Arabic Speech Recognition Systems

By

Hamda M. M. Eljagmani

Bachelor of Science
Computer Engineering
Zawia University
Engineering College

A thesis submitted to the College of Engineering
At Florida Institute of Technology
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Electrical and Computer Engineering

Melbourne, Florida
May, 2017
Copyright 2016 Hamda Eljagmani

All right reserved

The author grants permission to make single copy____________________
We the undersigned committee hereby approve the attached thesis, “Arabic Speech Recognition Systems,” by

Hamda M. M. Eljagmani.

Veton Z. Këpuska, Ph.D.
Associate Professor
Electrical and Computer Engineering
Committee Chair

Samuel P. Kozaitis, Ph.D.
Professor and Department Head
Electrical and Computer Engineering

Ersoy Subasi, Ph.D.
Assistant Professor
Engineering System
Abstract

Title: Arabic Speech Recognition Systems
Author: Hamda M. M. Eljagmani
Advisor: Veton Këpuska, Ph.D.

Arabic automatic speech recognition is one of the difficult topics of current speech recognition research field. Its difficulty lies on rarity of researches related to Arabic speech recognition and the data available to do the experiments. Moreover, to build Arabic speech recognition system with an optimal word error rate (WER), the system has to be completely trained to the individual user. Even though speaker dependent system can effectively achieve this by training it explicitly for this one speaker, it requires a large amount of training data. In addition speaker dependent system requires to be trained to each speaker individually. For this reasons speaker dependent systems are too time expensive and not suitable for Arabic speech recognition systems where such training sets are not easily available. However, the mentioned problem related to amount of data can be tackled by using speaker independent systems. Since in speaker independent systems there are no relations between the training and test set, their performance is lower than in speaker dependent systems. Additionally, the word error rate is usually high for Arabic automatic speech recognition systems that are trained by native speakers and later used by non-native speakers. This is because of both acoustic and pronunciation differences and varying accents. The challenge that non native speech recognition faces is to maximize the recognition performance with small amount of non native data available.

The novelty of this work relies on the application of an open source research software toolkit (CMU Sphinx) to train, build, evaluate and adapt Arabic speech recognition system. First, Arabic digits speech recognition system is built by using speaker dependent and speaker independent systems to
show how the relations between training set and test set affect the recognizer's performance. Furthermore, different test sets are used to test speaker independent system in order to see how variety among speakers will contribute to the recognition performance. Second, Arabic digits speech recognition system is constructed by using native Arabic speakers and tested by both native Arabic and non-native Arabic speakers to show how the differences in pronunciations among non-native speaker and native Arabic speakers have a direct impact on the performance of the system.

Finally, Maximum Likelihood Linear Regression (MLLR) adaptation technique is proposed to improve the accuracy of both speaker independent system and native Arabic digits system that is used by non-native speakers. This start off sampling speech data from the new speaker and update the acoustic model according to the features which are extracted from the speech in order to minimize the difference between the acoustic model and the selected speaker. The results show the acoustic model adaptation technique is beneficial to both systems. The systems were evaluated using word level recognition. An overall improvement in absolute recognition rate of 13% and 6.29% for speaker independent and Arabic digits speech recognition system to foreign accented speakers adaptation have been obtained respectively.
Table of Contents

Abstract ........................................................................................................... iii
Table of Contents ............................................................................................... v
List of Figures ........................................................................................................ viii
List of Tables ........................................................................................................ x
Acknowledgements ............................................................................................... xi
Preface ................................................................................................................ 1
Chapter 1 Introduction into Automatic Speech Recognition system .................. 3
  1. Overview of Automatic Speech Recognition .............................................. 3
     1.1 Automatic Speech Recognition history progress ..................................... 3
     1.2 Automatic Speech Recognition Classification ........................................ 5
     1.3 Difficulties in ASR .................................................................................. 6
        1.3.1 Speaker variability ............................................................................ 6
        1.3.2 Amount of data and search space ...................................................... 7
        1.3.3 Human comprehension of speech compared to ASR ......................... 8
        1.3.4 Noise .............................................................................................. 8
        1.3.5 Continues speech ............................................................................ 8
        1.3.6 Spoken language is opposite to written language ............................. 9
Chapter 2 Systems and theories ......................................................................... 10
  2. CMU Sphinx engine ...................................................................................... 10
5.1.2 Building Language model ........................................... 67
5.1.3 Start training ......................................................... 71
5.1.4 Feature extraction ................................................... 72
5.1.5 Building and training the acoustic model ................. 72
5.1.6 Decoding ................................................................ 80
5.2 Adapting the acoustic model ....................................... 81

Chapter 6 Results and discussions ..................................... 85
6.1 Evaluation of isolated Arabic digits recognition systems ........ 85
  6.1.1 Evaluate speaker independent system (SI) ................. 85
  6.1.2 Evaluate speaker dependent system (SD) ................. 88
  6.1.3 Result of native Arabic system to foreign accented speakers .... 90
6.2 Results of adaptation .................................................... 91
  6.2.1 Adapting the acoustic model for independent system ........ 91
  6.2.2 Adaptation of foreign accented speaker to Isolated Arabic digits
      recognition system .................................................... 94

Chapter 7 Conclusion and future work ................................ 96
7.1 Conclusion ............................................................... 96
7.2 Future work ............................................................. 97

References ........................................................................ 98
List of Figures

Figure 1 - A typical speech recognition system ................................................................. 13
Figure 2 - Basic system architecture of any speech-recognition system ......................... 14
Figure 3 - Architecture of CMU Sphinx recognizer .......................................................... 15
Figure 4 - Process of MFCC ............................................................................................ 19
Figure 5 - An HMM for the word six that has four emitting states, two non-emitting states, the transition probabilities A, the observation probabilities B and a sample observation sequence .................................................................................. 21
Figure 6 - A standard 5-state HMM ................................................................................ 22
Figure 7 - A composite word model for word six, formed by four phone model each with three emitting sates ..................................................................................... 22
Figure 8 - Welch training method .................................................................................... 25
Figure 9 - Acoustic model training process ...................................................................... 26
Figure 10 - Adaptation framework for two fixed regression classes. Each regression class has mixture component. In order to maximize the likelihood of the adaptation, the transformation matrices Wi are estimated .................................. 46
Figure 11 - Regression class tree .................................................................................... 47
Figure 12 - Waveforms and spectrograms of all Arabic digits for Speaker 12 during trial 1 ......................................................................................................................... 53
Figure 13 - Structure of isolated Arabic digits speech recognition system ................. 59
Figure 14 - Construct the acoustic model ....................................................................... 61
Figure 15 - Structure of database (ArabicDigits) folder .................................................. 62
Figure 16-Use of CMU Cambridge toolkit .................................................. 68
Figure 17-Snapshot of ArabicDigits.html .................................................. 73
Figure 18-HMM topology 3-state model ....................................................... 76
Figure 19-Phases of adaptation ................................................................. 82
Figure 20-Error rate for individual Arabic digits for both test sets ................. 88
Figure 21-Total accuracy for both systems (Speaker dependent) and (Speaker independent) ................................................................. 90
Figure 22-Arabic System accuracies when is tested using both native Arabic and non native Arabic test sets ......................................................... 91
Figure 23-The accuracy for speaker independent before and after adaptation ..... 94
Figure 24-Overall Accuracy for Arabic digits system before and after adaptation to foreign speakers ................................................................. 95
List of Tables

Table 1-Parameter setting for mk_model_gen. ................................................................. 27
Table 2-Parameters for mk_flat. ....................................................................................... 28
Table 3-Parametr setting of cp_parm. ................................................................................ 29
Table 4-Parameters of bw program. .................................................................................. 30
Table 5-Parameters of init_mixture. .................................................................................. 35
Table 6-Arabic digits from zero to nine. ......................................................................... 49
Table 7-Recording system parameters. ........................................................................... 60
Table 8-The purpose of each folder/file in the database (ArabicDigits) ......................... 63
Table 9-ArabicDigits. dic file structure. ......................................................................... 63
Table 10-ArabicDigits. phone file that is used in the training. ........................................ 65
Table 11-Accuracy for individual Arabic digits for speaker independent system using test1. .............................................................................................................. 86
Table 12- Accuracy for individual Arabic digits for speaker independent system using test2. ........................................................................................................... 87
Table 13-Accuracy for individual Arabic digits for speaker dependent system. ....... 89
Table 14-for individual Arabic digits for speaker independent system using test1. ................ ................................................................. 92
Table 15-Accuracy for individual Arabic digits for speaker independent system using test2. ........................................................................................................... 93
Acknowledgements

First, I would like to express my sincere gratitude to my advisor Prof. Veton Këpuska for the continuous support of my Master study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Master study.

Next, I would like to thank my parents for allowing me to realize my own potential. All the support they have provided me over the years was the greatest gift anyone has ever given me. Also, I need to thank my aunts Zakia and Sokaina, who taught me the value of hard work and an education. Without them, I may never have gotten to where I am today.

Finally, I would also like to acknowledge my committee members: Prof. Samuel Kozaitis and Prof. Ersoy Subasi, who graciously agreed to serve on my committee.
This thesis is dedicated to my loving husband for his unconditional support and encouragement.
Preface

The whole thesis consists of seven chapters and one appendix. Chapter one is introduction about Automatic Speech Recognition systems that includes review; a brief history and the progress made; the present state of the art of these systems; main parameters that categorize ASR systems; and the difficulties that ASR system face.

Because thesis is based on the open source CMU Sphinx recognizer, in chapter two a brief review of CMU sphinx engine and its versions Sphinx1, Sphinx2, Sphinx3, Sphinx4, Sphinxbase, PocketSphinx, SphinxTrain, and CMU Cambridge Language Modeling Toolkit are first introduced. The architecture of CMU Sphinx recognizer is explained in detail namely, feature extraction, acoustic model, language model and decoding. This chapter focuses on the acoustic model training. Finally chapter two defines how the performance of Automatic Speech Recognition systems is evaluated.

Third chapter summarizes the previous studies that investigate different adaptation techniques, and explain one of the most used adaptation techniques namely Maximum likelihood linear regression (MLLR).

The fourth chapter is introduction, mainly about the Arabic language, Arabic Dialects and the characteristics of Arabic alphabets. Moreover, this chapter presents a description of Arabic digits from zero to nine. The end of chapter introduces the research that is done in Arabic speech recognition field.

In chapter five, three isolated Arabic digits recognition systems are constructed: speaker dependent, speaker independent and Native Arabic speaker system. The different stages are also explained in detail, starting from data preparation, feature extraction, building the language model, building and training acoustic model and decoding. Furthermore, chapter five proposes adaptation technique for both speaker independent and Native Arabic speakers systems in order to increase the performance.
The evaluation and the result of all constructed systems before and after adaptation are discussed in chapter six. Figures and tables are provided to clarify each result.

Conclusion of all experiments and recommendation for future work are provided in chapter seven. Finally, running, compiling, and testing of isolated.
Chapter 1

Introduction into Automatic Speech Recognition system

1. Overview of Automatic Speech Recognition

Speech recognition, or more commonly known as Automatic Speech Recognition (ASR) is a technology that converts humans' speech signals into a sequence of words; these words can be the final output or the input to natural language processing. The main purpose of ASR systems is to recognize natural languages that are spoken by human beings (Mustaquim, 2011). In the last few years, Automatic Speech Recognition technologies have changed the way we live, work, and interact with devices.

Main advantages of ASR are reducing cost by replacing human achieving specific tasks with machines, new income opportunities since speech and understanding systems provide a high quality customer services care without the need to use keyboards, and customer conservation by improving the customer experience (Rabiner & Juang, 2006).

ASR technology has a wide range of applications such as command recognition (computers that have voice user interface), foreign languages' application, dictation, and hands free operations and controls which make machines and humans interactions much easier. According to Mustaquim (2011), most of ASR systems are built using the Hidden Markov Models (HMM) one of the powerful statistical techniques for modeling the acoustics of speech and use either statistic language (n-grams) or rule based grammars to model the language components.

1.1 Automatic Speech Recognition history progress

Human beings have been interested in creation of machine that can talk and understand human speech long time ago (Huang, Benesty, & Sondhi, 2008). Early attempts to design systems for automatic speech recognition were in 1952
by Davis, Biddulph, and Balashek of Bell Laboratories. Their system was built for isolated digit recognition based on single person, and the system measured the formant frequencies for each numerical digit vowel segment. During 1960s multiple ASR systems were developed, most notable was Suzuki and Nakata Radio Research Lab vowel recognizer in Tokyo. The recognizer analyzed and recognized speech in various portions of the input utterance by using a speech segment for the first time (Juang & Rabiner, 2006). Another significant discovery came out in this period was dynamic time warping solved the problem of speech signal length unequal (Huang, Benesty, & Sondhi, 2008).

A major progress has been made in ASR systems field in the late 1960s and early 1970s by introducing the statistical methods of hidden Markov modeling (Rabiner, 1989). In parallel studies moved towards large vocabulary speech recognition by international business machine corporation (IBM). AT & T Bell laboratories also focused on the design of a speaker independent system that was able to deal with acoustic diversity.

Breakthrough happened in 1980s when researchers started to focus on large vocabulary independent continues speech recognition systems. The most famous is Sphinx system from Carnegie Mellon University (CMU). Another considerable development in speech recognition researches was characterized by a movement from template matching to a statistical modeling framework based on HMM and artificial neural networks (ANNS) (Juang & Rabiner, 2006).

In the 1990’s, a number of innovations took place in the field of Automatic Speech Recognition with the presence of multimedia era. ASR technology is widely used on telephone communication network and other commercial field services. Modeling relied on very large vocabulary and continues speech recognition system have had a significant progression in this decade.

In the recent century, ASR systems have been used in verity of fields particularly with the development of Internet and mobile communications. Human machine interaction, keyword spotting, natural spoken dialogue and
multi-lingual language interpretation became new application directions (Froomkin, 2015).

1.2 Automatic Speech Recognition Classification

Following are some task parameters that classify ASR systems:

**Speaking style:** this indicates whether the task is for isolated words (digits recognition) or connected words (series of digits).

**Vocabulary size:** speech recognition task is easier when the vocabulary is smaller. However, not only the vocabulary size determines the task complexity, but also the grammar constraints of the tasks especially tasks with no grammar constraints since all words can follow any word (Adami, n.d.).

**Speaker mode:** there are two modes that can be used in the recognition system, specific speaker (speaker dependent) or by any speaker (speaker independent). Although speaker dependent systems require to be trained with the user speaker's data, they generally achieve better recognition results since there is no much variability from multiple users. In addition, speaker dependent (SD) modes are not reusable since they need complete re-training for each new user speaker, which make this kind of models are impractical for most applications. In contrast to speaker independent that is more appealing since it does not require training for each new user speaker. Moreover, in speaker independent acoustic model there is no fixed relation between training and production speakers. ASR systems that use speaker independent can give better results for new speakers than any adapted ones and perform adaptation to the individual user's voice to improve their recognition performance. In general SI modes have poor overall performance (Lee & Gauvain, 1993).

