IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS), ISSN: 2249-9555
Vol. 2, No.5, October 2012

Email by Voice

[1]Suma Swamy, K V Ramakrishnan
[1]Department of ECE,
Anna University, Chennai, India
[1]suma_swamy@yahoo.com, ramradhain@yahoo.com

[2]Smitha Crystal D' Almeda, [2]Shwetha Rani G, [2]Radha K
[2]Department of Computer Science and Engineering,
Sir M Visvesvaraya Institute of Technology, Bengaluru, India
[2]crystalsmitha24@gmail.com,
[2]shwetha457@gmail.com, anuradha7425@gmail.com

Abstract— This paper presents an approach to the development of a speech to text conversion system with three demarcating features: Continuous speech, Speaker independent and Text dependent. A speech-to-text engine is developed, and is implemented as a system on programmable chip (SOPC) solution. "IBM via Voice" for speech recognition converts the speech to text and stores it as a wave file. The system acquires speech at run time through a microphone and processes the sampled speech to recognize the uttered text. It converts the recognized speech to text. The process involves building an application that makes use of voice to generate email content and send the mail without human physical interaction using Java Development Kit (JDK 1.6), Java Programming Language as the platform (J2SE). Net beans IDE 6.1 and Google Voice Recognizer (IBM via Voice).

Keywords— SOPC, JDK, J2SE, IDE, JSGF

I. INTRODUCTION

Speech to text is a technology that uses speech recognition to translate spoken words into text. In Speaker Dependent speech recognition systems, the system is trained by the user. Speaker Independent speech recognition systems do not require training. [1]

A speech recognition system is able to identify words and phrases in spoken language. These words or phrases are converted to a machine-readable format. Isolated word speech recognition system uses a limited vocabulary. These systems can identify only limited number of words when they are spoken very clearly and with pauses.

Speech recognition systems can be divided into two types: (i) Text dependent (ii) Text independent.

In Text-dependent systems, a speaker utters a pre-selected phrase whereas in Text-independent system it can be any text or phrase. [1][2]

This paper describes how the speech to text conversion is implemented for the purpose of sending emails, where all the operations required are carried out solely by means of speech. This system is speaker independent and is capable of converting speech into text for any speaker.

II. PROPOSED MODEL

Program Database

The system is text dependent. Hence the words that need to be uttered are all stored in the grammar file. It uses XHTML coding.

The Fig. 1 illustrates the proposed model of the system.

The pseudo code for the grammar file (XHTML) is shown in the BOX 1.

JSGF V1.0 stands for Java Speech Grammar Format: It is a textual representation of grammars for use in speech recognition technologies like XHTML + voice. JSGF adopts the style and conventions of Java Programming language. [3]

The grammar file (Order_Search.gram) and contacts file (Contacts.txt) are stored as program database.

The Order_search.gram is a grammar file, where the text (words) is being stored as an xml program. This depicts the Text-Dependent feature. However, it can be dynamically changed.

