**B.Tech. Project Report**

**COT-415**

**on**

**MEL FILTERBANK FEATURE ANALYSIS FOR SPEECH RECOGNITION**

**BY**

**KAPIL YADAV (11510167)**

**MANAS SWARGIARY (11510182)**

**MOHINI AGARWAL (11520015)**

**Under the Supervision of**

**Vishal Passricha, (Ph.D. Scholar, NITK)**



**DEPARTMENT OF COMPUTER ENGINEERING**

**NATIONAL INSTITUTE OF TECHNOLOGY**

**KURUKSHETRA – 136119, HARYANA (INDIA)**

**July – Dec, 2018**

****

**CERTIFICATE**

We therefore declare that this work which being presented in this B.Tech. Minor Project (COT-415) report entitled “**MEL FILTERBANK FEATURES ANALYSIS FOR SPEECH RECOGNITION”,** in requirements for the award of the **Bachelor of Technology in Computer Engineering** is an authentic record work performed during the period from July 2018 to December 2018 under the supervision of **Vishal Passricha (**Ph.D. Scholar, NITK) Computer Engineering Department.

The matter presented in this project report has not been submitted for the award of any other degree elsewhere.

*Signature of Candidates*

**KAPIL YADAV (11510167)**

**MANAS SWARGIARY (11510182)**

**MOHINI AGARWAL (11520015)**

This is to certify that the above statements made by the mentioned candidates is correct as per my knowledge.

Date:  *Signature of Supervisor*

**Vishal Passricha**

**(Ph.D. Scholar, NITK)**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Section No.** | **TITLE** | **Page no.** |
|  | ABSTRACT | 1 |
| I | INTRODUCTION | 3 |
| II | MOTIVATION | 4 |
| III | LITERATURE SURVEY | 6 |
|  | II.1 FRONT END SIGNAL PROCESSING | 6 |
|  | II.2 BACK – END CLASSIFICATION | 7 |
| IV | AUTOMATIC SPEECH RECOGNITION BACKGROUND | 9 |
| V | MEL FILTERBANK SPECTRAL COEFFIECIENTS | 13 |
| VI | CONVOLUTIONAL NEURAL NETWORK | 14 |
| VII | TIMIT | 16 |
| VIII | DATA FLOW DIAGRAM | 19 |
|  | VI.1 Level 0 DFD | 19 |
|  | VI.2 Level 1 DFD | 19 |
|  | VI.3 Level 2 DFD | 20 |
| VII | RESULTS | 21 |
| VIII | CONCLUSION | 22 |
|  | REFERENCES | 23 |
| APPENDIX: | |  |
| A | COMPLETE CONTRIBUTARY SOURCE CODE | 24 |

**ABSTRACT**

Conventionally, Mel-frequency Cepstral Coefficients (MFCC) is preferably used in automatic speech recognition (ASR) systems for feature extraction but these features show limited performance in CNN-based ASR systems. The main reason for degradation in performance is the absence of locality in the acoustic features. The performance of the CNN-based system can be improved by adding locality component in the features. Log Mel-filterbank features show the locality component in their features due to the absence of discrete cosine transformation. In this work, a CNN-based ASR system is designed that uses log Mel filterbank features. It is a good substitute for MFCC. We will compare the recognition rate of both kinds of features results. Kaldi, an open-source toolkit for speech recognition will be used to implement on TIMIT corpus.

**ENVIRONMENT**

1) Kaldi

Kaldi is a speech recognition unit written on C ++ and approved under Apache License v2.0. Kaldi aims to be used by researchers to identify the statement. It is a WFST-based toolkit created by Daniel Povey. It was originally born at the JHU speech workshop in 2009, and some young people from the Brno University. Kaldi is specifically designed for research based recognition.

2) Linux

An UNIX like free and open source community developed Operating System licensed under GNU. A general OS platform for development.

3) C++

Preferred general purpose programming language for application development at front end, coding and implementation.

4) TIMIT

TIMIT is a collection of speeches and phrasal verbs of American English speakers of the opposite sex and dialects. Each recorded element was described in time. TIMIT is designed to enhance the knowledge of phonetics and speech recognition systems. It was staged by “DARPA and the corpus design were a joint effort between Massachusetts Institute of Technology (MIT), SRI International, and Texas Instruments (TI)”. At TI it was recorded, at MIT it was transcribed, and National Institute of Standards and Technology (NIST) verified and prepared for publication by the National Institute of Standards and Technology (NIST). TIMIT and NTIMIT are not freely available - either as a member of the Linguistic Data Consortium, or as a cash payment, is required to access the data.

**INTRODUCTION**

Automatic Speech Recognition is a technology based on artificial intelligence that enables the automatic translation and recognition of spoken language into text by computers. Human use their knowledge for understanding the utterance while listening to it. But for computer we only have the speech signal. So, to enable a computer to translate and recognise speech require a lot of pre-processing and training. There are different approaches to achieve the end result.

Both acoustic modeling and language modeling are important parts of modern statistically-based speech recognition algorithms. Modern general-purpose speech recognition systems are based on Hidden Markov Models (HMM). These are statistical models that output a sequence of symbols or quantities.

The primary purpose of this project is to learn about both the ASR system working and Kaldi toolkit. It will make us able to build an Automatic Speech Recognition system using Mel-filter bank features for feature extraction and to analyse its performance.

