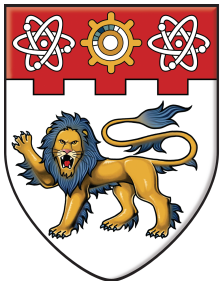
Project Number: A1119-151



NANYANG TECHNOLOGICAL UNIVERSITY

SOCIOFEEDBACK BY GOOGLE GLASS

Submitted by: Zhong Sailin

Matriculation Number: U1220669F

Supervisor: Associate Professor Justin Dauwels

School of Electrical Electronic Engineering

A final year project report presented to the Nanyang Technological University in partial fulfilment of the requirement of the degree of Bachelor of Engineering

2016

TABLE OF CONTENTS

ABSTRACT

This project concentrates on the development of an Android application for Sociofeedback. The available real-time Sociofeedback system and applications requires a series of devices to analyse the audio input and perform classification. In this project, all the signal processing and classification are aimed to be accomplished on the mobile devices themselves. Two platforms are discussed in this report — Google Glass and Android mobile phone.

The report illustrates the development process with reference to the system development life cycle (SDLC). The functionalities of the applications have been varying through the weekly meetings. This application initially takes monologues as audio input to identify the speech mannerisms and provide feedback in order to enhance users' presentation skills. Thinking of distributing such application to broader audience, we start to accommodate the application for two-person. The Sociofeedback app can further fit into workplace such as call centres and police offices. Users are able to select their preferred low-level features to be monitored, including volume, pitch, speaking percentage, and MFCC so far. The selected low-level features will be shown graphically on the app. Twelves low-level features are derived from the volume, pitch and MFCC of the speech from the two users to perform classification.

Google Glass is one of the most cutting-edge wirable devices in the market. It is light, unsophisticated and portable. Although the processing capability of Google Glass is not satisfying in this project, the future generation of such smart glass would be an appreciable platform for this app. We use Android phone as an alternative device to showcase the application. The iterative design procedure will be elaborated in this report.

ACKNOWLEDGEMENT

Firstly, I would like to express my deep sense gratitude to Associate Professor Justin Dauwels, my FYP supervisor whom I had worked closely with, for his aspiring guidance in the course of this project. The weekly FYP group meeting conducted by Professor Justin motivated us to work on our individual FYP projects in a consistent and constructive way. He directed us to explore a varieties of potential implementation for our project while sustaining feasibility.

Even though the initial goal of the project was to create an Android app which provides sociofeedback to individual user on Google Glass, Professor Justin led me to think forward to adjust the app for analysing dialogs and design an interface for Android phone.

Secondly, current PhD students Yasir Tahir and Tomasz Maszczyk had also assisted and gave valuable guidance for the investigation of the app. Yasir explained the current Sociofeedback system and illustrated low-level features. Tomasz introduced the classifier to me and propose several models for adjusting the app in order to analyse dialogs.

Lastly, I would like to thank to Dhayalan Priya, my project partner, a master student who wrote the code for extracting volume, pitch and MFCC and gave me feedback on the interface.

I. INTRODUCTION

Nowadays, people have to deal with plenty of interviews and presentations in a professional manner. Interpersonal communication skills are highly valued especially for people who are taking customer service or sales roles. To realise effective communication, people need to minimise potential misunderstanding and deal with barriers during the conversation. Phenomenal communicators understand their audience. They will look for feedback from their audience to see how the message is perceived and make every effort to counteract any misunderstanding or confusion timely [15].

Speaking mannerisms are a vital factor of human behaviour [2]. A dynamic speaker emphasises his or her main topic and maintains the attention of audience by varying his or her vocal behaviour [1]. If we consider the communication process as an encoding and decoding procedure shown in Figure 1, we can design an application to enhance the productiveness of this procedure by providing sociofeedback to the speaker intelligently. Here, the communication Channels includes face-to-face conversations and telephone calls. Considering the recording capability of the android devices, this project focus on simulating the communication over telephone calls. The objective of this project is to develop an Android application for providing Google Glass users or Android phone users with instantaneous sociofeedback. The app is primarily designed for delivering a real-time feedback on the Google Glass screen panel, but the platform is finally replaced by Android phone because of the poor

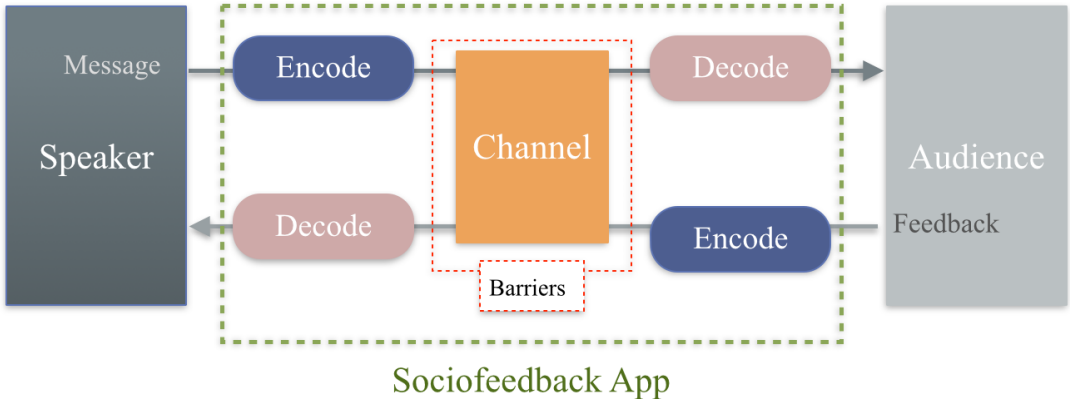


Figure 1. The communication process

performance of the Google Glass's processor. The Glass version only can display speaking mannerisms for monologues.

Various professors, researchers and students has cooperated to create an interactive system for analysing social behaviour and providing real-time feedback on NAO Robot. Last year, Vuzix M100 Smart Glass and Google Glass was introduced to display the feedback message. It was a breakpoint that remained us the Sociofeedback system can be a handy tool if it is installed in our mobile devices. The previous system operated on MATLAB. The final year project A1119-151 highlights the procedure of rebuilding Sociofeedback platform on the Google Glass and Android mobile phone.

1.1 OBJECTIVES

The main objectives of this projects include:

- Work with the project partner to retrieve low-level features of a speech such as volume, pitch, MFCC and speech rate in Android
- Design a user-friendly and consistent interface for the Google Glass and Android mobile phone.
- Adjust the application for dialogs. Display the low-level features of two audio inputs to simulate the usage of this app in conversations.
- Implement the Support Vector Machine (SVM) in Android for future classification

1.2 THE DIFFERENCE BETWEEN THIS PROJECT AND EXISTENT SOCIOFEEDBACK SYSTEM

As I mentioned earlier, a few generations of Sociofeedback system has been developed. According to a survey for the latest system, 68% of the 50 users are satisfied with the system and 70% of them demands to use the Sociofeedback app in

the future. It has been confirmed by the users that the Sociofeedback system can provide reliable and helpful feedback for their speech. However, the system has to rely on various devices to operate. The audio data was recorded using H4N recorder and was sent to a running MATLAB of central server [4]. The recording system was controlled through a wireless connection on a laptop. The system running on the laptop will process the audio signal and define the sociofeedback. The feedback message will be sent remotely to the Vuzix Glass or Google Glass.

