Speech Recognition Research for Recognize Digits in Sinhala

R.M.D.T. Asiri

139154A

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

April 2016

Declaration

I declare that this thesis is my 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 Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.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 the student: Asiri RMDT

Student Number: 139154A

Signature of the student: ....Asiri..... Date: 29/04/2016

Supervised By

Name of the supervisor: Mrs. Indika Karunarathne

Signature of the supervisor: ***UOM Verified Signature*** Date: 29/04/2016

Dedication

This Dissertation is dedicated to my loving Family for being part of me and encouraging me always being by my side.

Acknowledgement

First I express my heartfelt appreciation and gratitude to my supervisor Mrs. Indika Karunaratne for his most valued guidance, commitment and kind support to make this research success

Also I express my kind appreciation to our MSc Coordinator Mr. Saminda Premarathne to for his guidance and support and given flexibility with some timelines in order to complete the research activities

Also sincere appreciation is extended to Eng. Mr. Kanishka Jayasekara, Deputy General Manager (Information Technology) for his valuable opinions and encouragement given to me in this endeavor

It is my great pleasure to thank all the other Senior lecturers, Lecturers, Instructors, and staff members who helped us in many ways to make this research. The guidance and support received from all the members who contributed and who are contributing to this project, was vital for the success of the Research.

I would also like to thank my all the batch mates of the MSc in Information Technology batch 7 in faculty of Information Technology for their various help and support. And also other friends of the faculty as well as friends for outside who gave me support and encourage me with their best wishes.

Abstract

There is growing tendency in the human computer interaction activities especially within last decade with smart world. Day by day lot of smart options are coming to the market which has more powerful human computer interactive options. With this kind of an environment speech recognition is also became very vast and interested research area among modern computer researches. When we considering speech recognition there are lot of smart applications even in smart phones are available all around the world right now, but major problem of those application are the accuracy and the localization. Especially considering a language like Sinhala, current industry doesn't have much accurate of efficient recognition system to cater with. This research is basically focuses on how to identify user audio signals in Sinhala language and how to convert them in to text. Speech recognition becomes very popular research topic with the highly incremental human computer interactions today. Most of the popular or vastly using languages are already have well developed speech recognition systems, but as mentioned earlier languages like Sinhala it is very rare, but for Sri Lanka it is very useful to have a recognition system.

Ceylon Electricity Board (CEB) is the sole agent to generate and distribute electricity power within the Sri Lanka. So it has several call centers running around the country to get customer feedbacks especially regarding power failures. Recently CEB management came with an idea to employ some disable (blind) people to those call centers to collect customer complains thru telephone line and log them into a call center web application and then that application will process those complains to forward to correct maintenance party to attend to the problem quickly as possible. In order to get the input first idea was to use a brail keyboard; same time management has an idea to get the input thru a microphone. But that audio input needs to be in Sinhala language. So now our problem is to develop good speech recognition for Sinhala language to identify those vocal signals. As part of solution to the above mention problem, this research is conducting to create well trained Sinhala identification system. In this phase research is focus only to identify digit's vocal inputs created using Sinhala language. Than can be used to identify customer according to his/her electricity account number. Once that is done call center application can validate the customer using customer billing database.

Table of Contents

List of Figures

List of Tables

1. Introduction

Ceylon Electricity board is the main electricity generating company in Sri Lanka. With the market share of almost 95 – 99%, it controls the electricity generation, distribution and maintenance activities all around the county. To help those activities Ceylon Electricity Board has established several customer call centers all around the island to make customer complaints regarding any electricity issue. In those call centers there are several agents are working for collect the customer complaints especially thru phone calls. All call centers are currently running a web based software solution to capture those complaints and to redirect them in to correct maintenance parties.

As a government owned organization, management of the CEB is planning to offer some call center jobs for disable persons as a charity work and also to get disabled people's contribution to the government work force. So CEB management is researching to use those disabled people (especially blind) as call center agents. Their duty is to get the customer complaints through telephone line and log them into the call center software without using brail system. Call center application needs to identify the agents' voice signals and need them to convert into numbers and text as inputs. This need to be done for identifies digits and words in Sinhala language. This research is conducting as the first phase of above mention process of acquire disable work force to CEB. Research activities conducting to identify the digits accurate as possible in Sinhala as first phase

Java based speech recognition libraries and APIs will be using to read and recognize the user inputs. Users need to speak to a microphone and system will automatically identify the user by his/her vocal profile which is captured at the user registration process. Then user needs to input customers account number to the system by speak out one number at a time in the account number. At the end system identify the customer according to the audio input and load relevant customer details to the complaint. This process of recognizing will be discussed in details in the later chapters of this document.

2. Review of others work – Literature Review

2.1. Introduction

This chapter discusses the others work in the related research area. Here we take some example how the researches approached towards speech recognition and what are the pros and cons of those researches and how we can use them to generate good speech recognition model for Sinhala language.

2.2. Classification of Speech Recognition Systems

Most speech-recognition systems are classified as isolated or continuous. Isolated word recognition requires a brief pause between each spoken word, whereas continuous speech recognition does not. Speech-recognition systems can be further classified as speaker-dependent or speaker-independent. A speaker-dependent system only recognizes speech from one particular speaker's voice, whereas a speaker-independent system can recognize speech from anybody.[1]

This literature review will discuss some of various available speech recognition frameworks and researches in the field of speech recognition.

2.3.Speech Recognition in MATLAB

This section will focus on how to use the MATLAB built in facilities and related products to develop an algorithm for isolated speech recognition approach. Unfortunately this algorithm is speaker dependent according to the MATLAB which means it recognizes speech only from one particular speaker's voice. [1] This approach is having two major stages in order to identify vocals

1. Training stage - in this stage MATLAB developer is creating the dictionary for each word, acoustic model for each word, in a digit identification this dictionary will contain digits from zero to nine
2. Testing stage - here they use the created acoustic models algorithm recognizes the words or the digits

Now let's see how they capture vocals for testing

Seconds of speech from a microphone input at 8000 samples per seconds, which is the frequency of the microphone. The MATLAB code shown below is used to read the microphone input from Windows sound card

```
% Define system parameters
framesize = 80;          % Framesize (samples)
Fs = 8000;               % Sampling Frequency (Hz)
RUNNING = 1;             % A flag to continue data capture

% Setup data acquisition from sound card
ai = analoginput('winsound');
addchannel(ai, 1);

% Configure the analog input object.
set(ai, 'SampleRate', Fs);
set(ai, 'SamplesPerTrigger', framesize);
set(ai, 'TriggerRepeat',inf);
set(ai, 'TriggerType', 'immediate');

% Start acquisition
start(ai)

% Keep acquiring data while "RUNNING" ~= 0
while RUNNING
    % Acquire new input samples
    newdata = getdata(ai,ai.SamplesPerTrigger);

    % Do some processing on newdata
      ...
      <DO_SOMETHING>
      ...

    % Set RUNNING to zero if we are done
    if <WE_ARE_DONE>
        RUNNING = 0;
    end

end

% Stop acquisition
```

Figure 2-1 - MATLAB Code Sample

MATLAB uses Data Acquisition toolbox to set up continuous acquisition of the speech signal and simultaneously extract frames of data for processing.

This process is running until the RUNNING flag is set to ZERO (While loop), in this code we need to set the target action to <WE_ARE_DONE> after system reach that stage recording will be stopped.

Now next question, How they Analyzing this acquired speech?
They begin it by developing a word-detection algorithm that separates each word from ambient noise. Then derive an acoustic model that gives a robust representation of each word at the training stage. Finally, select an appropriate classification algorithm for the testing stage.

2.4. Speech Detection Algorithm

The speech-detection algorithm is developed by processing the prerecorded speech frame by frame within a simple loop to detect isolated digits, they use a combination of signal energy and zero-crossing counts for each speech frame zero crossing is a point where the sign of a mathematical function changes (e.g. from positive to negative)

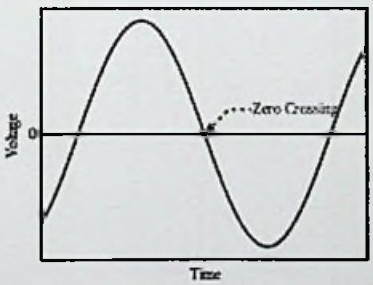

Figure 2-2- Zero Crossing

Signal Energy works well for detecting voiced signals and zero crossing of the signal works well to detect unvoiced signals

They have use inbuilt MATLAB functions to calculate above two metrics
To avoid ambient noise they assumed that each isolated image lasts at least 25 milliseconds
Next step is the Acoustic Model Development

2.5. Acoustic Model Development

Here first of all to investigate frequency characteristics they examined the power spectral density (PSD).Power spectral density basically describes the signal power distribution over the frequency

Since the human vocal tract can be modeled as an all-pole filter, they use the Yule-Walker parametric spectral estimation technique from Signal Processing Toolbox™ to calculate these PSDs.

