Static Dictionary for Pronunciation Modeling

A major project report submitted
in partial fulfillment of the requirements
for the award of the degree of

Master of Technology
in
Computer Science

By

SUBRAMANYAM KUNISETTI
(07MCMT02)

Under the Guidance of

Prof. P. N. Girija

Department of Computer & Information Sciences
School of Mathematics & Computer/Information Sciences
University of Hyderabad
Hyderabad -500046, INDIA.
June – 2009.

Certificate

This is to certify that the thesis entitled **" Static Dictionary for Pronunciation Modeling "** being submitted by **SUBRAMANYAM KUNISETTI** in partial fulfillment for the award of the degree of **Master of Technology** in **Computer Science**, is a record of the bonafide work carried out by him under my supervision at the University of Hyderabad.

The matter embodied in this project report has not been submitted to any other university for the award of any other degree or diploma.

Prof. P. N. Girija
Supervisor,
Dept. of CIS,
University Of Hyderabad,
Hyderabad.

Head,
Dept. of CIS,
University of Hyderabad,
Hyderabad.

Dean,
School of MCIS,
University of Hyderabad,
Hyderabad.

Acknowledgments

There are a lot of people without whose support: physical, technical and moral, neither the project nor this manuscript could possibly have neared completion. Though I would have liked very much to do so, it is unfortunately not feasible to mention all of them individually here, as that would probably occupy half of this report.

I would like to take this opportunity to thank my guide **Prof. P. N. Girija,** for her guidance, valuable suggestions, support in every task of my work, and freedom in expressing my opinions during my project work.

I would like to specially thank the Head of the department **Prof. Arun Agarwal,** who very kindly provide me the facilities required to proceed with my work.

I would like to thank the Dean of the School of MCIS, **Prof. Amaranath** for giving me valuable suggestions.

My heartful thanks to my **friends,** and thanks to **classmates** for encouragement and support which they given me in completion of my course.

Before closing, I would like to extend special attention to my **Parents,** who gave me this position and providing me inspiration and moral support throughout my studies.

SUBRAMANYAM KUNISETTI

Abstract

Automatic Speech Recognition (ASR) or Speech-to-text conversion is a sequential pattern recognition problem. It comprises of three major components- acoustic models, language model and the pronunciation dictionary, which aims to correctly hypothesize a spoken utterance into a string of words. During the training, the system is provided with speech data, the corresponding transcription and a pronunciation dictionary. At the decoding run-time, the acoustic models and language models trained on the task are used along with one of the standard dictionary as lexicon. After Decoding is completed, the confusion pairs are used as arguments for Levenshtein Distance algorithm, which gives the maximum number of operations required to convert one string into another. The pair which has minimum distance will be considered for adding to the static dictionary. Later the decoding process will be repeated to find improved recognition accuracy.

Table of Contents

TOPIC **PAGE NO**

Chapter1

Introduction

1 Introduction:

Speech Recognition is the process by which a computer can recognize spoken words. Basically it means talking to your computer and having it correctly recognize what you are saying. It has a wide area of applications in Command recognition, Dictation, Interactive voice Response etc. Speech Recognition can help also, handicapped people to interact with society. It is a technology that makes life easier and very promising.

The methods of statistical pattern recognition have been applied for several decades. Speech Recognizers based upon Hidden Markov Model have achieved a high level of performance in controlled environments. One naive approach for robust speech recognition in order to handle the mismatch between training and testing environments, many techniques has been proposed.

Some methods work on acoustic model and some methods work on Language model. Pronunciation dictionary is the most important component of the Speech Recognition system. New pronunciations for the words will be considered for speech recognition accuracy.

What is pronunciation?

Since the goal of this dissertation is to improve models of pronunciation for ASR systems. Pronunciation refers to two related constructs: the act of producing an acoustic message or the actual acoustic message itself in an abstract sense.

The pronunciation dictionary (or the lexicon) is a mapping table, a representation of the system's vocabulary in terms of its acoustic modeling units [1]. In general, while

acoustic and language models are outputs of statistical optimization procedures, lexicons are usually taken off the shelf. During the training, the system is provided with speech data, the corresponding transcription and a pronunciation dictionary. At the decoding run-time, the acoustic models and language models trained on the task are used while one of the standard dictionaries is used as the lexicon. Standard lexicons are manually built by linguists to provide the most generic pronunciations of the words. However, variations occur in pronunciation due to a number of factors including gender, accent, dialect, mode of speaking etc. While a generic pronunciation dictionary remains to be the safest but, it may not be optimally suited for a test condition. Thus, there is a need to adapt a lexicon in order to best match the test conditions. This dissertation shows that the lexicon can also be improved for a task by adding the variants to pronunciations, inferred using the resources already provided for acoustic model training.

Pronunciation dictionaries are generally hand-crafted by linguists to reflect the most agreed pronunciations of each word. In theory, sound units are divided into two basic types: phones and phonemes [1]. Phones are the fundamental sound categories that describe the range of acoustic features found in languages. Phonemes on the other hand are abstract, language specific entities that may represent one or more phones.

1.1 Statement of the Problem:

Most current leading edge speech recognition systems are based on an approach called Hidden Markov Modeling (HMM). By adding new pronunciations to the Static Dictionary the accuracy can be improved. There are 2 methods to add pronunciations to the static dictionary. First, calculate Levenshtein distance between the strings in the confusion pairs. Second, add the pronunciation by frequency of occurrence.

1.2 Aim of the Project:

Automatic Speech Recognition (ASR) or Speech-to-text conversion is a sequential pattern recognition problem. It comprises of three major components- acoustic models,

language model and the pronunciation dictionary, which aims to correctly hypothesize a spoken utterance into a string of words. During the training, the system is provided with speech data, the corresponding transcription and a pronunciation dictionary. At the decoding time, the acoustic models and language models trained on the task are used along with one of the standard dictionary (CMUdict) as lexicon. After Decoding is completed, the confusion pairs are used as arguments for Levenshtein Distance algorithm, which gives the maximum number of operations required to convert one string into another. The pair which has minimum distance will be considered for adding to the static dictionary. Later the decoding process will be repeated to find improved recognition accuracy. In the other method the word which has occurred more number of times will be considered for new entry in the static dictionary.