**Transducer type:** this parameter is based on the type of device used to record the speech. The recording may range from high-quality microphones to telephones (landline) to cell phones to array microphones (used in applications that track the speaker location).

**Channel type:** the properties of the recording channel can impact the speech signal. It may range from a simple microphone connected to digital speech
acquisition hardware, telephone channels (with a bandwidth about 3.5 kHz) to wireless channels with fading and with a sophisticated voice or a mobile phone channel characterized by packet losses (Adami, n.d.). Each channel has its characteristics such as frequency limits (e.g. a 16000 or 55100 sample per second microphone in contract to telephony system that has 5000 Hz, 8000 sample per speech). In addition to channel noise due to channel properties that remain consistent to variable factors such as vicinity of electronic equipment which varies greatly are some of the salient feature of speech environment (Ravishankar, 1996).

1.3 Difficulties in ASR

1.3.1 Speaker variability

Researcher O'Shaughnessy (2008) supports that the most challenging task is building a reliable ASR system because of significant diversity in human speech and accent due to their unique physical body and personality. Humans have major different voices and pronunciations of the same content. Not only the voice is different between speakers, but also there are wide diversities within one particular speaker.

More explanation is given by (Forsberg, 2003) in Why is Speech Recognition Difficult article, where some of these variations are listed below:

Realization

The output speech signal will not be identical when the same words were uttered over and over again. The realization of speech changes over time even if the speaker tries to pronounce it exactly the same. There will be some small differences in the acoustic wave.

Speaking style

All human beings speak differently to express their personality. They have personal vocabularies and unique ways to utter and emphasize these vocabularies. The speaking style also depends on the context and the situation; we speak differently in the bank, with our parents…etc. Humans also express their emotions and feeling via speech. If we are disappointed, we might lower
our voice and speak more slowly. In contrast to if we are frustrated, we might speak more loudly.

**The gender and age of the speaker**

Men and women with different ages have different voices due to difference in vocal tract length. In general women have shorter vocal tract and higher tone than men.

**Anatomy of vocal tract**

Not only is the length of the vocal cords differ among different speakers, also the formation of the cavities and the size of the lungs. These physical attributes change over time depending on the age and health of the speaker.

**Speed of speech**

Humans speak with different pace. We tend to speak faster if we are stressed, and decrease the speed if we are tired. In addition, we speak in different modes of speech if we talk about something unknown or known.

**Regional and social dialects**

The features of pronunciation, vocabulary and grammar differ according to the geographical area the speaker come from and the social group of the speaker.

### 1.3.2 Amount of data and search space

A large amount of speech data are produces every second when communicating with a computer via microphone. This data must be matched to set of sounds, words, sentences, and phones that consist of monophones, diphones and triphones. The numbers of sentences that can be break down into groups of groups of phones and words are enormous.

The quality of speech signals are affected by lowering the sampling rate, resulting in incorrect analysis. Whilst, the quality and the amount of input data can be controlled by the quantity of samples of the input signal. However, if the intended word is not in the lexicon, then another problem is called out-of-vocabulary will introduce and ASR system has to handle it.
1.3.3 Human comprehension of speech compared to ASR

Humans can communicate with speech and body language (signals) such as hand waving, eye movement and postures. Additionally, when listening humans use more than their ears, they use the knowledge they have learned about the speaker and the subject to predict words not yet spoken. Moreover, idioms and how we usually say things can make prediction easier.

Nevertheless, in ASR system is difficult to measure up humans' comprehension because it only has speech signal. It can be possible to build models for the grammatical structure, and use statistical models to enhance prediction, but how to model word knowledge is still difficult.

1.3.4 Noise

The greatest difficulties in designing an ASR are handling noise background and other external distortions that exist in the environment when the speech is uttered. For example, a clock ticking, music playing, another human speaker etc. ASR system must be able to identify and filter out theses unwanted information from the speech signal. Many methods are used to enhance ASR system ability to recognize stops only appear after a phrase or a sentence.

1.3.5 Continues speech

The speech that has no natural stops between the word boundaries, the stops only appear after a phrase or a sentence. This introduces another problem for Automatic Speech Recognition systems. First ASR should recognize phones and then group them into words, also ASR should be able to distinguish pauses between words which still difficult especially when the possible length of utterances increases and the pauses get unclear.
1.3.6 Spoken language is opposite to written language

In ASR, we have to address the main differences between spoken and written language since the spoken language has more performance errors. Another issue that has to be identified is that the grammaticality of spoken language is quite less complex and different to written language. For instance, 30-50% of all spoken language utterances consist of short utterances of 1-2-3 words with no predicative verb. Furthermore, collocations, grammatical constructions and frequencies of words are different to written language. In addition, in spoken language pronunciation there is a radical reduction of morphemes and words (Forsberg, 2003).

However, Automatic Speech Recognition system has overcome most of these difficulties and tried to tackle three constraints of ASR namely speaker independent, isolated words and small vocabulary (Lee, Hon, & Reddy, 1990). Therefore, ASR systems have become a grading role for many applications, hence variety of open source speech recognition systems have been developed, such as HTK and CMU Sphinx-4 which developed at Cambridge University and Carnegie Mellon University respectively (Satori, Harti & Chenfour, 2007).
Chapter 2

Systems and theories

2. CMU Sphinx engine

CMU Sphinx is a combination of multiple Automatic Speech Recognizers, and supports various libraries and training tools. Research of CMU Sphinx has lasted over two decades started with Sphinx 1, which was developed by Kai-Fu Lee and his staff until Sphinx 4 in present. CMU Sphinx is the first system that shows the feasibility of accurate Large Vocabulary Continuous Speech Recognizers (LVCSR). CMU Sphinx is an open source that has powerful robustness and good extensibility that allow researchers to use it as their speech recognition research tool.

The sphinx group at Carnegie Mellon University CMU in 1987 developed this open source speech recognizer with cooperation of Mitsubishi Electric Research Laboratories (MERL), Sun Microsystems laboratories and Hewlett Packard's Cambridge Research Lab (HP). CMU Sphinx was funded by University of California and Massachusetts Institute of Technology (MIT). It supports various operating system platforms, such as Microsoft Windows, Mac OS X, Linux and Android. CMU Sphinx developed multiple versions which include:

1-Sphinx 1: It was constructed by Kai-Fu Lee and his staff in 1987. It provided high performance speaker independent English ASR. This system introduced HMM into Automatic Speech Recognition which used 3 states discrete HMM, 256 vocabularies with high correct recognition rate about 89% (Ravishankar, 1996).

2-Sphinx 2: It is a high speed large vocabulary speech recognizer that was developed on the basis of Sphinx1 in 1922. It used in pronunciation learning systems, dialogue systems and interactive applications. Sphinx 2 introduced the design of PocketSphinx. Sphinx 2 uses five states semi-continuous HMM with probability density functions, and its source code is written in C language.
Sphinx 2 correct recognition rate was 90% when Wall Street Journal speech database was used (Raza, 2009). The latest version provided both a number of library function and hardware interface for live applications.

3- Sphinx3: It is slower than Sphinx2, but provides more accurate Large Vocabulary Speech Recognition System. Both semi-continuous HMM and continuous HMM were combined in Sphinx3. Two research branches were produced during the process of Sphinx 3 development in 1995 in order to support multiple operation modes. Flat decoder which is came from Sphinx3, and had more accuracy than tree decoder. Tree decoder was developed separately, but it is faster. According to (Danezis & Goldberg, 2009) flat decoder had a 10% higher accuracy than tree decoder. On the other hand, tree decoder ran 10 times faster than flat decoder. These two decoders did not merge together until the development of Sphinx 3.5.

4- Sphinx4: CMU Sphinx developed Sphinx4 in 2005. It is a completely rewritten version of Sphinx decoder in Java therefore it provides a powerful portability and flexible multi-threaded interface. It uses discrete, semi continuous HMM and continues HMM that can choose number of states from 3, 4 or 5. Sphinx 4 uses models trained by Sphinx 3 trainer and also recognize isolated and continuous speech.

5- PoketSphinx: It is the fastest version of CMU Sphinx speech recognition system that uses semi-continues output PDFs with HMM. It can be used in embedded devices and live applications even though it is as accurate as Sphinx 3 and Sphinx4.

6- SphinxTrain: It represents CMU Sphinx's training package tool that carry out acoustic model. It performs model training in Sphinx3 format. This format can be converted to Sphinx2 format.

7- CMU Cambridge Language Modeling Toolkit: This tool is used to train language models.

8- SphinxBase: It is a set of library that can be used by multiple CMU Sphinx projects (Raza, 2009).
2.1 Structure of CMU Sphinx

CMU Sphinx recognizer is based on the principles of statistical pattern recognition; in particular the use of hidden Markov models (HMMs) which is used to formulate speech recognition problems (Tan & Lindberg, 2008). As shown in figure -1 the speaker's mind decides what to say and then establishes the concepts in a sentence ,W, which is a sequence of words with pauses and other acoustic events such as uh's, um's, etc.). Then, W is passed into a noisy communication channel. This channel consists of the speaker’s vocal apparatus in order to produce the speech waveform and the speech signal-processing component of the speech recognizer X. At the end, the speech decoder tries to decode the acoustic signal X into a word sequences Ŵ which is close to the original word sequence W (Indurkhya & Damerau, 2010).

The dotted box in figure-1 represents the basic components of a typical speech recognition system. Both decoder and application interface represent outcomes that might be used to adapt other elements in the system. Acoustic models represent the knowledge about phonetics, acoustics, environment, microphone variability, and speakers' differences, etc. Language models contain information about what constitute a possible word, what words are likely to occur together, and in what sequence. Other factors are necessary for language models like the meanings and functions of operation that user might wish to perform. Speaker characteristics, speech style and rate, the recognition of basic speech segments, possible words, likely words, unknown words, grammatical variation, noise interference, nonnative accents, and the confidence scoring of results contribute in many uncertainties.
A successful speech-recognition system must be deal with all of these uncertainties. For example, the different accents and speaking styles of individual speakers are compounded by the lexical and grammatical complexity and variations of spoken language, which are all represented in the language model. The speech signal is passed through signal-processing module that extracts the most noticeable feature vectors for the decoder as shown in figure-2. Both acoustic and language models are used by decoder to produce the word sequence that has the maximum backward probability for the input feature vectors. Also, it provides information to adaptation elements in order to modify either the acoustic or language models so that improved performance can be obtained.

Both acoustic and language modeling can be described by the fundamental equation of statistical speech recognition:
\[
P(W)P(O|W) \frac{\hat{W}}{P(O)} = \arg_w \max \frac{P(W|O)}{P(O)} = \arg_w \max P(W|O)
\]

(1)

Where \( O = o_1, o_2, o_3, \ldots, o_n \) is the acoustic observation or feature vector sequence. The objective of speech recognition is to find out word sequences \( \hat{W} = w_1, w_2, w_3, \ldots, w_m \) which has the maximum backward probability \( P(W|O) \) as illustrated in Eq. (1). Because the maximization of Eq. (1) is carried out with the observation \( O \) fixed, the above maximization is equivalent of the maximization of the numerator:

\[
\hat{W} = \arg_w \max P(W) P(O|W)
\]

(2)

Where the probabilistic quantities computed by the language modeling and acoustic modeling components of speech recognition consist of \( P(W) \) and \( P(O|W) \) respectively.
Building accurate acoustic models $P(O|W)$, and language models $P(W)$, which can truly reflect the spoken language to be recognized is the biggest challenge. We need to analyze a word into a subword sequence in a large vocabulary speech recognition (often called pronunciation modeling). $P(O|W)$ should consider speaker variations, pronunciation variations, environmental variations, and context-dependent phonetic coarticulation variations. It is crucial to adapt $P(W)$ and $P(O|W)$ to increase $P(O|W)$ while using spoken language systems. Because one faces a practically infinite number of word patterns to search in continuous speech recognition, the decoding process of finding the best-matched word sequence, $W$, to match the input speech signal, $X$, in speech-recognition systems is more than a simple pattern recognition problem (Indurkhya & Damerau, 2010).

The main components and processes of CMU Sphinx recognizer as shown in figure-3 are described in more details in the following sections.

Figure 3-Architecture of CMU Sphinx recognizer.
2.1.1 Feature Extraction

Feature extraction is responsible for transforming the speech signal into a stream of feature vectors coefficients that have only the required information to identify a given utterance. These extracted features should have the following characteristics while dealing with speech signals:

1- Should be measured easily.
2- Should be consistent with time.
3- Should be robust to noise and environment.

The most widely used spectral analysis technique for feature vector extraction is Mel-Frequency Cepstral Coefficients (MFCC), which is used to mimic the human ear (Madan & Gupta, 2014).

First, convert the analog speech signal into a digital signal. This process is called an analog to digital conversion that has two steps:

1- A signal is sampled by measuring its amplitude at a specific time.
2- Store the amplitude measurement as integer. This process is known as quantization.

Second is pre-emphasis stage where the amount of energy in high frequencies is boosted using a high pass filter. Raising the energy of high frequencies makes information from these higher formants more available to the acoustic model.

Because we want to extract spectral features from a small window of speech that characterize a particular subphone, we cut speech signal into sections by adding window function. A more common window used in MFCC extraction is the Hamming window, which shrinks the values of the signal toward zero at the window boundaries, avoiding discontinuities. Hamming window is defined as the formula below:

\[
w[n] = \begin{cases} 
0.54 - 0.46 \cos \left( \frac{2\pi n}{L} \right) & 0 \leq n \leq L - 1 \\
0 & \text{Otherwise}
\end{cases}
\]

(3)
Where $L$ is frame long.

Following knows how much energy the signal at different frequency bands has. The tool for extracting this spectral information from a windowed signal is Discrete Fourier Transform (DFT).

$$x[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi nk}$$  \hspace{1cm} (4)

Where $x[n]$ is a windowed signal $x[n] \ldots \ldots x[m]$, and the output for each of $N$ discrete frequency band is a complex number $x[k]$ representing the magnitude and phase of that frequency component in the original signal. Then, the periodogram estimate of the power spectrum is computed by taking the absolute value of complex Fourier transform and square the result.

$$p[k] = \frac{1}{N}|x[k]|^2$$  \hspace{1cm} (5)

In order to improve speech recognition performance, some human hearing properties must be modeled. One of these properties is the less sensitivity at higher frequencies above 1000 Hz. This model can be done by warping the frequencies output by the DFT onto the Mel scale. The mapping between frequency in Hz and Mel scale is linear below 1000 Hz and the logarithmic above 1000 Hz. This intuition is implemented by several filters that collect energy from each frequency band. The formula for converting from frequency to Mel scale is:
\[ m(f) = 1125 \ln(1 + \frac{f}{700}) \] \hspace{1cm} (6)

Once we have the filterbank energies, we can take the logarithm of them.

