**2018** **The Degree of Master of Computer Science** **K A D C Dilshan**

# Transcribing Number Sequences in Continuous Sinhala Speech

**K A D C Dilshan**

**2018**

# Transcribing Number Sequences in Continuous Sinhala Speech

**A dissertation submitted for the Degree of Master of Computer Science**

**K. A. D. C. Dilshan**
**University of Colombo School of Computing**
**2018**

# Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.
To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: K. A. D. C. Dilshan
Registration Number: 2015/MCS/023
Index Number: 15440233

_____

Signature:

Date:

This is to certify that this thesis is based on the work of
Mr. K. A. D. C. Dilshan
under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:
Supervisor Name: Dr. A. R. Weerasinghe

_____

Signature:

Date:

# Abstract

Human speech recognition is still far more superior than that of the global performance of state-of-the-art speech recognition systems. Rudimentary speech recognition tools have various restrictions such as limited vocabulary of words and phrases, limited language support, and limited context support. More sophisticated tools have the ability to transcribe context independent natural speech, but the research has been done on very few languages.

There are many inefficiencies in telecommunication services, such as Interactive Voice Response (IVR) navigation and call center work flows, due to unavailability of a proper online number extractor for continuous Sinhala speech. Through this research, an attempt has been made to generate number transcriptions from continuous speech for under resourced Sinhala language using readily available tools in order to optimize telecommunication services.

This research focuses on a comprehensive architectural design of the Sinhala speech decoder pipeline including Gaussian Mixture Model (GMM) based acoustic modeling and feature rich language modeling for improved performance. However, data collection and annotation tool is developed exclusively for this study. Modeling tools such as Kaldi[27], that are used in this study, have more open license and comprehensive documentation and are also backed by a large community of researchers.

As this study has a data analytic perspective and also an annotated speech corpus that is suitable for this study is not readily available, a careful effort is taken to build and evaluate a corpus of moderate number vocabulary with the voluntary participation of a friendly team.

Further, the GMM based acoustic model with various input feature transformations are evaluated using a standard and an intrinsic scoring criteria which can be found in the general Automatic Speech Recognition (ASR) literature. An algorithm is formulated to calculate the Word Error Rate (WER) produced by each model. The best model returns an accuracy level of $80.09\%$.

In conclusion, research objectives of the online number transcription tool are analyzed against the output of the study in terms of the performance of the models which are investigated throughout the research.

# Acknowledgement

Special thanks to my supervisor Dr. A. R. Weerasinghe for his invaluable guidance to make this project a success. I owe a debt of gratitude to the team who helped me in many ways to collect native Sinhala speech and annotate. It is my duty to record my thankfulness to my loving parents, my wife and daughter and my teachers.

# Table of Contents

# List of Figures

# List of Tables

# Abbreviations

**AI** Artificial Intelligence. 10

**AMFCC** Autocorrelation Mel Frequency Cepstral Coefficients. 14

**ANN** Artificial Neural Network. 15–17

**ASR** Automatic Speech Recognition. i, 1–5, 7, 13–17, 19–29, 32, 35, 37–41, 44, 58–62, 69, 73–75, 77, 78

**BFCC** Bark Frequency Cepstral Coefficient. 15

**BLSTM** Bidirectional Long Short Term Memory. 78

**CMLLR** Constrained Maximum Likelihood Linear Regression. 46, 47, 58

**CMVN** Cepstral Mean and Variance Normalization. 16, 45, 46, 57, 58

**CNMF** Convolutive Non-Negative Matrix Factorization. 24

**CNN** Convolutional Neural Network. 78

**COTS** commercial off-the-shelf. 2

**CSR** Continuous Speech Recognition. 9, 10, 20

**CWR** Connected Word Recognition. 21

**DCT** Discrete Cosine Transform. 29, 31

**DNN** Deep Neural Network. 6, 27, 47

**DTW** Dynamic Time Warping. 9, 21

**EM** Expectation Maximization. 32

**ET** Exponential Transform. 16, 17, 34, 46

**FFT** Fast Fourier Transform. 29

**fMLLR** Feature-space Maximum Likelihood Linear Regression. 34, 58, 59, 69, 71–73, 75

**FST** Finite State Transducer. 37, 38, 48, 54, 56

**G2P** Grapheme to Phoneme. 7

**GMM** Gaussian Mixture Model. 17

**GMM** Gaussian Mixture Model. i, 27, 31–34, 37, 45, 47–50, 53–57, 60, 68, 75, 76, 78

**GUI** Graphical User Interface. 76, 78

**HCNF** Hidden Conditional Neural Fields. 18

**HCRF** Hidden Conditional Random Fields. 18

**HMM** Hidden Markov Model. 9, 10, 13, 16–18, 22, 25, 31–37, 45, 47–53, 55, 56

**HSR** Human Speech Recognition. 1, 6, 74

**HTK** Hidden Markov Model Toolkit. 6, 25, 39

**IPA** International Phonetic Alphabet. 36, 39

**IVR** Interactive Voice Response. i, 2, 38, 64, 77

**IWR** Isolated Word Recognition. 21

**LDA** Linear Discriminant Analysis. 16, 68, 69, 72, 75

**LPC** Linear Predictive Coding. 15, 16

**LPCC** Linear Predictive Cepstral Coding. 16

**LVCSR** Limited Vocabulary Continuous Speech Recognition. 20, 21

**MAP** Maximum A posteriori Probabilities. 18

**MCE** Minimum classification Error. 18

**MDT** Missing Data Technique. 24

**MFCC** Mel Frequency Cepstral Coefficient. 10, 14–17, 28, 30, 31, 45, 46, 49, 57

**ML** Maximum Likelihood. 18, 59

**ML** Machine Learning. 3

**MLE** Maximum Likelihood Estimation. 55

**MLLR** Maximum Likelihood Linear Regression. 34

**MLLT** Maximum Likelihood Linear Transform. 68, 69, 72, 75

**MMI** Maximum Mutual Information. 18, 70–72

**MPE** Minimum Phone Error. 18

**P2G** Phoneme to Grapheme. 7

**PCA** Principal Component Analysis. 16

**PLP** Perceptual Linear Prediction. 14–16, 28, 45

**RASR** RWTH ASR toolkit. 38, 39

**RPLP** Revised Perceptual Linear Prediction. 15

**RTF** Real Time Factor. 23

**SAT** Speaker Adaptive Training. 34, 69–73

**SER** Sentence Error Rate. 62, 66–71

**SGMM** Subspace Gaussian Mixture Model. 31, 34, 38

**STFT** Short Time Fourrier Transform. 28

**SVM** Support Vector Machine. 11, 12

**TTS** Text to Speech. 7

**VODER** Voice Operating DEmonstratoR. 5

**VTLN** Vocal Tract Length Normalization. 16, 17, 34, 46

**WER** Word Error Rate. i, 15, 21, 22, 25, 34, 38, 57, 60, 62, 65–72, 74, 76

**WFST** Weighted Finite State Transducer. 37, 39

# Glossary

**grapheme** A grapheme is the smallest meaningful contrastive unit in a writing system. 7

**phoneme** A phoneme is the smallest contrastive unit in the sound system of a language. 7

**phonetic** analyzes the production of all human speech sounds, regardless of language. 7

**viterbi** A Viterbi decoder uses the Viterbi algorithm for decoding a bit stream that has been encoded using convolutional code or trellis code.. 11, 18

# Chapter 1: Introduction

## 1.1 Overview

Speech is the most essential and natural form available for human communication, thoughts and ideas are conveyed over speech [10]. Key elements of human speech are a collection of multilayered temporal-spectral variation that embed words, intentions, style of speaking, accent, gender, intonation, expression, state of health and emotion of the speaker, speaker identity, sex, and age [5]. Human Speech Recognition (HSR) is still far more robust, accurate and speaker independent than that of Automatic Speech Recognition (ASR). Deep understanding of human speech variability factors is required to reach a human level comparable automatic speech recognizer. A spoken language must have speech recognition capability irrespective of the usage of that language. Advancement of technology and application has identified speech based technologies as vital components to economy.

Automatic recognition of speech and its transcriptions have been active in the computer science community for the past nine decades as it primarily fills in a capability gap between humans that speak Latin and non-Latin languages across the globe and a computer that understands only binary. Automatic Speech Recognition (ASR) has displayed appreciable progress in both technology and use cases. Despite this progress, there still exists a performance gap between Human Speech Recognition (HSR) and Automatic Speech Recognition (ASR) [42]. From transcriptions of voice logs to automation of call centers are all use cases of this domain based on the solution of automatic speech recognition problem. With the advancement of the power of computing and knowledge areas like machine learning, the gap is closing in and yet the current research and developments are capable of producing promising results to address many use cases in resourced languages.

Speech recognition has evolved over the past decades along with improvements in computing power in such a way that widely spoken languages such as English, Russian can be transcribed at a success rate as high as almost 96%. Research and development, that lead to this astonishingly low error rate, include latest developments in artificial intelligence and computing infrastructure. Many studies and implementations done in the past, have guided speech recognition and

transcriptions to this advanced stage. Earlier research proposed various feature extraction, language modeling and acoustic modeling to output a transcription for a given voice input. Various models such as female voice vs. male voice, speech through a telephone and speech through a condenser microphone were studied to improve success rate. Figure 1.1 illustrates a general pipeline for the speech recognition and various extensions of that framework are in the scope of this study.

## 1.2 Problem

Unavailability of a speech recognizer, that can transcribe under resourced non-Latin Sinhala voice, precludes the achievable performance of many local systems compared to other global contemporary systems of well-versed languages such as multilingual call centers and voice command-and-control systems. Most commercial off-the-shelf (COTS) and open speech recognition software and services are unable to output transcriptions for Sinhala voice due to various geographical and technological shortcomings such as the very low percentage of global population that speaks and understands Sinhala and less contribution towards Sinhala speech recognition development even though many invaluable studies and technology research such as building a Sinhala speech corpus, have been carried out in Sinhala language research centers. It is a good sign, that Sri Lanka as a rapidly developing country, adopts state-of-the-art technologies such as wireless telecommunication systems (4G LTE), as soon as those innovations are commercialized. Therefore, many such technological systems can benefit from continuous developments on automating Sinhala speech recognition as it has a variety of applications in command and control centers, IVR systems and many other highly local interactive operations.

For example, GIC 1919[14] has a huge archive of voice logs of its Sinhala call center and all those logs are nearly inaccessible because of being unable to generate transcriptions. A speech recognition tool that can transcribe Sinhala voice to a stream of text is required to address the above use case. Even though the above use-case requires a sophisticated decoder to perform accurately, a simple speech decoder can help an IVR subscriber to browse through a service using his/her own voice by decoding Sinhala number sequences. Therefore, many organizations including both local and global establishments can make use of an online Sinhala Automatic Speech Recognition (ASR) number sequence decoding tool to optimize existing business processes such as automation of contact centers to maximize the saving of queue lengths by reducing human intervention and also to expand businesses to new dimensions such

as Google's addition of speech recognition capability to its existing Machine Learning (ML) stack as an effort to make their work publicly available.

## 1.3 Objectives

This study specifically focuses on building a simple, but robust and accurate, online decoder to stream number sequences from a continuous utterance of Sinhala number sequence at the end of this study. A brief pipeline illustrated in figure 1.1 is studied at length to arrive at the previously mentioned primary research objective, which is the Sinhala ASR tool. In depth research on various acoustic models, language models and online decoders are also in the scope of this study. An online Sinhala speech decoder is the ultimate result of this effort after evaluating the models and adapting a best fit to implement the results.

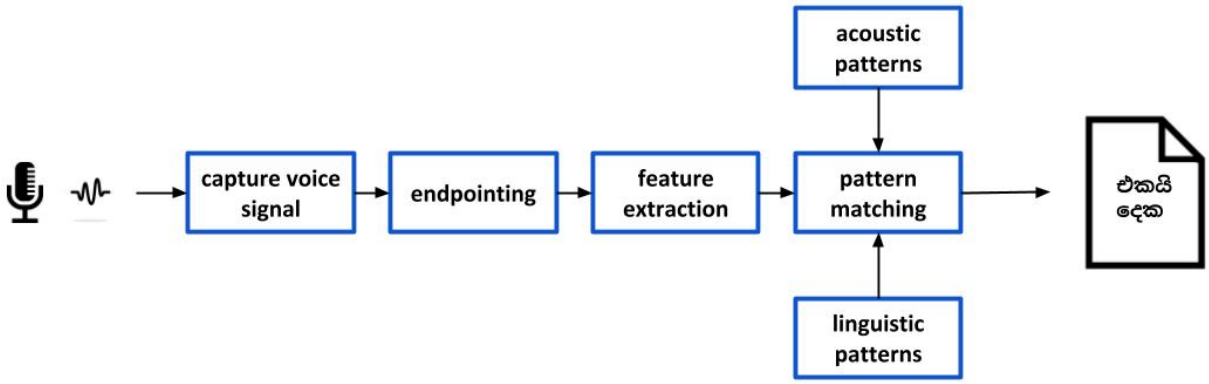


Figure 1.1: Speech recognition framework

A number of approaches have been proposed and executed to obtain a solution to the automatic speech recognition after studying human variability factors thoroughly, acoustic-phonetic approach being the early attempt to automatic speech recognition, pattern recognition approach and knowledge based approach being the state-of-the-art of speech recognition. Stochastic methods are in primary focus to solve this ASR problem for Sinhala language. ASR pipeline in 1.1 will follow a probabilistic and a discriminative approach towards the solution.

Evaluation matrices are also introduced to rate the performance of any given speech recognizer tool. Certain approaches have suitable evaluation criteria to measure their performance. It is vital to select an appropriate evaluator for performance measurement.

## 1.4 Structure of the Dissertation

A comprehensive review on the Automatic Speech Recognition (ASR) literature is presented in the chapter 2. In that chapter, a classification of ASR methodologies, ASR pipeline, implementation techniques, various readily available tools and performance evaluation are analyzed and reviewed. In chapter 3 ASR pipeline is thoroughly analyzed. Further, it describes the models on which this research is based on. Design steps of the ASR pipeline to develop an online Sinhala transcription tool is mentoned in chapter 4. That chapter consists of all the steps and guidelines to traverse the ASR process and implement. Chapter 5 critically evaluates the results returned by various feature transforms. Further, a suitable model is selected to implement the online decoder. In conclusion, chapter 6 summarizes the findings of this reserach and talks about how this research can be continued to further develop a comprehensive ASR tool.

# Chapter 2: Background

## Introduction

Homer Dudley's Voice Operating DEmonstratoR (VODER) is considered as the first and the earliest attempt to electronically synthesize a human auditory output in the history, it was designed in Bell Labs nearly nine decades ago [8]. It has been an active research area and also a concept in science fictions ever since because of the high impact of human variability factors on speech recognizer performance and now it is an important and must have tool for information and communication societies. It has evolved over ages from recognizing a set of selected sounds to state of the art automatic speech recognizers which respond to fluently spoken natural languages. Human auditory system is capable of producing voice for language symbols and it is a unique system from one speaker to another that serves a common and an indispensable purpose. Figure 2.1 illustrates how human auditory system is organized and concepts are converted to speech.

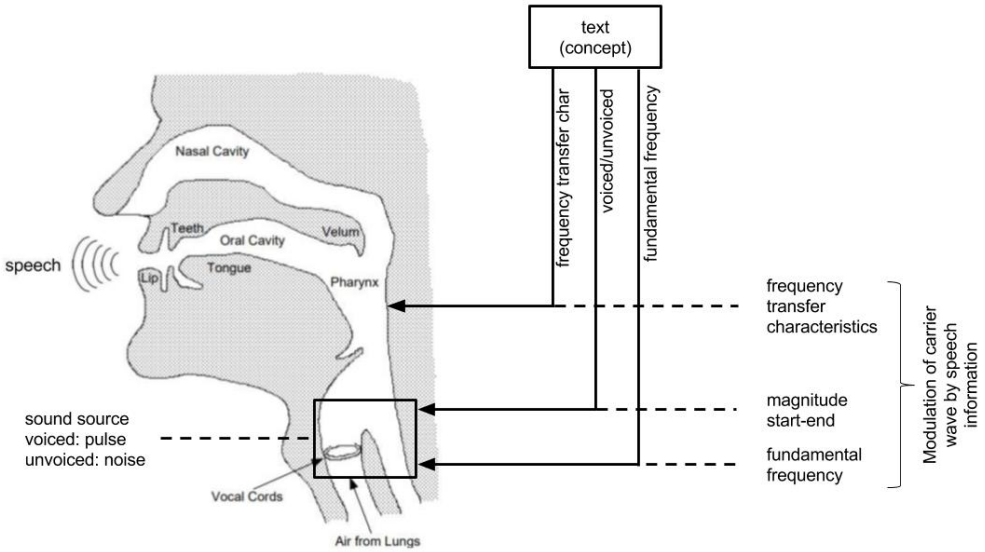


Figure 2.1: Human speech production system[44]

Various approaches and types of speech recognition tools have gradually evolved over last five to six decades. This evolution has greatly influenced the development of advanced speech synthesis tools for vastly spoken languages worldwide. The earliest acoustic-phonetic approach to Automatic Speech Recognition (ASR)[12] is viewed as sequential transformations of acoustic

micro structure of audio signal into its implicit phonetic macro structure. Languages, on which automatic speech recognizers have been researched and designed are mostly well resourced and a handful of a total of nearly 7300 existing languages. Russian, Portuguese, Chinese, Vietnamese, Japan, Spanish, Filipino, Arabic, English, Bengali, Tamil, Malayalam, Sinhala, Hindi are distinguishable among them. English is the language for which maximum research and development for speech recognition is done.

Sinhala is an under-resourced non-Latin language for which substantial research is being done towards the development of speech recognition. A speaker dependent continuous Sinhala speech recognizer was designed using readily available open source tools [21]. Speech corpora is fundamental and vital for the statistical modeling of speech recognition, a design of speech corpus for Sinhala language is also available as a result of another study [22]. But Sinhala speech recognition greatly falls behind compared to the speech recognition work done on languages such as English and Russian. This research aims at the problem of designing and developing of an online text streaming tool for Sinhala using readily available frameworks and results of past research.

## 2.1 Literature Review

A number of techniques ranging from acoustic-phonetic approach to Deep Neural Network (DNN) based acoustic modeling have been proposed and implemented thus far. However those techniques exhibit certain characteristics that affect the performance of the same and the performance has been improved from one to the other such that modern techniques, that are comprised of some form of neural network at both learning side and signal capturing side such as neural beam forming [23], are capable of reaching more than 95% of accuracy, which is as close as to the Human Speech Recognition (HSR). Modern techniques exploit the availability of large volumes of data collected by big internet companies such as Google and Facebook. Open frameworks such as Hidden Markov Model Toolkit (HTK) [43], Kaldi [28] and CMU Sphinx [17], are explored and analyzed in order to complete the study from theoretical aspects to implementation strategies.

### 2.1.1 Automatic Speech Recognition (ASR) Classification

Following strategies are in chronological order as they have been evolving over past six decades [12]. Variability of speech largely affect the performance of all approaches which eventually lead speech recognition to a pattern recognition problem.

#### 2.1.1.1 Classification Based on Acoustic Models

1. Acoustic Phonetic classification — 1964

   Acoustic-Phonetic model primarily refers to the acoustic aspects of a corresponding ensemble of phonemes or basic speech sounds. This speech recognition approach is also known as Phoneme to Grapheme (P2G) where computer indexes the grapheme values to the input phoneme or phoneme combination according to Phoneme to Grapheme (P2G) matching algorithm. Acoustic-Phonetics explores the features on the time domain and it is implemented in sequence: spectral analysis, features detection, segmentation and labelling, recognizing valid word. Components of speech, that bears the messages are to be extracted explicitly with the determination of relevant binary acoustic properties such as nasality, frication, voiced-unvoiced classification and continuous features such as formant locations, high and low frequency ratio according to this approach. This method has not provided a viable platform for commercial applications as the assumption of perfect speech fails to establish. The Sinhala language is a member of the Indo-Aryan subfamily as well as a member of a still larger family of languages known as Indo-European. Sinhala is the official language of Sri Lanka and the mother tongue of the majority of the people constituting about 74% of its population [40]. Major traits of Sinhala language related to phonetics and phonology are well examined over a longer period of time. Phonemes are basically classified against their articulators and manner of articulators, human articulatory system is illustarted in figure 2.2. Even though Text to Speech (TTS) systems are massively benefited from the principle aspects of this approach its further examination is discontinued as it does not aid the problem that is being attempted to solve.

