A Project Report

On

AUTOMATIC SPEECH RECOGNITION SYSTEM USING KALDI

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

ELECTRICAL ENGINEERING

By

Emmadi Rakesh (EE13B025)

Under esteemed guidance of

Prof. S. Umesh

(2013-2017)

Department of Electrical Engineering

INDIAN INSTITUTE OF TECHNOLOGY MADRAS, CHENNAI

(An Autonomous Institution under MHRD, GOI)

PROJECT CERTIFICATE

This is to certify that the dissertation titled **Automatic Speech Recognition System using Kaldi**, submitted by **Emmadi Rakesh (EE13B025)**, to the Indian Institute of Technology Madras, Chennai in partial fulfillment for the award of the degree of **Bachelor of Technology in Electrical Engineering**, is a bona fide record of the project (B. Tech) work done by him under the supervision of **Prof. S Umesh** during Jan-May 2017 semester. The contents of this dissertation, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. S Umesh
Project Guide
Professor
Dept. of Electrical Engineering
IIT Madras, 600 036

ACKNOWLEDMENTS

First and foremost, I would like to thank my guide **Prof. S Umesh** for guiding me thoughtfully and efficiently through this project, giving me an opportunity to work at my own pace along my own lines, while providing me with very useful directions whenever necessary.

I would also like to thank **K Sandeep** and **Vishwas**, Research Associates for Speech processing lab for being great source for encouragement throughout the length of this project and for providing required speech data.

I also express my thanks to Kaldi community for providing such an excellent open source software.

I offer my sincere thanks to all other persons who knowingly or unknowingly helped me complete this project.

ABSTRACT

The purpose of this project is to learn about both ASR and Kaldi toolkit, to be able to build an Automatic Speech Recognition system. We will study first all the process related to recognize speech automatically to understand how it works, and therefore be able to build a basic system. In addition, we pretend to retrain and adapt the original ASR system to recognize the speech in Indian English accent. As available data, it will be limited by our resources, after evaluate different approaches, we would like to find a commitment between the quality and the data used to build a system easy to handle. We will finally build ASR system based on hmm-gmm and neural networks and evaluate performance respectively. Custom language model will be build based on SRI language modelling for new corpus and incorporate the same in training of new data.

INDEX

SECTION 1

INTRODUCTION TO AUTOMATIC SPEECH RECOGNITION

1.1 Need for Speech Recognition:

Speech recognition is becoming part of our daily lives. Hardware and software system developers, consumer product designers, researchers, and innovative computer users are creating speech recognition applications that range from voice control of dishwashers to meaningful human-computer dialogues. Using Speech Recognition is a comprehensive, unbiased examination of the speech recognition industry. It explains the technology using clear, understandable language and explores differences among existing commercial speech recognition products. Using Speech Recognition describes successful applications along with the technology and human factors involved in designing good speech recognition applications. Most chapters contain a "Technology Focus" section that explains aspects of the technology and an "Application Focus" section that addresses application-development issues.

1.2 Introduction:

The statistical approach to automatic speech recognition aims at modeling the stochastic relation between a speech signal and the spoken word sequence with the objective of minimizing the expected error rate of a classifier. The statistical

paradigm is governed by Bayes' decision rule: given a sequence of acoustic observations x_1 $T= x_1…..x_T$ as the constituent features of a spoken utterance, Bayes' decision rule decided for that word sequence $w_1^N=w_1,\cdots,w_N$ which maximizes the class posterior probability Provided that the true probability distribution is used, Bayes' decision rule is optimal among all decision rules, that is, on average it guarantees the lowest possible classification error rate. However, for most pattern recognition tasks the true probability distribution is usually not known but has to be replaced with an appropriate model distribution. In automatic speech recognition, the generative model, which decomposes the class posterior probability into a product of two independent stochastic knowledge sources, became widely accepted:

$$P(w_1^N|x_1^T)=\frac{P(x_1^T)\cdot(P(x_1^T|w_1^N))}{P(x_1^T)}$$

The denominator $P(x_1^T)$ in above equation is assumed to be independent of the word sequence w_1^N and hence, the decision rule is equivalent to:

$$P(w_1^N|x_1^T)=P(x_1^T)\cdot(P(x_1^T|w_1^N))$$

The word sequence $[w_1^N]$ opt which maximizes the posterior probability is determined by searching for that word sequence which maximizes the product of the following two stochastic knowledge sources:

• The acoustic model P (x_1^T | w_1^N) which captures the probability of observing a sequence of acoustic observations x_1^T given a word sequence w_1^N.

• The language model $P(w_1^N)$ which provides a prior probability for the word sequence w_1^N.

1.3 Signal modelling:

1.3.1 Introduction:

Parameterization of an analog speech signal is the first step in the speech recognition process. Several popular signal analysis techniques have emerged as de facto standards in the literature. These algorithms are intended to produce a "perceptually meaningful" parametric representation of the speech signal: parameters that emulate some of the behavior observed in the human auditory and perceptual systems. Of course, and perhaps more importantly, these algorithms are also designed to maximize recognition performance.

