Implementation of Speech Recognition System for Bangla

Thesis Report

Supervisor: Prof Mumit Khan

Conducted by:
Shammur Absar Chowdhury
07110071

School of Engineering and Computer Science
BRAC University

Submitted on
August 2010

DECLARATION

I hereby declare that this thesis is based on the results found by myself. Materials of work found by other researcher are mentioned by reference. This Thesis, neither in whole nor in part, has been previously submitted for any degree.

Signature of Supervisor	Signature of Author
____________________________	____________________________
Prof. Dr. Mumit Khan	Shammur Absar Chowdhury

ACKNOWLEDGMENTS

It is an honor for me to thank those who made this thesis possible. I owe my deepest gratitude to my supervisor, Dr Mumit Khan, whose encouragement, guidance and support from the initial level to the end, enabled me to develop an understanding of the subject and helped me to fulfill my thesis work.

This thesis would not be possible without help from The Center for Research on Bangla Language Processing (CRBLP). I am very thankful to them, for letting me uses their previous research works. Especially I would like to show my gratitude to Firoj Alam, Research Programmer of CRBLP, for constantly helping and guiding me to complete the work.

I would like to give my thanks to all my faculties, friends for their constant support especially to Annajiat Alim Rasel for helping me.

I also offer my regards to Khondker Nayef Reza and all others for helping me in different processes in data preparation and testing phase.

Last but not at least, thanks to the Almighty for helping me in every steps of this Thesis work.

Abstract

Implementation of Speech Recognition System for Bangla

Speech recognition and understanding of spontaneous speech have been a goal of research since 1970. It is a process of conversion of speech to text. The object of human speech is not just a way to convey words from one person to another but also to make the other person to understand the depth of the spoken words. For understanding speech human not only consider for information passed to the ears but also judge the information by the context of the information. That's why human can easily understand the spoken language convey to them even in noisy environment. Recognizing speech by machine is so difficult for the dynamic characteristics of spoken languages. People used different approaches for automated speech recognition system. For recognizing speech people always prefer English as most of the research and implemented for them. So I am intended to have my research on Continuous Speech Recognition (CSR) system but preferably in our mother tongue –Bangla. It is an area where a lot to contribute for our language to establish in computer field.

So in this Thesis semester, my contribution is to show how to use CMU-Sphinx tools to build a domain based continuous speech recognition system, what is the methodology; from preparing text to speech corpus, training and integrating it with the system. Then the work is also extended to test the CSR in various environments in basic level.

This report also contains all the related studies done and steps taken to build a continuous speech recognizer.

TABLE OF CONTENTS

List of Figures

List of Tables and Charts

Literature Survey

Speech Recognition has been one of the important research areas since 1930. But research work to recognize Bangla speech has been started approximately around 1999. So while studying for my thesis, research works that I found and from where I have taken data to prepare myself for Thesis are mentioned some over here with details and others in the references section.

At the starting era most of the research works are done by using Artificial Neural Network (ANN), but as I am using HMM based technique so some HMM based and related research are mentioned below.

Isolated and Continuous Bangla Speech Recognition: Implementation, Performance and application perspective *(by Md. Abul Hasnat, Jabir Mowla and Mumit Khan-BRACU)* – I have studied the past works and to the best of my knowledge this work is the first reported attempt to recognized Bangla speech using HMM Technique, so from this publication I have taken most of my suggestion about the steps to build Speech Recognition System for my Thesis. From here I have learned how to increase the quality of audio signal given as input by noise elimination process and end detection algorithm, from this paper I have also learned that how feature of a sound is extracted and what are the parameters taken in feature files, I have also learn the algorithm for creating HMM models though for my Pre – Thesis semester, this was not exactly related but it helped to understand other specifications that I have read about internal structure about Sphinx and use it in my Thesis work. I have also got to know the basic difference between an Isolated SR and Continuous SR from the above research work.

Recognition of Spoken Letters in Bangla *(Abul Hasanat, Md. Rezaul Karim, Md. Shahidur Rahman and Md. Zafar Iqbal - SUST)*, **Extraction of Bangla Vowel and Representation in the Vowel Space** *(Syed Akhter Hossain-East West, M Lutfar Rahman-Du and Farruk Ahmed-NSU),* **Acoustic Analysis of Bangla Consonants***(Firoj Alam , S. M. Murtoza Habib and Mumit Khan) -* From here I have learn the technique used to recognize letters, vowels and consonant, basically here I found out the basic steps towards a recognizer and what are the common steps to build a full functioning recognizer.

Comparative Study of Feature Extraction Methods for Bangla Phoneme Recognition *(by Md. Farukuzzaman Khan and Ramesh Chandra Debnath)* – From here I have learned more about the Feature Extraction Process.
Another online article that I must mention in this part of my report is by **Stephen Cook – Speech Recognition HOWTO**: From here I have studied about different types of SR, Hardware Requirement for any standard SR, gained basic knowledge on digital audio and found many important references that I have taken as a guide in the whole semester.

I would also like to mention about some sites maintained by CMU Sphinx group: **Sphinx Train Document**, **CMU – Robust Group Tutorial** etc - from where I have found all the required tutorials to learn how to train the Sphinx, installation technique, platforms required, download links, file formats etc the links for the mentioned pages are given in reference section.

There are many other publications, online forums, websites from which I have taken information to complete my Thesis work; those related links and information about them are given in reference section.

Chapter 1: Introduction

Speech recognition is widely researched topic around the world. Many scientists and researchers are busy with doing works on speech recognition. Worldwide speech recognition is mostly done in different languages mostly English. Most of the languages in the world have speech recognizers of its own. But our mother tongue Bengali is not enriched with a speech recognizers. Small research works has been carried on Bengali speech recognizer, but it really does not have a great outcome. Implementing continuous speech recognizers for Bengali is our main goal throughout the thesis work. But implementing a recognizer is huge task within a short span of time. So the thesis work then narrowed down to a domain based continuous speech recognizer. Throughout the whole period of work, we tried to learn about different tools and we choose to use Sphinx, another high quality and widely used system which is gaining more popularity due to its BSD license. We continued to learn the tools, learn and prepared the data and files/scripts which we used to train, decode and test the system. All the steps followed are described in the whole report. But before going in to it let us just review few basics.

1.1 Speech Recognition basics:

Speech recognition (SR) in terms of machinery is the process of converting an acoustic signal, captured by a microphone or a telephone, to a set of words. It is a broad term which means it can recognize almost anybody's speech, but to make the machine independent of voice, huge training data is required. There are basically two types of SR:

1. Isolated speech recognition - ISR
2. Continuous speech recognition - CSR

An isolated-word speech recognition system requires that the speaker pause briefly between words, whereas a continuous speech recognition system does not. The continuous speech consists of continuous utterance which is representative of real speech. On the other hand a sentence constructed from connected words does not represent real speech as it is actually concatenation of isolated words. For Isolated word the assumption is that the speech to be recognized comprises a single word or phrase and to be recognized as complete entity with no explicit knowledge or regard for the phonetic content of the word or phrase.

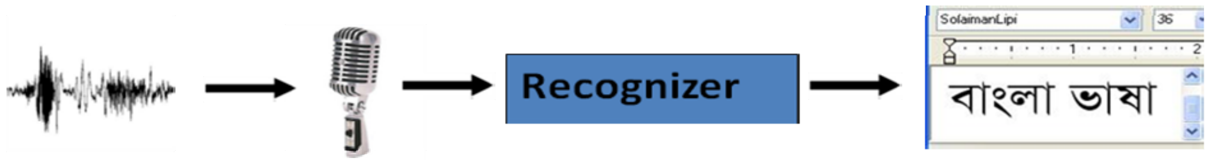


Figure 1.1 Overview of Speech Recognition System

To understand SR technology, some of the terms that are used throughout the full report and needed to know are:

1. Utterance
   An utterance is the vocalization (speaking) of a word or words that represent a single meaning to the computer. Utterances can be a single word, a few words, a sentence, or even multiple sentences.