1.3 Scope of the Project:

To improve the Speech recognition accuracy many people working with acoustic model and Language model. But it is good to work with pronunciation variations to improve the Speech recognition accuracy. There are two types of pronunciation dictionaries. Static dictionary and dynamic dictionary. This project used static dictionary.

1.3.1 Goals of the Proposed System:

To develop a project that gives the improved speech recognition accuracy for Telugu sentences using different approaches.

Chapter 2

Previous work on Static Dictionary

2.1 Predicting Variations in Pronunciations:

One primary source of pronunciation is a hand-written Dictionary, in which experts carefully write Pronunciation models for each word. When care is taken to minimize dictionary confusions and ensure consistency of pronunciation across similar words, the resulting Dictionary is often excellent. The pronunciation dictionary and the acoustic models are components of the system to handle variations in pronunciation. Pronunciation variations concentrates on coverage of the particular ways in which one word is uttered among different speakers, speaking styles, speech stress, dialectal variations, speaking rates etc.

Basically sound units are divided into two types. Phones and phonemes. Phones are the fundamental sound categories that describe the range of acoustic features found in languages. Phonemes on the other hand are abstract, language specific entities that may represent one or more phones.

2.1.1 Automatic base form learning:

The simple method of learning pronunciation variants is to learn each word's various pronunciations on a word-by-word basis. Typically a phone recognizer is used to determine possible alternatives for each word by finding a best-fit alignment between the phone recognition string and canonical pronunciations provided by a Static Dictionary.

Wooters and Westenrndorf and Jelitto used alignments between base form pronunciations and frequent pronunciations derived from a phone recognizer. We can also use HMM generalization, which allows induction of new base forms not seen in the training data by finding common variations among pronunciations seen during training [2].

2.2 Dictionary Refinement:

Some times Dictionary pruning is used to improve the Speech Recognition accuracy. Dictionary pruning is done based on the speech training database. We may arrive 2 types of problems

1. Words that are not included in the data do not have information to be treated with and
2. Some words tend to keep pronunciations that were rarely observed.

To solve the unobserved words problem, we can use central or summary pronunciations in the pruned Dictionary [3]. The aim of this sort of pronunciation is to capture the phonetic contents included in the set of pronunciation variants of each word and to consolidate them in a reduced pronunciation set.

2.3 Enhanced Tree Clustering:

This approach is contrast to decision tree based approach, which allows parameter sharing across phonemes [4]. In this approach a single decision tree is grown for all sub-states. The clustering procedure starts with all polyphones at the root. Questions are asked regarding the identity of the center phone and its neighboring phones. At any node, the question that yields the highest information is chosen and the tree is split. This process is repeated until either the tree reaches a certain size or a minimum count threshold is crossed. Compared to the traditional multiple-tree approach, a single tree allows more flexible sharing of parameters any nodes can potentially be shared by multiple phones.

2.4 Pronunciation Variations Captured by Triphone Model:

We can observe which pronunciations are well captured by trip hone model, and which are not. We have to consider two factors to know which pronunciations are handled by triphone model [5].

1. The exposure to more triphones in training.
2. The kinds of phonetic variation in the sentence.

2.4.1 Syllable Deletion:

There are 3 kinds of variations which are syllable deletion, vowel reduction and phone substitution [6].

Syllable deletions are cases in which the surface pronunciation misses the syllable which is in canonical pronunciations. Syllable deletion is not well modeled by simply having more data for the triphones.

The vowel reduction is an important category of variations, because it is linked with prosodic effects. Vowels which are stressed or accented are not reduced. Whether a syllable received lexical stress or not is already modeled in the lexicon.

Within word variation is the kind of variation that can be modeled at the level of the lexicon by adding pronunciation variants.

Adaptation of the Static Dictionary is done by adding pronunciation variants to it. So the first stage in the process is generation of candidate variants. This is either done manually or by automatic procedures [7]. The procedures are

• using learnt rules to generate the possible candidates of variants [8].
• Artificial neural networks.
• Grapheme to phoneme converters.

2.5 Approach Adopted in this Dissertation:

This dissertation proposes different methods to improve the Speech Recognition accuracy. At the first I take the wave files, removed the noise using Praat. In the later stage after Decoding, i take confusion pairs and calculate the distance using Levenshtein Distance algorithm. The minimum distance variants are added to the Static Dictionary. In the other approach which variant repeated frequently is considered to add into Static Dictionary.

Chapter-3
Design

3.1 Speech Recognition System:

A machine or software capable of recognizing spoken language. The machine or software may take the spoken language and translate it into written text, or follow the spoken instructions to perform other functions.

3.2 Speech Recognition Techniques:

There are three major types of speech recognition techniques [9].

First, the acoustic-phonetic approach assumes that the phonetic units are broadly characterized by the set of features, such as formant frequencies, voiced/unvoiced, and pitch. These features are extracted from the speech signal and are used to segment and label the speech.

Second, the pattern recognition approach requires no explicit knowledge of speech. This approach has two steps- namely, one training of speech patterns based on some generic spectral parameter set and another recognition of patterns via pattern comparison. The popular pattern recognition techniques include template matching, Hidden Markov Model (HMM), and Artificial Neural Network.

Third, the Artificial Intelligence approach attempts to mechanize the recognition procedure according to the way a person applies its intelligence in visualizing, analyzing, and finally making a decision on the measured acoustic features. Expert systems are used widely in this approach.

3.3 Speech Recognition using Sphinx:

SPHINX is the one of the best and most versatile Recognition systems in the world today. SPHINX-III is a large vocabulary, speaker-independent, Hidden Markov Model (HMM)-based continuous speech recognition system like its predecessors, the original SPHINX system and SPHINX-II.

This SPHINX developed at CMU in 1988 and it was one of the systems which demonstrate the feasibility of accurate, speaker-independent, large vocabulary continuous speech recognition. This sphinx trainer consists of a set of programs which have been compiled for two operating systems: LINUX and ALPHA. This project uses the SPHINX-III Recognition system.