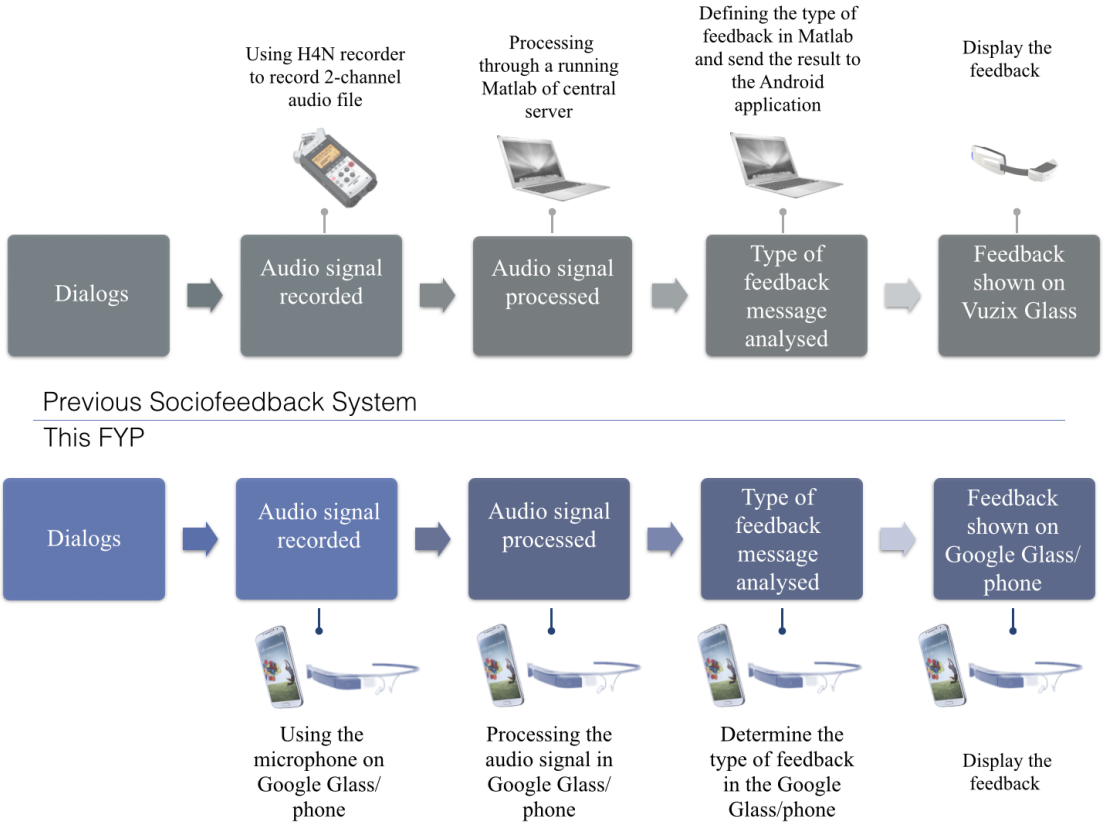


Figure 2. The previous Sociofeedback system versus Sociofeedback app

Concerning that users may not possess all the devices mentioned above and may not have the knowledge of installing the system in MATLAB, we decide to rebuild the system on Android. For this final year project, the dialogs will be recorded by two Android devices separately. Then the audio signal will be processed and be classified on these two Android devices in real-time. Users can view the feedback on the screen simultaneously. The unique feature of this final year project is enabling the whole

Sociofeedback analysis on Android devices. It allows users to monitor their social behaviour in vocal communication anytime and anywhere.

1.3 PROJECT FLOW

The project follows the interactive and incremental development model. During the development of the Sociofeedback app, three iterations of the software development cycle has been went through. In each circle, the app undergoes five main stages - Modelling, Analysis and Design, Test and Deployment, and Evaluation [16].

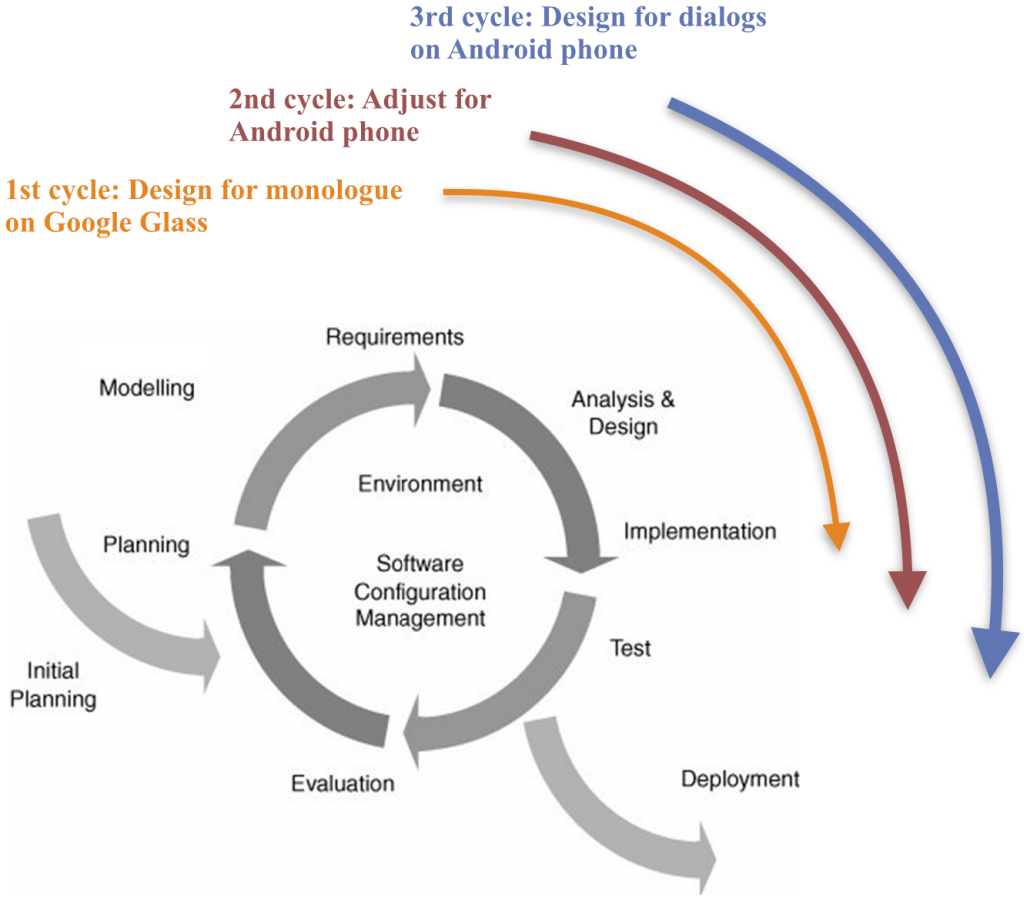


Figure 3. Iterative software development lifecycle for the Sociofeedback app

In the first life cycle, the app is designed for the Google Glass. The initial plan is to extract low-level features including volume, pitch and speaking rate of a real-time monologue, and to use the threshold obtained from the previous Sociofeedback system to categorise the feedback. We analyse the methods of obtaining low-level

features and design the prototype of the interface. After implementation, it turns out that the CPU of the Google Glass is not powerful enough to provide stable feedback. Besides the microphone does not capture the speech in the same way as H4N recorder. We may need to re-conduct experiments to define the threshold for the feedback. Hence, in the testing stage, we only deploy the volume detector and pitch detector on the Glass version.

In the second life cycle, the app is migrated to Android mobile phone. The mobile version of the app runs smoothly. Coming to design phase, to maintain consistency between different platforms, it maintains the same architectural structure and style to the Glass version, but its layout is adjusted for the phone. A classification function is added during this cycle. However, our current data are collected from dialogs. To show basic feedback such as agreement, dominance, interest, we have to obtain the low-level features from two users.

After evaluating the feasibility of detecting dialogs using the Sociofeedback app, we decide to simulate the conversation by transmit the low-level features of one user to another one's in the third life cycle. The layout of the app is re-arranged for two speakers. Correspondent code for feed the real-time data to the classifier is added so that we can train the system to provide feedback and link it to the phone call in the future.

In the subsequent chapters of this report, investigation during the project, interface design procedure, and the algorithm behind each implementation will be illustrated.

II. LITERATURE REVIEW

In this project, digital signal processing and machine learning algorithms has been touched. The glass version inherits the basic structure from the previous project. This chapter will cover all the background information of the functions for the Sociofeedback app.

2.1 AUDIO SIGNAL PROCESSING

Sound, caused by vibrations, is a pressure wave that goes through a medium [17]. In this project, the medium refers to air. We can detect sound by measuring the pressure level at a location [18]. When we record the sound by our mobile devices or a computer, it is digitised automatically, that is, the sound transfers from an analog signal to a digital signal. The digitised sound data can be stored in .WAV(Microsoft waveform) files. If the data is stored in uncompressed form and use linear samples, it can be specified as a .PCM(Pulse Code Modulation) file which is a subclass of the .WAV file format [19]. In PCM, the sound wave is stored in an array of samples with a pre-defined sampling rate. It is the rate of taking and playback the audio data [20]. The sample rate 44,100 Hz is commonly used in Audio Compact Disc(CD). It can cover the 20 ~ 20,000 Hz range of human hearing and is ideal for professional recording. The sample rate 8,000 Hz is adequate for human speech and the Sociofeedback system was trained on 98 brief dialogs stored in .WAV format with this sampling rate. Hence, in this project, the audio dispatchers' sampling rates are set to be 8,000 Hz.

Sample Rate	Maximum Frequency to be reproduced	Usage
8,000 Hz	3,600 Hz	Telephone transmission; adequate for human speech
11,025 Hz	5,000 Hz	Lower-quality PCM
22,050 Hz	10,000 Hz	Used in early 20th for audio digitising
44,100 Hz	20,000 Hz	Suitable for Audio CD
96,000 Hz	43,600 Hz	Suitable for DVD-Audio

Table 1. Common sample rate and their usage

When the sound is captured, its amplitude is stored in the audio file. The precision and noise level of the audio is determined by bit depth [17], the number of bits of information in each sample. For example, if each sample is recorded at 16-bit resolution, it can be assigned with 2^{16} (i.e. 65,536) unique values. 8-bit audio commonly used and 16-bit is considered as relatively high resolution. The higher the resolution, the better the quality of the digital audio, but also the higher processing capability required for hardware. The audio files used in this project is 16-bit audio.

As I mentioned above, amplitude is directly stored in the input data stream. If we aims to display the volume of the speech, we can direct read the data from the audio file. For conventional usage, the volume can be expressed in dB. The relationship between decibels and amplitude is:

$$\text{Volume (in dB)} = 20 \log_{10}(\text{amplitude}).$$

The current digital signal processing techniques allow us to convert the sound to the frequency domain whereby we can identify the amplitude versus frequency over the entire time span of the input data stream. To convert the audio signal from time domain to frequency domain, we can apply a fast Fourier transform (FFT) to the input file. In short, FFT is a relatively low time-consuming method to compute discrete Fourier transform (DFT). DFT is the methodology for converting a finite sequence of samples extended by time steps into the coefficients of its complex sinusoids extended by frequency steps. As such, we can further acquire the pitch.