   Many research efforts have been taken to arrive at reasonably acceptable outcomes such as 98% error rate solving Sinhala G2P problem [41]. As this conversion strategy is one to one, it is worth noting and few examples can be found in table 2.1.

| | | |
|---|---|---|
| "අම්මා" (mother) | ⟶ | /amma:/ |
| "අක්කා" (sister) | ⟶ | /akka:/ |
| "ගත්තා" (taken) | ⟶ | /gatta:/ |

Table 2.1: Example G2P conversion in Sinhala

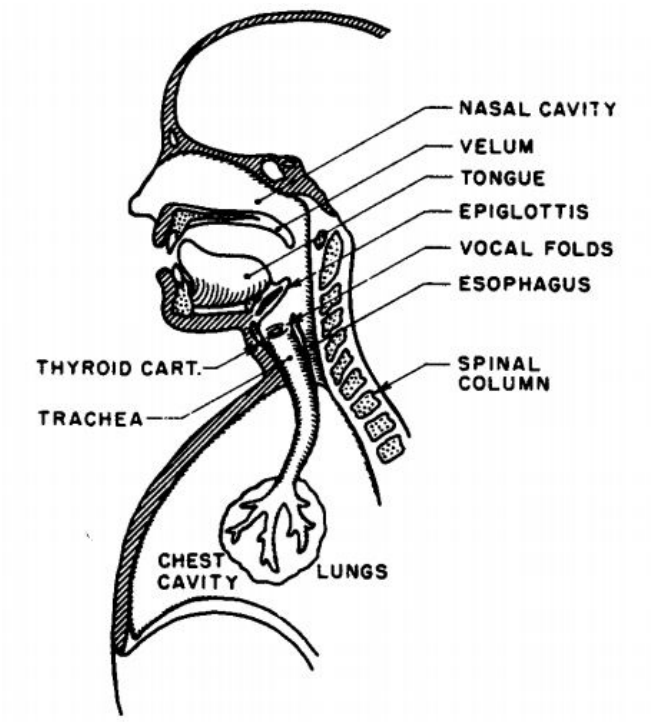


Figure 2.2: Human articulatory system[30]

2. Pattern recognition classification — 1975

   This method has become the common method for speech decoding in the last decades. This approach was first described in the paper, that applied minimum prediction residual principal to speech recognition [16], which got substantial support from the publication, Fundamentals of Speech Recognition [30] for its further acceptance among the researchers. Training patterns and comparison of patterns are the two essential steps in this method. Uniqueness of this classification is that it makes use of a well formulated mathematical framework and then constitutes a consistent representations of speech patterns for accurate pattern comparison from a set of tagged training samples via a formal training algorithm. A representation of speech pattern generally takes the form of a speech template, which is leading to methods based on templates, or a statistical model, which is leading to stochastic methods, which is applicable to a sound, a word, or a phrase equally. A direct comparison is made between the words that are spoken and required to be identified with each possible

pattern learned in the training stage for determining the identity of the unknown during pattern comparison stage of the process. There are 2 different methods that address the pattern recognition based solution.

- *Template based method*

  In this template based or example based method, a collection of templates of speech patterns are stored as reference patterns which represents the repository of candidate words. The term *template* is often used for fundamentally different two concepts, either for a single unit of speech representation with a known transcription, or for some type of an average of a number of different units of speech. The best matching pattern class is selected after an unknown spoken utterance is matched with each of these templates in the repository. Usually for each word, a template is constructed. Advantage of such a decision is that, segmentation or classification errors of less acoustic and more variable segments such as phonemes can be avoided. As a result, each word must have its own complete reference template. As vocabulary size increases, template preparation and template matching become prohibitively impractical or expensive . Watcher et al. [39] have attempted to overcome the key problems of Hidden Markov Model (HMM) framework, i.e. by discarding the information about time dependencies and over generalization problem, by applying continuous speech recognition based on templates with Dynamic Time Warping (DTW). As a result no modelling and no training procedure are required. DTW causes explosion of search space and it was kept in mind. Results obtained, were compared with the HMM based CSR. A reference CSR pipeline is shown in figure 2.3.

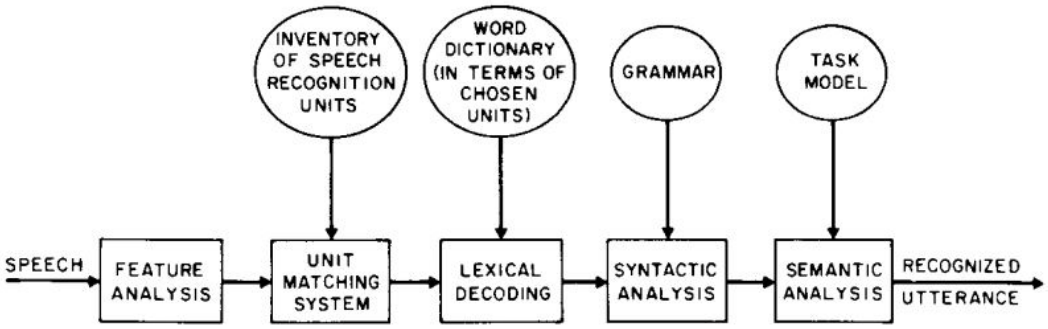


Figure 2.3: Block diagram of a continuous speech recognizer[30]

- *Stochastic method*

This method is based on the application of probabilistic models to deal with uncertain or incomplete information, such as ambiguity in sounds, speaker variability factors, contextual effects, and word sense ambiguity. Compared to template based approach, HMM modelling is more general and based on a strong mathematical foundation.

3. Connectionist classification — 1991

   The representation of knowledge and integration of knowledge sources are focus of this approach. Connectionist modelling of speech lies in the earliest development in speech recognition. Connectionist models prefer distribution of knowledge or constraints across many computing elements to an encoding in individual units, rules, or procedures. Modeling is based on rule based activity patterns in many units but not as likelihoods or stochastic functions of a single unit.

4. Knowledge based classification — 1994

   This classification focuses on applying the way a person applies intelligence in visualizing, analyzing, and characterizing speech based on a set of measured acoustic features in order to digitize the speech recognition process. The Artificial Intelligence (AI) technique refers to a hybrid of the two classifications, namely *acoustic phonetic approach* and *pattern recognition approach*. However, only template based approach and acoustic phonetic were unable to participate in human speech processing with substantial insight. Evaluation of models and knowledge based system enhancement cannot be improved as a result. Production rules of the system are created by the heuristics from empirical linguistic knowledge or from the observations from the speech spectrum in conventional knowledge based approach. The better performance of the algorithm and also in the selection of a suitable feature representation, the definition of segments of speech and the recognition algorithm design itself are the areas improved by the knowledge. For CSR, a data driven methodology was proposed by Samouelian [32], where the knowledge about the characteristics and structure of the speech signal is obtained explicitly from the repository by using inductive inference. Further, this method was capable of generating decision trees and known to have advantages of solving inter-person and intra-person speech variability problems. Very small data set of speakers causes the decoding performance of this approach to fall short. Proposed strategy has Mel Frequency Cepstral Coefficient (MFCC) based feature analysis and a schematic of the overall approach is illustrated in figure 2.4.

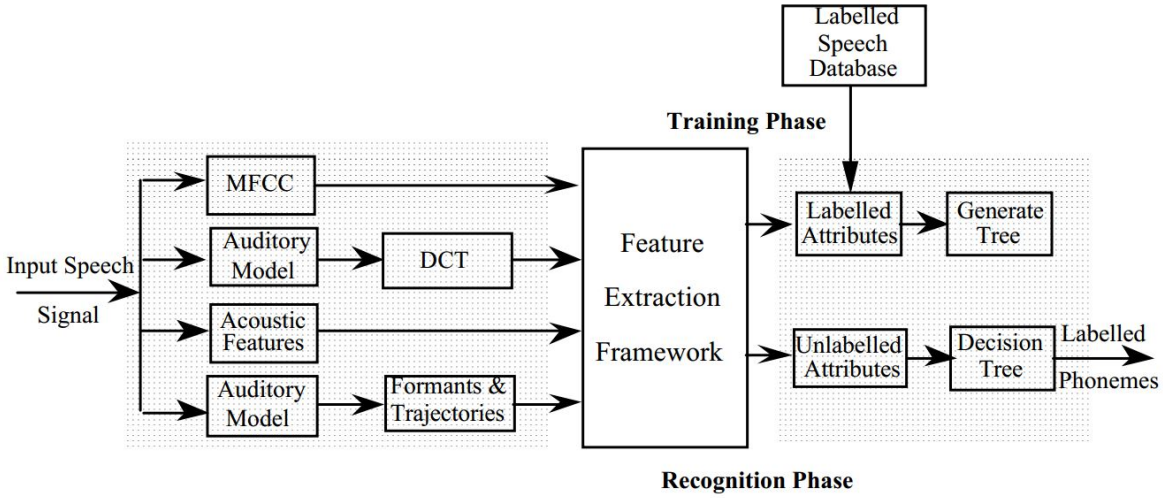


Figure 2.4: Block schematic of training and recognition strategy

5. Support Vector Machine (SVM) — 2006

Support Vector Machine (SVM) is one of the powerful supervised learning state-of-the art classifiers for pattern recognition which uses a discriminative approach [25]. Unseen patterns can be generalized by intuitive use of optimized margin between the samples and the classifier border. SVMs supports linear and nonlinear separating hyper-planes for data classification. However, this method cannot be readily employed to tasks that involve variable length data classification as SVMs can only classify fixed length data vectors. A simple solution to that limitation is the transformation of the variable length data to fixed length vectors before SVMs can be used. Linear classifier is generalized with maximum-margin fitting functions. In order for the classifier to generalize better, this fitting function provides regularization. Rather than controlling model complexity by using a small number of features, SVM controls the model complexity by controlling the vector classification dimensions of its model . This method is dimensionality independent and can utilize spaces of very large dimensions of spaces, which permits a construction of very large number of non-linear features and subsequent performing adaptive feature selection during training. SVM can employ a linear model for which vector classification dimensions is known by shifting all non-linearity to the features. Sendra et al. [25] have worked on an only SVM based continuous speech recognizer by applying SVM for making decisions at frame level and a Token Passing Model to obtain the chain of recognized words. To manage the uncertainty about the number of words in a sequence, the Token Passing Model algorithm, as shown in figure 2.5, is proposed as an extension to the Viterbi algorithm that is meant for continuous speech recognition. The results

achieved from the experiments have concluded that with a small database, recognition accuracy improves with SVMs but with the large database, same result is obtained at the expense of huge computational effort.

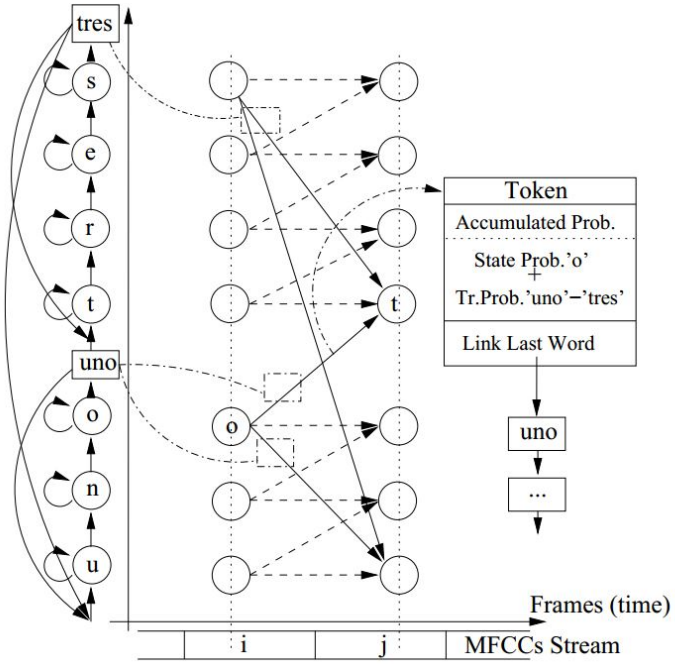


Figure 2.5: An illustration of the Token Passing Algorithm for a very simple grammar[25]

#### 2.1.1.2 Speaker Based Classification

- Speaker dependent system - A speaker dependent system is developed and targeted to operate for a single speaker. These systems are not as flexible as speaker independent or speaker adaptive systems but are comparatively accurate, easier to develop and less costly.
- Speaker independent system - A speaker independent system functions for any speaker of a given type such as an American English speaker. Further, these platforms have the most lengthy development cycles, most expensive cost structures. However, they are more flexible although the accuracy is low compared to speaker dependent systems.
- Speaker adaptive system - A speaker adaptive system is developed to learn the relevant characteristics of new speakers to operate. Its development effort lies somewhere between speaker independent and speaker dependent systems.

### 2.1.2 Automatic Speech Recognition (ASR) Pipeline

Any automatic speech recognition system involves two main stages, training stage and decoding stage. In order to map the basic speech unit such as phone, syllable to the acoustic observation, a rigorous training procedure is followed. In training phase, known speech is captured, pre processed and then enters the first stage of the pipeline that is feature extraction. Furthermore, this study is based on HMM based ASR systems. The following three stages are HMM creation, HMM training and HMM storage. The acoustic analysis of unknown audio signal marks the start of recognition phase. The recorded audio signal is converted to a sequence of acoustic feature vectors. The input observations are processed using a suitable algorithm. The speech data is compared against the HMM's networks and the utterance which is spoken is displayed. Behavior of an ASR system is that it can only recognize what it has learned during the training process. This system is able to recognize even those words that exist only in the system dictionary, which are not available in the training corpus and for which sub-word segments of the new word are known to the system. Figure 2.4 outlines a reference pipeline that illustrates the training and recognition phases.

Automatic Speech Recognition (ASR) pipeline will be discussed and analyzed to a further extent in following sections and also in chapter 3.

### 2.1.3 Functional Blocks of Automatic Speech Recognition (ASR) Pipeline

This section describes the main functional components of an Automatic Speech Recognition (ASR) pipeline. Following modules can be identified as the main modules of any ASR strategy described in section 2.1.1,

1. Capturing speech signal
2. Feature extraction
3. Acoustic modeling
4. Language and lexical modeling
5. Recognition

Capturing voice signals and feature extraction are common to both training and recognition phases. Figure 2.6 illustrates a detailed ASR procedure.

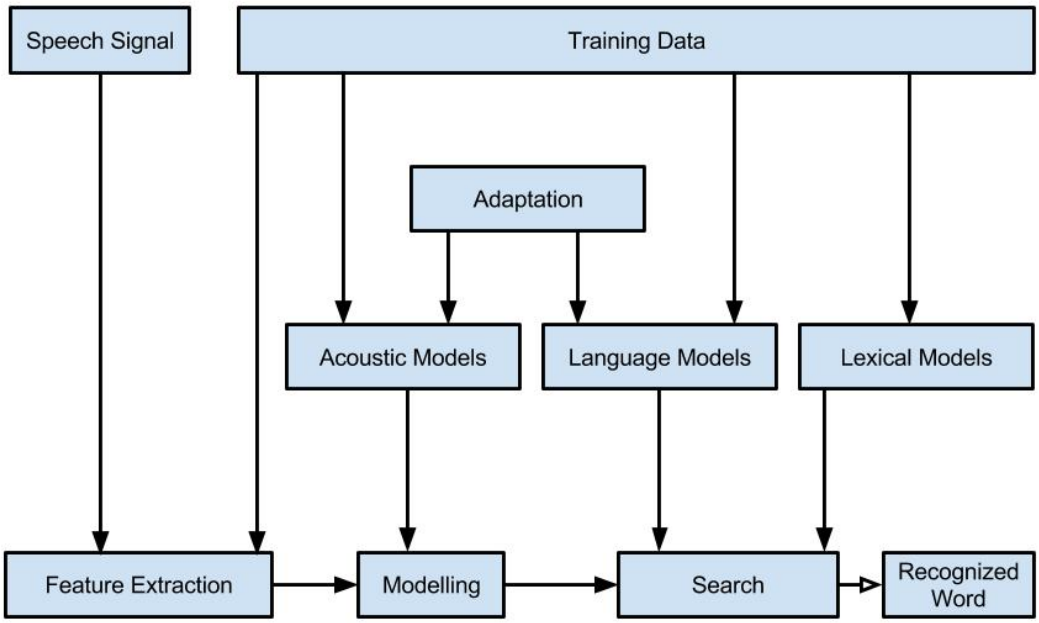


Figure 2.6: An illustration of the detailed ASR block diagram

Model adaptation is meant for minimizing the external dependencies on speakers' voice, acoustic environment, microphones and transmission channel, and to improve the generalization capability of the system. For example acoustic model requires adaptive noise cancellation to generalize the pipeline.

#### 2.1.3.1 Feature Extraction

Feature extraction is the second functional block of Automatic Speech Recognition (ASR) pipeline, after the pre-processing stage. In order to produce a classification of sounds. this block is capable of extracting descriptive features from the windowed and enhanced speech signal. As the raw audio stream contains additional information in addition to the linguistic data and also has higher dimensionality, feature extraction is required. Therefore, both characteristics of the raw audio signal can degrade the performance of the classification of phonemes and result in a high word error rate. As a remedy, the feature extraction technique produces feature vectors with reduced dimensionality to classify sounds. Following list includes many feature extraction techniques for speech recognition systems[12],

- Perceptual Linear Prediction (PLP)
- Mel Frequency Cepstral Coefficient (MFCC)
- Autocorrelation Mel Frequency Cepstral Coefficients (AMFCC)

- $\Delta$MFCC
- Linear Predictive Coding (LPC)
- MF-PLP
- Bark Frequency Cepstral Coefficient (BFCC)
- Revised Perceptual Linear Prediction (RPLP)

A feature vector in general should bring the important information regarding the specific task to the foreground and hide all other unnecessary information ideally. The information about this message needs to be emphasized, as the goal of automatic speech recognition is to transcribe the linguistic message. The speaker variability factors, the state of the background and the recording gear should be highly attenuated because these independent noise factors do not contain any information about the spoken utterance. Further, this non linguistic information would introduce additional variability and also severly impact on the separability of the phone classes unless taken care of. In order to reduce the dimensionality of the feature vector to obtain faster computation time and the less number of training samples, the feature extraction process should lower the dimensionality[30].

Frequency domain features are obtained using discrete fourier transform or wavelet transform, during the feature extraction process. Wavelet transform provides better time-frequency localization to detect sudden changes in audio signals. Wavelet transform uses wavelets in place of sine waves of different frequencies and provides different resolution for each scale. Other aspects of ASR pipeline, that hold very great influence on implementation of ASR are, Acoustic Model, Lexical Model and Language Model.

As a different approach, an Artificial Neural Network (ANN) can post process low-level features to generate more robust features for ASR.

All of the previously described combinations of low level features are used by many modern feature extraction components for ASR system. The feature extraction process of the RWTH Aachen speech recognition system [35] produced the best Word Error Rate (WER) for English and German languages. Furthermore, it uses successive low level features as a single feature vector, aggregating different feature streams and post processing low level features with an ANN and the tandem architecture with lowered hindering dimensions. The feature extraction component of the RWTH Aachen speech recognition system uses MFCC and PLP as low level

features. A collection of 20 band-pass filters computes the first 16 Cepstral coefficients of MFCCs, also known as Cepstral Mean and Variance Normalization (CMVN). Nine MFCC feature vectors within a sliding window are stacked and using a Linear Discriminant Analysis (LDA), the feature dimensions are reduced to 45. Extraction and processing of the extra feature stream of PLP features is identical to the MFCC features. Finally a hierarchical bottleneck feed forward ANN computes the phone posterior probabilities. Principal Component Analysis (PCA) decorrelates the probabilistic bottleneck features and those features are added to the feature vector.

Mel Frequency Cepstral Coefficient (MFCC) is widely used in both automatic speech recognition and automatic speaker recognition as the feature. Davis and Mermelstein[7] presented these 2 feature types. However, MFCC and PLP have been state-of-the-art feature types ever since. Linear Predictive Coding (LPC) and Linear Predictive Cepstral Coding (LPCC) were the popular type of feature for ASR with HMM classifiers specifically prior to the introduction of MFCC.

There are implicit algorithms, that can reduce the inter speaker variations, that are resulting from vocal tract length. One of the considerable source of inter-speaker variations [45] is the variation in length and/or shape of vocal tract. Vocal Tract Length Normalization (VTLN) is a computationally expensive operation and it can be made effective through adaptation techniques. VTLN significantly improves the performance of ASR systems by lowering the inter-speaker variations. Figure 2.7 illustrates the application of VTLN on MFCC pipeline.

