Implementation of Bangla Speech Recognition System on Cell Phones

Thesis Report

Supervisor: Prof Dr Mumit Khan

Conducted by: K.M Tasbeer Ahsan

07101003

School of Engineering and Computer Science
BRAC University

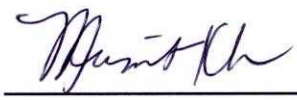
Submitted on 13th April 2011

DECLARATION

I, K.M Tasbeer Ahsan, student of Computer Science and Engineering department, BRAC University represent my thesis work on "Implement of Bangla Speech Recognition System on Cell phones" as requirement of completion of bachelor degree. This thesis research was performed under supervision of Dr. Mumit Khan, Professor, BRAC University, Dhaka, Bangladesh.

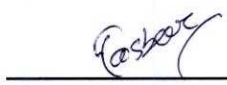
This is to declare that the thesis work was done by me and it has not been submitted before. Help that was taken from Internet and books was mentioned at references.

Signature of Supervisor



Prof. Dr. Mumit Khan

Signature of Author



K. M Tasbeer Ahsan

ACKNOWLEDGEMENTS

I will take this opportunity to express my gratitude to all those who helped and supported me throughout this pre-thesis work. I would like to thank my supervisor Prof. Dr. Mumit Khan to allow and encourage me to work with this project.

I am also very thankful to my friend Shammur Absar Chowdhury who shared her experiences with development of speech recognition system in Bangla. I express my gratitude to Mr. Hammad Ali (Lecturer) who helped me during the recording of the speech corpus for this project.

I would also like to thank my parents, siblings and all my friends for their support.

Abstract

Implementation of Bangla Speech Recognition System on Cell Phones

Speech Recognition refers to the process of converting analogue speech signals into text. Since the 1970s a lot of work has undergone in this particular field. The complex nature of speech due to it's contextual meaning , dialects, accents as well as the environment makes the task of recognizing speech very difficult. A lot of research is currently under progress to increase the accuracy of this system.

Although extensive research has been done on other language, the field of automated speech recognition in Bangla is at it's early level. With almost 200 million Bangla speakers all over the world, there can be a lot of applications of speech recognition. Cellular phone is one such domain , where voice commands can be used to initiate tasks. High accuracy can be achieved as the number of commands are rather limited and since it is dependent upon the user's voice, extensive training will yield better results. If an application is developed for cellular phones that can execute Bangla voice commands, it will make it easier for a large number of disabled people to use cell phones in their own languages. Since a lot of the users are not literate about how to use commands by pressing keys of phones will be facilitated by this application. However, the limited processing power of the cell phones are also a challenging factor.

The aim of this thesis was to show a way to develop an application that will be able to recognize Bangla voice commands for cell phone and execute them followed by an attempt to implement the system according to research findings and analyze it.

TABLE OF CONTENTS

FIGURES & TABLES

Literature Survey

In order to understand how a speech recognition system works and how it can be implemented for Bangla using existing tools, I have done some background research from various sources. Some of the references are explained here in detail whereas the rest is given in the reference section.

Speech and Language Processing (by Daniel Jurafsky , James H. Martin)

The chapters in this book elaborately explains the steps in speech recognition processes, the model building, the algorithms used in recognition etc. Since I am using a HMM model for recognition, this book taught me the basics of HMM and the related calculations of n-gram that is required to build a language model.

Lectures from CMUSpeech

This is an online resource that I used to download lectures given in Carnegie Mellon University. This helped me to understand the basics of feature extraction like how MFCC (Mel Frequency Cepstral Coefficients) are extracted from speech signal, why they are needed etc. It also explains the HMM model in detail.

Implementation of Speech Recognition System for Bangla (Thesis Paper by Shammur Absar Chowdhury)

I used this as a base research paper to understand how a speech recognizer can be developed for Bangla using Sphinx. It describes the issues related with speech recognition, how the language models can be built and step by step instruction in running the trainer and the decoder of Sphinx.

CMU Robust Group Tutorial

From this online tutorial I have learned how to use the trainer and decoder in Sphinx, the explanation of the different parameters that are important to decoding and training, the file formats on how to prepare the language models , language dictionary , phone files etc. It also gives a detailed explanation of how the HMM model is implemented and a probabilistic model can be used.

CMU Sphinx wiki

This online resource lists a number of very useful articles that details out steps to develop an application using PocketSphinx and the training parameters required for a telephony model which is very useful for cell phones

Android Developers Guide

This is the guide I referred to most in order to understand how to develop an application in Android starting from the very beginning. It lists a number of very useful tutorials and other technical resources along with a fantastic documentation on the APIs of Android

SourceForge CMUSphinx Forum

I found most of the solutions to my problems, some which were beyond my guess or ability to comprehend, from this help forum.

In addition to these I have used various other resources and websites which have been referred to the Reference section of this report.

SECTION 1 INTRODUCTION

1.1 Motivation

Continuous Speech Recognition for Bangla is still at it's early level. Although a number of speech recognition system were developed for research purposes, the number of applications of Bangla speech recognition is very small. One such useful application could be for cellular phones to allow voice commands to perform the tasks. It would be a handy tool for the disabled people who are unable to use the cell phone keypads properly. Not only that, a large number of mobile phone users in Bangladesh are not literate as a result using the cell phone is limited to certain tasks. If they could use speech to control the tasks it would be much much easier from them. During the thesis semester, I have tried to develop the language models necessary for this system, build an application for cell phone to use the models to recognize the user's speech and execute the commands followed by an analysis of the system.

1.2 Goal for the thesis

The aim of the thesis is to venture the different methodologies available to implement a continuous speech recognition system for Bangla in cell phones, and proceed to developing an application that will allow the user to initiate tasks like calling a contact, sending short messages, sending emails using speech commands. The first task is to develop a strong language model and train the system to generate the acoustic models which could be used for recognition. Then the next phase involves integrating them in cell phone I.e. developing an application for cell phone that will use those models to execute user commands.

1.3 Outline of the report

The report is divided into three major sections. The first one describes how an acoustic model can be generated for use of a speech recognizer, the available speech recognition engines and the chosen one. In addition to that it also explains ways of developing an application in cell phone. The second section describes the process how the acoustic models were generated in great detail. The section after describes how the application was built for a cellphone and integrated with the acoustic models. The final section sheds light on the performance, problems faced, possible solutions to increase performance and future works to be done.

SECTION 2 EXISTING SYSTEM OVERVIEW AND PROPOSING A PLAN

2.1 Basics of Speech Recognition

By definition, Speech Recognition is referred to the process of convert analog speech signals, captured via microphone or telephony into their corresponding text. This is achieved by training the system using different voices and a large number of different speeches. The system stores the patterns respective to the speech signals so that during recognition phase when an input signal is given it analyses the signal and tries to find the corresponding word observing the pattern.

Speech Recognition can be performed, mainly, in two ways. The first one being Isolated Speech Recognition (ISR) and the second one is Continuous Speech Recognition (CSR). In Isolated Speech Recognition, the user has to pause between each word, for a considerable amount of time, in order for recognition to be performed. On the other hand, in CSR the speaker delivers each sentences without any pauses between the words. CSR resembles to the way how speech is delivered hence in this system CSR is used.

Before progressing into the next sections the reader must get familiarized with the following terms as they will be used throughout the next sections

Speech Corpus: A speech corpus (or spoken corpus) is a database of speech audio files and text transcriptions in a format that can be used to create acoustic models (which can then be used with a speech recognition engine).

