International Journal of Engineering & Technology, 7 (3.31) (2017) 93-97

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Analyzing Call Data Through Live Calls Using Sphinx Tool

K. V. Krishnam Raju*, V. N. S. Manaswini

Computer Science Engineering, SRKR Engineering College, Bhimavaram, Andhra Pradesh, India-534204
**Corresponding Author E-mail:kvkraju.srkr@gmail.com; Tel;+91-9948771117*

Abstract

For Improving the Business growth, the Business people try to know the customer's intension about their products. One of the best methods of collecting customer's feedback is telephone or mobile survey where customer service representatives(CSR) can interact with customers through phone calls and also record to analyze the customer's call data. The main issue of call data analysis through recorded files is a large amount of storage is required to store the audio files. This results increased costs, maintaining the hardware and software systems and manage a database system. In this paper we can directly convert the live calls into text files using speech to text (STT) algorithm and analyze these text files using Hadoop and MapReduce Framework for improving their future purchasing behavior.

Keywords: *Analysis; Big data; Live Calls; STT; Sphinx.*

1. Introduction

Speech [1] is the best way to communicate with others. It is nothing but expressing our feelings in vocalized form with others. Speech is a combination of words, sentences and phrases. Each word is formed with combination of consonants and vowels this is known as phonetic combination. Phonetic is defined as the study of the speech sounds which are generated by the human with their different organs like mouth, throat, nasal, cavities, sinus and lungs. There are some important methods to follow for converting speech to text that is Speech recognition, speech analysis, speech authentication. For getting better accurate data we have to follow these methods.

Speech Recognition [2] is also known as "automatic speech recognition" (ASR), "speech to text" (STT) or "computer speech recognition". It is used to identify a limited number of words and phrases with the help of machine or program or web applications in particular spoken language, converting speech to human readable format **Figure 1**. It works on acoustic and language modeling where Acoustic modeling [3] is used to represent the relationship between audio signals and phonemes (one unit of sound). Language modeling provides probability distribution over word sequences for accuracy of data and It is used to differentiate the words which is having similar sounds.

Fig. 1: Converting speech to human readable format.

Speech recognition is mainly divided into two types called as speaker dependent.

Speaker dependent systems are mainly useful to recognize an particular user's voice those who are trained the system. This is more accurate than speaker independent because Speaker dependent system is static in nature where this system consists of a limited number of words and phrases. So, it is very easy to recognize the words. Speaker dependent systems are commonly observed in dictation software.

Speaker Independent systems are mainly useful to recognize everyone's voice, so there is no need to "train" the system. This system is very difficult to develop because every person has its own voice and the word pronunciations are also different from others, high expensive, less accuracy than speaker dependent systems, more flexible than speaker dependent system, dynamic in nature. Speaker dependent systems are commonly observed in both telephone services and interactive voice response systems.

Using Speech analysis [4], we remove unwanted properties from speech signals. Speech signal is denoted by S(n). This method is used to remove unnecessary repetitions of data in the speech signals S(n) and to simplify the speech signals in **Figure 2**. It provides accuracy to speech signal and It simplifies the explanation of phonetics.

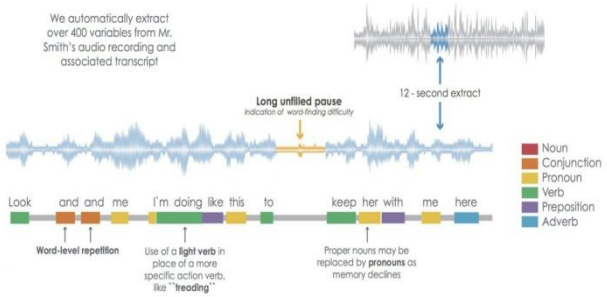

Fig. 2: Analyzing the speech.

Speech signals are divided into parameters and features. Speech parameters contain related data in an effective format. To know the value of parameters simple mathematical formulae will be used. The main issue of parameters is consisting less amount of related data. Whereas speech features are useful to represent the speech. It is calculated by complex mathematical formulae. Most

of the Speech analyzers are considers only parameters for better result. Linear predictive analyzer is best method for removing unnecessary data in speech. In this method both features and parameters are involved. The speech analysis can be used in either frequency domain or time domain. Accurate time resolution is useful for splitting speech signals to determine periods in speech while good frequency resolution useful to identify various sounds.
Speech authentication [5] is process of recognizing a user's identity through their voice. Generally, Authentication provides the security in the form of passwords. The main drawback of providing passwords is forgetting passwords or password hacking by hackers to access our data. For more security consider voice is an authenticate because everyone having a unique voice.
There are two methods which are used for voice authentication they are a) text-based method: It is used for some specified words which may also having verbal passwords; b) text-independent method: It is used for every word which are spoken to recognize individuals. **Figure 3** represents identification of user's voice with the characteristics like sound frequencies, velocity of speech, pronunciation of word etc.