2. Speaker Dependence
   Speaker dependent systems are designed around a specific speaker. They generally are more accurate for the correct speaker, but much less accurate for other speakers. They assume the speaker will speak in a consistent voice and tempo. Speaker independent systems are designed for a variety of speakers. Adaptive systems usually start as speaker independent systems and utilize training techniques to adapt to the speaker to increase their recognition accuracy.

3. Vocabularies
   Vocabularies (or dictionaries) are lists of words or utterances that can be recognized by the SR system. Generally, smaller vocabularies are easier for a computer to recognize, while larger vocabularies are more difficult. Unlike normal dictionaries, each entry doesn't have to be a single word. They can be as long as a sentence or two. Smaller vocabularies can have as few as 1 or 2 recognized utterances (e.g."Wake Up"), while very large vocabularies can have a hundred thousand or more.

4. 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)

5. A Language Dictionary
   Accepted Words in the Language are mapped to sequences of sound units representing pronunciation, sometimes includes syllabification and stress.

6. A Filler Dictionary

   Non-Speech sounds are mapped to corresponding non-speech or speech like sound units

7. 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

8. HMM
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.

9. 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.

1.2 Overview of the Full system:

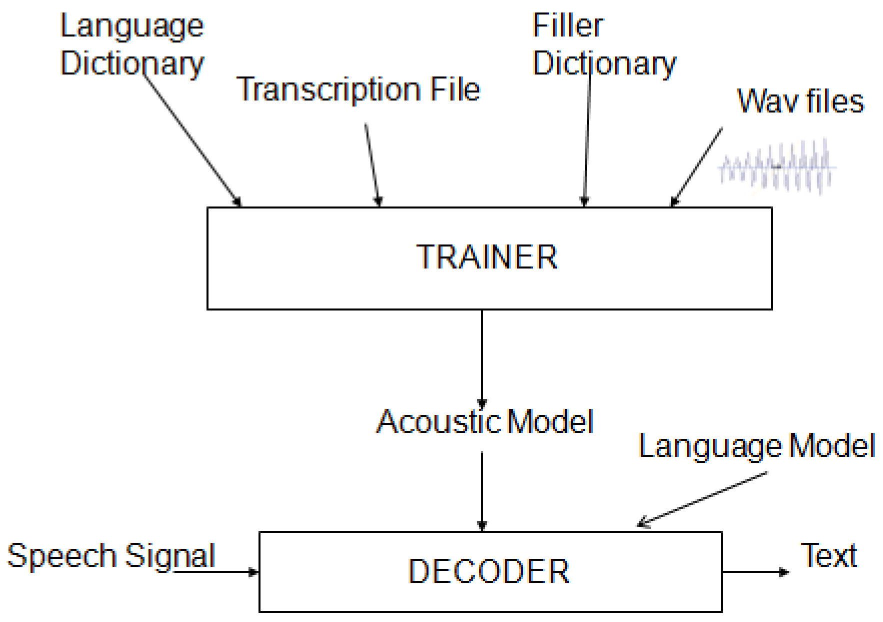


Figure 1.2 Overview of the Project

Chapter 2: Methodology

To build a general purpose speech recognizer, a huge amount of data would have been needed which was not feasible in this short period of time, so before proceeding with the continuous speech recognizer, a domain has been chosen which for this work is a product pricing query system. After choosing the domain, steps that have been followed are:

2.1 Data Preparation:

2.1.1 Text Preparation:

The products that are going to be included in the project has been selected first, then different ways by which a human being can be asking the price of the item has been considered.

For example:
Item list: আলু, বেগুন, পেঁয়াজ etc

List of possible questions selected:

আজকে পেঁয়াজের দাম কত?
পেঁয়াজ প্রতি কেজি কত করে?
এক কেজি পেঁয়াজ কত? etc

2.1.2 Building Speech Corpus:

After selecting text for the recognizer, recording of this chosen data is required. For this work, the recording has been taken place using eight different speakers. And their personal profile includes information like

- Name
- Age
- Gender
- Language dialect

Some other information like

- Environmental condition of recording (for example: class room condition, number of students present, sources of noise like fan, generator sound etc)
- Technical details of device (pc , microphone specification)
- Date and time of recording has also been noted down.

Speaker Id of three characters (100 to 800) has been assigned to each speaker.

Ids have also been assigned to the items (details given in appendices) and to sentence type.

For example:

Item	Item ID
আলু	01
শশা	02

Table 2.1 Item selected with IDs

Sentence type	Id
আজকে --- দাম কত	01
--- প্রতি কেজি কত করে	02
এক কেজি --- কত	03

Table 2.2 Sentence type selected with IDs

During recording session the following parameters of the wave file has been maintained throughout:

- Sampling rate of the audio: 16 kHz
- Bit rate (bits per sample) : 16
- Channel : mono (single channel)

For this work 16 kHz sample rate has been chosen because it provides more accurate high frequency information and 16 bit per sample will divides the element position in to 65536 possible values.

After the recording, the splitting of the audio files per sentence has been done manually using Audacity and saved in a .wav format, where each wav file has been named by using speaker id, sentence id and item id

For example: 1000103.wav stands as

Speaker Id: 100 Sentence Id: 01 Item Id: 03

2.1.3 Transcription File:

A transcript is needed to represent what the speakers are saying in the audio file. So in a file the dialogue of the speaker noted exactly the same precise way it has been recorded, with silence tag (starting tag <s> , ending tag </s>), followed by the file id which represent the utterance. This file is known as transcription file and basically there are main two types of .transcription file. One of them is used to train the system and another is to test. Named the files using the project name, here the name of the project is “crblp_asr” therefore files are named as crblp_asr _train.transcription and another test transcription file as crblp_asr_test.transcription

For example:

```
<s> আজকে আলুর দাম কত  </s> (1000101)

<s> আজকে শশার দাম কত  </s> (1000102)
```

2.1.4 Pronunciation Dictionary:

A language dictionary file where each word maps to a sequence of sound units to derive the sequence of sound units associated with each signal. The file should contain unique entry and sorted accordingly. To make this file, resources from CRBLP pronunciation dictionary is used; if any pronunciation of an entry is not available in the CRBLP dictionary then G2P (grapheme to phoneme) module of CRBLP is used to generate the sound unit sequence. In both cases the output of the pronunciation sound unit is given in Unicode IPA but for training a system the dictionary files need sound units in ASCII. For this reason a small program is made which can translate IPA Unicode to ASCII. After all this, the file is saved with .dic extension. For this project the file is named as crblp_asr.dic [projectName.dic].

