Speech Diarization on a Mobile Platform

by

Daniel Di Matteo

A thesis submitted in conformity with the requirements
for the degree of Bachelor of Applied Science
Division of Engineering Science
University of Toronto

Abstract

Speech Diarization on a Mobile Platform

Daniel Di Matteo

Bachelor of Applied Science

Division of Engineering Science

University of Toronto

2011

This document outlines the design and development of a speech-processing application for the Android mobile platform. The function of the application is to process ambient audio from a conversation between N speakers and to calculate how much time was spent speaking by each member of the conversation. Formally, this is framed as a problem within the research field of speech diarization. An existing speech diarization toolkit is ported to a smartphone device for the first time. The functionality of the application is confirmed and then explored under various speaker environments. Finally, possible avenues for improving accuracy and computation time are presented.

Acknowledgements

Many thanks to Professor Jonathan Rose for not only acting as my supervisor, but for offering me the freedom to work on something unfamiliar and the motivation to see it through to fruition. Thanks to Jason Foster for the time spent on discussing software engineering, prototyping and testing. Thanks to Mark Harfouche for help with everything research related, and especially for forcing me to learn LaTeX. Finally, I'd like to thank my family and friends for their interest in my work.

Contents

List of Abbreviations

BIC	Bayesian Information Criterion
DFT	Discrete Fourier Transform
EM	Expectation Maximization
ESTER	Evaluation of Radio Broadcast Rich Transcription System (French acronym)
FT	Fourier Transform
GLR	Generalized Likelihood Ratio
IDFT	Inverse Discrete Fourier Transform
LTI	Linear Time-Invariant
MFCC	Mel-Frequency Cepstral Coefficient
ML	Maximum Likelihood
NDK	Native Development Kit
PC	Personal Computer
PCM	Pulse Code Modulation
SIMD	Single Instruction, Multiple Data

List of Figures

Chapter 1

Introduction

Smartphones are a relatively new addition to the computing environment, and what they lack in processing power (when compared to desktop workstations), they make up for in mobility, price, and ease-of-use. Furthermore, the divide between the computational ability between PCs and smartphones is rapidly decreasing. Tasks such as 3D graphics, imaging and sound processing, which were once the domain of PCs, are now becoming well established in the mobile computing scene.

This pattern of increasing smartphone power and widening of smartphone applications is a circular process. As phone makers enhance the capabilities of their products, existing tasks that were once too intensive become tractable on the mobile platform. These novel applications grow in complexity, causing a demand for higher-performance devices which is supplyed by industry.

This thesis explores the design and implementation of one such application. The proposed application is intended to listen to a conversation and calculate what percentage of time was spent speaker by each member of the conversation. Formally, such a task can be viewed as problem within the domain of speech diarization.

1.1 Speech Diarization

Speech diarization is the process of segmenting a speech recording (i.e., an audio signal) according to speaker identity, effectively organizing all the speech clusters according to speaker. Colloquially, this task can be summarized as answering the question ”who spoke when?” [2].

This type of audio processing is motivated by the increasing need to index and categorize audio transcripts from sources such as broadcasts, voice mails, and meeting transcripts [2]. Diarization of these sources can provide discrete segments corresponding to individual speakers, which can then be processed to determine true speaker identity (via a speaker identification system) to aid in the cataloguing of these ”spoken documents” [2].

1.2 State of the Art

Diarization systems exist and can achieve relatively high levels of accuracy. Open Source toolkits include AudioSeg, Mistral_Seg, and LIUM_SpkDiarization [3]. In general, these systems diarize audio using a mix of signal processing to generate unique speaker signatures and statistical analysis to model speakers and detect change points in conversations [4].

These toolkits have been tested in national research campaigns for diarization and segmenting of broadcast audio (e.g., ESTER campaign in France) [5]. However, none of the systems are intended to be used to segment/diarize audio in a real time manner (intended for pre-recorded broadcast news or telephone recordings, as opposed to live meeting audio). Furthermore, none of these systems have been implemented on an embedded or mobile platform.

The ability to perform such processing on a mobile device offers many benefits over the traditional approach, including: reduced cost (i.e., smart-phone / tablet vs. PC/ Workstation) and convenience (processing can be performed in any setting where audio can be recorded). Finally, with increasingly powerful mobile devices becoming all the more pervasive within the domain of consumer electronics, such a project is now feasible from a financial and computational perspective.

1.3 Thesis Structure

The next chapter provides the theoretical background for the signal processing involved in speech diarization. Chapter 3 outlines the design steps taken and how the application was developed moving forward. Chapter 4 explores the performance of the application in two different environments. Finally Chapter 5 conludes with a discussion of the applications and contributions of this work, and how it may be optimized.

Chapter 2

Background

2.1 Overview

The process of speech diarization of an audio signal is ideally performed in three steps:

1. Feature Extraction from Audio Data
2. Speaker Segmentation
3. Speaker Clustering

Feature extraction is performed upon the audio to highlight the unique components of each speakers voice, which is ultimately exploited by the algorithm. Using the data produced by the feature extraction, speaker segmentation follows, which identifies points in the audio where speaker identity changes [4]. After effective speaker segmentation, what remains are a number of smaller discrete audio segments (discrete in the sense that each segment is the utterance of a single speaker only). Finally, speaker clustering is performed to sort the segments based upon unique speaker identity, resulting in a group of speech from each speaker.

2.2 Feature Extraction from Audio Data

2.2.1 The Cepstrum

The raw audio data is not used directly to train speaker models. Instead, the audio data is used to extract a series of Mel-Frequency Cepstral Coefficients (MFCCs). The Cepstrum is defined as the inverse Fourier Transform of the Power Spectrum of a signal (the Power Spectrum being the logarithm of the Fourier transform of a signal) [6]:

$$c(k) = IDFT\{log|DFT\{x(n)\}|\} \tag{2.1}$$

Each MFCC feature is a vector consisting of the first 13 coefficients of the cepstrum, with an additional Mel-Frequency filtering applied to the power spectrum of the speech signal.

Mel Scale Frequency Warping

The human ear is more sensitive to changes in pitch at lower frequencies. The Mel scale was designed to provide a nonlinear transformation to frequencies such that a linear change in frequency would be result in a linear perception in pitch change by the listener [7]. Application of Mel scale frequency scaling allows the MFCCs to more closely model perception, resulting in more robust identification of unique speaker spectral qualities [8]. See Figure A.1 in Appendix A for a representation of this nonlinear transformation.

2.2.2 Cepstral Analysis

The use of MFCCs is motivated by the desire to obtain a unique identifier for each speaker. Such an identifier could be the speaker's vocal tract, which is responsible for creating each speaker's unique voice. Viewing the vocal tract as an LTI system, a speaker's voice could be considered a signal which is the result of the convolution between an input signal (bursts of air from the speaker's lungs) and a system (the speaker's vocal tract). The Cepstrum effectively performs a deconvolution upon the output signal (i.e., the speaker's voice), allowing us to parametrize each person's unique voice by modelling the vocal system as a vector of cepstral coefficients.

Consider the following running example [6], given an input voice signal S(ω) :

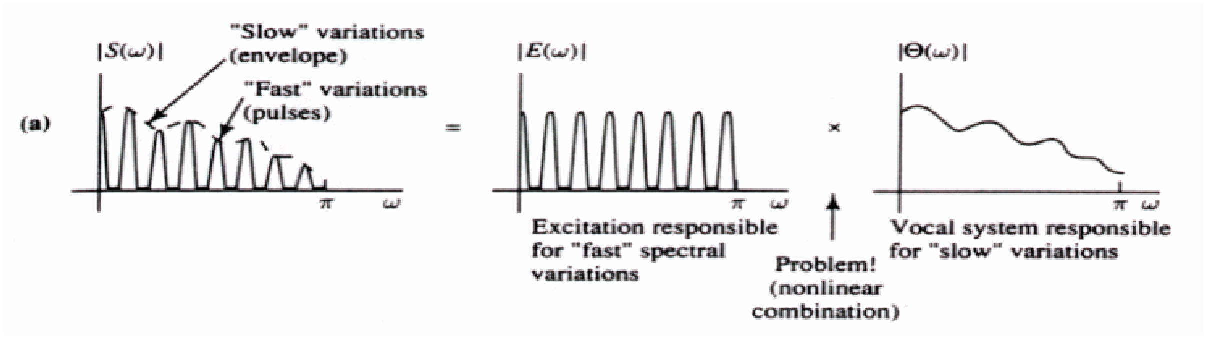

(a)
$|S(\omega)|$
"Slow" variations (envelope)
"Fast" variations (pulses)
=
$|E(\omega)|$
×
$|\Theta(\omega)|$
π ω
Excitation responsible for "fast" spectral variations
Problem! (nonlinear combination)
Vocal system responsible for "slow" variations

Figure 2.1: Frequency domain representation of arbitrary audio signal

Figure 2.1 demonstrates how an arbitrary audio signal can be viewed as the convolution (equivalently, multiplication in the frequency domain) of an excitation signal, E(ω), and a vocal system, $\theta(\omega)$. At this stage it is still not possible to easily separate the system's characteristics from those of the input signal.

