Audio Processing on Constrained Devices

by

Amod Gupta

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2009

Abstract

This thesis discusses the future of smart business applications on mobile phones and the integration of voice interface across several business applications. It proposes a framework that provides speech processing support for business applications on mobile phones. The framework uses Gaussian Mixture Models (GMM) for low-enrollment speaker recognition and limited vocabulary speech recognition. Algorithms are presented for pre-processing of audio signals into different categories and for start and end point detection. A method is proposed for speech processing that uses Mel Frequency Cepstral Coeffcients (MFCC) as primary feature for extraction. In addition, optimization schemes are developed to improve performance, and overcome constraints of a mobile phone. Experimental results are presented for some prototype applications that evaluate the performance of computationally expensive algorithms on constrained hardware. The thesis concludes by discussing the scope for improvement for the work done in this thesis and future directions in which this work could possibly be extended.

Acknowledgements

I would like to take this opportunity to express my gratitude to all those people who supported me in one way or another. First and foremost, I'd like to deeply thank my supervisor, Dr. Tamer Ozsu for his unstinting support and patient guidance which led my research towards the right direction. His hardworking spirit, experience and enthusiasm has always been a source of inspiration for me.

I would also like to thank Dr. Serhan Dagtas and Nolwen Mahe, my team leader and manager at SAP Labs during my internship there in the summer of 2008. Their valuable input and support helped me to look at this work from an industrial point of view.

I dedicate this thesis to my parents and my younger brother who supported me throughout my years of study. Without their love, support and encouragement this would not be possible.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

As technological improvements allow for the development of more powerful and ubiquitous hand-held devices, such as mobile phones and PDAs, the need for better utilization of the capabilities of these devices intensifies. No longer merely luxurious items, these devices have advanced far beyond the realms of simple phones or calendars and are now often used as cameras, music players and gaming consoles. A lot of business applications these days are launched on mobile platforms, which add to the different functions these devices are required to perform.

Due to the myriad of different functions that these devices are required to perform, there is also need to look beyond the traditional way of interacting with the device. Currently, by far the most prevalent way of communication with a mobile phone is the limited keyboard present on the device. This keyboard is constrained by the size of the device. The keys are so small that in some cases anything except punching a few numbers becomes tedious. This difficulty has been identified and some attempts to overcome it have been made by introducing full QWERTY keyboards and touch screen functionality. These alleviate the problem in some way as demonstrated by the increasing popularity of PDAs. In this thesis we investigate an alternative mode of interaction, namely the use of voice.

1.1 Objectives and Motivation

Most of the work on speech technologies (throughout this thesis, speech technology refers to speech-to-text) has focused on desktop computers with comparatively little work focusing on mobile phones. However, with the rapid increase in the capabilities of mobile phones, huge advances are being made in this field. More and more phones with sophisticated technologies and new sensors are introduced every day. For example, Apple recently launched its iPhone 3GS which comes loaded with

- Accelerometer - Used to detect rotation and position of the device for better user experience.
- Ambient Light Sensor - Adjusts the brightness of the screen according to the surroundings of the phone.
- Proximity Sensor - Senses when the device is close to the ear and turns off the display to save battery.

Devices like Blackberry Storm, HTC Touch, Samsung Omnia, Samsung Instinct are other phones that have used new sensors to enhance user experience. Looking at this recent trend one can see that there is a major shift in the market towards smart phones. Two major reasons emerge for this shift

1. Mobile phones are not just pocket phones anymore. They have become multipurpose devices and the industry recognizes that treating them as mere phones is no longer a viable option. The popularity of a device apart from other things will also depend on its usefulness. This partially explains the race to make a mobile phone also work as a camera, music player, PDA, internet surfing device etc.

2. Mobile phones are small in size and the size cannot be increased beyond a certain measure. In fact, size is one of the biggest limiting factors and it makes the use of mobile phones cumbersome. Therefore by adding new sensors there is an attempt to enhance the user experience, making it richer and more comfortable.

Apart from the smart phones, ordinary mobile phones have had many built-in sensors. A built-in camera has almost become a standard for mobile phones today. However one of the most under-used sensor on a phone is the microphone. In addition to its obvious function, it can be used for speech recognition and environment sensing. This functionality, if properly utilized, can prove to be beneficial for context-aware (i.e. intelligent) business applications. One of the objectives of this work is to provide context-aware information to business applications that may need it. As the devices become increasingly powerful more and more business applications are being released in their mobile versions. Some of these applications can benefit a great deal if provided with information about the surroundings of the device, such as the noise level, the brightness of light, presence of humans in the surroundings etc. As will be shown later, new sensors are not required to gather a lot of context aware information.

Speech is a natural mode for man-machine interaction. Speech input implemented in user interfaces and speech output, such as reading out SMS and emails, play an important role in enhancing the usability of a device. This consideration becomes more relevant for small devices like mobile phones where the size of the keyboard and the screen limits the ease of interaction between the user and the device. More convenient ways of interacting with the mobile phone are now needed. With the increase in the computational capabilities, on board memory and the data handling capabilities of the phone, more and more sophisticated business applications are now launched over mobile platforms. Few buttons at the bottom of the screen are not enough for navigation, nor is the keyboard convenient enough for typing anything more than a short message. Most of the phones don't have the advanced sensory capabilities of the iPhone or its other rivals like HTC Touch or Blackberry Storm to help enhance interaction. On the other hand, speech communication with a mobile device overcomes the barriers introduced by other modes of interaction.

Most of the phones are bundled with speech processing capabilities to some extent. Traditionally these capabilities were limited to applications such as name dialling, digit recognition or (somewhat limited) voice commands. Some current phones claim to have advanced speech processing capabilities like SMS dictation or speech recognition. However, most of these traditional and advanced applications are native to the phone and are tightly coupled with the phone's operating system for efficiency and performance. As such, they serve a specific purpose and are not portable or general. This means that a similar application on a different platform has to be developed from scratch. A portable modular framework that provides speech support across different applications is useful for a number of business applications that need to run on different mobile platforms. An objective of the work reported in this thesis is to develop a framework that provides speech support to different applications across different platforms.

Depending on where the processing is being done, mobile speech applications can be divided into three categories

1. Embedded - client side processing of audio data. The applications perform all the processing of audio data on the mobile phone itself. These applications are limited by the hardware present on board the mobile phone. Memory and processing power are the resources constraining such applications.

2. Network - server based processing. Applications falling in this category collect

audio data using the mobile phone but send it over the network to a server that does the processing for them and then hands the result back. Audio data is passed over the network and these applications are limited by the availability of the network hence can be unstable at times. Also, issues related to network latency need to be addressed.

3. Distributed - processing load is shared between the client and the server. Audio data are collected on the client side, some processing is done and the results are passed over the network to the server. The server provides intermediate results back to the device and then the results (as they are or with some modifications) are presented to the user. These types of applications try to have the best of both the worlds, but also inherit the limitations of both.

1.2 Speech applications on mobile phones

Phones possessing speech processing capabilities are now quite common. This is a sign of both the maturity of the speech processing technology and the enhancement in the capabilities of mobile phones. A historical survey of applications on mobile phones [6] covers the history of embedded speech applications from the Samsung A500 that had a digit dial application, A600 that had name dialling to Samsung P-207 that incorporates a large vocabulary speech recognition system.

Many commercial applications are now available that use the speech technology. Nuance, Voice Signal and Scansoft were the market leaders in providing speech solutions before they merged to form one entity. They provide an impressive range of speech solutions like Device Control, Voice SMS, Name and number dialling and boast applications for Blackberry, Nokia, Samsung and other leading mobile phone manufacturers. For advanced phones like iPhone, new applications are being released continually. These range from voice dialling to speech interfacing. Speech solutions have made immense progress from the time of when Samsung launched number dialling support for A500; however one aspect where they still fall short is platform independence. The functionalities come in different flavors depending on the phone manufacturer and are native to the phone. Also, these applications have specific functionalities like voice commands or name dialling, thereby restricting the voice support present on the device to these few applications. Our framework differs from these applications in that it provides platform independent support to applications on the mobile phone. Any application developed on the J2ME platform can make use of this framework to make itself speech enabled.

1.3 User interface perspective

Due to the limited size of mobile phones, the traditional way of interacting with the phone, namely using a keyboard is not very convenient. An effort has been made to discover different ways of interacting with the device. The most recent trends in the industry suggest that 'touch' is currently preferred way of interaction. Apple, Nokia, Blackberry and HTC have all come out with their touch-based phones. Touch enhances the user interaction by providing a bigger screen and larger space for selection. Although touch phones have been a success since they were launched, speech can yet offer an alternative and possibly a better way of interaction with the device.

From the user interface perspective, experts are generally of the opinion that voice provides a very convenient and useful way of interacting with the mobile device. Hands-free and eyes-free UI for the phone will not only increase the safety of using mobile phones but it will also increase the ease of use with which these devices can be handled. Cohen [6, 5] argues that the future is bright for the speech technologies. However, while voice-centric interfaces have been widely available in modern phones (even the low cost versions) they are not very popular with the users. Cohen [6] argues that the customer indifference can be attributed to 3 major reasons: marketing, interfaces and applications. From the marketing perspective the original voice support had sub-optimal performance and was oversold. None of the recent marketing campaigns have been able to differentiate between the modern, competent speaker-independent technologies and the earlier implementations. In the cell phone industry, marketing is controlled by the carriers, and they have not viewed speech interfaces as critical for their financial success. As for interfaces, they are not intuitive enough and there is no "must-have" voice application. When it comes to applications, Cohen argues that once such a killer application is developed this user interface will catch up rapidly. He goes on to suggest that voice search may turn out to be that killer application. However its not only the lack of a killer application but also the lack of modular voice support across all applications on the device that has restricted the popularity of voice as a viable user interface.

1.4 Challenges and Contributions

In this thesis we propose a framework for embedded speech processing functions that can be used to provide speech support and environment sensing capabilities to any mobile business application. While this is not meant to be full-fledged speech recognition or speaker recognition system, the framework is capable of providing limited vocabulary speech recognition for business applications to develop voice interface, low enrolment speaker identification and environment sensing for context aware applications that need it. This framework is platform-independent and can be used on any phone supporting J2ME. It does all the processing on the client side and does not require any backend support over the network which enhances its usability.

The methodology and algorithms that we developed for implementing this framework have their foundation in mathematical fundamentals that have formed the theoretical core for any speech processing technology. This approach ensures the correctness and the accuracy of the framework. However, considerable adaptation is necessary to match the unique constraints of a mobile phone. Proven desktop speech processing technologies, when ported to the phone without careful adjustments, result in unacceptable performance.

When compared to desktop computers, mobile phones and PDAs have only a few megabytes of RAM and a few hundred megahertz of processing power. This severely limits the complexity of applications that can be run on these devices. These constraints assume an even greater significance for speech technology, because the response time requirement for such applications is often real-time. To provide real time response on mobile phones by modifying, adapting and optimizing the existing technologies for desktop computers is one of the challenges addressed in this thesis.

Most of the speech applications on the mobile phones and other such constrained devices available today are very tightly integrated with the hardware and the operating system of the device. These applications have very specific functionalities and are native to the phone. The challenge is to develop platform independent framework for providing speech support. J2ME is used as the preferred technology due to the fact that it's platform independent and very popular (hence easily available on a lot of phones). However, any J2ME application is inherently slow when compared to the native application and since J2ME is the micro edition of JAVA it is not as powerful as J2EE or J2SE. To develop a framework for supporting speech in J2ME that ensures reasonable response time is another challenge that is addressed.

The main contributions of this thesis are

- Platform-independent speech processing and environment sensing support for mobile phones;
- Framework for providing a voice interface to business applications;
- Adaptation and optimization of proven speech processing algorithms on the mobile phone.

1.5 Outline of thesis

The rest of this thesis is organized as follows: In Chapter 2 a review is provided of some existing applications for speech processing on mobile phones and desktop computers. The chapter also includes the role of voice-centric interface on the usability of mobile phones, and why it has not been very successful till now. Some speech processing algorithms are presented and basic mathematical fundamentals are identified that form the core of the proposed framework.

In Chapter 3 presents the general architecture of the proposed speech processing system, focussing on the extraction of information for both speech recognition and speaker identification from the same speech engine. Existing models for speech processing applications are reviewed and reasoning behind the proposed approach is provided. In Chapter 4, a description of the main algorithm used for audio signal processing in general and speech processing in particular is presented. A step-by-step walk-through of the whole procedure from speech audio input collection to speaker verification is provided. Following the description of the methodology, Chapter 5 describes the settings for the experiments that were conducted for performance analysis. It also describes in detail the optimization techniques that were used to adapt the framework to the unique hardware and computational constraints of a mobile phone and present proof of concept applications. In Chapter 6, the business application scenarios in which this framework would be useful are considered. These applications may use the framework for providing navigational support or gathering input for context aware processing. Finally a conclusion with a short summary and directions for future work is presented.

Chapter 2

Background and Related Work

In this chapter, a comprehensive survey of the work done in the areas related to speech processing is presented. Before an audio signal can be processed to derive meaningful speech information, it has to be pre-processed. This pre-processing might determine whether the signal has any speech content in it. On finding speech content, the pre-processing module might be responsible for amplifying that part of the signal, removing periods of silence etc. After this pre-processing has been done, the audio signal is then sent to different modules for transformation, feature extraction and then verification. Figure 2.1 shows a simplified pipes and filter architecture for a generic speech processing system. First the background for a general speech processing system is presented and then related work done in these different areas is discussed.

2.1 Background

Physically the speech signal (actually all sound) is a series of pressure changes in the medium between the sound source and the listener. The most common representation of the speech signal is the oscillogram, often called the waveform,

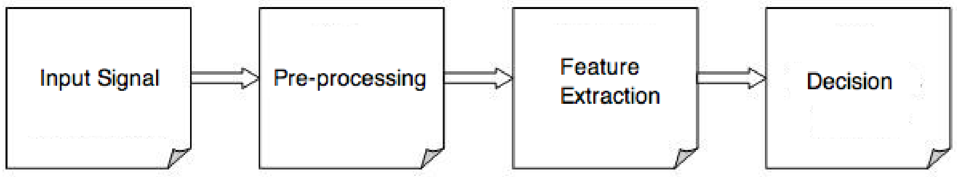

Figure 2.1: A simplified overview of speech processing architecture

which will be used in figures in the following chapters. In this representation the time axis is the horizontal axis from left to right and the curve shows how the pressure increases and decreases in the signal.

2.1.1 Sampling

Speech is a continuous signal but is stored electronically as discontinuous samples. This process of reducing a continuous signal to discrete samples is called sampling. The continuous signal varies over time and the sampling process is performed by measuring the continuous signal's value after a fixed interval of time, called the *sampling interval*. This process of sampling is governed by the Nyquist theorem which states that an analog signal that has been sampled can be reconstructed from its samples if the sampling frequency is at least twice the highest frequency present in the analog signal. In practice, for a signal that is a function of time, the sampling interval is typically quite small, of the order of milliseconds, microseconds, or less. This results in a sequence of numbers, called *samples*, to represent the original signal. Each sample value is associated with the instant in time when it was measured. Though the range of human hearing lies between 20 to 20,000Hz most of the energy in a speech signal (signals intended to carry human speech) is contained in the band between 5Hz - 4kHz allowing a sampling rate of 8kHz. Most telephony systems use 8kHz or 16kHz as their sampling frequency.

2.1.2 Audio file formats

Once the speech signal has been sampled it is stored in the form of samples in a file. Some of the most popular encoding formats are MP3 (.mp3) format, WAVE (.wav) format and AMR (.amr) format. These formats differ in the amount and type (lossy or lossless) of compression, their headers and some other specifics. WAVE file formats are one of the pure (lossless) ways of storing sound and will be discussed here. Standard format for a wav file is 44,100 samples per second and 16 bits per sample. The wav format is based on the Resource Interchange File Format (RIFF), which stores audio files in indexed chunks and sub-chunks. The main two chunks are the format chunk and the data chunk. The format chunk contains information about how the data are stored. Information such as number of channels, sampling rate, data size, etc are needed to make any meaningful use of the data. The data chunk contains the digital audio sample data which contains speech and related information. Speech applications, depending on the audio file format, first read the format chunk and then read the data chunk to derive any meaningful information. Since each sample is stored (generally 16 bits) in 2 bytes, each speech application first re-constructs the samples and stores them in an array. Pre-processing and further processing is then performed on arrays thus obtained.

2.1.3 Pre-processing

After the array of samples is obtained, pre-processing is performed on it before any speech specific information is extracted from it. This pre-processing is not a standard feature and might be totally absent from some speech applications.

Further, pre-processing will also depend on the goal of the speech application. Pre-processing is generally performed initially to determine if an audio signal contains any speech content. Once speech content has been determined, emphasizing of the speech content and end point detection may also be performed during the pre-processing stage itself.