Following figures demonstrate the calculated PSD for Number one and to two for three various voices

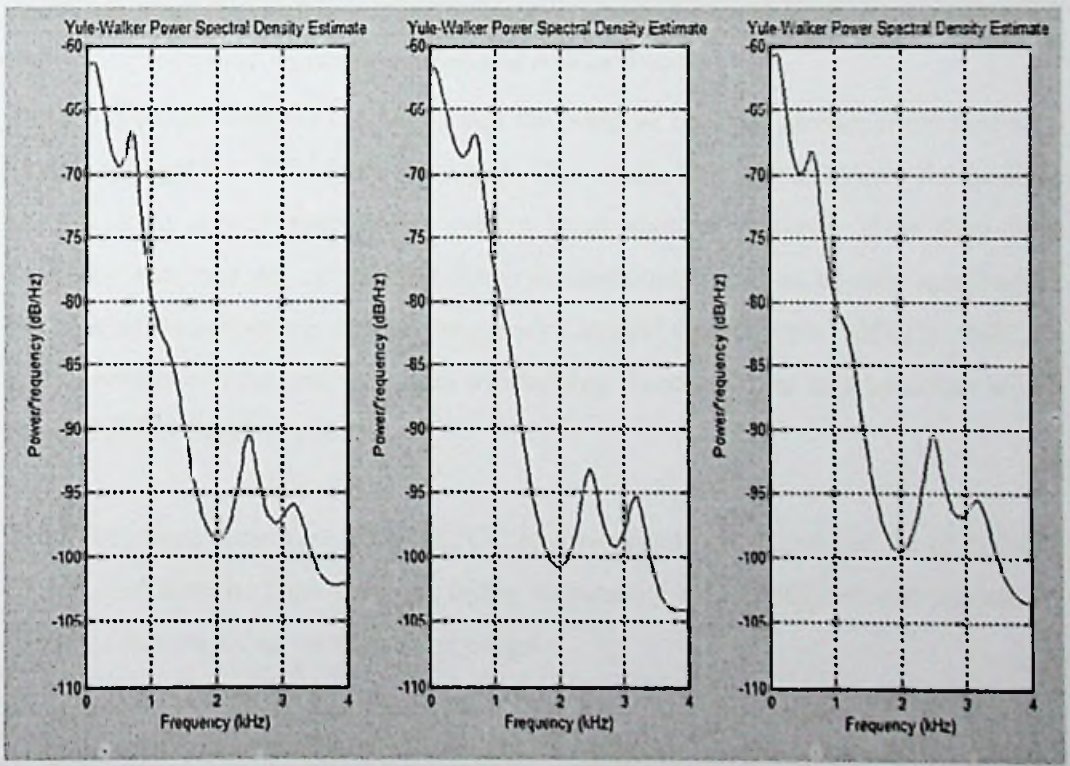

Figure 2-3- PSD estimate of three different utterances of the word "ONE."

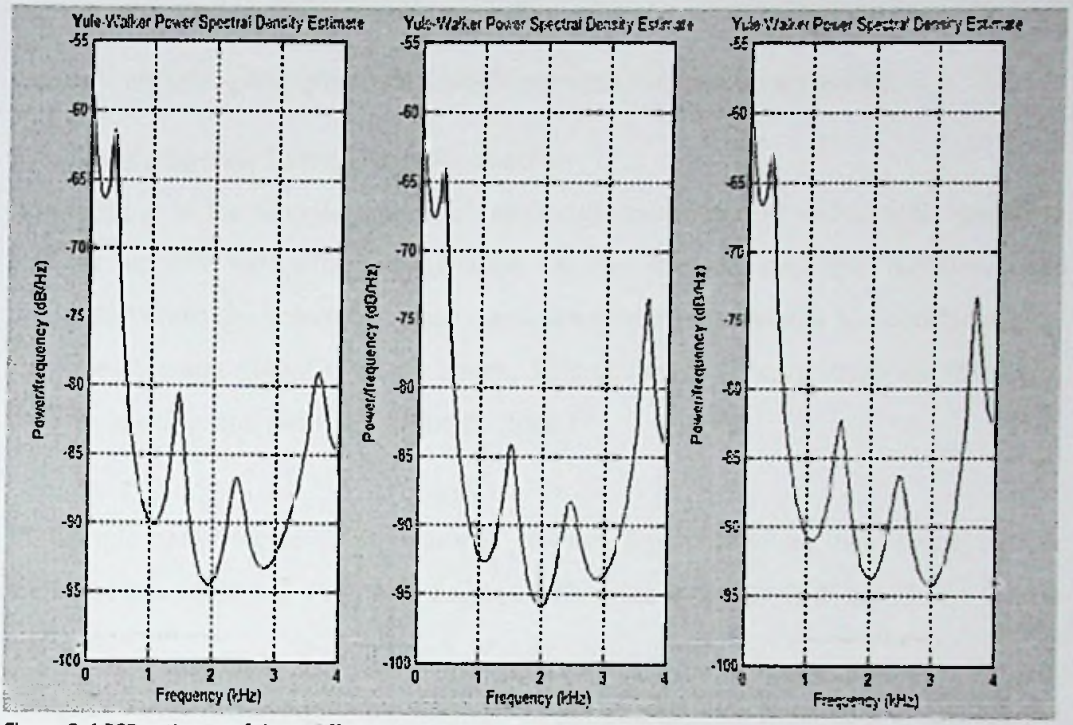

Figure 2-4 PSD estimate of three different utterances of the word "TWO."

Most important thing we can see is that the peaks in the PSD remain consistent for a particular digit but differ between digits. This means that we can derive the acoustic models in a speech recognition system from spectral features. Here they have specified that one set of spectral features commonly used in speech applications because of its robustness is Mel Frequency Cepstral Coefficients (MFCCs). MFCCs give a measure of the energy within overlapping frequency bins of a spectrum with a warped (Mel) frequency scale1.

What they have done here is that MFCC feature vectors are calculated for each frame of detected speech. Then they are trying to estimate a multidimensional probability density function for an each and every digit

They planned to extract the MFCC vectors from the test speech and use a probabilistic measure to determine the source digit with maximum likelihood, they use Gaussian mixture model (GMM) also to get the most likelihood log value from the MFCC parameters

This will conclude the summarizing the effort of MATLAB to have an Isolated Speech Recognition system

Next we are going to explore the Google research for speech recognition

2.6. Google Speech Recognition Researches

Today most of the Google apps such as Google maps are embedded with speech to text or speech recognition application in the internet and the mobiles.Those applications are the results of these continuous researches which are conducting by the Google using global research teams. This chapter will summarize the efforts of those researches and problems faced by them [2]

In Google speech processing research is basically focused on two areas, one is speaking to mobile and computer devices. Other one is to search or access the videos in the web.

Google also look at parallelism and cluster computing in a new light to change the way experiments are run, algorithms are developed and research is conducted. The field of speech recognition is data-hungry, and using more and more data to tackle a problem tends to help performance, Google kind of organization easily can generate larger data set because millions of users are trying to access and search by calling in to the Google software, so lot of samples are available with Google to go with speech processing research successfully comparing to others.

But according to Google research page as well as increasing accuracy, more data have other problems too which need to overcome.

Some of the problems identified by Google are as follows

1. How do you deal with data overload?
2. How do you leverage unsupervised and semi-supervised techniques at scale?
3. Which class of algorithms merely compensate for lack of data and which scale well with the task at hand?

As this review mentioned earlier researches of Google have the facility to get various vocal inputs all around the world using different mobile and computational interfaces which are running the Google apps. Also Google Researches are expanding over 25 languages due to this data availability and spread of researchers.

