Smart device controlling through voice commands given in malayalam language

Muneer.V.K.^{1*}, Muhamed Basheer K.P.² and Rababa Kareem Kollathodi³

Abstract

In this paper, a new method to read and recognize the voice command given in Malayalam language with the help of advanced language processing algorithms. Tensor flow is taken as the tool for voice processing in python language and various methods to train the signals and test them in second level. The acoustic model will check whether the signal can be recognized by these methodologies for further actions to be taken down. The accuracy is also to be considered with large data set.

Keywords

Speech processing, MFCC, Fourier Transform, Acoustic model, Tensor flow, Delta filtering, double delta, feature extraction, speech corpus.

^{1,2,3} Department of Computer Science, Sullamussalam Science College, Areekode, Kerala-673639, India.

*Corresponding author: ¹ vkmuneer@gmail.com; ² mbasheerkp@gmail.com; ³ rababakareem@gmail.com

Article History: Received 24 January 2019; Accepted 24 May 2019

©2019 MJM.

Contents

1	Introduction	445
2	Literature Review	446
3	Problem Definition	446
4	Objectives	447
5	Speech Corpus Description	447
6	Challenges of Speech Corups	447
7	Speech Recognition System Architecture	447
8	Modules	447
9	Experimental Values	448
10	Conclusion	449
	References	449

1. Introduction

In India there are more than 1683 spoken languages with 126 crore people, out of which, the literacy rate is 74.04% as per the Population Census of India, 2011. With so many spoken languages, there is huge language barrier between the citizens of India. Automatic Speech Recognition (ASR) technology can be used to share information and knowledge within the society easily, without even learning other language. Thereby, speech technology can remove or reduce the language barrier between the people and states.

Developing SRS (Speech Recognition system) model for Indian languages is more challenging as; the individual languages have many dialects, and also have different meaning for same words in different geographical regions. SRS is a computerized process. Computer /Machine receive as its input a speech recording and it produces as its output a transcription. The goal of speech recognition is for a machine to be able to “hear”, “understand,” and “act upon” spoken information.

Voice can combine two factors, namely, personal voice recognition and telephone recognition. Voice recognition systems are cheap and easily understood by users. In today's smart world, voice recognition plays a very critical role in many aspects. Voice based banking, home automation and voice recognition based gadgets are some of the many applications of voice recognition[1].

The main aim of an ASR system is to accurately and efficiently convert a speech signal into a text message transcription of the spoken words, independent of the device used to record the speech (i.e., the transducer or microphone), the speaker's accent, or the acoustic environment in which the speaker is located (e.g., quiet office, noisy room, outdoors).

The speech recognition is divided into two parts that means speaker dependent and speaker independent modes. In the speaker independent mode of the speech recognition the computer should ignore the speaker specific characteristics of the speech signal and extract the intended message. On the other hand in case of speaker dependent recognition machine should extract speaker characteristics in the acoustic signal.

SRS (Speech recognition systems) can be classified into several categories by describing what types of utterances the algorithm have the ability to recognize. These categories are based on the fact that one of the difficulties of ASR is the ability to determine when a speaker starts and finishes an utterance. The types of utterances are Isolated Words, Connected Words, Continuous Speech and Spontaneous Speech.

Isolated Word recognizers usually require each utterance to be in silent mode on both sides of the sample window. It does not mean that it accepts single words, but only it means that it requires a single utterance at a time. Often, these systems have “Listen /Not-Listen” states, where they require the speaker to wait between utterances (usual processing during the pauses). Isolated Utterance might be a better name for this type of Speech recognition.

Connected Word systems (or more correctly ‘connected utterances’ or ‘Phrase’) are similar to isolated words, but allow separate utterances to ‘run-together’ with minimal pause between them. An utterance is the vocalization (speaking) of a word or words that represent a single meaning to the computer. Utterance can be a single word, or a collection of a few words, a single sentence, or even multiple sentences. In the word-level stage, each stored word-pattern is matched against all possible regions in the connected word-input pattern. An adjustment window is used to define a region in which each word in the connected word-pattern may start and end for connected speech recognition.

Continuous Speech recognizers must be capable of recognizing continuous speech. The difficult task is to create continuous sentences because such a construction requires special methods to determine utterance boundaries. Continuous speech recognizers allow users to speak almost naturally, while the computer takes up the task of determining the content. Computer dictation and recognizing capability are basic to determine the quality of the work.