Fig. 3: Authentication on voice.

2. Related Work

Betul KARAKUS, Galip AYDIN [6] were proposed distributed call monitoring system for call center agents here the main goal of this paper is that to convert the recorded files in to text file for getting accurate data by the customers using Apache Hadoop but the main issue of this paper is that the large number of recorded files were stored it requires large amount of storage, provides security for recorded files and giving unique id for each and every recorded file it consumes time.
J.P. Shim , J. Koh , S. Fister , H. Y. Seo [7] had analyzed the customer's call data through text files where text files were converted from recorded files and they used phonetic search technology to the recorded files for getting accurate output. The main issue of this paper is that they were maintained a large number of recorded files.
Speech is one of the best ways of communication which is in form of sounds which are generated by humans and some animals and It is nothing but combination of words/phrases/sentences. Each word includes either vowel or consonants.
Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, Joe Woelfel [8] used sphinx4 framework for better accuracy of recognizing speech. In this paper they compared number of implementations in different modules where frontend supports MFCC, PLP and LPC, Linguist supports various language models and some other techniques like FST, N gram and CFG, the most important primary module is decoder it includes implementation of search manager, viterbi and some other parallel searches. After comparing various implementations Sphinx4 tool provides better accuracy and speed than those modules.
Paul Lamerel, Philip Kwok, William Walker, Evandro Gouvea, Rita Singh, Bhiksha Raj, Peter Wolf [12] were explained about decoder' functionality which were included in sphinx4. The main challenge of this paper is Sphinx4 is static language model and it is not optimized on bigram and trigram tasks at submitting time. So, they were included some techniques like graph constructions mainly for multi-level parallel decoding using multiple features by not using the HMM compounds, search algorithm like viterbi, Token stack decoding for maintaining multiple paths effectively in search process, Designing a general language graph of HMM from language models and grammars from multiple standard methods such as a potential piece between tree and flat search structures for better performance of speed and accuracy.
Saptarshi Boruah, Subhash Basishtha [13] were giving basic information about HMM that how it is works and what is the importance of HMM. In past few years ago for speech recognition was handled by Dynamic time wrapping but that more complicated in continuous speech recognition HMM is used in speech recognition for producing speech and it captures the temporal elasticity and rigorous framework for better result. Sphinx4 is also follows HMM.

3. Speech to Text (STT)

There are many ways of communication to exchange the information. In this, the communication through speech is very effective because by this way of communication human effectively exchanges their information. In this paper we are converting the speech to text for getting accurate data.
Sphinx4 is an Open Source Speech Recognition library tool which works in java [9]. It is the Hidden Markov Model(HMM) [10] based Speech Recognition system. Sphinx4 provides an interface to convert the speech to text by using CMU Sphinx acoustic models. It is used to identify speakers, to adapt models like generate acoustic feature files, convert send-ump and mdef files, updating and recreating acoustic files, to align existing speech to text over timestamping.

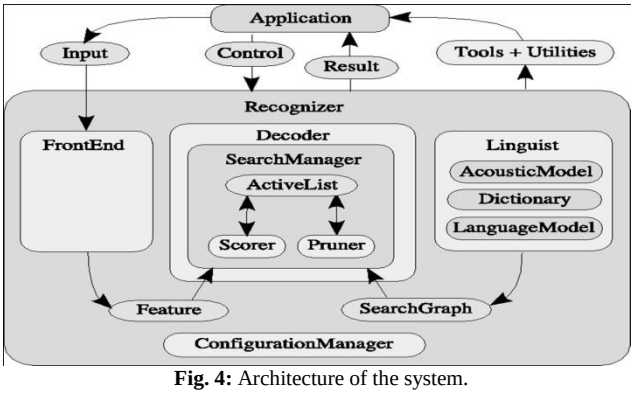

Fig. 4: Architecture of the system.

In **Figure 4** the architecture is basically divided into three primary modules in the Sphinx-4 framework those are Front End, Decoder, and Linguist.

