Speech-to-Text System
for
Phonebook Automation

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Submitted By
Nishant Allawadi
(Roll No. 801032017)

Under the supervision of:
Mr. Parteek Bhatia
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004
June 2012

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*Speech-to-Text for Phonebook Automation*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Mr. Parteek Bhatia* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Signature:

Nishant

(Nishant Allawadi)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Parteek Bhatia

(Mr. Parteek Bhatia)
Asst. Professor, CSED

Countersigned by

(Dr. Maninder Singh)
Head
Computer Science and Engineering
Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

First of all, I would like to express my gratitude towards THAPAR UNIVERSITY, for providing me a platform to do my thesis work at such an esteemed institute.

I wish to express my deep gratitude to Mr. Parteek Bhatia, Assistant Professor, Computer Science and Engineering Department, Thapar University, Patiala for his valuable advice and guidance in carrying out my thesis.

I would like to thank Dr. Maninder Singh, Head, Computer Science and Engineering Department, Thapar University, Patiala who has been a constant source of inspiration for me throughout this work.

I am also thankful to all the staff members of the department for their full cooperation and help.

I am thankful to my parents, my sisters and all my friends for their blessing and moral support. Thanks for boosting me with their constant encouragement, support and confidence.

Last, but not the least, I am thankful to God for providing me with the strength and ability to complete my work.

Nishant

Nishant Allawadi

Abstract

In the daily use of electronic devices, generally, the input is given by pressing keys or touching the screen. There are a lot of devices which give output in the form of speech but the way of giving input by speech is very new and it has been seen in a few of our latest devices. The speech input technology needs a Speech-to-Text system to convert input speech into its corresponding text and speech output technology needs a Text-to-Speech system to convert input text into its corresponding speech.

In the work presented, a Speech-to-Text system has been developed with a Text-to-Speech module to enable speech as input as well as output. A phonebook application has also been developed to add new contacts, update previously saved contacts, delete saved contacts and call the saved contacts. This phonebook application has been integrated with Speech-to-Text system and Text-to-Speech module to convert the phonebook application into a complete speech interactive system.

The first chapter, the basic introduction of the Speech-to-Text system has been presented. The second chapter discusses about the evolution of the Speech-to-Text technology along with the existing Speech-to-Text systems. The problem statement has been discussed in the third chapter. This chapter also describes the objectives of the proposed system and the methodologies that have been followed to develop the proposed system.

The fourth chapter discusses about the implementation of the proposed system. It describes the requirements for the development of the proposed system. It also discusses about the installation of the tools like ***Cygwin, HMM***, ***Audacity*** *etc.* and explains the architecture of the Speech-to-Text system. This chapter describes the data preparation phase in the implementation of the proposed system. It also describes the training phases of the proposed system by monophones, triphones and tied-state triphones, using ***HMM Tool Kit*** commands and ***perl*** scripts. Along with it, the implementation of the phonebook application and Text-to-Speech module has been discussed in the last sections of this

chapter. The fifth chapter presents the results of the developed system. In the sixth chapter, the conclusion and future scope has been discussed.

The proposed system represents an idea to develop such a full fledge speech interactive phonebook automation system that can be used by blind people to use their mobile phones easily by controlling their phonebook application by speech.

Keywords: *HMM, monophones, triphones, dictionary, vocabulary, grammar, training.*

Table of Contents

List of Figures

Chapter 1

Introduction

As there are a few input devices, with the help of which human interacts with computer system, the mostly used devices are keyboard and mouse. It is difficult for users to use these devices when a lot of data is required to be entered in the system. Also the physically challenged people find difficult to use the computer systems due to these input devices. Even partially blind people find difficult to read from a monitor.

The above problem can be solved by changing the method of interaction with computer system. A new method that can be very easily used for the interaction is speech. A speech interface can help to tackle these problems. In order to achieve this, two kinds of systems are required. These are Speech-to-Text system and Text-to-Speech system.

Speech synthesis, *i.e.*, Text-to-Speech system and speech recognition, *i.e.*, Speech-to-Text system, together form a speech interface. A speech synthesizer converts the input text into its corresponding speech output. Thus it can read out the textual contents from the screen. A speech recognizer has the ability to understand the spoken words and convert it into text. So, it converts the speech input into its corresponding text output.

1.1 A Speech-to-Text system

Speech-to-Text (STT) is a technology that allows a computer to identify the words that a person speaks in a microphone. It is the process by which a computer maps an acoustic speech signal to text. Basically, it is the process of converting an acoustic signal that has been captured by a microphone, to a set of words. The recognized words may be the final results for a Speech-to-Text system but these words may act as input for applications like linguistic processing for further processing [2].

All Speech-to-Text systems rely on at least two models: an ***acoustic model*** and a ***language model***. In addition to this, large vocabulary systems use a pronunciation model.

An ***acoustic model*** is created by taking audio recordings of speech and their transcriptions. These are compiled and converted into a statistical representation of the sounds that make up each word through a process called ***training***. A ***language model*** is a file containing the probabilities of sequences of words. In this, the recognition is based on the probability of sequence that is this particular sequence of words possible.

It is important to understand that there is no such thing as a universal speech recognizer or Speech-to-Text system. To get the best transcription quality, all of these models can be specialized for a given language, dialect, application domain, type of speech, and communication channel. Like any other pattern recognition technology, Speech-to-Text conversion cannot be error free. The speech transcript accuracy is highly dependent on the speaker, the style of speech and the environmental conditions. Speech-to-Text conversion is a harder process than what people commonly think. Humans are used to understanding speech, not to transcribe it, and only speech that is well formulated can be transcribed without ambiguity.

1.2 Basic Terminology

There are some basic terms related to Speech-to-Text system. These are as follows.

- **Utterance**

  An utterance is the vocalization or speaking of a word or a number of words that represent a single meaning to the computer. Utterances can be a single word, a few words, a sentence, or even multiple sentences [17].

- **Phoneme**

  A phoneme is a sound or a group of different sounds for any word. For example, for a word ***delete***, the phoneme combination is ***"d ih l iy t"***. These five phonemes are different sounds of English language. The combination of these sounds form a word ***delete***. There are 44 phonemes or sounds in English language.

- **Monophone**

  As the word ***monophone*** suggests that it is a combination of two words, ***mono*** and ***phones.*** Here ***mono*** means single and ***phone*** refers to phoneme. So, a ***monophone*** is a single phoneme or a single sound. For example, a word ***update*** has monophones ***"ah p d ey t".***

- **Triphone**

  A ***Triphone*** is a combination of two or three monophones. The triphones for the word ***update*** are as shown in (1.1) .

```
ah+p
ah-p+d
p-d+ey
d-ey+t
ey-t                                                    …(1.1)
```

- **Vocabularies**

  Vocabularies or dictionaries are lists of words or utterances given with their phoneme combination that can be recognized by a Speech-to-Text system [6]. Generally, smaller vocabularies are easier for a computer to recognize, while larger vocabularies are more difficult. Unlike normal dictionaries, each entry doesn't have to be a single word. There can be multiple words in a single entry. For example ***Stand Up***, it is a multi word entry in vocabulary file.

- **Grammar**

  Grammar of a language is a set of structural rules which governs the composition of different classes of the words or classes of a combination of words [17]. A grammar is defined for a Speech-to-Text system to govern, which combination of words the system can recognize. For example, if a rule is like ***Phrase -> Dial Number.*** Here ***Dial*** and ***Number*** are non-terminals which further derive terminals

as ***Dial*** -> ***dial*** and ***Number*** -> ***one*** or ***Number*** -> ***two*** and so on. So, the ***Phrase*** may derive ***dial one, dial two*** and so on.

- **Training**

  The training refers to the process of re-estimation of the statistics of each monophone [6]. The statistics include mean and variance of the waveforms of the sounds for each monophone. At every step of training, the values of mean and variance get modified to more accurate values.

1.3 Types of Speech-to-Text system

Basically, Speech-to-Text system is of two types. These are as follows.

1.3.1 Speaker-Dependent system

Speaker-dependent systems are designed around a specific speaker. They generally are more accurate for the correct speaker, but much less accurate for other speakers. They assume the speaker will speak in a consistent voice and tempo. Speaker dependence describes the degree to which a Speech-to-Text system requires knowledge of a speaker's individual voice characteristics to successfully process speech. The Speech-to-Text system can learn how one speaks words and phrases [17].

1.3.2 Speaker-Independent system

Speaker-independent systems are designed for a variety of speakers. Adaptive systems usually start as speaker-independent systems and utilize various training techniques to adapt to the speaker to increase their recognition accuracy [17].

Speaker-dependent systems are trained to recognize the speech of only one particular speaker at a time. On the other hand, speaker-independent systems can recognize speech from anyone. Only one acoustic model is trained using training data from many speakers, this single model is used for recognition of speech by anyone whose voice it might not have seen. Speaker-dependent systems have higher recognition accuracies than speaker-independent systems. Thus, it is preferable to use speaker-dependent systems. However, in order to train speaker-dependent systems, a large amount of speaker specific training

data would be required which is not possible to have in case of a multi-user application such as telephonic enquiry system.

Speaker adaptation is a technique which reduces the difference between training and testing conditions by transforming the acoustic models using a small set of speaker specific speech data. It is also necessary to have the correct word transcriptions for the adaptation data for robust adaptation [35]. There are two types of speaker adaptation. In supervised adaptation, the text of the adaptation speech is known to the system. Supervised adaptation provides good improvement in the recognition rates as it is same as re-training the system. However, this process is not practical in a system designed for a telephonic enquiry that is used by practically everyone, and the user does not have the patience to provide enough adaptation data. Unsupervised adaptation is the approach in which the system automatically adapts itself to the present speaker at the same time as he keeps using the system. The system uses the first sentence spoken by a new speaker as the adaptation data. The, possibly incorrect, output of the speaker-independent system is assumed as the correct transcription. Using this transcription and the adaptation data, the system transforms its acoustic models in order to recognize the speech of the current user better. Then, it re-recognizes the unknown utterance with adapted models, hopefully resulting in better recognition accuracy. Since the speaker adaptation is carried out as and when a speaker is using the system, this approach is called online speaker adaptation.

1.4 Categories of Speech-to-Text system

A Speech-to-Text system falls under two categories. These are as follows.

1.4.1 Large vocabulary or general purpose Speech-to-Text system

This type of Speech-to-Text system has very large vocabulary. So, the system is able to recognize a very large number of words of a language. The recognition accuracy is not very excellent as the vocabulary is large. But still this type of system can be used in general purpose environment for general use. For example, speech recognition in ***Windows 7*** is almost a general purpose Speech-to-Text system. It supports almost all

commonly used vocabulary. These systems are very difficult to develop and need a huge training of the system.

1.4.2 Small vocabulary or specific purpose Speech-to-Text system

This type of Speech-to-Text system has a small vocabulary. So, the system is able to recognize a small number of words of a language. As this type of system can be used in specific purpose environment for specific applications, the recognition accuracy is very high. For example, a credit card reader may need a few words, *i.e.,* digits. The system is need to be trained on 10 digits, *i.e.,* 0-9 to develop this type of specific use applications. These systems are comparatively easy to develop and don't need a huge training of the system.

1.5 Applications of the Speech-to-Text system

Speech-to-Text system has a lot of applications in various fields. These applications are as follows.

- **Health Care**

Speech-to-Text system is applicable in hospitals for health care instruments. The instruments may take speech as input and perform actions corresponding to the speech. For example, an MRI machine can perform various tests. So, to perform a particular test, speech input can be given to perform that test. Along with the input of test, the personal information of the patient required at the time of test can also be given by speech input.

The Electronic Medical Records (EMR) applications can be filled more effectively and may be processed more easily when these are used in conjunction with a Speech-to-Text system. The searching process, queries and form filling may become faster to perform by speech than by using a keyboard.

- **Banking**

In banking, Speech-to-Text system can be implemented in input devices where numbers are given input as speech. These numbers can be account number of account holders. These also might be the credit card number or debit card number. The ATMs also may be

developed with input method of speech. It will be easier for blind people to use these type of machines in banks.

- **Aircrafts**

Speech-to-Text system can be widely used in aircraft systems, where pilots can give audio commands to manage operations in the flight. They can give speech inputs to increase the speed of the flight. The command to incline the flight can also be issued by a speech input. The Speech-to-Text system in aircraft is a life critical system. So, its accuracy can't be compensated to anything at any cost.

- **Mobile Phone**

Mobile phones are devices which use Speech-to-Text technology in its many applications. These applications may include writing text messages by speech input. The e-mail documentation can also be performed by speech as input. The mobile games can also be played by speech commands instead of pressing keys. Song selection from a list of songs can be made by speech input in the music player. Along with it, a contact can be called by speaking the contact name.

- **Computers**

Speech-to-Text systems can also be used in computers for writing text documents. It can be used for opening, closing and operating various applications in computers. For example, by speaking ***open my computer*** opens ***my computer*** window. So, computer can be operated by speech input [42].

- **Robotics**

Robotics is a new emerging field where inputs are given in speech format to robots. Robot processes the speech input command and perform actions according to that. For example, if a robot is asked to move forward by speech input, then it moves forward [13].

1.6 Model used in Speech-to-Text system

The model that has been used in the development of Speech-to-Text system is Hidden Markov Model (***HMM***).

The Hidden Markov Model was first described by Ruslan L. Stratonovich in 1960. It was first used as a basic concept of speech recognition in the mid of 1970's. It is a statistical model in which the system which is being modeled is assumed to be a markov process. The ***HMM*** is basically a stochastic finite set of states where each state is associated with a probability distribution. The transitions among these states depend upon a set of probabilities called transition probabilities. In a particular state, an outcome can be generated according to its associated probability distribution. It is only the outcome which is visible to an external observer. The states are not visible to the external observer and hence its name is Hidden Markov Model.

In speech recognition, a speech signal could be visualized as a piecewise stationary signal or a short-time stationary signal [3]. So, assuming a short-time signal that could be in the range of 10 milliseconds, the speech can be considered as a stationary process. Speech could thus be thought of as a markov model for many stochastic processes. This is the reason for adapting HMM for speech recognition. Also, the HMM can be trained automatically and it is simple and computationally feasible to use. So, this concept was used in the speech recognition technology.

In this chapter, the basics of a Speech-to-Text system have been discussed. Along with it, the types and categories of the Speech-to-Text system have been elaborated. The applications of the proposed system have been discussed. A brief introduction on the concept of Hidden Markov Model has been given. In the next chapter, the review of literature of the proposed system has been discussed.

Chapter 2
Literature Review

The Speech-to-Text conversion technology has always been an interesting field of research for the researchers working in Natural Language Processing (NLP). The research on this field had been started in early 1950's.

The research work is still going on to make this technology more accurate and generic. This chapter has been divided into three sections. The first section discusses about the research work that has been done since 1950's till now. The second section discusses about the existing Speech-to-Text systems. In the third section, Speech-to-Text systems for different languages have been discussed.

2.1 History of Speech-to-Text technology

The history of the Speech-to-Text technology is very immense as the research work that has been done in this field is appreciable. Moreover, the work has been started in 1950's. The size of the computer can be imagined of that time. It was as large as a size of a room. In that time, to work on this technology would have been a typical task. In order to reveal the history of the Speech-to-Text system, the discussion has been done decade by decade in next section.

2.1.1 Research work since 1950's

In this decade, various researchers tried to exploit fundamental ideas of acoustic phonetics. As the signal processing and computer technologies were not well developed, so, the most of the speech recognition systems that were investigated used spectral resonances during the vowel region of each utterance. The spectral resonances were extracted from output signals of an analogue filter bank and logic circuits.

In 1952, Davis *et al.* developed a system at Bell laboratories. This was an isolated digit recognition system for a single speaker [7]. This system was developed using the formant

frequencies estimated during vowel regions of each digit. After this in 1956, Olson and Belar made an effort in RCA laboratories [25]. They tried to recognize 10 distinct syllables of a single speaker, as embodied in 10 monosyllabic words. In 1959, Fry built a phoneme recognizer at University College in England [10]. The phoneme recognizer was built to recognize four vowels and nine consonants. They increased the overall phoneme recognition accuracy for words consisting of two or more phonemes. They did this by incorporating statistical information concerning allowable phoneme sequences in English. This work marked the first use of statistical syntax at the phoneme level in automatic speech recognition. In the same year, Forgie and Forgie devised a system at MIT Lincoln Laboratories [9]. Their system which was able to recognize 10 vowels embedded in a /b/ - vowel - /t/ format in a speaker-independent manner.