For example:

```
আজকে aa j k e

আদা   aa da aa

আদার   aa da aa r

আলু   aa l u

আলুর  aa l u r
```

2.1.5 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. The tool is best for small domains. For this project the file is crblp_asr.bigram.arpabo

For example:

```
-0.3010 পটলের দাম 0.0000

-0.6021 পালংশাক কত -0.1237

-0.6021 পালংশাক প্রতি 0.0000
```

2.1.6 Control File:

It is just a plain text file containing all the name of each of the audio file without extension. Two of this file will be needed. One of them used for training named as crblp_asr_train.fileids and another for testing, named crblp_asr_test.fileids

For example:

```
1000101
1000102
1000103
…..
```

2.1.7 Filler Dictionary:

Dictionary where a non speech sounds are mapped to corresponding non speech sound units. This file is named as crblp_asr.filler

For Example:

```
<s>     SIL
<sil>   SIL
</s>    SIL
```

2.1.8 Phone File:

A simple text file that tells a trainer what phonemes are part of the training set. The file has one phone in each line, no duplicity is allowed. This file can be generated using a small program written for this project which takes the *.dic file as input and gives *.phone file as output. For this project the file name is crblp_asr.phone

Example:

```
e

o

aa~

ae
```

2.2 Setting up the System Environment:

2.2.1 Required Software:

To setup the task, Perl and C compiler is needed. perl will be useful to run the scripts where as to compile the source code a C compiler is needed. This project has been entirely setup in Linux platform, so to check whether both perl and C compiler are present, in the terminal:

For perl type: `$ dpkg -s perl`

If it is installed in the system then it will show all the details else to download and install Perl type in the terminal:

```
$ sudo apt-get install perl
```

For C compiler type: `$ dpkg -s gcc`

If gcc is already installed then related information will be shown else download it by typing in terminal:

```
$ sudo apt-get install gcc
```

2.2.2 Setting up the Trainer:

Download SphinxTrain from http://sourceforge.net/projects/cmusphinx/files/

To setup: untar it by the below command

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

To build the trainer go to SphinxTrain folder and typed:

```
$ ./configure
```

```
$ make
```

This will setup the trainer for further use.

2.2.3 Setting up the Project folder:

First a folder for the project (this project is named as crblp_asr) has been created, which must be in the same folder as SphinxTrain is. Then navigated to the project folder and there we run the below command in the terminal

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

After this in the project folder, few files has been created.

In wav folder, all the audio files (*.wav) has been copied. Then in etc folder all the previous prepared files have been copied so the folder looks like:

crblp_asr/etc:

>crblp_asr.dic

>crblp_asr.filler

>crblp_asr.phone

>crblp_asr_train.fileids

>crblp_asr_train.transcription

>feat.params [created automatically]

>sphinx_train.cfg [created automatically]
There must not be any extra line at the end of all the files mentioned above.

Few changes has been bought in the configuration file for further proceeding. The changes are:

1. The extension of the wavfiles has been changed from 'sph' to 'wav' i.e
   $CFG_WAVFILE_EXTENSION = 'wav'
2. There are few common sound file formats, some of them are nist, raw, mswav etc. In our cfg file the wav type has been altered to 'raw' from 'nist'
   $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 next step is to convert the raw speech data from the wav files to MFCC (Mel Frequency Cepstral Coefficients)and store them in crblp_asr/feat directory. For this we move to the project folder, crblp_asr and run the below command:

```
$perl scripts_pl/make_feats.pl -ctl etc/crblp_asr_train.fileids
```

2.3 Training

After all system setup and data preparation is done, the next step is to train the system and to train the system, audio files from six speaker has been used as data, the user profile has been given in Appendix section. And for training the system scripts in scripts_pl directory will needed to be executed. There are two option, one of them is to run:

```
perl scripts_pl/RunAll.pl
```

As this scripts will execute all the available perl scripts. Another option is run only required scripts one by one in the sequence given below.

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

As described in data preparation section, that in phone file there should not be any duplicate entries and no extra phone which is not in the dictionary should be allowed but if there is then warning message like

WARNING: "main.c", line 144: No triphones involving d FATAL_ERROR: "main.c", line 771: Initialization failed

If step 5 works properly then step 6 can be carried on

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

9.perl scripts_pl/99.make_s2_models/make_s2_models.pl

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

2.4 Preliminary Testing with PocketSphinx:

Download PocketSphinx from the below link:

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

Setup the recognizer using the below command

1. To unpack type in the terminal

$tar zxf pocketsphinx-0.5-20080912.tar.gz

2. To build navigate to pocketsphinx folder and run command
$./build_all_static.sh

3. To setup decoder navigate to the task folder (crblp_asr) and type
$../pocketsphinx0.5/pocketsphinx/scripts/setup_sphinx.pl -task crblp_asr

Then add files in

crblp_asr/etc

>crblp_asr_test.transcription

>crblp_asr_test.fileids

>sphinx_decode.cfg [generated during PocketSphinx setup]

crblp_asr/wav

> all test wav file

Then convert .wav files to .mfc files using command:

```
$perl scripts_pl/make_feats.pl -ctl etc/crblp_asr_test.fileids
```

4. Before starting to decode the test files a modification is required to make. We had to add

```
-dictcase => "yes"
```

to scripts_pl/decode/psdecode.pl

Then to decode type

```
$perl scripts_pl/decode/slave.pl
```

Then after the process is completed we will be getting an output like:

```
MODULE: DECODE Decoding using models previously trained
 Decoding 529 segments starting at 0 (part 1 of 1) 0%
This step had 2 ERROR messages and 45 WARNING messages.  Please check the log file for details.
        Aligning results to find error rate
        SENTENCE ERROR: 2.8% (15/529)   WORD ERROR RATE: 1.0% (23/2319)
```

Then to check the result of the decoder, move to crblp_asr/result and open the file called “crblp_asr.align”.

2.5 Integration with the decoder - Sphinx4

2.5.1 How to install Sphinx4

Download binary distribution of Sphinx4 from the below link

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

Extract the zip file and in Sphinx4*/lib, there we can see jar files like

js.jar

shinx4.jar

WSJ*.jar etc.

To use sphinx4 decoder, another jar file is required called jsapi.jar, which is not present in the lib folder. For jsapi.jar, there are few legal issue for downloading jsapi.jar so sphinx4 do not include this jar directly but it contains two files, lib/jsapi.exe and lib/jsapi.sh, if we run jsapi.sh for Linux, the corresponding jar file appears in lib folder which we can use. Similarly for windows we need to run jsapi.exe.

Along with lib folder, we have few other folders like:

Sphinx4*/bin

Sphinx4*/demo etc

2.5.2 Integrating the decoder with the system:

To create the project we can use any IDE, but for the thesis we choose to work with Netbeans.

Elements that we will need to include in this project are:

> js.jar

> jsapi.jar

> sphinx4.jar

> Acoustic model

> Configuration files

> Language Model

> System libraries provided by Java

2.5.2.1 Building Acoustic model:

Here we will use the models created by SphinxTrain. There are basically two ways in which we can use the models created by the trainer. Among the two we choose to create an acoustic model jar file, which we can easily included in our project. So in this section we will talk about how to create the model jar, where as another method description can be found from:

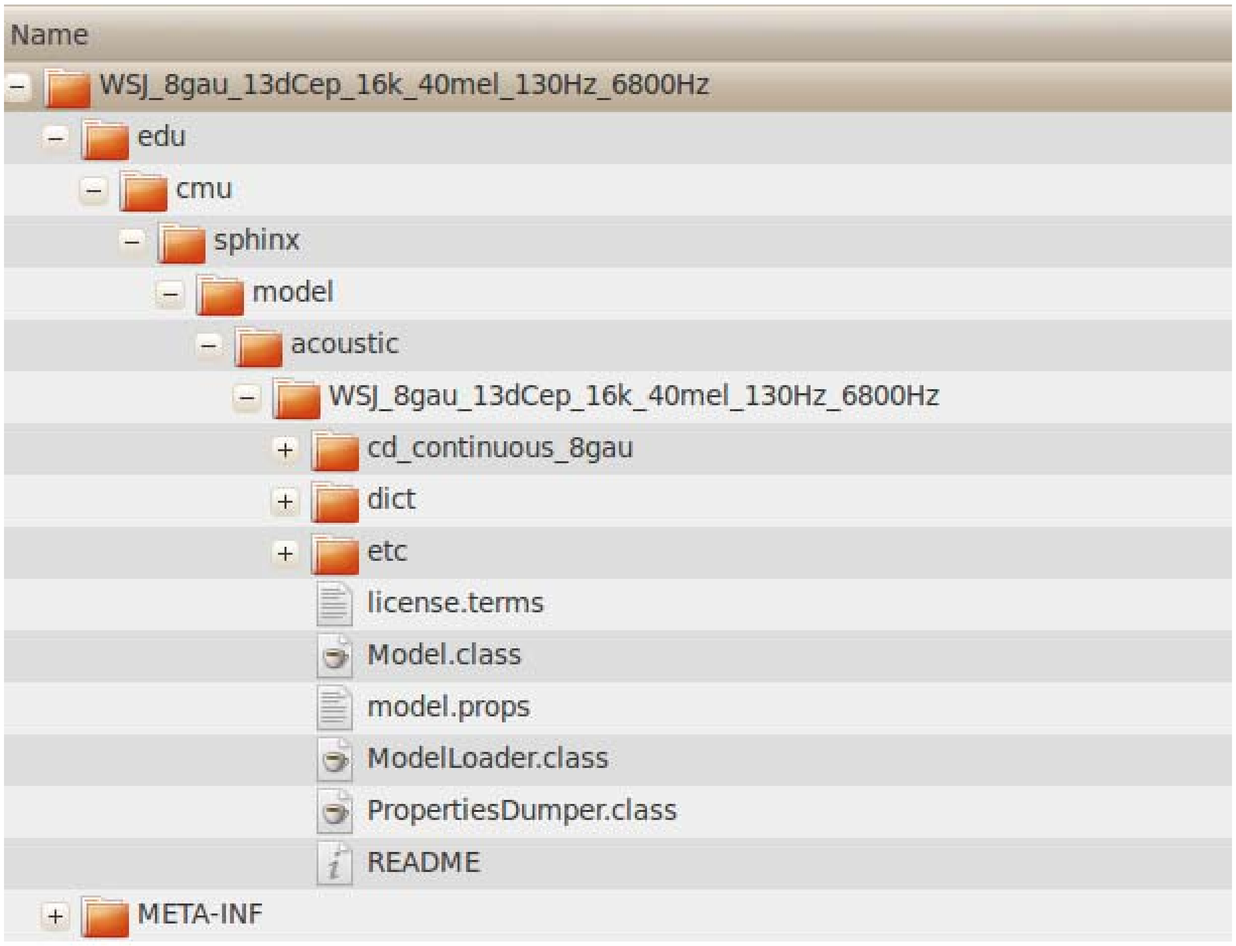
http://cmusphinx.sourceforge.net/sphinx4/doc/UsingSphinxTrainModels.html

The model jar file can be build manually but for ease of working we choose to replace files in a demo acoustic model given with Sphinx4 package and bring few changes to make it usable.

The steps followed for this are:

Unpack lib/WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz.jar from sphinx4 folder

The structure of the files is shown below:



```
Name
– WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz
  – edu
    – cmu
      – sphinx
        – model
          – acoustic
            – WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz
              + cd_continuous_8gau
              + dict
              + etc
                license.terms
                Model.class
                model.props
                ModelLoader.class
                PropertiesDumper.class
                README
  + META-INF
```

Figure 2.1 File Structure Tree in an Acoustic Model jar file

In folder:

cd_continuous_8gau, we have:

>cd_continuous_8gau/means

>cd_continuous_8gau/mixture_weights

>cd_continuous_8gau/variances

>cd_continuous_8gau/transition_matrices

Replace all of them with files from (Training files)
crblp_asr/model_parameters/crblp_asr.cd_cont_100

For small vocabulary task we can also take this files from ci_cont*

In dict folder, replace cmudict.0.6d with the language dictionary which we have prepared. We will also replace the filler dictionary with ours. Keeping both the name of the file same.

In etc, we will replace the inside contents of

> WSJ_clean_13dCep_16k_40mel_130Hz_6800Hz.4000.mdef with
model_architecture/crblp_asr.100.mdef

>WSJ_clean_13dCep_16k_40mel_130Hz_6800Hz.ci.mdef with
model_architecture/crblp_asr.ci.mdef

All other files in the WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz folder remains the same. Compress the folder to make a .jar file and this is our acoustic model jar.

2.5.2.2 Configuration files:

In the system both audio and live speech is taken as input to decode, so we need to write configuration file to define all the components we have.

Some essential tasks to define in config files are:

> Decoder configuration

> Search manager

> Dictionary configuration

> Language Model configuration

> Acoustic model configuration

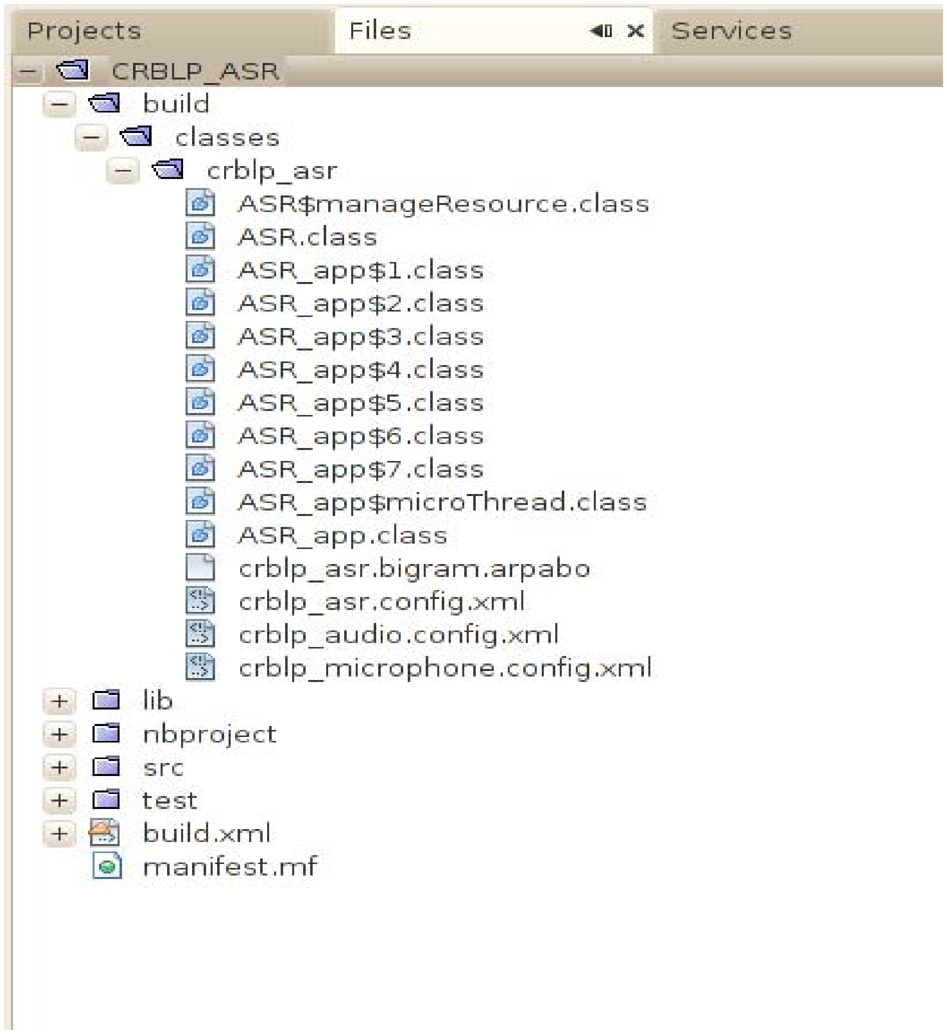
> Front end (of Shinx4 decoder) configuration

> Microphone and Stream data source configuration etc

To learn more about configuration file we can see examples “HelloNgram”, “Transcriber” provided with in Sphinx4, and used them to write our own file.