First, we will examine all the processes related to the automatic recognition of the language, to understand how it works and, therefore, to be able to build a basic system.

Additionally, original ASR system is retrained and repurposed to improve recognition precision. Since the available data is limited by our resources, after evaluating different approaches, we want to find a “compromise between the quality and the data used to build an easy-to-use system”.

Automatic Speech Recognition (ASR) in a spoken dialogue system converts speech to text so that the dialogue system is able to extract semantic meaning from the text. Dialogue systems are able to exploit multiple alternative text hypotheses. State-of-the-art speech recognisers are able to extract multiple hypotheses in real time. We prefer extracting multiple hypotheses in form of a word lattice because it eﬃciently represents multiple hypotheses. The Alex dialogue system had used the HTK toolkit and the OpenJulius lattice speech recogniser to train acoustic models and to decode lattices in real time respectively. Unfortunately, OpenJulius crashes when extracting lattices. Fixing OpenJulius’s complicated source code seemed unrealistic due to lack of documentation and community support. As a result, we decided abandon OpenJulius and HTK. As an alternative, we decided to use the Kaldi toolkit because its speech recognisers are able to produce high-quality lattices and are suﬃciently fast1 for real-time recognition. In addition, the Kaldi toolkit is actively maintained, and is distributed under the permissive Apache 2.0 license2. We still need to implement a speech recogniser which supports incremental speech processing, prepare acoustic modelling scripts and evaluate the developed recogniser, so that the Kaldi toolkit can be used in Alex dialogue system.

Modern automatic speech recognition (ASR) systems typically model the “relationship between the acoustic voice signal and the phonemes” in two separate steps that are independently optimized. In a first step, the speech signal is converted into characteristics, which generally consist of a phase of dimensionality reduction and a phase of selection of the information, based on the specific knowledge of the task of the phenomena. These two phases have been carefully crafted by hand, leading to breakthrough properties such as the frequency cepstral coefficients after mel (MFCC) or the “perceptual linear prediction features (PLP)” for cepstral. In a second step, the likelihood of sub-word units, such as phonemes, is estimated using generative models or discriminative models. In recent years, there has been a growing interest in the hybrid HMM / ANN framework in the use of "intermediate" representations, rather than conventional features such as, e.g. Cepstral-based features, as an input to systems based on neural networks. Deep-lane networks (DNNs), which can produce a better system than a single hidden MLP layer, have been proposed to address different aspects of acoustic modelling. Specifically, the use of context-dependent phonemes, the use of spectral features as opposed to cepstrum features, the CNN-based system with the energies of melter banks as a combination of inputs of different properties, to name but a few.

In our ASR system, we use Kaldi, a toolkit for speech recognition written in C++ and licensed under the Apache License v2.0. In kaldi toolkit we have options to use different approaches for extracting features from the sample speech signal. The two main features used are MFCC and MFSC. MFCC include computation of DCT as a last step which is avoided in the MFSC feature.

**MOTIVATION**

The field of speech recognition is still open for research as 100% effective system is yet to be developed. Speech being the primary means of human communication can be used to develop natural interfaces for both literate and illiterate users. Nowadays, ASR systems are widely used. Some of the popular ASRs are WhatsApp ASR system, Siri, Closed Caption by YouTube and Google voice etc. They work well in a clean environment but their performance degrades in noisy conditions. Improving the performance of automatic speech recognition (ASR) systems in reverberant environments is still a major challenge. Also, automatic speech recognition will be able to contribute towards preservation of endangered languages.

**LITERATURE SURVEY**

Research in speech recognition began in the late 1950s with the advent of the digital computer. As the ASR has two main components, the review is broadly divided into two categories: front-end signal processing and back-end classification.

**Front-End Signal Processing**

The main objective of the front-end processing is to “obtain a projection of the speech signal to a compact parameter space where the information related to a speech content can be extracted easily”. The important development here, was the concept of cepstral analysis first suggested by Tukey in 1959. From 1960s to 1980s, the cepstral coefficients were derived directly based on the spectrum of speech without applying the filter bank. Modern techniques apply a bank of filters with linear spaced centre frequencies on the power spectrum, as it seems to decrease their variability from the speaker specific noise traits. There are several important cepstral based parameterization techniques introduced in speech processing, namely Linear Prediction Coefficient (LPC) proposed by Atal and Hanauer (1971), Linear Predictive Cepstral Coefficient (LPCC) derived by Atal (1974), Mel Frequency Cepstral Coefficient (MFCC) formulated by Davis and Mermelstein (1980), Perceptual Linear Prediction (PLP) proposed by Hermansky (1990) and the MF-PLP by Woodland *et al.* (1997) . In and later on in other studies, it was demonstrated that the MFCC outperformed the LPC, LPCC, and other speech features on the task of speech recognition. Standard ASR systems primarily use cepstral features which tend to capture the “envelope of short term magnitude spectrum of speech (frequency domain information)”.

**LPC**: Speech sample can be approximated as a linear combination of the past speech samples. LPC is computed to minimize the prediction error. In this technique, human articulatory organs are modelled by signal processing methods.

**MFCC**: First, the speech signal is analyzed with the STFT, then DFT values are grouped together in critical bands and weighted according to the triangular weighting function. This technique is based on human auditory system.

Wavelet based features were introduced for phoneme recognition at the end of 1990s proving their superiority over the typical Fourier transformation based approaches. However, these features did not gain the popularity due to their relatively more sophisticated computation and to the lack of open-source implementation.