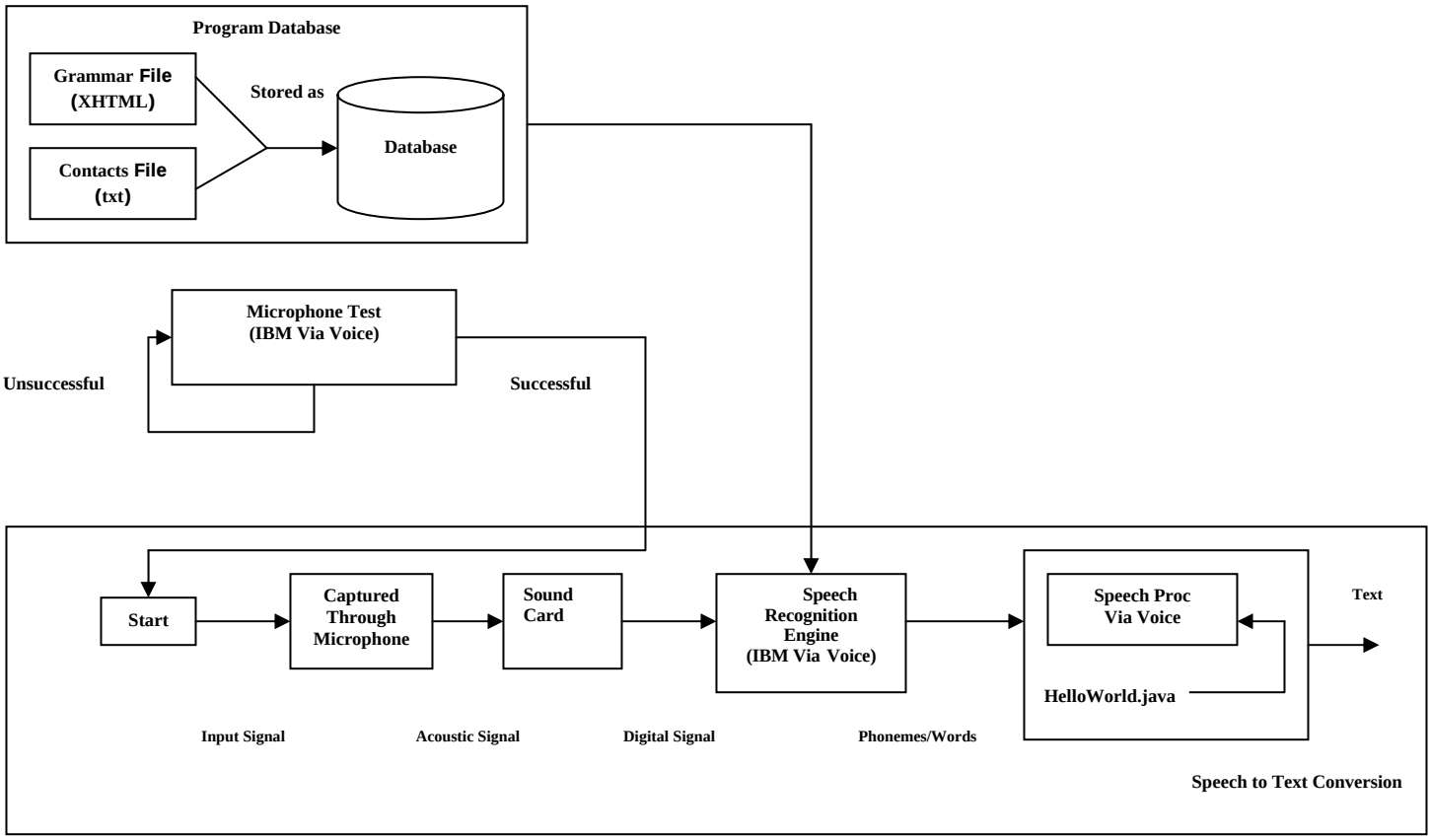

Figure 1. Proposed Model

```
#JSGF V1.0;
grammar startText;
// Body
public <startText> = (please | smitha | hello | kindly |
udith | nitesh | vinod | amar |
could you | apoorva | oh mighty computer | Hai How are
u | mode1 | stage one |
stage two | stage three | clear email address | gunjan |
vishal | rajesh | see see how it is
| cleartext | raghavendra | space | endaddress | send mail |
fullstop | newline | comma |
vin | email address | hai deepti | jack | abhishek |
backspace) *;
```

BOX 1. Creation of grammar File

The Contacts.txt is a text file that is used to store the E-mail addresses of the users to whom the mail needs to be sent. This uses two arguments as the format to store contacts, i.e the user name must be followed by the mail address.

The contacts file (Contacts.txt) is shown in BOX 2.

```
(shwetha,shwetha457@gmail.com)
(radha,anuradha7425@gmail.com)
(smitha,crystalsmitha24@gmail.com)
```

BOX 2. Creation of contact File

Microphone Testing Phase

In this phase, the microphone is being tested to check if it is working or not. This is done by initializing the "IBM via Voice" user wizard. The steps involved are:

1. Open the IBM viaVoice user wizard of test.
2. Connect the microphone.
3. Select the number of connectors.
4. Click start and read the displayed passage.
5. If successful message is displayed, click finish else repeat.

On a successful note proceed with the next phase.

Speech-to-Text conversion Phase

Once the microphone is tested successfully, the speech signals (analog signals) captured by microphone are converted to digital signals by the sound card. This digital signal acts as an input to the speech recognition engine. The speech recognition engine is implemented using IBM via Voice, which generates Phonemes (segmental units of speech) or words. The HelloWorld.java function is used to

read the input phonemes or words spoken by the user and converts it into text, which is printed on the screen.