SPHINX-III has a more flexible structure. The user can determine whether it is continuous or semi-continuous modes to be used and the number of streams of data the system will be used and the organization of these streams. SPHINX-III includes both an acoustic trainer and various decoders i.e. text recognition, phoneme recognition, N-best list generation, etc.

3.3.1 Signal Processing:

Speech can be represented in terms of its message content, or information. An alternative way of characterizing speech in terms of the signal carrying the message information, i.e. the acoustic waveform [9]. All the Speech recognition systems use a parametric representation of speech instead of the waveform itself which is based as the pattern recognition. These parameters carry the information related to the short-time spectrum of the signal. SPHINX-III uses the Mel-frequency cepstral coefficients (MFCC) as static features for speech recognition.

3.3.2. Recognition Unit:

HMM is used to model the specific unit of speech .This Specific unit may be word, a sub word, or a complete form of sentence. For a large-vocabulary systems, HMM model is used to model the sub word units i.e. phonemes. For the small-vocabulary systems, it is used to model the word itself. The amount of training data and storage required for word models is enormous, that why SPHINX-III based on phonetic models. However it is inadequate to capture the variability of acoustical behaviors for the given phoneme in different contexts, for those particular contexts it will be modeled using triphones.

Training using sphinx-III:

The training procedure involves optimizing HMM parameters given training data. An iterative procedure, the Baum-Welch or forward-backward algorithm is employed to estimate transition probabilities, output distributions, and codebook means and variances under the probabilistic framework.

Recognition using sphinx-III:

For large-vocabulary tasks in the continuous speech recognition, the search algorithm should involved the concepts of acoustic and linguistics in order to maximize the accuracy of the recognition. To apply all these, SPHINX-III uses the Viterbi algorithm.

The SPHINX-III decoder is designed in such a way that it incorporates all the available acoustic and linguistic concepts in several phases. In initial stage, Viterbi beam search produces a single recognition hypothesis as well as a word lattice that includes word segmentations and acoustic scores. The word lattice is then transformed into a directed acyclic graph (DAG) This DAGs are for quick search for the best hypothesis. DAGs are also used to generate N-best lists for re-scoring empirically optimized parameters like the language weight and insertion penalty. This speech recognition system uses the process of learning the set of sound units which is called as the Training. The process of using the knowledge acquired to deduce the most probable sequence of units in the given signal is termed as the Decoding or simply Recognition. SPHINX system has the SPHINX trainer and the SPHINX decoder.

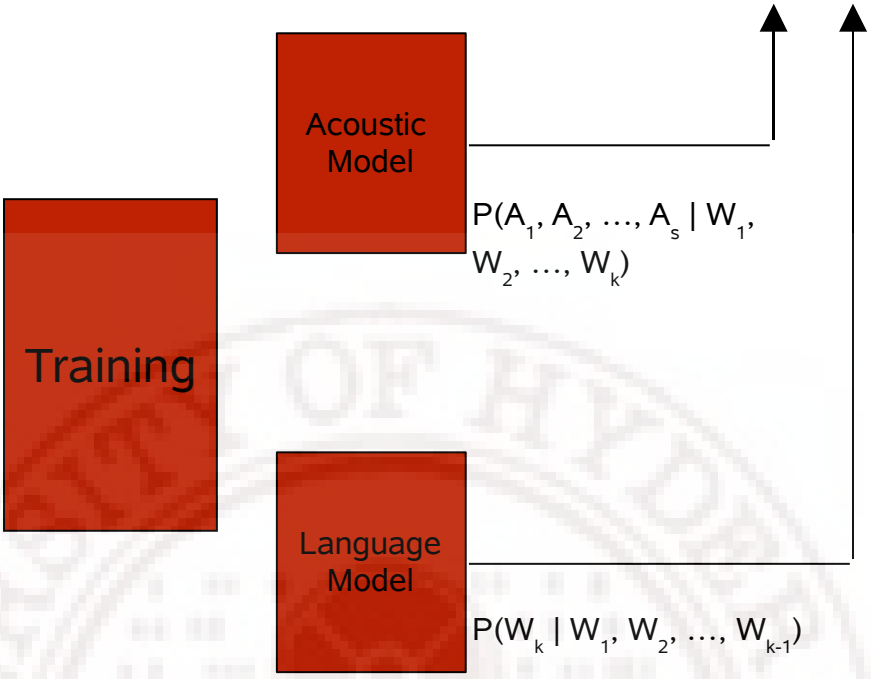

Fig 3.1: Training

Probabilistic Formulation:

Let A = {A1, A2, … ,At} be a sequence of acoustic observations. Let W = {W1, W2, …, Wm} be sequence of words.

Given the acoustic observations A, the probability of word sequence W.

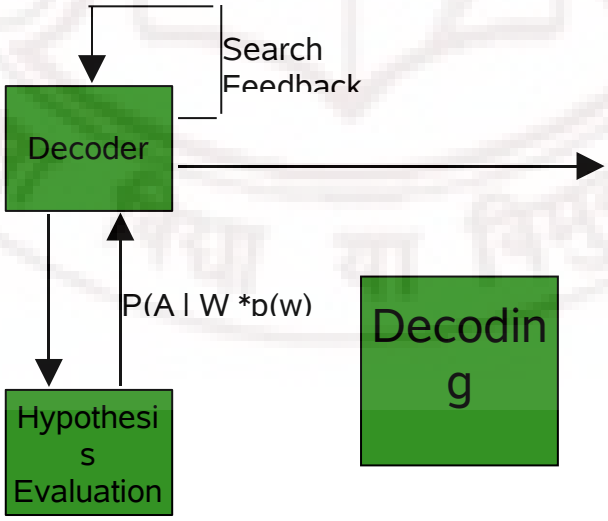

Figure 3.2: Decoding

For recognition, we choose

Best = argmaxW(P(W | A)).....Eq 3.1

Bayes Rule:

argmaxW (P(W | A)) = argmaxW (P(W, A) / P(A)).....Eq 3.2

= argmaxW(P(W, A) .....Eq 3.3

= argmaxW(P(A | W) * P(W)).....Eq 3.4

A model for the probability of acoustic observations given the word sequence, P(A | W), is called an “acoustic model”. A model for the probability of word sequences, P(W), is called a “language model”.

