Speech Recognition of Hearing Impaired Children

By

Sidrah Azhar

MSIT-17-172437

Supervisor

Dr. Sharifullah Khan

Co-supervisor

Dr. Ali Tahir

School of Electrical Engineeringand Computer Science (SEECS)

National University of Sciences and Technology (NUST)

Islamabad, Pakistan

May 2019

This thesis is dedicated to *my beloved parents*

Abstract

In this world of digital technologies, such as artificial intelligence and voice-driven applications, Automatic Speech Recognition (ASR) is becoming essential to our daily life activities specially for disabled people. Now a days, people who are hearing impaired use lipreading visual cues for speech therapy. For our research work, data was acquired from speech therapy sessions of hearing- impaired children. In these sessions, hearing impaired children learnt how to articulate English words through speech therapy. Language therapy of single words as well as pronunciation of short English sentences were practiced by lip reading visual cues. The improvement in speech therapy sessions can be made possible by checking the accuracy of articulated words. The goal of our research is to build speech recognition engine for abrupt and disrupted speech of hearing impaired. We have checked how accurately those words were articulated. Depending on the accuracy achieved by recognition results, therapists can make hearing impaired children practice those words again which are not correctly pronounced in prior sessions. Hearing impaired individuals have their own way of pronouncing the words of English due to the abrupt and disrupted speech of hearing impaired. Hence their language model (LM) and dictionary is also unique. We built a language model (LM) for disrupted and cumbersome speech of hearing impaired. We have checked the impact of different model sizes of acoustic model on accuracy of isolated recognized words. We used Kaldi speech recognition toolkit for building our system. In this model, we have checked the performance measure that is, Word Error Rate (WER) based on isolated word recognition. The accuracy of 61% was achieved by the proposed recognizer. We have validated our system performance in comparison with the Google Cloud speech-to-text API built for normal hearing with speech data of hearing impaired. The difference of 45% accuracy in existing work was observed. The recognition results can be further improved with increase in data size. Graphical user interface can be built in future to facilitate hearing

impaired children and instructor.

Acknowledgments

I owe my deepest gratitude to Allah Almighty. I would also like to acknowledge the efforts of my parents to make such opportunity available for me. I am very thankful to Dr. Sharifullah Khan and Dr. Ali Tahir for patiently guiding me through out the process. I want to thank my friend Ms. Sahar Shafait for supporting and sharing her valuable advice. I would like to thank everyone who directly or indirectly contributed in this journey.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

In this world of digital technologies, artificial intelligence and voice-driven applications like Siri [1], Amazon Alexa [2] and Cortana [3], Automatic Speech Recognition (ASR) is becoming essential to our daily life activities specially for disabled people. This chapter present the overview of the research carried out for building Automatic Speech Recognition (ASR) system for Hearing Impaired speech.

The breakdown of this chapter is as follows. First the chapter describes the background of the research carried out followed by the motivation for building Automatic Speech Recognition (ASR) system for Hearing Impaired people. Then problem statement and need of this research followed by research objectives. Finally we conclude with the proposed solution and organization of the thesis.

1.1 Significance of this research

According to World Health Organization (WHO), almost 5% of the people are Hearing Impaired [4]. For these people, Automatic Speech Recognition (ASR) provides significant help, so they can communicate effectively with other people using computer with voice input [5]. So this research focuses on providing a unique solution to profound hearing impaired children by building a Speech Recognition system on speech data of hearing impaired. We designed a Language Model (LM) for the speech data of hearing impaired. Since the language model is different for the pronunciation of hearing impaired disrupted and abrupt speech.

Hearing disability can affect a child's development of speech and communication skills

[6]. If hearing disability is detected and managed in the initial phase then a child can overcome its language impairment and become a good communicator.

There is an application that uses Automatic Speech Recognition (ASR) system build for hearing impaired [7]. It uses the speech data of hearing impaired. Google cloud speech-to-text API was used. Performance was low due to the fact that speech data of hearing impaired was inserted into Automatic Speech Recognition (ASR) that are build for normal hearing person. There is another research work carried out on the speech data of hearing impaired which only uses total of five words (first five digits) as their input to Automatic Speech Recognition (ASR) and even the performance was low [8]. Another research was accomplished on speech data of hearing impaired by Kanwal Yousaf et al [9]. As a performance metric, they used accuracy in terms of precision and recall which is not suitable for Automatic Speech Recognition (ASR) systems.

For this research work, speech therapy sessions was held for hearing impaired children. In these sessions, hearing impaired children learned how to articulate English words through speech therapy.

Language therapy of single word pronunciation of English was practiced. The structure of the words were based on the placement of vowels in the words that is, in the beginning, middle and end of the word like apple, Lozina and purple respectively. Total of 72 words were used as shown in table. Lip-reading videos were shown for each word pronounced to hearing impaired children and asked them to practice those words with instructor and therapist. The age group of the hearing impaired children were 5-8 years (both genders; male and female).

For this speech data that we acquire [10], we want to check how accurately those words were articulated. Depending on the accuracy achieved by recognition results, therapist can make hearing impaired children practiced those words again which were not correctly pronounced.

1.2 Automatic Speech Recognition (ASR)

Automatic Speech Recognition (ASR) systems allows to convert speech or utterances into its corresponding text [11]. Text can be either words or syllables or any other sub-word units like phones. Daily life examples are like YouTube captioning [12], Dictation

Table 1.1: Words used in data set

A	Ankle	Ant	Apple	Arm	Arrow
Banana	Blue	Cat	Chili	Cow	Cup
Dog	Dot	E	Ears	Egg	Elephant
Elbow	Eyes	Fig	Glue	Gorilla	Hut
I	Ice	Igloo	Ill	Ink	Insect
Jam	Jet	Jug	Kiwi	Knee	Koala
Leg	Lid	Lip	Lit	Lozina	Mango
Map	Mug	O	Old	Orange	Oval
Oven	Owl	Peg	Pot	Potato	Purple
Queue	Rat	Red	Rod	Table	Tap
Tomato	Tub	Two	U	Umbrella	Umpire
Under	Up	Vase	Wet	Zero	Zip

systems [13].

The very first component of Automatic Speech Recognition (ASR) is an acoustic analysis component which look at the raw speech waveform of audio track and convert it into discrete representation, like Mel Frequency Cepstral Coefficient (MFCC) features [14]. The mapping of words to discrete unit and distinctive unit of language (Phonemes) was written down by experts of the language. For English language, there is CMU online dictionary [15] which gives pronunciation of commonly occurring words. It has around 150,000 words.

The acoustic model takes the MFCC features and maps into a sequence of phonemes in order to interpret all the different little parts that make up a waveform [16]. The acoustic model looks at the fine detail of the signal and provides the foundation for everything that comes after it. In order to model the phone units in the acoustics, there is a technique called Hidden Markov Model which is a way of breaking the signal label, down into three units per phone, the beginning, middle, and end of the sound [17]. These labels form the target for the training algorithms. For more advanced models, instead of phonetic units, triphone units [18] are used. Triphones are just like phones but they are aware of the phonetic context they are in, because when we are speaking fluently, the target sounds we make for a particular phone change based on the context around it.

The pronunciation model provides link between sub-word units (phonemes) and the word [19]. A dictionary of pronunciations is maintained. The final model that is N-Gram language model [20] tells how likely is it for different words to occur given the recent word context. SRI Language Modeling Toolkit [21] to design language model.

Now the recognizer forms a decoding graph which search through the large graphs and throw out what is the most likely word sequence to corresponding test utterance [17]. Speech recognition decoding is the process of finding the word sequence that jointly maximizes the language model score and acoustic model score. It is a path search algorithm through the decoding graph, where the score of the path is the sum of the score given to it by the decoding graph, and the score given to it by the acoustic model [22].

This is how the speech recognition of hearing impaired works. To train them from data, and to bring these different models together in a recognizer.

1.3 Motivation

According to Don Grushkin, Automatic Speech Recognition (ASR) for hearing impaired is a one-way process [23]. Since there are many applications and speech recognition systems like Text Hear [1], which is a speech recognition application made for hearing impaired. It listens the voice of normal hearing person and convert it into text format for hearing impaired. It convert voice to text in real time.

Another application on Apple store is Hearing Helper [2] which is made available for deaf. It is a speech-to-text application used for captioning to help deaf which uses Siri [1] as its speech recognition system. It also translates the voice of normal hearing person. There is another speech-to-text recognition system called Ava [3] and SpeeakSee [4] which are used for group conversations. They provides captions to hearing impaired by translating the voice of a normal hearing person.

The fact is that they all takes the input as the voice of normal hearing person and the translation of normal hearing person's voice displayed to hearing impaired. But these

[1] https://texthear.com/
[2] https://itunes.apple.com/us/app/hearing-helper/
[3] https://www.ava.me/
[4] https://speak-see.com/

Automatic Speech Recognition (ASR) systems does not cater the process other way around, that is, if hearing impaired wants to reply back to normal hearing person. From all perspective, hearing impaired have to write down his reply in return.

The analyses on speech recognition systems for hearing impaired showed that there is a need to build a speech recognition system on the speech data of hearing impaired so that they can also communicate effectively with normal hearing person.

1.4 Problem Statement

Every unique language has its Language Model (LM). All languages have their own set of words and their pronunciations. Therefore hearing impaired have their own way of pronouncing the words of English. Due to the abrupt and disrupted speech of hearing impaired. Hence their language model (LM), lexicon and dictionary is also different.

The main concern is that the speech recognition systems that are build for hearing impaired uses the speech data of normal hearing. Therefore the speech recognition systems are trained and tested on speech data of normal hearing person. There are some research carried out that on speech recognition systems that build on speech data of hearing impaired.

There is a need to recognize the speech of hearing impaired. To check how accurately the words of English are pronounced by hearing impaired. Depending on the accuracy achieved by recognition results, therapist can make hearing impaired children practiced those words again which were not correctly pronounced.

Speech recognition technology offers a new approach to the development of communication solutions for hearing-impaired. Language model (LM) is designed for every unique language according to its words and pronunciations. We must try to recognize the speech of profound hearing impaired.

The articulation of words by hearing impaired, during speech therapy sessions are single words of English. We have to recognize how accurately words pronounced by hearing impaired were articulated. We can check the percentage of correctness of isolated words since the nature of data is pronunciation of isolated words.

The research that have already been carried out has also some flaws like inaccurate performance metric, data set size, nature of data, feature sets and performance of Au-

tomatic Speech Recognition (ASR) system has to be checked and improved. The main concern is that these speech recognition systems are build for normal hearing persons and uses the speech data of normal hearing persons. Therefore these speech recognition systems are trained and tested on the speech data of normal hearing persons.

1.5 Research Objectives

This research aims to design an Automatic Speech Recognition (ASR) system to recognize the abrupt and disrupted speech of hearing impaired children. Our main focus is on designing a Language Model (LM) for the speech of hearing impaired children which is abrupt and disrupted. Their way of pronouncing the English words are different. Hence different lexicon and dictionary is prepared. These are the following objectives that we want to fulfil in our research.

- To design a Language Model (LM) for disrupted speech of Hearing Impaired
- To check the impact of different model sizes of acoustic model on accuracy of recognized words
- To build a speech recognition system (ASR) for cumbersome and complicated speech of hearing impaired children
- To check the performance measure on the basis of isolated word recognition
- Validating the system performance in comparison with the Google Cloud speech-to-text API build for normal hearing with speech data of hearing impaired

1.6 Proposed Solution

This research aims at building a speech recognizer for the speech of hearing impaired children. It focuses on applying the Artificial Intelligence (AI) techniques on speech data of hearing impaired. Due to the cumbersome and complicated speech of hearing impaired, there are some research carried out on speech recognition of hearing impaired. Therefore the first piece of work is to build the ground truth that is, the transcriptions

for the speech of hearing impaired since listening to the voice of deaf children for the utterance transcription are tough and challenging. Due to the abrupted speech of hearing imapired.

The crucial stage of machine learning is to clean data. Hence for data preprocessing we will do data cleaning which includes smoothing the noisy data from the audio clips which contained background noise of instructor talking with deaf during sessions and other interference's. The other problem is the microphone frequency problem due to the low quality of mobile microphone and not a good quality set-up which will be solved by amplification of each audio track.

Our main focus will be on designing a Language Model (LM) for the speech of hearing impaired children. Their way of pronouncing the English words are different and hence they have different lexicon and dictionary. Beside using CMU online dictionary [24] which is build for normal hearing person, we have designed our own dictionary and hence a Language Model (LM).

We will use Kaldi [17] speech recognition toolkit for building our system. For model training we will use Gaussian Mixture Model and Hidden Markov Model (GMM-HMM) as an acoustic model. Monophone and triphone training with MFCC features will be used in Kaldi setup. For validation of results we will use a speech recognizer which is built for normal hearing person that is, Google Cloud speech-to-text API.

1.7 Organization of thesis

The next chapter 2 explain the overview of the Automatic Speech Recognition (ASR) and all its components. Models that are, acoustic model, pronunciation model, language model. Recognizer, and how all these components make up to one speech recognizer.

Chapter 3 discusses some recent research carried out on speech data of hearing impaired. Speech recognizer built for deaf and hard of hearing. Then chapter 4 discusses methodology used to build our own speech recognizer. It explains the speech data of hearing impaired. It explains all the methods used, from preprocessing to validation of results.

Chapter 5 discusses the results obtained from our own designed speech recognizer. The results obtained from validation. Finally the last chapter 6 concludes the whole research with limitations and future work.

Chapter 2

Background

In this world of digital technologies, artificial intelligence and voice-driven applications like Siri and Alexa, Speech recognition is becoming essential to our daily life activities. This chapter is divided into an introduction and demand of ASR system to daily life activities. Then structure of ASR system is explained. And finally, performance measurement parameters of ASR is discussed.

2.1 Introduction to ASR system

Automatic Speech Recognition (ASR) systems allows to convert speech or utterances into its corresponding text [11]. Text can be either words or syllables or any other sub-word units like phones. Daily life examples are like YouTube captioning, Dictation systems.

2.1.1 Phonetics

Phonetics is the part of linguistics that focuses on the study of the sounds produced by human speech [25]. It encompasses their production (through the human vocal apparatus), their acoustic properties, and perception. The atomic unit of speech sound is called a phoneme. Words are comprised of one or more phonemes in sequence, if you change it, it can change the meaning of a word like “pat” to “bat”.The acoustic realization of a phoneme is called a phone. One major way to categorize phonemes is into vowels and consonants [26].

2.1.2 Vowels

Vowels can be distinguished by two attributes. First, they are voiced sounds, meaning that the airflow from the vocal chords into the mouth cavity is created by the vibration of the vocal chords at a particular fundamental frequency (or pitch). Second, the tongue does not in any way form a constriction of air flow during production [27]. The placement of the tongue, lips, and jaw distinguishes different vowel sounds from each other. These different positions form different resonances inside the vocal tract called formants and the resonant frequencies of these formants characterizes the different vowel sounds.

2.1.3 Consonants

Consonants are characterized by significant constriction of air flow in the airway or mouth. Like vowels, some consonants can be voiced, while others are unvoiced. Unvoiced phonemes do not engage the vocal cords and therefore do not have a fundamental frequency or pitch. Some consonant phonemes occur in pairs that differ only in whether they are voiced or unvoiced but are otherwise identical. For example, the sounds /b/and /p/ have identical articulatory characteristics (your mouth, tongue, jaw are in the same position for both) but the former is voiced and the latter is unvoiced [27]. The sounds /d/ and /t/ are another such pair. One important aspect of phonemes is that their realization can change dependent on the surrounding phones. This is called phonetic context and it caused by a phenomenon called co-articulation. The process of producing these sounds in succession changes their characteristics. Modified versions of a phoneme caused by co-articulation are called allophones.