**Back-End Classification**

At back-end, the statistical framework of hidden Markov model (HMM) has been used as a dominant approach for classification. It is a doubly stochastic process, generated by two interrelated mechanisms, an underlying Markov chain having a finite number of states, and each of those states is being associated with a specific probability distribution to compute the likelihood of acoustic features. State emission probabilities can be modelled via discrete probability distributions, semi-continuous probability distributions, or continuous probability distributions. Commonly used models for continuous probability distributions are mixture distributions composed of a weighted sum of Gaussian or Laplacian probability density functions.

The basic theory of HMM was published by Baum and its colleagues in the late 1960sand early 1970s, and applied to speech recognition by Jelinek and Baker. But it became popular in the context of ASR after a classical tutorial presented by Rabiner in 1989. An excellent theoretical overview can be found in and a comprehensive discussion on learning and inference in HMM with Bayesian networks is available.

When each HMM condition is associated with a separate set of mixtures, it is known as the continuous HMM. These HMMs have two problems: they are expensive to evaluate and can not be compared to force when the number of surveys in each case in the training data is small. In order to deal with these problems with ecosystems, many similar model states are closed and data related to all these states is used to teach one international state. This leads to a large amount of data in each case and, therefore, the criteria are well considered. The option that connects is used is used by decision trees. HMM and this type of sharing were proposed under the half-standing names of the HMM mix.

Inspired by the great success of margin-based classification techniques (a machine learning approach), Jiang et al. (2006) incorporated this concept into hidden Markov modelling for speech recognition at York University. After that, second time Sha and Saul (2006) , computer science researchers at the University of Pennsylvania used large margin Gaussian mixture modelling for phonetic classification and recognition. Soft margin estimation (SME) is the third one proposed by Li et al. (2006) to overcome the difficulties of Large Margin Estimation (LME). Margin based training of HMMs seeks not only to “minimize the empirical error rate, but also to separate the scores of correct and incorrect transcriptions by the largest possible amount”, thus achieving better generalization on unseen data. Discriminative algorithms, MMIE, MCE, and MWE/MPE can also be cast in the rigorous SME framework by defining corresponding separation functions.

Artificial Neural Network is “a computer system inspired from the organization of cells in the human brain”. Multilayer perceptron (MLP) are the best studied class of ANN frequently applied in speech recognition. Artificial NN and more specifically MLP appeared to be a promising alternative in this respect to replace or help HMM in the classification mode. Some ANN approaches have been proposed to improve the state of the art of ASR system.

**AUTOMATIC SPEECH RECOGNITION**

**Automatic Speech Recognition**

The statistical approach for automatic voice recognition aims to model the stochastic relationship between a voice signal and the pronounced sequence to minimize the expected error rate of a specific classifier.

Given that the true probability distribution is used, the Bayes decision rule guarantees on average the lowest possible classification error rate. For most pattern recognition tasks, the actual probability distribution is not generally known, but rather we use appropriate model distribution.

**Signal analysis**

The initial step in an automated speech recognition system is to “extract features, i.e. identify the components of the audio signal that are adequate for detecting the linguistic content and discarding all the other stuff which are non-essential”. No two utterances of the same words or sentences are giving the same digital signal. In other words, the objective of signal analysis is to “derive a feature vector such that the vectors for the same phoneme are as close to each other as much possible, while the vectors for different phonemes are maximally differ to each other”.

The main factors which could cause two random speech samples to differ from one another are:

• Phonetic identity: The pronunciation difference depends on gender, dialect, voice, etc.

• Microphone: Including transmission medium.

• Environment: Surrounding noise and acoustics, etc.

Automatic Speech Recognition with Kaldi toolkit.

Automatic speech recognition uses common signal processing techniques thatare based on Mel Frequency Cepstral Coefficients (MFCC). In this project, MFCC is used.

Important to remember that the sounds produced by a human are altered by the shape of the vocal chords of the human. This determines how voice does come out and bundled in the short time power spectrum or cepstrum. Representation of the individual phoneme being produced is given by correct determination of this cepstrum. MFCC’s task is to represent this feature.

MFCC transformations are applied on a sampled and quantized audio signal. The overall MFCC calculation that Kaldi follows is:

• Calculate frames in the file.

• For each frame:

1) Data extraction, dithering options, unload dc offset before emphasis, and multiply it by windowing.

2) Determine power output.

3) Perform Fast Fourier Transformation (FFT) to calculate power spectrum.

4) Factor energy in each mel bin.

5) Calculates the log of energies and takes cosine transformations (DCT), keeping the coefficients as determined.

Feature extraction is primary initial step in speech recognizing applications. Along static features extracted from each frame of speech data, it is best to utilize some transformations to improve results.

Transforms, projections and other feature operations that are typically not speaker specific include:

• Frame division.

• Delta feature extraction.

• Linear Discriminant Analysis (LDA) transform.

• Heteroscedastic Linear Discriminant Analysis (HLDA).

• Maximum Likelihood Linear Transform (MLLT) estimation.

**Delta feature computation**

The “relationship in phonetic frames without considering the relationship between them” are accounted by MFCC. Phonetic data are orderly continuous, thus the capture of the dynamic spectrum data under phonetic frames will improve the performance of recognition model.