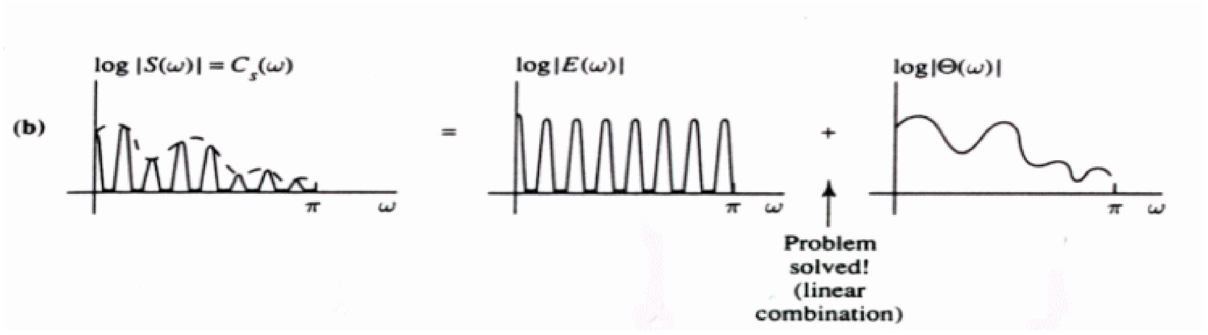

(b)
$\log|S(\omega)| = C_s(\omega)$
=
$\log|E(\omega)|$
+
$\log|\Theta(\omega)|$
π ω
Problem solved! (linear combination)

Figure 2.2: Logarithm of the Fourier transform of the speech signal

Figure 2.2 demonstrates how the logarithm of the Fourier transform of the audio signal effectively solves the problem of the nonlinear combination of the input signal and voice system by exploiting the logarithm's property of transforming products into sums.

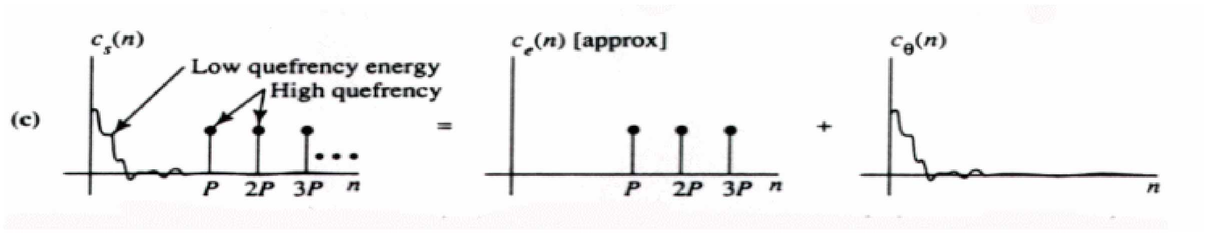


Figure 2.3: Cepstrum of the audio signal

Figure 2.3 depicts the cepstrum of the speech signal, arrived at by applying the inverse Fourier transform to the signal depicted in Figure 2. By isolating the data located on the lowest values of the abscissa (i.e., quefrency, the equivalent of frequency in the cepstral domain), the characteristics of the vocal system are preserved, while discarding the influence of the excitation signal. This is why, in the discrete time system which is used in practice, the first 13 values are preserved and subsequently adopted as a method of determining speaker identity.

2.3 Segmentation

Segmentation is the processing of detection speaker change points in the conversation. A segment is defined as a continuous portion of speech originating from a single speaker. Consider figure 2.4 as a visual aid for the following description.

Segmentation is performed by first observing MFCC vectors within a specific window (e.g., 500 vectors). Using the data contained in this window, three models are computed. First, a model N' is generated using all of the MFCC vectors as training data. Second, two other models N_1 and N_2 are computed, using the first and second half of the total sum of vectors, repectively. A distance metric is then computed, which is used to assess if the data is better fit by the single model N' or the two smaller models N_1 and N_2. After the distance is computed, this value is saved and the window shifted (by a single vector), and the process is repeated until all the vectors have been assessed in this manner.

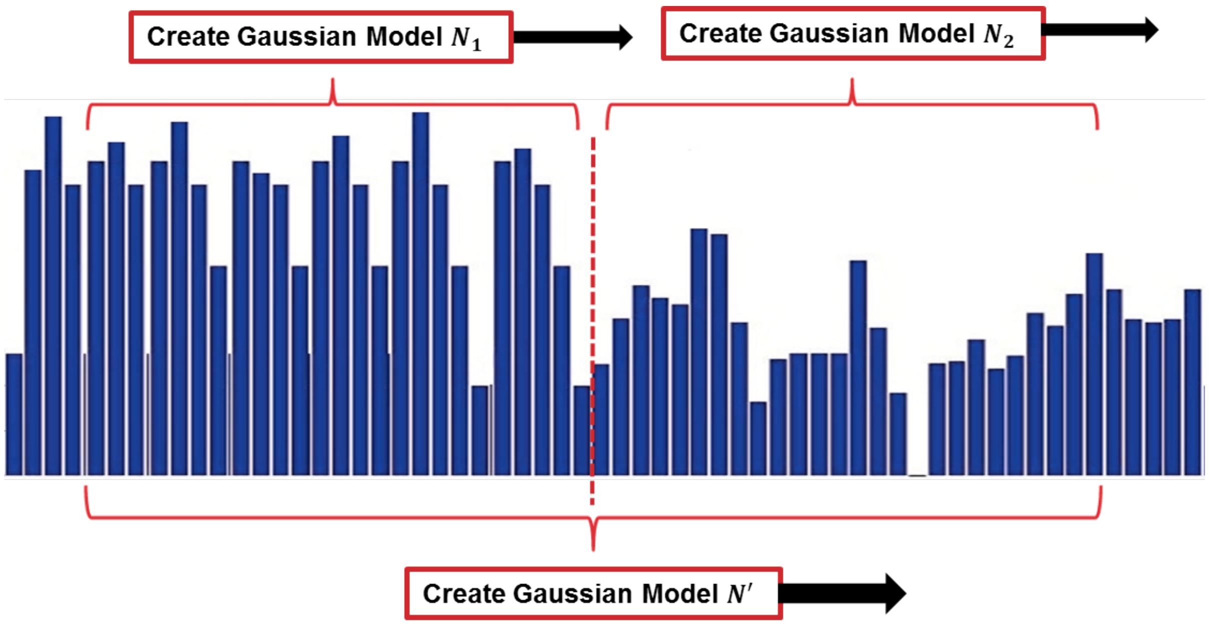

Create Gaussian Model N_1
Create Gaussian Model N_2
Create Gaussian Model N'

Figure 2.4: Model construction in Segmentation step

Defining Change Points

Change points are defined after all the distance values have been computed. The array of distance values is evaluated and local maximas of the array are compared to some threshold. Values greater than the threshold are defined as changepoints, creating segment boundaries. Maxima are only considered within a fixed licality, such that in practice, changepoints must be separated by some nominal value (e.g., 250 MFCC vectors, or 2.5s). This ensures that all segments are at least a given length in size/time. Furthermore, the threshold value to be exceeded is another system parameter that can be varied such that the decision for defining change can be made more or less stringent.