1.3.2 Motivation for signal modelling:

1. To obtain the perceptually meaningful parameters i.e. parameters which are analogous to those used by human auditory system

2. To obtain the invariant parameters i.e. parameters which are robust to variations in channel, speaker and transducer.

3. To obtain parameters that capture spectral dynamics, or changes of spectrum with time.

The signal modeling involves four basic operations: spectral shaping, feature extraction, parametric transformation, and statistical modeling. Spectral shaping is the process of converting the speech signal from sound pressure wave to a digital signal; and emphasizing important frequency components in the signal. Feature extraction is process of obtaining different features such as power, pitch, and vocal tract configuration from the speech signal. Parameter transformation is the process of converting these features into signal parameters through process of differentiation and concatenation. Statistical modeling involves conversion of parameters in signal observation vectors.

1.3.3 Spectral Shaping:

Spectral shaping involves two basic operations: digitization i.e. Conversion of analog speech signal from sound pressure wave to digital signal; and digital filtering i.e. Emphasizing important frequency components in the signal. This process is shown in Fig.2.1.

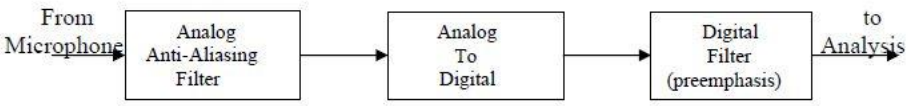


Fig 1.1

The main purpose of digitization process is to produce a sampled data representation of speech signal with as high signal-to-noise ratio (SNR) as possible. Once signal conversion is complete, the last step of digital post filtering is most often executed using a Finite Impulse Response (FIR) filter given as

$$H_{pre}(z) = \sum_{k=0}^{N_{pre}} a_{pre}(k) z^{-k}$$

Normally, a one coefficient digital filter known as pre-emphasis filter, is used

$H_{pre}(z) = 1 + a_{pre}\, z\text{-}1$

A typical range of values for a_{pre} is [-1.0, -0.4]. The pre-emphasis filter boosts the signal spectrum approximately 20 dB per decade.

Advantages of pre-emphasis filter

1. The voiced sections of speech signal naturally have a negative spectral slope (attenuation of approximately 20 dB per decade due to physiology of speech production system). The pre-emphasis filter serves to offset this natural slope before spectral analysis, thereby improving the efficiency of the analysis.

2. The hearing is more sensitive above the 1-kHz region of the spectrum. The pre-emphasis filter amplifies this area of the spectrum. This assists the spectral analysis algorithm in modelling the perceptually important aspects of speech spectrum.

1.3.4 Features Extraction:

In speaker independent speech recognition, a premium is placed on extracting features that are somewhat invariant to changes in the speaker. So feature extraction involves analysis of speech signal. Broadly the feature extraction techniques are classified as temporal analysis and spectral analysis technique. In temporal analysis the speech waveform itself is used for analysis. In spectral analysis spectral representation of speech signal is used for analysis.

a. Mel Frequency Cepstrum Coefficients (MFCC)

The most prevalent and dominant method used to extract spectral features is calculating Mel-Frequency Cepstral Coefficients (MFCC). MFCCs are one of the most popular feature extraction techniques used in speech recognition based on frequency domain using the Mel scale which is based on the human ear scale. MFCCs being considered as frequency domain features are much more accurate than time domain features

Mel-Frequency Cepstral Coefficients (MFCC) is a representation of the real cepstral of a windowed short-time signal derived from the Fast Fourier Transform (FFT) of that signal. The difference from the real cepstral is that a nonlinear frequency scale is used, which approximates the behavior of the auditory system. Additionally, these coefficients are robust and reliable to variations according to speakers and recording conditions. MFCC is an audio feature extraction technique which extracts parameters from the speech similar to ones that are used by humans for hearing speech, while at the same time, deemphasizes all other information. The speech signal is first divided into time frames consisting of an arbitrary number of samples. In most systems overlapping of the frames is used to smooth transition

from frame to frame. Each time frame is then windowed with Hamming window to eliminate discontinuities at the edges.

The filter coefficients w (n) of a Hamming window of length n are computed according to the formula:

$$W(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), 0 \leq n \leq N-1$$
$$= 0, \text{otherwise}$$

Where N is total number of sample and n is current sample. After the windowing, Fast Fourier Transformation (FFT) is calculated for each frame to extract frequency components of a signal in the time-domain. FFT is used to speed up the processing. The logarithmic Mel-Scaled filter bank is applied to the Fourier transformed frame. This scale is approximately linear up to 1 kHz, and logarithmic at greater frequencies. The relation between frequency of speech and Mel scale can be established as:

Frequency (Mel Scaled) = [2595 * log (1+f_{Hz} /700]