Therefore, the Delta element is a Fourier change of the system's timing of system frames. For example: If we have 11 MFCC coefficients, with a change of Δ + ΔΔ we also get 11 + 13 coefficients of the delta, which will combine a vector providing a height of 33 (11 + 11 + 11). Then, the original vector has decreased by 33 MFCC vector Δ + ΔΔ acoustic components.

**Acoustic Model**

The type of small units used in the speech recognition depends on the amount of existing data for the training and complexity of the waste model: when the least-known vocabulary designs (<100 words) use full word models, developed vocabulary recognition systems massive (> 5000 words) often use smaller units that can be created with weapons, phonics, or phonemes in the environment. Reliable dependent rodents are also known as telephone numbers. Commonly used features in large vocabulary speech recognition systems are mobile phones in one or two nearby phonemics, known as triphones or quinphones. Examples of dependent phonometer enable infectious infections that occur in different contexts of solidarity (solidarity).

**Evaluation**

There are different ways to assess the quality of the ASR system. Word Error Rate (WER) is a typical speech recognition functionality.

The main difficulty of measuring performance is that the sequence of the word recognition can have different lengths from the word line chain. WER is based on Levenshtein's distance, but it works at a level of word. This problem is solved by integrating the first sequence of word recognition with a sequence of word word by using the equity string.

WER=100∗(M+Q+T)/ D

Where:

-D: Is the number of words in the reference

-M: Is the number of substitutions

-Q: Is the number of insertions

-T: Is the number of deletions

**KALDI**

“Kaldi is a toolkit for speech recognition written in C++ and licensed under the Apache License v2.0. Kaldi is intended for use by speech recognition researchers”.

The goal is to have modern and flexible code, written in C++, that is easy to modify and extend. Important features include:

* Code-level integration with Finite State Transducers (FSTs)
  + We compile against the OpenFst toolkit (used as a library).
* Extensive linear algebra support
  + We include a [matrix library](http://kaldi-asr.org/doc/matrix.html) that wraps standard BLAS and LAPACK routines.
* Extensible design
  + As far as possible, “we provide our algorithms in the most generic form possible.” For instance, “our decoders are templated on an object that provides a score indexed by a (frame, fst-input-symbol) tuple”. This means the “decoder could work from any suitable source of scores, such as a neural net.”
* Open license
  + The code is licensed under Apache 2.0, which is one of the least restrictive licenses available.
* Complete recipes

The goal of “providing a complete recipe is an important feature of kaldi”. Since the principle is available to the public under a license that allows restructuring and re-opening, we would like to encourage people to release their code, including their script titles, in the same format as a copy of the Kalid model.

We have tried to make the documents as “complete” as possible due to time limitations, but for a while we can not expect to produce documents that are similar to the HTK. Especially there is a lot of introductory material in the HTKBook, describing the recognition of statistical speech for uninitiated, which may never appear on Calendars. Many Caldary documents are written in a manner that will only be available to the expert. In the future we expect to do as much as possible, considering that our viewing observers are researchers to identify statements or researchers in the training. In general, Kaldi is not a speech recognition identifier for "dummies." It will enable you to perform multiple operating types that are not available.

**MEL FILTERBANK SPECTRAL COEFFICIENTS**

The first step in any automatic speech recognition system is to extract features i.e. identify the components of the audio signal that are good for identifying the linguistic content and discarding all the other stuff which carries information like background noise, emotion etc. Mel Filterbank features or Mel Filterbank Spectral Coefficients (MFSC) are a feature that are used in ASR. Following are the steps involved in computing Mel filterbank:

*Pre-Emphasis***:** The first step is to apply a pre-emphasis filter on the signal to amplify the high frequencies. A pre-emphasis filter is useful in several ways:

(1) balance the frequency spectrum since high frequencies usually have smaller magnitudes compared to lower frequencies.

(2) avoid numerical problems during the Fourier transform operation.

(3) may also improve the Signal-to-Noise Ratio (SNR).

The pre-emphasis filter can be applied to a signal *x* using the first order filter in the following equation:

*y*(*t*)=*x*(*t*)−*αx*(*t*−1)

where typical values for the filter coefficient (*α*) are 0.95 or 0.97

*Framing:* After pre-emphasis, we need to split the signal into short-time frames. The rationale behind this step is that frequencies in a signal change over time, so in most cases, it doesn’t make sense to do the Fourier transform across the entire signal in that we would lose the frequency contours of the signal over time. To avoid that, we can safely assume that frequencies in a signal are stationary over a very short period of time. Therefore, by doing a Fourier transform over this short-time frame, we can obtain a good approximation of the frequency contours of the signal by concatenating adjacent frames.

Typical frame sizes in speech processing range from 20ms to 40ms with 50% (+/-10%) overlap between consecutive frames. Popular settings are 25ms for the frame size.

*Window:* After slicing the signal into frames, we apply a window function such as the Hamming window to each frame. A Hamming window has the following form:

*w*[*n*]=0.54−0.46*cos*(2*πn/N*−1)

where, 0≤*n*≤*N*−1, *N* is the window length.

There are several reasons why we need to apply a window function to the frames, to counteract the assumption made by the FFT that the data is infinite and to reduce spectral leakage.