2.7. Automatic Pronunciation Verification for Speech Recognition

This research is conducted to verify the automatic pronunciation for speech recognition, research creates various lexicons and a lexicon understanding processes to identify the automatic pronunciations. Basically this is a data driven pronunciation learning research [3]

At the start system has a defined lexicon to verify the speech; with the data system will have kind of a learning process according to the new pronunciations

Learning set up is divided into three major recognition engines

1. Baseline : This is the baseline pronunciations for words to be verified
2. Append: This is the engine who adding new sounds to baseline
3. Replace: This engine will replace the old base with new pronunciations

Those three engines will work to fulfill the lexicon with the new incoming utterances. This is an ongoing training process to improve the baseline engine to identify more and more utterances

Then research focused on some mathematics specially log based metrics and recognition based metrics to gauge if the result difference is good, bad or neutral to update the baseline engine

2.8. Open Source Speech Recognition Toolkit –Source forge Project

This is a speech recognition toolkit which is developed as a source forge project; [4] this toolkit kit is having a larger number of supportive tools and functions to conduct speech recognition research.

Name is given to this framework is **CMUSphinx** toolkit, basically this has four major parts build in

1. Pocketsphinx — lightweight recognizer library written in C.
2. Sphinxbase — support library required by Pocketsphinx
3. Sphinx4 — adjustable, modifiable recognizer written in Java
4. Sphinxtrain — acoustic model training tools

Pocketsphinx is the lighter version to do a speech recognition when comparing to the Sphinx4, Pocketsphinx is speed and portable (even can work with android kind of

applications) but less flexible and manageable than the Sphnix4. But when it comes to accuracy it depends on many factors, not just the engine. The thing is that engine is just a part of the system which should include many more components. If we are talking about large vocabulary decoder, there must be idolization framework, adaptation framework and post processing framework. They all need to cooperate somehow. Flexibility of sphinx4 allows you to build such a system quickly. It's easy to embed sphinx4 into flash server like red5 to provide web-based recognition; it's easy to manage many sphinx4 instances doing large-scale decoding on a cluster.

On the other side, if your system needs to be efficient and reasonably accurate, if you are running on embedded device or you are interested in using recognizer with some exotic language like Erlang, pocketsphinx is your choice. It's very hard to integrate Java with other languages not supported by JVM pocketsphinx is way better here.

3. Technology Adapted: Sphinx 4

3.1. Introduction

This chapter will discuss the implementation technology of the research in detail manner, also discuss reasons to select the kind of technology and reasons to reject some of other available mechanisms

3.2. Why Sphinx 4?

One of the major reason to select this technology is the easy integration with the already implemented call center application. Since underlying technology is JAVA, it is very easy to integrate with other application and it is platform independent

Another reason to select such a technology is with the package we get the source code too, because this is a freely available open source bundle. If we need to adjust or have to do some changes, it is totally doable with this package

One reason to refuse MATLAB kind of an implementation which is discussed in the literature review is the integration difficulty and the high resource requirement of MATLAB package.

3.3. Overview

Sphinx stands for Site-oriented Processor for HTML INformation eXtraction is one of speech recognition engine that developed by The Sphinx works based on Hidden Markov Model (HMM) [6]algorithm. [7]

Sphinx4 is a pure Java speech recognition library. It provides a quick and easy API to convert the speech recordings into text with the help CMUSphinx acoustic models.

It can be used on servers and in desktop applications. Beside speech recognition Sphinx4 helps to identify speakers, adapt models, and align existing transcription to audio for time stamping and more.

There are several high-level recognition interfaces in Sphinx-4:

- LiveSpeechRecognizer
- StreamSpeechRecognizer
- SpeechAligner

For the most of the speech recognition jobs high-levels interfaces should be enough. And basically you will have only to setup four attributes:

- Acoustic model
- Dictionary
- Grammar/Language model.
- Source of speech

First three attributes are setup using Configuration object which is passed then to a recognizer. The way to point out to the speech source depends on a concrete recognizer and usually is passed as a method parameter.

One of the major reasons to select this technology is, it is written in java and published ad set of java libraries. CEB call centre application is running as a web application it is very easy to couple java application to that to identify the speech, in that case we don't need to provide several interfaces to user, we can use one interface get the audio signal and then we can presume with call center application.

Following diagram illustrate the Sphnix architecture in brief

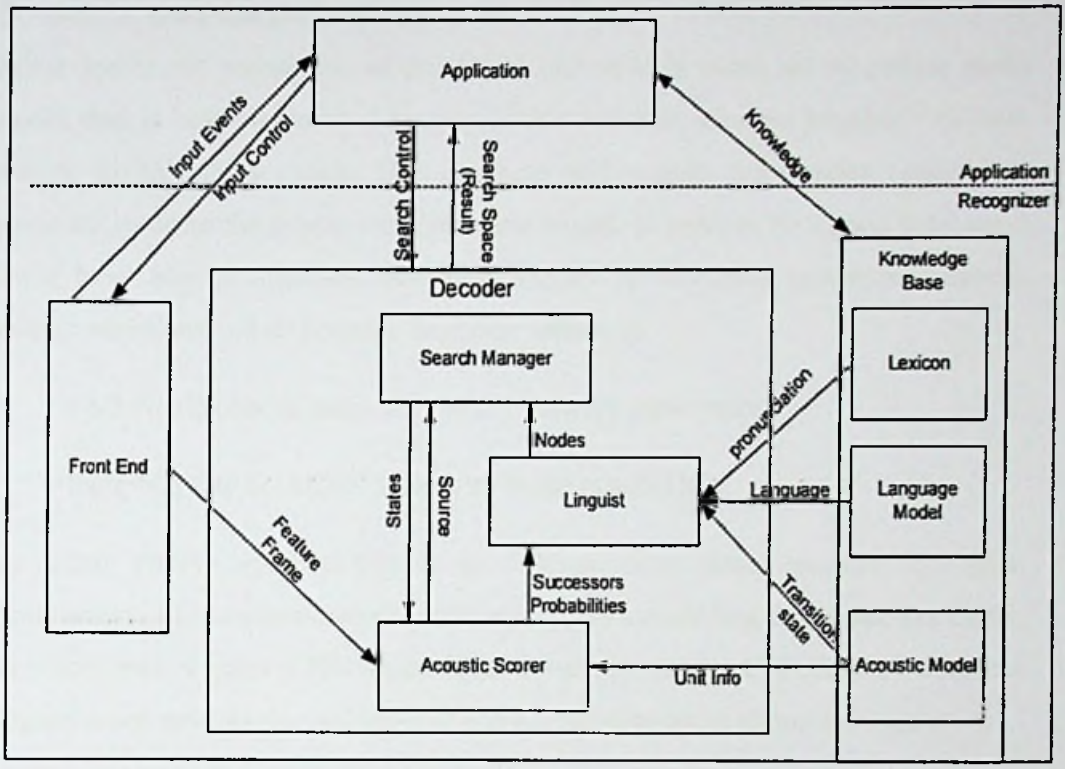

Figure 3-1 - Sphinx 4 - Architecture

Now let's look in to each of the above mentioned attributes briefly

3.4. Acoustic Model

An **acoustic model** is used in Automatic Speech Recognition to represent the relationship between an audio signal and the phonemes or other linguistic units that make up speech. The model is learned from a set of audio recordings and their corresponding transcripts. Models are Creates by taking audio recordings of speech, and their text transcriptions, and using software to create statistical representations of the sounds that make up each word. CMUSphinx libraries acts as the intermediate software to create statistical representations

CMUSphinx is coming with several high quality acoustic models. But here our problem is currently it does not have in built acoustic model for Sinhala language. So we need to create sample acoustic model to use Sinhala language within the CMUSphinx.

3.5. Data Preparation for Training

Trainer learns the parameters of the sound unit models using set of sample audio signals, that is called training database. In this research we need training data base contains Sinhala digit vocals. This database will contain information required to extract statics from the speech using acoustic model. In order to be a good database it should have enough speakers recording, variety of recording conditions, enough acoustic variations and all possible linguistic sentences.

CMUSphinx needs audio files with following parameters

Sampling rate is 16KHZ 16bit with mono (single) line

For small vocabulary CMUSphinx is different from other toolkits. It's often recommended to train word-based models for small vocabulary databases like digits. But it only makes sense if HMMs could have variable length. **CMUSphinx does not support word models.** Instead, need to use a word-dependent phone dictionary.

3.6. Language Model

Statistical language models describe more complex language. They contain probabilities of the words and word combinations. Those probabilities are estimated from a sample data and automatically have some flexibility.