2.3.1 Modelling and Training

Each speaker is modelled as a random Gaussian process from which their corresponding sequence of MFCC vectors are sampled from. In full generality this process is multivariate, and as such can be considered a D-dimensional Gaussian of the form:

$$b(\vec{x}) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu})'\Sigma^{-1}(\vec{x}-\vec{\mu})\right) \tag{2.2}$$

with mean vector $\vec{\mu}$ and covariance matrix Σ. In practice, 13 dimensinal Gaussians can be used, with each dimension of the Gaussian corresponding to a dimension of the MFCC vectors.

Model Training

In practice the aforementioned parameters of the Gaussians are not known, but must be computed given some MFCC vectors. Traditionally a Maximum Likelihood (ML) method would be used to compute the necessary parameters, which would provide the ideal model parameters to maximize the likelihood of the model [9]. Unfortunately the Gaussian likelihood expression cannot be maximized directly, and instead an iterative approach is used to estimate the parameter values leading to maximum likelihood.

This more practical technique, referred to as Expectation Maximization (EM), begins by guessing parameters, and then estimates a new set of parameters that result in a Gaussian of greater likelihood. This new set of parameters is then used as the guess in the next iteration, and the parameters are improved upon until some threshold is reached [9].

2.3.2 Generalized Likelihood Ratio as a Distance Metric

A commonly used distance metric used to assess model fit is the Generalized Likelihood Ratio. For operating upon multivariate Gaussians, the GLR is computed as follows [10]:

$$d_{GLR}(S_i, S_j) = \frac{n_i + n_j}{2} log|\Sigma_{ij}| - \frac{n_i}{2} log|\Sigma_i| - \frac{n_j}{2} log|\Sigma_j| \tag{2.3}$$

where n is the number of MFCC vectors used to train the model, and $|\Sigma|$ is the determinant of the model's covariance matrix. Subscript ij refers to the aggregate model, where subcripts i and j refer to the two smaller models.

What this does, essentially, is to provide the most likely solution to the following binary hypothesis testing problem [11]:

$$\begin{cases} \mathcal{H}_0 : & \text{Data better fit by a single model} \\ \mathcal{H}_1 : & \text{Data better fit by two models} \end{cases}$$

Higher score values suggest hypothesis $\mathcal{H}_1$ as being more likely, and thus segment boundaries are placed at the maxima of d_{GLR}. Recall, however, that these maxima are still required to be higher that a threshold value to be considered as segment boundaries.

2.4 Clustering

Clustering is the process of assigning speaker identity to each segment, essentially sorting and fusing the segments based on their source. A cluster is defined as all the segments originating from a single speaker. This task can be considered the dual of segmentation. One of the key differences, however, is that this comparison must be done between each pair of models (i.e., not only between adjacent models). Consider figure 2.5 as a visual aid for the following description.

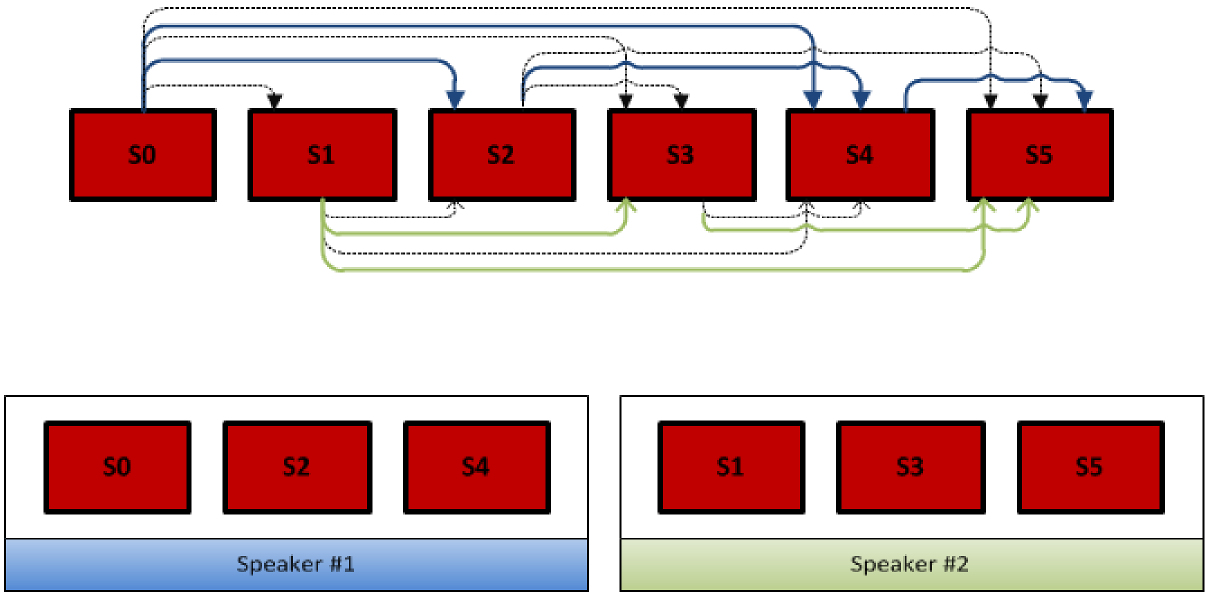


Figure 2.5: Segment recombination in Clustering step

At the onset of the clustering stage, an initial set of clusters is created, where each cluster is composed of a single segment only (i.e., the segments resulting from the segmentation phase). Clustering is then performed by computing the distance between every pair of clusters. At each iteration the pair of clusters with the smallest distance are selected for merging together into a single cluster [3]. This process continues until the distances between all of the remaining clusters are greater than some threshold value.

Figure 2.5 represents a sample situation in which there are two speakers, who both take turns speaking. Speaker 1 speaks, followed by Speaker 2, and this repeats for two more iterations. Ideally, segments S0, S2 and S4 should be clustered together (as they were spoken by Speaker 1), and likewise segments S1, S3 and S5 should be clustered together.

2.4.1 Modelling and Training

As in the segmentation phase, the clusters are modeled as 13-dimensional Gaussians. These Gaussians are trained using the body of MFCCs that correspond to all of the clusters' segments, using the same EM algorithm.

2.4.2 Bayesian Information Criterion as a Distance Metric

A commonly used distance metric used as the decision factor in the cluster stage is the Bayesian Information Criterion (BIC) [12]. The BIC can be derived from the GLR as follows (see equation 2.3):

$$d_{BIC}(C_i, C_j) = d_{GLR}(S_i, S_j) - \lambda P \tag{2.4}$$

$$d_{BIC}(C_i, C_j) = \frac{n_i + n_j}{2} log|\Sigma_{ij}| - \frac{n_i}{2} log|\Sigma_i| - \frac{n_j}{2} log|\Sigma_j| - \lambda P \tag{2.5}$$

The BIC penalty factor P, is defined as

$$P = \frac{1}{2}\left(d + \frac{d(d+1)}{2}\right) log(n_i + n_j) \tag{2.6}$$

where λ is the penalty weight (a parameter that varies across systems and implementations), and d is the dimension of the model space (e.g., 13). In BIC-based clustering, the two candidate clusters i and j are merged if $d_{BIC}(C_i, C_j) < 0$ [12].

N.B.:

The introduction of this penalty term has the effect of lowering the overall score. Since the score must be negative to trigger a clustering action, modifying the penalty weight, λ, can make a clustering decision process more or less likely.

2.4.3 Linear vs. Hierarchical Clustering

In practive, clustering is not often done in a single step, as explained earlier. Instead, the distinction is made between two type of clustering procedures: linear and hierarchical. Hierarchical clustering is the traditional procedure, which was outlined in this section.

Linear clustering can be though of as an intermediate step between segmentation and hierarchical clustering. What it does is essentially re-segmentation. That is, it fuses together adjacent segments that originate from the same speaker. Theoretically, this should not be necessary, since a changepoint should not be defined between two adjacent segments originating from the same speaker. In practice, however, the segmentation threshold is set very low to avoid false negatives (i.e., true changepoints that the segmentation algorithm missed). Since lowering this threshold will increase the rate of false positive detection (i.e., detecting a changepoint were there is not one in reality), the linear clustering phase is introduced to correct this problem.

