Hindi Speech Recognition

Akanksha Saxena

Student at Galgotias College of Engineering and Technology, Greater Noida, Uttar Pradesh, India

Abstract: ***This work is to develop a speech recognition system for speaker dependent, real time, isolated words of Hindi Language. Speech is a useful and effective communication medium with machines, especially in the environment where keyboard input is awkward or impossible. The Speech Processing technology (Automatic Speech Recognition and Speech Synthesis) has made great progress for European languages. In India, almost three-fourth of the population lives in rural areas and most of the population is unfamiliar with computers and English. It would be a great boon for Indian society if communication with machines, mainly with computers, in native languages can be made possible. It will enable people to interact with computers in their own language and without the use of keyboard. This report is a study of the technology and modeling techniques, having been used for Speech Processing. The focus of the present work is Hindi Speech Recognition.***

Keywords: Natural language processing, Hidden Markov Model, Mel frequency cepstral coefficient, Human computer interaction, Viterbi algorithm

1. Introduction

Hindi speech Recognition is the field of computer science that deals with designing computer systems that can recognize spoken words. They generally require an extended training session during which the computer system becomes accustomed to a particular voice and accent. Such systems are said to be speaker dependent. Many systems also require that the speaker speak slowly and distinctly and separate each word with a short pause. These systems are called discrete speech systems. Recently, great strides have been made in continuous speech systems – voice recognition systems that allow you to speak naturally. There are now several continuous-speech systems available for personal computers. Because of their limitations and high cost, voice recognition systems have traditionally been used only in a few specialized situations. For example, such systems are useful in instances when the user is unable to use a keyboard to enter data because his or her hands are occupied or disabled. Instead of typing commands, the user can simply speak into a headset. Increasingly, however, as the cost decreases and performance improves, speech recognition systems are entering the mainstream and are being used as an alternative to keyboards. It appears that most computer users can create and edit documents more quickly with a conventional keyboard, despite the fact that most people are able to speak considerably faster than they can type. Using both keyboard and speech recognition simultaneously, however, can in some cases be more efficient than using any one of these inputs alone. Additionally, heavy use of the speech organs results in vocal loading. Also, the typical office environment with high amplitude of background speeches is among the most adverse environment for current speech recognition technologies. For use with computers, analog audio must be converted into digital signals. This requires analog-to-digital conversion. For a computer to decipher the signal, it must have a digital database, or vocabulary, of words or syllables, and a speedy means of comparing this data with signals. The speech patterns are stored on the hard drive and loaded into memory when the program is run. A comparator checks these stored patterns against the output of the A/D converter. It has been noticed that the success of isolated-word automatic speech recognition systems requires a combination of various techniques and algorithms, each of which performs a specific task for achieving the main goal of the system. Therefore, a combination of related algorithms improves the accuracy or the recognition rate of such applications. A speech recognizer system comprises of two distinct blocks, a Feature Extractor and a Recognizer.

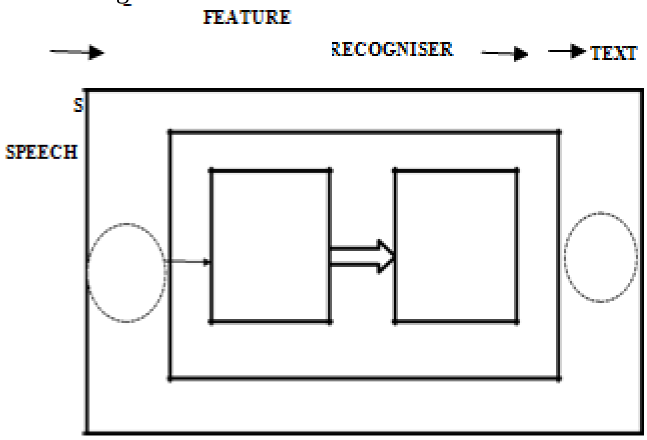

Figure 1: Building blocks of speech recognition system

One of the major problems in speech recognition is to find suitable front end features. Various front end features used by different researchers are linear prediction coefficients (LPC) ,the reflection coefficients (RC), the linear prediction cepstrum coefficients (LPCC), mel-frequency cepstrum coefficients (MFCC), linear frequency cepstrum coefficients (LFCC) etc.