Spontaneous Speech is natural sounding and unrehearsed. An SRS with spontaneous speech ability should be able to handle a variety of natural speech features such as words being run together, some disfluencies like “ums” and “ahs”, and even slight stutters.

SRS (Speech recognition system) software can be designed in three different ways;

- a) Speaker-dependent
- b) Speaker-independent and
- c) Speaker-adaptive.

Speaker dependent software only recognizes the speech of user it is trained to understand. It allows for very large vocabulary. These systems are usually easier to develop, cost-effective and more accurate, then but not as flexible as speaker adaptive or speaker independent systems. SRS software that can recognize a variety of speakers without any prior training is known as SI (**Speaker-Independent**). SI software generally limits the number of words in a vocabulary, but it is the only realistic option for applications such as Interactive Voice Response System (IVRS) that must accept input from a

large number of users. A speaker independent system is developed to operate for *anyspeaker* of a particular type. These systems are the most difficult to develop, most expensive and accuracy is lower than speaker dependent systems. However, they are more flexible. A **Speaker Adaptive (SA)** system is developed to adapt its operation to the characteristics of new speakers. Its limitation lies somewhere between speaker independent and speaker dependent systems [3]

2. Literature Review

Research on speech recognition on regional languages is still growing on. Even though there different obstacles in these languages, few of the languages has proved the processing is successful. These translation is performed by using well know algorithms like HMM, Neural network and Language processors(LP).

In context of Indian languages, few of the prominent languages has already been reached far ahead in research. Hindi Speech Recognition system which has been trained on 40 hours of audio data and has a trigram language model that is trained with huge collection of words. For a vocabulary size of 65000 words, the system gives a word accuracy of 75% to 95% approximately [5].

Paper [2] describes a novel ‘Language Processor’ (LP) which uses syntactic and semantic properties of Indian language. They experimented for Speech recognition system for railway reservation inquiry. The acoustic level recognizer provides several alternatives for each word with associated ‘confidence levels’. Using these confident levels, the LP has to generate a ‘most likely’ sentence consistent with the restrictions imposed by syntax and semantics. Paper [4] has experimented to automatically segment and label continuous speech signal into syllable-like units for Indian languages (Tamil and Telugu). In this approach, the continuous speech signal is first automatically segmented into syllable-like units using group delay based algorithm. Google has recently released an application for translation of speech to text to regional Indian languages, in which the sentence speaks in Malayalam is taken as input and corresponding Malayalam text is generated. However no action or control on smart devices is implemented by the application.

3. Problem Definition

Speech is the most essential and major communication mode among human beings. Speech has the importance to become an essential interaction mode with computer when such communication becomes a reality. The applications of automatic speech recognition are becoming useful and similar in this day as several of the modern devices are configured and generated user friendly for the convenience of individuals. Although numerous interactive applications of software are feasible the uses of automatic speech recognition (ASR) are restricted due to barriers of language. Hence the growth of automatic speech recognition in local languages will help

individuals to make use of this technological growth. The performance of ASR is also depend on the speech signal recording devices, environments of recording, sampling rate, speakers dialects and gender.

In this study automatic speech recognition is designed for controlling the smart devices using Malayalam language. This study considers Malayalam language which has not been used by many studies in speech recognition context. Here our automatic speech recognition system has to work similar to Apple Siri software, but it is accepting the input as Malayalam language and it will process that input signal and do the necessary activities based on the command given through voice.

4. Objectives

To design and implement the smart device controlling through the Malayalam language voice commands. The designed model should work for natural recording environment. The model must be capable to recognize the inputted Malayalam speech signal recorded using any type of recording devices namely microphone, head phone inserted in any type of smart phones and trigger an activity based on the input command.

5. Speech Corpus Description

In this experiment we have listed all basic operations/commands given to smart phone and recorded each commands for 10 times. The signals are recorded at sampling rate of 44 KHz, 16 bps, stereo channel. These signals were recorded at a little noisy room environment, while computer's built in voice recorder Software was used to record with the help of mini microphone of frequency response 50 – 12500 Hz.