2.1.4 Syllables and Words

A syllable is a sequence of speech sounds, composed of a nucleus phone and optional initial and final phones. The nucleus is typically a vowel or syllabic consonant, and is the voiced sound that can be shouted or sung. As an example, the English word “bottle” contains two syllables. The first syllable has three phones, which are “b aa t” in the Arpabet phonetic transcription code. The “aa” is the nucleus, the “b” is a voiced consonant initial phone, and the “t” is an unvoiced consonant final phone. The second syllable consists only of the syllabic cosonant “l”. A word can also be composed of a

single syllable which itself is a single phoneme, e.g. “Eye,” “uh,” or “eau”. In speech recognition, syllable units are rarely considered and words are commonly tokenized into constituent phonemes for modeling.

2.1.5 Syntax and Semantics

Syntax describes how sentences can be put together given words and rules that define allowable grammatical constructs. Semantics generally refers to the way that meaning is attributed to the words or phrases in a sentence. Both syntax and semantics are a major part of natural language processing but neither plays a major role in speech recognition.

2.2 Structure of ASR system

The pipeline of what is typically in an ASR system is shown in figure 1.1 [28].

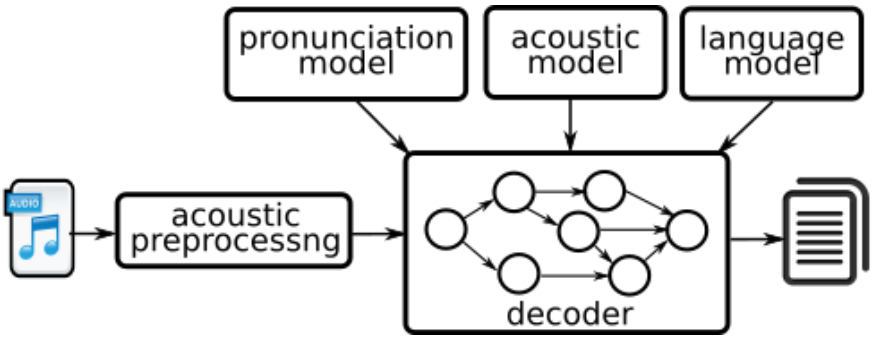

Figure 2.1: Architecture of Automatic Speech Recognition System

2.2.1 Acoustic Analysis

As speech sound is analog and computers are digital so, we need to digitize them. Speech sound waves propagate through the air and are captured by a microphone which converts the pressure wave into electrical activity which can be captured [29]. The electrical activity is sampled to create a sequence of waveform samples that describe the signal. Speech signals have less high frequency (only up to 8000 Hz) information so a sampling rate of 16,000 Hz is typically used. Speech over conventional telephone lines and most mobile phones is band-limited to about 3400 Hz, so a sampling rate of 8000 Hz is typically used for telephone speech. So there is an acoustic analysis component,

which sees the speech waveform and converts it into some discrete representation or features. So the features are then feed in to what is known as acoustic model. So the very first component is just looking at the raw speech waveform and converting it into some representation, like Mel Frequency Cepstral Coefficient (MFCC) features.

This is a very high level, so there is a lot of underlying machinery, so you have the raw speech signal. Which then is discretized, because you can't really work with a raw speech signal. So, you sample it and generate these discrete samples. And each of these samples are typically of the order of 10 to 25 milliseconds of speech. And the idea is that once you have each of these what are known as frames, speech frames, which are round 25 sometimes even larger. But typically 25 milliseconds. Then you can extract acoustic features, which are representative of all the information in the signal. And another reason why to discretized at particular sampling rates is that the assumption is that within each of these frames the speech signal is stationary. If speech signal is not stationary within the speech frames, then you can not effectively extract features form that particular slice. So, around 25 milliseconds of speech. So, we extract features, and this feature extraction is a very involved process, and it requires a long of signal processing know how. But this is also kind of motivated by how our ears work, so actually, one of the most common acoustic feature of presentations, which are known as Mel-frequency cepstral coefficients or MFCCs. They're actually motivated by what goes on in our ear or the filter banks, which apply in ear. So that might be too much detail, so let's just think of it as follows.

So we will start with the raw speech wave form. Then generate these tiny slices which are also known as speech frames. And now each speech frame is going to be represented as some real valued vector of features, which is capturing all the information and as well as possible is not redundant. So we do not have different dimensions here, which are redundant, so we want it to be as compact as possible. So, now we have these features for each frame, and now these are inputs to the next component in the ASR system, which is known as the acoustic model [30]. And it's a very important component. So, the process of Feature Extraction is,

- Speech is analyzed over short analysis window
- For each short analysis window, a spectrum is obtained using Fast Fourier Transfrom (FFT)

- Spectrum is passed through Mel-Filters to obtain Mel-Spectrum
- Cepstral analysis is performed on Mel-Spectrum to obtain Mel-Frequency Cepstral Coefficients

Thus speech is represented as a sequence of Cepstral vectors [31]. So, the basic units of Acoustic Information is phoneme. A phoneme is just a discrete unit and distinctive unit of language that can be used to differentiate between words. It is a speech sound in a language, which can be used to differentiate between words.

Each word can be represented as a sequence of phonemes like “five” consist of three phonemes and that is “f”, “ay”, and “v”. So, this mapping is written down by experts in the language. So, this pronunciation information is actually given to us by experts. So in English, of course, we have a very well developed pronunciation dictionaries, where experts have given us pronunciations corresponding to most of the commonly occurring words in English. And actually we have CMU to thank for that, so one of the most popularly used translation dictionaries in English is CMUdict, which is freely available [32]. And it has around 150,000 words. And that is actually one of the more extensive dictionaries. So typically, this is not a very easy resource to create. And it is clear why, because it is pretty tedious.

First of all, you need to find linguistic experts in the language. And then you need to find what are the most commonly occurring words in the language, and so on. So this takes time. And not many languages have very well developed pronunciation dictionaries. And most languages have around 20 to 60 phonemes. So the phoneme inventory size is roughly 20 to 60. And this is also something which needs to be determined by experts. So why are phonemes useful, why not just use words, instead of trying to split it into these sub- word units. So look at a very simple example.

So say that the speech wave form is the string of digits, five, four, one, nine. And during training, you have seen lots and lots of samples of five, four and one. So, now you can build models to identify when someone said five, and when someone said four, and when someone said one. But when it come to nine, during test time, when evaluating this particular speech utterance. If, during training, there is no nine. So now five is this sequence of phonemes “f”, “ay”, “v”, four is this sequence of phonemes “f”, “ow”, “r”, one is this sequence of phonemes “w”, “ah”, “n”, and so on. And now acoustic speech

samples, which correspond to each of these phonemes in the training. So during test time, when nine appears, you might be able to put together this string of phonemes “n”, “ay”, “n”. Because there is enough acoustic evidence for each of these individual phonemes. So it helps generalize, just to move to a more fine-grained inventory. But of course, with that said, if there is a very limited vocabulary task that are almost certain that during test time you are only going to get utterances, which are going to stick to that particular vocabulary. And if there is enough samples during training, you are probably well off just building word level models, and not even moving to the phoneme level. So that is why we need phonemes.

2.2.2 Acoustic Model

So as mentioned above there are acoustic features which we extract from raw speech signal. And we want the output on this acoustic model to be what is a likely phone sequence, which corresponded to the particular speech utterance.

So initially in the 80s and even now actually, Hidden Markov models is a paradigm which is used to learn this mapping of phones into HMM sequences. So now how to associate a set of frames, or set of features. Each of the acoustic feature vectors corresponds to a speech frame. Now we check how to chunk, how many frames are going to correspond to a particular phone.

There is a chain of what are known as hidden states, in the Hidden Markov Model [33]. So it is just like a graph. And it is a weighted graph, the weights are all probabilities. At each point one can transition to any phone, because initially do not know which phone is going to appear next. So the entire model is probabilistic. There is a need to have lots of hypotheses as to what the next phone could have been. And then there are estimates which say that for example initial ten frames most likely corresponds to a certain phone. And the transition probabilities, the probabilities which transition from one phone state to the other determine how many are going to chunk. How many speech frames are going to correspond to each phone. And once in the particular state, there are probabilities for having generated each of these vectors. And those probabilities come from what are known as Gaussian Mixture Models. So it is a probability distribution which determines that if in some particular state, that particular speech vector could be generated with a certain probability. And these probabilities are learned from training

data.

So, we have a probabilistic model, which maps sequences of feature vectors to a sequence of phonemes. So Hidden Markov Models were the state-of-the-art for a long time. And now, of course, we have Deep Neural Networks, which are used for a similar mapping. So now we have speech signal and again, we are extracting fixed windows of speech frames. Considering a speech frame and then look at a fixed window around it, a fixed window springs around it and generate, put all of these features together and that is the input to a deep neural network.

And the output is what is the most likely phone to have been produced, given this particular set of speech rates. But this is a posterior probability over the phones. But, again, also we get an estimate of what is the phone given a speech frame. What is the most likely phone. So it is actually a probability distribution over all the phones. And there are two ways in typically how Deep Neural Network (DNN) are used in acoustic models. So as mentioned above, when we have HMMs, we have the states and the probability distributions, which govern how the speech vectors correspond to a particular states. And these probability distributions are mixtures of Gaussian. So that is one way in which the HMM models can be combined and used within an acoustic model. So the idea is that we want to map acoustic features to phone sequences. And this is all probabilistic. So this is the only one best sequence but this is the likely probability distribution of phone sequences. So we have observation probabilities and observation probabilities are either they can be Gaussian mixture models or observation probabilities can be scaled posteriors from DNNs.

So that is the output from the acoustic model, which produces phone sequences.

2.2.3 Pronunciation Model

So now, we eventually want to get a word sequence, from the speech utterance. The pronunciation model provides link between sub-word units (phonemes) and the word [19]. Now how do we move from the phones to words. So as mentioned above that there are large pronunciation dictionaries. So this model, which provides a link between these phone sequences and words. So, typically just a simple dictionary of pronunciations is maintained. So we have these large dictionaries which say that the words correspond to these sequences of phones.

And this is the only module in an ASR system that is not learned. It is not learned from data. So the acoustic model was learned from training data, and we will come to the language model that is also learned from data. But the pronunciation model is actually expert derived. So an expert give these mappings.

2.2.4 Language Model

And the final model is what is known as the language model. As in the pronunciation model, the output was words. So now we have mapped a phone sequence to a particular word, and now the language model comes and says how should these words be ordered, according to a particular language [19]. So the language model looks at lots and lots of texts in that particular language. The statistical language models tells how likely is it for different words to occur given the recent word context.

Pr("she taught a class") > Pr("sheet or tuck lass")

So the thing is, we are not getting a single phone sequence. It is probabilistic. We have probabilities for every phone sequence appearing. And even if it does not exactly match, maybe it will exactly match it with a lower probability. But then the language model also comes in and then the probability when we add up, then we get the most likely sequence here. For example, if the word context is "dog", obviously the most likely next word to follow this particular word context is "ran", maybe even "can", but definitely not "pan". So, "pan" would have a very very low probability of following the "dog".

The language model is very crucial because it can be used to disambiguate between similar acoustics. For example if utterance is "baby crying", it could also very well map to this particular word sequence "bay bee crying", but obviously the first word sequence is much more likely. Because if we look at large volumes of English text, "baby crying" is probably a much more likely word ordering than "bay bee crying".

So SRILM, has lots of in-built functionalities implemented, this is a good tool kit to use [20].

Okay, so language models has many applications. Speech recognition is just one of them. Machine translation is another application where language models are heavily used. Handwriting recognition, optical character recognition, all of these also would use language models on either letters or characters. Spelling correction, again, language

models are useful here because you can have language models over your character space. Summarization, dialog generation, information retrieval, the list is really long. Language models are used in a large number of applications.

There is very popular language model which is used are NGram language models [20]. The idea is really straightforward. Look at co-occuring, either two words, or three words, or four words. If our n is two, we are looking at bigrams. If n is three, we are looking at trigrams, four-grams and so on. If we are already looking at five-grams, we can pretty much reconstruct English sentences really well. But of course then we are running into really, really large number of NGrams, as we increase the order of the NGram. So here we look at a four-gram.

Let us suppose that the four-gram is "she taught a class" then what is the probability of this particular four- gram. That is the word "class" follows, this particular word context "she taught a". We look at counts of, "she taught a class", in large volumes of English text. And then we normalize it with the count of, "she taught a", which is the word context.

2.2.5 Decoder

We looked into each of the individual components. But there is the main component that is actually a very important component.

The different parts of the ASR system which are giving various estimates of what is the most likely phoneme sequence. What is the most likely word sequence and so on. But finally we just want to get the most clear word sequence corresponding to speech utterance, and then it is a search problem. We have these various components, and now we need to search, putting all of them together, we need to search through this entire space.

So we start to a particular point and say that we only expect it to be nine or one, just these two words. Then we need to transition to nine. So here, every single arc does not have a weight but these are all weighted, because they all come with their associated probabilities. From start, we can transition in to either producing the word nine or one. But each nine is a sequence of phones, and each sequence of each phones, corresponds to its corresponding HMM. So, which has its own probabilities.

So this is quite a large graph just for these two words. And at least like a half decent system we will be looking at at least 20,000 or 40,000 words. So these are really large search graphs.

So, a vocabulary size of around 40,000 gives us a search graphs of the order of tens of millions of states. So these are really large graphs, and so now we need to search through these graphs and throw out what is the most likely word sequence to correspond to the speech utterance. So that is the decoder. This is the entire kind of pipeline of how an ASR system works.

2.3 Measuring Performance

When we build and experiment with speech recognition systems it is obviously very important to measure performance. Because speech recognition is a sequence classification task (in contrast to image labeling where samples are independent), we must consider the entire sequence when we measure error. To measure accuracy of Automatic Speech Recognition systems, we have Word Error Rate (WER). There are three types of errors a system can make,

- A substitution, where one word is incorrectly recognized as a different word
- A deletion, where no word is hypothesized when the reference transcription has one, and
- An insertion where the hypothesized transcription inserts extra words not present in the reference

The overall WER can be computed as

$$WER = \frac{N_sub + N_ins + N_del}{N_ref}$$

where Nsub, Nins, Ndel are the number of substitutions, Number of insertions and Number of deletions, respectively and Nref is the number of words in the reference transcription. Where Reference transcription is the ground truth which means "what was actually said". The WER is computed using a string Edit Distance between the reference transcription and the hypothesized transcription. String edit distance can

be efficiently computed using dynamic programming. Because string edit distance can be unreliable over a long body of text, we typically accumulate the error counts on a sentence-by sentence basis and these counts are aggregated overall sentences in the test set to compute the overall WER. Where Edit Distance is to determine how dissimilar two strings (for example words) are to one another by counting the minimum number of operations required to transform one string into the another. In the example below, the hypothesis "how never a little later he had comfortable chat" is measured against the reference "however a little later we had a comfortable chat" to reveal two substitution errors, one insertion error, and one deletion error as shown in table 1.1.

It is commonly believed that a lower word error rate shows superior accuracy in recognition of speech, compared with a higher word error rate.

Table 2.1: Example of Word Error Rate

Reference	***Hypothesis***	***Error***
However	How	Substitution
-	Never	Insertion
A	A	Correct
Little	Little	Correct
Later	Later	Correct
We	He	Substitution
Had	Had	Correct
A	-	Deletion
Comfortable	Comfortable	Correct
Chat	Chat	Correct

Sentence error rate (SER) is a less commonly used evaluation metric which treats each sentence as a single sample that is either correct or incorrect. If any word in the sentence is hypothesized incorrectly, the sentence is judged incorrect. SER is computed simply as the proportion of incorrect sentences to total sentences.

2.3.1 Other Considerations

Besides accuracy, there may be computational requirements that impact performance,such as processing speed or latency. Decoding speed is usually measured with respect to a

real-time factor. In general, any ASR system can be tuned to trade-off speed for accuracy. But, there is a limit. For a given model and test set, the speed-accuracy graph has an asymptote that is impossible to cross, even with unlimited computing power. The remaining errors can be entirely ascribed to modeling errors. Once the search finds the best result according to the model, further processing will not improve the accuracy.