Since we are considering digits for this research, we don't need more complex language model. But need to have accurate model in Sinhala language.

3.6.1. JSGF (Java Speech Grammar Format)

This the widely used grammar format to create language models in Sphinx 4. JSGF is a textual representation of grammars for use speech recognition [2]

3.6.2. Dictionary

This is responsible for determining how the words are pronounced for a given phase or for a given digit.

Following example demonstrate a dictionary created for English digits

Word	Pronunciation
ONE	HH W AH N
ONE(2)	W AH N
TWO	T UW
THREE	TH R IY
FOUR	F AO R
FIVE	F AY V
SIX	S IH K S
SEVEN	S EH V AH N
EIGHT	EY T
NINE	N AY N
ZERO	Z IH R OW
ZERO(2)	Z IY R OW
OH	OW

Table 3-1- Sample Dictionary

3.7. Summary

As conclusion to this chapter we can summarize following as the adapted technologies to implement the this research

Java based Spinix 4 libraries used as the main recognition framework, and to support that JSGF is used as the grammar formatter to the system

4. Research Approach

4.1. Introduction

This chapter will discuss the approach of this research step by step in later paragraph. This approach contains details regarding how the research conducted and what are steps followed and why they followed and obstacles faced during the research and how they overcome.

4.2. Approach in Steps

This research uses the Sphinx4 as the underlying technology to implement outcome of the research activity

First of all, need to create Language model and Dictionary in Sinhala to identify digits. By doing that need to identify all possible phonetics of the Sinhala digit speech from 0 to 9, then list down all the possible combination of phonetics to make words representing all the ten digits.

After creating language model with dictionary, system trained using several individual vocals in order increase the efficiency. After create the trained language model, it needs to be test the quality of the tested database in order to select best parameters, understand how application performs and optimize performance. With the trained and tested acoustic model, research implemented front end UI level application to capture the user inputs via microphone and dashboard to show the results

Once user sends the vocal input system needs to verify it and pass it to the recognizer to do the analysis process. Once recognizer completes the recognition it needs to send the identified digit back to the front level application. Above steps describes the basic approach to the research with Sphinx 4, furthermore to train the system we need to use several users and need to create as much as accurate language model.

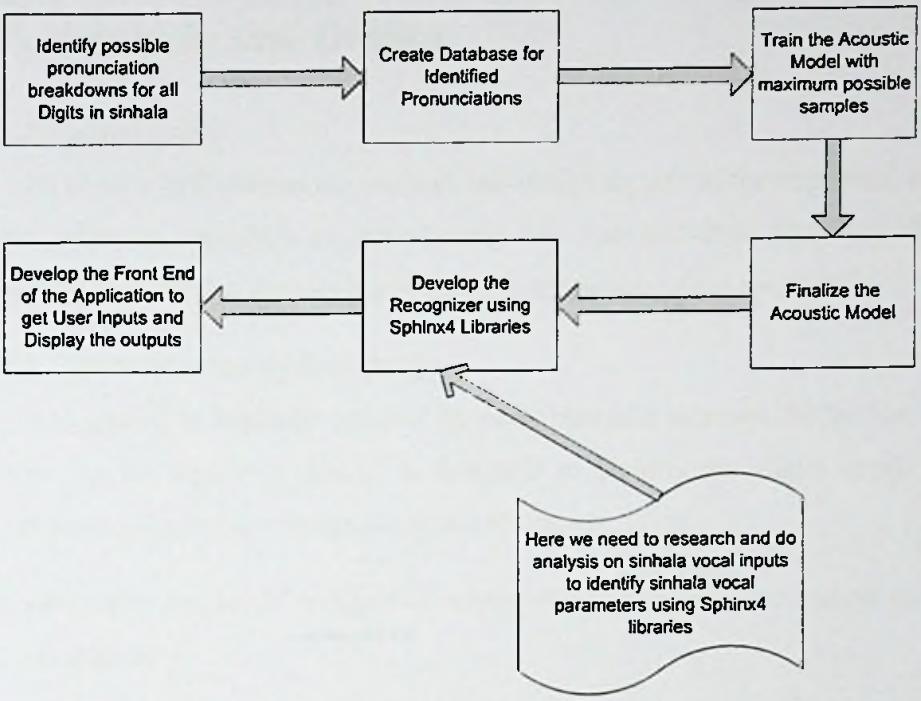

Figure 4-1- Research Approach in brief

4.3. Obstacles faced and how they overcome with this Approach?

Major obstacle is creating accurate acoustic model to compare with user input once the user has spoken in to the microphone. Lot of unnecessary noises are adding up to user inputs especially in a call center kind of environment. So we have to use well equipped and less noisily microphones to get agent voice to the system

Another obstacle is accuracy and the speed of the software package classes to identify a given voice using recognizer. To overcome this we have to increase some system memory variables and has to do some code level changes by removing unnecessary bindings happens at the runtime.

5. Analysis and Design

5.1. Introduction

This chapter will discuss the analysis and design aspects of the implementation part of this research. Design is explained using use cases and their decomposition. System architecture section discuss the overall design aspects of the research.

5.2. Abstract of the System Design

This research is basically focused for recognize and interpret the human voice in to text. As the first part system is designed to identify the digits speak in Sinhala language. Microphone is use to capture the human voice

System also has set of recognized words which are used as command such as start, exit, stop etc...

Basically system contains two major components, one use to capture the user data and process the acoustic signal. Other component is meant to interpret the processed signal into matching digits

5.3. Functional overview of the system

As mentioned in the abstract system contains two major components at first. When we are going to detail design system has three major functional behaviors under those two major components

First component takes the audio input and process it, extract the features from input signal to help the recognition. This is basically the front end of the system. Most of the user interactions are happening with this component

Second component is the most important part of this system design. It is the decoder, decoder use the output of the first component the match the output with the knowledge base and performs search to match the correct digit according to the vocal signal

Next is the knowledge base, it basically contains three parts

1. Dictionary - Matching the Digit with Pronunciation
2. Language Model – This contains the model vocals, this will help to refine the search

3. Acoustic Model – This is the acoustic database of statistical models

5.4. System Architecture and structure

This phase describes the system architecture and the interconnection between the modules within and outside. This recognition system comprises of several modules, so it is very important function all those interconnection properly in order to function the recognition as whole.

5.4.1. Architecture

System contains three major components as describes above

Front end captures the user inputs thru a microphone and analyze and break the vocal in to frames and then to features to pass into the decoder

Row data captured using a microphone sends to a data framer to frame the row data, afterwards frames will be extracts in to signal features. Then processes the extracted features and pass them to decoder

Following image describes front end behavior

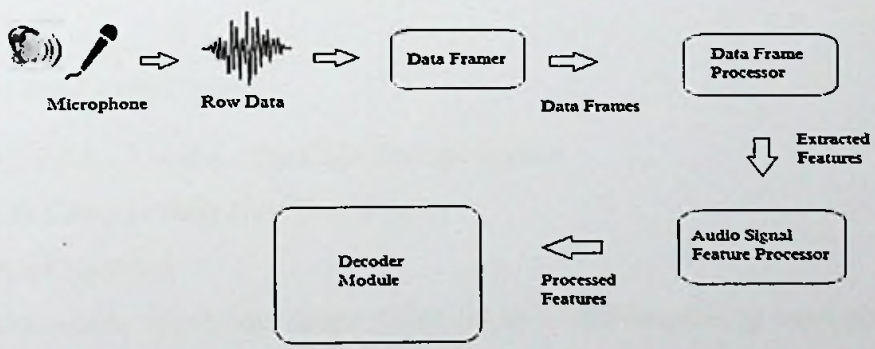

Figure 5-1- Frontend Behavior of the System

After the signal features send to the decoder, decoder uses the knowledge base to interpret the received features. Once decoder finishes the decoding output or the identified digit sends to the frontend for display.