Vocabulary: Vocabularies/Dictionaries lists down all the possible words that a SR system is expected to recognize. The size of the vocabulary is dependant on the domain.

Training : Process of learning the characteristics of sound units is called Training. The trainer learns the parameters of the models of sound units using a set of sample speech signals called training database (TD)

Testing/Decoding: Process of extracting patterns from sound units and matching them with patterns learnt in Training phase in order to recognize the word.

Pronunciation Dictionary: Lists down all the valid words in the vocabulary

with mappings of their corresponding sequences of sound unit

Filler Dictionary: Non-Speech sounds are mapped to corresponding non-speech or speech like sound units
Phone : Way of representing the pronunciation of words in terms of sound units. The standard system for representing phones is the International Phonetic Alphabet or IPA. English Language use transcription system that uses ASCII letters where as Bangla uses Unicode letters.

HMM (Hidden Markov Model) : The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multidimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to the associated probability distribution. It is only the outcome, not the state visible to an external observer and therefore states are ``hidden'' to the outside; hence the name

Hidden Markov Model Language Model: assigns a probability to a sequence of m words by means of a probability distribution. To model it we can use a regular grammar.

2.2 Speech Recognition in Machine

The Speech recognition that is performed in machine consists of two phases.

The stages are:

1. Training

2. Decoding/Recognition

2.2.1 Training

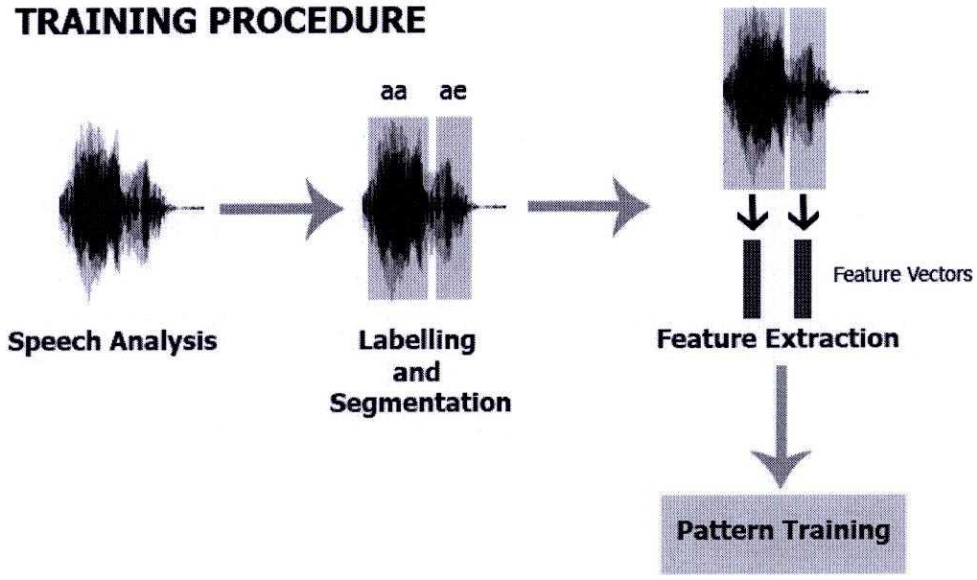


Fig 2.1: The figure above illustrates the phases of the training process in a speech recognition system

The training process involves preparing a sample speech corpus, that consists of a considerable number of sample spoken sentences. Each of these speech are input to the training module. The trainer then performs the labeling and segmentation operation on these input signals. This involves labeling the phonemes and word boundaries of the speech input signals. After features like Mel Frequency Cepstral Coefficients are extracted from the audio input and then are represented in frequency vectors. These vectors are used to train the machine for patterns. These patterns can be phonemes or words. Using these features a model classifier is built to classify any pattern next time.

2.2.2 Recognition

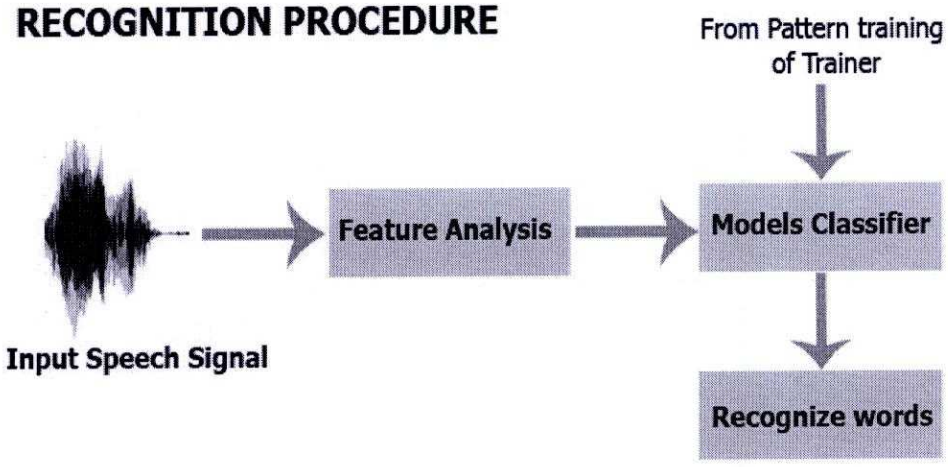


Fig 2.2 : The figure above illustrates the phases of the recognition process in a speech recognition system

In the recognition phase, the system is tested with a number of speech signals that are entirely different from the ones that were used to train the system. From these input the recognizer/decoder performs a feature analysis and extracts the features. From there on using the patterns extracted , the model classifier identifies which phonemes matches these pattern and concatenate them to for individual words.

2.3 Speech Recognition Tools

There are a number of Speech Recognition tools that are available to use and build a recognition system for any language. Each of these alternatives were compared against for factors like availability, performance, documentation, supported platforms and languages. The table below shows the comparison between the available tools

	License	Development Language	Platforms	Support
Julius	**BSD**	**C**	**Linux/Unix, Windows**	**-**
HTK	**Prohibits redistribution and commercial use but R&D allowed**	**C**	**Windows, Linux/Unix, MAC OS X**	**HTK book, Active Mailing List**
Sphinx	**BSD**	**C,JAVA**	**Linux/Unix, Windows, MAC OS X**	**Very well documented**
ISIP ASR	**Public domain (no restrictions)**	**C++**	**Windows**	**-**

Table 1: Comparison between different Speech Recognition Tools that are available

By considering the factors mentioned above and few others Sphinx was chosen for this thesis. The reasons are :

- Runs on any platform
- Very well documented. Help is available from SourceForge group.
- Development language is C and Java both of which is very familiar
- BSD license available , so it can be used for free
- Sphinx has a decoder called PocketSphinx that is specifically designed for

cell phones and hand held devices and also runs in desktop

- Implementation of a Bangla SR in Sphinx for a different domain exists.

2.4 Implementation paradigms for cell phones

There are a number of ways in which the speech recognition application can be developed for cell phones. By analyzing them, two methods were found to be feasible given the time line of the project.
The two approaches are:
1.Performing the Training and Decoding in a web server and connecting the application with the server.
2.Performing the training and loading the models on the phone and use an existing decoder for recognition locally.