Here we focus on the scenario when the Feature extractor block uses a standard MFCC coder, which translates the incoming speech into a trajectory in the Mel Frequency cepstrum feature space. The trajectories on such reduced dimension spaces can provide reliable representations of spoken word, while reducing the training complexity and the operation of the recognizer. The output of the FE block is blind i.e. it does not care about the word that is being represented by that trajectory. The FE block only transduces an incoming pressure wave into a trajectory in some feature space. It is the Recognizer block that discovers the relationships between the trajectories and recognizes the word. The Recognizer is to be designed in two different ways using Neural Networks. The Neural Network architectures used are Recurrent Neural Networks and the Multi Layer Perceptrons. Here we will focus on Multi Layer Perceptron Neural Network.

A speech recognition system, using the pattern recognition capabilities of neural networks, and other mathematical and signal processing tools will be able to correctly identify simple words. The system will recognize samples that it trained with, and will also be able to generalize to other samples of the same word. As larger vocabularies are used, recognition accuracy will decrease.

1.1 Speech Sample Collection

Speech samples collection is mostly concerned with recording various speech samples of each distinct word by different speakers. However, Rabiner and Juang [1] identified four main factors that must be considered when collecting speech samples, which affect the training set vectors. Those factors include who the talkers are; the speaking conditions; the transducers and transmission systems and the speech units

The first factor is the profile of the talkers/speakers. For the purpose of increasing robustness of the IWASR we can train the system using speakers of various age, sex and regions. The second factor is the speaking conditions in which the speech samples were collected from, which basically refer to the environment of the recording stage. The IWASR speech samples collection is usually done in a noisy environment. The rationale behind collecting the speech samples from noisy environments is to represent a real world speech samples collection, because most speech recognition systems are meant to be used in different environments and spheres. Therefore, collecting speech samples from noisy environments is purposely done. After the speech samples are collected they are converted from analog to digital form by sampling at a frequency of about 16,000 Hz. Sampling involves recording the speech signals at a regular interval. Since a high sampling frequency is used it doesn't involve loss of critical data. The collected data is now quantized if required to eliminate inherent noise in the speech samples. The collected speech samples are then going to pass through the features extraction, features training and features testing stages

1.2 Feature Extractor

The first step in developing this speech recognition program is to design a feature extractor. The FE block can be modeled after the stages evidenced in the human biology and development. This is a block that transforms the incoming sound into an internal representation such that it is possible to reconstruct the original signal from it. This stage can be modeled after the hearing organs, which first transduce the incoming air pressure waves into a fluid pressure wave and then convert them into aspecific neuronal firing pattern. The Feature Extraction block used in speech recognition should aim towards reducing the complexity of the problem before later stages start to work with the data.Once the Feature Extraction block completes its work**,** the Recognizer module classifies its output. It integrates the sequences of phonemes into words. This module sees the world as if it where only composed of words and classifies each of the incoming trajectories into one word of a specific vocabulary. The process of correlating utterances to their symbolic expressions, translating spoken language into written language, is called speech recognition. The recognizer is built using the neural network.

2. Design and Implementation using HTK

HTK is the “Hidden Markov Model Toolkit” developed by the Cambridge University Engineering

Department (CUED). This toolkit aims at building and manipulating **Hidden Markov Models** (HMMs). HTK is primarily used for **speech recognition** research HTK consists of a set of library modules and tools available in C.

2.1 Steps in the development of application using HTK:

2.1.1 Creation of directories in the home directory

- data/ : to store training test data (speech signals, labels, etc.), with 3 sub-directories
- data/sig and data/lab
- model/proto
- model/hmm0
- model/hmm1
- model/hmm2
- model/hmm3

This directory structure is created inside PROJECT directory.

2.1.2 Creation of a training database

First, we have to record the words speech signals with which word models will be trained (the *training corpus*). Each speech signal has to be labelled, that is: associated with a text (a label)describing its content. Recording and labelling can be done with the HSLab HTK tool (t any other tool could be used).