2.4 The Fundamental Equation

In this section, the Bayes rule and the fundamental equation of speech recognition is discussed. Speech recognition is cast as a statistical optimization problem. Specifically, for a given sequence of observations O=O1,...,ON, we seek the most likely word sequence W=W1,...,WM. That is, we are looking for the word sequence which maximizes the posterior probability P(W|O). Mathematically, this can be expressed as,

$$W = argmax_W \, P(W|O)$$

To solve this expression, we employ Bayes rule,

$$P(W|O) = \frac{P(O|W)\, P(W)}{P(O)}$$

Because the word sequence does not depend on the marginal probability of the observation , this term can be ignored. This, we can rewrite this expression as

$$W = argmax_W \, P(O|W)\, P(W)$$

This is known as the fundamental equation of speech recognition. The speech recognition problem can be cast as a search over this joint model for the best word sequence.

The equation has a component P(O|W) known as an acoustic model, that describes the distribution over acoustic observations O given the word sequence W. The acoustic model is responsible for modeling how sequences of words are converted into acoustic realizations, and then into the acoustic observations presented to the ASR system.

The equation has a component P(W) called a language model based solely on the word sequence W. The language model assigns a probability to every possible word sequence. It is trained on sequences of words that are expected to be like those the final system will encounter in everyday use. A language model trained on English text will probably assign a high value to the word sequence “I like turtles” and a low value to “Turtles

sing table." The language model steers the search towards word sequences that follow the same patterns as in the training data. Language models can also be seen in purely text-based applications, such as the auto-complete field in modern web browsers. For a variety of reasons, building a speech recognition engine is much more complicated that this simple equation implies.

2.5 HMM-GMM Acoustic Model

Now lets dig deeper into the Acoustic Model. Acoustic Modeling is the technique of taking an acoustic wave form and breaking it down into individual sounds or phones in order to interpret all the different little parts that make up a waveform. The acoustic model looks at the fine detail of the signal and provides the foundation for everything that comes after it. In order to model the phone units in the acoustics, there is a technique called Hidden Markov Model which is just a way of breaking the signal label, down into three units per phone, the beginning, middle, and end of the sound. And these labels form the target for the training algorithms. For more advanced models, instead of phonetic units, use triphone units. And the triphones are just like phones but they are aware of the phonetic context they are in, because when we are speaking fluently, the target sounds we make for a particular phone will change based on the context around it. People often refer to the labels as senones and this is an old term.

2.5.1 Markov Chains

Markov chains are a method for modeling random processes. In a Markov chains, discrete events are modeled with a number of states. The movement among states is governed by a random process.Let us consider an example. In a weather prediction application, the states could be " Sunny", " Partly Cloud", " Cloudy", and " Raining". If we wanted to consider the probability of a particular 5 day forecast, for example P(p,p,c,r,s), we would employ Bayes' rule to break up this joint probability into a series of conditional probabilities.

$$p(X1, X2, X3, X4, X5) = p(X5|X4, X3, X2, X1)\, p(X4|X3, X2, X1)\, p(X3|X2, X1)\, p(X2|X1)\, p(X1)$$

This expression can be greatly simplified if we consider the first-order Markov assumption, which states that

$$p(X_i|X_1, ..., X_i - 1) = p(X_i|X_i - 1)$$

Under this assumption, the joint probability of a 5-day forecast can be written as

$$p(X1, X2, X3, X4, X5) = p(X5|X4)\, p(X4|X3)\, p(X3|X2)\, p(X2|X1)\, p(X1)$$

$$p(X1, X2, X3, X4, X5) = p(X_1) \prod_{i=2}^{5} p(X_i|X_{i-1})$$

Thus, the key elements of a Markov chain are the state identities (weather forecast in this case) and the transition probabilities p(Xi | Xi-1) that express the probability of moving from one state to another (including back to the same state). Note that in addition to the condidional probabilities

$$p(X_i|X_{i-1})$$

in the equation above, there was also a probability associated with the first element of the sequence,

$$p(X_i)$$

So, in addition to the state inventory and the conditional transition probabilities, we also need a set of prior probabilities that indicate the probability of starting the chain in each of the states. Let us assume our prior probabilities are as follows,

$$p(p) = \pi_p\, , p(c) = \pi_c\, , p(r) = \pi_r\, , p(s) = \pi_s$$

Now, let us return to the example. We can now compute the probability of P(p,p,c,r,s) quite simply as

$$p(p, p, c, r, s) = p(s|r, c, p, p)\, p(r|c, p, p)\, p(c|p, p)\, p(p|p)\, p(p)$$

$$p(p, p, c, r, s) = p(s|r)\, p(r|c)\, p(c|p)\, p(p|p)\, p(p)$$

2.5.2 Hidden Markov Model (HMM)

The Markov chains described above are also known as observable Markov models. That is because once we land in a state, it is known what the outcome will be, for example. it will rain. A hidden Markov model is different in that each state is defined not by a deterministic event or observation but by a probability distribution over events or observations. This makes the model doubly stochastic.

The transitions between states are probabilistic and so are the observations in the states themselves. We could convert the Markov chain on weather to a hidden Markov model by replacing the states with distributions. Specifically, each state could have a different probability of seeing the various weather conditions, sun, partly cloudy, cloudy, or rainy. Thus, a HMM is characterized by a set of N states along with

- A transition matrix that defines probabilities of transitioning among states A with elements a subscript ij
- A probability distribution for each state

$$B = b_i(x)\,, i = 1, 2, ..., N$$

- A prior probability distribution over states

$$\pi = \pi_1, \pi_2, ..., \pi_N$$

Thus, we can summarize the parameters of an HMM compactly as,

$$\phi = A, B, \pi$$

There are two fundamental problems for hidden Markov models each with well-known solutions.

The Decoding Problem

Given a model and an observation sequence, what is the most likely sequence of states through the model that can explain the observations. This problem can be solved using the well-known Viterbi algorithm. The application of this algorithm to the special case of large vocabulary speech recognition is discussed later.

The Training Problem

Given a model and an observation sequence (or a set of observation sequences) how can we adjust the model parameters (phi). This problem can be efficiently solved using the BaumWelch algorithm, which includes the Forward-Backward algorithm.

A byproduct of the forward algorithm is that it computes the probability of being in a state i at time t given all observations up to and including time t. The backward algorithm has a similar structure, but computes the probability of being in state i at time t given all future observations starting at t+1. These two artifcats are combined in the forward- backward algorithm to produce the posterior probability of being in state i at time t given all of the observations.

Once we know the posterior probability for each state at each time, the Baum-Welch algorithm acts as if these were direct observations of the hidden state sequence, and updates the model parameters to improve the objective function.

2.5.3 HMM for speech recognition

In speech recognition, hidden Markov models are used to model the acoustic observations (feature vectors) at the subword level, such as phonemes. It is typically for each phoneme to be modeled with 3 states, to separately model the beginning, middle and end of the phoneme. Each state has a self-transition and a transition to the next state. Word HMMs can be formed by concatenating its constituent phoneme HMMs. Thus, a high-quality pronunciation dictionary which "spells" each word in the system by its phonemes is critically important for successful acoustic modeling. Historically, each state in the HMM had a probability distribution defined by a Gaussian Mixture Model (GMM) which is defined as

$$p(x|s) = \Sigma_m \omega_m N(x; \mu_m, \Sigma_m)$$

where

$$N(x; \mu_m, \Sigma_m)$$

is a Gaussian distribution and wm is a mixture weight, with

$$\Sigma_m \, \omega_m \; = 1$$

Thus, each state of the model has its own GMM. The Baum-Welch training algorithm estimated all the transition probabilities as well as the means, variances, and mixture weights of all GMMs.

As described above, the word HMMs can be constructed by chaining the HMMs for the individual phones in a word according to the pronunciation dictionary. These phonemes are referred to as "context-independent" phones, or CI phones for short. It turns out that the realization of a phoneme is in fact heavily dependent on the phonemes can precede and follow it. For example, the /ah/ sound in "bat" is different from the /ah/ sound in "cap".

For this reason, higher accuracy can be achieved using "context-dependent" (CD) phones. Thus, to model "bat" we'd use an HMM representing the context dependent phone /b-ah+t/ for the middle /ah/ sound and for the word "cap" we'd use a separate HMM that modeled /k-ah+p/. So, imagine the word "cup" was in the utterance "a cup of coffee". Then cup would be modeled by the following context-dependent HMMs.

- /ay-k+uh/
- /k-uh+p/
- /uh-p+uh/

When context-independent phones are used, there are a very managable number of states: N phones times P states per phone. US. English is typically represented using 40 phones, with three states per phone. This results in 120 context-independent states. As we move to context-dependent units, the number of triphones is N cube. This leads to a significant increase in the number of states, for example,

$$40^3 * 3 = 192,000$$

This explosion of the label space leads to two major problems:

1. Far less data is available to train each triphone
2. Some triphones will not be observed in training but may occur in testing

A solution to these problems is in widespread use which involves pooling data associated with multiple context-dependent states that have similar properties and combining them

into a single “tied” or “shared” HMM state. This tied state, known as a senone, is then used to compute the acoustic model scores for all of the original HMM states who’s data was pooled to create it. Grouping a set of context-dependent triphone states into a collection of senones is performed using a decision-tree clustering process. A decision tree is constructed for every state of every context-independent phone and explanation is out of the scope. The clustering process is performed as follows,

1. Merge all triphones with a common center phone from a particular state together to form the root node. For example, state 2 of all triphones of the form /*-p+*/
2. Grow the decision tree by asking a series of linguistic binary questions about the left or right context of the triphones. For example, "Is the left context phone a back vowel?" or "Is the right context phone voiced?" At each node, choose the question with results in the largest increase in likelihood of the training data.
3. Continue to grow the tree until the desired number of nodes are obtained or the likelihood increase of a further split is below a threshold.
4. The leaves of this tree define the senones for this context-dependent phone state.

This process solves both problems listed above. First, the data can now be shared among several triphone states so the parameters estimates are robust. Second, if a triphone is needed at test time that was unseen in training, it’s corresponding senone can be found by walking the decision tree and answering the splitting questions appropriately.

Almost all modern speech recognition systems that use phone-based units utilize senones as the context-dependent unit. A production-grade large vocabulary recognizer can typically have about 10,000 senones in the model. Note that this is far more than the 120 context-independent states but far less than the 192,000 states in a untied context-dependent system.

To label all frames of the training data with a corresponding senone label, a process known as forced alignment is used. In this process, we essentially perform HMM decoding but constrain search to be along all paths that will produce the correct reference transcription. Forced alignment then generates the single most-likely path, and thus, the senone label for every frame in the utterance.

The forced alignment process needs a speech recognition system to start from. This can be an initial GMM-based system.

The output of forced alignment is typically a file that lists for each utterance the start frame and end frame of a segment and the corresponding senone label. This format can be different depending on the toolkit being used. HTK is a well-known speech recognition toolkit. In HTK, the output from forced alignment is called an MLF file. Here is a snippet from an MLF file produced by forced alignment. The columns of the MLF can be interpreted as,

1. Start time (in 100ns time units)
2. End time (in 100ns time units)
3. Senone ID
4. Acoustic model score for that senone segment
5. Context-dependent triphone HMM model (appears at start of phone boundary)
6. Acoustic model score for the triphone HMM model
7. Word in the transcription (appears at start of word boundary)

2.6 Language Modelling

A language model is a component of a speech recognizer that knows how likely or unlikely where hypotheses are. For example the language model knows that the "dog bit the man" is much more likely than the "man bit the dog." Or even "man the bit dog the", which is not even English. It seems intuitive to us as English speakers but for a computer to parse these things out it is a little bit more complex. Most current recognizers have actually a finite list of words that they can recognize and it is important to carefully choose the vocabulary of the system, we learn how to do that based on training data so as to minimize the occurrence of so-called out of vocabulary words. N-gram modeling which is really the basis for most of the language models you will find today. The idea is simply to count the number of times that two or more words occur in an immediate sequence and we will learn how to use that to estimate the probability of an entire sentence or utterance.

Language Modeling is the component of a speech recognition that estimates the prior probabilities P(W) of possible spoken utterances. Recall that these prior probabilities are combined with the acoustic model likelihoods P(O|W) in the Fundamental Equation of Speech Recognition to arrive at the overall best hypothesis

$$W = argmax_W P(O|W)P(W)$$

Thus, the language model (or LM) embodies the recognizer's knowledge of what probable word sequences are, even before it has heard any actual speech sounds. Instead of encoding hard rules of syntax and semantics that allow some utterances and disallow others, the LM should assign high probabilities to likely utterances and low probabilities to unlikely ones, without ruling anything out completely (because one never knows what people might actually say). In language modeling we often talk about sentences as the word sequence corresponding to an entire speech utterance, without suggesting that these represent anything like a correct and complete sentence in the conventional grammatical sentences. In fact, a sentence for LM purposes can be anything a speaker would utter in the context of a speech application.

2.6.1 Vocabulary

We need to assign a probability to every possible sentence

$$W = w_1 w_2 ... w_n$$

where n is the number of words, which is unbounded in principle. First, we simplify the problem by limiting the choice of words to a finite set, the vocabulary of the LM. Note the vocabulary of the LM is also the vocabulary of the speech recognizer – we cannot recognize a word that is not considered possible by the LM (that is, it is probability would be effectively zero). Words outside the vocabulary are called out-of-vocabulary words, or OOVs. If we ever encounter an OOV in the input data we will incur (at least) one word recognition error, so it is important to choose the vocabulary so as to minimize the chances of OOVs. An obvious strategy is to pick the words that have the highest prior probability of occurring, as estimated from data. In other words, we choose the most frequently occurring words in a corpus of training data. For example, we can pick the N most frequent words, or all words occurring more than K times in the data, for suitable values of N or K There is often an optimal vocabulary size that represents a

good tradeoff between recognizer speed (a larger vocabulary means more computation in decoding) and accuracy (by reducing OOVs, but adding very rare words will have negligible effect on accuracy, and might even hurt accuracy, due to search errors and greater acoustic confusability within the vocabulary).

2.6.2 N-gram

Even with a finite vocabulary, we still have an infinite set of word sequences, so clearly we cannot parameterize the LM by listing the probability of every possible sentence. Even if we conceptually could do so, it would be impossible to get reliable probability estimates as the majority of possible sentences are very rare (the smaller a probability the more data is needed to estimate it reliably). We can work around both these problems by using a trick we already discussed in the acoustic modeling module: use the chain rule to factor the sentence probability into a product of conditional word probabilities, and then apply the Markov assumption to limit the number of states, and thereby, parameters.

$$P(W) = P(w_1)xP(w_2|w_1)xP(w_3|w_1w_2)x...xP(w_n|w_1...w_n - 1)$$

Now let us assume that the Markov state of the LM (often called the context or history) is limited to just one word. This gives us,

$$P(W) = P(w_1)xP(w_2|w_1)xP(w_3|w_2)x...xP(w_n|w_n - 1)$$

Note how each word is now predicted by only the immediately preceding one, that is, we are using a first-order Markov model. However, in language modeling this terminology is not usually used, and instead we call such a model a bigram model, because it uses only statistics of two adjacent words at a time. Correspondingly, a second-order Markov model would be called a trigram model, and predict each word based on the preceding two, and so forth.

The generalization of this scheme is the N-gram model, that is, each word is conditioned on the previous N – 1 words. The parameters of such a model as associated with N-grams, taht is, strings N words. It turns out that there is good improvement when going from bigrams to trigrams, but little improvement as N is increased further. Therefore, in practice we rarely use LMs beyond 4-grams and 5-grams.

To let our N-gram model assign probabilities to all possible finite word sequences we are left with one small problem: how will the model predict where to end the sentence? We

could devise a separate model component for the sentence length n, but it is far easier to introduce a special end-of-sentence tag </s> into the vocabulary that marks the end of a sentence. In other words, the LM generates words left to right, and stops as soon as </s> is drawn according to the conditional probability distribution. Importantly, this also ensures that the (infinite) sum of all sentences probabilities is equal to one, as it should be for a probability distribution.Similarly, we also introduce a start-of-sentence tag <s>. It is inserted before the first word w1 , and in fact represents the context for the first real word. This is important because we want the first word to be predicted with knowledge that it is occurring first thing in the sentence. Certain words such as "I" and "well" are especially frequent in first position, and using the start-of-sentence tag we can represent this using the bigram probabilities P(w1| <s>). The complete sentence probability according to the bigram model is now

$$P(W) = P(w_1| < s >)xP(w_2|w_1)xP(w_n|w_n - 1)xP(< s > |w_n)$$

Now turn to the problem of actually estimating these N-gram probabilities.