*Fourier-Transform and Power Spectrum:* We can now do an *N*-point FFT on each frame to calculate the frequency spectrum, which is also called Short-Time Fourier-Transform (STFT), where *N* is typically 256 or 512, NFFT = 512; and then compute the power spectrum (periodogram) using the following equation:

where, *xi* is the *ith* frame of signal *x*

*Filter Banks:* The final step to computing filter banks is applying triangular filters, typically 40 filters, nfilt = 40 on a Mel-scale to the power spectrum to extract frequency bands. The Mel-scale aims to mimic the non-linear human ear perception of sound, by being more discriminative at lower frequencies and less discriminative at higher frequencies. We can convert between Hertz (*f*) and Mel (*m*) using the following equations:

*Mean Normalization*: As previously mentioned, to balance the spectrum and improve the Signal-to-Noise (SNR), we can simply subtract the mean of each coefficient from all frames.

**CONVOLUTIONAL NEURAL NETWORK**

**Artificial Neural Networks**

Over the last 30 years, a new computer-based view of neural science has grown. His name, neural networks, comes from the fact that these techniques depend on analogies derived from the internal functions of the human nervous system. The main idea behind neural networking is imitating the brain behaviour by using the linked barriers to real neurons, finding systems that show complicated behaviour, and with optimism of mind.

Contrary to the biological neural networks, neural network architecture is incorporated into design processes, and criteria that define how their neurons process interfere with the process of learning. The learning process involves searches in the parameter position to determine the best values of the parameter according to the quality parameters.

The convolution neural network is given “a sequence of input input signals, split into frames, and score results for each class”. Architecture includes several filter steps, followed by grouping action. The physiological differentiating step involves a layer of solutions, followed by a long-term linking and non-linear (tanh ()). Our direct design included three stages of the fili. The refined signals derived from these steps are provided by group action, which in turn is a multi-layer perceptron, and one layer hidden. Generates conditional probabilities = p (i | x), for each class i, for each chapter x using the “SoftMax layer”. The network is trained “under cross-entropy criteria, increasing by using the gradient increase algorithm”.

**Multi-Layer Perceptron**

The Multi-Layer Perceptron (MLP) function can be well defined and the results verified by Kolmogorov, then reunited with Cybenko, Funahashi, and others. They indicated that the MLP and one layer hidden by neurons have enough capacity for almost any ongoing work. MLP is a total function of approximately. As in the SOM, the MLP does not associate with long-term behaviourpu.

MLP includes several neurons arranged for different layers. An array that has neurons that receives external input is called an insert layer. The output column is called output layer. All inputs between input and output layers are called hidden layers. Any input vector used on the Internet is spread from the input layer, through the hidden buttons, toward the output layer.

**Convolutional layer**

When standard classical layers in common MLPs accept vector size size tablets, the convolution layer is supposed to be fed to the T vectors / frame chain: X = {x1 x2 ... xT}. The convolutional chapter applies the same line change over each series (or inter spaced with windows dW) framed windows.

**Max-pooling layer**

These kind of layers perform “local temporal max operations over an input sequence”. More formally, the transformation at frame t is written as:

max t−(kW−1)/2≤s≤t+(kW−1)/2

xd s ∀d (2)

with x being the input, kW the kernel width and d the dimension.

**TIMIT**

**Corpus Speaker Distribution**

TIMIT has “a total of 6300 sentences, 10 sentences spoken by each of 630 speakers from 8 regions of the US dialect”.

Table 1 shows the number of speakers in 8 regions, sexually differing. Percentages are deduced from brackets. The language of the speaker in the US area where they lived during their childhood age. Geographical areas are related to regions known in the US (Language Files, Ohio University of the Linguistics Dept University, 1982), except by location Western (dr7) which language boundaries are unknown to any area of confidence in 8 languages where speakers moved very close during their childhood.

Table 1: “Dialect distribution of speakers

**Dialect**

Region(dr) #Male #Female Total

---------- --------- --------- ----------

1 31 (63%) 18 (27%) 49 (8%)

2 71 (70%) 31 (30%) 102 (16%)

3 79 (67%) 23 (23%) 102 (16%)

4 69 (69%) 31 (31%) 100 (16%)

5 62 (63%) 36 (37%) 98 (16%)

6 30 (65%) 16 (35%) 46 (7%)

7 74 (74%) 26 (26%) 100 (16%)

8 22 (67%) 11 (33%) 33 (5%)

------ --------- --------- ----------

8 438 (70%) 192 (30%) 630 (100%)