The two approaches will now be discussed in detail

2.4.1 Server side implementation

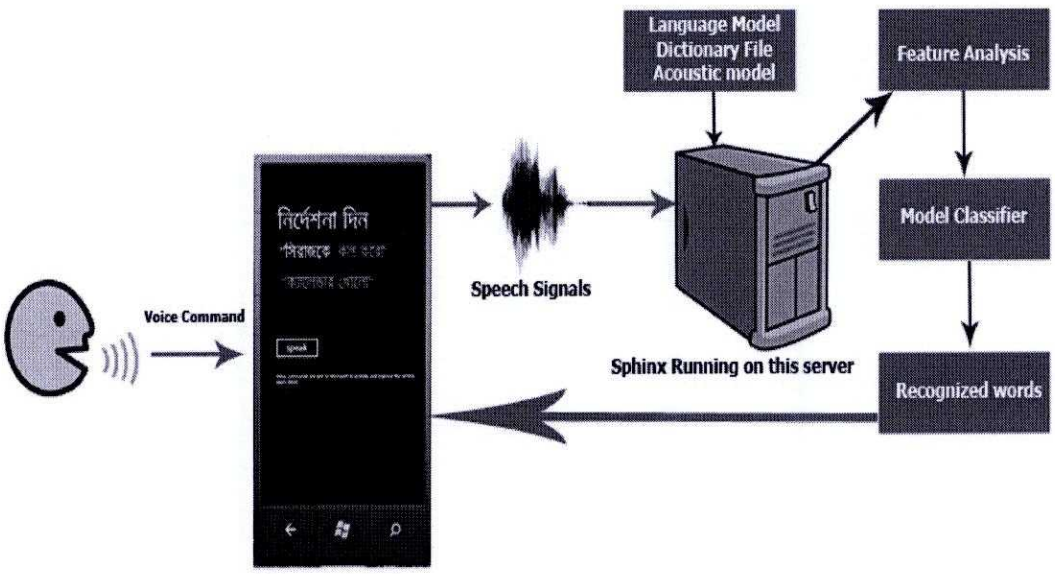


Fig 2.3 : Developing an application that will forward speech signals and recognition will be performed on a server and returned back

In this methodology, the application running on the cell phone will just take the user voice input and redirect the speech signal to a server using internet where Sphinx trainer and decoder is running. Feature analysis will be performed

within the server and using the language model, acoustic model and the dictionary file the decoder will recognize the words and send it back to the cell phone where the text will be parsed to understand the command and execute accordingly.

This method has quite a few advantages. The first is that it is independent of processing power of the cell phones. Almost any cell phone that is using internet can use this application. The second advantage is that training can be performed for contacts of names and a database of names can be stored in the server so it can use a wide range of . Existing telephony servers softwares are present (ASTERISK) to help connect a cell phone to a server and there is an implementation of a speech recognizer using ASTERISK and Sphinx. However the implementation uses an old version of Sphinx.

2.4.2 Training on computer and decoding using PocketSphinx

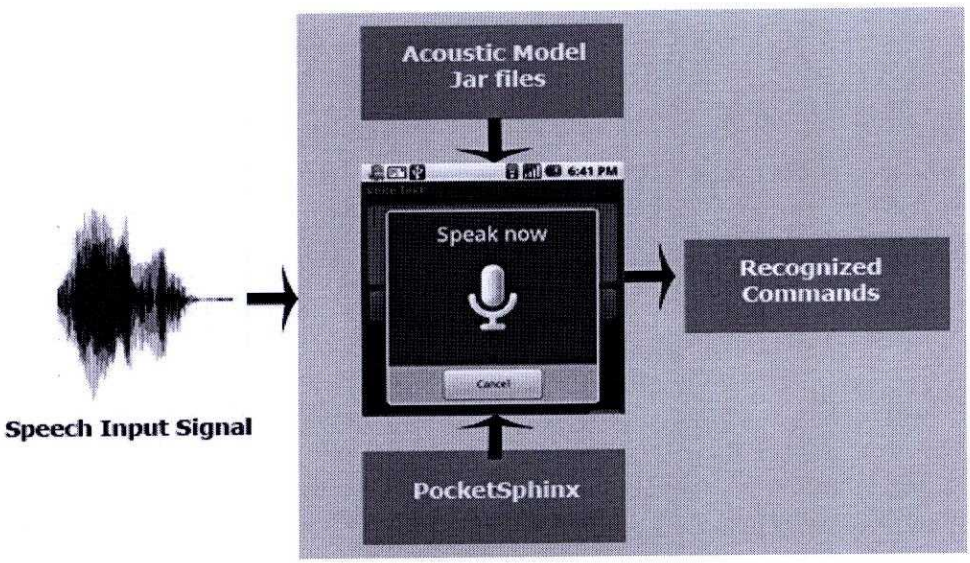


Fig 2.4 : An application that will integrate PocketSphinx decoder and recognize speech using the acoustic models provided

After doing a bit of research in the internet and the Sphinx groups, it was found that the acoustic models can be exported and PocketSphinx can be integrated in handheld devices. Considering the availability of NDK (Native development Kit) of the Android OS for cell phones and the availability of documents on ways to integrate the jar files and the decoder into android, decided that it will be a good option for developing the application. The Android NDK is a companion tool to the Android SDK that allows to build performance critical portions of applications in native code (in this case JAVA will be used). It

provides headers and libraries that allow to build activities, handle user input, use hardware sensors, access application resources. Acoustic models will be generated for 8-10 different speakers beforehand and the jar files copied to the cell phone. After integration with PocketSphinx it can decode the commands.

However, most of the replies in the group showed that very few succeeded in integrating pocketsphinx in Android phones. Accuracy rate is also not very high. Another problem exists is that of training. For instance, if one want's to dial a particular contact the SR has to be trained for the names in the users contact list. The processing power of cell phones is also a concern.

2.5 Final Choice

After analyzing the alternatives and their pros and cons, I have decided to develop the application using the approach mentioned in section 2.4.2. The reasons are mentioned in the sections above. The application will be developed in the following way :

- Train the SR system and generate acoustic models
- Integrate PocketSphinx with android
- Load the models into the android device
- Add modules that will be able to parse recognized words and carry out actions

SECTION 3 ACOUSTIC MODEL GENERATION

This part of the report will describe the procedures undertaken to generate the acoustic models for the system. The steps are described in detail, explaining the rational behind their design, how they were designed and using what tools.

3.1 Preparing the Data

In order to perform training the language dictionaries, phone files etc. had to be prepared beforehand for training. The steps are given below

Speech Corpus

In order to train the system in Sphinx, it is necessary to record some sample audio files consisting of sentences relative to this domain (in this case the sentences are voice commands used in cell phones). In order to be an acoustic model to be accurate it is necessary that the system is trained using recordings of same sentences of different speakers of different ages. This ensures that the system is trained on different utterances and hence be able to recognize more easily. In order to train the system I have built a speech corpus by recording 126 sentences of giving commands in different ways in the voice of 8 different speakers, 6 of whom were male and the rest were female of the age range 20-30. The recording parameters that were used are:

Bit rate (bits per sample) : 8
Sampling rate of the audio: 8 Khz (Standard for telephony model)
Channel: Mono (Single Channel)

The files were stored with the initials of their name followed by the number of the sentence. e.g t1.wav for Tasbeer's first recorded sentence and so on. For better training the noise conditions were varied from reasonably silent to high level of noise.