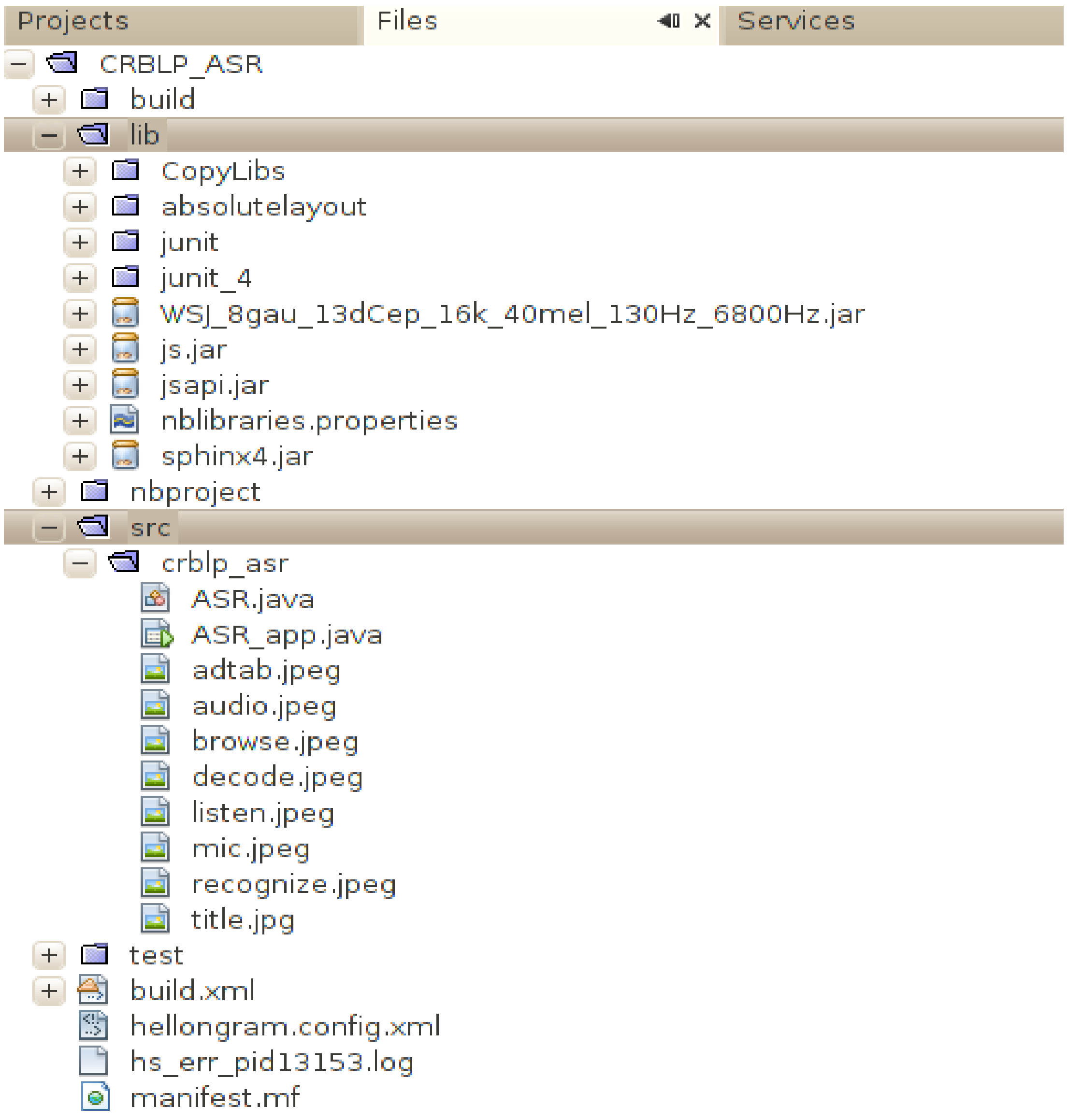
2.5.2.3 Structure of the files in the Netbeans project

After including all the essential files in the project, the file section of the project view is given below



```
CRBLP_ASR
  build
    classes
      crblp_asr
        ASR$manageResource.class
        ASR.class
        ASR_app$1.class
        ASR_app$2.class
        ASR_app$3.class
        ASR_app$4.class
        ASR_app$5.class
        ASR_app$6.class
        ASR_app$7.class
        ASR_app$microThread.class
        ASR_app.class
        crblp_asr.bigram.arpabo
        crblp_asr.config.xml
        crblp_audio.config.xml
        crblp_microphone.config.xml
  lib
  nbproject
  src
  test
  build.xml
  manifest.mf
```

Figure 2.2 Project Contents under build file in NetBeans



```
– CRBLP_ASR
  + build
  – lib
    + CopyLibs
    + absolutelayout
    + junit
    + junit_4
    + WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz.jar
    + js.jar
    + jsapi.jar
    + nblibraries.properties
    + sphinx4.jar
  + nbproject
  – src
    – crblp_asr
        ASR.java
        ASR_app.java
        adtab.jpeg
        audio.jpeg
        browse.jpeg
        decode.jpeg
        listen.jpeg
        mic.jpeg
        recognize.jpeg
        title.jpg
  + test
  + build.xml
    hellongram.config.xml
    hs_err_pid13153.log
    manifest.mf
```

Figure 2.3 Project Contents under lib and src files in NetBeans

Then using our main class which is ASR_app.java over here, we build our project to proceed for the testing phase.

Chapter 3: Testing and Performance Evaluation

We tried to test the system by varying speaker, environment, microphone etc. Testing the system is done using audio inputs of two test speaker and live test from the microphone in different environment (speaker profiles are given in Appendix section).

Tests are done using two different decoder:
1. Pocket Sphinx

2. Sphinx 4

3.1 Test results with PocketSphinx:

Experiment No	Details	Result
Experiment 1	Using Trained data set Number of Speaker: 6 Male: 3 Female: 3	TOTAL Words: 2292 Correct: 2276 Errors: 16 TOTAL Percent correct = 99.30% Error = 0.70% Accuracy = 99.30%
Experiment 2	Untrained Speaker Environment: Open space, Noisy, Different Microphone Model from training setup	TOTAL Words: 392 Correct: 289 Errors: 107 TOTAL Percent correct = 73.72% Error = 27.30% Accuracy = 72.70%
Experiment 3	Untrained Speaker Environment: In a Lab room Number of students present: 14	TOTAL Words: 142 Correct: 134 Errors: 9 TOTAL Percent correct = 94.37% Error = 6.34% Accuracy = 93.66%
Experiment 4	Trained Speaker Environment: Closed room Different Microphone Model from training setup	TOTAL Words: 392 Correct: 381 Errors: 12 TOTAL Percent correct = 97.19% Error = 3.06% Accuracy = 96.94%