2.1.2 Research work since 1960's

In this decade, the technology used in computer was very complex. So, the computer systems were not fast enough. To solve the purpose, several special purpose hardware systems were built. In 1960, Suzuki and Nakata designed a hardware system in Radio Research Lab in Japan [36]. They built a vowel recognizer. This hardware system was able to recognize the vowels of Japanese language. In 1961, Sakai and Doshita designed a hardware system at Kyoto University [32]. They built a phoneme recognizer. They built it by using a hardware speech segmenter and zero crossing analysis of different regions of the input utterance. In 1963, Nagata *et al.* developed a recognizer in NEC laboratories [24]. They built a hardware digit recognizer.

There were a number of challenges in the development these systems. One of the difficult problems of speech recognition existed in the non-uniformity of time scales in speech events. In the 1960s, Martin *et al.* developed some methods at RCA Laboratories. They developed a set of elementary time-normalization methods [22]. These methods were based on the ability to reliably detect the speech's start point and end point. This significantly reduced the variability of the recognition scores.

In 1968, Vintsyuk proposed the use of dynamic programming methods for time aligning a pair of speech utterances, generally known as dynamic time warping (DTW). He also included algorithms for connected word recognition [41]. At the same time, Sakoe and

Chiba made an effort at NEC laboratories, Japan [33]. They also started to use a dynamic programming technique to solve the non-uniformity problem. In the late 1960s, Reddy conducted a pioneering research in the field of continuous speech recognition by dynamic tracking of phonemes at Carnegie Mellon University [31].

2.1.3 Research work since 1970's

In this decade, a number of significant milestones were achieved in the research field of speech recognition. The area of isolated word or discrete utterance recognition became a viable and usable technology based on fundamental studies in Russia and Japan. The researchers of Russia, Velichko and Zagoruyko, extended the use of pattern-recognition ideas in speech recognition.

In early 1970's, Itakura proposed some new ideas of linear predictive coding (LPC) at Bell laboratories [14]. He showed how these ideas could be extended to speech recognition systems through the use of an appropriate distance measure based on LPC spectral parameters. At the same time, researchers studied large vocabulary speech recognition for three distinct tasks in IBM Labs. These three tasks were namely the New Raleigh language, the laser patent text language and the office correspondence task, called Tangora. The New Raleigh language was for simple database queries. The laser patent text language was developed for transcribing laser patents [16]. The Tangora was for dictation of simple memos.

The researchers tried to develop a speaker independent speech recognition system at AT&T Bell Labs. In order to achieve this goal, a wide range of sophisticated clustering algorithms were used [29]. These were used to determine the number of distinct patterns required to represent all variations of different words across a wide user population. In 1973, a good achievement was got by CMU. They also developed a system, namely Hearsay I, which was able to use semantic information to significantly reduce the number of alternatives considered by the recognizer. They developed a system, namely Harpy, which was able to recognize speech using a vocabulary of 1,011 words with reasonable accuracy. One particular contribution from the Harpy system was the concept of graph search, where the speech recognition language is represented as a connected network

derived from lexical representations of words, with syntactical production rules and word boundary rules. The Harpy system was the first to take advantage of a finite state network (FSN) to reduce computation and efficiently determine the closest matching string. In the late 1970s, dynamic programming in numerous variant forms had become an indispensable technique in automatic speech recognition. These forms included the Viterbi algorithm which came from the communication theory community [40].

2.1.4 Research work since 1980's

In this decade, the problem of creating a robust system capable of recognizing a fluently spoken string of connected word was the focus of research. At NEC, Sakoe formulated and implemented an algorithm based on matching a concatenated pattern of individual words including the two-level dynamic programming approach. In Joint Speech Research Unit (JSRU), UK, Bridle and Brown formulated and implemented one-pass method [5]. At Bell Labs, Myers and Rabiner implemented the level-building approach [23]. Researchers Lee and Rabiner implemented the frame-synchronous level-building approach.

The research in this decade took a turn in the direction of statistical modeling. Speech recognition research was characterized by a shift in methodology from the more intuitive template based approach, *i.e.*, a straightforward pattern recognition paradigm, towards a more rigorous statistical modeling framework. Today, most practical speech recognition systems are based on the statistical framework developed in these years and their results, with significant additional improvements that have been made in the 1990's.

In 1980's, a key concept of Hidden Markov Model (HMM) came into existence in the field of speech recognition [28]. Although the HMM was well known and understood in laboratories like IBM, Institute for Defense Analysis (IDA) and Dragon Systems, but later this technique was widely used for speech recognition. Furui proposed the spectral features of the speech recognition technology [12]. He proposed how to use the combination of instantaneous cepstral coefficients and their first and second-order polynomial coefficients, now called and cepstral coefficients, as fundamental spectral features for speech recognition.

In IBM, the researchers were primarily focusing on a structure of a language model, *i.e.*, grammar. The structure was represented by statistical syntactical rules describing how likely, in a probabilistic sense, does a sequence of language symbols could appear in the speech signal. The researchers proposed the *n*-gram model, which defines the probability of occurrence of an ordered sequence of *n* words. The use of *n*-gram language models, and its variants, has become indispensable in large-vocabulary speech recognition systems [15]. Along with it, the concept of Neural networks was proposed. The researchers got the achievement of understanding the strengths and limitations of this technology. They also understood the relationship of this technology to classical pattern classification methods.

2.1.5 Research work since 1990's

In this decade, some techniques were developed to make the speech recognition more robust. The major techniques include Maximum Likelihood Linear Regression (MLLR) technique [20], the Model decomposition technique [39], Parallel Model Composition (PMC) technique and the Structural Maximum a Posteriori (SMAP) method [34]. These techniques were developed to overcome the problems coming due to background noises and other disturbances.

2.1.6 Research work since 2000's

During the years of this decade, Defense Advanced Research Projects Agency (DARPA) conducted a program, the Effective Affordable Reusable Speech-to-Text (EARS). This program was conducted to develop the Speech-to-Text technology with the aim of achieving substantially richer and much more accurate output than before [21]. The tasks included detection of sentence boundaries, fillers and disfluencies. The program was focusing on natural, unconstrained human-human speech from broadcasts and foreign conversational speech in multiple languages. The goal of the program was to make it possible for machines to do a much better job of detecting, extracting, summarizing and translating important information, thus enabling humans to understand what was said by reading transcriptions instead of listening to audio signals.

It was noted that although the read speech and similar types of speech, *e.g.* news broadcasts reading a text, could be recognized with accuracy higher than 95% using state-of-the-art speech recognition technology. But the recognition accuracy drastically decreased for spontaneous speech. In order to increase recognition performance for spontaneous speech, several projects were conducted. In Japan, a 5-year national project "Spontaneous Speech: Corpus and Processing Technology" was conducted [11]. The world-largest spontaneous speech corpus, "Corpus of Spontaneous Japanese (CSJ)" consisting of approximately 7 millions of words, corresponding to 700 hours of speech, was built, and various new techniques were investigated. These new techniques include flexible acoustic modeling, sentence boundary detection, pronunciation modeling, acoustic as well as language model adaptation, and automatic speech summarization.

2.2 Existing Speech-to-Text systems

There are a number of Speech-to-Text systems which are running successfully. Some of them have been discussed below.

2.2.1 Windows Speech Recognition

It is a speech recognition application included in Windows Vista and Windows 7. Windows Speech Recognition allows the user to control the computer by giving specific voice commands [42]. This program can also be used for the dictation of text so that the user can control their Vista or Windows 7 computer. Applications that do not present obvious commands can still be controlled by asking the system to overlay numbers on top of interface elements, the number can subsequently be spoken to activate that function. Programs needing mouse clicks in arbitrary locations can also be controlled through speech, when asked to do so, a mousegrid of nine zones is displayed, with numbers inside each. The user speaks the number, and another grid of nine zones is placed inside the chosen zone. This continues until the interface element to be clicked is within the chosen zone. Windows Speech Recognition has fairly high recognition accuracy and provides a set of commands that assists in dictation. A brief speech-driven tutorial has been included in the application which helps to familiarize a user with speech recognition commands.

Currently, the application supports several languages including English, Spanish, German, French, Japanese and Chinese *etc*.

2.2.2 Dragon Naturally Speaking

It is a speech recognition software package developed by Nuance Communications for Windows personal computers. Its most recent package is version 11.5, which supports 32-bit and 64-bit editions of Windows XP, Vista and 7. The Mac OS version is called Dragon Dictate. NaturallySpeaking utilizes a minimal user interface. As an example, dictated words appear in a floating tooltip as they are spoken (though there is an option to set this feature so it is not displayed to increase speed), and when the speaker pauses, the program transcribes the words into the active window at the location of the cursor. The software has three primary areas of functionality: dictation, text-to-speech and command input. The user is able to dictate and have speech transcribed as written text, have a document synthesized as an audio stream, or issue commands that are recognized as such by the program.

2.2.3 IBM ViaVoice

The IBM's ViaVoice is an efficient tool but it has some sizeable system requirements compared to the more basic speech recognition systems. It is a speaker independent system. The speech recognition standard development toolkit is available for free. It includes IBM's SMAPI, grammar API, documentation, and a variety of sample programs.

2.2.4 XVoice

XVoice is a continuous speech recognizer that can be used with a variety of X Windows applications. It allows user-defined macros. Once it sets up, it performs with adequate accuracy. XVoice requires the IBM's Via Voice tool to be installed. It is also important to note that this program interacts with X windows.

2.2.5 CVoiceControl

Console Voice Control (CVoiceControl) was initially known as KDE Voice Control (KVoiceControl). It is a basic speech recognition system that allows a user to execute Linux commands by using spoken commands. CVoiceControl replaced KVoiceControl. The software includes a microphone level configuration utility, a vocabulary named

model editor, for adding new commands and utterances, and the speech recognition system.

2.2.6 CMU Sphinx

Sphinx was originally started at CMU and it has released as open source. This is a fairly large program that includes a lot of tools and information. It includes trainers, recognizers, acoustic models, language models and documentation.

2.2.7 Abbot

Abbot is a very large vocabulary, speaker-independent speech recognition system. It was originally developed by the Connectionist Speech Group at Cambridge University and it was transferred to SoftSound later.

2.3 Speech-to-Text systems for different languages

The work of Speech-to-Text systems has been done in a number of languages like German, Arabic, Urdu, Finnish, Spanish, English *etc.* Some of these have been discussed below.

2.3.1 German language

For German language, a large vocabulary continuous speech recognition system was developed by M. Adda-Decker *et al.* [8]. The recognition system was originally developed for French and American English language which was further adapted by German language. The accuracy of the system was reported as 82.7%.

2.3.2 Urdu language

For Urdu Language, A.A. Raza *et al.* worked on phonetically rich speech corpus in 2009 [30]. They presented details of designing and developed an optimal context based phonetically rich speech corpus for Urdu language. That speech corpus served as a baseline model for training a large vocabulary continuous speech recognition system for Urdu language.

2.3.3 Finnish language

For Finnish language, the Speech-to-Text system was designed by Lamel and Vieru [19]. Finnish is a Finno-Ugric language spoken by about 6 million of people living in Finland, but also by some minorities in Sweden, Norway, Russia and Estonia. The proposed system development was carried out without any detailed manual transcriptions, relying instead on several sources of audio and textual data that were found on the web.

2.3.4 Arabic language

For Arabic language, Tomalin M. *et al.* described two aspects of generating and using phonetic Arabic dictionaries [37]. Firstly, they investigated the use of single pronunciation acoustic models in the context of Arabic large vocabulary speech recognition. These have been found to be useful for English speech recognition systems, when combined with standard multiple pronunciation systems. The second area they examined was automatically deriving phonetic pronunciations for words that have standard approaches.

2.3.5 Spanish language

For Spanish language, a database design was proposed by Peiiagarikano and Bordel [27]. The goal of their work was the design and realization of a database to be used in an automatic speech recognition system for a fixed telephone network. One thousand speakers, native to five Argentine dialectal regions, were recorded. A strategy of uniform information collection and recording was designed. Phone calls were received through a digital line connected to a computer, and monitored by acquisition software. This procedure resulted in an efficient way to collect data from a large number of speakers representing several different dialects in only two months.

2.3.6 Hindi language

For Hindi language, Kumar and Aggarwal built a speech recognition system for Hindi language [18]. They used Hidden Markov Model Toolkit (HTK) to develop the system. The system recognized the isolated words using acoustic word model. The system has been trained for 30 Hindi words. Training data has been collected from eight speakers. Their experimental results showed the overall accuracy of 94.63%.

Bapat and Nagalkar worked on phonetic speech analysis [4]. Their work aimed at generating phonetic codes of the uttered speech in training-less, human independent manner. This work was guided by the working of ear in response to audio signals. They explained and proved the scientific arrangement of the Devnagari script. They tried to segment speech into phonemes and identify the phoneme using simple operations like differentiation, zero-crossing calculation and FFT.

2.3.7 Tamil language

Anumanchipalli *et. al.* made efforts in the development of Indian language speech databases in Tamil, Telugu and Marathi for building large vocabulary speech recognition systems [1]. They collected speech data from about 560 speakers in these three languages. They developed the design and methodology of collection of speech databases. They also presented preliminary speech recognition results using the acoustic models created on these databases using Sphinx 2 speech tool kit.

In this chapter, the history of the Speech-to-Text technology has been discussed since 1950's. Some of the latest existing Speech-to-Text systems have also been discussed. The research work in different languages for the proposed system has been elaborated. In the next chapter, the problem statement has been discussed with the objectives of this thesis work and methodologies to be followed to develop such a system.

Chapter 3
Problem Statement

A Speech-to-Text system converts the input speech into its corresponding text. Nowadays, a lot of research work is going for the development of Speech-to-Text system with better accuracy. Existing general Speech-to-Text systems are not 100% accurate but the systems developed for particular domains have been very successful.

There are various applications based upon Speech-to-Text system. This technology may be implemented in health care devices used in hospitals where doctors or lab technicians can input commands by speech. In the banking sector, this technology may be used to develop machines which take speech input like taking input by just spelling out the account number. It may be used in aircraft systems, where pilots give audio commands to manage operations in the flight. In the new emerging field of robotics, this technology has mandatory requirement to give orders to robot by speech input.

Nowadays, everybody is using mobile phone as a very important gadget in their life. But there are some physically challenged people who are blind and the use of mobile phone is very difficult for them. So, there is an immense need of a system which works on speech as input and as well as output, *i.e.*, completely speech interactive system, to make them able to operate mobile phones by their own. In order to solve the problem statement, objectives have been framed for this thesis work. These have been discussed in next section.

3.1 Objectives

In this thesis, the objectives that have been framed for the development of the Speech-to-Text system for phonebook automation are as follows.

- To study the architecture of acoustic model based on ***HMM***.
- To train the acoustic model with domain specific vocabulary.
- To test the accuracy of Speech-to-Text system.

- To develop an interface for phonebook automation system.
- To integrate the phonebook automation system with Speech-to-Text system.
- To integrate the phonebook automation system with Text-to-Speech module.

3.2 Methodology

To achieve all the objectives discussed in above section, the following three methodologies have been adopted.

- Development of Speech-to-Text system.
- Development of phonebook application.
- Development of Text-to-Speech module.

3.2.1 Methodology for the development of Speech-to-Text system

In order to develop the Speech-to-Text system, the following tasks have been performed step by step.

- The ***Cygwin*** tool has been installed which provides an environment to execute other tools.
- ***HMM Tool Kit (HTK)*** has been integrated with ***Cygwin***. This tool has been used to train the system.
- ***Julius*** tool has been integrated with ***Cygwin***. This tool processes the speech input given by the user from microphone.
- ***Perl*** has been integrated with ***Cygwin*** to execute the ***perl*** scripts.
- ***Audacity*** tool has been installed for recording the speech.
- The set of rules in the form of grammar of the system has been defined.
- Data preparation has been done for transforming it to usable form for training.
- Training with monophones has been performed.
- Training with triphones has been performed for better accuracy.
- System has been executed and tested for all possible inputs.
- Output of the system has been directed to a file.