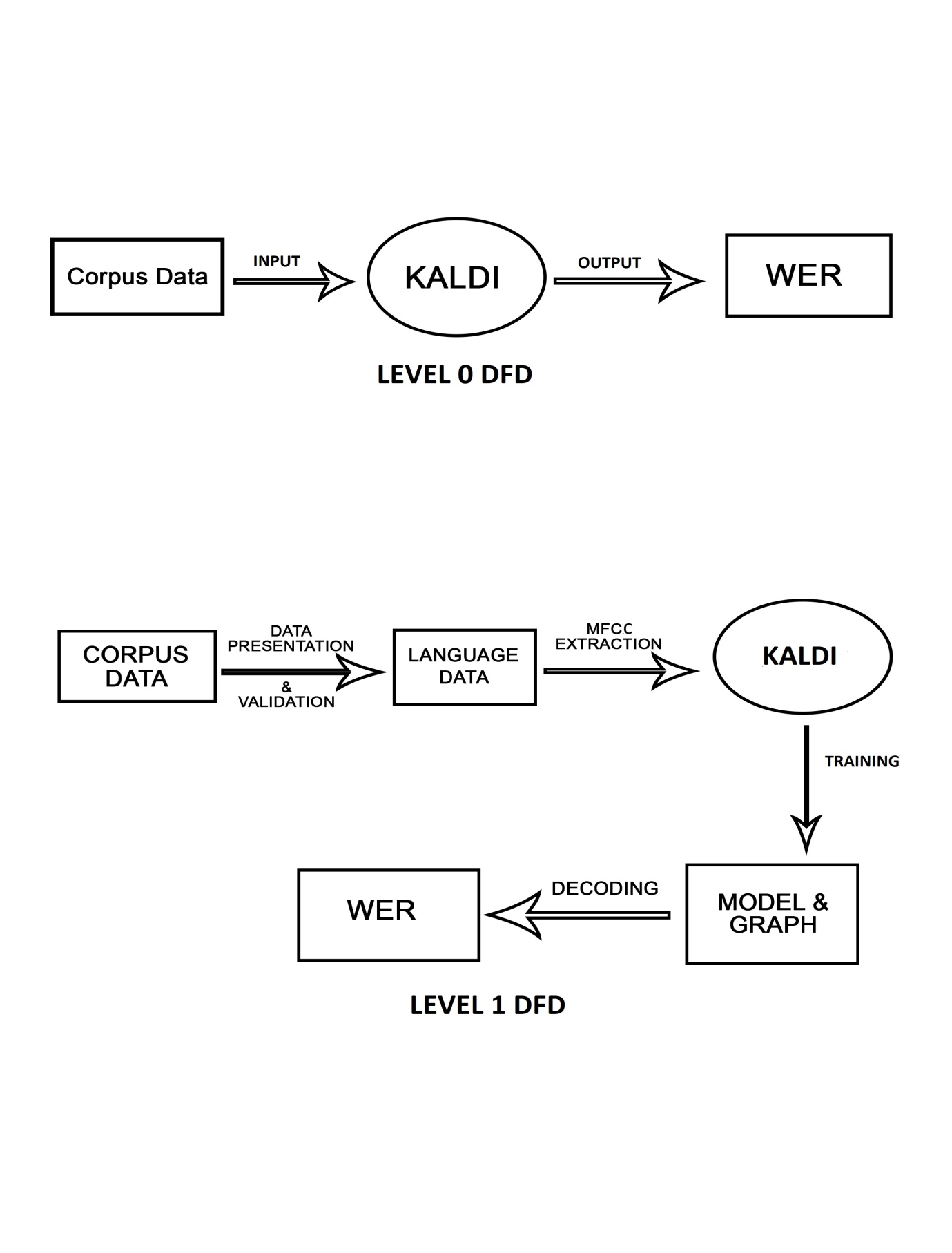
The dialect regions are:

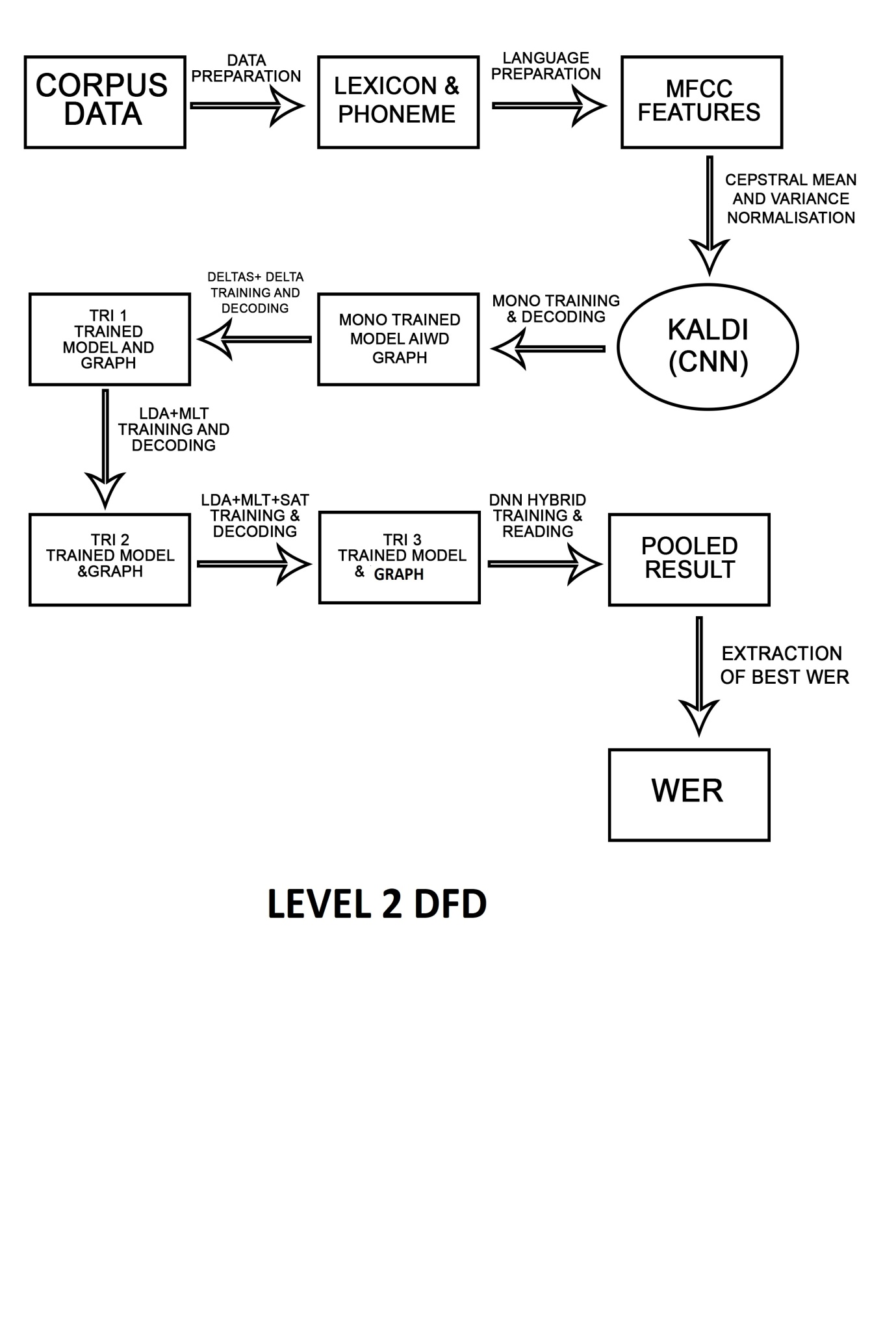
dr1: New England dr2: Northern

dr3: North Midland dr4: South Midland

dr5: Southern dr6: New York City

dr7: Western dr8: Army Brat (moved around)”

**DATA FLOW DIAGRAM**



**RESULT**

**CONCLUSION**

**REFERENCES**

[1] ADSF, The Alex Dialogue Systems Framework, April 2014, https://github.com/UFAL-DSG/alex.

[2] Lee Akinobu, Open-Source Large Vocabulary CSR Engine Julius, April 2014, http://julius.sourceforge.jp/en\_index.php.

[3] Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri, OpenFst: A general and eﬃcient weighted ﬁnitestate transducer library, Implementation and Application of Automata, Springer, 2007, pp. 11–23.

[4] Stefan Behnel, Robert Bradshaw, Lisandro Dalcín, Mark Florisson, Vitja Makarov, and Dag Seljebotn, Cython: C-Extensions for Python, 2014, http://cython.org/.

[5] Thorsten Brants, Ashok C Popat, Peng Xu, Franz J Och, and Jeﬀrey Dean, Large language models in machine translation, In Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Citeseer, 2007.

[6] http://kaldi.sourceforge.net

[7] Yu Hongzhi, A Research on Recognition of Tibetan Speakers Based on MFCC and Delta Features. Proceedings IFCSTA. IEEE, 2009, volume 2: pp, 234-238.

[8] R. Haeb-Umbach and H. Ney, Linear discriminant analysis for improved large vocabulary continuous speech recognition, Proceedings ICASSP. IEEE, 1992, pp, 13-16.

[9] Omar, M.K. and Hasegawa-Johnson, M. Model enforcement: a unified feature transformation framework for classification and recognition, Signal Processing, IEEE, 2004, volume 52: pp,27012710.

[10] Steve Renals, Automatic Speech Recognition. Search and decoding. 2012 ASR Lecture 10.

[11] D.Povey and A. Ghoshal, The Kaldi Speech Recognition Toolkit, ASRU 2011.

[12] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, “OpenFst: a general and efficient weighted finite-state transducer library,” in Proc. CIAA, 2007.

[13] Mohri, Pereira and Riley, Speech Recognition with Weighted Finite-State Transducers, in Springer Handbook on SpeechProcessing and Speech Communication, 2008.

**COMPLETE CONTRIBUTARY SOURCE CODE**

G:\Programming\Minor Project\kaldi-master\egs\timit\s5\run.sh>

#!/bin/bash

#

# Copyright 2013 Bagher BabaAli,

# 2014-2017 Brno University of Technology (Author: Karel Vesely)

#

# TIMIT, description of the database:

# http://perso.limsi.fr/lamel/TIMIT\_NISTIR4930.pdf

#

# Hon and Lee paper on TIMIT, 1988, introduces mapping to 48 training phonemes,

# then re-mapping to 39 phonemes for scoring:

# http://repository.cmu.edu/cgi/viewcontent.cgi?article=2768&context=compsci

#

. ./cmd.sh

[ -f path.sh ] && . ./path.sh

set -e

# Acoustic model parameters

numLeavesTri1=2500

numGaussTri1=15000

numLeavesMLLT=2500

numGaussMLLT=15000

numLeavesSAT=2500

numGaussSAT=15000

numGaussUBM=400

numLeavesSGMM=7000

numGaussSGMM=9000

feats\_nj=10

train\_nj=30

decode\_nj=5

echo ============================================================================

echo " Data & Lexicon & Language Preparation "

echo ============================================================================

#timit=/export/corpora5/LDC/LDC93S1/timit/TIMIT # @JHU

#timit=/mnt/matylda2/data/TIMIT/timit # @BUT

timit=/home/vishal/TIMIT

local/timit\_data\_prep.sh $timit || exit 1

local/timit\_prepare\_dict.sh

# Caution below: we remove optional silence by setting "--sil-prob 0.0",

# in TIMIT the silence appears also as a word in the dictionary and is scored.

utils/prepare\_lang.sh --sil-prob 0.0 --position-dependent-phones false --num-sil-states 3 \

data/local/dict "sil" data/local/lang\_tmp data/lang

local/timit\_format\_data.sh

echo ============================================================================

echo " MFCC Feature Extration & CMVN for Training and Test set "

echo ============================================================================