Transcription file

This file basically contains the text forms of each of the sentences in every .wav file on every new line. The format for writing the text is like this:
<s> Text in Bangla </s> (<file_name>.wav).
The transcription file is named as <Project_Folder_Name>_train.transcription

Example:

<s> ফোনবুকে খুঁজো </s> (h5)
<s> নাম্বার তালিকায় যাও </s> (h6)
<s> নাম্বার তালিকা দেখাও </s> (h7)
<s> নাম্বার তালিকায় দেখো </s> (h8)
<s> নাম্বার তালিকাতে যাও </s> (h9)
<s> নাম্বার তালিকাতে দেখো </s> (h10)

Dictionary File

This file consists of all the possible words that are accepted in this domain. Every new line consists of one such word and then a space, followed by the pronunciation of that word in ASCII. Bangla words are written in unicode. However Sphinx does not support unicode pronunciations. That is why the words were first mapped to IPA (International Phonetic Alphabet) pronunciations, from their to ASCII pronunciation. The Bangla word to IPA mapping was done using a tool from CRBLP (Center for Research on Bangla Language Processing) website. However, the IPA form of all the words were not available. For this reason another tool called G2P (grapheme to Phoneme) was used to obtain the IPA symbols. After all the IPA symbols were retrieved a script was run to get the corresponding ASCII pronunciation. It has to be kept in mind that no word can be repeated.
A total of 100 words were chosen that can identify 31 distinct commands when used in combination.
The file was
named as <Project_Folder_Name>.dic. The words were arranged in the following manner:

ন্তর্জাল o n ta o r j aa l
অন্তর্জালে a n ta o r j aa l e
আঁট aa~ t o
আট aa t o
ইন্টারনেট i n t oa r o n e t
ইন্টারনেটে i n t oa r o n e t e
এ e
এক ae k
এট e t o
এলার্ম e l aa r m o
এলার্মে e l aa r m e
এসএমএস e s a ae m a e s
এসএমএসে e s o ae m a e s e

ওয়াইফাই o y aa i ph aa i
কমাও k a m aa o
করো k a r o
কল k a l
কলের k a l e r
কাটো k aa t o
কারিগরী k aa r i g o r i
কিপ্যাড k i p p aa d o
ক্যামেরা k ae m e r aa
খুঁজো kh u~ j o
খুলো kh u l o
খুঁজো kh u~ j o
খোঁজো kh o~ j o
খোলো kh o l o
গান g aa n
ঘোষণা gh o sh a n aa
চার c aa r
চালাও c aa l aa o
চালু c aa l u
চিত্র c i ta ta r o
চিত্রগ্রহন c i ta ta r o g g r o h o n
ছবি ch o b i
ছাড়ো ch aa ra o
ছয় ch a y
তথ্য ta o ta tah
তথ্যতে ta o ta tah e
তালিকা ta aa l i k aa
তালিকাতে ta aa l i k aa ta e
তালিকায় ta aa l i k aa y
তিন ta i n
তুলো ta u l o
তোলো ta o l o
দাও da aa o
দি da i
দুই da u i
দেখাও da ae kh aa o
দেখো da ae kh o
দ্যা da ae

ধরো dah a r o
নতুন n o ta u n
নম্বর n a m b o r
নম্বরে n a m b a r e
নাম্বার n aa m b aa r
নাম্বারে n aa m b aa r e
নিরাপত্তা n i r aa p a ta ta aa
নেটওয়ার্ক n ae t o y aa r k o
নয় n a y
পাঁচ p aa~ c
পাঠাও p aa th aa o
প্লাস p l aa s
পড়ো p a ra o
ফোন ph o n
ফোনবুক ph o n b u k
ফোনবুকে ph o n b u k e
বন্ধ b a n dah o
বাতিল b aa ta i l
বার্তা b aa r ta aa
বাড়াও b aa ra aa o
ব্যবস্থাপনা b ae b o s tah aa p o n aa
ব্যবস্থাপনায় b ae b o s tah aa p o n aa y
ব্যবস্থাপনাতে b ae b o s tah aa p o n aa ta e
ব্লুটুথ b l u t u tah
ভিডিও bh i d i o
মেইল m e i l
মেইলে m e i l e
মেনু m ae n u
মেনুতে m ae n u ta e
মেসেজ m e s e j
যাও j aa o
যোগাযোগ j o g aa j o g
রিসিভ r i s i bh
রেট r ae t
লিখো l i kh o
লিস্ট l i s t o
লেখো l e kh o
শব্দ sh o b da o

শুন্য sh u n n o
শুরু sh u r u
সংকেত sh a ng k e ta o
সময় sh a m a y
সাত sh aa ta o
সেভ s ae bh
সেটিংস s e t i ng s o
সেটিংসেs e t i ng s e
সেন্ড s e n d o
স্টার s t oa r
হ্যাশ h ae sh

Phone File

This file consists of all the possible phones in ASCII for this domain (without repetition). It was named as <Project_Folder_Name>.phone e.g

a

aa

ae

a~

Language Model

The language model, LM file, describes the likelihood, probability taken when a sequence or collection of words is seen. This information is used to constrain search and as a result significantly improve recognition accuracy. The language model file is plain text. The format is the commonly used "arpa" format which is standard in speech recognition research. It lists 1-,2- and 3-grams along with their likelihood (the first field) and a back-off factor (the third field).To build this file, CMU lmtoolkit is used. lmtool is a web based tool that allows users to quickly compile two text-based components needed for using an ASR decoder. To do this, a corpus is needed, which in this case means a set of sentences (or more precisely, utterances) that is expected for recognition system to be able to handle (i.e., what people might reasonably want to say to the system). The corpus needs to be in the form of an ASCII text file but with new advanced version Unicode text file is also supported, with one sentence to a line. Upload this file, click the compile button. This will give a set of lexical (pronunciation dictionary) and language modeling files. Here the only file used is LM file as Pronunciation dictionary should be built as stated above. It is named as

ASR_mobile_demo.lm. The language model is used during the decoding/testing phase. E.g:

2.3871 ওয়াইফাই -0.2861
-3.5011 কমাও -0.2574
-2.1788 করো -0.2574
-2.1788 কল -0.2861
-2.4219 কলের -0.2905

Filler Dictionary

Dictionary where a non speech sounds are mapped to correspondingnon speech sound units. This file is named as <Project_Folder_name>.filler

e.g. <s> SIL <sil> SIL </s> SIL

Control File

This is a text file consisting of the names of each of the wav files. e.g.

t1

t2

t3

...

It was named as <Project_Folder_Name>_train.fileids.

3.2 Setting up the Trainer

The platform that was used for this testing were Linux based.