The conditional probabilities based on N-grams can be naively estimated by their relative frequencies. Let

$$c(w_1...w_k)$$

be the number of occurrences (or count) of the k-gram. For example, the conditional probability of "bites" following "dog" is the ratio

$$P(bites|dog) = \frac{c(dogbites)}{c(dog)}$$

More generally, k-gram probability estimates are

$$P(w_k|w_1...w_k - 1) = \frac{c(w_1...w_k)}{c(w_1...w_k - 1)}$$

N-gram smoothing

Relative frequencies as estimates for probabilities have one severe problem: they give probability zero to any N-gram that is not observed in the training data (the numerator in the equation becomes zero). Training data is finite, and we should not rule out a combination of words simply because our limited language sample did not contain it. (Another reason is that language is not a static system, and speakers come up with

new expression and even words all the time, either because they are being creative or because language production is error-prone.)

So we need a principled way to assign nonzero probability estimates to N-grams that we never seen, a process that is often called language model smoothing (we can think of the unobserved N-grams as "holes" in the model, that have to smoothed over). An entire sub- specialty of LM research has looked at this problem and many methods have been proposed, several of which are implemented by the SRILM tools.

2.7 Speech Decoding

Decoding is the process where we take our language model and combine it with the acoustic model that we've trained in order to create a best guess for the transcript of whatwas said. It's a dynamic programming search that, given all of this information that we have trained about the nature of speech and given acoustic sentence, it comes up with its best guess of what was said.All of the structures we've seen above like the phonetic model, the acoustic model, the language model, and just bringing it all together in the decoder. Older designs for speech recognition decoders took the different parts of the structure of a language and built it directly into the algorithms in the code. More modern designs encode all of the structures of the language into graphs that are then consumed by the decoder and the decoder becomes very simple because the structure of the language is in the data rather than in the algorithm. And the mathematical formalism we use is known as a weighted finite state transducer (WFST) technology.

The acoustic model scores sequences of acoustic model labels. For every time frame of input data, it computes the relative score for every label. Every sequence of labels has an associated score.

The language model scores sequences of words. To every valid sequence of words in the vocabulary, it assigns a language model score.

The glue that ties the two together is the decoding graph. It is a function that maps valid acoustic label sequences to the corresponding word sequences, together with the language model score.

2.7.1 The Search

Speech recognition decoding is the process of finding the word sequence that jointly maximizes the language model score and acoustic model score. A sequence of acoustic states is assigned an acoustic model score by the acoustic model, and a language model score by path it describes through the decoding graph.

It is a path search algorithm through the decoding graph, where the score of the path is the sum of the score given to it by the decoding graph, and the score given to it by the acoustic model. Due to the nature of our models, we can use a simple dynamic programming approach to find the shortest path. If the best path passes through state S at time T, then it includes the best prefix path ending at time T and state S.

So this is how ASR works and all the components of a speech recognition system, how to train them from data, and how to bring these different models together in a decoder in order to take audio we have on disk in the form of waveform and generate the words that were spoken by the user.

Chapter 3

Related Work

In this chapter, we discuss the research work carried out to build Automatic Speech Recognition engines for hearing impaired.

3.1 History of ASR

There are so many Automatic speech recognition systems developed for normal hearing person like Siri, Cortana and Amazon echo and so on, but there is not much work done for hearing impaired person.

3.1.1 SHOEBOX

The very first kind of ASR is the SHOEBOX, which was in 1962, and it was by IBM [34]. And they actually demoed the system. But what it was recognizing was just connected strings of digits. So it was just purely a digit recognizer and a few mathematics operations so, it could do basic mathematics like six plus five and so on. But it was very limited and just total of 16 words that is 10 digits and six operations and it done only isolated word recognition.

3.1.2 HARPY

In the 70's, there was a lot of interest in developing speech recognition systems and Artificial Intelligence (AI) based systems. ARPA which is a big agency in the US funded this dollar 3 million project in 1975, and three teams worked on this particular

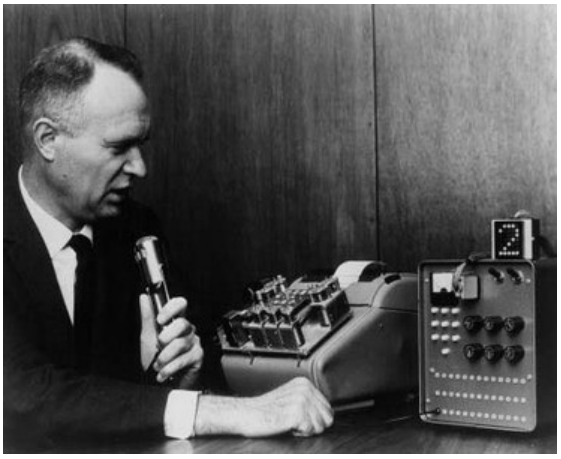

Figure 3.1: IBM SHOEBOX

project. The goal was to build a fairly advanced speech recognition system, which was not just doing some isolated word recognition and would actually evaluate continuous speech [35]. The winning system from this particular project was HARPY out of CMU. HARPY actually was recognized connected speech from 1,000 word vocabulary. It still did not use statistical models, which is kind of current setting. This was in 1980s, pioneered by Fred Yelenik at IBM and others around the same time.

3.1.3 LVCSR Systems

Statistical models became very popular to be used in speech recognition. The entire problem was formulated as a noisy channel. One of the main machine learning paradigm, which was used for this particular problem were hidden Markov models [36]. Statistical models were able to generalize much better than the previous models because the previous models are all kind of rule- based. Then move into 10K vocabulary sizes. The vocabulary size got larger, and these are now falling in what are known as large vocabulary continuing speech recognition systems. Now, large vocabulary is much larger than 10K, but that was in the 80's. In 2006, deep neural networks came to the forefront, now all the state-of-the-art systems are powered by deep neural networks. Any of these systems Cortana, Siri, Voice Search, at the back end they are powered by deep neural network based models.

3.2 ASR systems for Hearing Impaired

There is an application build for hearing impaired developed by students of Fatima Jinnah University of Pakistan [7]. The desktop application is designed for the children who are hearing impaired. The system is specially designed for the feasibility of deaf so that it will be easy to use interfaces for them. It caters two languages that is English and Urdu. The data was collected from the School for Special Children and Pakistan Sign Language Organization. The participants included was 8 hearing impaired boys and 12 girls. The application starts with the two options of learning that is, learning through articulation and other was learning through sign language. The first learning category was learned by watching the lip syncing videos and the other learning category was learned through sign language videos. Each learning category is further divided into four main categories with quiz for each of these four categories that is, Alphabets, Broken Words, Words, and Sentences. The results showed that Profound Hearing Loss (PHL) children were quick to give answers of sign language activities whereas Mild Hearing Loss (MHL) children were quick in speech learning activities.

Jeyalakshmi et al. [37] presents a speech recognition system by using the speech data of deaf as an input to the system. The system built for the recognition of isolated words. Total words were five that is, one, two, three, four and five. Total participants were also five deaf individuals of age five to ten years old. Perceptual linear predicton coefcents (PLP) were used as the acoustic feature. Training phase included unsupervised and supervides learning of Self organized feature map neural network (SOFM). SOFM output then given to the Back propagaton neural network (BPN) for learning. The word one, two three, four and fives acquires recognition accuracy of 50%, 10%, 60%, 60% and 40% respectively.

Yousaf et al. [38] proposed a mobile application called vocalizer to mute (V2M) which uses Automatic Speech Recognition (ASR) for the speech recognition of deaf. Total deaf participants were 15 children seven to thirteen years old. For speech samples, questions were asked by the researchers from the deaf and in response, deaf answer the question which was used as the testing sample. Twelve short length questions were selected. The recognition accuracy was measured in terms precision and recall which is not a good measure for speech recognition systems. HTK speech recognizer was used for recognition phase. The accuracy achieved for the proposed application was 97.9%.

Resmi K et al. [39] presents a Speech Training system for the development of speech and language skills of deaf children. A Graphical User Interface (GUI) is designed for deaf to learn for learning of words by seeing the visual feedback. Three categories of deaf children is considered in this system that is, slight,mild and moderate hearing loss. The system consists of two modes; Graphic mode and Training mode.In Graphics mode, the interface consist of object images that is to be learned, audio of word and sign description. To practice, there is a Training mode which records the audio of deaf children and then this audio is compared to speech recognition database. The administrator manages the application that can be teacher or parent. The administrator first selects the learning modes that is, Training mode or Graphics. To view the multimedia interface, graphics mode is selected and for recording of the speech , the training mode is selected then the control logic block decides the results that should be given to the child as a form of visual feedback. This application will help hard of hearing children to easily communicate with the normal hearing person.

To help the hearing impaired in speech learning process, Riella et al. [40] presented a voice analyzer called VOXsis. The speech was processed in real time for the spontaneous assessment of speech. The main objective of this work is to calculate the overall percentage of correctly identified specific word. Digital Signal Processor (DSP) were used for the analysis of the speech. S System is used to teach the correct pronunciation of words to the hearing impaired and gives the visual feedback. For the evaluation of the correct word pronunciation, three formants are used that is, resonance frequencies of nasal, larynx and buccal. Three digital filter banks were used for the acquisition of formants from speech. Then temporal normalization is performed which were used for extraction of parameters and comparison of the results. For the multiple frequencies, 95% correct recognition was achieved, whereas 65% correct recognition was achieved for only the speech of men.

Other Automatic Speech Recognition (ASR) applications that is specially build for Hard of Hearing or Deaf are Text Hear, which is used for real time captioning. Roger Voice which helps deaf in phone conversations by using Speech Recognizer and taking input of Normal Hearing person's speech and display the text to the deaf. Hearing Helper application only available for Apple devices by using the Apple's Siri technology. Ava is another application used for real time captioning for Hard of Hearing. Speak See is another application which helps Hard of Hearing individuals in meetings by using

Speech-toText recognition system.

3.3 Critical Analysis

There are number of Automatic Speech Recognition (ASR) systems that are developed for deaf and hard of hearing person but they take input as the voice of Normal Hearing (NH) person and output is the text or transcription of the speech of Normal Hearing (NH) so that Hearing Impaired (HI) person can understand what information the other person is trying to convey.

According to Don Grushkin, this is a one-way process which means that, a Normal Hearing (NH) can convey his information to Hard of Hearing through speech but what about the reverse process. If Hard of Hearing wants to reply. For this, Hard of Hearing person have to type back his replies.

And the other way around, there are Speech Synthesis systems (Text-to-Speech Recognition system) which take input as the text which is written by deaf or Hard of Hearing person and conveyed their information to Normal Hearing (NH) as the output in the form of speech.

We can build Automatic Speech Recognition (Speech-to-Text) systems for the speech of Hard of Hearing. There are very few systems build which work on the speech of Hearing Impaired (HI). We still need a lots and lots of speech data of Hard of Hearing to train a good Recognition engine. There is a lot of work still needs to be done on this research area.

Chapter 4

Proposed Methodology

This chapter illustrate the methodology of the proposed work. The objective of this dissertation was to build an Automatic Speech Recognition (ASR) system for hearing impaired children. This dissertation focuses on applying the Artificial Intelligence (AI) techniques on speech data of hearing impaired. To check the robustness of the designed system, Rauf et al. [7] approach is used to validate the results.

Division of this chapter starts with the description of proposed framework, the environment and tools used in this dissertation, detailed description of data set being used, data pre-processing techniques, acoustic data preparation and language data preparation, the formatting style of data, training models are then described and in the end, some modifications to the data were mention.

4.1 Overview of Methodology

The system is specially build for profound hearing impaired children. The speech of deaf is different than normal hearing person. The profound hearing impaired cannot listen to sound at all, have cumbersome and complicated speech which is very difficult to understand for normal hearing person. Listening of the audio of these deaf children for the utterance transcription were tough and challenging. Writing the transcriptions of almost four hundred files were truly a challenging task, which was time consuming.

Data that collected was in *mpeg* format. The kaldi supports input of *WAV* files. The script was written using *SOX* audio processing tool that converts the *mpeg* format into *WAV* format. The data was converted with special characteristics of an audio.

Data consist of twelve speakers which further consist of three male speakers and nine female speakers. The audio speech consist of children having age of five to eight years. Pre-processing on the audio was accomplished due to surrounding noisy environment and quality of microphone. For the pre-processing, *Audacity* tool is used. After pre-processing, formatting is done, the data is presented in the predefined format of Kaldi toolkit. As a basis, *Voxforge* setup is used in Kaldi. As the naming convention of *Voxforge* example script was used, renaming of five hundred and twenty three audio files were revised manually which was time consuming. *Speaker ID* and *Utterance ID* of five hundred and twenty three audio files were manually created.

The next step was to prepare the data. Transcriptions were prepared for Acoustic Modeling by listening to the speech of hearing impaired several times as the speech of the hearing impaired specially profound hearing impaired is not very clear and disrupted. Each audio file took several minutes for writing accurate utterance transcriptions.

Then dictionary, lexicon and phonemes were prepared for Language Modeling. Four different types of feature transformations were trained and tested on mono-phone and tri-phone Acoustic models. As the audio contains voices of both normal hearing person and profound hearing impaired, the next step was to extract speech of only hearing impaired which was done first by the tool LIUM Diarization and a python script but the results were not impressive on the speech of hearing impaired. The performance metric for Automatic Speech Recognition (ASR) was used that is Word Error Rate (WER). Word Error Rate (WER) was calculated for each speaker separately and data was analyzed after evaluating the performance of each profound hearing impaired. The Word Error Rate (WER) were also computed separately for the deaf voice.

A statistical speech recognizer evaluates and combines both models through generating and scoring a large number of alternative word sequences (so-called hypotheses) during a complex search process.

The proposed design of Speech Recognition of Hearing Impaired Children is shown in figure 3.1.

4.2 Data Pre-processing

The need for data pre-processing was needed because:

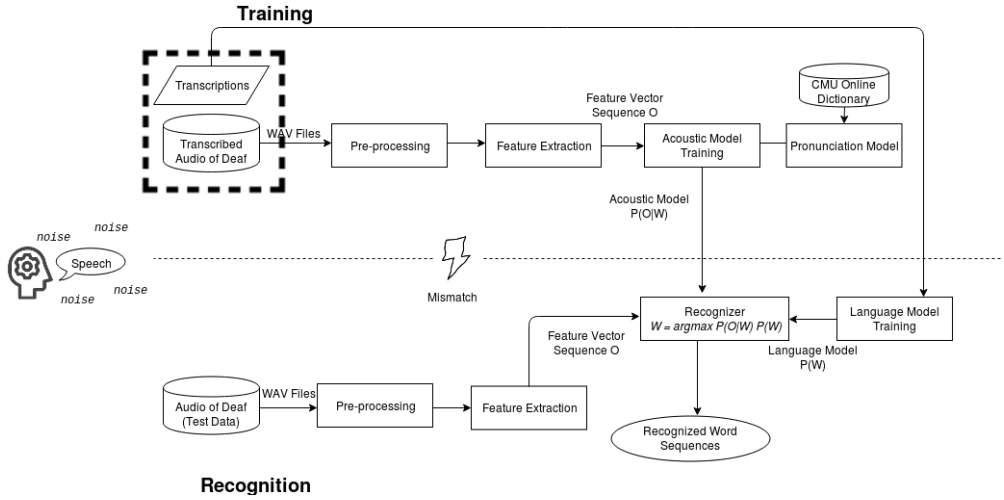

Figure 4.1: Proposed Design of Speech Recognition of Hearing Impaired Children

1. The data that was collected and recorded by the mobile microphone, which was not of good quality and that was the main barrier encountered for building good speech recognition system.

2. Another disadvantage was that that the microphone were placed far distant from the speaker and in the result the voice of the speaker was low and the speech recognition system was unable to capture the voice.

The collected video files were first converted into *WAV* file format.