3.1 Front End

The main purpose of frontend is to convert an input signals (audio through microphone) to a sequence of an output (text) [11].

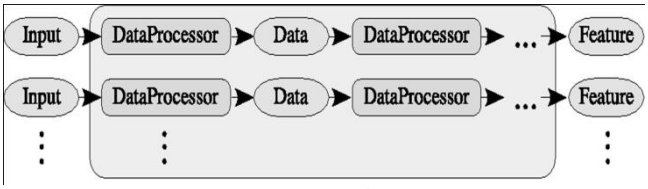

Fig. 5: Architecture of Front End.

According to **Figure 5**, the front end contains one or more parallel chains with communicating signal processing module is known as Data Processors. Simultaneous computations are done by the different parameters from the input signals with the help of multiple chains and also decode is also done by parameters such as PLP, MFCC, and also through video.
In Front End framework, Data Processors implements signal processing techniques includes:Reading from a variety of input formats for batch mode operation means run file automatically with

no interface. Reading from the system audio input device for live mode operation like microphone.
Pre emphasis and windowing with a raised cosine transform (e.g., Hamming and Hanning windows). It also contains discrete Fourier transform (via FFT), mel frequency filtering, bark frequency warping, discrete cosine transforms (DCT), linear predictive encoding (LPC), end pointing, cepstral mean normalization (CMN), mel-cepstra frequency coefficient extraction (MFCC) and perceptual linear prediction coefficient extraction (PLP).

3.2 Configuration Manager

Configuration manager system has two primary purposes:
Which components are to be used in the system and determining the detailed configuration of each of these components. Configuration file is used to configure in sphinx4.

Configuration file maintain the Information of each component. The component connections mean how each component is connected to other one. The detailing configuration on each component.

3.3 Linguist

Linguist [12] is used to represent and manage the search space by decoder and it also generates the search graph and it can hide the complexities which are involved in graph during the generating graph. Linguist consists three components those are language model which is used to create language structure, Dictionary is used by linguist for better pronunciation, mapping the words into sequence of Acoustic Model elements. The main intension of these components for constructing the search graph.
Language Model:
This module produces word level language structure. It can be represented in form of plugins. These are two ways stochastic N-gram model and graph-driven grammar where stochastic N-gram model performs probability for words and observe the previous (n-1) words. Where N-gram model represents as direct graph where each node represents as word each arc represents probability of word transition.
Language model consists of different formats such as simple word list grammar (grammar which is in form of list of words). Java Speech API Grammar: It defines a Backus-Naur Form style, works at any platform, works with any vendor.
Language Model grammar: This Grammar is based on statistical language model. It generates each grammar node for one word and it works effectively with single or double items in n-gram of up to approximately 1000 words. Finite State Transducer grammar: This grammar supports finite state transducer in ARPA format.
Example: I 5
F 0 4.30259
T 0 5 plank 2.60951
Explanation:
I - initial node, so "I 5" where 5 is the initial node.
F - final node, e.g., "F 0 4.30259" 0 is indicates final node and the probability at node 0 is 4.30259 (in -ln).
T - transition, "T 0 5 plank plank 2.60951" means "transitioning from node 0 to node 5, the output is plank and the machine is now in the node plank, and the probability associated with the transition is 2.60951 (in -ln)".

• **SimpleNGramModel:** This model is based on ASCII N-Gram models in the ARPA format. This model is satisfying no optimizes memory usage, so it is used for small files

• **Large Trigram Model**: This model is based on N-Gram models by the CMU Cambridge Statistical Language Modeling Toolkit. This model optimizes memory storage, it used for large files of >=100MB.

3.4 Dictionary

Dictionary is useful for word pronunciations in the language model. It breakdown word into sequence of sub words and it also allows word classification, to be in single word in multiple classes. Sphinx-4 provides interface of Dictionary to support the CMU Pronouncing Dictionary.

3.5 Acoustic Model

Acoustic model provides mapping between HMM and speech in form of unit. We all know that Linguist divides word into sequence of sub words in form of units then pass these units to acoustic model and may retrieve HMM graphs which are associated to those units these are performed to construct the search graph. This HMM graphs are directed graph in this each node represents HMM state and each arc represent transition probability from one state to another state in HMM. By using HMM acoustic can easily have supplied in various topologies. It can share various components at every level component such as transition matrices, Gaussian mixtures, mixture weights.