5.5. Detail System Software Design

5.5.1. Use Cases

There are two major use cases to this recognition system. Since this is developing to a call center, always only authorized personnel can access to the system to retrieve customer data by giving their account number, so first major use case is to create authorized profile to call center agents. Next use case is the speech recognition use case

5.5.1.1.1. Use Case - Level 0

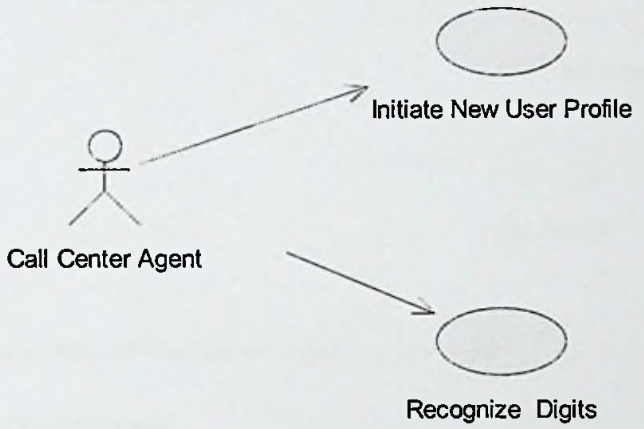

Figure 5-2- Level 0 Use case

5.5.1.1.2. Level 0 - Use Case Decomposition

Use Case: Initiate New User Profile

Pre-Condition

User needs to provide sample voice cut to a well-functioning microphone with low and consistent noise level

Description

This is the use case performs the user profile creation, once a person is allocated as a call center agent he/she needs to create their own profile in order to use the system. In order to create a user profile user needs to enter his/her sample voice cut to the system.

Then the profile handler analyses the input voice and obtain specific features for the given user and create new user profile by saving those data to a storage (to Database or to a Flat file).

Level 1: Use Case for "Initiate New User Profile"

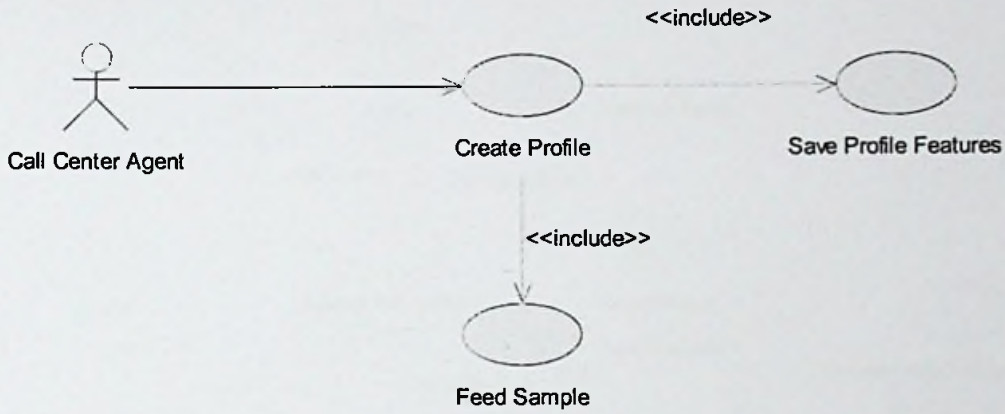

Figure 5-3- Level 1: Use Case for "Initiate New User Profile"

Post Conditions

New user profile needs to create against the user

Use Case: Recognize Digits

Pre-Conditions

User needs to have proper user profile and successfully spoke to the microphone

Description

This is the most important use case of the system, this will get the vocal from user and get the user profile from first use case and do the analytical and recognizing functions using system resources

This use case is responsible of following areas

1. Processing the input signal
2. Match the outcome of the signal processing with the knowledgebase
3. Retrieve the correct digit from knowledge base and return it to front end

4. Display the digit

Level 1: Use Case for "Recognize Digits"

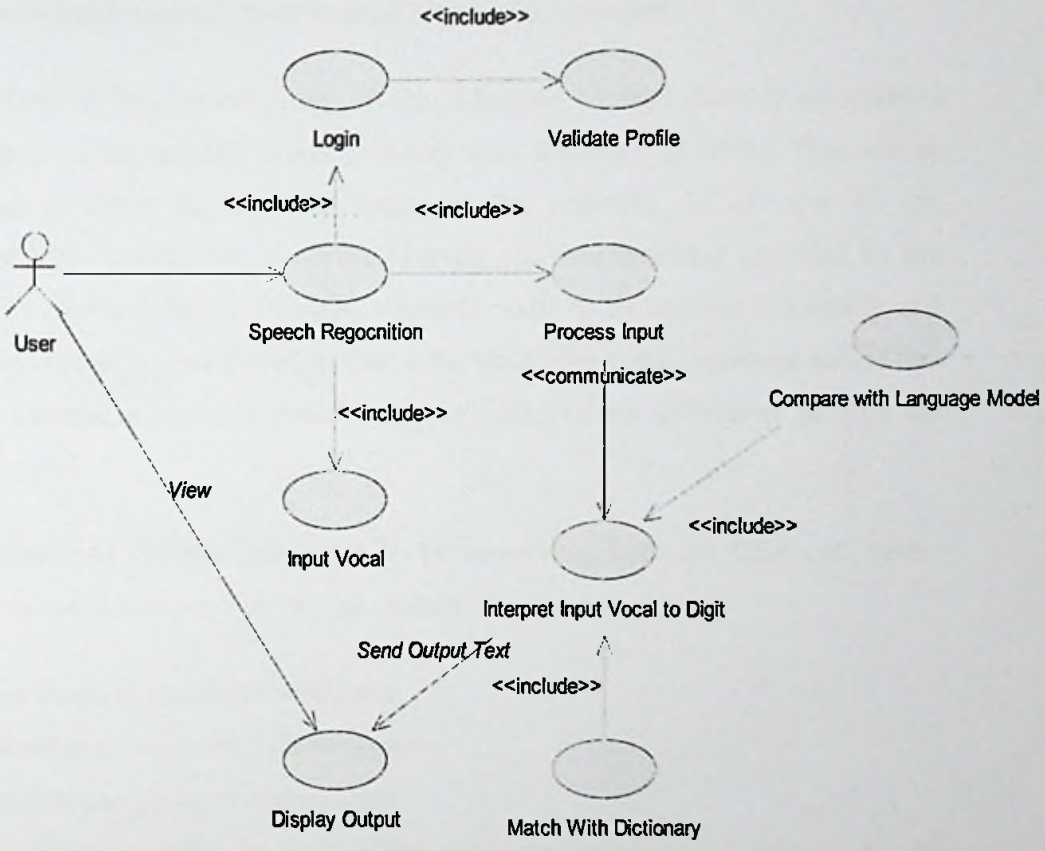

Figure 5-4 -Level 1: Use Case for "Recognize Digits"

Post Conditions

System needs to display correct digit according to the users' vocal input

Exception: If user input is invalid or login failed, system needs to provide appropriate error messages

6. Implementation

6.1. Introduction

When it comes to the implementation, first of all we need to create physical environment fulfilled with hardware requirements. Good quality microphone with less noise is very much essential to have good output in this research

Since Sphinx 4 libraries are using Hidden Makarov Method (HMM) so software development of the research is tightly bound with the basics of HMM. This will be discussing in detail in the later stages of the research. As of now all the implementation is doing based on the libraries and functionalities provided by the Sphinx 4 framework and additionally research needs to be conduct to identify and implement Sinhala language model. That is the most critical and important part of this research. Several user inputs need to capture with various differences to train the acoustic model.

Finally output of the research needs to be integrating with the CEB call center application in order to use with the call centers all around the country.

6.2. Major Parts of the Implementation

Implementation is consisting following parts

- Interface to get agent voice inputs
- Speech processing and Recognition using Acoustic Model, Language Model and the Dictionary created using sample and trained data
- Interface to call center application to pass the identified account number

Now let's briefly discuss how those components are implemented

6.3. Acoustic Model

Acoustic model is a database of statistical models. Each statistical model represents a single unit of speech such as a word or a phoneme.

6.4. Language model

Describe what is likely to be spoken in particular context. It will help to restrain search space.

6.5. Dictionary

The task of dictionary is mapping word to the pronunciations. Single word may have multiple pronunciations.

Those three parts are well explained under technology adopted chapter of this document.

6.6. Outline Architecture in Brief

This speech recognition speech is conducting to implement good speech to text conversion software to the Ceylon electricity board call centers. As we mentioned earlier in this document currently Ceylon electricity board call centers are running .Net application to track the user complaints especially regarding the power break downs from the customers. So this implementation need to couple with that .Net web application to identify the input voice signals.

In order to meet above implementation since we need to find a way to communicate Java based sphinx to .Net based call center application.