Create synthesizer command is used which reads the words spoken by the speaker. These spoken words are recognized using a recognizer command. Once the words are recognized threads are created. New threads are created for different words spoken. These threads call the run method which is used to identify whether the words spoken are valid or not i.e., it maps the (name,value) pair. "Text.equals" command is used to check if the two are equal i.e., whether the word spoken is stored in the grammar file. The tokens present in the words spoken are retrieved using the "text.contains" command.

The word spoken is checked with the grammar file using the getsource() function. The words are broken into tokens and is checked with the words stored in grammar file using the gettokens() function. Each letter is compared and if the word matches any of the words present in the grammar file then it is printed on the screen using "println" command.

The tokens are also compared to check if it is with respect to the modes i.e., if the tokens match "stage" command then it is with respect to the modes of working.

If the user tells "stage-1" then the mode is set to 1 where the words are typed as said by the user. For e.g., if the user tells newline in this mode then a newline is appended into the text using "append" command. If the user tells comma, fullstop, space or cleartxt then a ',' '.' ' ' and the text entered are cleared respectively. When the user gives any of this command he has to tell some other words because these commands are used to indicate break in the sentences. "Continue" command is used after each of these conditions to indicate user needs to speak some words.

If the user speaks "stage 2" then the control goes to the second mode where the user needs to tell the mail id to which he wishes to send the mail.

If the user tells "stage3 "then the control goes to third stage, which is the command mode. The user can use send or drafts command where the mail is sent to the specified mail id or stored in his email which can be sent later respectively.

The mail id entered by the user must match the mail id's that are stored in contacts.txt file. If the id entered does not exist in the file then it is considered to be an invalid one and an exception is produced which is to be handled by the user. If the user exceeds the timeout period i.e., if the user remains silent after a specified period of time a timeout exception is produced. Thus the converted speech is displayed as text.

III. IMPLEMENTATION

The programming is done in JAVA. The program layout consists of FrameThread.java, .HelloWorld.java, LoginForm.java, MainFrame.java, SendMail.java and SignUpForm.java.

The FrameThread.java is used to start a thread (process) and thus initiates the LoginForm.java and runs it. And when the sign up option is been selected, the thread control moves to the SignUpForm.java. Once the data has been entered and the validation process has been done, control logic and flow of the program is stored in HelloWorld.java. and the sending of mail is by invoking SendMail.java. Contacts.txt and Order_search.gram files are also included.

The Order_search.gram is a grammar file, where the text (words) is being stored as an xml program. This depicts the Text-Dependent feature. However, it can be dynamically changed.

The Database command is used to store the user information entered in SignUpForm.java as shown in BOX 3.

```
Create database email_data;
Use email_data;
Create table login( username varchar(100) , password
varchar(100));
```

BOX 3. Creating Database

A. *Major Tools*

The major tools used are.

JDK Tool kit: This Java Development Kit (JDK) from Oracle Corporation is very widely used by software developers. [4]

NetBeans 6.1 IDE: The NetBeans IDE is written in Java and can run anywhere. A JDK is required for Java development functionality. [5]

Speech Pro-IBM viaVoice: Current version of IBM ViaVoice is a language-specific continuous speech recognition software products offered by IBM specially designed for use in embedded systems. [6]

Talking Java SDK: This is a full implementation of Sun's Java Speech API for Windows platforms from Cloud garden allowing the use of wide range of SAPI4 and SAPI5 compliant Text-To-Speech and Speech-Recognition engines in different languages. These are programmed using the standard Java Speech API. [7]

MySql and MySql Tomcat Connector: MySQL is a high performance, very fast, reliable and flexible open source Relational Database Management System. It is a multithreaded and multiuser Relational Database management system. [8]

B. *Program Layout*

The program layout basically illustrates the actual implementation of the system as shown in Fig. 2, and depicts every level of description discussed in previous section as coding concept.

