

HUman computer Interaction REPORT



December 6, 2013

Department of computer science

RISHI REDDY CHERUKU

**Human-Computer Interaction** **Report**

**Problem Statement:**

The rapid growth of computing is bringing the world closer day by day. The need to interact with computer needs to be improved. Human-Computer Interaction (HCI) is the interaction between man and machine the ultimate goal of Human-Computer Interaction Research is exploring how to design the computer to help people complete the necessary tasks more safely and efficiently [1]. The traditional approach to speech recognition system design has been to create an entire system optimized around a particular methodology. Sphinx-4 is a state-of-the-art speech recognition system written entirely in the Java programming language. It was created via a joint collaboration between the Sphinx group at Carnegie Mellon University, Sun Microsystems Laboratories, Mitsubishi Electric Research Labs (MERL), and Hewlett Packard (HP) [2]. In this context we would like to build an voice based application which reduces the usage of keyboard to the extent possible. We would like to build the application with which we would be able to operate a computer to perform operation like opening files and also able to perform some basic calculations on the calculator.

**Approach:**

In this Project we have used Rule Based System where there are set of rules to be followed. The working memory is the Grammar file. If the rule or rules matches then we reach to the final state and the Goal is performed. In case the rule is not matched or satisfied then we don't move the final state or the solution is not obtained. The basic approach we followed to build the application is:

1.) Main file: This file contains the exact logic of how the application should work.

2.) Grammar file: This file contains the speech recognition syntaxes which we need to speak to open the application.

3.) Configuration file: This file is used to link the main file with the Sphinx-4 jar files.

*A. The Structure of the Main File:-*

* ConfigurationManager this is main class that collects information about the machine and Configuration file to obtain the resources to the machine. Then we link the Configuration file (.xml) with the ConfigurationManager.
* We look up for Voice recognizer based on the Configuration file to allocate the system resources. We look up for a microphone to accept voice commands then recognizer will match the commands to the tokens provided in the Grammar File.
* We check if there is any input to the micro phone if it returns False we will remove the resources from the heap and exit the program else we will get the input from the microphone then match it with the grammar file (.gram) and then get the string to perform the respective action.
* Based on the Input from Microphone say "Open Notepad" we will check this with the token in the grammar file "Open Notepad". If it matches then we try to open the file from the command prompt and again ask the user for another input.
* For Calculator when we say "one hundred add two result" we will get the output as 102. we have also implemented intelligent calculator like "one zero zero add two result" will also give us the output as 102.
* Then we used Text To Speech (TTS) an open source API provided by sun-micro systems. In FreeTTS we have used "Kevin16" voice to make the computer to speak to the human.

so we have built an application where it interacts with us after we speak like if the system is not able to understand what we speak it speaks out "Oops! didn't hear you".

*The Structure of the Grammar File:-*

* The Grammar file contains all the list of the tokens which should be detected when we speak to the system.
* Example:

grammar menu;

public <command> = show statistics { stats } |

exit the program { exit } |

open note pad { note pad } |;

* So here when speak "open note pad" the system will recognize and then match with the grammar file "open notepad" and returns notepad. Then we will use notepad as input to open the file.

*The Structure of the Configuration File:-*

* This file specifies the configuration of the system to test the data. The basic version would be provided by CMU and we need to build over it.
* Example:

<component name="jsgfGrammar" type="edu.cmu.sphinx.jsapi.JSGFGrammar">

<property name="dictionary" value="dictionary"/>

<property name="grammarLocation"

value="resource:/com.dialog.Dialog!/com/dialog/"/>

<property name="grammarName" value="menu"/>

</component>

* Here the we need to specify the file path and grammar file name in order to use it by sphinx-4.

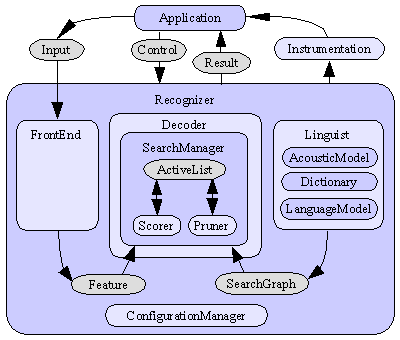
**Design and Implementation:**

We have first understood the sphinx-4 architecture and then started implementing a basic program. The sphinx-4 architecture is described below:

***Sphinx 4 Architecture:***

Figure 1 illustrates the Sphinx 4 Architecture [3]. Each component of Sphinx 4 is explained below.

*Recognizer* – The Recognizer Contains three main components of Sphinx-4, the front end, the linguist, and the decoder. The application interacts with the Sphinx-4 system mainly via the Recognizer.