In that context this research suggest to implement the speech recognition as a java web service, then any kind of a third party application can call it by passing the required parameters. So dealing with microphone will be transfer to the client (here that functionality needs to implement in the call center application). Input voice signal need to pass to the web service and it will recognize the input and return to the client side. Advantage of this approach is whatever the client technology does not matter to use the speech recognition application. Even though call center application change from the .Net technology in the future it doesn't affect the speech recognition.

6.7. Logical View of the Implementation

Using above mention models and dictionaries Decoding component is implemented according to the following class diagram. This is basically developed with sphinx 4 library and we have add our own requirement to this structure

Following Figure will illustrate the class diagram of the implementation

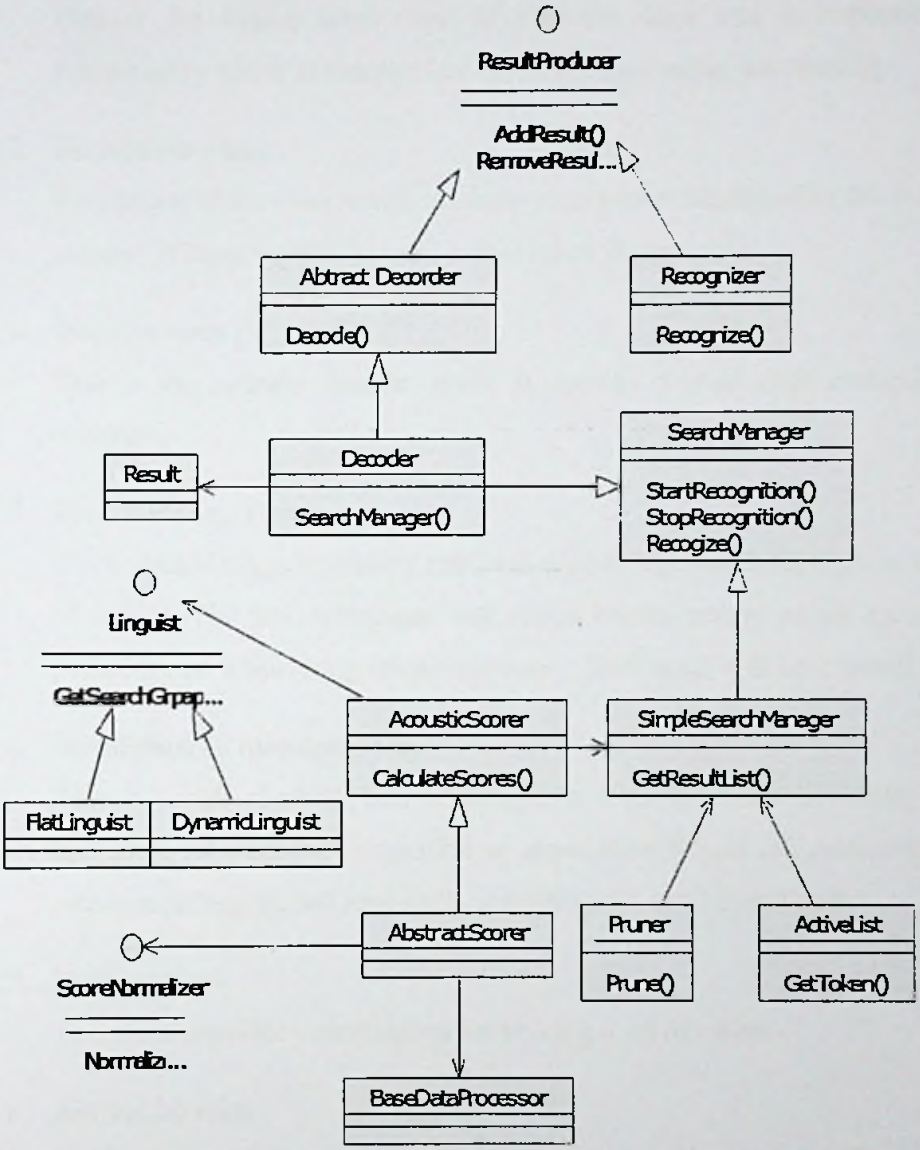

Figure 6-1- Main Class Diagram

6.8.Class Description

6.8.1. ResultProducer Interface

A higher level interface which is shared by components which are able to produce results.

6.8.2. AbstractDecoder class

This is the higher level class of Decoder class and it implements all functionality which is independent of the used decoding functionality.

6.8.3. Recognizer class

Recognizer is the class which contains recognition functionality for the given number of input frames, or until a final result is generated

6.8.4. Decoder class

This is the primary decoder class. It decodes frames until recognition is complete.

6.8.5. SearchManager class

The SearchManager's primary role is to execute the search for a given number of frames. The SearchManager will return interim results as the recognition proceeds and when recognition completes a final result will be returned.

6.8.6. SimpleSearch manager class

This is a sub class of SearchManager and provides the advanced search operation. To perform recognition an application should call initialize before recognition begins, and repeatedly call recognize until Result ends.

6.8.7. Pruner class

This class provides a mechanism for pruning a set of tokens.

6.8.8. ActiveList class

In this class an active list is maintained as a sorted list and gets the list of all tokens.

6.8.9. AcousticScorer class

AcousticScorer provides a mechanism for scoring a set of HMM states

6.8.10. Linguist class

The linguist is responsible for representing and managing the search space for the decoder. The role of the linguist is to provide, upon request, the search graph that is to be used by the decoder. The linguist is a generic interface that provides language model services.

6.8.11. AbstractScorer class

This class implements some basic scorer functionality but keeps specific scoring open for sub-classes.

6.9.Activity Diagram

These diagrams are drawn to show the main activity flow of the implementation

6.9.1. User Training Activity

When every time new user comes to the system, system capture that new user's voice input and add the acoustic values to the dictionary. In that case dictionary and training database increase with user base.

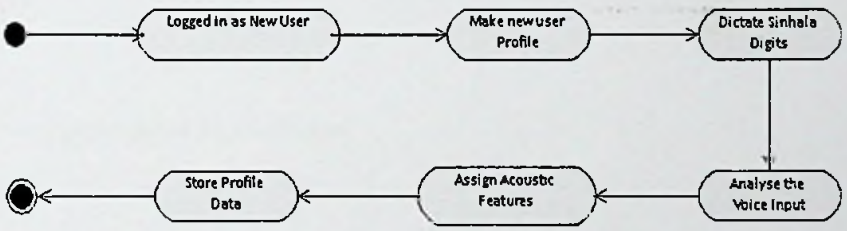

Figure 6-2 - User Training Activity Diagram

6.9.2. Recognizing speech Activity

This is the major activity of this implementation, once a user logged in and input his/her voice inputs to the system, by using the trained DB and decoder mechanisms system will generate results

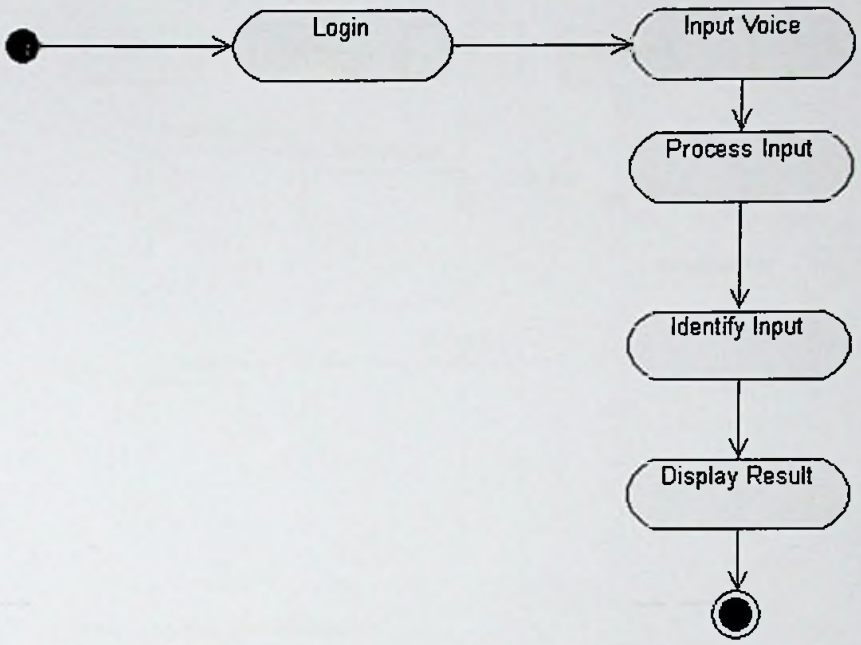

Figure 6-3 - Recognizing Speech Activity Diagram

6.10. Sequence Diagram for Decoding Speech

Following sequence diagram will explain the sequence of decoding process and class procedure calls

The input audio file processed in the front end and extract features needed for recognition. Then those *Feature Frames* forwarded to the *Scorer* where it get scored according to the data in knowledge base. With the knowledge base system can impose certain grammatical rules which are defined in language model. So *Scorer* allocates scores to the features against next likely states. Then *SearchManager* allocate linguist for the task and it return a *SearchData* which contain current state in the search space. Depending on those statistics *SearchManager* identify most possible answer.