Pitch is a perceptual property of sounds that orders the sounds on a frequency-related scale [21]. It is correlated with musical melodies. Pitch is a major attribute of tones in auditory system [22]. Pitch is subjective to each person, in the sense that it gives a person a feeling to define the sound as higher or lower. It is associated with frequencies. By comparing sound with pure tones, we can produce a rough measure of pitch. There are several approach for pitch detection, including time-domain approaches, frequency-domain approaches and spectral approaches. TarsosDSP, the

library that we are using for this project, implements a number of pitch detection algorithms: YIN, the Mcleod Pitch Method and a "Dynamic Wavelet Algorithm Pitch Tracking" algorithm [23].

2.2 THE SOCIOFEEDBACK SYSTEM

In the paper "Real-time comprehensive sociometric for two-person dialogs (2013)", a group of 5 researchers and professors proposed a real-time Sociofeedback system. It assesses speech mannerisms by low-level speech metrics involving volume, pitch and rate of speech [6]. The speech mannerisms includes Low Voice, High Voice, Low Speech Rate, High Speech Rate, Screaming, and Low Response Time. [6] Those low-level speech metrics are also used to compute the level of interest, agreement, and dominance. The system is trained with 98 labeled two-person dialogs in English. The accuracy is 80-90% which is verified by leave-one-out cross-validation [6]. The system overview is shown below.

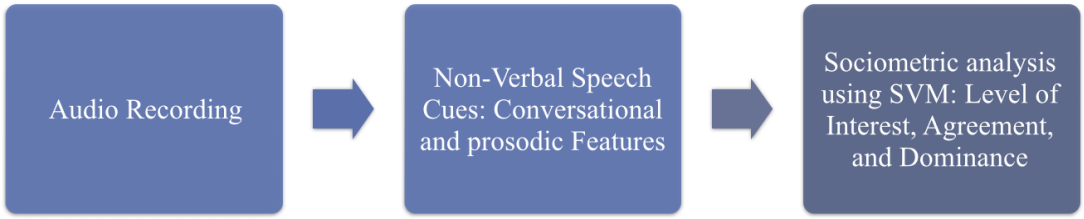


Figure 4. Sociofeedback system overview

The recorded dialogs covers students project discussion, and social and political views. In the conversations, problematic situations such as interruption, boredom, agreement, soft speech and fast pace are broadly revealed [6]. The duration of the recording clips is 2-3 minutes and is recorded in 2-channel audio .WAV files.

In the second stage of the system overview above, the team employed a hidden Markov model (HMM) to perform speech detection and compute conversational cues such as speaking percentage, interruptions and mutual silence percentage after the signals have been segmented. For prosodic cues, the team extracted amplitude, larynx

frequency, formants, and mea-frequency cexstral coefficients (MFCCs) every 10 ms [6].

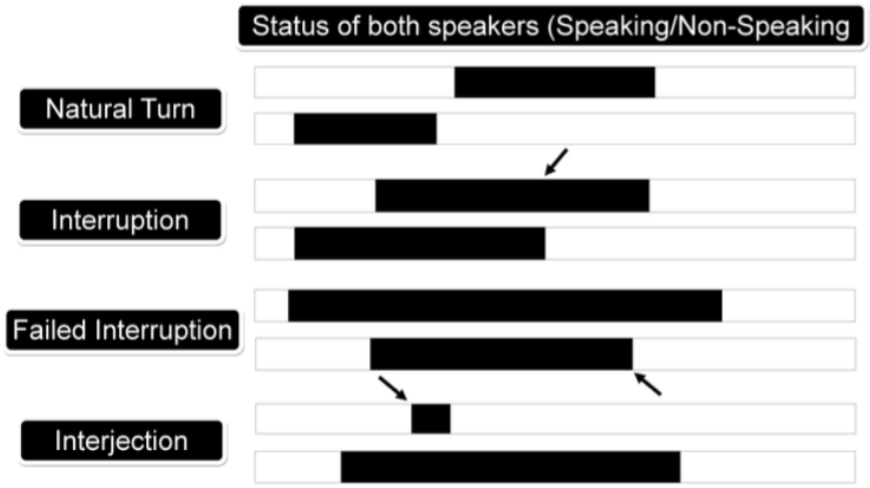


Figure 5. Conversational cues

In the third stage of the Sociofeedback system, the team made use of the manual annotation from 14 judges. The dialogs were distributed to the judges and they were asked to completed a questionnaire to assess the speech mannerism. Then the team constructed the following table to explain the relationship between speech mannerism assessed by the judges and low-level features detected by the system. The team further used the annotations to define the thresholds for volume, speech rate, screaming and low response time such that the false alarm rate is within 5%.

Speech Mannerism	Corresponding Features	Detection Rate (%)
Low Voice	Mean Volume	88
High Voice	Mean Volume	90
Low Speech Rate	Speech Rate	76
High Speech Rate	Speech Rate	78
Screaming	Mean Volume, Mean MFCCs	92
Low Response Time	Response Time	96

Table 2. Speech mannerism and low-level features

In the last stage, the team calculated Information Gain (IG) and Correlation-based Feature Selection (CFS) to identify the applicable low-level speech cues to classify social behaviour [24, 25].

Category	Feature from the speech cues	Info Gain	CFS Subset Merit
Interest	Speaking percentage	1.130	0.718
	Turn Duration	0.564	0.540
	Mutual Silence	0.468	0.392
	Volume	0.420	0.312
	Response Time	0.392	0.192
Agreement	Interruptions	0.888	0.605
	Total Overlap	0.508	0.400
	Mutual Silence	0.227	0.191
	Larynx Frequency	0.212	0.166
	Volume	0.167	0.218
Dominance	Speaking percentage difference	1.079	0.783
	Turns Difference	0.695	0.540

Table 3. IG and CFS for classification

Then the team applied four kinds of multi-class classifiers on the system: k-nearest neighbour (kNN), Artificial Neural Network (ANN), Support Vector Machine (SVM), and Naive Bayes [6], among which SVM produced the best accuracy. Hence, I decide to apply SVM to the Sociofeedback app in this project.

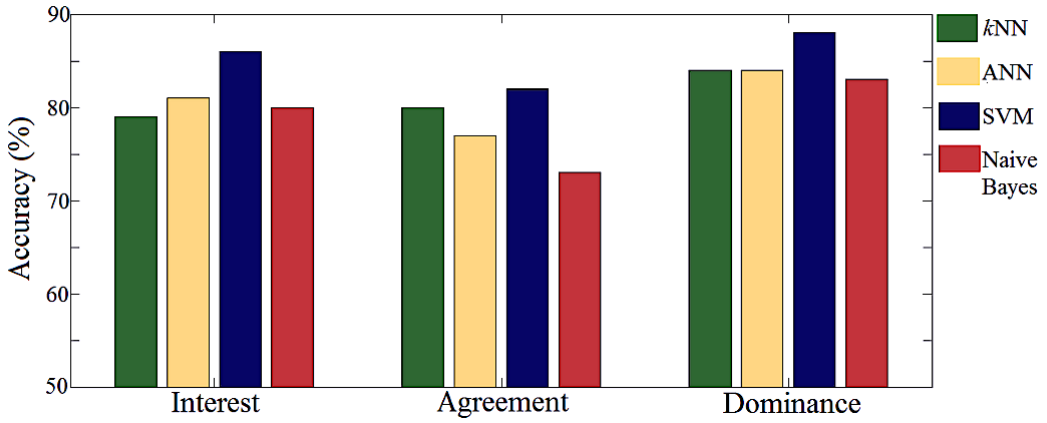


Figure 6. Classification performance for the two classifiers

2.3 MACHINE LEARNING ALGORITHM

The machine learning algorithm that we are using for data classification is Support Vector Machines (SVMs). During classification, data are separated into training and testing sets. Each instance in the training set contains one class label and several attributes[26]. SVMs can produce a model to predict the class labels for the testing data by analysing their attributes. SVMs are supervised learning models which classify the examples into categories linearly. In addition, SVMs can be extended to non-linear classifications by mapping data to high dimensional space. This is called the kernel trick. Some basic kernels includes:

- linear: $K(x_i, x_j) = x^T_i x_j$.
- polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$.
- radial basis function (RBF): $K(x_i, x_j) = \exp(- \gamma ||x_i - x_j||^2), \gamma > 0$.
- sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$.

Here, γ, r, and d are kernel parameters [26].