6. Challenges of Speech Corups

There are the several challenges to be addressed in speech recognition system. One of the biggest hurdles is collecting proper input commands. This is the fundamental element to be collected for building speech corpus. The recording environment, selection of commands, collection of different inputs from different speakers, organizing the corpus and retrieval other added challenges. The more input increases, accuracy can proportionally increase.

In this experiment, the inputs are taken in a noisy environment by using small microphone. All fundamental commands commonly used to handle smart phone have been listed and recorded ten times each. The files were in m4a format with 44100 sample rate. Those further undergone editing to make them in wav format with 16000 Hz by using audacity software. Then they numbered by using customized naming convention and organized in such a way that they can easily accessed later.

7. Speech Recognition System Architecture

Audio signal is taken as input to the system. The analog input signal to be converted to digital format by using PCM and Linear PCM. This signal is then forwarded to perform pre-processing to remove noise, filtering and strengthen the signal.

Figure 1: Architecture of Speech Recognition System

8. Modules

Sampling and Pre-processing

The pre-processing stage in speech recognition systems is used in order to increase the efficiency of subsequent feature extraction and classification stages and therefore to improve the overall recognition performance. Commonly the pre-processing includes sampling step, a windowing and a de-noising step.

Feature Extraction

After the pre-processing step, feature extraction is the second component of automatic speech recognition (ASR) systems. This component should derive descriptive features from the windowed and enhanced speech signal to enable a classification of sounds. The feature extraction is needed because the raw speech signal contains information besides the linguistic message and has a high dimensionality. Both characteristics of the raw speech signal would be unfeasible for the classification of sounds and result in a high word error rate. Therefore, the feature extraction algorithm derives a characteristic feature vector with a lower dimensionality, which is used for the classification of sounds.

There are a couple of well defined algorithms used in feature extraction of voice commands. One of the most popular algorithm is Mel Factor Cepstral Coefficient(MFCC) which we used here for feature extraction. As the data set is comparatively small for this experimental environment only basic filtering is performed on the signals.

Sample Code:

```

import scipy.io.wavfile as wav
import python_speech_features
from python_speech_features import mfcc
  
```

```

import scipy.io.wavfile as wav
(rate,sig)=wav.read("wav\16K\train\name\200100101.wav")
print(rate)
mfcc_feat = mfcc(sig,rate,512)
print(mfcc_feat)

```

While performing the mfcc filter on each signal, the noise and unwanted areas are removed and the essential features sufficient to identify a particular signal is extracted and stored in a dataset.

Sample Code:

```

(rate,sig) = wav.read(str1)
mfcc_feat=mfcc(sig,rate,512)
print("Delta mfcc")
d_mfcc_feat = delta(mfcc_feat, 2)
print(d_mfcc_feat)Delta

```

Block diagram of MFCC

Training and Matching

Once the feature extraction completes, the voice parameters then passed for training phase. The machine has to learn from the predefined input features. Training of system will end up with building of language model. The training and further testing is implemented by the technology called Tensor Flow

Tensor Flow is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. It is a symbolic math library, and also used for machine learning applications such as neural networks. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

9. Experimental Values

For the development stage of this project, the signals are recorded and stored in .wav format. Ten sample signals are recorded for each command. Out of this 7 commands are

reserved for training and 3 for testing. Signal Processing is done in Python language. Development environment is in Anaconda 3.6 version. Feature extraction is done by using MFCC algorithm. Training and Testing is performed in Tensor flow environment.

The stages of processing are listed below:

1. Inclusion of Packages

```

import speech_recognition
import sounddevice as sd
import numpy as np
import scipy.io.wavfile as wav
import python_speech_features
from python_speech_features import mfcc
import scipy.io.wavfile as wav
import matplotlib.pyplot as plt
from scipy import signal
import csv
from python_speech_features import delta
from python_speech_features import logfbank

```

2. Pre processing of signal : PCM model

```

with open("example1.csv","r") as f:
    reader=csv.reader(f)
    for row in reader:
        print(row)
f.close()

