Learning behavioural patterns in a mobile context using smartphones

Jakub Ratajczak

Kongens Lyngby 2011
IMM-M.Sc.-2011-76

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

Abstract

This thesis is focused on presenting a system for conversation and speaker detection based on the audio data collected by the smartphone devices. The system allows to detect whether two people are participants of the same conversation and state who was speaking when and for how long. The approach is privacy preserving so it is not possible to state what was said during the conversation.

The system is based on already existing solution but it presents a more practical approach. It uses audio data recorded by the smartphone's built-in microphone as a source of information to analyze. Smartphone application is responsible for the voiced frames detection and it is used as an inteface for displaying conversations' information to the user. Server side web services collect audio features data sent by the smartphone prototype and they perform conversation and speaker detection based on comparison of a data obtained from different smartphones. They also provide detected conversations' data so it can be obtained by the smartphone application.

Firstly audio data analyzing tool has been created in order to facilitate calculations correctness verification of each part of the process. Afterwards smartphone application and server side web services have been implemented. All parts of the process have been thoroughly tested and the results have been analyzed.

Preface

This thesis was prepared at the Department of Informatics and Mathematical Modeling (IMM), at the Technical University of Denmark (DTU) in partial fulfillment of the requirements for acquiring the degree of Master of Science in Computer Science and Engineering.

The thesis supervisors are Jakob Eg Larsen, Michael Kai Petersen and Sune Lehmann Jørgensen, Department of Informatics and Mathematical Modeling, Technical University of Denmark.

Kongens Lyngby, September 2011

Jakub Ratajczak

Contents

Chapter 1

Introduction

During the past few years mobile devices have become integral part of human life. People are carrying their phones everywhere they go to always be in touch with their friends or business partners. Moreover smartphones raised mobile communication to the whole new level by transforming simple phone into advanced handheld computer. Such transformation brought many new oportunities going far beyond the phone related aspects. One of the possible directions is utilizing data collected from different mobile sensors.

1.1 Motivation

Nowadays data science has become very important topic as a way of developing meaningful and well presented information. Scientists from all over the world in many different areas are working on data analysis and visualization. There are many different projects focused on the creation of customized visualizations of collected data (eg. Many Eyes project conducted by IBM [11]). The main challenges to face in this field are defining what useful information can be acquired from gathered data and how to present them in the most convenient way.

Modern smartphones are equipped with many sensors facilitating their use (accelerometer, Wi-Fi, Bluetooth) or providing useful information (GPS, compass).

One of the most popular areas where such data are useful is social life. Based on the information provided by the sensors one can learn about friend's position (GPS), availability (phone in silent or vibrating mode), music tastes (audio player - virtual sensor) or even motion (accelerometer).

Several projects were conducted in this area but this thesis will go beyond their findings. It will add the microphone as a sensor for disambiguating the context based on audio snapshots of the surrounding environment. There are two main applications for microphone data utilization. First one is a detection of environmental sound which can be used to determine the place where person currently stays (e.g. bus, street, silent place).

Second application which is described in this thesis allows to detect the identity of people in conversation and their speaking turns. Such information can be used in many different ways, for example:

- networks of people who knows each other created automatically without user participation;
- history of conversations presenting statistics such as who the person speaks with most frequently, where they usually talk with each other or when was their last conversation.

Conversation and speaker detection technique described in this thesis is based on already existing solution created by Danny Wyatt, Tanzeem Choudhury and Jeff Bilmes [22] but in that case special wearable devices were used instead of smartphones. Thanks to the technical progress modern smartphones are able to handle complex computations and as they are integral part of people's life it makes them perfect for this task.

1.2 Project goals

The main goal of the project is to design and implement prototype application which is able to detect conversation between two people and indicate who has spoken when during the conversation.

One of the most controversial aspects in processing recorded audio data is a privacy issue [22]. It should not be possible to reconstruct conversation speech from collected data but at the same time gathered data should give enough information to state whether conversation occured and who was speaking at a given time.

One of the earliest goals is to create analyzing tool. It should run on stationary computer and allow to perfom first tests of implemented algorithms. Moreover creation of graphic interface will simplify analysis of correctness of all algorithms.

The prototype application should work on a smartphone and it should be able to cooperate with server responsible for comparing data sent by other mobile devices. Here the next goal appears - server side application creation.

The last goal is to test and evaluate results of conversation and speaker detection of different participants and in different environments.

1.3 Thesis structure

The report is divided into several chapters presenting process of making system for conversation and speaker detection.

At the beginning the project on which thesis is based is presented. It describes conversation and speaker detection system which uses wearable devices as a source of data to analyze.

Next chapter is focused on thorough analysis of requirements and steps resulting in conversation and speaker detection. It consists of three main parts: voice and speech detection, conversation detection and speaker segmentation.

In Chapter 4 design of the protopype application is presented. It describes methods for data collecting, processing and visualization as well as the server side structure for analysis of the streams obtained from different sources.

Implementation process is described in Chapter 5. Firstly, the analyzing tool running on a stationary computer had been implemented to test early results of each detection step. Working on proper mobile application started after analyzing tool results had been successfully validated.

Next chapter presents evaluation of the prototype. Several kinds of tests were performed including detection of voiced frames, conversations and speakers.

Finally, the discussion and conclusion chapters focuses on usefulness of collected information and practical examples of its application.

Chapter 2

Related work

So far there are only several different approaches for conversation detection based on recorded audio data. Most of them uses Hidden Markov Model to detect conversation and segment its participants [1, 2, 4, 22] but there are also other approaches which uses Bayesian network [21] or normalized cross-correlation between raw audio signals [6].

2.1 System for wearable devices

Conversation and speaker detection process described in this thesis is based on the work of Danny Wyatt, Tanzeem Choudhury and Jeff Bilmes [22]. They have presented:

> *"...privacy-sensitive methods for (1) automatically finding multi-person conversations in spontaneous, situated speech data and (2) segmenting those conversations into speaker turns."* [22]

Their earlier approach to this topic which was using Bayesian network for voicing and speaker segmentation is described in [21], the social aspects of conversation detection process are presented in [20].

Privacy-sensitive approach ensures that it is not possible to reconstruct from collected data what was said during the conversation. Only information about three audio features - number of autocorrelation peaks, maximum peak and relative spectral entropy - is stored. Whole detection process is divided into three main steps:

- voiced frames detection - based on the Hidden Markov Model whose hidden state is a binary variable indicating whether frame is voiced or unvoiced and whose observations are three features mentioned earlier;
- conversation detection - based on the mutual information algorithm which compares audio streams from different sources and indicates whether they belongs to the same conversation;
- speakers segmentation - based on the Hidden Markov Model whose hidden state has four values indicating whether no one, first person, second person or someone else is speaking and whose observations are log ratios of the speaker frames energies.

Voiced frames and conversation detection part presented in [22] extends one described by Sumit Basu [1, 2]. Detailed analysis and thorough description of each step is presented in the next chapter.

Approach described in [22] was tested using special wearable devices. Data was first collected and then conversation and speaker detection was performed. Approach described in this thesis differs mainly in that it is designed for smartphones and detection process is performed in real-time. It makes this approach more practical.

Chapter 3

Analysis

This chapter is focused on the detailed analysis of the conversation and speaker detection process. One of the main assumptions for the adopted approach is recorded data privacy preservation. To satisfy this condition an approach based on storing only several features that characterizes recorded data is introduced. Afterwards all process's steps and required calculations are thoroughly described.

3.1 Privacy preserving approach

To detect conversation and speakers participating in it the audio data from each participant's phone has to be compared. Such comparison needs to be handled in one central point which collects samples from different sources. Therefore it should be impossible to reconstruct from retrieved data what each participant or person within the range of the microphone was saying.

To satisfy this condition privacy preserving approach is introduced. It ensures that the potential intruder will not be able to obtain any sensitive information based on the collected data.

The approach is based on obtaining three audio features for voiced frames detection [22]. The features are:

- non-initial maximum autocorrelation peak;
- the total number of autocorrelation peaks;
- relative spectral entropy.

One additional feature - energy - is also calculated. It is used for speakers segmentation. All this features provide enough information to detect conversations and determine who was speaking when but not what was said. They are also robust to noise [22].

3.2 Features calculation

All features are calculated on per frame basis. Each frame consists of 256 samples with overlap of 128 samples. The frequency of recorded data is set to 8 kHz so each frame lasts 32 milliseconds.

3.2.1 Autocorrelation peaks

Autocorrelation allows to find periodic components in the signal by applying cross-correlation with itself [14]. Voiced frames have a property that the signal periodically repeats itself (Figure 3.1). Therefore voiced frames are characterized by small number of strong autocorrelation peaks. On the other hand unvoiced frames result in a large number of small peaks [1].

The normalized autocorrelation of the signal s of length N is calculated as follows:

$$a[k] = \frac{\sum_{n=k}^{N} s[n]s[n-k]}{(\sum_{n=0}^{N-k} s[n]^2)^{\frac{1}{2}} (\sum_{n=k}^{N} s[n]^2)^{\frac{1}{2}}} \tag{3.1}$$

To preserve all information about peaks nature, maximum peak value and number of peaks are chosen as the first two features.

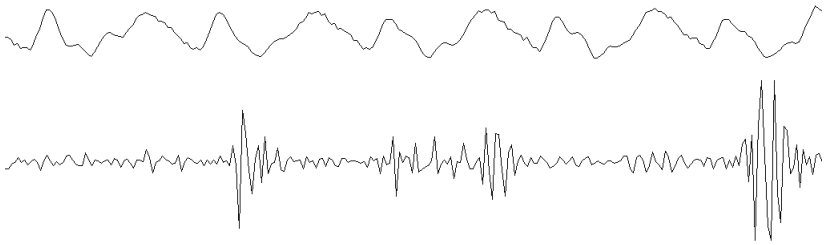

Figure 3.1: Waveforms of voiced (top) and noisy (bottom) signal.