The final step in MFCC feature extraction is the computation of the Cepstrum by applying the Inverse of Discrete Fourier Transform. There are two main reasons this is performed. Because of overlapping of the filterbanks, filterbank energies are correlated with each other. This extraction results in 12 cepstral coefficients for each frame.

Because energy correlates with phone identity and its useful for phone detection, it is a good idea to add energy from the frame. Another important factor about speech signal is that it is not constant from one frame to another. This also can provide a useful cue for phone detection. Therefore, adding the feature related to the change in cepstral feature for each of 13 features (12 cepstral features plus energy). These features are a delta or velocity, and a double delta or acceleration. Each of the 13 delta features (12 delta cepstral coefficients plus delta energy coefficient) represent the change between frames in the corresponding cepstral/energy features. While each of the 13 double delta features (12 double delta cepstral coefficients plus double delta energy coefficient) represent the change between frames in the corresponding delta features. In the result we end up with 39 MFCC features (Jurafsky & Martin, 2006).
2.1.2 Acoustic models

One of the most challenges of automatic speech recognition is the accuracy. Acoustic modeling plays an important role in improving this accuracy. The main purpose of acoustic model is to compute the likelihood of the observed
feature vectors given linguistic units (phones, words, subparts of phones) using a statistical method known as the Hidden Markov Model (HMM) with a mixture density Gaussian distribution. For instance, Gaussian Mixture Model (GMM) is used to compute the likelihood of a given feature vector \( P(o|q) \) for each HMM state \( q \), corresponding to a phone or subphone. The output of this stage will be a sequence of probability vectors, one for each time frame; each vector at each time frame contains the likelihood that each phone or subphone generated the acoustic feature vector at that time.

HMM components are described below:

\[
Q = q_1, q_2, \ldots, q_N
\]

Is a set of states.

\[
A = a_{01}, a_{02}, \ldots, a_{n1}, \ldots, a_{nn}
\]

\( A \) is a transition probability matrix, each \( a_{ij} \) represents the probability of moving from state \( i \) to state \( j \), where \( \sum_{j=1}^{n} a_{ij} = 1 \) \( \forall i \).

\[
O = o_1, o_2, \ldots, o_N
\]

A set of observations, each one is drawn from a vocabulary \( V = v_1, v_2, v_3, \ldots, v_N \).

\[
B = b_i(o_t)
\]

A set of observation likelihoods each expresses the probability of an observation \( o_t \) that generated from a state \( i \).

\( q_0, q_{\text{end}} \)

Start and end states which are not associated with observations.

For speech, the hidden states are phones, parts of phones, or words. The observation sequence for speech recognition is a sequence of acoustic feature vectors which are extracted during the previous stage. Each observed acoustic feature vector represents information about the amount of energy in different
frequency bands at a point in time. As mentioned in the feature extraction stage, each observation consists of a vector of 39 real valued feature indicating spectral information. These observations are drawn every 10 milliseconds.

Each HMM represents a single phone, and these states are concatenated together. Figure -5 shows an HMM for the word six.

![HMM diagram](attachment:image.png)

**Figure 5** - An HMM for the word six that has four emitting states, two non-emitting states, the transition probabilities A, the observation probabilities B and a sample observation sequence.

It can be clearly seen that certain connections or transitions are allowed. These transitions are constrained by the sequential nature of speech. For example, HMMs for speech do not allow transitions from states to earlier states in the word. In other words, states can transitions to themselves (self-loop) or to successive states only. This kind of HMM structure is called left to right HMM.

Since phone durations vary hugely, dependent on the phone identity, the speaker’s rate of speech, the phonetic context, and the level of prosodic prominence of the word, the use of self-loops allow a single phone to repeat in order to cover a variable amount of the acoustic input.

For recognizing small numbers of words like 10 digits, using HMM state to represent a phone is sufficient. The most common configuration to represent the phone is three HMM states, the beginning, middle, and end states. Each phone
has three emitting HMM states instead of one plus two non-emitting states at two ends. This 5 states phone HMM is known as a word model or phone model as shown in figure-6.

![Standard 5-state HMM](image)

**Figure 6- A standard 5-state HMM.**

To create a HMM for whole word using phone model, each phone of the word model in figure-5 is replaced with a 3 state phone HMM. The non-emitting start and end state for each phone model are replaced with transition immediately to the emitting state of the preceding and following phone. This leaves only two non-emitting states for the whole word as shown below.

![Composite word model](image)

**Figure 7- A composite word model for word six, formed by four phone model each with three emitting states.**
To summarize, the components of HMM model for speech recognition can be rewritten as follow:

\[ Q = q_1, q_2, q_3, \ldots, q_N \]

Is a set of states corresponding to subphones.

\[ A = a_{01}, a_{02}, \ldots, a_{n1}, \ldots, a_{nn} \]

A is a transition probability matrix, each \( a_{ij} \) represents the probability for each subphone of taking self-loop or moving to the next subphone, where \( \sum_{j=1}^{n} a_{ij} = 1 \forall i \).

\[ B = b_i(o_t) \]

A set of observation likelihoods each expresses the probability of an observed cepstral feature vector \( o_t \) that generated from subphone state \( i \).

Together represent a lexicon, a set of pronunciations for words. Each pronunciation has a set of subphones with the order of the subphones specifies by the transition probabilities \( A \).

The probability \( A \) and the states \( Q \) of Hidden Markov Models should be characterized by fundamental steps:

1- Computing Likelihood: Given an HMM \( \lambda = (A,B) \) and an observation sequence \( O \), determine the likelihood \( P(O|\lambda) \).

2- Decoding: Given an observation sequence \( O \) and an HMM \( \lambda = (A,B) \), discover the best hidden state sequence \( Q \).

3- Learning: Given an observation sequence \( O \) and the set
of states in the HMM, learn the HMM parameters A and B.

4- Re-assessing the parameters of \( \lambda \) to increase \( P(O|\lambda) \) (Jurafsky & Martin, 2006).

All previous steps are important to determine the best HMM model for speech recognition. There are effective algorithms to produce effective and accurate solution to every step of the previous steps. In order to train and use HMM in a speech recognition system, a forward-backward algorithm or the Baum-Welch re-estimation method is used.

Figure-8 illustrates the training procedure for re-estimating model parameters using the Baum-Welch method. Most recent successful statistical methods have been merged with a number of techniques that try to improve the recognition accuracy and make the recognizer more efficient with multiple talkers, background noise conditions, and channel effects. One of these techniques concentrates on conversion of the observed or measured features. The conversion encourages by the need for vocal tract length normalization. For example, minimize the effect of variations in vocal tract length of different speakers. Another conversion is known as maximum likelihood linear regression method is emerged in the statistical model to find the mismatch between the statistical characteristics of the training data and the actual unknown utterances to be recognized (Juang & Rabiner, 2006).
To train the acoustic model SphinxTrain is used. The main flow is shown in figure 9. The training tool can be trained to semi-continuous or continuous HMM models. Since the decoder that is used in these experiments isphinx3, the training is done using continuous HMM model.
Figure 9-Acoustic model training process.
1) Training CI modeling

First step in training context independent phones is generating the model definition file (mdef), which is part of model_architecture and model_parameters directories. The basic purpose of this file is to provide a unique numerical identity to each state of HMM that is going to be trained, and to provide a sequence that will be followed to construct the model parameters files. Hence, the states are indicated only by these numbers during the training. To generate CI model definition file we need to carry out multiple parameters as shown in table-1.

Table 1-Parameter setting for mk_model_gen.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonelstfn</td>
<td>Phonelist.</td>
<td>model_architecture/ArabicDigits.phonelist</td>
</tr>
<tr>
<td>Moddeffn</td>
<td>Name of the CI model definition file with full path.</td>
<td>model_architecture/ArabicDigits.ci.mdef</td>
</tr>
<tr>
<td>n_state_pm</td>
<td>Numbers of states per HMM model that will be trained.</td>
<td>3 for continuous HMM.</td>
</tr>
</tbody>
</table>

Second, generating the HMM topology file. This file contains a matrix with boolean entries, where every entry refers whether a particular transition from state is allowed in HMM or not. Third is the flat initialization of CI model parameters, which consists of four parameter files:

- **Mixture_weights** is the weights for each Gaussian in the Gaussian mixture corresponding to a state.
- **Transition_matrices** is the matrix of state transition probabilities.
- **Means** is means of all Gaussians.
- **Variances** is variances of all Gaussians.
The mixture_weights and transition_matrices are initialized using the executable `mk_flat`, which requires the following parameters:

Table 2-Parameters for mk_flat.

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>moddeffn</td>
<td>CI model definition file.</td>
<td>model_architecture/ArabicDigits.ci.mdef</td>
</tr>
<tr>
<td>topo</td>
<td>HMM topology file.</td>
<td>model_architecture/ArabicDigits.topology</td>
</tr>
<tr>
<td>mixwfn</td>
<td>File which writes the initialized mixture weights.</td>
<td>model-parameters/ArabicDigits.ci_cont_flatinitial/mixture-weights</td>
</tr>
<tr>
<td>tmatfn</td>
<td>File which writes the initialized transition matrices.</td>
<td>model-parameters/ArabicDigits.ci_cont_flatinitial/transition_matrices</td>
</tr>
<tr>
<td>nstream</td>
<td>Number of independent feature streams.</td>
<td>For continuous models is 1.</td>
</tr>
<tr>
<td>ndensity</td>
<td>Number of Gaussians modeling each state.</td>
<td>For CI models is 1.</td>
</tr>
</tbody>
</table>
Global means and variances must be computed using both executables *init_gau* and *norm*. The flat means and variances file can be created using the executable *cp_parm*, but *cp_parm* has to be run twice, once for copying the means, and once for copying the variances. *cp_parm* requires the following arguments.

**Table 3 - Parametr setting of cp_parm.**

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpopsfn</td>
<td>Copy operations map file.</td>
<td>model_architecture/ArabicDigits.cpmeanvar</td>
</tr>
<tr>
<td>igaufn</td>
<td>Input global mean (or variance) file.</td>
<td>model-parameters/ArabicDigits.ci_cont_flatinitial/globalmeans</td>
</tr>
<tr>
<td>ncbout</td>
<td>Number of phones times the number of states per HMM (ie, total number of states).</td>
<td>60</td>
</tr>
<tr>
<td>ogaufn</td>
<td>Output initialized means (or variances) file.</td>
<td>model-parameters/ArabicDigits.ci_cont_flatinitial/means</td>
</tr>
</tbody>
</table>

Fourth is CI training stage. During this stage, the previous flat initialized models are re-estimated using Baum-Welch algorithm. The re-estimation is iterated many times to get better set of models for the CI phones. Since the objective function in the iteration are maximum likelihood, making too many
iterations of model parameters will result in models that fit very closely to the training data. Generally 5-8 times iteration can get good estimates of the CI models.

To run a Baum-Welch algorithm, \textit{bw} executable file is executed. The following parameters in table -4 need to be set. After each execution of \textit{bw}, executable called \textit{norm} must be run to estimate the final model parameters, namely the means, variances, mixture-weights and transition matrices. Finally, the iterations of Baum-Welch and norm result in CI models. The model parameters are computed by norm in the final iteration are used to initialize the models for CD phones with united states.

Table 4-Parameters of bw program.

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>moddeffn</td>
<td>CI phones model definition.</td>
<td>model_architecture/ArabicDigits.ci.mdef</td>
</tr>
<tr>
<td>ts2cbfn</td>
<td>Types of HMM</td>
<td>in this case is .cont.</td>
</tr>
<tr>
<td>mixwfn</td>
<td>Names of the file where the mixture-weights from the previous iteration are stored.</td>
<td>model-parameters/ArabicDigits.ci_cont_flinitial/mixture-weights</td>
</tr>
<tr>
<td>mwfloor</td>
<td>Minimum value of the mixture weights and any number below it will</td>
<td>1e-08</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
<td>Example Path</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>tmatfn</td>
<td>Name of the file in which the transition matrices from the previous iteration are stored.</td>
<td>model-parameters/ArabicDigits.ci_cont_flattest/transition_matrices</td>
</tr>
<tr>
<td>meanfn</td>
<td>Name of the file where the means from the previous iteration are stored.</td>
<td>model-parameters/ArabicDigits.ci_cont_flattest/means</td>
</tr>
<tr>
<td>varfn</td>
<td>Name of the file where the variances from the previous iteration are stored.</td>
<td>model-parameters/ArabicDigits.ci_cont_flattest/variance</td>
</tr>
<tr>
<td>dictfn</td>
<td>Dictionary.</td>
<td>etc/ArabicDigits.dic</td>
</tr>
<tr>
<td>fdictfn</td>
<td>Filler dictionary.</td>
<td>etc/ArabicDigits.filler</td>
</tr>
<tr>
<td>ctlfn</td>
<td>Control file.</td>
<td>etc/ArabicDigits_train.fileids</td>
</tr>
<tr>
<td>part</td>
<td>After splitting the training data.</td>
<td></td>
</tr>
</tbody>
</table>

be set to the minimum value.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>npart</td>
<td>Number of parts where the training data is split.</td>
<td>1</td>
</tr>
<tr>
<td>Cepdir</td>
<td>Directory where your feature files are stored.</td>
<td>ArabicDigits/feat</td>
</tr>
<tr>
<td>Cepext</td>
<td>The extension that comes after the name of control file.</td>
<td>mfc</td>
</tr>
<tr>
<td>Lsnfn</td>
<td>Transcript file name.</td>
<td>etc/ArabicDigits_train.transcription</td>
</tr>
<tr>
<td>Accumdir</td>
<td>Intermediate directory where training result</td>
<td>bwaccumdir/ArabicDigits_buff_1</td>
</tr>
</tbody>
</table>

into N equal parts. If there are M utterances in the control file, then training can be run separately for each (M/N)th part.
is stored

<table>
<thead>
<tr>
<th>varfloor</th>
<th>Minimum variance value.</th>
<th>0.0001</th>
</tr>
</thead>
<tbody>
<tr>
<td>topn</td>
<td>Number of Gaussians.</td>
<td>1</td>
</tr>
<tr>
<td>abeam</td>
<td>Forward beam width.</td>
<td>1e-90</td>
</tr>
<tr>
<td>bbeam</td>
<td>Backward beam width.</td>
<td>1e-10</td>
</tr>
<tr>
<td>age</td>
<td>Automatic gain control.</td>
<td>None</td>
</tr>
<tr>
<td>cmn</td>
<td>Cepstral mean normalization</td>
<td>current</td>
</tr>
<tr>
<td>varnorm</td>
<td>Normalize variance or not.</td>
<td>no</td>
</tr>
<tr>
<td>meanrees t</td>
<td>Re-estimate mean or not.</td>
<td>yes</td>
</tr>
<tr>
<td>varreest</td>
<td>Re-estimate variance or not.</td>
<td>yes</td>
</tr>
<tr>
<td>passvar</td>
<td>Use means from previous iteration in the variance re-estimation</td>
<td>yes</td>
</tr>
</tbody>
</table>
or not.