Note, however, that hierarchical clustering could be used to perform the same task, since it will also fuse together adjacent segments. The problem is that the complexity of the hierarchical algorithm grows quadratically, while the linear algorithm grows linearly. Therefore, if linear clustering is done first, this reduces the total number of segments presented to the hierarchical stage, thereby reducing overall computation time.

Chapter 3

Design and Implementation

3.1 Investigation of Available Resources

Faced with the existence of several well-documented and open source diarization toolkits, it was necessary to decide whether the design should incorporate one of these existing packages or if an entirely new system should be coded from the ground up. It was decided that an existing package would be ported to the Android platform to serve as the computational back end. In this manner it was possible to leverage the documentation and community support of the open source community. Furthermore, the existing packages have all achieved acceptable results for the target application, making a redesign unnecessary.

3.2 Choosing a Target Toolkit

Having decided to port an existing toolkit, the choice between toolkits remained to be made. The two main toolkits considered were AudioSeg (a C-based toolkit) and LIUM_SpkDiarization (Java-based). LIUM_SpkDiarization was chosen as the nominal toolkits for two main reasons. Firstly, being Java-based, this allows the code to be run directly within Android's Dalvik VM. This allows one to avoid using the Java Native

Interface provided by the Android NDK necessary to call C-based toolkits. The Android NDK was avoided since its future support within the Android platform is dubious. Secondly, during a brief testing period, LIUM_SpkDiarization achieved better diarization results upon sample data than the AudioSeg package.

3.3 Prototype Development

Having narrowed the scope of the project, it was then necessary to begin the development work. Prior to developing directly for the Android platform, a prototype was developed on a PC to confirm the viability of the project and the selected toolkit. Appendix B contains a prototype script, in which the LIUM_SpkDiarization package is used to perform a complete diarization upon a sample audio file.

3.3.1 Assessing Prototype Performance

Performing a diarization upon an audio file containing 3 different speakers (in which the 'correct' diarization was determined by actively listening to the recording and noting the change points), the following results were gathered:

LIUM_SpkDiarization	**Actual**
`Speaker 1:  0.47%`	`Speaker 1:  23%`
`Speaker 2:  27.27%`	`Speaker 2:  41%`
`Speaker 3:  37.48%`	`Speaker 3:  36%`
`Speaker 4:  33.14%`	
`Speaker 5:  1.64%`	

Clearly the system erroneously identifies new speakers, but the accuracy was sufficient such that the design was still considered still viable. Consider, for example, applying some minimum threshold of time spent speaking by which to consider a new speaker, these can be eliminated. For example, by imposing that all new speakers speak at least 3% of the total time the extraneous speakers would be eliminated.

3.4 Application Development

Having deemed the project viable using the LIUM_SpkDiarization toolkit as a computational backend, it was then necessary to port the toolkit into an Android environment and code the necessary features to complete the design.

3.4.1 Application Structure

The application was developed through the completion of five main subcomponents:

1. Graphical Frontend
2. Audio Recorder
3. Feature Extractor
4. Diarization Backend
5. Results Processor

Graphical Frontend

The interface of the application can be seen in Figure C.1 in Appendix C.

Audio Recorder

Digital audio is encoded as single-channel 16-bit signed integer PCM. This is achieved through use of the Android AudioRecord class [13].

Feature Extractor

Functionality from the Sphinx4 set of feature extraction classes was adapted to perform the feature extraction as outlined in Section 2.2 [14].

Diarization Backend

The LIUM_SpkDiarization toolkit was employed to perform the following functions:

- GLR-based Segmentation
- BIC-based Linear Clustering
- BIC-based Hierarchical Clustering

Results Processor

The end results of the diarization are displayed to the user in the form of a pie chart, which represents the relative length of time spent speaking by each user (see Figure C.1 in the Appendix). These lengths are calculated by parsing the output data produced by the LIUM_SpkDiarization toolkit. This output contains the length of each segment uttered by each speaker. As the total length of the conversation is known, the relative lengths are easily calculated. See Appendix D for sample output from LIUM_SpkDiarization.

3.4.2 Development Considerations

The software development process (i.e., coding, bug fixes, testing, etc) will not be discussed in this document. The scope of this document is to simply highlight the viability of the smartphone as a computational platform to perform speech diarization.

Chapter 4

Results and Discussion

4.1 Experimental Procedures

4.1.1 Comparison of Results

To assess the accuracy of the diarization results, the computed results were compared to a diarization accepted as 'truth'. This true diarization was the result of listening to the recording of the conversation and noting when speaker changes occured. Since no systems exist which can surpass the accuracy of the human ear, a diarization arrived at in this manner forms the basis for all comparisons.

4.1.2 Data Acquisition

The results of the computed diarization (i.e., the application's analysis of the conversation) were obtained directly from the output of the LIUM_SpkDiarization backend by enabling logging within the application. These results provide a much higher resolution than the pie chart analysis, and they also allow one to view the intermediate representations of the conversation (i.e., what occurs between each step in the diarization process).

4.1.3 Analysis

To assess the accuracy of the results, plots were created which depict where each speaker speaks within the conversation, by colouring a time-dependent waveform by speaker identity. Instead of plotting the audio waveform of the conversation directly, what is plotted is the value of the distance metric from the segmentation step vs. feature index. This is preferred for the following reasons:

- this will confirm if segment boundaries are indeed occuring at the local maxima of the distance score
- this representation is more readable as the score varies less rapidly than the audio waveform
- feature index varies linearly with time (i.e., one feature every 10ms), so the temporal aspects of the conversation are still accurately conveyed

4.1.4 Accuracy Metric

The metric used to assess the accuracy of the diarization is defined as follows:

$$Accuracy = \left(1 - \frac{total\ time\ during\ which\ audio\ is\ correctly\ clustered}{length\ of\ conversation}\right) \times 100\% \quad (4.1)$$

4.2 Results

4.2.1 Conversation Among Two Speakers

Figure 4.1 displays the results of the segmentation process upon a conversation between 2 speakers. The vertical red lines show the segment boundaries. This confirms that segment boundaries occur at local maxima of the GLR distance score. Note that numerous false positives occur.

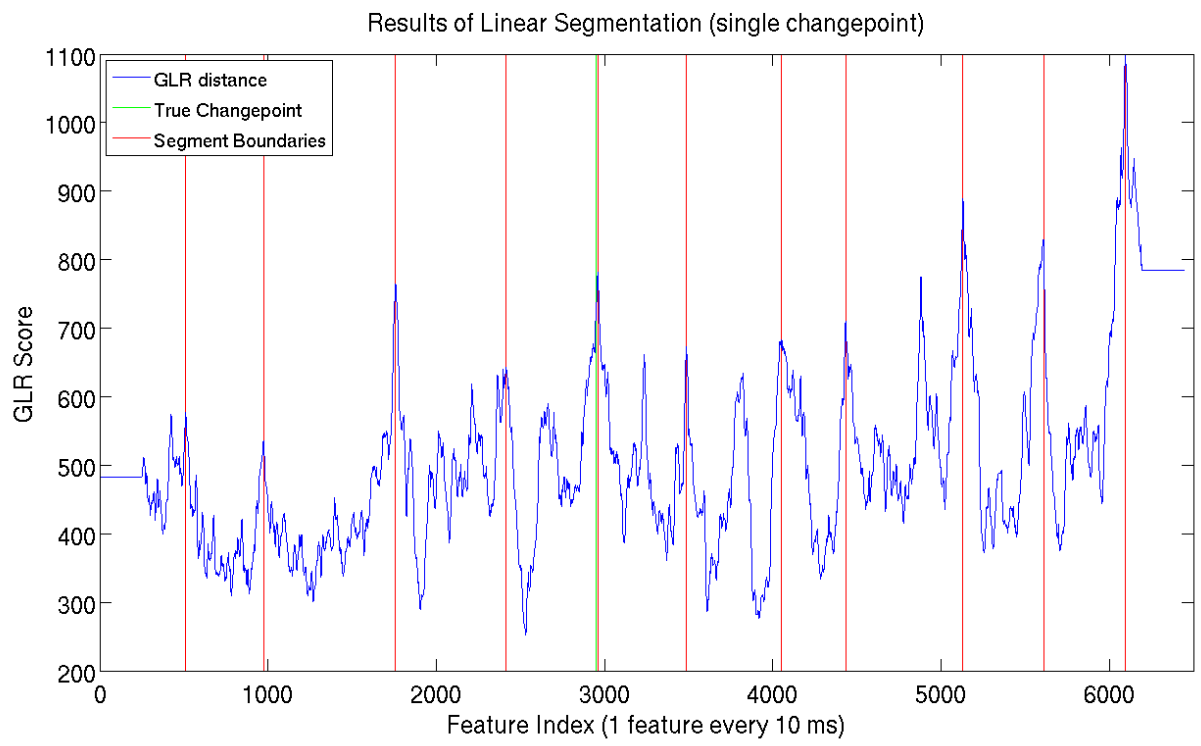


Figure 4.1: Results of linear segmentation upon a converation with a single changepoint.