3.2.2 Relative spectral entropy

It can be observed on the spectrogram (Figure 3.2) that the fast Fourier transform (FFT) magnitudes of voiced frames are easily distinguishable from unvoiced ones. They are characterized by a series of very strong peaks while in case of unvoiced frames the spectrum is noisy. Taking it into account it can be stated that entropy of unvoiced frame's spectrum is relatively high. Therefore spectral entropy [17] analysis allows to distinguish voiced frames from other noisy ones.

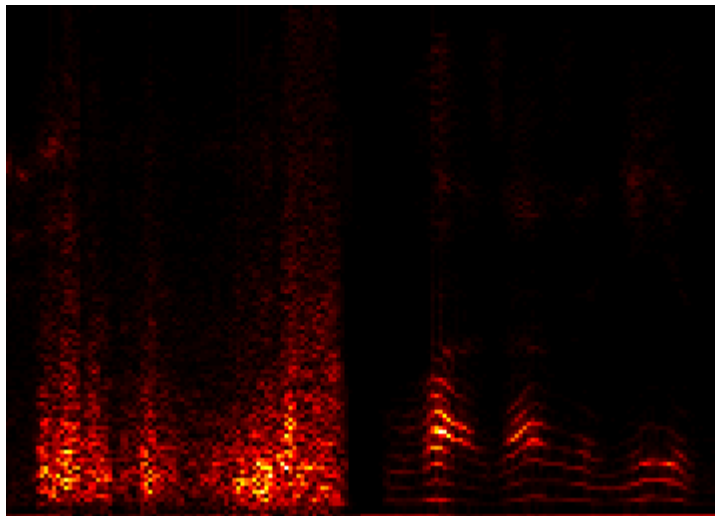

Figure 3.2: Spectrogram of unvoiced (left part) and voiced (right part) frames.

Instead of calculating spectral entropy, relative spectral entropy term is introduced. It is a Kullback-Leibler divergence [18] between the current spectrum and the local mean spectrum of neighboring 500 frames [1]. It improves results in case of constant noise presence. It is calculated as follows:

$$H_r = -\sum_w p[w] \log \frac{p[w]}{m[w]} \tag{3.2}$$

where $p[w]$ is normalized Fourier transform and $m[w]$ is the mean spectrum of neighboring 500 frames [1].

3.3 Voiced frames detection

The first step in conversation and speakers detection is to find voiced frames. This part of the process is based on Hidden Markov Model whose hidden states are variables indicating whether frame is voiced or unvoiced and whose observations are three features presented in the previous section.

3.3.1 Hidden Markov Model

Hidden Markov Model (HMM) is a temporal probabilistic model which is widely used in speech recognition. In the model the state of the process is described by a single discrete random variable where the possible values of it are the possible states of the world. The current state of the process is unknown (hidden) but the output dependent on this state can be obtained [16].

HMM is characterized by the following elements:

- hidden states;
- observations;
- the state transition probability matix;
- the observation probability matrix;
- the initial state probability matrix [15].

In the case of the project there are 4 hidden states specifying whether current frame is unvoiced, voiced, contains speech or contains both voice and speech. The graphical representation of such model is presented on Figure 3.3.

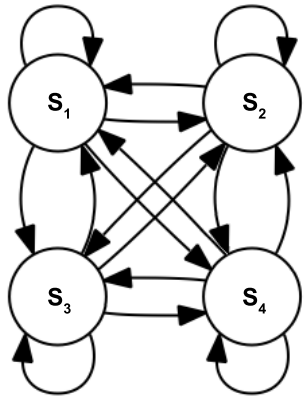

Figure 3.3: HMM with four hidden states (figure from [5]).

As it was mentioned before three features are used as observations for HMM. Their probabilities are modeled with a single three-dimensional, full covariance Gaussian. Probability distributions are calculated during learning process which uses observations sequences with manually marked speech and voiced frames.

There are three main problems which can be solved using HMM:

- calculating the probability of the observation sequence;
- finding the most likely state sequence;
- adjusting the model parameters to maximize the probability of the observation sequence [15].

HMM in this project is used to find the most likely sequence of hidden states based on given observation sequence. It is done by the use of the Viterbi algorithm [3].

3.4 Conversation detection

Comparison of states sequences acquired from different sources allows to detect whether there was conversation between two or more people. Such comparison can be done using mutual information algorithm.

3.4.1 Mutual information

Mutual information algorithm allows to determine similarity of data from two different sources. By performing it on the states sequences calculated by HMM it can be determined whether conversation occured.

The mutual information between streams A and B for conversation window w is calculated as follows:

$$I(A_w; B_w) = \sum_{v,v'} P(A_w = v, B_w = v') log \frac{P(A_w = v, B_w = v')}{P(A_w = v)P(B_w = v')} \tag{3.3}$$

where $P(A_w = v, B_w = v')$ is the joint probability distribution function of A_w and B_w, $P(A_w = v)$ and $P(B_w = v')$ are the marginal probability distribution functions of A_w and B_w [13].

The mutual information algorithm result should be significantly higher for the samples' pair which were recorded in the same place than for the samples recorded in different locations.

3.5 Speaker segmentation

The last step in the detection process is to determine when each person taking part in the conversation was speaking. It is done by the use of new Hidden Markov Model which is described in this section.

3.5.1 Energy of the signal

One of the main features used in the speaker segmentation process is the energy of each frame. Differences between energies' log ratios from two different sources are used as observations for the new HMM [22].

Each frame energy e_{raw} is calculated as follows:

$$e_{raw} = (\sum_{i=1}^{N} h[n](s[n])^2)^{\frac{1}{2}} \tag{3.4}$$

where N is the frame size and $h[n]$ is a Hamming window of length N.

To improve robustness to noise a regularized energy term is introduced [1]. The regularized energy e_{reg} of the K-frame voicing segment is calculated as follows:

$$e_{reg} = \frac{1}{K}[(\sum_{i=1}^{K} e_{raw}[i]) - e_n^2]^{\frac{1}{2}} \tag{3.5}$$

where e_2^n is the per-frame energy of the noise (average per-frame energy over non-speech regions).

Such approach decreases impact of the noise in speakers segmentation process.

3.5.2 HMM for speakers turns detection

HMM for speaker segmentation consists of the following states:

- no one was speaking;
- person A was speaking;
- person B was speaking;
- someone else was speaking.

The observations are the differences between energies' log ratios from two different sources:

$$r_s = \log g_s^A - \log g_s^B \tag{3.6}$$

where g_s^A is the mean energy of frame s in stream A.

The observations values should be positive when person A was speaking and negative when it was person B turn, whereas they should be close to 0 when there was silence or someone else was speaking [22].

As it was with the previous HMM, learning process of this HMM is based on observations sequence with corresponding states set manually.

Chapter 4

Design

The approach presented in [22] was tested using special wearable devices. Gathered data were analyzed and compared after collection process was finished. Such approach was appropriate only for the test purposes and could not be applied in the real world.

The approach described in this thesis aims to be much more practical and useful in the daily life. The main difference is that smartphone devices are used as the core of the system and detection process can be executed in real-time.

4.1 General idea

The main purpose of the conversation and speaker detection process is to provide different useful information to the user about convesations in which he took part and their other participants. To make this information as useful as possible it is important to be able to acquire them up-to-date, shortly after they were collected.

Audio recording and its analysis needs to be performed continuously so any data about current conversation would not be lost. To provide the most recent infor-

mation new data needs to be available continuously and in small time intervals after they were collected. Moreover to enable comparison of the data from different sources (devices) one central point which will perform proper calculations needs to be defined.

Simple client-server architecture should be able to handle all needed actions. Each user's smartphone will exchange information with the server which can store all data in the database (Figure 4.1).

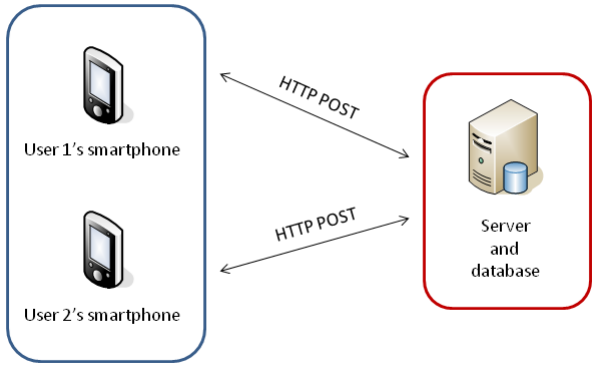

Figure 4.1: Proposed system architecture.

Process which will satisfy presented requirements can be described by the following sequence of events:

- audio recording is started on the smartphone device and is performed continuously;
- after every certain amount of time recorded data is stored as a smaller sample;
- appropriate calculations are performed on such sample's data (voiced frames detection);
- calculated data is sent to the server just after it become available;
- every certain amount of time the server compares available samples data acquired from different devices (conversation and speaker detection);
- calculations results become available on the server;
- the results are obtained by the smartphone device and they are displayed to the user.

Figure 4.2 presents described sequence as a communication flow sequence diagram.

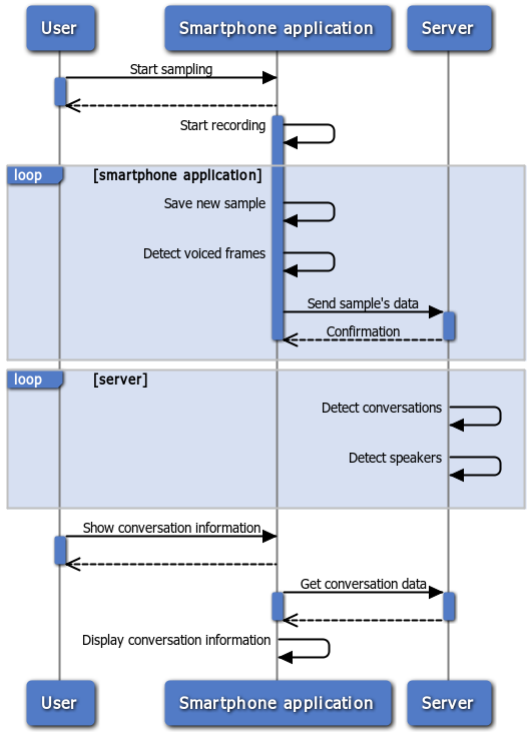

Figure 4.2: Communication flow sequence diagram.

