
String question = "question=" + textFieldQuery.getText().trim();
System.out.println(question);
System.out.println(webAddress +" "+ cgibinFileName+" "+ question);
HTTPRequest postRequest = new
HTTPRequest(webAddress,cgibinFileName,question);
String result = postRequest.getResults();
System.out.println ( result );
textAreaDisply.appendText(result);
public String valueString(String token, String delimiter) {
    StringTokenizer tokensNameValue = new StringTokenizer(token, delimiter);
    String Name = tokensNameValue.nextToken();
    String Value = tokensNameValue.nextToken();
    return Value;
}

public String checkGrammarChange(String result) {
    /* tokenise the string based on "=" */

    StringBuffer result1 = new StringBuffer();
    StringBuffer vType = new StringBuffer();

    if (result.startsWith("STACKREMOVE=")) {
        String name = "";
        /* tokenise the string and the appropriate variables */
        StringTokenizer tokens = new StringTokenizer(result, ",;");
        while (tokens.hasMoreTokens()) {
            result1.append(" you are now in ");
            filename = valueString(tokens.nextToken(), delimiterValue);
            webAddress = valueString(tokens.nextToken(), delimiterValue);
        }
    }
}
cgibinFileName = valueString(tokens.nextToken(), delimiterValue) + 
"/"+ 
( grammarFileName=valueString(tokens.nextToken(), delimiterValue) 
)+".so";
nname = grammarFileName;
silhlonames[Integer.parseInt(filename) - 1] = name;
grgrammarFileName = grammarFileName+".gram";
ptype.append(valueString(tokens.nextToken(), delimiterValue));
}
/** load new grammar, check if the recogniser has a grammar attached to it
 * yes delete the grammar and add the new grammar make a cgi-bin
 * call */

try{
    RuleGrammar[] delGrammar = recog.listRuleGrammars();
    for (int j = 0; j < delGrammar.length; j++) {
        recog.deleteRuleGrammar(delGrammar[j]);
        recog.commitChanges();
    }
    readRemoteGrammar(recog);
    recog.commitChanges();
    result1.append(name);
    result1.append(" department.");
} catch( Exception delGram) {
    System.err.println("Exception deleteing Grammar");
    delGram.printStackTrace();
}

if (vType.toString().equalsIgnoreCase("female")) {
    flagFemaleVoice = true;
    flagMaleVoice = false;
} else {
    flagFemaleVoice = false;
    flagMaleVoice = true;
}
return result1.toString();

else
if ( result.startsWith("STACK="))
{
    Stringtokenizer tokens = new Stringtokenizer(result,";");
    while(tokens.hasMoreTokens()) {

        result1.append("You are now in ");

        filename = valueString( tokens.nextToken(), delimiterValue );
        grammarFileName = filename + ".gram";
        vType.append("female");
    }
    /** load new grammar,check if the recogniser has a grammar attached to it
     * yes delete the grammar and add the new grammar. */
    try{
        RuleGrammar[] delGrammar = recog.listRuleGrammars();
        for (int j = 0; j < delGrammar.length;j++){
            recog.deleteRuleGrammar( delGrammar[j] );
            recog.commitChanges();
        }
        readLocalGrammar(recog);
        recog.commitChanges();
        String name = sihlonames[Integer.parseInt(filename) - 1];
        result1.append(name);
        result1.append(" sihlo");
    }catch( Exception delGram ){
        System.err.println( "Exception deleteing Grammar");
        delGram.printStackTrace();
    }

    return result1.toString();
}

return result;
}

void textFieldQuery_actionPerformed(ActionEvent e){

}
String question = "question=" + textFieldQuery.getText().trim();
HTTPpost postRequest = new
    HTTPpost(webAddress, cgibinFileName, question);
String result = postRequest.getResults();
System.out.println(result);
/** append the text Area */
textAreaDisplay.appendText(result);
speak(result);
}

void buttonConnect_actionPerformed(ActionEvent e) {
    DataInputFrame getInfo = new DataInputFrame(this);
    Dimension getInfoSize = getInfo.getPreferredSize();
    Dimension frmSize = getSize();
    Point loc = getLocation();
    getInfo.setLocation((frmSize.width - getInfoSize.width) / 2 + loc.x,
                        (frmSize.height - getInfoSize.height) / 2 + loc.y);
    getInfo.show();
    /*System.out.println (webAddress);
    System.out.println(cgibinFileName);
    System.out.println(grammarFileName); to be deleted */
    buttonConnect.setEnabled(false);
}

void buttonStop_actionPerformed(ActionEvent e) {
    System.out.println("Inside button stop action performed");
    try {
        synth.deallocate();
        synth.allocate();
    } catch (Exception dealloc) {
        dealloc.printStackTrace();
    }
}

void buttonPause_actionPerformed(ActionEvent e) {
    System.out.println("Inside button Pause action performed");
    synth.pause();
}

void buttonResume_actionPerformed(ActionEvent e) {

System.out.println("Inside button Resume action performed");
try{
    synth.resume();
}catch (Exception exRes){
    exRes.printStackTrace();
}