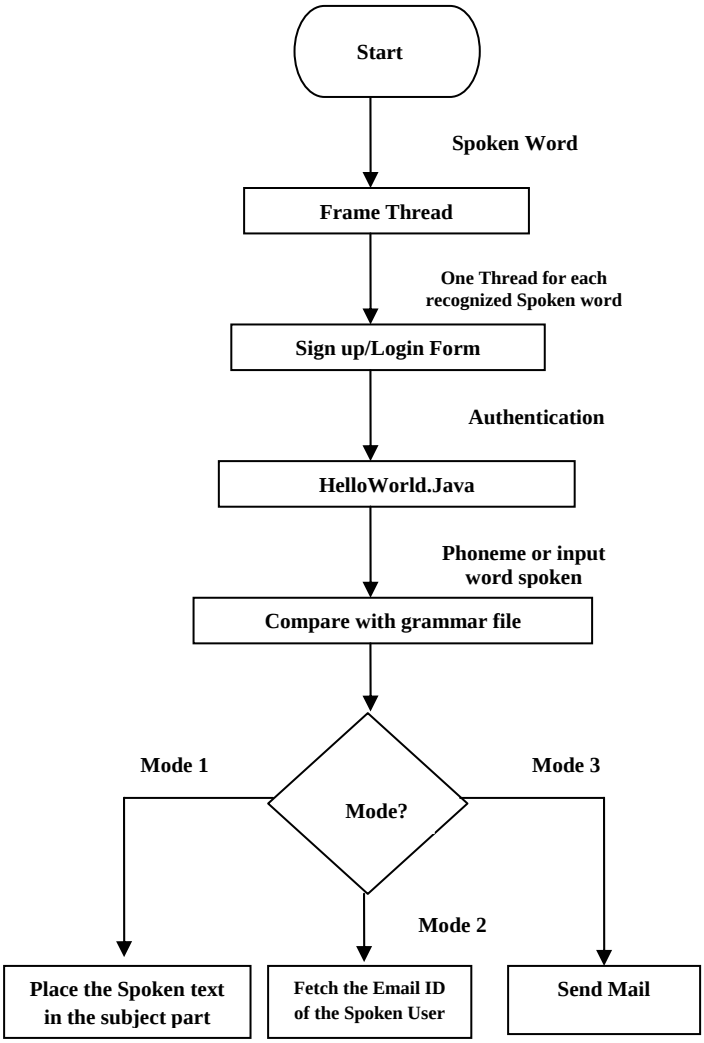

Figure 2. Flow Diagram of the program layout

1) Frame Thread

The BOX 4. shows a pseudocode for initializing thread.

```
package speechrecognitionexample;
public class FrameThread implements Runnable {
public void run()
{
new MainFrame().setVisible(true);
//MainFrame.To_Email_Adress.setText("");
new LoginForm().setVisible(true);
//MainFrame.To_Email_Adress.setText("");
}
}
```

BOX 4. Initializing Thread

As mentioned in the program modules the NetBeans IDE v6.1 is used to compile and execute. It requires that the program code needs to be stored under the package. Thus in the beginning line as shown above we are including the package of source code 'speechrecognitionexample'. The class is named as FrameThread and is declared as public and it implements Runnable in order to create threads. In order to construct a thread on any object a single method called run() is needed. The run function is called whenever new threads i.e., whenever a new user logs in to send Email using Speech to Text Conversion Systems. This is set using new LoginForm().setVisible(true).where the user enters the Username and Password to gain access.

2) Signup Form/Login Form

From the previous module the login form is displayed. This login form contains two text fields namely username and password and two buttons namely signup and login. If the user is already registered he can directly access the system by entering the username and password by clicking the login button. If the user is not yet registered he clicks the Sign up button. Sign up button leads to the signup form where the user creates a new account. To create a label jLabel.setText is used. The parameter passed to this function is the name that is set to the textbox. The pseudo code is as shown in BOX 5.

```
public class LoginForm extends javax.swing.JFrame {
String url = "jdbc:mysql://localhost:3306/";
String db = "email_data";
String driver = "com.mysql.jdbc.Driver";
/** Creates new form LoginForm */
public LoginForm() {
jLabel1.setText("Username");
jLabel2.setText("Password");
Login_Btn_Submit.setText("Login");
Login_Btn_Signup.setText("Sign Up");
if(!username.equals("")&&!pass.equals("")){
system.println("wrong password");}}
```

BOX 5. Login form

3) HelloWorld.java

The HelloWorld.java function is used to read the input phonemes or words spoken by the user and converts it into text as shown in BOX 6, which is printed on the screen. "Create synthesizer" command is used which reads the words spoken by the speaker. These spoken words are recognized using a recognizer command. Once the words are recognized threads are created. New threads are created for different words spoken. These threads call the run method which is used to identify whether the words spoken are valid or not i.e., it maps the (name,value) pair. "Text.equals" command is used to check if the two are equal i.e., whether the word spoken is stored in the grammar file. The tokens present in the words spoken are retrieved using the "text.contains " command.