4.2 System overview

The system for audio data collection, analysis and conversation and speaker detection presented in this thesis is composed of three main components:

- audio data analyzing tool for early tests purposes;
- mobile application prototype for audio data collection, features extraction and user interaction;
- web services and database for data collection, comparison and analysis.

All these components will be implemented as a separate applications where analyzing tool is designed to run on a stationary computers, mobile application works on smartphone devices with Andrioid operating system, whereas web services and database are hosted by Google App Engine.

4.3 Audio data analyzing tool

Analyzing tool designed to run on a stationary computer will be created to test all elements of conversation and speaker detection process. It will allow to load files containing audio signal data recorded on the smartphone and analyze their content. Application will consist of three main views with various options for each one:

- waveform and spectrogram analysis - its main purpose is to perform HMM learning process, waveform and spectrogram visualization will allow to manually mark voiced frames in the loaded sample and based on so-created observations sequence teach new HMM, this view can be also used to check the correctness of detected voiced frames by the smartphone prototype;
- calculated features presentation - it will allow to check correctness of dependancies between calculated features and voiced frames occurences;
- audio samples comparison - its main purpose is to perform conversation and speaker detection process and validate its correctness, it wiil allow to compare spectrograms of two different samples.

4.4 Mobile application prototype

The main objective of the prototype application is to record audio samples, calculate their features and pass this information to the server whose web services are responsible for conversation and speaker detection. The second important goal is to provide an interface for the user where he will be able to check different information about detected conversations or their participants and display other useful statistics.

4.4.1 Mobile platform

Nowadays one of the most popular mobile platform is Android operating system and mainly because of that it has been chosen as the environment for the prototype application. Moreover by the use of Java programming language it is easy to combine parts of code of the analyzing tool and server side web services which are also implemented in Java language [9].

4.4.2 Data collection and processing

The first step in the conversation and speaker detection process is to collect audio data of the surrounding environment. Such collection needs to be performed constantly so any data about the conversation would not be lost. Moreover to be able to display information about current conversation, collected data need to be processed and passed to the server on the fly. Table 4.1 presents detailed description of such case.

Table 4.1: Collect and process audio recording data use case

Name	Collect and process audio recording data
Actors	Smartphone application, server
Description	Smartphone application is responsible for continuous audio recording and collected data processing.
Preconditions	Application is installed and started on the mobile device. The smartphone is equipped with microphone and has active internet connection.
Steps	Recording thread is started automatically. New audio sample data is stored every 30 seconds. Available data is processed (voiced frames detection) by a separate thread.
Post conditions	Processed data is sent to the server where further processing is performed.

To meet all conditions audio data are recorded constantly but they are stored as a separate samples every 30 seconds. In order to send all necessary information to the server collected data need to be firstly processed. Mobile application is responsible for finding voiced frames in recorded sample data. As it was mentioned before three features need to be calculated to perform detection process. Based on their values and by the use of Hidden Markov Model voiced frames

are detected. Each frame energy also needs to be calculated so the server would be able to detect speakers turns. All this calculations are performed on currently available 30 seconds samples. Afterwards information about each sample's frames states and energies is sent to the server. Thus sample data can be processed by the server with only small delay.

4.4.3 Results visualization

When the process of conversation and speaker detection is successfully finished different conversation and participants information can be displayed to the user. As this part of the process is performed on the server side all data are obtained by the mobile application using appropriate web service. The prototype application will provide only basic information and very simple visualization of acquired data but different ideas for the final version of the application are pesented later in this chapter.

4.5 Server side

Data collected by different mobile devices needs to be compared with each other to provide any useful information about conversations and their participants. Such comparison has to be done in one place accessible for all devices. The reasonable solution is to store and analyze this data on the server side.

4.5.1 Google App Engine

Google App Engine has been chosen as a server for hosting different web services of the project. It allows to run web applications written in Java language which is very useful as all parts of the project are implemented in this language. Reusage of different pieces of the code is very convenient. Moreover integrated database can be used to store all data about samples, conversations and participants [8].

To handle all needed actions four web services are introduced. Each of them is responsible for differerent part of the process.

4.5.2 Time synchronization service

This web service returns current server time. It allows to synchronize time on all devices which uses prototype application.

4.5.3 Sample data upload service

This web service allows to pass information about audio samples collected by the smartphone application. Only information about each frame energy and voiced frames states is transferred to the server. Server is storing this data in the database from where they can be further analyzed.

4.5.4 Conversation and speaker detection service

This web service is responsible for finding conversations and users who are their participants. It also calculates speakers turns. It runs as a cron job every 5 minutes. All samples which were not yet processed are extracted from the database and they are analyzed together with other ones which were recorded at the same time. Table 4.2 presents detailed description of such use case.

Table 4.2: Compare audio samples' data use case

Name	Compare audio samples' data
Actors	Google App Engine
Description	Web service is responsible for comparing different audio samples' data in order to detect conversations and speakers turns.
Preconditions	New samples' data sent by the smartphone application are stored in the database.
Steps	All not yet processed samples' data are obtained from the database, the ones recorded at the same time are compared with each other, conversations and speakers turns are detected.
Post conditions	Conversation and speaker detection data is stored in the database and become available to obtain by the smartphone application.

4.5.5 Statistics provider service

This web service provides different information about found conversations and their participants which can be presented to the user. Table 4.3 presents detailed description of such use case.

Table 4.3: Get conversations and speakers data use case

Name	Get conversations and speakers data
Actors	User, Google App Engine, Smartphone application
Description	User chooses type of the information which he wants to have displayed.
Preconditions	Application is installed and started on the mobile device. The smartphone has active internet connection.
Steps	User chooses type of the information. Application invokes server's web service responsible for obtaining conversations and speakers data. Depending on the chosen information type proper data is returned.
Post conditions	Application displays information visualization to the user.

For the purpose of this thesis the service will provide only basic information about conversations which can be processed by the smartphone prototype. Examples of other information which could be also provided by the service are presented in the next section.

4.6 Thoughts on the final version

For the purpose of the project only the application prototype will be implemented. It will provide basic functionality with very simple user interface. In this section ideas for the final application appearance and functionality are presented.

4.6.1 Conversations

One of the most basic data which can be presented to the user is information about his conversations with other participants. Based on collected data such summary can contain following information about each conversation:

- date (10.09.2011, 13:42);
- length (34 minutes, 13:42-14:16);
- participants data;
- each participant's speaking time (12 minutes, 35%);
- speakers turns.

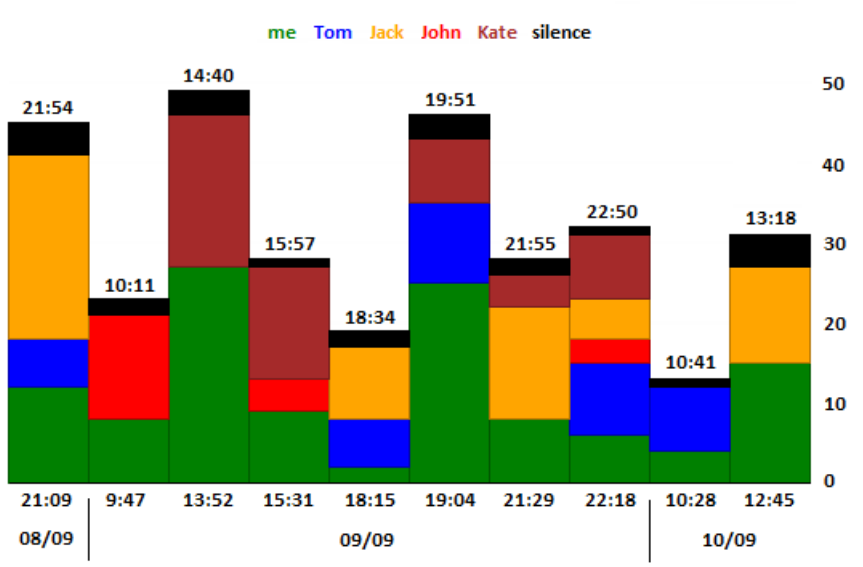

Figure 4.3: Conversations data visualization example.

The proposition of conversations data visualization is presented on Figure 4.3. Each conversation is represented by a vertical bar. Each bar is divided into several parts characterized by different colors. Each color is assigned to different participant (eg. blue color is assigned to Tom). The size of a bar segment represents how much time a participant was speaking during given conversation.

For example there are two detected conversations on 10th September. First one took place at 10:28 and finished at 10:41. There were two participants: the user and Tom. It can be noticed that Tom was speaking twice longer than the user (the height of the blue segment is twice higher then the green one). There was also small amount of silence.

Such visualization could be also used to present conversations information only with specified participant. From presented chart it can be concluded who is usually dominating during the conversations or how often is the user in conversation with given person.

Figure 4.4: Conversation timeline visualization example.

The more detailed information about specified conversation (speakers turns) can be presented as shown on Figure 4.4. The conversation data is displayed on a timeline. Different segments' colors are specifying the participants. Segment's width defines speaking duriaton.

4.6.2 Summaries

Beside presenting detected conversations data it would be also interesting for the user to see different statistics and summaries of collected data. Figure 4.5 presents example of conversations summary visualization.

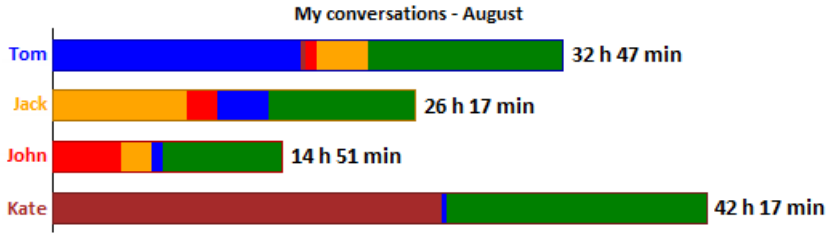

Figure 4.5: Conversations summary visualization example.