<table>
<thead>
<tr>
<th>tmatreest</th>
<th>Re-estimate transition matrices or not.</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ceplen</td>
<td>Length of basic feature vector.</td>
<td>13</td>
</tr>
</tbody>
</table>

2) Training CD untied models

First, generate a model definition file for all the triphones occurring in the training set. This is done by running the executable file `mk-mdef-gen`.

Next step in CD united training is flat initialization of CD united model parameters. First, the model parameter files corresponding to the CD united model definition file are constructed. Then, means, variances, transition matrices and mixture weights files are generated. For each file, the values from corresponding CI model parameters file are copied. Each state of a particular CI phone contributes to the same state of the same CI phone in the Cd-untied model parameter file. In addition, each state of a particular CI phone contributes to the same state of all the triphones of the same CI phone in the CD united model parameter file. To do this the executable `init_mixw` is run with the following arguments in table -5.
Table 5-Parameters of init_mixture.

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>src_moddeffn</td>
<td>CI model definition file.</td>
<td>model_architecture/ArabicDigits.ci.mdef</td>
</tr>
<tr>
<td>src_ts2cbfn</td>
<td>Types of HMM</td>
<td>which is .cont in this case.</td>
</tr>
<tr>
<td>src_mixwfn</td>
<td>CI mixture-weight file</td>
<td>model-parameters/ArabicDigits.ci_cont/mixture-weights</td>
</tr>
<tr>
<td>src_meanfn</td>
<td>CI means file.</td>
<td>model-parameters/ArabicDigits.ci_cont/means</td>
</tr>
<tr>
<td>src_varfn</td>
<td>CI variances file</td>
<td>model-parameters/ArabicDigits.ci_cont/variances</td>
</tr>
<tr>
<td>src_tmatfn</td>
<td>CI transition matrix file.</td>
<td>model-parameters/ArabicDigits.ci_cont/transition_matrices</td>
</tr>
<tr>
<td>dest_moddeffn</td>
<td>United CD model definition file.</td>
<td>model-architecture/ArabicDigits.united.mdef</td>
</tr>
<tr>
<td>dest_ts2cbfn</td>
<td>Types of HMM</td>
<td>which is .cont in this case.</td>
</tr>
<tr>
<td>dest_mixwfn</td>
<td>United CD mixture-weight file.</td>
<td>model-parameters/ArabicDigits.ci_cont_united/mixture_weights</td>
</tr>
<tr>
<td>dest_meanfn</td>
<td>United CD means file.</td>
<td>model-parameters/ArabicDigits.ci_cont_united/means</td>
</tr>
<tr>
<td>-dest_varfn</td>
<td>United CD</td>
<td>model-</td>
</tr>
</tbody>
</table>
Final step in training CD untied models is to train the CD united models. The Baum-Welch forward-backward algorithm is used for this purpose. As explained in CI model, each iteration consists of bw buffers that is generation by executing bw on the training data. In order to compute the final parameters at the end of each iteration, the executable norm is run. Following this step is the normalization step where the norm executable must be executed for this purpose. The typical iteration is normally between 6-10 iterations.

3) Building decision tree

Decision trees are used to decide which of the HMM states of all the triphones (seen and unseen) are similar to each other, so that data from all these states are collected together and used to train one global state, which is called a "senone". One decision tree is built for each state of each phone.

The decision trees require the CD-untied models and a set of predefined phonetic classes. These classes or questions share some common properties. Therefore, they are used to partition the data at any given node of a tree. Each question produces one partition, and the question that has the best partition is used to partition the data at that node. There is only one single file for all linguistic questions. When the linguistic question is generated, each CI phone
presents in phonelist except the filler and SIL phone in the phonelist must have decision tree.

Decision tree building processes are:

a) Pruning the decision trees once they are built in order to have as many leaves as the number of senones that required for training.

b) Creating the CD tied model definition file once the trees are pruned. This file contains all the triphones which are seen during training, and has the states corresponding to these triphones identified with senones from the pruned trees.

4) Initializing and training CD tied models

Single Gaussian distribution or a mixture of Gaussian distributions is used to model HMM states. The number of Gaussians in a mixture-distribution must be even, and a power of two. For example, to model HMM states by a mixture of 8 Gaussains, one Gaussian per state is first trained. Then, each Gaussian distribution is split into two by perturbing its mean. The produced two distributions are used to initialize the training for 2 Gaussian per state models. Further these are perturbed to initialize for 4 Gaussians per state models and a further split is done to initialize the 8 Gaussian per state models. Therefore, the CD-tied training for models with $2^N$ Gaussians per state is done in $N+1$ steps. Each of these $N+1$ steps consists of:

a) Initialization of the 1 Gaussian per state models. First, the model parameters form the CI model parameters are copied into a location in the CD tied model parameters files. The means, variances, transition matrices and mixture weights files are created. The each state of CI phone contributes to the same state of the same phone in the CD tied model parameters file. Furthermore, to the same state of all triphones of the same CI phone in the CD tied model parameters file.

b) Iterations of Baum-Welch using $bw$ followed by norm.

c) Gaussian splitting (not done in the $N+1$ th stage of CD-tied training)
using *inc-comp* (SphinxTrain Documentation, n.d.).

### 2.1.3 Language model

Methods of language modeling can be statistical based or rule based. Statistical based language model is widely used, which uses N-gram algorithm for modeling. N-gram is statistical model that predicts the next word from previous N-1 words. In other words, computes the probability of a sequence of words.

The prior probability of a word series \( W = w_1, w_2, w_3, \ldots, w_k \) in Eq. (2) is provided by:

\[
P (w) = \prod_{k=1}^{K} P (w_k | w_{k-1}, \ldots, w_1)
\]

This assumption is called a Markov which assumes that the probability of a word depends only on the previous word. Hence, the bigram can be generalized (looks one word into the past) to the trigram (looks two words into the past) and thus to the N-gram (looks N-1 word into the past).

The conditioning word history in previous Eq. (7) is amputated to N-1 words to form an N-gram language model for large vocabulary recognition.

\[
P (w) = \prod_{k=1}^{K} P (w_k | w_{k-2}, w_{k-3}, \ldots, w_{k-N+1})
\]

\( N \) in above equation is in the range 2-4.

N-gram probabilities are predicted from set of training text. This is done by counting N-gram occurrences to compute likelihood (ML) parameter estimates. For instance, suppose \( C (w_{k-2}, w_{k-1}, w_k) \) is the number of occurrences of the three words \( w_{k-2}, w_{k-1}, w_k \) and \( C (w_{k-2}, w_{k-1}) \) is the number of occurrences of the two words \( w_{k-2}, w_{k-1} \), then (Gales & Young, 2007).
For the general case of ML N-gram parameter estimation:

\[
P(w_k | w_{k-1}, w_{k-2}) \approx \frac{C(w_{k-2}, w_{k-1}, w_k)}{C(w_{k-2}, w_{k-1})}
\]  

(9)

2.1.4 Decoding

The process of decoding trained acoustic model and language model is known as a search process since it finds a sequence of words \( \hat{W} \) whose acoustic and language models best match the acoustic signal represented by the input feature vector sequence (Indurkhya & Damerau, 2010). For decoding, three information sources must be available:

1- An acoustic model with an HMM for each unit (phoneme or word).
2- A dictionary, typically a list of words and the phoneme sequences they consist of.
3- A language model with word or word sequence likelihoods.

Knowing which words can be spoken is a mandatory condition for decoding. These words are listed in the dictionary (lexicon), together with the according phoneme sequence. The acoustic model has a probability density function that is a mixture of Gaussians and gives likelihood for each observed vector \( P(O|W) \).

A language model is not an absolute requirement for decoding but increase word accuracy. In case of digit recognition 0-9, it is acceptable to consider all words equally likely.

In decoding process, search is done to find the word \( \hat{W} \) that fits best to the observation \( O \) as given in Eq. (2). With \( P(W) \) coming from the language model and \( P(O|W) \) calculated from the sequence of phonemes in the word as defined by the dictionary. When the space of possible state sequence is large, it is not
possible to compute the probabilities of all existing paths through the state network: for \( N \) states and \( T \) observation, the complexity is \( O(N^T) \). To find the most likely sequence of hidden states, the Viterbi search algorithm is used which is based on dynamic programming methods (Gruhn, Minker, & Nakamura, 2011).

### 2.2 Evaluating the Performance of ASR

A key issue in speech recognition is how to measure the performance of the system. A commonly metric is the word error rate (WER). For the isolated words tasks, three errors must be taken into account. The first one is word substitution which occurs when an incorrect word is recognized in place of the correctly spoken word. The second error is word deletion (some spoken words are not recognized). Finally, word insertion error means extra words are not in the spoken sentence might be inserted. The definition of word error rate based on the three errors is

\[
\text{WER} = 100 \% \times \left( \frac{S + D + I}{|W|} \right)
\]

(11)

Where \( S \) is the number of substitutions, deletions \( D \), insertions \( I \), and \( |W| \) is the number of words in the sequence of word \( W \) (Adami, n.d.).
Chapter 3

Adapting the acoustic model

3. Overview

According to the previous chapter building a typical ASR system includes four main components: feature extraction, acoustic model training, language model construction, and decoding. After constructing all these components it is critical to evaluate the performance of ASR system. The performance of ASR system might degrade, and the source of this degradation can be grouped into environmental noise, different channels, and speaker variability (Merino, 2002). The speaker variability factor is the toughest one to eliminate, particularly when the automatic speech recognition systems are trained by native speakers and later are used by non-native speakers (Lee et al., 2000; Zheng et al., 2005). This is due to the vocal tract, accent, dialect, cultural and emotional voice characteristics that each speaker has. As result, there are many different studies have been proposed to improve automatic speech recognition systems used by new speakers (non-native speakers or new speakers speak the same language).

Number of important issues related to the application of Bayesian learning techniques to speaker adaptation are investigated by (Lee & Gauvaint, 1993). They showed that the seed models required to build previous densities can improve the performance of both speaker dependent and speaker independent speech recognition systems.

Fung et al. (2000) worked on principal mixture speaker adaptation for improved continuous speech recognition. They introduced a method known as a principle mixture speaker adaptation. This method reduced HMM complexity by choosing only the principle mixtures corresponding to particular speaker’s characteristics. In addition to recognition accuracy improvement by 31.8% and recognition speed reduction by 30% when compared to full mixture speaker adaptation models.
Wang et al. (2003) explored how the acoustic models can be adapted to better handle the non-native speech by using a multilingual recognizer to do the decoding on non-native speech. They tested on a conversational speech task. To do speaker adaptation, they used Maximum Likelihood Linear Regression (MLLR) and Maximum and A-Posteriori (MAP) with multiple test sets to see how speaker variability will contribute to the recognition performance. Furthermore, they explored how interpolation can be useful in building acoustic models for non-native speech recognition. Additionally the Polyphone Decision Tree Specialization was used to see whether it can also help to improve the performance on non-native speech recognition. Later in 2005, Bartkova and Jouvet showed that error rate can be significantly reduced when standard acoustic models of phonemes are adapted using speech data from other languages. In their case, the acoustic model of French phonemes is adapted with speech data from three other languages: English (US and UK), German, and Spanish. Their results obtained for 11 language groups of speakers in their outputs. The highest error rate reduction of 50% was obtained on English native speakers.

Also in 2005 Fakotakis worked on the adaptation of standard Greek speech recognition systems to work with Cypriot dialect by using Hidden Markov Models toolkit (HTK) toolkit, MLLR, MAP, and combined MLLR and MAP techniques. He considered Cypriot Greek as a variation of standard Greek with the same set of phonemes. He used utterances read. He used utterances from 500 native Greek speakers, 550 from them are used for training phase and 50 as testing set. The system performance degraded when trained using pure Cypriot Greek.

3.1 Adaptation techniques

A number of methods for handling non-native speech and compensate for speaker variability in speech recognition have been proposed. All these adaptation methods reduce the differences between the acoustic model and the selected speaker. This is done by sampling speech data from the new speaker
and updating the acoustic model according to the features that are extracted from the speech (Woodland, 2001). The adaptation can be either supervised if the transcription of the speech data is known or unsupervised if the transcription is unknown. When the adaptation data is available at once and is used to adapt the final system during a single run, the adaptation is called a static mode. While in the dynamic adaptation mode, the data is acquired in parts and the system is continuously adapted over time (Woodland & Leggetter, 1995).

One of most used adaptation techniques is presented in the following section: Maximum likelihood linear regression (MLLR).

### 3.1.1 Maximum likelihood linear regression (MLLR)

It belongs to a family of adaptation techniques that computes set of linear transformations for the mean of Gaussian mixture HMM system in order to minimize the mismatch between an initial model and adaptation data. These transformations shift Gaussian mean parameters in the initial system in order to make every state in the HMM generate the adaptation data (Selouani & Alotaibi, 2011). A linear transformation is estimated as follows:

\[ \hat{\mu} = A \mu + b \]  \hspace{1cm} (12)

Where:
- \( A = n \times n \) matrix.
- \( n \) is dimensionality of the observations (data) which is 39 in case of an MFFC observation vector.
- \( b = \) dimensional vector.

Eq. (12) can be rewritten as:

\[ \hat{\mu} = W \xi \]  \hspace{1cm} (13)
Where:

\[ W = n \times (n+1) \text{ matrix.} \]

\[ \xi = \text{the extended mean vector that can be defined as follow:} \]

\[ \xi^T = [1 \mu_1 \mu_2 \ldots \ldots \mu_n] \]

(14)

The Expectation-Maximization (E-M) algorithm can be used to estimate the matrix \( W \) in order to maximize the likelihood of the adaptation data. Usually, for HMMs this task is performed by the BaumWelch algorithm, also known as the forward-backward algorithm. MLLR technique needs only a small amount of adaptation data to predict a global transformation matrix \( W \). Then the global transformation matrix \( W \) can be used in Eq. (13) to transform Gaussian means of phonemes or triphones that haven’t even been observed in the adaptation data. The transformation matrices can be chosen to be diagonal:

\[
\begin{bmatrix}
M_1 & 0 & 0 \\
0 & \ddots & 0 \\
0 & 0 & M_n
\end{bmatrix}
\]

The observation vector consists of three partitions: MFCC Features, and their first and second derivative. Consequently, the block diagonal transforms usually consist of three quadratic matrices \( M_i \).