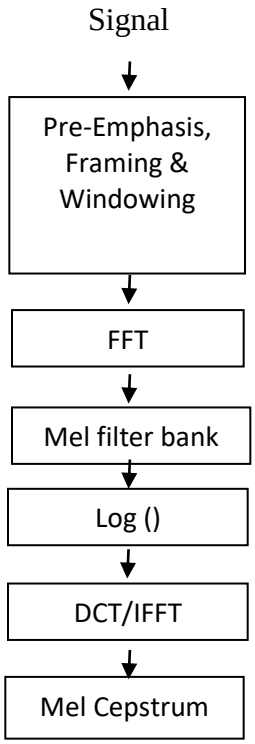


Fig. 1.2

The last step is to calculate Discrete Cosine Transformation (DCT) of the outputs from the filter bank. DCT ranges coefficients according to significance, whereby the 0th coefficient is excluded since it is unreliable. The overall procedure of MFCC extraction is shown in figure 1.2.

For each speech frame, a set of MFCC is computed. This set of coefficients is called an acoustic vector which represents the phonetically important characteristics of speech and is very useful for further analysis and processing in Speech Recognition. We can take audio of 2 Second which gives approximate 128 frames each contain 128 samples (window size = 16 ms). We can use first 20 to 40 frames that give good estimation of speech. Total of forty-Two MFCC parameters include twelve original, twelve deltas (First order derivative), twelve delta-delta (Second order derivative), three log energy and three 0th parameter.

b. Body Linear Predictive Codes (LPC)

It is desirable to compress signal for efficient transmission and storage. Digital signal is compressed before transmission for efficient utilization of channels on wireless media. For medium or low bit rate coder, LPC is most widely used. The LPC calculates a power spectrum of the signal. It is used for formant analysis. LPC is one of the most powerful speech analysis techniques and it has gained popularity as a formant estimation technique.

While we pass the speech signal from speech analysis filter to remove the redundancy in signal, residual error is generated as an output. It can be quantized by smaller number of bits compare to original signal. So now, instead of transferring entire signal we can transfer this residual error and speech parameters to generate the original signal. A parametric model is computed based on least mean squared error theory, this technique being known as linear prediction (LP). By this method, the speech signal is approximated as a linear combination of its p previous samples. In this technique, the obtained LPC coefficients describe the formants. The frequencies at which the resonant peaks occur are called the formant frequencies. Thus, with this method, the locations of the formants in a speech signal are estimated by computing the linear predictive coefficients over a sliding window and finding the peaks in the spectrum of the resulting LP filter. We have excluded 0th coefficient and used next ten LPC Coefficients.

In speech generation, during vowel sound vocal cords vibrate harmonically and so quasi periodic signals are produced. While in case of consonant, excitation source can be

considered as random noise. Vocal tract works as a filter, which is responsible for speech response. Biological phenomenon of speech generation can be easily converted in to equivalent mechanical model. Periodic impulse train and random noise can be considered as excitation source and digital filter as vocal tract.

c. Perceptual Linear Prediction (PLP)

The Perceptual Linear Prediction PLP model developed by Herman sky. PLP models the human speech based on the concept of psychophysics of hearing. PLP discards irrelevant information of the speech and thus improves speech recognition rate. PLP is identical to LPC except that its spectral characteristics have been transformed to match characteristics of human auditory system.

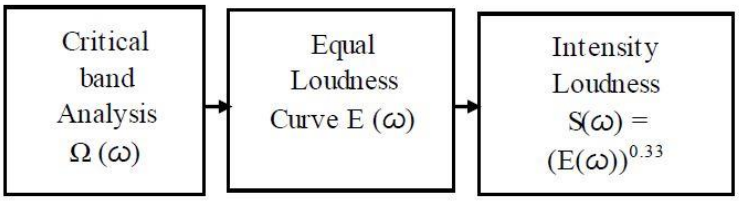


Figure 2.4 shows steps of PLP computation. PLP approximates three main perceptual aspects namely: the critical-band resolution curves, the equal-loudness curve, and the intensity-loudness power-law relation, which are known as the cubic-root.

In this project, we use **MFCC** features extraction method to develop Speech recognition in Kaldi tool kit.

1.3 Acoustic Modelling:

The acoustic model P (x_1^T | w_1^N) provides a stochastic description for the realization of a sequence of acoustic observation vectors x_1^T given a word sequence w_1^N. Due to data sparsely, the model for individual words as well as the model for entire sentences is obtained

by concatenating the acoustic models of basic sub-word units according to a pronunciation lexicon. Sub-word units smaller than words enable a speech recognizer to allow for recognizing words that do not occur in the training data. Thus, the recognition system can ensure that enough instances of each sub-word unit have been observed in training to allow for a reliable estimation of the underlying model parameters.

The type of sub-word units employed in a speech recognizer depends on the amount of available training data and the desired model complexity: while recognition systems designed for small vocabulary sizes (<100 words) typically apply whole word models, systems developed for the recognition of large vocabularies (> 5000 words) often employ smaller sub-word units which may be composed of syllables, phonemes, or phonemes in context. Context-dependent phonemes are also referred to as n-phones. Commonly used sub-word units employed in large vocabulary speech recognition systems are n-phones in the context of one or two adjacent phonemes, so-called triphones or quinphones. Context-dependent phoneme models allow for capturing the varying articulation that a phoneme is subject to when it is realized in different surrounding phonetic contexts.