Each bar represents sum of user's conversations durations in a given period of

time with a given participant. There can be more than two participants in the conversation. That is why other users' segments can be visible in the given participant bar. For example John during some conversations with the user was speaking also with Tom and Jack. That is why on the John's bar also Tom's and Jack's segments colors are visible. Kate during conversations with the user was also speaking for a short time with Tom but she never spoke to John or Jack.

Such summary gives information on how often the user is speaking with given person, who is usually dominating in the conversation and how often other people are taking part in the conversation with given person.

Chapter 5

Implementation

As it was presented in the previous chapter conversation and speaker detection system described in this thesis consists of three main components:

- audio data analyzing tool;
- mobile application prototype;
- server side webservices and database.

In order to test algorithms used in the process, analyzing tool was implemented first. After successful verification of all its elements smartphone prototype application and server side implementation have been started.

5.1 Overview

All parts of the project are implemented in Java programming language which simplifies reuse of the analyzing tool code in the smartphone prototype and web services. HTC Desire phone [10] with Android operating system is used as a hardware platform for the smartphone prototype. Web services and database are hosted by Google App Engine.

5.1.1 Limitations

The implemented system is considered as a prototype. There exists some limitations which would be overcame in the final version:

- smartphone prototype does not work in background, it runs as a normal Android application;
- voiced frames algorithms can be further optimized to run efficiently;
- only conversation between two users can be detected;
- threshold for mutual informaton algorithm specifying whether conversation occured is set manually (it is not calculated dynamically);
- the server provide only basic information about detected conversations.

5.1.2 Resources used in the project

One of the most important libraries used in all parts of the project (analyzing tool, prototype application and web services) is *Jahmm* library designed by Jean-Marc François [7]. It is a Java implementation of Hidden Markov Model and different algorithms connected with it. For the project purposes the library has been modified to be able to handle learning process based on already specified states for the observations sequence.

Some components also reuse small parts of the SenseBook [12] application code (mainly HTTP connection part).

5.2 Audio data analyzer

As it was mentioned before analyzing tool has been created to provide convenient testing environment. Through the use of graphical interface it allows to observe results of different algorithms calculations and validate their correctness.

5.2.1 General structure

Audio data analyzer was implemented as a Java applet designed to run on a stationary computer. It consists of several different components which overview

is presented in Table 5.1.

Table 5.1: Audio data analyzer components overview

Class	Description
Analyzer	resposible for updating application state depending on user's actions, it also invokes other components actions
DataHolder	stores all samples' data during application execution
VoiceDetection	performs all algorithms required by voiced frames detection process
ConversationDetection	performs all algorithms required by conversation detection process
SpeakerSegmentation	performs all algorithms required by speakers turns detection process
Features	calculates and provides all features data (to display them to the user)
Views	responsible for displaying chosen view to the user
FilesManager	handles creation and loading of different files used by the application
Recording	allows to record new sample using built-in microphone

5.2.2 Supported files types

The analyzing tool uses data stored in several different files types (Table 5.2). To simplify files' load and save actions all data are stored as a serializations of different Java classes.

The ".hmm" file type contains learned HMM data. This file is created during learning process which is performed only by the analyzing tool. Data from this file is used by the smartphone prototype and server side webservices. Thus it is not needed to perform learning process caculations on their side. They just use already learned HMM.

The ".sample" file type contains digital signal values of recorded sample. Such data allows to reconstruct recorded audio (for the test purposes) and calculate all needed features. This files are created by the smartphone application and are used by the analyzing tool. There is also a ".learn" file type which is used

Table 5.2: Supported files types

File type	Used by	Action
.hmm	analyzing tool smartphone application web services	creation and utilization utilization utilization
.sample	analyzing tool smartphone application	creation and utilization creation
.learn	analyzing tool	creation and utilization
.obs	analyzing tool	creation and utilization
.states	analyzing tool smartphone application	creation and utilization creation

in HMM learning process for voiced frames detection. It differs from ".sample" file type only in having different extension.

The ".obs" file type contains information about log ratios differences between two samples. This files are created and used only by the analyzing tool in the HMM learning process for speaker segmentation.

Depending on the process's current step the ".states" files type contain information about each sample's frame voicing state (voiced frames detection step) or values defining speaker turns (speakers segmentation step). This files, together with corresponding ".learn" or ".obs" files, are used in the HMM learning processes.

5.2.3 Waveform and spectrogram visualization

This view (Figure 5.1) displays waveform of the chosen frame (256 samples) and spectrogram of 512 consecutive frames. Visual representation combined with possibility to play audio sample allows to manually detect and mark (checkboxes) voiced and speech states and save them as a file (*Save states* button). Data from both audio signal file and states file is used in Hidden Markov Model learning process (*Learn HMM* button). Files containing learnt HMMs are used by the mobile prototype and server side web services.

This view is also used to check whether voiced frames are detected correctly using already learnt HMM (*Detect voiced frames* button). *Play voiced frames* button allows to play only frames marked as voiced.

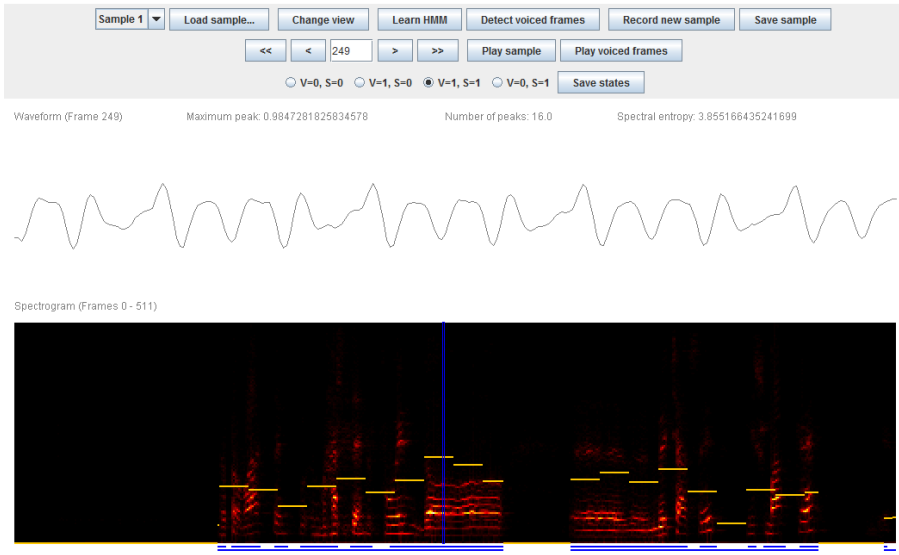

Figure 5.1: Waveform and spectrogram view.

5.2.4 Calculated features presentation

This view (Figure 5.2) displays visual representation of all features calculated during detection process: maximum peak, number of peaks, spectral entropy and energy. Based on the information about voiced frames it can be observed what features magnitudes characterizes voiced frames.

5.2.5 Audio samples comparison

This view (Figure 5.3) displays spectrograms of two loaded samples with marked voiced and speech frames. It allows to perform conversation and speaker detection process (*Detect speakers* button) in order to validate its correctness. Information about mutual infomation value and each speaking frame energy log ratios is displayed and speakers turns are marked on the spectrograms as a green lines. Since speakers detection process is based on Hidden Markov Model this view contains also option to create learning data for the model and then perform learning process (*Learn HMM* button). Speakers turns can be marked using checkboxes and data about energy log ratios and speakers turns can be saved to a file (*Save speakers data* button).

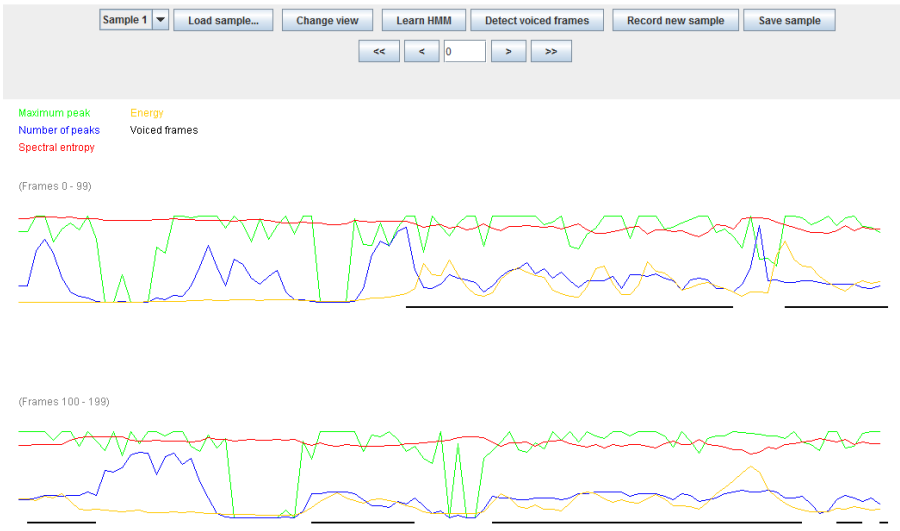

Figure 5.2: Calculated features view.

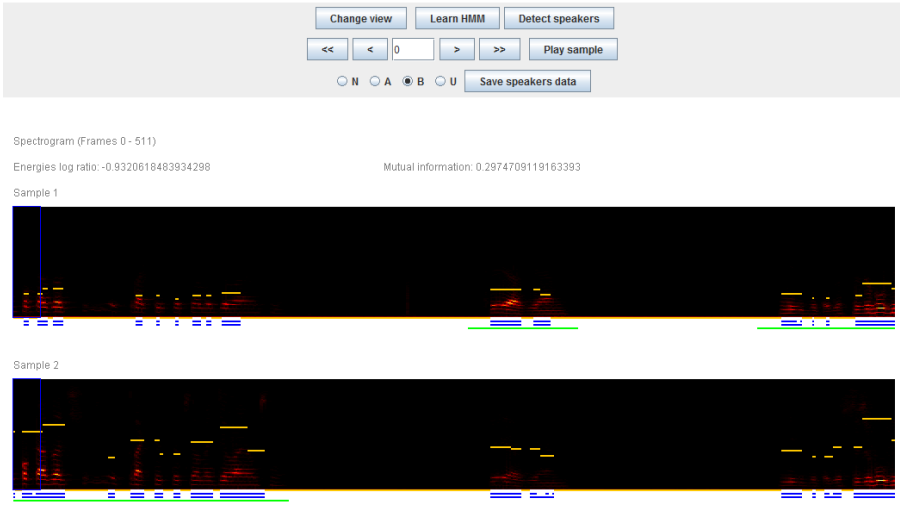

Figure 5.3: Audio samples comparison view.