3.2.2 Methodology for the development of phonebook application

In order to develop phonebook application, the following tasks have been performed step by step.

- The business logic for the application has been written in Java language using Netbeans IDE.
- Data has been saved to MySQL database.
- The application has been executed by giving input from the output file of Speech-to-Text system.
- The whole system has been tested for various speech inputs.

3.2.3 Methodology for the development of Text-to-Speech module

In order to develop the Text-to-Speech module, the following tasks have been performed step by step.

- The speech files have been recorded for each digit.
- The speech files have been recorded for confirmation questions.
- Java code has been written to play the speech files according to the input text.
- Testing has been done with complete system.

The above mentioned methodologies have been followed in the implementation of the Speech-to-Text system with phonebook automation to achieve the objectives defined in thesis. The implementation of the proposed system has been described in the next chapter.

Chapter 4
Implementation of Speech-to-Text System for Phonebook Automation

In this chapter, the implementation of Speech-to-Text system with phonebook automation has been discussed. This chapter has been divided into two parts. The first part is the architecture of Speech-to-Text system. The second part is the phonebook automation. The architecture of the Speech-to-Text system includes the requirements for the development of the system. It also includes the installation process of the tools. It includes the different phases for the development of Speech-to-Text system and the execution of the system. The phonebook automation part includes the design of the phonebook application. This chapter also includes a Text-to-Speech module which has been integrated with the system to make the system completely speech interactive.

4.1 Process for automation of phonebook using Speech-to-Text system

The Speech-to-Text system for phonebook automation consists of two parts as follows:

- **Speech-to-Text system**

The Speech-to-Text system converts the speech input into its corresponding text. This ***HMM*** based system passes its output text to the phonebook application part for further processing.

- **Phonebook automation**

The output of the Speech-to-Text system works as input for the phonebook automation part. This part performs action according to the input text and updates the database connected to it. This part also provides a user interface so that user can easily interact with the system.

The basic process of Speech-to-Text system with phonebook automation has been shown in figure 4.1. The process shows that how the two parts have been integrated together to make a complete working system.

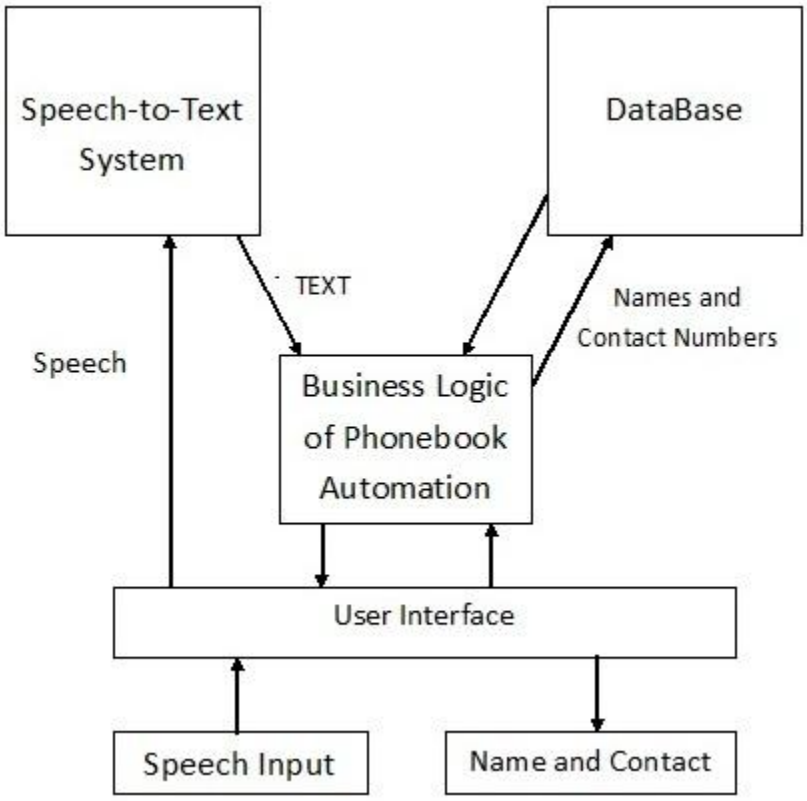

Figure 4.1: Speech-to-Text system with phonebook automation

4.2 Requirements for implementation of the Speech-to-Text system

There are some pre requirements for developing the Speech-to-Text system. These have been given below.

4.2.1 Hardware requirements

To develop a Speech-to-Text system, the basic devices which are required, are as follows.

- **Good quality microphone**

Microphone has been used for recording the speech files. It should be noise free so that speech files get recorded clearly. A desktop microphone doesn't work for Speech-to-Text

system as it captures a lot of noise with recording sound. A headset microphone serves the purpose in the best way [26].

- **Sound card**

As speech requires relatively low bandwidth, so a medium-high quality 16-bit sound card is sufficient for the Speech-to-Text system. Sound card must be enabled in the kernel and drivers must have correctly installed. Sound cards with the cleanest analog to digital conversions should be used.

4.2.2 Software requirements

The software tools which have been used for developing the Speech-to-Text system are as follows:

- ***Cygwin***

Cygwin tool integrates the other tools which have been used to develop the Speech-to-Text system. This tool provides an environment to execute other software tools like ***HTK*** and ***Julius***. This tool also provides the interface for the training of the Speech-to-Text system. It is a linux-like environment for Windows. It contains the bash shell scripting language and the ***perl*** scripting language. Both are required to run the scripts while development. It can be downloaded from *www.cygwin.com*.

- ***Hidden Markov Model Tool Kit (HTK)***

This tool provides all the commands of Hidden Markov Model to train the system. These commands include converting the speech files into mathematical form and re-estimating the statistical values. This tool kit is freely available on *www.*htk.eng.cam.ac.uk.

- ***Julius***

Julius is a speech recognition engine. It uses acoustic models in ***HTK*** format and grammar files in its own format. It is also used in dictation applications. ***Julian*** is a special version of ***Julius*** that performs grammar based speech recognition. ***Julian*** has been used in the development of our system. It can be downloaded from *http://julius.sourceforge.jp/en_index.php?q=index-en.html#download_julius.*

.

- ***Perl***

Perl is an interpreter which interprets perl scripts. A number of ***perl*** scripts have been used in this system. So, this tool is required for the execution of ***perl*** scripts.

- ***Audacity***

This tool has been used to record the speech files during the development of the system. This tool gives out the speech files in ***.wav*** format. Attributes of the ***.wav*** file can be set before recording like sample rate, channel, bit rate *etc.*

4.3 Installation of the tools for Speech-to-Text system

The tools that are required to install are ***Cygwin, HTK, Julius, Audacity*** and ***Perl.*** In order to do that ***Cygwin*** has been installed first. After performing all the steps to install ***Cygwin***, ***HTK*** has been installed. In the ***HTK*** package, it provides three zipped files. These are ***htk-3.2.1-windows-binary.zip***, ***HTK-samples-3.2.1.zip*** and ***htkbook_html.tar.gz***. The ***htk-3.2.1-windows-binary.zip*** file provides all the executables files to be used as commands in the development of Speech-to-Text system. The ***HTK-samples-3.2.1.zip*** file provides the ***perl*** scripts to be executed to manipulate files during development. The ***htkbook_html.tar.gz*** file provides a manual for using ***HTK*** commands. It is important to unzip all files in *HTK* directory under *Cygwin* directory.

Then ***Julius*** has to be installed. In order to install ***Julius***, ***julius-3.5.2-multipath-win32bin.zip*** has been downloaded. It is again important to unzip this file in *Julius* directory under *Cygwin* directory.

In order to integrate ***HTK*** and ***Julius*** with ***Cygwin***, the variable given in (4.1) has to be added in ***bash.bashrc*** file in *etc* directory under *Cygwin* directory.

```
PATH=/Julius/bin:/HTK/htk-3.3-windows-binary/htk:$PATH
export PATH                                                    …(4.1)
```

In order to check the integration, the user can execute ***HCopy*** command and ***julian*** command in the ***Cygwin*** shell**.** If it displays its switch options, then integration is successful.

In this thesis, the installation and implementation of the Speech-to-Text system has been done by following the steps given on the website of an organization, voxforge [38].

4.4 Architecture of Speech-to-Text system

The conversion process of speech to text has been divided into four phases as shown in figure 4.2.

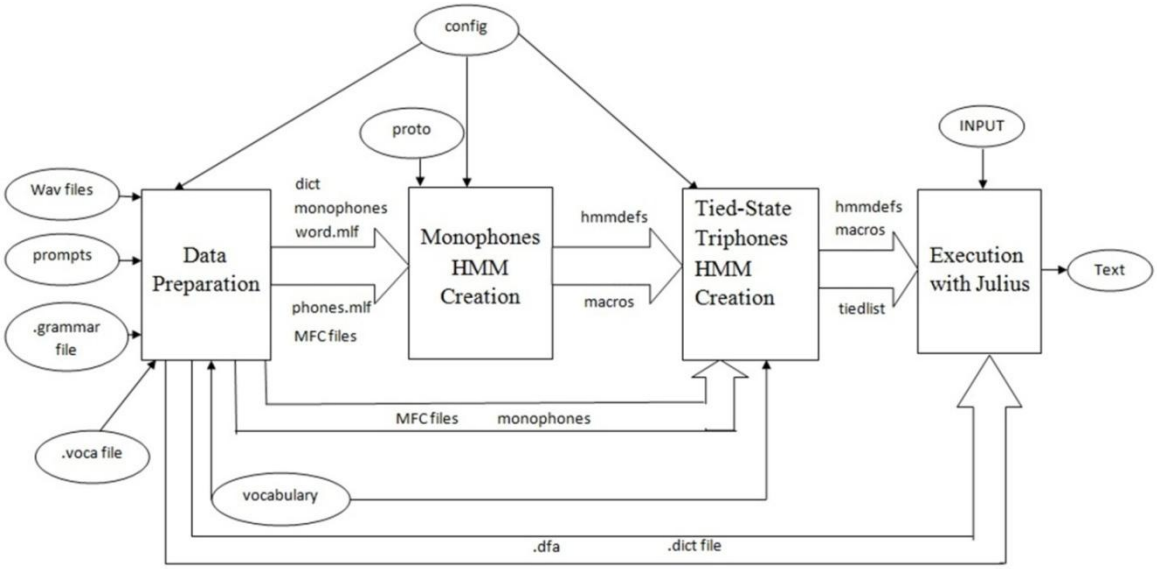

Figure 4.2: Architecture of Speech-to-Text system

These four phases of the architecture of Speech-to-Text system are as follows:

- Data preparation phase
- Creation of monophones ***HMM*** phase
- Creation of tied-state triphones ***HMM*** phase
- Execution with ***Julius*** phase.

The description of each of these phases has been discussed in subsequent sections.

4.4.1 Data preparation phase

This is the first phase in the development process of Speech-to-Text system. This phase has been used to prepare data to a form that can be recognized and processed by the

system. This phase requires grammar file, speech files, vocabulary file and training text file as raw input for processing. This phase has been divided into three sub-phases. These are as follows:

- Creation of grammar files
- Creation of master label file and dictionary file
- Creation of speech files and its conversion into mathematical form.

These sub-phases have been discussed below.

4.4.1.1 Creation of grammar files

In this sub-phase, the grammar of the language in the form of rules, is provided in ***.grammar*** file and words are provided in ***.voca*** file. The ***.grammar*** file is used to define the recognition rules. The ***.voca*** file is used to define the actual words in each word category and their pronunciation information. The description of ***.grammar*** file is given in (4.2).

```
S : NS_B SENT NS_E
SENT: DIGIT                                                    …(4.2)
```

The description of ***.voca*** file is given in (4.3).

```
% NS_B
<s>       sil
% NS_E
</s>      sil
% DIGIT
ONE       w ah n
TWO       t uw                                                 …(4.3)
```

As given in (4.2), ***S*** refers to start symbol of input, while ***NS_B*** indicates the beginning of silence and ***NS_E*** indicates end of silence by ***sil*** monophone. The data to be recognized is given by the keyword ***SENT*** which refers to ***DIGIT*** as given in (4.2). The details of

DIGIT has been provided in ***.voca*** file as given in (4.3). The ***DIGIT*** provides the recognition of words with their monophones combination. For example, ***ONE*** has monophones combination of **"w ah n".** The ***.grammar*** file and ***.voca*** file have been compiled to generate a dictionary file and finite automata file, namely ***.dict*** and ***.dfa*** file, respectively as shown in figure 4.3.

In our case, file name of these files are ***xyz.grammar*** and ***xyz.voca*** as it is mandatory to have same prefix of ***.grammar*** and ***.voca*** file. A perl script named ***mkdfa.pl*** has been used to create the **.dfa** and **.dict** file. This script has been used as given in (4.4).

```
$ mkdfa.pl xyz                                                    …(4.4)
```

The command given in (4.4) creates ***xyz.dfa*** and ***xyz.dict*** with a supplement file ***xyz.term*** which consists of all the terminal symbols. These files are required at the time of execution of the system.

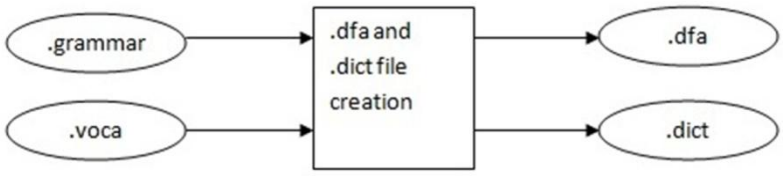

Figure 4.3: Process of creation of ***.dfa*** file and ***.dict*** file

4.4.1.2 Creation of master label file and dictionary file

In order to create master label file and dictionary file, some input files are required as shown in figure 4.4. These files have been given as follows.

- **Training text file**

Training text file is named as ***prompts***. It contains a list of words that are to be recorded and the names of their corresponding audio files that are to be stored. The description of this file is given in (4.5).

```
*/sample1 ONE TWO THREE FOUR FIVE SIX SEVEN EIGHT NINE ZERO
```

```
*/sample2 ZERO NINE EIGHT SEVEN SIX FIVE FOUR THREE TWO ONE

*/sample3 ONE ONE TWO TWO THREE THREE FOUR FOUR FIVE FIVE          …(4.5)
```

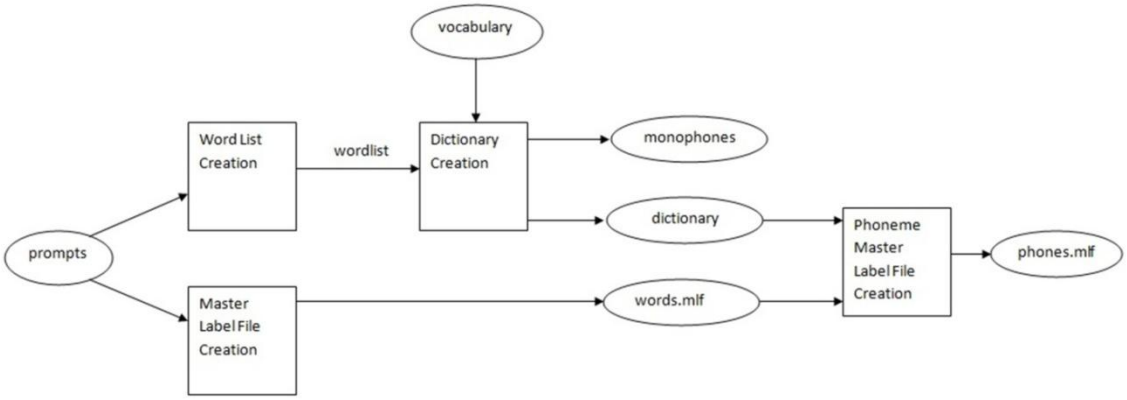

Figure 4.4: Master label file creation

In this case, an audio file has to be recorded with name ***sample1.wav***, which has content as follows: ONE TWO THREE FOUR FIVE SIX SEVEN EIGHT NINE ZERO.