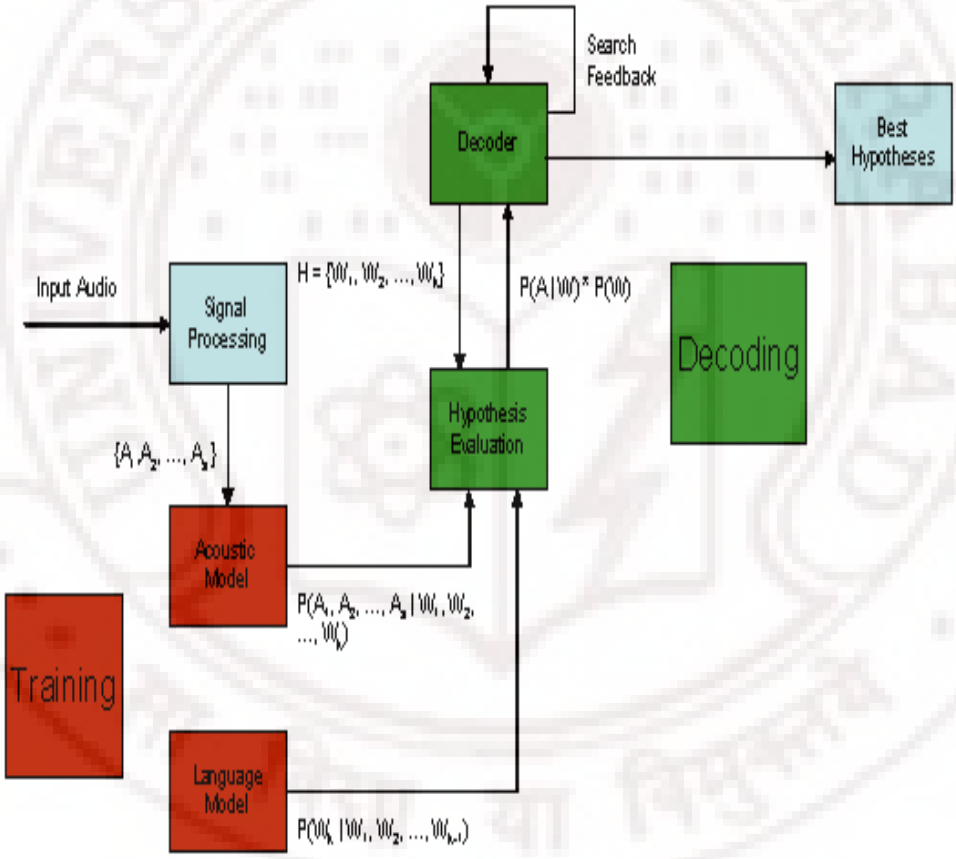

Fig 3.3: Block Diagram of the speech Recognition System

Search Problem:

Find the sequence WBest = { W1, W2, …, Wm }.....Eq 3.5

= argmaxW(P(A | W) * P(W)).....Eq 3.6

This is a search of a space of Vm word sequences, where 'V' is the vocabulary size (which may be 100,000 words or larger), and 'm' is the sentence length (which may be 20 words long or longer).Clearly, an exhaustive search is impossible. It is necessary to have a structured, intelligent search.

The components provided for the trainer:

(i) The trainer executables.
(ii) The Acoustic signals.
(iii) The corresponding transcription file.
(iv) The dictionary.
(v) The filler dictionary.

The components provided for decoding:

(i). The decoder executable.
(ii). The dictionary.
(iii).The filler dictionary.
(iv).The language model.

3.4 Evaluation of Speech Recognition Accuracy:

Word error rate (WER) is a common metric of the performance of a speech recognition system. The general difficulty of measuring performance lies in the fact that the recognized word sequence can have a different length from the reference word sequence (supposedly the correct one). The WER is derived from the Levenshtein distance, working at the word level instead of the phoneme level.

This problem is solved by first aligning the recognized word sequence with the reference (spoken) word sequence using dynamic string alignment.

Word error rate can then be computed as:

WER=(S+I+D)/N .....Eq 3.7

where

- *S* is the number of substitutions,
- *D* is the number of the deletions,
- *I* is the number of the insertions,
- *N* is the number of words in the reference.

When reporting the performance of a speech recognition system, sometimes word recognition rate (WRR) is used instead [1] :

WRR=1-WER=(N-S-D-I)/N=(H-I)/N.....Eq 3.8 where

- *H* is N-(S+D), the number of correctly recognized words.

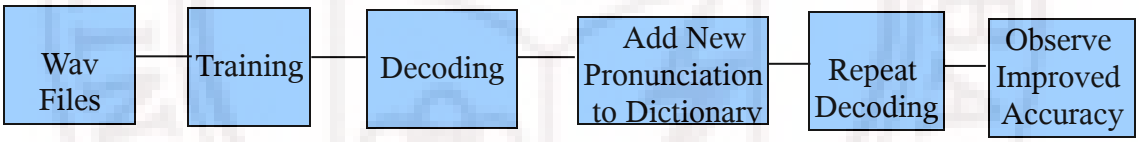

Fig 3.4 Use of Static Dictionary for Speech Recognition

3.5 Data Flow Diagrams:

Data Flow Diagrams illustrates how data is processed by a system in terms of inputs and outputs. The diagrams use four symbols to represent any system at any level of detail [9].

Data flow diagrams do not show decisions or timing of events. Their function is to illustrate data sources, destinations, flows, stores and transformations [10]. The capabilities of data flow diagramming align directly with general definitions of systems. Data flow diagrams are an implementation of a method for representing system concepts including boundaries, inputs/outputs, processes/sub processes, etc.

