Dictionary Application with Speech Recognition and Speech Synthesis

[1]Sanket Kamble, [2]Ajinkya Kundur, [3]Akash Hulbutti, [4]Shubham Thorve, [5]Vijayendra Gaikwad

[1 2 3 4] Graduate Student, [5] Assistant Professor
Department of Computer Engineering,
Savitribai Phule Pune University, Pune, India

Abstract : *In today's environment while reading documents (online/offline) or browsing through many websites, we face many unfamiliar words, as we cannot get the meaning of those, we lack in complete understanding of the concepts or situation. So, to overcome this problem we are developing a Speech–To-Text and Text-To-Speech input and output method for our voice based dictionary system. The System is implemented using Java Sphinx-4 API for Speech Synthesis and Speech Recognition. The application or the system uses Java concepts like Swing API and JAVAFX for Graphical User Interface. The system will be deployed as the background process using Multithreading support.*

Index Terms - Speech Recognition, Searching and Mapping, Speech Synthesis, Sphinx, HMM, MARY TTS.

I. INTRODUCTION

The fastest growing technologies such as Speech Recognition and Speech Synthesis have gained a lot new netizens [1]. Speech based research is the widest growing research nowadays. One demerit/limitation of speech based recognition is the sharpness of the voice quality that is to be captured or recognized [2]. In Speech Recognizer module the input data which needs to be recognized has to be provided slowly and clearly [3]. The Dictionary consists of many words and meanings so, it is being stored as same as the operating system's file system database. The microphone quality also matters because of noise issues. Speech Recognition has many complex applications [4]. Also, since it is used as a replacement for typing. Speech Recognition also lacks in voice sharpness, due to which user has to speak slowly and clearly [3].

The system consists of these speech domains in a well-defined way. Domains can be easily interpreted. System recognizes voice and synthesizes for displaying and dictating the word and meaning. The Sphinx API internally consists of noise removal techniques which has impact on Performance metrics of the system [2]. System manipulates the input data/text, obtained by converting the speech-to-text, then it attempts to look for the word from database and it searches the particular meaning with the same method, when a specific match is found it is shown on the console window and dictated to the user.

To use this Speech Based Domains in system, two methods are used: The Hidden Markov Model(HMM), and a slight different and not used much called time warping method. Due to its high Accuracy and low computational need the HMM method is known to be DE-facto standard [3]. Sphinx uses HMM method.

II. MOTIVATION

As there are many voice assistant technologies been developed like Google Assistant, Apple's Siri which have the capability of Human Speech Recognition in a best possible way it can be, but all of this assistants continues to work with internet connectivity and also there are many online dictionary application available in the market which has a traditional method of inputting data to the system via keyboard. Hence in order to achieve human computer interaction with offline Human Speech Recognition we found the need of developing a dictionary application using Human Speech.

III. EXISTING SYSTEM

1) CMU Sphinx 4 :

CMU Sphinx 4 is an open source speech recognition tool. The Sphinx 4 API is developed in Java, which consists of different modules. Architecture of Sphinx 4 is shown in Figure 1. Modules in Sphinx 4 consist of Front End, Decoder, and Knowledge Base [2]. Front end of the Sphinx 4 is responsible taking input wave-forms and converts them into an appropriate form. Second module, Decoder is responsible for decoding/recognizing but before recognizing, knowledge base must be loaded. Knowledge base is nothing but the information required for speech decoder/recognition phase. Knowledge base can vary for different languages. Knowledge base consists of language model, Acoustic model and Dictionary contains information which will be used by decoder for recognizing purpose.

1) Front End: Front end has the responsibility of transforming the input waveform into the sequence of feature. Cepstral audio signals are extracted in this block/section of Sphinx 4. This signals then forwarded to decoder block.

2) Decoder: Decoder consists of 3 components as search manager, linguist, and acoustic scorer.

a. Search Manager:

Search manager constructs the tree for better hypothesis comparison and search [2]. The tree is generated using the information available, in obtained linguist. On top of that the search manager also communicates with the acoustic scorer for incoming data.

b. Linguist:

The linguist block is responsible for translating the linguist constraints provided through the grammar [2], Which is also used by search manager. Linguistic constraints are generally provided as context free grammar (CFG) through grammar file (. gram file).

c. Acoustic Scorer:

The acoustic scorer generates state output probability for various states. Acoustic scorer provides these probabilistic scores on demand of search manager [2]. For computing these scores, the scorer must share the probabilities with the front end.