- **Vocabulary file**

This file contains a sorted collection of commonly used words of a language along with their combination of monophones. This file is used as a reference to create a dictionary for the training words. A snapshot of this file, namely *lexicon*, is given in (4.6).

```
ABACK       [ABACK]      ax b ae k
ABACUS      [ABACUS]     ae b ax k ax s
ABALON      [ABALON]     ae b ax l aa n                    … (4.6)
```

It is a very large file containing approximately 24000 entries. Each entry contains a word and its monophone combination. All the words that are present in ***prompts*** file should be present in this vocabulary file. Otherwise an error would be shown while creating the dictionary file.

In order to create master label file and dictionary file, a ***wordlist*** file is required to be created. The ***prompts*** file has been used to create ***wordlist*** file. This file has been created by using command as shown in (4.7).

```
$ perl prompts2wlist prompts wordlist                    …(4.7)
```

The ***prompts2wlist*** file is a ***perl*** script which has been provided in ***htk-sample-3.2.1***. This script created a ***wordlist*** file. The ***wordlist*** file contains all the unique words among all the words in prompts file. The ***prompts2wlist*** script selected the distinct words and separated them in ***wordlist*** file.

The ***prompts*** file is used to create a master label file. A master label file (***.mlf***) contains same text as ***prompts*** file but every word occurs in a new line. It also divides the lines of ***prompts*** file into ***.lab*** headers in master label file. This has been done using a perl script as given in (4.8).

```
$ perl prompts2mlf prompts words.mlf                     …(4.8)
```

The ***words.mlf*** file looks like as given in (4.9).

```
#!MLF!#
"*/sample1.lab"
ONE
TWO
THREE
 ….
"*/sample2.lab"
ONE
THREE
FIVE
…                                                        …(4.9)
```

The ***wordlist*** file has been used with vocabulary file, namely ***lexicon***, to create ***monophones1*** file and ***dict*** file. This has been done by using command as given in (4.10).

```
$ HDMan -A -D -T 1 -m -w wordlist -n monophones1 -i -l dlog dict lexicon        …(4.10)
```

HDMan is an ***HTK*** command which has been used with ***A*** switch, ***D*** switch and ***T*** switch. The ***A*** switch is used to print command line arguments which has been set off by default. The ***D*** switch has been used to not to display configuration variables. The ***T*** switch has been used to set trace flags to 1, *i.e.,* its value written next to it. The ***w*** switch has been used to load ***wordlist*** file. The ***l*** switch has been used to create log of the command in a file with file name written next to it, *i.e,* ***dlog*** in this case.

The ***monophones1*** file contains all the unique monophones among all the monophones of each word in the ***wordlist*** file. The ***dict*** file is like a subset of ***lexicon*** file containing only those words that are present in ***wordlist*** file, *i.e.*, all the words with which training has to be done. The snapshot of ***monophones1*** file is as shown in (4.11).

```
aa
sh
iy
sp
...          `                                                    …(4.11)
```

The snapshot of ***dict*** file is as shown in (4.12).

```
ADD          [ADD]          ae d sp
DELETE       [DELETE]       d ih l iy t sp
...                                                              …(4.12)
```

A ***monphones0*** file is required to create here which contains the same content as ***monophones1*** file except that it should not have ***sp*** monophone, *i.e.,* short-pause

monophone. The ***monophones0*** and ***monophones1*** has been used in subsequent phases for training of the system.

As the ***words.mlf*** file has been created for training text words, there is a need to create a phoneme master label file. This file contains monophones of all the words in a new line of each word present in the ***words.mlf*** file.

As per the requirement of further phases, two phoneme master label files are required to create, one file without ***sp*** monophone and other file with ***sp*** monophone. These files are ***phones0.mlf*** and ***phones1.mlf*** respectively. In order to do this, their corresponding script files are needed. These script files are ***makephones0.led*** and ***makephones1.led*** as shown in (4.13).

makephones0.led

```
EX
IS sil sil
DE sp
```

makephones1.led

```
EX
IS sil sil
```

…(4.13)

In the above scripts, ***EX*** command expands the word read by it. ***IS*** command inserts the ***sil*** monphone at beginning and end of each line read. The ***sil*** monophone is silence monophone. ***DE*** command deletes ***sp*** monophones, i.e., short pauses in between the words.

Now, the ***phones0.mlf*** file has been created by the ***HTK*** command as given in (4.14).

```
$ HLEd -A -D -T 1 -l '*' -d dict -i phones0.mlf makephones0.led words.mlf
```

…(4.14)

The ***phones1.mlf*** file has been created by the same ***HTK*** command as given in (4.15).

```
$ HLEd -A -D -T 1 -l '*' -d dict -i phones1.mlf makephones1.led words.mlf
```

…(4.15)

The ***phones0.mlf*** file is shown as in (4.16).

```
#!MLF!#
"*/sample1.lab"
sil
w
ah
n
...                                                    …(4.16)
```

The ***phones1.mlf*** file is shown as in (4.17).

```
#!MLF!#
"*/sample1.lab"
sil
w
ah
n
sp
...                                                    …(4.17)
```

4.4.1.3 Creation of speech files and its conversion into mathematical form

Speech files have been recorded by ***audacity***. There are some attributes which have to be set before recording are as follows:

- **Sample Rate** : It is required to be set to 48000Hz.
- **Sample Format** : It is required to be set to 16-bit.
- **Channel** : It is required to set to Mono.
- **Uncompressed Export Format :** It is required to set as WAV (Microsoft 16 bit PCM).

Along with these attributes, it has to be checked that the microphone volume in ***audacity*** should be set to 1.0. The recorded waveform level should be in between 0.5 and -0.5. Average level may be in between 0.3 to -0.3.

The training text written in ***prompts*** file has been recorded and saved in these files. These files are to be stored in ***.wav*** format. The recorded files are to be placed in *wav* directory. The names of these files should be like ***sample1.wav, sample2.wav*** *etc.*

These speech files have been converted into mathematical form. The new mathematical file for each speech file has extension ***.mfc****, i.e.,* Mel Frequency Cepstral file. This file contains the mel frequency cepstral coefficients of the speech files.

The conversion of ***.wav*** file to ***.mfc*** file has been done by using a configuration file namely ***config***. The ***config*** file is as shown in (4.18).

```
SOURCEFORMAT = WAV
TARGETKIND = MFCC_0_D
TARGETRATE = 100000.0
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 26
CEPLIFTER = 22
NUMCEPS = 12                                                    …(4.18)
```

The ***config*** file sets the attributes of the ***.mfc*** to be created from ***.wav*** files. For example, TARGETKIND has been set to ***MFCC_0_D***. TARGETRATE has been set to ***100000.0*** and so on.

The command used to convert the ***.wav*** files to ***.mfc*** files is shown in (4.19).

```
$ HCopy -A -D -T 1 -C config -S codetrain.scp                   …(4.19)
```

The ***HCopy*** command has been used with ***config*** file and ***codetrain.scp*** file. The ***codetrain.scp*** file contains the source path of ***.wav*** files and destination path of ***.mfc*** files to locate these files. The ***codetrain.scp*** file is as shown in (4.20).

```
../train/wav/sample1.wav   ../train/mfcc/sample1.mfc
../train/wav/sample2.wav   ../train/mfcc/sample2.mfc
../train/wav/sample3.wav   ../train/mfcc/sample3.mfc
```
…(4.20)

This command has populated the ***mfcc*** directory with all ***.mfc*** files.

The data preparation phase gets complete here now. In data preparation phase, the following files have been created: ***dict*** file, ***monophones0*** file, ***monophones1*** file, ***phones0.mlf*** file, ***phones1.mlf*** file, ***words.mlf*** file, ***.mfc*** files, ***xyz.dict*** file, ***xyz.dfa*** file and ***xyz.term*** file.

4.4.2 Creation of monophone *HMM* phase

This phase has been used to create a well-trained set of monophones ***HMM*** files. In this phase, the training of the ***HMM*** files has been done by monophones. Initialy, flat values are given in the ***HMM*** files to initialize the training process. Then re-estimation is done for training. This phase requires a prototype for ***HMM*** file, ***.mfc*** files, ***config*** file, monophone files and phoneme master label file for creating the HMM files. Each ***HMM*** file follows the prototype given in ***proto*** file, *i.e.*, prototype file.

There are a number of monophones in ***HMM*** file. Generally, each monophone has five states. Here, state 1 and state 5 are opening and closing states, while state 2, 3 and 4 has values for means and variances for its corresponding monophone.

This phase has been divided into three sub-phases as shown in figure 4.5 and given as follows:

- Creation of flat start monophones ***HMM*** files and its re-estimation
- Fixing the silence models
- Realigning the training data

These three sub-phases have been discussed below.

4.4.2.1 Creation of flat start monophones *HMM* files and its re-estimation

In this sub-phase, ***HMM*** files have been created at different training levels. An ***HMM*** file contains some parameters for each monophone. There are 5 states for each monophone. State 1, state 2, state 3, state 4 and state 5 are the five states. Among these states, state 1 and state 5 are opening and closing states for that monophone. So, state 1 and state 5 don't have any parameters. Each monophone has some parameters for other three states. These parameters are mean and variances of the waveform of each monophone for each state. Along with it, a global constant value is associated with these three states. Also a 5x5 is associated with each monophone which shows the transition in between the states.

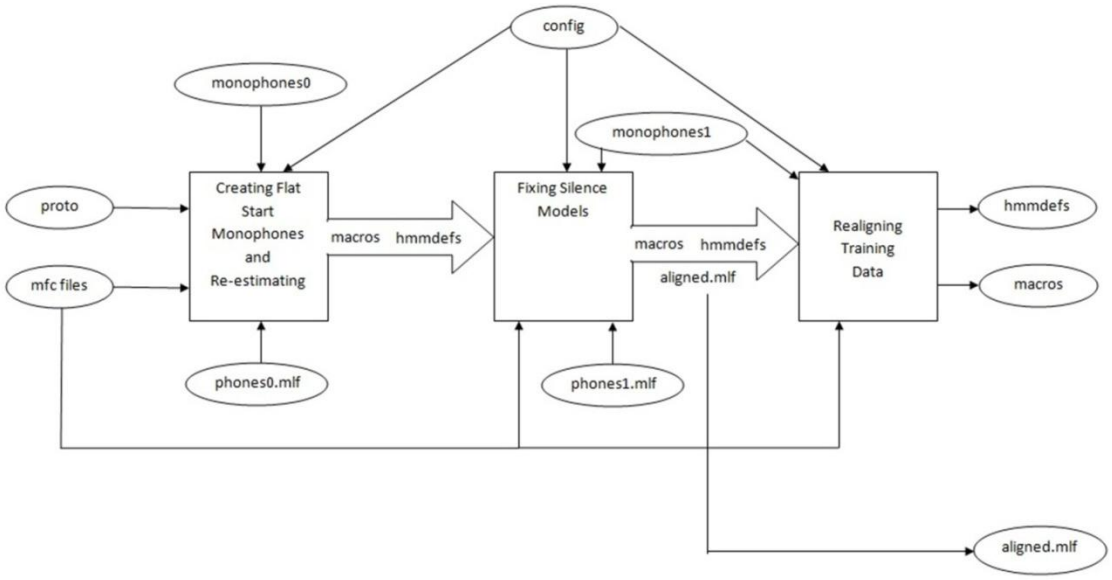

Figure 4.5: Monophone ***HMM*** creation and training

A prototype of a ***HMM*** file has been given in ***proto*** file. The ***proto*** file is as shown in (4.21).

```
~o <VecSize> 25 <MFCC_0_D_N_Z>
~h "proto"
<BeginHMM>
 <NumStates> 5
 <State> 2
```

```
  <Mean> 25
   0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
  <Variance> 25
   1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 3
  <Mean> 25
   0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
  <Variance> 25
   1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 4
  <Mean> 25
   0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
  <Variance> 25
   1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<TransP> 5
 0.0 1.0 0.0 0.0 0.0
 0.0 0.6 0.4 0.0 0.0
 0.0 0.0 0.6 0.4 0.0
 0.0 0.0 0.0 0.7 0.3
 0.0 0.0 0.0 0.0 0.0
<EndHMM>                                                                    …(4.21)
```

As given in (4.21), proto is a monophone. It has three states with their mean and variance parameters. It is only a prototype, so the parameters have been taken as dummy.

As the name of this sub-phase suggests, ***Flat Start HMM***, the first ***HMM*** file should have the same constant parameters for each monophone, just to start the training process. At each step of training, these parameters would be changed by ***HTK*** commands. For calculating these initial constant parameters, an ***HTK*** command has been used. This is given as in (4.22).

```
$ HCompV -A -D -T 1 -C config -f 0.01 -m -S train.scp -M hmm0 proto          …(4.22)
```

This command given in (4.22) has been used with ***config*** file, *i.e.*, configuration file. The ***proto*** file has been used for referencing a prototype. The ***train.scp*** file provides the path of all the ***.mfc*** files. The snapshot of ***train.scp*** file is as shown in (4.23).

```
../train/mfcc/sample1.mfc
../train/mfcc/sample2.mfc
../train/mfcc/sample3.mfc                                    …(4.23)
```

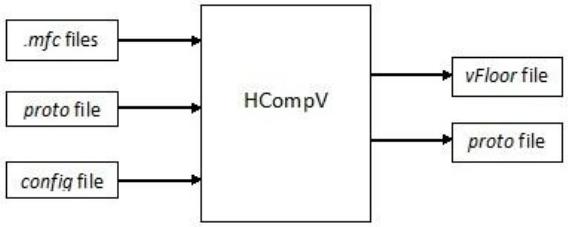

Figure 4.6: Creation of initial constant parameters

The output of the above command has given two files, ***vFloors*** and ***proto*** in *hmm0* directory as shown in figure 4.6. The ***vFloors*** file gives an initial constant parameter. This initial constant parameter has to be used for each monophone in ***monophones0***. In order to do this, a new file has been created with name ***hmmdefs***. The ***hmmdefs*** file contains all the monophones of ***monophones0*** file with their initial constant parameter as given in ***vFloors*** file. These constant parameters have been same for all monophones. This justifies the flat start ***HMM*** in the name of the sub-phase. A new file with name ***macros*** has also been created. This file contains 25 variance values as given by newly created ***proto*** file. The newly created ***hmmdefs*** and ***macros*** files act as first step HMMs trained files. On the basis of these files, further training of the system has been done.

The training of the system has to be done after the initialization of ***HMM*** files with constant parameters. In order to do this, the ***HERest*** command has been used. The ***HERest*** stands for re-estimation. So, re-estimation of the ***HMM*** files has been done by executing the command as given in (4.24).

```
$ HERest -A -D -T 1 -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm0/macros
-H hmm0/hmmdefs -M hmm1 monophones0                          …(4.24)
```

The command given in (4.24) used previous ***HMM*** files for generating new trained ***HMM*** files, *i.e.*, ***hmmdefs*** and ***macros*** files. The other files used for executing the command are configuration file, *i.e.*, ***config***, phoneme master label file, *i.e.*, ***phones0.mlf***, ***.mfc*** files and ***monophones0*** file. After first training of the system, hmm1 ***HMM*** files have been created.

This re-estimation process has been done for two more times to train the system for two more training levels. The further re-estimation of the system has been done by commands as given in (4.25).

```
$ HERest -A -D -T 1 -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm1/macros -
H hmm1/hmmdefs -M hmm2 monophones0

$ HERest -A -D -T 1 -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S train.scp -H hmm2/macros -
H hmm2/hmmdefs -M hmm3 monophones0                                                    …(4.25)
```

The re-estimation process has created ***HMM*** files up to third level. In second re-estimation, all the input parameters have been same except the ***HMM*** files. The output of first level re-estimation, *i.e.*, ***HMM*** files have been used as input in second time. The output of the second level re-estimation, *i.e.*, ***HMM*** files have been used as input in third time. So a cycle has been created in this process. This has been shown in figure 4.7.

