LB/DON/106/2016

IT 01/128

# Speaker independent Sinhala Speech to Text SMS application for Mobile Phones

B.G.D. Anuradha Bopagama

139157K

Dissertation submitted to the Faculty of Information Technology, University of Moratuwa, Sri Lanka for the partial fulfillment of the requirements of the Master of Science in Information Technology.

004"16"

004(043)

April 2016

TH 3163
+
1 DVD - ROM
(TH 3160 - TH3180)

# Declaration

We declare that this thesis is our own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

Name of Student (s)

B.G.D.A. Bopagama

Signature of Student (s)

[illegible]

Date: 29/04/2016

Supervised by

Name of Supervisor(s)

G.T.I. Karunarathne

Signature of Supervisor(s)

***UOM Verified Signature***

Date: 29/04/2016

# Dedication

To my parents and my wife for their constant support and encouragement over the years.

# Acknowledgement

I would first like to thank my project supervisor Lecturer Mrs. Indika Karunaratne of the Faculty of Information Technology at University of Moratuwa Sri Lanka. She was there for me whenever I ran into a trouble or had a question about my research or writing. She consistently allowed this paper to be my own work, and steered me in the right direction whenever she thought I needed it.

I would also like to thank the Students in University of Moratuwa Information Technology Faculty who were involved in the survey and Mr. Amal Wijewardane who did a great job by coordinate with other fellow students and get the required voice recordings for this research project. Without their passionate participation and input, this project could not have been successfully conducted.

Finally, I must express my very profound gratitude to my parents, parents in law, sister in law, brother in law and to my loving wife for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Author

Anuradha Bopagama

# Abstract

Speech recognition is one of the most discussed topics by researchers in recent years. Because of the limitless applications and the competition of making more user-friendly systems, lots of researchers put their effort on speech recognition system developments. There are lot of applications have already developed for English language. But for the languages like Sinhala, Hindi, Tamil are still at their preliminary stages.

The main purpose of this study was to develop a speaker independent automatic speech recognition android application for Sinhala language. Sinhala is the native language for Sri Lankans and they are the only people who speak Sinhala Language. So this study will create a great opportunity for the Sri Lankans to build their own speech recognition applications using and enhancing this language model.

The products of the study include frequently used SMS phrases speech corpus in Sinhala language which can be used in sending SMS. A survey has conducted among university students in order to collect frequently used SMS phrases. Those data will be used to create and implement the speech recognition system. Since the speech recognition training task is a time consuming one and the time limitation for the project restrict the size of the SMS phrases corpus to a limited one. At the moment audio recordings were only taken from one female and a one male and more recordings needed to build more accurate speech model.

The System was implemented using a HMM toolkit call CMUsphinx. CMUSphinx toolkit is a leading open source speech recognition toolkit with various tools used to build speech applications. CMU Sphinx toolkit has a number of packages for different tasks and applications. Pocketsphinx is one of the tools that support Android operating system which comes under CMUSphinx. Pocketsphinx tool used to create a speech model that can be used in various applications. To build the speech model it needs audio recordings of text and corresponding text. Once the model created it can be used in various applications. The main aim of this project is to build an accurate speech recognition model for Sinhala language that can be used in Android operating system.

# Contents

## List of Tables

## List of Figures

# Chapter 01

## Introduction

During the recent past speech recognition area has become a big attraction to the researchers and developers. More accurate speech recognition can lead to do so many developments in wide range of areas. Human speech is the most widely used communication methodology among humans. So the researchers and developers were trying to simulate the human speech into computer systems and try to recognize what is being said. This is the beginning of speech recognition systems.

Once the speech recognition systems for PCs become more and more accurate developers were trying to focus on implementing speech recognition on mobile phones. With the emerging of Android operating system and IOS operating system for Apple iPhones, development in speech recognition on mobile phones developed rapidly. The development on Google Voice Search and Apple's SIRI shows higher accuracy rates than their earlier versions. Currently Google, Microsoft and Apple are doing a considerable amount of work on this context.

Human speech does have a clear difference from person to person as well as it has many different languages. It's said that up to 7,000 different languages are spoken around the world [1]. So the speech recognition system developers are trying to increase the number of supported languages while they are trying to increase the accuracy

Sinhala is the native language for Sri Lankans. Currently there are many researches and developments are under going to implement a good voice recognition system for Sinhala language. Sinhala is less resourced non-Latin language and Sinhala language does have more letters than English alphabet. Complete Sinhala alphabet contains 60 letters, 20 for vowels and 40 for consonants [2]. So when converting speech to text it has some major difficulties. This was addressed using Sinhala Uni-codes.

## 1.1 Background and Motivation

For a long time speech-recognition software was poor in recognition human speech. But lately it has got much better and most modern smart phones now have a host of voice-activated features. Speech activated means you don't have to use traditional methods like typing on a keyboard using fingers or selecting a menu using your finger tip on a touch device. All the things can be done using voice commands. It will save time and give the capability to do simultaneous tasks like sending an urgent SMS while cooking or driving. And also people who don't have a good knowledge in technical things like mobile phones or computer systems will get the benefit to operate them much easier way by giving voice commands. Most importantly this speech recognition system concept is beneficial for disabled personnel, like blind people. They will be able to operate mobile phones much easier.

By introducing Sinhala speech recognizing system Sri Lankan natives will get benefited. This can be used to enhance the mobile phone literacy within Sri Lankan community and it makes people to have communication facilities much more conveniently. Operating a mobile phone has some challenges as stated below,

1. Lack of knowledge to operate mobile devices.
2. People with weak eye sight or with disabilities may face difficulties in text messaging using a mobile phone.
4. Language barrier: Most of the mobile devices use English as the default language.

The language barrier issue is common among the Sri Lankans due to the lack of English language knowledge. Most of the people in Sri Lanka send SMSs using "Singlish" words, like "mama yanawa". Currently there are lots of Sinhala character typing applications introduced for mobile phones, but still there is one difficulty on them. Typing Sinhala words is really hard due to Sinhala alphabet. It is a big set of character combinations with comparing to English alphabet. Sinhala speech recognition comes handy in these situations. By developing a speech recognition application for mobile devices, people in Sri Lanka will get benefited. And also the Sinhala Language model created in this project can be used for any other speech recognition system which uses CMUSphinx [3].

## 1.2 Aim and Objectives

**Aim:** The main aim of this research project is to create more accurate Sinhala speech recognition system for mobile phones running on Android operating system and thereby support mobile users who are not familiar with using mobile phones due to the challenges stated above will be able to use mobile phones much easily using their native language.

**Objectives:**

- To develop an Android mobile phone application that will be capable of recognizing human speech in Sinhalese.
- Application is capable of converting recognized Sinhala voice input into Sinhala text as a SMS and sending to any mobile number.
- Include speaker independent capabilities to recognize any human voice speak in Sinhala language.

## 1.3 Structure of dissertation

In chapter 02 Current tendency in Speech recognition systems described. And in chapter 03 technology behind the speech recognition on mobile devices is discussed. Then chapter 04 has described the approach to Sinhala speech recognition using Pocketsphinx. In chapter 05 Design of the Sinhala speech recognition system for Android OS described. After that in chapter 06 explains the implementation process of the system. Then in chapter 07 present the evaluation part and chapter 08 describe the conclusion part.

## 1.4 Summary

Speech recognition systems for mobile phones are rapidly developing to cater day to day requirements. As Sri Lankans we also need to develop speech recognition systems for our own language. This study is an attempt to create a speech recognition system for Sinhala language which can be used for various applications in the future.

Chapter 02

# Literature Review: Current tendency in Speech recognition systems

## 2.1 Introduction

Until recently, giving voice commands to control computer systems is considered to be a pure science fiction. But speech technologies have grown rapidly and the science fiction became reality. This is one of the most discussed topics among researchers because of its usefulness. Speech recognition systems give people more convenient and more reliable source of communication between man and machines. Rather than typing or clicking, giving voice commands is much faster and more precise.

Speech recognition has been a most discussed topic and had decades of progress including the successful introduction of commercial voice-based systems. Those developed systems were capable to convert your speech into text or to control some devices like robots, TV, mobile phone etc. Especially in mobile devices due to the small sized keyboards speech-to-text systems are widely used. With the emerging of Android operating system and IOS operating system for Apple iPhones, development in speech recognition on mobile phones developed rapidly.

Human speech does have a clear difference from person to person as well as it has many different languages. So the speech recognition system developers are trying to increase the number of supported languages while they are trying to increase the accuracy. Now there are open source tools available to construct your own speech recognition system for any language.

## 2.2 Comparison

Nadungodage and Weerasinghe developed a continuous speech recognizer to build a prototype of a continuous Sinhala speech recognizer developed [4]. The system they built is a speaker dependent system which is accurate only for a single voice. Currently the speech recognition has developed to a certain level that Automatic Speech recognition[5]

successfully implemented for many languages. In this research authors try to apply existing speech recognition mechanisms to develop a Sinhala speech recognition system. Paper discussed about Speech recognition Network[6] in condensed manner. As explained above this paper discussed only about speaker dependent ASR system. It's much easier than speaker independent ASR system. Because in speaker independent ASR systems with better accuracy need large vocabulary and voice recordings to create Language Models[4] and Acoustic models[4]. Major drawback of this work is that authors didn't try to implement the speech recognition system on a practically usable application. They just created the system and test it for accuracy on a testing toolkit.
In a different approach to a speech recognizer development a Speech Corpus for Sinhala Speech Recognition system has been developed by Nadungoda and colleagues [7]. Speech corpus is a large collection of audio recordings and text transcription of the audio recordings [8]. This is the main part of the mathematical model based speech recognition developments. Here researchers try to develop a Sinhala speech corpus as a contribution to Sinhala speech recognition system development. Authors of the paper describe how to design a proper speech corpus focusing better speech recognition accuracy. There are two methods to develop a speech corpus. One way is to collect existing speech data and manually transcribe those audio recordings to text. Other way is create text transcription first and then record the speech by reading the text [7]. In this project they were trying to build a music request application so the corpus they created mostly relevant to the application they are creating. So it's not a general corpus. But they have collected various categories of Sinhala words like Currency, digits, Boolean data, etc... Perplexity of the language model [9] also measured to indicate the quality of the language model. The corpus they have built was a good one but in the paper they didn't discuss about the accuracy of the corpus they built.