Figure 4.2 display the results of linear clustering applied to the previous group of segments. Note the elimination of the false positive change points.

Figure 4.3 displays the end results of the diarization. In this situation an acuracy of 93% is achieved. The main source of innacuracy is the fact that the noise at the end of the conversation is considered a third speaker.

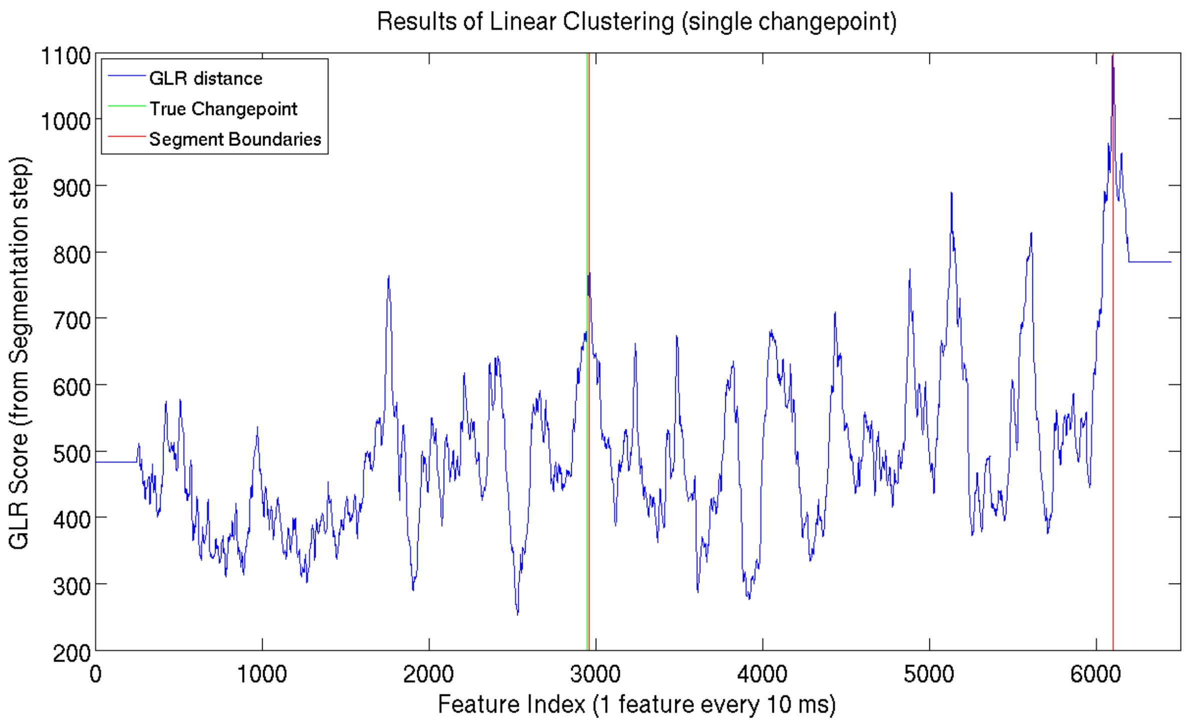


Figure 4.2: Results of linear clustering upon the results shown in figure 4.1.

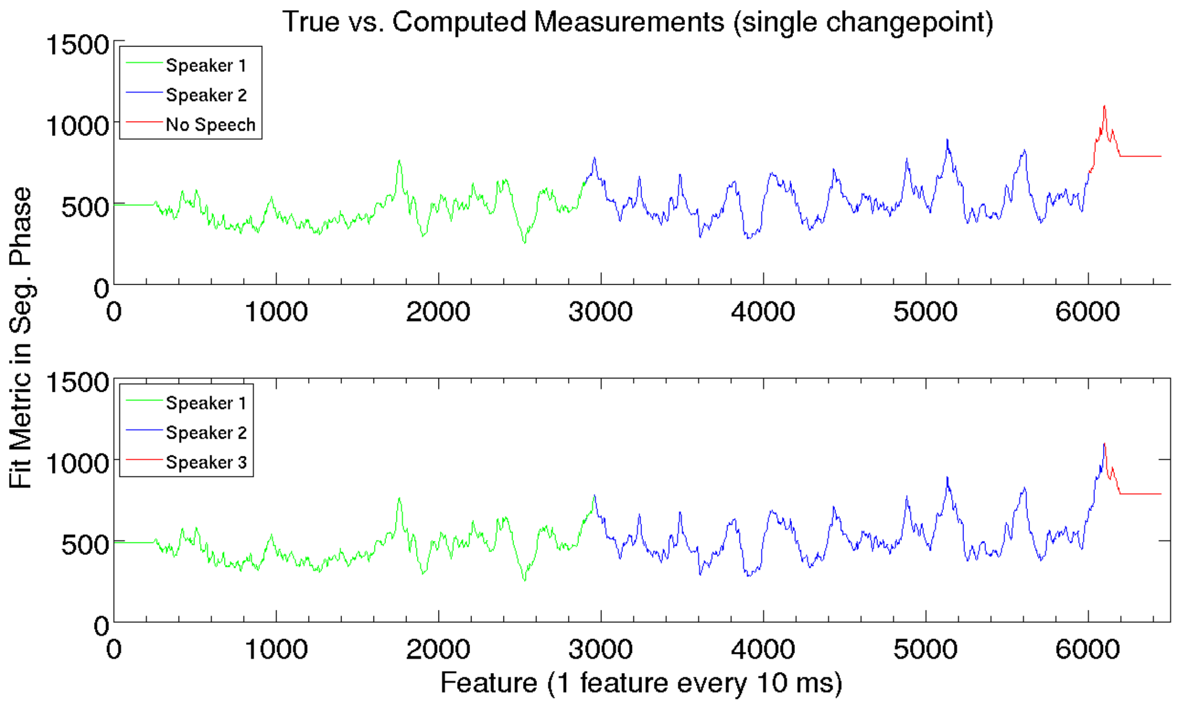


Figure 4.3: Results of complete diarization upon the conversation with a single changepoint.

4.2.2 Conversation Among Three Speakers

Figure 4.4 displays the final results of the full diarization process upon a conversation between 3 speakers. In this situation an acuracy of 77% is achieved. The main source of innacuracy is the fact that some of the clusters from the true Speaker 2 are incorrectly classified as new speakers. This indicated that the clustering stage is failing to identify that the computed Speakers 2,3 and 6 are the same speaker.