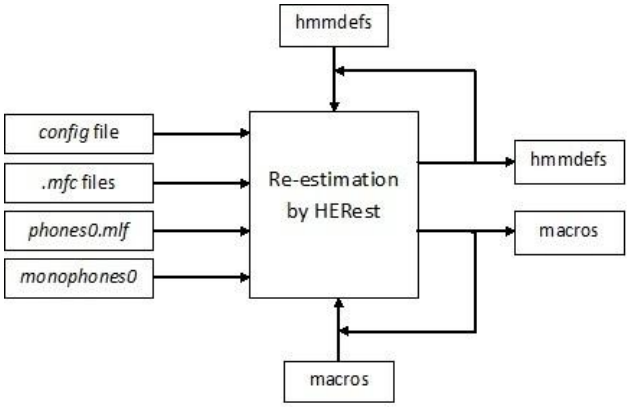

Figure 4.7: Re-estimation of the ***HMM*** files without ***sp*** monophone

4.4.2.2 Fixing the silence models

At time of recording, there were short pauses between the words spoken, to absorb these short pauses between words, the system is required to get trained for these short pauses. This has been done by adding the ***sp*** monophone in the ***HMM*** file last created.

Each monophone presented in the ***hmmdefs*** file has 5 states but the ***sp*** monophone has 3 states. State 1 and state 3 are opening and closing states. So these states don't have any parameters. State 2 should have parameters. In order to add ***sp*** monophone, a temporary copy of ***sil*** model has been created and saved with name ***sp***. The State 2 and state 4 have been removed from ***sp*** model and only state 3 has been kept in ***sp*** model with state 2 as its name. The ***HMM*** files for the next training level, *i.e.,* 4th level, have been formed by above mentioned procedure. The ***HMM*** files now have the ***sp*** monophone. At this time, there is a need to tie up these middle states, *i.e.*, state 3 of ***sil*** monophone and state 2 of ***sp*** monophone. The tie up means that the states would share the same set of parameters. The tie up has been completed by executing the command as given in (4.26).

```
$HHEd -A -D -T 1 -H hmm4/macros -H hmm4/hmmdefs -M hmm5 sil.hed monophones1 …(4.26)
```

The command given in (4.26) used previously created ***HMM*** files, *i.e.,* 4th level files, to generate new ***HMM*** files for 5th level of training as shown in figure 4.8

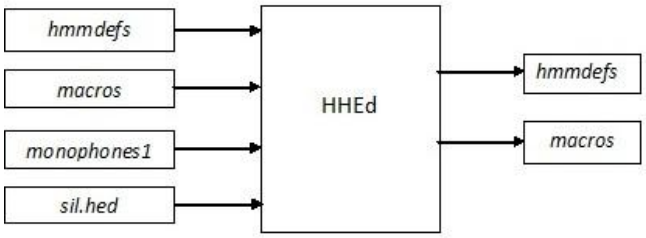

Figure 4.8 Tie-up of middle states of ***sil*** monophone and ***sp*** monophone

The ***monophones1*** file has been used instead of ***monophones0*** file as ***monophones0*** file doesn't contain the ***sp*** monophone. A script ***sil.hed*** has been used to tie up the states. This script is as shown in (4.27).

```
AT 2 4 0.2 {sil.transP}
AT 4 2 0.2 {sil.transP}
AT 1 3 0.3 {sp.transP}
TI silst {sil.state[3],sp.state[2]}
```
…(4.27)

The last line of the script given in (4.27) has tied up the state 3 of ***sil*** monophone and state 2 of ***sp*** monophone. The ***HMM*** files need to be re-estimated to generate next level ***HMM*** files, *i.e.*, 6th level. So, the ***HERest*** command has been used to re-estimate the ***HMM*** files. The commands executed are given in (4.28).

```
$ HERest -A -D -T 1 -C config  -I phones1.mlf -t 250.0 150.0 3000.0 -S train.scp -H hmm5/macros -
H  hmm5/hmmdefs -M hmm6 monophones1

$ HERest -A -D -T 1 -C config  -I phones1.mlf -t 250.0 150.0 3000.0 -S train.scp -H hmm6/macros -
H  hmm6/hmmdefs -M hmm7 monophones1
```
…(4.28)

The commands given in (4.28) have created 6th and 7th level ***HMM*** files. The ***phones1.mlf*** file has been used as phoneme master label file. The ***monophones1*** file has been used because of ***sp*** monophone. The previous ***HMM*** files have been used to create new ***HMM*** files at next level. The ***config*** file and ***.mfc*** files have also been used as input parameters as shown in figure 4.9.

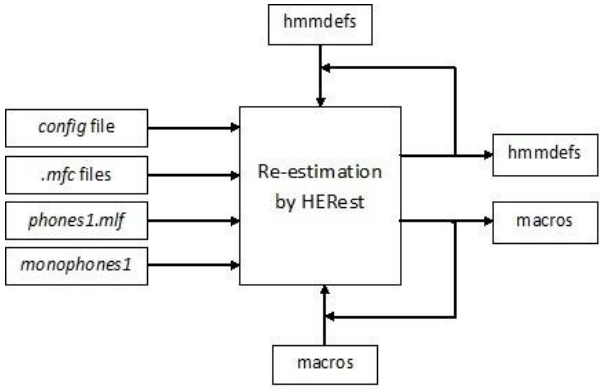

Figure 4.9: Re-estimation of the ***HMM*** files with ***sp*** monophone

This sub-phase has trained the system up to 7th level of ***HMM*** training. This sub-phase has made the Speech-to-Text system more robust to absorb various impulsive noises in the training data.

4.4.2.3 Realigning the training data

As the system has been trained by the training text, it might be a possibility that there may be multiple pronunciations of a text in a word. These pronunciations have been saved as monophones in the ***phones0.mlf*** file and ***phones1.mlf*** file. The process of selecting the best pronunciation of the text among multiple pronunciations is called as **realignment** of training data. A new phoneme master label file is need to be created for realignment of the training data. So, the ***HTK*** command used for realignment is as given in (4.29).

```
$ HVite -A -D -T 1 -l '*' -o SWT -b SENT-END -C config -H hmm7/macros -H hmm7/hmmdefs -i aligned.mlf -m -t 250.0 150.0 1000.0 -y lab -a -I words.mlf -S train.scp dict monophones1> HVite_log
```
…(4.29)

As given in 4.29, the command has created a phoneme master label file named as ***aligned.mlf***. This file has the best selected monophones corresponding to the training text. The configuration file, *i.e.*, ***config***, has been used as input parameter in this command. The word master label file, *i.e.*, ***words.mlf***, has been used as a reference to create new phoneme master label file. The previously generated ***HMM*** files, *i.e.,* 7th level ***HMM*** files, have also been used as input parameters as shown in figure 4.10.

The dictionary file, *i.e.*, ***dict***, and ***monophones1*** has been used to refer monophones. The ***.mfc*** files have also been used as input parameters. The log of the execution of the command has been saved in a log file, namely ***HVite_log***.

As the system has got a new phoneme master label file, so it has to be retrained by using re-estimation. The re-estimation has been done two times.

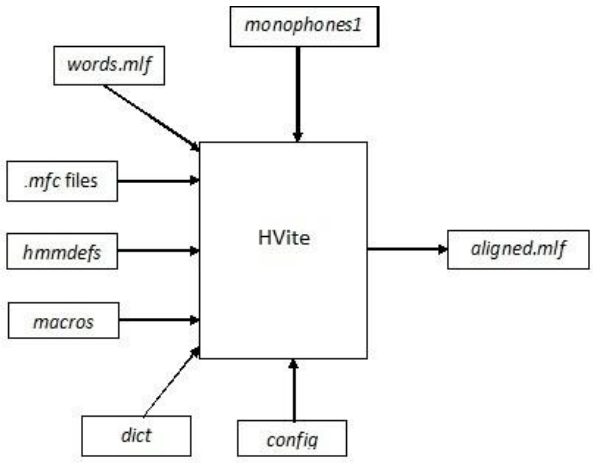

Figure 4.10: Realignment of training data

The commands for re-estimation are as given in (4.30).

```
$ HERest -A -D -T 1 -C config -I aligned.mlf -t 250.0 150.0 3000.0 -S train.scp -H hmm7/macros -H
hmm7/hmmdefs -M hmm8 monophones1

$ HERest -A -D -T 1 -C config -I aligned.mlf -t 250.0 150.0 3000.0 -S train.scp -H hmm7/macros -H
hmm7/hmmdefs -M hmm8 monophones1                                                    …(4.30)
```

As given in 4.30, in the first command, the previously generated ***HMM*** files, *i.e.,* 7th level ***HMM*** files, have been used to generate next level, *i.e.,* 8th level, ***HMM*** files. In the second command, 8th level ***HMM*** files, *i.e.,* the output of the first command, have been used to generate next level, *i.e.,* 9th level, ***HMM*** files. The ***aligned.mlf*** has been used as phoneme master label file. The ***.mfc*** files and ***monophones1*** file have also been used as input parameters for this command as shown in figure 4.11.

Here, the second phase of the development for Speech-to-Text system, *i.e.,* Monophone ***HMM*** creation phase, gets completed. In this phase, the following files have been created:

- ***hmmdefs*** of 9th level
- ***macros*** of 9th level

- ***aligned.mlf***

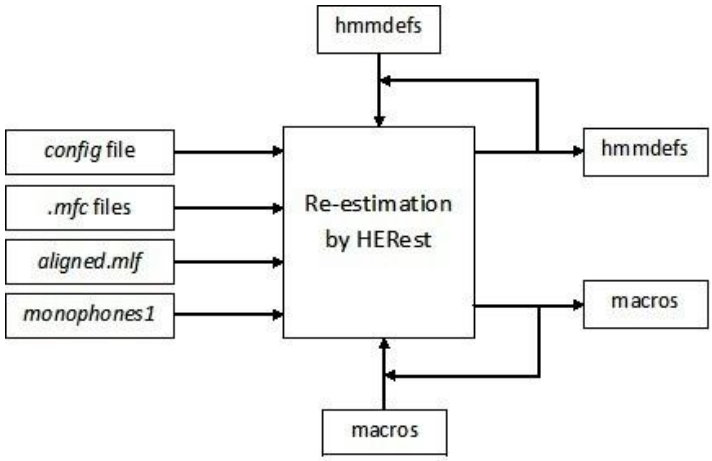

Figure 4.11: Re-estimation with realigned phoneme master label file

4.4.3 Creation of tied-state triphones *HMM*

The system has been trained by using monophones till now. The accuracy of the system that has been trained by monophones is not so good because as a word is split in to a number of monophones, then these monophones are recognized individually. If the speech recognition engine tries to recognize a sequence of monophones, its accuracy gets increased. For example, if a keyword is searched on any search engine or if a combination of three keywords is searched on that search engine, the result of latter case would be more accurate. The result would be more precise towards the goal. In order to train the system with help of a sequence of monophones, instead of individual monophone, then training with triphone is required.

In this phase, the training of the system has been done with triphones and tied-state triphones. This phase has two sub-phases. These are as follows:

- Creation of triphones and training
- Creation of tied-state triphones and training

These sub-phases have been discussed one by one. The process of this phase has been shown in figure 4.12

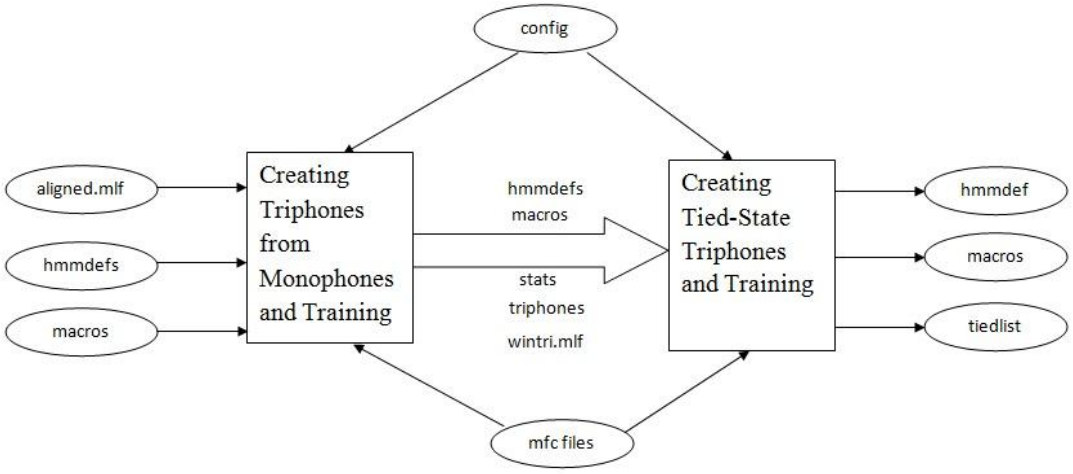

Figure 4.12: Creation of triphones and tied-state triphones and their training

4.4.3.1 Creation of triphones and their training

A triphone is a sequence of two or three monophones. It can be denoted by ***L-X+R***, where ***L***, ***X*** and ***R*** are three monophones. ***L*** denotes that it is a left side monophone and ***R*** denotes that it is a right side monophone. ***L*** monophone precedes ***X*** monophone and ***R*** monophone follows ***X*** monophone. For example, triphones of a word, ***DELETE***, are as shown in (4.31).

```
d+ih
d-ih+l
ih-l+iy
l-iy+t
iy-t                                                    …(4.31)
```

The system has been trained by single phoneme master label file and monophones file till now. But in order to train the system with triphones, a triphone master label file and triphones file is need to be created. So, an ***HTK*** command has been used to create triphone master label file, *i.e.*, ***wintri.mlf*** and a triphone file, *i.e.*, ***triphones1***. The ***HTK*** command for this is as given in (4.32).

```
$ HLEd -A -D -T 1 -n triphones1 -l '*' -i wintri.mlf mktri.led aligned.mlf    …(4.32)
```

As given in (4.32), the command has created two files. First file is ***triphones1*** and second file is ***wintri.mlf*** as shown in figure 4.13. The ***aligned.mlf*** has been used as phoneme master label file. A script file has been used with this command, *i.e.*, ***mktri.led***. The ***mktri.led*** is as shown in (4.33).

```
WB sp
WB sil
TC                                                    …(4.33)
```

In this script as given in (4.33), ***TC*** command has been used to expand each monophone that has been given in ***aligned.mlf***. The expansion has been done to convert monophones into triphones. The ***WB*** command has been used to instruct the ***HLEd*** command to not to process ***sp*** and ***sil*** monophones. All the monophones except ***sp*** and ***sil*** have been converted into triphones.

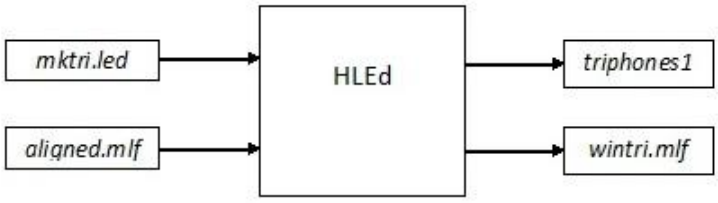

Figure 4.13: Creation of triphones

As the triphones have been created above, so, an initial set of triphones model is need to be created. A perl script has been used to create triphone models. The script reads each monophone from ***monophones1*** file and forms its triphone model. This has been done by executing the command as given in (4.34).

```
$ perl maketrihed monophones1 triphones1              …(4.34)
```

The command given in (4.34) creates a triphone model file, *i.e.*, ***mktri.hed***. This file contains triphone models of monophones like *aa, sh* and *iy* monophones as shown in (4.35).

```
TI T_aa {(*-aa+*,aa+*,*-aa).transP}
TI T_sh {(*-sh+*,sh+*,*-sh).transP}
TI T_iy {(*-iy+*,iy+*,*-iy).transP}                                    …(4.35)
```

In the script given in (4.35), the ***TI*** command has been used to tie-up the items of a model. For example, for ***aa*** monophone, the items are ****-aa+****, ***aa+**** and ****-aa***. The tie-up has been done by an ***HTK*** command as given in (4.36).

```
$ HHEd -A -D -T 1 -H hmm9/macros -H hmm9/hmmdefs -M hmm10
mktri.hed monophones1                                                  …(4.36)
```

The command given in (4.36) has created new ***HMM*** files for 10th level as shown in figure 4.14. The ***monophones1*** file has been used to provide list of monophones. The ***mktri.hed*** script has commands that have to be executed by ***HHEd*** command.