Sandasarani has discussed about a suitable method to develop a Sinhala speech recognition system in a different research project [10]. In order to implement the speech recognition this paper suggested 2 methods. One is isolated word recognition and the other is continuous speech recognition approach. Isolated word recognition is difficult to match with practical scenarios. Because human speech is a continuous flow of words rather than word by word utterance. Here the author uses Artificial Neural Network concept for the continuous speech recognition implementation ,and Mel Frequency Cepstral Coefficients (MFCC)[11] and Dynamic Time Warping (DTW) [12] techniques used for isolated word recognition. In the research results isolated word recognition has showed better accuracy with smaller vocabulary. But with the vocabulary increases the accuracy rate become low. On the other hand continuous speech recognition showed different behavior. It didn't decrease the accuracy with the vocabulary increase. When vocabulary becomes larger the accuracy rate becomes much higher. But this project was limited to a small vocabulary of words. So the researcher has to enhance the vocabulary to implement better accurate and more general purpose speech recognition applications.

In this speech recognition related research paper Molligoda and Wijerathne discussed about the applicability of Hidden Markov Model based pattern recognition algorithms in Sinhala Speech recognition [13]. Here authors didn't try to implement any speech recognition applications but do an analysis on HMM and it's relation to the speech recognition with the help of published research papers. Main goal for this study is to elaborate the need of a Hidden Markov model and it's variants to Sinhala Speech Recognition development. According to the research paper applicability of HMM is measured using 3 methods, which are inclusion, exclusion criteria and search strategy .HMM is the most commonly used model in speech recognition research area. it is a tool for representing the possibility of occurring an event over a sequence of observation [14]. Here authors have talked about variants of HMM like Discrete Hidden Markov Model-DHMM and Continuous Density Hidden Markov Model-CDHMM[15]. According to the research HMM based recognizer use single Gaussian distribution to model the output by assuming feature models are symmetric and single modeled. But in reality multiple modes are created due to reasons like speaker, gender, language, accent differences. To resolve this problem, researchers introduced Gaussian mixture model [16],which is capable of handling asymmetric and multi model speech data. In Sinhala Speech recognition context authors found few research papers. In those researches they used only the conventional HMM with the support of HTK software packages. In this paper researcher discussed only about HMM and it's usability with related to speech recognition. But paper didn't discuss about how to apply HMM to speech recognition in depth.

Another research has conducted on developing a methodology to translate discrete Sinhala speech to Sinhala Unicode text in real time[17]. Here they used HMM approach with the support of HTK used speech recognizer. Ability to recognize Sinhala words in speaker independent scope and noisy environment are the main achievements they expect in this research project. According to the paper speech corpus was created by a single person and recorded in a quiet environment for the training. In the data collection stage vocabulary of the training was limited to 50 words, which is a medium size vocabulary according to the researcher. Once the required data was collected features of the collected data has to be analyzed. In order to do a feature analysis in speech recognition domain there are several techniques available to use. Some of those techniques are Linear Predictive Cepstral Coefficients (LPCC) , Mel-Frequency Cepstral Coefficients (MFCC) [18] , Perceptual Linear Predictive Coefficients (PLPC)[19] , Relative Spectra Filtering of Log Domain Coefficients (RASTA)[20] and Integrated Phoneme Subspace (IPS) [21]. For the feature extraction portion researchers had used MFCC technique because of the Julius decoder [22]. They use desktop application to demonstrate the Sinhala speech recognition with the support of integrated Sinhala Unicode. Here the researchers didn't thought about variants in the speech training process. Variants in speech recognition can be context, speaker and environment and they will increase the accuracy of the speech recognition system[23]. At the conclusion researchers indicate the difficulty of implementing continuous speech recognition system with Sinhala Unicode support. And also this paper points out the usefulness of Sinhala Speech recognition implementation for mobile devices.

### 2.2.1 Problem derivation/statement - Research questions

The above study shows numerous limitations of the speech recognition system. These issues are summarized in Table 2.1

Table 2.1 Summarization

| Research | Limitations |
|---|---|
| Continuous Sinhala speech recognizer [4] | Didn't implement any usable application for Sinhala Speech recognition and it is a speaker dependent system. |
| Developing a Speech Corpus for Sinhala Speech Recognition [7] | Here paper discussed only about a Sinhala Speech corpus and they didn't implement any application to demonstrate their findings and accuracy details |
| Sinhala Speech Recognition [10] | This research project vocabulary limited to a small amount of words. So the accuracy in continuous speech recognition can be a very low. |
| Applicability Of Hidden Markov Model Approach For Sinhala Speech Recognition – A Systematic Review [13] | In this paper researcher discussed only about HMM and it's usability with related to speech recognition. But paper didn't discuss about how to apply HMM to speech recognition in depth. |
| Real time Translation of Discrete Sinhala Speech to Unicode Text [17] | Here the researchers didn't thought about variants in the speech training process. |

Based on above, the research problem is defines as the inadequate attention given to methods of speech recognition.

## 2.3 Speech Recognition system approach

Speech recognition is a process of identifying the exact match of a voice signal to human readable word or a phrase. By understanding the word said by the speaker system will be able to print that word or phrase in text format. As for some applications that words recognized by the system can be a command or control input to the system.

Speech recognition systems have many different types. Some systems are capable of recognizing the continuous speech of a speaker while some other systems need a little pause in between two words when speaks. Other than that speech recognition systems have many parameters like speaking model, speaking style, vocabulary, language. There are two approaches in speech recognition. One is speaker independent and the other is speaker dependent. Speaker dependent method is much simpler than speaker independent method. Because speaker dependent method only need to train by a single speaker and that person alone can use the system. But for speaker independent method system should be trained by more than one person.

There are three main popular methodologies used in speech recognition,

- Hidden Markov model (HMM)-based speech recognition
- Dynamic time warping (DTW)-based speech recognition
- Deep Neural Networking

## 2.4 Hidden Markov model (HMM)-based speech recognition

This is the most popular modeling method used in speech recognition development. Most modern general-purpose speech recognition systems are based on Hidden Markov Models. These are statistical models that output a sequence of symbols or quantities.
In HMM-based speech recognizers, each unit of sound (usually called a phoneme) is represented by a statistical model that represents the distribution of all the voice data for that phoneme. This is called the acoustic model for that sound unit. In acoustic model creation, speech signals are transformed into a sequence of vectors at the beginning that represent identified characteristics of the speech signal, and the parameters of the acoustic model are then estimated using these vectors. Those estimated vectors are called features of the speech signal. This process is called training the acoustic models. During speech recognition, features are derived from the incoming speech in the same way as in the training process. The component of the recognizer that generates these features is called the front end. These live features are scored against the acoustic model [24].
The score obtained indicates how likely that a particular set of features (extracted from live audio) belongs to the phoneme of the corresponding acoustic model.

The process of speech recognition is to find the best possible sequence of words that will match with the given input speech. In HMM-based recognizers this process is called a graph search problem. The graph represents all possible sequences of phonemes in the entire vocabulary of the relevant context. The graph is typically composed of the HMMs of sound units concatenated in a guided manner, as specified by the grammar of the task. As an example, let's look at a simple search graph that decodes the words "එක" and "දෙක". It is composed of the HMMs of the sounds units of the words "එක" and "දෙක":

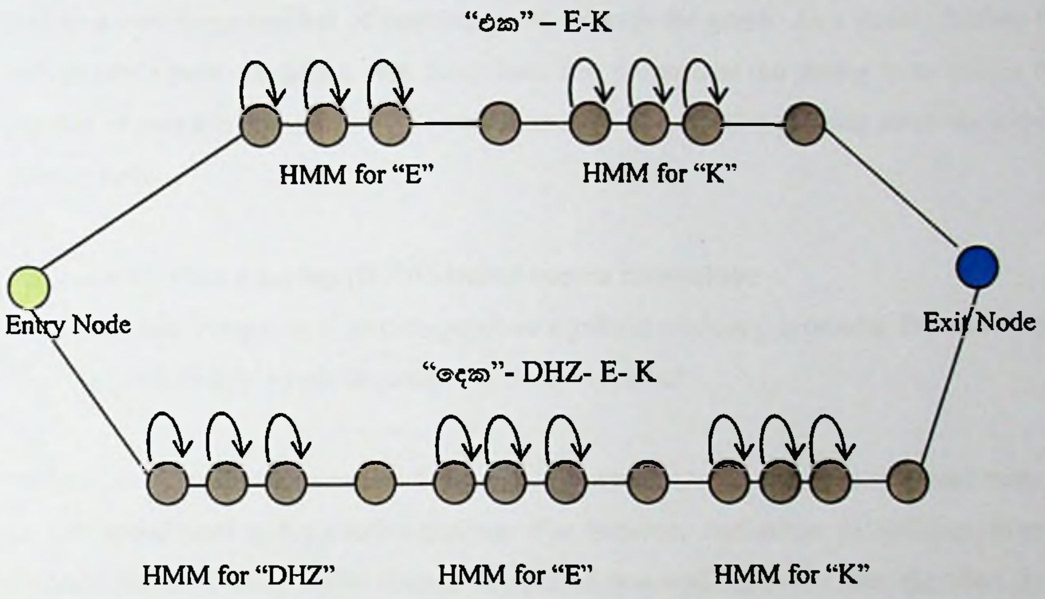


Figure 2.1: Search Graph

Constructing the above graph requires information from various sources. It requires a dictionary, which maps the word "එක" to the phonemes E and K, and the word "දෙක" to DHZ, E and K. It requires the acoustic model to obtain the HMMs for the phonemes E, K and DHZ.

Usually, the search graph also has information about probability of occurring certain words. This information is supplied by the language model. Suppose that, in our example, the probability of someone saying "එක" (e.g. 0.8) is much higher than saying "දෙක" (e.g. 0.2).Then, in the above graph, the probability of the transition between the entry node and the first node of the HMM for E will be 0.8, while the probability of the transition between the entry node and the first node of the HMM for DHZ will be 0.2. The path to "එක" will consequently have a higher score.

Once this graph is constructed, the sequence of parameterized speech signals (i.e., the features) is matched against different paths through the graph to find the best fit [25]. The best fit is usually the least cost or highest scoring path, depending on the implementation. As can be seen from the above graph, a lot of the nodes have self transitions. This can lead to a very large number of possible paths through the graph. As a result, finding the best possible path can take a very long time. The purpose of the pruner is to reduce the number of possible paths during the search, using heuristics like pruning away the lowest scoring paths.