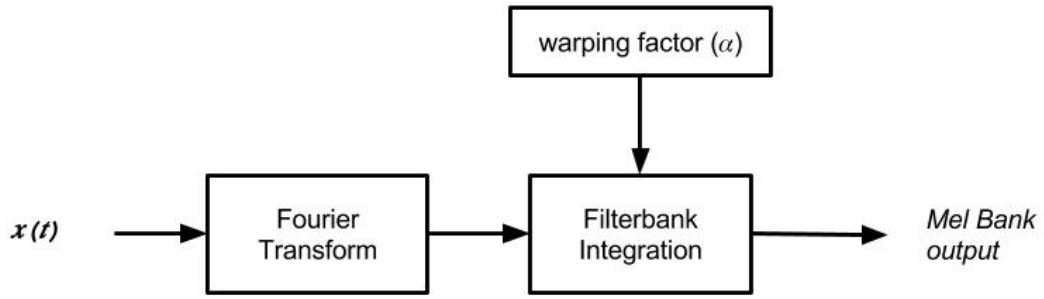


Figure 2.7: Speaker-specific Mel scale VTLN

In order to normalize the formant positions based on per-speaker, frequency axis is scaled. Due to the contribution made by gender and other factors to affect the length of the vocal tract, there can be a 20% variation between speakers. Currently, the most commonly used approach to VTLN operates by repeating feature extraction multiple times as much as 20 times for example, for a discrete set of warping factors, and selecting the warp that provides the highest likelihood features with respect to a simple model.

Exponential Transform (ET) is another popular technique which integrates aspects of VTLN, regression and transform techniques into a single transform with components that are collectively learned. Being able to adapt effectively small amounts of speaker specific data and the requirement of a very small number of speaker specific parameters are the main advantages of this approach. Our formulation shares some characteristics of VTLN, and is intended as a substitute for VTLN. A single parameter specific to the speaker which is analogous to a VTLN warp factor controls the key part of the transform. However, there are non speaker specific parameters of this transform that are learned from data. Furthermore, the axis along which male and female speakers differ is learned automatically. The Exponential Transform having no explicit frequency warping based notion, makes it applicable in principle to non parametric features such as those derived from neural nets, or when the key axes may not be gender specific as male or female. However, standard MFCC features promise to give better results experimentally than traditional VTLN.

#### 2.1.3.2 Acoustic Model

Acoustic model is the most critical component of the ASR pipeline and it contributes to the computational overhead and system performance mostly. The output features of the audio signals are mapped by the acoustic model with the ground truth sentence of the hypothesis phone sequence. The *acoustic model* is constructed to identify the spoken phoneme string. Creation of acoustic models involves the use of captured audio of speech and their corresponding transcripts and then developing them into a statistical representation of sounds which make up graphemes. Aggarwal and Dave[2] have attempted to provide a comprehensive general overview to acoustic modelling in the context of ASR. Figure 2.8 illustrates the proposed statistical modeling of the ASR. Currently Gaussian Mixture Models (GMMs) are the dominant and popular technique for modelling the emission distribution of HMMs for ASR.

However, two major drawbacks hinder the HMM performance. Given a state, strong independence assumption in HMM states that frames are independent. As a result of that shortcoming, it is unable to deal with a feature that straddles over several frames. Various feature techniques have been developed to counter the straddling phenomena such as delta coefficient, segment statistics and modulation spectrum. Aggarwal and Dave [1] have reviewed a variety of modifications, extensions and improvements adopted for the HMM based acoustic models in the form of refinements such as variable duration models, discriminative techniques, connectionist strategy

(HMM and ANN) to overcome the limitations of conventional HMM and improvements such as margin based methods, wavelets and dual stream approach. Ostendorf et al.[24] also introduced segment models to overcome this weakness of HMM.

Secondly, HMM fails to discriminate sequences as it is a generative model. This weakness has become dominant due to the maximization of Maximum Likelihood (ML) instead of Maximum A posteriori Probabilities (MAP). Maximum Mutual Information (MMI), Minimum classification Error (MCE), Minimum Phone Error (MPE) algorithms have been developed for training as some alternative procedures. Above two major drawbacks of HMM can be overcome using Hidden Conditional Random Fields (HCRF) technique while ensuring the integrity of the merits of HMM such as efficient algorithms including forward backward algorithm and Viterbi decoding. Neglecting non linearity among features which would be crucial for speech recognition is a drawback of HCRF. As number of features increases, HCRF approach becomes more and more difficult. Yasuhisa et al. [9] have introduced gating functions as a modified Hidden Conditional Neural Fields (HCNF) which can easily consider non linearity between feature. However, It has also been found that HCNF can be trained without any initial model and any kinds of features can be Incorporated. It was found that results of Hidden Conditional Random Fields (HCRF)s were inferior to the results of HMMs because of the implementation of HCRF without using mixtures. HCNFs have clearly outperformed HCRFs and the result showed the effectiveness of incorporating the gate function into HCRF. This result was superior to the results of HMMs and comparable with the best ones of previous results in mono-phone setting. Mohamed et al [20] have proposed a technique in which Gaussian mixture models have been replaced by multi-layer feed-forward neural networks where multiple layers of features have been generatively pre-trained. It has been the very first application of neural networks to acoustic modelling. This approach provides a hierarchical framework where each layer is designed to capture a set of distinctive feature landmarks. A specialized acoustic representation is constructed for each feature in which the corresponding feature is easy to detect. Discriminative fine tuning was performed using back propagation to slightly adjust the features so as to make them better at predicting a probability distribution over the states of mono-phone HMMs.

#### 2.1.3.3 Lexical Model

Lexicon is a pronunciation dictionary which compiles the pronunciation of each word in a given language. Through lexical model, various combinations of phones are defined to give valid

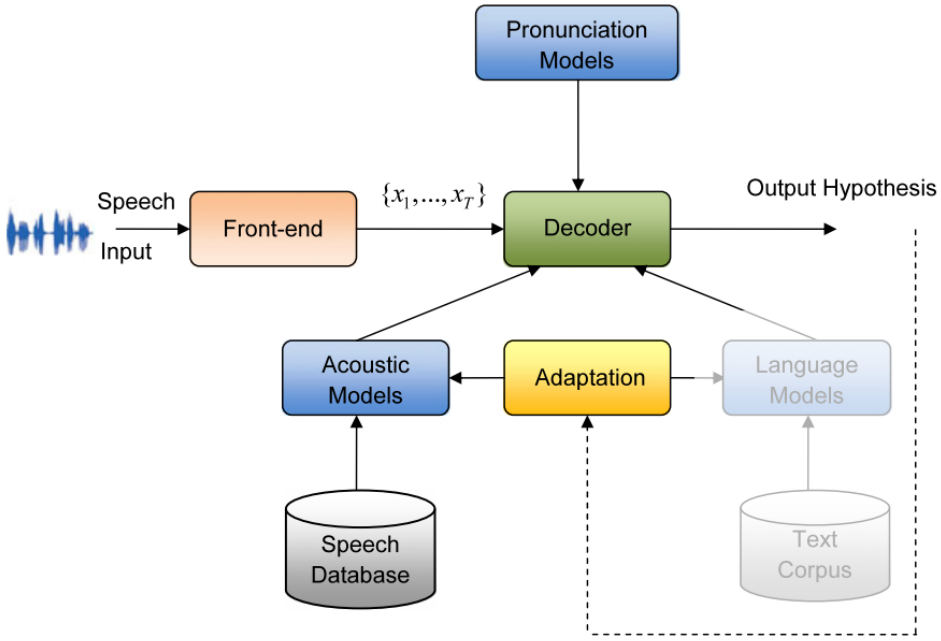


Figure 2.8: An illustration of the statistical ASR block diagram[2]

words for the recognition. Neural networks have enabled the speech recognition of non native speakers through development of lexical model.

#### 2.1.3.4 Language Model

Language model is the single largest known component to comprise of millions of utterances of words, consisting of a large collection of parameters and compiled for deducing the word level relationships in a sequence of words with the aid of a lexicon. $n-gram$ language models help ASR tools to guide the search for correct word sequence by computing the likelihood of the word at hand on the basis of the preceding words. The probability of a word sequence $W$ is computed as:

$$P(W) = P(w_1 w_2 w_3 ... w_{m-1} w_m) = P(w_1) P(w_2|w_1) P(w_3|w_2 w_1) ... P(w_m|w_1 w_2 ... w_{m-1})$$

During the construction of $n-gram$ language models for large vocabulary speech recognizers, two problems are being faced. Large amount of training data generally leads to large models for real applications. Second is the sparseness problem, which is being faced during the training of domain specific models. Language models are non-deterministic as well as cyclic. Those 2 features complicate the compression of representation. A technique, that does not affect the preformance in terms of access time, has been proposed by Sorensen and Allauzen [34].

### 2.1.4 Variants of ASR

Type of speech or speaking mode, dependence on speaker, size of vocabulary and bandwidth are the different basis on which researchers have worked. Among them, speaking mode is one of the main criteria which lead to the evolution of following ASRs. Rest of the factors have been dealt along with.

#### 2.1.4.1 Continuous Speech Recognition (CSR)

Continuous speech recognition deals with the speech where words are connected together instead of being separated by pauses. As a result of variability introduced by the boundaries of words, effects such as coarticulation, rate of speech and production possibility of surrounding phonemes largely impact the performance of Continuous Speech Recognition systems. It has been known that there are 3 bases to speech recognition in terms of the selection of sub-word segments, word based, syllable based and phone based.

- Word Based

  Word or lexical unit is well defined in terms of acoustics and the variations in acoustics occurs in the boundary, beginning and the end of the word generally. Parameter sharing is not possible because there is a problem with the choice of word unit having each word trained individually. This requires the setting up a very large set of training data and an increasing memory requirement. Word models have been employed to build limited vocabulary ASR systems successfully in spite of the aforementioned issue. However, it is not practically used for Limited Vocabulary Continuous Speech Recognition (LVCSR) systems.

- Syllable Based

  Syllable, larger unit of sub word, is set as a acoustic modelling primitive. Nucleus has no dependencies in context unlike the other constituents onset and coda. Contextual effect of coda of current syllable with the onset of the following syllable is the issue to be dealt further. There are languages where syllable is being used as a sub word model as pronunciation mainly depends on the syllable. English has a fuzzy syllabication whereas in phonetic languages, syllables are formed by strong linguistic rules. Thangarajan et al. [33] have developed a Tamil continuous speech recognizer using syllable as a sub word

unit for building acoustic model. Each word was segmented into its prosodic constituent syllables to create a syllable based lexicon through an algorithm. A fairly good recognition accuracy was achieved but about 10% increase in WER has been attributed to a set of large number of syllables to be modelled with the limited training set available. Abushariah et al.

- Phone Based

  Phone based model has the ability to solve this problem by incorporating the sharing of phoneme parameters and thereby saving the computing resources. But the phones are highly context dependent and their aspiration varies too across the word. Over-generalization by phone models in comparison to no generalization by word models is a strange observation for the researchers. Phone-in-context and tri-phone have provided solution to over-generalization by adding right and left contexts. Tri-phone sub-word models have provided better acoustic modelling and considerable reduction in word error rate for LVCSR systems. Thangarajan et al. [37] have developed a triphone medium vocabulary and a word based small vocabulary continuous speech recognizer for Tamil language. Triphone sub word segment model requires a lot of memory during the computation and it is known as a limitation of this approach.

#### 2.1.4.2 Connected Word Recognition (CWR)

Connected Word Recognition (CWR) of spoken signal systems consist of words are separated by silent pauses. Connected Word Recognition involves fluent speech speech strings extracted from small to moderate size vocabulary such as number sequences, spelled letter sequences, combination of alphanumeric. Similar to Isolated Word Recognition (IWR), this ASR type also has a property that the basic unit of speech recognition is to the extent of word or phrase. Rabiner et al. [30] designed three algorithms Connected Word Recognition; two level dynamic programming approach, level building approach and one pass approach. Furthermore, three algorithms provide the identical best matching string with the identical matching score for CWR. However, they differ in following metrics; computational efficiency, storage requirement and realization feasibility in real time hardware. Speaker dependent connected digits recognition system have been developed by Garg et al. [15] by applying technique of unconstrained Dynamic Time Warping (DTW) where each digit is recognized by calculating the distance with respect comparison of input spoken number to the stored template.

### 2.1.5 Implementation of an Automatic Speech Recognition (ASR) Tool

ASR development pipeline has the following general steps,

1. All speech data is recorded using a uni-directional noiseless microphone.
2. Employ a relevant feature extraction method for signal parameters.
3. Perform acoustic analysis to convert training waveforms into a sequence of vectors of coefficients.
4. Define a prototype HMM for elements in the task vocabulary.
5. Initialize each HMM and train with training data.
6. Define grammar or language model for the speech recognizer.
7. Recognize unknown input speech signal during the execution.

Advent of new techniques and modification from time to time in different acoustic structure of languages, there have been changes in one or more nodes applied by the researchers in order to improve existing techniques.

### 2.1.6 Performance of Automatic Speech Recognition (ASR)

Accuracy and speed are the two main performance criterion of an ASR system.

#### 2.1.6.1 Accuracy

1. Word Error Rate (WER)

   The WER is calculated by comparing the test set to the hypothesized document. However, it is done by counting the number of substitutions ($S$), deletions ($D$), and insertions ($I$) and dividing the total by the word count in the test set.

   E.g. Reference one | **fourteen** | six
   ASR one | **forty** | six

   The substitution error of the word *forty* for the word *fourteen* would be counted as one substitution error, as opposed to one deletion error of (*fourteen*) and one insertion error (*forty*).

#### 2.1.6.2 Speed

Real Time Factor (RTF) is a parameter to score ASR rate. If it takes time $P$ to process an input of duration $I$, the RTF is defined as,

$$RTF = \frac{P}{I}$$

E.g. RTF would be 2 if it the computation time taken to decode a speech stream of 3 hours long is 6 hours. $RTF \leq 1$ implies real time processing.

### 2.1.7 Robust Automatic Speech Recognition (ASR)

All variability sources are to be addressed at a priority to achieve the real robustness in ASR. Following are inevitable sources of critical environmental conditions in real world applications and ASR suffers recognition performance degradation accordingly.

1. Prosodic and phonetic context
2. Accent and Dialect
3. Variability and distortions of
4. Vocabulary Size and domain
5. Transmission channel variability and distortions
6. Adverse speaking conditions
7. Noisy acoustic environment
8. Pronunciation
9. Speaking behaviour

#### 2.1.7.1 ASR Compensation Efforts

Few researchers have made following attempts to deal with the variability sources listed above.

1. It is always a challenging task to identify the noise sources and suppress it from the audio stream in a multiple source attenuated environment due to the stochastic nature of

background noise. In order to model noise variation reliably, a substantial amount of training speech data is required. Computationally optimized Convolutive Non-Negative Matrix Factorization (CNMF) enabled researchers to attend to the problem of identifying the speech stream of target speaker captured under typical ambient noise conditions. Ter approach to problem was from enhancement aspect of the signal. This strategy is known to be extremely helpful in situations where long hours of recorded data of environments aid the learning of global noise floors.

2. Inevitable sources; Acoustics of environment, per speaker variability, background noise and transducer, corrupt the audio streams in reality. The acoustic features observed being unable to no longer match the acoustic models causes degraded performance. By integrating Missing Data Technique (MDT)s with three conventional methods; multiple condition training, dereverberation and spectral subtraction, many researchers have been able to enhance the performance of ASR

3. There is variability in the spoken sequences by different speakers due to varibaility factors such as different geographical boundaries, social background, age, gender, occupation etc. Researchers have been able to model the same isolated utterances spoken by different speakers by using only four features, first three formant frequencies and zero crossing rate of the signals. Multiple layer neural networks for the classification were used in the process. Better results were achieved compared to wavelet based features.

4. Variation in pronunciation majorly causes the degradation of performance for a variety of ASR systems. Pronunciations in a speech recognizer is modelled by a lexicon taken ahead by a set of rewrite conditional rules to apply for variation in phonology. Even a well defined pronouncing dictionary fails to support each and every variation in human pronunciation some time or most of the times. Without any dictionary modification, Gaussian density sharing across phonetic models and decision tree have stated true to be efficient. Researchers were able to address this issue using an alternative modified relabel approach to obtain a pronunciation based on rules and to develop real acoustic models of phonetics. Initial transcriptions phonetics and training approaches were combined and tried. Phonemic transcriptions, that initialize the pronunciation variability system, that is generated by relabel approach produced acceptable results.

5. A linear model transformation and a frequency warping based speaker normalization have been able to suppress the variability sources such as inter speaker, channel, environmental

and microphone variability. There was a considerable improvement in the WER reduction resulting from combined frequency warping and spectral shaping.

### 2.1.8 Tools and Open Frameworks for ASR

**Audacity**: It is free, open source software available with latest version of 1.3.14(Beta) which can run on wide range of OS platforms and meant for recording and editing sounds.

**HTK** — Open source Hidden Markov Model Toolkit (HTK) is written completely in ANSI C. The primary application of it, is to build Hidden Markov Models (HMMs) and manipulate. This tool kit supports characters in 8-bit ASCII standard code as it has been designed to recognize English originally. However, there are languages which do not follow this format. Vietnamese language as an example, is stored in utf-8 format support HTK.

**CMU Sphinx** — CMUSphinx 4 is the latest version of Sphinx series of speech recognizer tools, written completely in Java programming language. It provides a more high level framework for research in speech recognition.

**Kaldi** — Kaldi is a toolkit for speech recognition written in C++ and licensed under the Apache License v2.0.

**Microphones** — They are being used by researchers for recording speech database. Some unidirectional and noiseless microphones are fabricated by Sony and I-ball.

### 2.1.9 Areas of Application

Growing contribution and interest of researchers in the development of new techniques for different stages of automatic speech recognition pipeline, have lead to the applications of ASRs in different fields such as,

1. Command and control centers
    - Automated call-type recognition
    - Call distribution by voice commands
    - Directory listing retrieval
    - Voice repertory dialer

2. Call re-routing
3. Data entry automation
4. Speech to text processing
5. Home automation e.g. PAs like Google Home, Amazon Alexa
6. Vehicle navigation system
7. Transcription of speech to short messages

## Summary

Researchers, working on the very promising and challenging field of automatic speech recognition, are collectively heading towards the ultimate goal that is natural conversation between human beings and machines, are applying the knowledge from areas of neural networks, psycho-acoustics, linguistics, speech perception, artificial intelligence, acoustic-phonetics etc.. The challenges to the recognition performance of ASR are being provided concrete solutions so that the gap between recognition capability of machine and that of a human being can be reduced to maximum extent. An attempt has been made through this research to give a comprehensive survey and growth of automatic speech recognition over the last six decades through the never ending efforts of researchers in countries like China, Russian, Portuguese, Spain, Saudi Arab, Vietnam, Japan, UK, Sri-Lanka, Philippines, Algeria and India.

# Chapter 3: Analysis of the ASR Pipeline

## Introduction

This analysis mainly targets on the conventional modeling of an Automatic Speech Recognition (ASR) whereas state-of-the-art is DNN based ASR. Regardless of the techniques, almost all speech recognition systems follow the same basic structure. Figure 3.1 illustrates the end-to-end ASR process. It outlines all important steps from capturing audio signal to word decoding to meaning of the spoken sentence. In this section, the analysis is done up to the word(digit) level along the pipeline. Each block of the diagram will be analyzed in terms of the techniques and the algorithms and adapted to arrive at the online Sinhala speech decoder. As Kaldi[26] framework is used to implement the working decoder, many aspects of the blocks are related to the conventional model but the analysis extends beyond Gaussian Mixture Model (GMM) to discriminative criteria. Kaldi[26] is written in such a way that it is extensible to new kinds of models.