void resultSpeech(ResultEvent e)
{
    FinalRuleResult resultSentence = (FinalRuleResult)( e.getSource());
    String tags[] = resultSentence.getTags();
    ResultToken tokens[] = resultSentence.getBestTokens();
    StringBuffer textSpoken = new StringBuffer();
    if (tags[0].equals("sentence")) {
        for (i = 0; i < tokens.length ; i++) {
            textSpoken.append((tokens[i].getSpokenText() + " ");
            System.out.print(tokens[i].getSpokenText()+" ");
        }
    textFieldQuery.setText(" ");
    textFieldQuery.setText(textSpoken.toString());
    astTextSpoken = textSpoken.toString();/** this is for echo command */
    String question = "question=" + textFieldQuery.getText().trim();
    HTTPpost postRequest = new HTTPpost(webAddress,cgibinFileName,question);
    String result = postRequest.getResults();
    System.out.println( result );
    /** pass the result to string processing check for any change in grammar file */
    String result1 = checkGrammarChange( result );
    textAreaDisplay.appendText(result1);
    textAreaDisplay.appendText("\n");
    speak(result1);
    /** change the voice if the flag is set true */
    if ( flagFemaleVoice ) {
        synthProp = synth.getSynthesizerProperties();
        try{
            voice[1] = new Voice();
            voice[1].setGender(Voice.GENDER_FEMALE);
            voice[1].setAge(Voice.AGE_DONT CARE);
synthProp.setVoice(voice[1]);
} catch (Exception femVoice) {
    System.err.println("Set Female voice failed ");
}
} else if (flagMaleVoice) {
    try {
        Voice maleVoice = new Voice();
        maleVoice.setGender(Voice.GENDER_MALE);
        maleVoice.setAge(Voice.AGE_MIDDLE_ADULT);
        synthProp.setVoice(maleVoice);
    } catch (Exception femVoice) {
        System.err.println("Set Female voice failed ");
    }
}
} else if (tags[0].equals("abort")) {
    try {
        synth.deallocate();
        synth.allocate();
    } catch (Exception dealloc) {
        dealloc.printStackTrace();
    }
} else if (tags[0].equals("pause")) {
    synth.pause();
} else if (tags[0].equals("resume")) {
    try {
        synth.resume();
    } catch (Exception resume) {
        resume.printStackTrace();
    }
} else if (tags[0].equals("bye")) {

    String s = "Thanks for coming to the speechWeb. Please come again. Have a nice day. " ;

    /** set output to text area */
    textFieldDisplay.appendText(s);
    /** set output to the speaker */
speak(s);
try{
    synth.waitForEngineState(Synthesizer.QUEUE_EMPTY);
synth.deallocate();
} catch (Exception ealldelloc) {
    System.err.println("ERR IN DEALLOCATING");
}
System.exit(0);

} else if (tags[0].equals("back")) {
    counter = 0;
}

if(Integer.parseInt(filename) > 1) {
    filename = String.valueOf(Integer.parseInt(filename) - 1);
    webAddress = webaddresses[Integer.parseInt(filename)-1];
    cgibinFileName = cgibinfilenames[Integer.parseInt(filename) - 1];
    grammarFileName = filename + ".gram";
    flagFemaleVoice = true;

    /** load new grammar, check if the recogniser has a grammar attached to it
    * yes delete the grammar and add the new grammar.  */
    try{
        RuleGrammar[] delGrammar = recog.listRuleGrammars();
        for (int j = 0; j < delGrammar.length;j++) {
            recog.deleteRuleGrammar( delGrammar[jj]);
            recog.commitChanges();
        }
        readLocalGrammar(recog);
        recog.commitChanges();
        grammarFileName = filenames[Integer.parseInt(filename) - 1];
        String name = silhonomes[Integer.parseInt(filename) - 1];
        String Result = "You are now in the " + name + " silho. Please say help for assistance";
    }
speak(Result);
} catch (Exception delGram) {
    System.err.println( "Exception deleteing Grammar");
    delGram.printStackTrace();
}
}

if(Integer.parseInt(filename) == 1) {

    filename = "1";
    String rejResult = " Already at home can not go back. Please say help ";
    speak(rejResult);
}

} else if (tags[0].equals("home")) {

    counter = 0;
    filename = "1";
    webAddress = webaddresses[Integer.parseInt(filename)-1];
    cgibinFileName = cgibinfileNames[Integer.parseInt(filename) - 1];
    grammarFileName = filename + ".gram";
    flagFemaleVoice = true;

    /** load new grammar,check if the recogniser has a grammar attached to it
     * yes delete the grammar and add the new grammar. */
    try {
        RuleGrammar[] delGrammar = recog.listRuleGrammars();
        for (int j = 0; j < delGrammar.length; j++) {
            recog.deleteRuleGrammar( delGrammar[j]);
            recog.commitChanges();
        }
        readLocalGrammar(recog);
        recog.commitChanges();
    }
String Result = "You are now in the main sihlo. Please say help for assistance";
        speak(Result);
    } catch( Exception delGram){
        System.err.println( "Exception deleteing Grammar");
        delGram.printStackTrace();
    }

else if (tags[0].equals("stay"))
{
    counter = 0;
    webAddress = webaddresses[Integer.parseInt(filename)-1];
    cgibinFileName = cgibinfilenames[Integer.parseInt(filename) - 1];
    grammarFileName = filename + ".gram";
    flagFemaleVoice = true;