```

```

22.92136609, "5.956635901", "1.3322534", "-9.512082762", "20.04966842", "-5.713601305"
["mfcc", "8.504489002", "-2.512494551", "-2.3004043", "26.48851626", "-2.670449576", "12.23152354", "-6.646573628"
"3.670925961", "-23.5723685", "13.9969621", "-15.50906804", "-16.37283997", "-1.273915434"]
["cmph", "13.00427812", "18.52884293", "13.51159247", "17.84235517", "11.3734333", "6.375945831", "11.51390615",
"15.94579842", "-6.8558113", "12.12897244", "1.696549547", "-2.598263309", "1.811169102"]
["cmph", "12.92071703", "18.73336816", "13.00113919", "17.14829864", "13.39357187", "4.616527361", "11.54399518"
"14.77257972", "-9.487513", "16.41898875", "-5.496269404", "0.118193018", "12.25284234"]
["cmph", "8.401645706", "4.312437395", "0.025807935", "13.89875586", "2.389243097", "12.13923016", "7.317036851"
"17.7917903", "6.394860859", "12.81144418", "-10.06305003", "14.6383306", "5.655258462"]
["cmph", "10.39865257", "0.271716289", "5.565978891", "24.47218883", "6.250826182", "6.105819238", "6.545200773"
"25.90515672", "-12.14428579", "23.0777244", "3.279723817", "-2.185963771", "12.25799185"]
["cmph", "9.419518427", "4.18002187", "0.846915475", "13.93717917", "16.12073265", "16.45886136", "12.14507626"
"12.46347252", "-12.36054294", "10.96413274", "1.232724668", "14.10124257", "4.067294654"]
["cmph", "11.90501529", "13.2992976", "9.306042474", "22.19935218", "13.5748971", "6.966918831", "0.843090113",
20.91456315", "1.244828345", "6.261694220", "3.677207109", "7.84688624", "-5.091206861"]
["cmph", "14.08559529", "21.391871", "16.21704296", "14.58244736", "7.981290667", "11.90361681", "13.6865521",
.887908561", "2.136696407", "8.769813426", "0.642426359", "3.833372547", "10.11188739"]
["cmph", "11.54819695", "13.92990845", "12.5162558", "24.17851936", "4.644157441", "5.728050468", "10.74165887"

```

3. Extract of Code Snippet

```

myFile = open('example1.csv', 'w')
with myFile:
    with open("path.txt","r") as f2:
        for row in f2:
            for index, item1 in enumerate(row.split()):
                item1.strip()
                str1=item1.strip()
                print(str1)
                (rate,sig) = wav.read(str1)
                mfcc_feat=mfcc(sig,rate,512)
                #d_mfcc_feat = delta(mfcc_feat, 2)
                #fbank_feat = logfbank(sig,rate)
                #print(fbank_feat[1:3,:])
                print(mfcc_feat)
                writer = csv.writer(myFile)

```

```
writer.writerow(mfcc_feat)
print("Writing complete")
del sig,rate,str1
```

```
#f1.close()
f2.close()
#print(mfcc_feat)
```

4. Input signal : Plotted Graphical representation

5. MFCC Feature extraction

```
import scipy.io.wavfile as wav
import matplotlib.pyplot as plt
import python_speech_features
from python_speech_features import mfcc
import matplotlib.pyplot as plt

(rate,sig) = wav.read("wav/16K/train/makeCall/200100101.wav")
mfcc_feat = mfcc(sig,rate)

print(mfcc_feat)
plt.plot(mfcc_feat)
plt.show()

f1=open("pre_data.xls","w")
f1.write(str(mfcc_feat))
f1.close()
```

[[11.63800093	19.5883892	12.12007157	...	-0.92992861	1.56789163
-10.33510269]					
[12.86059787	19.43498177	12.01193945	...	-0.42464516	2.70106031
-6.86815176]					
[13.06206545	13.25253908	11.62713262	...	-0.15487113	-2.93420202
-3.3016054]					
...					
[16.79890418	2.65988654	-7.00612406	...	-3.64094059	3.94835811
2.13762253]					
[16.70081699	2.45704018	-7.30495928	...	-4.84228931	2.68248008