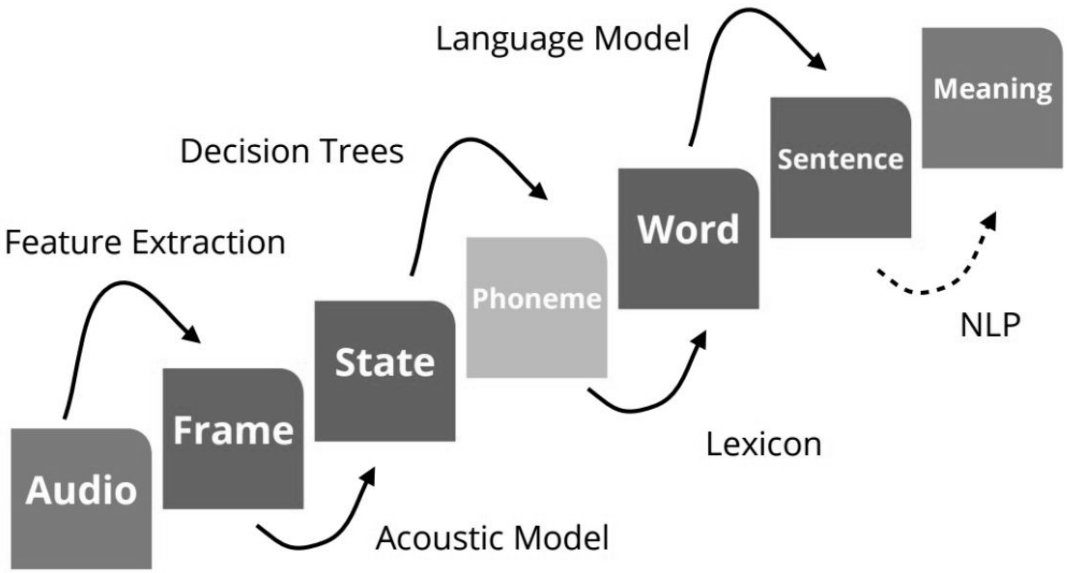


Figure 3.1: An illustration of the ASR pipeline

## 3.1 Feature Extraction

Feature extraction is the first step in any ASR system , that is identifying the attributes of the audio signal, that are better fits for identifying the linguistic description and disregarding all the other irrelevant components which carries information like background noise, emotion etc.

The most important thing about speech is that shape of the vocal tract, teeth, tongue and other organs filter the human generated sound. These organ shapes determines the sounds that come out. An accurate phonemic representation can be produced when an accurate mapping from those shapes to abstractions can be done. The shape of the vocal tract can be recovered in the envelope of Short Time Fourrier Transform. State-of-the-art feature extraction techniques are MFCC and PLP. In general, job of those techniques are to accurately represent this envelope. However, after optimal setting both parameterization techniques provided almost comparable results[29]. Comparable statistics in table 3.1 was achieved by a group of researchers who implemented an ASR system for meeting transcriptions[13]. Their research findings tabulate the WER% results for various corpora and it can be seen, that the performance of MFCC and PLP are comparable. Kaldi framework has constructs for both techniques mentioned above but MFCC is selected to continue the analysis.

| Features | Adapt/Normalize | TOT | CMU | EDI | NIST | TNO | VT |
|---|---|---|---|---|---|---|---|
| MFCC | — | 39.7 | 39.9 | 37.0 | 34.2 | 38.9 | 45.8 |
| MFCC | VTLN HLDA | 34.2 | 34.2 | 32.6 | 29.9 | 32.0 | 41.0 |
| PLP | — | 39.0 | 39.0 | 35.4 | 33.7 | 40.3 | 45.6 |
| PLP | VTLN HLDA | 31.8 | 31.9 | 29.0 | 29.1 | 30.0 | 37.9 |

Table 3.1: MFCC and PLP WER comparison[13]

### 3.1.1 Mel Frequency Cepstral Coefficient (MFCC) Parameterization

Mel Frequency Cepstral Coefficients (MFCCs) features are widely used in automatic speech and automatic speaker recognition. Davis and Mermelstein[7] introduced both techniques, and ever since they have been the state-of-the-art. This analysis will cover the main aspects of MFCC, on why they return a good feature for ASR, and also on their implementation.

A high level introduction to the steps of implementation are itemized below. Each step is further investigated in depth with a more detailed description of how to calculate MFCCs.

1. Frame the signal into short frames.
2. For each frame calculate the periodogram estimate of the power spectrum.
3. Apply the Mel filter-bank to the power spectra, sum the energy in each filter.
4. Take the logarithm of all filterbank energies.
5. Take the DCT of the log filterbank energies.
6. Keep DCT coefficients $2 - 13$, discard the rest.

There are a few more commonly done things, such as appending the frame energy to each feature vector. Usually, Delta and Delta-Delta features are also appended. Commonly final features are also applied with liftering[30].

In the following discussion, each step is analytically investigated.

A speech signal is varying constantly. In order to simplify the step, it is assumed that speech signal does not vary much statistically although it is obvious that the signal samples are varying regularly on even short scale of time. This is why signals are framed into $20 - 40ms$ frames. There will not be sufficient samples to get a reliable spectral estimate, if it is much shorter. However, if it is longer the signal varies a lot throughout the frame.

Computation of the Fourier transform or power spectrum of each frame is the next step. This step is inspired by the human organ resides inside the ear, namely cochlea, which vibrates depending on the incoming sound frequency at different places. Different nerves gets activated and informs the brain that certain frequencies are present depending on the point in the cochlea that vibrates with wobbling small hairs. Periodogram estimate does an equivalent by identifying which frequencies are available in the frame. This is known as the periodogram estimate of the power spectrum. Absolute value of the complex Fourier transform is taken, and the result is squared. Generally a 512 point FFT is performed and kept the first 257 coefficients. Figure 3.2 is a sample power spectrum of the speech signal එක—තුන—පහ. Onset, fundamental frequency and harmonics are clearly visible in the 512 point FFT.

The spectral estimate of the periodogram still consists of a lot of unnecessary information for ASR. The cochlea can not understand the difference between two closely separated frequencies in particular. As the frequencies increase, this effect surfaces more. Clumps of periodogram bins are taken and summed up for this reason to understand the amount of energy present in many frequency regions. Mel filter-bank is assigned to this task, the first filter is very narrow

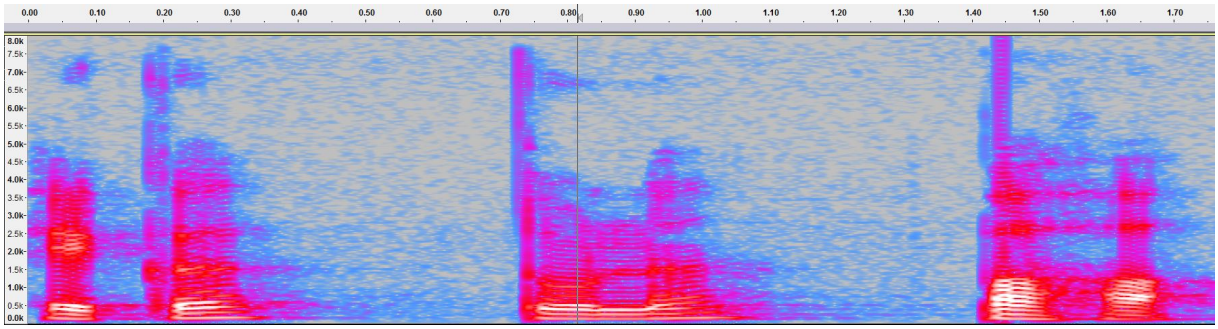

Figure 3.2: Spectrum of speech signal of එක—තුන—පහ

and gives an indication of how much energy exists near $0Hz$. Filters get wider as frequencies go higher and variations get less concern. A rough estimate on the amount energy occurs at each place is the only interested spot. The Mel scale defines how exactly to determine the space of filter banks and how to widen them. Following equation maps frequencies to Mel scale frequencies[30].

$$M(f) = 1125ln\left(1 + \frac{f}{700}\right)$$

Figure 3.3 is the 26 filter Mel-frequency filter bank.

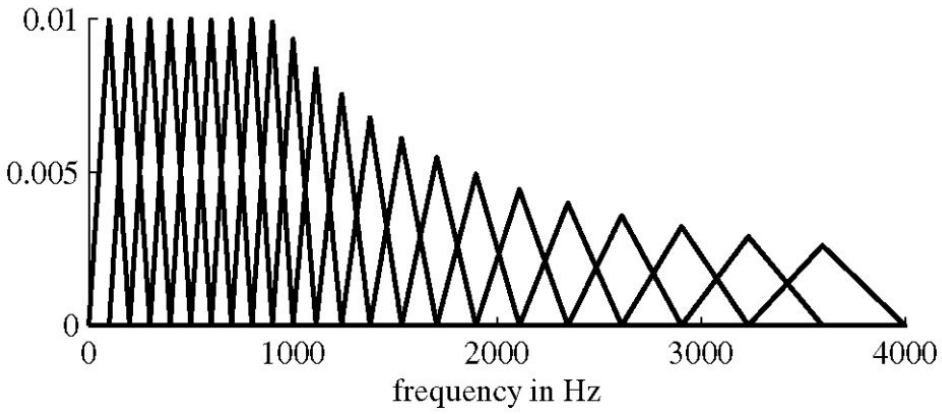


Figure 3.3: Weighting functions for Mel-frequency filter bank[31]

Once filter bank energies are calculated, logarithm of those energies are taken. This is also motivated by human hearing, we do not hear loudness on a linear scale. In order to double the received volume of an audio stream, 8 times as much energy must be put into it. If the sound is highly audible to begin with, large changes in energy may not significantly sound that different. This compression technique makes MFCC features match what humans actually hear more closely. The logarithm is computed instead of cubic root as the logarithm enables the use of Cepstral mean subtraction, and also a channel normalization technique.

The Discrete Cosine Transform (DCT) of the log filter bank energies are computed as the final step. Following are the 2 main reasons to why this is done. Since all filter-banks are overlapping with neighboring bins, their energies are quite correlated with each other. The energies are decorrelated by DCT which means features in an HMM classifier for example are modeled by the diagonal covariance matrices. However, only 12 of the 26 DCT coefficients are allowed. The reason for that decision is that the higher DCT coefficients produce rapid changes in the filter bank energies and it can be seen that these sudden changes actually cause decoder performance degradation. However, the improvement that can be obtained by dropping those coefficients is small.

Kaldi framework has scripts to generate MFCC features of a speech file. In the design section, those scripts are outlined and explained.

## 3.2 Acoustic Model

Fundamental approach behind this section is pattern recognition based approach described in the chapter 2. This analysis mainly focuses on Gaussian Mixture Model (GMM) based acoustic models and various techniques are applied to obtain improved results. Kaldi framework has a number of acoustic models and configurations to adapt to any language of choice. A rigorous analysis on the models are required to test and optimize available scripts to arrive at a good model. Kaldi enables all traditional models such as diagonal GMM and Subspace Gaussian Mixture Models (SGMMs), but also expansion to new model variants. All these models are attempting to arrive at the best fit to decode a given human speech signal as described by the below fundamental equation of statistical speech recognition,

$$\hat{w} = \underset{w \in \sum *}{argmax} \quad P(w|o) = \underset{w \in \sum *}{argmax} \quad P(o|w)\, P(w)$$

, where $w, \hat{w}$ - lexical utterance and $o$ is audio

Further, above equation is known as the popular Bayes rule and a simplified version is given by the following equation,

$$P(uttereance|audio) = \frac{P(audio|utterance)\,.P(utterance)}{p(audio)}$$

### 3.2.1 Hidden Markov Model (HMM) Algorithms

HMMs are the fundamental algorithms for context independent phones and states. There are 3 algorithms under HMM.

- Likelihood computation (forward algorithm) - Determine the overall likelihood of an observation sequence $X = (x_1, x_2, , , , , x_t, , , x_T)$ being generated by an HMM.
- Most probable state sequence (Viterbi algorithm) - An approximation algorithm for HMM training.
- Estimating the parameters (Expectation Maximization (EM) algorithm)- An iterative algorithm for mizture of gaussians for HMM training.

Figure 3.4 illustrates the probabilistic finite state representation of HMM. Kaldi ASR toolkit has ability to compute likelihoods using forward algorithm and Viterbi algorithm.

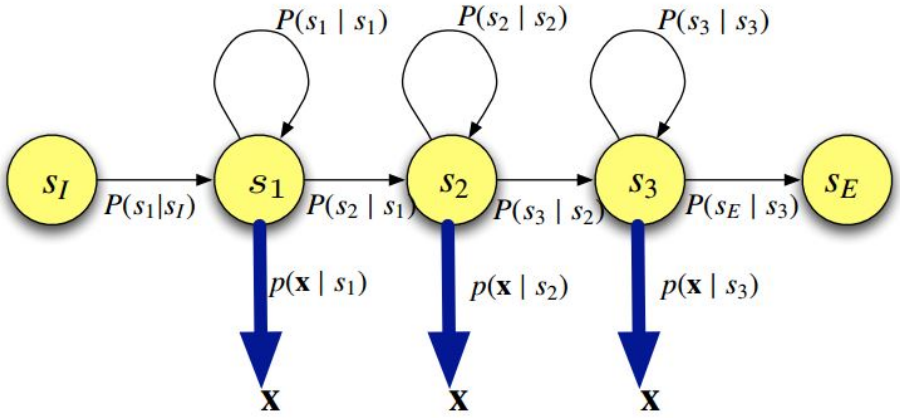


Figure 3.4: Probabilistic finite state automaton as HMM

where parameters are,

- Transition probabilities: $a_{kj} = P(S = j|S = k)$
- Output probability density function: $b_j(x) = (x|S = j)$

### 3.2.2 Gaussian Mixture Model (GMM)

A Gaussian Mixture Model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. Gaussian Mixture Models (GMMs) with diagonal covariance structures and full covariance structures are

supported by Kaldi[27]. a GMM class with natural parameters, which are inverse covariance and means times inverse covariance, is directly implemented without separately representing the Gaussian densities. The constant term, that consists of all independent terms on the data vector, in likelihood computation is also stored by the GMM classes. An efficient computation of log likelihood with dot products is supported by such an implementation. Figure 3.5 illustrates how GMM is associated with HMM states and phones.

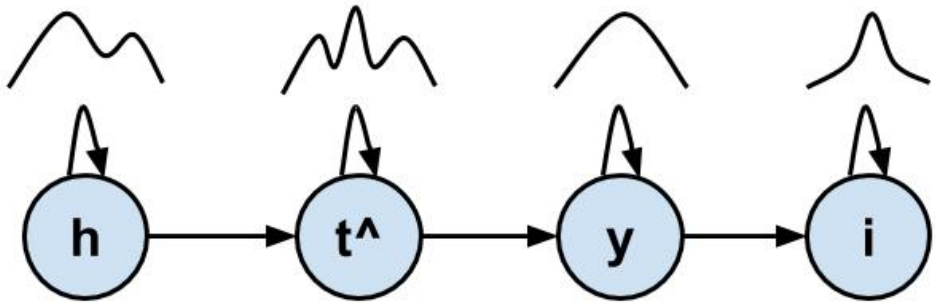


Figure 3.5: GMM to model HMM states (හතයි -7)

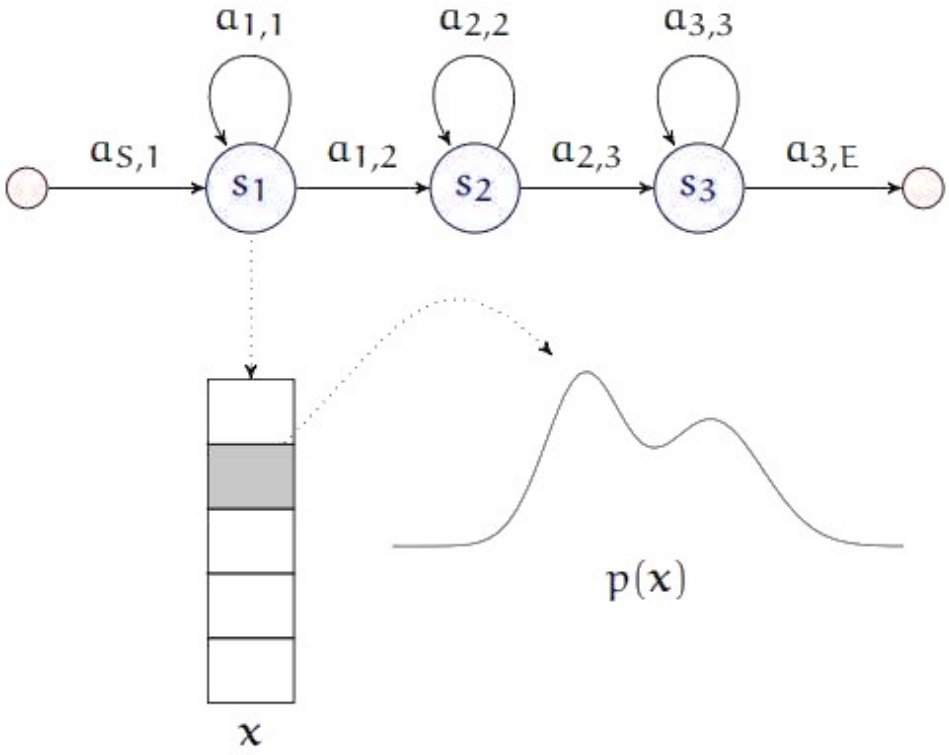


Figure 3.6: Single Gaussian mixtures corresponding to the HMM state $s_1$

where parameters are,

- Transition probabilities: $\alpha_{S,1}, \alpha_{3,E}, \alpha_{x,y}$

Figure 3.6 shows how each state of HMM relates to s mixture of Gaussians.

### 3.2.3 Speaker Adaptive Training (SAT)

Both Maximum Likelihood Linear Regression (MLLR) doing model space adaptation[18] and Feature-space Maximum Likelihood Linear Regression (fMLLR), also known as constrained MLLR, doing feature-space adaptation is supported by Kaldi[27]. A regression tree can be used to estimate multiple transforms for both MLLR techniques. In the feature pipeline, the requirement of a single fMLLR can be achieved using another processing step in addition. Kaldi[27] also supports normalization at speaker level using a linear approximation to VTLN, [45], or similar to traditional feature level VTLN, or using ET as a gender normalization approach in general. Both fMLLR and VTLN can be used for Speaker Adaptive Training (SAT) of the acoustic models.

### 3.2.4 Subspace Gaussian Mixture Model (SGMM)

The Subspace GMM acoustic model has both globally shared parameters and parameters specific to acoustic states, and this makes it possible to do various kinds of tying. In the past we have investigated sharing the *global* parameters among systems with distinct acoustic states; this can be useful in the multilingual setting. In the current paper we investigate the reverse idea: to have different global parameters for different acoustic conditions (gender, in this case) while sharing the specific parameters to acoustic state. We experiment with modeling gender dependency in this way, and we show Word Error Rate (WER) improvements on a range of tasks.

### 3.2.5 Monophone Model

An observation vector set and a prototype model are used to create first sets of single Gaussian monophone HMMs. Each HMM has a set of five states including the silence model as first and last state. A single Gaussian with its mean, variance and mixture weights represents each state. First a set is created with identical monophone HMMs where mean and variance of each of them are also identical. A set of data files is scanned, the global mean and variance is computed and a given HMM is set to have same mean and variance in all its Gaussians. An embedded retraining is executed in several iterations after the creation of initial models to build new HMMs by updating model values to improve the observation sequence probability. A phonetic decision tree, figure 3.7, without splits is created to support decoding in further steps. Decision trees are discussed in the subsequent subsection.

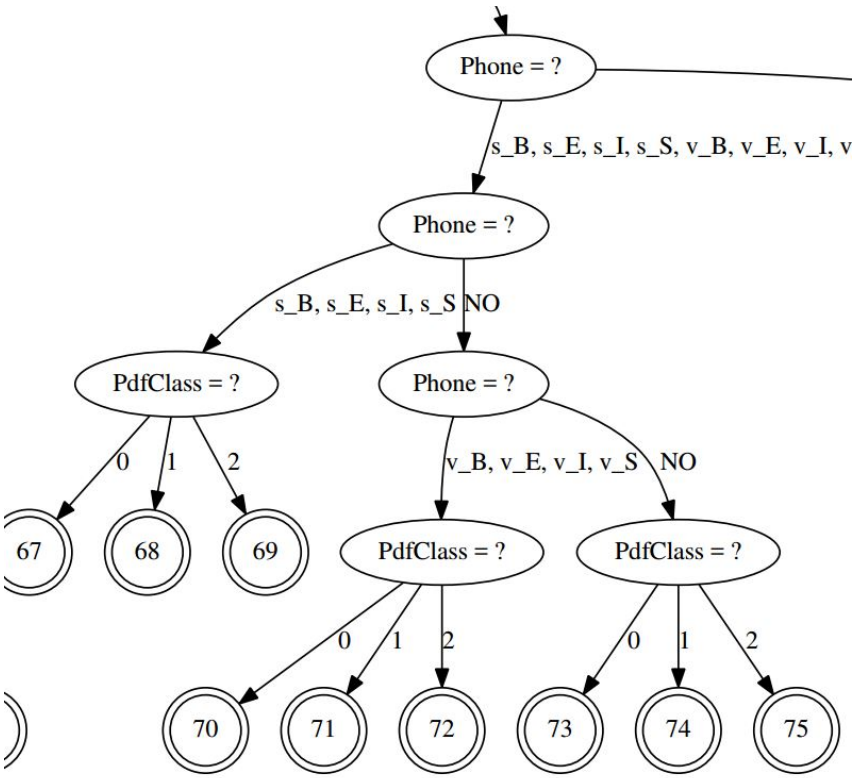


Figure 3.7: Phonetic context decision tree

### 3.2.6 Triphone Model

Monophone model has several shortcomings that could have a sever effect on the ASR performance drastically, *coarticulation* or difference of sounds of phones in different scenarios, phones that are located immediately before and after impacts most strongly. In order to build a triphone model, it requires $38x38x38$ models per triphone model to build for 38 phones. However, it is quite a large number of models to train. By converting transcriptions of monophone to triphone which is also known as word internal can be used to build context dependent triphone HMMs. By copying monophones and estimating them again, a set of triphone models are created.