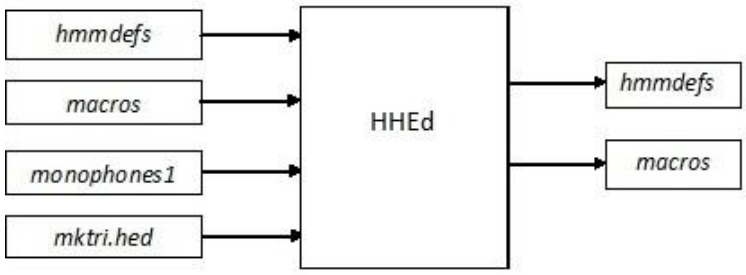

Figure 4.14: Creation on 10th level ***HMM*** files

As the system has been updated with ***triphones1*** and ***wintri.mlf***, it is required to re-estimate the ***HMM*** files by using newly created files. So, the ***HTK*** commands used to re-estimate the ***HMM*** files are as given in (4.37).

```
$ HERest  -A -D -T 1 -C config -I wintri.mlf -t 250.0 150.0 3000.0 -S train.scp -H hmm10/macros -H
hmm10/hmmdefs -M hmm11 triphones1

$ HERest  -A -D -T 1 -C config -I wintri.mlf -t 250.0 150.0 3000.0 -s stats -S train.scp -H
hmm11/macros -H hmm11/hmmdefs -M hmm12 triphones1                                    …(4.37)
```

The commands given in (4.37) have created next level ***HMM*** files, *i.e.,* 11th and 12th level. The 11th level ***HMM*** files have been created by using 10th level ***HMM*** files. The 12th level ***HMM*** files have been created by using 11th level ***HMM*** files. In the second command, a ***stats*** file has also been generated as shown in figure 4.15. This ***stats*** file is needed in the next sub-phase. This file contains some statistical information about the triphones.

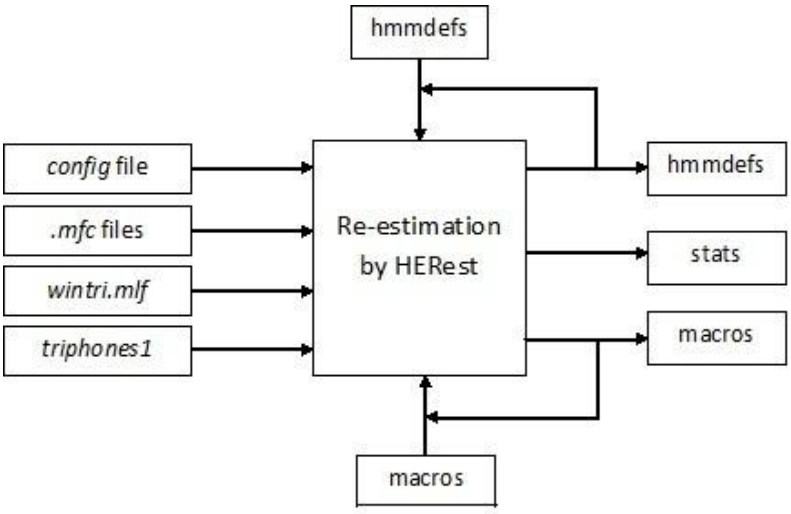

Figure 4.15: Re-estimation using triphones

4.4.3.2 Creation of tied-state triphones and their training

In this sub-phase, tied-state triphones have been created. In order to do this, a full vocabulary dictionary has to be created. Along with it, a full list of monophones and triphones has also to be created. These two files have been created by ***HTK*** command as shown in (4.38).

```
$ HDMan -A -D -T 1 -b sp -n fulllist -g global.ded -l flog dict-tri lexicon          …(4.38)
```

As given in (4.38), the above command is using ***lexicon*** file, *i.e.*, vocabulary file, as an input parameter to create dictionary file, *i.e.*, ***dict-tri*** as shown in figure 4.16. The ***–b*** switch has been used to set ***sp*** as boundary symbol. The ***global.ded*** is a script file as shown in (4.39).

```
AS sp
RS cmu
MP sil sil sp                                                    …(4.39)
```

As shown in (4.39), the ***AS*** command in the script appends silence model ***sp*** to each pronunciation in the newly created ***dict-tri*** file. The ***RS*** command removes stress marking. Currently the only stress marking system supported is that used in the dictionaries produced by Carnegie Melon University (CMU). The ***MP*** command merges any sequence of phones. In this case ***sil*** and ***sp*** monophones have been merged and renamed as ***sil.***

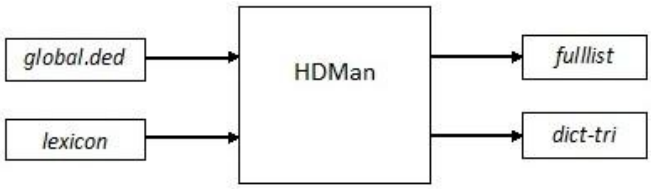

Figure 4.16: Creation of ***fulllist*** and ***dict-tri*** files

The monophones and triphones in files ***fulllist*** and ***triphones1*** have been collected in one file, ***fulllist1***. The ***fulllist1*** file might have duplicate entries. To remove these duplicate entries, a perl script has been used, namely ***fixfulllist.pl***. In this file, simple perl code has been written to remove duplicate entries. The unique entries have been overwritten in ***fulllist*** file.

A script ***tree.hed*** has been used to create tied state triphones. The ***tree.hed*** script contains commands to create tied state triphones and also contains reference to the ***fulllist*** file. An ***HTK*** command has been used to execute the ***tree.hed*** script. This command creates next

level, *i.e.*, 13th level, ***HMM*** files and a ***tiedlist*** file as shown in figure 4.17. The ***tiedlist*** file contains tied state triphones. The ***HTK*** command executed is as shown in (4.40).

```
$ HHEd -A -D -T 1 -H hmm12/macros -H hmm12/hmmdefs -M
hmm13 tree.hed triphones1                                    …(4.40)
```

The commands given in (4.40) used previous level, *i.e.*, 12th level, ***HMM*** files to generate new ***HMM*** files. The ***triphones1*** file has also been used to provide all the *triphones*.

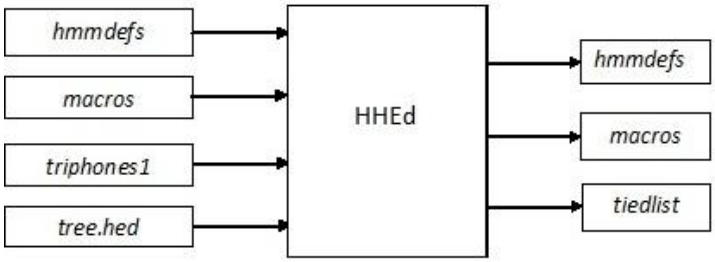

Figure 4.17: Tied-state triphones creation

As the system has been updated by new tied state triphones file, *i.e.*, ***tiedlist*** file, re-estimation of the system is required. So, the ***HTK*** command has been used to re-estimate the system. The commands executed to re-estimate are as given in (4.41).

```
$ HERest -A -D -T 1 -T 1 -C config -I wintri.mlf -s stats -t 250.0 150.0 3000.0 -S train.scp -H
hmm13/macros -H hmm13/hmmdefs -M hmm14 tiedlist

$ HERest -A -D -T 1 -T 1 -C config -I wintri.mlf -s stats -t 250.0 150.0 3000.0 -S train.scp -H
hmm14/macros -H hmm14/hmmdefs -M hmm15 tiedlist              …(4.41)
```

The ***HERest*** command has been executed with configuration file, *i.e.*, ***config*** file, the triphone master label file, *i.e.*, ***wintri.mlf*** file, ***stats*** file that has been created while creation of 12th level ***HMM*** files, ***.mfc*** files and ***tiedlist*** file as shown in figure 4.18. The first command has used 13th level ***HMM*** files to create 14th level ***HMM*** file. The second

command has used 14th level ***HMM*** files to create 15th level, *i.e., the last level,* ***HMM*** files.

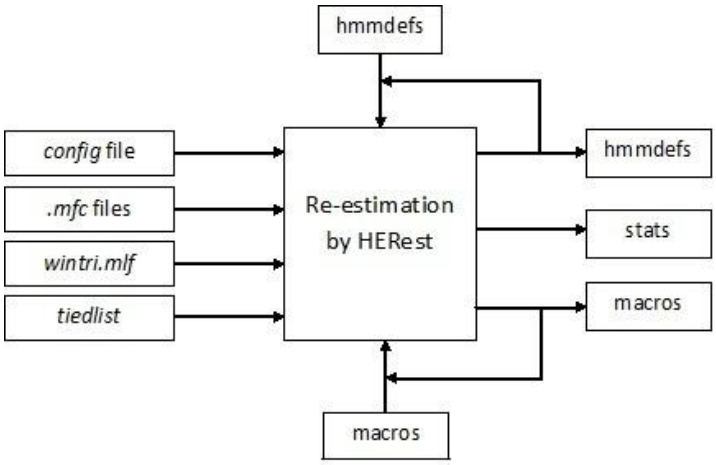

Figure 4.18: Re-estimation with tied-state triphones

This phase gets completed here after creating final ***HMM*** files that are to be executed in the next phase.

4.4.4 Execution with *Julius*

Julius has been used to execute the developed Speech-to-Text system. ***Julius*** requires four files for execution. These are as follows:

- ***xyz.dfa*** file
- **xyz*.dict*** file
- last generated ***HMM*** file, *i.e.,* ***hmmdefs*** of 15th level
- ***tiedlist*** file

The first two files, *i.e.,* ***xyz.dfa*** file and **xyz*.dict*** file, have already been created in first phase and ***HMM*** file and ***tiedlist*** file have been created in the third phase. In order to execute the system, these files are passed as parameters in ***Julius*** configuration file, *i.e.,* ***julian.conf*** as shown in (4.42).

```
-hlist        tiedlist
-h            hmm15/hmmdefs
-dfa          xyz.dfa
-v            xyz.dict
```

```
-smpFreq  48000                                                    ...(4.42)
```

The ***julian*** command has been used to execute the system. It requires ***julian.conf*** and ***mic*** as input parameters. The command executed is as shown in (4.43).

```
$ julian -input mic -C julian.jconf                                ...(4.43)
```

After execution of this command, the system prompts the user to speak the sentence as shown in figure 4.19.

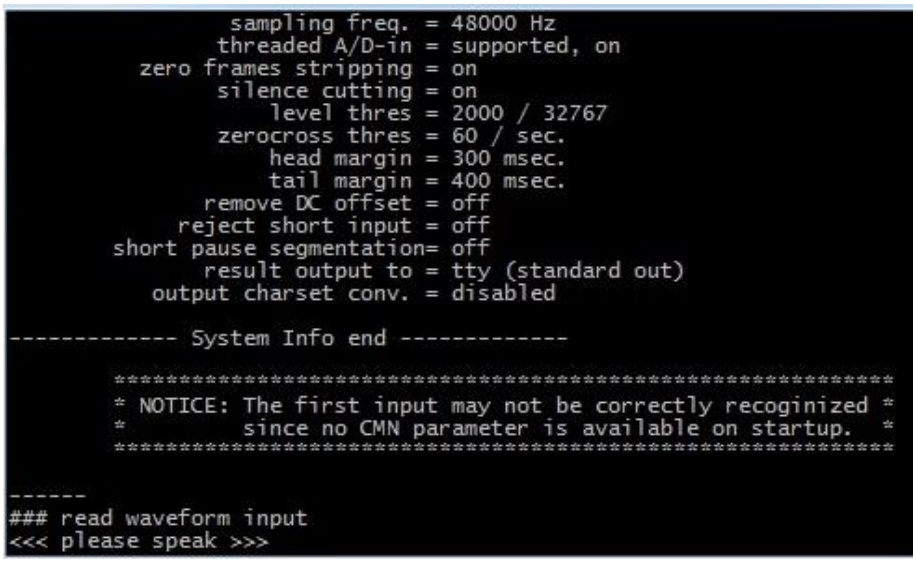

Figure 4.19: Execution of the system

The user speaks some text, for example "zero", the executing system converts this speech into text and prompts in the window. The recognized text has been shown in figure 4.20.

In this manner, the Speech-to-Text system has been developed. In the next section the development of phonebook application has been discussed.

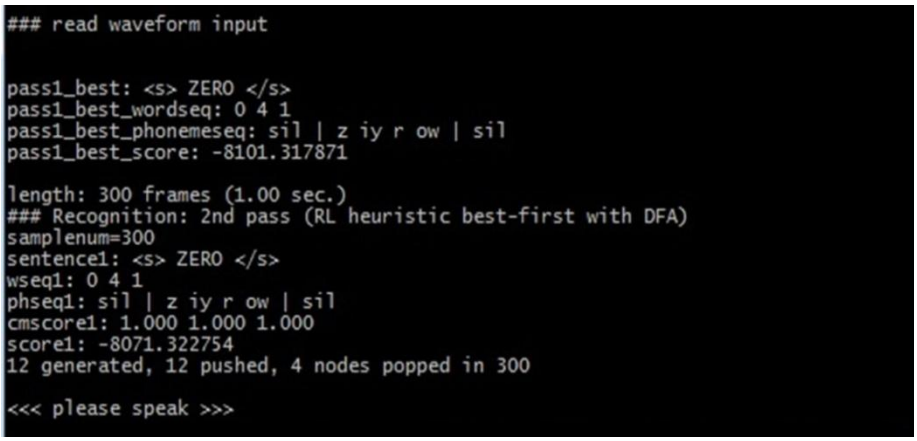

Figure 4.20: Speech-to-Text conversion of spoken word “zero”

4.5 Phonebook automation

The phonebook automation application has been developed in Java language using swing package. This application takes input from the output of the Speech-to-Text system, *i.e.*, text. The input text is converted to an action to be performed which makes the changes.

The Speech-to-Text system developed in previous section has been trained according to the requirement of this application. The system recognizes text like ***ADD, UPDATE***, ***DELETE*** and ***CALL***. It also recognizes numbers as ***ONE***, ***TWO***, ***THREE***, ***FOUR***, ***FIVE***, ***SIX***, ***SEVEN***, ***EIGHT***, ***NINE***, ***ZERO***. The system recognizes some replies like ***YES*** and ***NO***.

As the system has been trained by the spellings of the numbers, like ***ONE***, ***TWO*** *etc.*, these spellings have been mapped to numeric digits. The ***ONE*** has been mapped to ***1***. The ***TWO*** has been mapped to ***2*** and so on. The basic flow of the data has been shown in figure 4.21.

In Speech-to-Text system, the output is directed to a file named ***test***. The output file contains a lot of information as output of the Speech-to-Text system. The text corresponding to speech is retrieved from this file. The output text is matched with the predefined string and corresponding action is performed.

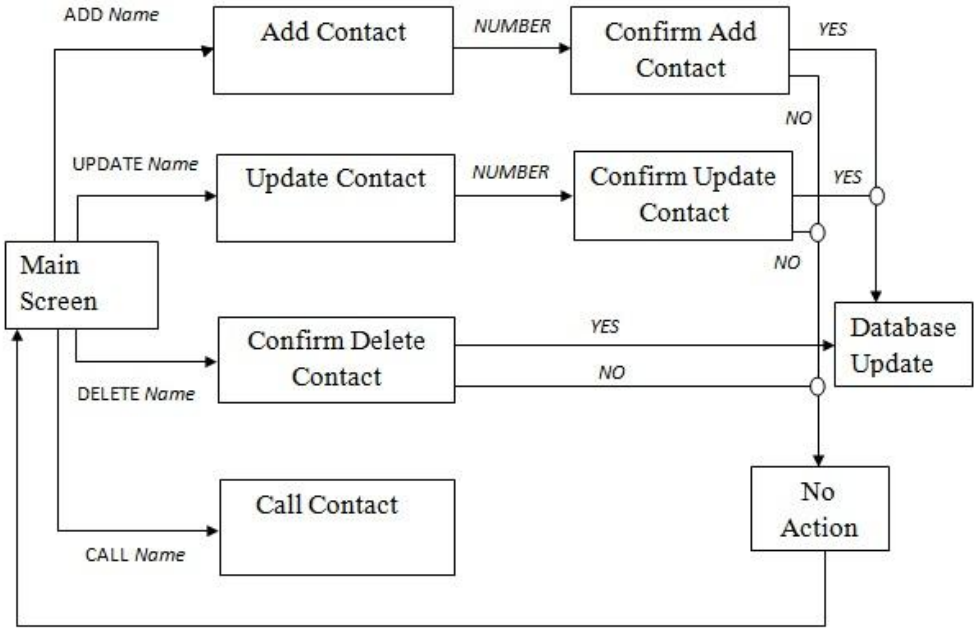

Figure 4.21: Data flow of phonebook application

4.6 Text-to-Speech module

Text-to-Speech is a technology which converts the input text into speech. A small module of this technology has been used in this system. In the phonebook application, whenever any output prompts out on the screen, the Text-to-Speech module comes in the action.