MLLR also transform the Gaussian variances parameters. The transformation of the covariance matrix \( \sum \) is stated as:

\[ \sum \approx H\sum H^T \]

(15)

Both Gaussian means and variances are transformed independently by using Eq. (13) and Eq. (15). For each parameter set, separate transformation matrices
W and H are estimated. In the constrained transform case, the means and variances are set to use the same transformation matrix $A_C$. This is known as constrained MLLR (cMLLR).

$$\hat{\mu} = A_C \mu - b_C$$  \hspace{1cm} (16) \\
$$\Sigma = A_C^T \Sigma A_C$$  \hspace{1cm} (17)

Different transformation matrices can be tied to Gaussians that are close to each other in acoustic space instead of using the same global transformation matrix W for all Gaussian models. These transformation matrices are arranged into regression classes (Woodland, 2001). Regression classes can be either fixed or dynamic. In the fixed regression classes, the classes’ definition is predetermined by assessing the amount of adaptation data available. Figure 10 shows the adaptation framework for two fixed regression classes. The optimal number of regression classes is proportional to the amount of adaptation data. For instance, a division into Gaussian models representing vowel and consonant sounds could be made. Consequently, two transformation matrices for each group can be estimated. The mixture components are divided into an optimal number of different regression classes after determing the size of the adaptation data. The class definitions and what mixture components belong to which class specified in advance. Lastly. In order to maximize the likelihood of the adaptation, the transformation matrices $W_i$ are estimated.
Figure 10-Adaptation framework for two fixed regression classes. Each regression class has mixture component. In order to maximize the likelihood of the adaptation, the transformation matrices \( W_i \) are estimated.

However, if the amount of adaptation data is equally distributed among the classes, then fixed regression classes will work. Nevertheless, when the classes are assigned with insufficient amount of adaptation data, the estimates will be poor. Hence, determination of the content distribution of the adaptation data and making the division into regression classes based on this will be helpful. This means that the regression classes are defined dynamically based on the type of adaptation data that is available.

As depicted in figure- 11, the dynamic regression classes with their mixture elements are organized into a tree. The root node indicates to the global transform matrix where all mixture elements are combined. Leaves of the regression tree represent individual mixture elements. The mixture components
are merged into groups of similar elements based on a distance measure between elements at the higher levels. The purpose of regression tree is to determine which classes have enough amounts of data in order to estimate the transformation matrix properly.

A search is made through the tree starting from the top level through the leaves. Then, a transformation matrix estimated is done at the lowest level of the tree for which regression class there is sufficient data. This allows adaptation data to be used in more than one regression class and it ensures the mixture components are updated with the most specific transformation matrix (Woodland & Leggetter, 1995).

![Figure 11-Regression class tree.](image_url)
Chapter 4

Introduction about Arabic language

4. The Arabic language

The Arabic language is the largest and oldest Semitic language in the world, and it has various differences with other European languages such as English. The basic difference is the pronunciation of 10 digits from zero to nine. Arabic language is considered one of the six official languages of the United Nations. In addition Arabic is the official language of the Arab world that consists of 22 countries, and also used in many other languages such as Persian and Urdu. Based on the number of first language speakers, Arabic is ranked as the sixth most spoken languages. Moreover, Arabic language has more than 250 million first language speakers.

The official linguistic of Arabic language is Modern Standard Arabic which is used and taught in schools, universities, media, offices, moving subtitling, books, news broadcast and formal speech. In addition, Modern Standard Arabic is used in writing all Arabic text resources. MSA is considered the second language of all Arabic speakers. Therefore, in order to target a broad Arabic audience, most of TV, radio, and news broadcast use it.

Arabic sentences are written from right to left, some letter in the sentences might change in shape depending on their position in a word (Elmahdy, Gruhn & Minker, 2012).

4.1 Arabic alphabet

Standard Arabic language has 35 main phonemes, 28 of them are consonants phonemes, and the rest are vowels phonemes. If we compare the Arabic language with the English language, we will notice that Arabic has
fewer vowels than English. American English has at least 12 vowels, in contract to Arabic that has three long and three short vowels (Satori et al., 2007).

Arabic is characterized by two distinctive classes emphatic and pharyngeal phonemes. These kinds of classes can be found in Semitic languages like Hebrew (Elmahdy et al., 2012). Arabic digits from zero to nine are considered as polysyllabic words except zero which is considered as a monosyllable word (se`fr, wa`-he`d, _aa`th-n_a`yn, tha`-la`-th-a`h, _aa`r-ba`-_aa`h, kha`m-sa`h, se`t-ta`h, su`b-_aa`h, tha`-ma`-ni-ye`h, and te`s-_aa`h).

The only allowed syllables in Arabic are CV, CVC and CVCC, where V is considered as a long or short vowel while C is considered as a constant. Consequently, the CVCC pattern is only permitted at the end of a word and all Arabic utterances can only begin with a constant phoneme (Elmahdy et al., 2012).

Arabic syllables can not start with vowels and must contain at least one vowel. Arabic syllables are categorized as long or short. CVC and CVCC types are long while CV type is a short. Additionally, syllables also classified as open or closed. The close does not end with a vowel, and the open syllable ends with a vowels. A vowel in Arabic language always forms a syllable nucleus and there are various syllables in it. Table-6 shows the pronunciation of Arabic digits, IPA representation, type of syllable, and number of syllables in each Arabic digit (Alotaibi, 2005).

Table 6-Arabic digits from zero to nine.

<table>
<thead>
<tr>
<th>Digit</th>
<th>Arabic Writing</th>
<th>Pronunciation</th>
<th>Syllables</th>
<th>IPA Representation</th>
<th>No. of Syllables</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>سَفْر</td>
<td>_aa`f</td>
<td>CVCC</td>
<td>_aa`f</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>وَاحِد</td>
<td>wa`-h1d</td>
<td>CV-CVC</td>
<td>wa`-h1d</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>ثَانِين</td>
<td>th-n_ayn</td>
<td>CVC-CVC</td>
<td>?1h-n_ayn</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>ثَلَاث١</td>
<td>th-la-th1h</td>
<td>CV-CVC</td>
<td>th-la-th1h</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>أَرْبَع١</td>
<td><code>ar-ba</code>-a`h</td>
<td>CVC-CVC</td>
<td><code>ar-ba</code>-a`h</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>خَمْسَة</td>
<td>khm-sa`h</td>
<td>CVC-CVC</td>
<td>xam-sa`h</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>سَبْعَة</td>
<td>sib-<code>a</code>h</td>
<td>CVC-CVC</td>
<td>sib-<code>a</code>h</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>ثَامِن١</td>
<td>tham-n_1n</td>
<td>CVC-CVC</td>
<td>tham-n_1n</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>تَمْمٍ</td>
<td>tamm_1n</td>
<td>CV-CV-CVC</td>
<td>tamm_1n</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>ثَلَاث١</td>
<td>th-la-th1h</td>
<td>CVC-CVC</td>
<td>th-la-th1h</td>
<td>3</td>
</tr>
</tbody>
</table>

49
4.2 Description of Arabic digits

The only monosyllable digit word is zero that has long syllable CVCC, where C means consonant and V means long or short vowel. Zero begins with long consonant /$\text{\v{S}}$/ which is a fricative unvoiced non-emphatic consonant followed by short vowel /i/ then two consonants end the digit are /f/ and /r/ one of them is liquid voiced non-emphatic consonant. The duration of zero is the shortest among all other Arabic digits.

Digit one has two syllables CV-CVC. CV syllable begins with the semivowel /w/ and ends with the long vowel /a:/, While, CVC starts with the consonant /$\text{\f{H}}$/ which is a pharyngeal fricative unvoiced sound then followed by short vowel /i/ and ends with the stop voiced non-emphatic sound /d/. These two syllables make this digit relatively long in duration.

Digit two consists of two syllables CVC-CVC. CVC syllable starts with a glottal unvoiced stop consonant /?/ and the second phoneme is the short vowel /i/ followed by the inter-dental fricative unvoiced stop /$\text{\f{H}}$/, The second syllable begins and ends with the same consonant /n/ which is voiced nasal sounds in Arabic. While the second phoneme is long vowel /i:/, It is obvious that the middle part in two syllables is voiced sound but both ends are unvoiced consonants.

Digit three has three syllables CV-CV-CVC and has long duration. First and last CV-CVC syllables begin with the consonant /$\text{\f{H}}$/ which is an inter-dental fricative unvoiced sound. The second CV syllable begins with liquid voiced consonant /l/ and ends with the long vowel /a:/, The first syllable ends with the short vowel /a/. The last syllable ends with the glottal unvoiced fricative preceded by the short vowel /a/.

Digit four has three syllables CVC-CV-CVC where the first and last syllables are of the same type. The first syllable CVC has stop unvoiced non-emphatic glottal /?/, short vowel /a/, and liquid emphatic alveolar /r/ phonemes. Whilst, the second syllable CV has voiced non-emphatic bilabial /b/ and short /a/ phonemes. The last syllable CVC has fricative voiced non-emphatic
pharyngeal /?/, short vowel /a/, and fricative unvoiced non-emphatic glottal /h/ phonemes. Digit four consists of same three vowels /a/ in all syllables.

Digit five consists of two syllables CVC-CVC. Two syllables are identical. The first one has fricative unvoiced non-emphatic uvular /x/, short vowel /a/, and nasal voiced emphatic /m/ phonemes. In contract to second syllables that consists of fricative unvoiced non-emphatic alveolar /s/, short vowel /a/, and fricative unvoiced non-emphatic glottal /h/ phonemes. Both of these syllables have vowel /a/ in the middle. Because of the nasal sound /m/ in the middle of digit five, it has low energy signal.

Digit six also consists of two identical syllables CVC-CVC. In the first syllables, there are fricative unvoiced non-emphatic alveolar /s/, short vowel /i/, and stop unvoiced non-emphatic alveolar /t/ sounds. While, in the second syllable there are stop unvoiced non-emphatic alveolar /t/, short vowel /a/, and fricative unvoiced non-emphatic glottal /h/ phonemes. Since all consonants in it are unvoiced, this digit is mostly unvoiced.

Digit seven has two syllables CVC-CVC. The first syllable consists of fricative unvoiced non-emphatic alveolar /s/, short vowel /a/, and stop unvoiced non-emphatic bilabial voiced /b/. While, the second has fricative voiced non-emphatic pharyngeal /?/, short vowel /a/, and fricative unvoiced non-emphatic glottal /h/ phonemes. The vowels in both syllables are the short vowel /a/.

Digit eight has CV-CV-CV-CVC syllables. In the first syllable there is a fricative unvoiced non-emphatic inter-dental /h/ and the short vowel /a/, and a nasal voiced non-emphatic bilabial /m/ and long vowel /a:/ in the second syllable. The third syllable has a nasal voiced non-emphatic alveolar /n/ and long vowel /i/. In the fourth syllable together with the semi-vowel, unvoiced non-emphatic voiced palatal /j/, long vowel /a/ and fricative unvoiced non-emphatic glottal /h/. Because this digit has four syllables, it is the longest utterance among Arabic digits.

Digit nine consists of two syllables CVC, where the first one has stop unvoiced non-emphatic alveolar /t/ followed by the short vowel /i/ and ends with the fricative unvoiced non-emphatic alveolar phoneme /s/. The second
syllables begins with the fricative voiced non-emphatic pharyngeal phoneme /ʔ/ and followed by the short vowel /a/ and finally this syllable ends with the fricative unvoiced non-emphatic phoneme /h/ (Alotaibi, 2005). Figure-12 shows the Waveforms and spectrograms of all Arabic digits for speaker 12 during trial 1.
Figure 12-Waveforms and spectrograms of all Arabic digits for Speaker 12 during trial 1.
4.3 Spoken digit recognition

One of the most challenging tasks in the speech recognition field is automatic recognition of spoken digits. This kind of recognition is important for many applications especially that need digits as input like airline reservations and automatic directories to retrieve or send information, etc.

In general, automatic speech recognition researchers were interested in spoken digits such as English and Japanese (Alotaibi, 2005). For instance, Cosi et al built and tested a high performance telephone bandwidth speaker independent continuous digits recognizer. The system gave 99.22% word accuracy, 92.62% sentences accuracy, and it was based on artificial neural network.

Few numbers of researchers have been conducted on Arabic digits recognition. In addition, most of previous work on Arabic Automatic Speech recognition was done on either Modern Standard Arabic or Egyptian Colloquial Arabic, and by training the system using one of Romanized format or Standard Arabic script (Abu Zitar & Hyassat, 2006).

4.4 Arabic Speech Recognition studies

Two of the previous Arabic alphadigits recognizers were designed by Hagos and Abdullah in 1985. Hagos built a speaker independent Arabic digit recognizer that is based on the LPC parameters for feature extraction and log likelihood ratio for similarity measurements. The system used template matching for input utterances. While, Abdullah designed different Arabic digits recognizer which used positive slope and zero crossing duration as the feature extraction. This system has 97% accuracy rate. Later, another automatic Arabic vowel recognition system was built by Al-otaibi. He studied the nature of Arabic language syllables (Alotaibi, 2005).

Most recent Arabic speech recognition has been addresses by Satori et al., 2007 by using sphinx tools for recognition of isolated Arabic digits. The corpus
was collected from six speakers (3 males and 3 females) and the system has 86.66% accuracy. Another tool is developed to construct Arabic pronunciation dictionaries by Hiyassat, 2007 in his PhD thesis. The produced dictionaries are based on a small Modern Standard Arabic corpus which contains digits. Another technique is used by Elshafei et al., 2008 to generate Arabic phonetic dictionaries for a large vocabulary (a 5.5 hours corpus of broadcast news) speech recognition system. The system used classic Arabic pronunciation rules, morphologically driven rules and Modern Standard Arabic pronunciations rules. This system achieved high accuracy of 88.29%.

A major work is done by Abu Zitar & Hyassat, 2006, where they introduced the first SPHINX-IV based Arabic recognizer that is based on three corpus, namely the Holly Qura'an, the command and control corpus and Arabic digits corpus. Following in 2010 Alghamdi et al. developed a system to recognize an isolated whole word speech. The data in training and testing phases is taken from the telephony Arabic speech corpus, SAAVB. The recognition system had 93.72% overall correct rate. Recently, The International Arab Journal of Information Technology in 2012 published a paper that proposed a design of speaker independent continues automatic Arabic speech recognition system based on CMU sphinx tool and 515 sentences recorded by 50 speakers.

### 4.5 Arabic Dialects

As mentioned by Zaidan in his research in the University of Pennsylvania 2015, the only written form of the Arabic language is Modern Standard Arabic. Moreover, Modern Standard Arabic is the only variety that is standardized and taught in schools. Nevertheless, there are a variety of spoken regional dialects of Arabic which differ from Modern Standard Arabic and used basically for day to day spoken communication, dealing, blogs, forums and chat rooms. Even though dialectal Arabic stays absent from written communication compared with Modern Standard Arabic, it is possible to construct speech recognizer that is able to recognize both Modern Standard Arabic and different Arabic dialects.
The regional dialects variation of Arabic are divided into groups as following:

**Levantine**: a group of dialects which have different pronunciations and intonations, but they are equivalent in written form.

**Gulf**: it is the closest dialect to Modern Standard Arabic because the Modern Standard Arabic evolved from Arabic in Gulf region.

