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

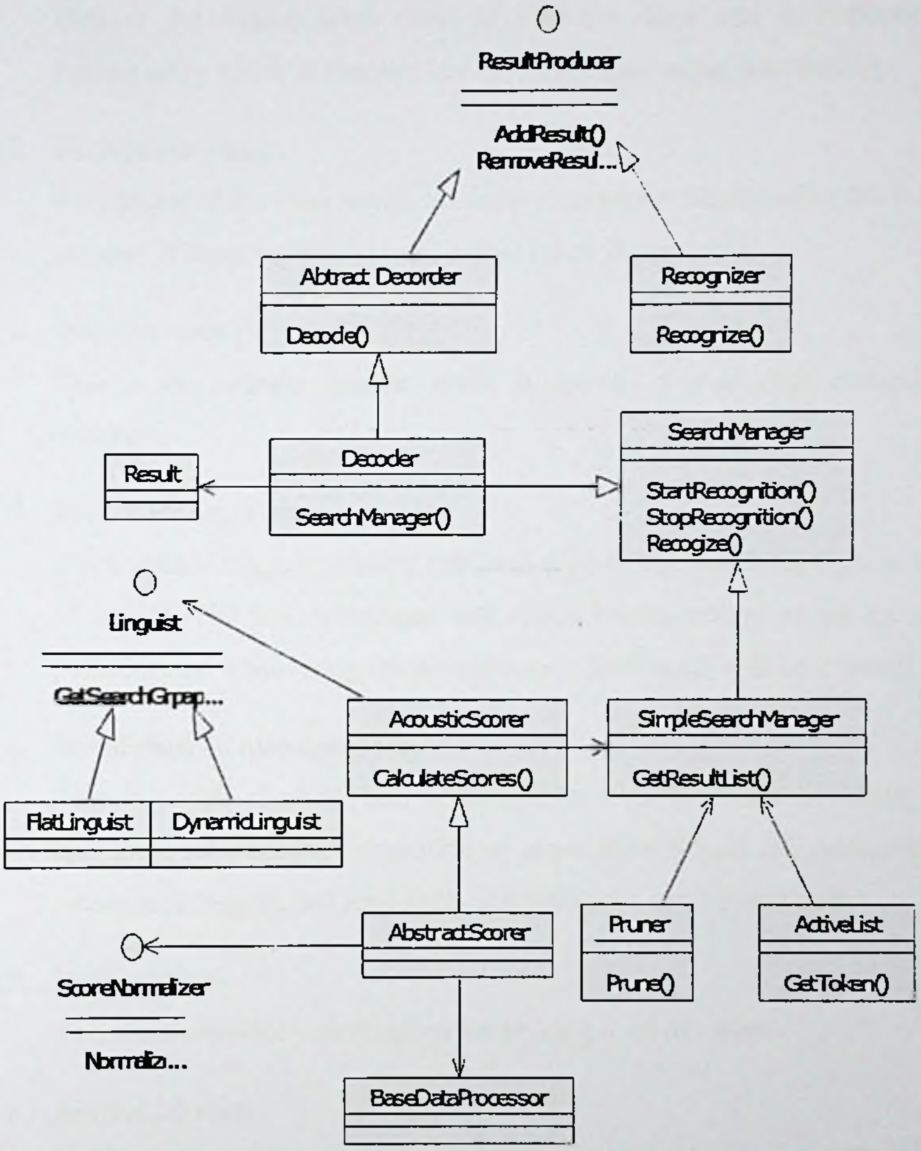

Figure 6-1- Main Class Diagram

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.