3) Knowledge Base: The knowledge base provides whatever resources needed by the decoder in order to carry out its functionality [2]. Knowledge base provide/consists of language model (. lm file), Dictionary (. dic file), Acoustic model (included in Sphinx 4 API jar file).

Dictionary consists of words with their phoneme sequence. Language model consists of grammar required for understanding of sequences [2]. The acoustic model consists of acoustic data for every phoneme in dictionary.

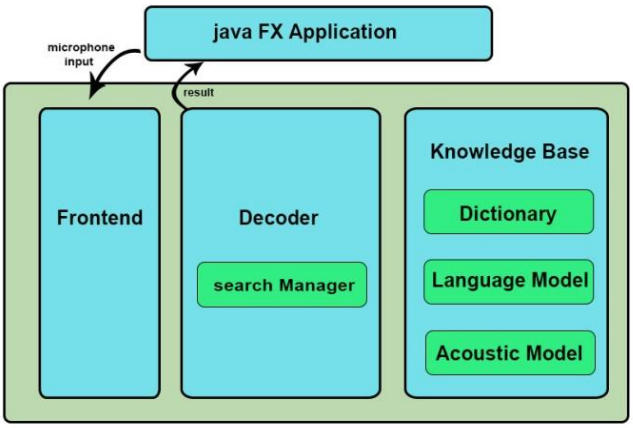

Figure 1 – Sphinx 4 Architecture

2) Hidden Markov Model:

A HMM is a statistical model for sequences of discrete symbols [5]. From many years HMM is being used, for speech recognition and for perfect gene finding task. HMM provides effective framework which is used for modelling time-varying spectral vector sequences. HMM is applied efficiently on well-known biological problems, such as [5],

1. Protein Secondary Structure Recognition.
2. Multiple Sequence Alignment.
3. Gene Finding.

HMM is a heart of all modern Speech Recognition Systems [5]. Well known applications of HMM are in Reinforcement Learning, Temporal Pattern Recognition such as Speech-Gesture Recognition.

3) MARY TTS:

MARY TTS is an open source voice creation toolkit of unit selection [6]. It generates the voice based on HMM [6]. MARY stands for Modular Architecture for Research on Speech Synthesis. The MARY TTS is successfully implemented in British English, Turkish, Telugu and Mandarin Chinese languages. The voice can be easily generated for the supported languages. MARY TTS also gives support for new language support/implementation [6].

The MARY TTS is developed in Java. It consists of following modules.

A. The Pre-processing:

The pre-processing module consist tokenizer, abbreviation expansion and numeral expansion [7].

B. Natural Language Processing:

To calculate the speech-relevant data the input text natural language processing is done. Initially in NLP part of speech labelling is done and shallow parsing (chunking) [7].

IV. PROPOSED SYSTEM

The current applications, which uses such functions does not uses the off-line speech recognition and off-line database for storage purpose. The proposed system is overcoming this drawback of existing desktop applications. On top of that the proposed system application will be running in background, in the form of process. Which makes it completely hidden from the user but, will be functioning in background.

The Application will be totally functioning on user's voice/speech which improves the interaction of user with system. The application initially will be running in background. To invoke the application a specific command is predefined by developers, the application will be continuously listening for the invocation command. Whenever the application detects the specific command application will start listening again but now it will be listening for the word from the user voice, for which the meaning needs to be searched. This phase of application will be running for a specific period of time, if it couldn't locate any word then it will again go in listening mode for invocation command.

After listening to the word, the interpreted word will be searched in the off-line dictionary database, stored in the system. If the word is not present in the dictionary then application will show appropriate error messages through pop-ups. If word is found in dictionary then it will collect all the variations of the word and returns an appropriate meaning and also shows the variations.

The word and meaning will be shown in a pop-up window at a bottom-right corner of the screen; also it will dictate the meaning to the user.

Following figure 2 shows the architecture of application with speech recognition and speech synthesis.

