Sphinx 4 Speech Recognition in ATC

Vatsala Mathapati[1], Anjaney Koujalagi[2], Naveen Kumar C[3]

[1]Software Engineering, Department: CSE, AMC Engineering College Bangalore, India
[2]VLSI and Embedded systems, Department: ECE, Nitte Meenaxi Institute of Technology Bangalore, India
[3]Assistant professor, Department: CSE, AMC Engineering College Bangalore, India

Abstract**—***Speech Recognition plays a very important role in day to day life. Speech Recognition is widely used and addicted by this world as it allows users to communicate with computers by recognizing their spoken language. Communication with Speech Recognition made our lives easy. There are many types of open source Speech Recognition Engine. Different types of application are built using Speech Recognition Engine. An Application is built for Air Traffic Controller. A new Software library with good grammar is proposed according to Air Traffic Controller commands. This Software library is used to build successful working application.*

***Keywords— ATC commands; JSAPI; JSGF; LM ;Sphinx Decoder Framework*.**

I. INTRODUCTION

This paper is determined for training ATC, represents an excellent application for SR system .

Fig.1: Pilot handling Aircraft

Fig.2: Air Traffic Controller handling pilot

The Figure 1 represents Pilot handling Aircraft. The Figure 2 represents ATC handling pilot with commands.

A. Abbreviations and Acronyms

Speech Recognition (SR), Language Model (LM), Air Traffic Controller (ATC), Java Speech Application Programming Interface (JSAPI), Java Speech Grammar Format (JSGF).

SR converts spoken words into text. There are many open source SR software. This software should be installed in computer, which helps computer to understand spoken language and be able to translate it in computer language so that it acts according to spoken language. Speech is recognized through Signal Analyzer.

- Signal Analyzer:
  High quality microphone picks up the speaker sound and converts to signal and signal analyzer separates the background noise and passes only speaker's speech and sends to Acoustic model.

There are two types of Speech Recognition. User can use either of them. Both Speech Recognition systems contains corpus collection individually.

- Speaker Dependent SR system:
  Single person voice is preconfigured into system in the form of corpus as shown in figure 3 and only that unique person's pattern is recognized to perform task. This is applied for small scale industries and requires fewer databases.
- Speaker Independent SR system:
  Many people's voice is preconfigured into system in the form of corpus as shown in figure 4 and any one's voice is recognized to perform task. This is applied for large scale industries and requires large database as it stores many person's voice samples.

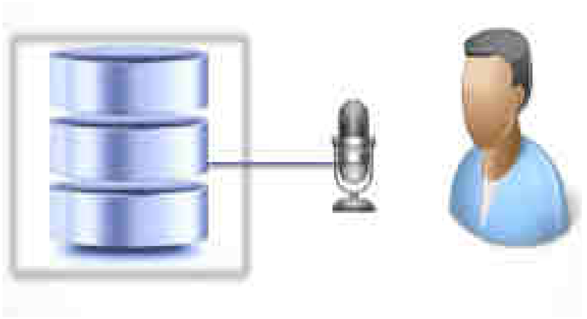


Fig.3: Corpus collection for Speaker Dependent SR

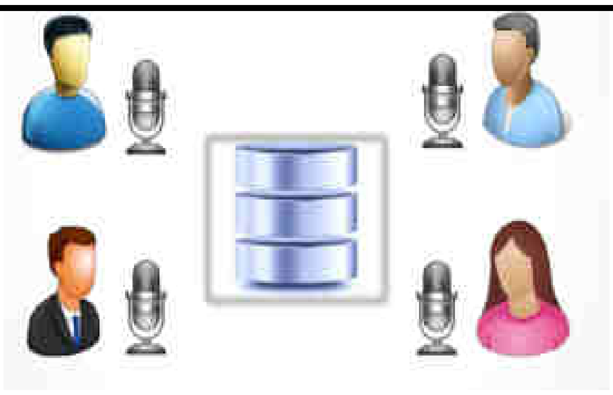


Fig.4: Corpus collection for Speaker Independent SR

Speech Recognition contains two models:

1) Acoustic Model:
   Acoustic Model recognizes speech samples and breaks into phonemes using probability based on mathematical model. The samples of phonemes with highest probability of a match is selected as correct phoneme and passed to Language Model
2) Language Model:
   Once phonemes in speech samples are identified, the LM is used to recognize and decide what words has been spoken. Thousands of phonetic spellings and meaning need to be provided in dictionary file. The grammar file is also generated to compare words found in dictionary.