To create and label a speech file:
1. Recording- We have used linux command arecord for the recording.
2. Labelling- We have used wavesurfer for labeling the wav files.

2.1.2.1 Recording the signal

The command used for recording a signal is:

arecord -d 10 -f cd -c1 -t wav -r 16000 test.wav

- -d: duration
- -f: format
- -t: file type
- -r: rate(Hz)

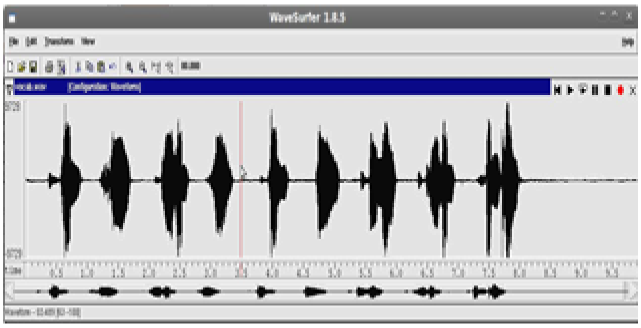

Figure 2: Recorded voice

2.1.2.2 Label the file

To label the speech waveform, first press “*Mark*”, then select the region you want to label. When the region is marked, press “Label as”, type the name of the label, then press Enter.

For each signal, we have to label 3 successive regions: start silence (with label sil), the recorded word with label of the digit uttered), and end silence (with label sil). These 3 regions cannot overlap with each other (but no matter if there is a little gap between them).When the 3 labels have been written, press “*Save*”: a label file called any_name_0.lab is created.

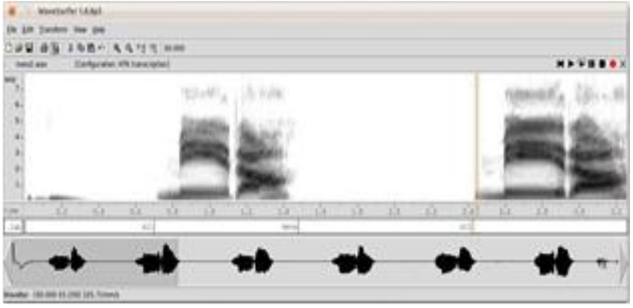

Figure 3: LabelledVoice

2.1.2.3 Renaming the files

We need to rename our files .the convention we used is digit_number.sig. 10 samples of each digit are recorded. The signal files should be stored in a data/train/sig/ directory (the training corpus), the labels in a data/train/lab/ directory (the training label set).

2.1.3 Acoustical Analysis

The speech recognition tools cannot process directly on speech waveforms. These have to be represented in a more compact and efficient way. This step is called “acoustical analysis”:

- The signal is segmented in successive frames (whose length is chosen between 20ms and 40ms, typically), overlapping with each other.
- Each frame is multiplied by a windowing function (e.g. Hamming function).
- A vector of acoustical coefficients (giving a compact representation of the spectral properties of the frame) is extracted from each windowed frame

The conversion from the original waveform to a series of acoustical vectors is done with the HCopy HTK tool:

```
HCopy -A -D -C analysis.conf -S targetlist.txt
```

analysis.conf is a configuration file setting the parameters of the acoustical coefficientextraction.

targetlist.txt specifies the name and location of each waveform to process, along with thename and location of the target coefficient files.

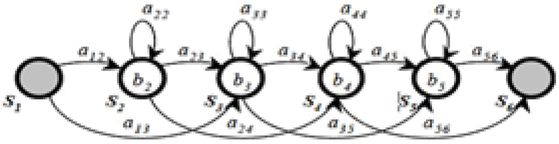

Figure 4: HMM Parameter

2.1.3.1 Configuration Parameters

The configuration file (analysis.conf) is a text file (“#” is used to introduce a comment). For the development of our application, the following configuration file is used:

SOURCEFORMAT = HTK # Gives the format of the speech files TARGETKIND = MFCC_0_D_A # Identifier of the coefficients to use