Listing 4.1: Conversion of *mpeg* to *wav* format

```
#!/bin/bash
mkdir "WAV"
for f in *.MOV;
do
        fname=$f;
        echo $fname;
        base=$(echo $fname | cut -f 1 -d '.');
        echo $base;
        ffmpeg -i $fname $base.mp3;
        echo "Creating WAV files";
        sox $base.mp3 -r 16000 -c 1 -b 16 WAV/$base.wav;
end
```

This piece of code (listing 4.1) written in Bash Scripting, converts *mpeg* files (video files) first into *mp3* file (audio file) and then into *WAV* file as the input to the speech recognition system is the *WAV* files and storing them into the directory called *WAV*. The *Sox* command is used for the conversion of audio files into different format as the *Sox* is an audio processing tool for command-line.

The *Sox* command options that is, "-r" represents that the input audio file is a "raw" file that is, it has no header or no explanation of characteristics of signal. The next options show the "sample rate", "sample size", "data encoding" and "number of channels".

The sample rate of 16,000 (16kHz samples per second) is used, 16-bit sample size is used and the number of channels is one that is "mono". The audio data goes to the "/home-/sidrah/kaldi/egs/voxforge/s5/voxforge/selected" directory. In this directory there are total of 23 folders in which 11 folders are for test data and 12 are for training data based on 70%-30% data division.

For the above mentioned reasons, audacity tool is used for amplification and noise reduction of audio WAV files which also results in improvement of accuracy of speech recognition system. Audacity is the free audio editor software distributed under the GNU General Public License (GPL). For the low voice problem, "Amplify" effect was applied to the specific portions of the audio as shown in figure 3.1.

The data were presented in the predefined format of Kaldi toolkit. As a basis, *Voxforge* setup was used in Kaldi. As the naming convention of *Voxforge* example script was used, renaming of five hundred and twenty three audio files were revised manually which was time consuming. Then *Speaker ID* and *Utterance ID* of five hundred and twenty three audio files were manually created and again it was burdensome. Detailed characteristics of the speaker is mentioned in README file.

4.3 Feature Extraction

As speech sound is analog and computers are digital so, there is need to digitize them. Speech sound waves propagate through the air and are captured by a microphone which converts the pressure wave into electrical activity which can be captured [29]. The electrical activity is sampled to create a sequence of waveform samples that describe the signal. Speech signals have less high frequency (only up to 8000 Hz) information

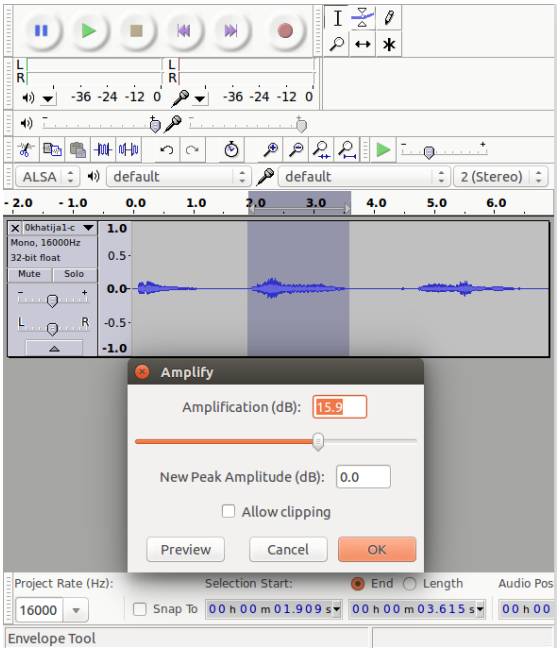

Figure 4.2: Amplification of the audio WAV files

so a sampling rate of 16,000 Hz is typically used. Speech over conventional telephone lines and most mobile phones is band-limited to about 3400 Hz, so a sampling rate of 8000 Hz is typically used for telephone speech. There is an acoustic analysis, which sees the speech waveform and converts it into some discrete representation or features. The features are then feed in to what is known as acoustic model. The very first component looked at the raw speech waveform and convert it into some representation, called Mel Frequency Cepstral Coefficient (MFCC) features.

The raw speech signal, which then is discretized, because raw speech signals are not useful for this work. First sampling is done and generate the discrete samples. Each of these samples are typically of the order of 10 to 25 milliseconds of speech. The idea is that once speech frames are extracted, which are round 25 sometimes even larger but typically 25 milliseconds. Then extract acoustic features, which are representative of all the information in the signal. Now each speech frame is represented as some real valued vector of features, which captures all the information. After getting features for each frame, these are the inputs to the next component in the ASR system, which is known as the acoustic model [30]. The process of feature extraction is,

- Speech is analyzed over short analysis window

- For each short analysis window, a spectrum is obtained using Fast Fourier Transfrom (FFT)

- Spectrum is passed through Mel-Filters to obtain Mel-Spectrum

- Cepstral analysis is performed on Mel-Spectrum to obtain Mel-Frequency Cepstral Coefficients

Thus speech is represented as a sequence of Cepstral vectors [31]. Another reason why to discretized at particular sampling rates is that the assumption is that within each of these speech frames the speech signal is stationary. If speech signal is not stationary within the speech frames, then features are not extracted from that particular slice. So, around 25 milliseconds of speech, features are extracted, and this feature extraction is a very involved process, it requires a long of signal processing know how which is out of scope. This is also kind of motivated by how ears work, one of the most common acoustic feature of presentations, which are known as Mel-frequency cepstral coefficients or MFCCs. They are motivated by what goes on in our ear or the filter banks, which apply in ear.

4.4 Acoustic Model Training

As mentioned above there are acoustic features which were extracted from raw speech signal. The output on this acoustic model to be what is a likely phone sequence, which corresponded to the particular speech utterance. Now we check how to chunk, how many frames are going to correspond to a particular phone. HMMs and states and the probability distributions govern how the speech vectors correspond to a particular states. These probability distributions are mixtures of Gaussian. The idea is that the acoustic features are mapped to phone sequences. This is all probabilistic. This is the only one best sequence but this is the likely probability distribution of phone sequences. There are observation probabilities and observation probabilities are Gaussian mixture models. These probabilities are learned from preparing the acoustic data.

4.4.1 Acoustic Model

As the audio data is prepared, there is a need to prepare transcripts that is the translation of speech spoken by the speaker in an audio file. Two files are stored for transcriptions called *PROMPTS* and *prompts-original* and a *README* file in each speaker folder to describe the characteristics of the speaker. All the *Acoustic Data* is divided into two sub-folders in *data* directory that is test and train. In each of these two sub-folders, there are four files prepared for five hundred and twenty three acoustic audio files that is created manually for training and testing data separately which is cumbersome and time taking process. All these four files with their pattern and description are shown in table.

Table 4.1: Acoustic Data Files

File Name	***Pattern of File***
spk2gender	speakerID, gender
wav.scp	uterranceID,full_path_to_audio_file
text	uterranceID, text_transcription
utt2spk	uterranceID, speakerID

The speakers are both male and female and the file *spk2gender* tells about the gender information of the speaker. Data contained 3 male speakers and 9 female speakers whereas *speakerID* represents the speaker's particular name. The *wav.scp* file represents the path of the audio file related to that specific utterance whereas the utterance is the transcription of the speech spoken by the specific speaker. The *utteranceID* is nothing more than the *speakerID* joined with the name of the *wav* file. The *text* file contains the utterance transcriptions of the audio files. The *utt2spk* file shows that the particular utterance associated with that particular speaker. There is another final *Acoustic* file called *corpus.txt* which contains all the transcriptions of the train and test data located in the */data/local* directory.

4.5 Pronunciation Model

The pronunciation model provides link between sub-word units (phonemes) and the word. There are large pronunciation dictionaries. Simple dictionary of pronunciations is maintained. This is the only module in an ASR system that is not learned. It is not learned from data. The acoustic model was learned from training data, and then language model also learned from data. But the pronunciation model is actually expert derived. An expert give these mappings. Output of the pronunciation model are words.

4.6 Language Model Training

Language model looks at a word sequence, from the speech utterance. As mentioned in above section that the pronunciation model provides link between sub-word units (phonemes) and the word. From the phones to words. As mentioned above that there are large pronunciation dictionaries. Simple dictionary of pronunciations is maintained. And this is the only module in an ASR system that is not learned. It is not learned from data. The acoustic model was learned from training data, language model also learned from data. But the pronunciation model is actually expert derived. An expert give these mappings. Mapped a phone sequence to a particular word

As in the pronunciation model, the output was words. Mapped a phone sequence to a particular word, and now the language model comes and says how should these words be ordered, according to a particular language. The language model looks at lots and lots of texts in that particular language. The statistical language models tells how likely is it for different words to occur given the recent word context. There is very popular language model called NGram language models. Look at co-occuring, either two words, or three words, or four words. If our n is two, the model is bigrams. If n is three, the model is trigrams, four-grams and so on.

4.6.1 Language Model

To define the language of the specific audio data, there is a need to prepare the dictionary. Dictionary includes the phonemes, non-silence phonemes and lexicon. CMU dictionary is normally used in Kaldi for English pronunciation, as the speech of deaf is different so

there is a need to make our own dictionary. The *kaldi/egs/voxforge/s5/data/local/dict* directory contains the dictionary. The following table defines the four files of language data.

Table 4.2: Language Data Files

File Name	***Pattern of File***	***Pattern Example***
lexicon.txt	<word> <phone 1> <phone 2> ...	BLUE B L UW, BNAANA B N A N A
nonsilence_phones.txt	<phone>	AB, AE, AH
silence_phones.txt	<phone>	sil, spn
optional_silence.txt	<phone>	sil

All the unique words pronounced by the speaker are presented in *lexicon.txt*. Each word is associated with phoneme. There can be two or more phonemes for each word. British accent have its own set of phonemes and American accent have its own set of phonemes. Phonemes are the way of how the word is pronounced whcih is described in *Background* chapter. Whereas the *nonsilence_phones.txt* represents the unique phonemes in *lexicon.txt* file. This is how language data is prepared. The pronounced words of deaf speech is different and according to words of nine hundred and ninety one (991), phonemes were prepared manually according to American accent as shown in table.

Table 4.3: Words and their Phonemes

Word	***Phoneme***	***Word***	***Phoneme***	***Word***	***Phoneme***
A	AH	AAGUB	A G UB	ABPAH	A B P A H
A	EY	AAGUU	A G OU	AEYBIL	AE B IY L
AAB	A B	AAHULLO	A H UL OU	ALIPHAANT	A L IY PH A N T
AABAL	A B UL	AAIE	A IE	ALLAUP	AL L A UP
AAEAH	A E AH	AALBOW	A L B OW	ALVO	AL V OU
AAEB	A EB	AALIBOW	A L IY B OW	AMMAIN	A M EY N
AAEE	A EY	AALLOH	A L OU H	AMMEH	A M EH
AAEYB	A AE EY UB	AAROW	A R OU	ANNIE	AE N IY

4.7 Recognizer

The different parts of the ASR system which are giving various estimates of what is the most likely phoneme sequence. What is the most likely word sequence and so on. But finally the most clear word sequence corresponding to speech utterance, and then it is a search problem.There are various components, and now there is a need to search, putting all of them together, to search through this entire space. A vocabulary size of around 40,000 gives us a search graphs of the order of tens of millions of states. So these are really large graphs, and so now there is a need to search through these graphs and throw out what is the most likely word sequence to correspond to the speech utterance. So that is the *Recognizer*. This is the entire kind of pipeline of how an ASR system works. The Acoustic model is prepared from Acoustic data and Language model is prepared from Language data mentioned above. After the preparation of both models, next come the training models. Four training models are selected that are,

1. Monophone Training
2. Triphone1 Training
3. Triphone2a Training
4. Triphone2b Training

In all of the training models, the feature transformations were different. Delta features are used, delta plus delta-delta features and Linear Discriminant Analysis (LDA) transformation of features and Maximum Likelihood Linear Transform (MLLT) estimation. Triphone1 model uses delta features for training, Triphone2a uses delta plus delta-delta features and Triphone2b uses Linear Discriminant Analysis (LDA) and Maximum Likelihood Linear Transform (MLLT) estimation. In decoding, Language model was combined with the acoustic model that was trained in order to create a best guess for the transcript of what was said. It is a dynamic programming search that, given all of this information that was trained about the nature of speech and given acoustic sentence, it comes up with its best guess of what was said. All of the structures was seen above like the acoustic model, the language model, and bringing it all together in the *Recognizer*. The acoustic model scores sequences of acoustic model labels. For every time frame of input data, it computes the relative score for every label. Every sequence

of labels has an associated score. The language model scores sequences of words. To every valid sequence of words in the vocabulary, it assigns a language model score. The glue that ties the two together is the decoding graph. It is a function that maps valid acoustic label sequences to the corresponding word sequences, together with the language model score. So, Speech recognition decoding is the process of finding the word sequence that jointly maximizes the language model score and acoustic model score. It is a path search algorithm through the decoding graph, where the score of the path is the sum of the score given to it by the decoding graph, and the score given to it by the acoustic model. Due to the nature of the models, simple dynamic programming approach can be used to find the shortest path. If the best path passes through state S at time T, then it includes the best prefix path ending at time T and state S. So this is how the speech recognition of hearing impaired works and all the components of a speech recognition system, how to train them from data, and how to bring these different models together in a *Recognizer* in order to take audio that is on disk in the form of waveform and generate the words that were spoken by the user.

4.8 Speaker Diarization

The data that were collected from [41] contained the speech of both normal hearing person and hearing impaired children. The goal of this research work was to recognize the speech of the hearing impaired children and the recognition of the speech of normal hearing person. This was the main reason behind speaker diarization. Based on speaker identity, segmentation of speech is done. The output contains the time segments in which speaker A or speaker B speaks.

The speakers in audio does not spoke in alternate turns which was the major disadvantage faced by speaker diarization process. To best, only those audio in which normal hearing person and hearing impaired children spoke in alternate turns. *LIUM_SpkDiarization* software was used. The main drawback was that, that the audio was recorded by mobile phone. *LIUM_SpkDiarization* does not give good results for mobile recorded audios which was mentioned on their official website. The diarzation was conducted on two types of data

1. Data before pre-processing

2. Data after pre-processing

There are two main directories in */LIUM-master* which were useful for diarzation. The one is */test_wav*, which stores *wav* files on which the diarization was to be processed. The other directory *test_out* stores the output segments of each *wav* file separately. The output is stored in a file with extension *ev_is.120.seg* called the diarization file. Diarization file is created for each *wav* file seperately. This file mentions two things that is the number of clusters and characteristics of each segment with respect to each cluster. The number of clusters represents the number of speakers. For each cluster there are multiple segments as the speaker xyz can speak multiple times that is why each cluster can have multiple segments. The number of lines in diarization file represents the number of segments. The segment format is

1. Field 1: The name of audio file
2. Field 2: Represents the number of channels that is mono or stereo
3. Field 3: The start time of the segment in milliseconds
4. Field 4: The duration of the segment in milliseconds
5. Field 5: Represents gender of the speaker (U=Unknown, F=Female, M=Male)
6. Field 6: Shows which type of band is used (T=Telephone, S=Studio)
7. Field 7: Shows the type of environment (Music, Speech only, Broadcast news)
8. Field 8: Represents the label of the speaker

The diarization file is generated by executing the file *go.sh* on *Bash*. The code is written in *Bash Scripting*. It execute the file *ilp_diarization2.sh* and outputs the segments and cluster of each audio file in *.seg* diarization file in output folder called *test_out*.

Listing 4.2: Processing Diarization file

```
1 inputfolder=./test_wav
2 outputfolder=./test_out
3
4 mkdir -p $outputfolder
```

```
5
6  for file in $inputfolder/*.wav
7  do
8   echo "Processing $file"
9   ./ilp_diarization2.sh $file 120 $outputfolder
10 end
```

4.9 Evaluation

Rauf et al. [7] presented a desktop application for the children who are hearing impaired in which speech evaluation of children were carried out by using the *Google Cloud Speech-to-Text* API. There approach is followed to check the robustness of our system.

4.9.1 System Setup

Google Cloud Speech-to-Text API uses neural network models for the conversion of speech into text.