Table 3.1 Experimental Details with Results for PocketSphinx

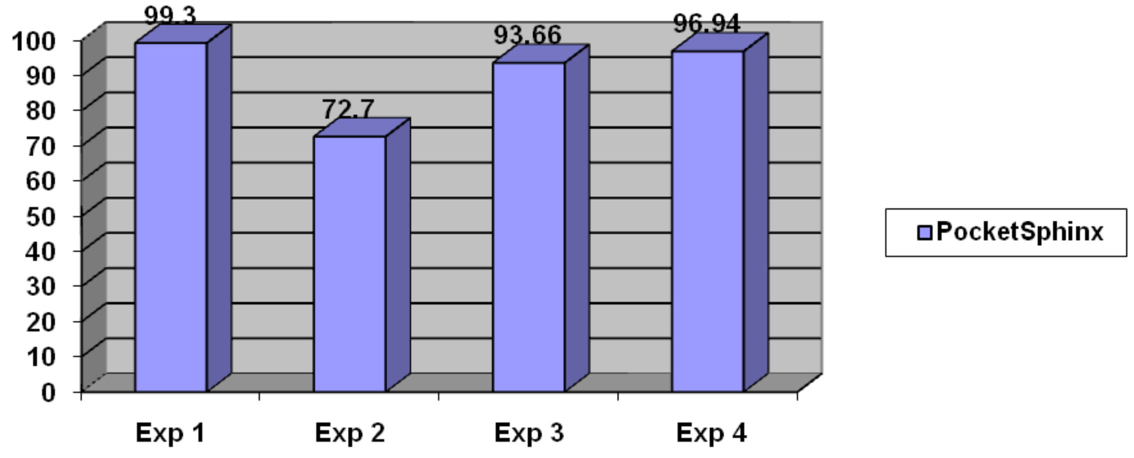


Chart 3.1 Experiment Results with PocketSphinx

Average Accuracy Rate = 90.65%

3.2 Test results with Sphinx4:

3.2.1 Input Type: Microphone

Experiment No	Details	Result
Experiment 1	Untrained Speaker Lab room Male Age = 22	TOTAL Words: 48 Correct: 41 Errors: 7 TOTAL Percent correct = 85.42% Error = 14.58% Accuracy = 85.42%
Experiment 2	Untrained Speaker Lab room Male Age = 22	TOTAL Words: 48 Correct: 43 Errors: 5 TOTAL Percent correct = 89.58% Error = 10.42% Accuracy = 89.58%
Experiment 3	Untrained Speakers Class room Age = 20-22	TOTAL Words: 20 Correct: 19 Errors: 3 TOTAL Percent correct = 95.00% Error = 15.00%

		Accuracy = 85.00%
Experiment 4	Untrained Speaker Cafeteria Age = 23	TOTAL Words: 96 Correct: 88 Errors: 15 TOTAL Percent correct = 91.67% Error = 15.62% Accuracy = 84.38%
Experiment 5	Untrained Speaker Room Number of people in surrounding=3 Age = 17	TOTAL Words: 93 Correct: 79 Errors: 14 TOTAL Percent correct = 84.95% Error = 15.05% Accuracy = 84.95%
Experiment 6	Untrained Speaker Room Number of people in surrounding=3 Age = 52	TOTAL Words: 128 Correct: 118 Errors: 11 TOTAL Percent correct = 92.19% Error = 8.59% Accuracy = 91.41%

Table 3.2 Experimental Details with Results for Sphinx 4 Live

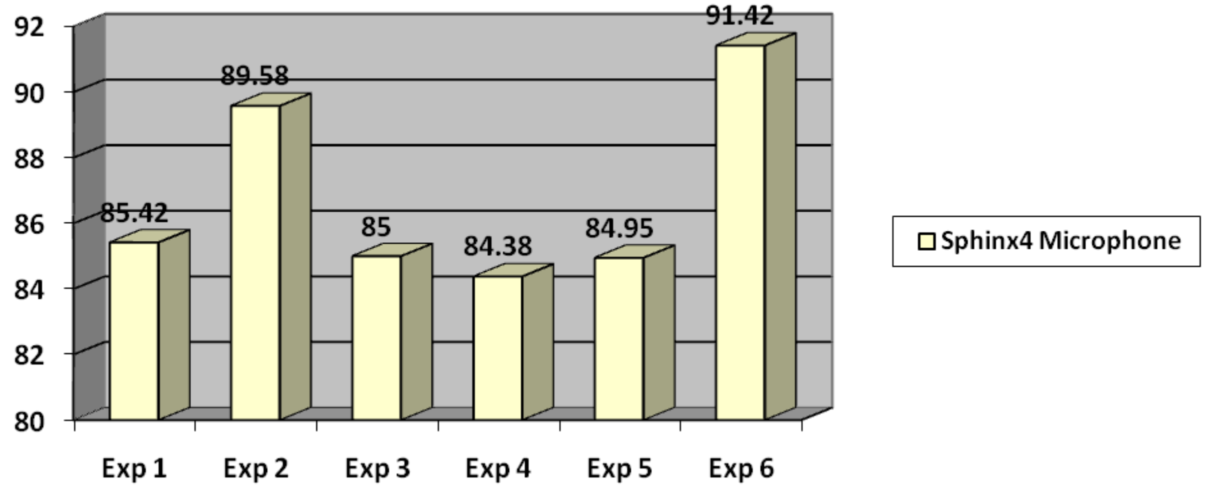


Chart 3.2 Experiment Results with Sphinx-4 Live Input

Average Accuracy Rate = 86.79%

3.2.2 Input Type: Audio

Experiment No	Details	Result
Experiment 1	Audio File Speaker 100*	TOTAL Words: 389 Correct: 311 Errors: 78 TOTAL Percent correct = 79.95% Error = 20.05% Accuracy = 79.95%
Experiment 2	Audio File Speaker 200*	TOTAL Words: 389 Correct: 330 Errors: 59 TOTAL Percent correct = 84.83% Error = 15.17% Accuracy = 84.83%
Experiment 3	Audio File Speaker 300*	TOTAL Words: 379 Correct: 234 Errors: 145 TOTAL Percent correct = 61.74% Error = 38.26% Accuracy = 61.74%
Experiment 4	Audio File Speaker 400*	TOTAL Words: 380 Correct: 249 Errors: 132 TOTAL Percent correct = 65.53% Error = 34.74% Accuracy = 65.26%
Experiment 5	Audio File Speaker 500*	TOTAL Words: 380 Correct: 206 Errors: 174 TOTAL Percent correct = 54.21% Error = 45.79% Accuracy = 54.21%
Experiment 6	Audio File Speaker 600* *[Details in Speaker Profile - Appendices]	TOTAL Words: 384 Correct: 316 Errors: 68 TOTAL Percent correct = 82.29% Error = 17.71% Accuracy = 82.29%

Table 3.3 Experimental Details with Results for Sphinx 4 Audio

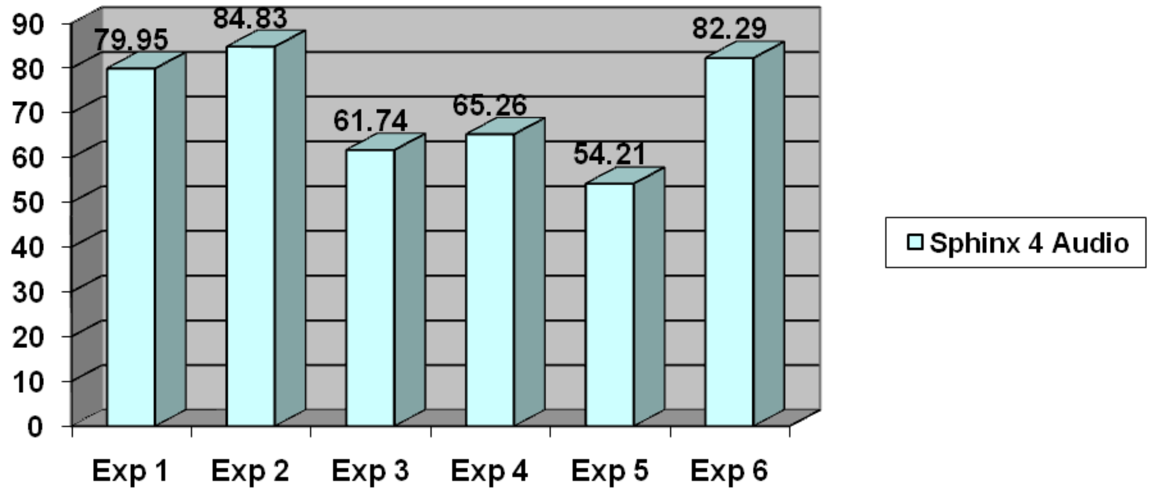


Chart 3.3 Experiment Results with Sphinx-4 Audio Input

Average Accuracy Rate = 71.38%

3.2.2.1 Limitation of Sphinx4 in case of audio files input

When running the test it is observed that when there is a bit long period of silence in the middle of the speech, the performance of the decoder decreases and simultaneously a warning is printed shown below:
" NonSpeechDataFilter: ALERT: getting a SpeechStartSignal while in speech, removing it."