## 2.5 Dynamic time warping (DTW)-based speech recognition

Dynamic Time Warpping is an old speech recognition modeling approach. But now it has been replaced largely by more successful HMM approach.

DTM is an algorithm to measure the similarity between two sequences that may vary in time or speed used in time series analysis. For instance, similarities in walking patterns could be detected using DTW, even if one person was walking faster than the other, or if there were accelerations and decelerations during the course of an observation. DTW has been applied to temporal sequences of video, audio and graphics data — indeed, any data which can be turned into a linear sequence can be analyzed with DTW. A well-known application has been automatic speech recognition, to cope with different speaking speeds. Other applications include speaker recognition and online signature recognition. Also it is seen that it can be used in partial shape matching application[26].

## 2.6 Deep Neural networking

Deep neural network is a combination of multiple hidden layers of units between input and output of the network. In this hidden layer units, each of which takes all outputs of the lower layer as input and process according to mathematical calculations such as *sigmoid or tanh* [27] . DNNs can model complex non-linear relationships which is more suitable for speech recognition pattern detection. And the DNN architectures itself gives a huge learning capacity and thus the potential of modeling complex patterns of speech data. Currently DNN is in use for large vocabulary speech recognition systems and it shows better performance and more accuracy. But implementation of this concept is much more complicated than HMM. More sophisticated speech recognition systems such as Google Now[27], Apple Siri [28], Microsoft Cortana [29] adopted this DNN concepts to improve their recognition capabilities.

For Sinhala language above methodologies can be applied as well. There were researches for Sinhala speech recognition using Hidden Markov Method. But there were no proper implemented researches for the mobile platforms.

## 2.7 Similar projects carried out

A similar project was carried out by a group of students at University of Moratuwa. They have developed a system called KATHANA. But it was not specifically designed for mobile devices. And KATHANA system was designed to recognize isolated words, but continuous speech recognition. Further to that there were several developments at the Language Research Laboratory at University of Colombo. They have created a large corpus of Sinhala words as well for the benefit of speech recognition researches [30]. And there were several other developments in Text- to –Speech systems by University of Colombo but very few attempts were made to develop speech-to-text systems speaker independent systems.

## 2.8 Summary

There is a rapid development on speech recognition systems in past decade. Now there are commercial products capable of recognizing the human. These systems are capable of converting the speech to text or control devices.

Speech recognition is a complex process. It can have many parameters like speaking model, speaking style, vocabulary, language. Speech recognition systems can be categorized as speaker dependent and speaker independent. For all these variations there are two main models build by researchers.

Hidden Markov Model and Dynamic Time Warping are the two models that use by most of the speech recognizer implementations. These two algorithms try to match human voice and its correlated word and use it to fulfill the application needs.

Chapter 03

# Technology enrichment of speech recognition on mobile devices

## 3.1 Introduction

Speech is a complex phenomenon. People rarely understand how is it produced and perceived. The naive perception is often that speech is built with words, and each word consists of phones. The reality is unfortunately very different. Speech is a dynamic process without clearly distinguished parts. All modern descriptions of speech are to some degree probabilistic. That means that there are no certain boundaries between units, or between words. Speech to text translation and other applications of speech are never 100% correct.

PocketSphinx is a lightweight open source large vocabulary, speaker-independent, language independent continuous speech recognition engine designed by Carnegie Mellon University specifically tuned for handheld and mobile devices; though it works equally well on the desktop as well [31]. Android operating system is fully supported by this recognition engine. Android operating system is most widely used mobile operating system on the planet. It powers hundreds of millions mobile devices in more than 190 countries around the world.

## 3.2 Pocketsphinx process

The common way to recognize the speech is by taking the waveform, split it on utterances by silences then try to recognize what's being said in each utterance. To do that system needs to take all possible combinations of words and try to match them with the audio. Pocketsphinx engine accept audio data in a form of files (.wav) or live voice input from a microphone. Once it takes the input audio signal and puts it through an analysis process that includes pre-emphasis, signal segmentation, and frequency analysis. And then the matching process starts. In matching process there are three models used,

**Acoustic Model:** Holds extracted information from sound units. These information are called data packets. Each data packet represents a part of a sound that we make when we say a word. Each data packet represents a piece of a syllable in a word. Collection of few data packets will create one syllable and many data packets will eventually create a word. This database is large because of the extent of the Sinhala language and the different parts that can make up each word.

**Phonetic Dictionary:** This contains all the Sinhala words recognized by the recognition engine and its phonetic mappings. By using this dictionary recognition engine will recognize how each and every Sinhala word is pronounced.

**Language Model:** This contains information about probabilities of a single word with related to the phrase it can be appeared. Given a set of sound data packets, this model will help to determine which word that it is likely to be the next. It also holds grammar rules so that it can determine what word will likely follow another word in a phrase. This methodology is called an N-gram based method of recognition. N-gram consists of three types, which are uni-gram, big-gram and tri-gram. Here the language model was build using tri-gram model. A tri-gram would look at the previous two words of a phrase and then using probability to determine the next likely word to come. To create language model for Sinhala language a toolkit call SRILM was used. SLILM is a statistical language modeling toolkit which support uni-code. So this was the best tool to create Sinhala language model for the purpose of speech recognition system.

**Decoder:** Brings together all the components to output the results to the user.
Other than the above components Pocketsphinx must have two parts to build into as a speech recognition engine. Those two parts are,

1. Sphinxbase – A library which is a prerequisite for the Pocketsphinx to run
2. Sphinxtrain – An acoustic model training tool, which is used to create the language model for the recognition engine.

## 3.3 Sinhala Speech Recognition with Pocketsphinx

Since the Pocketsphinx toolkit is language-independent, it was a perfect choice to develop a speech recognition system for Sinhala language. This toolkit is an offline speech recognition methodology. So it does not need any online severs to do the recognition process such as Google Now or Apple Siri. And also it support Android operating system. So the Pocketsphinx is the best option to build a good and accurate speech recognition system for mobile devices.

Following is the architectural design of the Pocketsphinx and the explanation of the process.

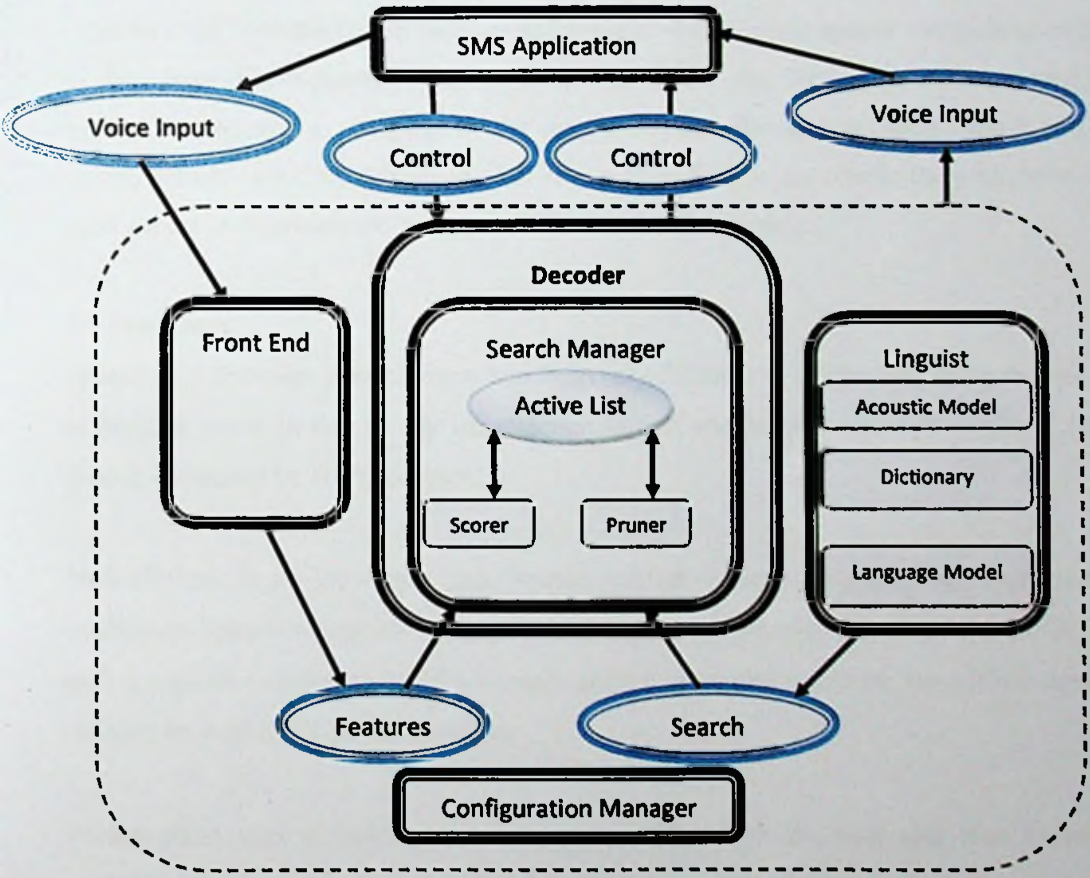


Figure 3.1: Pocketsphinx Architecture

The **recognizer** constructs the **front end** (which extract features from input speech ), the **decoder**, and the **linguist** generates the search graph according to the configuration specified by the user. Some of these components create their own subcomponents also. For example, the linguist component will use the combination of the **acoustic model**, the **dictionary**, and the **language model**. Decoder will use the knowledge gain from those three components to construct a **search graph** that is used in recognition process. Then it will build the **search manager**, which creates the **scorer**, the **pruner**, and the **active list** using the search manager information.

There are lots of parameters in Pocketsphinx that can be fine tuned. For an example sample rate of the incoming speech data can be changed accordingly. And also the architecture explains itself how this toolkit behaves as language –independent speech recognition engine. In the **Linguist** part Acoustic Model, Language Model and Dictionary are containing the words that are going to recognize by the engine. So those file contain Sinhala words and the system does not need to understand what are on those files. It just checks the probability and other aspect to determine the perfect match for the audio segment.