    
    /** load new grammar, check if the recogniser has a grammar attached to it
     * yes delete the grammar and add the new grammar. */
    try{
        RuleGrammar[] delGrammar = recog.listRuleGrammars();
        for (int j = 0; j < delGrammar.length;j++){
            recog.deleteRuleGrammar( delGrammar[j]);
            recog.commitChanges();
        }
        readLocalGrammar(recog);
        recog.commitChanges();
        grammarFileName = filenames[Integer.parseInt(filename) - 1];
        String name = siholenames[Integer.parseInt(filename) - 1];
        String Result2 = "You are now at the " + name + " sihlo. Please say help for assistance.";
        speak(Result2);
    } catch( Exception delGram){
        System.err.println( "Exception deleteing Grammar");
        delGram.printStackTrace();
    }
}

else if (tags[0].equals("where"))
{  
counter = 0;
flagFemaleVoice = true;
grammarFileName = filenames[Integer.parseInt(filename) - 1];
String name =  sihilationnames[Integer.parseInt(filename) - 1];
String Result2 = "You are now at the "+ name + " sihilation. Please say help for assistance.";
speak(Result2);
}
else if (tags[0].equals("help"))
{
counter = 0;
if((Integer.parseInt(filename) - 1) > 0 )
{

    String Result1 = "You are now at the "+ sihilationnames[Integer.parseInt(filename) - 1] + " sihilation."
    Result1 += "You have these options. If you want to stay in this sihilation, please say, stay.";
    Result1 += " If you want to go to the " + sihilationnames[Integer.parseInt(filename) - 2] + " sihilation, please
        say, go back.";
    Result1 += " If you want to go home, please say, home. ";
    Result1 += " If you want to ask where you are, say, where am i";
    Result1 += " If you want to end the application, please say, bye.";
speak(Result1);
}
else
{
    filename = "1";
    webAddress = webaddresses[Integer.parseInt(filename)-1];
cgibinFileName = cgibinfilenames[Integer.parseInt(filename) - 1];
grammarFileName = filename + ".gram";
flagFemaleVoice = true;

    /** load new grammar,check if the recogniser has a grammar attached to it
* yes delete the grammar and add the new grammar.  */
    try{
        RuleGrammar[] delGrammar = recog.listRuleGrammars();

    }catch (IOException e) {
    
        System.out.println("ERROR: IOException caught while trying to load new grammar.");
    
    }catch (Exception e) {
    
        System.out.println("ERROR: Exception caught while trying to load new grammar.");
    
    }
}
for (int j = 0; j < delGrammar.length; j++) {
    recog.deleteRuleGrammar(delGrammar[j]);
    recog.commitChanges();
}
readLocalGrammar(recog);
recog.commitChanges();
String Result = "You are now in the start sihlo. Please say help for assistance";
speak(Result);
} catch (Exception delGram) {
    System.err.println("Exception deleting Grammar");
    delGram.printStackTrace();
}
}

void speak ( String s ) {
    if (synth != null) {
        try {
            recog.pause();
            synth.speak( s, null);
            recog.resume();
        } catch (Exception speakExep) {
            speakExep.printStackTrace();
        }
    } else
        System.out.println(s);
}

void resultRejectedSpeech(RESULTEvent e) {
    if (counter < 3)
    {
        String Result = "Please repeat I do not understand." ; 
        speak(Result);
        counter++;
    }
else {
    counter = 0;
    grammarFileName = "navigation.gram";
    flagFemaleVoice = true;

    String Result3 = "I am sorry I could not understand. Please say, help for assistance."
    speak(Result3);

    /** load new grammar, check if the recogniser has a grammar attached to it
     * yes delete the grammar and add the new grammar. */
    try{
        RuleGrammar[] delGrammar = recog.listRuleGrammars();
        for (int j = 0; j < delGrammar.length; j++) {
            recog.deleteRuleGrammar(delGrammar[j]);
            recog.commitChanges();
        }
    }
    try{
        HTTPget get = new HTTPget("www.cs.uwindsor.ca", 80);
        get.submit("/cgi-bin/iqures1/h.sh", "navigation.gram");
        System.out.println(grammarFileName);
        System.out.println(get.result());
        StringReader strReaderGrammar = new StringReader(get.result());
        rulegram = recog.loadJSGF(strReaderGrammar);
        rulegram.setEnabled(true);
        recog.commitChanges();
        /** create info object and add to history */
        history.addElement(loc); /*
    } catch(Exception gram){
        System.err.println("Grammar file exception unable to get from URL");
        gram.printStackTrace();
    }
    recog.commitChanges();
}
catch( Exception delGram){
System.err.println("Exception deleting Grammar");
delGram.printStackTrace();
}
}

void audioDisplay (RecognizerAudioEvent e ){
    textFieldStatus.setText(" ");
    textFieldStatus.setText("Volume:" + e.getAudioLevel());
}
}// end class speech client