3.5.1 Data Flow Diagrams for Speech recognition using HMM:

:

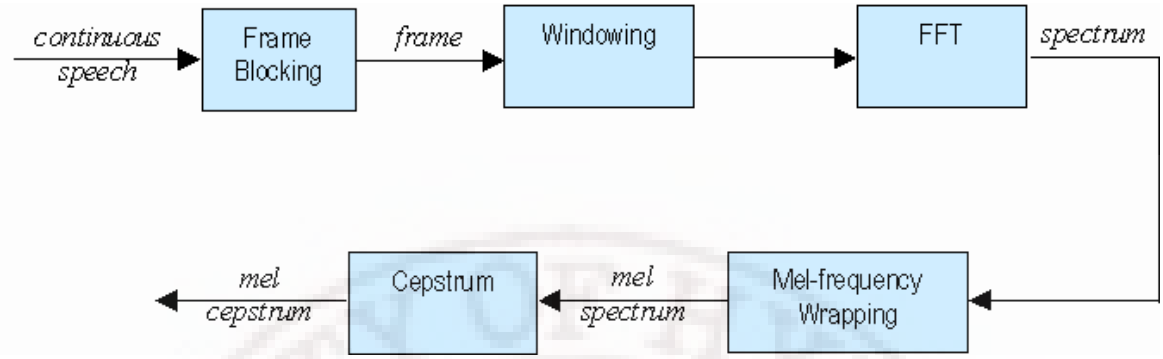

Fig. 3.5.1 Data flow diagram for MFCC Extraction

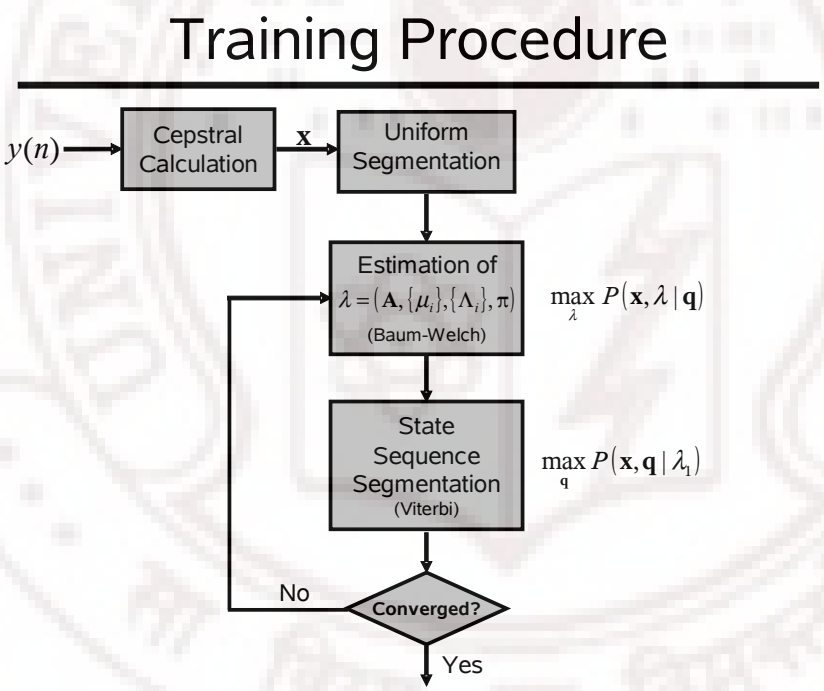

Fig. 3.5.2. Speech Recognition Training Using HMM

3.6 Sphinx User Manual:

Before Training:

1. Download TUTORIAL.tar.gz from

2. http://www.cs.cmu.edu/`robust/Tutorial

Unzip downloaded file using the command

tar -zxvf singlemachine.tar.gz.

This will give the directory called TUTORIAL in this base directory it will install all the files necessary to train and test the Sphinx.

Note: We have to install in root privilege only.

3. In TUTORIAL create new experiment by giving the command

./create_newexpt.csh file1 [file name]

4. In this file all the files required to train and decode the system will be loaded.

- Copy the ***until*** directory *from the Sphinx3* to this file.
- Create new ***feature_files*** folder by deleting existing feature_files folder.
- Replace ***s3trainer*** folder from previous experiments.
- Create train and test directory in file1 experiment.
- Copy "***wavtoraw.csh***" & "***dictophn.c***" program from previous experiments or create as your own.

In this train folder

- create a folder named wav and paste all wav files. And create a folder named raw
- Raw files, this will get by running the program "wavtoraw.csh" by giving the wav files path and giving the path where raw files should be stored as input to this program.

  Ex: In the way to wavtoraw.csh program

  set wavedir=/home/sphinx/TUTORIAL/file1/train/wav

  set rawdir=/home/sphinx/TUTORIAL/file1/train/raw

  file1>./wavtoraw.csh

  now we will get all raw files in the raw folder in train.

5.Write:

a. text file: which is exact pronunciation of a word. Then send it to

http://www.speech.cs.cmu.edu/tools/lmtool.html

Then you get

b. sentence file

Ex: <s> amma </s> I.e., filename.sent

c. Dictionary file

Ex: amma am mma I.e., filename.dic

d. Language model (lm) file

I.e., filename.lm

e. write transcription file manually all follows

Example of transcription file: one space <s> consonant/words/sentences then space </s> (name01)

Ex: <s> A </s> (akbar01) I.e., filename.trans

f. create filename.phonelist using phone list, this will get by running the program "dictophn.c"by giving the dictionary file as input to it.

Then compile it using

train>cc dictophn.c

- Then execute it using

./a.out

(or)

./a.out | tee fname.txt

Then u will get phone list at the command prompt and save that list in a new text file and give name as filename.phonelist

Note: Eliminate duplication of phonemes in phone list. using command

#sort -u filenm.phonelist

Training:

1..Create the control raw file using the command train#

ls /home/sphinx/TUTORIAL/file1/train/raw/*.raw>file.ctlNote: go c_scripts->compute_mfcc.csh2.Copy compute_mfcc.csh file from c_scripts in the railway folder into the current working c_scripts(replace it). Run the file script in c_scripts by the command

c_scripts#] ./compute_mfcc.csh /home/sphinx/TUTORIAL/file1/train/file.ctl

Note: Before compute mfcc change the path in compute_mfcc.csh.