**Egyptian**: because of the popularity of Egyptian television and movie industry, Egyptian dialect is the most widely understood dialect.

**Maghrebi**: significantly affected by Berber and French languages and difficult to understand by speakers from other areas in the Middle East.

**Iraq**: Even though it has distinguishing features of prepositions, it is considered one of dialects.
Chapter 5

Development of isolated Arabic digits ASR system based on CMU Sphinx

5.1 Different isolated Arabic digits speech recognition systems

In this thesis three different isolated Arabic digits speech recognition systems are built, following sections and figure-13 outline the differences between these systems and detailed phases for construct and train different acoustic models and decoding stage.

a) Speaker independent system

Building an acoustic model of spoken alpha digits with accurate speaker independent system is considered to be difficult task due to similarity among certain groups of letters and digits (Loizou & Spanias, 1996). During the training phase just ten native Arabic speakers (one through ten) were used. The total training set consists of 1000 tokens (10 speakers × 10 repetitions × 10 digits). During the test phase, two test sets are used. Each test set has 200 tokens (2 speakers × 10 repetition × 10 digits). In test1 set only speakers 11 and 12 were used, and speakers 13 and 14 are used in test2 set. The acoustic model produces in this system is called (SI) for later references.

b) Speaker dependent system

In the Speaker dependent recognition system, the training set consists of first and second repetition of each digit uttered by all native Arabic speakers. Thus, the total sample dedicated for the training phase was 600 tokens (30 speakers × 2 repetitions × 10 digits). For the testing phase, all 3000 tokens uttered by 30 speakers were used. This indicates that the training data set is a subset of the testing data set. The acoustic model produces in this mode is called (SD) for later references.
c) Native Arabic speakers system

In this system only 80% of Arabic speakers were used for the training purpose. Thus, the total number of tokens considered for training was 2400 tokens (24 speakers× 10 repetitions ×10 digits). For testing mode, two test sets are used. The rest of Arabic speakers 20% were used as test1 set; therefore test1 has 600 tokens (6 native speakers× 10 repetitions ×10 digits). Also, test2 has 600 tokens (6 non native speakers× 10 repetitions ×10 digits). The acoustic model produces in this system is called (NA) for later references.

In order to define, train, and build different acoustic models for the above systems based on CMU Sphinx tool, the following steps must be followed (Juang & Rabiner, 2006).
Figure 13 - Structure of isolated Arabic digits speech recognition system.
5.1.1 Data preparation

According to (Training Acoustic Model for CMUSphinx, n.d.) first is preparing the corpus which is a collection of unit sounds defined by certain word in the vocabulary. The corpus was created from all 10 Arabic digits (zero to nine). A number of 30 Arabic native speakers were asked to utter all digits 10 times. Hence, the database consists of 10 repetitions of every digit produced by each speaker. Depending on this, the database for native Arabic speakers consists of 3000 tokens. In addition, a number of 20 non-native Arabic speakers were asked to utter all digits 10 times. Depending on this, the database for non-native Arabic speakers consists of 2000 tokens. During the recording session, each utterance was played back to ensure that the entire digit was included in the recorded signal. Table 7 shows recording system parameters.

Table 7-Recording system parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling rate</td>
<td>16 kHz, 16 bits</td>
</tr>
<tr>
<td>Wave format</td>
<td>Mono, wav</td>
</tr>
<tr>
<td>Corpus</td>
<td>Isolated 10 Arabic digits (zero to nine)</td>
</tr>
<tr>
<td>Arabic native speakers</td>
<td>30 males</td>
</tr>
<tr>
<td>Non native Arabic speakers</td>
<td>20 males</td>
</tr>
<tr>
<td>Repetitions</td>
<td>10 times</td>
</tr>
<tr>
<td>Window type and size</td>
<td>Hamming, 256</td>
</tr>
<tr>
<td>Window step size</td>
<td>65</td>
</tr>
<tr>
<td>Pronunciation</td>
<td>MSA Arabic.</td>
</tr>
</tbody>
</table>

Second, the recorded corpus is divided into two sets the training corpus and the test corpus depending on the system run mode. Training corpus is used to
prepare training database that contains information required to extract statistics from the speech in form of the acoustic model as shown in figure-14.

The file structure for the database is shown in figure-15 and table-8:

Third, we must tell the trainer which unit sounds it should learn the parameters of, and at least their order in every speech signal in the training database. A transcript file provides this information, where a series of words and non speech sounds are written according to their order in a speech signal, followed by a tag to join this order with the corresponding speech signal.

In order to derive the order of sound units associated with each signal, the trainer looks into two dictionaries files one is called dictionary file that connects every word to a series of sound units (or sub word units), and the another is
called the filler file that connects non speech sounds to corresponding non speech or speech like sound units. Following is the description and the format of some files:

Figure 15-Structure of database (ArabicDigits) folder.
Table 8-The purpose of each folder/file in the database (ArabicDigits).

<table>
<thead>
<tr>
<th>Folder/File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>etc folder</strong></td>
<td></td>
</tr>
<tr>
<td>ArabicDigits.dic</td>
<td>Phonetic dictionary file</td>
</tr>
<tr>
<td>ArabicDigits.phone</td>
<td>Phoneset file</td>
</tr>
<tr>
<td>ArabicDigits.lm.DMP</td>
<td>Language modeling</td>
</tr>
<tr>
<td>ArabicDigits_train.filelds</td>
<td>List of files for training</td>
</tr>
<tr>
<td>ArabicDigits_trian.transcription</td>
<td>Transcription for training</td>
</tr>
<tr>
<td>ArabicDigits_test.filelds</td>
<td>List of files for testing</td>
</tr>
<tr>
<td>ArabicDigits_test.transcription</td>
<td>Transcription for testing</td>
</tr>
<tr>
<td><strong>wav folder</strong></td>
<td></td>
</tr>
<tr>
<td>Native Arabic folder</td>
<td>Contains all native Arabic</td>
</tr>
<tr>
<td></td>
<td>speakers wave files. For example</td>
</tr>
<tr>
<td></td>
<td>Speaker1_repetition1.wav</td>
</tr>
<tr>
<td></td>
<td>Speaker1_repetition2.wav</td>
</tr>
<tr>
<td>Non-native Arabic folder</td>
<td>Contains all non-native Arabic</td>
</tr>
<tr>
<td></td>
<td>speakers wave files. For example</td>
</tr>
<tr>
<td></td>
<td>Speaker1_firstdigit.wav</td>
</tr>
</tbody>
</table>

**Phonetic Dictionary (ArabicDigits.dic)**

In ArabicDigits.dic dictionary file, I define each digit vocabulary from the corpus with the phone unit as shown in table-9.

Table 9-ArabicDigits.dic file structure.

<table>
<thead>
<tr>
<th>Word</th>
<th>Phone unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEFR</td>
<td>S EH F R</td>
</tr>
<tr>
<td>SEFR(2)</td>
<td>S F R</td>
</tr>
<tr>
<td>WAHID</td>
<td>W AA HH D</td>
</tr>
<tr>
<td>ETHNAYN</td>
<td>AE TH N AY N</td>
</tr>
<tr>
<td>ETHNAYN(2)</td>
<td>AA T N EY AH N</td>
</tr>
<tr>
<td>Arabic Digit</td>
<td>Pronunciation</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
</tr>
<tr>
<td>THALATAH</td>
<td>TH AA L AA TH AH</td>
</tr>
<tr>
<td>THALATAH(2)</td>
<td>T AA L AA T AH</td>
</tr>
<tr>
<td>ARBAAH</td>
<td>AA R B AA AH</td>
</tr>
<tr>
<td>KHAMSAH</td>
<td>K AA M S AH</td>
</tr>
<tr>
<td>SETTAH</td>
<td>S T AA H</td>
</tr>
<tr>
<td>SETTAH(2)</td>
<td>S T AA</td>
</tr>
<tr>
<td>SABAHH</td>
<td>S B AA</td>
</tr>
<tr>
<td>SABAHH(2)</td>
<td>S AA B AA AH</td>
</tr>
<tr>
<td>THAMANEYAH</td>
<td>T M AA N EY AH</td>
</tr>
<tr>
<td>THAMANEYAH(2)</td>
<td>TH AA M AA N EY AH</td>
</tr>
<tr>
<td>TESAH</td>
<td>T S AA</td>
</tr>
</tbody>
</table>

**Filler Dictionary (ArabicDigits.filler)**

This file contains user's definition of any non speech events (background noise like breath, hmm or laugh) and maps them to user defined phones. It also can contain just silences. This dictionary should have at least the following entries:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Pronunciation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;s&gt;</td>
<td>SIL</td>
</tr>
<tr>
<td>&lt;/s&gt;</td>
<td>SIL</td>
</tr>
<tr>
<td>&lt;sil&gt;</td>
<td>SIL</td>
</tr>
</tbody>
</table>

Where:

<s> stands for beginning of the speech.

<sil> stands for silence.

</s> stands for end of speech.

**Phonelist (ArabicDigits.phone)**
In this file, all phonemes and all background noises that are used in dictionary files (ArabicDigits.dic and ArabicDigits.filler) must be listed as illustrated in table-10.

Table 10-ArabicDigits.phone file that is used in the training.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Arabic alphabet</th>
<th>Transliteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>س</td>
<td>Sin</td>
</tr>
<tr>
<td>SS</td>
<td>ص</td>
<td>Emphatic Sad</td>
</tr>
<tr>
<td>F</td>
<td>ف</td>
<td>Alfa</td>
</tr>
<tr>
<td>R</td>
<td>ر</td>
<td>Ra</td>
</tr>
<tr>
<td>W</td>
<td>و</td>
<td>Waw</td>
</tr>
<tr>
<td>AA</td>
<td>أ</td>
<td>Alef</td>
</tr>
<tr>
<td>HH</td>
<td>ح</td>
<td>Ha</td>
</tr>
<tr>
<td>D</td>
<td>د</td>
<td>Dal</td>
</tr>
<tr>
<td>TH</td>
<td>ث</td>
<td>Tha</td>
</tr>
<tr>
<td>N</td>
<td>ن</td>
<td>Non</td>
</tr>
<tr>
<td>T</td>
<td>ت</td>
<td>Ta</td>
</tr>
<tr>
<td>L</td>
<td>ل</td>
<td>Lam</td>
</tr>
<tr>
<td>B</td>
<td>ب</td>
<td>Ba</td>
</tr>
<tr>
<td>K</td>
<td>ك</td>
<td>Kaf</td>
</tr>
<tr>
<td>M</td>
<td>م</td>
<td>Mem</td>
</tr>
</tbody>
</table>

Transcription file (ArabicDigits_trian.transcription)

This file is a text file listing the transcription for each audio file under sphinx format. For instance,

<s> SEFR </s>  (Speaker1_repetition1.wav)
<s> SEFR </s>  (Speaker1_repetition2.wav)
<s> SEFR </s>  (Speaker1_repetition3.wav)
Transcription file (ArabicDigits_test.transcription).

SEFR (Speaker1_repetition1.wav)
SEFR (Speaker1_repetition2.wav)
SEFR (Speaker1_repetition3.wav)

TESAH (Speaker1_repetition10.wav)

Fileids(ArabicDigits_train.fileid or ArabicDigits_test.fileid).

This is a text file has the name of the recordings (utterance ids) one by line.

For instance,

Digit0/ Speaker1_repetition1.wav
5.1.2 Building Language model

There are multiple types of models that can be used to describe any language to recognize during the decoding phase. For example, keyword list, grammars, statistical language models and phonetics statistical language models.

a) N-gram Language Model

A language model is a statistical model of word sequences of a corpus. It uses N-gram to achieve the idea of word prediction. In order to prepare the N-gram language model, I used the CMU Statistical Language Modeling (CMU-SLM) toolkit, which is a set of Unix software tools facilitates the construction and testing of conventional bigram and trigram language models. Figure-16 shows the procedure uses by CMU toolkit (Clarkson & Rosenfeld, n.d).
Figure 16-Use of CMU Cambridge toolkit.

First of all, CMU toolkit needs only transcription file as input. This transcription file should be in the form of normalized test without fileids and fillers, and the utterances should be delimited by <s> and </s> tags. For example, ArabicDigits.txt file is shown below:

<s> SEFR </s>
<s> WAHID </s>
<s> ETHNAYN </s>
<s> THALATAH </s>
<s> ARBAAH </s>
1) Assigning frequency

An initial step in building a language model is to obtain word frequencies of the training data in ArabicDigits.txt. The tool text2wfreq outputs the number of occurrences of each word (digit) in the input file.

`text2wfreq < ArabicDigits.txt> ArabicDigits.wfreq`

The output looks like this:

`SEFR 1
SETTAH 1
ARBAAH 1
ETHNAYN 1
SABAAM 1
KHAMSAH 1
THALATAH 1
TESAH 1
WAHID 1
THAMANEYAH 1`

2) Creating a vocabulary

The next step is to build the model's vocabulary. The tool wfreq2vocab turns the words list (digits) into a vocabulary file, in this case containing the most common 20,000 words.

`wfreq2vocab < ArabicDigits.wfreq > ArabicDigits.vocab`

The output file looks like this:

`ARBAAH`
ETHNAYN
KHAMSAH
SABA AH
SEFR
SETTAH
TESAH
THALATAH
THAMANEYAH
WAHID

3) Constructing n-gram models

The first step toward building n-gram model is to create the file ArabicDigits.idngram which is an intermediate file for creating the language model file. Using the tool text2idngram to turn the training text into a list of id N-grams (N-grams with each word mapped to an integer id, which will be zero for OOVs).

```
text2idngram -vocab ArabicDigits.vocab< ArabicDigits.txt>
```

**ArabicDigits.idngram**

The second step is converting the id N-gram stream into a binary language model file. To do this, ArabicDigits.ccs file must be provided which has following 2 lines:

```
<s>
</s>
```

Then running the command

```
idngram2lm -vocab ArabicDigits.vocab -idngram ArabicDigits.idngram -arpa ArabicDigits.ug.lm -context ArabicDigits.ccs
```

However, this is not the binary format which is required for decoding. The command **sphinx3_lm_converter** is used for this purpose (Clarkson & Rosenfeld, n.d).

```
sphinx3_lm_converter -i ArabicDigits.ug.lm -o ArabicDigits.ug.lm.DMP
```
b) Finite State Grammar (FSG)

Grammar permits to specify the language accepted by the speech recognizer. Sphinx provides a tool is called `sphinx_jsgf2fsg` to construct a format that describes this grammar. First I prepared the input file (ArabicDigits) as shown below:

```
Arabic</t>fsg</t>SEFR</s>WAHID</s>ETHNAYN</s>THALATAH</s>
ARBAAH</s>KHAMSAH</s>SETTAH</s>SABAAH</s>THAMANEYAH</s>
TESAH
```

Then run the command:
```
perl fsg.pl -l ArabicDigits
```

This will build a file with JSGF format which look like:

```
#JSGF V1.0;
grammar ArabicDigits_fsg;
Public <Arabic_fsg> = (SEFR | WAHID | ETHNAYN | THALATAH |
ARBAAH | KHAMSAH | SETTAH | SABAAH | THAMANEYAH |
TESAH);
```

In order to convert it into a Sphinx FSG file required for decoding, I run the command:
```
sphinx_jsgf2fsg < ArabicDigits_fsg > ArabicDigits_fsg.fs
```

5.1.3 Start training

SphinxTrain is used to facilitate training process where the source code is written in C language, and the script is under Perl programming language (Training Acoustic Model For CMUSphinx, n.d.). The training tool can be trained to semi-continues, or continues HMM models depending on the decoder whether it is Pocketsphinx, or Sphinx3. In this case I used Sphinx3 which uses continuous HMMs.