According to the requirement of SVM, each data instance in the training set needs to be represented as a vector of real numbers. For example, if we have low-level features {volume = 100 dB, pitch = 210 Hz, speech rate = 190 syllables/min } and it is classified as aggressive, it should be represented as {1 1:100, 2: 210, 3: 190} where aggressive is labeled as class 1. To avoid the domination of attributes in greater numeric ranges and to decrease numerical difficulties during the calculation, we may need to linearly scale each attribute to the range [0, 1] [26],

Generally, RBF kernel is the first choice to be experimented on. It is applicable to the condition when class labels and attributes are nonlinearly distributed. It also has less hyper parameters so that it requires less complexity [26]. This kernel is suitable for classifying relatively small number of features. It meets the classification requirement of the Sociofeedback system.

SVMs aim to find the solution for the following optimisation problem [26]:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{\cdot}\xi_i$$
$$\text{subject to} \quad y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1$$

where (x_i, y_i) is instance-label pairs and $\Phi(x_i)$ is the non-linear instance. $C > 0$ is the penalty parameter of the error term [26]. Hence, RBF kernel has two parameters to be adjusted: C and γ. Cross-validation, the approach employed in the Sociofeedback system, could be a good way to determine (C, γ).

2.4 PAST YEAR INTERFACE FOR GOOGLE GLASS AND USER STUDY

The past year Sociofeedback app is a display panel which retrieves the social behaviour from the Sociofeedback system and provide graphic feedback to the user. The connection between Google Glass and the server follows the "Three-way Handshake" model.

The app has six panels in total: Welcome, Plain Text, Image, Combination, Audio, and About. When users enter the Welcome panel, they can select the feedback mode via the menu by tapping on the touchpad. The order of the panels in the menu is shown below.

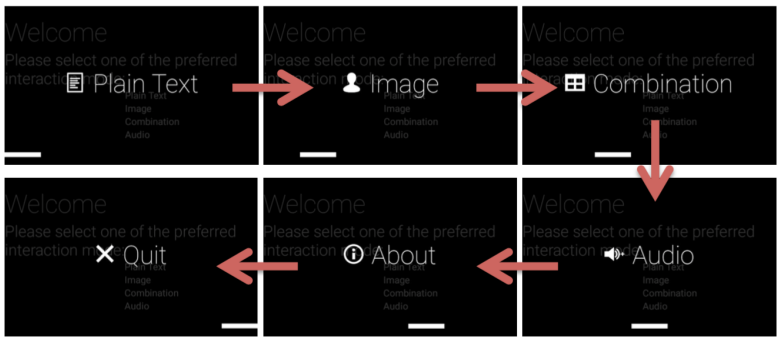


Figure 7. Available panels from the menu

The plain text mode only shows the feedback message in phrases and the low-level features are shown both in percentage and in progress bar. The Image mode shows the

feedback message in images and the low-level features are shown on the right hand side of the image. The combination mode is the consolidation of plain text mode and image mode. The audio mode provides vocal feedback. The feedback messages are recorded as .MP3 files and the Google Glass will notify the user by reading the feedback message for users.

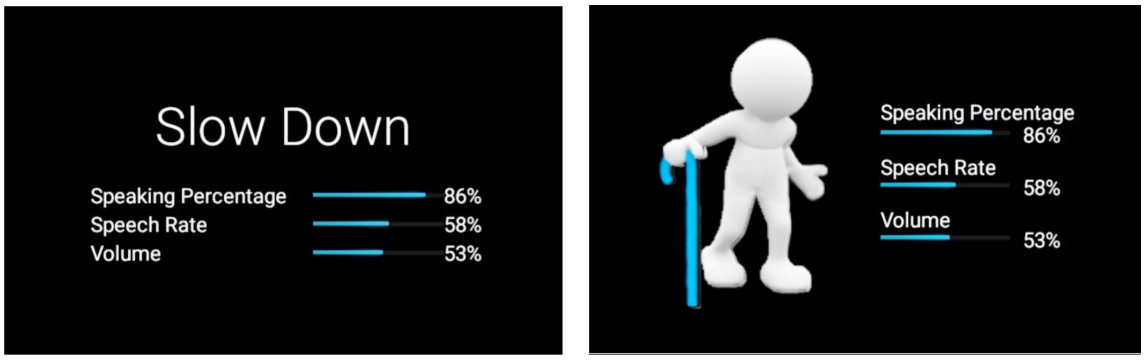


Figure 8. Plain Text mode and Image mode

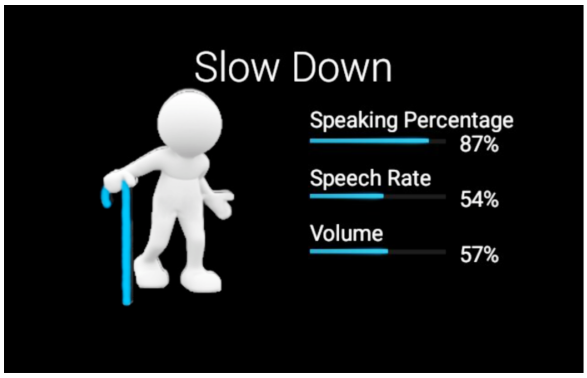


Figure 9. Combination mode

The previous FYP student tended to analyse user preference by provide four modes above. This is highly valuable for future enhancement of the Sociofeedback app. Some of the feedback from the users are cooperated in the new design this year.

According to the user study last year. Users are keen to receive sociofeedback regarding their pace and volume during conversations. Hence, we begin with extract ing volume and speaking rate from the speech this year.

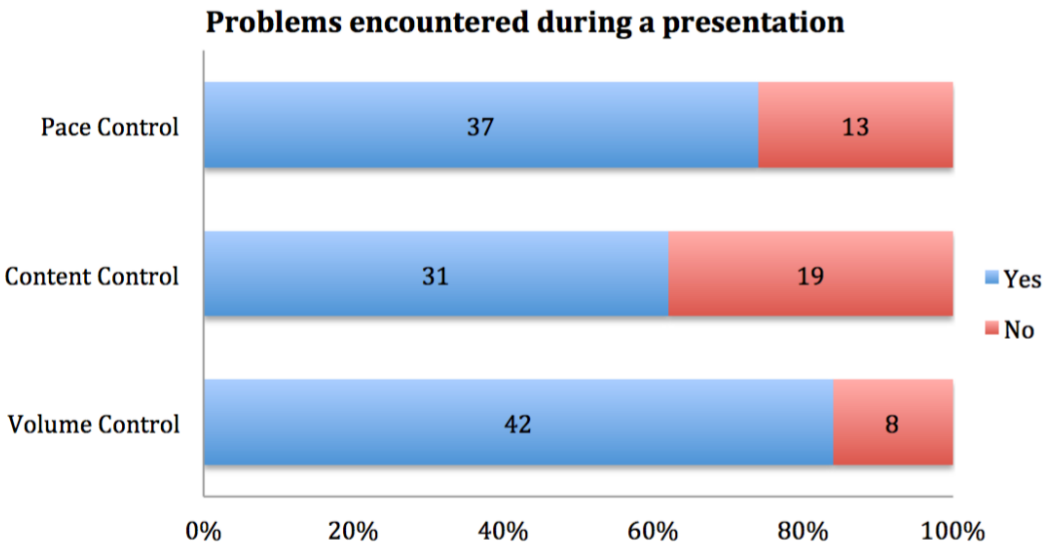


Figure 10. Problem statement for the Sociofeedback app

In the user study, the participants also indicate that they prefer image to data value. They also mentioned that the data values shown on the progress bar do not provide as much information as the feedback messages. To preserve the quantitive feedback, we retain the progress bar [4], but we decide to vary the colour of the progress bar based on the feedback message.

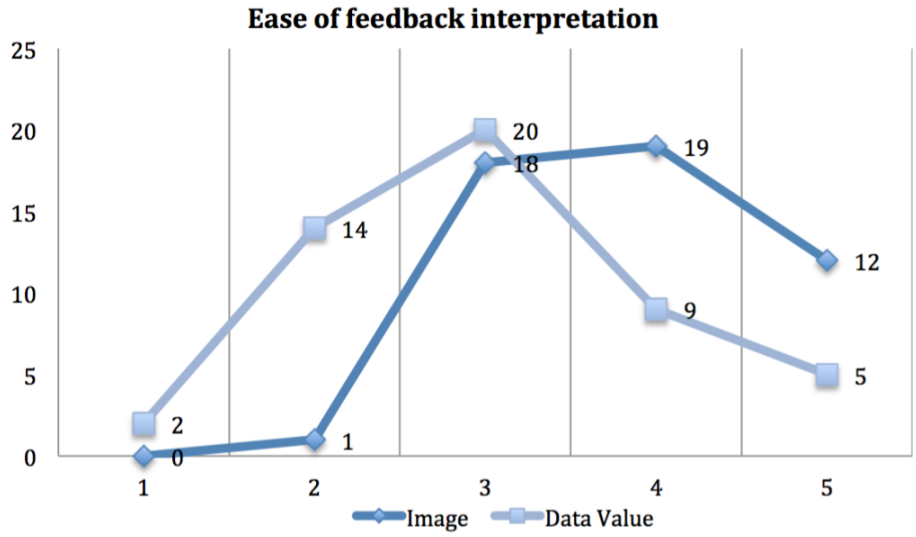


Figure 11. Preference of feedback messages

III. PROJECT INVESTIGATION

Google Glass is only compatible with android based mobile applications. Hence this project is developed using Android Studio. The previous FYP student implemented a preliminary version of this application on the Google Glass. Last year, all the real-time data has to be received from a server. The current application will detect the speech mannerisms by itself. Volume, pitch and speaking rate were aimed to be obtained. In this Section, I will elaborate the volume and speaking rate detector that I have implemented. In the final version of the Sociofeedback application, those algorithms are replaced by the functions in Tarsos library because of the accuracy and stability.