Short time energy is one of the most popular metrics used for any kind of speech processing. Its popularity can be attributed to the fact that computing energy from a speech signal is a relatively easy operation. In their paper Wasson et al [38] present speaker recognition experiments in which they use short term energy to pre-process the signal before feature extraction. Wu, Ren, Liu and Zhang [44] present a Voice Activity Detection (VAD) system in which short term energy in the speech signal forms one of the most important metrics for VAD in moderate Signal-to-Noise ratio (SNR) conditions. They also use the short term energy for beginning and end point detection but argue that short term energy, as a metric for VAD, loses its usefulness in low SNR conditions. Mathematically, short term energy for a frame can be calculated as

$$Energy_i = \sum_{j=0}^{n} a_j^2$$

where a_j represents the amplitude of the jth sample. $Energy_i$ represents the energy of the ith frame and n the number of samples in the frame.

Fundamental frequency is another useful metric that is used in a number of speech processing systems. Fundamental frequency is the dominant frequency of repetition in any audio signal. Fundamental frequency is often used for speaker recognition purposes. Gerhard [13] presents an overview of audio signal classification, where it is argued that fundamental frequency is the lowest frequency of repetition in a signal and can be used to gauge the periodicity. This in turn can be used to distinguish if the audio is music, birdcall etc. According to Gerhard, most information that can be derived from the fundamental frequency is obtained by observing how the sound changes over time. According to Wasson et al [38], apart from fundamental frequency, formant frequencies also play an important role in classification of sound. Formants are distinguishing meaningful frequencies in human speech and singing. For instance, the information needed to distinguish between vowels is contained in the formant frequencies. The calculation of fundamental and formant frequency for speech as described in [19] is not a trivial process and hence the mathematical algorithm for it will not be presented here.

Another important metric used for speech analysis and pre-processing is Zero Crossing Rate (ZCR). ZCR, like short term energy, is a popular metric due to the ease with which it can be measured. As the name suggests, ZCR is the measure of how often the signal crosses from positive to negative and vice-versa. A detailed explanation of how ZCR is computed is presented in the next chapter. According to Gerhard [13] ZCR is useful for estimating fundamental frequency and extracting features such as spectral content, voicedness and noise content. Wasson et al [38] show how ZCR can be used to identify speakers in an automated manner. Kedem [21] shows how different statistical metrics can be collected using the ZCR and how they can be used to perform spectral analysis. The emphasis of this work is on higher order ZCR analysis.

Marolt [25] cites the importance of pitch as a feature for computing similarity specially of musical fragments. Though Marolt's work focuses on music information retrieval by finding melodic lines through a given piece of music, it shows that pitch can be used as a useful discriminator between speech music and other audio categories. Apart from these, other metrics that have been used during the pre-processing stage of speech processing systems are Artificial Neural Networks (ANN) [25] and Linear Discriminant Analysis [3]. Both ANNs and LDAs are computationally expensive as compared to using simple metrics like Short term energy and ZCRs. ANN and LDA might be suitable for performing speech processing when powerful computers are available, but on limited computing hardware as the mobile phone, this option is unviable. Another distinct disadvantage of using ANN is that the whole system has to be re-trained on addition of a new entity.

2.1.4 Speech processing systems

The metrics mentioned in the previous subsection can be used for pre-processing on different types of speech processing systems. These systems might have different end goals and consequently might require different approaches for pre-processing. Some of the possible speech processing system goals might be the following:

- **Emotion Detection** A lot of research today is devoted to identifying the emotion in a human voice. Giripunje [14] discusses the use of Artificial Neural Networks for this purpose using metrics like Pitch, formant frequencies, energy and speaking rate. The aim of this work is to determine the emotional state such as anger, sadness, happiness, fear etc.

- **Gender Classification** Gender classification could be another objective of a speech processing system. It is a well studied problem that finds application in the fields of speaker indexing, speaker recognition, annotation and retrieval of multimedia databases, voice synthesis, human computer interaction, biometrics, robotics etc. Kotti et al [22] discuss gender classification from speech samples using support vector machines.

- **Speech Recognition** Speech recognition is another well studied speech application problem. The aim of a speech recognition system is to identify spoken speech. Such a system could be an exhaustive or a limited vocabulary speech recognition system. Speech-to-text conversion is one of the most popular applications of speech recognition.

- **Speaker Recognition** Speaker recognition is the process of automatically recognizing who is speaking on the basis of individual information included in speech signals. It can be divided into Speaker Identification and Speaker Verification. Speaker identification determines which registered speaker provides a given utterance from amongst a set of known speakers. Speaker verification accepts or rejects the identity claim of a speaker - is the speaker the person he/she claim to be?

While the above mentioned types of speech processing systems are fairly popular, there might be other applications with a different set of goals. The emphasis of this work is on speech recognition and speaker recognition on constrained devices, hence the focus from now on would be towards these two applications of speech processing systems.

2.1.5 Signal Processing

After the pre-processing stage, the audio signal is passed through the signal processing framework. This framework is responsible for doing majority of the work required for speech recognition or speaker recognition. Since this thesis basically deals with speech recognition and speaker recognition, this background will be discussed in detail. Many of the techniques for both speech recognition and speaker recognition are common. Speech recognition and speaker recognition, in general, both require models to be developed for a word or a speaker. Various techniques have been studied and used to develop such models. Some of the more popular ones are the following:

1. **Hidden Markov Models (HMM)** The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multidimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to the associated probability distribution. It is only the outcome, not the state visible to an external observer; therefore states are "hidden" to the outside, hence the name Hidden Markov Model. In order to define an HMM completely, following elements are needed:

   - The number of states
   - The number of observation symbols
   - A set of state transition probabilities

   In a typical speech recognition system based on HMMs, words are usually represented by networks of phonemes. Phonemes are the sounds that constitute each word. Each path in a word network represents a pronunciation of the word. The same phoneme can have different acoustic distributions of observations if pronounced in different contexts. The HMM model moves from one state to another depending on the probabilities ascertained from the training set. The end state is identified as the spoken word. Speaker recognition systems are similar, where a speaker instead of a word is modeled as an entity. There has been a lot of research for speech processing systems based on HMMs. They are generally considered to be a very good approach towards solving speech recognition or speaker recognition problems. For instance Jiang et al [18] describe large margin hidden markov models for speech recognition. Without going into the details, in their work, Jiang et al propose to estimate the HMM problem using a minmax function. They show that large margin HMM perform significantly better than traditional HMMs. Dumitru et al in [8] describe a speaker recognition system based on HMM for Romanian language.

Though HMMs have been proven to be fairly successful for speech recognition tasks, they need a fairly exhaustive training set. Also, building models is computationally expensive. The accuracy of HMM-based systems depends on the exhaustiveness of the training set. The bigger the training set, the more computationally expensive it is to build the system. Since this work focuses on speech processing systems embedded on a constrained device, this computationally expensive building of the models (even if only once) is not preferable. Further, with HMM-based systems, addition of any entity to the system needs the system to be re-trained. Thus HMMs are not a good choice considering the objectives of this work.

2. **Artificial Neural Networks (ANN)** An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. In general for a speech recognition system based on ANNs the features extracted from the training set are fed into the neural nets as input. Each output neuron represents a voice command or a word. The training is generally iterative and the weights are adjusted for each training set. Back propagation is carried out to minimize the error.

   ANNs suffer from the same drawbacks that HMMs do as far as speech recognition on constrained devices is concerned. Training has to be exhaustive and intensive. Any addition to the training set requires retraining for the whole system, which is a very costly process to perform on a device like a mobile phone.

3. **HMM/ANN Hybrid** Though both ANNs and HMMs have proven their usefulness in the field of speech recognition, they both suffer from a number of limitations. HMMs were the first modeling technique to be used for speech recognition. Its success lies in its mathematical simplicity; efficient and robust algorithms have been developed to facilitate its practical implementation. Standard HMMs model speech as a series of stationary regions in some representation of the acoustic signal. However speech is a continuous process, and ideally should be modeled as such. Furthermore, HMMs assume that state and phone boundaries are strictly synchronized with events in the parameter space, whereas in fact different parameters do not necessarily change value simultaneously at boundaries. As an alternate to HMMs Artificial Neural networks have been used. However, they have their own drawbacks. ANNs were found to be unsuccessful in dealing with long time-sequences of speech signals. This is because they are originally static qualifiers. They do not inherently model the temporal evolution of data and in speech processing they have to deal with the duration variability of speech units. While neural networks can readily be applied to acoustic modeling, it is not yet clear how to use them for temporal modeling. Some researchers have explored a new approach that combines HMMs and ANNs within a single, hybrid architecture. Joe Tebelski [37] states that the goal in hybrid systems is to take advantage of the properties of both HMMs and ANNs, improving efficiency and recognition performance. Therefore, Tebelskis et al [37] explore a class of systems called NN-HMM hybrids, in which neural networks perform acoustic modeling, and

HMMs perform temporal modeling.

In general, in such hybrid models, ANNs are used to estimate the transition probabilities that are then used when the system is modeled as a HMM. However, in terms of constrained devices, such hybrid models suffer from the same drawback as ANNs and HMMs.

4. **Gaussian Mixture Models (GMM)** In statistics, a mixture model is a probabilistic model for density estimation using a mixture distribution. A mixture model can be regarded as a type of unsupervised learning or clustering. When the distribution is assumed to be Gaussian, the model is called gaussian mixture model.

   Typically in a speech processing system modeled as a GMM, multidimensional statistical models are built for both speech recognition and speaker recognition. Each dimension is a feature(such as ZCR, energy amplitude, MFCC etc) and every entity has a value in all the dimensions. An entity here may represent a word for a speech recognition system or a speaker for a speaker recognition system. Clustering is later done for the formation of the model during training and shortest distance from the cluster identifies the entity during testing. Reynolds et al [31] describe a speaker recognition system based on GMMs. The focus of their work is on applications that require high identification rates using short utterances from unconstrained conversational speech.

   There are some distinct advantages to using a GMM-based model for the constraints mentioned in this work. GMM-based models can be trained with a very limited dataset as will be shown later in the experiments section. Further, addition of an entity to the model does not require re-training of the whole system. Thus the computational overhead can be reduced. While this is sufficient reason to adopt a GMM-based statistical model, this work does not intend to prove that GMM-based models are better suited to this task than any of the above as different techniques might be useful under different constraints.

Since, this thesis is concerned with a GMM-based speech processing system, a deeper look at a typical GMM based speech processing system is provided in this section. Typically, such a system has two main parts

1. Feature Extraction
2. Pattern Recognition

2.1.6 Feature Extraction

Feature Extraction is a special form of dimensionality reduction. When the input data to an algorithm are too large to be processed and it is suspected to be highly redundant (much data, but not much information) then the input data will be transformed to a reduced representative set of features (also named *features vector*). Transforming the input data into the set of features is called features extraction. If the extracted features are carefully chosen, it is expected that the feature set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input. In terms of audio processing, feature extraction involves information retrieval from audio signals. Since frequency is one of the most important pieces of information necessary to accurately recognize sound, audio signal is generally transformed from time domain to frequency domain to break the signal into its frequency components.

One of the most popular transformation techniques is the Fourier transformation. The Fourier Transform, in essence, decomposes or separates a waveform or function into sinusoids of different frequency that sum to the original waveform. It identifies or distinguishes the different frequency sinusoids and their respective amplitudes. The Fourier Transform is based on the discovery that it is possible to take any periodic function of time $x(t)$ and resolve it into an equivalent infinite summation of sine waves and cosine waves with frequencies that start at 0 and increase in integer multiples of a base frequency $f_0 = \frac{1}{T}$, where T is the period of $x(t)$. Fourier transformation is generally performed on continuous signals; even though speech is a continuous signal, sampling makes it discrete. Hence, Discrete Fourier Transformation (DFT) is used for transformation. For most computational purposes, the Fast Fourier Transformation algorithm is used. The fast Fourier transform (FFT) is a discrete Fourier transform algorithm that reduces the number of computations needed for points from $2 \cdot N^2$ to $2 \cdot NlgN$, where lg is the base-2 logarithm.

Another popular technique is the discrete wavelet transformation technique. The Wavelet Transform provides a time-frequency representation of the signal. It was developed to overcome the shortcoming of the Short Time Fourier Transform (STFT), which can also be used to analyze non-stationary signals. While STFT gives a constant resolution at all frequencies, the Wavelet Transform uses multi-resolution technique by which different frequencies are analyzed with different resolutions. A wave is an oscillating function of time or space and is periodic. In contrast, wavelets are localized waves. They have their energy concentrated in time or space and are suited to analysis of transient signals. While Fourier Transform and STFT use waves to analyze signals, the Wavelet Transform uses wavelets of finite energy. Though the wavelet transformation was developed to overcome the shortcomings of the fourier transformation, it has not been proven that wavelet transformation is better suited to speech processing. Furthermore, FFT is a well studied algorithm and computationally fast, which is why it is preferred in this work.

Mel frequency cepstral coefficients (MFCC) are used as the primary feature in this speech analysis framework. Different speech processing systems have experimented with various types of features for speech analysis. Due to the sheer number of such features, this section will concentrate on the feature used here that is Mel Frequency Cepstral Coefficients.

The word Mel comes from Melody and Mel scale is a perceptual scale of pitches. Mel Frequency Cepstrum (MFC), a representation of short term power spectrum of a sound based on linear cosine transform, and MFCC collectively make up the MFC. MFCCs have been shown to have good discriminatory properties for both limited vocabulary speech recognition and speaker verification. They come in various flavors(depening on the method of extraction) as shown in [12]. Each MFCC, once extracted, is a dimension in itself. Using these MFCCs, the GMM is built where every entity has a value for each MFCC dimension, and taken together, these define the multi-dimensional Gaussian mixture model. The details of how MFCCs are extracted in this framework are presented theoretically and graphically in the next chapter.

2.1.7 Pattern Recognition

After feature extraction, clustering is performed for pattern recognition. Clustering can be considered the most important unsupervised learning problem; so, as every other problem of this kind, it deals with finding a structure in a collection of unlabeled data. A loose definition of clustering could be the process of organizing data into groups whose members are similar in some way. Consequently, the aim of clustering is to determine the intrinsic grouping. Some of the main clustering algorithms are described in this section.

One of the simplest and most popular algorithms used for clustering is called the

K-means clustering algorithm [24]. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters. The number of clusters k is fixed before running the algorithm and k centroids are defined, one for each cluster. The placement of these centroids affects the results and hence should be decided with a lot of care. A thumb's rule is to place these centroids as far as possible from each other. The next step is to take each point in the data set and associate it to the nearest centroid. After this has been done for all the points in the dataset, first round is completed. At this point k new centroids are recalculated as barycenters (centre of mass) of the clusters resulting from the previous step. A new binding is then done between the same data set points and the nearest new centroid. Thus, a loop has been generated and is executed until no change in cluster centres is observed. Various implementations of Kmeans algorithm are freely available. An efficient implementation of the algorithm is given by Tapas et al[36].

Another popular clustering algorithm is the Fuzzy C-means algorithm. Its very similar to the K-means clustering algorithm, the main difference being that each point in the dataset belongs to each cluster with a degree of belonging. Thus, points on the edge of a cluster, may be in the cluster to a lesser degree than points in the center of cluster. Thus for each point x there is a coefficient giving the degree of being in the kth cluster. Usually, the sum of those coefficients for any given x is defined to be 1. This method was developed by Dunn [9]

Clustering has been well studied. There has been a lot of research on which clustering algorithm is better than the other [45]. Different clustering algorithms are suited to different types of datasets. In this thesis, different clustering algorithms were experimented with as has been explained in Chapter 4. Template matching is performed for pattern identification once the model has been built after training. The test data is compared with all the training sets and the one with the least Euclidean distance is considered to be the match. Dynamic Time Warping, a technique that uses both frequency and time domain characteristics (where as template matching uses only frequency domain characteristics) was also considered. But computational constraints allow don't allow for a very elaborate pattern matching. More details about this are presented in Chapter 4.

2.2 Related Work

Context-aware applications are touted to be the next big thing for mobile phones and provide a better experience through collaboration with each other [15]. The framework presented here can go some distance in providing audio input to such context-aware applications. There have been a few similar attempts in the past, albeit with a slightly different objective. We'll describe a few such attempts in this section.

1. **Sensay** SenSay, described in [34], is a context-aware mobile phone that adapts to dynamically changing environmental and physiological states. Sensay comes mounted with external sensors like accelerometer, light sensors and microphones mounted at different points of the users body. Sensay has abilities such as providing the remote caller with an option to convey the urgency of their call, make call suggestions to users when they are idle, provide feedback on the current status of the sensay user, manipulate ringer volume, vibrations and phone alerts. While some of the functions performed by sensay are not related to the objective of this thesis, functions like manipulation of ringer, vibration and phone alerts can be performed by any application making use of the output from the framework presented in this thesis. Furthermore, these functions can be performed without the need of mounting extra sensors on the user at different places, which can clearly be cumbersome. These functions can be performed by measuring the loudness of the

environment using the built-in microphone and taking pictures and measuring the brightness levels. Apart from measuring the loudness of the audio signal, the framework is also capable of differentiating speech and silence from noise in indoor environments. This can be vital input to some applications and will be discussed in more detail in the discussion section.

2. **Exploration of small enrollment speaker verification on handheld devices** This work [43] focuses on low enrollment speaker verification in the context of very limited training data. It discusses the importance of normalization techniques, such as handset normalization and zero normalization as well as model training methodologies to minimize the detrimental impact of highly variable environment and microphone conditions on speaker verification robustness. The focus is on speaker verification process from a security point of view rather than speech recognition and speaker verification for providing contextual information to business applications that is the focus of this thesis. Furthermore, the speaker verification process takes speaker's utterance and his or her identity as input and outputs whether or not they were both a match. These methods require a lot of computational resources to this one aspect of speech processing. Hence, while similar mathematically this is not directly applicable to the framework presented in this thesis.