3.6 Decoder

This is main recognition part where considering features from frontEnd and using search graph which are conjunct by linguist to generate main result. The decoder block consists of a pluggable "search manager" and Search manager contains sub-components like Active List, scorer, pruner for simplifying the process. In decoder block search manager is main component.

3.7 Search manager

It is used to recognize the structure of features. Search manager creates an object which is related to result. In this object every path is reaches the final state where that state is belonging to non-emitted one. Search manager is not fixed to any implementations like linguist module. It follows token-passing algorithm.
i. Active List: It contains set of active tokens depend on time. These tokens are to be sorted in form of list.
ii. Pruner: The main function of pruner is that collecting garbage in java. It removes the unwanted terminal token's path from Activelist. Also, it identifies unshared tokens which is not used, allows the garbage collector to recover the associated Memory.
iii. **Scorer:** It is used to access the features depend on time. By using the mathematical calculations for computing the scorer.

4. Implementation

Algorithm:
Input: Speech from microphones
Output: text file
Step 1: Speak some word/phrases in microphones.
Step 2: Sphinx4 tool detects the speech and recognizes the words/phrases.
Step 3: Get the output in form of text.
Step 4: Comparing the words/phrases whether it is exact word or not.
Step 5: If it generates exact words then save this text as text file.
Step 6: Else go to Step 2.

Process:
First create a java project. Import Sphinx4-core.jar and Sphinx4-data.jar files in this project. Add grammar file with ". gram" extension it contains words, phrases and sentences. Add Sphinx4-core-1.00-javadoc jar file in library. Then create configuration object in main class, it contains Acoustic model, Dictionary, Language model. In Live Speech Recognition we will use grammar file instead of Language model. Create Live Speech Recognizer object to recognize the speech. Pass in configuration object. Now

recognition process has started by using start Recognition() method. For speech analysis we use get Hypothesis() method. Get an output in form of text file.

Figure 6 represents the process of creating an object which is named as "Configuration" and how it passes to the recognizer. The created configuration object used for creating and importing the required files.

```
public Main() {

    // Loading Message
    logger.log(Level.INFO, "Loading..\n");

    // Configuration
    Configuration configuration = new Configuration();

    // Load model from the jar
    configuration.setAcousticModelPath("resource:/edu/cmu/sphinx/models/en-us/en-us");
    configuration.setDictionaryPath("resource:/edu/cmu/sphinx/models/en-us/cmudict-en-us.dict");

    // if you want to use LanguageModelPath disable the 3 lines after which
    // are setting a custom grammar->

    // configuration.setLanguageModelPath("resource:/edu/cmu/sphinx/models/en-us/en-us.lm.bin")

    // Grammar
    configuration.setGrammarPath("resource:/grammars");
    configuration.setGrammarName("grammar");
    configuration.setUseGrammar(true);
```

Fig. 6: Creating Configuration object.

After creating a configuration object, the recognition process in **Figure 7** was started. This process is useful to take our speech through micro phone and then start the processes.

```
    try {
        recognizer = new LiveSpeechRecognizer(configuration);
    } catch (IOException ex) {
        logger.log(Level.SEVERE, null, ex);
    }

    // Start recognition process pruning previously cached data.
    recognizer.startRecognition(true);

    // Start the Thread
    startSpeechThread();
    startResourcesThread();
}
```

Fig. 7: Live Speech Recognition.

Figure 8 represents get Hypothesis() method in speech Result object for checking that what type of speech is giving by the user. This method uses the recognizer to display the accurate result to the user.

```
*/
protected void startSpeechThread() {

    // alive?
    if (speechThread != null && speechThread.isAlive())
        return;

    // initialise
    speechThread = new Thread(() -> {
        logger.log(Level.INFO, "You can start to speak...\n");
        try {
            while (true) {
                /*
                 * This method will return when the end of speech is
                 * reached. Note that the end pointer will determine the end
                 * of speech.
                 */
                SpeechResult speechResult = recognizer.getResult();
                if (speechResult != null) {

                    result = speechResult.getHypothesis();
                    System.out.println("You said: [" + result + "]\n");
                    // logger.log(Level.INFO, "You said: " + result + "\n")

                } else
                    logger.log(Level.INFO, "I can't understand what you said.\n");

            }
        } catch (Exception ex) {
            logger.log(Level.WARNING, null, ex);
        }

        logger.log(Level.INFO, "SpeechThread has exited...");
```

Fig. 8: GetHypothesis() method in SpeechResult object.