## 3.3 Decision Trees

Objective of building the phonetic decision tree code is to make it efficient for arbitrary context sizes that is to avoid all contexts enumeration, and also to adequately generalize to support a wide range of ASR approaches. The traditional approach is, in each monophone of each HMM state, to attach a decision tree that asks questions about the left phones and right phones. Sharing of decision tree roots is encouraged among both the phones and the states mentioned in Kaldi[28]

toolkit. Further, any phone in the context window can be questioned, and also any HMM state. In order to supply phonetic questions, linguistic knowledge can be taken as the base, but in Kaldi recipes, phonetic tree clustering is taken as the base to generate the questions automatically. Questions on lexicons such as phonetic stress marked in the pronunciation dictionary and an extended phone set supports the start and end of word information of such a question. In that case, the decision tree roots are shared among the different versions of the same phone.

## 3.4 Lexicon

Lexicon is also known as a pronunciation dictionary where each entry has its legal set of constructive phonemes. There are many popular phonetic alphabets, International Phonetic Alphabet (IPA) being the prominent among computer society, various other transliteration schemes are available. In this study, Sihnala transliteration scheme developed by UCSC is used. Table 3.2 tabulates some phonemes of the transliteration scheme. There are few challenges in building a valid lexicon for a vocabulary. Several words may have multiple pronunciations. For example word දාහතර may have 2 different pronunciations, /d^ a: h a t^ @ r @/ and /d ^a h a h a t^ @ r @/. Another challenge is adding variant dialects and pronunciations to a valid lexicon such as Southern dialects.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ◌ං | x | ක | k | ඵ | t^ | ෆ | f | ඍ | ri | ඤ | cn | ඹ | mb | ◌ෟ | i^ |
| ◌ඃ | h | ඛ | k | ද | d^ | ◌ා | a: | ඎ | ri: | ඥ | jn | ය | y | ◌ෟ | i^: |
| අ | a | ග | g | ධ | d^ | ◌ැ | ae | ප | i^ | ජ | nj | ර | r | ◌ෙ | e |
| ආ | a: | ඝ | g | න | n | ◌ෑ | ae: | ඵ | i^: | ට | t | ල | l | ◌ෙ᪽ | e: |
| ඇ | ae | ඞ | x | ඳ | nd^ | ◌ි | i | එ | e | ඨ | t | ව | w | ◌ෛ | ai |
| ඈ | ae: | ඟ | ng | ප | p | ◌ී | i: | ඒ | e: | ඩ | d | ශ | s^ | ◌ො | o |
| ඉ | i | ච | c^ | ඵ | p | ◌ු | u | ඓ | ai | ඪ | d | ෂ | s^ | ◌ෝ | o: |
| ඊ | i: | ඡ | c^ | බ | b | ◌ූ | u: | ඔ | o | ණ | n | ස | s | ◌ෞ | au |
| උ | u | ජ | j | භ | b | ◌ෘ | ru | ඕ | o: | ඬ | nd | හ | h | | |
| ඌ | u: | ඣ | j | ම | m | ◌ෲ | ru: | ඖ | au | ත | t^ | ළ | l | | |

Table 3.2: Sinhala transliteration scheme[19]

## 3.5 Language Model

The language model represents the probability of word sequence that forms a valid sentence in the given language. $P(w)$ of the fundamental probabilistic ASR equation refers to the language model. Furthermore, it effects on the performance of the ASR system significantly. A language model for any language is represented by the standard format of ARPA language model. However, Kaldi ASR framework favours Finite State Transducer (FST) to represent any language model, Kaldi[27] converts these standard formats to its friendly Weighted Finite State Transducer (WFST) format. There is a variety of language model configurations such as uni-gram, bi-gram and tri-gram language models which formulates the probabilistic dependency of previous words to the next word. The probability of a word in a word sequence is conditioned on only the previously spoken $n$ words as known as an n-gram language model, can be utilized to obtain a simple statistical method, namely Markovian assumption. Equation below formulates a probabilistic n-gram equation for an utterance,

$$P(W) = \prod_{i=1}^{n} P(w_i|,,,w_{i-1}) = \prod_{i=1}^{n} P(w_i|h)$$

## Summary

This chapter describes the functional modules of the ASR pipeline. Further, the proposed methodology attempts to solve general ASR equation that can be found in literature. Also, this section seeks the possibilities of adapting Sinhala language to a general ASR pipeline which has many language dependent components such as lexical model. Acoustic model of this research is based on HMM-GMM. Therefore, all acoustic model features are related to base HMM model.

# Chapter 4: Design Methodology of Sinhala Number Sequence Decoder

## Introduction

One of the key objectives of this research is to implement an online Sinhala digit decoder of an acceptable WER. Tools from the Kaldi[26] ASR framework are adopted to implement the decoder. Final decoder is able to decode any digit sound from 0 to 999 pronounced in various contexts by various individuals such as voice mail prompts, IVR prompts. Digit vocabulary has substantial content of utterances, hence it is selected for the design of ASR pipeline. Each and every step in the design of ASR pipeline to suit Sinhala language, construction of the lengthy lexicon for 2 different representations and various online decoders are discussed in this chapter.

## 4.1 Kaldi Framework

Kaldi[26] presents a speech recognition system based on finite-state transducers using the freely available OpenFst[4], together with comprehensive documentation and scripts for building complete speech decoder systems. Kaldi is written is C++ language, and the core library supports modeling of arbitrary phonetic-context sizes, acoustic modeling with Subspace Gaussian Mixture Model (SGMM) as well as standard Gaussian mixture models, together with all commonly used linear and affine transforms. Kaldi is released under the Apache License v2.0, which is highly nonrestrictive, making it suitable for a wide community of users.

### 4.1.1 ASR Toolkits

Researchers on Automatic Speech Recognition have several potential choices of open-source frameworks and toolkits for building a decoder system. Notable among these are: HTK[43] and Julius[3] which are both written in C, CMU Sphinx-4 which is written in Java, and the RWTH ASR toolkit which is written in C++. Kaldi[27] specific requirements that are FST

based framework, extensive linear algebra support, and a non-restrictive license led to the development of Kaldi.

Kaldi is mainly intended on focused acoustic modeling research; thus, closest parallel choices can be viewed as the HTK and the RWTH ASR toolkit. The chief advantage versus HTK is modern, flexible, cleanly structured code and better WFST and math support; also, Kaldi license terms are more open than either HTK or RASR.

## 4.2 Online Sinhala Digit Decoder

One of the final outcomes of this research is an online continuous number sequence decoder as mentioned in the introduction of this chapter. Sinhala digit lexicon consists of a relatively large vocabulary and one great advantage is that the lexicon can be machine generated. This digit lexicon is also challenged by numerous pronunciations for a single number. In the following sections, design methodology of the ASR pipeline to build this decoder and algorithms that helped to overcome several challenges are described.

### 4.2.1 Sinhala Digit Lexicon

This section has several linguistics aspects because identifying phonemes of a given pronounced word requires some knowledge of phonology to deduce a correct phonetic sequence. As the author of this research has no such background to linguistics or phonology or any related subject area, a study on an online lexicon is done that is the CMU Pronouncing Dictionary[38]. The CMU Pronouncing Dictionary is an open-source machine-readable pronunciation dictionary for North American English that contains over 134,000 words and their pronunciations. CMUdict[38] is being actively maintained and expanded. Its phoneme or precisely the phone set is based on the ARPAbet[6] symbol set and developed for speech recognition tasks.

First attempt is based on the ARPAbet to derive a lexicon for the Sinhala digit vocabulary. ARPAbet can also be translated to International Phonetic Alphabet (IPA). Table 4.1 tabulates some areas of the Sinhala digit lexicon done on ARPAbet.

That study also reveals that it only requires a lexicon of **115** phone sets to derive the whole lexicon from 0 to 999. <SIL> and <UNK> refer to the silence and spoken noise.

Secondly, an attempt was taken on Sinhala transliteration scheme[19], a result of the PAN

| | | | |
|---|---|---|---|
| <SIL> | SIL | 5-පහ | p aa hh ay |
| <UNK> | spn | 6-හය | hh ah y er |
| 0-බින්දුව | b ih n d uw w er | 6-හය | hh ah y ay |
| 0-බින්දුව | b ih n d uw w ay | 7-හත | hh ah th er |
| 1-එක | eh k er | 7-හත | hh ah th ay |
| 1-එක | eh k ay | 8-අට | ah t er |
| 2-දෙක | dh eh k er | 8-අට | ah t ay |
| 2-දෙක | dh eh k ay | 9-නවය | n ah m er y er |
| 3-තුන | th uw n er | 9-නවය | n ah m er y ay |
| 3-තුන | th uw n ay | 9-නවය | n ah w er y er |
| 4-හතර | hh ah th er r er | 9-නවය | n ah w er y ay |
| 4-හතර | hh ah th er r ay | 10-දහය | dh ah hh ah y er |
| 5-පහ | p aa hh ah | 10-දහය | dh ah hh ah y ay |

Table 4.1: Sinhala digit lexicon (ARPAbet)

localization project support, and the same algorithm was executed to machine generate the entire lexicon. Table 4.2 shows first few utterances of the lexicon. This table follows the phone alphabet in table 3.2.

| | | | | | |
|---|---|---|---|---|---|
| <SIL> | SIL | 3-තුනයි | t^ u n a y i | 8-අට | a t @ |
| <UNK> | spn | 4-හතර | h a t^ @ r @ | 8-අටයි | a t a y i |
| 0-බින්දුව | b i n d^ u w @ | 4-හතරයි | h a t^ @ r a y i | 9-නමය | n a m @ y @ |
| 0-බින්දුවයි | b i n d^ u w a y i | 5-පහ | p a h a | 9-නවය | n a w @ y @ |
| 1-එක | e k @ | 5-පහයි | p a h a y i | 9-නමයයි | n a m @ y a y i |
| 1-එකයි | e k a y i | 6-හය | h a y @ | 9-නවයයි | n a w @ y a y i |
| 2-දෙක | d^ e k @ | 6-හයයි | h a y a y i | 10-දහය | d^ a h a y @ |
| 2-දෙකයි | d^ e k a y i | 7-හත | h a t^ @ | 10-දහයයි | d^ a h a y a y i |
| 3-තුන | t^ u n @ | 7-හතයි | h a t^ a y i | | |

Table 4.2: Sinhala digit lexicon (Transliteration)

Sinhala transliteration alphabet is used to design the rest of the steps of the ASR pipeline.

#### 4.2.1.1 Digit Lexicon Generator

As described in the previous section, algorithm 1 is formulated using **`Java`** to auto generate the entire lexicon **`3047`** utterances. Following pseudocode outlines the auto generating algorithm,

Implementation of the above pseudocode can be found in the appendix B.

#### 4.2.1.2 Complexity of Digit Lexicon

As described in the previous section, the chosen lexicon has a quite large number of tokens including all inflections. Following table 4.3 describes how a given token can have the various inflections depending on the speaker variability.

| | | |
|---|---|---|
| 19 | දානමය | d^ a: n @ m @ y @ |
| 19 | දානමයයි | d^ a: n @ m @ y a y i |
| 19 | දානවය | d^ a: n @ w @ y @ |
| 19 | දානවයයි | d^ a: n @ w @ y a y i |
| 19 | දහනමය | d^ a h a n @ m @ y @ |
| 19 | දහනමයයි | d^ a h a n @ m @ y a y i |
| 19 | දහනවය | d^ a h a n @ w @ y @ |
| 19 | දහනවයයි | d^ a h a n @ w @ y a y i |

Table 4.3: Inflections of number 19 in Sinhala

Digit lexicon should have all variations to address these variations in spoken language.

### 4.2.2 Kaldi Structure

Prior to the discussion on feature extraction methods, it is important to look at the structure of a Kaldi[26] model project. Kaldi decoder for Sinhala numbers follows the project structure below,

- **steps** contains a set of scripts for creating an ASR system.
- **utils** contains scripts to modify Kaldi files in certain ways, for example to subset data directories into smaller pieces.
- **local** this directory typically contains files that relate only to the corpus we are working on (e.g. Sinhala Digit corpus). In this case it also may contain files relevant to the online

**Algorithm 1** Digit Lexicon Generator

```
1: function AutoGen(PrefixList p, Lexicon l)
        ▷ p contains all indexed prefixes such as විසි, තිස්, හැට, පන්සිය and l is the complete lexicon
2:     for i = 21; i < 1000; i++ do
3:         ldigit ← i%10
4:         mdigit ← (i − ldigit) %100
5:         fdigit ← (i − ldigit − mdigit) %1000
6:         if fdigit = 0 and ldigit ≠ 0 then
7:             for each lphone in l do
8:                 for each mphone in l.get(lphone) do
9:                     l.get(i).add(concat(mphone + lphone))
10:                end for
11:            end for
12:        end if
13:        if fdigit > 0 then
14:            if mdigit > 0 or ldigit > 0 then
15:                for each fphone in p do
16:                    for each mphone in l.get(i%100) do
17:                        l.get(i).add(concat(lphone + mphone))
18:                    end for
19:                end for
20:            end if
21:        end if
22:    end for
23:    return l                                  ▷ return the completed lexicon
24: end function
```

decoding.

- **data** will contain any data directories, such as a train and test directory for Sinhala number data.
- **exp** contains the actual experiments and models, as well as logs.
- **conf** contains configurations for certain scripts that may read them.
- **path.sh** contains the path to the Kaldi source directory.
- **digits_audio** contains all train and test speech data in 16kHz 8-bit PCM wav format.
- **online** contains all dynamic speech data when the online decoder is running.

### 4.2.3 Kaldi I/O Mechanisms

Classes defined in Kaldi have a uniform interface for I/O. The standard interface is shown below,

```
class SomeKaldiClass {
 public:
   void Read(std::istream &is, bool binary);
   void Write(std::ostream &os, bool binary) const;
};
```

Kaldi has a concept called **Table** where a Table is a concept rather than actual C++ class. It consists of a collection of objects of some known type, indexed by strings. These strings must be tokens (a token is defined as a non-empty string without whitespaces).

Kaldi has 2 different table specifiers known as wspecfier and rspecifier. The **Table** classes require a string that is passed to the constructor or to the Open method. This string is called a wspecifier if passed to the **TableWriter** class, or a rspecifier if passed to the **RandomAccessTableReader** or **SequentialTableReader** classes. Examples of valid rspecifiers and wspecifiers include,

```
std::string rspecifier1 = "scp:data/train.scp"; // script file.
std::string rspecifier2 = "ark:-"; // archive read from stdin.
// write to a gzipped text archive.
std::string wspecifier1 = "ark,t:|␣gzip␣-c␣>␣/some/dir/foo.ark.gz";
std::string wspecifier2 = "ark,scp:data/my.ark,data/my.ark";
```

### 4.2.4 Feature Extraction

In general, feature extraction involves several sub level steps such as, data capturing, data preparation, pre processing and feature computation finally.

#### 4.2.4.1 Data Preparation

Data preparation requires a certain amount of time as rest of the ASR pipeline highly depends on the consistency and integrity of the data preparation step. Kaldi requires data to be organized in a way that it supports all Kaldi underlying programming constructs.

There is a minimum of 3 file configuration that Kaldi requires as a minumum requirement. Those files are, **text**, **utt2spk** and **wav.scp**. The files are closely related by *utterance* and *speaker ids*, abbreviated to *utt_id* and *spk_id* in figure 4.1. If each utterance is from a different speaker, or if we have no information about speakers, then the utterance and speaker ids match. Otherwise the speaker information is used to pool statistics across utterances for speaker adaptation and for speaker specific scoring. In the absence of a segments file, which sets out what portions of each audio file should be used for an utterance id, the recording ids are equivalent to the utterance ids. In this case we use the entire length of each audio file set out in *wav.scp*.

To check that the data directories conforms to Kaldi specifications, validate them by running the following two lines of bash scripts,

```
./utils/validate_data_dir.sh data/train
./utils/validate_data_dir.sh data/test
```

It is a mandatory requirement to have contents of all these files to be in a sorted order. The reason for the sorting is to enable something equivalent to random-access look-up on a stream that does not support `fseek()`, such as a piped command. Kaldi also has a script to fix those sorting errors. Both train and test directories can be fixed by executing the following script,

```
./utils/fix_data_dir.sh data/train
./utils/fix_data_dir.sh data/test
```

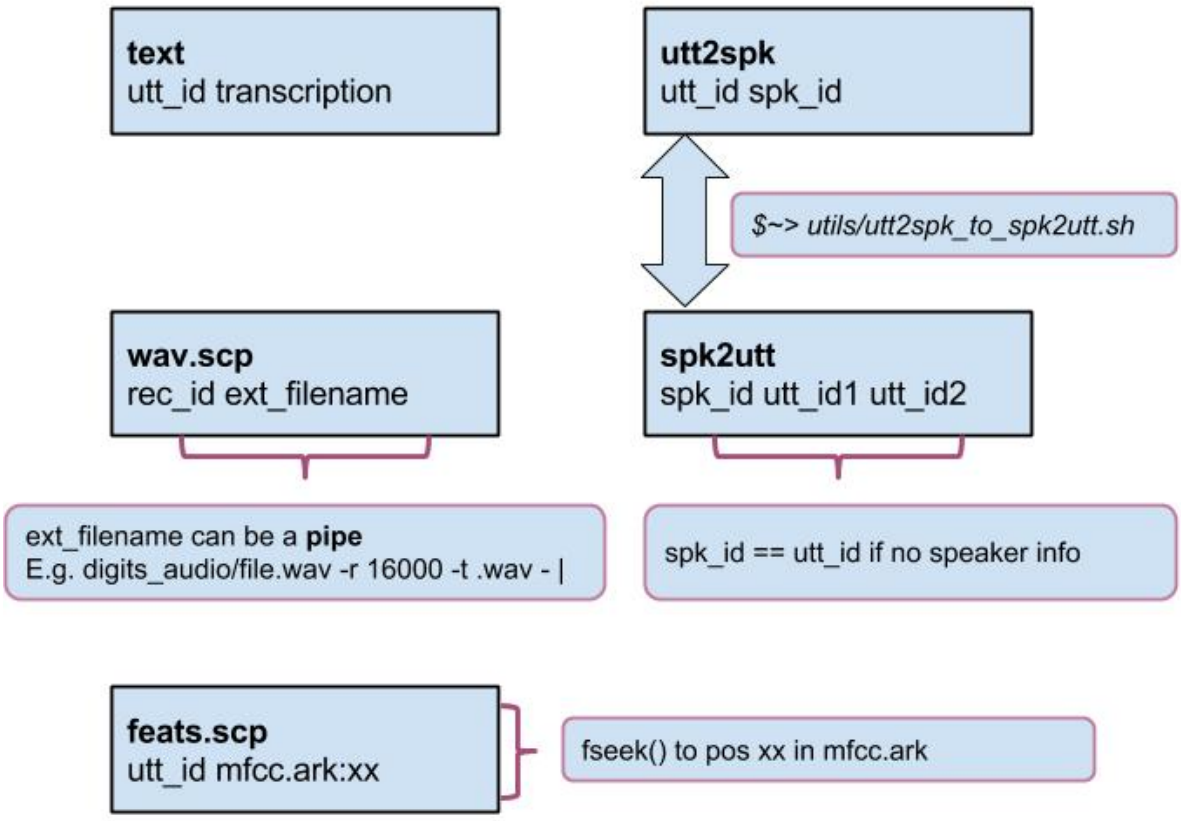


Figure 4.1: Illustration of a Kaldi data directory structure

#### 4.2.4.2 MFCC Feature Extraction

For GMM-HMM systems MFCC or PLP features are typically used, and then Cepstral Mean and Variance Normalization is applied to derive speaker independent features. MFCC for our Sinhala digit data is created by running the following bash script, which iterates over the data directories and extracts features for each.

```
for dir in train test; do
    ./steps/make_mfcc.sh data/$dir exp/make_mfcc/$dir mfcc
done
```

Above bash script has feature computation and normalization algorithms. Following discussion is about the fine tuning on those steps to arrive at a reasonable decoder.