3. **Fixed-point GMM-based Speaker Verification over Mobile Embedded System** As has been mentioned previously, speech processing is not a novel area for research. Although by no means a solved problem, the topic has been the focus of research for quite a few years now. Long before devices like PDAs and mobile phones came into existence there has been research in this field focussing on performing speech processing on powerful desktop computers. Due to the difference in the hardware configuration of handheld devices

and powerful desktop computers, algorithms employed for speech processing could not be directly ported to handheld devices. Moon et al [26] describe the implementation of a speaker verification system on a PDA. They describe the step-by-step porting of an existing speaker verification system on a desktop computer to a PDA. They make use of fixed point implementation to speed up the existing speaker verification system. This is similar to the framework presented in this thesis, because a GMM based model is built using Mel frequency cepstral co-efficient. However, the difference lies in the fact that fixed point implementation has been used. Though faster for computationally constrained devices, fixed point implementation of any algorithm has its drawbacks. Fixed point implementations introduce a tradeoff between accuracy and performance and these have been studied in detail [23].

4. (Commercial Systems) Commercial applications are also available in this area though none of them have been a huge success or considered the answer to the problem of speech processing. Due to their commercial nature its not possible to comment on their technical details. Commercial systems developed by Scansoft and Nuance (two of the pioneers in this field) are tailored towards industrial applications such as health care and financial sectors, that require high levels of security to protect sensitive customer account information. In addition to providing security, commercial speaker verification systems allow companies to reduce costs by replacing expensive live call centers with automated systems for speaker verification.

Chapter 3

Pre-processing

This chapter focuses on the design issues, algorithms and techniques that are used to pre-process an audio signal before it is further processed by applications such as word recognition or speaker recognition. One of the objectives of pre-processing is to categorize the audio into a number of categories. Once the audio is determined to be in the speech category, start and end point detection is performed on the audio recording. Pre-processing is done in two stages:

1. Categorization
2. Start and end point detection for speech recordings

An audio signal can be categorized into various classes depending on different parameters chosen. For instance, a possible categorization is shown in Figure 3.1. This is by no means an exhaustive categorization but it serves to illustrate the purpose of pre-processing. The images on the right hand side are waveforms in time domain of a sample from each category.

Figure 3.2 depicts the data flow within the pre-processing module. One of the advantages of this architecture is that the design is modular so different modules for

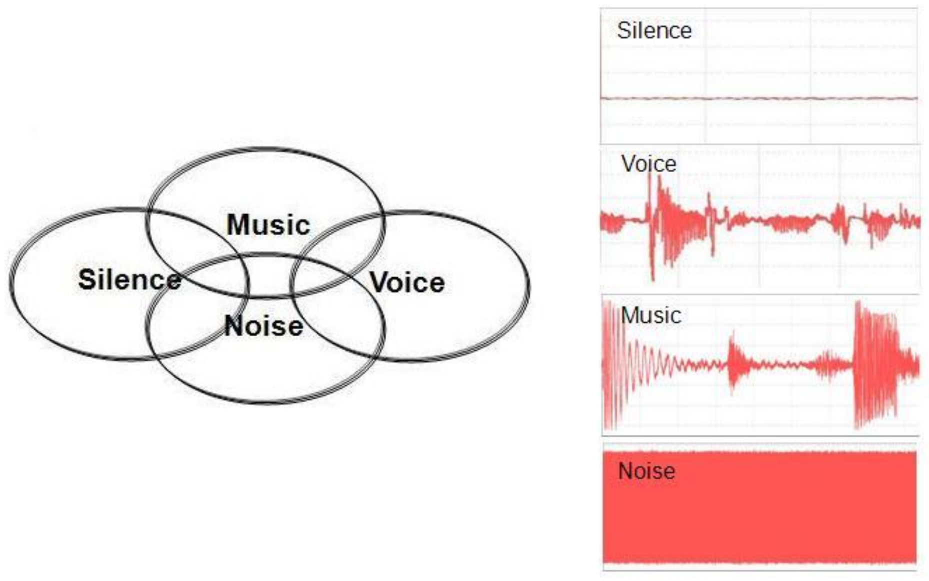

Figure 3.1: A possible categorization of Audio

deriving context-aware information can be added. For example, a module for differentiating between noise and music could be added to increase the functionality. New modules can also be added before feature extraction such as Vocal Tract Normalization (VTN) [39], Dynamic Time Warping (DTW) [16] for speech recognition, or modules for gender classification. These modules can then work independently of the existing modules thereby adding more functionality.

Each module can provide different applications with the sort of information that it needs. For example the silence or non-silence (speech or noise) classification may be used to determine whether or not the user is sleeping. There's a clock built in the phone, and if the time of the day is late at night and this module concludes there is silence then it is probably safe to assume that the user is sleeping. On the other hand if the user is traveling in a car and pre-processing suggests that there is noise in the background then the volume of the ringer can be adjusted to suit the noise levels of the environment. This stage of pre-processing not only makes

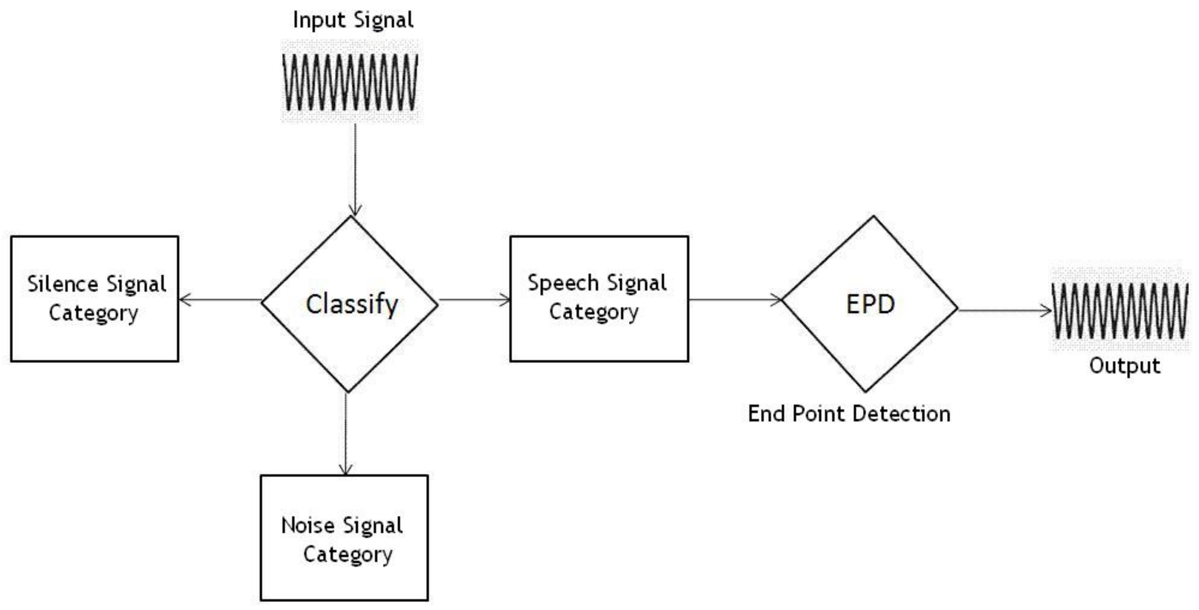

Figure 3.2: Data flow within the speech processing module. The square boxes represent the categories into which the audio signal could be classified and the diamond box represents the decision making unit.

speech detection much more efficient, but it also provides contextual information. This forms the first stage of pre-processing.

For our proof of concept applications, we instantaneously classify the input audio signal as silence, speech or noise during the first stage of pre-processing. For our proof of concept applications and those applications that need to do speech recognition or speaker recognition, the pre-processing module in the second stage does end point detection and then extracts the portion out of the original recording that has speech content.

3.1 Stage I

The audio signal captured by the microphone is sent through the classification module to identify whether the captured audio recording is silence, has speech

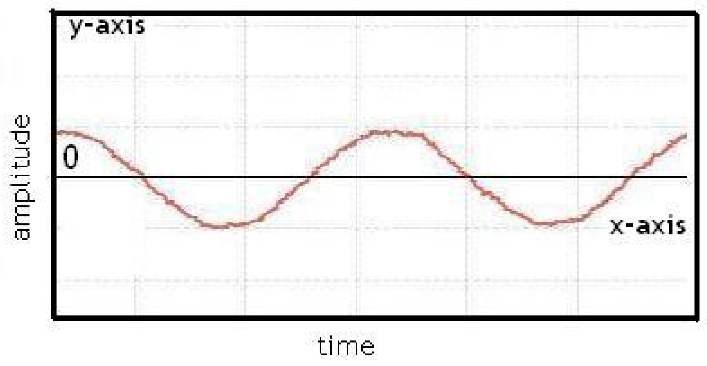

Figure 3.3: Zero crossings

content, or is noise. Different functions can then be performed depending on the class of the signal. If the audio signal is identified as having speech content, it undergoes further processing to extract meaningful information from it. As has been mentioned in the previous chapter, pre-processing stage can make use of a lot of features to determine the category of an audio signal. In this work, we make use of two features:

1. Short term energy

2. Zero crossing rate skew

Short term energy is the energy present in the audio signal over a very short duration. While short term energy is sufficient to classify a signal as silence, its not sufficient to derive any meaningful information out of it. Zero crossing rate skew is used to further categorize the audio into noise or having speech content.

Zero crossings is defined as the number of times an audio waveform in time domain crosses the zero line. As the name suggests, zero crossing rate (ZCR) is the number of zero crossings per unit time. For instance the number of zero crossings in Figure 3.3 are four. Zero crossing rate is determined over the entire signal by counting the zero crossings and then dividing them by the length of the signal measured in units of time. The signal is then broken into small frames and the ZCR for each frame is measured. ZCR skew is defined as the difference between the number of frames whose ZCR is greater than the mean ZCR and number of frames whose ZCR is less than the mean ZCR. This is given in Algorithm 1

Algorithm 1: Calculation of ZCR skew

```
Data: Total zero crossings $zc$, Total time $t$, Frame length $f$, Zero crossings
      for each frame $zf$
Result: ZCR skew
// Calculate mean ZCR
$mz \leftarrow \frac{zc}{t}$ ;
// Initialize counters
$p \leftarrow 0$ and $n \leftarrow 0$ ;
while all frames have not been processed do
    if $\frac{zf_i}{f} > mz$ then
        $p = p + 1$;
    else if $\frac{zf_i}{f} < mz$ then
        $n = n + 1$;
    else
        Ignore this frame and move to the next
    end
end
$skew = |p - n|$;
return skew
```

Algorithm 1 describes how ZCR skew is calculated. It begins by calculating mean ZCR over the entire signal. Then each frame is processed separately to measure the ZCR for that particular frame. The number of frames whose ZCR is greater than the mean is stored in p, and the number of frames having ZCR less

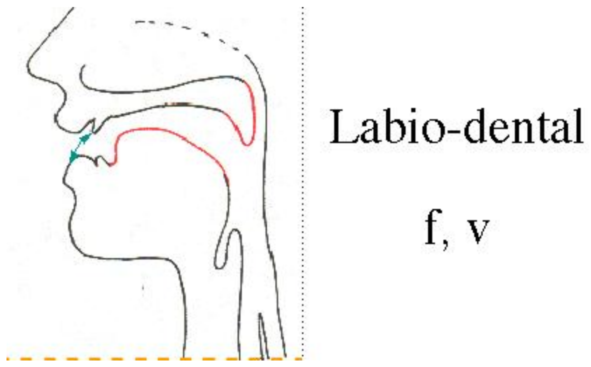

Figure 3.4: Fricatives ‘f’ and ‘v’ [17]

than the mean is stored in n. The difference of these two provides the skew. In actual implementation, for counting the number of positive frames, ZCR in a frame is compared with $mz + X$, and for counting negative frames, ZCR in a frame is compared with $mz - X$. X is a small percentage of the mean value that is added or subtracted to smooth out minor fluctuations and get a stable value for the skew.

This skew is higher for speech than other categories because of the presence of fricatives in human speech. Fricatives are sounds of consonants, such as ‘f’ or ‘v’ in English, produced by the forcing of breath through a constricted passage. For both of these sounds, the constriction is caused by upper teeth touching the lower lip, as depicted in Figure 3.4. The fricatives are high energy sounds and are easy to detect. ZCR skew catches this property of sound to a certain extent. For, silence or uniform noise the ZCR skew for obvious reasons will be relatively small. Variable noise is not considered in this work, because without a noise pattern, distinguishing variable noise from speech would be computationally very expensive and, thus, is not possible during the pre-processing stage.

Figure 3.5 shows the plot of ZCR-per-frame on y-axis and frame-number on x-axis for an audio recording that has speech content. The dark band in the center is the mean$\pm$’X’. All the points above the band are positive frames and all the plots below the band are negative frames.

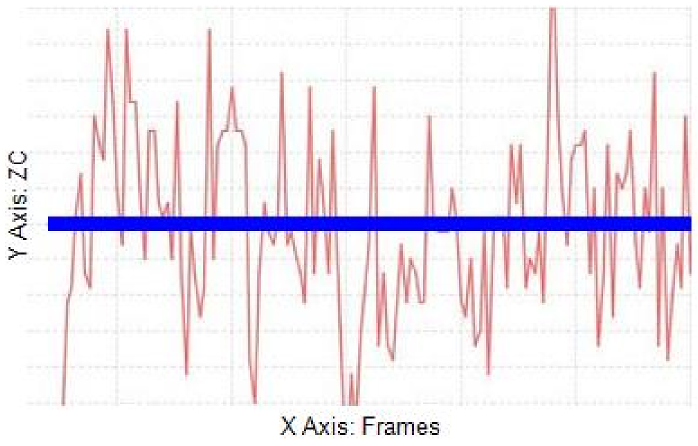

Figure 3.5: Plot of ZCRframe vs Frame number

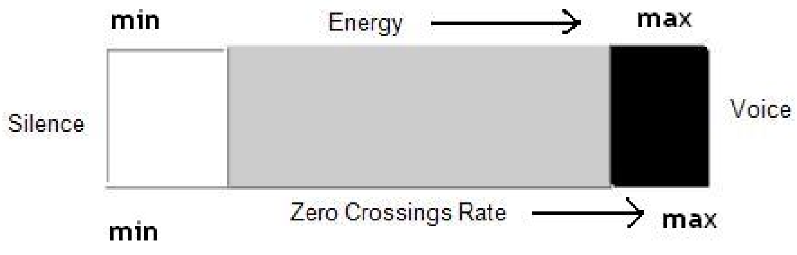

Figure 3.6: Categorization into Silence, Noise and Speech

After Stage I pre-processng categorization as represented by Figure 3.6 is achieved. The grey color in the center represents different types of noise. The white on the left hand side is meant to denote silence and the black at the other extreme depicts noise. The arrows in front of short term energy and ZCR skew represent the increase in the values of these parameters from silence to speech.

3.2 Stage II

If stage I decides that the recorded audio signal has speech content, second stage of processing is performed. This stage is responsible for detecting the start and the endpoint of speech content in the audio signal and then "trimming" it. Trimming means removing those parts of the audio signal from the beginning and end that do not have speech content. Naturally, if the speech content lasts from the start to the end of the recording no trimming will be necessary. Otherwise trimming is necessary because the next module (speech processing) transforms the pre-processed speech signal from the time domain to the frequency domain. Fast fourier transform algorithm is used to achieve this transformation. The transformation module, which is discussed in detail in the next chapter, forms the bottleneck in terms of performance. Therefore, the length of the speech signal needs to be kept small to achieve reasonable transformation performance. Second stage of pre-processing reduces the length of the audio signal drastically, thereby increasing the performance many-folds. Though this stage is important for future processing of the audio signal, by itself it doesn't provide any useful information. Therefore, a lot of processing time or power should not be spent at this stage.

Algorithm 2: Start point detection

```
Data: Array containing samples from audio signal to be processed
Result: Starting frame number for speech content
Divide input signal into $nf$ frames ;
$TotalEnergy = 0$ ;
forall All the samples in the audio signal do
    $TotalEnergy = TotalEnergy + {amplitude_i}^2$ ;
end
Compute total energy of the signal ;
// Calculate mean energy
$me = \frac{TotalEnergy}{nf}$ ;
for $i = 0$ to $nf$ do
    if $FrameEnergy_i \leq \frac{me}{4}$ then
        // Skip $n$ frames
        i = i + n ;
    else
        Mark $i$ as possible starting frame ;
        // Check next $n$ frames
        for $j = 1$ to $n$ do
            if $FrameEnergy > \frac{me}{4}$ then
                Confirm $i$ as the starting frame;
                return $i$ ;
            else
                $j++$;
            end
        end
    end
end
return $i$ ;
```

This section presents a simple algorithm to detect the start and end point of speech content in an audio signal. It is simple in the sense that it doesn't require a lot of computation power, it is fast, and it re-uses some of the statistics already collected in the previous stage of pre-processing. Stage II of pre-processing assumes that there might be periods of silence at the start and end of a recording and it strives to remove these periods of silence if they exist. On desktop computers this step might be done to improve efficiency of feature collection during speech processing, but on mobile phones it assumes even greater importance as it enforces efficient use of scarce resources.