Run this command to start training phase:
perl scripts_pl/setup_tutorial.pl ArabicDigits

This will add several new directories in the main directory (ArabicDigits). These directories have files which are being generated in the course of the training.

5.1.4 Feature extraction

After preparing the database and all required files is a voice signal feature extraction phase. Feature extraction has a corresponding acoustic model library mainly by using make_feats.pl. For this stage, the file of the system must be specified either in WAV or SPHERE format. In this paper, all speech signals are in WAV format. File format can be set by editing the configuration file in etc folder etc/sphinx_train.cfg

```
$CFG_WAVFILES_DIR = "$CFG_BASE_DIR/wav";
$CFG_WAVFILE_EXTENSION = 'wav';
$CFG_WAVFILE_TYPE = 'mswav'; # one of nist, mswav, raw
```

Then run the command to generate the features:

Perl scripts_pl/make_feats.pl --ctl etc/ArabicDigits_train.fileids

5.1.5 Building and training the acoustic model

After preparing the data and language model, next step toward constructing acoustic model is training the models in order to build acoustic word models from gathering input data (training data set) of multiple occurrences of each of the vocabulary words by one or more speakers. In addition, construct dictionary from a text training data. This dictionary describes how each word should be pronounced by using subword units to characterize individual words), language model that defines how words are connected to produce valid sentences, and finally a task grammar that specifies which valid word strings are meaningful in the task application (Juang & Rabiner, 2006).

To construct and train the Acoustic models run this command:
perl scripts_pl/RunAll.pl
This will add ArabicDigits.html file in the main directory and the following output should be seen:

MODULE: 00 verify training files
O.S. is case sensitive ("A" != "a").
Phones will be treated as case sensitive!
Phase 1: DICT - Checking to see if the dict and filler dict agrees with the phonelist file.
Found 13 words using 20 phones
Phase 2: DICT - Checking to make sure there are not duplicate entries in the dictionary
Phase 3: CTL - Check general format; utterance length (must be positive); files exist
Phase 4: CTL - Checking number of lines in the transcript should match lines in control file
Phase 5: CTL - Determine amount of training data, see if n_tied_states seems reasonable.
Total Hours Training: 0.171830982905982
This is a small amount of data, no comment at this time
Phase 6: TRANSCRIPT - Checking that all the words in the transcript are in the dictionary
Words in dictionary: 10
Words in filler dictionary: 3
Phase 7: TRANSCRIPT - Checking that all the phones in the transcript are in the phonelist, and all phones in the phonelist appear at least once

Figure 17-Snapshot of ArabicDigits.html.

The training process is organized in a list of successive stages, each of which builds on the results of the previous one as described in chapter 2 and next sections.

1) Training CI modeling:
   - First create the CI model definition file as explained in chapter 2. model-definition file will look like this:
# Generated by
/home/hamda/Downloads/Thesis/ArabicDigits/bin/mk_mdef_gen on Mon Apr
0.3

20 n_base
0 n_tri
80 n_state_map
60 n_tied_state
60 n_tied_ci_state
20 n_tied_tmat

#Columns definitions
#base  lft  rt  p  attrib  tmat ... state id's ...
AA   -   -   -  n/a  0   0   1   2  N
AE   -   -   -  n/a  1   3   4   5  N
AH   -   -   -  n/a  2   6   7   8  N
AY   -   -   -  n/a  3   9  10  11  N
B    -   -   -  n/a  4   12  13  14  N
D    -   -   -  n/a  5  15  16  17  N
EH   -   -   -  n/a  6  18  19  20  N
EY   -   -   -  n/a  7  21  22  23  N
F    -   -   -  n/a  8  24  25  26  N
HH   -   -   -  n/a  9  27  28  29  N
K    -   -   -  n/a 10  30  31  32  N
L    -   -   -  n/a 11  33  34  35  N
M    -   -   -  n/a 12  36  37  38  N
N    -   -   -  n/a 13  39  40  41  N
R    -   -   -  n/a 14  42  43  44  N
S    -   -   -  n/a 15  45  46  47  N
SIL -   -   -  filler 16  48  49  50  N
<table>
<thead>
<tr>
<th>Phone</th>
<th>n_base</th>
<th>n_tri</th>
<th>n_state_map</th>
<th>Tmat</th>
<th>State Ids</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>-</td>
<td>-</td>
<td>n/a</td>
<td>17</td>
<td>51 52 53</td>
</tr>
<tr>
<td>TH</td>
<td>-</td>
<td>-</td>
<td>n/a</td>
<td>18</td>
<td>54 55 56</td>
</tr>
<tr>
<td>W</td>
<td>-</td>
<td>-</td>
<td>n/a</td>
<td>19</td>
<td>57 58 59</td>
</tr>
</tbody>
</table>

Where:
- **n_base** indicates the number of phonemes in the ArabicDigits.phone file.
- **n_tri** is the number of triphones.
- **n_state_map** indicates the total number of HMM states (emitting and non-emitting). The Sphinx appends an extra terminal non-emitting state to each HMM. Therefore, for 20 phones, each is modeled by a 3-state HMM, this number will be 20 phones * 4 states = 80 states.
- **n_tied_state** refers to the number of states of all phones after state-sharing is done. At this stage there is no state shared. Therefore, the number of states is 20 * 3 = 60.
- **n_tied_ci_state** represents the number of states for phones after state-share the original phonemes. Hence, the states of total emission state are 20 * 3 = 60.
- **n_tied_tmat** indicates each transition probability matrix that associated with each HMM CI phone. Therefore, the total number of transition matrices for the given model is 20.
- **base** represents the phonemes.
- **Lft** is the left-context of the phone.
- **rt** is the right-context of the phone.
- **p** refers to the location of a triphone.
- **Attrib** is attribute of phone. In the phone list, filler indicates non-voice and n/a indicates the voice phoneme.
- **tmat** indicates the id of the transition matrix associated with the phone.
- **state id** is the ids of the HMM states for each phone.

N at the end of list stands for non-emitting state.

- Second Generating the HMM topology file.
The HMM topology file look like this:

```
4
3.0  1.0  0.0  0.0
0.0  3.0  1.0  0.0
0.0  0.0  3.0  1.0
```

Where 4, represents the total number of HMM states. Sphinx system automatically adds a non-emission state. The first entry 3.0 means that there is a transition from state 1 to itself. The first 1.0 represents the transition from state 1 to state 2. The HMM topology is illustrated in next figure.

![HMM topology 3-state model](image)

Figure 18-HMM topology 3-state model.

To begin training the CI models, each of the files (means, variances, transition_matrices, mixture_weights) must have initial values. Global variances and means are calculated using the vectors in the feature file, and copied into means and variances of each state of the HMMs. The model_parameters/ArabicDigits.ci_cont_flatinitial contains the global mean and variance.

After this the CI model file is initialized by setting some parameters such as ArabicDigits.ci.mdef, ArabicDigits.topology, mixture_weights files and transition_matrices file in ArabicDigits.ci_cont_flatinitial folder. Acoustic models for CI phones are ready for training. This is done through the Baum-Welch algorithm. The training is done until the convergence ration is reached.
Model_params/ArabicDigits.ci_cont is used to store the trained CI models. The model parameters that computed during the last iteration are used to initialize the models for CD triphones with united states.

2) Training united CD model

First, generating the model definition file for all the triphones in the training set as shown below. This file like the CI mdef file assigns unique Ids to every HMM state and serve as reference for handling CD united model parameters. This file is stored in model_architecture/ArabicDigits.alltriphines.mdef.

0.3
20 n_base
162 n_tri
728 n_state_map
546 n_tied_state
60 n_tied_ci_state
20 n_tied_tmat

# Columns definitions
#base lft rt p attrib tmat ... state id's ...
AA - - - n/a 0 0 1 2 N
AE - - - n/a 1 3 4 5 N
AH - - - n/a 2 6 7 8 N
AY - - - n/a 3 9 10 11 N
B - - - n/a 4 12 13 14 N
D - - - n/a 5 15 16 17 N
EH - - - n/a 6 18 19 20 N
EY - - - n/a 7 21 22 23 N

77
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>n/a</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>n/a</td>
<td>8</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>N</td>
</tr>
<tr>
<td>HH</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>n/a</td>
<td>9</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>N</td>
</tr>
<tr>
<td>K</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>n/a</td>
<td>10</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>N</td>
</tr>
<tr>
<td>L</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>n/a</td>
<td>11</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>N</td>
</tr>
<tr>
<td>M</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>n/a</td>
<td>12</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>n/a</td>
<td>13</td>
<td>39</td>
<td>40</td>
<td>41</td>
<td>N</td>
</tr>
<tr>
<td>R</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>n/a</td>
<td>14</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>N</td>
</tr>
<tr>
<td>S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>n/a</td>
<td>15</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>N</td>
</tr>
<tr>
<td>SIL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>filler</td>
<td>16</td>
<td>48</td>
<td>49</td>
<td>50</td>
<td>N</td>
</tr>
<tr>
<td>T</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>n/a</td>
<td>17</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>N</td>
</tr>
<tr>
<td>TH</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>n/a</td>
<td>18</td>
<td>54</td>
<td>55</td>
<td>56</td>
<td>N</td>
</tr>
<tr>
<td>W</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>n/a</td>
<td>19</td>
<td>57</td>
<td>58</td>
<td>59</td>
<td>N</td>
</tr>
<tr>
<td>AA</td>
<td>AA</td>
<td>R</td>
<td>b</td>
<td>n/a</td>
<td>0</td>
<td>60</td>
<td>61</td>
<td>62</td>
<td>N</td>
</tr>
<tr>
<td>AA</td>
<td>AH</td>
<td>R</td>
<td>b</td>
<td>n/a</td>
<td>0</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>N</td>
</tr>
<tr>
<td>AA</td>
<td>B</td>
<td>AA</td>
<td>e</td>
<td>n/a</td>
<td>0</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>N</td>
</tr>
<tr>
<td>AA</td>
<td>B</td>
<td>AE</td>
<td>e</td>
<td>n/a</td>
<td>0</td>
<td>69</td>
<td>70</td>
<td>71</td>
<td>N</td>
</tr>
<tr>
<td>AA</td>
<td>B</td>
<td>AH</td>
<td>i</td>
<td>n/a</td>
<td>0</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>N</td>
</tr>
<tr>
<td>AA</td>
<td>B</td>
<td>K</td>
<td>e</td>
<td>n/a</td>
<td>0</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>N</td>
</tr>
<tr>
<td>AA</td>
<td>B</td>
<td>S</td>
<td>e</td>
<td>n/a</td>
<td>0</td>
<td>78</td>
<td>79</td>
<td>80</td>
<td>N</td>
</tr>
<tr>
<td>AA</td>
<td>B</td>
<td>SIL</td>
<td>e</td>
<td>n/a</td>
<td>0</td>
<td>90</td>
<td>82</td>
<td>83</td>
<td>N</td>
</tr>
<tr>
<td>AA</td>
<td>B</td>
<td>T</td>
<td>e</td>
<td>n/a</td>
<td>0</td>
<td>93</td>
<td>85</td>
<td>86</td>
<td>N</td>
</tr>
<tr>
<td>AA</td>
<td>B</td>
<td>TH</td>
<td>e</td>
<td>n/a</td>
<td>0</td>
<td>96</td>
<td>88</td>
<td>89</td>
<td>N</td>
</tr>
</tbody>
</table>

Where:

- n_base refers to the number of CI phones, which is 20 in this case.
- n_tri is the number of triphones, which is 162 in this case.
\textbf{n\_state\_map} indicates to the total number of HMM states (emitting and non-emitting). The Sphinx appends an extra terminal non-emitting state to each HMM. Therefore, for 20+162 phones, each is modeled by a 3-state HMM, this number will be 182 phones*4 states = 728 status.

\textbf{n\_tied\_state} refers to number of states of all phones after state-sharing is done. At this stage there is no state shared. Therefore, the number of states is 65*3=195.

\textbf{n\_tied\_ci\_state} represents the number of states for phones after state-share the original phonemes. Hence, the state of total emission state is 182*3=546.

\textbf{n\_tied\_tmat} indicates to each transition probability matrix that associated with each HMM CI phone .Therefore the total number of transition matrices for the given model is 20.

\textbf{base} represents the phonemes.

\textbf{lft} is left-context of the phone.

\textbf{rt} is right-context of the phone.

\textbf{p} refers to the locations of a triphone that can be represented by four markers; b= word beginning triphone; e=word ending triphone; i=word internal triphone; s=single word triphone.

\textbf{attrib} is attribute of phone. In the phone list.

\textbf{filler} indicates to non voice and n/a indicates the voice phoneme.

\textbf{tmat} indicates the id of the transition matrix associated with the phone.

\textbf{State id} $^S$ is the ids of the HMM states for each phone.

\textbf{N} at the end of list that stands for non emitting state.

Second is flat initialization of CD united model parameters, the model parameter files corresponding to the CD united model definition file are constructed. Then, means, variances, transition matrices and mixture weights files are generated. For each file, the values from corresponding CI model parameters file are copied. Each state of a particular CI phone contributes to the same state of the same CI phone in the CD -untied model parameter file. In addition, each state of a particular CI phone contributes to the same state of all the triphones of the same CI phone in the CD united model parameter file.
Finally, once the initialization is done, Baum Welch algorithm is used for tainting the CD united models. The model parameters then are stored in model_parameters/ArabicDigits.cd_cont_united.

3) Building decision tree for parameter sharing

The main purpose of decision tree is to decode which HMM states of all triphones are similar to each other. Collected data from these states is used to train one global state. As mentioned in chapter 2 decision trees require CD united models and a set of predefined acoustic classes which share some common property. Following is creating linguistic questions to partition the data at any given node of a tree. The result will be one partition from each question. The questions that results in the best partition are chosen to partition data at the node.

All linguistic questions are stored in one file in model_architecture/ArabicDigits.tree_questions file. Decision trees must be built for every state of each CI phone in the phonelist after generating the linguistic questions. For 20 base phones with 3 states HMM, there are 20×3=60 trees. These trees are pruned once they are built.