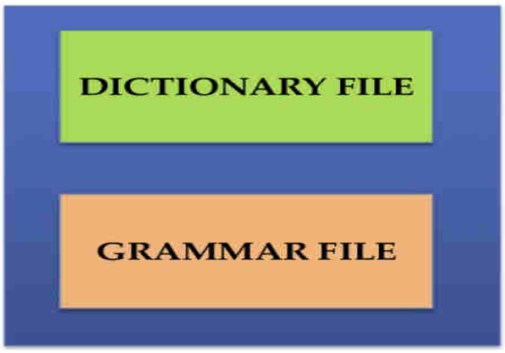


Fig.5: Language Model

JSAPI and Sphinx 4 tools used to build Speech to text applications.

- JSAPI:
  To support commands and control recognizers the java speech application programming interface is required. JSAPI supports two cores. First is Speech Synthesis used to build the text to speech technology.
  Second, Speech Recognition is used to build the speech to text technology. It provides computer with the ability to listen the person's speech and identify what has been said. JSAPI have some classes and interfaces, which are grouped into packages
  1) Javax.speech:
     It provides classes and interfaces for a speech engine.
  2) Javax.speech.synthesis:
     It provides classes and interfaces for speech synthesis.
  3) Javax.speech.recognition:
     It provides classes and interfaces for speech recognition.
- Sphinx 4:
  The figure 6 represents the overall architecture of Sphinx 4 Framework. It has three modules to perform high flexibility and modularity. The three modules are FrontEnd, Decoder and Linguist.

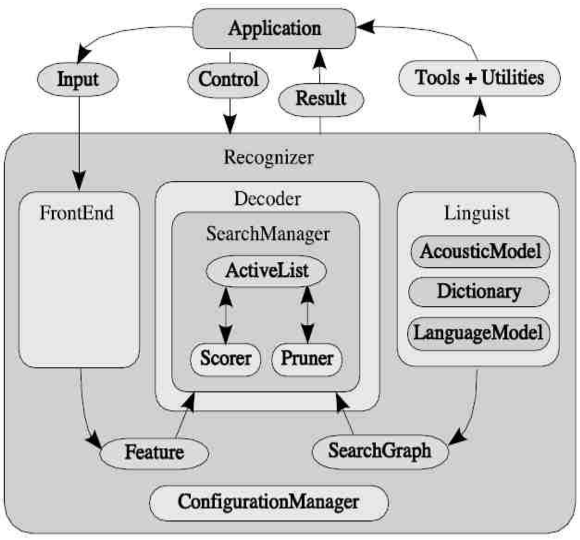


Fig.6: The Framework for Sphinx 4

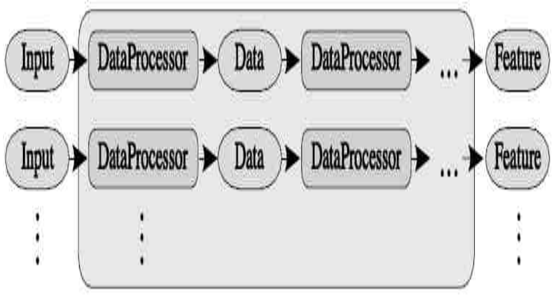


Fig.7: Sphinx 4 FrontEnd

FrontEnd:
Front end picks up input as audio signal and parameterize an input signal into sequence of output features. Different types of parameters from same or different signals are also performed by parallel chains of communicating Data processors. Each Data processor have input and output to

communicate with other Data processor. The last Data processor provides feature as output.

Decoder:

The decoder request Search manager to recognize set of feature frames which are output of Data processor. Search Manager creates Result object containing all the paths that have reached final state. Search Manager is implemented with Search algorithm. The Decoder has sub Frameworks. First, ActiveList actives token using pluggable pruner. Second, Pruner performs a garbage collection in java platform. Third, Scorer provides score to requested SearchManager by calculating score using mathematical operations.

Configuration Manager configures the parameter and provides Sphinx 4 to perform dynamically load and configure modules at run time.

Linguist:

The Linguist generates SearchGraph which is then used by Decoder during search. It contains three pluggable components are Acoustic Model, Language Model, Dictionary and SearchGraph. The SearchGraph is the directed graph with n nodes present in it and each node represent search state either an emitting or a non-emitting.

II. EXISTING SYSTEM

The JSAPI and Sphinx 4 libraries are already used in Java Language to build applications such as Google application, Speech to Text Android application and so on. Sphinx 4 applications are not yet applied for Air Traffic Controller in Aerospace. There is no such software libraries present in the existing Sphinx 4, Lot of required words are missing in sphinx 4 pronunciation dictionaries. In sphinx 4, pronunciation dictionary is maintained with only normal English words used in daily life but not related to Indian ATC command dictionary.