## 3.4 Summary

Speech is a complex phenomenon which is very difficult to understand even though we as humans use it in day to day life. Speech to text translation and other applications of speech can never be 100% correct.

PocketSphinx is a lightweight open source speaker-independent, language independent continuous speech recognition engine which supports Android operating system. So this tool is a perfect option to build a Sinhala speech recognition system for mobile devices running on Android Operating system.

Pocketsphinx uses a technique that uses the audio from the user and tries to match possible combinations of words and output the result to the user. In order to do the matching process Pocketsphinx uses Acoustic Model, Phonetic Dictionary, Language model and Decoder.

Chapter 04

# Approach to Sinhala speech recognition Using Pocketsphinx

## 4.1 Introduction

To implement Sinhala speech recognition capability on Android powered mobile devices needs a time consuming complex process. And it gets more complicated when implementing speaker- independent speech recognition system. Speaker-independent speech recognition systems need large sets of audio files for training the system from various human voices. It helps the Pocketsphinx engine to identify the most suitable matching word to the audio input.

In order to successfully implement the Sinhala speech recognition system, there are several steps to be followed. And also large amount of audio data and related text data should be gathered. It is not possible to define the limitation of collecting data set, because words in a language can be a huge number. So the data gathering part is the most complex and time consuming part in any speech recognition system for any language.

## 4.2 Proposed setup

Speech recognition with Pocketsphinx requires a huge amount of audio and word corpus. This is the main challenge when setting up the system. It is recommended having a corpus with relate to the system implementation. In this project the corpus used was related to SMS phrases commonly used by the people. Creating a general text corpus will be very complicated and time consuming task. For this project the researcher has defined the text corpus with related to the application he is going to build. In this case the text corpus was created with related to SMS phrases.

After creating the text corpus with the help of survey participants audio recordings for the relevant texts have to be recorded. Those recordings should be in mono mode and the sampling rate should be 16 KHz [32]. Since the implementation focus on speaker-independent speech recognition system, there should be recordings of the same text using different people. Accuracy of the speech recognizer depends on the number of speakers and the amount of audio data and text data. That is the technique used by Pocketsphinx to train the system to produce a language model. Once this language model created it can be used by any application using CMUsphinx tools (Eg: Pocketsphinx , Sphinx).

After creating the text corpus and the audio files, training process starts. In this training process Sphinxtrain tool will assess the word probability and frequency of occurring against the audio files. This matching process is very crucial for the final product.
Because the statistical model which creates in this process will be used in the speech recognition application.

**4.3 Survey**

In order to create a text corpus for the purpose of this project, researcher gathered most commonly used SMS phrases by conducting a survey. In this survey responses were gathered from 100 University students. The reason to select University students for the survey is that they are the people who use SMS most of the time. So they have a better knowledge about most common SMS phrases used in day to day life. The survey was designed using Google form online tool. In the survey there were 9 categories defined and participants for the survey has to write whatever they feel to be suitable for each category. Defined categories were mentioned in Appendix A and sample questionnaire stated in Appendix B.

To guide the survey participants researcher gave example phrases for each category and ask participants to write any phrase they like according to the category. The categories were defined, because without categories participants may face difficulties in responding to the survey. All the phrases were in Sinhala language and the development of this speech recognition will be based on the output of the survey. Part of the results from survey mentioned in Appendix C. After collecting the data researcher will be able to create a good text corpus to train the system accordingly.

## 4.4 Android Testing Environment

Table 4.1: provides the Android testing environment details where the speech recognition application will be installed.

Table 4.1 Android test environment details

| | |
|---|---|
| Equipment | Samsung Galaxy Star GT-S5282 |
| OS | Android OS, v4.1.2 (Jelly Bean) |
| CPU | 1 GHz Cortex-A5 |
| RAM | 512 MB |
| Internal Memory | 4 GB |

Android OS v 4.1.2 Jelly Bean fully support Sinhala Unicode characters. This was an added advantage for this project. Otherwise the Sinhala font has to be installed separately by rooting the mobile device. But there is an option in Android programming to add any font to the code. Once it is done there is no need to install the Sinhala font to the mobile device.

## 4.5 Android interface development

Designing the Android interface for the speech recognition application was done using the following tools,

- Android Studio 1.3.2
- Java jdk 1.7.0
- Android SDK 4.2

## 4.6 Recording

A speech recording for training is a vital part in training process. In order to record speech and manipulate the recordings according to the needs of CMUSphinx was done by using a tool call Audacity[33]. Audacity is an open source free to use tool. It has features to edit voice recordings much easily.

**4.7 Summary**

In order to implement speech recognition system there should be a good training data set. This training data should include a large set of Sinhala words and related audio clips. When doing audio recordings there are certain rules to be followed.

Then an Android interface should be created to demonstrate the speech recognition.

Create the Sinhala language model by using the collected data (audio clips, related text words) and input that into the speech recognition engine.

Chapter 05

# Design of Sinhala speech recognition system for Android OS

## 5.1 Introduction

Designing process is the most important in implementing speech recognition system. Because the speech recognition is a complex process and need to have covered all the possible inputs and outputs of the system.

As the initial step training process for the speech recognition system has to be designed. It has to be designed in order to get more accurate output from the speech recognition system. Once the training completes then the recognition engine part comes. Recognition engine will come under the user application part. In this project it will be the Android SMS application. This application acts as the testing environment for accuracy of the speech recognition.

## 5.2 Speech recognition design

Speech recognition process needs to follow a complex process in order to get an accurate output. Simple form of speech recognition process adapted into this project is shown below.

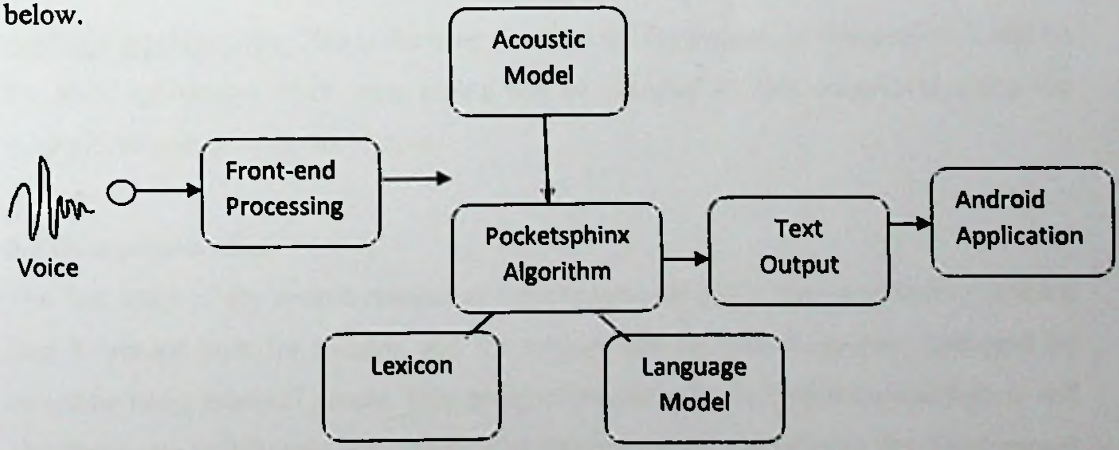


Figure 5.1: Speech recognition system design

**Front-end Processing:** The front-end of a speech recognition system is the part that transforms speech to a vector of features that is suitable for further processing. In this stage features of the input voice will be extracted using CMUsphinx algorithms.

**Pocketsphinx Algorithm:** This part will process the features extracted in the previous step in more details and create a search graph. This graph will be used in audio to text matching using acoustic model, language models and lexicon.

**Acoustic model:** Acoustic model is a database of statistical models. Each statistical model represents a single unit of speech such as a word or phoneme. In spoken Sinhala language consists of 40 distinct sounds that are useful for speech recognition.

**Language Model:** Speech recognition system tries to match sounds with word sequences. The language model provides context to distinguish between words and phrases that sound similar.

**Lexicon:** Lexicons contain the mapping between the written representations and the pronunciations of words or short phrases. And also it contains a list of Sinhala words that can be recognized by the Speech recognition engine.

**Android Application:** This is the user interface for the system. In this project it will be the SMS application. User voice inputs will be accepted by this component using the microphone and give the text output.

## 5.3 Data preparation

The first stage of any speech recognizer development project is data preparation. Speech data is needed both for training and for testing. Speech recordings were gathered by recording using group of people. This group of people includes both male and female and also they were in different age groups. The training data is used during the development of the system. Test data provides the reference transcriptions against which the recognizer's performance can be measured. For the training data a pronunciation dictionary created to provide initial phone level transcriptions that are needed by the HMM training process.

Structure of the data files needed by the CMUsphinx speech recognition system can be elaborate as follows,

Table 5.1: CMUSphinx data file details

| File Type | Detail |
|---|---|
| db_name.dic | Phonetic dictionary |
| db_name.phone | Phoneset file |
| db_name.lm | Language model |
| db_name.filler | List of fillers (eg: silence, cough, noises etc.) |
| db_name _train.fileids | List of recording's location for training |
| db_name_train.transcription | Speech recordings and it's correlated text for training |
| db_name _test.fileids | List of recording's location for testing |
| db_name _test.transcription | Speech recordings and it's correlated text for testing |

## 5.4 Recordings

Speech recordings in this project were done in a quite environment to prepare a better training data set. Noises were recorded separately and added to the relevant data files to ignore whenever it occurs. Since the project is speaker independent speech recognition system, wide varieties of voices need to be recorded. All the recordings should be labeled properly to use those in next steps in the process more conveniently. All the recordings must be in .wav format, mono mode and 16 kHz sampling rate. Those are the settings needed by the CMUsphinx to works properly. Microphone used for recordings must product correspondingly uninterrupted voice level within the session of operating. Some of the voice recordings and noise recordings recorded from mobile phone. A tool called Smart Voice Recorder was used in this situation.