3.1 INSTALLATION OF ANDROID STUDIO AND IMPORTING PREVIOUS YEAR PROJECT

Android Studio is the preferred tool for enabling Google services or developing Android based applications. The latest Android Studio requires Max OS X 10.8.5 (or higher version) and JDK 7 configured [13]. Current laptop meets the system requirements.

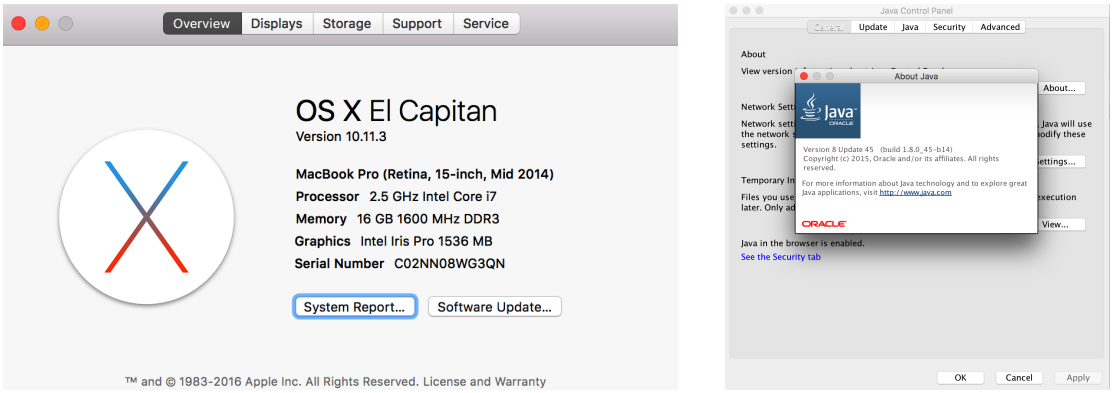


Figure 12. System requirements

API 19 or SDK version Android 4.4.2 is needed to be selected to support the GDK (Glass Development Kit) configuration. The GDK is an SDK adds-on that enables robust Glassware to be launched on the Glass without going through Google's API server. In another word, the application can perform the tasks without connecting to a web server and more customised functions can be built. The GDK uniquely supports

live cards, Immersions, Voice input and output, Network and Bluetooth connectivity, Graphics and animations, sensors and camera[5]. The voice input function allows users to choose their target panel by voice comment. With the GDK installed in Android Studio, offline processing and dynamic user interaction are enabled on the Google Glass.

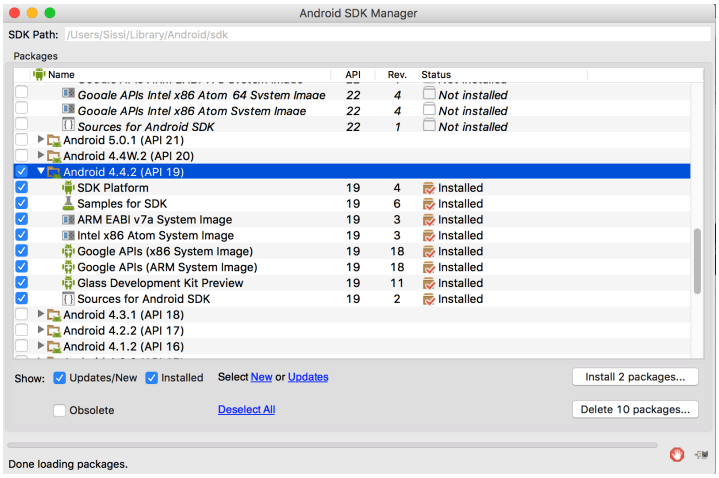


Figure 13. Android SDK version

To create an application for glass, the "Glass" platform has to be installed in Android Studio. Only Android Studio 1.2.1.1 or Android Studio 1.5.1 supports this configuration. Hence, the Android Studio was downgraded to 1.2.1.1 and was upgraded after the new version 1.5.1 is released.

Figure 14. Glass option in Android Studio

After setting up Android Studio, a new project is created and the project folder is replaced by the previous FYP project source code. Then the project is synced with the

Gradle Files. The layout is shown as expected. Android Screen Monitor is downloaded to mirror the Google Glass Screen on a laptop and debugging mode is enabled on the Google Glass.

Figure 15. Welcome page of the previous FYP project

3.2 IMPLEMENTING RELEVANT PARAMETERS ON GLASS

3.2.1 RECORDING AUDIO

The user speech may need to be recorded for user study. Android provides two methods to capture audio. The first one is MediaRecoder, a high-level API which automatically records into a compressed file. Only output format, encoding method and output file name are required to be set. The second method is to use AudioRecord. AudioRecord and AudioTrack are low-level APIs which save the raw audio data. Other than the parameters required by MediaRecoder, AudioRecord asks the developer to define buffer size and channel. The raw data is usually saved in a PCM file. The recorded audio file can be exported by executing an Android Debug Bridge

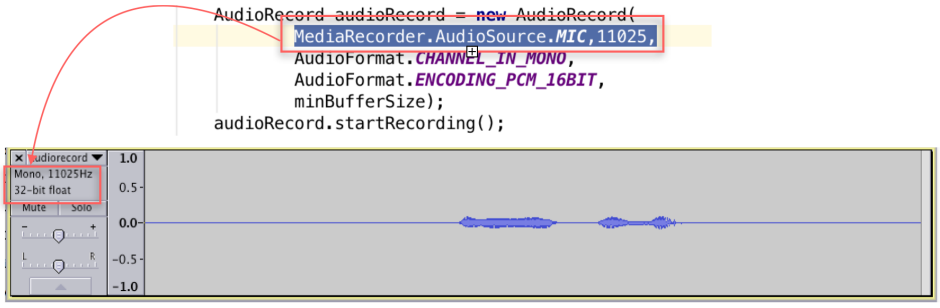


Figure 16. The waveform in time domain (shown in Audacity)

(adb) command in terminal. Audacity is an open source software for editing sounds. It is used to display the raw data in time domain. The sampling rate is 11025 Hz which results in a passable voice quality for speech[3].

These two recorder are tested on the Google glass. Both of them can record and play speeches. In order to test the capacity of recording dialogs, I place my phone 30 cm away from the glass and talk with Siri. It turns out that MediaRecorder is able to record the dialog but the volume difference is not distinguishable when I play the sound. The AudioRecord embedded in function from TarsosDSP library is not able to detect such dialogs as the second speaker is relatively far from the microphone.