First of all SphinxTrain (the trainer) was downloaded from

http://sourceforge.net/projects/cmusphinx/files/

the file was then extracted using the command

> $ gunzip -c SphinxTrain.nightly.tar.gz | tar xf -

The trainer was built by navigating to the SphinxTrain folder and

writing

$ ./configure

$ make

3.3 Setting up the project folder

A project folder was created with the name crblp_asr and within it two sub-folders were created named “etc” and “wav”. Inside the wav folder another directory called “train_wav” was created which contains all the training data. The transcription file, control file, language model, dictionary file, phone file, filler file were copied to the “etc” folder. After this the following command was run in terminal:

> $../SphinxTrain/scripts_pl/setup_SphinxTrain.pl -task crblp_asr

Two new files were created automatically in the etc folder : feat.params and sphinx_train.cfg

The following changes were made to the sphinx_train.cfg file:

1.$CFG_WAVFILE_EXTENSION = 'wav'

2. $CFG_WAVFILE_TYPE = 'raw'

3. States per HMM will be 3

4.$CFG_FEATURE will be "1s_c_d_dd"

5.$CFG_FINAL_NUM_DENSITIES = 1

6. $CFG_N_TIED_STATES = 100

The following lines were added to the “feat.params” file

-samprate 8000.0

-dither yes

-alpha 0.97

-doublebw no

-nfilt 31

-ncep 13

-lowerf 200.0

-upperf 3500.0

-nfft 256

-wlen 0.0256

The folder called “scripts_pl” is opened and then the lines 84 to 93 is replaced by the following lines

-samprate 8000.0

-dither yes

-alpha 0.97

-doublebw no

-nfilt 31

-ncep 13

-lowerf 200.0

-upperf 3500.0

-nfft 256

-wlen 0.0256

This is done so that the features that are extracted from the wav files follow the parameters required for a model of 8K sampling rate
Now from the raw speech (wav) files the features are extracted using

the following command

$perl scripts_pl/make_feats.pl -ctl etc/ASR_mobile_demo_train.fileids

3.4 Training

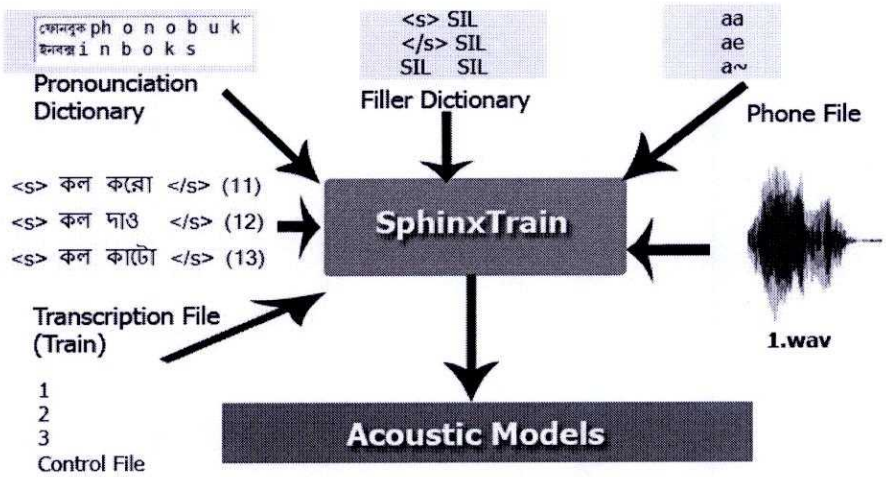


Fig 3.1 : Training to generate acoustic models

In order to perform training the following steps were executed:

1.perl scripts_pl/00.verify/verify_all.pl

This scripts verify whether all the files are in correct format, checks if there is any mismatch in transcription file, phone file and dictionary files, detects extra space etc.

2. perl scripts_pl/01.vector_quantize/slave.VQ.pl

This script is only needed to execute if the training is given for continuous model

3. perl scripts_pl/20.ci_hmm/slave_convg.pl

This script can give error messages if the dictionary have any duplicate entries but while data preparation this problem should be taken care.

4. perl scripts_pl/30.cd_hmm_untied/slave_convg.pl

5. perl scripts_pl/40.buildtrees/slave.treebuilder.pl

6. perl scripts_pl/45.prunetree/slave.state-tying.pl

7. perl scripts_pl/50.cd_hmm_tied/slave_convg.pl

8.perl scripts_pl/90.deleted_interpolation/deleted_interpolation.pl

During the training process several new directory has been created which contains files that are going to be included in the acoustic model

The models that are required can be found in the "model_parameters" folder in the project folder as shown below.

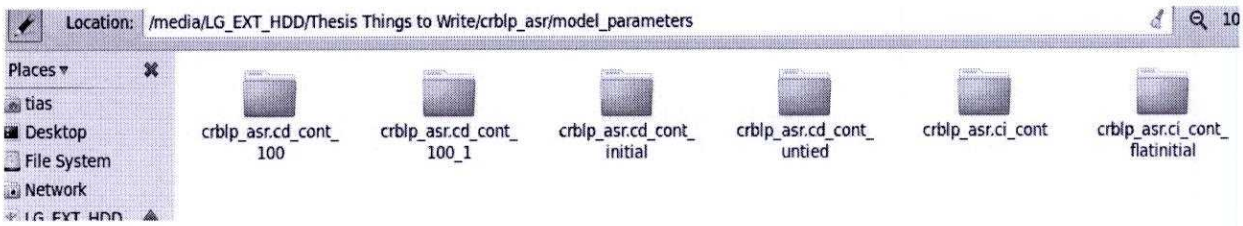


Fig 3.2 : Models to be used

For the purpose of this project the models on folder "crblp_asr_ci_cont and "crblo_asr.cd_cont_100" will be used.

Both the folders has the following files

-feat.params
-mdef
-means
-mixture_weights
-noisedict
-transition_matrices
-variances

In addition to these two more files are required for pocketsphinx. One is the mdef.txt file which is a text version of the mdef file and a "sendump" file which basically consists the data from means and mixture_weights together.

In order to generate the mdef.txt file the following line has to be executed in the terminal

> pocketsphinx_mdef_convert -text <model_directory>/mdef <model_directory>/mdef.txt

In order to generate the "sendump" file, first the program file called "mk_s2sendump" has to be copied to the model directory. It can be found in the directory called bin.i686-pc-linux-gnu or bin-x86_64-unknown-linux-gnu (on Linux)

or in bin\Debug or bin\Release (on Windows) inside the SphinxTrain directory. Then navigate to the model directory and execute this command

```
./mk_s2sendump \
-pocketsphinx yes \
-moddeffn mdef.txt \
-mixwfn mixture_weights \
-sendumpfn sendump
```

The models are now ready to be used by pocketsphinx.

SECTION 4 RUNNING POCKETSPHINX ON ANDROID

4.1 Setting up PocketSphinxAndroidDemo Project

The developers of Sphinx have provided a sample demo project for pocketsphinx which is basically a rewrite of the recognition code in JAVA use with Android, that was previously written in C . The user has to buildthe necessary java files on their own using the swig interface of pocketsphinx. The next sections describe how to do so.

4.1.1 Setting up Android SDK

The project was run in Eclipse. For this purpose the "Eclipse Classic" version has to be downloaded from http://www.eclipse.org/downloads/

After installing Eclipse on the computer, the android ADT plugin for Eclipse has to be downloaded from http://developer.android.com/sdk/eclipseadt.html In this project version 10is used.

The next step is to download the android sdk from http://developer.android.com/sdk/index.html

A tutorial on how to set this up to run Android is provided in the links mentioned above. It has to be followed to develop to setup the environment.

4.1.2 Importing PocketSphinxAndroidDemo

PocketSphinxAndroidDemo Project can be downloaded from the following link: http://cmusphinx.svn.sourceforge.net/viewvc/cmusphinx/trunk/PocketSphinxAndroidDemo/

After downloading the Project it has to be imported into Eclipse. After opening the project it would look like this. Eclipse will show a lot of errors but they have to be ignored until the libraries are built.