## 5.5 Training

Once the text files and recordings are prepared, the training process can begin. All the data should be in compatible form to input as training data. If any inconsistency persists in the process of training CMUSphinx tool will show the errors occurred. If any errors occurred it has to be manually corrected until the training process successfully complete. Most of the errors occur due to training algorithm detect recordings were not matched to the corresponding text. In those cases most of the time only solution is to rerecord the particular phrase and do the training process again. Basic training process is shown in the figure below,

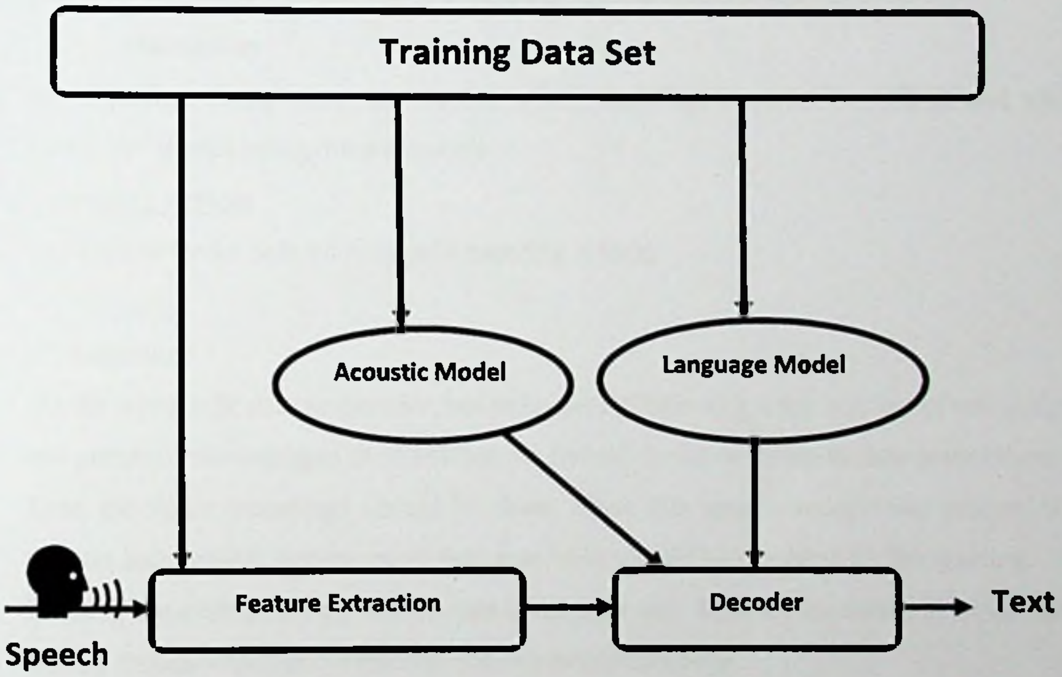


Figure 5.2: CMUSphinx Training Process

**Training Data Set:** All the text data and recorded voice data.

**Feature Extraction:** Here it is same as Front-end processing.

**Decoder:** The most important component of the system is the decoder. It does most of the work. It reads features extracted in previous stage, couple those data with the knowledge base (Acoustic model and Language model) and performs a search to determine the most appropriate sequence of words that can be represented by those features.

## 5.6 Design assumptions and dependencies

Here discuss about assumptions and dependencies which may relate to hardware, software and other things.

<u>**Hardware**</u>

- Dependencies

Speech recognition system highly depends on the clarity of input wave signal. So a good quality microphone will increase the accuracy level to a considerable percentage.

- Assumption

Good quality microphones were used in when recording the voice recordings and when testing the speech recognition accuracy.

<u>**Operating System**</u>

The system works only on Android Operating system.

## 5.7 Summary

As the initial task data preparation has to be done. Collecting a fair amount of text corpus and prepare it according to the CMUSphinx formats is the next step in data preparation. Then the voice recordings should be done. Since this speech recognition system is a speaker independent system, more than one voice should be recorded for the training. Training the system using gathered data is the next step. Here all the data will be checked for any inconsistency and trained to achieve better accuracy.

Chapter 06

# Implementation process of Speech Recognition System

## 6.1 Introduction

For the implementation process first the data should be prepared for the speech recognition system.

Then the gathered data put into the training tool and get the output to use in Pocketsphinx.

Once the trained data is prepared the Android interfaces for the testing should be prepared. As for the interim stage prepared interface only used as the output interface to show recognized words. There are no options to send or receive SMSs as it now.

## 6.2 Collecting data set

This is the most important and most difficult step in the whole process. Text corpus need to be a large one in order to get an accurate speech recognition output. One option to collect data is the survey conducted among University students. All the data collected were used as the testing data set. Other texts were copied from sources like web sites, news papers etc...

More than 10 people were contributing for the voice recordings both male and female. And also different age groups also considered in recording the corpus. Around 1000 words and more than 400 phrases were recorded in different locations. It was not a total silent environment. Some of the noticeable noises were removed using the Audacity tool and labeled properly to avoid confusions in next phases.

## 6.3 Creating Language Model File

In this step the SRILM toolkit used to create the Sinhala language model. SRILM is an open source toolkit which gives the probability of words using N-Gram and Tri-Gram methodologies. First download the toolkit and installed on Windows 8.1 operating system using Cygwin [34]. Cygwin is a Linux like environment design for Windows platforms. Once the toolkit installed, using text corpus created earlier the language model file created. Sample format of a Language Model (LM) file can be seen in Appendix D. once the language model file created it has to be sort. To do the sorting a researcher uses a tool comes with sphinxtrain. It will sort the language model based on the N-gram value given during the language model creation phase.

## 6.4 Creating Phonetic Dictionary File

This is another important file which consists of all the words that can be recognized by the recognizer and the phonemes for each word. Sample file can be seen in Appendix E. and a complete phonetic list of Sinhala letters can be seen in Appendix F[2]. In order to create the phonetic representation of the Sinhala words English letters were used. To get the correct phonetic letters which corresponds to the Sinhala word a MS Excel macro was used.

## 6.5 Creating other files

All the other files were created manually with the help of MS Excel. The files are,

- db_name.phone – sample shown in Appendix G
- db_name.filler - sample shown in Appendix H
- db_name _.fileids - sample shown in Appendix I
- db_name _.transcription - sample shown in Appendix J

## 6.6 Training

This is the most important stage in the whole process. This is where all the data is going through an estimation process, which creates a model that can be used in Pocketsphinx recognition engine. To setup the training environment a virtualized Ubuntu 14.02 system was used. In the Ubuntu system all the dependencies and main two components were installed.

- Sphinxbase
- Sphinxtrain

Those two main tools were used to do the training process. Once the training process completes output model files were copied to the local PC.

## 6.7 Creating Android Environment

First step to download Android studio from official Android web site. Once it's installed all the required components for the development of Android applications were installed. Components like SDK platform, Android Development Tools and Android Virtual Device Emulator installed.

Once the Android Studio installation completes Pocketsphinx recognizer engine development can be done. All the Pocketsphinx Libraries can be imported to the program using,

```
Import edu.cmu.pocketsphinx.Hypothesis;
import edu.cmu.pocketsphinx.RecognitionListener;
import edu.cmu.pocketsphinx.SpeechRecognizer;
```

All the coding was written using Android studio environment. Detailed coding of the recognizer module can be seen in Appendix K.

Once the Android interfaces build model parameters created from the CMUSphinx training can be import into the android coding using the Asset folder. The content of the model parameters folder in the CMUSphinx training environment copied into the Asset folder of the android development folder. Then the recognition system can be tested using AVD or a real phone by connecting it to the Android Studio through USB cable.

## 6.8 Summary

First prepare the text corpus and voice record data for the purpose of training the system. Then create the Language Model using the SRILM toolkit. Once the LM created Phonetic dictionary and other required files were created with the help of MS Excel.

Start the training using prepared data in the training environment created in the Ubuntu operating system. Once the training completes it will creates the model parameter set according to the input data.

Then create the Android environment using Android Studio software. Android interface and Pocketsphinx libraries were imported into the coding. By using the model parameters created in the training process Android application can be tested for the accuracy.

Chapter 07

# Evaluation

## 7.1 Introduction

Analyzing the results, test it and evaluate the product is the most important part in any project. As the final stage of the project researcher did this analyzing part to check the system accuracy. Speech recognition accuracy can be measured in two stages.

- During the training process
- Testing on real application (Manual test)

## 7.2 During training process

In this phase Sphinxtrain tool will output the error rate of the trained dataset at the decoding stage of the training process. CMUsphix calculate the error rate using the below mentioned formula,

$$\mathbf{WER = (I + D + S) / N}$$

WER- Word Error Rate

I – Number of Inserted words

D – Number of Deleted Words

S – Number of Substituted Words

N- Total Number of Words

Using the above formula a script will compute the WER in percentage. If the percentage value is low means better accuracy of the system. If it's very high means very poor accuracy in the system.

## 7.3 Testing on Real Application

Here the testing is done using a manual process. Researcher uses the same android application created to demonstrate the project outcome to test the accuracy. According to the researcher point of view this is the better accuracy measurement procedure rather than the system generated WER. Therefore researcher did the testing using this method to evaluate the system.

In this process 4 people were participated to test the application. Bellow table illustrate the conditions of the selected people.

Table 7.1 Group of people use to test the system

| Person | Gender | Status |
|---|---|---|
| Person 1 | Male | Participate in Training Process |
| Person 2 | Male | Not Participate in Training Process |
| Person 3 | Female | Participate in Training Process |
| Person 4 | Female | Not Participate in Training Process |

They were given a set of SMS phrases randomly selected from trained data set. Total numbers of words in the selected SMS phrases were 144. And then Sinhala speech recognition application installed mobile device has given them for the testing. And then ask each participant to go inside a room and read out the SMS phrase one by one. Ask the participant to repeat the task for 3 iterations to gather more reliable test data set. In the room, only the researcher and the participant were present. Once done with the testing another participant was asked to come to the room to do the testing.

## 7.4 Testing Conditions

Testing conditions were as follows,

- Testing was done by isolating each participant in a room.
- All the people were give same set of SMS phrases used during the testing process.
- 40 randomly selected SMS phrases were given to the participants. (Which is nearly 10% of the total number of phrases.)
- All the given 40 SMS phrases were used in training process.
- 3 iterations were tested using the same 40 SMS phrases.
- Noise measuring application was used to measure the environmental noise level during the testing and recorded. (Testing person sound also included in here).
- Accuracy measurements have captured for words uttered by the participants.
- Uses same mobile device during the whole testing process.

- If a participant made a mistake in pronouncing a word or struggle to complete a sentence perfectly, participant was asked to repeat the same sentence again.

## 7.5 Assumptions

Following are the assumptions made during the testing process.