3.Mfcc files are created in the folder 'raw' in the feature_files folder of your experiment.

4.Create the control mfcc file using the command

expt#ls /home/sphinx/TUTORIAL/exp/feature_files/raw/*.mfc>mfc.ctl

then in mfc.ctl file delete/replace all /home/sphinx/TUTORIAL/file1/feature_files/ and **.ctl** extension before& after each line with empty space then we can get as **raw/***.

Note: * means filename, then save it.

5. Copy lm3g2dmp from any directory in sphinx software, then extract in test folder.

Ex: ./lm3g2dmp /home/sphinx/TUTORIAL/speech0/train/ak.lm /home/sphinx/TUTORIAL/speech0

6.Change the paths in the file called variables.def contained in the c_scripts as per files located.

- Go to c_scripts file, change the command for all the scripts contained in the directories 01* through 05*, by keeping unlimit as #unlimit.

Note: We may copy 01* to *05 files from previous experiments.

7.Then run the slave*.csh script(in all starting from 01* through 05*) as below. If the input is error free, then we get number of files by running each script. The errors will be checked in the log directory contained in file1 directory.

Remove all temporary files in c_scripts folder as follows:

c_scripts # ls

01.ci-chmm 05.cd-chmm create_newexpt.csh .variables.def.swp

02.cd_untied cleanall.csh runall.csh

03.buildtrees compute_mfcc.csh variables.def

04.tiestate compute_mfcc.csh~ variables.def~

c_scripts # rm *.*~

c_scripts # ls

01.ci-chmm 04.tiestate compute_mfcc.csh variables.def

02.cd_untied 05.cd-chmm create_newexpt.csh .variables.def.swp
03.buildtrees cleanall.csh runall.csh

Note: Remove all modified files like *.*~ as above shown commands.

Running slave.csh files from each folder of 01* to 05* in c_script folder, as follows:*

linux-5qm4:/home/sphinx/TUTORIAL/03062008/c_scripts # cd 01.ci-chmm/
linux-5qm4:/home/sphinx/TUTORIAL/03062008/c_scripts/01.ci-chmm
./slave_convg.csh
USAGE: ./slave_convg.csh <iteration number (def 1)>
Setting iter value to 1
Continue? (y/n) y
Cleaning up accumulator directories...
/bin/rm: No match.
Cleaning up log directories...
/bin/rm: No match.
Cleaning up qmanager directories...
/bin/rm: No match.
/bin/rm: No match.
/bin/rm: No match.
Cleaning up model directories..
/bin/rm: No match.

linux-5qm4:/home/sphinx/TUTORIAL/03062008/c_scripts/01.ci-chmm #
cd ../02.cd_untied/
linux-5qm4:/home/sphinx/TUTORIAL/03062008/c_scripts/02.cd_untied
./slave_convg.csh
USAGE: ./slave_convg.csh <iteration number (def 1)>
Setting iteration value to 1
Continue? (y/n) y
Cleaning up accumulator directories...

Cleaning up log directories...

Cleaning up qmanager directories...

linux-5qm4:/home/sphinx/TUTORIAL/03062008/c_scripts/02.cd_untied #
cd ../03.buildtrees/

linux-5qm4:/home/sphinx/TUTORIAL/03062008/c_scripts/03.buildtrees
./slave.treebuilder.csh

Phone = AA, State = 0 - completed

Phone = AA, State = 1 - completed

Phone = AA, State = 2 - completed

..

..

Phone = Z, State = 2 – completed

linux-5qm4:/home/sphinx/TUTORIAL/03062008/c_scripts/03.buildtrees #
cd ../04.tiestate/

linux-5qm4:/home/sphinx/TUTORIAL/03062008/c_scripts/04.tiestate
./slave_tiestate.csh

/bin/rm: No match.

/bin/rm: No match.

/bin/rm: No match.

linux-5qm4:/home/sphinx/TUTORIAL/03062008/c_scripts/04.tiestate # cd ../05.cd-chmm/

linux-5qm4:/home/sphinx/TUTORIAL/03062008/c_scripts/05.cd-chmm
./slave_convg.csh

USAGE: ./slave_convg.csh <ngau (def 1)> <iteration number (def 1)>

Setting ngau to 1, iteration no. to 1

Continue? (y/n) : y

Assuming initial models are 1 gaussian per state

Setting iter value to 1

Cleaning up accumulator directories...

/bin/rm: No match.

Cleaning up log directories...

Cleaning up qmanager directories...

(Or)

Simply Run

c_scripts # ./runall.csh

All the above operations can be done.

Copy **lm3g2dmp** from any directory in sphinx software, then extract

Ex: ./lm3g2dmp /home/sphinx/TUTORIAL/speech0/train/ak.lm /home/sphinx/TUTOR IAL/speech0/train

Decoding:

1.Go to decoding directory then give the command

- # ./launch_decode.ci.1gaumodels

2.Then run

#./compute_acc.ci.csh

WORD ACCURACY= 100.000% (150/ 150) ERRORS= 0.000% (0/150)

Chapter4

Present Work

I tried with various approaches to improve the speech recognition accuracy for Telugu sentences. The approaches are

- Calculating distance using Levenshtein distance algorithm and minimum distance variants are added to the Static Dictionary.
- Addition of the frequently occurring errors.
- Addition of variant in Language model.
- Changing the probability.
- Transcription Modification.

4.1 Levenshtein Distance Algorithm:

Levenshtein distance algorithm is to calculate the distance between the variants. The variants which are having minimum distance will be added to the Static Dictionary. Then we can observe the improved accuracy.