The start and end-point detection algorithm 2 and 3 uses short term energy of the audio signal for removing the silent frames from the beginning and the end. A silent frame is defined as a frame whose energy is less than a certain threshold. The value of this threshold varies from implementation to implementation, we've chosen 25% of the mean energy of the entire signal, as our threshold value.

Algorithm 2 starts from the first frame and then proceeds by checking the frame energy initially after every n frames. The aim is to avoid having to check every frame, thereby speeding up the computation. This might cause deviation of a maximum of $n-1$ speech frames from the actual start of the speech content, which is an error that is considered acceptable for small values of n. The assumption here is that when the algorithm reaches the part of the signal that has speech content, then any consecutive n frames will have more than one frame that will be identified as non-silent frames by the algorithm. This serves two purposes. Firstly, each frame need not be checked, and secondly, a false non-silent frame is not marked at the start point.

Algorithm 3 for detecting end point is similar to the start point detection. The algorithm begins by testing every nth frame from the start point till it finds a silent frame. The next n frames are then tested for the presence of any non-silent frame, whose energy is greater than 25% of the mean energy. If all of the next n frames are found to be silent frames then the initial frame is marked as the end point. If however one of the frames turns out to be non-silent while checking the consecutive n frames, then the current index is updated to this frame's index and the algorithm reiterates.

Algorithm 3: End point detection

```
Data: Array containing samples from audio signal to be processed, Starting
      frame number $sf$
Result: Ending frame number for speech content
for $i = sf$ to $nf$ do
    if $FrameEnergy_i > \frac{me}{4}$ then
        // Skip $n$ frames
        i = i + n ;
    else
        Mark $i$ as possible starting frame ;
        // Check next $n$ frames
        for $j = 1$ to $n$ do
            if $FrameEnergy > \frac{me}{4}$ then
                $i = i + j$;
                Break ;
            else
                $j++$;
                if $j == n$ then
                    return $i$ ;
                end
            end
        end
    end
end
return $i$ ;
```

This algorithm was tested under indoor conditions with moderate noise and was

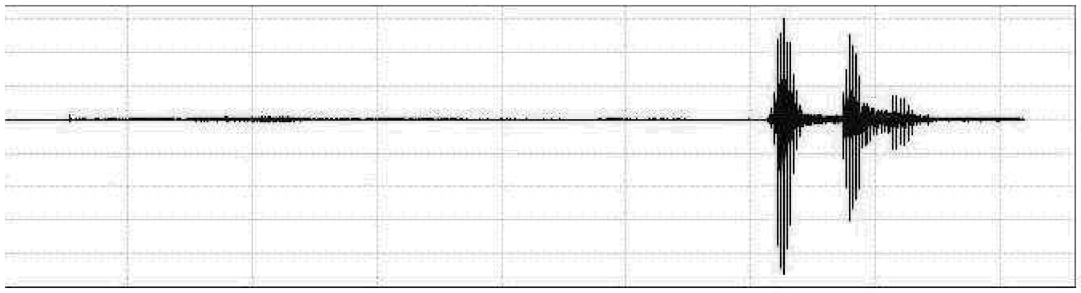

Figure 3.7: Audio recording of 3 seconds with the word ‘back’ in it

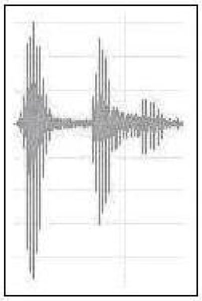

Figure 3.8: Extracted speech after passing the 3 sec long recording through the start-end point detection stage

found to be working satisfactorily. Test results are presented in Chapter 5. There are other algorithms for end-point detection that are robust in high noise environment that we discussed in Chapter 2. These can easily be integrated in this module but they will increase the computational complexity, and are, therefore, not considered in this thesis.

Figure 3.7 represents the word ‘back’ spoken in an audio recording which lasted 3 seconds. Figure 3.8 represents the extracted speech portion from this audio recording. Both of these are plots in time domain of the audio signal. The x-axis is time and y-axis is the amplitude of the signal. The parameter n and threshold can be tuned to make the algorithm more strict or relaxed in terms of detecting the start and end point. A relaxed algorithm would run faster but could be off by a few frames while detecting start and end point. The strictest algorithm would have $n = 1$ as the parameter value.

Chapter 4

Speech Processing

In this chapter the techniques that are employed for speech processing are discussed in detail. Starting from feature extraction to vector quantization, each step is presented in detail. The term 'speech processing', as used in this chapter, refers to both speech recognition and speaker recognition, since a majority of the steps involved for both of these are identical although some differences exist. These differences will be highlighted as and when needed. Speech processing starts with the trimmed audio signal that is generated at the end of pre-processing as discussed in chapter 3

A simplified architecture of the speech processing module is presented here in Figure 4.1. The design is modular and the implementation of each of the sub-systems shown is also modular. Pipes-and-filter design is specially evident in this speech processing module, and different components are designed to be re-usable and replaceable. This architecture is then used for both training and testing.

The input speech is the output from the pre-processing sub-system described in the previous chapter. This audio signal is then fed to the Feature Extraction module. Once the feature set is extracted from the input signal, it is then sent to the Classification module, which is responsible for measuring the similarity of

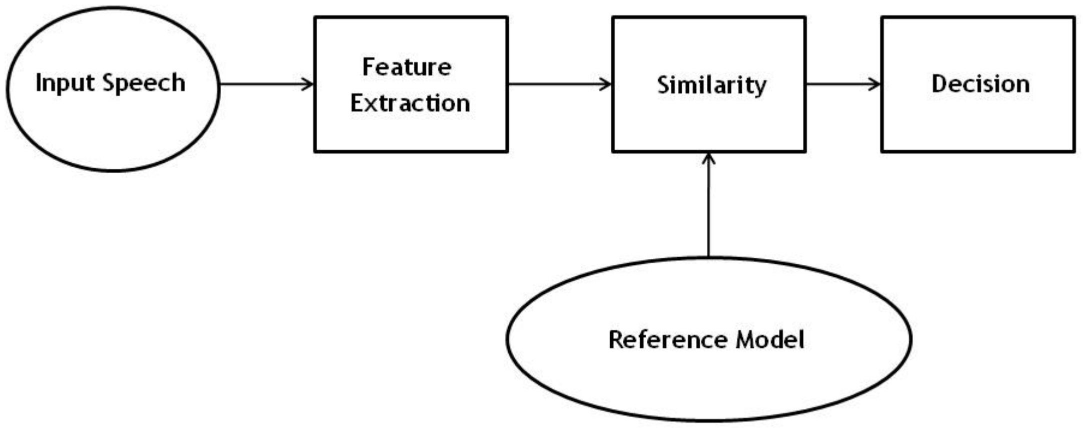

Figure 4.1: Architecture of the speech processing module

the extracted feature set to the models already present in the reference set. The computed measure of similarity is passed on to the Decision module that decides on a positive or a negative match. The reference models are computed in the same way as has been just described and are already present in the system before it's used. The following sections look at each of these components in greater detail.

4.1 Feature Extraction

The procedure to extract MFCC from audio signal is common to both speech and speaker recognition. In fact, one of the reasons for selecting MFCC as the primary feature for extraction is the fact that MFCC is a popular feature for extraction which has been used for both speech and speaker recognition. Different number of MFCC need to be extracted for speech recognition and speaker recognition, as explained later.

Speech processing is illustrated through the application of speaker recognition. Figure 4.2 shows two speakers uttering the same phrase that is a few seconds in length. Y-axis represents the amplitude of the signal and X-axis represents the

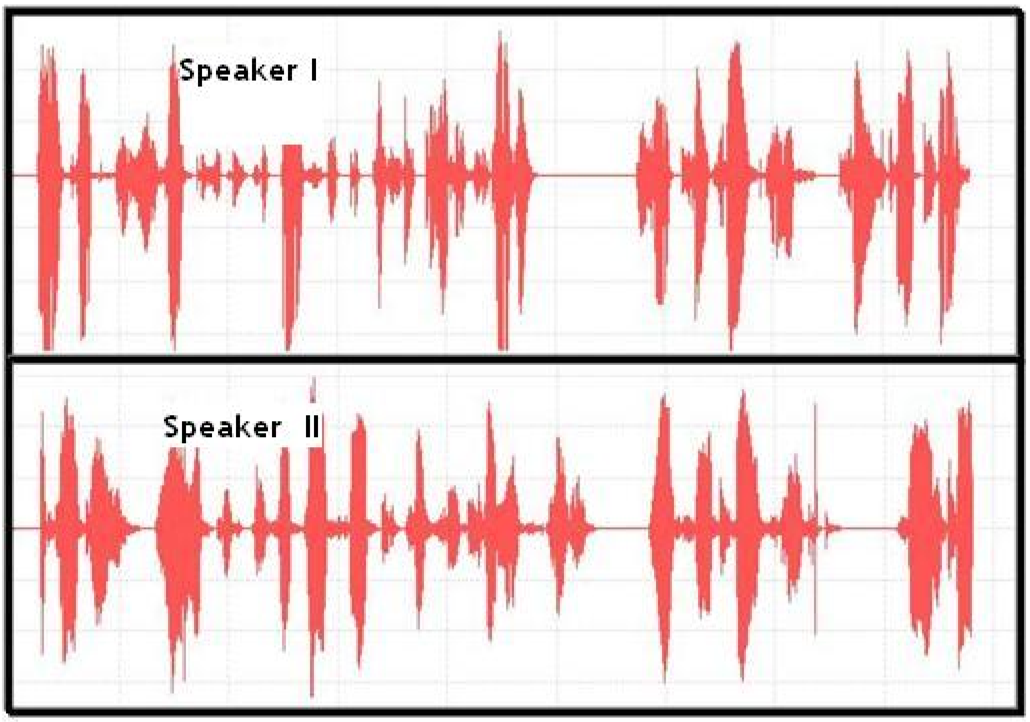

Figure 4.2: Two speakers uttering the same phrase

different samples in the utterance. The utterances shown in Figure 4.2 will be passed through different stages of the extraction procedure shown in Figure 4.3. The output from each stage will be graphically shown and each step explained in detail in the following subsections.

4.1.1 Segmentation

Segmentation is the process of breaking the speech signal into small peices called frames which are then processed separately. Speech is fundamentally a non-stationary signal, so small segments of speech signal are cut and speech is approximated as a stationary random process during each frame. Speech applications typically choose a framesize of somewhere between 8 milliseconds to 30 milliseconds. After some experimentations a fixed framesize of approximately 10 milliseconds was chosen. In the segmentation step, entire speech is blocked into frames. A blocked frame is a

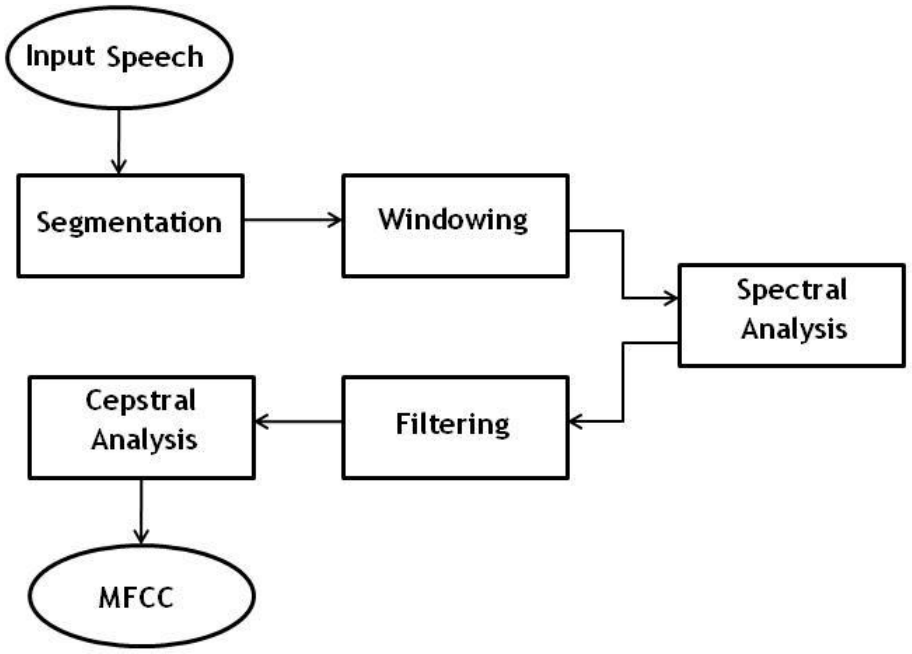

Figure 4.3: Extraction procedure for MFCC

frame which overlaps partially with the next frame.

Overlapping successive frames emphasizes the features in each frame and allows for a smoother extraction of features. The amount of overlapping varies from system to system and depends on the amount of speech content expected to be present in the signal. The more speech content expected in the signal, the higher the amount of overlapping is done although there is no precise way of determining how much overlapping is optimal. In this thesis, frames overlap by 50%. The number of frames with a 50% overlap can be calculated as follows:

$$nFrames = \frac{2 \times SampleLength}{FrameLength} - 1$$

Figure 4.4 shows frames from two different speakers. Y-axis represents the amplitude of the speech signal in time domain and X-axis represents the sample number in a frame. These frames have a 50% overlap and have been zero padded

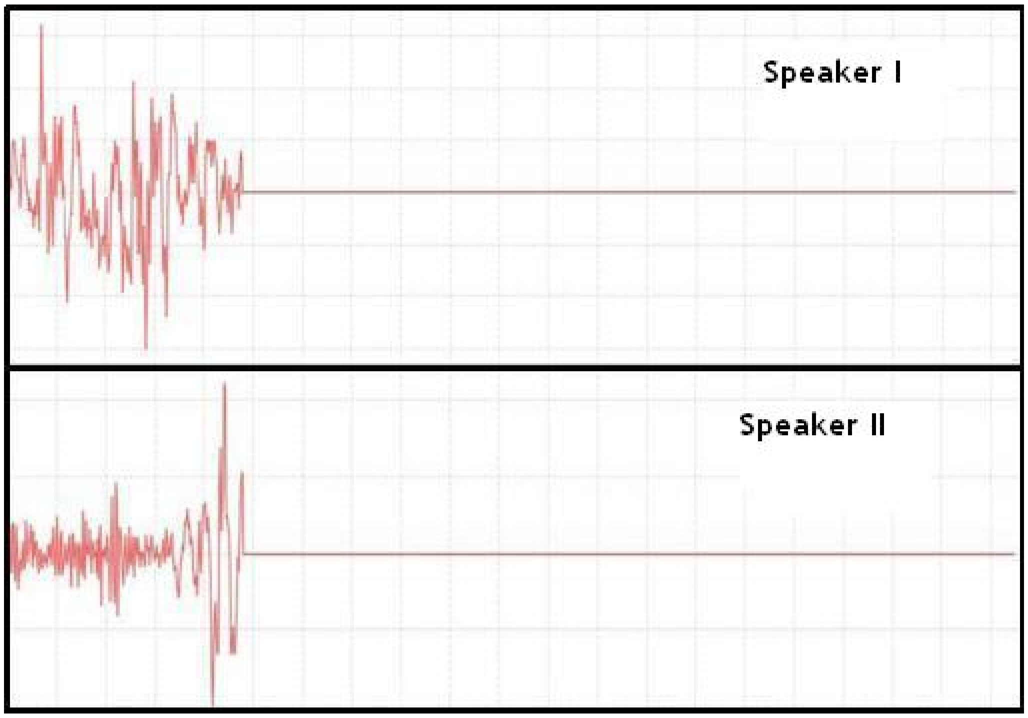

Figure 4.4: Speech frames with 50% overlap and zero padding.

to 1024 samples. This is done to quickly transform the signal from time domain to frequency domain using the Fast Fourier Transform (FFT) algorithm. Transformation from time domain to frequency domain is a computationally expensive process. FFT is a fast way of doing this but it requires that the length of the signal to be transformed be a power of 2. That is why each frame of the speech signal is padded to 1024 before transformation.

4.1.2 Windowing

In the next step of feature extraction, windowing is performed on all the frames to minimize spectral distortion caused by framing. Each frame of the original function is multiplied with a *window function* also called *tapering functions.* These functions are called *window functions* because they exist only inside a window and evaluate to zero outside some chosen interval. When multiplied with the speech frame they taper down the beginning and the end to zero to minimize spectral distortion. :

Some of the popular windowing functions used are:

- Hamming window function
- Hann window function
- Cosine window function
- Lanczos window function

In this work a Hamming window function is applied to the frames to minimize the discontinuity of the signal at the beginning and at the end caused by framing in the previous step. The hamming window function used in this thesis is given by the following equation where n represents

$$F(n) = 0.54 - 0.46 \cos(\frac{2\pi n}{N-1})$$

the sample where n represents the sample number in the frame and N is the total number of samples in a frame.

Figure 4.5(a) shows the hamming window function [20]. The x-axis for the window function represents the samples and the y-axis represents the amplitude of these samples.