Typically, the constituent phones for various acoustic realizations of the same word are produced with different duration and varying spectral configuration, even if the utterances are produced by the same speaker. Each phone will therefore aggregate an a-priori unknown number of acoustic observations. The temporal distortion of different pronunciations as well as the spectral variation in the acoustic signal can be described via a Hidden Markov Model (HMM). A HMM is a stochastic finite state automaton that models the variation in the acoustic signal via a two-stage stochastic process. The automaton is defined through a set of states with transitions connecting the states. The probability P ($x_1^T \mid w_1^N$) is extended by an unobservable (hidden) variables representing the states:

$$P\left(w_1^N \middle| x_1^T\right) = \sum_{s_1^T} P\left(x_1^T, s_1^T \middle| w_1^N\right)$$

1.4 Language Modelling:

The language model P (w_1^N) provides a prior probability for the word sequence w_1^N = $w_1 \ldots w_N$. Thus, it inherently aims at capturing the syntax, semantics, and pragmatics of a language. Since language models are independent of acoustic observations, their parameters can be estimated from large text collections as, for instance, newspapers, journal articles, or web content. Due to a theoretically infinite number of possible word sequences, language models require suitable model assumptions to make the estimation problem practicable. For large vocabulary speech recognition, m-gram language models have become widely accepted. An m-gram language model is based on the assumption that a sequence of words follows an (m-1) th order Markov process, that is, the probability of a word w n is supposed to depend only on its m-1 predecessor words.

$$P\left(w_1^N\right)=\prod_{n=1}^{N} P\left(w_n|w_1^{n-1}\right)$$

$$P\left(w_1^N\right) model\,assumption=\prod_{n=1}^{N} P\left(w_n|w_{n-m-1}^{n-1}\right)$$

1.5 Evaluation of ASR

Exist different methods to evaluate the quality of an ASR system. Word Error Rate (WER) is a common metric of the performance of a speech recognition.

The main difficulty of measuring performance lies in the fact that the recognized word sequence can have a different length from the reference word sequence. The WER is derived from the Levenshtein distance, but working at the word level. This problem is solved by first aligning the recognized word sequence with the reference word sequence using dynamic string alignment.

$$WER=\frac{100*(S+I+D)}{N}$$

Where:

-N: Is the number of words in the reference

-S: Is the number of substitutions

-I: Is the number of insertions

-D: Is the number of deletions.

A basic alignment example:

Ref: portable	***	phone	upstairs	last	night	so	***
Hyp: preferable	form	of	stores	next	light	so	far
Eval: S	I	S	S	S	S		I

N = 5, S = 5, I = 2, D = 0 Then WER = 80%

1.7 What is

a. Corpus:

Corpus is a large collection of texts. It is a body of written or spoken material upon which a linguistic analysis is based. The plural form of corpus is corpora. Some popular corpora are British National Corpus (BNC, http://www.natcorp.ox.ac.uk/corpus/index.xml), COBUILD/Birmingham Corpus, and IBM/Lancaster Spoken English Corpus. Monolingual corpora represent only one language while bilingual corpora represent two languages. European Corpus Initiative (ECI) corpus is multilingual having 98 million words in Turkish, Japanese, Russian, Chinese, and other languages. The corpus may be composed of written language, spoken language or both. Spoken corpus is usually in the form of audio recordings. A corpus may be open or closed. An *open corpus* is one which does not claim to contain all data from a specific area while a *closed corpus* does claim to contain all or nearly all data from a particular field. *Historical corpora,* for example, are closed as there can be no further input to an area.

b. Phoneme:

A phoneme is one of the units of sound (or gesture in the case of sign languages, see chereme, https://en.wikipedia.org/wiki/Cherology) that distinguish one word from another in a particular language. The difference in meaning between the English words *kill* and *kiss* is a result of the exchange of the phoneme /l/ for the phoneme /s/. Two words that differ in meaning through a contrast of a single phoneme form a minimal pair (https://en.wikipedia.org/wiki/Minimal_pair).

c. Phone:

A phone is any distinct speech sound or gesture, regardless of whether or not the exact sound is critical to the meanings of words. In contrast, a phoneme (https://en.wikipedia.org/wiki/Phoneme) is a speech sound that, in a given language, if it were swapped with another phoneme, would change the meaning of the word. Phones are absolute, not specific to any language, but phonemes can be discussed only in reference to specific languages.

d. Lexicon:

It is dictionary where each and every word in a language is defined with its phonetic transcriptions. US English lexicon can be found on CMU speech research website. Following figure shows how a lexicon dictionary look like.

```
abrego  AA B R EH G OW
abreu   AH B R UW
abridge AH B R IH JH
abridged    AH B R IH JH D
abridgement AH B R IH JH M AH N T
abridges    AH B R IH JH AH Z
abridging   AH B R IH JH IH NG
abril   AH B R IH L
abroad  AH B R AO D
abrogate    AE B R AH G EY T
```

SECTION 2

INTRODUCTION TO KALDI TOOL KIT

2.1 Introduction:

2.1.1 Abstract:

This section describes the design of Kaldi, a free, open-source toolkit for speech recognition research. Kaldi provides a speech recognition system based on finite-state transducers (using the freely available Openfst), together with detailed documentation and scripts for building complete recognition systems. Kaldi is written is C++, and the core library supports modeling of arbitrary phonetic-context sizes, acoustic modeling with subspace Gaussian mixture models (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.