The 'src' folder consists of the following 3 files

–PocketsphinxAndroidDemo.java

–

This file basically does the task of creating a button and a text viewfor the android app so that user can start recording their speech and see the outcome of recognition. It records the sound at a sampling rate of 8K and then writes it to a block memory queue and then forwards itto

the RecognizerTask for recognition which in turn returns the result toit

–RecognitionListener.java

This is the interface file for the RecognizerTask file

–RecognizerTask.java

It stores the speech in a block in queue and then creates a configuration object using the acoustic models to generate the required features for recognition. Then it takes the input speech which it saved in raw format , process it for extraction of features and then perform pattern matching to recognize the word using language model and dictionary

4.1.3 Building the libraries

In order to run the project the files that are written in C that are used for recognition in pocketsphinx has to be imported via the JNI (java native interface). This is achieved in the following way:

1. Download a pocketsphinx snapshot and sphinxbase-0.6 from http://sourceforge.net/projects/cmusphinx/files/

2. Navigate to the SphinxBase directory and execute the following command
   % ./configure
   % make

   % make install

   %export LD_LIBRARY_PATH=/usr/local/lib

   %export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig

3. Change to the pocketsphinx directory and install using the following commands

   % ./configure
   % make
   % make install

4. Open pocketsphinx directory and navigate to the “swig” folder

5. Open the Android.mk file and change the following parameters

   SPHINX_PATH := $(HOME)/Projects/Sphinx/trunk

   to the root directory where pocketsphinx and sphinxbase is installed
   and save the file

6. Run %make command on the swig folder.

7. Several files called Config.java, Decoder.java etc are generated. Copy all of them and then paste it in the jni folder of PocketSphinxAndroidDemo project under the directory edu/cmu/pocketsphinx/demo (if the directories doesn't exist one has to be made like this)

Now the project is ready to be run. The project should be tried to build now. If an "External Builder tool" error appears then the NDK build and SWIG has to be uncheked from the Java Build Path in eclipse.

4.2 Loading the models into sdcard

An sdcard image has to be made manually in order to load the models. This can be done following the steps below

1. An sdcard image has to be created only once. First navigate to the android_sdk installation directory and to the 'tools' subdirectory from the terminal.
2. Execute this command in the terminal
   ./mksdcard 128M my128MbCard

3. The image can now be mounted as a loopback device. First createa folder called 'mycard' inside the "/media" directory. Then using the following command :
   sudo mount -o loop <Card_name> \media\mycard

4. Once mounted the card can be accessed as a device.

5. In order to load the models first a folder called "Android" has to be created. Within it a subfolder called "data", within which another folder called "edu.cmu.pocketsphinx" has to be created. Now the files from the 'models' folder has to be copied to Android/data/edu.cmu.pocketsphinx.

6. Then open location '/hmm/en_US/hub...' and replace the models with the ones that were generated in the crblp_asr/model_parameters folders.

7. Navigate to the crblp_asr/etc folder and then run the following :

sphinx_lm_convert -i crblp_asr.lm -o crblp_asr.DMP

8. Copy 'crblp_asr.dic' and 'crblp_as'r.DMP' files to Android/data/edu.cmu.pocketsphinx/lm/en_US/ and rename them to 'hub4.5000.dic' and 'hub4.5000.DMP' files

9. Copy the 'Android' folder and it's contents to the /media/mycard folder and unmount

10. Create an android virtual device using AVD manager in eclipse. Change the API level to 8, in the place of sdcard browse to the place where the sdcard image was created (android_sdk_installation_folder/tools). Add audio record and play back support and save.

11. Upon running the application now the emulator will load like this:

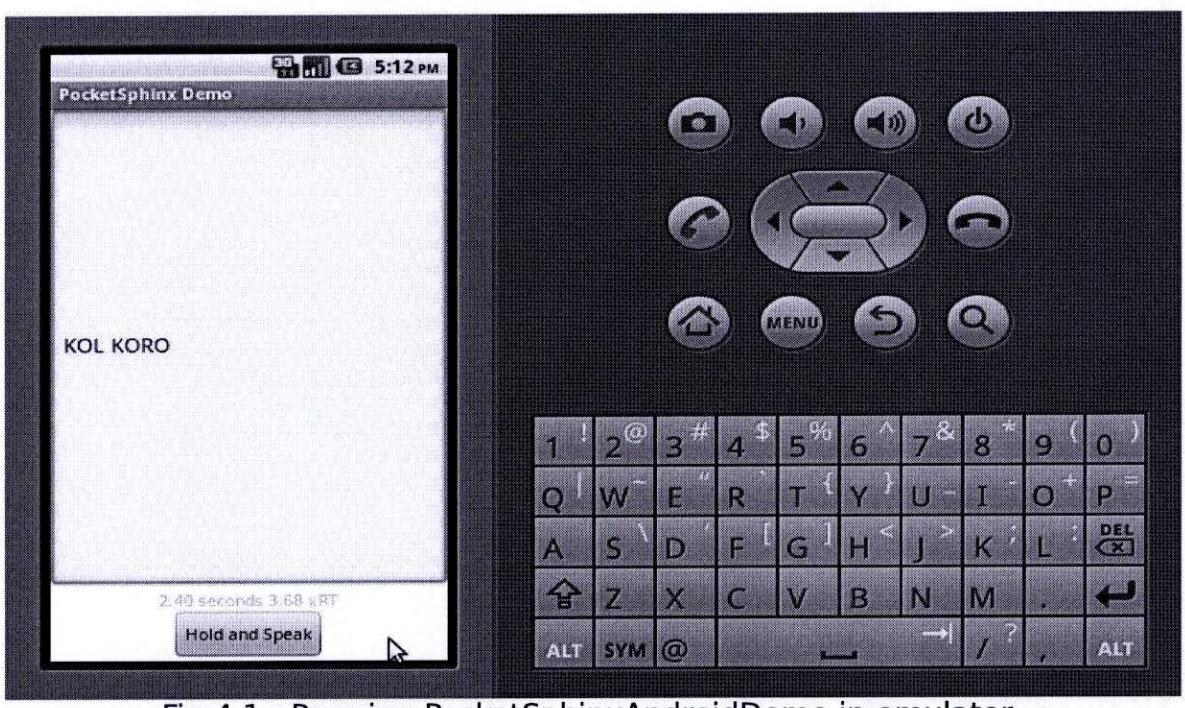


Fig 4.1 : Running PocketSphinxAndroidDemo in emulator

4.3 Adding a method to parse the recognized word

The final step is to add methods to the PocketSphinxDemo.java file to parse the recognized string and carry out the command. In order to test the application a simple method was added that would take in the recognized string as a parameter check if it contains the word "phone" or "number" and open the phonebook or the call menu. The code is shown below.

```
public void execCommand (String cmd){
        Uri mSmsinboxQueryUri = Uri.parse("content://sms/inbox");
        Intent intent;
        if(cmd.contains("???")){
                intent = new Intent(Intent.ACTION_VIEW,
ContactsContract.Contacts.CONTENT_URI);
                startActivity(intent); }
        }else if(cmd.contains("???????") || cmd.contains("????????")){
                intent = new Intent(Intent.ACTION_VIEW,
CallLog.Calls.CONTENT_URI);
                startActivity(intent);
        }

}
```

SECTION 5 ERRORS FACED, PERFORMANCE, FUTURE WORKS

5.1 Errors Faced

Error 1: Unable to create the Config.java, Decode.java etc files in the pocketsphinx snapshot folder using swig

Reason: While installing the SphinxBase, the pkg_config_path variable has to be set and exported, otherwise the swig interface does not work properly

Solution: The variables are exported using the following command
%export LD_LIBRARY_PATH=/usr/local/lib
%export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig

Error 2: Linux doesn't recognize the sphinx_lm_convert tool to convert the .lm files to .DMP files

Reason: The older versions of SphinxBase had bugs with sphinx_lm_convert which was solved in the new releases.