- All the participants have same amount of knowledge of this project.
- All the participants have same knowledge in operating mobile devices.
- Any of the participants didn't have any speech disorders.

## 7.6 Results

Results of the testing are illustrated in bellow table.

Table 7.2 Results

| **Participant** | **Number of words Displayed correctly in the Application** | | | **Average Environmental Noise During the Testing (dB)** | | | **Average Accuracy** |
|---|---|---|---|---|---|---|---|
| | **Attempt 1** | **Attempt 2** | **Attempt 3** | **Attempt 1** | **Attempt 2** | **Attempt 3** | |
| Person 1 | 29 | 30 | 30 | 20 | 19 | 20 | 20.6 |
| Person 2 | 17 | 19 | 19 | 21 | 20 | 20 | 12.7 |
| Person 3 | 26 | 27 | 27 | 20 | 18 | 19 | 18.5 |
| Person 4 | 16 | 16 | 17 | 19 | 21 | 20 | 11.3 |

Researcher uses a simple formula to calculate accuracy in the testing. Following is the formula used,

**Accuracy = (C/N) x 100%**

C – Total Number of words displayed correctly in the SMS application

N – Total number of words used in the testing

Chapter 08

# Conclusion and Future Work

## 8.1 Introduction

Speech is the most fundamental way of communicating with each other. This has evolved thousand years and now there are lots of languages spoken by humans. After the discovery of computers researchers and scientists try to couple speech and IT systems. Because of these attempts speech recognition concepts, methods and algorithms were discovered. At the earlier stages accuracy of the speech recognition was so poor and researchers try to improve the accuracy of the recognition process. Now there is a huge development in speech recognition area. For the commonly used languages like English more than 90 % accuracy can be guaranteed. But for the languages like Sinhala lots of improvements need to be made.

## 8.2 Completed work and future work

This project is a tryout to improve the recognition accuracy of the Sinhala language using an open source toolkit call CMUSphinx. CMUSphinx is a speech recognition engine developed by Carnegie Mellon University. It's a language independent speech recognition engine. That was the main reason to select this tool for the development of Sinhala speech recognition system. Researchers in Sri Lanka are more into this subject in recent past. There are some Sinhala speech recognition developments in Sri Lanka, but there couldn't be found any developments done using CMUSphinx for Android Operating system. And also most of the developments are speaker dependent or restricted to limited number of words like recognizing digits. But this project doesn't contain such small amount of recognition dictionary. So this system is more generic than other systems. Once this project completes any other researcher can improve the model and can be used in their applications.

Since the speech recognition for Sinhala language developed only a little I have to do the implementation from the scratch. Speech recognition is a complex process. So it was a difficult task to understand the concepts behind the process and the toolkits used for the process.

Another challenge was the preparation of data set. Most of the projects on speech recognition systems take more than 2 -3 years, because of the data preparation process consumes lot of time. Since this project duration limited to a 1 year, scope of the project was reduced to match with the time limitations. However I have managed to reduce the time needed for some tasks by using tools, scripts and applications like MS Excel.
And also collecting audio recordings in a minimum noise environment was a challenge. I have tried to do the recordings in a quite environment as much as possible. Editing the recorded voice clips also consumes lot of time and effort. In addition to that collecting voice recordings from various people was a big challenge. Since the system is speaker independent, different voices has to be used to train the system. To overcome the challenge I had to get help from my friends and family members.

For the successful implementation of speech recognition systems a huge number of data sets need to be created. Both audio and text corpus need to be prepared and it consumes lot of time. At the end of this project only a limited number of data sets were created and tested. By calculating the test results researcher has come to a conclusion that the average accuracy of the system is15.7%.It has to be improved a lot and should gain less error rate by increasing the data set and reduce inconsistencies in audio recordings.

According to the results captured during the testing process, speech recognition system build in this project does not produce 100% accuracy. None of the speech recognition system reaches to that point so far. Therefore the researchers who are interested in speech recognition projects have to improve the speech recognition concepts and methods.
Here the researcher has shared his work with the speech recognition community as a contribution to the researchers who are interested in speech recognition area. They can refer the work done by this project and continue with this work by adding more words to the corpus and more voice recordings to improve the accuracy. As further work researcher can enhance the application to accept speech commands and controls to

operate the SMS application by improving the accuracy, do certain changes in android application and add certain files to training process.

## 8.3 Summary

Here the research tries to implement a better accurate Sinhala Speech recognition system for Android powered mobile devices. In his attempt to successfully complete the work he faces many difficulties like time limitations, resource limitations etc... speech recognition systems cannot be build to produce 100% accurate results. All most all the speech recognition systems currently available are still improving to get better performance and better accuracy.

However researcher has managed to gain a respectable accuracy rate for a language which is not used very commonly in speech recognition research works. In addition to that researcher has contributed his work for the betterment for the researchers who are interested in Sinhala speech recognition developments.

# References

[1] "Bbc.co.uk, 'BBC - Languages - Languages - Languages of the world - Interesting facts about languages.'" [Online]. Available: http://www.bbc.co.uk/languages/guide/languages.shtml. [Accessed: 14-Nov-2015].

[2] A. Wasala and K. Gamage, "Research Report on Phonetics and Phonology of Sinhala," University of Colombo School of Computing, Colombo.

[3] "CMU Sphinx." .

[4] T. Nadungodage and R. Weerasinghe, "Continuous Sinhala speech recognizer," in *Conference on Human Language Technology for Development, Alexandria, Egypt*, 2011, pp. 2–5.

[5] S. Furui, "AUTOMATIC SPEECH RECOGNITION AND ITS APPLICATION TO INFORMATION EXTRACTION." Tokyo institute of Technology.

[6] "Neural Networks." [Online]. Available: https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html. [Accessed: 28-Jan-2016].

[7] T. Nadungodage, V. Welgama, and R. Weerasinghe, "Developing a Speech Corpus for Sinhala Speech Recognition." .

[8] C. Richey, "Speech Corpora." Apr-2000.

[9] S. Chen, D. Beeferman, and R. Rosenfeld, "EVALUATION METRICS FOR LANGUAGE MODELS," Carnegie Mellon University, Pittsburgh, PA 15213.

[10] N. a. C. Sandasarani, "Sinhala Speech Recognition," *Int. J. Eng. Res. Technol.*, vol. Vol. 4, no. Issue 10, pp. 391–394, Oct. 2015.

[11] S. Molau, M. Pitz, R. Schl¨uter, and H. Ney, "COMPUTING MEL-FREQUENCY CEPSTRAL COEFFICIENTS ON THE POWER SPECTRUM," RWTH Aachen – University of Technology, 52056 Aachen, Germany.

[12] C. A. Ratanamahatana and E. Keogh, "Everything you know about Dynamic Time Warping is Wrong," University of California, Riverside, CA 92521.

[13] L. A. D. S. A. Molligoda and P. G. Wijayarathna, "APPLICABILITY OF HIDDEN MARKOV MODEL APPROACH FOR SINHALA SPEECH RECOGNITION – A SYSTEMATIC REVIEW," University of Kelaniya.

[14] Z. Ghahramani, "An introduction to Hidden Markov Model and Bayesian Networks," *Int. J. Pattern Recognit. Artif. Intell.*

[15] P. Dymarski, *HIDDEN MARKOV MODELS, THEORY AND APPLICATIONS*, First. India: InTech, 2011.

[16] D. Reynolds, "Gaussian Mixture Models." MIT Lincoln Laboratory.

[17] M. K. H. Gunasekara and R. G. N. Meegama, "Real time Translation of Discrete Sinhala Speech to Unicode Text," University of Sri Jayewardenepura.

[18] T. Gulzar, A. Singh, and S. Sharma, "Comparative Analysis of LPCC, MFCC and BFCC for the Recognition of Hindi Words using Artificial Neural Networks," *Int. J. Comput. Appl.*, vol. Volume 101– No.12, p. 6, Sep. 2014.

[19] F. H¨onig, G. Stemmer, C. Hacker, and F. Brugnara, "Revising Perceptual Linear Prediction (PLP)," Universit¨at Erlangen-N¨urnberg, Germany.

[20] U. Shrawankar, "TECHNIQUES FOR FEATURE EXTRACTION IN SPEECH RECOGNITION SYSTEM : A COMPARATIVE STUDY," SGB Amravati University, Amravati.

[21] U. Shrawankar and V. Thakare, "Feature Extraction for a Speech Recognition System in Noisy Environment: A Study," 2010, pp. 358–361.

[22] "Open-Source Large Vocabulary CSR Engine Julius." [Online]. Available: http://julius.osdn.jp/en_index.php. [Accessed: 03-Feb-2016].

[23] M. A. Anusuya and S. K. Katti, "Speech Recognition by Machine: A Review," *Int. J. Comput. Sci. Inf. Secur.*, vol. Vol. 6, p. 25, 2009.

[24] K. Lee, *Context-independent phonetic hidden Markov models for speaker-independent continuous speech recognition*, vol. vol. 38, 4 vols. 1990.

[25] K. Lee, "Automatic speech recognition." Boston,Mass: Kluwer, 1989.

[26] "GenTXWarper - Mining gene expression time series." .

[27] "The Google app," *The Google app*. [Online]. Available: https://www.google.com/search/about/. [Accessed: 04-Mar-2016].

[28] "iOS - Siri," *Apple*. [Online]. Available: http://www.apple.com/ios/siri/. [Accessed: 04-Mar-2016].

[29] "What is Cortana? - Windows Help," *windows.microsoft.com*. [Online]. Available: http://windows.microsoft.com/en-us/windows-10/getstarted-what-is-cortana. [Accessed: 04-Mar-2016].

[30] "Language Technology Research Laboratory." [Online]. Available: http://www.ucsc.cmb.ac.lk/ltrl/?page=downloads&lang=en&style=default. [Accessed: 04-Mar-2016].

[31] "cmusphinx/pocketsphinx," *GitHub*. [Online]. Available: https://github.com/cmusphinx/pocketsphinx. [Accessed: 04-Mar-2016].

[32] "Training Acoustic Model For CMUSphinx [CMUSphinx Wiki]." [Online]. Available: http://cmusphinx.sourceforge.net/wiki/tutorialam. [Accessed: 04-Mar-2016].