/** Bunch of Inner classes to satisfy the action events */

class SpeechClient_textFieldQuery_actionAdapter implements java.awt.event.ActionListener{
    SpeechClient adaptee;
    SpeechClient_textFieldQuery_actionAdapter(SpeechClient adaptee){
        this.adaptee = adaptee;
    }
    public void actionPerformed(ActionEvent e) {
        adaptee.textFieldQuery_actionPerformed(e);
    }
}

class SpeechListener extends ResultAdapter{
    SpeechClient adaptee;
    SpeechListener( SpeechClient adaptee ){
        this.adaptee = adaptee;
    }
    public void resultAccepted( ResultEvent e){
        adaptee.resultSpeech(e);
    }
    public void resultRejected( ResultEvent e){
        adaptee.resultRejectedSpeech(e);
    }
}//end class speech listener
class AudioRecoListener extends RecognizerAudioAdapter{
    SpeechClient adaptee;
    AudioRecoListener (SpeechClient adaptee){
        this.adaptee = adaptee;
    }
    public void audioLevel( RecognizerAudioEvent e ){
        adaptee.audioDisplay ( e );
    }
}// end class audio listener

class SpeechClient_buttonConnect_actionAdapter implements java.awt.event.ActionListener{
    SpeechClient adaptee;
    SpeechClient_buttonConnect_actionAdapter(SpeechClient adaptee){
        this.adaptee = adaptee;
    }
    public void actionPerformed(ActionEvent e) {
        adaptee.buttonConnect_actionPerformed(e);
    }
}

class SpeechClient_buttonStop_actionAdapter implements java.awt.event.ActionListener{
    SpeechClient adaptee;
    SpeechClient_buttonStop_actionAdapter(SpeechClient adaptee) {
        this.adaptee = adaptee;
    }
    public void actionPerformed(ActionEvent e) {
        adaptee.buttonStop_actionPerformed(e);
    }
}

class SpeechClient_buttonPause_actionAdapter implements java.awt.event.ActionListener{
    SpeechClient adaptee;
    SpeechClient_buttonPause_actionAdapter(SpeechClient adaptee) {
        this.adaptee = adaptee;
    }
    public void actionPerformed(ActionEvent e) {
        adaptee.buttonPause_actionPerformed(e);
    }
}
class SpeechClient_buttonResume_actionAdapter implements java.awt.event.ActionListener{
    SpeechClient adaptee;
    SpeechClient_buttonResume_actionAdapter(SpeechClient adaptee) {
        this.adaptee = adaptee;
    }
    public void actionPerformed(ActionEvent e){
        adaptee.buttonResume_actionPerformed(e);
    }
}

class DataInputFrame extends Frame implements java.awt.event.ActionListener{
    SpeechClient adaptee;
    Panel displayPanel = new Panel();
    Button okButton = new Button();
    Button cancelButton = new Button();
    Choice locationChooser = new Choice();
    Label labelWebAddress = new Label("WebAddress");
    Label labelCgi_bin = new Label("ExecutableFile");
    Label labelGrammarFile = new Label("GrammarFile");
    Label labelFileLocation = new Label("GrammarFileLocation");
    TextField txtWebAddress = new TextField(25);
    TextField txtCgi_bin = new TextField(25);
    TextField txtGrammarFile = new TextField(25);
    GridLayout gridlayoutDIF = new GridLayout();

    /** constructor */
    DataInputFrame (SpeechClient adaptee) {
        try{
            this.adaptee = adaptee;
            init();
        }catch (Exception e){
            e.printStackTrace();
        }
    }
    pack();
}
private void init() {
    this.setTitle("Information of Remote Executable ");

    gridlayoutDIF.setRows(6);
    gridlayoutDIF.setColumns(2);
    displayPanel.setLayout(gridlayoutDIF);

    okButton.setLabel("OK");
    okButton.addActionListener(this);
    cancelButton.setLabel("Cancel");
    cancelButton.addActionListener(this);
    this.add(displayPanel, null);
    displayPanel.add( labelWebAddress );
    displayPanel.add( txtWebAddress );
    txtWebAddress.setText("www.cs.uwindsor.ca");
    displayPanel.add( labelCgi_bin );
    displayPanel.add( txtCgi_bin );
    txtCgi_bin.setText("cgi-bin/iqures1/welcome.so");

    displayPanel.add( labelGrammarFile );
    displayPanel.add( txtGrammarFile );
    txtGrammarFile.setText("welcome");

    displayPanel.add( labelFileLocation );

    locationChooser.add("Local");
    locationChooser.add("Remote");
    displayPanel.add( locationChooser );

    displayPanel.add( okButton );
    displayPanel.add( cancelButton );
    pack();
}

public void actionPerformed(ActionEvent e) {
    if ( e.getSource() == okButton ){
/** process the information obtained */
adaptee.webAddress = txtWebAddress.getText().trim();
adaptee.cgiBinFileName = txtCgi_bin.getText().trim();
adaptee.grammarFileName = txtGrammarFile.getText().trim()+".gram";