- Mel Frequency Cepstral Coefficient Computation
  - **compute-mfcc-feats**

This program is given two command-line arguments: an rspecifier to read the .wav data, which is indexed by utterance, and a wspecifier to write the computed MFCC features , which are also indexed by utterance. In common usage, data is written to one big "archive" file and also written out an "scp" file for easy random access. Shown below is only the option to write to an archive file.

```
./compute-mfcc-feats --config=conf/mfcc.conf \
    scp:exp/make_mfcc/train/wav1.scp \
        ark:/data/mfcc/raw_mfcc_train.1.ark;
```

The configuration file **mfcc.conf** has several parameters related to speech signals. For example, the lower and upper cutoff of the frequency range covered by the triangular mel bins are controlled by the options **--low-freq** and **--high-freq** , which are usually set close to zero and the *Nyquist* frequency respectively, e.g. **--low-freq=20** and **--high-freq=7800** for 16kHz sampled speech.

- Cepstral Mean and Variance Normalization

Cepstral Mean and Variance Normalization performs the normalizing of the mean and the variance of the raw Cepstra, normally to produce zero-mean, unit-variance Cepstra, either on a per-utterance or per-speaker basis depending on the data that is available. In general model-based approaches are preferred to others to mean and variance normalization; e.g., the code for VTLN also learns a mean offset and the code for Exponential Transform does a diagonal Constrained Maximum Likelihood Linear Regression (CMLLR) transform that has the same power as Cepstral Mean and Variance Normalization. There is also the capability in the feature extraction code to subtract the mean on a per-utterance basis (the **--subtract-mean** option to **compute-mfcc-feats** and **compute-plp-feats**). In order to generate per-utterance and per-speaker mean and variance normalization we provide the programs **compute-cmvn-stats** and **apply-cmvn**.

These programs, despite the names, do not care whether the features in question consist of Cepstra or anything else; they are simply regarded as matrices. Of course, the features supplied to **compute-cmvn-stats** and **apply-cmvn** must have the same dimension. It is observed that it would probably be more consistent with the overall design of the feature transformation code, to supply a version of **compute-cmvn-stats** that would write out the mean and variance normalizing transforms as generic affine transforms (in the same

format as CMLLR transforms), so that they could be applied by the program **transform-feats**, and composed as needed with other transforms using **compose-transforms**.

- **compute-cmvn-stats**

  The program **compute-cmvn-stats** will, by default, compute the sufficient statistics for mean and variance normalization, as a matrix, and will write out a table of these statistics indexed by utt_id. If it is given the **--spk2utt** option, it will write out the statistics on a per-speaker basis instead.

```
./compute-cmvn-stats --spk2utt=ark:data/train.1k/spk2utt \
    scp:data/train.1k/feats.scp \
        ark:exp/mono/cmvn.ark;
```

### 4.2.5 Training Acoustic Models

Kaldi offers a collection of stochastic acoustic models ranging from monophone model to DNN models. In this study, only GMM-HMM models have been studied and adapted to design the online decoder.

#### 4.2.5.1 Monophone Training

Kaldi has several bash scripts to train acoustic models. Kaldi has a script called **steps/train_mono.sh** that can create a monophone model as mentioned below,

```
./steps/train_mono.sh --nj 4 data/train data/lang exp/mono
```

Monophone training involves several binary programs which initializes and trains a model, aligns monophone with data.

- Monophone Model Initialization

  - **gmm-init-mono**

    This binary program accepts two inputs and returns two outputs. As inputs, first a topology file which describes the structure of the HMM phones of the acoustic model (e.g. **data/lang/topo**) is needed and then the number of dimensions of

each Gaussian mixture component (e.g. **39**) must follow.

As outputs, a GMM-HMM model (e.g. **0.mdl**) and a phonetic decision tree (e.g. **tree**) are returned. For example,

```
./gmm-init-mono data/lang/topo \
    39 \
        exp/mono/0.mdl \
            exp/mono/tree;
```

Output **tree** is phonetic-context decision tree in this case which does not have any splits as it is a monophone model.

**– compile-train-graphs**

Above step builds a model and a tree. The following command creates an archive that consists of the graph HCLG corresponding to each of the transcriptions training data. That is, this program compiles FST automata, one for each training utterance.

```
./compile-train-graphs exp/mono/tree \
    exp/mono/0.mdl \
        data/L.fst \
            ark:data/train.tra \
                ark:exp/mono/fsts;
```

The input file train.tra contains integer representation of the training transcripts. Kaldi has a Pearl script (**utils/sym2int.pl --map-oov data/lang/phones/oov.txt -f 2- data/lang/words.txt data/train/text**) to generate integers from symbols. E.g. a given line might look like the following line, where the first token of the line (**s_3025419786_9**) is the utterance id.

```
s_3025419786_9 20 1 15 30 25 2 50 40 45 35
```

The output of the **compile-train-graphs** program is the archive **fsts**; it contains an FST (in binary form) for each utterance id in **train.tra**. The FST automata in this archive correspond to HCLG. The overall picture for decoding-graph creation is that we are constructing the graph $HCLG = H \circ C \circ L \circ G$.

1. H refers to the HMM definitions; output symbols of H represent context-dependent phones and input symbols of H are transition-ids, which encode

the pdf-id.

2. C represents the context dependency: its output symbols are phones and its input symbols represent context-dependent phones.
3. L is the lexicon; its output symbols are words and its input symbols are phones.
4. G represents a grammar; it is a finite-state acceptor which encodes the language model.

Graph creation recipe during training time is simpler than during test time, mainly because the disambiguation symbols are not required during the test time.

- Monophone Model Training

  Now that the monophone model is initialized, there are several binary programs to train the model. Following binary programs were used to train the model,

  - **align-equal-compiled**

    This program returns an wspecifier for alignments when given the inputs: an acoustic model, an rspecifier for graphs and an rspecifier for features. An alignment is a vector of integers (per utterance). This is the E-Step in the HMM EM training algorithm which aligns hidden states to the audio input. During the very first stage in the initialization, each utterance gets an equally spaced alignment.

```
./align-equal-compiled 0.mdl \
    graphs.fsts \
        scp:train.scp \
            ark:equal.ali;
```

    There is another program called **show-alignments** that produces alignments in more readable format against phones.

  - **gmm-acc-stats-ali**

    This program returns a file of accumulated stats (e.g. **0.acc**) for GMM training, given the three inputs that are a compiled acoustic model (e.g. **0.mdl**), MFCC features saved in **train.scp** and the alignment of hidden states to audio previously calculated.

```
./gmm-acc-stats-ali 0.mdl \
    scp:train.scp \
```

```
        ark:equal.ali \
            0.acc;
```

- **gmm-est**

  This is the M-Step in the HMM EM training algorithm. This program will output a new acoustic model, which has been updated via maximum likelihood re-estimation, given an acoustic model and a file of accumulated statistics for GMM training.

```
./gmm-est --min-gaussian-occupancy=3 \
    --mix-up=250 \
        exp/mono/0.mdl \
            exp/mono/0.acc \
                exp/mono/1.mdl;
```

  As the amount of data that is being dealt with is very small, an option called **--min-gaussian-occupancy** to the above program ensures the correct estimation of rare phones and avoids alignment problems.

- Monophone Alignment

  - **gmm-align-compiled**

    This program returns an wspecifier for alignments, given an acoustic model, an rspecifier for graphs and an rspecifier for features. This is the E-Step in the EM training algorithm (i.e. hidden state alignment to the audio input). These alignments are between GMM states and feature vectors. Each HMM state has an output Gaussian distribution, and the feature vectors assigned to that distribution are then used to update the Gaussian parameters (i.e. $\mu$ and $\Sigma$).

```
gmm-align-compiled 1.mdl \
    ark:graphs.fsts \
        scp:train.scp \
            ark:1.ali;
```

All binary programs associated with 3 above steps of the monophone training are bundled in 3 bash shell scripts: **steps/train_mono.sh**, **utils/mkgraph.sh** and **steps/align_si.sh**.

#### 4.2.5.2 Triphone Training

Phones are found to vary depending on the preceding and succeeding phones and this aspect needs to be captured within the acoustic models to improve performance. This can be achieved by building triphone models.

- Building of a Phonetic Decision Tree for Context-Dependent Triphones

  It is required to follow a few steps to build a conventional phonetic decision tree. First, a monophone system is trained (or use a pre-built triphone system) to get time alignments for the audio clips. Then statistics can be accumulated to train a single Gaussian per HMM state for each seen triphone, given those alignments.

  The sufficient parametric statistics to compute a Gaussian are count, sum, and squared sum. This implies that the total statistics size (for 39-dim feats) is $(38 \times 38 \times 38) \times (3 - HMM - states) \times (39 + 39 + 1)$.

  - **`acc-tree-stats`**

    This program returns an accumulation of tree statistics given an acoustic model as the first input, an rspecifier for the acoustic features as the second input, an rspecifier for previously made alignments as the third input.

    Monophone alignments are taken and **`AccumulateTreeStats()`** function (invoked from **`acc-tree-stats`**) is used to accumulate statistics for training the tree, after training a monophone system from a flat start.

    This binary program not only reads monophone alignments but it works for context dependent alignments too so trees based on triphone alignments can also be built. The tree building statistics takes the type **`BuildTreeStatsType`** when written to the disk. The function **`AccumulateTreeStats()`** takes in the values $N$ and $P$, as explained in the previous section, these parameters are set to 3 and 1 respectively by default by the command-line programs. Further, this can be overridden using the **`--context-width`** and **`--central-position`** options. In order to reduce the size of the statistics, **`acc-tree-stats`** is given a given a set of context independent phones such as **`SIL`** even though it is not mandatory. This binary program can accumulate corresponding statistics for context independent phones without the keys pertaining to the left and right phone defined in the context free events maps. In

the following invocation, **JOB** refers to the number of the jobs that being executed as every where else in the Kaldi scripts. Given a scenario of a computing environment that consists of four cores, each job yields a file in the form of **$JOB.treeacc**.

```
acc-tree-stats final.mdl \
    scp:train.scp \
    ark:JOB.ali \
    JOB.treeacc;
```

**– sum-tree-stats**

Function of this binary program is to sum statistics for phonetic context tree building. This program outputs an accumulated tree stats file when the program is given ***.treeacc** files.

```
sum-tree-stats treeacc \
    *.treeacc;
```

**– cluster-phones**

This executable clusters phones or sets of phones for various applications. This program returns a list of questions (**questions.int**) which can be further used to cluster phones when it is given accumulated tree statistics (**treeacc**) and a list of phones to be clustered (**phonesets.int**). Phones are clustered to obtain questions which are just sets of phones. A question normally refers to a phonetic category. However, acoustic similarity is the feature for clustering. A hierarchy of sets of all sizes is yielded by tree clustering.

```
cluster-phones treeacc \
    phonesets.int \
    questions.int
```

**– compile-questions**

This program returns a set of phonemes transformed into a C++ object which contains questions for each key in **EventMap** when it is given an HMM topology (**topo**) and a set of phonemes (**questions.int**). Questions on different HMM states are prepared by this program. There are parameters that can be set to make changes that affect tree building.

```
compile-questions data/lang/topo \
```

```
    exp/tri1/questions.int \
    exp/tri1/questions.qst;
```

- **build-tree**

  The program **build-tree** is used to build and output the final tree when the statistics are accumulated in earlier steps. This program takes the accumulated tree statistics (**treeacc**), the questions configuration (**questions.qst**) and the file that contains roots (**roots.int**). The questions configuration is output by the **compile-questions** program. The roots file specifies the lists of phones which are having shared roots in the decision tree clustering process. For each phone set, "shared" or "not-shared" describes if there should be or should not be separate roots for each of the pdf classes. If it is "shared", a single root tree for all HMM states are defined. If it is "not-shared" there would be three of those tree roots, allowing one for each pdf class.

  Splitting of the decision tree should be done whether that split parameter is "split" or "not-split". This program returns a collection of decision trees. However, a post clustering to some level is done after the splitting.

```
build-tree treeacc \
    roots.int \
    questions.qst \
    topo \
    tree;
```

  There is a tool known as **draw-tree** in the Kaldi framework to illustrate the phonetic decision tree. For example,

```
draw-tree data/lang/phones.txt \
    exp/mono/tree | \
    dot -Tps -Gsize=8,10.5 | \
    ps2pdf - ~/tree.pdf
```

- Triphone Model Initialization

  - **gmm-init-model**

    A GMM acoustic model (**1.mdl**) is initialized using a decision tree (**tree**), accumulated tree stats (**treeacc**) and an HMM model topology (**topo**).

```
gmm-init-model tree \
    treeacc \
    topo \
    1.mdl;
```

**– gmm-mixup**

This executable program returns a new GMM acoustic model with an increased number of Gaussian components (**2.mdl**) when it is given a GMM acoustic model from the previous step (**1.mdl**) and per transition id occupation counts (**1.occs**)

```
gmm-mixup --mix-up=$numgauss \
    1.mdl \
    1.occs \
    2.mdl;
```

**– convert-ali**

This program converts alignments from one decision tree model to another, returning a set of new alignments given a previous GMM model, a new GMM model, a new decision tree and an rspecifier for a set of old alignments.

```
convert-ali monophones_aligned/final.mdl \
    triphones_del/2.mdl \
    triphones_del/tree \
    monophones_aligned/ali.*.gz \
    triphones_del/ali.*.gz \
```

**– compile-train-graphs**

This program creates training graphs without an transition probability by default. However, this program was extensively described in the monophone training. This program returns a wspecifier for training graphs, given a decision tree, an acoustic model for the training, a Finite State Transducer for the lexicion and an rspecifier for the training data labels as inputs.

```
compile-train-graphs tree \
    1.mdl \
    L.fst \
    text \
```

```
    fsts.*.gz;
```

- Triphone Model Training

  Those three programs, mentioned below, which are identical to those used in monophone training, are used to train a triphone model. Collectively, these three programs are executed recursively, generating alignments (E-Step), and updating model parameters when it is given the newly aligned feature vectors (M-Step). In general it is required to improve the GMM-HMM model in training to make good predictions on future audio data. The type of Expectation Maximization that is used during the training is the Maximum Likelihood Estimation. The likelihood as in Bayes theorem is the probability of the data given the model.

  - **gmm-align-compiled**
  - **gmm-acc-stats-ali**
  - **gmm-est**

### 4.2.6 Training Language Models

Kaldi[27] project structure containing lang/ directory did not describe how to create the file G.fst in it, which is the finite state transducer form of the language model or grammar that is used to decode with. However, it is possible to have multiple **lang*** directories for various testing purposes with different language models and dictionaries. The Wall Street Journal (WSJ) setup is an example that contains multiple lang directories.

Depending on whether a statistical language model or some type of grammar is going to be used, the building process of **G.fst** is different. In the Sinhala digit decoder setup there is a unigram grammar, which allows word level likelihoods. A unigram model is sufficient for the Sinhala digit decoder.

SRILM[36] is a famous toolkit for building language models. Kaldi framework has various language modeling toolkits. However, SRILM is selected and used for the language model of Sinhala digit decoder as it has the most comprehensive documentation and fully featured specification. Its only shortcoming is that it does not come with the most free license. Following is the usage message of **utils/format_lm_sri.sh**

```
Usage: ./utils/format_lm_sri.sh [options] <lang-dir> <arpa-LM> [<
   lexicon>] <out-dir>
The <lexicon> argument is no longer needed but is supported for back
   compatibility
E.g.: utils/format_lm_sri.sh data/lang data/local/lm/foo.kn.gz data/
   local/dict/lexicon.txt data/lang_test
Converts ARPA-format language models to FSTs. Change the LM
   vocabulary using SRILM.
```

The binary program **ngram-count** returns a language model in ARPA format when given a **corpus.txt**,

```
ngram-count -order $lm_order -write-vocab $local/tmp/vocab-full.txt -
   wbdiscount -text $local/corpus.txt -lm $local/tmp/lm.arpa
```

The Kaldi binary program, **arpa2fst**, converts the ARPA format language model into a weighted Finite State Transducer.

```
arpa2fst --disambig-symbol=#0 --read-symbol-table=$lang/words.txt
   $local/tmp/lm.arpa $lang/G.fst
```

### 4.2.7 Online Decoder

Online decoding is a type of decoding where the feature vectors are produced in real time. It is not necessary to wait and capture all audio before starting online decoding. Kaldi has 2 online decoding setups which were released in 2 different places in the timeline and the oldest one is deprecated now. This design involves the latest online decoding programs for HMM-GMM model.

#### 4.2.7.1 Kaldi Scope for Online Decoding

Online decoding usually comes with a set of subjective requirements. Therefore, tools that can do end-to-end online decoding could be irrelevant and this research hypothesis also supports that idea. Kaldi has basic and general constructs that aid the development of online decoders.

#### 4.2.7.2 Design of the Online Decoder

As all models that are tested in this study are GMM based, the program **online2-wav-gmm-latgen-faster** is used as the online decoder which is currently the primary online decoding program of Kaldi framework. This program is capable of reading the entire **wav** file internally and processing it chunk by chunk independently. Kaldi framework has sample scripts (**egs/rm/s5/local/online/run_gmm.sh**) on how to build suitale models and evaluate them. The main objective of this decoder program is to apply the GMM based online decoding procedure within a typical batch processing framework. Further, such as program can make the Word Error Rate evaluation simple and easy. One important thing to note here is that oversampling of an audio signal does not work whereas subsampling does.

#### 4.2.7.3 Feature Extraction in Online Decoding

Online decoding complexity mostly relates to the feature extraction and adaptation. However, Kaldi toolkit has several classes inheriting from **OnlineFeatureInterface** that provides a number of components for feature extraction. They can be found in the definition file known as **online-feature.h**. **OnlineFeatureInterface** is a base class for online feature extraction which specifies the object that provides features to the invoking method (**OnlineFeatureInterface::GetFrame()**) and which also declares the available frames (**OnlineFeatureInterface::NumFramesReady()**). How to obtain those features is not implemented in this interface.

Classes **OnlineMfcc** and **OnlinePlp** are the lowest level features defined in **online-feature.h**. The Cepstral Mean and Variance Normalization (CMVN) is an important part of this decoding pipeline subsequent section describes the internals of the implementation.

#### 4.2.7.4 CMVN in Online Decoding

Cepstral mean normalization is the most popular normalization technique where mean of the raw Mel Frequency Cepstral Coefficient (MFCC) features is deducted. However, Cepstral refers to the type of feature where the cepstrum is the inverse Fourier transform of the log spectrum, although it is really the cosine transform that is applied. Futher, in the variance counterpart of the Cepstral normalization, each feature dimension is scaled to make the variance equal to one.

Cepstral variance normalization is disabled in all scripts of the Sinhala digit decoder, although it is available for use at anytime. For brevity, only Cepstral mean normalization is referred to.

In all Kaldi scripts, Cepstral Mean and Variance Normalization (CMVN) is applied on a per speaker basis. Because of the non causal behavior of online decoding context, it is impossible to do on per speaker basis.

The basic technique, that is used in this design, is moving window Cepstral mean normalization where mean of a moving window of by default 6 seconds is accumulated. In order to improve the estimate for the first few seconds of each utterance, prior information of the same speaker or global average of the Cepstra is given to the decoder using **`OnlineCmvnOptions`**. The executable program **`apply-cmvn-online`** can perform this normalization from the training pipeline.

#### 4.2.7.5 Adaptation in Online Decoding

Feature-space Maximum Likelihood Linear Regression (fMLLR) is the most standard adaptation technique that is used in speech recognition domain. That technique is also known as Constrained Maximum Likelihood Linear Regression (CMLLR) in the ASR literature. fMLLR consists of an affine, i.e both linear and offset, transform of the features. Transformation is in the form of $d \rightarrow d \times (d + 1)$ where $d$ refers to the final feature dimension, typically 40. This online decoding design requires a basis method to estimate increasing transform parameter count incrementally as decoding of more data continues. **`SingleUtteranceGmmDecoder`** class implements the top level logic at the decoder level.