This module converts the text written on the output forms into speech and plays the ***.wav*** files according to it. For example, if a contact has been asked to update and user says ***UPDATE AASHISH***, then it prompts a new screen to enter a 10 digit number by speech input. When the input gets completed, it confirms from user to update the contact by prompting a screen asking for ***YES*** or ***NO***. Along with it, an audio file also gets played asking for confirmation of update and repeats the new 10 digit number entered by the user. If user says ***YES***, then it updates the contact.

The working of the Text-to-Speech module is based on Java code. The speech files for each digit has already been recorded and saved. Also for confirmation queries, speech files have been saved. When the system confirms the number to be updated or added, the query audio file is played. When the system repeats the 10 digit number entered by the

user, the corresponding audio files for the digits of the entered number are played by Java methods and the system gets able to generate the audio output. In this way, the complete system works in a complete speech interactive mode. The user gives input in speech form and even gets output in speech form.

In this chapter, the Speech-to-Text system has been developed by installing the tools and training the system. The phonebook application has also been developed in Java. In the next chapter, the result and description of the proposed system has been presented.

Chapter 5
Testing and Results

In this chapter, the complete execution of the Speech-to-Text system for phonebook automation has been discussed. This chapter discusses about the functionality of the system with the snapshots during execution. Along with it, this chapter discusses the exceptional cases during the execution of the system.

5.1 Speech-to-Text system with phonebook automation

As the complete system gets executed, the main screen gets appeared as shown in figure 5.1.

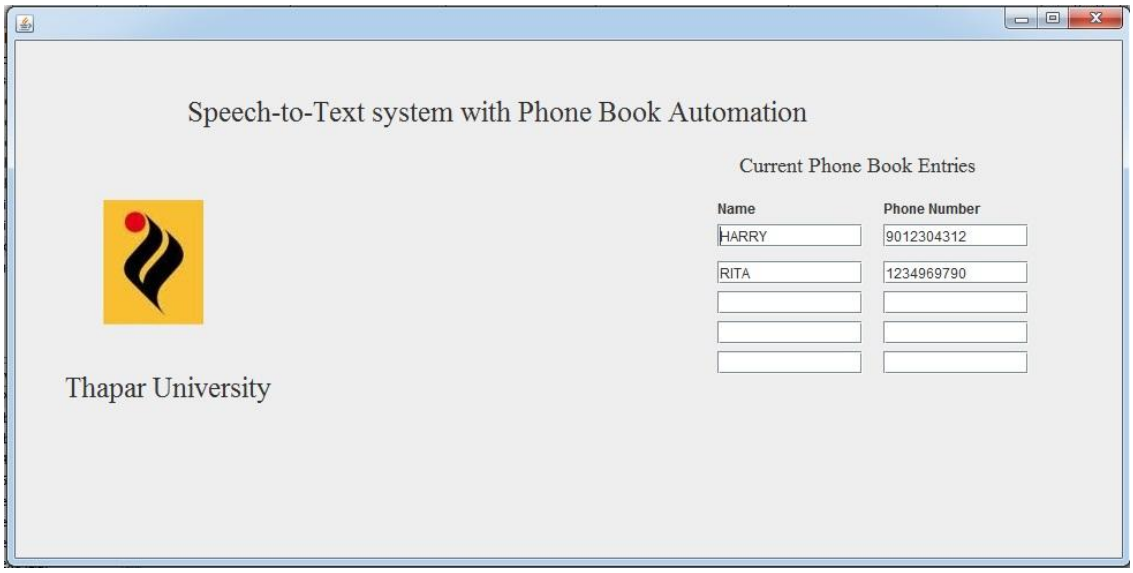

Figure 5.1: Main screen after execution

A table has been displayed on the main screen to show currently available contacts in the database. This main screen waits for the command to be given by the user. The speech commands that a user can give as input are as follows.

- **Add *Name***

If the input text gets started with ***ADD***, then this action has been performed. The name from the text has been captured. A new window titled as ***ADD CONTACT***, has been displayed with name already shown on that window, and asking to input number as shown in figure 5.2.

Figure 5.2: Add contact screen

A 10 digit number should be spoken by the user. The number gets displayed in the text box, digit after digit as user speaks as shown in figure 5.3.

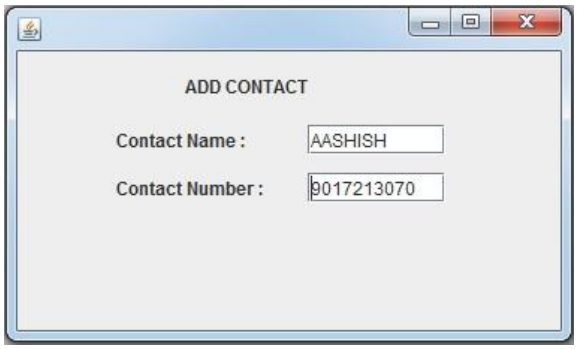

Figure 5.3: Add contact screen with number filled

As the 10 digits get completed, the control goes over to the confirmation window. This new window has been titled as ***ADD CONTACT***. The name and 10 digit phone number has already been written on this window as shown in figure 5.4.

Figure 5.4: Confirm add contact screen

The system waits here for input of ***YES*** or ***NO***. If the user speaks ***YES***, the contact gets added and the database gets updated as shown in figure 5.5.

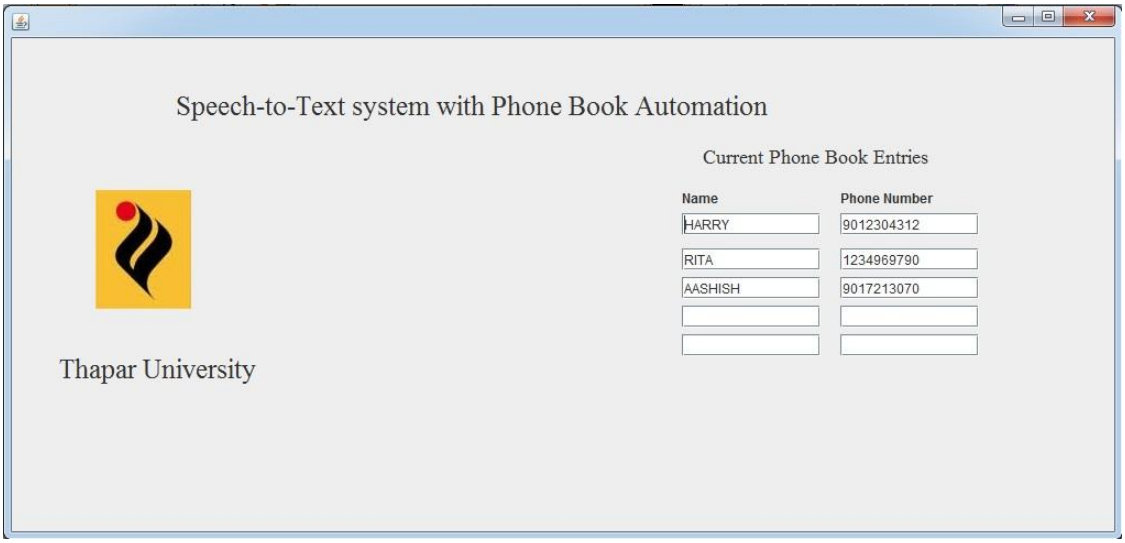

Figure 5.5: Updated database after contact addition

If the user speaks ***NO***, the control gets transferred to the main screen waiting for next command.

On the main screen, if the user speaks **ADD *NAME***, and that name already exists in the contacts then system prompts out an error message showing that name already exists in contacts as shown in figure 5.6. ***ADD HARRY*** was spoken in this case.

Figure 5.6: Prompt dialogue box

- **UPDATE *Name***

If the input text gets started with ***UPDATE***, then this action has been performed. The name from the text has been captured. A new window titled as ***UPDATE CONTACT***, has been displayed with name and old number already shown on that window, and asking to input number as shown in figure 5.7.

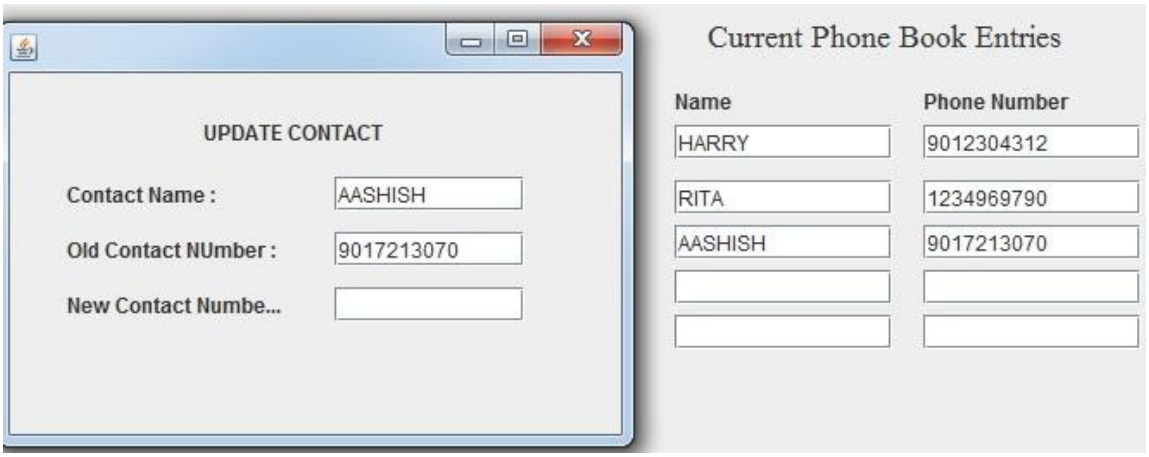

Figure 5.7: Update contact screen

A new 10 digit number should be spoken by the user. The number gets displayed in the text box, digit after digit as user speaks as shown in figure 5.8.

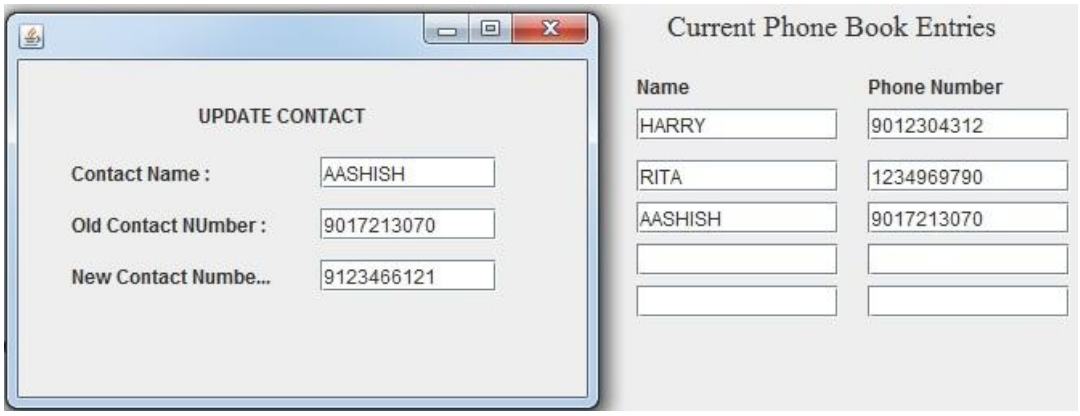

Figure 5.8: Update contact screen with new number filled

As the 10 digits get completed, the control goes over to the confirmation window. This new window has been titled as ***UPDATE CONTACT***. The name and the new 10 digit phone number has already been written on this window as shown in figure 5.9.

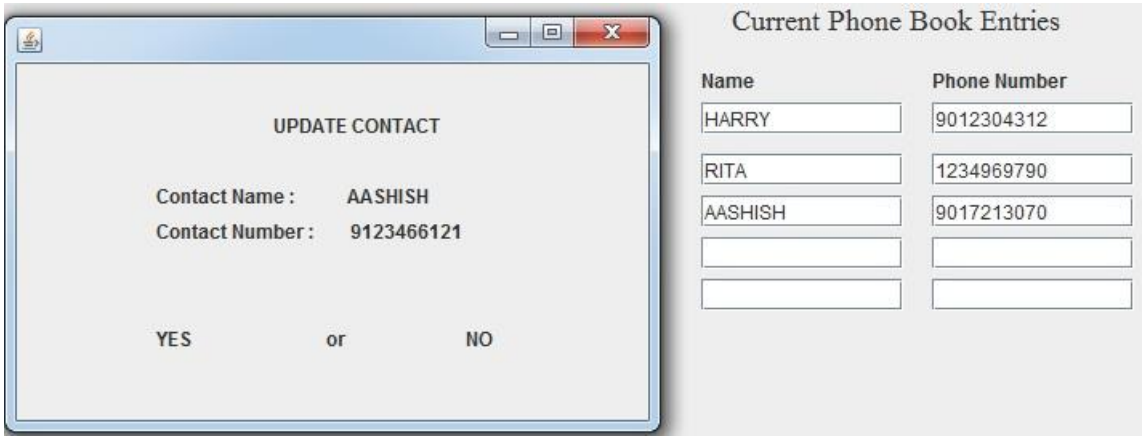

Figure 5.9: Confirm update contact screen

The system waits here for input of ***YES*** or ***NO***. If the user speaks ***YES***, the contact gets changed and the database gets updated as shown in figure 5.10.

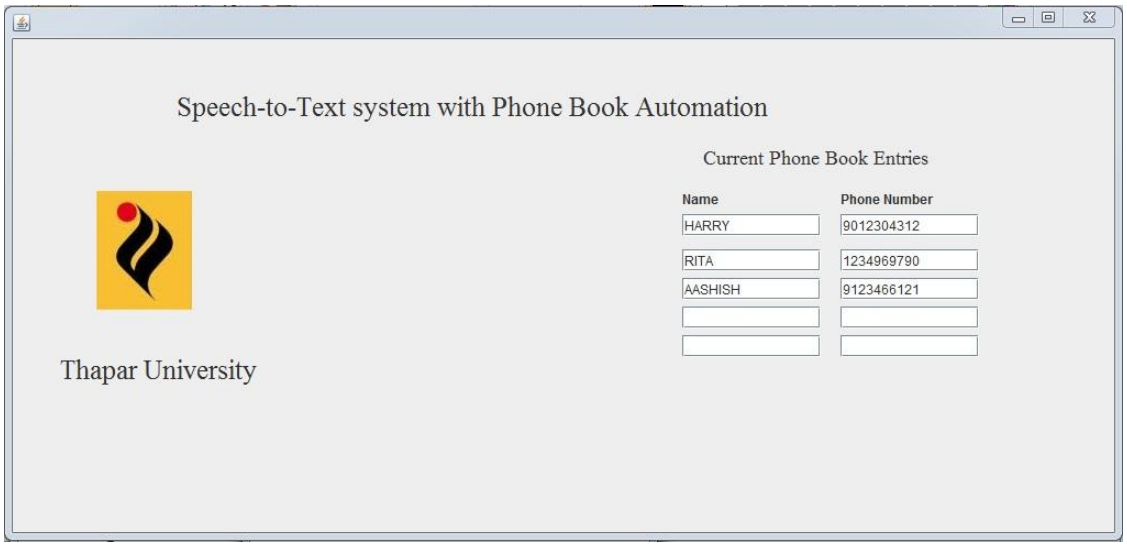

Figure 5.10: Updated database after contact update

If the user speaks ***NO***, the control gets transferred to the main screen waiting for next command.

On the main screen, if the user speaks **UPDATE** ***NAME***, and that name doesn't exist in the contacts then system prompts out an error message showing that name doesn't exists in contacts as shown in figure 5.11. ***UPDATE SUNIL*** has been spoken in this case.