The *Learn HMM* button also creates ".hmm" file used by the server's web service responsible for speakers segmentation.

5.3 Mobile application prototype

The prototype has been designed to run on Android operating system. Its main purpose is to perform audio recording, collect recorded data, process them and send processed information to the server. It also provide user interface for displaying server calculations results.

5.3.1 Overview

The prototype components are presented in Table 5.3. Application allows to perform three basic actions:

- start/stop audio sampling;
- record training sample;
- get data about conversations from the server.

The main screen of the prototype is presented on Figure 5.4. When the application is started learnt HMM model (provided by analyzing tool) is loaded first. Then user can perform previously listed actions. When the *Start* action is performed sampling process is started. Three different threads take part in it:

- Detector - Firstly it obtains current server time by invoking proper web service. Then local phone time is synchronized with the server time. To synchronize sample's recording start times on all devices recording process is started at the moment when current time's seconds number reaches 0 or 30. The countdown indicating when recording will be started is displayed to the user. When the coundown reaches 0 recorder and sender threads are started. Thread starts to wait for the first sample to appear in the recorded samples list. If there are new samples to process, voiced frames detection process is performed. Processed data is added to the processed samples list. Afterwards thread again waits for the new sample to process.

Table 5.3: Prototype components overview

Class	**Description**
Constants	stores url adresses for the server web services
ConversationInfoData	stores all conversation's information obtained from the server
ConversationsData	transforms JSON format conversations data (obtained from the server) into ConversationInfoData objects
Detector	controls all threads and voiced frames detection process
Features	calculates sample's features
HttpHelper	performs communication with web services
MainActivity	Android OS activity class, responsible for the application initialization and interaction with the user
RecordedData	stores unprocessed data about recorded sample
SampleData	stores processed data about recorded sample
ServerDataCollector	responsible for collecting and processing data from the server
ServerResponse	specifies whether the server response was successfully obtained

- Recorder - Audio recording is performed continuously but every 30 seconds new separate sample is created based on the data recorded during last 30 seconds. Sample's data is added to the recorded samples list.

- Sender - If there exists already processed samples (processed samples list is not empty) their data is converted into JSON format and proper web service with request message containing converted data is invoked. If the web service response is successful sent sample's data is removed from the processed samples list. Thread starts to wait for new processed sample.

All described threads are running concurrenlty until the *Stop* action is performed.

The *REC Training sample* action allows to record one minute sample which is stored on the phone's flash card. All recorded samples and files containing information about detected voiced frames are also automatically stored so they could be later analyzed using the analyzing tool.

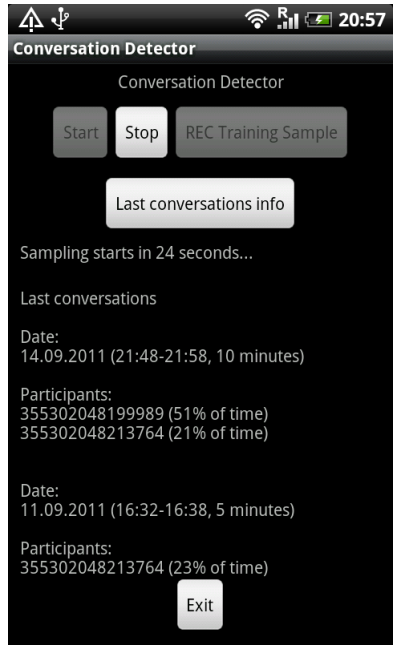

Figure 5.4: Prototype application screen.

The *Last conversation info* action displays information about last ten conversations the user was participating in. The information include conversation's date, duration, participants ids and percentage of time given participant was speaking during the conversation.

5.3.2 Voiced frames detection process

When a new sample is added to the recorded samples list, voiced frames detection process on a new sample data is performed. Part of the code responsible for this process is presented on Figure 6.2.

Firstly sample's data is saved as a file to allow its further analysis by the analyzer tool. Then two-pass process is started. In the first pass all features are calculated and converted to the observations vector. Afterwards using the vector and sample signal data voiced frames are detected but the process is not finished yet because sample data improvement is performed [2]. It adds low-power Gaussian noise signal to each frame in order to minimize infuence of small

```
for (int i=0; i<2; i++){
    for (short[] aData : sample){
        float[] spectrum = features.calculateSpectrum(aData);
        float spectralEntropy
            = features.calculateSpectralEntropy(spectrum);
        double[] peaksInfo = features.calculatePeaks(
            features.calculateAutoCorrelation(aData));
        if (i==0) energies[index] = features.calculateEnergy(aData);
        double[] obs = {peaksInfo[0], peaksInfo[1], spectralEntropy};
        observations.add(new ObservationVector(obs));
        index++;
    }
    if (i==0){
        sample = features.improveSample(
            sample, getHmmResult(observations));
        observations.clear();
    }
}
int[] states = getHmmResult(observations);
```

Figure 5.5: Voiced frames detection source code.

noisy periodic signals. The amount of noise to use is estimated using non-speech frames detected during first pass. The second pass performs the same actions excluding signal energies calculations and sample improvement.

5.4 Server side

This part of the system consists of four web services and datastore component, both hosted by Google App Engine.

5.4.1 Google App Engine

To simplify required web services implementation process and database management Google App Engine service was selected as a server side platform. It offers *"fast development and deployment; simple administration, with no need to worry about hardware, patches or backups; and effortless scalability"* [8].

One of the key factor is that web services can be implemented using Java programming language (Python and Go languages are also available). Eclipse de-

velopment environment used in the project allows to deploy new versions of web services quickly and in a very convenient way (using Google App Engine plugin). The database tables and their records, error logs and other information about web application can be accessed using web dashboard. Web services are publicly available through *.appspot.com domain.

5.4.2 Datastore entities

For the project purposes seven different kinds of entities (tables) were created in the database. Their detailed descriptions is presented in Table 5.4. They store all required information about samples recorded by the smartphone application and data calculated during conversation and speaker detection process. Datastore entities are firstly defined as Java classes used by the web application. The sample class is presented on Figure 5.6.

```
@PersistenceCapable(identityType = IdentityType.APPLICATION)
public class States {
    @PrimaryKey
    @Persistent(valueStrategy = IdGeneratorStrategy.IDENTITY)
    private Long Id;

    @Persistent
    private Long SampleId;

    @Persistent
    private Text States;

    public States(Long SampleId, Text States) {
        this.SampleId = SampleId;
        this.States = States;
    }

    [...]
}
```

Figure 5.6: States entity class.

Table 5.4: Datastore entities

Entity kind	**Field**	**Field description**
Sample	Id	entity id
	Processed	specifies whether sample was already used in conversation detection process
	Timestamp	sample's recording time
	ConversationId	conversation id (updated after detection process)
	UserId	user's id that sample belongs to
States	Id	entity id
	SampleId	sample's id that states belongs to
	States	array containing each sample's frame state (voiced or unvoiced)
Energies	Id	entity id
	SampleId	sample's id that energies belongs to
	Energies	array containing each sample's frame energy
Conversation	Id	entity id
	StartTime	conversation start time
	EndTime	conversation end time
ConversationInfo	Id	entity id
	ConversationId	conversation id
	ParticipantId	conversation's participant id
	StartTime	the time when participant has joined the conversation
Participants	Id	entity id
	ConversationId	conversation id
	Timestamp	beginning of the conversation between two users
	User1Id	first participant id
	User2Id	second participant id
SpeakersTurns	Id	entity id
	ConversationId	conversation id
	Timestamp	beginning of the conversation between two users
	ParticipantsId	Participants entity id
	Sample1Id	id of the first sample used in speaker detection process
	Sample2Id	id of the second sample used in speaker detection process
	MI	mutual information value

Datastore entity is created in the database at a time when web service invokes `makePersistent()` method on a given entity class object (Figure A.2).

```
States st = new States(sampleId, new Text(states.toString()));
pm = PMF.get().getPersistenceManager();
try {
    pm.makePersistent(st);
} finally {
    pm.close();
}
```

Figure 5.7: Creating new entity in the database.

Data from the database are obtained using simple Java Data Objects queries. The sample query is presented on Figure 5.8.

```
private List<ConversationInfo> getConversationsInfo(long userId){
    javax.jdo.Query query = PMF.get().getPersistenceManager()
            .newQuery(ConversationInfo.class, "this.ParticipantId==id");
    query.setOrdering("this.StartTime desc");
    query.declareParameters("Long id");
    query.setRange(0, 10);
    List<ConversationInfo> conversationsFound
        = (List<ConversationInfo>) query.execute(userId);
    return conversationsFound;
}
```

Figure 5.8: Query for obtaining customized ConversationInfo entities data.

5.4.3 Servlets

As it was previously described in the Design chapter four different web services were implemented:

- time synchronization service (`GetTimeServlet`);
- sample data upload service (`CreateSampleDataServlet`);
- conversation and speaker detection service (`FindConversationsServlet`);
- statistics provider service (`GetDataServlet`).

All web services are implemented as a Java servlets (they are extensions of `HttpServlet` class). Request data needs to be send in JSON format via post request method. Response data are returned also in JSON format. Sample servlet class code with JSON parsing example is presented on Figure 5.9.

```
public class CreateSampleDataServlet extends HttpServlet {
    public void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws IOException {

        String content = getPostData(req);
        try {
            JSONObject allData = new JSONObject(content);
            String userId = allData.getString("userId");
            JSONArray samples = allData.getJSONArray("samples");

            [...]
        }
    }
    [...]
}
```

Figure 5.9: Sample web service class with JSON parsing example.

Time synchronization service is the simplest one. It does not take any request data and it only returns current server time. Smartphone application calculates difference between phone's local time and the server time so it could set sample's recording time regarding to the server time.