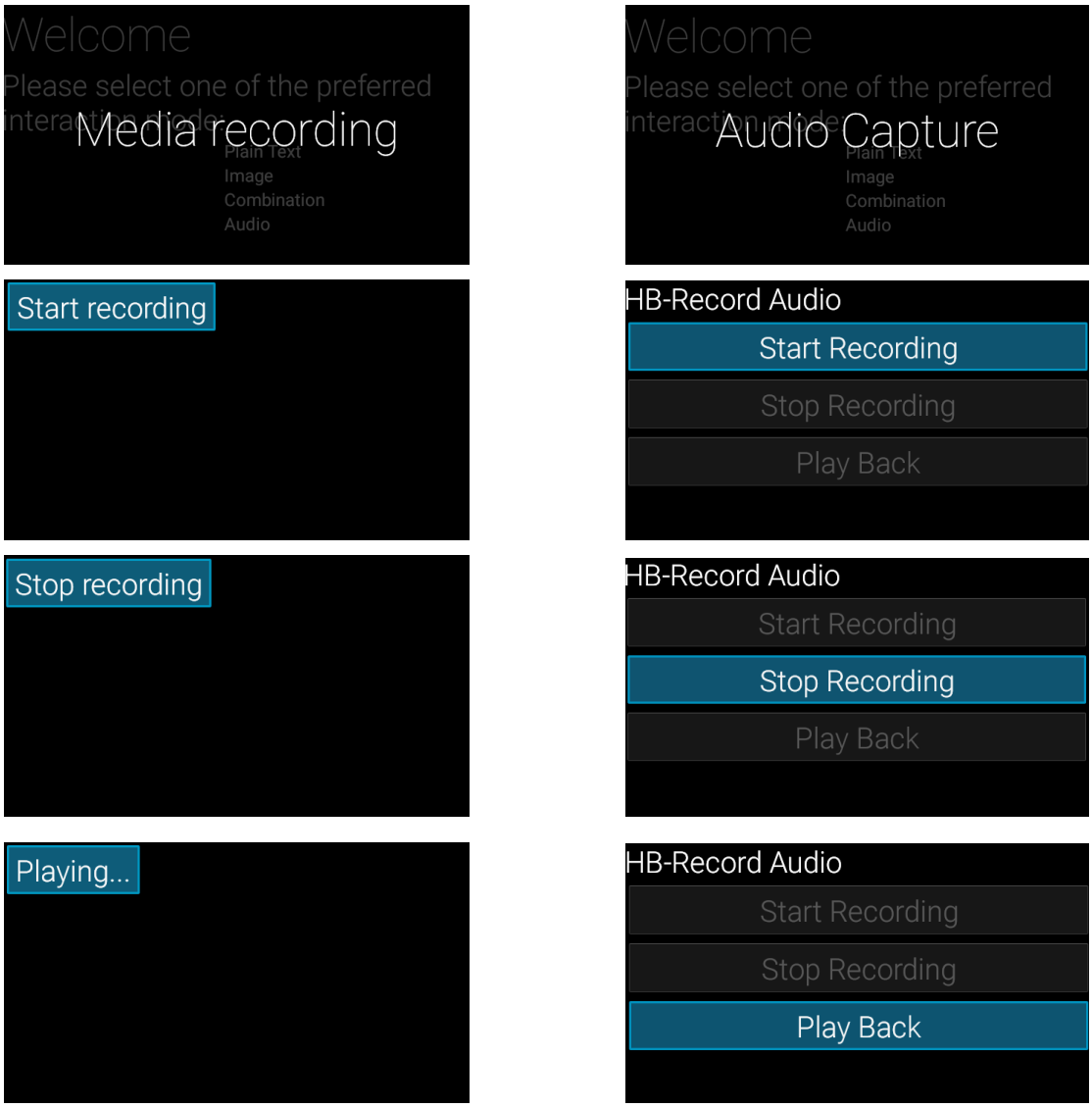


Figure 17. MediaRecorder and MediaPlayer (left) versus AudioRecord and AudioTrack (right)

3.2.2 DETECTING VOLUME

Volume is the amplitude of the waveform in time domain[7]. Again, we have two approaches to capture the volume — MediaRecorder and AudioRecord. If the speech is recorded using MediaRecoder, we can get the amplitude by calling getMaxAmplitude() every 100 milliseconds. Considering that low-level features such as pitch have to be detected simultaneously and the pitch requires to apply FFT on the raw data, we choose to use AudioRecord. The amplitude samples can be stored in buffers according to the properties of AudioRecord object[8]. The average value of the data stored in a buffer is the current voice amplitude [9].

```
/**
 * Functionality that gets the sound level out of the sample
 */
private void readAudioBuffer() {

    try {
        short[] buffer = new short[bufferSize];

        int bufferReadResult = 1;

        if (audio != null) {
            // Sense the voice...
            /*Reads audio data(current voice amplitude) from the audio hardware
            for recording into a direct buffer. return int */
            bufferReadResult = audio.read(buffer, 0, bufferSize);
            double sumLevel = 0;
            for (int i = 0; i < bufferReadResult; i++) {
                sumLevel += buffer[i];
            }
            lastLevel = Math.abs((sumLevel / bufferReadResult));    //take average
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

The minimum buffer size for Google glass is 3072. The volume to be displayed is the average of the data stored in 3072 bytes.

```
double l;
l = Math.abs((lastLevel-4.5)/11.5*100);
if(lastLevel > 3.6 && lastLevel <= 5.4){
    imageView.setImageResource(R.drawable.participate);
    textView7.setText("Please Participate");
    //textView7.setText(String.valueOf(lastLevel));

}else
if(lastLevel > 5.4 && lastLevel <= 8){
    imageView.setImageResource(R.drawable.speaklouder);
    textView7.setText("Speak Louder");
}else
if(lastLevel > 8 && lastLevel <= 13){
    imageView.setImageResource(R.drawable.right);
    textView7.setText("Right Volume");
}
if(lastLevel > 13){
    imageView.setImageResource(R.drawable.tooloud);
    textView7.setText("Too Loud");
}
volume.setProgress((int) l);
textViewV.setText((int) l + "%");
```

Scale the average amplitude

Figure 18. Code for capturing amplitude

This function is tested in a quiet environment. The lower boundary of the level of loudness is around 4 and the upper boundary is around 17. Base on these values, I integrate this function into the Sociofeedback app. There are four feedback messaged — 'Please Participate', 'Speak Louder', 'Right Volume' and 'Too Loud'. The threshold is calculated by (Current Loudness Level-Lower Boundary) / (Upper Boundary-Lower Boundary)*100%. The progress bar reflects the level of loudness which is calculated using the data fetched from the previous buffer.

	Please Participate	Speak Louder	Right Volume	Too Loud
Volume	0 ~ 8%	9% ~ 31%	32% ~ 77%	78% ~ 100%
Feedback	Please Participate	Speak Louder	Right Volume	Too Loud

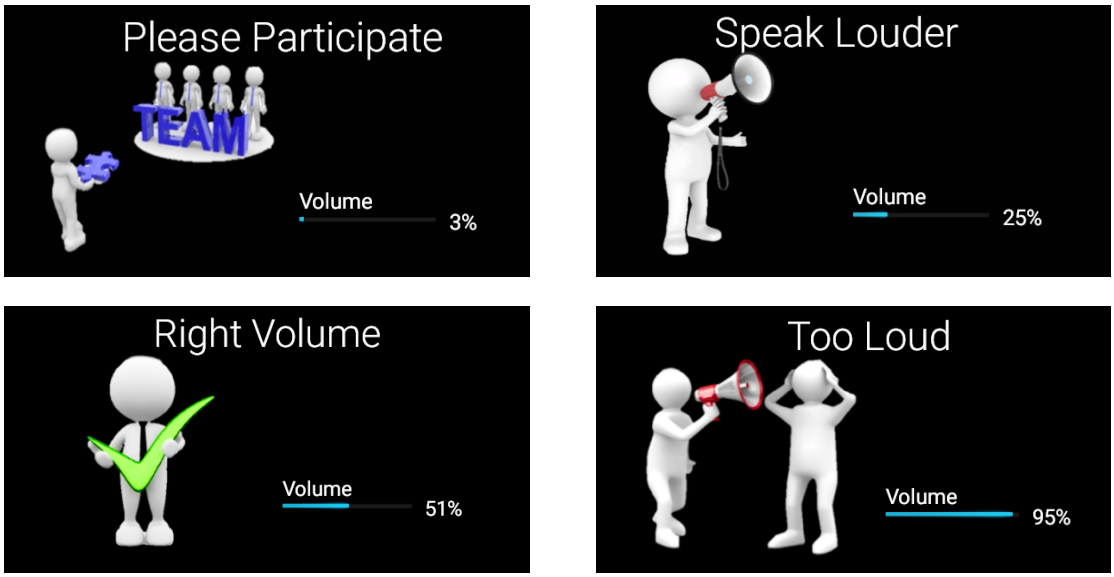


Figure 19. The thresholds and corresponding feedback

Concerning unity and power consumption, this part of code is replaced by the getFloatBuffer() function from TarsosDSP library and my project partner calculated the volume in dB based on the byte array with the audio data acquired from this function in the final version of the app,

3.2.3 MONITORING SPEAKING RATE

Google Glass has speech recognition capability. It waits for users to speak and returns the transcribed text after they finish [12]. The glass has to connected to the internet

and the internet can be set up through MyGlass site. The glass can be connected to the internet by scanning the QR code.

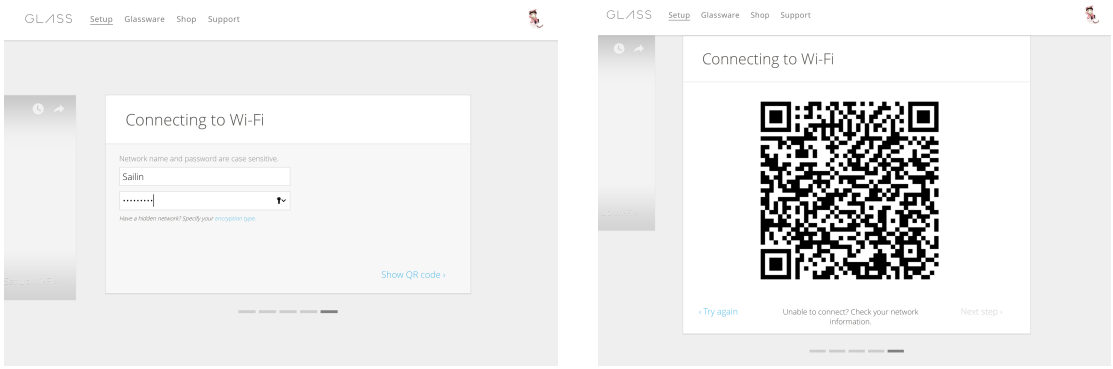


Figure 20. Wifi connection settings on MyGlass site

A chronometer is created to record the time that the user spend on a sentence. The number of words will be counted from the transcribed text after user press the off button. The speaking rate is the total number of words divided by the time.

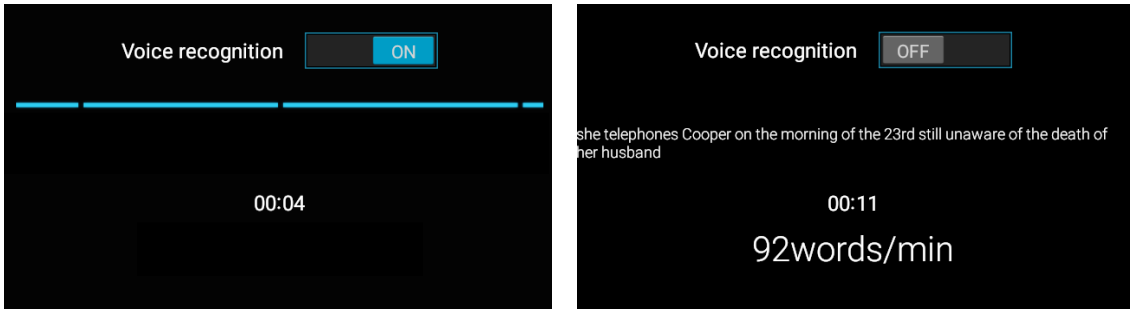


Figure 21. Voice recognition progress and result

The voice recognition system returns the transcribed text in a String. To obtain the total number of words within the elapsed time, I need to trim the sentence. This is done by searching the space between the words until there are no more characters. Take the sentence in the screenshot as an example. The Google voice recognition system will produce the sentence, “She telephones Cooper on the morning of the 23rd, still unaware of the death of her husband” with its speech-to-text function. The duration is calculated based on the difference between the current time and the start time of chronometer. This voice recognition system does not add punctuation mark in

the sentence. After searching for spaces, we can find 17 words in this sentence. The speaking rate is calculated as (17words / 11 sec) * 60 sec/min, which is 92 words/min.