2.1.2 about Kaldi:

According to legend, Kaldi was the Ethiopian goat herder who discovered the coffee plant. Kaldi1 is an open-source toolkit for speech recognition written in C++ and licensed under the Apache License v2.0. The goal of Kaldi is to have modern and flexible code that is easy to understand, modify and extend. Kaldi is available on SourceForge. The tools compile on the commonly used Unix-like systems and on Microsoft Windows.

Researchers on automatic speech recognition (ASR) have several potential choices of open-source toolkits for building a recognition system. Notable among these are: HTK, Julius (both written in C), Sphinx-4 (written in Java), and the RWTH ASR toolkit (written in C++). Yet, our specific requirements—a finite-state transducer (FST) based framework, extensive linear algebra support, and a non-restrictive license—led to the development of Kaldi. Important features of Kaldi include:

Integration with Finite State Transducers: Kaldi compile against the Openfst toolkit (using it as a library).

Extensive linear algebra support: Kaldi include a matrix library that wraps standard BLAS and LAPACK routines.

Extensible design: Kaldi's attempt to provide its algorithms in the most generic form possible. For instance, kaldi decoders work with an interface that provides a score for a particular frame and FST input symbol. Thus, the decoder could work from any suitable source of scores.

Open license: The code is licensed under Apache v2.0, which is one of the least restrictive licenses available

Complete recipes: Kaldi make available complete recipes for building speech recognition systems, that work from widely available databases such as those provided by the Linguistic Data Consortium (LDC).

2.2 Overview of the tool kit:

A schematic overview of the Kaldi toolkit in figure 2.1. The toolkit depends on two external libraries that are also freely available: one is Openfst for the finite state framework, and the other is numerical algebra libraries. We use the standard "Basic Linear Algebra Subroutines" (BLAS) and "Linear Algebra Package" (LAPACK) 2 routines for the latter. The library modules can be grouped into two distinct halves, each depending on only one of the external libraries (c.f. Figure 2.1). A single module, the Decodable Interface, bridges these two halves. Access to the library functionalities is provided through command-line tools written in C++, which are then called from a scripting language for building and running a speech recognizer. Each tool has very specific functionality with a small set of command line arguments: for example, there are separate executables for accumulating statistics, summing accumulators, and updating a GMM-based acoustic model using maximum likelihood estimation. Moreover, all the tools can read from and write to pipes which makes it easy to chain together different tools.

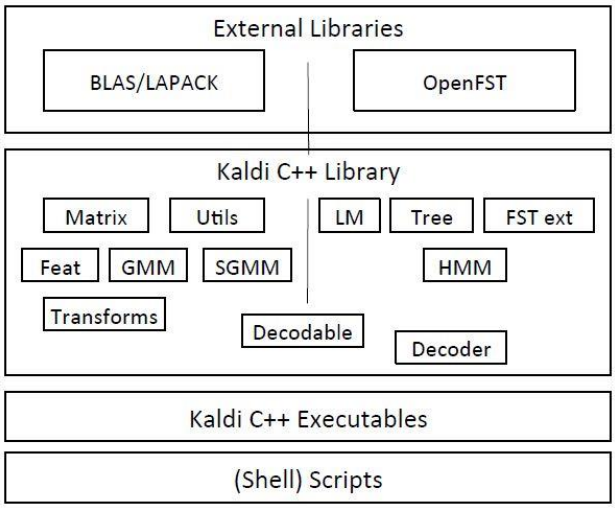


Fig 2.1: A simplified view of the different components of Kaldi

2.3 Features Extraction:

Kaldi feature extraction and waveform-reading code aims to create standard MFCC and PLP features, setting reasonable defaults but leaving available the options that people are most likely to want to tweak (for example, the number of Mel bins, minimum and maximum frequency cutoffs, etc.). Kaldi support most commonly used feature extraction approaches: e.g. VTLN, cepstral mean and variance normalization, LDA, STC/MLLT, HLDA, and so on.

2.4 Acoustic Modelling:

Kaldi supports conventional models (i.e. diagonal GMMs) and Subspace Gaussian Mixture Models (SGMMs), but to also be easily extensible to new kinds of model.