Sample data upload service takes samples' data array as a request (Figure 5.10). Then it creates appropriate database entities which store all required samples' information (time, states, energies). At the end information specifying whether sample was successfully stored in the database is returned.

Unlike other services which are invoked by the smartphone application, conversation and speaker detection service runs as a cron job. It means that it is automatically invoked by the server every specified amount of time. This service uses only already stored data in the database and does not need any user's interaction. It goes through all not yet processed samples data. Then it compares this data with other samples' data recorded at the same time by other devices. It performs mutual information algorithm to detect conversations and then it finds speakers turns in the discovered conversations. Finally it stores obtained data in the database and finishes its execution if there is no any unprocessed samples.

```
{
    "userId":355302048213764,
    "samples":
    [
        {
            "energies":[503.76126166000915,661.9734903993069,...],
            "states":[0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,...],
            "timestamp":1313920320000
        },
        {
            "energies":[4086.4310881789934,4277.28339284066,...],
            "states":[1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,...],
            "timestamp":1313920350000
        }
    ]
}
```

Figure 5.10: JSON request containing two samples' data.

```
{
    "isSuccess":true,
    "errorMessage":"",
    "cids":
    [
        {
            "id":74034,
            "startDate":1313670420173,
            "endDate":1313670450173,
            "participants":["355302048199989","355302048213764"],
            "durations":[0,66,166]
        },
        {
            "id":79034,
            "startDate":1313670000664,
            "endDate":1313670030664,
            "participants":["355302048213764","355302048199989"],
            "durations":[0,231,1]
        },
        [...]
    ]
}
```

Figure 5.11: JSON response containing two conversations' data.

The last service is responsible for providing the user with different statistics about already detected conversations in which he was participating. Different kinds of data are obtained by specifiying the type of the information in the JSON request. One of the sample data provided by the service are presented on Figure 5.11. It contains information about ten last user's conversations - start time, end time, participants and durations (how many frames were unvoiced, how many of them were voiced by the first participant and how many by second one).

Chapter 6

Evaluation

There are three main parts of the system which were tested to check how accurate the whole detection process is. The first part concerns voiced frames detection performed by the smartphone application. Two other parts are responsible for conversation and speaker detection performed on the server side. The calculations results verification process was performed in the following order:

- the calculations result is obtained;
- the manual detection is performed using the analyzer tool;
- both results are compared in order to check calculations correctness.

HMM learning process was performed using couple minutes long conversation between two people (including environmental noise) who have not participated in the evaluation process.

6.1 Voiced frames detection

Voiced frames detection process is performed by the smartphone application. To verify results correctness, the accuracy of frames detection in a silent and noisy environment was calculated. The manual detection was performed using waveform and spectrogram view of the analyzer tool (Figure 6.1). Voiced frames were marked based on a waveform and spectrogram appearance.

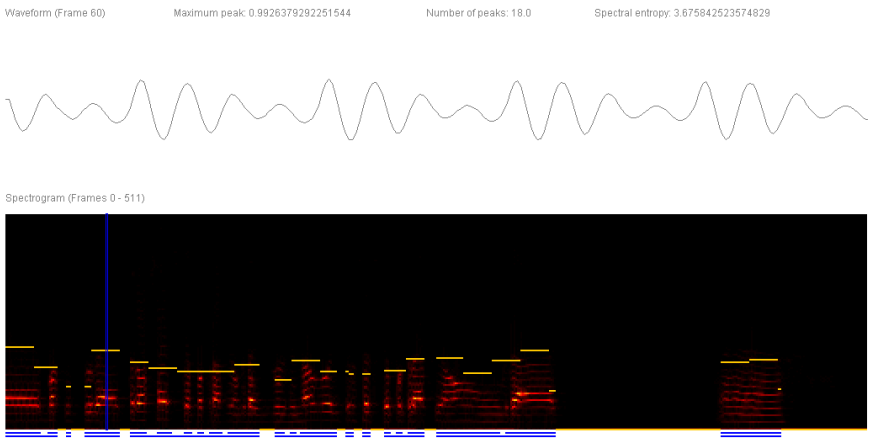

Figure 6.1: Waveform and spectrogram analysis.

6.1.1 Detection accuracy in a silent environment

This part presents voiced frames detection correctness between two people in a silent environment. Detailed results are presented in Table 6.1. The test was performed on ten 30 seconds long samples. Each sample consisted of 1874 frames.

Based on the test results it can be observed that there is only slight difference in the number of voiced frames detected manually and by the use of implemented algorithms. The number of incorrectly detected voiced frames is also small which results in high accuracy.

Table 6.1: Voiced frames detection correctness in silent environment

Voiced frames detected		Incorrect detection	Accuracy
Manual detection	**Calculations result**		
695	674	39	98%
728	659	76	96%
634	591	51	97%
539	491	50	97%
699	666	33	98%
746	701	46	98%
729	689	45	98%
846	796	56	97%
839	755	85	95%
1016	953	64	97%

6.1.2 Robustness to noise

This part presents voiced frames detection correctness between two people in noisy environment. Two first samples were recorded with low-level noise (quiet background music, approximately 40dB), the other two with loud background noise (urban noise [19], approximately 60dB).

Table 6.2: Voiced frames detection correctness in noisy environment

Voiced frames detected		Incorrect detection	Accuracy
Manual detection	**Calculations result**		
1084	1133	53	97%
1043	1073	40	98%
341	679	364	81%
400	565	303	84%

In contrast to silent environment test, smartphone application detected more voiced frames than when manual detection was performed. This is because some noisy frames were detected as voiced ones. Low-level noise has only slight effect on the detection result, the number of incorrectly detected voiced frames is similar to the one observed in the previous test. Loud noise discrupts detection process significantly but the accuracy is still quite high.

6.1.3 Conclusion

Figure 6.2 presents graphical representation of both tests results. Samples 9 and 10 were recorded with loud background noise.

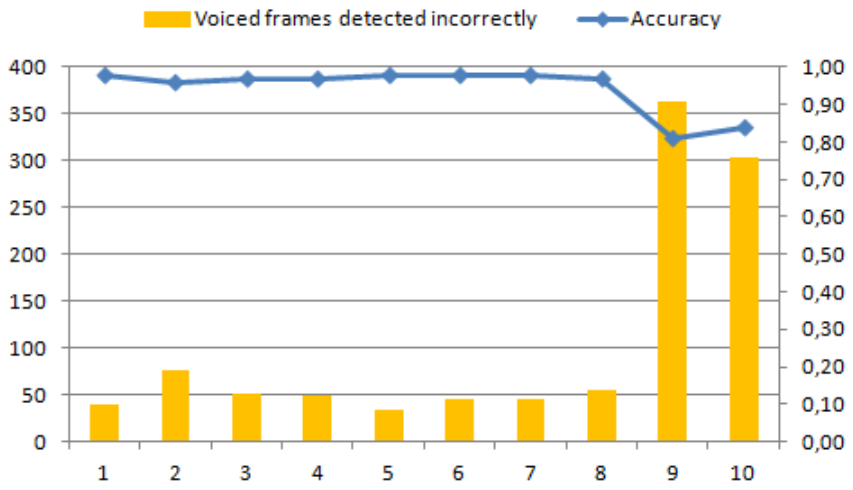

Figure 6.2: Voiced frames detection.

It can be concluded that the voiced frames detection in a silent environment is precise and the error ratio is very small. Low-level noise does not affect the calculations results, however loud noise has moderate impact on the detection process which results in greater amount of incorrectly detected voiced frames. Notwithstanding the accuracy in all tested cases is high which makes the voiced frames detection correctness satisfactory.

6.2 Conversation detection

Conversation detection process is performed by the proper server side web service. The accuracy of samples' allocation to different conversations is the most important issue to check in this case.

6.2.1 Samples' comparison

Figure 6.3 presents mutual information values of ten samples' pairs comparisons made using audio data recorded at the same time by two smartphones. It also contains information about differences in number of detected voiced frames. First seven samples' pairs were part of the same conversation whereas pairs 8,9 and 10 were recorded in different locations. Each sample consisted of 1874 frames.

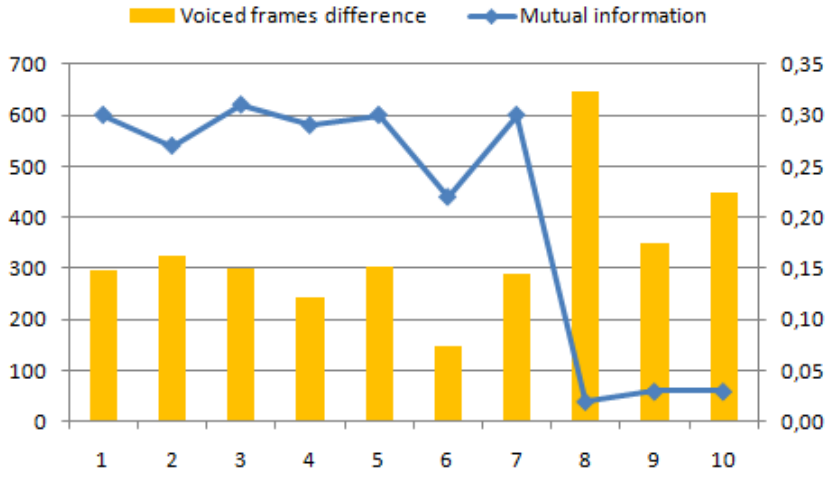

Figure 6.3: Samples comparison.

It can be observed that mutual information for samples' pairs of the same coversation have significantly higher values than for samples which were recorded in different locations. Differences are also visible in the detected voiced frames dissimilarities. It is higher for samples' pairs recorded in different places.

It can be concluded that the mutual information values clearly state whether samples' pair belongs to the same conversation.

6.3 Speakers segmantation

Speakers segmantation process is performed by the proper server side web service just after conversation detection. The accuracy of speakers turns detection is

examined in order to verifiy results correctness for this part of the process.