```
aSwitch.setOnCheckedChangeListener((buttonView, isChecked) → {
        if (isChecked) {
            speakingRate.setVisibility(View.VISIBLE);
            speakingRate.setIndeterminate(true);
            speech.startListening(recognizerIntent);
            duration.setBase(SystemClock.elapsedRealtime());
            duration.start();
              if ((SystemClock.elapsedRealtime() - duration.getBase()) == 10000){
                  Log.i(LOG_TAG, "10s");
              }

        } else {
            speakingRate.setIndeterminate(false);
            speakingRate.setVisibility(View.INVISIBLE);
            speech.stopListening();
            elapsedMillis = SystemClock.elapsedRealtime() - duration.getBase();
            duration.stop();
            Log.i(LOG_TAG, "duration:" + Long.toString(elapsedMillis));
        }
});
```

Calculate the total time elapsed

```
@Override
public void onResults(Bundle results) {
    Log.i(LOG_TAG, "onResults");
    ArrayList<String> matches = results
            .getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);
    String text = "";
    for (String result : matches)
        text += result + "\n";

    //count no. of words
    String trimmed = text.trim();
    noOfWords = trimmed.isEmpty() ? 0 : trimmed.split("\\s+").length;
    Log.i(LOG_TAG, "No. of words: " + Integer.toString(noOfWords));
    Log.i(LOG_TAG, "text: " + text);
    errMsg.setText("");
    if (elapsedMillis != 0){
        speakingRate.setProgress(((int) (60 * noOfWords / (elapsedMillis / 1000)))/300);
        returnedSpeakingRate.setText(Integer.toString((int) (60 * noOfWords / (elapsedMillis / 1000))) + "wpm");
    }
```

Calculate the total number of words

Display the speaking rate in words per minute

Figure 22. Code for calculating speaking rate

After that, I consolidate the volume detector and speaking rate monitor together on the app. When user turns the switch on, the app will start recognise the speech of the user and reflect his or her volume in real-time. After the user turns off the switch, the speaking rate will appear.

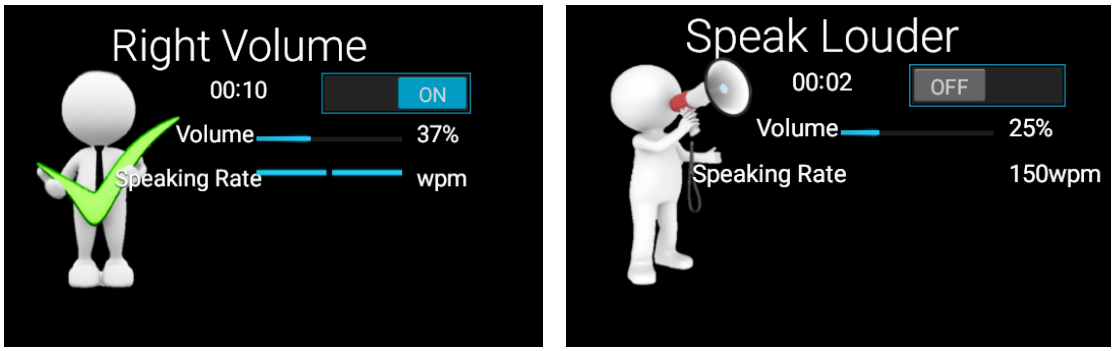


Figure 23. Sociofeedback app volume and speaking rate detection enabled

Due to the instability of Google speech recognition system, this part is abandoned in the final version. It has been observed that the recogniser will crash if the switch is toggled too rapidly. Besides, the recogniser needs pause for around 2 seconds before it starts again. A client-side error will occur if the sentence is too long or the recogniser is occupied. Hence, it is hard to define the frequency for updating speaking rate. This app requires to reflect low-level features in real-time, but the speaking rate obtained by using the speech recogniser does not meet our expectation.

IV INTERFACE DESIGN

The interface of Sociofeedback has been deliberately designed. In consideration of the two platforms we are using - Google Glass and Android mobile phone, I created two interactive interface. The interfaces are stylised according to the screen size, colour and conventional layout of the two platforms and they share common functions and elements to sustain consistency. This app can provide intuitive information and become visually appealing with the interface.

4.1 INTERFACE FOR GOOGLE GLASS

4.1.1 APPLICATION REQUIREMENTS AND SPECIFICATIONS

The app requires 1 to 2 seconds to warm up. In case of losing user's interest, a welcome page is created to grab user's attention and guide user to understand the fundamental structure of the app. Google Glass does not have a touchable screen, so the app needs to react according to user's gesture on touchpad. Other than displaying low-level features and feedback, the app also ensure that user can define the parameter they would like to monitor. Pitch can be displayed in waveform while volume is supposed to be shown in progress bar.

The feedback should be displayed both graphically and in plaintext to minimise the time for understanding the message. Thus the user can react instantaneously. The app is initially considered to analyse user's performance during the total time elapsed. The details of the application and functional requirements are listed in table.

No.	Application requirement and specifications
R1	Put up a welcome page when the application is loading
R2	Provide guidance to the app
R3	Allow user to personalise the parameters to be displayed
R4	Show the low-level features of the user's speech