# Unit = 0.1 micro-second:

WINDOWSIZE=250000.0# = 25 ms = length of a time frame TARGETRATE = 100000.0 # = 10 ms = frame periodicity NUMCEPS = 12 # Number of MFCC coeffs (here from c1 to c12)

USEHAMMING = T # Use of Hamming function for windowing frames PREEMCOEF = 0.97 # Pre-emphasis coefficient

NUMCHANS = 26 # Number of filterbank channels CEPLIFTER = 22 # Length of cepstralliftering

With such a configuration file, an MFCC (Mel Frequency Cepstral Coefficient) analysis is performed (prefix “MFCC” in the TARGETKIND identifier). For each signal frame, the following coefficients are extracted:

2.1.4 HMM Definition

In this step, 40 acoustical events have to be modelled with a Hidden Markov Model (HMM) for words and 1 for silence: all the 40 words and “sil”. For each one we will design a HMM.

The first step is to choose *a priori* a topology for each HMM:

- number of states
- form of the observation functions (associated with each state)
- disposition of transitions between states

Figure 5: Basic HMM topology

2.1.5 HMM training

The following command line initializes the HMM by time-alignment of the training data with a Viterbi algorithm.

HInit -A -D –T 1 -S trainlist.txt -M model/hmm0 -H model/proto/hmmfile -l label -L label_dirnameofhmm

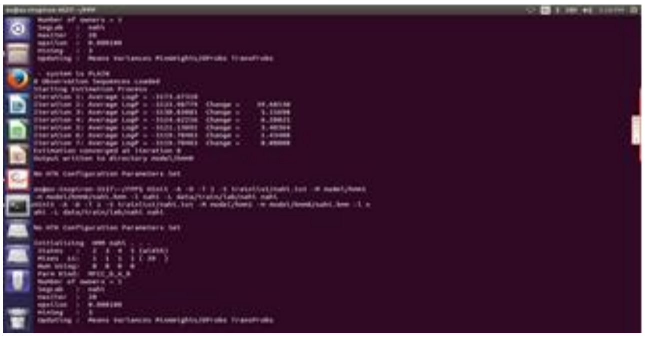

Figure 6: HInit Command

2.1.5.1 Training

HRest is the final tool in the set designed to manipulate isolated unit HMMs. Its operation is very similar to HInit except that, it expects the input HMM definition to have been initialised and it uses Baum-Welch re-estimation in place of Viterbi training. This involves finding the probability of being in each state at each time frame using the *Forward-Backward* algorithm. This probability is then used to form weighted averages for the HMM parameters. Thus, whereas Viterbi training makes a hard decision as to which state each training vector was “generated” by,. This can be helpful when estimating phone-based HMMs since there are no hard boundaries between phones in real speech and using a soft decision may give better results.

The following command line perform one re-estimation iteration with HTK tool HRest, estimating the optimal values for the HMM parameters (transition probabilities, plus mean and variance vectors of each observation function).

HRest -A -D -T 1 -S trainlist.txt -M model/hmm*i* -H vFloors -H model/hmm*i-1*/hmmfile -l label -L label_dirnameofhmm

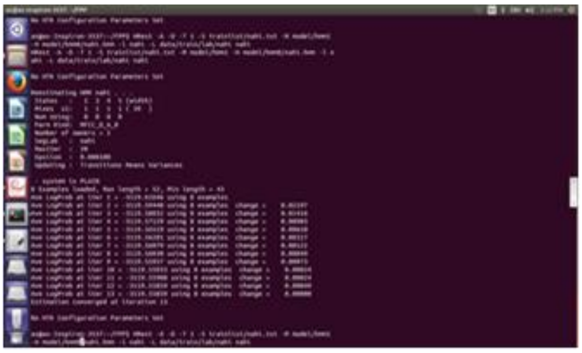

Figure 7: HMMs

2.1.6 Task Definition

Every files concerning the task definition are stored in a dedicated def/ directory.