1. Step 1: To avail the services of *Google Cloud Speech-to-Text* API, an account was created and debit card information was provided.

2. Step 2: Open the *Quick Start* window and select from the three options to send the speech and get the transcription. *Command Line* approach was selected.

3. Step 3: To Enable Cloud Speech API, *GCP Console* project was set up. Enter the name of the project which was named *Speech-of-deaf*. Next was the selection of the permissions of service account which was "Project Viewer". Then downloaded a private key as JSON which was stored named as *Speech of deaf-8a0fa9d8d8d3.json*.

4. Step 4: Next step was to set up the environment variables called

   GOOGLE_APPLICATION_CREDENTIALS

   by opening the terminal from the directory which contains the JSON file contain the private key and set the command.

   export GOOGLE_APPLICATION_CREDENTIALS="[PATH]"

5. Step 5: Next step was to install and initialize the *Cloud SDK*. Selected the *DEBIAN/UBUNTU* as our system. Then *python* version was checked as *Cloud SDK* supports Python 2.7.9 or later versions.

6. Step 6: For *Cloud SDK*, the environment variable was set by the command

   export CLOUD_SDK_REPO="cloud-sdk-$(lsb_release -c -s)"

   Then Cloud SDK distribution URI was added by the command

   echo "deb http://packages.cloud.google.com/apt $CLOUD_SDK_REPO main" | sudo tee -a /etc/apt/sources.list.d/google-cloud-sdk.list

   Then imported the public key of Google Cloud by using the *curl* command. To install the SDK, *sudo apt-get update sudo apt-get install google-cloud-sdk* command was entered on terminal. Installation was ended by executing the command *gcloud init*.

4.9.2 Transcription Request

The system was ready to send a transcription request to *Google Cloud* API. Selected the option for short audio recognition. Short audio time was less than one minute. To perform the recognition on our own data, two steps had to be performed. Set the environment variable again. To initiate the new session, environment variable had to be set again. To send the transcription request.

gcloud ml speech recognize PATH-TO-LOCAL-FILE –language-code='en-US'

were executed separately for each audio file which was a time taking process. Different options were used for this command. *–hints[]* command was used to give the idea of a spoken word in the speech. to mention the time stamps of each spoken word, option *–include-word-time-offsets* was also used. The results were shown on the *Bash* terminal separately for each audio. Two types of data (before pre-processing and after pre-processing) were used. The confidence value was also shown with each word to represents that if the confidence value was low then there were chances that another word was more accurate.

Chapter 5

Results and Discussion

This chapter presents results of the proposed work.To build and experiment with speech recognition systems it is obviously very important to measure performance. To measure the accuracy of Automatic Speech Recognition systems, Word Error Rate (WER) was used. Accuracy is a reciprocal of Word Error Rate (WER). To check the robustness of the system, results are compared with the approach used in [7]. diarization results are also shown in the end of the chapter which was not impressive for our data.

5.1 Experimental Setup

To build the speech recognition system, Kaldi toolkit was used. First Ubuntu 14.04 LTS was installed, as the Kaldi toolkit best works on Linux rather than on Windows. HP-Pavilion Notebook PC was used with 4GB RAM and intel core i3. After the successful installation of Ubuntu 14.04, Kaldi was acquired from git repositories. Kaldi needs C compiler (GCC), version 7.3.0 were compatible with Kaldi and G++ version 4.8.5 because version greater than 4.7 were compatible. ATLAS library was needed, GNU Awk, Flac decompressor, SRILM tool, OpenFST, Sequitur, SWIG, and Numpy. The installation of these libraries took a lot of time and configuration. To run the operations on a single operating system, queue.pl was changed to run.pl in directory of /home-/sidrah/kaldi/egs/voxforge/s5/cmd.sh. The main directory of Kaldi is "Kaldi-trunk" which further contains 5 sub-directories:

1. /egs

2. /misc
3. /src
4. /tools
5. /windows

The "egs" directory contain the example scripts for building Automatic Speech Recognition (ASR) system which contains almost 30 corpus examples of speech. The "misc" directory contain extra tools and supplies which is not necessarily needed for Kaldi. All the source code is in "src" directory. All the libraries and related tools to different example scripts are in "tool" directory. For using Kaldi in windows, the tool for windows are in "windows" directory. All the stuff that is audio data, acoustic data and language data are in /home/kaldi/egs/voxforge/s5 directory.

5.2 Data Set Specification

The data set used in this research was collected from the research article [41]. The data was collected during the speech therapy of the profound hearing impaired children. The session (individual and group) was held specially designed for the language therapy. The special school was located in H-9/4 Islamabad, Pakistan; National Special Education Center for Hearing Impaired Children (NSEC-HIC). Total number of speakers in this research were 12 that is 1 normal hearing person and 11 hearing impaired. The structure of the words were based on the placement of vowels in the words that is, in the beginning, middle and end of the words like apple, Lozina and purple respectively. Total of 72 words were used as shown in table 3.1.

The age group of the hearing impaired children were 5-8 years (both genders; male and female). The visual cues used for speech therapy was lip-reading as the deaf children can not hear, they have other sharp senses; they understand lip-reading more clearly as compared to normal hearing person. Total of 523 audio files (contained speech of normal hearing person and hearing impaired children) were collected (396 audio files of training data and 127 audio files for testing data) and each audio file is of 15 to 20 seconds long.

Table 5.1: Words used in data set

A	Ankle	Ant	Apple	Arm	Arrow
Banana	Blue	Cat	Chili	Cow	Cup
Dog	Dot	E	Ears	Egg	Elephant
Elbow	Eyes	Fig	Glue	Gorilla	Hut
I	Ice	Igloo	Ill	Ink	Insect
Jam	Jet	Jug	Kiwi	Knee	Koala
Leg	Lid	Lip	Lit	Lozina	Mango
Map	Mug	O	Old	Orange	Oval
Oven	Owl	Peg	Pot	Potato	Purple
Queue	Rat	Red	Rod	Table	Tap
Tomato	Tub	Two	U	Umbrella	Umpire
Under	Up	Vase	Wet	Zero	Zip

5.3 Performance Metric

There are three types of errors a system can make,

- A substitution, where one word is incorrectly recognized as a different word
- A deletion, where no word is hypothesized when the reference transcription has one, and
- An insertion where the hypothesized transcription inserts extra words not present in the reference

The overall WER can be computed as

$$WER = \frac{N\_sub + N\_ins + N\_del}{N\_ref}$$

where Nsub, Nins, Ndel are the number of substitutions, Number of insertions and Number of deletions, respectively and Nref is the number of words in the reference transcription. Where Reference transcription is the ground truth which means "what was actually said". The WER is computed using a string Edit Distance between the reference transcription and the hypothesized transcription. Edit Distance is to determine

how dissimilar two strings (for example words) are to one another by counting the minimum number of operations required to transform one string into the another. It is commonly believed that a lower word error rate shows superior accuracy in recognition of speech, compared with a higher word error rate.

5.4 Recognition Results

Our data set was split into two subsets that is training data and test data on 70:30 basis. Fitted our model on training data and then predicted our model on test data. Training of models were done on four parallel threads. The word error rate (WER) was not calculated for each speaker seperately but calculated word error rate (WER) of overall system (Combined WER of all speakers). Recognized our results on two types of data which was

1. Recognized word error rate (WER) before pre-processing
2. Recognized word error rate (WER) after pre-processing

5.4.1 Before Pre-processing

Before pre-processing, the data contained noise and the microphone from the speaker was at distant and for this reason the voice of the speaker was not properly recognizable by the speech recognition system. The background noise, like the conversation of instructor or other hearing impaired children were also interrupted in speech therapy sessions. Trained model four times by changing parameters of the model during training. The result of first model monophone in each of four training is shown below. Only best word error rate is shown in the table.

Table 5.2: Recognition Result of Monophone Model

Model	***Word Error Rate***
Monophone	71.2%

There were total of 47 insertions, 408 deletions and 466 substitution of words. Total words were 1294. According to our performance metric, the calculation of word error

rate is: The overall WER can be computed as

$$WER = \frac{N_{sub} + N_{ins} + N_{del}}{N_{ref}}$$

$$WER = \frac{466 + 47 + 408}{1294}$$

$$WER = 71.2\%$$

Now the other three models that is three models of triphone were trained by changing two different parameters. The first parameter was the *Total number of leaves* of the decision tree. As mentioned in the *Background* chapter that in speech recognition, hidden Markov models HMMs are used to model the acoustic observations (feature vectors) at the subword level, such as phonemes. It is typically for each phoneme to be modeled with 3 states, each state in the HMM had a probability distribution defined by a Gaussian Mixture Model (GMM). Thus, each state of the model has its own GMM. Grouping a set of context-dependent triphone states is performed using a decision-tree clustering process. A decision tree is constructed for every state of every context-independent phone and explanation is out of the scope. So, the number of leaves is the maximum number of leaves in tree. The tree is used for phones combining in the same classes (triphones which have the same or almost the same pronunciation). So it means that program will be able to recognize up to *num_leaves* different triphones. The second parameter was *Total Gaussian* which returns the number of mixture components in the gaussian mixture model (GMM). After tree composition each leave (each triphone) is associated with one Gaussian. In each iteration number of Gausses is increased. The number of Gaussians is the total over all leaves. The recognition results of the three models are shown in table 4.2, 4.3 and 4.4.

1. Number of mixture components in GMM
2. Number of leaves of decision tree

The best results were obtained by *Triphone2a* model.

5.4.2 After Pre-processing

Explained in *Methodology* chapter, the pre-processing was done due to the data that was collected was recorded by the mobile microphone, which was not of good quality

Table 5.3: Word error rate (WER) obtained when decode with delta transformation *Triphone1* model using different model sizes before pre-processing

Number of Leaves	***Total Gaussian***	***Word Error Rate (WER)***
3,000	22,000	56.8%
2,000	11,000	60.1%
1,000	11,000	59.8%
1,000	9,000	60.1%

Table 5.4: Word error rate (WER) obtained when decode with delta plus delta-delta transformation *Triphone2a* model using different model sizes before pre-processing

Number of Leaves	***Total Gaussian***	***Word Error Rate (WER)***
3,000	22,000	56.9%
2,000	11,000	59.9%
1,000	11,000	60.4%
1,000	9,000	59.8%

and that was the main barrier encountered for building good speech recognition system. Another reason was that the microphone were placed far distant from the speaker and in the result the voice of the speaker was low and the speech recognition system was unable to capture the speech signals. First the segments of audio which contain the background conversations of instructor and other hearing impaired children were cut out by using the *Audacity* audio processor tool. After that the voices was amplified and after that the noise was removed. This results in great change of word error rate which shows that much better results still can be produced if more clean data set is obtained.

In pass one, pre-processing was done which only amplifies the speech of hearing impaired. In pass two the audio was listened again and the segments of audio which contain the background conversations of instructor and other hearing impaired children were cut out by using the *Audacity* audio processor tool. And remove the noise. The other reason of high word error rate before pre-processing was that there were more than four hundred out of vocabulary (OOV) words which was solved by adding them in in vocabulary (IV) words. The other reason was that there were some words missing in the symbol table

Table 5.5: Word error rate (WER) obtained when decode with LDA and MLLT transformation *Triphone2b* model using different model sizes before pre-processing

Number of Leaves	***Total Gaussian***	***Word Error Rate (WER)***
3,000	22,000	61.8%
2,000	11,000	61.2%
1,000	11,000	61.6%
1,000	9,000	61.1%

Table 5.6: Word error rate (WER) obtained after pre-processing

Model	***Pass 1***	***Pass 2***
Monophone	72.9%	71.3%
Triphone1	45.1%	39.4%
Triphone2a	44.1%	39.4%
Triphone2b	45.2%	39.6%

and the problem was resolved by adding the missing words which were in transcripts but not mention in */data/lang/words.txt* symbol table. In first pass, only the audio of test data was amplified. In second pass, both the audio of training data and test data was amplified. Each word error rate in table 4.6 is calculated below by describing the insertions, deletions and substitutions. [1]. [2]

$$WER = \frac{N_{sub} + N_{ins} + N_{del}}{N_{ref}}$$

$$WER_{Mono,Pass1} = \frac{639 + 57 + 247}{1294} = 72.9\%$$

$$WER_{Tri1,Pass1} = \frac{257 + 33 + 294}{1294} = 45.1\%$$

$$WER_{Tri2a,Pass1} = \frac{268 + 35 + 267}{1294} = 44.1\%$$

[1] See the 5.4.2 for more details about Pass 1 and Pass 2

[2] See the 5.4.1 for more details about Before Pre-processing and After Pre-processing

$$WER_{Tri2b,Pass1} = \frac{199 + 27 + 359}{1294} = 45.2\%$$

$$WER_{Mono,Pass2} = \frac{633 + 70 + 314}{1294} = 71.3\%$$

$$WER_{Tri1,Pass2} = \frac{240 + 40 + 282}{1294} = 39.4\%$$

$$WER_{Tri2a,Pass2} = \frac{210 + 38 + 314}{1294} = 39.4\%$$

$$WER_{Tri2b,Pass2} = \frac{180 + 25 + 359}{1294} = 39.6\%$$

Out of four models, *Triphone2a* shows the best word error rate.

5.4.3 Alignment Results

For the simplicity, audio was divided into three segments. Segment of audio in which only the hearing impaired children spoke, segment of audio in which only the normal hearing person spoke and segment in which both spoke simultaneously. These results were needed to estimate the ratio of speech of hearing impaired and normal hearing participation in each audio on average. This helped us in evaluating the word error rate of only deaf voices which was the main goal and objective of this speech recognition system. Because audio was also contained the speech of normal hearing person which in return affects the word error rate. Audio files were selected randomly to avoid bias selection of data.

Each audio was listened and calculate the number of words spoken by deaf and how many words were spoken by the normal hearing person which was divided by the total number of words spoken in a single audio file. There were some segments of audio in which both speakers started to speak simultaneously. Overlapping segment was considered as one segment or one word. Detailed calculations of some speakers are shown here.

This overlapped segments were not detected by speech recognition system that was also the reason behind high word error rate. The formula used for calculating the ratio of overlapped segments was number of overlapped segments divided by the total number of unique segments. Each overlapped segment (Voices of both speaker simultaneously) is considered as one segment.

Table 5.7: Ratio of Speech of Hearing Impaired and Normal Hearing in an audio

Speaker	*Word*	*HI Words*	*NH Words*	*Total Words*	*Ratio of HI Voice*	*Ratio of NH Voice*
Shiza	Arrow	7	4	11	63.6%	36.4%
Ahmed	Elephant	14	7	21	66.7%	33.3%
Ahmed	Igloo	11	5	16	68.8%	31.3%
Ahmed	Ink	10	4	14	71.4%	28.6%
Ahmed	Jet	15	7	22	68.2%	31.8%
Ahmed	Leg	10	4	14	71.4%	28.6%
Ahmed	Ill	9	5	14	64.3%	35.7%
Ayesha	Arrow	12	6	18	66.7%	33.3%
Fatima	Ant	10	6	16	62.5%	37.5%

Table 5.8: Alignment of reference and hypothesis transcription of speaker Ayesha with pronounced word jam

Speaker	*HI*	*NH*	*HI*	*HI*	*NH*	*HI*	*HI*
Reference	JAAP	JAM	AAPP	AAPP	JAM	JAAP	AAPP
Hypothesis	JAAP	***	AA	AAPP	JAM	JAAP	AAPP
Operation	C	D	S	C	C	C	C
Overlap	-	O	O	-	-	-	-

On the basis of these speech ratios, word error rate was estimated of hearing impaired voice only. Now the alignments results with their word error rates are shown below.

The word error rate of only hearing impaired is shown below for the alignment of speaker Ayesha with pronounced word jam of table 4.8.

$$WER = \frac{N_{sub} + N_{ins} + N_{del}}{N_{sub} + N_{del} + N_{corr}}$$

$$WER_{Ayesha,Jam}HI = \frac{1 + 0 + 0}{1 + 0 + 4} = 20\%$$

$$WER_{Ayesha,Jam}NH = \frac{0 + 1 + 0}{0 + 1 + 1} = 50\%$$

$$WER_{Ayesha,Jam} = \frac{0.7 * 20\%}{0.3 * 50\%} = 29\%$$

The subscript of WER (Ayesha, Jam) represents the file name of the audio whereas the HI and NH of WER in left hand side of the equation represents the speech of HI and NH in the same audio file which is 29%.