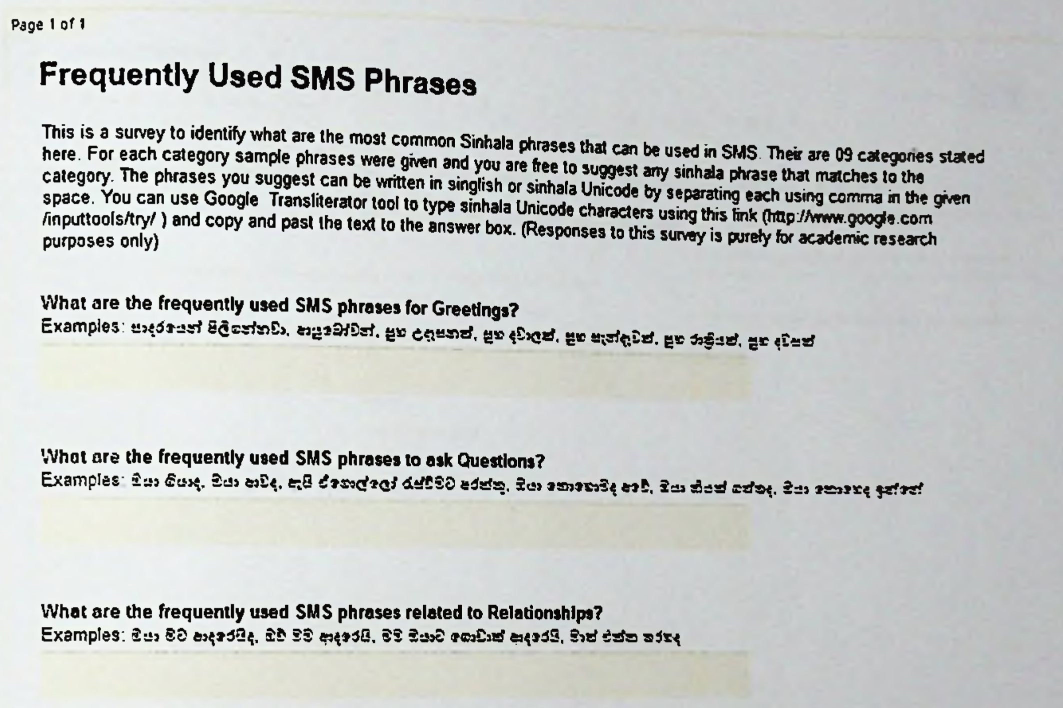
[33] "Audacity®." .

[34] "Cygwin." [Online]. Available: https://www.cygwin.com/. [Accessed: 04-Mar-2016].

## Appendix A

1. SMS phrases for Greetings
2. SMS phrases for asking Questions
3. SMS phrases related to Relationships
4. SMS phrases related to Students & Exams
5. SMS phrases to mention When & Where
6. SMS phrases for Invitations
7. SMS phrases for appreciation
8. SMS phrases for Emergency
9. SMS phrases to express feelings

# Appendix B



Page 1 of 1

## Frequently Used SMS Phrases

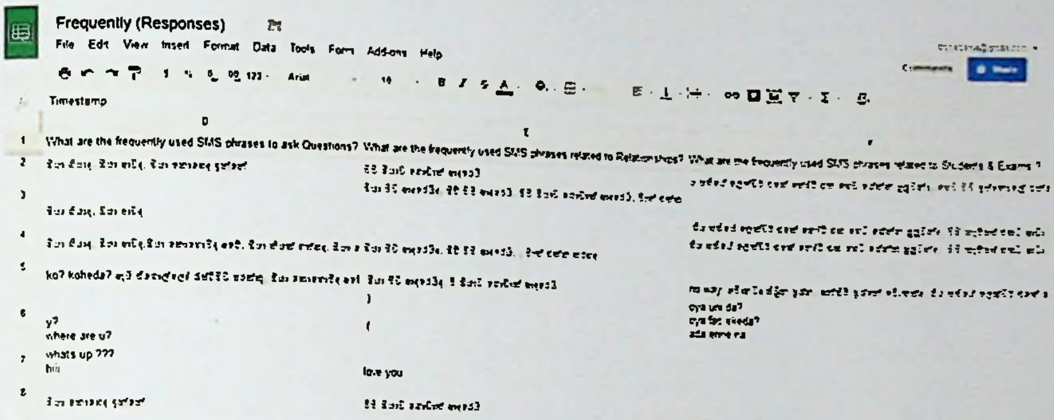
This is a survey to identify what are the most common Sinhala phrases that can be used in SMS. Their are 09 categories stated here. For each category sample phrases were given and you are free to suggest any sinhala phrase that matches to the category. The phrases you suggest can be written in singlish or sinhala Unicode by separating each using comma in the given space. You can use Google Transliterator tool to type sinhala Unicode characters using this link (http://www.google.com/inputtools/try/ ) and copy and past the text to the answer box. (Responses to this survey is purely for academic research purposes only)

**What are the frequently used SMS phrases for Greetings?**
Examples: සාදරයෙන් පිළිගන්නවා, ආයුබෝවන්, සුබ උදෑසනක්, සුබ දවාලක්, සුබ සැන්දෑවක්, සුබ රාත්‍රියක්, සුබ දවසක්

**What are the frequently used SMS phrases to ask Questions?**
Examples: ඔයා කියද, ඔයා කවුද, [illegible] [illegible] [illegible] [illegible], ඔයා කොහොමද [illegible], ඔයා [illegible] [illegible], ඔයා කොහෙද [illegible]

**What are the frequently used SMS phrases related to Relationships?**
Examples: ඔයා මට ආදරෙයිද, ඔබි මම ආදරෙයි, මම ඔයාට ගොඩක් ආදරෙයි, මාත් එක්ක තරහද

# Appendix C



Frequently (Responses)

File Edit View Insert Format Data Tools Form Add-ons Help

Timestamp

|  | D | E | F |
|---|---|---|---|
| 1 | What are the frequently used SMS phrases to ask Questions? | What are the frequently used SMS phrases related to Relationships? | What are the frequently used SMS phrases related to Students & Exams ? |
| 2 | [illegible] | [illegible] | [illegible] |
| 3 | [illegible] | [illegible] |  |
| 4 | [illegible] | [illegible] | [illegible] |
| 5 | ko? koheda? [illegible] | [illegible] | [illegible] |
| 6 | y?<br>where are u? | )<br>( | cya [illegible] da?<br>cya [illegible] eeda?<br>[illegible] |
| 7 | whats up ???<br>hiii | love you |  |
| 8 | [illegible] | [illegible] |  |

# Appendix D

```
\data\
ngram 1=578
ngram 2=3540
ngram 3=407

\1-grams:
-5.2121 0 -0.1680
-5.2121 1 -0.1680
-5.2121 2 -0.1680
-5.2121 3 -0.1680
-5.2121 4 -0.1680
-5.2121 5 -0.1680
-5.2121 6 -0.1680
-5.2121 7 -0.1680
-5.2121 8 -0.1680
-5.2121 9 -0.1680
-0.5786 </s> 0.0000
-99.0000 <s> -1.0147
-3.6439 අංකය -0.4351
-3.3992 අඟහරුවාදා -0.2699
-4.2170 අඩන්න 0.0000
-3.4412 අතරමං -0.6011
-3.2078 අත්සන් -0.7666
-4.1707 අතුරුපස -0.1328
-3.3488 අතේ -0.2976
-1.9804 අද -0.3949
-4.2170 අදනම් 0.0000
-5.2121 අදවත් -0.3009
```

```
-2.6619 අධ්‍යාපන -1.3828
-3.5589 අන්තිම -0.3010
-4.1707 අනිද්දා -0.1657
-4.5131 අනිවාර්යෙන්ම -0.1680
-3.2436 අනුමත -0.6554
-3.3036 අනේ -0.4151
-4.5131 අප්සෙට් 0.0000
-1.9793 අපි -0.1885
-2.7921 අපිට -0.3704
-1.9561 අපේ -0.5345
-2.8055 අමතක -0.8056
-4.2170 අමතකම 0.0000
-4.3670 අමතන්න -0.6451
```

# Appendix E

| | |
|---|---|
| ආයුබෝවන් | AAZ Y U B OO V N |
| සුභ | SHZ U B |
| උදෑසනක් | U DHZ AAE SHZ N K |
| රාත්‍රියක් | R AA THZ RA R I Y K |
| දවාලක් | DHZ V AA L K |
| සැන්දෑවක් | SHZ AE N DHZ AAE V K |
| කොහොමද | K O H O M DHZ |
| බුදුසරණයි | B U DHZ U SHZ R N Y I |
| දවසක් | DHZ V SHZ K |
| වේවා | V EE V AA |
| දෙවි | DHZ E V I |

# Appendix F

| | |
|---|---|
| අ | a |
| ආ | aaz |
| ඇ | aez |
| ඈ | awz |
| ඉ | i |
| ඊ | iiz |
| උ | u |
| ඌ | uuz |
| ඍ | ri |
| ඎ | riiz |
| ඏ | ilu |
| ඐ | iluuz |
| එ | e |
| ඒ | eez |
| ඓ | aiz |
| ඔ | o |
| ඕ | ooz |
| ඖ | auz |
| ක | k |
| ඛ | k |
| ග | g |
| ඝ | g |
| ඞ | q |
| ඟ | ngz |
| ච | c |
| ඡ | c |
| ජ | j |
| ඣ | j |
| ඤ | cnz |

| | |
|---|---|
| ඥ | jcnz |
| ඦ | njz |
| ට | t |
| ඨ | t |
| ඩ | d |
| ඪ | d |
| ණ | n |
| ඬ | ndz |
| ත | thz |
| ථ | thz |
| ද | dhz |
| ධ | dhz |
| න | n |
| ඳ | ndz |
| ප | p |
| ඵ | p |
| බ | b |
| භ | b |
| ම | m |
| ඹ | mbz |
| ය | y |
| ර | r |
| ල | l |
| ව | v |
| ශ | shz |
| ෂ | shz |
| ස | shz |
| හ | h |
| ළ | l |
| ෆ | f |
| ං | q |
| ඃ | h |