Algorithm

Step	Description
1	Set n to be the length of s. Set m to be the length of t. If n = 0, return m and exit. If m = 0, return n and exit. Construct a matrix containing 0..m rows and 0..n columns.
2	Initialize the first row to 0..n. Initialize the first column to 0..m.
3	Examine each character of s (i from 1 to n).
4	Examine each character of t (j from 1 to m).
5	If s[i] equals t[j], the cost is 0. If s[i] doesn't equal t[j], the cost is 1.
6	Set cell d[i,j] of the matrix equal to the minimum of:

	a. The cell immediately above plus 1: d[i-1,j] + 1. b. The cell immediately to the left plus 1: d[i,j-1] + 1. c. The cell diagonally above and to the left plus the cost: d[i-1,j-1] + cost.
7	After the iteration steps (3, 4, 5, 6) are complete, the distance is found in cell d[n,m].

Example

This section shows how the Levenshtein distance is computed when the source string is "GUMBO" and the target string is "GAMBOL".

Steps 1 and 2

		G	U	M	B	O
	0	1	2	3	4	5
G	1					
A	2					
M	3					
B	4					
O	5					
L	6					

Steps 3 to 6 When i = 1

		G	U	M	B	O
	0	1	2	3	4	5
G	1	0				
A	2	1				
M	3	2				
B	4	3				
O	5	4				
L	6	5				

Steps 3 to 6 When i = 2

		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1			
A	2	1	1			
M	3	2	2			

B	4	3	3			
O	5	4	4			
L	6	5	5			

Steps 3 to 6 When i = 3

		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1	2		
A	2	1	1	2		
M	3	2	2	1		
B	4	3	3	2		
O	5	4	4	3		
L	6	5	5	4		

Steps 3 to 6 When i = 4

		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1	2	3	
A	2	1	1	2	3	
M	3	2	2	1	2	
B	4	3	3	2	1	
O	5	4	4	3	2	
L	6	5	5	4	3	

Steps 3 to 6 When i = 5

		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1	2	3	4
A	2	1	1	2	3	4
M	3	2	2	1	2	3
B	4	3	3	2	1	2
O	5	4	4	3	2	1
L	6	5	5	4	3	2

Step 7

The distance is in the lower right hand corner of the matrix, i.e. 2. This corresponds to our intuitive realization that "GUMBO" can be transformed into "GAMBOL" by substituting "A" for "U" and adding "L" (one substitution and 1 insertion = 2 changes).

4.2 Addition of the frequently occurring errors:

In the result file we get confusion pairs with number of times the error was repeated. I took frequently occurring errors and added to the Static Dictionary. Then I got the improved Accuracy.

4.3 Addition of variant in Language model:

I also tried to include the variant in the Language model also. But i got reduced accuracy. So i did not try this procedure later.

4.4 Changing the probability:

I tried to change probabilities of the states, because I want to whether the accuracy will be increased or decreased. But i could unable to open the following files in the folders which are in model_parameters.

(i) means.

(ii) mixture_weights.

(iii) transition_matrices.

(iv) variances.

4.5 Transcription Modification:

when I get some type of errors , I modified the Transcription for those words, i observed improved accuracy. The following are the Examples of errors.

MEEREKKADA	-	MEE
EKKADIKI	-	EKKADA
HYDERAABAD	-	MEERAEMI
BHAARATHADESAM	-	BHAARATHEEYULANDARU

The modified Transcription words are

MEERREKKADA

EKKADAKKI

HHYDERAABAD

BHAARATTHADESAM

In all the approaches, in which I succeeded one main observation is that when I add new variant to the dictionary, there is reduced error rate. Which is contradiction to other papers [3-5]. For all approaches initially I used Praat to eliminate the noise present in the wave files. if we have noise in the wave files I got insertion errors, because of these errors the error rate is increasing. After deleting noise using Praat the error rate is decreased.

Chapter 5

Results

The following are the Results when the wave files are noisy.

EXPERIMENT NAME	WORD ACCURACY(%)	ERRORS(%)
50	61.364	59.091
51	88.696	21.739
52	84.348	31.304
53	80.870	65.217
54	93.913	16.522
55	87.826	24.348
56	73.913	68.696
57	96.522	16.522
58	89.565	18.261
59	77.391	61.739
60	88.696	38.261
61	62.609	95.652
62	90.435	21.739
63	93.043	16.522
64	93.043	11.304
65	92.174	26.087
66	81.739	20.870
67	91.304	20.000
68	93.913	12.174
69	66.957	72.714
70	83.478	24.348
71	83.478	36.522
72	76.522	31.304
73	55.000	50.833

After eliminating the noise using Praat the results are as follows

EXPERIMENT NAME	WORD ACCURACY(%)	ERRORS(%)
50	80.870	40.000
51	88.696	19.130
52	86.957	21.739
53	81.739	45.217
54	91.304	18.261
55	92.174	14.783
56	87.826	21.739
57	94.783	12.174
58	93.913	10.435
59	84.348	37.391
60	96.522	10.435
61	73.913	53.043
62	92.174	13.043
63	93.043	12.174
64	92.174	13.043
65	94.696	22.957
66	81.739	20.870
67	93.043	14.783
68	93.913	12.174
69	78.261	51.304
70	80.870	29.565
71	85.217	21.739
72	86.957	19.130
73	93.913	9.565

After addition to the Dictionary the results are as follows

EXPERIMENT NAME	WORD ACCURACY(%)	ERRORS(%)
50	99.130	22.609
51	95.652	12.174
52	98.261	10.435
53	96.522	28.696
54	99.130	10.435
55	97.391	8.696
56	97.391	16.522
57	98.261	8.696
58	99.130	5.217
59	96.522	23.478
60	99.130	7.826
61	94.696	30.739
62	98.261	6.957
63	98.261	6.957
64	96.522	8.696
65	100.000	15.652
66	94.783	7.826
67	99.130	6.957
68	97.391	6.957
69	93.913	36.522
70	93.913	16.522
71	93.913	13.913
72	96.522	9.565
73	99.130	4.348

TRAINING (number of speakers)	TESTING (number of speakers)	ACCURACY	ERRORS	AFTER DICTIONARY ADDITION	
				ACCURACY	ERRORS
12	12	51.159	85.797	69.420	88.478
16	8	58.370	71.196	76.413	65.000
20	4	59.826	73.696	79.130	65.870

From the above table we can observe that the error rate is decreasing when new variants are added to the dictionary.