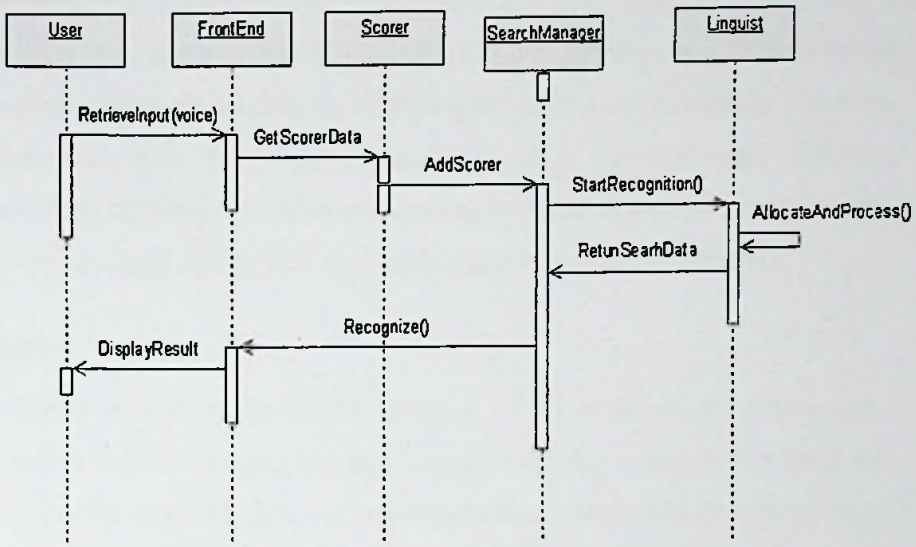

Figure 6-4 - Decoding Speech Sequence Diagram

6.11. Deployment Architecture

As we mentioned in the architecture outline section this recognition will be hosted as a web service, which can be reach by any application outside as client. Deployment environment can be within a cloud or within a privet hosting environment.

7. Evaluation

Evaluation is done by selecting various kind of CEB call employees with a different kind environments. Different employees used to evaluate the pronunciation effect on the application and how the implementation a reach to overcome different pronunciations. Also different environments used to evaluate how the environmental aspects especially artificial and natural noise effect to the recognition process

8. Discussion

This report's intend to discuss the interim progress of the research. As a summary, over all this report discuss the purpose and intend to do the research and how the progress up to current stage. Design and implementation chapters include high level facts regarding the Software and Identified Functional and Non Functional requirements to support the hypothesis.

Literature review contains some details regarding related works done by the different people and some technologies used to do them. Main difference in this research is Sinhala language model to identify Sinhala words. Although we have lot of implemented language models for other languages, for Sinhala it is very rare to find out good language model. By doing that research is planning to use Sphinx 4, which is a framework consist of several java libraries to support voice recognition.

Basically this research output implementation has major functionalities to cater with. They are as follows

1. Create and Train Acoustic model
2. Read and record user input from microphone and pass it to recognizer
3. Analyze the audio input and using the trained acoustic model to identify the digit and pass it to the front end
4. Get the recognizer identified result and display it in the application. Handle the input events such as start of speech, end of speech etc...

From above, Step 1 and step 3 are the most critical and researching parts of the overall activity.

9. Conclusion

In this research we have try to achieve to implement efficient digit recognition system in Sinhala language to support Ceylon electricity board call center application. Once the system identify the account number of the complaint call center application can retrieve his/her details from the database for further processing. Implementation of the research used JAVA based speech libraries to work with trained acoustic model for archiving the research targets. But still those acoustic models needs to do some tuning and training to make the recognition more accurate

10. Future Work

As the future work this research need to conduct to identify Sinhala words, then call center application can log the customer complain at the first place too. That is the main target of this research. So lot of training and recognition needs to do to create good dictionary for Sinhala language. Once it build we have to update and train it day by day to improve the efficiency.

Since Ceylon electricity board has its call centers all around the county and more age variation within its employees, with the time progress we can create good trained dictionary to Sinhala language with different pronunciations used around the country

11. References

[1] M. Daryl Ning, "Developing an Isolated Word Recognition System in MATLAB," 2010. [Online]. Available: http://in.mathworks.com/company/newsletters/articles/developing-an-isolated-word-recognition-system-in-matlab.html.

[2] Google, "Speech Processing," 2015. [Online]. Available: http://research.google.com/pubs/SpeechProcessing.html.

[3] Kanishka Rao, Fuchun Peng, Franc,oise Beaufays, "Automatic Pronounciation Verification For Speech Recognition," [Online]. Available: http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43262.pdf.

[4] sourceforge, "CMU Sphinx," 2015. [Online]. Available: http://cmusphinx.sourceforge.net/.

[5] G. Brandl, "sphinx-doc.org," 2015. [Online]. Available: http://sphinx-doc.org/sphinx.pdf.

[6] D. Paul, "Speech Recognition Using Hidden Makrov Models," 2010. [Online]. Available: https://www.ll.mit.edu/publications/journal/pdf/vol03_no1/3.1.3.speechrecognition.pdf.

[7] Bronto, "javasphinx User's Guide," 2012. [Online]. Available: https://bronto.github.io/javasphinx/.

[8] A. Hunt, "JSpeech Grammar Format," 2015. [Online]. Available: http://www.w3.org/TR/jsgf/.