2.4.1 Gaussian Mixture Models:

Kaldi support GMMs with diagonal and full covariance structures. Rather than representing individual Gaussian densities separately, it directly implements a GMM class that is parameterized by the natural parameters, i.e. means times inverse covariance and inverse covariance. The GMM classes also store the constant term in likelihood computation, which consist of all the terms that do not depend on the data vector. Such an implementation is suitable for efficient log-likelihood computation with simple dot-products.

2.4.2 GMM-based acoustic model:

The “acoustic model” class AmDiagGmm represents a collection of DiagGmm objects, indexed by “pdf-ids” that correspond to context-dependent HMM states.

This class does not represent any HMM structure, but just a collection of densities (i.e. GMMs). There are separate classes that represent the HMM structure, principally the topology and transition-modeling code and the code responsible for compiling decoding graphs, which provide a mapping between the HMM states and the pdf index of the acoustic model class. Speaker adaptation and other linear transforms like maximum likelihood linear transform (MLLT) or semi-tied covariance (STC) are implemented by separate classes.

2.4.3 Speaker Adaptation:

Kaldi support both model-space adaptation using maximum likelihood linear regression (MLLR) and feature-space adaptation using feature-space MLLR (fMLLR), also known as constrained MLLR. For both MLLR and fMLLR, multiple transforms can be estimated using a regression tree. When a single fMLLR transform is needed, it can be used as an additional processing step in the feature pipeline. The toolkit also supports speaker normalization using a linear approximation to VTLN, similar to, or conventional feature-level VTLN, or a more generic approach for gender normalization which we call the "exponential transform". Both fMLLR and VTLN can be used for speaker adaptive training (SAT) of the acoustic models.

2.4.4 Subspace Gaussian Mixture Models:

For subspace Gaussian mixture models (SGMMs), the toolkit provides an implementation of the approach described in. There is a single class AmSgmm that represents a whole collection of pdf's; unlike the GMM case there is no class that represents a single pdf of the SGMM. Similar to the GMM case, however, separate classes handle model estimation and speaker adaptation using fMLLR.

2.5 Language Modelling:

Since Kaldi uses an FST-based framework, it is possible, in principle, to use any language model that can be represented as an FST. Kaldi provide tools for converting LMs in the standard ARPA format to FSTs. In its recipes, kaldi have used the IRSTLM toolkit 3 for purposes like LM pruning. For building LMs from raw text, users may use the IRSTLM toolkit, for which we provide installation help, or a more fully-featured toolkit such as SRILM.

2.6 Training:

All kaldi training and decoding algorithms use Weighted Finite State Transducers (WFSTs). In the conventional recipe, the input symbols on the decoding graph correspond to context-dependent states (in our toolkit, these symbols are numeric and we call them pdf-ids). However, because it allows different phones to share the same pdf-ids, we would have a number of problems with this approach, including not being able to determine the FSTs, and not having sufficient information from the Viterbi path through an FST to work out the phone sequence or to train the transition probabilities. In order to fix these problems, kaldi put on the input of the FSTs a slightly more fine-grained integer identifier that we call a "transition-id", that encodes the pdf-id, the phone it is a member of, and the arc (transition) within the topology specification for that phone. There is a one-to-one mapping between the "transition-ids" and the transition-probability parameters in the model: we decided make transitions as fine-grained as it could without increasing the size of the decoding graph.

Kaldi decoding-graph construction process is based on the recipe described in; however, there are a number of differences. One important one relates to the way we handle "weight-pushing", which is the operation that is supposed to ensure that the FST is stochastic. "Stochastic" means that the weights in the FST sum to one in the appropriate sense, for each state (like a properly normalized HMM). Weight pushing may fail or may lead to bad pruning behavior if the FST representing the grammar or language model (G) is not stochastic, e.g. for back off language models. Kaldi's approach is to avoid weight-pushing altogether, but to ensure that each stage of graph creation "preserves stochasticity" in an appropriate sense. Informally, what this means is that the "non sum- to-one-ness" (the failure to sum to one) will never get worse than what was originally present in G.fst.

2.7 Decoding:

Kaldi has several decoders, from simple to highly optimized; more will be added to handle things like on-the-fly language model rescoring and lattice generation. By "decoder" we mean a C++ class that implements the core decoding algorithm. The decoders do not require a particular type of acoustic model: they need an object satisfying a very simple interface with a function that provides some kind of acoustic model score for a particular (input-symbol and frame).

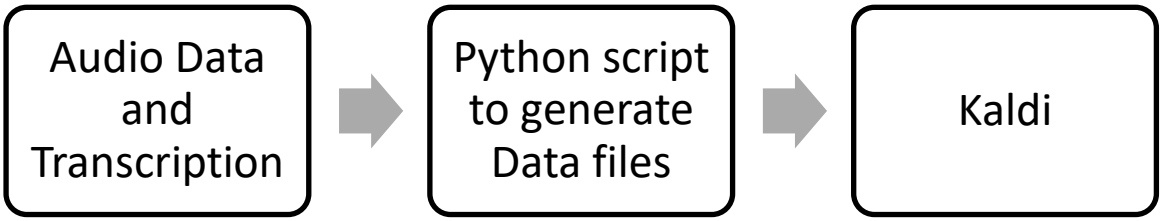
```
class DecodableInterface {
```

```
public:

virtual float LogLikelihood(int frame, int index) = 0;

virtual bool IsLastFrame(int frame) = 0;

virtual int NumIndices() = 0;

virtual ˜DecodableInterface() {}

};
```

Command-line decoding programs are all quite simple, do just one pass of decoding, and are all specialized for one decoder and one acoustic-model type. Multi-pass decoding is implemented at the script level.