6. Extracted Values

```
wav\16K\train\six\200101206.wav
WARNING:root:frame length (8192000) is greater than FFT size (512),
[[ [ 11.25437043 12.73722107 8.13183663 21.28678395 14.61203838
4.92997541 1.3362473 21.64886155 -6.45652405 15.15912406
5.85372316 8.42867766 -2.44613134]]
Writing complete
wav\16K\train\six\200101207.wav
WARNING:root:frame length (8192000) is greater than FFT size (512),
[[ [ 1.37549461e+01 2.17832075e+01 1.48087052e+01 1.25746174e+
1.39752061e+01 8.72232121e+00 1.10037517e+01 9.46369594e+
2.60107546e+00 4.10228714e+00 -3.62742839e-03 2.88236329e+
4.12713922e+00]]
Writing complete
wav\16K\train\seven\200101301.wav
WARNING:root:frame length (8192000) is greater than FFT size (512),
[[ [ 12.05645026 13.12435072 13.73233838 24.86209228 -1.09298296
16.86421833 11.95614709 9.23826713 -13.50680072 26.68281898
-3.73412812 -14.78022691 27.95241722]]
Writing complete
wav\16K\train\seven\200101302.wav
WARNING:root:frame length (8192000) is greater than FFT size (512),
```

7. Training Tensor Flow

```
import tflearn #pip install tflearn
import speech_data
learning_rate=0.001 # 0.0001
training_iters=1000 #300000
batch = word_batch = cav.mfcc_batch_generator(13)
X,Y = next(batch)
trainX,trainY = X,Y
testX, testY, X, Y
net = tflearn.input_data([None, 20, 80])
net = tflearn.lstm(net, 128, dropout=0.8)
net = tflearn.fully_connected(net, 10, activation='softmax')
```

10. Conclusion

Speech is the most prominent and the primary mode of communication among human beings. Speech has the potential to become an important mode of interaction with computer when such an interaction becomes a reality. Designing a speech recognition system to control smart devices for Malayalam language will take much effort in present scenario. However it can be done once the effective speech corpus is designed in Malayalam. Hence, it can be design to make a smart device to control via human natural voice in natural environments. This technology will leads to revolution in the field of speech recognition technology and all the electronic smarts device can be controlled by voice.

References

- [1] Koustav Chakraborty, Asmita Talele, Voice Recognition Using MFCC Algorithm, *International Journal of Innovative Research in Advanced Engineering*, 1(2014), 2349–2163.
- [2] Rabiner L, Jung B-H Fundamentals of speech recognition, Pearson Education (Singapore) Private Limited, Indian Branch, 482 F.I.E Patpargans, Delhi 110092, India.(1993)
- [3] G Lakshmi Sarada, et al., Automatic transcription of continuous speech into syllable-like units for Indian languages, *Sadhana*, 34(2009), 221–233.
- [4] <http://www.lumenvox.com/resources/tips/tipsglossary.asp/speechRecognition>.
- [5] Rabiner L. R., Juang. B. H., Speech recognition: Statistical methods. *Encyclopedia of language & linguistics*, (2006) 1–18.
- [6] Nitendra Rajput, Ashish Verma and Chalapathy Neti, A Large Vocabulary Continuous Speech Recognition System For Hindi, NCC, Mumbai (2002).
- [7] Hemakumar G, Punithavalli M and Thippeswamy K, Speech Recognition Technology: A Survey on Indian Languages, *International Journal of Information Science and Intelligent System*, 2(2013).
- [8] Steve Renals, Multilingual Speech Recognition, *Automatic Speech Recognition Lecture* 16(2017).
- [9] L. Lamelet al., Speech Transcription In Multiple Languages, *Spoken Language Processing Group LIMSI-CNRS*.
- [10] Zhirong Wanget al., Towards Universal Speech Recognition, Carnegie Mellon University, Pittsburgh.

- [11] Hemakumar G, Punithavalli M and Thippeswamy K, Large Vocabulary in Continuous Speech Recognition Using HMM and Normal Fit, *International Journal of Computer Trends and technology (IJCTT)*,42(2016), 2231–2803.
- [12] Hemakumar G. and Punitha P., Large Vocabulary Speech Recognition: Speaker Dependent and Speaker Independent, *Advances in Intelligent and Soft Computing*,339(2015),73-80.
- [13] Hemakumar G. and Punitha P, Automatic Segmentation of Kannada Speech Signal into Syllables and Sub-words: Noised and Noiseless Signals, *International Journal of Scientific & Engineering Research*, 5(2014),1707-1711.

ISSN(P):2319 – 3786

Malaya Journal of Matematik

ISSN(O):2321 – 5666