Solution: Replace the older SphinxBase files with the new files

Error 3: PocketSphinxAndroidDemo fails to load when run and shows "External tools build error" in Eclipse

Reason: There were two external tools added to the Java Build Path that is not required.

Solution: remove the NDK build tool and SWIG tool from the Java Build Path

Error 4: Pocketsphinx app crashes when loaded with a model

Reason: While training the models the upperf parameter was set to 4000 according to the nyquist frequency. However for the models to operate smoothly the frequency has to be 3500.

Solution: during training the "upperf" parameter value in the feat.params file was changed to 3500.

5.2 Performance

The application runs successfully and records speech and shows results. However the accuracy is almost equals to zero. The only word it managed to decode with success was "Phonebook". The training was done again with repetitions of same voice and utterance but the results didn't improve much. The accuracy rate achieved was 30.28%. The results of a test is given below.

Test Results

ফোনবুক খুলো (USER-TEST_1)
চালু করো (USER-TEST_1)
Words: 2 Correct: 0 Errors: 2 Percent correct = 0.00% Error = 100.00% Accuracy = 0.00%
Insertions: 0 Deletions: 0 Substitutions: 2
মেনুতে যাও (USER-TEST_2)
*** বার্তা (USER-TEST_2)
Words: 2 Correct: 0 Errors: 2 Percent correct = 0.00% Error = 100.00% Accuracy = 0.00%
Insertions: 0 Deletions: 1 Substitutions: 1
কল করো (USER-TEST_3)
কল করো (USER-TEST_3)
Words: 2 Correct: 2 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%
Insertions: 0 Deletions: 0 Substitutions: 0
কল রিসিভ করো (USER-TEST_4)
কল তথ্য পড়ো (USER-TEST_4)
Words: 3 Correct: 1 Errors: 2 Percent correct = 33.33% Error = 66.67% Accuracy = 33.33%
Insertions: 0 Deletions: 0 Substitutions: 2
ইন্টারনেট খুলো (USER-TEST_5)
তালিকা খুলো (USER-TEST_5)
Words: 2 Correct: 1 Errors: 1 Percent correct = 50.00% Error = 50.00% Accuracy = 50.00%
Insertions: 0 Deletions: 0 Substitutions: 1
কল লিস্ট দেখাও (USER-TEST_6)
*** তালিকা দেখাও (USER-TEST_6)
Words: 3 Correct: 1 Errors: 2 Percent correct = 33.33% Error = 66.67% Accuracy = 33.33%
Insertions: 0 Deletions: 1 Substitutions: 1
নাম্বার তালিকা দেখাও (USER-TEST_7)
নাম্বার তালিকা দেখাও (USER-TEST_7)
Words: 3 Correct: 3 Errors: 0 Percent correct = 100.00% Error = 0.00% Accuracy = 100.00%

Insertions: 0 Deletions: 0 Substitutions: 0
শুন্য এক সাত এক সাত দুই চার তিন নয় নয় পাঁচ নাম্বারে কল করো (USER-TEST_8)
শুন্য এক চার পাঁচ পাঁচ দুই *** *** *** সাত বার্তা পাঠাও কল করো (USER-TEST_8)
Words: 14 Correct: 5 Errors: 9 Percent correct = 35.71% Error = 64.29% Accuracy = 35.71%
Insertions: 0 Deletions: 3 Substitutions: 6
শুন্য এক *** আট এক নয় পাঁচ আট দুই চার সাত শূন্য নাম্বারে কল করো (USER-TEST_9)
শুন্য এক আট আট এক নয় পাঁচ পাঁচ দুই চার *** চার চার বার্তা দেখাও (USER-TEST_9)
Words: 14 Correct: 8 Errors: 7 Percent correct = 57.14% Error = 50.00% Accuracy = 50.00%
Insertions: 1 Deletions: 1 Substitutions: 5
স্টার আট তিন দুই ছয় হ্যাশ এ কল করো (USER-TEST_10)
পাঁচ আট শুন্য দুই *** *** চার পাঁচ পাঠাও (USER-TEST_10)
Words: 9 Correct: 2 Errors: 7 Percent correct = 22.22% Error = 77.78% Accuracy = 22.22%
Insertions: 0 Deletions: 2 Substitutions: 5
নেটওয়ার্ক ব্যবস্থাপনাতে যাও (USER-TEST_11)
*** বার্তা ব্যবস্থাপনা (USER-TEST_11)
Words: 3 Correct: 0 Errors: 3 Percent correct = 0.00% Error = 100.00% Accuracy = 0.00%
Insertions: 0 Deletions: 1 Substitutions: 2
নিরাপত্তা মেনুতে যাও (USER-TEST_12)
বার্তা মেনু দেখাও (USER-TEST_12)
Words: 3 Correct: 0 Errors: 3 Percent correct = 0.00% Error = 100.00% Accuracy = 0.00%
Insertions: 0 Deletions: 0 Substitutions: 3
নিরাপত্তা ব্যবস্থা খুলো (USER-TEST_13)
*** ব্যবস্থাপনা খুলো (USER-TEST_13)
Words: 3 Correct: 1 Errors: 2 Percent correct = 33.33% Error = 66.67% Accuracy = 33.33%
Insertions: 0 Deletions: 1 Substitutions: 1
সময় সংকেত ঘোষণা করো (USER-TEST_14)
সময় *** সাত পাঠাও (USER-TEST_14)
Words: 4 Correct: 1 Errors: 3 Percent correct = 25.00% Error = 75.00% Accuracy = 25.00%
Insertions: 0 Deletions: 1 Substitutions: 2
এলার্ম সংকেত দাও (USER-TEST_15)
*** পাঠাও দাও (USER-TEST_15)
Words: 3 Correct: 1 Errors: 2 Percent correct = 33.33% Error = 66.67% Accuracy = 33.33%
Insertions: 0 Deletions: 1 Substitutions: 1
ব্লুটুথ খুলো (USER-TEST_16)
ব্লুটুথ খোলো (USER-TEST_16)
Words: 2 Correct: 1 Errors: 1 Percent correct = 50.00% Error = 50.00% Accuracy = 50.00%
Insertions: 0 Deletions: 0 Substitutions: 1