if( locationChooser.getSelectedItem().equalsIgnoreCase("Local")){
    adaptee.flagLocal = true ;
}
adaptee.initRecogniser();
setVisible(false);
dispose();
}
if ( e.getSource() == cancelButton ){
/** process the information obtained */
System.out.println("CANCEL BUTTON CLICKED");
setVisible(false);
dispose();
}
2) File HTTPget.java

```java
import java.net.*;
import java.io.*;
import java.io.IOException;

public class HTTPget {

    private String contentType = "Text/plain"; // by default it is plain
    private String receiveData = " ",
    private String urlName;
    private int port;

    public StringBuffer storageBuf = new StringBuffer();

    public HTTPget( String urlBase, int urlPort ){
        urlName = urlBase;
        port = urlPort;
        if (port == -1) port = 80;
    }

    public void submit (String scriptName , String queryData ){
        receiveData = " ";
        /** create a get query */
        String urlLine = scriptName+"?";
        try {
            String url = urlName + urlLine;
            URL u = new URL("http://" + url + queryData);
            DataInputStream line = new DataInputStream (u.openStream());
            while ( ( receiveData = line.readLine() ) != null){
                storageBuf.append(receiveData);
            }
            line.close();
        } catch (Exception et) {
            et.printStackTrace();
        }
    }

    public String result(){
return (storageBuf.toString());
}

3) File HTTPpost.java

import java.net.URL;
import java.net.URLConnection;
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.DataInputStream;
import java.io.IOException;
public class HTTPpost {

private URL urlToGo;
private String urlHostName;
private String urlFileName;
private String query;
private String receiveData;
public StringBuffer storageBuf = new StringBuffer();
public HTTPpost(String hostName, String fileName, String query) {
    try {
        urlToGo = new URL("http", hostName, fileName);
        URLConnection uc = urlToGo.openConnection();
        uc.setDoOutput(true);
        uc.setDoInput(true);
        uc.setAllowUserInteraction(false);
        uc.setRequestProperty("Content-type", "text/plain");
        uc.setRequestProperty("Content-length", query.length() + "\n");

        DataOutputStream dos = new DataOutputStream(uc.getOutputStream());
        System.out.println("INSIDE QUERY HTTP POST " + query);
        dos.writeBytes(query);
    }

}
dos.close();
    // read the response
    DataInputStream dis = new DataInputStream(uc.getInputStream());
    String nextline;
    while ( (nextline = dis.readLine()) != null) {
        receiveData = nextline;
    }
    dis.close();
} catch (Exception be){
    System.err.println(be);
}

public String getResults (){
    return (receiveData) ;
}
4) File MC.java

import java.awt.*;
import SpeechClient;

public class MC {
    boolean packFrame = false;
    public MC(){
        SpeechClient frame = new SpeechClient();
        if (packFrame)
            frame.pack();
        else
            frame.validate();
        // center the window
        Dimension screenSize =
            Toolkit.getDefaultToolkit().getScreenSize();
        Dimension frameSize = frame.getSize();
        if ( frameSize.height > screenSize.height )
            frameSize.height = screenSize.height;
        if ( frameSize.width > screenSize.width )
            frameSize.width = screenSize.width;
        frame.setLocation( ( screenSize.width - frameSize.width
            )/ 2,
            ( screenSize.height - frameSize.height)/ 2);
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        MC mC = new MC();
    }
}
APPENDIX B: PROGRAMS ON THE SERVER SIDE

1) C Program That Wraps Miranda Code

#include <errno.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include "preferences.h"
#include "printer.h"

char *makeword(char *line, char stop);
char *fmakeword(FILE *f, char stop, int *len);
void unescape_url(char *url);
char x2c(char *what);
void plastospace(char *str);

char *makeword(char *line, char stop) {
    int x = 0, y;
    char *word = (char *) malloc(sizeof(char) * (strlen(line) + 1));

    for (x = 0; (line[x] && (line[x] != stop)); x++)
        word[x] = line[x];

    word[x] = '\0';
    if (line[x]) ++x;
    y = 0;

    while (line[y++] = line[x++]);
    return word;
}
void unescape_url(char *url) {
    register int x, y;

    for (x = 0, y = 0; url[y]; ++x, ++y) {
        if ((url[x] = url[y]) == '%' ) {
            url[x] = x2c(&url[y+1]);
            y += 2;
        }
    }
    url[x] = '\0';
}

char x2c(char *what) {
    register char digit;

    digit = (what[0] >= 'A' ? ((what[0] & 0xdf) - 'A') + 10 : (what[0] - '0'));
    digit *= 16;