Chapter 4: Limitations

This implementation of Speech Recognition System has been build on small data, domain based and trained with only 6 speakers, whereas for a general Speech Recognition System at least 50 Speaker is needed. Its performance is dependent on speaker, environment, microphone, distance between speaker and microphone, stress. It recognizes the sentences better when said together and it do make mistake to recognize some particular word if said in a slow speed or if broken down.

Chapter 5: Future Works

As this implementation is in preliminary stage, so in future we need to test it more to find out how we can improve its performance. We do plan to increase its capability to recognize speech more accurately and enhance its vocabulary.

We also want to train this system, with different parameters to see how it affects the performance of the speech recognizer.

We also have an intention to integrate the system with Interactive voice response. For all this, one of the regularly used tool that we are interested to test in future is Asterisk. Basically it is a open source telephony software, which have every single components from call routing to VoIP Gateways etc of telecommunication. It is not a NLP tool but has few tools that help it to integrate with Speech processing systems.

A future proposed system diagram is shown below:

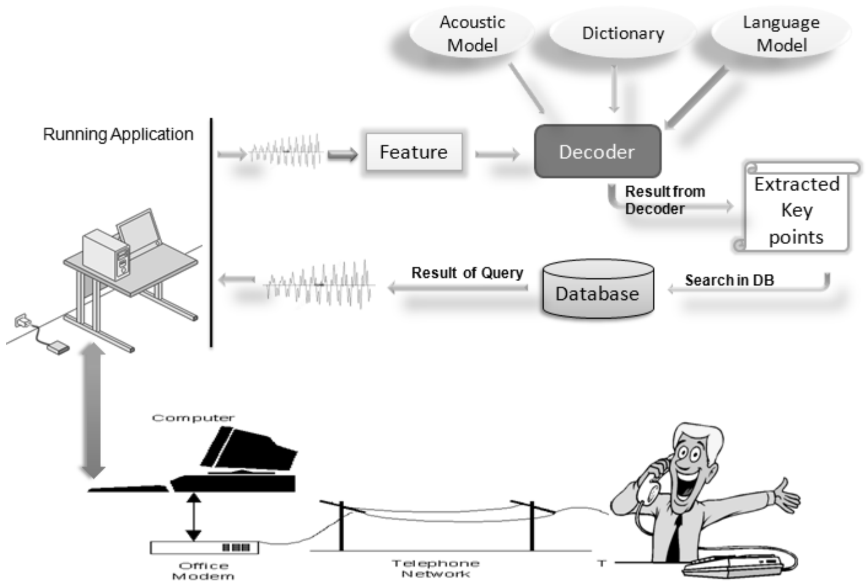


Figure 5.1 Proposed future project

Chapter 6: Conclusion

The described above system is in its preliminary level, all the tools, data used to train and build the system has been discussed throughout the whole report. To make the system more natural, lots of improvement in case of data, tools and its parameters is required, but again no speech recognizer till now has 100% accuracy. But if we can include more accurate training, adaption of system with environment, speaker adaption, then system resulted will be enough to serve our purpose.

References

Publications

1. Firoj Alam, S. M. Murtoza Habib, Dil Afroza Sultana and Mumit Khan, "**Development of Annotated Bangla Speech Corpora**", Spoken Language Technologies for Under-resourced language (SLTU'10), Universiti Sains Malaysia, Penang, Malasia, May 3 - 5, 2010.

2. Firoj Alam , S. M. Murtoza Habib and Mumit Khan. "**Acoustic Analysis of Bangla Consonants**", Spoken Language Technologies for Under-resourced language (SLTU'08), Hanoi, Vietnam, May 5 - 7, 2008.

3. Abul Hasanat Md. Rezaul Karim, Md. Shahidur Rahman and Md. Zafar Iqbal, "**Recognition of Spoken Letters in Bangla**",SUST

4. Sadaoki Furui*, Fellow, IEEE, Tomonori Kikuchi, Yousuke Shinnaka, and Chiori Hori, Member, IEEE,"* **Speech-to-Text and Speech-to-Speech Summarization of Spontaneous Speech**"

5. Md. Farukuzzaman Khan and Ramesh Chandra Debnath, "**Comparative Study of Feature Extraction Methods for Bangla Phoneme Recognition**"

6. Syed Akhter Hossain-East West, M Lutfar Rahman-Du and Farruk Ahmed-NSU, "**Acoustic Feature Extraction of Bangla Vowel and Representation in the Vowel Space"**

7. Md. Abul Hasnat, Jabir Mowla, Mumit Khan, "**Isolated and Continuous Bangla Speech Recognition: Implementation, Performance and application perspective**"

8. Nipa Chowdhury – DUET, Md. Abdus Sattar – BUET and Arup Kanti Bishwas – ERL, "**Separating Words from Continuous Bangla Speech** "

9. Kamrul Hayder- BRACU, "**Research Report on Bangla Lexicon**"

10. A K M Mahmudul Hoque, BRACU,Thesis paper –May 2006," **BENGALI SEGMENTED AUTOMATED SPEECH RECOGNITION**"

11. KAI-FU LEE, Haiao-Wuen Hon and Raj Reddy, "**An Overview of the SPHINX Speech Recognition System**"

Books

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. How to Build a Speech Recognition Application". B. Balentine, D. Morgan, and W. Meisel. 1999. ISBN: 0967127815

4. Applied Speech Technology". A. Syrdal, R. Bennett, S. Greenspan. 1994. ISBN: 0849394562.

5. Speech Recognition : The Complete Practical Reference Guide". P. Foster, T. Schalk. 1993. ISBN: 0936648392

Internet

1. news:comp.speech

   Newsgroup dedicated to computer and speech.

   – US: http://www.speech.cs.cmu.edu/comp.speech/

   – UK: http://svr-www.eng.cam.ac.uk/comp.speech/

2. news:comp.speech.research

   Newsgroup dedicated to speech software and hardware research

3. MIT's Spoken Language Systems Homepage

   http://groups.csail.mit.edu/sls//sls-blue-noflash.shtml

4. Speech Technology

   ASR software and accessories. http://www.speechtechnology.com/

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

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

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

8. Sphinx-4 A speech recognizer written entirely in the Java^{TM} programming language
http://cmusphinx.sourceforge.net/sphinx4/

9. Sphinx-4 Application Programmer's Guide -
http://cmusphinx.sourceforge.net/sphinx4/doc/ProgrammersGuide.html

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

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

12. Acoustic Model Creation using SphinxTrain
http://forum.visionopen.com/viewtopic.php?f=39&t=1130

13. Acoustic Model Creation using SphinxTrain
http://www.bakuzen.com/?p=16

14. Sphinx-4 Instrumentation
http://cmusphinx.sourceforge.net/sphinx4/javadoc/edu/cmu/sphinx/instrumentation/doc-files/Instrumentation.html

15. Sphinx-4 Configuration Management
http://cmusphinx.sourceforge.net/sphinx4/javadoc/edu/cmu/sphinx/util/props/doc-files/ConfigurationManagement.html

16. Hello World Decoder Quick Start Guide
http://sphinx.subwiki.com/sphinx/index.php/Hello_World_Decoder_QuickStart_Guide