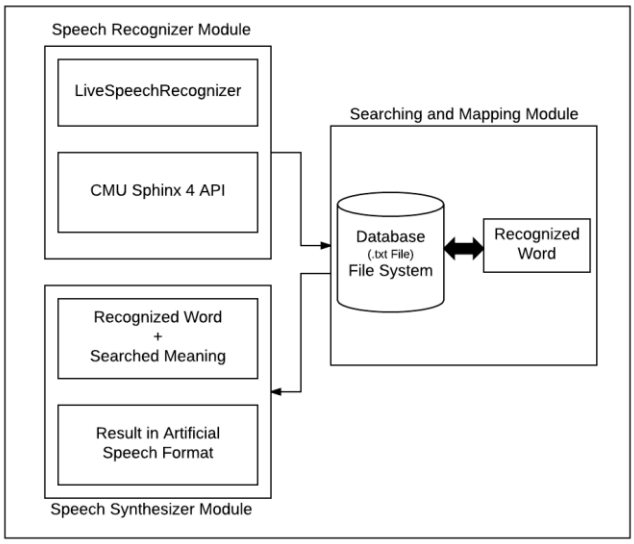

Figure 2 – System Architecture

V. WORKING

The System works in basic 3 phases, Speech Recognition, Searching & Mapping and Speech Synthesis. The detailed working of the entire system is described as below:

1) Initialization:

The System initiates its resources like every other system, starting with the followings:

1. Compatibility Check: The compatibility check will check whether the system is able to do the required operation successfully i.e. resources like microphone speaker or headphones are verified for attachment to achieve the successful execution.
2. Retrieve Configuration: As the application uses many voice modes, the configuration related to those context/functionality needs to be maintained or saved in order to access it every time. To achieve that we are using a text file (.txt) to store the configuration related to those voice modes. Every time user changes the voice mode the details will get updated to that text file.

2) Execution:

After initialization of all resources the application will actually starts executing in following order:

1 . System Tray : The application starts and keeps working in background, which can be seen by a tray icon in system tray, implemented using JAVA AWT TrayIcon Class. The icon will display following options on right clicking on it:

-Take Word: It will start the recognition.
-Open: It will open the GUI based dictionary.
-Exit: It will terminate the application.

2. Speech Recognition Based Execution: When the user clicks on 'Take Word' it will start the speech recognizer then it will start listening to the word on initial launch (directly start listening the word). If it gets the word then the searching will begin that is explained further. If it is couldn't be able to get the word in first attempt, then it will go in sleep state. To wake up the recognizer user will have to invoke it manually using "OPEN DICTIONARY" command, and then it will start recognizer. This procedure of sleeping and waking up the recognizer will happen for each attempt of word detection.

For Speech Recognition purpose Sphinx4 API is used with acoustic model and language model. These files can be modified according to words needs to be recognized.

3 . Search Meaning: For storing words and its meaning we have used serialization feature of java for storing object containing words and meaning. The object is de-serialized when system initiates as explained earlier. The data structure in object used for storing data is "HashMap<String,String>". It is easy to use, as searching and adding elements is dynamically possible.

4 . Front End: The application also provides manual interface for using dictionary with JAVAFX for attractive user interface. In System Tray it can be accessed by selecting "options" option in popup menu. It is a Java Library used to build Rich Internet Application. The applications developed using this library can be executed on multiple platforms. It supports FXML based GUI designing using different editing tools. It gives user different set of API's and also provides CSS like styling feature.

5 . Speech Synthesis : System provides the meaning of word in voce format. User can select the voice mode of its choice by going in the options in popup menu. System is using the MaryTTS open source API available for speech synthesis purpose. It has the functions in order to access the configurations and creating the artificial voice.

Algorithm:

Step 1: Start

Step 2: Choose an option from applications system tray by right clicking on it,
A . 'Taka a Word'.
B . 'Open'.
C . 'Exit'.

Step 3: If 'a' then - start recognition().
- A notification window will be shown indicating that system has started by loading all the resources.
- User will have to speak a word, else it will go to sleep mode.
- The application will ignore all words, till it listens to wake up command 'OPEN DICTIONARY'.

Step 4: If b: open application.
- It will show various features to search a word manually, change voice mode, information guide.

Step 5 : The entire application will be terminated from further execution.

Step 6: Stop.