    return (digit);
}

void plastospace(char *str) {
    register int x;

    for (x = 0; str[x]; x++)
        if (str[x] == '+') str[x] = ' ';
the above function to be put in a .h file

/* convert upper-to-lower and remove all unnecessary characters such as
tabs, LF, CR, \n ...
return modified string as a result */
char *process_question(char *string)
{
    char *question;
    int i, ind;

    question = (char *) malloc(strlen(string)+1)*sizeof(char));
    /* skip leading spaces */
    for (i = 0; *(string + i) != '\0' &&
            ((*(string + i) == ' ') || (*(string + i) == 't') ||
             (*(string + i) == 10) || (*(string + i) == '\n') ||
             (*(string + i) == 13)); i++);
    /* remove new_lines, tabs, convert upper-to-lower */
    for (ind = 0, i = i; *(string + i) != '\0'; i++, ind++)
    {
        if (((*(string + i) == 't') || (*(string + i) == 10) ||
            (*(string + i) == '\n') || (*(string + i) == 13))
            *(question + ind) = ' ';
        else
            if ((*(string + i) <= 90) && (*(string + i) >= 65))
                *(question + ind) = *(string + i) + 32;
            else
                *(question + ind) = *(string + i);
    }
    *(question + ind) = '\0';

    return question;
}
main()
{
    int fifo[2], proc;

    char *argv[4], quest[300], *name, *ques, *questmira, answer[5000],
    tmp[500], tmp1[500], tmp2[500], tmp3[500];

    fgets(quest, 300, stdin);
    plustospace(quest);
    unescape_url(quest);

    name = makeword(quest, '=');
    questmira = process_question(quest);

    if (strcmp(questmira, "")) {

        argv[0] = malloc(10*sizeof(char));
        strcpy(argv[0], "welcome");
        argv[1] = malloc(strlen(questmira)*sizeof(char));
        strcpy(argv[1], questmira);
        argv[2] = "/quit";
        argv[3] = NULL;

        /* env settings needed to start Miranda interpreter */

        sprintf(tmp1, "HOME=%s", MIRA_PREFIX);
        putenv(tmp1);
        sprintf(tmp2, "MIRALIB=%s", MIRA_LIB);
        putenv(tmp2);
        sprintf(tmp3, "PATH=%s:%s", MIRA_BIN, MIRA_PREFIX);
        putenv(tmp3);

        printf("Content-type: text/plain\n\n"]; /* printf(questmira); */
pipe(fifo);

if ((proc = fork()) == -1)
   {printf("error: Can't fork");
    exit(errno);
   }

if (proc == 0) /* child */
   {
    close(1);
    dup(fifo[1]);
    sprintf(tmp, "%s/welcome", MIRA_PREFIX);

    execvp(tmp, argv);

    printf("error: Can't exec");
    exit(errno);
   }

/* Parent */
close(0);
dup(fifo[0]);

fgets(answer, 5000, stdin);

}
else{
    printf("\n bad question\n");
}

print_answer(answer);

}
2) Preferences.h

/* Change these settings if necessary */

/* url prefix of cgi-programs */
#define EXEC_PREFIX "/cgi-bin/iquresl"

/* url prefix of HTML documents */
#define DOC_PREFIX "/users/i/iquresl/miranda"

/* Miranda home directory -- this directory should contain .mirarc file as well as all Miranda programs. The file .mirarc should contain the following (or similar) line:

hdve 100000 24000 2020 vi +!

The main purpose of this line is to increase the heap to 100000 (default on janus is 20000) */
#define MIRA_PREFIX "/opt/local/WWW/Servers/NCSA/cgi-bin/iquresl/mira_data"

/* directory where Miranda is installed */
#define MIRA_BIN "/opt/local/mira/bin"

/* directory where Miranda's stdlib is located */
#define MIRA_LIB "/opt/local/mira/lib"
3) Unix Shell Script (File h.sh)

#!/bin/csh -f
#
echo "Content-Type: text/plain"
echo ""

#echo "<b>REQUEST_METHOD</b> = &quot;$REQUEST_METHOD&quot; <br>"
if ( $REQUEST_METHOD == "GET" ) then
  # echo "<b>QUERY_STRING</b> = &quot;$QUERY_STRING&quot; <br>"
  # echo " This is query string: " $QUERY_STRING
  # echo " The content length " $CONTENT_LENGTH
  # echo " 
  # echo " 
  # echo " >> cat $QUERY_STRING
  # echo " 
  # echo " 
  # echo "This test was a success This is a test database which you can query "
else if ( $REQUEST_METHOD == "POST" ) then
  echo "<b>CONTENT_LENGTH</b> = &quot;$CONTENT_LENGTH&quot; <br>"
  setenv v `/bin/cat`
echo $v
/opt/local/mira/bin/mira /home/cs/misc/www/cgi-bin/iquires1/mira_data/welcome.m
<<zzz (interpret "$v")
/q
zzz

endif
APPENDIX C:

SAMPLE GRAMMAR AND LANGUAGE FILES

A) Grammar Files

1) Navigation Grammar

grammar navigation;

public <where> = where am I {where};

public <help> = help {help};

public <back> = go back {back};

public <stay> = stay {stay};

public <bye> = bye {bye};

public <home> = home {home};
2) Welcome Grammar

grammar welcome;

public <s> = <simple> {sentence};

public <bye> = bye {bye};

<simple> = | help
               | where am I
               | clothing
               | games
               | accessories
               | electronics;