Figure 4.6 represents the same two frames from from the example earlier after the hamming window function was applied to them. When compared with Figure 4.4, which shows the same frames before the hamming function was applied to them, the smoothened edges at the beginning and end are easily noticeable.

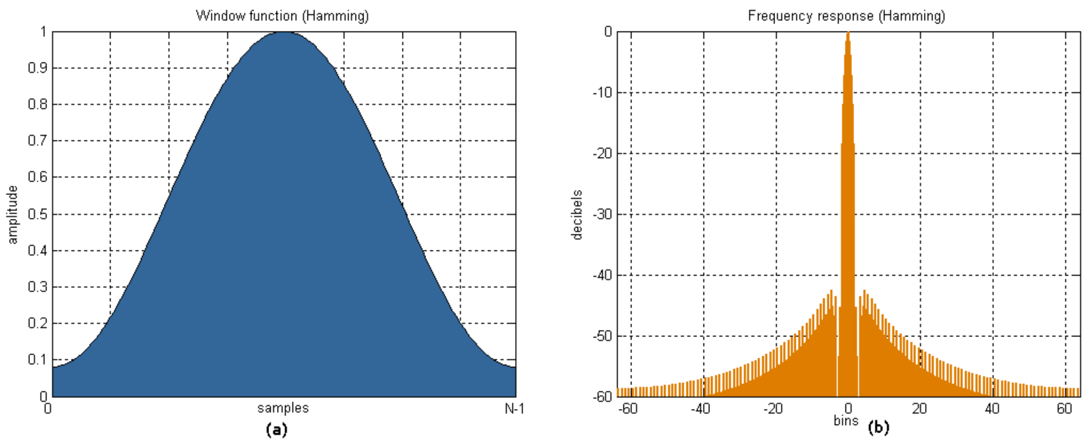

Figure 4.5: Hamming window function and frequency response

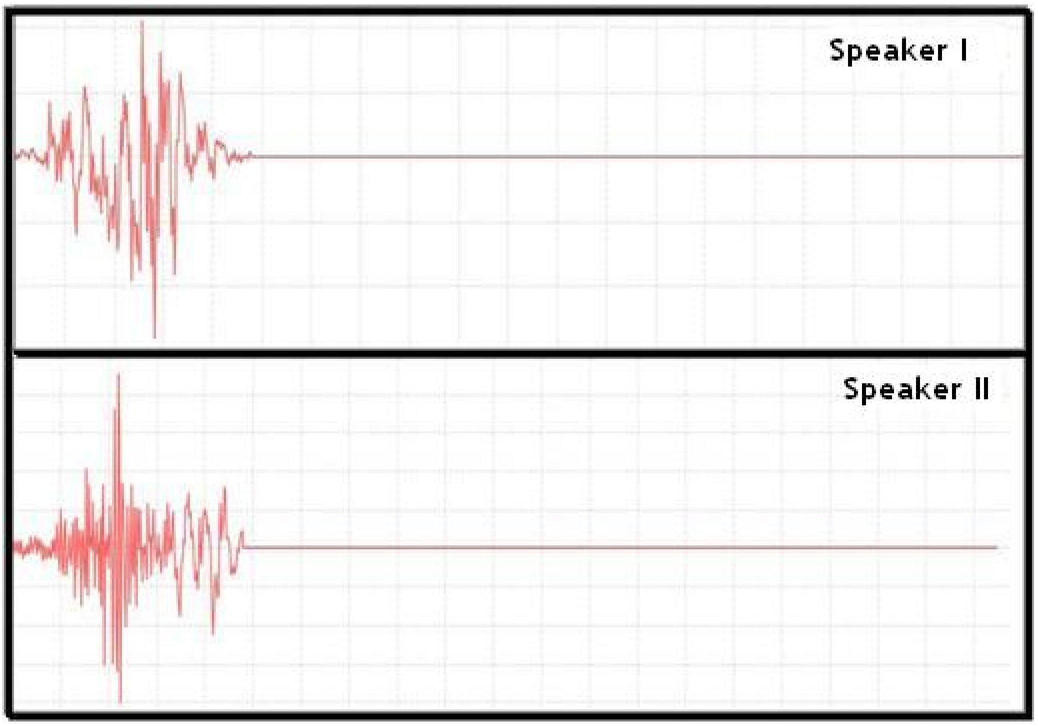

Figure 4.6: Speech frames after windowing

4.1.3 Spectral Analysis

After windowing, each frame is transformed from the time domain to the frequency domain using Discrete Fourier Transformation (DFT). The frequency domain response is called the *spectrum* and the fast fourier transform (FFT) algorithm to obtain the spectrum. The zero padding of frames up to 1024 samples per frame performed during the framing step enables the use of FFT to obtain the spectrum, as FFT requires the number of samples per frame to be a power of 2. The fourier equation governing the transformation from time domain to frequency domain is given by the following:

$$F(\nu) = \int_{-\infty}^{\infty} f(t)e^{2\pi\nu t}\, dt$$

MFCCs can be obtained by taking either a fourier transform or a short term Linear Predictive Coding (LPC) transform. LPC is a tool like DFT to obtain the spectral envelope of a digital signal. This is done by modelling the system as a linear predictive model. Linear prediction is an operation where future values of a discrete-time signal are estimated as a linear function of previous samples. Various studies have been undertaken to compare the performances of LPC-based vs FFT-based MFCCs [40, 41]. There is no general consensus as to which one is better. Some reports claim comparable performance while others indicate a slightly better performance for LPC. Without getting involved in the discussion of which one is better, FFT based MFCC are chosen in this thesis only because it is easier to implement.

Finally, Figure 4.7 represents frames from the previous example, transformed to their frequency domains. Notice that it is easy to discriminate between the two speakers by comparing the dominant frequencies in this figure which can be identified by looking at the frequencies with high amplitude 4.7. However, this is

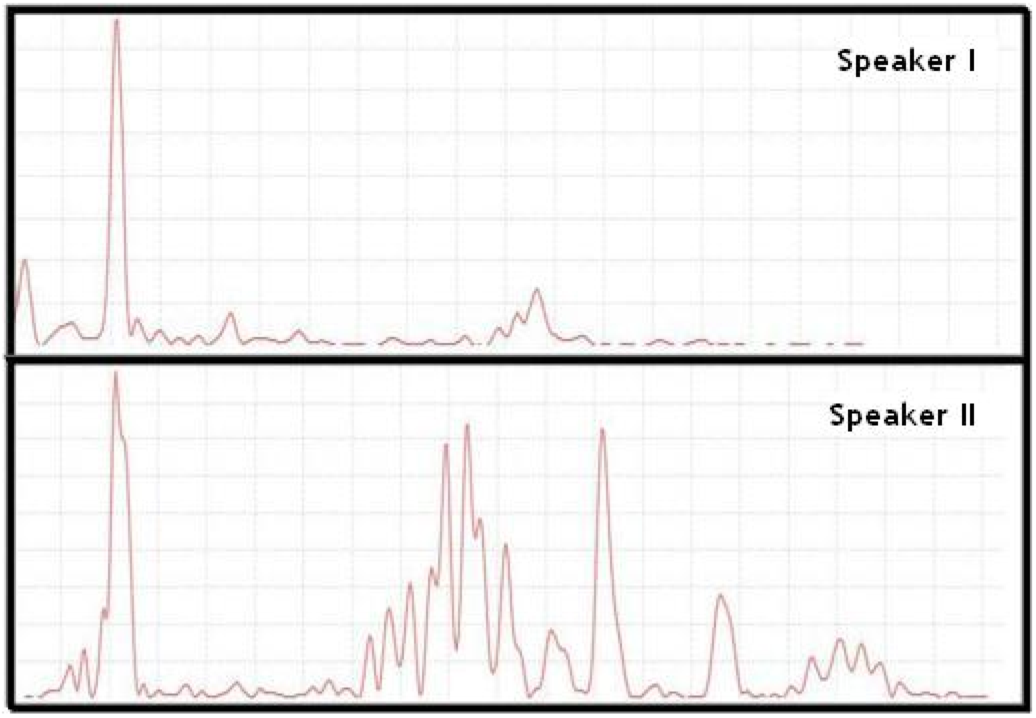

Figure 4.7: Frames after fast fourier transformations

just one frame from the entire utterance. When all the dominant frequencies over the entire utterances are summed for both speakers, the difference may not be so prominent. Hence, its not possible to identify the entities using frequency response alone.

4.1.4 Filtering

In this step of feature extraction, each frame (now in frequency domain) is passed through a filterbank (set of filters). These filters are equally spaced on the Mel scale and are called Mel filters. Mel scale is a perceptual scale of pitches judged by listeners to be equal in distance from one another. The name mel comes from the word melody to indicate that the scale is based on pitch comparisons.

This scaling to mel scale is performed because equally spaced frequency bands on the Mel scale approximate the human auditory system's response more closely

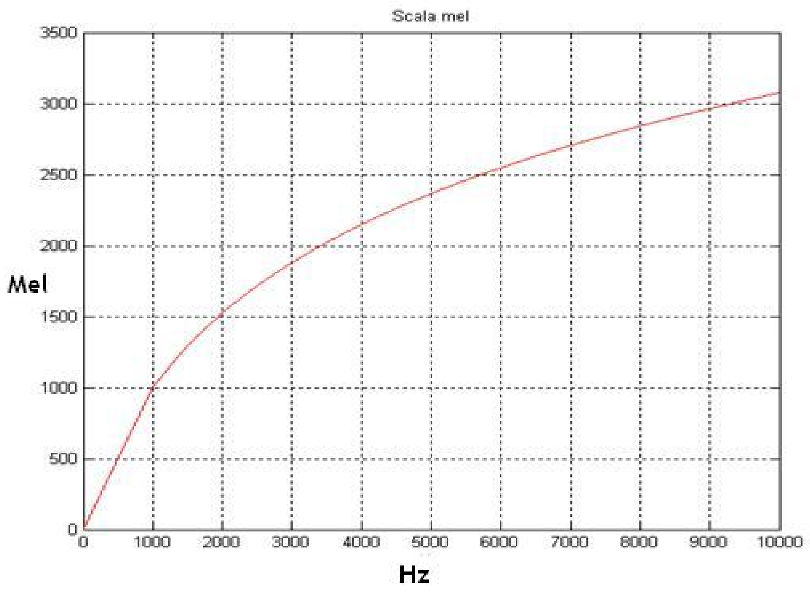

Figure 4.8: Mel scale vs linear frequency in Hertz

than the linearly-spaced frequency bands. The scaling of linear frequency to mel scale is governed by the following equation:

$$Mel(f) = 2595 \times \log_{10}(1 + \frac{f}{700})$$

where f represents linear frequency before warping. Equal spacing on Mel scale translates to linear spacing from 0 to 1000Hz but logarithmic spacing after that as shown in Figure 4.8.

This is the first step that is different for limited vocabulary speech recognition and low enrollment speaker recognition. For limited vocabulary speech recognition it was observed during the experiments that a few MFCCs were enough to differentiate one word from another, but for speaker recognition acceptable accuracy levels were obtained when at least 30 MFCCs were extracted from the speech signal. This point is relevant because the number of MFCCs that can be extracted from a speech signal is directly proportional to the number of used Mel filters as will be shown shortly.

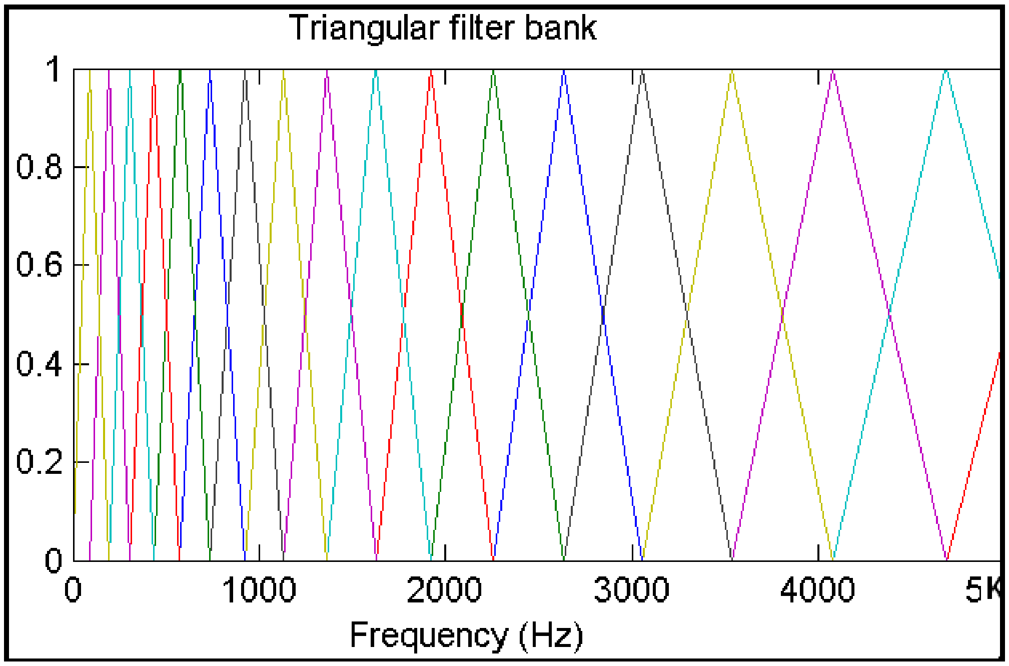

Figure 4.9: Triangular Mel filterbank

Figure 4.9 shows Mel filters on a normal frequency scale where x-axis represents the frequency in Hertz and y-axis represents the scaling response of these filters. The range in Figure 4.9 goes from 0 to 5000 Hz but during implementation the range depends on the number of Mel filters to be extracted. Mel filterbanks can be defined by three parameters:

1. Starting Frequency F_s
2. Ending Frequency F_e
3. Number of Filters M

Since human voice lies between 200Hz and 16Khz, this range is a good choice for speech processing applications. However, one must take into account the sampling frequency of the recorded audio before deciding on these parameters. The maximum frequency has to be lower than half the sampling rate as dictated by the Nyquist-Shannon sampling theorem. In Mel frequency domain, these filters are equally

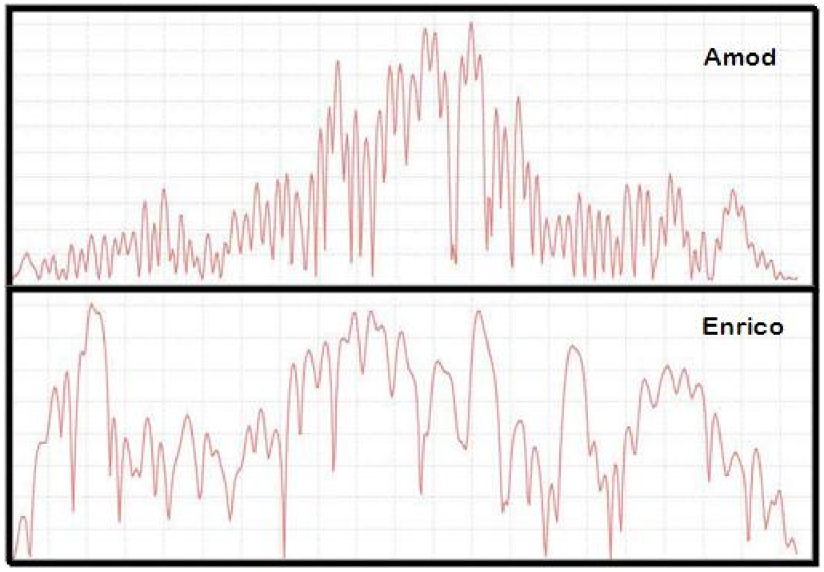

Figure 4.10: Speech frames from two different speakers after filtering

spaced and overlap each other at the center. The start and end point of one filter is the mid-point of its neighbors. The end point of these filters is governed by the following equation:

$$F_s + i \times \frac{F_e - F_s}{M + 1}$$

where i goes from 0 to 1 and M represents the total number of filters. F_e and F_s are frequencies on Mel scale.

The speech frames shown in Figure 4.7 when passed through the filterbank shown in Figure 4.9 are converted to frames shown in Figure 4.10.

4.1.5 Cepstral Analysis

The word Cepstral in Mel Frequency Cepstral Coefficients comes from the analysis that is performed after filtering, which will be described in this section. Speech signal is often assumed to be the output of a linear time-invariant system that is the convolution of the input and impulse response. Convolution is a mathematical operation on two functions which produces a modified version of one of them. In speech processing, one of the functions is the input and impulse response the other one. De-convolution is necessary to characterize or model the system. Cepstral analysis is a common procedure used for such de-convolution [33].

During cepstral analysis (de-convolution), the power of the filterbank is summed and their logarithm is taken to compress the sum. Mathematically, it is represented by

$$X_i = \log[\sum_{k=0}^{N-1} |X(k)| \cdot H_i(k)] \quad \text{i=1,2...M}$$

where M is the total number of filters, N is the number of samples in a frame, $H_i(k)$ represents the response of the filters calculated in the previous section.