The word spoken is checked with the grammar file using the getsource() function. The words are broken into tokens and is checked with the words stored in grammar file using the gettokens() function. Each letter is compared and if the

word matches any of the words present in the grammar file then it is printed on the screen using println command. The tokens are also compared to check if it is with respect to the modes i.e., if the tokens match stage command then it is with respect to the modes of working. If the user tells stage-1 then the mode is set to 1 where the words are typed as said by the user. If the user speaks Stage-2 then the control goes to the second mode where the user needs to tell the mail id to which he wishes to send the mail. If the user tells Stage-3 then the control goes to third stage, which is the command mode. The user can use send or drafts command where the mail is sent to the specified mail id or stored in his email which can be sent later respectively.

The mail id entered by the user must match the mail id's that are stored in contacts.txt file. If the id entered does not exist in the contacts.txt file then it is considered to be an invalid one and an exception is produced which is to be handled by the user.

```
public class HelloWorld1 extends ResultAdapter {
public void resultAccepted(ResultEvent e) {
for i =0 to tokens.length{
String Text=tokens[i].getSpokenText();
if(mode==1){ if(Text.equals("newline")) {
MainFrame.Email_Txt.append("\n");
continue; }
else if(mode==2) {if(Text.contains("stage")) {
if (tokens.length>1){
if(tokens[1].getSpokenText().equals("one")) {
mode=1;
System.out.println("code set to 1");}
else if(tokens[1].getSpokenText().equals("two")) {
mode=2;
System.out.println("code set to 2");}
else if(tokens[1].getSpokenText().equals("three")) {
mode=3;
System.out.println("code set to 3");}} break; }
```

BOX 6. HelloWorld.java

IV. RESULT ANALYSIS

The experimental results for a speaker-independent, Text-dependent, continuous, Speech-to-Text Conversion system is depicted as follows in Table 1 and Fig. 3 for 50 trials:

TABLE I. EXPERIMENTAL RESULT TABULATION

Spoken Words	Efficiency (%)
Hello Welcome	90
Hi How are you	90
Could you please	92
Its necessary	86
Shall be Done	92

The overall efficiency of the system is 90%

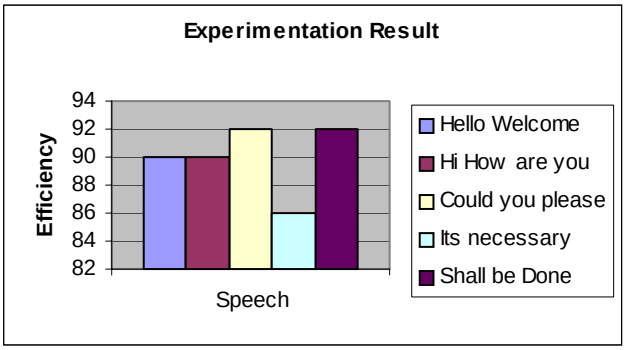

Figure 3. Graph of Experimental Results

V. CONCLUSION

An application for sending e-mail by voice using the existing tools such as Java Development Kit (JDK 1.6), Java Programming Language as the platform (J2SE). Net beans IDE 6.1 and Google Voice Recognizer (IBM via Voice) is developed. It helps to generate email content and send the mail without human physical interaction.

ACKNOWLEDGMENT

The author acknowledges Anna University, Chennai for encouragement and the permission to publish this paper. The author would like to thank the Principal of Sir M Visvesvaraya Institute of technology, Dr. M S Indira for her constant encouragement and would like to thank Prof. Dilip K Sen, Head of the department, CSE/ISE for his invaluable guidance and suggestions from time to time.

REFERENCES

[1] www.oppapers.com/essays/Voice- Recognition/986291
[2] www.findbiometrics.com/voice-recognition/
[3] www.w3.org/TR/jsgf/
[4] www.tutors161.com/.../java-development-kit-java-software-development-kit-jdk-jsdk/
[5] www.oldapps.com/netbeans.php
[6] ibm-viavoice.software.informer.com/wiki/
[7] www.cloudgarden.com
[8] www.roseindia.net/mysql/

AUTHORS PROFILE

1. Suma Swamy, Assistant Professor, Department of CSE, Sir M. Visvesvaraya Institute of Technology, Bengaluru, India. Research Scholar, Department of ECE, Anna University, Chennai, India. Corresponding Author.
2. Dr. K.V Ramakrishnan, Supervisor, Anna University Chennai, India.
3. Smitha Crystal D' Almeda, Shwetha Rani G, Radha K, Final Year Students, Department of CSE, Sir M. Visvesvaraya Institute of Technology, Bengaluru, India.