3) **Sports Grammar**

Grammar Sports;

Public <S> = <Simple> {Sentence};

<Simple> = | Help
            | Where Am I
            | Winter Games
            | Water Games
            | Indoor Games
            | Outdoor Games
            | Go Back
            | Home
            | Clothing
            | Accessories
            | Electronics;
4) Wintersports Grammar

grammar wintersports;

public <s> = <simple> {sentence};

<simple> = | where am I
               | help
               | ice hockey
               | snow skiing
               | snow boarding
               | go back
               | home
               | water games
               | indoor games
               | outdoor games;
5) Ice Hockey Grammar

grammar icehockey;

public <s> = <simple> {sentence};

<simple> = | where am I
  | help
  | snow skiing
  | snow boarding
  | go back
  | home
  | water games
  | indoor games
  | outdoor games
  | what is the price of helmets
  | what is the price of ice hockey skates
  | what is the price of hockey sticks
  | what is the price of stick bags
  | what is the price of gloves;
B) Language Files

1) File Welcome.m

interpret "help" = "hello there. This is the first speech department store. "
++ "it is the first department store information "
++ "service on speech web."
++ "we deal in clothes, sports, electronics, and"
++ "accessories."
++ "we have help available in each department and"
++ "their sub sections. you have these options to select"
++ "from. to go to"
++ "clothes department, say, clothes, to go to"
++ "electronics department, say,"
++ "electronics. to go to games department, say,"
++ "games. if you want to quit the application, say,"
++ "bye. please speak your choice now."

interpret "where am i"
= "This is Speech Department Store."
++ "It is the first department store information service on speech"
++ "Web. We deal in clothing, sports, accessories, and electronics."
++ "for assistance, say, help."

interpret "clothes" = "STACKREMOVE=2;"
++ "SIHLOURL=www.cs.uwindsor.ca;"
++ "CGI=cgi-bin/iqures1;"
++ "APPLICATION=clothes;"
++ "VOICE=female;"

interpret "games"
= "STACKREMOVE=2;"
++ "SIHLOURL=www.cs.uwindsor.ca;"
++ "CGI=cgi-bin/iqures1;"
++ "APPLICATION=sports;"
++ "VOICE=female;"
interpret "accessories"
  = "STACKREMOVE=2;"
  ++ "SIHLOURL=www.cs.uwindsor.ca;"
  ++ "CGI=cgi-bin/iquires1;"
  ++ "APPLICATION=accessories;"
  ++ "VOICE=female;"

interpret "electronics"
  = "STACKREMOVE=2;"
  ++ "SIHLOURL=www.cs.uwindsor.ca;"
  ++ "CGI=cgi-bin/iquires1;"
  ++ "APPLICATION=electronics;"
  ++ "VOICE=female;"
2) File Sports.m

interpret "where am i"
  = "you are in sports department. We deal in sports goods for"
    ++ "winter sports, water sports, outdoor sports and indoor sports."
    ++ "for assistance, please say help."

interpret "help"
  = "we deal in sports goods for winter sports, water sports, outdoor"
    ++ "sports, and indoor sports. you have these options to select from. To"
    ++ "go to winter games, say, winter games. to go to water games, say,
    ++ "water games."
    ++ "to go to indoor games, say, indoor games. to go to outdoor games,
    ++ "say,"
    ++ "outdoor games. to go back to home, say, go back, or say, home. to"
    ++ "go to accessories department, say, accessories. to go to electronics"
    ++ "department, say, electronics. to go to clothing department, say,"
    ++ "clothing. to ask for help again, say, help. please, speak your choice"
    ++ "now."

interpret "winter games"
  = "STACKREMOVE=3;"
    ++ "SIHLOURL=www.cs.uwindsor.ca;"
    ++ "CGI=cgi-bin/iqures1;"
    ++ "APPLICATION=wintersports;"
    ++ "VOICE=male;"

interpret "water games"
  = "STACKREMOVE=3;"
    ++ "SIHLOURL=www.cs.uwindsor.ca;"
    ++ "CGI=cgi-bin/iqures1;"
    ++ "APPLICATION=watersports;"
    ++ "VOICE=male;"

interpret "indoor games"
  = "STACKREMOVE=3;"
    ++ "SIHLOURL=www.cs.uwindsor.ca;"
++ "CGI=cgi-bin/iqures1;"
++ "APPLICATION=indoorsports;"
++ "VOICE=male;"

interpret "outdoor games"
  = "STACKREMOVE=3;"
    ++ "SIHLOURL=www.cs.uwindsor.ca;"
    ++ "CGI=cgi-bin/iqures1;"
    ++ "APPLICATION=outdoorsports;"
    ++ "VOICE=male;"

interpret "go back" = "STACK=1;"

interpret "home" = "STACK=1;"

interpret "clothing"
  = "STACKREMOVE=2;"
    ++ "SIHLOURL=www.cs.uwindsor.ca;"
    ++ "CGI=cgi-bin/iqures1;"
    ++ "APPLICATION=clothes;"
    ++ "VOICE=male;"

interpret "accessories"
  = "STACKREMOVE=2;"
    ++ "SIHLOURL=www.cs.uwindsor.ca;"
    ++ "CGI=cgi-bin/iqures1;"
    ++ "APPLICATION=accessories;"
    ++ "VOICE=male;"

interpret "electronics"
  = "STACKREMOVE=2;"
    ++ "SIHLOURL=www.cs.uwindsor.ca;"
    ++ "CGI=cgi-bin/iqures1;"
    ++ "APPLICATION=electronics;"
    ++ "VOICE=male;"