12. Appendixes

Following code sniffs outline the major interface and functions of the implementation

```
package edu.cmu.sphinx.decoder;

import edu.cmu.sphinx.util.props.Configurable;

/**
 * Some API-elements shared by components which are able to produce
<code>Result</code>s.
 *
 * @see edu.cmu.sphinx.result.Result
 */
public interface ResultProducer extends Configurable {

    /** Registers a new listener for <code>Result</code>.
     * @param resultListener*/
    void addResultListener(ResultListener resultListener);


    /** Removes a listener from this <code>ResultProducer</code>-
instance.
     * @param resultListener*/
    void removeResultListener(ResultListener resultListener);
}
```

```
package edu.cmu.sphinx.recognizer;

import edu.cmu.sphinx.decoder.Decoder;

import edu.cmu.sphinx.decoder.ResultProducer;

import edu.cmu.sphinx.decoder.ResultListener;

import edu.cmu.sphinx.instrumentation.Monitor;

import edu.cmu.sphinx.instrumentation.Resetable;

import edu.cmu.sphinx.result.Result;

import edu.cmu.sphinx.util.props.*;


import java.util.ArrayList;

import java.util.Collections;

import java.util.List;


/**
```

```
 * The Sphinx-4 recognizer. This is the main entry point for Sphinx-
4. Typical usage of a recognizer is like so:
 * <p/>
 * <pre><code>
 *  public void recognizeDigits() {
 *      URL digitsConfig = new URL("file:./digits.xml");
 *      ConfigurationManager cm = new
ConfigurationManager(digitsConfig);
 *      Recognizer sphinxDigitsRecognizer
 *          = (Recognizer) cm.lookup("digitsRecognizer&quot");
 *      boolean done = false;
 *      Result result;
 * <p/>
 *      sphinxDigitsRecognizer.allocate();
 * <p/>
 *     // echo spoken digits, quit when 'nine' is spoken
 * <p/>
 *      while (!done) {
 *           result = sphinxDigitsRecognizer.recognize();
 *           System.out.println(&quot;Result: &quot; + result);
 *           done = result.toString().equals(&quot;nine&quot;);
 *      }
 * <p/>
 *      sphinxDigitsRecognizer.deallocate();
 *   }
 * </code></pre>
 * <p/>
 * Note that some Recognizer methods may throw an
IllegalStateException if the recognizer is not in the proper state
 */
public class Recognizer implements Configurable, ResultProducer {
```

```
    /** The property for the decoder to be used by this recognizer.
*/
    @S4Component(type = Decoder.class)
    public final static String PROP_DECODER = "decoder";

    /** The property for the set of monitors for this recognizer */
    @S4ComponentList(type = Monitor.class)
    public final static String PROP_MONITORS = "monitors";

    /** Defines the possible states of the recognizer. */
    public static enum State { DEALLOCATED, ALLOCATING, ALLOCATED,
READY, RECOGNIZING, DEALLOCATING, ERROR }

    private String name;
    private Decoder decoder;
    private State currentState = State.DEALLOCATED;

    private final List<StateListener> stateListeners =
Collections.synchronizedList(new ArrayList<StateListener>());
    private List<Monitor> monitors;

    public Recognizer(Decoder decoder, List<Monitor> monitors) {
        this.decoder = decoder;
        this.monitors = monitors;
        name = null;
    }

    public Recognizer() {
    }

    /* (non-Javadoc)
```

```
     * @see
edu.cmu.sphinx.util.props.Configurable#newProperties(edu.cmu.sphinx.u
til.props.PropertySheet)

     */

    @Override

    public void newProperties(PropertySheet ps) throws
PropertyException {

        decoder = (Decoder) ps.getComponent(PROP_DECODER);

        monitors = ps.getComponentList(PROP_MONITORS, Monitor.class);


        name = ps.getInstanceName();

    }




    /**

     * Performs recognition for the given number of input frames, or
until a 'final' result is generated. This method

     * should only be called when the recognizer is in the
<code>allocated</code> state.

     *

     * @param referenceText what was actually spoken

     * @return a recognition result

     * @throws IllegalStateException if the recognizer is not in the
<code>ALLOCATED</code> state

     */

    public Result recognize(String referenceText) throws
IllegalStateException {

        Result result = null;

        checkState(State.READY);

        try {

            setState(State.RECOGNIZING);

            result = decoder.decode(referenceText);

        } finally {
```

```
            setState(State.READY);
        }
        return result;
    }



    /**
     * Performs recognition for the given number of input frames, or
until a 'final' result is generated. This method
     * should only be called when the recognizer is in the
<code>allocated</code> state.
     *
     * @return a recognition result
     * @throws IllegalStateException if the recognizer is not in the
<code>ALLOCATED</code> state
     */
    public Result recognize() throws IllegalStateException {
        return recognize(null);
    }



    /**
     * Checks to ensure that the recognizer is in the given state.
     *
     * @param desiredState the state that the recognizer should be in
     * @throws IllegalStateException if the recognizer is not in the
desired state.
     */
    private void checkState(State desiredState) {
        if (currentState != desiredState) {
            throw new IllegalStateException("Expected state " +
desiredState
```

```
                    + " actual state " + currentState);
        }
    }

    /**
     * sets the current state
     *
     * @param newState the new state
     */
    private void setState(State newState) {
        currentState = newState;
        synchronized (stateListeners) {
            for (StateListener sl : stateListeners) {
                sl.statusChanged(currentState);
            }
        }
    }

    /**
     * Allocate the resources needed for the recognizer. Note this
method make take some time to complete. This method
     * should only be called when the recognizer is in the <code>
deallocated </code> state.
     *
     * @throws IllegalStateException if the recognizer is not in the
<code>DEALLOCATED</code> state
     */
    public void allocate() throws IllegalStateException {
        checkState(State.DEALLOCATED);
        setState(State.ALLOCATING);
```

```
        decoder.allocate();

        setState(State.ALLOCATED);

        setState(State.READY);

    }



    /**

     * Deallocates the recognizer. This method should only be called
if the recognizer is in the <code> allocated

     * </code> state.

     *

     * @throws IllegalStateException if the recognizer is not in the
<code>ALLOCATED</code> state

     */

    public void deallocate() throws IllegalStateException {

        checkState(State.READY);

        setState(State.DEALLOCATING);

        decoder.deallocate();

        setState(State.DEALLOCATED);

    }



    /**

     * Retrieves the recognizer state. This method can be called in
any state.

     *

     * @return the recognizer state

     */

    public State getState() {

        return currentState;

    }
```

```
    /** Resets the monitors monitoring this recognizer */
    public void resetMonitors() {
        for (Monitor listener : monitors) {
            if (listener instanceof Resetable)
              ((Resetable)listener).reset();
        }
    }


    /**
     * Adds a result listener to this recognizer. A result listener
is called whenever a new result is generated by the
     * recognizer. This method can be called in any state.
     *
     * @param resultListener the listener to add
     */
    @Override
    public void addResultListener(ResultListener resultListener) {
        decoder.addResultListener(resultListener);
    }


    /**
     * Adds a status listener to this recognizer. The status listener
is called whenever the status of the recognizer
     * changes. This method can be called in any state.
     *
     * @param stateListener the listener to add
     */
    public void addStateListener(StateListener stateListener) {
```

```
        stateListeners.add(stateListener);
    }


    /**
     * Removes a previously added result listener. This method can be
called in any state.
     *
     * @param resultListener the listener to remove
     */
    @Override
    public void removeResultListener(ResultListener resultListener) {
        decoder.removeResultListener(resultListener);
    }



    /**
     * Removes a previously added state listener. This method can be
called in any state.
     *
     * @param stateListener the state listener to remove
     */
    public void removeStateListener(StateListener stateListener) {
        stateListeners.remove(stateListener);
    }
    /* (non-Javadoc)
    * @see java.lang.Object#toString()
    */
    @Override
    public String toString() {
        return "Recognizer: " + name + " State: " + currentState;
    }}
```

```
package speechrecognition.view;

import java.awt.EventQueue;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;

import edu.cmu.sphinx.frontend.util.Microphone;
import edu.cmu.sphinx.recognizer.Recognizer;
import edu.cmu.sphinx.result.Result;
import edu.cmu.sphinx.util.props.ConfigurationManager;
import speech.SpeakString;
import speechrecognition.digits.english.EnglishRecognizer;
import speechrecognition.digits.english.MainRecognizer;

public class AccountDetailView {

	private JFrame frame;
	private static JTextField accNoTxt;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					AccountDetailView window = new
AccountDetailView();
```

```
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });

        try {

            SpeakString spk = new SpeakString();
            EnglishRecognizer engRec = new EnglishRecognizer();
            String str = "";
            accNoTxt.setText(str);

            ConfigurationManager cm;
            cm = new
ConfigurationManager(MainRecognizer.class.getResource("englishdigits.
config.xml"));

            Recognizer recognizer = (Recognizer)
cm.lookup("recognizer");
            recognizer.allocate();

            // start the microphone or exit if the program if
this is not possible
            Microphone microphone = (Microphone)
cm.lookup("microphone");
            if (!microphone.startRecording()) {
                //System.out.println("Cannot start
microphone.");
                spk.dospeak("Cannot start microphone.",
"kevin16");
                recognizer.deallocate();
```

```
                        System.exit(1);
                    }

                    spk.dospeak("Enter Account Number. Please speak out
number by number.", "kevin16");

                    Result result = recognizer.recognize();

                    String resultText =
result.getBestFinalResultNoFiller();

                    str =
str.concat(engRec.convertStringToDigit(resultText));


                    while (str.length() < 10 && str.length() > 0) {

                        Result resultInLoop = recognizer.recognize();

                        str =
str.concat(engRec.convertStringToDigit(resultInLoop.getBestFinalResul
tNoFiller()));

                        accNoTxt.setText(str);

                    }


                    spk.dospeak("Data is Loading Wait For While !!!!",
"kevin16");
            } catch (Exception e) {


            }
        }


        /**
         * Create the application.
         */
        public AccountDetailView() {
            initialize();
        }


        /**
```

```
     * Initialize the contents of the frame.
     */
    private void initialize() {
        frame = new JFrame();
        frame.setBounds(100, 100, 450, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(null);

        JLabel lblAccountNumber = new JLabel("Account Number");
        lblAccountNumber.setBounds(10, 31, 121, 14);
        frame.getContentPane().add(lblAccountNumber);

        accNoTxt = new JTextField();
        accNoTxt.setBounds(141, 28, 158, 20);
        frame.getContentPane().add(accNoTxt);
        accNoTxt.setColumns(10);

        JButton btnLoadDetails = new JButton("Load Details");
        btnLoadDetails.setBounds(309, 27, 115, 23);
        frame.getContentPane().add(btnLoadDetails);
    }
}
```