Table 5.9: Alignment of reference and hypothesis transcription of speaker Ahmed with pronounced word Ill

Speaker	***NH***	***HI***	***NH***	***HI***	***NH***	***HI***	***NH***	***HI***	***Insertion***
Reference	ILL	AH	ILL	ULL	ILL	ULL	ILL	ULL	***
Hypothesis	ILL	***	ILL	ULL	ILL	ULL	ILL	ULL	ILL
Operation	C	D	C	C	C	C	C	C	I

The word error rate of only hearing impaired is shown below for the alignment of speaker Ahmed with pronounced word Ill of table 4.9.

$$WER_{Ahmed,Ill}HI = \frac{0 + 1 + 1}{0 + 1 + 3} = 50\%$$

$$WER_{Ahmed,Ill}NH = \frac{0 + 0 + 1}{0 + 0 + 4} = 25\%$$

$$WER_{Ahmed,Ill} = \frac{0.7 * 50\%}{0.3 * 25\%} = 42.5\%$$

Table 5.10: Alignment of reference and hypothesis transcription of speaker Shiza with pronounced word Tap

Speaker	***HI***	***HI***	***HI***	***HI***	***NH***	***HI***	***NH***	***HI***	***NH***	***HI***
Reference	TAP	TAP	TAP	TAP	TAP	TAAP	TAP	TAP	TAP	TAP
Hypothesis	TAP	TAP	TAP	TAP	***	AP	***	***	TAP	TAP
Operation	C	C	C	C	D	S	D	D	C	C
Overlap	-	-	-	-	-	-	O	O	-	-

The word error rate of only hearing impaired is shown below for the alignment of speaker Shiza with pronounced word Tap of table 4.10.

$$WER_{Shiza,Tap}HI = \frac{1 + 1 + 0}{0 + 1 + 6} = 28.6\%$$

$$WER_{Shiza,Tap}NH = \frac{0 + 2 + 0}{0 + 2 + 1} = 66.7\%$$

$$WER_{Shiza,Tap} = \frac{0.7 * 28.6\%}{0.3 * 66.7\%} = 40.03\%$$

Table 5.11: Alignment of reference and hypothesis transcription of speaker Khatija with pronounced word Cat

Speaker	***HI***	***HI***	***NH***	***HI***	***Insertion***	***NH***	***NH***	***HI***
Reference	AAEYY	YAAEY	CAT	YAAEY	***	CAT	CAT	EE
Hypothesis	AAEYY	YAAEY	CAT	YAAEY	CAT	CAT	***	***
Operation	C	C	C	C	I	C	D	D
Overlap	-	-	-	-	-	-	O	O

The word error rate of only hearing impaired is shown below for the alignment of speaker Khatija with pronounced word Cat of table 4.11.

$$WER_{Khatija,Cat}HI = \frac{0 + 1 + 1}{0 + 1 + 3} = 50\%$$

$$WER_{Khatija,Cat}NH = \frac{0 + 1 + 1}{0 + 1 + 2} = 66.7\%$$

$$WER_{Khatija,Cat} = \frac{0.7 * 50\%}{0.3 * 66.7\%} = 55.01\%$$

5.5 Speaker Diarization Results

Speaker diarization was performed on data because the audio contain both the voices of hearing impaired and normal hearing person and our objective was to estimate the word error rate of only hearing impaired children. As the audio was the speech of hearing impaired that is why the results of speaker diarization was not successful. The other reason

was that the audio contained too much overlapped voices which was not recognizable by the system. The LIUM toolkit which we have used for speaker diarization does not processed mobile recordings very efficiently that was also the reason behind not success full results of speaker diarization. For these reasons we have to calculate manually word error rate for hearing impaired by estimating the ration of hearing impaired voice in an audio which was described above in sub section *Alignment Results*.

Detailed explanation of output file with extension *.seg* was described in *Methodology* chapter. The results are shown here.

5.5.1 Before Pre-processing

The results shown in this section of speaker diarization are before pre-processing the data.

Table 5.12: Speaker Diarization Results Before Pre-processing

Speaker	***Word***	***Cluster***	***Segment***	***Start of Segment***	***Duration of Segment***	***Gender***	***Label***
Ayesha	Koala	1	1	183	182	F	S0
Izza	Arm	1	2	78	624	F	S0
Izza	Arm	1	2	702	345	F	S0
Izza	Arrow	2	4	0	281	F	S0
Izza	Arrow	2	4	301	262	F	S0
Izza	Arrow	2	4	1594	270	F	S0
Izza	Arrow	2	4	739	273	F	S1
Laiba	Jug	1	3	0	633	F	S0
Laiba	Jug	1	3	644	156	F	S0
Laiba	Jug	1	3	886	387	F	S0
Muneeb	Queue	1	2	0	236	M	S0
Muneeb	Queue	1	2	251	428	M	S0
Ahmed	Jet	1	4	0	428	M	S0
Ahmed	Jet	1	4	856	376	M	S0
Ahmed	Jet	1	4	1328	152	M	S0
Ahmed	Jet	1	4	1525	171	M	S0

5.5.2 After Pre-processing

The results shown in this section of speaker diarization are before pre-processing the data.

Table 5.13: Speaker Diarization Results After Pre-processing

Speaker	*Word*	*Cluster*	*Segment*	*Start of Segment*	*Duration of Segment*	*Gender*	*Label*
Ayesha	Koala	1	2	0	163	F	S0
Ayesha	Koala	1	2	201	157	F	S0
Izza	Arm	1	2	81	621	F	S0
Izza	Arm	1	2	702	341	F	S0
Izza	Arrow	1	1	0	156	F	S0
Laiba	Jug	1	2	0	318	F	S0
Laiba	Jug	1	2	1010	263	F	S0
Muneeb	Queue	1	2	251	198	M	S0
Muneeb	Queue	1	2	470	209	M	S0
Ahmed	Jet	1	3	103	205	M	S0
Ahmed	Jet	1	3	855	272	M	S0
Ahmed	Jet	1	3	1525	171	M	S0

The first audio file (Ayesha, Koala) had two clusters in an audio file means two speakers but the results showed that there are only one cluster which is not correct. There were 6 to 7 segments in this audio but the only two were recognizable after pre-processing. Before pre-processing only hearing impaired voice was recognized and after pre-processing, normal hearing person's voice was recognized. In the same way, the audio file (Izza, Arm) has two speakers per file but LIUM Diarizer recognized only one speaker. From time 0.78 milliseconds to 3.45 milliseconds , both deaf and hearing impaired were speaking in alternate turns but diarizer considers them as one speaker. These were the problems faced by using LIUM diarizer.

5.6 Evaluation Results

To check the robustness of the system, we adopt another approach. Another speech recognizer called *Google Cloud Speech-to-Text* API was used. At each time the new session was started, the environment variable was set on *Bash* terminal every time. Another major drawback was that the single audio file had to be processed separately by using the command *gcloud ml speech recognize <path of wav file>*. Set three parameters with this command

1. –language-code='en-US'
2. –include-word-time-offsets
3. –hints [HINTS...]

With each options of alternate transcripts, each was given a confidence score between 0.0 to 1.0 which shows that from multiple options of transcripts, the high confidence transcript is the most likely correct transcript. If the result shows blank curly braces, it shows that no transcript was detected by the *Google Cloud Speech-to-Text* API and the reason behind this, is the poor quality of audio. The results are shown here before processing and after processing of data. Alignment of reference and hypothesis transcriptions are also shown. [3]

$$WER_{Ayesha,Jam} = \frac{0 + 6 + 0}{0 + 6 + 1} = 85.7\%$$

$$WER_{Shiza,Tap} = \frac{1 + 9 + 0}{1 + 9 + 0} = 100\%$$

$$WER_{Izza,Cat} = \frac{0 + 13 + 0}{0 + 13 + 1} = 93\%$$

$$WER_{Khatija,Ill} = \frac{1 + 9 + 0}{1 + 9 + 0} = 100\%$$

Table 5.14: *Google Cloud Speech-to-Text API Results Before Pre-processing*

Speaker	***Word***	***Hypothesis Transcript***	***Start-End Time***	***Confidence***
Ayesha	Koala	Koala Koala	0.500s-1.200s, 1.200s-2s	0.7746
Izza	Arm	-	-	-
Izza	Banana	Banana Banana Banana	0.100s-1s, 1s-1.500s, 1.500s-3.300s	0.9727
Laiba	Jug	Jacqueline	5.200s-6.100s	0.9507
Laiba	Map	Math Papa	0.600s-1.200s, 1.200s-1.600s,	0.8884
Laiba	Map	Map of	2.800s-3.400s, 3.400s-4s,	0.6659
Ahmed	Jet	JetBlue	11.700s-12.600s	0.8933
Ahmed	Jet	Vector	14.100s-14.900s	0.5870
Ahmed	Jet	Jackson	16.400s-17.400s	0.9407

$$WER_{Muneeb,Umbrella} = \frac{0 + 9 + 0}{0 + 9 + 3} = 75\%$$

5.7 Discussion

If we compare our system with the approach used for evaluation, our system performs very well. The *Google Cloud Speech-to-Text* API build for normal hearing person whereas we build our dictionary, pronunciation model, language model so according to the speech of hearing impaired very well. Let us compare our results of both systems, speaker by speaker. The results of speaker *Ahmed* with the pronunciation word *Ill* had correctly recognized 7 out of 8 words with 1 deletion and 1 insertion with error rate of 42.5% whereas this same audio of *Ahmed* with the pronunciation word *Ill* was processed by using the *Google Cloud Speech-to-Text* API, not a single word was correctly recognized. If the result shows blank curly braces, it shows that no transcript was detected by the *Google Cloud Speech-to-Text* API because of the poor quality of audio. With each options of alternate transcripts, each was given a confidence score between 0.0 to 1.0 which shows that from multiple options of transcripts, the high confidence transcript is the most likely correct transcript. The results of speaker *Ayesha* with the pronunciation

[3]See the 5.4.1 for more details about Before Pre-pocessing and After Pre-processing

Table 5.15: *Google Cloud Speech-to-Text API Results After Pre-processing*

Speaker	***Word***	***Hypothesis Transcript***	***Start-End Time***	***Confidence***
Ayesha	Koala	cola-cola	0.500s-2.900s	0.7010
Izza	Arm	-	-	-
Izza	Banana	Banana Banana	0.100s-1s, 1s-1.500s	0.9540
Laiba	Jug	Jacqueline	0.300s-1.200s	0.7912
Laiba	Jug	Joplin	2.700s-3.400s	0.8864
Laiba	Jug	Japan	7.300s-8.200s	0.8757
Laiba	Map	Math Papa	0.500s-1.200s, 1.200s-1.600s,	0.4973
Laiba	Map	Map of	2.700s-3.400s, 3.400s-4s,	0.6616
Ahmed	Jet	Jack Depp	0s-0.400s, 0.400s-1.100s	0.8695
Ahmed	Jet	Jack Splash Jack Jack	2.400s-3s, 3s-4.200s, 4.200s-5.100s, 5.100s-7.700s	0.6800
Ahmed	Jet	Vector	14.100s-14.900s	0.7823
Ahmed	Jet	Jackman	11.700s-12.600s	0.6121

word *Jam* had correctly recognized 5 out of 7 words with 1 deletion and 1 substitution because of overlapped voices with error rate of 29% whereas this same audio of *Ayesha* with the pronunciation word *Jam* was processed by using the *Google Cloud Speech-to-Text* API, only 1 word was correctly recognized out of 7 with confidence of 85.4%. The results of speaker *Ayesha* with the pronunciation word *Rat* had correctly recognized 7 out of 10 words with 1 deletion and 2 substitution with error rate of 30% whereas this same audio of *Ayesha* with the pronunciation word *Rat* was processed by using the *Google Cloud Speech-to-Text* API, not a single word was correctly recognized. The results of speaker *Shiza* with the pronunciation word *Tap* had correctly recognized 6 out of 10 words with 3 deletion and 1 substitution because of overlapped voices, with error rate of 40% whereas this same audio of *Shiza* with the pronunciation word *Tap* was processed by using the *Google Cloud Speech-to-Text* API, only 1 word was recognized which was incorrect out of 10 words that is *Stop* with confidence of 86.9%.

We concluded from the evaluation results that speech recognition systems that was built for the voice recognition of normal hearing person can not be used for the speech recognition of hearing impaired children. Special dictionary has to be prepared for this kind of speech. Hearing impaired children had their own way of spoken language whether

Table 5.16: Evaluation results of alignment of reference and hypothesis transcription of speaker Ayesha with pronounced word Jam

Speaker	***HI***	***NH***	***HI***	***HI***	***NH***	***HI***	***HI***
Reference	JAAP	JAM	AAPP	AAPP	JAM	JAAP	AAPP
Hypothesis	***	***	***	***	JAM	***	***
Operation	D	D	D	D	C	D	D

Table 5.17: Evaluation results of alignment of reference and hypothesis transcription of speaker Shiza with pronounced word Tap

Speaker	***HI***	***HI***	***HI***	***HI***	***NH***	***HI***	***NH***	***HI***	***NH***	***HI***
Reference	TAP	TAP	TAP	TAP	TAP	TAAP	TAP	TAP	TAP	TAP
Hypothesis	STOP	***	***	***	***	***	***	***	***	***
Operation	S	D	D	D	D	S	D	D	D	D

they pronounce English words like *apple*, *chair*, *gorilla* but they have different fluent and way of articulating that word which must be understood by the speech recognition system. That was our goal which we achieved to design a speech recognition system specially hearing impaired children.

Table 5.18: Evaluation results of alignment of reference and hypothesis transcription of speaker Izza with pronounced word Cat

Speaker	***NH***	***HI***	***NH***	***NH***	***HI***	***NH***	***HI***	***NH***	***HI***	***HI***	***NH***	***HI***	***NH***	***I***
Reference	CAT	KAT	CAT	CAT	KAT	CAT	AAT	CAT	AAT	AAT	CAT	AAT	CAT	A
Hypothesis	***	***	***	***	***	CAT	***	***	***	***	***	***	***	*
Operation	D	D	D	D	D	C	D	D	D	D	D	D	D	

Table 5.19: Evaluation results of alignment of reference and hypothesis transcription of speaker Khatija with pronounced word Ill

Speaker	***NH***	***HI***	***NH***	***HI***	***NH***	***HI***	***NH***	***HI***	***HI***	***NH***
Reference	ILL	ELLEYA	ILL	ELLEYA	ILL	ILL	ILL	ILL	ELLEYA	ILL
Hypothesis	***	***	***	***	ANIMALS	***	***	***	***	***
Operation	D	D	D	D	S	D	D	D	D	D

Table 5.20: Evaluation results of alignment of reference and hypothesis transcription of speaker Muneeb with pronounced word Umbrella

Speaker	***NH***	***HI***	***NH***	***HI***	***NH***	***HI***	***NH***	***HI***
Reference	UMBRELLA	BELLA	UMBRELLA	ELLAH	UMBRELLA	ALLAH	UMBRELLA	BALL
Hypothesis	UMBRELLA	***	UMBRELLA	***	UMBRELLA	***	***	***
Operation	C	D	C	D	C	D	D	D

Chapter 6

Conclusion

The objective of this research was to build an Automatic Speech Recognition (ASR) system for hearing impaired children. This dissertation focuses on applying the Artificial Intelligence (AI) techniques on speech data of hearing impaired children. Speech of the profound hearing impaired children were abrupt and was a challenging task for listener to end up with accurate transcriptions of the utterances.

For this research work, speech therapy sessions was held for hearing impaired children. In these sessions, hearing impaired children learned how to articulate English words through speech therapy.

Language therapy of single word pronunciation of English was practiced. The structure of the words were based on the placement of vowels in the words that is, in the beginning, middle and end of the word like apple, Lozina and purple respectively. Total of 72 words were used as shown in table. Lip-reading videos were shown for each word pronounced to hearing impaired children and asked them to practice those words with instructor and therapist. The age group of the hearing impaired children were 5-8 years (both genders; male and female).