2.1.6.1 Grammar and Dictionary

Before using our word models, we have to define the basic architecture of our recognizer (the *task grammar*). We have used the simplest one: a start silence, followed by a single word, followed by an end silence.

In HTK, the task grammar is written in a text file, according to some syntactic rules. In our case, the grammar is quite simple:

For example:

$WORD = ONE | TWO | THREE | FOUR | FIVE |
SIX | SEVEN | EIGHT | NINE | ZERO; ({
START_SIL } [$WORD] { END_SIL })

The system must off course know to which HMM corresponds each of the grammar variables ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, ZERO, START_SIL and END_SIL. This information is stored in a text file called the *task dictionary*.

2.1.6.2 Network

At this stage, our speech recognition task (Fig.17), completely defined by its **network**, its **dictionary**, and its **HMM set** (the 3 models stored in model/hmm3/), is ready for use.

The task grammar (described in file **gram.txt**) have to be compiled with tool HParse, to obtain the *task network* (written in **net.slf**):

HParse -A -D -T 1 gram.txt net.slf At this stage, our speech recognition task, is completelydefined by its **network**, its **dictionary**, and its **HMM set** (the models stored in model/hmm0/). To be sure that no mistakes were made when writing the grammar, the tool HSGen can be used to test it: **HSGen -A -D -n 10 -s net.slf dict.txt**

Where **dict.txt** is the task dictionary. Option **-n** indicates that 10 grammatically conform sentences (i.e. 10 possible recognition results) will be randomly generated and Recognitioned.

2.1.7 Recognition

An input speech signal input.sig is first transformed into a series of "acoustical vectors" (here MFCCs) with tool HCopy, in the same way as what was done with the training data (Acoustical Analysis step). The result is stored in an input.mfcc file (often called the *acoustical observation*). The input observation is then process by a Viterbi algorithm, which matches it against the recogniser's Markov models. This is done by tool HVite

HVite -A -D -T 1 -H hmmsdef.mmf -i reco.mlf -w net.slf dict.txt hmmlist.txt input.mfcc

To make the recognition user interactive we have use the script file, called final The content of the final file is

```
#!/bin/sh

echo SPEAK DIGIT:
HSLabinput.sig

HCopy -A -D -C analysis.conf -S tgt.txt
Hvite –A D-T 1 –H hmmsdef.mmf  -I reco.mlf –w net.slf
dict.txt hmmlist.txt input.mfcc
```

3. Errors and Precautions

In all the HTK commands that we have utilized the pathname is very pivotal. The path name should be correct.

During the acoustical analysis, the path specified in the source/target specification text file should be correct. Its always desirable to write all the source .sig file names and their respective target .MFCC file names in one text file and that text file should begiven as a parameter instead of writing individual file names. This file is called the script file.

4. Conclusions and Future work

The work has been done to implement speech recognition systems for Hindi language which will recognize isolated words from the predefined set of vocabulary in Hindi as well as English Text .The brief introduction to the various basic concepts of speech recognition.
The work can be further extended in many directions.

The dissertation deals with the recognition based upon small vocabulary size and isolated words. The work can also be extended from a limited vocabulary to large and from connected words to continuous speech or spontaneous speech.

- System developed for connected & continuous speech recognition for Hindi Language.
- Large vocabulary size
- Independent from pre defined vocabulary.
- User can add new set of vocabulary/word.
- Speaker independent.
- Real user interface.

References

[1] L. R. Rabiner and B. H.Juang, "Statistical Methods for the Recognition and Understanding of Speech", Encyclopedia of Language and Linguistics, 2004.

[2] P. Jancovic, J. Ming, "A Multi- Band Approach Based on the Probabilistic Union Model and Frequency Filtering Features for Robust Speech Recognition", In: Euro Speech' 01. pp. 579–582, 2001.

[3] Claudio Becchetti and LucioPrinaRicotti, "Speech Recognition Theory and C++ Implementation", John Wiley & So

[4] Alan and Ronald, (1952) 'Automatic Recognition of spoken Digits', J. Acoust. Soc.

[5] Owens, F. J. (1993) Signal Processing of Speech, Macmillan Press Ltd., London, UK.