*** শব্দ বন্ধ করো (USER-TEST_17)
সাত এ কল করো (USER-TEST_17)
Words: 3 Correct: 1 Errors: 3 Percent correct = 33.33% Error = 100.00% Accuracy = 0.00%
Insertions: 1 Deletions: 0 Substitutions: 2
নাম্বার তালিকাতে খোঁজো (USER-TEST_18)
নাম্বার তালিকা খোলো (USER-TEST_18)
Words: 3 Correct: 1 Errors: 2 Percent correct = 33.33% Error = 66.67% Accuracy = 33.33%
Insertions: 0 Deletions: 0 Substitutions: 2
নাম্বার তালিকাতে খুঁজো (USER-TEST_19)
নাম্বার তালিকাতে খূঁজো (USER-TEST_19)
Words: 3 Correct: 2 Errors: 1 Percent correct = 66.67% Error = 33.33% Accuracy = 66.67%
Insertions: 0 Deletions: 0 Substitutions: 1
কিপ্যাড খুলো (USER-TEST_20)
কিপ্যাড খোলো (USER-TEST_20)
Words: 2 Correct: 1 Errors: 1 Percent correct = 50.00% Error = 50.00% Accuracy = 50.00%
Insertions: 0 Deletions: 0 Substitutions: 1
ফোন নাম্বার দেখাও (USER-TEST_21)
*** প্লাস আট (USER-TEST_21)
Words: 3 Correct: 0 Errors: 3 Percent correct = 0.00% Error = 100.00% Accuracy = 0.00%
Insertions: 0 Deletions: 1 Substitutions: 2
ইন্টারনেট এ যাও (USER-TEST_22)
নাম্বারে কল দাও (USER-TEST_22)
Words: 3 Correct: 0 Errors: 3 Percent correct = 0.00% Error = 100.00% Accuracy = 0.00%
Insertions: 0 Deletions: 0 Substitutions: 3
চিত্র তুলো (USER-TEST_23)
তথ্য খোলো (USER-TEST_23)
Words: 2 Correct: 0 Errors: 2 Percent correct = 0.00% Error = 100.00% Accuracy = 0.00%
Insertions: 0 Deletions: 0 Substitutions: 2
*** চিত্রগ্রহণ মেনুতে যাও (USER-TEST_24)
চিত্র তোলো কল দাও (USER-TEST_24)
Words: 3 Correct: 0 Errors: 4 Percent correct = 0.00% Error = 133.33% Accuracy = -33.33%
Insertions: 1 Deletions: 0 Substitutions: 3
নম্বর তালিকা দেখাও (USER-TEST_25)
নাম্বার তালিকা দেখাও (USER-TEST_25)
Words: 3 Correct: 2 Errors: 1 Percent correct = 66.67% Error = 33.33% Accuracy = 66.67%
Insertions: 0 Deletions: 0 Substitutions: 1
ভিডিও ব্যবস্থাপনাতে যাও (USER-TEST_26)

*** নিরাপত্তা পাঠাও (USER-TEST_26)
Words: 3 Correct: 0 Errors: 3 Percent correct = 0.00% Error = 100.00% Accuracy = 0.00%
Insertions: 0 Deletions: 1 Substitutions: 2
নেটওয়ার্ক মেনুতে যাও (USER-TEST_27)
বার্তা মেনু দেখাও (USER-TEST_27)
Words: 3 Correct: 0 Errors: 3 Percent correct = 0.00% Error = 100.00% Accuracy = 0.00%
Insertions: 0 Deletions: 0 Substitutions: 3
নতুন মেসেজ লিখো (USER-TEST_28)
*** বার্তা তথ্য (USER-TEST_28)
Words: 3 Correct: 0 Errors: 3 Percent correct = 0.00% Error = 100.00% Accuracy = 0.00%
Insertions: 0 Deletions: 1 Substitutions: 2
গান তালিকা দেখাও (USER-TEST_29)
কল তালিকা দেখাও (USER-TEST_29)
Words: 3 Correct: 2 Errors: 1 Percent correct = 66.67% Error = 33.33% Accuracy = 66.67%
Insertions: 0 Deletions: 0 Substitutions: 1
TOTAL Words: 109 Correct: 36 Errors: 76
TOTAL Percent correct = 33.03% Error = 69.72% Accuracy = 30.28%
TOTAL Insertions: 3 Deletions: 15 Substitutions: 58

Possible reasons for such performance

- Accuracy of detection reduces with sampling rate. The performance is much better for a 16K model as the loss of information is lower compared to 8K models where the utterances of closely related words may overlap. In addition to that the noise filters have to be reduced for 8K model which also results in inaccurate models

- For good acoustic models it is necessary to have a training data of 2 hours. In this case the training data was only of 45 minutes. This may also be a reason for such low accuracy rate

- For the trainer to label and split the wav files accurately it is better to have each sentences used in recording to have a length of greater than 5 seconds and less than 30 seconds. However due to small length of commands in this system the average length was 3-3.5 seconds.

- In order to build a good language model it is necessary to have a sufficiently large corpus. In this case however no such corpus is present as a result it had to be hand made and may not be good enough.

- The decoding parameters used in PocketSphinx are reduced to a value

that enables faster recognition hence compromising with accurate recognition in most cases. This is also a reason for low accuracy rate of the system

5.3 Future Work

- A major part of future work would include improving the accuracy of the existing acoustic model. One way can be by limiting the dictionary size to 40 specific commands and then performing rigorous training with more than 20 different speakers and checking for perfect utterance of words.

- In this system a probabilistic approach, that calculates the likelihood of a word using bigram technique, was used to develop the language model. However, another approach exists where a rule based grammar is developed to be used as a language model to guess a word from their sequence of utterances. Since the set of commands are small, this can be particularly useful in this case and can be tried in future. Although it has the obvious disadvantage that introduction of new word in the dictionary will mean changing of some rules if not many

- In order to improve accuracy it is best to train the system using the user's own voice. However there is no lightweight trainer available yet for running with limited processing power of handheld devices like cell phones. One way can be training the system on a server via the cell phone where the server can take the training speech from the user via the cell phone and then generate the acoustic models which in turn will be sent to the user's phone where the existing model will be updated. However the level of complexity of such system will be very high.

- Some tuning can be performed on PocketSphinx at the loss of slight speed in order to achieve higher accuracy.

SECTION 6 CONCLUSION

Although my goal was to achieve a complete system that will have a high accuracy and at the same time can be able to execute user commands efficiently, I have managed to develop a system and a guideline to on how an application can be developed for Android using existing tools. I have encountered many errors and fixed them and tried my best to move forward with it and I hope to continue working with the project to improve it's accuracy.

7 References

1. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, D. Jurafsky and J. Martin

2. Fundamentals of Speech Recognition". L. Rabiner & B. Juang. 1993. ISBN: 0130151572.

3. Implementation of Speech Recognition System in Bangla - Thesis Paper by Shammur Absar Chowdhury, CRBLP

4. MIT's Spoken Language Systems Homepage
http://groups.csail.mit.edu/sls//sls-blue-noflash.shtml

5. Speech Technology
ASR software and accessories. http://www.speechtechnology.com/

6. CMU – Instruction set for Training
http://www.speech.cs.cmu.edu/sphinxman/fr4.html

7.CMU – Robust Group Tutorial http://www.speech.cs.cmu.edu/sphinx/tutorial.html

8. CMU Sphinx – wiki http://cmusphinx.sourceforge.net/wiki/

9.Speech Recognition Software http://speech.blau.in/

10. ASR Assignment
http://www.speech.cs.cmu.edu/15-492/homework/hw2/index.html

11. Adding Speech Recognition to your embedded Platform
http://hackaday.com/2010/07/11/adding-speach-recognition-to-your-embedded-platform/

12. Android NDK - http://developer.android.com/sdk/ndk/index.html

13. PocketSphinx-Android http://androidforums.com/computers/185254-pocketsphinx-android.html

14. PocketSphinx-Android Demo
https://github.com/cjac/cmusphinx/tree/trunk/PocketSphinxAndroidDemo

15. http://blog.jayway.com/

16. http://cmusphinx.sourceforge.net/wiki/