6.3.1 Segmentation accuracy

Speakers turns are detected using differences between energies log ratios of the signals obtained from two different sources. The segmentation results correctness was analyzed using audio samples view of the analyzer tool (Figure 6.4). Ten 30 seconds long samples' pairs were recorded for the test purposes, each one contains conversation between two speakers.

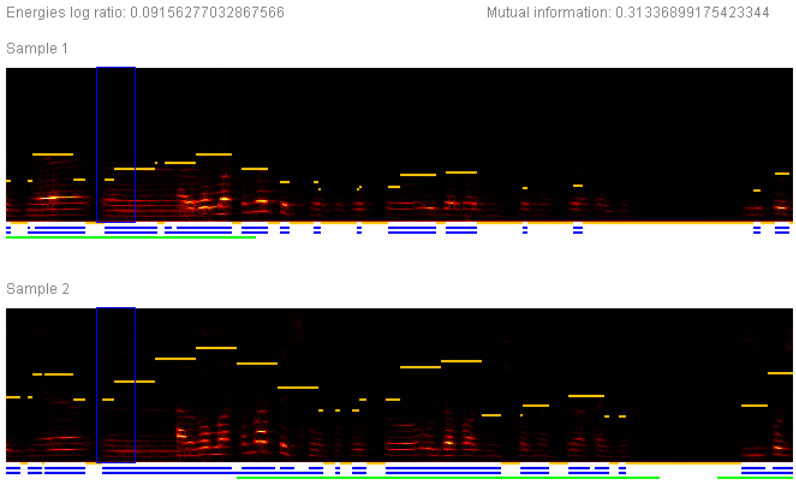

Figure 6.4: Speakers segmentation analysis.

Figure 6.5 presents calculations results for each conversation participant. The charts shows the difference between the number of speaking frames detected manually and by the use of implemented algorithms.

It can be observed that usually the differences are quite small, however for samples's pairs 3 and 6 they are more observable as they result in assigning significantly more speaking time to the Speaker B and taking part of Speaker A time. Such situation was caused by the very small distance between two participants (sitting side by side). The difference in energies log ratios were so small that based on them it could not be clearly stated who was speaking when.

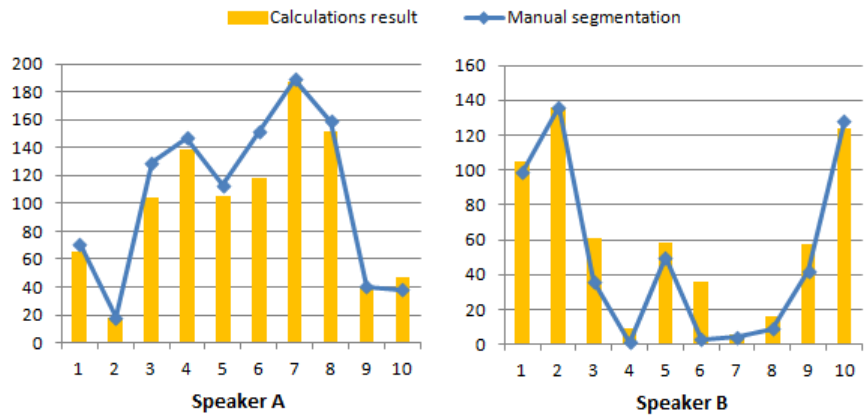

Figure 6.5: Speakers segmentation.

Table 6.3 presents the accuracy of all samples' pairs. As expected the accuracy is quite low for pairs 3 and 6 but it is high for other pairs.

Table 6.3: Speakers segmentation detection correctness

Samples' pair	**Accuracy**
1	93%
2	100%
3	70%
4	89%
5	90%
6	57%
7	98%
8	92%
9	82%
10	92%

It can be concluded that the speaker segmentation is highly dependent on the distance between the conversation's participants. If it is very small (approximately less than 1 meter) then it can result in incorrect speakers turns detection. In the other cases accuracy of spekers segmentation is high.

Chapter 7

Discussion

The aim of this thesis was to design and implement the prototype of the system for conversation and speaker detection based on the audio data recorded by the users' smartphones.

7.1 What has been accomplished

Firstly analyzing tool designed for the early tests purposes has been created. It allowed to verify in a convinient way correctness of different algorithms and parts of the system. It consisted of three main components which included waveform and spectrogram visualizations, detected features charts and samples comparison. It also allowed to create training samples and perform HMM learning process.

Afterwards the smartphone application prototype and server side web services have been designed and created. Smartphone application has been designed to run on Android operating system, whereas web services and database were hosted by the Google App Engine.

Smartphone prototype was responsible for continuous audio data recording and

processing collected data in order to detect voiced frames. It was also used as an interface for the user to display different conversation information.

Data processed by the smartphone application was transferred to the server using proper web service. Web services were responsible for data storage (in the provided database), processing collected samples' information in order to detect conversations and their participants and provide information about them back to the smartphone application.

At the end three main parts of the conversation and speaker detection process (voiced frames detection, conversation detection, speakers segmentation) has been tested and the results were presented.

7.2 Future work

From the start the system created for the project purposes was defined as a prototype. It can not be treated as a complete system since it misses many components which should be taken into account for the final version considerations. The complete system should be efficient, secure and most importantly it should provide the user many different information displayed in an interesting form.

7.2.1 Basic improvements

Since the prototype system can handle only conversations between two people, one of the main improvements would be to enable conversation detection between different number of participants. In the current version if there are more participants then each pair is considered as a participants of different conversation.

The prototype is running as a normal application and the calculations are not optimized enough to be able to run in the background without phone's performance drop. The next important improvement would be to optimize the algorithms to enable operating in the background so the voiced frames detection process could be perfomed continuously without interrupting the user.

Smartphone application has limited functionality which is useful only for the tests purposes. It allows to record training sample, start/stop sampling process and display simple information about already detected conversations. It

includes conversation date, duration, participants ids and percent of time each participant was speaking. The main feature which would make the smartphone application interesting for the user is conversation data presentation. Information should not be displayed as a text but it should be visualized in a clear and simple to understand way using different charts forms. Such visualizations examples were presented in section 4.6.

7.2.2 Social networking aspect

Collecting data about user's conversations gives opportunity to automatically create connections between different users. If two users are taking part in the same conversation then it means that they probably know each other. Based on this information connection between these users can be created. By checking how often users are speaking with each other it can be also defined how strong relation between the users is.

Interesting infomation could be concluded from analysing the time of day when the users are usually in conversation with each other. For example if the conversations occurs only during the working hours then it probably means that users are working or studying together. On the other hand if the conversation occurs only in the morning, in the evening and during the weekends then it is very likely that the users are living together (family). When the conversations occurs occasionally in the evening and during the weekends then it can mean that the users are colleagues or friends (frequent conversations). These are only examples on how could be the time of day information used.

The more accurate information can be acquired by combining processed audio data with other sensors information. There already exist applications which are collecting information from different phone's sensors and utilizing them (e.g. SenseBook application [12]). In the case of conversations data utilization, location information seems to be most useful. It could give information about the places where the user meets given person, where the conversations with given person usually take place or what is a chance to meet given person in a given place at given time.

Chapter 8

Conclusion

This thesis presented practical look on the conversation and speaker detection process. So far similar systems were implemented only using special wearable devices. The system presented in this thesis introduced the smartphone devices as a source for audio data collection. It made the process convenient to apply in a people's daily life.

The tests results indicate that there is strong potential for further development. Detection results are characterized by high accuracy, however there are some cases which need to be improved. There is still a lot of work to make the system useful for the ordinary user. This thesis gave strong foundations for it and they have proved that the modern smartphones are capable to handle complex computations associated with the voiced frames detection process.

Further system development can result in creation of very useful applications for those who like to be updated with different information about themselves or their friends. As it was presented in this thesis, conversation data can provide many useful information. Not only different summaries about user's conversations can be created. There are also many opportunities to explore on the social networking field.

Appendix A

Trained HMMs parameters

```
State 0
  Pi: 0.875     Aij: 0,982 0 0,017 0
  Opdf: Multi-variate Gaussian distribution --- Mean: [ 0,125 1,086 4,129 ]

State 1
  Pi: 0.0       Aij: 0,25 0,25 0,25 0,25
  Opdf: Multi-variate Gaussian distribution --- Mean: [ 0 0 0 ]

State 2
  Pi: 0.125     Aij: 0,016 0 0,901 0,083
  Opdf: Multi-variate Gaussian distribution --- Mean: [ 0,836 10,758 4,098 ]

State 3
  Pi: 0.0       Aij: 0,002 0 0,109 0,89
  Opdf: Multi-variate Gaussian distribution --- Mean: [ 0,268 2,763 4,214 ]
```

Figure A.1: HMM parameters for voiced frames detection, *Pi* - probability of being inital, *Aij* - transition probability (from state i to state j), *Opdf* - observation probability function.

```
State 0
  Pi: 0.4     Aij: 0,798 0,069 0,134 0
  Opdf: Gaussian distribution --- Mean: 0 Variance 0

State 1
  Pi: 0.2     Aij: 0,041 0,924 0,035 0
  Opdf: Gaussian distribution --- Mean: 0,618 Variance 1,166

State 2
  Pi: 0.4      Aij: 0,068 0,023 0,909 0
  Opdf: Gaussian distribution --- Mean: -0,828 Variance 1,829

State 3
  Pi: 0.0      Aij: 0,25 0,25 0,25 0,25
  Opdf: Gaussian distribution --- Mean: 0 Variance 1
```

Figure A.2: HMM parameters for speakers segmentation, *Pi* - probability of being inital, *Aij* - transition probability (from state i to state j), *Opdf* - observation probability function.