# Now make MFCC features.

mfccdir=mfcc

for x in train dev test; do

steps/make\_mfcc.sh --cmd "$train\_cmd" --nj $feats\_nj data/$x exp/make\_mfcc/$x $mfccdir

steps/compute\_cmvn\_stats.sh data/$x exp/make\_mfcc/$x $mfccdir

done

echo ============================================================================

echo " MonoPhone Training & Decoding "

echo ============================================================================

steps/train\_mono.sh --nj "$train\_nj" --cmd "$train\_cmd" data/train data/lang exp/mono

utils/mkgraph.sh data/lang\_test\_bg exp/mono exp/mono/graph

steps/decode.sh --nj "$decode\_nj" --cmd "$decode\_cmd" \

exp/mono/graph data/dev exp/mono/decode\_dev

steps/decode.sh --nj "$decode\_nj" --cmd "$decode\_cmd" \

exp/mono/graph data/test exp/mono/decode\_test

echo ============================================================================

echo " tri1 : Deltas + Delta-Deltas Training & Decoding "

echo ============================================================================

steps/align\_si.sh --boost-silence 1.25 --nj "$train\_nj" --cmd "$train\_cmd" \

data/train data/lang exp/mono exp/mono\_ali

# Train tri1, which is deltas + delta-deltas, on train data.

steps/train\_deltas.sh --cmd "$train\_cmd" \

$numLeavesTri1 $numGaussTri1 data/train data/lang exp/mono\_ali exp/tri1

utils/mkgraph.sh data/lang\_test\_bg exp/tri1 exp/tri1/graph

steps/decode.sh --nj "$decode\_nj" --cmd "$decode\_cmd" \

exp/tri1/graph data/dev exp/tri1/decode\_dev

steps/decode.sh --nj "$decode\_nj" --cmd "$decode\_cmd" \

exp/tri1/graph data/test exp/tri1/decode\_test

echo ============================================================================

echo " tri2 : LDA + MLLT Training & Decoding "

echo ============================================================================

steps/align\_si.sh --nj "$train\_nj" --cmd "$train\_cmd" \

data/train data/lang exp/tri1 exp/tri1\_ali

steps/train\_lda\_mllt.sh --cmd "$train\_cmd" \

--splice-opts "--left-context=3 --right-context=3" \

$numLeavesMLLT $numGaussMLLT data/train data/lang exp/tri1\_ali exp/tri2

utils/mkgraph.sh data/lang\_test\_bg exp/tri2 exp/tri2/graph

steps/decode.sh --nj "$decode\_nj" --cmd "$decode\_cmd" \

exp/tri2/graph data/dev exp/tri2/decode\_dev

steps/decode.sh --nj "$decode\_nj" --cmd "$decode\_cmd" \

exp/tri2/graph data/test exp/tri2/decode\_test

echo ============================================================================

echo " tri3 : LDA + MLLT + SAT Training & Decoding "

echo ============================================================================

# Align tri2 system with train data.

steps/align\_si.sh --nj "$train\_nj" --cmd "$train\_cmd" \

--use-graphs true data/train data/lang exp/tri2 exp/tri2\_ali

# From tri2 system, train tri3 which is LDA + MLLT + SAT.

steps/train\_sat.sh --cmd "$train\_cmd" \

$numLeavesSAT $numGaussSAT data/train data/lang exp/tri2\_ali exp/tri3

utils/mkgraph.sh data/lang\_test\_bg exp/tri3 exp/tri3/graph

steps/decode\_fmllr.sh --nj "$decode\_nj" --cmd "$decode\_cmd" \

exp/tri3/graph data/dev exp/tri3/decode\_dev

steps/decode\_fmllr.sh --nj "$decode\_nj" --cmd "$decode\_cmd" \

exp/tri3/graph data/test exp/tri3/decode\_test

echo ============================================================================

echo " SGMM2 Training & Decoding "

echo ============================================================================

steps/align\_fmllr.sh --nj "$train\_nj" --cmd "$train\_cmd" \

data/train data/lang exp/tri3 exp/tri3\_ali

#From this point you can run Karel's DNN : local/nnet/run\_dnn.sh

steps/train\_ubm.sh --cmd "$train\_cmd" \

$numGaussUBM data/train data/lang exp/tri3\_ali exp/ubm4

steps/train\_sgmm2.sh --cmd $numLeavesSGMM $numGaussSGMM \

data/train data/lang exp/tri3\_ali exp/ubm4/final.ubm exp/sgmm2\_4

utils/mkgraph.sh data/lang\_test\_bg exp/sgmm2\_4 exp/sgmm2\_4/graph

steps/decode\_sgmm2.sh --nj "$decode\_nj" --cmd "$decode\_cmd"\

--transform-dir exp/tri3/decode\_dev exp/sgmm2\_4/graph data/dev \

exp/sgmm2\_4/decode\_dev

steps/decode\_sgmm2.sh --nj "$decode\_nj" --cmd "$decode\_cmd"\

--transform-dir exp/tri3/decode\_test exp/sgmm2\_4/graph data/test \

exp/sgmm2\_4/decode\_test

echo ============================================================================

echo " MMI + SGMM2 Training & Decoding "

echo ============================================================================

steps/align\_sgmm2.sh --nj "$train\_nj" --cmd "$train\_cmd" \

--transform-dir exp/tri3\_ali --use-graphs true --use-gselect true \