Figure 5.11: Prompt dialogue box

- **DELETE *Name***

If the input text gets started with ***DELETE***, then this action has been performed. The name from the text has been captured. A new window titled as ***DELETE CONTACT***, has been displayed with name and number already shown on that window, and asking to delete contact as shown in figure 5.12.

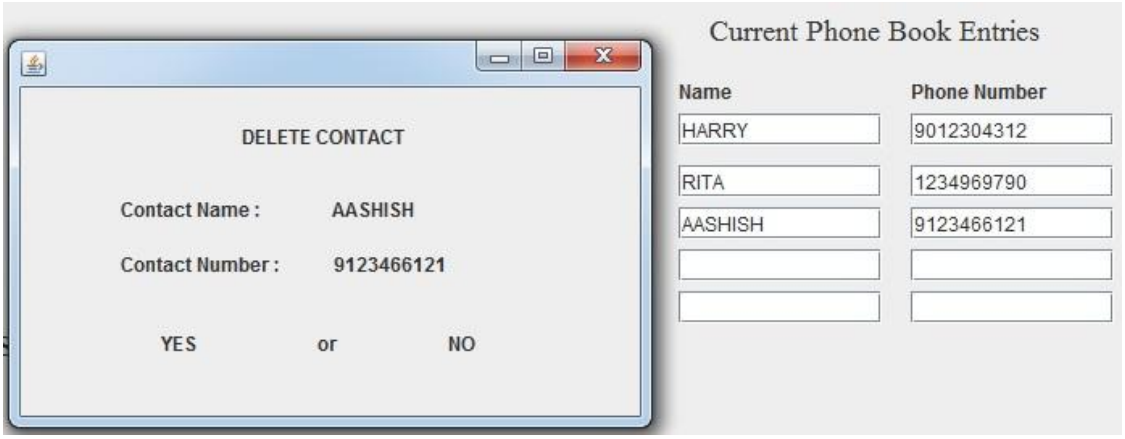

Figure 5.12: Confirm delete contact screen

The system waits here for input of ***YES*** or ***NO***. If the user speaks ***YES***, the contact gets deleted and the database gets updated as shown in figure 5.13. If the user speaks ***NO***, the control gets transferred to the main screen waiting for next command.

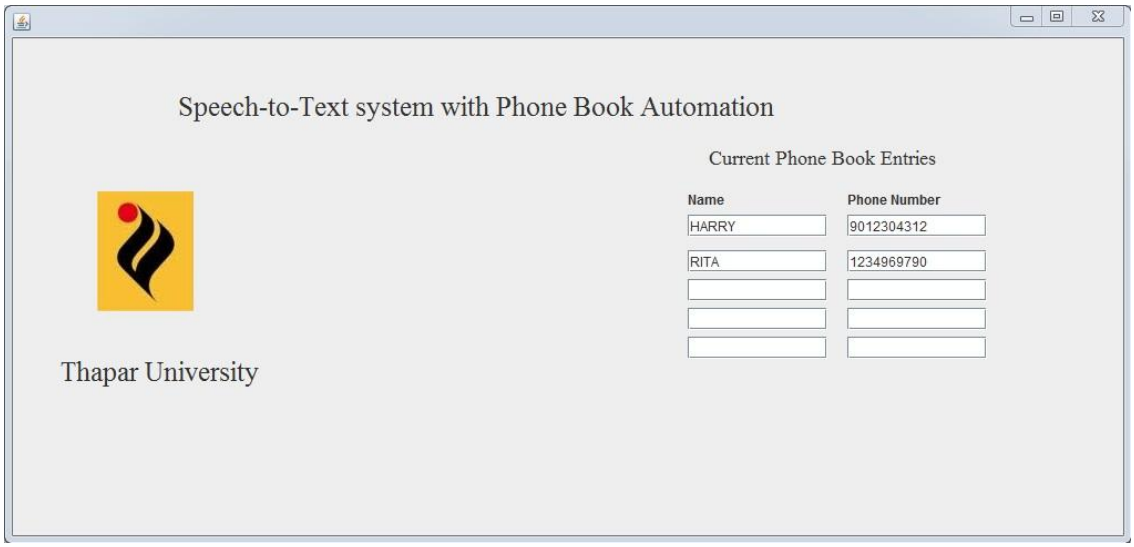

Figure 5.13: Updated database after contact deletion

On the main screen, if the user speaks **DELETE *NAME***, and that name doesn't exist in the contacts then system prompts out an error message showing that name doesn't exists in contacts as shown in figure 5.14. ***DELETE SUNIL*** has been spoken in this case.

Figure 5.14: Prompt dialogue box

- **CALL *Name***

If the input text gets started with *CALL*, then this action has been performed. The name from the text has been captured. A new window titled as ***CALL CONTACT***, has been displayed with name. The new window shows calling process as shown in figure 5.15.

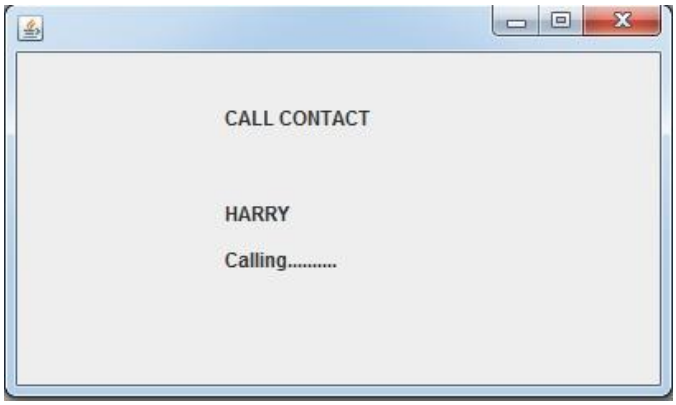

Figure 5.15: Call contact screen

This has been coded just to show the functionality of the system. This action may be actually linked with original calling process of the mobile phone when this application would be implemented in that environment.

The Speech-to-Text system with phonebook automation has been tested completely with exceptional cases. The result of the proposed system is satisfactory. In the next chapter, the conclusion and future scope of the proposed system have been presented.

Chapter 6
Conclusion and Future Scope

6.1 Conclusion

A Speech-to-Text system with phonebook automation has been developed in this thesis work. In order to do this, the raw input data was prepared to process it during the training of the system. The system was trained by monophones. The system was also trained by triphones and tied-state triphones to make it robust. An acoustic model had been created by complete training of the system.

An application was developed in Java which worked as an interface of a phonebook to the user. The data was saved in MySQL database. In this application, user is able to add new contact by speech input. By giving speech input, user may update, delete or call the saved contacts. A Text-to-Speech module was developed in Java using some speech files recorded for given text. This module speaks the output of the executing system.

So, the complete developed system is just a small application to reveal that a full fledge system can be developed in a mobile phone environment for the blind people so that they would be able to use their phone easily.

6.2 Future scope

In future, the proposed system can be improved as discussed below.

- As the system has been developed for a few contact names. So, the number of contact names can be increased from which the system can be trained.
- The current system is a desktop application. In order to convert it to work as a complete mobile application, coding can be done in J2ME.
- The proposed system has been developed for English language. So, it can be trained for other languages also.

- This system can be developed for multiple users by making it a speaker independent system.
- The Speech-to-Text system can be developed as a web application so that it can be easily accessed by the users.

References

[1] Anumanchipalli G., Chitturi R., Joshi S., Kumar R., Singh S. P., Sitaram R.N.V., Kishore S.P., "Development of Indian Language Speech Databases for Large Vocabulary Speech Recognition Systems", Hewlett Packard Labs, Bangalore.

[2] *Automatic Speech Recognition and Understanding*, [online], Available : *http://ewh.ieee.org/r10/bombay/news6/AutoSpeechRecog/ASR.htm*, [Accessed : November 10, 2011].

[3] Aymen M., Abdelaziz A., Halim S., Maaref H., "Hidden Markov Models for Automatic Speech Recognition", Laboratory of Micro-Opto-electronic and Nanostructure, Faculty of sciences, Monastir, Tunisia.

[4] Bapat A. V., Nagalkar L. K., "Phonetic Speech Analysis for Speech to Text Conversion", in *Proc. 2008 IEEE Region 10 Colloquium and the Third International Conference on Industrial and Information Systems*, Kharagpur, India, 2008, pp. 1-2.

[5] Bridle J. S., Brown M. D., "Connected word recognition using whole word templates", in *Proc. Institute of Acoustics Autumn Conference*, 1979, pp. 25-28.

[6] Cook S., *Speech Recognition HOWTO***,** [online], Available : *http://www.faqs.org/docs/Linux-HOWTO/Speech-Recognition-HOWTO.html*, [Accessed : October 24, 2011].

[7] Davis K. H., Biddulph R., Balashek S., "Automatic recognition of spoken digits", *J. of Acoustical Society of America*, vol. 24 (6), 1952, pp. 637-642.

[8] Decker M. A., Adda G., Lamel L., Gauvain J.L., "Developments in large vocabulary continuous speech recognition of German", in *Proc. Acoustics, Speech, and Signal Processing, ICASSP-96*, Atlanta, Georgia, 1996, pp. 153-156.

[9] Forgie J. W., Forgie C. D., "Results obtained from a vowel recognition computer program", *J. of Acoustical Society of America*, vol. 31(11), 1959, pp. 1480-1489.

[10] Fry D. B., "Theoretical aspects of mechanical speech recognition", *J. of British Inst. Radio Engr*., vol. 19(4), 1959, pp. 211-229.

[11] Furui S., "Recent progress in corpus-based spontaneous speech recognition", *IEICE Transactions on Information & Systems*, Vol. E88-D(3), 2005, pp. 366-375.

[12] Furui S., "Speaker independent isolated word recognition using dynamic features of speech spectrum", *IEEE Transactions of Acoustics, Speech, Signal Processing, ASSP*, vol. 34(1), February 1986, pp. 52-59.

[13] Hasanabadi H., Rowhanimanesh A., Yazdi H. Tabatabaee, Sharif N., "A Simple and Robust Persian Speech Recognition System and Its Application to Robotics", in *Proc. International Conference on Advanced Computer Theory and Engineering*, Phuket, Thailand, 20-22 December, 2008.

[14] Itakura F., "Minimum prediction residual applied to speech recognition", *J. of ASSP*, vol. 23 (1), 1975, pp. 67-72.

[15] Jelinek F., "The development of an experimental discrete dictation recognizer", in *Proc. IEEE*, vol. 73(11), 1985, pp. 1616-1624.

[16] Jelinek F., Bahl L.R., Mercer R.L., "Design of a linguistic statistical decoder for the recognition of continuous speech", *IEEE Transactions on Information Theory*, vol. 21(3), 1975, pp. 250-256.

[17] Kemble K. A., "An Introduction to Speech Recognition", Program Manager, Voice Systems Middleware Education IBM Corporation, unpublished.

[18] Kumar K., Aggarwal R.K., "Hindi Speech Recognition System using HTK", *J. of International Journal of Computing and Business Research*, vol. 2(2), 2011, pp. 3-7.

[19] Lamel L., Vieru B., "Development of a Speech-to-Text Transcription System for Finnish", in *Proc. 2nd Intl. Workshop on Spoken Languages Technologies for Under-Resourced Languages (SLTU'10)*, Penang, May 2010.

[20] Leggetter C.J., Woodland P.C., “Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models”, *Computer Speech and Language*, vol. 9, 1995, pp. 171-185.

[21] Liu Y., Shriberg E., Stolcke A., Peskin B., Ang J., Hillard D., Ostendorf M., Tomalin M., Woodland P., Harper M., “Structural metadata research in the EARS program”, in *Proc. ICASSP*, vol. 957, 2005, pp. 1-4.

[22] Martin T. B., Nelson A.L., Zadell H. J., “Speech recognition by feature abstraction techniques”, August 1964.

[23] Myers C. S., Rabiner L. R., “A level building dynamic time warping algorithm for connected word recognition”, *IEEE Transactions on Acoustics, Speech, Signal Processing, ASSP*, vol. 29(2), 1981, pp. 284-297.

[24] Nagata K., Yasuo K., Seibi C., “Spoken digit recognizer for Japanese language”, *J. of AES*, vol. 12(4), October 1964, pp. 336-342.

[25] Olson H. F., Belar H., “Phonetic typewriter”, *J. of Acoustical Society of America*, vol. 28 (6), 1956, pp. 1072-1081.

[26] Pavlovic N., “Natural Language Processing and Speech Enabled Applications”, Computer Science Department City Liberal Studies, Affiliated Institution of University of Sheffield.

[27] Peiiagarikano M., Bordel G., “Speech-to-Text Translation by a non-word Lexical based system”, in *Proc. Fifth International Symposium on Signal Processing and its Applications, ISSPA ‘99*, Brisbane, Australia, August, 1999, pp. 111-114.

[28] Rabiner L.R., “A tutorial on hidden Markov models and selected applications in speech recognition”, in *Proc. IEEE*, vol. 77 (2), February 1989, pp. 257-286.

[29] Rabiner L. R., Levinson S. E., Rosenberg A. E., Wilpon J.G., “Speaker independent recognition of isolated words using clustering techniques”, *IEEE Transactions on Acoustics, Speech, Signal Processing, ASSP*, vol. 27(4), 1979, pp. 336-349.

[30] Raza A. A., Hussain S., Sarfraz H., Ullah I., Sarfraz Z., "Design and development of phonetically rich Urdu speech corpus", in *Proc. Speech Database and Assessments, Oriental COCOSDA International Conference*, Beijing, China, 2009, pp. 38-43.

[31] Reddy D. R., "An approach to computer speech recognition by direct analysis of the speech wave", Computer Science Dept., Stanford Univ., 1966.

[32] Sakai T., Doshita S., "The phonetic typewriter information processing", in *Proc. IFIP Congress*, Munich, Germany, 27 August - 1 September, 1962.

[33] Sakoe H., Chiba S., "Dynamic programming algorithm optimization for spoken word recognition", *J. of ASSP*, vol. 26 (1), 1978, pp. 43-49.

[34] Shinoda K., Lee C. H., "A structural Bayes approach to speaker adaptation", *IEEE Transactions of Speech and Audio Processing*, vol. 9(3), 2001, pp. 276-287.

[35] Sivaraman G., Samudravijaya K, "Hindi Speech Recognition and Online Speaker Adaptation", in *Proc. International Conference on Technology Systems and Management, ICTSM,* Czech Technical University in Prague, Czech Republic, 2011, pp. 27-30.

[36] Suzuki J., Nakata K., "Recognition of Japanese vowels - preliminary to the recognition of speech", *J. of Radio Research Lab*, vol. 37 (8), 1961, pp. 193-212.

[37] Tomalin M., Diehl F., Gales M.J.F., Park J., Woodland P.C., "Recent Improvements to the Cambridge Arabic Speech to Text Systems", in *Proc. ICASSP*, Dallas, Texas, USA, 2010, pp. 4382-4385.

[38] *Tutorial: Create Acoustic Model – Manually,* [online], Available : *http://www.voxforge.org/home/dev/acousticmodels/windows/create/htkjulius/tutorial*, [Accessed: 15 December 2011].

[39] Varga A. P., Moore R. K., "Hidden Markov model decomposition of speech and noise", in *Proc. ICASSP*, vol. 2, 1990, pp. 845-848.

[40] Viterbi A. J., “Error bounds for convolutional codes and an asymptotically optimal decoding algorithm”, *IEEE Transactions on Information Theory*, vol. 13(2), April 1967, pp. 260-269.

[41] Vintsyuk T. K., “Speech discrimination by dynamic programming”, *J. of Kibernetika*, vol. 4 (2), 1968, pp. 81-88.

[42] *What can I do with Speech Recognition*, [online], Available : *http://windows.microsoft.com/en-US/windows7/What-can-I-do-with-Speech-Recognition*, [Accessed : March 15, 2012].

Publications

Published

Nishant Allawadi, Parteek Bhatia, "A Speech-to-Text System", Published in "*CSI Communications*", vol. 36(2), May 2012, pp. 18-21.

Communicated

Nishant Allawadi, Parteek Bhatia, "Speech-to-Text System for Phonebook Automation", Communicated to *International Journal of Computer Applications* (IJCA).