## Appendix G

NDZ

AEZ

EEZ

II

OOZ

J

NGZ

IIZ

SIL

+GARBAGE+

+NOISE+

+BREATH+

## Appendix H

```
<s> SIL
</s> SIL
<sil> SIL
++CHAIRSQUEAK++          +NOISE+
++COUGH++        +NOISE+
```

## Appendix I

tracks/713-Audio_Track

tracks/714-Audio_Track

tracks/715-Audio_Track

tracks/716-Audio_Track

tracks/717-Audio_Track

tracks/718-Audio_Track

tracks/719-Audio_Track

tracks/720-Audio_Track

tracks/721-Audio_Track

tracks/722-Audio_Track

tracks/723-Audio_Track

tracks/724-Audio_Track

tracks/725-Audio_Track

tracks/726-Audio_Track

tracks/727-Audio_Track

tracks/728-Audio_Track

# Appendix J

<s> මට තව පැය භාගෙකින් කෝල් කරන්න </s> (1390-Audio_Track)

<s> මට තව විනාඩි පහක් දෙන්න </s> (1391-Audio_Track)

<s> 0 </s> (1392-Audio_Track)

<s> 1 </s> (1393-Audio_Track)

<s> 2 </s> (1394-Audio_Track)

<s> 3 </s> (1395-Audio_Track)

++SINGING++ (noise_0001)

++MOVEMENT++ (noise_0002)

++TYPING++(noise_0005)

# Appendix K

```
package edu.cmu.pocketsphinx.sinhala;
import java.io.File;
import java.util.concurrent.LinkedBlockingQueue;
import android.media.AudioFormat;
import android.media.AudioRecord;
import android.media.MediaRecorder;
import android.os.Bundle;
import android.os.Environment;
import android.util.Log;
import edu.cmu.pocketsphinx.Config;
import edu.cmu.pocketsphinx.Decoder;
import edu.cmu.pocketsphinx.Hypothesis;
import edu.cmu.pocketsphinx.pocketsphinx;
public class RecognizerTask implements Runnable {

  class AudioTask implements Runnable {
    /**
     * Queue on which audio blocks are placed.
     */
    LinkedBlockingQueue<short[]> q;
    AudioRecord rec;
    int block_size;
    boolean done;

    static final int DEFAULT_BLOCK_SIZE = 1024;

    AudioTask() {
      this.init(new LinkedBlockingQueue<short[]>(), DEFAULT_BLOCK_SIZE);
    }
```

```
AudioTask(LinkedBlockingQueue<short[]> q) {
  this.init(q, DEFAULT_BLOCK_SIZE);
}

AudioTask(LinkedBlockingQueue<short[]> q, int block_size) {
  this.init(q, block_size);
}

void init(LinkedBlockingQueue<short[]> q, int block_size) {
  this.done = false;
  this.q = q;
  this.block_size = block_size;
  //HiepNH - CHANNEL_IN_MONO -> CHANNEL_IN_STEREO
  this.rec = new AudioRecord(MediaRecorder.AudioSource.DEFAULT, 16000,
      AudioFormat.CHANNEL_IN_MONO,
      AudioFormat.ENCODING_PCM_16BIT, 8192);
}

public int getBlockSize() {
  return block_size;
}

public void setBlockSize(int block_size) {
  this.block_size = block_size;
}

public LinkedBlockingQueue<short[]> getQueue() {
  return q;
}
```

```
  public void stop() {
    this.done = true;
  }

  public void run() {
    this.rec.startRecording();
    while (!this.done) {
      int nshorts = this.readBlock();
      if (nshorts <= 0)
        break;
    }
    this.rec.stop();
    this.rec.release();
  }

  int readBlock() {
    short[] buf = new short[this.block_size];
    int nshorts = this.rec.read(buf, 0, buf.length);
    if (nshorts > 0) {
      Log.d(getClass().getName(), "Posting " + nshorts + " samples to queue");
      this.q.add(buf);
    }
    return nshorts;
  }
}

/**
 * PocketSphinx native decoder object.
 */
Decoder ps;
/**
```

```
 * Audio recording task.
 */
AudioTask audio;
/**
 * Thread associated with recording task.
 */
Thread audio_thread;
/**
 * Queue of audio buffers.
 */
LinkedBlockingQueue<short[]> audioq;
/**
 * Listener for recognition results.
 */
RecognitionListener rl;
boolean use_partials;

enum State {
  IDLE, LISTENING
};
enum Event {
  NONE, START, STOP, SHUTDOWN
};

Event mailbox;

public RecognitionListener getRecognitionListener() {
  return rl;
}

public void setRecognitionListener(RecognitionListener rl) {
    this.rl = rl;
  }
  public void setUsePartials(boolean use_partials) {
    this.use_partials = use_partials;
  }
  public boolean getUsePartials() {
    return this.use_partials;
  }
  public RecognizerTask(File sphinxDirectory) {
     pocketsphinx.setLogfile(sphinxDirectory.getAbsolutePath()+"/pocketsphinx.log");
     Config c = new Config();
    c.setString("-hmm", sphinxDirectory + "/hmm/sinhala");
    c.setString("-dict", sphinxDirectory + "/dict/test06.dic");
    c.setString("-lm", sphinxDirectory + "/lm/test06.lm");
    c.setString("-rawlogdir", sphinxDirectory.getAbsolutePath()); // Only use it to store
the audio
    c.setFloat("-samprate", 16000.0);
    c.setInt("-maxhmmpf", 2000);
    //c.setInt("-maxwpf", 10);
    //c.setInt("-pl_window", 2);
    //c.setBoolean("-backtrace", true);
    //c.setBoolean("-bestpath", false);
    this.ps = new Decoder(c);
    this.audio = null;
    this.audioq = new LinkedBlockingQueue<short[]>();
    this.use_partials = true;
    this.mailbox = Event.NONE;
  }

  public void run() {
    /* Main loop for this thread. */
    boolean done = false;
    /* State of the main loop. */
    State state = State.IDLE;
    /* Previous partial hypothesis. */
    String partial_hyp = null;

    while (!done) {
      Event todo = Event.NONE;
      synchronized (this.mailbox) {
        todo = this.mailbox;
        if (state == State.IDLE && todo == Event.NONE) {
          try {
            Log.d(getClass().getName(), "waiting");
            this.mailbox.wait();
            todo = this.mailbox;
            Log.d(getClass().getName(), "got" + todo);
          } catch (InterruptedException e) {
            /* Quit main loop. */
            Log.e(getClass().getName(), "Interrupted waiting for mailbox, shutting
down");
            todo = Event.SHUTDOWN;
          }
        }

        this.mailbox = Event.NONE;
      }
      switch (todo) {
      case NONE:
        if (state == State.IDLE)
          Log.e(getClass().getName(), "Received NONE in mailbox when IDLE,
threading error?");
```

```
          break;
        case START:
          if (state == State.IDLE) {
            Log.d(getClass().getName(), "START");
            this.audio = new AudioTask(this.audioq, 1024);
            this.audio_thread = new Thread(this.audio);
            this.ps.startUtt();
            this.audio_thread.start();
            state = State.LISTENING;
          }
          else
            Log.e(getClass().getName(), "Received START in mailbox when
LISTENING");
          break;
        case STOP:
          if (state == State.IDLE)
            Log.e(getClass().getName(), "Received STOP in mailbox when IDLE");
          else {
            Log.d(getClass().getName(), "STOP");
            assert this.audio != null;
            this.audio.stop();
            try {
              this.audio_thread.join();
            }
            catch (InterruptedException e) {
              Log.e(getClass().getName(), "Interrupted waiting for audio thread, shutting
down");
              done = true;
            }
            short[] buf;
            while ((buf = this.audioq.poll()) != null) {
```

```
            Log.d(getClass().getName(), "Reading " + buf.length + " samples from 
queue");
            this.ps.processRaw(buf, buf.length, false, false);
          }
          this.ps.endUtt();
          this.audio = null;
          this.audio_thread = null;
          Hypothesis hyp = this.ps.getHyp();
          if (this.rl != null) {
            if (hyp == null) {
              Log.d(getClass().getName(), "Recognition failure");
              this.rl.onError(-1);
            }
            else {
              Bundle b = new Bundle();
              Log.d(getClass().getName(), "Final hypothesis: " + hyp.getHypstr());
              b.putString("hyp", hyp.getHypstr());
              this.rl.onResults(b);
            }
          }
          state = State.IDLE;
        }
        break;
      case SHUTDOWN:
        Log.d(getClass().getName(), "SHUTDOWN");
        if (this.audio != null) {
          this.audio.stop();
          assert this.audio_thread != null;
          try {
            this.audio_thread.join();
          }
        catch (InterruptedException e) {
        }
    }
    this.ps.endUtt();
    this.audio = null;
    this.audio_thread = null;
    state = State.IDLE;
    done = true;
    break;
}
if (state == State.LISTENING) {
    assert this.audio != null;
    try {
        short[] buf = this.audioq.take();
        Log.d(getClass().getName(), "Reading " + buf.length + " samples from queue");
        this.ps.processRaw(buf, buf.length, false, false);
        Hypothesis hyp = this.ps.getHyp();
        if (hyp != null) {
            String hypstr = hyp.getHypstr();
            if (hypstr != partial_hyp) {
                Log.d(getClass().getName(), "Hypothesis: " + hyp.getHypstr());
                if (this.rl != null && hyp != null) {
                    Bundle b = new Bundle();
                    b.putString("hyp", hyp.getHypstr());
                    this.rl.onPartialResults(b);
                }
            }
            partial_hyp = hypstr;
        }
    } catch (InterruptedException e) {
        Log.d(getClass().getName(), "Interrupted in audioq.take");
        }
      }
    }
  }
  public void start() {
    Log.d(getClass().getName(), "signalling START");
    synchronized (this.mailbox) {
      this.mailbox.notifyAll();
      Log.d(getClass().getName(), "signalled START");
      this.mailbox = Event.START;
    }
  }
  public void stop() {
    Log.d(getClass().getName(), "signalling STOP");
    synchronized (this.mailbox) {
      this.mailbox.notifyAll();
      Log.d(getClass().getName(), "signalled STOP");
      this.mailbox = Event.STOP;
    }
  }
  public void shutdown() {
    Log.d(getClass().getName(), "signalling SHUTDOWN");
    synchronized (this.mailbox) {
      this.mailbox.notifyAll();
      Log.d(getClass().getName(), "signalled SHUTDOWN");
      this.mailbox = Event.SHUTDOWN;
    }
  }
}
```