Figure 9 represents where Buffered Reader class. It is useful to read the output from console, with the help of file Writer object and print Writer class is used to print the output in text file.

```

    BufferedReader in =
        new BufferedReader(new InputStreamReader(System.in));
    String name = "Instructor";
    System.out.print("Give your name: ");
    try {
        name = in.readLine();
        PrintWriter fileWriter = new PrintWriter("write.txt");
        PrintWriter fileWriter1 = new PrintWriter("write1.txt");

        fileWriter.println( "You said: [" + result + "]\n" );
    fileWriter.close();
        fileWriter1.println("You said: [" + result + "]\n");
        fileWriter1.close();


        }
        catch(Exception e) {
           System.out.println("Caught an exception!");
        }
```

Fig. 9: Save output from console to text file.

The output is generated in both console and text file where, **Figure 10** represents the output in console and **Figure 11** represents the output which is generated in text file.

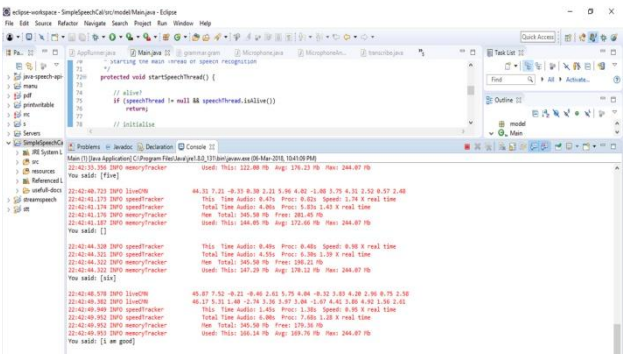

Fig. 10: Console output.

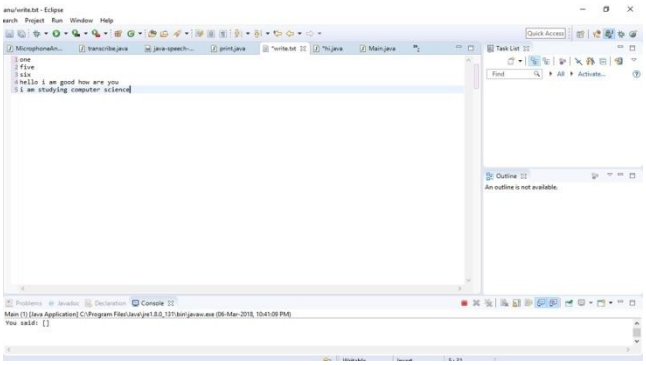

Fig. 11: Output in text file.

5. Conclusion

In this paper we proposed to convert live calls into text files. So, there is no need to store recorded files by this we can reduce the space, no need to provide any security for recorded files, reduces the cost and time. Here for converting speech to text we are using sphinx4 library It is used in java platform where it is easy to understand. Sphinx4 supports so many languages like US English, India-English, Hindi, etc. It is helpful to identify the speakers, maintaining dictionary having large vocabulary. The main idea of this paper is to convert speech to text with more accuracy So we are trying to improve our code or will use another method for getting better result.

References

[1] https://en.wikipedia.org/wiki/Speech.
[2] https://en.wikipedia.org/wiki/Speech_recognition.
[3] http://searchcrm.techtarget.com/definition/speech-recognition.
[4] https://en.wikipedia.org/wiki/Voice_analysis.
[5] http://whatis.techtarget.com/definition/voice-authentication.
[6] Betul Karakus, Galip Aydin, "Call center performance Evaluation Using Big Data Analytics",*ISNCC,IEEEXplore*,2016.

[7] J.P.Shim , J.Koh , S.Fister , H.Y.Seo, " Phonetic Analytics Technology and Big Data: Real-World Cases", *Communication of ACM*, (2016).
[8] Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, Joe Woelfel, "Sphinx-4: ‘‘A flexible open source framework for speech recognition", (2004).
[9] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.4395 &rep=rep1&type=pdf.
[10] Saptarshi Boruah, Subhash Basishtha, "A study on HMM based speech recognition system", *IEEE Xplore*, (2013).
[11] http://www.gavo.t.utokyo.ac.jp/~kuenishi/java/sphinx4/edu/cmu/sphinx/frontend/FrontEnd html.
[12] https://cmusphinx.github.io/doc/sphinx4/javadoc/edu/cmu/sphinx/linguist/Linguist.html.