For this speech data that was acquired [10], checked how accurately those words were articulated. Depending on the accuracy achieved by recognition results, therapist can make hearing impaired children practiced those words again which were not correctly pronounced.

The back bone of the system was to build the ground truth for speech data of hearing impaired children which was a time taking process. Twelve speakers were considered

for training data. The structure of the words were based on the placement of vowels in the words that is, in the beginning, middle and end of the words like apple, Lozina and purple respectively. Total of 72 words were used. The age group of the hearing impaired children were 5-8 years.Total of 523 audio files (contained speech of normal hearing person and hearing impaired children) were collected (396 audio files of training data and 127 audio files for testing data) w.r.t 70:30 ratio and each audio file is of 15 to 20 seconds long.

Four setups were used in this research. *Kaldi* speech recognition tool was used for building our own system. For data pre-processing, *Audacity* audio processing tool was used. *LIUM Diarization* was used for speaker identification to fulfill our goal of achieving only WER of the speech of hearing impaired. And to conclude and verify our results, robustness of the system was checked by using *Google Cloud Speech-to-Text* API which is a speech recognition engine build for normal hearing person.

Five phases that summarizes our research are as follows.

1. Data Pre-processing
2. Pronunciation Model
3. Languuage Model
4. Acoustic Model
5. Evaluation

For data pre-processing, amplification and noise reduction was processed on speech data of hearing impaired children. To represent the voice signal, there is a representation which contains all the important information of signal which are useful for acoustic analysis. The structure of phonemes and consonants (basic unit of speech recognition system) are visible in frequency domain and that was the reason to generate MFFC features. So the features are then fed in to what is known as acoustic model. The very first component of speech recognition system that is *Acoustic Analysis* just looks at the raw speech waveform and convert it into some representation, like Mel Frequency Cepstral Coefficient (MFCC) features and generate acoustic features which represent the whole information contained in the speech signal.

Four types of acoustic models were trained on our data. Acoustic models were differentiated by the type of features it use and what type of *Ngram* language model it uses. HMM-GMM model was used as an acoustic model. Ngram language models that is monophone and triphone with different types of feature extensions were used for acoustic model. The first HMM-GMM acoustic model used monophone language model which refers to single phoneme for each word and referred to as context-independent phonemes. This results in 228 context-independent HMM states (76 phones with three states per phone; 76*3). The second HMM-GMM acoustic model uses trigram language model with features as delta which refers to context dependent phonemes and results in 1,316,928 (76 power cube into 3 states per phone) HMM states as the number of triphones is N cube. The third HMM-GMM acoustic model also uses trigram language model with different set of features that delta+delta-delta. Fourth HMM-GMM acoustic model also uses trigram language model with LDA+MLLT as features.

Models were trained on data before pre-processing and on data after pre-processing. Before pre-processing, models were trained four times with different model sizes. *Triphone1* with *Total number of leaves* were 3,000 and *Total Gaussian* were 22,000 achieved word error rate (WER) of 56.80%. After pre-processing, again the four acoustic models were trained which gives the best word error rate (WER) of *Triphone1* and *Triphone2a* of 39.41%. The difference of 18% word error rate was reduced after pre-processing and other changes which was mentioned in *Methodology* chapter.

Speaker diarization was performed with the goal of separating the speech signals of hearing impaired children and normal hearing person. But the results was not impressive because the speech was not the speech of normal hearing person but that was the speech of hearing impaired children. The speech of hearing impaired is not clear and abrupt which is not recognizable by those speech recognition systems which are build for normal hearing person. *LIUM Diarization* toolkit build for normal hearing person. One more reason was that the recordings of hearing impaired children were recorded by microphone of mobile and *LIUM Diarization* does not work well for mobile recordings. *LIUM Diarization* for data set did not specify the number of speakers correctly and also the segments were not accurately identified. For these reasons, the ratios of hearing impaired and normal hearing voice were estimated in each audio and the ratio was 70:30 respectively. That was the ratio used to calculate the WER of only hearing impaired children. Alignment results of *Hypothesis* transcription generated by the system and

Reference transcriptions derived by us which was ground truth, were presented of each speaker separately.

The robustness of our system was checked by adopting an approach mentioned in [7]. Another speech recognizer called *Google Cloud Speech-to-Text* API which is build for speech of normal hearing person was used. It is concluded from this evaluation and comparison from our system that our system performs very well for the speech of hearing impaired children. Special language model, transcriptions, pronunciation model and acoustic model were build to work for hearing impaired children whereas the *Google Cloud Speech-to-Text* used for normal hearing person and that was reason that our data of hearing impaired on *Google Cloud Speech-to-Text* was not impressive. Compared our results of both systems, speaker by speaker like the results of speaker *Ahmed* with the pronunciation word *Ill* had correctly recognized 7 out of 8 words with 1 deletion and 1 insertion with error rate of 42.5% whereas this same audio of *Ahmed* with the pronunciation word *Ill* was processed by using the *Google Cloud Speech-to-Text* API, not a single word was correctly recognized. If the result shows blank curly braces, it shows that no transcript was detected by the *Google Cloud Speech-to-Text* API because of the poor quality of audio. With each options of alternate transcripts, each was given a confidence score between 0.0 to 1.0 which shows that from multiple options of transcripts, the high confidence transcript is the most likely correct transcript. It was concluded from the evaluation results that speech recognition systems that was built for the voice recognition of normal hearing person can not be used for the speech recognition of hearing impaired children. Special dictionary has to be prepared for this kind of speech. Hearing impaired children had their own way of spoken language whether they pronounce English words like *apple*, *chair*, *gorilla* but they have different fluent and way of articulating that word which must be understood by the speech recognition system. That was the goal which was achieved to design a speech recognition system specially hearing impaired children.

6.1 Discussion

The nature of the data, pre-processing was achieved because the data that was collected was recorded by the mobile microphone, which was not of good quality and that was the main barrier encountered for building good speech recognition system and another

disadvantage was that that the microphone were placed far distant from the speaker and in the result the voice of the speaker was low and the speech recognition system was unable to capture the voice.

Before pre-processing, the data contained noise and the microphone from the speaker was at distant and for this reason the voice of the speaker was not properly recognizable by the speech recognition system. The background noise, like the conversation of instructor or other hearing impaired children were also interrupted in speech therapy sessions. The other reason of high word error rate before pre-processing was that because there were more than four hundred out of vocabulary (OOV) words which was solved by adding them in in vocabulary (IV) words. The other reason was that there were some words missing in the symbol table and the problem was resolved by adding the missing words which were in transcripts but not mention in */data/lang/words.txt* symbol table.

After pre-processing, there was 18% reduction in error rate was observed. Pre-processing was carried out which only amplifies the speech of hearing impaired, the audio was listened again and the segments of audio which contain the background conversations of instructor and other hearing impaired children were cut out by using the Audacity audio processor tool and remove the noise. In the first pass of after pre-processing, only the audio of test data was amplified. In the second pass, both the audio of training data and test data was amplified. The other major problem faced during recognition was the overlapped voices; overlapped segment of audio was not detected by the speech recognition system and not even the *LIUM Diarization* tool.

6.2 Future Work and Limitations

Although the provided analyses and methodologies were quite good but there are some improvements that can still be made. Many new areas can yet be explored, and some points from this work further clarified.

First and foremost is the need of lots and lots of data which is the prominent source of variability, to get good recognition results, our data was not enough to train for a good speech recognition results which was the main limitation of our system.

The data of hearing impaired should be collected in quiet environment for future work but in our data acquisition process, the background noise like the conversation of instruc-

tor or other hearing impaired children were also interrupted in speech therapy sessions. For recording data clearly, only one hearing impaired child must be entertain at a time, in some of the sessions, multiple hearing impaired children were speaking at the same which results in overlapped voices and the speech recognition system does not quite well handle the overlapped segments.

The other barrier was encountered regarding data acquisition was that in therapy sessions, the voice of therapist was also included in audios which was not necessary as the profound hearing impaired children does not understand voice but they were just using lip-syncing to understand the spoken words.

In future, must focus on getting more accurate and clear data and lots and lots of data. As the speech recognition systems are build upon data, and the backbone of speech recognition system is hundreds of hours of data but we had a limitation of data for our system which can be handled in future work.

Regarding *Acoustic* models, this research used HMM-GMM models which can be extended to HMM-DNN acoustic model in future work. And graphical user interface can be build in future to facilitate hearing impaired children and instructor.

References

[1] Jacob Aron. How innovative is apple's new voice assistant, siri?, 2011.

[2] Gustavo López, Luis Quesada, and Luis A Guerrero. Alexa vs. siri vs. cortana vs. google assistant: a comparison of speech-based natural user interfaces. In *International Conference on Applied Human Factors and Ergonomics*, pages 241–250. Springer, 2017.

[3] Matthew B Hoy. Alexa, siri, cortana, and more: An introduction to voice assistants. *Medical reference services quarterly*, 37(1):81–88, 2018.

[4] Frank R Lin, John K Niparko, and Luigi Ferrucci. Hearing loss prevalence in the united states. *Archives of internal medicine*, 171(20):1851–1853, 2011.

[5] In-Chul Yoo and Dongsuk Yook. Automatic sound recognition for the hearing impaired. *IEEE Transactions on Consumer Electronics*, 54(4):2029–2036, 2008.

[6] McCay Vernon and Soon D Koh. Early manual communication and deaf children's achievement. *American Annals of the Deaf*, pages 527–536, 1970.

[7] Sadaf Abdul Rauf, Aatka Javed Butt, Aliza Zahid, Ayesha Jabeen, Abdul Rauf Siddiqi, and Hina Shafique. Urdu language learning aid based on lip syncing and sign language for hearing impaired children. *International Journal of Computer Science and Information Security*, 14(12):478, 2016.

[8] C Jeyalakshmi, V Krishnamurthi, and A Revathi. Speech recognition of deaf and hard of hearing people using hybrid neural network. In *2010 2nd International Conference on Mechanical and Electronics Engineering*, volume 1, pages V1–83. IEEE, 2010.

[9] Kanwal Yousaf, Zahid Mehmood, Tanzila Saba, Amjad Rehman, Muhammad Rashid, Muhammad Altaf, and Zhang Shuguang. A novel technique for speech recognition and visualization based mobile application to support two-way communication between deaf-mute and normal peoples. *Wireless Communications and Mobile Computing*, 2018, 2018.

[10] Lozina Shoaib, Sharifullah Khan, Muhammad Azeem Abbas, and Ahmad Salman. Enabling profound hearing impaired children to articulate words using lip-reading through software application. *JPMA. The Journal of the Pakistan Medical Association*, 68(3):432–436, 2018.

[11] Mohamed Benzeghiba, Renato De Mori, Olivier Deroo, Stephane Dupont, Teodora Erbes, Denis Jouvet, Luciano Fissore, Pietro Laface, Alfred Mertins, Christophe Ris, et al. Automatic speech recognition and speech variability: A review. *Speech communication*, 49(10-11):763–786, 2007.

[12] Hank Liao, Erik McDermott, and Andrew Senior. Large scale deep neural network acoustic modeling with semi-supervised training data for youtube video transcription. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 368–373. IEEE, 2013.

[13] Hy Murveit, John Butzberger, Vassilios Digalakis, and Mitch Weintraub. Large-vocabulary dictation using sri's decipher speech recognition system: Progressive search techniques. In *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 319–322. IEEE, 1993.

[14] Raymond D Kent, Charles Read, and Ray D Kent. *The acoustic analysis of speech*, volume 58. Singular Publishing Group San Diego, 1992.

[15] Robert L Weide. The cmu pronouncing dictionary. *URL: http://www. speech. cs. cmu. edu/cgibin/cmudict*, 1998.

[16] Biing Hwang Juang and Laurence R Rabiner. Hidden markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

[17] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz,

et al. The kaldi speech recognition toolkit. Technical report, IEEE Signal Processing Society, 2011.

[18] Keith Vertanen. Baseline wsj acoustic models for htk and sphinx: Training recipes and recognition experiments. Technical report, Technical report). Cambridge, United Kingdom: Cavendish Laboratory, 2006.

[19] Lalit R Bahl, Frederick Jelinek, and Robert L Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE transactions on pattern analysis and machine intelligence*, (2):179–190, 1983.

[20] Andreas Stolcke. Srilm-an extensible language modeling toolkit. In *Seventh international conference on spoken language processing*, 2002.

[21] Andreas Stolcke. Srilm-an extensible language modeling toolkit. In *Seventh international conference on spoken language processing*, 2002.

[22] Hong-Kwang Jeff Kuo, Brian Kingsbury, and Geoffrey Zweig. Discriminative training of decoding graphs for large vocabulary continuous speech recognition. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 4, pages IV–45. IEEE, 2007.

[23] Don Grushkin. Speech recognition systems. URL https://www.quora.com/Why-doesnt-Google-develop-a-Speech-to-Text-app-for-the-Deaf-Hard-of-hearing.

[24] Robert L Weide. The cmu pronouncing dictionary. *URL: http://www. speech. cs. cmu. edu/cgibin/cmudict*, 1998.

[25] Kenneth N Stevens. *Acoustic phonetics*, volume 30. MIT press, 2000.

[26] Ilse Lehiste and Gordon E Peterson. Vowel amplitude and phonemic stress in american english. *The Journal of the Acoustical Society of America*, 31(4):428–435, 1959.

[27] Christian E Stilp and Keith R Kluender. Cochlea-scaled entropy, not consonants, vowels, or time, best predicts speech intelligibility. *Proceedings of the National Academy of Sciences*, 107(27):12387–12392, 2010.

[28] Kai-Fu Lee. *Automatic speech recognition: the development of the SPHINX system*, volume 62. Springer Science & Business Media, 1988.

[29] Thomas F Quatieri. *Discrete-time speech signal processing: principles and practice*. Pearson Education India, 2006.

[30] Chin-Hui Lee, Frank K Soong, and Kuldip K Paliwal. *Automatic speech and speaker recognition: advanced topics*, volume 355. Springer Science & Business Media, 2012.

[31] Lindasalwa Muda, Mumtaj Begam, and Irraivan Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *arXiv preprint arXiv:1003.4083*, 2010.

[32] CMU Pronouncing Dictionary. version 0.6, 2003.

[33] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[34] Jason Kincaid. *Shoebox*, July 12, 2018. URL https://medium.com/descript/a-brief-history-of-asr-automatic-speech-recognition-b8f338d4c0e5.

[35] Bruce T Lowerre. The harpy speech recognition system. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1976.

[36] Mark Gales, Steve Young, et al. The application of hidden markov models in speech recognition. *Foundations and Trends® in Signal Processing*, 1(3):195–304, 2008.

[37] C Jeyalakshmi, V Krishnamurthi, and A Revathi. Speech recognition of deaf and hard of hearing people using hybrid neural network. In *2010 2nd International Conference on Mechanical and Electronics Engineering*, volume 1, pages V1–83. IEEE, 2010.

[38] Kanwal Yousaf, Zahid Mehmood, Tanzila Saba, Amjad Rehman, Muhammad Rashid, Muhammad Altaf, and Zhang Shuguang. A novel technique for speech recognition and visualization based mobile application to support two-way communication between deaf-mute and normal peoples. *Wireless Communications and Mobile Computing*, 2018, 2018.

[39] K Resmi, Satish Kumar, HK Sardana, and Radhika Chhabra. Graphical speech training system for hearing impaired. In *Image Information Processing (ICIIP), 2011 International Conference on*, pages 1–6. IEEE, 2011.

[40] Rodrigo Jardim Riella, André Guilherme Linarth, Jr Lippermann, and Percy Nohama. Computerized system to aid deaf children in speech learning. In *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*, volume 2, pages 1449–1452. IEEE, 2001.

[41] Lozina Shoaib, Sharifullah Khan, Muhammad Azeem Abbas, and Ahmad Salman. Enabling profound hearing impaired children to articulate words using lip-reading through software application. *JPMA. The Journal of the Pakistan Medical Association*, 68(3):432–436, 2018.