Conclusion and Future work

Conclusion:

The approaches discussed in this dissertation working well. Initially the wave files are noisy. At both ends of the sentence the noise was removed. But sound recorder can remove noise present at both the ends, not in the middle, Praat is used to remove the noise present in the middle part of wave files. When such wave files are trained and tested with Sphinx tool. Then obtained confusion pairs re added to the Dictionary. By using the approaches discussed in this dissertation, finally improved accuracy is observed.

Future Work

The speech database consisted of 24 speaker's voice and each speaker spoken 40 sentences. The accuracy can be verified with large number of speakers and large database.

References:

[1] Gopala krishna Anumanchipalli, "Modeling Pronunciation Variation for Speech Recognition", M.S Thesis , IIIT Hyderabad, February 2008.

[2] Eric John Fosler-Lussier, "Dynamic Pronunciation Models for Automatic Speech Recognition", Ph.d Thesis, University of California, Berkeley, CA, 1999.

[3] Gustavo Hernandez-Abrego, Lex Olorenshaw, "Dictionary Refinement based on Phonetic Consensus and Non-uniform Pronunciation Reduction", INTERSPEECH 2004 -ICSLP 8th International Conference on Spoken Language Processing, October 4-8, 2004.

[4] Hua Yu and Tanja Schultz, "Enhanced Tree Clustering with Single Pronunciation Dictionary for Conversational Speech Recognition", in Proceedings of Eurospeech, pp. 1869-1872, 2003.

[5] Dan Jurafsky, Wayne Ward, Zhang Jianping, Keith Herold, Yu Xiuyang, and Zhang sen, "What Kind of Pronunciation Variation is Hard for Triphones to Model", in proceedings of ICASSP, pp. 577-580, 2001.

[6] Adda-Decker M. and Lamel L., "Pronunciation Variants Across System Configuration", Speech Communication, 1999.

[7] J. M. Kessens, M. Wester, and H. Strik, "Improving the Performance of a Dutch csr by Modeling within-word and cross-word Pronunciation Variation", Speech Commun., vol. 29, no. 2-4, pp. 193–207, 1999.

[8] R. Rosenfeld, "Optimizing Lexical and N-gram Coverage via Judicious use of Linguistic data", in Proc. Eurospeech, (Madrid), 1995.

[9] Sk . Akbar, “ Comparing Speech Recognition Accuracy using HM Ms and Multi-layer Perceptrons ”, M.Tech dissertation, June 2008.

[10] kenneth A. Kozar, “Representing Systems with Data Flow Diagrams”, Spring, 1997. http://spot.colorado.edu/~kozar/DFD.html

[11] Finke Michael and Waibel Alex, “Speaking Mode Dependent Pronunciation Modeling in Large Vocabulary Conversational Speech Recognition”, in Procedures of Eurospeech, 1995.

[12] M . Ravishankar and M. Eskenazi, “Automatic Generation of Context-Dependent Pronunciations”, in Proc. Eurospeech ’97, (Rhodes, Greece), pp. 2467–2470, 1997.

[13] T. Sloboda and A. Waibel, “Dictionary Learning for Spontaneous Speech Recognition”, in Proc. ICSLP ’96, (Philadelphia), pp. 2328–2331, 1996.

[14] A. Xavier and D. Christian, “Improved Acoustic-Phonetic Modeling in Philip's Dictation System by Handling Liaisons and Multiple Pronunciations” , in Proc. Eurospeech ’95, (Madrid), pp. 767– 770, 1995.

[15] P. S. Cohen and R. L. Mercer, “The Phonological Component of an Automatic Speech Recognition System ”, Reddy, D. R. (Ed) Speech Recognition. Invited Papers Presented at the 1974 IEEE Symposium., pp. 275–319, 1975.

[16] N. Cremelie and J.-P. Martens, “In Search of Better Pronunciation Models for Speech Recognition”, Speech Communication, vol. 29, no. 2-4, pp. 115–136, 1999.

[17] B. C. S. and Y. S. J., “Pseudo-Articulatory Speech Synthesis for Recognition using Automatic Feature Extraction from X-ray Data”, in Proc. ICSLP ’96, (Philadelphia), pp. 969–972, 1996.

[18] I. Amdal, F. Korkmazskiy, and A. C. Surendran, “Joint Pronunciation Modeling of

Non-native Speakers using Data-Driven Methods", in Proc. ICSLP '00, (Beijing, China), pp. 622–625, 2000.

[19] M. Bacchiani and M. Ostendorf, "Joint Lexicon, Acoustic Unit Inventory and Model Design", Speech Commun., vol. 29, no. 2-4, pp. 99–114, 1999.

[20] T. Fukada, T. Yoshimura, and Y. Sagisaka, "Automatic Generation of Multiple Pronunciations based on Neural Networks", Speech Commun., vol. 27, no. 1, pp. 63–73, 1999.

[21] S.Greenberg, "Speaking in Shorthand – A Syllable-centric Perspective for Understanding Pronunciation Variation", in Proc. of the ESCA Workshop on Modeling Pronunciation Variation for Automatic Speech Recognition, (Kekrade, Netherlands, May 1998. ESCA.), 1998.

[22] T. Holter and T. Svendsen, "Maximum Likelihood Modeling of Pronunciation Variation", Speech Commun., vol. 29, no. 2-4, pp. 177–191, 1999.

[23] H. Strik and C. Cucchiarini, "Modeling Pronunciation Variation for ASR: a survey of the literature", Speech Commun., vol. 29, no. 2-4, pp. 225–246, 1999.

[24] M. Riley, W. Byrne, M. Finke, S. Khudanpur, A. Ljolje, J. McDonough, H. Nock, M. Saraclar, C. Wooters, and G. Zavaliagkos, "Stochastic Pronunciation Modeling from Hand-Labeled Phonetic Corpora", Speech Commun., vol. 29, no. 2-4, pp. 209–224, 1999.

[25] Grace Chung and Staphanie Seneff and Chao Wang and I. Hetherington, "A Dynamic Vocabulary Spoken Dialogue Interface", in Proc. ICSLP, (Jeju Island), 2004.