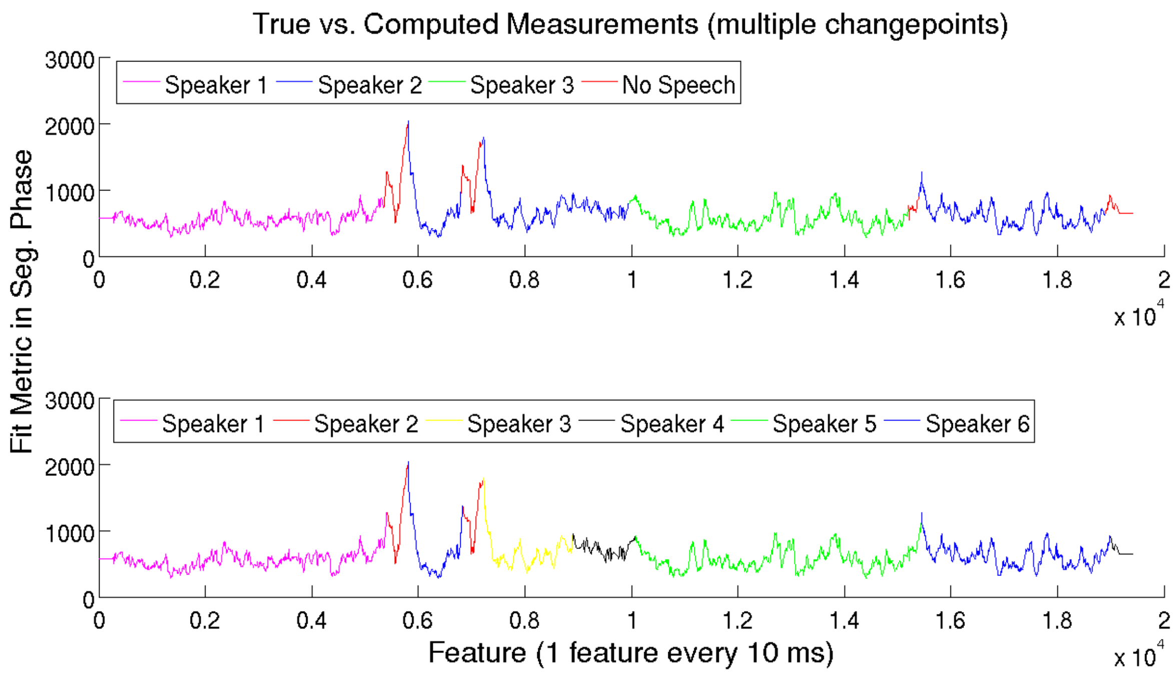


Figure 4.4: Results of complete diarization upon a conversation with multiple changepoints between 3 speakers.

4.2.3 Improving Upon Baseline Performance

System Parameters Varied During Experimentation

In the following section, additional processing trials upon the 3-speaker conversation will be discussed. During the experimentation process the following system parameters were varied:

λ_l :	BIC penalty weight in linear clustering stage
λ_h :	BIC Penalty weight in hierarchical clustering stage

Rationale for Modifying the BIC Penalty Weight

As depicted in Figure 4.4, the system sometimes fails to cluster audio which is truly from the same speaker. Upon reviewing the BIC-based clustering procedure from Section 2.4, it becomes evident that the clustering score can be lowered by increasing the value of λ. Lowering the score will have the effect of making a clustering decision more likely. This theory will now be confirmed experimentally.

Diarization with Modified λ_l

Figure 4.5 displays the final results of full diarization process upon the same 3 speaker conversation. In this experiment the BIC penalty weight for the linear clustering phase is raised. When compared to Figure 4.4, it shows how two of the extraneous clusters are now merged together. Note that only the two adjacent clusters are merged, which is to be expected for *linear* clustering.

In this situation, λ_l was increased from 2 (default) to 3, resulting in an accuracy which is still 77%.

Diarization with Modified λ_l and λ_h

Figure 4.6 displays the final results of full diarization process upon the same 3 speaker conversation. In this experiment the BIC penalty weight for the hierarchical clustering

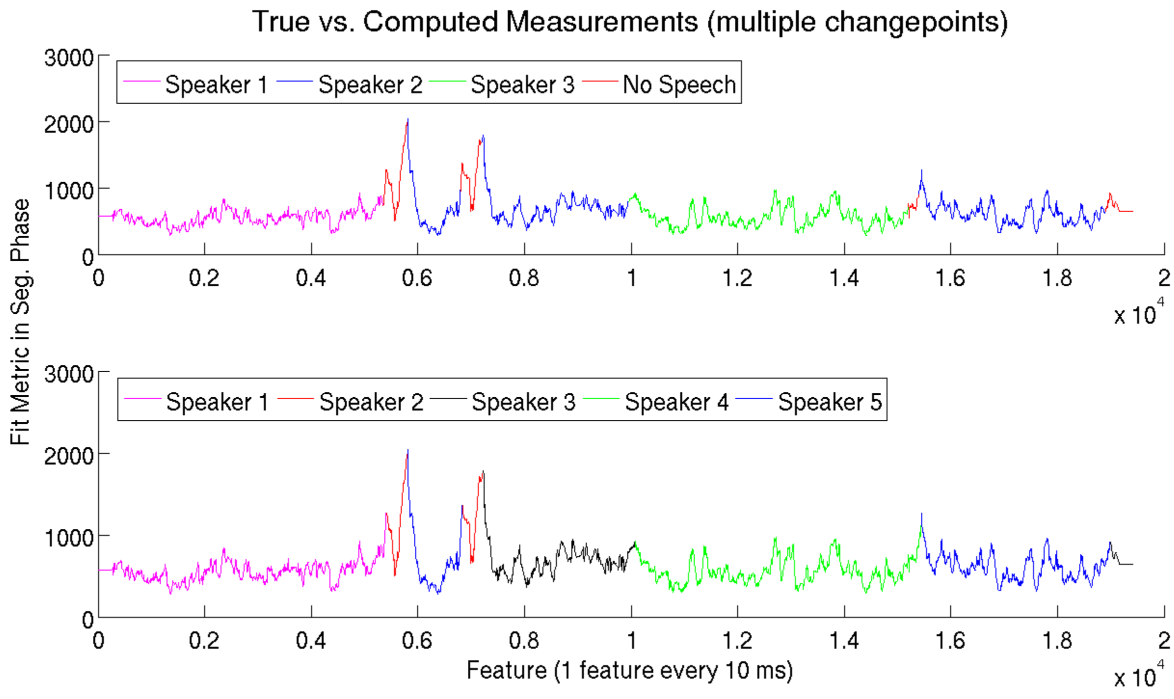


Figure 4.5: Results of complete diarization with increased λ_l

phase is also raised. When compared to Figure 4.5, it shows how all of the extraneous clusters are now merged together.

In this situation, λ_h was increased from 3 (default) to 7, increasing the achieved accuracy to 94%.

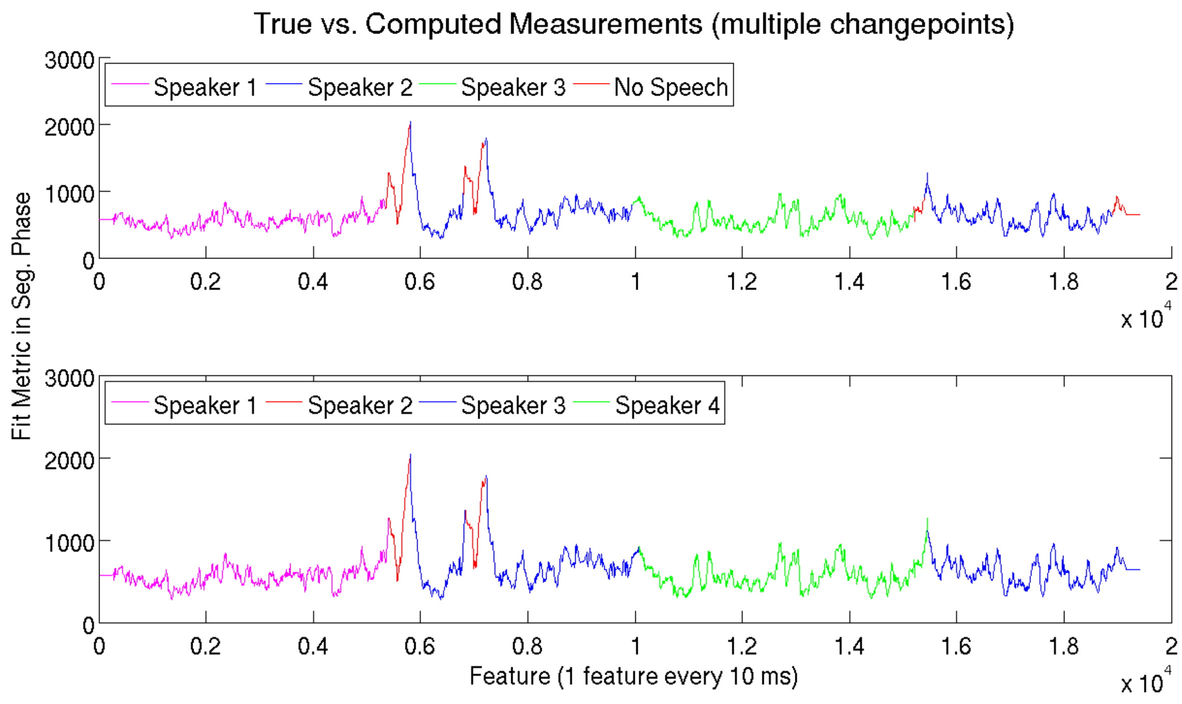


Figure 4.6: Results of complete diarization with increased λ_l and λ_h

4.2.4 Summary of Results

In the two experimental situations the following results were achieved:

Situation	**Accuracy** (Eq. 4.1)
2 Speakers, default λ_l and λ_h	93%
3 Speakers, default λ_l and λ_h	77%
3 Speakers, modified λ_l and default λ_h	77%
3 Speakers, modified λ_l and λ_h	94%

Figure 4.7: Summary of experimental results

Chapter 5

Conclusion

5.1 Contributions

As far as the author is aware, this is the first documented case of speech diarization being performed entirely upon a smartphone. There exist similar applications which perform related conversation logging and speaker identification tasks [15]. The main difference, however, is that these applications offload the raw data to a centralized server to perform the computation, and then send the results back to the phone [15]. This contrast in capabilities demonstrates the viability of a smartphone as a computational platform to perform speech diarization. In a general sense, this research supports the trend of mobile computing and works to further shrink the divide between what is possible on mobile and traditional computers.

5.2 Future Work

Improving Accuracy

As depicted earlier, the most significant source of inaccuracy is the fact that portions of the audio recording that do not contain speech are still modelled as speakers. The system accordingly assigns a speaker identity to these clusters, which skews the overall results. To circumvent this, the system should be modified to employ silence and noise detection as in [16]. Emplying silence detection would allow the system to identify clusters that do not contain speech, and afterwards remove them from the analysis.

Decreasing Computation Time

The feature extraction stage consumes significant processing power to perform the DFT and IDFT operations. The operations are consistent with numerous signal processing application, and there exist many platforms that use hardware acceleration to speed this process up (e.g., hardware multimedia codecs). Currently, this operation is being done entirely in software. Exploiting the availability of hardware acceleration would be one possible optimization.

A more general optimization would be the execution of native code (as opposed to Java - interpreted code) for the intensive calculations (i.e., feature extraction, segmentation, clustering). This is a general optimization than extends beyond the mobile platform, and can be facilitated by using the Java Native Interface provided by the Android Native Development Kit. Furthermore, the use of native code allows for compilation with SIMD instructions such as NEON on the ARM architecture, which can further decrease computation by exploiting parallel computation.

Appendix A

The Mel Scale

The figure on the following page depicts the relationship between frequency and Mel pitch, as originically explained in Feature Extraction from Audio Data, Section 2.2.

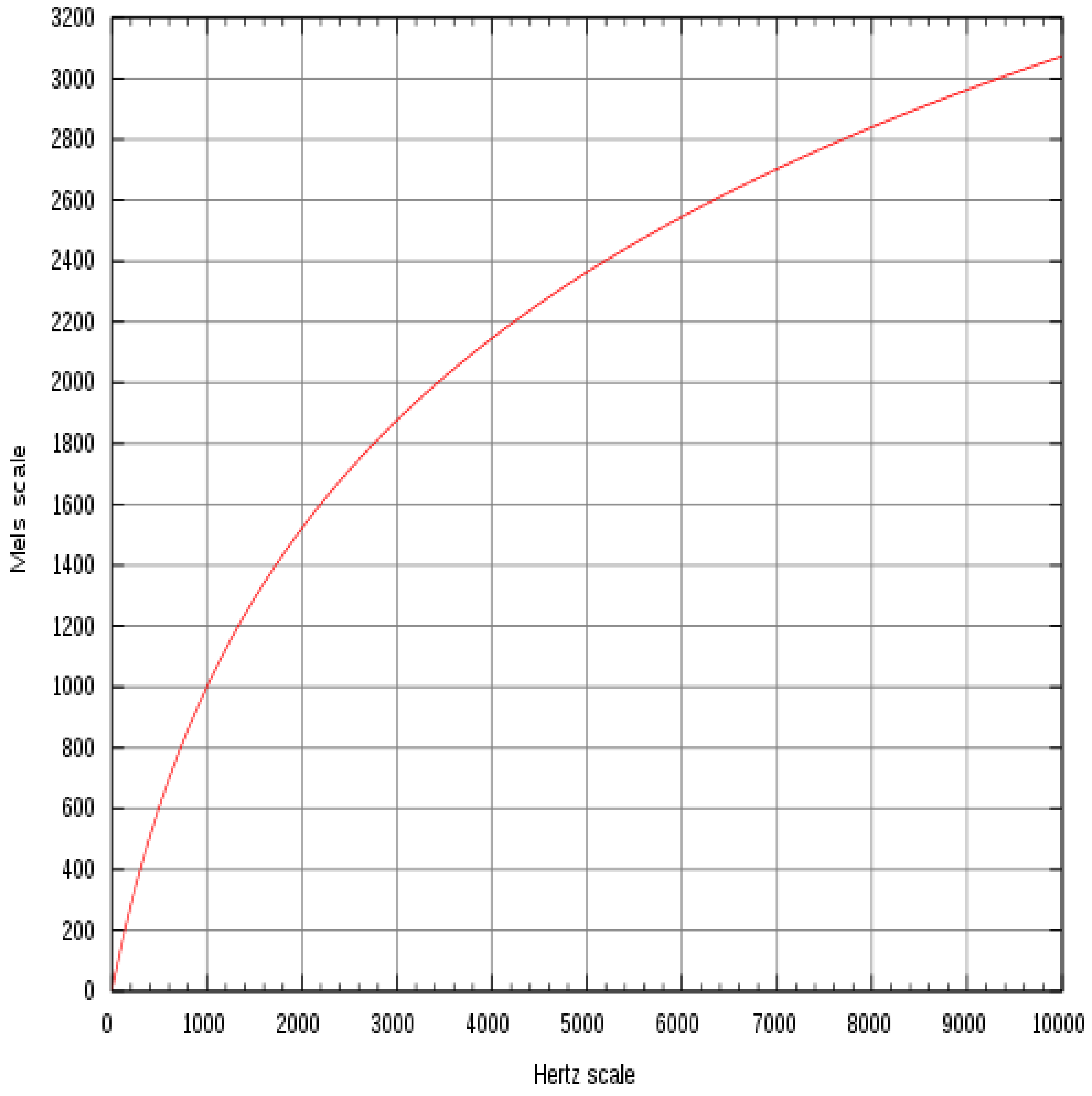


Figure A.1: Mel scale transformation [1]