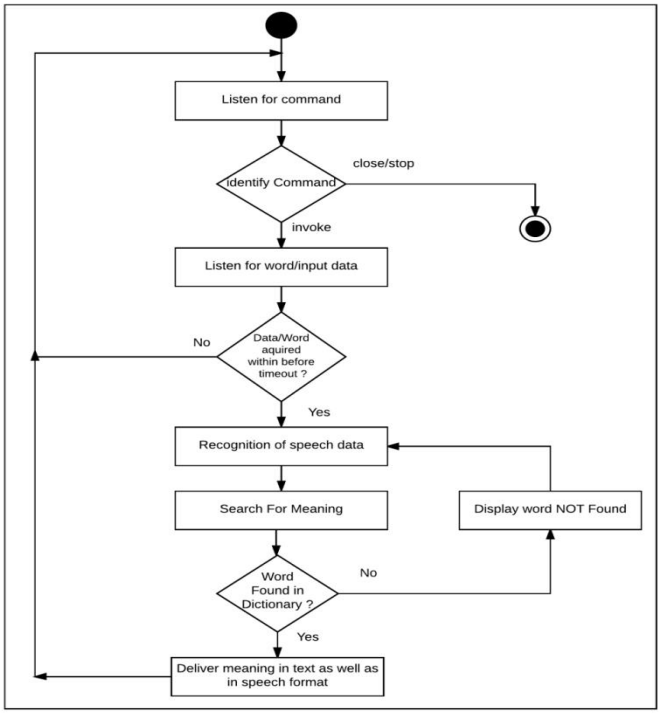

Figure 3 – System Activity Diagram

VI. RESULT

After complete implementation of application, we conducted various tests in order to analyze the system for its accuracy and its performance. The application uses *.dic(pronunciation dictionary file)* and *.lm(language model file)* as dictionary resources. The word and its meaning are stored in *HashMap* collections object, where it contains key-value format of word and its meaning. It is

used once the voice or speech is successfully interpreted and detected in both *.dic* and *.lm* files. The accuracy of word detection depends on the noise immunity, However Sphinx-4 API contains some noise removal part, but it doesn't improve the overall performance without training. Also it depends on the accent of user, which varies from individual to individual.

For example the pronunciation of words 'TASTE' and 'TEST' sounds same, but the system understands it completely different as shown below. Also the word 'Text' so far is similar to other words. The meaning of all these three words is different.

TEST	T EH S T
TASTE	T EY S T
TEXT	T EH K S T

The application contains huge number of words, the pronunciation of these words differs, but when system at run time implements phoneme of users' voice it may include noise as well. So the final word and the actual word may be different. Such problem occurs when *.dic* file contains larger number of words. So the accuracy can be reduced if the system is integrated with some training or some learning capability. However this application doesn't involve training the accuracy is less. If the count of words is less, then accuracy could be more than moderate.

Following are the screenshots of the application, Fig 4- Shows successful execution of manual search, Fig 5- Shows the execution of search if word and meaning not found, Fig 6- Shows System Tray implementation with popup menu.