Figure(1): This Illustrates the Architecture of Sphix-4 speech recognition system.

*Audio* – The Input for the Sphinx4 is generally a voice that speaks out to any hardware device that takes it as input and passes to decode it.

*Front End* - Performs digital signal processing (DSP) on the incoming data, processes the input data(the audio) into a list of features.

*Feature* - The output of the front end are features, which are used for decoding in the rest of the system.

*Linguist* –The body which contains the knowledge base of the entire Sphinx4, which are the acoustic model, the dictionary, and the language model. The output is generally a search graph, on which the search manager performs searching operations using different algorithms.

*Acoustic Model* - Contains a representation (often statistical) of a sound, often created by training using lots of acoustic data. Usually they are represented in the form of Phenomes.

*Dictionary* - Responsible for determining how a words are pronounced.

*Language Model* - Contains a representation (often statistical) of the probability of occurrence of words.

*Search Graph* - The graph structure produced by the linguist according to certain criteria (e.g., the grammar), using knowledge from the dictionary, the acoustic model, and the language model. Generally creates the phonemes or HMM’s for each and every letter.

*Decoder* - Contains the search manager. which performs various search operations such as searching the required phenome representation for the word.

*Search Manager* - Performs search using certain algorithm used, e.g., breadth-first search, best-first search, depth-first search, etc.. Also contains the feature scorer and the pruner.

*Active List* - A list of tokens representing all the states in the search graph that are active in the current feature frame.

*Scorer* - Scores the current feature frame against all the active states in the Active List.

*Pruner* - Prunes the active list according to certain strategies.

*Result* - The decoded result, which usually contains the most probable best results, or N –best results.

*Configuration Manager* - loads the Sphinx-4 configuration data from an XML-based file, and manages the component life cycle for objects.

Build the Main Class file with the logic to be implemented and link it with the Sphinx-4.

Build Grammar file with the list of tokens to be implemented.

Add the Grammar file and main class file to the configuration file.

Figure(2): Flow Chart representation of the Human Computer Interaction.

We have tried to implement the basic Desktop application to open the files like notepad, all ms office file, windows media player, Chrome, Facebook, File search and USB detection.

The Other module we tried to implement is Basic Speech recognition calculator which will perform basic calculations like Addition, Subtraction, Multiplication and Division. This is also an intelligent calculator like if the user speaks "one hundred five" or "one zero five" it will take it as 105.

**Results:**

Below are the list of the things we have implemented.

Table 1: Illustrates the application implemented and status of the application.

|  |  |
| --- | --- |
| **Implemented Applications** | **Status** |
| Basic Calculator application | success |
| Addition | success |
| Subtraction | success |
| Multiply | success |
| Division | success |
| open and close notepad | success |
| open and close paint | success |
| open and close calculator | success |
| open and close ms-word | success |
| open and close google chrome | success |
| open and close Microsoft Word | success |
| open and close Microsoft Excel | success |
| open and close Microsoft Power Point | success |
| open and close Microsoft Access | success |
| open and close Microsoft Outlook | success |
| open and close facebook | success |
| open and close gmail | success |
| File Search | success |
| Folder Search | success |
| Detect a USB | success |
| Open and close media player | success |

**Conclusions and Future work:**

This project work of human computer interaction started with a brief introduction of technology and its applications in the different sectors. This application has lot of importance in the present era since Voice recognition is used everywhere . There is an in Increased human – computer interaction where speech is one of the way to interact, and it is not necessary to sit at a keyboard and finally no training is required.

This work can be taken into more detail and more work can be done on the project in order to bring modifications and additional features. The current software doesn’t support a large vocabulary(grammar file), the work will be done in order to accumulate more number of samples and increase the efficiency of the software. In the future the computer might grasp the meaning of the words ,Maintenance of a log file(contains the questions that were not answered),involve in a learning process, greater use will be made of intelligent systems that helps the computer to clearly understand what the user is speak and finally accuracy will become better and better.

**References:**

[1]. Xianyi Yang; Guo Chen, "Human-Computer Interaction Design in Product Design," *Education Technology and Computer Science, 2009. ETCS '09. First International Workshop on* , vol.2, no., pp.437,439, 7-8 March 2009

[2]. "Sphinx Information",<http://cmusphinx.sourceforge.net/sphinx4/doc/Sphinx4Whitepaper.pdf> , (Accessed : 25 November 2013)

[3]. "Sphinx 4 Architecture",<http://cmusphinx.sourceforge.net/sphinx4/>,(Accessed : 25 November 2013)