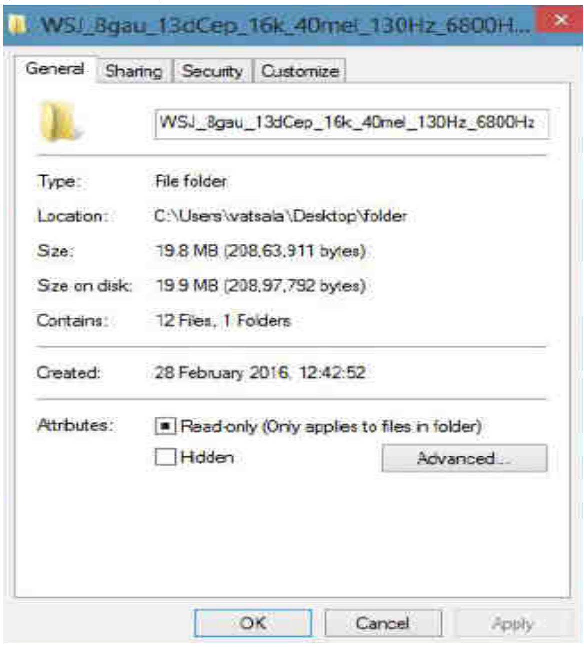
III. PROPOSED SYSTEM

A new software library is proposed using Air Traffic Controller commands in java platform. This library files then imported in Eclipse tool. Some steps need to follow are

Selecting high frequency microphone:

Speech Recognition system requires high frequency microphone to recognize speech and it should be able to recognize approximate voice sample of person. The microphone should be very near to speaker and should be surrounded with cool and silent area. The microphone should work accurately and be able to convert person's sound waves to signal.

1) Creating Dictionary file:
   The Dictionary file is created with collecting the phonemes pattern from the Acoustic Model and the correct word which matches to the phoneme pattern. Phoneme is the small unit of word which is created by Acoustic Model according to the pronunciation. When phonemes are identified, search is performed on the dictionary file and produce correct word. The dictionary contains thousands of phonetic spellings and corrects word of Aircraft name, numbers, commands and unit which are used for Aircraft application.
2) Creating a Software library
   To create a software library, four dictionary files are stored in a folder and should create a jar file to import in the Eclipse tool to support and run Java application. Along with the existing JSAPI and Sphinx 4, proposed Software library is also added by creating jar files. The command syntax for creating jar files:
   Jar cf jar-file input-files(s)
   c : Represents to create jar file
   f : Represent to save output in a file
   jar-file: Represents any file name can be given by the user with .jar extension.
   Input-file(s): Represent space separator, used when more than one file need to be added in a jar file.
   Figure 8 represent the Dictionary name and location, using this detailed location jar is created with commands in command prompt which is presented in figure 9.


WSJ_8gau_13dCep_16k_40mel_130Hz_6800H...

General | Sharing | Security | Customize

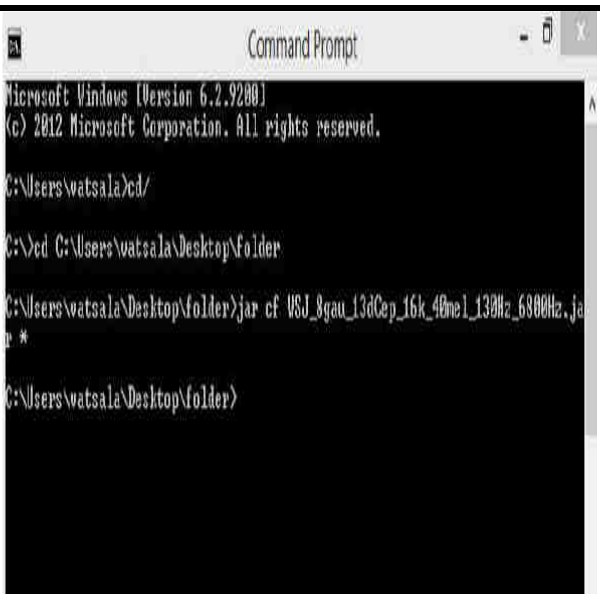
WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz

Type: File folder
Location: C:\Users\vatsala\Desktop\folder
Size: 19.8 MB (208,63,911 bytes)
Size on disk: 19.9 MB (208,97,792 bytes)
Contains: 12 Files, 1 Folders