Appendix B

Waveform samples

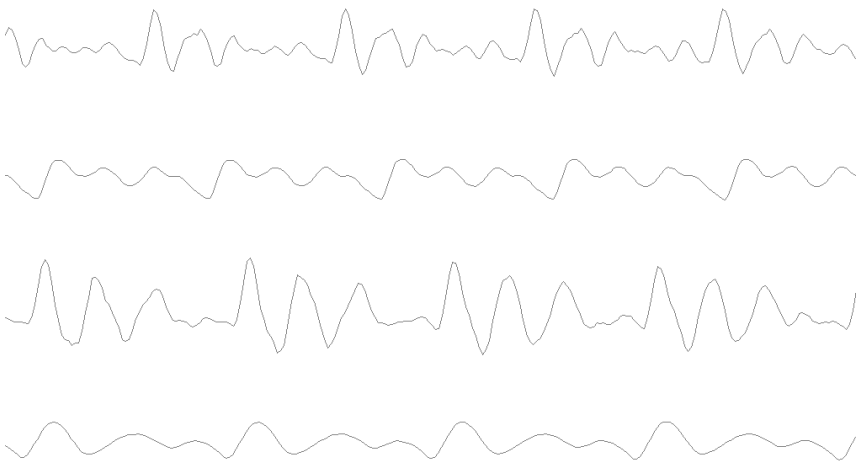

Figure B.1: Waveforms of a voiced signals. Repetitive parts are clearly visible.

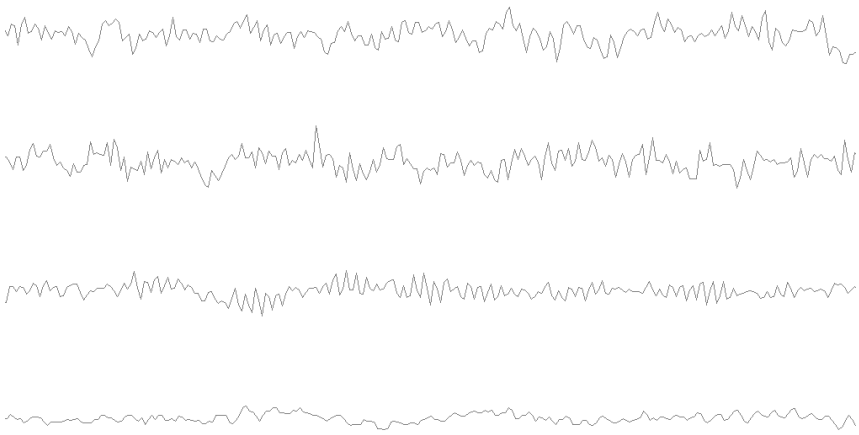

Figure B.2: Waveforms of a noisy signals. Repetitive parts are not present.

Appendix C

Spectrogram samples

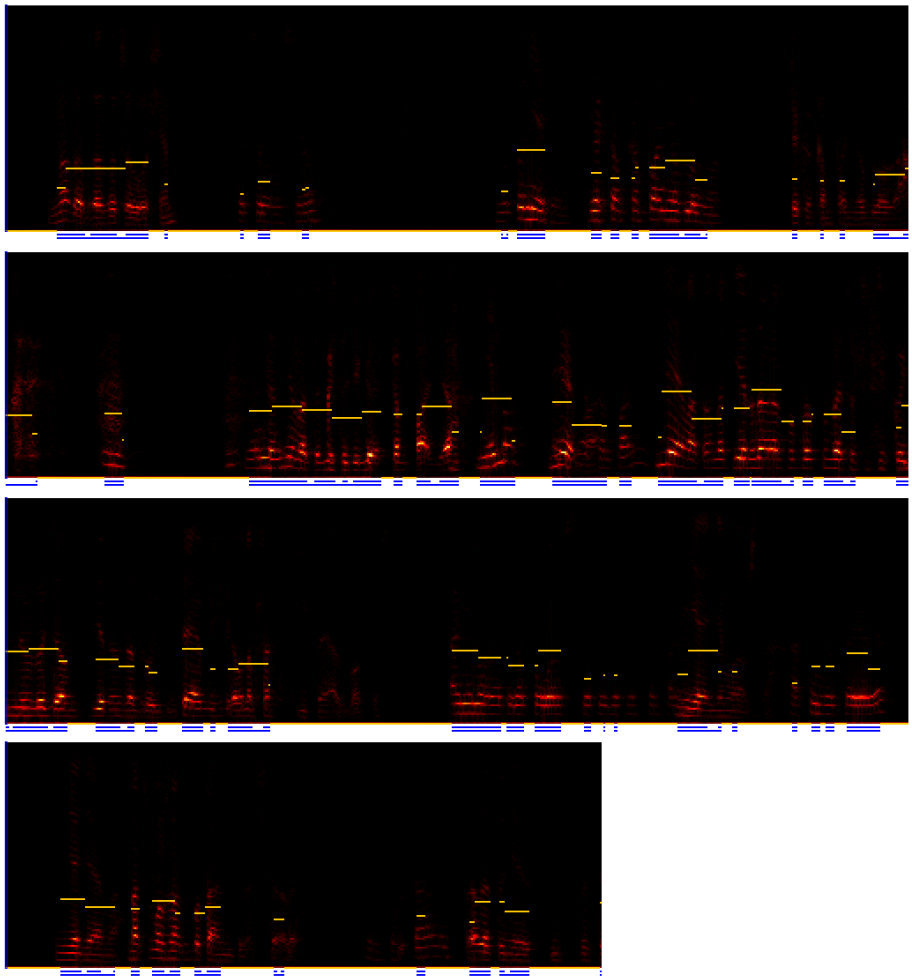

Figure C.1: Spectrogram of 30 seconds long sample. Yellow line - energy magnitude, bottom blue line - speech regions, top blue line - voiced frames.

Appendix D

Features dependancies

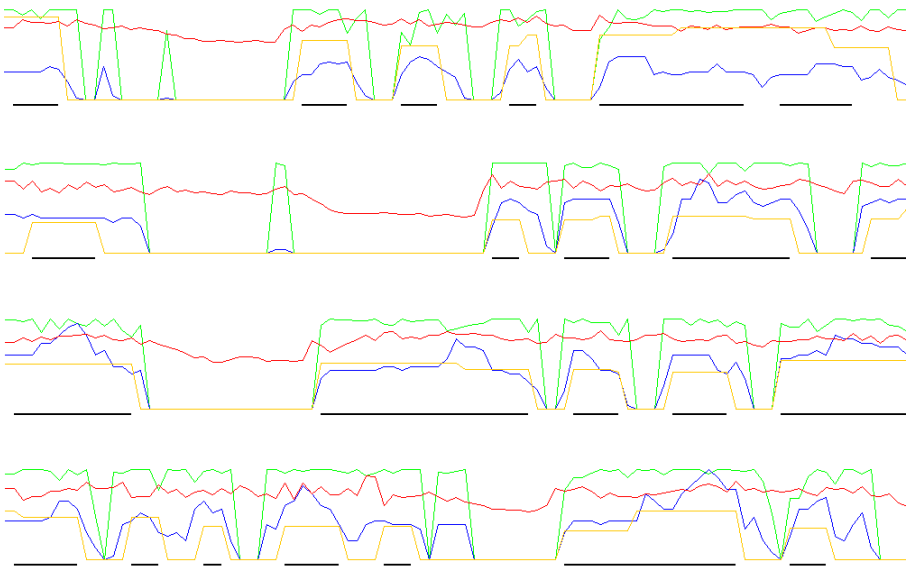

Figure D.1: Each line represents one feature magnitudes. Yellow line - energy (calculated only for the speech frames), blue line - number of peaks, green line - maximum peak, red line - spectral entropy, black line - voiced frames.

Bibliography

[1] Sumit Basu. Conversational scene analysis, 2002.

[2] Sumit Basu. A linked-HMM model for robust voicing and speech detection, 2003.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st ed. 2006. corr. 2nd printing edition, October 2007.

[4] Oliver Brdiczka, Jérôme Maisonnasse, and Patrick Reignier. Automatic detection of interaction groups. In *2005 International Conference on Multimodal interaction, ICMI '05, Trento It*, pages 32–36, 2005.

[5] Tim Carstens. What is a Hidden Markov Model? http://intoverflow.wordpress.com/2008/05/27/what-is-a-hidden-markov-model/, access: May 2011.

[6] S. R. Corman and C. R. Scott. A synchronous digital signal processing method for detecting face-to-face organizational communication behavior. In *Social Networks, vol. 16*, pages 163–179, 1994.

[7] Jean-Marc François. Jahmm - an implementation of Hidden Markov Models in Java (v.0.6.2). http://jahmm.googlecode.com, access: April 2011.

[8] Google. Google App Engine. http://code.google.com/appengine/, access: June 2011.

[9] Google. Android developers. http://developer.android.com, access: May 2011.

[10] HTC. HTC Desire specification. http://www.htc.com/us/products/desire-cellularsouth, access: April 2011.

[11] IBM. Many Eyes. http://www-958.ibm.com/, access: July 2011.

[12] Jakob Hommelhoff Jensen and Regin Larsen. *Collecting, Analysing and Visualising Context Data for Enriching Relationships - Using a Mobile Social Networking Application.* Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2007.

[13] Alexander Kraskov, Harald Stögbauer, Ralph G. Andrzejak, and Peter Grassberger. Hierarchical clustering based on mutual information. Cornell University Library, 2003.

[14] Geoff Martin. Introduction to sound recording - autocorrelation. http://www.tonmeister.ca/main/textbook/node715.html, access: May 2011.

[15] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[16] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, Englewood Cliffs, NJ, 3rd edition, 2010.

[17] Liang sheng Huung and Chung ho Yung. A novel approach to robust speech endpoint detection in car environments. In *Proc. ICASSP*, 2000.

[18] Jonathon Shlens. Notes on Kullback-Leibler Divergence and Likelihood Theory. http://www.snl.salk.edu/~shlens/kl.pdf, access: June 2011.

[19] The Noise of New York. Online video clip. http://www.youtube.com/watch?v=CslD_6L4oTs, access: September 2011.

[20] Danny Wyatt and Jeff Bilmes. Towards the automated social analysis of situated speech data, 2008.

[21] Danny Wyatt, Jeff Bilmes, Tanzeem Choudhury, and Henry Kautz. A privacy-sensitive approach to modeling multi-person conversations. In *Proc. of IJCAI-07*, 2007.

[22] Danny Wyatt, Tanzeem Choudhury, and Jeff Bilmes. Conversation detection and speaker segmentation in privacy-sensitive situated speech data. In *Proc. of Interspeech*, 2007.