4.1.6 Discrete Cosine Transform

Discrete Cosine Transform (DCT) is needed to convert the data back from frequency domain to time domain. The mathematically correct procedure is to use Inverse Fast Fourier Transform (IFFT) or apply Principal Component Analysis (PCA). However, since the signal is real, DCT is used because its computationally less expensive than IFFT or PCA. Discrete cosine transform expresses a sequence of finitely many data points in terms of a sum of cosine functions oscillating at different frequencies. The equation for Discrete Cosine Transformation is,

$$C_j = \sum_{i=1}^{M} X_i \cdot \cos(j \cdot (i - \frac{1}{2}) \cdot \frac{\pi}{M}) \quad \text{j = 1,2...J}$$

where J is the number of MFC coefficients to be extracted, M the number of filters in the filterbank and X_i the log energy output of the i^{th} filter.

As has been mentioned before, MFCCs are abstract mathematical quantities and may or may not correspond to something tangible in the real world. Each of these coefficients can be treated as a dimension and these dimensions are calculated for every frame. Once quantized, these dimensions can then be easily compared for different words or different speakers. As was noted, the number of dimensions or MFCCs to extract will depend on the problem. A good rule of thumb is that $J < M$. Also, the first MFCC coefficient is normally ignored for computations and comparisons as it represents the average loss in energy [27]. Finally, if the two frames that have been used so far to illustrate the MFCC extraction process were treated as the entire signal then the MFCCs extracted from them would look like Figure 4.11. The dashed line in the graph represents the x-axis. It immediately becomes clear that the y-axis is the value of these MFCCs and that these values can be both negative and positive. Since each MFCC is a separate dimension, we obtain a discontinuous graph as shown in Figure 4.11.

4.2 Decision and Similarity estimation

In Gaussian Mixture Model (GMM) based pattern recognition problems like speech recognition or speaker recognition, the next step after feature extraction is to normalize the extracted features. Normalization of features is important for vector quantization. Normalization could mean different things like spreading the data points obtained after feature extraction or compressing them inside a given range. Normalization is performed to facilitate the formation of clusters and reduce error during pattern matching. Vector quantization or clustering is performed to create codebooks for every entity (a word or a speaker) that exists in the system. A codebook represents the entity in a system in terms of its feature vectors. These codebooks are then used for pattern matching, which is explained below.

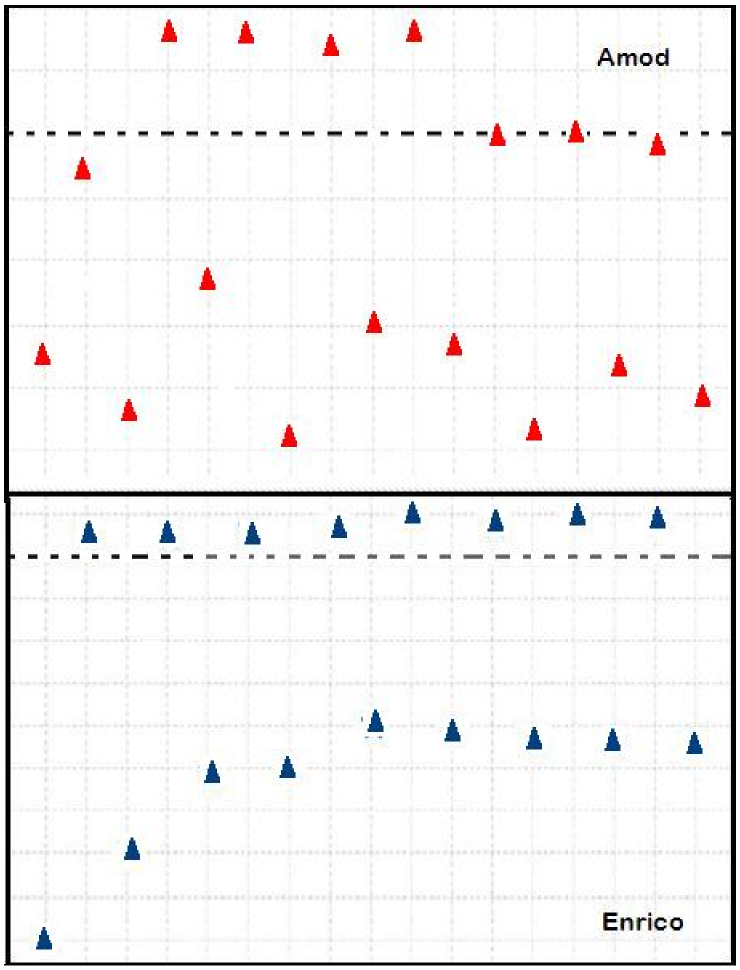

Figure 4.11: 17 MFC coefficients in a frame for two different speakers

Speech processing applications have, in the past, used different techniques for normalization, such as a weighting matrix. Normalization with weighting matrices involves multiplying all the MFFCs extracted with a weighting matrix to spread the vectors in a given range. This step helps in comparing different sets. Initial experiments were performed with weighting matrices but after some experiments it was observed that linear weighting produced better results. This was because, each MFCC is a dimension in itself and is only spread along one dimension (i.e changes only in amplitude), along y when plotted as a graph with amplitude on the y-axis and MFCC coefficient number on the x-axis. Hence for normalization instead of multiplying the feature set with a matrix, the feature set was multiplied by the equation of line. Furthermore, due to the constraints of the device, linear weighing provided a computationally cheaper option.

After normalization, clustering is performed. The clustering algorithm needs to cluster these MFCCs into different clusters. Clustering tends to expose a trade-off between accuracy of representing an entity and the computational expense of comparing a test entity with a reference model. Experimentation with different algorithms, pre-seeded, k-means [24], kmeans++ [1] led to some interesting observations. One of the clusters always forms around the mean and others close to each other above and below the mean, regardless of the training set. Moreover, if the number of clusters was increased to anything beyond 20 then a lot of cluster centers ended up having 0 members. This was true for all the data sets used in the experiments. This peculiar behavior is observed because MFCCs are spread in one dimension and they tend to group at certain places; hence if the number of clusters is preset to a large value a lot of cluster centers end up with no members.

All the MFCCs from different frames are clustered together to create what is called as codebooks. Different frames have different values for each MFCC. While Figure 4.11 shows the MFCC coefficients derived from a speech frame for

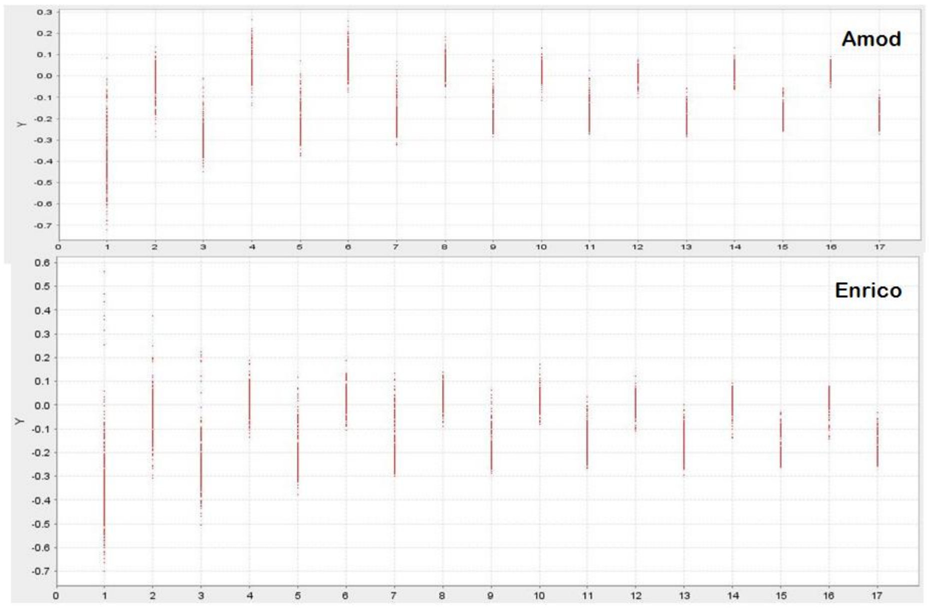

Figure 4.12: MFCC spread for the original utterances for two speakers

two speakers, Figure 4.12 shows the MFCCs extracted from all the frames of the utterance in Figure 4.2.

After experimenting with different clustering algorithms for clustering and observing the above mentioned behavior a pre-seeded clustering algorithm, which is called 'Regional Clustering', was used in this work. The central observation which led to the development of regional clustering and the (pre seeded cluster centers) was that in each dimension clusters were formed at the mean and the rest were spread close to each other above and below the mean. Clusters between these three groups were observed to be very sparse. Hence for regional clustering, the clusters were pre-seeded with their centers at the mean, positive mean and negative mean. Positive mean is defined as the mean of all the MFCCs for that particular dimension that lie above zero. Similarly, negative mean is defined as the mean of all the MFCCs for any dimension that lie below zero. One argument that could be made against this type of pre-seeding is that for those dimensions where no MFCCs lie above or below zero, the cluster would be empty. Such cases were encountered very rarely.

After this step, a codebook is created for every entity that exists in the system. Each codebook representing an entity, has three values in it.

1. Cluster center value at mean cm
2. Cluster center value at positive mean cp
3. Cluster center value at negative mean cn

That is, for a speech recognition system an entity would be a word. Similarly for speaker recognition the entity would be a speaker. While using the system, for instance when a speaker speaks, a codebook is created for the unknown speaker and distance calculated between the codebook for the unknown user and each of the codebooks already in the system. The speaker corresponding to the codebook with the least distance from the codebook for the unknown speaker is identified as the match. Distance between two codebooks C_1 and C_2 is defined as:

$$Distance(C_1, C_2) = \sqrt{(cm_1 - cm_2)^2 + (cp_1 - cp_2)^2 + (cn_1 - cn_2)^2}$$

where cm_i represents the center at mean for codebook i, cp_i denotes the positive mean and cn_i denotes negative mean for codebook i. A conscious effort was made to not build a complicated classifier owing to the processing constraints of the device. Figure 4.13 represents the codebooks prepared from the utterances shown in Figure 4.2 for two speakers. In Figure 4.13 x-axis represents the extracted features denoted by their coefficient numbers, and y-axis represents their values. Larger circles indicate two or more clusters formed very close to each other.

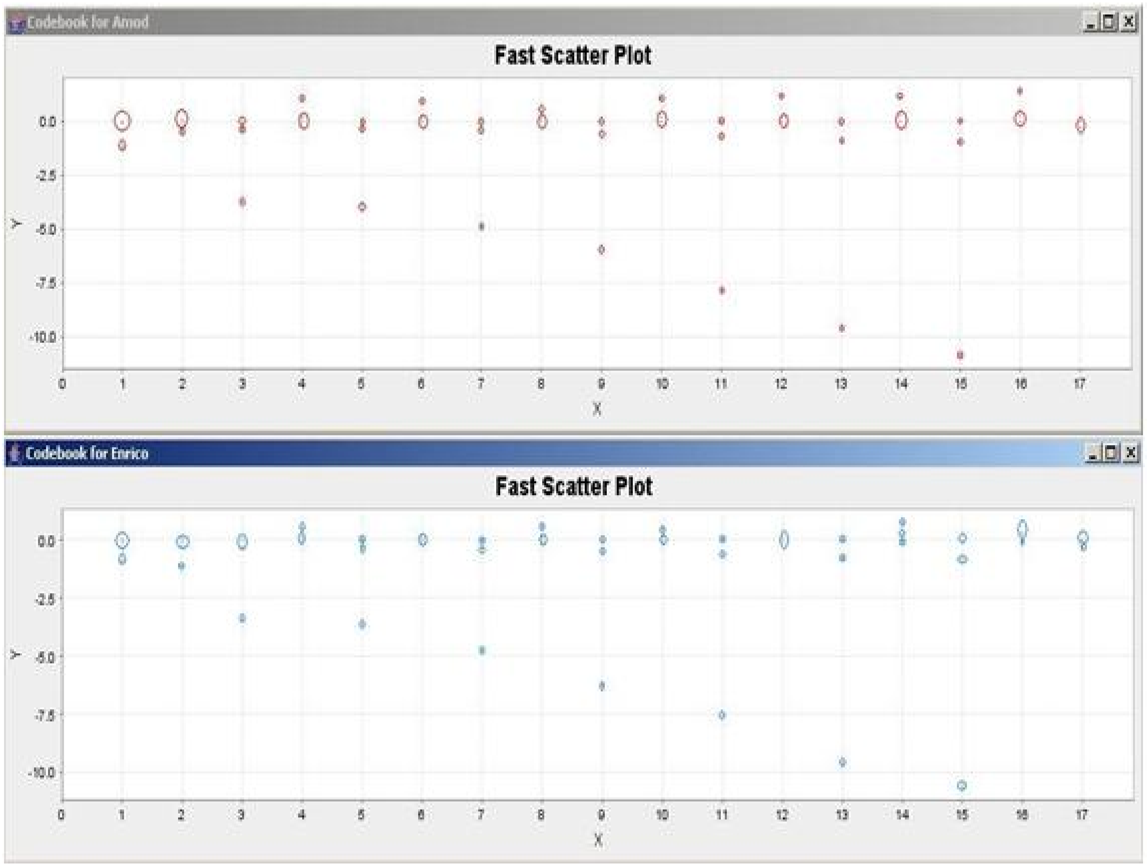

Figure 4.13: Codebooks for two different speakers

Chapter 5

Experiments and Results

In this chapter the experiments and accompanying results to test various modules of the system, described in the previous chapters, are described. These experiments measure different aspects of the system, such as memory consumption while execution, performance in terms of time taken during execution and accuracy of results. The experiments for different modules will be presented in different sections.

5.1 Classification Experiments (Module I)

The aim of the following experiments is to test the performance of the pre-processing module. The objective is to answer questions like, 'How long does it take to classify an input signal as silence, speech or noise?' or 'What is the percentage of misclassification?'. Three categories are used for classification :

1. *Silence* is the category where the input signal does not have any speech in it.
2. *Noise* is the category where the input signal has some noise that cannot be identified clearly as speech. It could be speech or it could be noise from some other source.

3. *Voice* is the category where the input signal has speech content.

To achieve this, these experiments aim to measure two things:

1. The percent accuracy of classification
2. Processing time required for classification

Experimental setup comprised of an indoor environment (office or home). The pre-processing module was loaded on a Nokia N95 and a Nokia N80 cellphone. Samples from the Silent and Voiced categories were used to measure accuracy of classification and processing time required by the device to classify the input in one of the three categories - Silence, Noise or Voice. Training set comprised of n samples from each the silent category and the voiced category. The value of n was varied in different experiments to get an idea of the dependence of classification accuracy over the size of training set. More than one speaker were used to test the system. Specifically, the speaker testing the system wasn't necessarily the one whose voice had been used to train it. Data for training and testing was obtained by making different speakers speak into the device. While recording samples for 'Voice' category the device was put on the table and not held close to the mouth. The idea was to replicate the situation where a phone kept on the table can identify human speech in its environment.

5.1.1 Accuracy of detection for Module I

In the first experiment *two* samples from the silent category were used to measure the accuracy of classification. The aim here is to find out how many of these silent samples would be classified as voice or noise. The reason for using only two samples is to find out if the system could perform at acceptable level with minimal training.

Of the 43 samples used about 83% of samples were correctly detected as silence while around 16% which were incorrectly classified as noise. None of the silent samples was classified as voice.

In the next experiment *two* samples from the voice category were used to measure the accuracy of classification. The aim here is to find out how many of these voice samples will be classified as silence or noise. Of the 32 samples used for testing about 81% of samples were correctly detected as voice while about 19% were misclassified as noise. Once again, none of the voice samples was classified as silence.

To gauge the effect of the size of training set on the accuracy of classification, the same experiment was performed again in the same setup with the difference that instead of two, ten samples from each category were used to train the system. Of the 31 silent samples used to test the system about 81% of samples were correctly detected as silence while around 19% were incorrectly classified as noise. None of the silent samples was classified as voice.

Similarly to measure the accuracy of the system for detection of voice when trained with 10 samples, 31 samples were used to test the system. Of these about 90% of samples were correctly detected as voice while around 9% which were incorrectly classified were classified as noise. None of the silent samples was classified as voice.

The following conclusions can be drawn from the above experiments.

1. *Increasing the number of training samples does not result in a better accuracy for detecting silence.* The reason for this is that silence has a very steady profile in terms of energy and ZCR contour (the two parameters used for detection). Samples are very similar to each other and increasing the number of samples does not necessarily provide any more information.

2. *Increasing the number of training samples results in a better accuracy for detecting voice.* We see that the percentage increase in accuracy for detecting voice increases from lower 80% to 90% when the number of training samples is increased from 2 to 10. This happens because, unlike silence, it is hard to capture the nature of voice even for a single speaker in 2 samples. More samples provide more information and therefore lead to an increased accuracy. However there is a limit to the number of samples that can be used for training. Keeping in mind that this is just a pre-processing module and the user might not want to spend a lot of time on training the system we did not test it with more than 10 samples in the training set. However, if this module was used with a feedback loop in an application, the accuracy of detection would get better with time. This is because with a feedback loop, the number of samples for training would increase with the usage.

Another minor point worth noting is that, even though the algorithm is prone to mistakes as shown by the accuracy levels, it never commits the gross miscalculation of classifying silence as voice or vice versa.