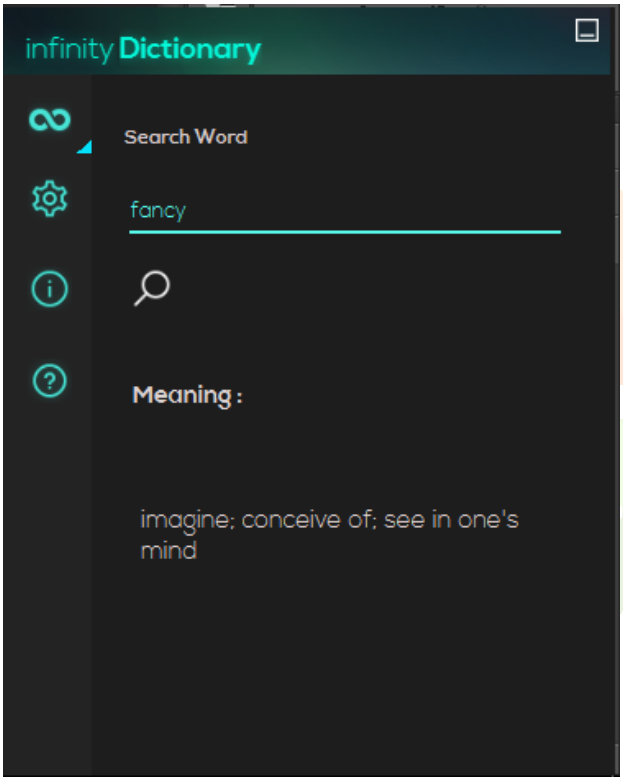

Figure 4 – Found Meaning

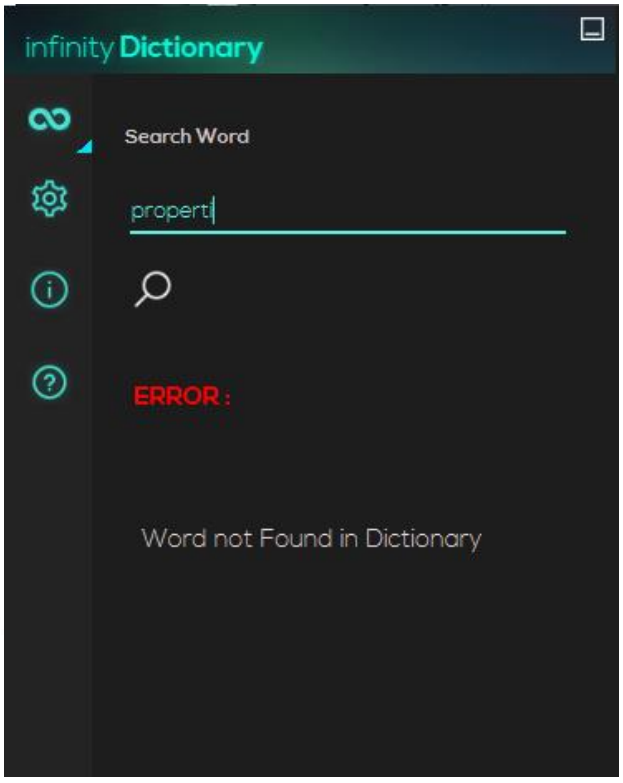

Figure 5 – Meaning Not Found

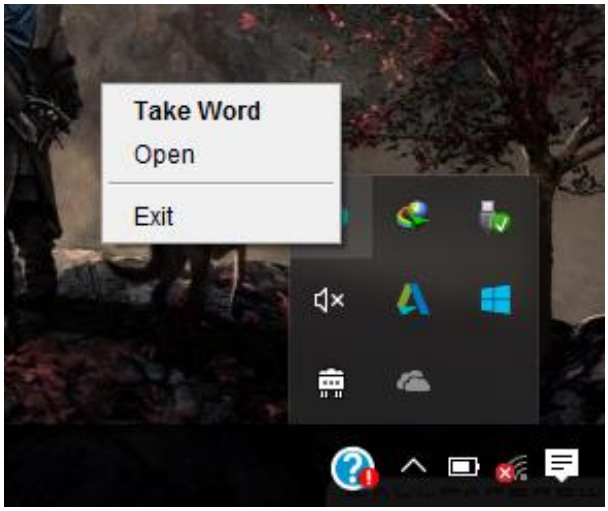

Figure 6 – System Tray Implementation

The execution of system in voice mode cannot be shown here, as it is speech based application.

VII. CONCLUSION

The recent advancements in software technologies have made human effortless. In such an environment when communicating and reading, the words play an important role. The complete understanding of each word is necessary, as it is an important physical interaction with people. Our application is a feature through which it can be made easy for individuals in their daily life. The application/software that has been developed is a dictionary application which works on Human speech and it is offline. The CMU Sphinx 4 API is used for speech recognition purpose. The application delivers the meaning of the word by searching and mapping the meaning from English dictionary (Database). The word and meaning are synthesized and converted to audio format using open source voice generation toolkit MARY TTS.

The main entities of the application are accuracy and number of words. Also the users' environment plays a vital role. The noise can reduce the efficiency in detecting the word and as the number of word increases the accuracy of the system decrease. As system doesn't have any learning or adapting capability it affects the overall system performance. We have found that by using high quality headsets increases the system performance.

REFERENCES

[1] J. Twiefel, T. Baumann, S. Heinrich and S. Wermter, "Improving Domain-Independent Cloud-Based Speech Recognition With Domain-Independent Phonetic Post-processing", Proceedings of the Twenty-Eight AAAI Conference on Artificial Intelligence, 2014.

[2] P. Lamere,P.Kwok,W. Walker,E. Gouva,R. Singh,B .Raj,P. Wolf and J. Woelfel. " Sphinx-4: A flexible open source framework for speech recognition" ,Sun Micro-system Inc. 2004.

[3] S. Batlouni, H. Karaki, F. Zaraket, F. Karameh, "Mathifier – Speech Recognition of Math Equation",18th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2011, Beirut, Lebanon, December 11-14, p. 301, 2011.

[4] Information retrieved on November 10th, 2017, from http://www.guidogybels.eu/asrp4.html

[5] J. Watada,IEEE ,Hanayuki. "Speech Recognition in a Multi-Speaker Environment by using Hidden Markov Model and Mel-frequency Approach", Third international conference on computing measurement control and sensor network.

[6] S. Pammi,M. Charfuelen,M. Schröder, "Multilingual Voice Creation Toolkit for MARY TTS Platform".

[7] Documentation retrieved on October 20th, 2017 from, mary.dfki.de/documentation/index.html.