The fMLLR estimation is performed periodically as opposed to continuously because of the lattice posterior calculation that is not computationally feasible for the online decoder.

#### 4.2.7.6 Multiple Models in Online Decoding

Mostly three or fewer models can be given for the online decoding interface of Kaldi toolkit. **`OnlineGmmDecodingModels`** class implements all functionality required to decide which model to use in different circumstances whenever fewer models are given. Three possible models that can be given are,

- A speaker independent model that is trained usinf **`apply-cmvn-online`**

- A speaker adapted model that is estimated using fMLLR
- A discriminative speaker adapted model of which adaptation parameters are estimated using Maximum Likelihood

## 4.3 Data Collection

Data collection plays a major role in any data analytic problem as well as in this ASR system development. There are many iterations of data collection which are performed in this study. In order to train the ASR system efficiently, data of acceptable quality must be obtained. Section 5.2 outlines the strategy of data collection. Following list summarizes various data collection sources,

- telephony
- embedded device
- single or multiple speaker
- prompt variation
- speech modality
- text corpora and other resources

As this study focuses on decoding number sequences for telecommunication applications, data that was captures are examples of credit card numbers and telephone numbers.

## 4.4 Design of the Python Demo Program

A comprehensive program is implemented using Python to demonstrate the results of this Sinhala text streaming application for continuous digits strings. Following figure 4.2 illustrates the main functional blocks of the demo program.

- system recorder — Python demo program interact with the system recording application to capture the audio signal. System amplification of the audio signal is disabled in order to reduce noise. A predefined activation threshold is set so the recording thread starts capturing the digital audio stream when the signal level at the microphone exceeds the threshold.

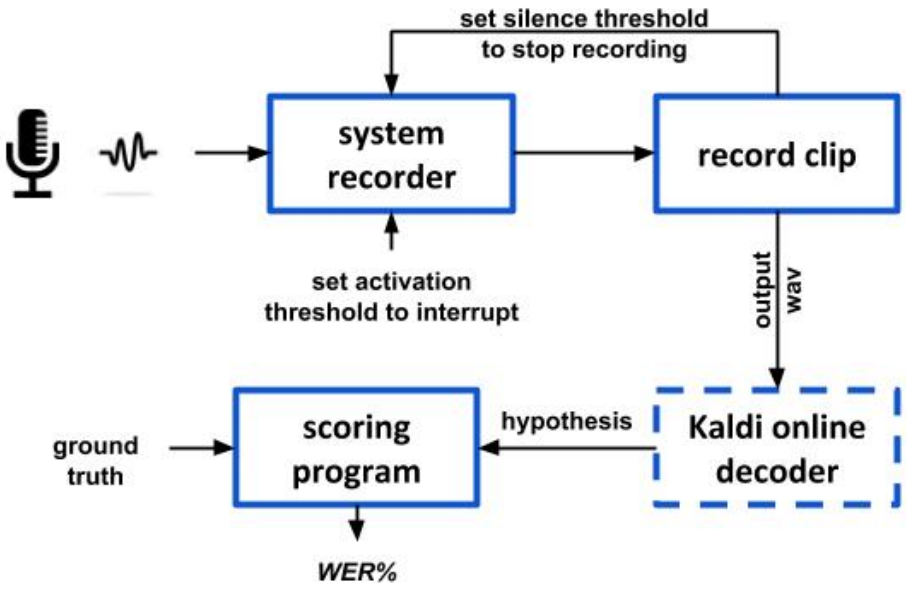


Figure 4.2: Block diagram of Python demo program

- record clip — Recording thread continuously listens to the microphone until a predefined period of silence is detected. As soon as the silence period exceeds, thread stops recording and creates an inter-process communication channel to communicate with the Kaldi[27] online decoder script. Prior to inter-process communication, Python demo program prepares all files required by the Kaldi online decoder program (**`wav.scp`**, **`utt2spk`**, **`text`**)
- Kaldi online decoder — This program is a script that executes all binary programs to extract features, decode and generate a hypothesis.
- scoring program — This method implements the algorithm 2 to calculate the WER when the ground truth is given.

## Summary

Design of the ASR pipeline requires extensive study on various aspects along the process. Final outcome of this design is an implementation of an online decoding program that can transcribe spoken Sinhala digit sequences. Any general ASR pipeline involves various feature transformations and modeling techniques. Techniques related to GMM model are studied with the help of tools available in Kaldi[27].

Further, priori functions such as lexical modeling are also important to complete the design of

the ASR pipeline. Requirements for lexical modeling and modeling tools are identified in this study. However, final research outcome of this effort is to produce an online decoder that can transcribe Sinhala speech. Therefore, several online decoding programs are studied in general to achieve the design objectives.

# Chapter 5: Evaluation and Results

## Introduction

A number of evaluation methods are available to compute the performance of a speech decoding program. Kaldi[27] toolkit supports mainly 2 different evaluators, Word Error Rate (WER) and Sentence Error Rate (SER). As the outcome of this research is an implementation of a Sinhala digit decoder, WER is calculated for the models that are trained. SER refers to the percentage of sentences with at least one error and that calculation is omitted in this study as the testing is done on limited spoken digit sequences at any given time.

## 5.1 Word Error Rate (WER)

The Word Error Rate (WER) is a way to measure performance of an Automatic Speech Recognition (ASR). It compares an output or a hypothesis to ground truth or what is actually said, and is defined like the equation below,

$$WER = \frac{S + D + I}{N}$$

where $S$ is the number of substitutions, $D$ is the number of deletions, $I$ is the number of insertions and $N$ is the number of words in the reference. Kaldi[27] has a local script to score the the accuracy of the hypothesis generated by the decoding program. A script known as **`score.sh`** has the follwing usage to compute the **`WER|SER`** pair,

```
Usage: local/score.sh [--cmd (run.pl|queue.pl...)] <data-dir> <lang-
   dir|graph-dir> <decode-dir>
```

Algorithm 2 outlines the pseudocode for WER calculation. This algorithm is inspired by the *Levenshtein* distance for words. Further, it is implemented in Python demo program to calculate the WER when the reference sentence is given. Detailed implementation can be found in appendix A.

**Algorithm 2** Compute WER between a hypothesis and a reference

```
1: function WER(Reference r, Hypothesis h)
2:     int [|r| + 1] [|h| + 1] D                                  ▷ Initialization
3:     for i = 0; i ≤ |r| ; i + + do
4:         for j = 0; j ≤ |h| ; j + + do
5:             if i == 0 then
6:                 D [0] [j] ← j
7:             else if j == 0 then
8:                 D [i] [0] ← i
9:             end if
10:        end for
11:    end for
12:    for i = 1; i ≤ |r| ; i + + do                              ▷ Calculation
13:        for j = 1; j ≤ |h| ; j + + do
14:            if r [i − 1] == h [j − 1] then
15:                D [i] [j] ← D [i − 1] [j − 1]
16:            else
17:                sub ← D [i − 1] [j − 1] + 1
18:                ins ← D [i] [j − 1] + 1
19:                del ← D [i − 1] [j] + 1
20:                D [i] [j] ← min (sub, ins, del)
21:            end if
22:        end for
23:    end for
24:    return D [|r|] [|h|]
25: end function
```

## 5.2 Training Data for Sinhala Digit Decoder

Training the models involves a data set from `31` different speakers. Training data set has `10` utterances from each speaker and all utterances are `10` digit numbers spoken in various contexts such as IVR for credit card call centers. Utterances in table 5.1 cover the all reduced combinations of all possibilities.

| | |
|---|---|
| 23 34 45 67 56 | විසි තුනයි තිස් හතරයි හතළිස් පහයි හැට හතයි පනස් හය |
| 3 1 2 5 4 6 0 7 8 9 | තුනයි එකයි දෙකයි පහයි හතරයි හයයි බින්දුවයි හතයි අටයි නමය |
| 0 71 691 36 85 | බින්දුවයි හැත්තෑ එකයි හයසිය අනූ එකයි තිස් හයයි අසූ පහ |

Table 5.1: Utterance possibilities

Table 5.2 briefs the gender distribution of the training data set.

| Gender | Count |
|---|---|
| Male | 15 |
| Female | 16 |

Table 5.2: Gender distribution of training data set

Table 5.3 shows the age distribution of the training data set of local speakers.

| Age Range | Count |
|---|---|
| 0 – 20 | 4 |
| 21 – 30 | 12 |
| 31 – 50 | 8 |
| 51 – 60 | 4 |
| 60 > | 3 |

Table 5.3: Age distribution of training data set

## 5.3 Testing Data for Sinhala Digit Decoder

Testing the models involves data set from `4` different speakers, `10` utterances of `10` digits each.

Table 5.4 briefs the gender distribution of the testing data set.

| Gender | Count |
|---|---|
| Male | 2 |
| Female | 2 |

Table 5.4: Gender distribution of testing data set

| Age Range | Count |
|---|---|
| 0 – 20 | 1 |
| 21 – 30 | 2 |
| 31 – 50 | 1 |
| 51 – 60 | 0 |
| 60 > | 0 |

Table 5.5: Age distribution of testing data set

Table 5.5 shows the age distribution of the testing data set of local speakers.

A decoding graph is created for each model in the training phase. The test utterances are decoded by a Kaldi script **steps/decoding.sh** with the help decoding graphs as well. However, this script works only for a small set of feature types. Usage of **steps/decoding.sh** is below,

```
steps/decode.sh [options] <graph-dir> <test-data-dir> <decode-dir>
```

There are various options that can be given to the above script. Those parameters are changed to achieve the best decoding WER. However, there are two parameters that are important to mention,

- **acwt** — Acoustic scale applied to acoustic likelihoods, applied in lattice generation (default **0.083333**). It affects the pruning of the lattice (low enough likelihood will be pruned).
- **min_lmwt** and **max_lmwt** — minimum and maximum language model weight for lattice rescoring (defualt **7** and **17**)

So the best idea is to do an initial decoding with the default acoustic scale of $0.1$. Then rescore the lattice and set the language model weight to integer values between $7$ and $15$. Then do another decoding with the acoustic scale of inverse of language model weight with best results and rescore the lattice with the best language model weight again.

As the final step of **steps/decoding.sh** the results are recorded. The results can be found in

**<decode-dir>** under filenames called **wer_<N>** where *N* is the language model scale.

Further, Kaldi provides another handy script (**utils/best_wer.sh**) to extract the language model weight that produces the best WER. Usage of the above script is as below,

```
grep WER exp/mono/decode/wer_* | utils/best_wer.sh
```

### 5.3.1 WER of Monophone Model

Following table 5.6 consists of the test results of the monophone model decoding for the test dataset. All files related to the monophone model along with its phonetic tree, decoding graph and etc. are in the **exp/mono**.

| LM weight | Insertions | Deletions | Substitutions | WER% | SER% |
|---|---|---|---|---|---|
| 7 | 7 | 1 | 18 | 23.64 | 81.82 |
| 8 | 7 | 1 | 18 | 23.64 | 81.82 |
| 9 | 7 | 1 | 18 | 23.64 | 81.82 |
| 10 | 6 | 1 | 17 | 21.82 | 81.82 |
| 11 | 5 | 1 | 17 | 20.91 | 81.82 |
| 12 | 5 | 1 | 16 | 20.00 | 81.82 |
| 13 | 3 | 1 | 19 | 20.91 | 90.91 |
| 14 | 3 | 1 | 19 | 20.91 | 90.91 |
| 15 | 2 | 1 | 19 | 20.00 | 90.91 |
| 16 | 2 | 1 | 19 | 20.00 | 90.91 |
| 17 | 1 | 1 | 22 | 21.82 | 90.91 |

Table 5.6: Test WER of the monophone model

Best WER script generates the following output that contains the accurate decoding model,

```
%WER 20.00 [ 22 / 110, 5 ins, 1 del, 16 sub ] exp/mono/decode/wer_12
```

Therefore, best WER of monophone decoding is,

$$WER - 20.00\%$$

### 5.3.2 WER of Triphone Model

Triphone model basically solves the coarticulation problem hence the improvement in WER.

#### 5.3.2.1 First Pass Decoding of Triphone Model

Following table 5.7 consists of the test results of the triphone model first pass decoding for the test dataset. All files related to the triphone model first pass along with its phonetic tree, decoding graph and etc. are in the **`exp/tri1`**.

| LM weight | Insertions | Deletions | Substitutions | WER% | SER% |
|---|---|---|---|---|---|
| 7 | 9 | 6 | 8 | 20.91 | 54.55 |
| 8 | 8 | 6 | 8 | 20.00 | 54.55 |
| 9 | 8 | 6 | 8 | 20.00 | 54.55 |
| 10 | 5 | 5 | 9 | 17.27 | 54.55 |
| 11 | 5 | 5 | 9 | 17.27 | 54.55 |
| 12 | 4 | 5 | 9 | 16.36 | 54.55 |
| 13 | 4 | 5 | 8 | 15.45 | 54.55 |
| 14 | 3 | 5 | 8 | 14.55 | 54.55 |
| 15 | 3 | 5 | 8 | 14.55 | 54.55 |
| 16 | 3 | 5 | 8 | 14.55 | 54.55 |
| 17 | 2 | 5 | 8 | 13.64 | 54.55 |

Table 5.7: Test WER of the first triphone pass model

Best WER script generates the following output that contains the accurate decoding model,

```
%WER 13.64 [ 15 / 110, 2 ins, 5 del, 8 sub ] exp/tri1/decode/wer_17
```

Therefore, best WER of triphone first pass decoding is,

$$WER - 13.64\%$$

#### 5.3.2.2 Second Pass Decoding of Triphone Model

Following table 5.8 consists of the test results of the triphone model second pass decoding for the test dataset. All files related to the triphone model second pass along with its phonetic tree, decoding graph and etc. are in the **`exp/tri2`**. Triphone second pass decoding takes alignments from the first pass to arrive at a reasonable estimate. This second pass has no effect on the WER improvement.

Best WER script generates the following output that contains the accurate decoding model,

| LM weight | Insertions | Deletions | Substitutions | WER% | SER% |
|---|---|---|---|---|---|
| 7 | 18 | 5 | 10 | 30.00 | 54.55 |
| 8 | 16 | 5 | 10 | 28.18 | 54.55 |
| 9 | 14 | 5 | 10 | 26.36 | 54.55 |
| 10 | 12 | 5 | 12 | 26.36 | 54.55 |
| 11 | 11 | 5 | 12 | 25.45 | 54.55 |
| 12 | 11 | 5 | 12 | 25.45 | 54.55 |
| 13 | 11 | 5 | 12 | 25.45 | 54.55 |
| 14 | 10 | 5 | 12 | 24.55 | 54.55 |
| 15 | 8 | 6 | 12 | 23.64 | 54.55 |
| 16 | 8 | 6 | 12 | 23.64 | 54.55 |
| 17 | 6 | 6 | 12 | 21.82 | 54.55 |

Table 5.8: Test WER of the second triphone pass model

```
%WER 21.82 [ 24 / 110, 6 ins, 6 del, 12 sub ] exp/tri2/decode/wer_17
```

Therefore, best WER of triphone second pass decoding is,

$$WER - 21.82\%$$

From the above results, it can be seen that second pass has no effect on the overall improvement of the WER. Therefore, second pass is omitted for future decoding. The reason for this degrading performance is coarticulation problem does not span beyond the triphone model that is modeled by a GMM.

### 5.3.3 WER of Triphone (LDA and MLLT) Model

This is also a triphone model with feature transforms applied to it. The overall process is described as if MLLT on top of LDA features are done, but it is applicable on top of traditional delta features as well. Following table 5.9 consists of the test results of the triphone model with LDA and MLLT decoding for the test dataset. All files related to the triphone model along with its phonetic tree, decoding graph and etc. are in the **`exp/tri3a`**.

Best WER script generates the following output that contains the accurate decoding model,

| LM weight | Insertions | Deletions | Substitutions | WER% | SER% |
|---|---|---|---|---|---|
| 7 | 14 | 3 | 11 | 25.45 | 54.55 |
| 8 | 14 | 3 | 11 | 25.45 | 54.55 |
| 9 | 13 | 3 | 11 | 24.55 | 54.55 |
| 10 | 12 | 3 | 11 | 23.64 | 54.55 |
| 11 | 12 | 3 | 11 | 23.64 | 54.55 |
| 12 | 12 | 3 | 11 | 23.64 | 54.55 |
| 13 | 11 | 4 | 10 | 22.73 | 54.55 |
| 14 | 10 | 4 | 10 | 21.82 | 54.55 |
| 15 | 9 | 5 | 10 | 21.82 | 54.55 |
| 16 | 9 | 5 | 10 | 21.82 | 54.55 |
| 17 | 9 | 5 | 10 | 21.82 | 54.55 |

Table 5.9: Test WER of the triphone model (LDA + MLLT)

```
%WER 21.82 [ 24 / 110, 10 ins, 4 del, 10 sub ] exp/tri3a/decode_test/
   wer_14
```

Therefore, best WER of triphone (LDA + MLLT) decoding is,

$$WER - 21.82\%$$

### 5.3.4 WER of SAT (fMLLR) Model

fMLLR is a feature space transformation technique that is widely used to obtain SAT models in ASR. Following table 5.10 consists of the test results of the SAT model with fMLLR for the test dataset. All files related to the SAT model along with its phonetic tree, decoding graph and etc. are in the **`exp/tri4a`**.

Best WER script generates the following output that contains the accurate decoding model,

```
%WER 10.91 [ 12 / 110, 7 ins, 0 del, 5 sub ] exp/tri4a/decode_test/
   wer_16
```

Therefore, best WER of triphone SAT decoding is,

$$WER - 10.91\%$$

| LM weight | Insertions | Deletions | Substitutions | WER% | SER% |
|---|---|---|---|---|---|
| 7 | 12 | 0 | 5 | 15.45 | 54.55 |
| 8 | 12 | 0 | 5 | 15.45 | 54.55 |
| 9 | 12 | 0 | 5 | 15.45 | 54.55 |
| 10 | 12 | 0 | 5 | 15.45 | 54.55 |
| 11 | 10 | 0 | 5 | 13.64 | 54.55 |
| 12 | 9 | 0 | 5 | 12.73 | 45.45 |
| 13 | 9 | 0 | 5 | 12.73 | 45.45 |
| 14 | 8 | 0 | 5 | 11.82 | 45.45 |
| 15 | 8 | 0 | 5 | 11.82 | 45.45 |
| 16 | 7 | 0 | 5 | 10.91 | 45.45 |
| 17 | 7 | 0 | 5 | 10.91 | 45.45 |

Table 5.10: Test WER of the SAT model

### 5.3.5 WER of SAT (MMI) Model

MMI is another linear transform available in latest Kaldi[27] framework. Following table 5.11 consists of the test results of the SAT model with MMI for the test dataset. All files related to the SAT model along with its phonetic tree, decoding graph and etc. are in the **`exp/tri4a_mmi`**.

| LM weight | Insertions | Deletions | Substitutions | WER% | SER% |
|---|---|---|---|---|---|
| 7 | 22 | 0 | 6 | 25.45 | 54.55 |
| 8 | 21 | 0 | 6 | 24.55 | 54.55 |
| 9 | 20 | 0 | 6 | 23.64 | 54.55 |
| 10 | 18 | 0 | 6 | 21.82 | 54.55 |
| 11 | 18 | 0 | 6 | 21.82 | 54.55 |
| 12 | 18 | 0 | 6 | 21.82 | 54.55 |
| 13 | 16 | 0 | 6 | 20.00 | 54.55 |
| 14 | 15 | 0 | 6 | 19.09 | 54.55 |
| 15 | 15 | 0 | 6 | 19.09 | 54.55 |
| 16 | 12 | 0 | 7 | 17.27 | 54.55 |
| 17 | 12 | 0 | 7 | 17.27 | 54.55 |

Table 5.11: Test WER of the MMI model

Best WER script generates the following output that contains the accurate decoding model,

```
%WER 17.27 [ 19 / 110, 12 ins, 0 del, 7 sub ] exp/tri4a_mmi/decode/
   wer_16
```

Therefore, best WER of triphone MMI decoding is,

$$WER - 17.27\%$$

From the results of the SAT model, fMLLR returns the lowest WER of all models. Further, the fMLLR model is superior than that of the Maximum Mutual Information (MMI) based SAT model.