4) Training CD tied Gaussian mixture models

This is done after creating the CD tied model definition file that will be stored in model_parameters/ArabicDigits.cd_cont_${no_of_gaussians} and initializing the CD tied Gaussian mixture models as described in chapter 2.

The final set parameters which are used for decoding are stored in model_parameters/ArabicDigits.cd_cont_${no_of_stones}_${no_of_gaussians}.

5.1.6 Decoding

This task is done by using Sphinx3 decoder which is the successor to the Sphinx-II speech recognition system from Carnegie Mellon University. The required files in decoding step are trained models, dictionary, filler dictionary, test data, and language model or FSG.
First, the MFCCs features for all the test utterances (test wav files) in the test data set are computed and stored in directory called feat by typing the following command:

```
perl scripts_pl/make_feats.pl -ctl etc/ArabicDigits_test.fileids
```

Then perform the recognition on test data

```
perl scripts_pl/decode/slave.pl
```

This command will start decoding process using the different trained acoustic models and the language model that are configured in etc/sphinx_train.cfg file. When the recognition is complete, the script computes recognition Word Error Rate (WER) which will be shown in next chapter.

### 5.2 Adapting the acoustic model

The previous acoustic models, speaker independent model (SI) and Native Arabic model (NA) that constructed earlier in this chapter are adapted using adaptation technique namely, MLLR. Phases of adaptation is explained and shown below.
Figure 19-Phases of adaptation.

Speaker independent acoustic model

Native Arabic speakers acoustic model

Adaptation data=1600 tokens
Test data=200 tokens

Adaptation data=120 tokens
Test data=480 tokens

Acoustic model feature files

Gather Statistics from the adaptation data

Create transformation matrix

Apply the transformation to the mean vector

Test the adaptation using the new set of means
1-Install both Sphinx3 and SphinxBase.

2- Prepare the adaptation data:

   For speaker independent adaptation, 16 native Arabic males speakers were
   used as adaptation data (speakers fifteen through thirty (1600tokens)). Two test
   sets are used, test1 (speaker 11and speaker12) and test2 (speaker 13 and
   speaker14).

   For adapting non native Arabic to native Arabic system, First and second
   repetitions for all 6 non native speakers were used as adaptation data (6
   speakers× 2 repetitions × 10 digits=120 tokens). For testing purposes,
   repetitions from three through ten are used (6 speakers× 8 repetitions × 10
digits= 480 tokens).

3- Prepare fileids and transcription files for both adaptation and test data.

4-Copy both acoustic model files of speaker independent system and native
   Arabic males system( mdef, means, variances and transition - matrices) from
   model_parameters/ArabicDigits.ci_cont into different directory in order to work
   on them. Then perform the following steps on them separately.

5-Generate a set of acoustic model feature files from WAV audio records in
   order to run the adaptation tools. This can be achieved with the sphinx_fe tool
   from SphinxBase.

   sphinx_fe -argfile en-us/feat.params -samprate 16000 -c arctic20.fileids -di
   . -do . -ei wav -eo mfc -mswav yes

6-Gather statistics from the adaptation data. This is done by running Baum-
   Welch program from SphinxTrain:

   ./bw -hmmdir en-us -moddeffn en-us/mdef.txt -ts2cbfn .cont. -feat
   1s_c_d_dd -cmn current -age none -dictfn arctic20.dict -ctlfn

7-Create transformation matrix using the program mllr_solve:

   ./mllr_solve -meanfn en-us/means -varfn en-us/variances -outmllrfn
   mllr_matrix -accumdir .

8- Apply the transformation to the mean vector. This is done using
   mllr_transform program:

83
9- Finally testing the adaptation using the new set of means:

```
Sphinx3_decode -adcin yes -cepdir wav -cepext .wav -ctl adaptation-test.fileids -lm en-us.lm.dmp -dict arctic20.dict -hmm en-us-adapt -hyp adaptation-test.hyp
```

This is done by word_align.pl script from SphinxTrain distribution

```
perl word_align.pl adaptation-test.transcription adaptation-test.hyp
```

All results are shown in next chapter.
Chapter 6

Results and discussions

6.1 Evaluation of isolated Arabic digits recognition systems

Last step is to evaluate the recognizer performance by determining the word error. This step is the last stage in decoding which uses all the components, including the different generated acoustic model explained in the previous chapter and the model index file that generated during the training run. To perform Regression test on pre-recorded wave files run the command:

`perl scripts_pl/decode/slave.pl`

This command will test the recognizer performance using different test sets according to each training mode. The following sections show recognition performance results for each system:

6.1.1 Evaluate speaker independent system (SI)

Speaker independent had a lower overall accuracy rate when is used to configure the system as shown in table-11 and table-12. Depending on the testing database set, the system must try to recognize 20 samples for every digit where the total number of tokens is 200. The overall system accuracy is 88.5% and 85% for test1 and test2 respectively. For test1, the worst performance was found in case of digit two with accuracy equal to 70%; and the best performance was encountered in the case of digit one with accuracy equal to 100%. The system failed in recognizing 23 words out of 200 words. While, for test2 the system failed in recognizing 30 words out of 200 words. The worst performance was found in case of digits two, three, eight, and nine with accuracy equal to 80%: digits one, four, five, and six had the highest performance 90%. Table-11 and table-12 display the total accuracy and
individual Arabic digit accuracy in speaker independent system using test1 and test2. Figure -20 shows Error rate for individual Arabic digits for both test sets.

Table 11-Accuracy for individual Arabic digits for speaker independent system using test1.

<table>
<thead>
<tr>
<th>Digit</th>
<th>No. of insertions</th>
<th>No. of deletions</th>
<th>Substitution with</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>-</td>
<td>1</td>
<td>4,6</td>
<td>85</td>
</tr>
<tr>
<td>One</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Two</td>
<td>-</td>
<td>3</td>
<td>8,1,9</td>
<td>70</td>
</tr>
<tr>
<td>Three</td>
<td>-</td>
<td>-</td>
<td>0,5</td>
<td>90</td>
</tr>
<tr>
<td>Four</td>
<td>-</td>
<td>-</td>
<td>7,7</td>
<td>90</td>
</tr>
<tr>
<td>Five</td>
<td>-</td>
<td>-</td>
<td>6,7,9</td>
<td>85</td>
</tr>
<tr>
<td>Six</td>
<td>-</td>
<td>-</td>
<td>0,3</td>
<td>90</td>
</tr>
<tr>
<td>Seven</td>
<td>-</td>
<td>1</td>
<td>6</td>
<td>90</td>
</tr>
<tr>
<td>Eight</td>
<td>-</td>
<td>1</td>
<td>2</td>
<td>90</td>
</tr>
<tr>
<td>Nine</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>95</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td>88.5</td>
</tr>
</tbody>
</table>
Table 12- Accuracy for individual Arabic digits for speaker independent system using test2.

<table>
<thead>
<tr>
<th>Digit</th>
<th>No. of insertions</th>
<th>No. of deletions</th>
<th>Substitution with</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>-</td>
<td>1</td>
<td>7,6</td>
<td>85</td>
</tr>
<tr>
<td>One</td>
<td>-</td>
<td>1</td>
<td>3</td>
<td>90</td>
</tr>
<tr>
<td>Two</td>
<td>-</td>
<td>2</td>
<td>8,8</td>
<td>80</td>
</tr>
<tr>
<td>Three</td>
<td>-</td>
<td>1</td>
<td>8,0,5</td>
<td>80</td>
</tr>
<tr>
<td>Four</td>
<td>-</td>
<td>-</td>
<td>7,0</td>
<td>90</td>
</tr>
<tr>
<td>Five</td>
<td>-</td>
<td>-</td>
<td>6,1</td>
<td>90</td>
</tr>
<tr>
<td>Six</td>
<td>-</td>
<td>-</td>
<td>5,7</td>
<td>90</td>
</tr>
<tr>
<td>Seven</td>
<td>-</td>
<td>-</td>
<td>4,6,5</td>
<td>85</td>
</tr>
<tr>
<td>Eight</td>
<td>-</td>
<td>2</td>
<td>2,2</td>
<td>80</td>
</tr>
<tr>
<td>Nine</td>
<td>-</td>
<td>1</td>
<td>6,6,6</td>
<td>80</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td>85</td>
</tr>
</tbody>
</table>
6.1.2 Evaluate speaker dependent system (SD)

In the speaker dependent system, the system must try to recognize 300 samples for each digit, where the total number of tokens is 3000. The system overall accuracy was 99.83%. The system failed to recognize 5 tokens out of the 3000 total tokens. The best performance was encountered with 0, 4, 5, 6, 7, 8, and 9 with 100% accuracy. Digits 2 had the worst performance that decreased to 99%. Although the database size was small and the speaker variability that was used in training, the system with speaker dependent showed significant improvement compared to speaker independent. Table-13 displays the overall accuracy and individual Arabic digit accuracy in speaker dependent system. The performance for both systems is displayed in figure-21.
Table 13-Accuracy for individual Arabic digits for speaker dependent system.

<table>
<thead>
<tr>
<th>Digit</th>
<th>No. of insertions</th>
<th>No. of deletions</th>
<th>Substitution with</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>One</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>99.67</td>
</tr>
<tr>
<td>Two</td>
<td>-</td>
<td>-</td>
<td>8,1,5</td>
<td>99</td>
</tr>
<tr>
<td>Three</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>99.67</td>
</tr>
<tr>
<td>Four</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Five</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Six</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Seven</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Eight</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Nine</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td>99.83</td>
</tr>
</tbody>
</table>
6.1.3 Result of native Arabic system to foreign accented speakers.

This system is trained using 80% of the native speakers' data (2400 tokens), and tested using two test sets. First test set consists of 20% of the native speakers' data (600 tokens), and second test set consists of 600 tokens (6 non-native Arabic speakers × 10 digits × 10 repetitions). The system must recognize 60 samples for each digit. Obviously, the performance of the system degrades when it is tested by non-native speakers to 40.17%, and failed to recognize 359 words out of 600 words with total of 81 insertions, 57 deletions and 221 substitutions. This performance is low compared to the same system that is tested with native Arabic speakers. In this case, the system failed to recognize just 17 words out of 600 words with only 17 substitutions. Figure-22 compares the system accuracies with test 1 and test2.
6.2 Results of adaptation

6.2.1 Adapting the acoustic model for independent system

Adapting the acoustic model for independent system has improved the system overall performance by reducing number of deletions for both test sets. When testing the system using test1, the total tokens tested by the system are 200 (20 for every digit). The overall performance was 99% with total of 2 miss-recognized tokens. The worst performance was found in case of digit 3 with accuracy equal to 90%; the system replaced this digit with digit eight two times. Table-14 displays detailed information about the performance of individual digits of speaker independent adaptation when test 1 is used. In addition, the overall performance of the system when test 2 is used was 98% which is reasonably high. The system failed in recognizing only four tokens out of the 200 total tokens. Digit two is substituted with digits three and eight, which reduced the accuracy for this digit into 80%. Table-15 displays detailed
information about the performance of individual digits of speaker independent adaptation when test 2 is used. Figure 23 shows the accuracy for speaker independent before and after adaptation.

Table 14—For individual Arabic digits for speaker independent system using test 1.

<table>
<thead>
<tr>
<th>Digit</th>
<th>No. of insertions</th>
<th>No. of deletions</th>
<th>Substitutions with</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>One</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Two</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Three</td>
<td>-</td>
<td>-</td>
<td>8,8</td>
<td>90</td>
</tr>
<tr>
<td>Four</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Five</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Six</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Seven</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Eight</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Nine</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Total</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>99</td>
</tr>
</tbody>
</table>
Table 15-Accuracy for individual Arabic digits for speaker independent system using test2.

<table>
<thead>
<tr>
<th>Digit</th>
<th>No. of insertions</th>
<th>No. of deletions</th>
<th>Substitutions with</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>One</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Two</td>
<td>-</td>
<td>-</td>
<td>3,3,3,8</td>
<td>80</td>
</tr>
<tr>
<td>Three</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Four</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Five</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Six</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Seven</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Eight</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Nine</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td>98</td>
</tr>
</tbody>
</table>
6.2.2 Adaptation of foreign accented speaker to Isolated Arabic digits recognition system.

Adaptation of foreign accented speakers into acoustic model completely trained using native Arabic speakers has improved by 6.29% even though the adaptation data is quite small (120 tokens). The miss-recognized tokens were reduced from 359 tokens to 257 tokens. Figure-24 shows the accuracy of the system before and after adaptation.
Figure 24 - Overall Accuracy for Arabic digits system before and after adaptation to foreign speakers.
Chapter 7

Conclusion and future work

7.1 Conclusion

Spoken isolated Arabic digit recognition systems were designed to investigate the process of automatic recognition. These systems were based on the Sphinx-based system that is developed by the Carnegie Mellon University. A database of 3000 tokens were created using 30 Arabic native speakers, 2000 tokens were created using 20 non native speakers. The overall system performance was 99.83% in speaker dependent system. For speaker independent system the performance was 88.5% and 85% depending on the test set. This proves that Arabic digit recognition systems must be trained to the individual user to get higher performance and the performance is affected by speaker's variety. With the limited amount of spoken Arabic digit database, training the Arabic system explicitly for each user is difficult. Therefore, adapting the acoustic model of speaker independent system is the solution that has desirable speaker dependent properties but requires only a small fraction of speaker specific training data needed to build a full system. Adapting the acoustic model of speaker independent improved the performance by 10.5% and 13%.

On the other hand, isolated Arabic digit recognition system was trained by all Arabic native speakers and tested using native Arabic speaker and non native speakers' data sets. The Word Error Rate of the system increased from 2.83% to 59.83% when used by non native. This is mainly because of both acoustic and phonological differences between accents. Acoustic model adaptation successfully dropped WER to 53.54%.

To conclude, acoustic model adaptation is the best guarantee for improving Arabic speech recognition performance. MLLR framework is a good basis for speaker independent systems. Its main advantages are reliable results and its
ability to adapt without need of any prior knowledge. In addition, MLLR gives the fastest improvements for adaptation of Arabic speech recognition systems to foreign accented speakers with relative reduction of 6.29%.

7.2 Future work

Further investigation is required to study methods and factors that raise Arabic speech recognition system performance whether it is for isolated or continues speech. This is can be done by:
1- High quality and larger speech corpora.
2- Adaptation of acoustic model using Maximum a posteriori (MAP).
3- Adaptation of acoustic model using combination of MLLR and MAP.
4- Evaluate the performance of adaptation Arabic speech recognition to foreign accents not only at word level, but also at phone levels.
References

Adami, A. (n.d.). Automatic Speech Recognition: From the Beginning to the Portuguese Language.


P. C. Woodland. Speaker Adaptation for Continuous Density HMMs: A Review. ITRW on Adaptation Methods for Speech Automatic Recognition, pp. 11-18, 2001