Appendix B

Prototype Diarization Script

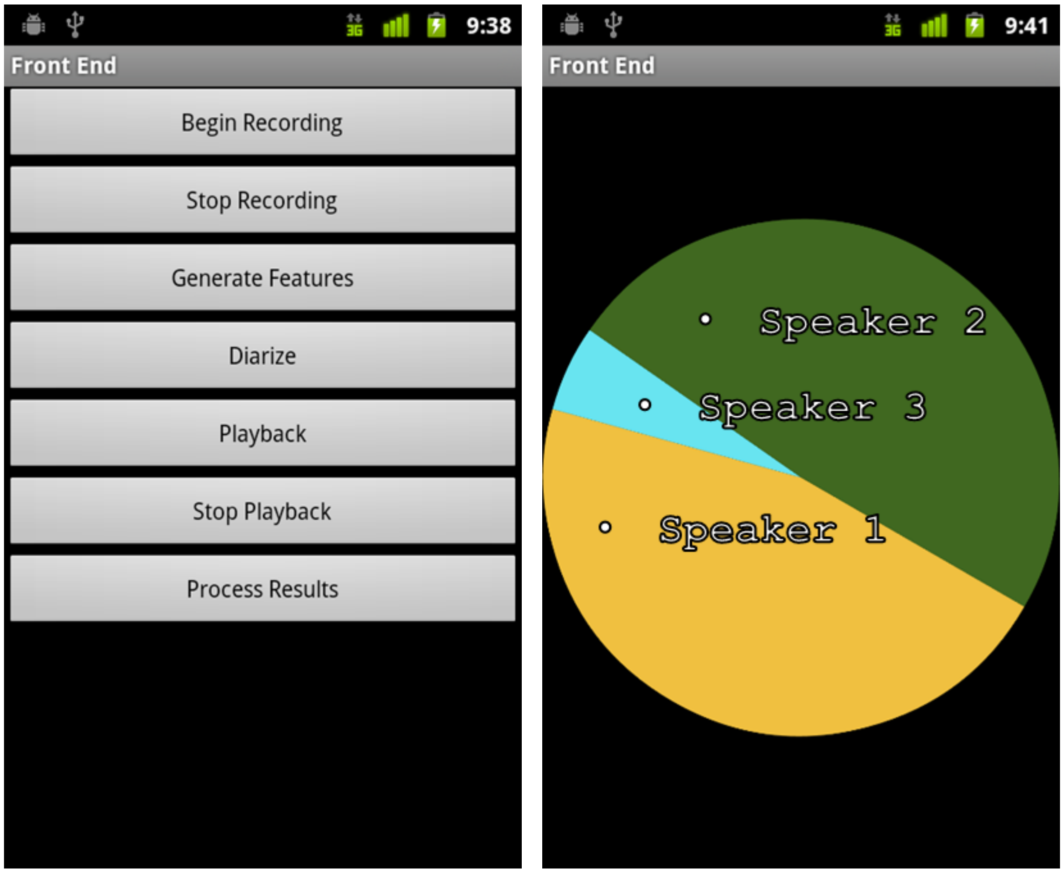
The following is the prototype diarization script, referred to in Prototype Development, Section 3.3

```
#!/bin/bash


show=$1
fDesc="sphinx,1:1:0:0:0:0,13,0:0:0"
fDescCLR="sphinx,1:3:2:0:0:0,13,1:1:300:4"
features="./$show.sphinx.mfc"
uem="./$show.uem.seg"

sphinxTrain="/home/daniel/Development/SphinxTrain-1.0/bin.i686-pc-linux-gnu"

# generate MFCC using SphinxTrain-1.0
$sphinxTrain/wave2feat -i $show.wav -o $features -mswav yes

# Check the validity of the MFCCs
java -Xmx1024m -classpath ./LIUM_SpkDiarization-3.6.jar \
fr.lium.spkDiarization.programs.MSegInit --trace \
--help --fInputMask=$features --fInputDesc=$fDesc \
--sInputMask=$uem --sOutputMask=$show.i.seg $show
```

```
# Speech / non-speech segmentation using a set of GMMs
java -Xmx1024m -classpath ./LIUM_SpkDiarization-3.6.jar \
fr.lium.spkDiarization.programs.MDecode --trace \
--help --fInputDesc=sphinx,1:3:2:0:0:0,13,0:0:0 \
--fInputMask=$features --sInputMask=$show.i.seg \
--sOutputMask=$show.sms.seg --dPenality=10,10,50 \
--tInputMask=sms.gmms $show \

# GLR-based segmentation, make small segments
java -Xmx1024m -classpath ./LIUM_SpkDiarization-3.6.jar \
fr.lium.spkDiarization.programs.MSeg  --trace --help \
--kind=FULL --sMethod=GLR  --fInputMask=$features \
--fInputDesc=$fDesc --sInputMask=$show.i.seg \
--sOutputMask=$show.s.seg $show

# Linear clustering, fuse consecutive segments of the same speaker from the start to the end
java -Xmx1024m -classpath ./LIUM_SpkDiarization-3.6.jar \
fr.lium.spkDiarization.programs.MClust --trace --help \
--fInputMask=$features --fInputDesc=$fDesc \
--sInputMask=$show.s.seg --sOutputMask=$show.l.seg \
 --cMethod=l --cThr=2 $show

# Hierarchical bottom-up BIC
java -Xmx1024m -classpath ./LIUM_SpkDiarization-3.6.jar \
fr.lium.spkDiarization.programs.MClust --trace --help \
--fInputMask=$features --fInputDesc=$fDesc \
--sInputMask=$show.l.seg --sOutputMask=$show.h.seg \
--cMethod=h --cThr=3 $show
```

Appendix C

Application Graphical Interface



(a) Home screen interfance (b) Results of a diarization

Figure C.1: Graphical User Interface of the application

Appendix D

Sample Output of LIUM_SpkDiarization

The following is the output after diarization of a conversation containing two speakers and some noise:

```
test 1 0 2960 U U U S0
test 1 6094 349 U U U S11
test 1 2960 3134 U U U S5
```

Figure D.1: Sample output after clustering

Where the data can be interpreted as following [3]:

- field 1: show name
- field 2: channel number
- field 3: start of the segment (in features)
- field 4: length of the segment (in features)

- field 5: speaker gender (U=unknown, F=female, M=Male)
- field 6: type of band (T=telephone, S=studio)
- field 7: type of environment (music, speech only,)
- field 8: speaker label

Bibliography

[1] K. Vedala. (2008) Scaled vector graphics (svg) image of plot of mel scale vs hertz scale. Mel-Hz_plot.svg. [Online]. Available: http://en.wikipedia.org/wiki/File:Mel-Hz_plot.svg

[2] D. Reynolds and P. Torres-Carrasquillo, "Approaches and applications of audio diarization," in *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, vol. 5, March 2005, pp. 953 – 956.

[3] S. Meignier and T. Merlin, "Lium_spkdiarization: An open source toolkit for diarization," in *CMU SPUD Workshop 2010*, 2010.

[4] H. Aronowitz, "Trainable speaker diarization," in *INTERSPEECH-2007*, 2007, pp. 1861 –1864.

[5] Y. Esteve, T. Bazillon, J. Antoine, F. Bechet, and J. Farinas, "The epac corpus: Manual and automatic annotations of conversational speech in french broadcast news," in *Proceedings of the Seventh conference on International Language Resources and Evaluation*, May 2010.

[6] J. Deller, J. Hansen, and J. Proakis, *Discrete-Time Processing of Speech Signals*. Wiley-IEEE Press, 1999, ch. 6, pp. 352–366.

[7] B. Logan, *Mel Frequency Cepstral Coefficients for Music Modelling*, Compaq Research Laboratory Std.

[8] A. Klapuri. (2004, Oct.) Clustering with gaussian mixtures. classification.pdf. [Online]. Available: http://www.cs.tut.fi/sgn/arg/klap/classification.pdf

[9] D. Reynolds and R. Rose, "Robust text-independent speaker identification using gaussian mixture speaker models," *Speech and Audio Processing, IEEE Transactions on*, vol. 3, no. 1, pp. 72 –83, Jan 1995.

[10] G. Gravier, M. Betser, and M. Ben. (2010, Jan.) audioseg audio segmentation toolkit, release 1.2. audioseg-1.2.pdf. [Online]. Available: https://gforge.inria.fr/projects/audioseg

[11] A. Tadaion, M. Derakhtian, M. Nayebi, and M. Aref, "Glr detector for coded signals in noise and interference," *Scientia Iranica*, vol. 15, no. 2, pp. 170–174, April 2008.

[12] S. S. Chen and P. S. Gopalakrishnan, "Speaker, environment and channel change detection and clustering via the bayesian information criterion." IBM T.J. Watson Research Centre, 1998.

[13] A. Developers. (2011) Audio record — developers. [Online]. Available: http://developer.android.com/reference/android/media/AudioRecord.html

[14] Sphinx4. (2010) Package edu.cmu.sphinx.frontend. [Online]. Available: http://cmusphinx.sourceforge.net/sphinx4/javadoc/edu/cmu/sphinx/frontend/package-summary.html

[15] G. Gopalakrishnan, K. Chimalamarri, and M. Achuthan, "Konversa : A personal audiolog," April 2009.

[16] C. Barras, X. Zhu, S. Meignier, , and J.-L. Gauvain, "Multistage speaker diarization of broadcast news," *EEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1505–1512, September 2006.