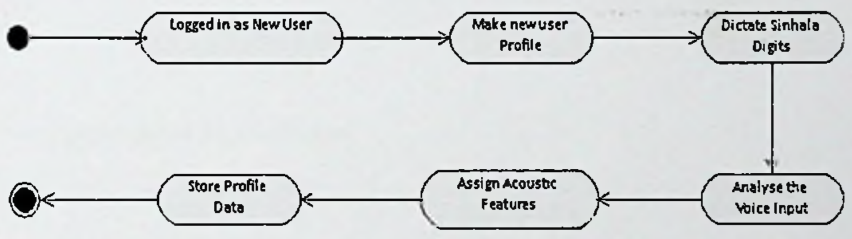

Figure 6-2 - User Training Activity Diagram

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

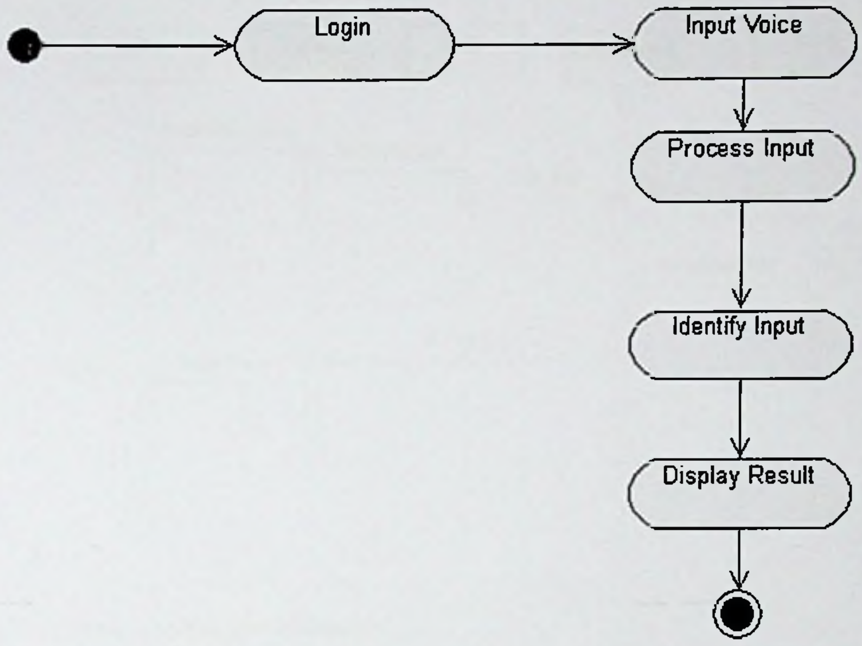

Figure 6-3 - Recognizing Speech Activity Diagram

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.

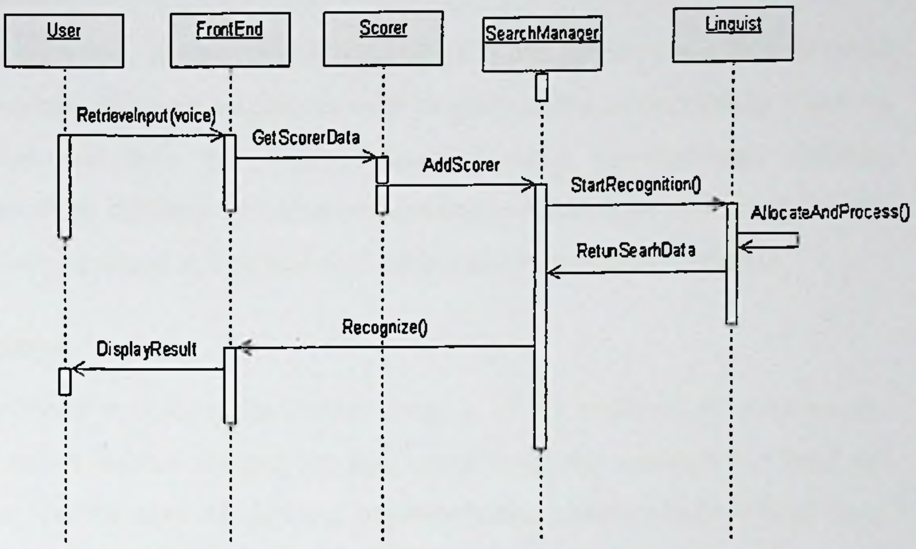

Figure 6-4 - Decoding Speech Sequence Diagram

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.