SECTION 3

DATA REPRESENTATION IN KALDI



Data Files:
wav.scp: <utterance id> <path>
utt2spk: <utterance id> <utterance id>
text: <utterance id> <transcription>
*utterance id is name of audio file

a) wav.scp: This file connects every utterance (sentence said by one person during particular recording session) with an audio file related to this utterance. Here 'utteranceID' is nothing more than 'speakerID' glued with audio file name without '.wav' ending (look for example below).

```
PATTERN: <utteranceID> <full_path_to_audio_file>
------------ example wav.scp starts -------------
Ramu_1_2_5 /home/{user}/kaldi-
trunk/egs/example/s5/audio_data/train/ramu/1_2_5.wav
Seetha_2_8_5 /home/{user}/kaldi-
trunk/egs/example/s5/audio_data/train/Seetha/2_8_5.wav
Bharath_4_9_5 /home/{user}/kaldi-
trunk/egs/example/s5/audio_data/train/Bharath/4_9_5.wav
Ravan_4_1_7 /home/{user}/kaldi-
trunk/egs/example/s5/audio_data/train/Ravan/4_1_7.wav
# and so on ……
------------ example wav.scp ends ------------------
```

b) text: This file contains every utterance matched with its text transcription.

```
PATTERN: <uterranceID> <text_transcription>
------------ example text starts -------------------
Ramu_1_2_5 one two five
Seetha_6_8_3 six eight three
Ravan_4_4_2 four four two
# and so on...
------------ example text ends --------------------
```

c) utt2spk: This file tells the ASR system which utterance belongs to particular speaker

```
PATTERN: <uterranceID> <speakerID>
------------- example utt2spk starts ---------------
Ramu_1_2_5 Ramu
Seetha_6_8_3 Seetha
Ravan_4_4_2 Ravan
# and so on...
------------- example utt2spk ends -----------------
```

3.1 Script to generate data files:

Kaldi requires data to be submitted in specific format in form of files. We've written python script to generate those data files.

```
#Python script to generate data files
#Author: Emmadi Rakesh

import os
import io
import sys
from subprocess import call

wavscpLines = []
textLines = []
idLines = []
corpusLines = []
os.chdir("Bangalore_English_Transcription")
call(["rm","id","wav.scp","text","corpus.txt","utt2spk"])
for dir0 in os.listdir(os.getcwd()):
        currDir0 = os.getcwd();
        os.chdir(dir0)
        for dir1 in os.listdir(os.getcwd()):
                currDir1 = os.getcwd();
                os.chdir(dir1)
                for fil in [doc for doc in os.listdir(os.getcwd()) if
doc.endswith(".wav")]:
                        transcriptFile = open(fil[:-4]+".txt", 'r')
                        trans = transcriptFile.readlines()[0]+"\n"
                        idLines.append(fil[:-4])
                        corpusLines.append(trans)
                        textLines.append(fil[:-4] + " "+ trans)
                        wavscpLines.append(fil[:-4]+"
"+str(os.getcwd())+"/"+fil[:-4]+".wav"+"\n")
                os.chdir(currDir1);
        os.chdir(currDir0);

wavscp = io.FileIO("wav.scp","w")
text = io.FileIO("text","w")
corpus = io.FileIO("corpus.txt","w")
id = io.FileIO("id","w")
utt2spk = io.FileIO("utt2spk","w")

for line in sorted(wavscpLines):
        wavscp.write(line)
for line in sorted(textLines):
        text.write(line)
for line in sorted(idLines):
        id.write(line+"\n")
        utt2spk.write(line+" "+line+"\n")
for line in corpusLines:
        corpus.write(line)
```

SECTION 4
DATA SETS AND EXPERIMENTS

We have given two data sets to train our model. Every utterance in those data files are spoken in English language. Thus, our model recognizes English language. For our convenience, we named our data sets. Brief description to those datasets are given below:

1. CCC English Data
    - Number of utterances: 12284
    - Approximate Hours: 36

2. New Bangalore English Data
    - Number of utterances: 495
    - Approximate hours: 10.6

4.1 Experiment with CCC English Data:

CCC English data contains audio files extracted from phone calls. It has various speakers from Tamil Nadu, Chhattisgarh and Jharkhand.

Train: 9180 utterances (approx. 26.8 hours)

Test: 3104 utterances (approx. 9 hours)

A research associate from speech lab had already built a model on this data based on gmm-hmm. Word Error Rate (WER) achieved was **20.2**. They used Tri2 training and decoding. Tri2 is based on LDA (Linear Discriminant Analysis) and MLLT (Maximum Likelihood Linear Transforms) algorithms.