By training the system with n samples from each category, the system builds a model for each category. The higher the value of n, the better and more accurate is the model. Instead of allowing the device to calibrate this model automatically, values describing this model were hard coded. The hard coded values were obtained by observing the experiments performed. To test this set up, 12 samples each from silence and voice category were used. This experiment resulted in a 100% accuracy but one limitation of the experiment was that the test samples were recorded in the same environment in which thresholds were determined for hardcoding. When the same thresholds were used in a moderately noisy environment outside a library, or inside an office with people talking or very noisy environment (inside a bus) the accuracy levels dropped alarmingly. On using 2 samples to train the device, accuracy levels in the range of lower 80% were observed for both silence and noise in moderately noisy environment. However, detection was impossible in very noisy conditions like inside a bus. This was because the noise was so overwhelmingly loud that it completely enveloped the profile for both silence and voice.

This technique can be further improved to handle high noise environment or additional modules can be added for the same purpose but that would increase the computational complexity and the classification would not be instantaneous. Being the pre-processing module, the classification needs to be instantaneous and increasing the accuracy in noisy environments at the expense of slowing down the classification in all environments is not an acceptable tradeoff for this thesis.

5.1.2 Processing time

The aim of these experiments described below was to determine how much processing time was taken to classify an audio clip as silence, noise or voice. The experimental setup described in the previous section was used for these experiments too. System clock built in the mobile device was used for measuring the processing time for classification.

This experiment was performed in 4 sets with about 30 samples in each set. Each set differs from the other in terms of the environment in which the samples were collected. The processing time for each audio clip is directly proportional to its length. Irrespective of the category the length of each audio clip, was between 3 to 4 seconds. The results are presented in the Table 5.1.

To calculate these processing times, time for each sample in each set was measured in milliseconds. Afterwards, for each set, the highest and the lowest processing time was discarded and the mean for each set was computed. 5.1 show that the classification process is instantaneous and satisfies the time constraint of the

Table 5.1: Processing time in Milli seconds

SetNo.	Processing Time	Number of Samples
1	55.51	32
2	73.41	43
3	67.50	31
4	59.83	31

pre-processing module.

5.2 Experiments for Start and End point Detection

Module II is the end point detection module and is responsible for narrowing down an audio clip to the part which contains speech. By trimming down the audio clip to only that part which contains speech, higher accuracy and efficiency is obtained since the system has to process a smaller audio clip. Higher accuracy is obtained because the models built for speech recognition and speaker recognition are not influenced by the silent portion of the audio clip. Higher efficiency is obtained because the modules after this step have to process smaller clips. Therefore, efficiency is measured in terms of shortening of the audio clips. For instance if the input audio clip is i units in length and the start and end points are given by s and e respectively then,

$$Efficiency\% = \frac{i-(e-s)}{i} \times 100$$

To measure the efficiency, audio clips of about 3 and a half seconds (88000 bytes) are collected (longer clips are trimmed by hand). These clips have one word spoken somewhere in them to mimic a scenario where the user gives a command to the device. In this experiment 5 words of different lengths are used and data are collected. Before proceeding, lengths of audio clips and words are measured in terms of number of samples. Each sample is stored in 2 bytes. So, for a byte array of 88000 elements the length is 44000 samples.

$$NumSamples = LengthByteArray \div 2$$

Table 5.2 represents the data about each word measured after start and end point detection. The *Length* column in Table 5.2 represents the length of each word measured in terms of the number of samples. The *% length* column represents the percentage of the time the window is occupied by a word. To compute this percentage, lengths of each word were measured 5 times and mean of all the lengths computed.

Table 5.2: Start and end point detection module

Word	Internet	SMS	Menu	Hornbook	Back
Length	4640	5540	3560	4220	2360
% length	10.54	12.5	8.1	9.6	5.4

Each audio clip initially recorded was 44000 samples long and the percentage efficiency achieved by reducing the audio clip to the lengths mentioned above is shown in Figure 5.1.

This reduction in size matters a lot, because in the next step, Fourier transform of the entire audio clip is taken. That is the slowest step, and any increase or decrease in the length of audio clip being sent for Fourier transformation affects the performance directly.

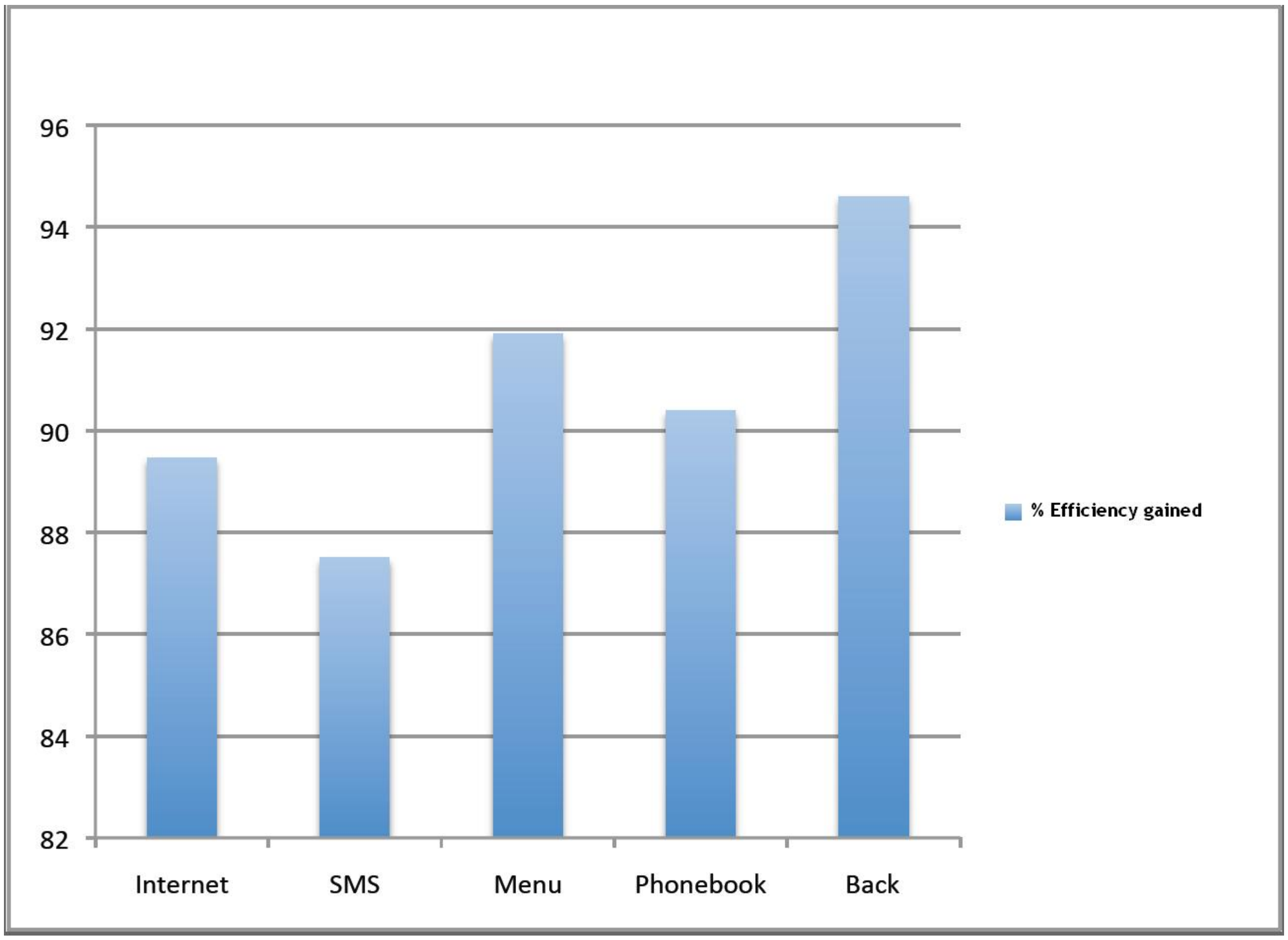

Figure 5.1: Efficiency gain in terms of reduction in size for five words

5.3 Experiments for Speech Processing Module

In this section experiments related to both low enrollment speaker recognition and limited vocabulary speech recognition are described. These experiments test the prototype of speech recognition and speaker recognition applications built using the algorithms described in the previous chapters. The statistics presented in all the experiments are mean values derived by repeating the same experiments a number of times. The aim of these experiments is to measure the accuracy of recognition (word recognition and speaker recognition) and time taken for processing. This section begins by describing the preliminary experiments first. They are preliminary, because, apart from gathering data for measuring the performance of the system they were also used to fine tune different parameters of the system like the number of MFCCs to be extracted or the length of the audio clip to be used.

5.3.1 Text Independent Speaker Recognition I

This experiment was aimed at testing the accuracy of recognition and time taken for processing during a speaker recognition process. This process was text independent because it did not depend on the content of what was spoken. A different audio clip (4 sec long) was used for training and 4 to 5 audio clips (each different in content), also 4 sec long, were used for testing. The number of audio clips depended on the availability of the testers hence for some users there were 4 while for others there were 5 clips. The number of MFCCs extracted were varied over different experiments because this number would directly influence the accuracy of detection and processing time. The experiments were performed in the same experimental setup as the previous ones. In the first experiment 18 MFCC (Mel Frequency Cepstral Coefficients) were extracted. The accuracy levels obtained were around 60%.

5.3.2 Text Independent Speaker Recognition II

The same experiment was performed again with a 15 sec long speech. This was to make sure that the low levels of accuracy obtained are not due to the small length of the audio input speech. This time 20 MFCCs were extracted instead of only 18. This was done to gauge if increasing the number of coefficients by a small value would have any drastic effect on the accuracy of recognition . However the accuracy increased by only 5% to become 65%. Though a greater number of MFCC are expected to increase the accuracy and this number is increased in subsequent experiments, due to the computational expense involved and the limited hardware available on a mobile phone, it is not possible to extract a very large number of MFCC.

Hence it was not the length of speech that was responsible for the low levels of accuracy. What was instead responsible was lack of proper training, but in an application without a feedback loop, it cannot be assumed that the user would take the trouble to train the device with 15 seconds of speech multiple times. The whole process of classification as voice, end point detection and speaker recognition took over 2 minutes on the device. These experiments were also found to be very sensitive to environment noise. Therefore, it was decided not to proceed in the direction of "Text independent speech recognition".

5.3.3 Text Dependent Speaker Recognition

Next, experiments to determine the accuracy and processing times for text dependent speaker recognition were performed. In the first experiment 18 MFCCs were extracted from an approximately 12 sec long utterance. Since this is text dependent speaker recognition the system was tested with utterances with the same sentences. This does not mean that the same audio clip was used for both training and testing.

In fact, it means that the audio clips used for training and testing had the user speaking the same sentences. The system could detect the correct speaker with a 71% accuracy.

After seeing the unacceptable processing times, the system was broken down into different modules which were run independently to discover the bottleneck. As expected, the time to frequency domain transformation step was found to be the bottleneck.

When compared with the experiment for text independent speaker recognition where the accuracy levels were around 60%-65% , though the accuracy increased slightly to 71% it is still not good enough. Moreover, it still took over 90 seconds to process on the phone. The reduction of about 30 seconds in processing time was due to shortening the length of the audio clips from 15 seconds to 12 seconds. From the preliminary experiments described so far, the following conclusions can be drawn:

1. As has been stated in the experiments described above, utterances of three different lengths, 4 seconds, 15 seconds and 12 seconds respectively were used. The number of MFCCs to be extracted was varied only by a small number to ensure that it did not have any effect on the processing time. Comparing the different times it took to process the utterances of different lengths, it can be seen that processing times depend directly on the length of the utterance. This is also the expected behavior utterances with longer length will have more frames that need to be transformed from time to frequency domain. Therefore, the first conclusion to be drawn is that the length of utterance would have to be reduced significantly since the devices were taking an unacceptable time to process it.

2. Even though the length of utterances was varied from 4 sec to 15 seconds and

experiments were performed with both text independent speaker recognition and text dependent speaker recognition systems, acceptable accuracy levels or recognition were not obtained. Therefore, it can be concluded that the accuracy of speaker recognition depends primarily on the number of MFCCs extracted and 18-20 MFCC coefficients are not enough to characterize a speaker. More coefficients will have to be extracted to improve the accuracy.

At this point one of the major concerns was to reduce the processing time and the following optimizations were performed to achieve that.

Algorithmic Optimizations

1. Windowing step was removed during feature extraction. This step would ensure that every sample in the audio clip will not be multiplied by a complex valued function and thus computational time could be saved.

2. Linear line weighing was used to spread the MFCCs instead of the matrix weighing. The reasoning behind this has been explained previously and will not be mentioned here again.

3. Pre-seeded clustering algorithm with a fixed number of cluster centers , described in the previous chapters was used. This would save time since the algorithm will not have to count the number of cluster centers during every iteration.

4. Zero padding before FFT was reduced from 1024 samples per frame to 512 samples per frame. This is the optimization that sped up the computation drastically. This is to be expected because FFT is the bottleneck and this optimization halves the amount of samples to be transformed from time domain to frequency domain.

Coding Optimizations

1. Strength reduction - Replacing multiplicative and division operators with bitwise operators wherever possible. Bitwise operators are faster.

2. Sub-expression elimination. All common expressions were calculated once, outside the loop and not multiple times inside the loop.

3. Loop Unrolling. ‘For’ loop of the following form

   ```
   for(int i = 0; i < vector.length; i++){...}
   ```

   was changed to

   ```
   int N = vector.length;
   for(int i = N; -i > 0; ){...}
   ```

   The rest of the code was adjusted accordingly. This resulted in comparison with ‘0’ which is always faster than comparing with any other number and it also resulted in one less condition to be evaluated.

After making these changes, the experiment for text dependent speaker recognition were performed again.

5.3.4 Text Dependent Speaker Recognition II

Different speakers were made to speak a very small utterance like one sentence containing 5-7 words. Four sets of this utterance were used to train the system. The number 4 was not chosen for any mathematical reason, it was because of the limited availability of the speakers . 30 MFCCs were extracted to form the speaker profile. Recognition rates of up to 78% were achieved and the processing time came down to just over 5 seconds. All the changes and optimizations were not employed at once. Incremental improvements in time were observed but nothing significant till the zero padding before FFT was reduced from 1024 to 512 per frame. Since, the bottleneck had already been located, this was not unexpected.

5.3.5 Limited Vocabulary Speech recognition

Experiments similar to the ones that have been described above were also performed for limited vocabulary speech recognition. One of the first questions that springs to mind on reading limited vocabulary speech recognition is "How Limited?". The answer to this question is, using a smart architecture like a 'dictionary of pages' a greater number of words can be accommodated. In this type of an architecture there are a few words present on each page and each word when spoken links to another page which has a new set of words. This way a greater number of words can be accommodated without having to compare all of them at the same time. It also suits mobile devices because due to their limited screen size, only a few words can be displayed at once.

In this experiment a model of each word was created in the system by using 4 samples of the same word. 12 MFCCs were extracted for each word. This number was reached after observing that consistent results were obtained using 10-12 MFCC coefficients. During testing, a window of 3 seconds was provided to the user to utter a one-word command which the device recognized. Accuracy levels of 85% were observed and a processing time of under 3 seconds was achieved on average. Higher accuracy was obtained for speech recognition and with lesser processing time. Higher accuracy can be attributed to the fact that word models can be formed more accurately than speaker models. The reason being that there is more scope for variation from one utterance to another when the utterances are as long as a sentence. Factors like the gap between words, the speed of utterance may vary even for the same speaker and cause variations which lead to inaccuracy during detection. On the other hand, for a word model, there is no gap between start to end, the modeling has to be done for a smaller utterance and the speed with which a word is spoken remains fairly constant. This leads to higher accuracy. The faster processing time is due to the fact that after trimming from the start and end point detection module the word utterances are very small. Also fewer MFCC coefficients are extracted. These factors combine to reduce the processing time for limited vocabulary speech recognition.

5.4 Results and Interpretation

Apart from accuracy, another metric that is crucial to performance is processing time. Even after all the optimizations, processing is not real time. The reason for the prototype applications being slower than the ones that come pre-installed on the phone are

1. *J2ME code as against native code.* J2ME is good for developing applications from the portability point of view. It is also very popular and J2ME applications can easily be run on a lot of handheld devices. But there is a drawback to writing applications in J2ME. These applications run comparatively much slower than the applications written in code native to the device. Sometimes the difference can be as much as an order of magnitude [4]. These differences have been studied in detail in [4].

2. On some phones, third party applications do not have all of the memory available to them. For example, on Nokia N95, there is 64MB RAM out of which 3rd party applications can use up to 26 MB of RAM. These experiments were

performed on different phones with about the same amount of RAM. Two characteristics, RAM and Processing power were identified as possible bottlenecks. Experiments using on open source application, Tasky, were performed to measure the size of the heap (free memory). Different speaker recognition applications were being run in the background and it was found that a decreasing heap size was not a performance bottleneck unless the size of heap decreased to 3-4 MB. Hence, it was concluded that the processing power of a device is responsible for speed rather than the memory.

3. Native applications on the phone have fixed point implementations. Due to a lack of a floating point processor there's no direct support for floating point applications which decrease the speed further.