(3) File Wintersports.m

interpret "where am i"
  = "you are in winter sports department. we deal in sports goods for ice
  ++ "hockey snow skiing and snowboarding for assistance, please, say, help."

interpret "help"
  = "we deal in sports goods for ice hockey, snow skiing and snow"
  ++ " you have these options to select from. to go to ice hockey"
  ++ "section,"
  ++ " say, ice hockey. to go to snow skiing section, say, snow skiing. to"
  ++ " go to snow boarding section, say, snow boarding. to go to water"
  ++ " games"
  ++ " department, say, water games. to go to indoor games department, say"
  ++ " indoor games. to go to outdoor games department, say outdoor games"
  ++ " to go"
  ++ " back to sports department, say, go back. to go home, say, home. to"
  ++ " ask for help again, say, help. please, speak your choice now."

interpret "ice hockey"
  = "STACKREMOVE=4;"
  ++ "SIHLOURL=www.cs.uwindsor.ca;"
  ++ "CGI=cgi-bin/queries1;"
  ++ "APPLICATION=icehockey;"
  ++ "VOICE=female;"

interpret "snow skiing"
  = "STACKREMOVE=4;"
  ++ "SIHLOURL=www.cs.uwindsor.ca;"
  ++ "CGI=cgi-bin/queries1;"
  ++ "APPLICATION=snowskiing;"
  ++ "VOICE=female;"
interpret "snow boarding"
    = "STACKREMOVE=4;"
    ++ "SIHLOURL=www.cs.uwindsor.ca;"
    ++ "CGI=.cgi-bin/iquires1;"
    ++ "APPLICATION=snowboarding;"
    ++ "VOICE=female;"

interpret "go back" = "STACK=2;"

interpret "home" = "STACK=1;"

interpret "water games"
    = "STACKREMOVE=3;"
    ++ "SIHLOURL=www.cs.uwindsor.ca;"
    ++ "CGI=.cgi-bin/iquires1;"
    ++ "APPLICATION=watersports;"
    ++ "VOICE=female;"

interpret "indoor games"
    = "STACKREMOVE=3;"
    ++ "SIHLOURL=www.cs.uwindsor.ca;"
    ++ "CGI=.cgi-bin/iquires1;"
    ++ "APPLICATION=indoorsports;"
    ++ "VOICE=female;"

interpret "outdoor games"
    = "STACKREMOVE=3;"
    ++ "SIHLOURL=www.cs.uwindsor.ca;"
    ++ "CGI=.cgi-bin/iquires1;"
    ++ "APPLICATION=outdoorsports;"
    ++ "VOICE=female;"
3) File Icehockey.m

interpret "where am i"
    = "you are in ice hockey department. "
    ++ "we have helmets, ice hockey skates, hockey sticks, "
    ++ "stick bags, and gloves. "
    ++ "for assistance, please, say, help."

interpret "help"
    = "you are in ice hockey department. "
    ++ "we have helmets, ice hockey skates, hockey sticks, "
    ++ "stick bags, and gloves. "
    ++ "you have these options. to go to snow skiing"
    ++ " department, say, snow skiing, to go to snow boarding"
    ++ " department, say, snow boarding, to go back to winter games "
    ++ " deapartment, say, go back. to go home, say, home. to go to water"
    ++ "games"
    ++ " department, say, water games. to go to indoor games department,
    ++ " say, "
    ++ " indoor games. to go to outdoor games department, say, outdoor"
    ++ " games. to "
    ++ "ask for help again, say, help. to ask the price of an item, say, what is "
    ++ " the price for item. for example, to ask the price for helmets,"
    ++ " say, what is the price for helmets. please, speak your choice now."

interpret "snow skiing"
    = "STACKREMOVE=4;"
    ++ "SIHLOURL=www.cs.uwindsor.ca;"
    ++ "CGI=cgi-bin/iques1;"
    ++ "APPLICATION=snowskiing;"
    ++ "VOICE=female;"

interpret "snow boarding"
    = "STACKREMOVE=4;"
    ++ "SIHLOURL=www.cs.uwindsor.ca;"
    ++ "CGI=cgi-bin/iques1;"
    ++ "APPLICATION=snowboarding;"
    ++ "VOICE=female;"
interpret "go back" = "STACK=3;"

interpret "home" = "STACK=1;"

interpret "water games"
  = "STACKREMOVE=3;"
  ++ "SIHLOURL=www.cs.uwindsor.ca;"
  ++ "CGI=cgi-bin/iqures1;"
  ++ "APPLICATION=watersports;"
  ++ "VOICE=female;"

interpret "indoor games"
  = "STACKREMOVE=3;"
  ++ "SIHLOURL=www.cs.uwindsor.ca;"
  ++ "CGI=cgi-bin/iqures1;"
  ++ "APPLICATION=indoorsports;"
  ++ "VOICE=female;"

interpret "outdoor games"
  = "STACKREMOVE=3;"
  ++ "SIHLOURL=www.cs.uwindsor.ca;"
  ++ "CGI=cgi-bin/iqures1;"
  ++ "APPLICATION=outdoorsports;"
  ++ "VOICE=female;"

interpret "what is the price of helmets"
  = "they are hundred dollars each"

interpret "what is the price of ice hockey skates"
  = "they are two hundred dollars each"

interpret "what is the price of hockey sticks"
  = "they are fifty dollars each"

interpret "what is the price of stick bags"
  = "they are hundred dollars each"

interpret "what is the price of gloves"
  = "they are thirty dollars each"
**VITA AUCTORIS**

**NAME:** Irfan H. Qureshi

**PLACE OF BIRTH:** Karachi, Pakistan

**YEAR OF BIRTH:** 1971

**EDUCATION:**
- Comprehensive High School, Karachi, Pakistan. 1976 – 1986
- Dehli College, Karachi, Pakistan. 1987 – 1989
  Bachelors of Engineering in Electrical Engineering.
- University of Windsor, Canada. 1997 – 2000
  Master’s in Computer Science.