17. baküzen » Blog Archive » Sphinx4
http://www.bakuzen.com/?p=4

18. How to Use Models form SphinxTrain in Sphinx-4
http://cmusphinx.sourceforge.net/sphinx4/doc/UsingSphinxTrainModels.html

Appendices

A. IPA Unicode to ASCII Chart

Bangla phoneme with IPA	IPA Unicode	IPA ASCII
ক /k/	k	K
খ /k^h/	k^h	Kh
গ /g/	g	G
ঘ /g^h/	g^h	Gh
চ /c/	c	C
ছ /c^h/	c^h	Ch
য,জ /ɟ/	ɟ	J
ঝ /ɟ^h/	ɟ^h	Jh
ট /t/	t	T
ঠ /t^h/	t^h	Th
ড /d/	d	D
ঢ /d^h/	d^h	Dh
ত /t̪/	t̪	Ta
থ /t̪^{h}/	t̪^{h}	Tah
দ /d̪/	d̪	Da

Bangla phoneme with IPA	IPA Unicode	IPA ASCII
ধ /d̪^{h}/	d̪^{h}	dah
প /p/	p	p
ফ /p^h/	p^h	ph
ব /b/	b	b
ভ /b^h/	b^h	bh
শ,ষ,স/ʃ/	ʃ	sh
শ,স/s/	s	s
ম /m/	m	m
ঙ,ং /ŋ/	ŋ	ng
ণ, ন /n/	n	n
র /r/	r	r
ল /l/	l	l
ড়, ঢ় /ɾ/	ɾ	ra
য় /j/	j	y

Bangla phoneme with IPA	IPA Unicode	IPA
অ/ɔ/	ɔ	a
আ/a/	a	aa
ই/i/	i	i
উ/u/	u	u
এ/e/	e	e
ও/o/	o	o
এ্যা/æ/	æ	ae
অঁ/ɔ̃ /	ɔ̃	a~
আঁ/ã/	ã	aa~
ইঁ/ĩ/	ĩ	i~
উঁ/ũ/	ũ	u~
এঁ/ẽ/	ẽ	e~
ওঁ/õ/	õ	o~
এ্যাঁ/æ̃ /	æ̃	ae~
অও/ɔo/	ɔo	ao

Bangla phoneme with IPA	IPA	IPA ASCII
আই/ai/	ai	aai
আউ/au/	au	aau
আয়া/aja/	aja	aya
ইউ/iu/	iu	iu
ইএ-ইয়ে/ie/	ie	ie
ইও/io/	io	io
ইয়া-ইআ/ia/	ia	ia
উই/ui/	ui	ui
উয়া-উআ/ua/	ua	ua
উয়ে/ue/	ue	ue
উয়ো-উও/uo/	uo	uo
এই/ei/	ei	ei
এউ/eu/	eu	eu
এও/eo/	eo	eo
এয়া-এআ/ea/	ea	ea

Bangla phoneme with IPA	IPA Unicode	IPA ASCII
এ্যায়া/æa/	æa	eea
ওই/oi/	oi	oi
ওউ/ou/	ou	ou
ওয়া-ওআ/oa/	oa	oa
ওয়ে/oe/	oe	oe

B. Speaker Profile

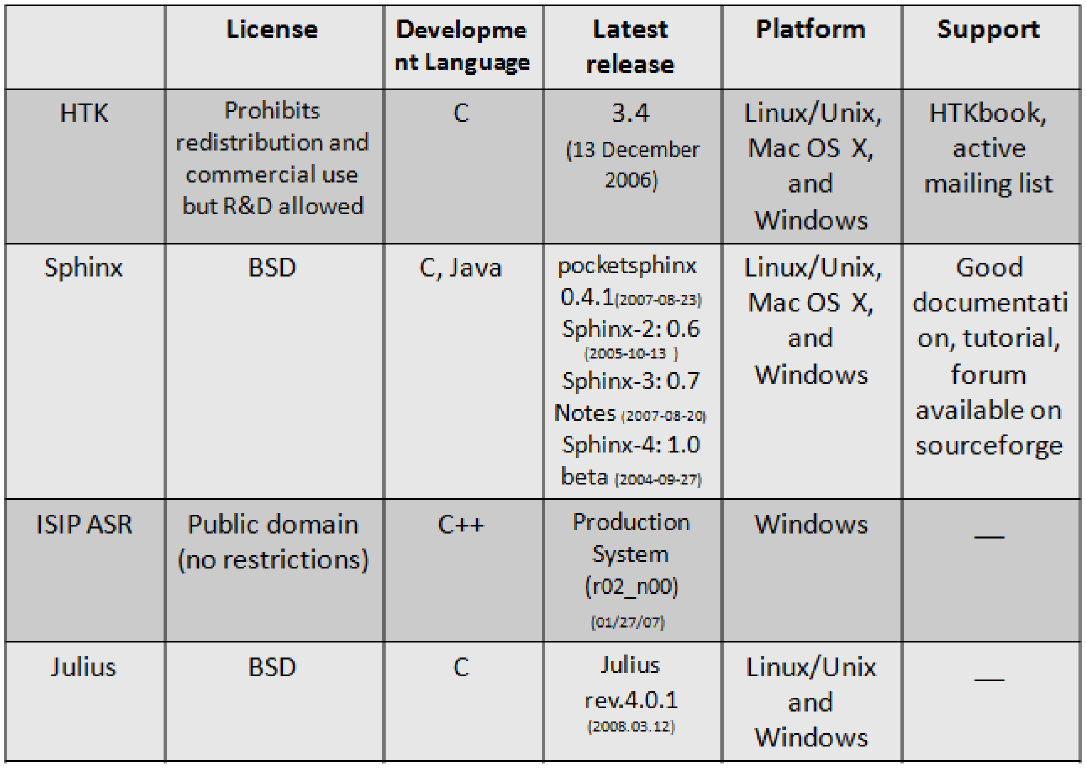
Speaker ID	Age	Gender	District	Environment
100	27	Male	Bramonbaria	University Department
200	21	Female	Feni	Class Room
300	20	Male	Barishal	Lab
400	21	Male	Chitagong	Lab
500	23	Female	Pabna	Lab
600	22	Female	Comilla	Department
700	21	Male	Norshindi	Lab
800	20	Male	Dinajpur	Open Space

C. Item List and Code:

Item	Id
আলু	01
শশা	02
বেগুন	03
পটল	04
পেয়াজ	05
রসুন	06
মূলা	07
ফুলকপি	08
লাউ	09
গাজর	10
শালগম	11
বাঁধাকপির	12
সীম	13
করলা	14
পালংশাক	15
কাচা মরিচ	16

পেঁপে	17
কলমিশাক	18
মিষ্টি কুমড়া	19
বরবটি	20
কচুশাক	21
টমেটো	22
লালশাক	23
মটর শুটি	24
পুদিনা পাতা	25
পুঁইশাক	26
কাঁচকলা	27
লাল মরিচ	28
আদা	29
ধনে পাতা	30

D. Software/Tools available for Speech Recognizers:



	License	Development Language	Latest release	Platform	Support
HTK	Prohibits redistribution and commercial use but R&D allowed	C	3.4 (13 December 2006)	Linux/Unix, Mac OS X, and Windows	HTKbook, active mailing list
Sphinx	BSD	C, Java	pocketsphinx 0.4.1(2007-08-23) Sphinx-2: 0.6 (2005-10-13) Sphinx-3: 0.7 Notes (2007-08-20) Sphinx-4: 1.0 beta (2004-09-27)	Linux/Unix, Mac OS X, and Windows	Good documentation, tutorial, forum available on sourceforge
ISIP ASR	Public domain (no restrictions)	C++	Production System (r02_n00) (01/27/07)	Windows	—
Julius	BSD	C	Julius rev.4.0.1 (2008.03.12)	Linux/Unix and Windows	—