As for the accuracy, the prototyped applications turned out to be more accurate than some applications that come pre-installed on the phones. This accuracy can be increased further by building a feedback loop. Almost all the pre-installed applications have in built feedback loop. A typical command recognition application for instance displays a list of results after a command has been spoken into the phone. Once the user chooses from that list, the system assumes that the chosen word was spoken. This helps the system to re-calibrate the word models built in. Continuous feedback can greatly improve the recognition accuracy of the prototype applications described here.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Mobile phones have become extremely popular and pervasive. They are not merely portable telephones, in fact they have become portals to access information from anywhere. Many businesses have realized this fact and are keen to take advantage of it. Consequently, mobile versions of many business applications have been released, ready to run on the phone. These applications perform complex functions and require new and better ways of interaction with the device. Several attempts are being made to facilitate this interaction such as touch screen phones, full keypad phones etc.

In this thesis we examine the use of speech as a possible way of communicating with a mobile device. We also examine other ways in which speech could be useful such as providing contextual information to business applications. Another aspect that we look at in this thesis is embedded processing. Mobile devices, both simple phones and PDAs have seen major improvements in the hardware that they carry on board. However, they still fall under the category of constrained devices when it comes to running applications meant for PCs or even laptops.

Speech processing applications can be computationally very expensive. A full fledged speech recognition application may require hardware that's not available to majority of mobile devices. In this thesis, we try to develop a library for speech processing on mobile phones. Instead of being a speech processing application, this library helps business applications to perform the kind of processing that they may need.

The speech processing framework that is developed in this thesis consists of the following steps:

1. It captures audio signal from the environment.
2. The speech framework then performs pre-processing on the captured audio signal to determine the category to which the audio belongs. Some business applications may require the results of only pre-processing such as determining whether the audio was voice, silence or noise.
3. For those applications that need complicated speech processing, the framework then performs processing to perform speech recognition or speaker recognition as required by the business application.

The results of our experiments show that, the pre-processing performed at the beginning can be performed instantaneously. This stage is not only important for speech processing but can also be useful in itself. Applications that require information like the loudness of noise in the background or the presence of speech in the noise can make use of the pre-processing module.

Experiments performed for accuracy show that the accuracy of the prototype applications is acceptable. These prototype applications demonstrate how other business applications can make use of the speech processing module. One of the benefits of this approach is that each application can choose to implement their own speech processing application thus providing each with a speech interface.

Mobile phones and PDAs are considered constrained devices, because of the lack of computation power on board when compared to desktop PCs. One of the aims of this thesis was to explore whether computationally expensive algorithms used during speech processing could be performed on devices like mobile phones. Our experiments show that with careful adaptation and improvisation, speech processing can indeed be performed on constrained devices. Furthermore, reasonable performance in terms of accuracy and processing time can also be achieved.

In conclusion, the research described in this thesis shows that constrained devices such as mobile phones now have sufficient processing power on board to handle embedded speech processing applications. It also shows that microphone present on the mobile phones can be used to provide limited contextual information to business applications.

6.2 Future Work

Although the speech processing framework described in this thesis performs reasonably well for the type of prototype applications that were intended and the environment it was used in, there is still significant scope for improvement. The first thing that comes to mind is the performance in noisy environment. One approach to tackle this would be to model noise as an entity in the prototype applications and then audio recordings matched to the noise model could be identified as such. The drawback with this approach is that noise comes in various flavors, hence numerous noise models could be required depending on the environments the user operates in. Another way of tackling this problem of noise is to weed it out at the beginning during pre-processing. This could be done by amplifying those frequencies where speech exists. But even this might not cancel out all kinds of noise.

A major performance improvement can be brought about by building a feedback loop in those applications that make use of the speech processing framework described here. Accuracy reported in Chapter V was brought about by minimal training. The reason behind minimal training was to reduce the effort required on the part of the user. But, as shown in some of the experiments, performance improvements can be brought about by more training. Constant training by implementing a feedback loop would be a good way of doing this.

Microphone is just one of many sensors available to a mobile device. One other sensor that can be used in much the same way is the camera. A camera is now standard on many mobile phones. It can be used to gather more information about the environment of the user. Pictures taken from the camera at regular intervals can provide information about the surroundings. Such contextual information, when combined with the information from microphone or even separately, can be useful for some applications. Taking this principle forward, sensors such as

1. Microphone
2. Camera
3. Built-in clock
4. Signal Strength

Figure 6.1 can be used to gather more information about a user's surrounding. For example, the built-in clock can be used to check if it is day or night time, the camera can be used to take pictures to infer whether the background is dark, the microphone can be used to find out if there is noise in the background or if it is silent. Using these sensors, it may be possible to infer whether the user is

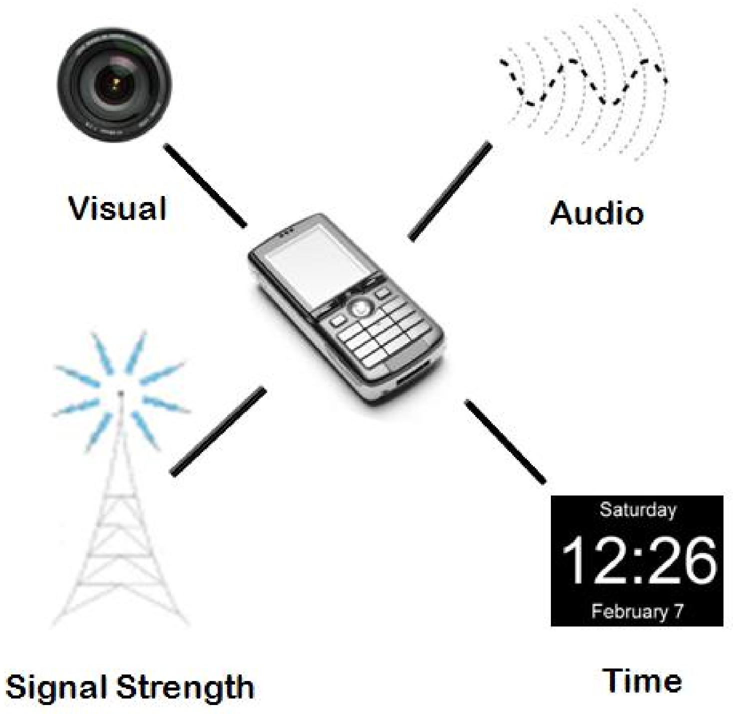

Figure 6.1: Sensors on a mobile device

sleeping or not. If the user is sleeping then the ringer's volume, for example, can be lowered or muted. Let's call such an application a 'Sleep Detector'. Now if the sleep detector had a signal profile for different places where the user spent a lot of time, then that could further be used to ascertain that the user was at home. Taking this a step forward, if information could be collected from applications like calendar, memos, alarms etc then a lot of applications could benefit. For example, the phone could know when the user was in a meeting and consequently the volume of the ringer could be adjusted automatically.

The architecture of such an information gathering system could look like Figure 6.2. The controller module would be responsible for gathering data from all the sensors and then processing it to generate meaningful information. Different applications could then call the functions inside the controller module to get information gathered in such a manner from all the sensors or sensors of their choice. The ovals

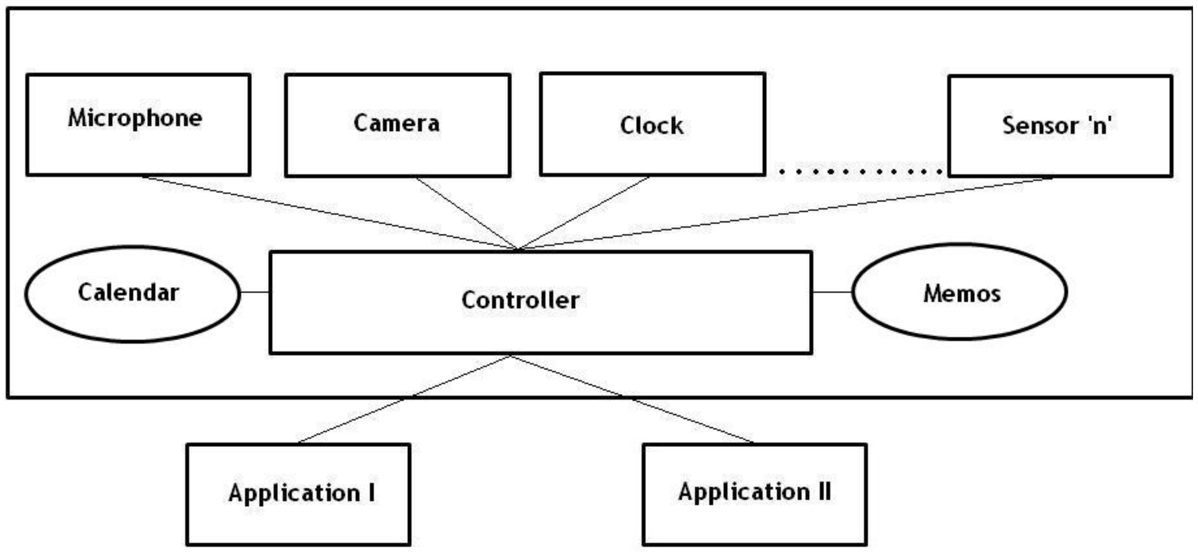

Figure 6.2: Framework Architecture for multiple sensors

in Figure 6.2 have been used to differentiate between applications and sensors, both of which can be used to gather data.

References

[1] Arthur and Vassilvitskii. k-means++: The advantages of careful seeding. In *Proc. of SODA: ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, 2007. 57

[2] B. S. Atal. Automatic speaker recognition based on pitch contours. *J. Acoust. Soc. Amer.*, 52(6 (Part 2)):1687–1697, 1977.

[3] Alessandro Bugatti, Alessandra Flammini, and Pierangelo Migliorati. Audio classification in speech and music: A comparison between a statistical and a neural approach. *Eurosip Journal on Applied Signal Processing*, 3:372–378, 2002. 15

[4] Carlos Garcia-Rubio Celeste Campo and Alberto Cortes. Performance evaluation of j2me and symbian applications in smart camera phones. In *Proc. of 3rd Symposium of Ubiquitous Computing and Ambient Intelligence*, pages 1–2, 2007. 75

[5] Jordan Cohen. Is embedded speech recognition disruptive technology? In *CTO: Voice Signal*, 2004. 6

[6] Jordan Cohen. Embedded speech recognition applications in mobile phones: Status, trends and challenges. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5352–5355, 2008. 5, 6, 7

[7] George R. Doddington. Speaker recognition identifying people by their voices. *Proceedings of IEEE*, 73(11):1651–1664, 1985.

[8] C.O. Dumitru, I. Gavat, and R. Vieru. Speaker verification using hmm for romanian language. In *In proc. Multimedia Signal Processing and Communications, 48th International Symposium ELMAR-2006*, pages 131–134, 2006. 19

[9] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, (3):32–57, 1973. 25

[10] Masakiyo Fujimoto and Yasuo Ariki. Speech recognition in a noisy environment using a speech signal estimation method based on the kalman filter. *Systems and Computers in Japan*, 35(3):46–57, 2004.

[11] A. Ganapathiraju, J. Hamaker, and J. Picone. Hybrid svm/hmm architectures for speech recognition. In *Proc. Speech Transcription Workshop*, pages 504–507, 2000.

[12] Todor Ganchev, Nikos Fakotakis, and George Kokkinakis. Comparative evaluation of various mfcc implementations on the speaker verification task. In *In Proc. 10th International Conference on Speech and Computer*, pages 191–194, 2005. 24

[13] David Gerhard. Audio signal classification: An overview. *Canadian Artificial Intelligence*, 45:4–6, 2000. 14, 15

[14] Shubhangi Giripunje and Anshish Panat. Speech recognition for emotions with neural network: A design approach. In Mircea Gh. Negoita, Robert J. Howlett, and Lakhmi C. Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems, 8th International Conference, Part II*, volume 3214 of *Lecture Notes in Computer Science*, pages 640–645. Springer, 2004. 16

[15] Jonna Häkkilä and Jani Mäntyjärvi. Collaboration in context-aware mobile phone applications. In *In Proc. 38th Annual Hawaii International Conference on System Sciences*, page 33.1, 2005. 26

[16] S. Haltsonen. An endpoint relaxation method for dynamic time warping algorithms. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 9.8.1–9.8.4, 1984. 30

[17] Caroline Heycock. Lecture 6: Phonetics ii. `http://www.ling.udel.edu/colin/courses/ling101_f99/lecture6.html`. ix, 34

[18] Hui Jiang, Xinwei Li, and Chaojun Liu. Large margin hidden markov models for speech recognition. *IEEE Transactions on Audio, Speech & Language Processing*, 14(5):1584–1595, 2006. 18

[19] G. S. Jovanovic. A new algorithm for speech fundamental frequency estimation. *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP-34(3):626, 1986. 15

[20] Bob K. Hamming window and frequency response, 15-Dec-2005. 47

[21] B. Kedem. Spectral analysis and discrimination by zero-crossings. *In Proc. of the IEEE*, 74:1477–1493, 1986. 15

[22] M. Kotti and C. Kotropoulos. Gender classification in two emotional speech databases. In *In Proc. International Conference on Pattern Recognition*, pages 1–4, 2008. 16

[23] C. C. Leung and Y. S. Moon. Effect of window size and shift period in mel-warped cepstral feature extraction on GMM-based speaker verification. In *Proc. Audio- and Video-Based Biometric Person Authentication*, pages 438–445, 2003. 28

[24] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. LeCam and J. Neyman, editors, *In Proc. 5th Berkeley Symp. on Mathematics Statistics and Probability*, pages 281–297, 1967. 25, 57

[25] Matija Marolt. On finding melodic lines in audio recordings. In *In Proc. 7th International Conference on digital Audio Effects*, pages 217–221, 2004. 15

[26] Y.S. Moon, C.C. Leung, and K.H Pun. Fixed-point gmm-based speaker verification over mobile embedded system. In *Proc. ACM SIGMM 2003 Multimedia Biometrics Methods and Applications Workshop*, pages 53–57, 2003. 28

[27] D. O'Shaughnessy. Efficient automatic speech recognition. In *Proc. Internet and Multimedia Systems and Applications*, page 427, 2004. 55

[28] Joeseph Picone. Signal modeling techniques in speech recognition. In *Proc. CTO: Voice Signal*, pages 1215–1247, 1993.

[29] Richard C. Price, Jonathan P. Willmore, William J. J. Roberts, and Kathleen Zyga. Genetically optimised feedforward neural networks for speaker identification. In *Fourth International Conference on Knowledge-Based Intelligent Information Engineering Systems & Allied Technologies*, pages 479–482, 2000.

[30] Steve Renals and Vincent Wan. Speaker verification using sequence discriminant support vector machines. Technical Report EDIINFRR0659, The University of Edinburgh, 2005.

[31] Douglas A Reynolds and Richard C Rose. Robust text- independent speech recognition using gaussian mixture models. *IEEE Trans. Speech Audio Process.*, 3:72–83, 1995. 21

[32] Aaron E. Rosenberg. Automatic speaker recognition: A review. *Proceedings of IEEE*, 64(4):475–487, 1975.

[33] Xavier Serra. Cepstral analysis. `http://www.dtic.upf.edu/~xserra/cursos/TDP/8-Cepstral-Analysis.pdf`. 54

[34] Daniel P. Siewiorek, Asim Smailagic, Junichi Furukawa, Andreas Krause, Neema Moraveji, Kathryn Reiger, Jeremy Shaffer, and Fei Lung Wong. Sensay: A context-aware mobile phone. In *Proc. International semantic web conference*, pages 248–249, 2003. 26

[35] Zheng-Hua Tan and Børge Lindberg. *Automatic speech recognition on mobile devices and over communication networks*. Springer-Verlag, London, 2008.

[36] Nathan S. Netanyahu Christine D. Paitko Ruth Silverman Tapas Kanungo, David M. Mount and Angela Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(7):881, July 2002. 25

[37] Joe Tebelskis. Speech recognition using neural networks. Technical report, 1995. 20

[38] D. Wasson and R. Donaldson. Speech amplitude and zero crossings for automated identification of human speakers. In *Proc. IEEE International conference on Acoustics, Speech and Signal Processing*, pages 390– 392, 1975. 14, 15

[39] L. Welling and X. Aubert. A study on speaker normalization using vocal tract normalization and speaker adaptive training, October 05 1998. 30

[40] Febe De Wet. Comparing acoustic features for robust asr in fixed and cellular network applications, February 16 2000. 49

[41] Febe De Wet, Bert Cranen, Johan De Veth, and Loe Boves. A comparison of LPC and FFT-based acoustic features for noise robust ASR, July 03 2001. 49

[42] Wikipedia. Cepstrum. `http://en.wikipedia.org/wiki/Cepstrum`.

[43] Ram H. (Ram Han) Woo. Exploration of small enrollment speaker verification on handheld devices. Master's thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 2005. 27

[44] Bian Wu, Xiaolin Ren, Chongqing Liu, and Yaxin Zhang. A robust, real-time voice activity detection algorithm for embedded mobile devices. 1997. 14

[45] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Trans. on Neural Networks*, 16(3):645–678, May 2005. 25

[46] Dmitry Zaykovskiy and Alexander Schmitt. Java (J2ME) front-end for distributed speech recognition. In *Proc. nternational Conference on Advanced Information Networking and Applications Workshops*, pages 353–357, 2007.