Created: 28 February 2016, 12:42:52

Attributes: ■ Read-only (Only applies to files in folder)
☐ Hidden

Fig.8: Dictionary name and location

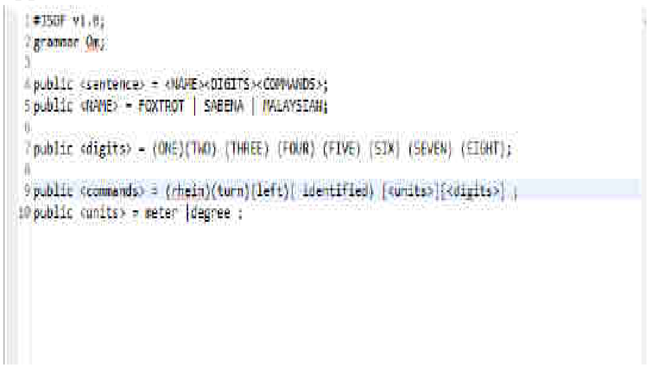


```
Command Prompt
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

C:\Users\watsala>cd/

C:\>cd C:\Users\watsala\Desktop\folder

C:\Users\watsala\Desktop\folder>jar cf WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz.ja
r *

C:\Users\watsala\Desktop\folder>
```

Fig.9: Creating software library

3) Creating Grammar file:
The Grammar file is created by using JSGF and it is platform independent. JSGF is Java Speech Grammar Format. It is format for textual representation of context free grammars. Grammar helps to determine what the recognizer should listen and describes the utterances of user sayings.

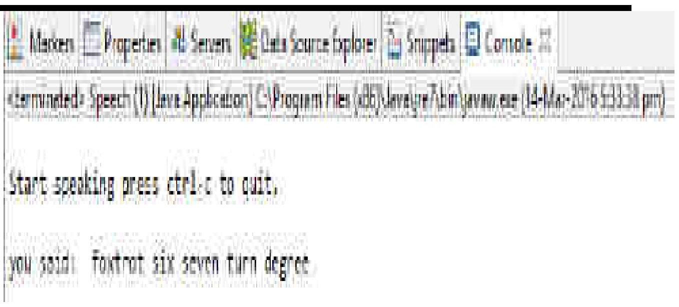
4) Set path in Java code:
The xml form is generated to use xml elements to represent grammar constructs. The grammars of certain application requirements are stored in xml form. This xml path is then set in java code to run the java application.



```
#JSGF v1.0;
grammar Om;

public <sentence> = <NAME><DIGITS><COMMANDS>;
public <NAME> = FOXTROT | SABENA | MALAYSIAN;

public <digits> = (ONE)(TWO) (THREE) (FOUR) (FIVE) (SIX) (SEVEN) (EIGHT);

public <commands> = (rhein)(turn)(left)( identified) [<units>][<digits>] ;
public <units> = meter |degree ;
```

Fig.10: Creating JSGF

The figure 10 shows Java Speech Grammar Format for Aircraft commands. First line represent version of JSGF. Remaining line represent voice tokens of Aircraft commands. The unit and digits within square bracketsshows they are optional. The or operator shows only one voice token is selected. The parenthesis shows any combination can be selected. The figure 11 shows the Result of speech in text form with the help of Grammar



```
<terminated> Speech (1) [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (14-Mar-2016 5:33:38 pm)

Start speaking press ctrl-c to quit.

you said:  foxtrot six seven turn degree
```

Fig.11: Result

IV. CONCLUSION

The new software library is proposed for Indian English ATC. It is developed by using java technology in Eclipse tool. This library helps and communicates best performance between ATC and pilot.

ACKNOWLEDGEMENTS

I have got a best opportunity to present the paper on "Sphinx 4 Speech Recognition in ATC". I am thankful to who all encouraged to complete this paper.

REFERENCES

[1] E. Craparo, E. Feron, Natural language processing in the control of UAV, Guidance, Navigation, 2004.
[2] S. Young, "The HTK hidden Markov model toolkit: Design and philosophy," Cambridge University Engineering Department, UK, Tech. Rep. CUED/FINFENG/ TR152, Sept. 1994.
[3] Voce http://voce.sourceforge.net
[4] K. F. Lee, H. W. Hon, and R. Reddy, "An overview of the SPHINX speech recognition system," IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 38, no. 1, pp. 35–45, Jan. 1990.
[5] J. K. Baker, "The Dragon system - an overview," in IEEE Transactions on Acoustic, Speech and Signal Processing, vol. 23, no. 1, Feb. 1975, pp. 24–29.
[6] http://www.speech.cs.cmu.edu/cgi/bin/cmusphinx/twiki/view/Sphinx4/Train%erDesign
[7] CMU Sphinx: http://cmusphinx.sourceforge.net
[8] R. G. Leonard and G. R. Doddington, "A database for speaker-independent digit recognition," in Proceedings of the International Conference on Acoustics, Speech and Signal Processing, vol. 3. IEEE, 1984, p. 42.11.
[9] JSGF: http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/
[10] The Native Interface in JAVA : http://java.sun.com