### 5.3.6 WER of SAT (fMLLR) Online Model

Kaldi[27] toolkit has generic constructs to decode a speech signal online as described in section 4.2.7. SAT model with fMLLR transform is taken to proceed with online decoding.

The reason it is online is that internally, it processes a given audio data (wav) file sequentially as if it is being captured from an audio stream, so that when the end of the file is reached it is ready with the decoded output, with very little latency.

Table 5.12 contains the results from online decoding. SER is taken out from the table as it does not have a meaning in the real online decoding environment.

Best WER script generates the following output that contains the accurate decoding model,

```
%WER 22.73 [ 25 / 110, 11 ins, 4 del, 10 sub ] exp/tri4a_online/
   decode/wer_13
```

Therefore, best WER of online decoding is,

$$WER - 22.73\%$$

## 5.4 Discussion

In order to train the acoustic models, **31** native Sinhala language speakers are selected with a uniform gender distribution. All speakers are distributed in multiple age groups to address the speaker variability across all possibilities. Each speaker is given minimum of **10** sequences of

| LM weight | Insertions | Deletions | Substitutions | WER% |
|---|---|---|---|---|
| 7 | 14 | 3 | 11 | 25.45 |
| 8 | 14 | 3 | 11 | 25.45 |
| 9 | 13 | 3 | 11 | 24.55 |
| 10 | 12 | 3 | 11 | 23.64 |
| 11 | 12 | 3 | 11 | 23.64 |
| 12 | 12 | 3 | 11 | 23.64 |
| 13 | 11 | 4 | 10 | 22.73 |
| 14 | 11 | 4 | 10 | 22.73 |
| 15 | 13 | 3 | 11 | 24.55 |
| 16 | 12 | 3 | 11 | 23.64 |
| 17 | 12 | 3 | 11 | 23.64 |

Table 5.12: Test WER of the online model

digits and recorded using a sub program in the Python demo program. It is sufficient to record that number of sequences because the number of possible scenarios are relatively low compared to other complex grammars.

A single Kaldi[27] script (**`run.sh`**) is executed to generate all models (monophone model, triphone first pass model, triphone second pass model, triphone LDA + MLLT model, SAT fMLLR model, SAT MMI model and online decoder model with feature and linear transform). Same training dataset is used for training of all acoustic models to ensure the integrity of the methodology.

A test dataset is acquired in the same way but with less number of speakers and considerable variability to calculate the WER of decoders of models mentioned above. WER of offline models vary from a minimum rate of $10.91\%$ to a maximum rate of $21.82\%$. Equivalent success rate varies from a minimum of $78.18\%$ to a maximum of $89.09\%$.

Online decoder model reports a success rate of $77.27\%$. Online decoder nearly falls behind by a $10\%$ when considering SAT + fMLLR offline decoder model. Success rate of the offline decoder can be further improved by providing more data to the training models.

## Summary

All model variants tested above are done using Sinhala continuous digit sequences spoken by native speakers in Sri Lanka. Above test results are achieved after several fine tuning of the training and decoding parameters. Testing and scoring criteria adapted here is the most popular technique in ASR literature. All important findings and results are tabulated in previous sections of this chapter. Acoustic scale and language model weight are two important parameters which are changed from default values to obtain acceptable decoding performance.

SAT model with fMLLR transform returns the nearly $\sim 90\%$ accuracy above all other decoding models. Hence, fMLLR model is used as the base model for online decoding of Sinhala spoken digit sequences. Sinihala digit vocabulary is not the largest vocabulary available, but it is substantial for this study to constitute that research objectives have been reached.

# Chapter 6: Conclusion and Future Work

## Introduction

This chapter recognizes all work carried out in previous sections in a critical manner to validate whether the research holds integrity and continuity. In conclusion, all activities done through out the scope of this study are analyzed and presented in such a way that any related future work is inspired.

## 6.1 Conclusion

Main research objective of this comprehensive study is to validate the hypothesis of an online transcription program for continuous spoken word sequences in Sinhala language. A software tool is developed to interact with various acoustic and language models, which are modeled, trained and tested using a popular extended framework, and to output a hypothesized transcription for a Sinhala audio signal. Minimum error rate (WER) that was reported during the study is $10.91\%$, which is a considerable achievement, compared to the performance rate of HSR, to establish that the main research objective is reached. The methodology on how this study is carried out from the beginning to support the continuity along this research area and also to achieve other research goals are discussed in following sections.

Initially, this research started with an in depth look at the Automatic Speech Recognition (ASR) literature in general. A comprehensive literature review is conducted to identify the gaps in this area of study. It becomes evident that less research has been done on the Sinhala speech decoding. Further, there are less resources to continue a similar study on the native language. Hence, the requirement for an online Sinhala speech transcription tool is identified and research problem on the same is finalized.

This research is conducted using a community backed free and open source toolkit, Kaldi[27] toolkit. It is one of the objectives to incorporate a more open framework to continue the research at a higher level abstraction. Kaldi framework has most of the ground level constructs developed using mainly C++ and exposed as shell scripts for customization. A similar programming style

can help to view the ASR pipeline from higher level perspective. Further, such an abstraction allows rapid testing as it basically involves fine tuning of parameters. All binary programs, that are relevant for the GMM based acoustic modeling, are studied in depth and documented for future use.

Capability to extend to a large corpora or a large vocabulary, using the findings of this study is another key objective of this effort. Sinhala digit vocabulary up to three digits is the vocabulary that constructs the corpus for this study. Initial assumption is to employ a substantial vocabulary to prove the results for a valid hypothesis. However, the models, that are trained and tested, should work for vocabularies of different sizes at the cost of varying error rates. It is merely a data collection problem as far as the developments of this study is concerned. In conclusion, this research has made available required tools and documentation to continue and support a similar or advanced study on this area.

One of the initial assumptions of this research is that GMM is a suitable fit to model Sinhala speech acoustics. Further, it is taken as a key objective of this research to achieve a successful outcome from that model. GMM with fMLLR feature transform yields an error rate as low as $10.91\%$, proving the assumption holds for the Sinhala language and achieving the objective as well. However, various transforms such as LDA, MLLT are applied to feature space and then decoder outputs are compared to find the most accurate training model. Many trial and error iterations are taken to fine tune and test various parameters on the model as described in chapter 4.

Priori functions or language models play an essential role in the general equation of ASR pipeline. Although there are various lexical models available for popular Latin languages, there are almost none for non-Latin languages like Sinhala. One of the goals is to build a lexical model for this specific task at minimum to complete the equation and receive outputs to study. Many tools with comprehensive documentation and features are investigated to create a good lexical model for Sinhala digit sequences. SRILM[36] is chosen as the toolkit to develop a language model as it has a wide variety of features attached with readable documentation.

Primary objective of this research is to build a software tool that aggregates all functional blocks along the ASR pipeline and produces a hypothesis for any given Sinhala spoken digit sequence. A python program is developed to capture an audio stream, send it to the online decoding program and score the hypothesis against a given ground truth.

In conclusion, this study has been able to arrive at all its milestones with an online Sinhala

transcription tool as the final outcome of this project.

## 6.2 Risks and Limitations

A moderate vocabulary of Sinhala digit corpus is used as the data to train all models discussed in the earlier chapter. Sinhala speech transcription is a data analytic problem and it requires an annotated speech corpus to trial out various models in the literature. However, the required corpus is created by the researcher with the help of a group of 35 people. Therefore, this transcription tool can decode only the continuous number sequences spoken by native Sinhala speakers in Sri Lanka.

Some tools such as the SRILM[36] language modeling toolkit is not a complete free tool. Although it has a comprehensive feature catalogue and a documentation, most parts of it are not free for any use. Kaldi[40] toolkit is wriiten in C++ and all its aggregate functions are exposed using bash scripts. Therefore, it makes it a bit difficult when integrating Kaldi online decoder with the Python demo program. However, the stable solution is to deploy Python inter-process communication module to interact with bash scripts.

Python demo program, that basically transcribes Sinhala spoken streams, is a command line tool rather than an interactive GUI. When an audio stream is handed over to the online decoder to transcribe, hypothesis and error rates are printed on the console application in different identifiable log formats.

Understanding Kaldi coding style, I/O internals, data structures and process communication styles is a tedious task compared to frameworks, that are written in higher level languages such as Java, because the Kaldi core is written in C++ and executable programs are consolidated in bash scripts which are from little to no readable. However, the performance of the Kaldi tools are far ahead than that of Java tools such as CMUSphinx.

Kaldi tools support CUDA processing and other distributed parallel processing such as Grid Engine. However, all the testing and debugging is done on a very limited processing environment with four CPUs keeping in mind only the WER besides the latency. The GMM based models that are tested do not require or benefit from parallel processing at large. But the latency of training and decoding can be reduced by a parallel processing environment.

## 6.3 Lessons Learned

This section discusses the important lessons learned throughout this course of study to implement a Sinhala transcription tool. Those aspects help greatly ti improve the quality of the study as well as the output of the study.

### 6.3.1 Sinhala Speech Transcription and its Importance

Sinhala being the widely spoken language among the majority of native Sri Lankans, a Sinhala speech transcription tool can have wide variety of applications in day-to-day life. Many industries such as telecommunication, health greatly benefits from this tool in their command and control applications, voice call and IVR applications and etc. The hearing impaired community can also be benefited from this tool in their education programs and other regular activities.

### 6.3.2 Use of Community Supported Toolkits

Building models, graphs and auxiliary function from ground up takes a considerable amount of time. Therefore, readily available toolkits and frameworks can greatly influence the rapid development from an idea to an implementation. However, the tools and frameworks should be supported by the respective community and the codebase should also be highly manageable. Kaldi is the best toolkit available to begin with this study because of it adequately large community base and online support via Google groups.

### 6.3.3 Practical Difficulties

From the beginning, it was very challenging because getting exposed to the unfamiliar domain knowledge, designing a proper ASR system requires a lot of background study. In tha latter part of the study, data collection and proper evaluation require a considerable amount of time as it needs well defined plans.

### 6.3.4 Target Audience and Design Usability

Main objective of this research is to implement an online Sinhala transcription tool that can be used in telecommunication applications such as voice mail prompts, call center applications and

etc. Since the ASR tool mostly focuses on the backend operations, an interactive GUI is not required.

### 6.3.5 Best Practices

Although it was challenging at the beginning to grasp Kaldi[27] coding style, it has adapted the best practices in C++ development. Even though the learning curve is steep in that area, it helps greatly to get familiar with the best practices in C++ development.

## 6.4 Future Work

This section focuses on the continuity and expansions of this study to fill all gaps and achieve a full featured Sinhala transcription tool. This report is composed in such a way that following future work is encouraged.

### 6.4.1 Machine Learning for ASR

GMM is the acoustic model that is tested under various transformations in this research. Bidirectional Long Short Term Memory (BLSTM) being the state-of-the-art modeling technique based on CNN, findings of this research can be extended to obtain a more sophisticated transcription tool. Parallel computing environments are required to solve this ASR problem using machine learning techniques.

### 6.4.2 Large Annotated Speech Corpus

A moderate vocabulary is chosen to validate the research hypothesis and address the usecases mentioned in this study. A large vocabulary that encompasses the complete Sinhala lexicon must be used to address all possible spoken data, native and non-native. It is a challenging task to construct such a large annotated corpus for a new language such as Sinhala. However, there are many resources such as call center logs are available at disposal.

### 6.4.3 Availability of an Online Decoder

An online decoder for Sinhala speech can be made available via web APIs to crowdsource the data collection activities and annotation operations. Many resourced languages have similar tools that can expedite the training tasks and then produce models with very low error rates.

# References


[1] R. K. Aggarwal and M. Dave. Acoustic modeling problem for automatic speech recognition system: advances and refinements (part ii). *International Journal of Speech Technology*, 14(4):309, Aug 2011.

[2] R. K. Aggarwal and M. Dave. Acoustic modeling problem for automatic speech recognition system: conventional methods (part i). *International Journal of Speech Technology*, 14(4):297, Sep 2011.

[3] L. Akinobu. Open-source large vocabulary csr engine julius. `http://julius.osdn.jp/en_index.php`.

[4] C. Allauzen and M. Riley. Openfst. `http://www.openfst.org/`.

[5] M. Benzeghiba, R. De Mori, O. Deroo, S. Dupont, T. Erbes, D. Jouvet, L. Fissore, P. Laface, A. Mertins, C. Ris, et al. Automatic speech recognition and speech variability: A review. *Speech communication*, 49(10):763–786, 2007.

[6] I. contributors. Arpabet. `https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Arpabet.html`.

[7] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, 28(4):357–366, 1980.

[8] H. Dudley, R. Riesz, and S. Watkins. *A synthetic speaker*. 1939.

[9] Y. Fujii, K. Yamamoto, and S. Nakagawa. Automatic speech recognition using hidden conditional neural fields. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5036–5039. IEEE, 2011.

[10] S. Furui. Fifty years of progress in speech and speaker recognition. *The Journal of the Acoustical Society of America*, 116(4):2497–2498, 2004.

[11] M. Gales and S. Young. The application of hidden markov models in speech recognition. *Found. Trends Signal Process.*, 1(3):195–304, Jan. 2007.

[12] W. Ghai and N. Singh. Literature review on automatic speech recognition. *International Journal of Computer Applications*, 41(8), 2012.

[13] T. Hain, L. Burget, J. Dines, G. Garau, M. Karafiat, D. Van Leeuwen, M. Lincoln, and V. Wan. The 2007 ami (da) system for meeting transcription. *Multimodal Technologies for Perception of Humans*, pages 414–428, 2008.

[14] ICTA. Government information center. `http://www.gic.gov.lk/gic/`.

[15] K. J. IIET. Connected digits recognition using distance calculation at each digit.

[16] F. Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):67–72, 1975.

[17] P. Lamere, P. Kwok, W. Walker, E. B. Gouvêa, R. Singh, B. Raj, and P. Wolf. Design of the cmu sphinx-4 decoder. In *INTERSPEECH*. ISCA, 2003.

[18] C. J. Leggetter and P. C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer speech & language*, 9(2):171–185, 1995.

[19] L. T. R. L. (LTRL). Subasa transliterator. `http://translate.subasa.lk/SiEn_Transliterator/`.

[20] A.-r. Mohamed, G. E. Dahl, and G. Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22, 2012.

[21] T. Nadungodage and R. Weerasinghe. Continuous sinhala speech recognizer. In *Conference on Human Language Technology for Development, Alexandria, Egypt*, pages 2–5, 2011.

[22] T. Nadungodage, V. Welgama, and R. Weerasinghe. Developing a speech corpus for sinhala speech recognition. In *ICON-2013: 10th International Conference on Natural Language Processing*, 2013.

[23] T. O'Donnell, J. Simmers, H. Southall, and T. Klemas. Neural beamforming for signal detection and location. In *Military Communications Conference, 1994. MILCOM'94. Conference Record, 1994 IEEE*, pages 326–330. IEEE, 1994.

[24] M. Ostendorf, V. V. Digalakis, and O. A. Kimball. From hmm's to segment models: A unified view of stochastic modeling for speech recognition. *IEEE Transactions on speech and audio processing*, 4(5):360–378, 1996.

[25] J. Padrell-sendra. Support vector machines for continuous speech recognition. In *in Proc. of the 14th European Signal Processing Conference*, 2006.

[26] D. Povey. Kaldi. `http://kaldi-asr.org/`.

[27] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011. IEEE Catalog No.: CFP11SRW-USB.

[28] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011. IEEE Catalog No.: CFP11SRW-USB.

[29] J. Psutka, L. Müller, and J. V. Psutka. Comparison of mfcc and plp parameterizations in the speaker independent continuous speech recognition task. In *Seventh European Conference on Speech Communication and Technology*, 2001.

[30] L. Rabiner and B. Juang. *Fundamentals of Speech Recognition*. Prentice-Hall Signal Processing Series: Advanced monographs. PTR Prentice Hall, 1993.

[31] L. R. Rabiner, R. W. Schafer, et al. Introduction to digital speech processing. *Foundations and Trends® in Signal Processing*, 1(1–2):1–194, 2007.

[32] A. Samouelian. Knowledge based approach to speech recognition. 12 1997.

[33] C. Sivaranjani and B. Bharathi. Syllable based continuous speech recognition for tamil language. *Int J Adv Engg Tech/Vol. VII/Issue I/Jan.-March*, 1:04, 2016.

[34] J. Sorensen and C. Allauzen. Unary data structures for language models. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[35] M. Sundermeyer, M. NuSSbaum-Thom, S. Wiesler, C. Plahl, A. E.-D. Mousa, S. Hahn, D. Nolden, R. Schlüter, and H. Ney. The rwth 2010 quaero asr evaluation system for

english, french, and german. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2212–2215, May 2011.

[36] S. S. Technology and R. Laboratory. Srilm - the sri language modeling toolkit. `http://www.speech.sri.com/projects/srilm/`.

[37] R. Thangarajan, A. Natarajan, and M. Selvam. Word and triphone based approaches in continuous speech recognition for tamil language. *WSEAS transactions on signal processing*, 4(3):76–86, 2008.

[38] C. M. University. The cmu pronouncing dictionary. `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`.

[39] M. D. Wachter, M. Matton, K. Demuynck, P. Wambacq, R. Cools, and D. V. Compernolle. Template-based continuous speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 15:1377–1390, 2007.

[40] A. Wasala and K. Gamage. Research report on phonetics and phonology of sinhala. 10 2017.

[41] A. Wasala, R. Weerasinghe, and K. Gamage. Sinhala grapheme-to-phoneme conversion and rules for schwa epenthesis. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 890–897, Sydney, Australia, July 2006. Association for Computational Linguistics.

[42] H. You. *Robust automatic speech recognition algorithms for dealing with noise and accent*. University of California at Los Angeles, 2009.

[43] S. J. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. *The HTK Book Version 3.4*. Cambridge University Press, 2006.

[44] H. Zen. Deep learning in speech synthesis. In *SSW*, page 309, 2013.

[45] P. Zhan and A. Waibel. Vocal tract length normalization for large vocabulary continuous speech recognition. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 1997.

# Appendix A WER Algorithm

Following algorithm is used in scoring program of the Python demo program. This is the Python implementation of pseudocode 2.

```
# -*- coding: utf-8 -*-
"""
Created on Thu Mar 22 21:06:44 2018

@author: chamikad
"""

# reference
r = '15␣2x␣5␣3x␣5␣4x␣5␣5x␣5␣6x␣5␣7x␣5␣8x␣5␣9x␣5'.split()

#hypothesis
h = '15␣2x␣6␣6␣3x␣5␣4x␣5␣5x␣5␣6x␣5␣7x␣5␣8x␣5␣9x␣5'.split()



subs = 0         # substitutions
ins = 0          # insertions
dels = 0         # deletions

# initialize D
D = [[0 for x in range(len(h) + 1)] for y in range(len(r) + 1)]

# initialize D
for i in range(len(r) + 1):
    for j in range(len(h) + 1):
        if i == 0:
            D[0][j] = j
        elif j == 0:
            D[i][0] = i



# WER calculation
```

```
for i in range(1,len(r) + 1):
    for j in range(1,len(h) + 1):
        if r[i - 1] == h[j - 1]:
            D[i][j] = D[i - 1][j - 1]
        else:
            subs = D[i - 1][j - 1] + 1
            ins = D[i][j - 1] + 1
            dels = D[i - 1][j] + 1
            D[i][j] = min(subs,ins,dels)



print("WER␣-␣%.2f%s"%(D[len(r)][len(h)]*100/len(r),'%'))
```

## Appendix B Lexicon Generator

Following algorithm is used to generate the lexicon of all possible tokens of the number corpus. This is the Python implementation of generating part of the algorithm 1 [11].

```
# -*- coding: utf-8 -*-
"""
Created on Thu Mar 22 21:06:44 2018

@author: chamikad
"""



finalList = [[] for y in range(1000)]
suffixList = [[] for y in range(1000)]




for i in range(21,1000):
    ldigit = i % 10
    mdigit = (i - ldigit) % 100
    fdigit = (i - ldigit - mdigit) % 1000
```

```
    if fdigit == 0 & ldigit != 0:
        for lphone in finalList[ldigit]:
            for mphone in suffixList[mdigit]:
                finalList[i].append(mphone + "␣" + lphone)

    if fdigit > 0:
        if mdigit > 0 | ldigit > 0:
            for fphone in suffixList[fdigit]:
                for mphone in finalList[i % 100]:
                    finalList[i].append(fphone + "␣" + mphone)
```