We tried to implement neural networks based training and decoding on same train and test set. We used scripts from nnet2 from Kaldi. GPU clusters are required for efficient training and decoding. WER achieved is **18.14** which is lower than past model.

4.2 Experiment with Bangalore English Data:

Bangalore English data contains audio files recorded through microphone. It has various speakers uttered context based on modern technology. As name suggests speakers are exclusively from Bangalore. We worked predominantly on building model for this data.

4.2.1 Data description:

- No of files: 495 (approx. 10.6 hours)
- Format: wave
- Mean duration: 76.97 secs

As we observe mean duration of this data is more than a minute. We need to change some parameters in source shell scripts of kaldi. We'll get back to it in training section.

4.2.2 Creating Data files:

Python script described in section 3 is used to generate data files such as wav.scp, utt2spk and text. These files are must to train the model. By using sublime text software empty lines in those files are removed.

4.2.3 Language modelling:

There is no language model available for this data. We've to prepare our own language model based on train corpus. We used SRI language model. But the problem is there are lot words in corpus whose phonetic transcription is missing. We used Logios tool provided by Carnegie Mellon University (CMU) to find phonetic transcription for unknow words. We used phones set provided by CMU.

4.2.4 Features Extraction:

Out of total 495 audio files we are able to generate MFCC features for only 399 audio files. Remaining 96 files are throwing "RIFF header missing" error. Solution we found is to re-encode those wav files using ffmpeg tool.

4.2.5 Training:

As discussed in 4.2.1 section mean duration of each audio file is more than one minute. We have to change "beam" parameter in steps/mono_train.sh file to 55. We implemented mono, tri1 and tri2 training to build our model.

4.2.6 Decoding:

Since test files are more than a minute, it is taking lot of time to decode. So, to test the performance of the model we choose 379 files for train and 20 files to decode. Test data corpus is not included in language modelling.

4.3 Results:

Results for each experiment are tabulated below.

Data	Model	WER	Insertions	Deletions	Substitutions
CCC English Data	Neural networks	18.17% (2211/12166)	561	533	1117
Bangalore English Data	Tri1	28.15 % (1064/3780)	134	134	796
Bangalore English Data	Tri2	26.27% (993/3780)	125	115	753

SECTION 5
CONCLUSION, LIMITATION AND FUTURE SCOPE OF THE PROJECT

5.1 Conclusion:

One who has better knowledge of shell scripting will able to develop a simple Automatic Speech Recognition system in Kaldi. All example scripts are written in bash. By experiment described in section 4.2, one can able to develop ASR to recognize any language provided proper phonetic dictionary and decent amount of speech data.

5.2 Project Limitations:

We need to have large amount of speech data recorder from various speakers to develop a perfect speech recognition model. It is highly impossible to decode words that are not trained. Some manual work is needed to filter corpus. One has to invest lot of time to decode large audio files.

5.3 Future Scope:

We can develop Large Vocabulary Continuous Speech Recognition (LVCSR) system of any language provided proper lexicon dictionary with accurate phonetic transcriptions and

large amount of data. For better we can implement Deep Neural Networks (DNN) in Kaldi which is cost function optimization technique than the one which we used, Gaussian Mixture Models (GMM) is of probabilistic estimation. We can develop real time speech recognizer by using online kaldi recipe based on online latgen recognizer.

[https://github.com/kaldi-asr/kaldi/blob/master/egs/vystadial_cz/online_demo/pykaldi-online-latgen-recogniser.py]

SECTION 6
REFERENCES

[1] Speech and language processing by Jurafsky and Martin (book)

[2] Kaldi website (http://kaldi-asr.org/)

[3] Feature extraction methods LPC, PLP and MFCC in Speech recognition by Namrata Dave (paper)

[4] Feature Extraction for Speech Recognition by Manish P.Kesarkar (paper)

[5] Signal Modeling Techniques in Speech Recognition by Joseph Picone (paper)

[6] Speech Recognition Wiki page (https://en.wikipedia.org/wiki/Speech_recognition)

[7] A Gaussian Mixture Model Spectral Representation for Speech Recognition by Matthew Nicholas Stuttle (thesis)

[8] The Application of Hidden Markov Models in Speech Recognition by Mark Gales and Steve Young (paper)

[9] Logios tool (http://svn.code.sf.net/p/cmusphinx/code/trunk/logios/Tools/)

[10] CMU lexicon dictionary (http://svn.code.sf.net/p/cmusphinx/code/trunk/cmudict/)

[11] FFmpeg tool (https://ffmpeg.org/)

[12] Voxforge repository

(http://www.repository.voxforge1.org/downloads/SpeechCorpus/Trunk/Audio/Original/16kHz_16bit/)

[13] Speech Recognition with weighted finite-state transducers by Mehryar Mohri, Fernando Pereira and Michael Riley (paper)