R5	Give feedback to the user
R6	Allow to be controlled through touchpad

Table 4. Application requirements and specifications

No.	Correspondent Application requirement	Functional requirement and specifications
F1	R1	The welcome page should be aesthetically designed and guides the user to enter the other panels
F2	R2	The user guide panel should provide basic information of the app, such as how to use the app and who owns the app
F3	R3	The settings panel records user's preference and the feedback panel will restores the settings accordingly
F4	R4	Display the low-level features graphically in feedback panel; volume is supposed to be shown in progress bar while pitch should be shown in waveforms
F5	R5	Show sociofeedback message in feedback panel
F6	R6	Enable the touchpad so that user can swipe forward, back, up and down to access their desired panels

Table 5. Functional requirements and specifications

4.1.2 SITE MAP AND STORY BOARD

Based on the above application and functionality requirements, the Sociofeedback app have 4 panels — Welcome panel, Settings panel, Help panel and Feedback panel. User can return to the Welcome panel, change their settings or check the user guide by applying gestures on Google Glass. The relationship between the 5 panels are shown in the site map below.

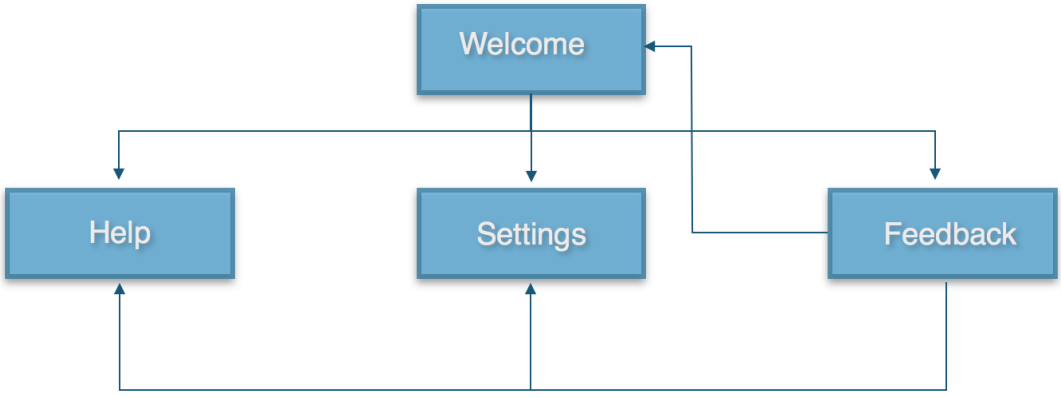


Figure 24. Site map of the Sociofeedback app on Google Glass

Users will enter the Welcome panel when they open the app. They can modify their settings or check the help information by swiping back or forward on the Welcome panel. They can proceed to feedback panel from Welcome panel by tapping on the touchpad. Users are allowed to access the settings panel and Help panel by swiping back and forward on feedback panel as well. The interaction is shown in the story board below.

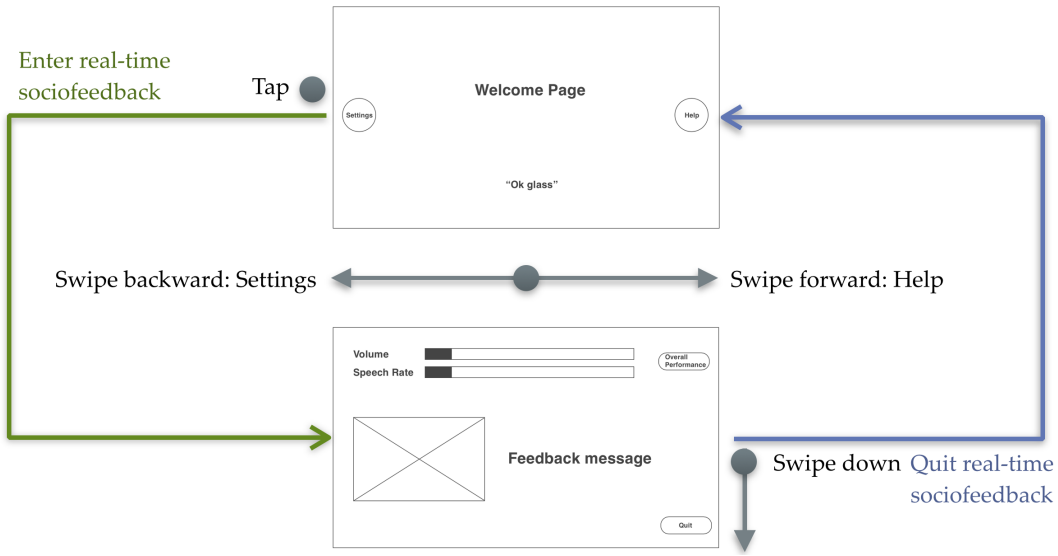


Figure 25. Storyboard of the Sociofeedback app on Google Glass

4.1.3 PROTOTYPE

The screen size of Google glass is relatively small. Looking at Google Glass's screen is equivalent to watching a 25'' TV screen at around 2.4 meters away. It cannot present abundant content to users. The text size has to be relatively larger with respect to images [14]. Thus, it is advisable to create a neat layout for the Sociofeedback app on the Google glass. The feedback messages and labels for the buttons needs to be highly contextual and in be short with large font.

The background of the app is coloured in black and the welcome page is decorated with a picture of galaxy with text in white to create a powerful and intelligible impression to the users. Conventional icons with descriptive labels are placed at the two side of the Welcome panel in order to prompt the user to swipe back or forward to enter the settings or Help panel. Users can proceed to the feedback panel by either

tapping on the touchpad or saying ‘Ok glass’ to trigger the voice comment.

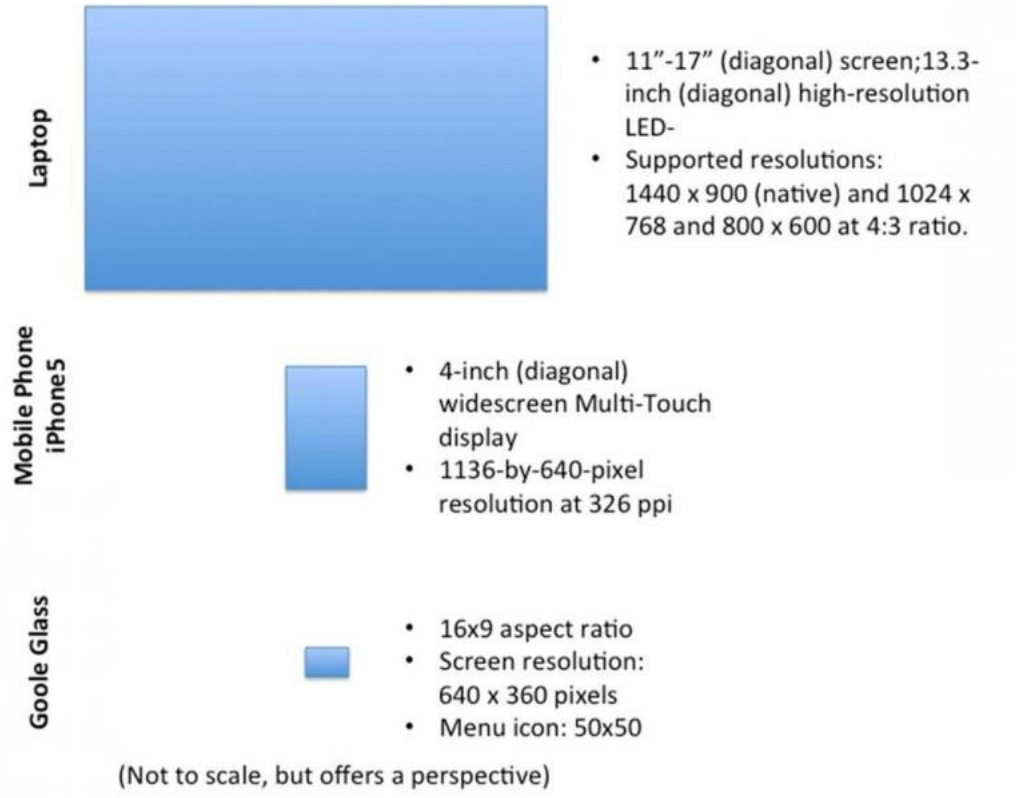


Figure 26. Screen size of Google glass

In the settings panel, users can turn off some low-level features. Once the low-level feature is turned off, it will not be shown in the feedback panel, for instance, the progress bar of volume will disappear if it is turned of in the settings panel. The Help panel will indicate all the available gestures for this app and give a brief introduction of this project and this app.

Once users enter the feedback panel, real-time sociofeedback will automatically start. All the low-level features will appear on the top part of the screen and feedback message will be revealed in a small pop-up window on the screen in ever 20 seconds. If users want to quit the feedback panel, they can simply swipe down on the touchpad. An icon with label for quit is allocated at the bottom corner of this panel to remind users this gesture.

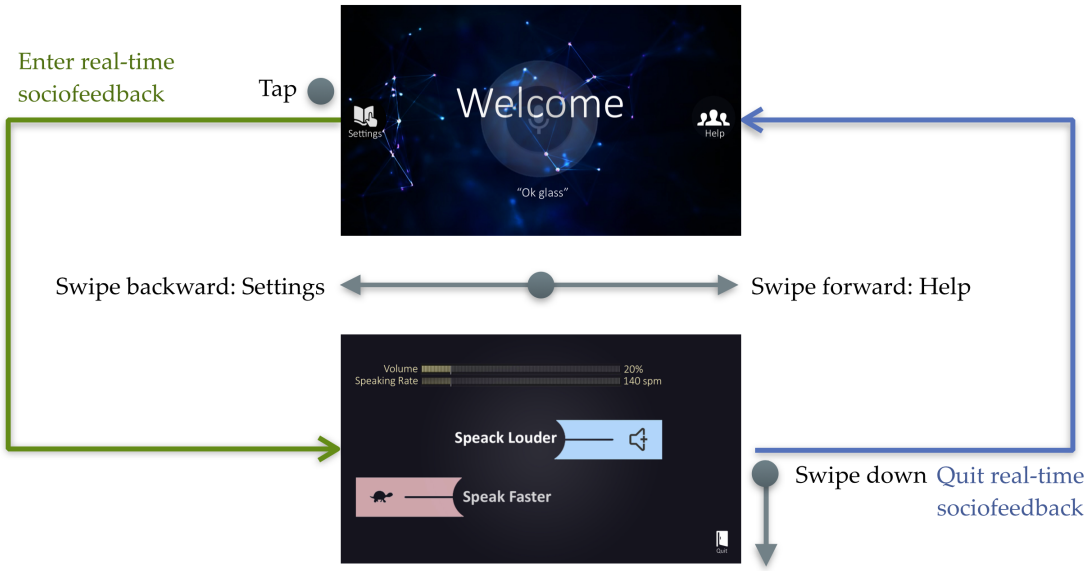


Figure 27. Prototype for Sociofeedback app on Google Glass

4.2 INTERFACE FOR ANDROID MOBILE PHONE

The application and functional requirement for the Sociofeedback on Android mobile phone remains the same to the Google Glass version except that users will interact with the app via touch screen instead of touchpad or voice comment. The difference between interaction with Google Glass and Android mobile phone is listed in Table 6.

Functional Requirements	Action on phone (Welcome panel)	Action on phone (feedback panel)	Action on Google Glass(on both panels)
Enter the feedback panel	Click the 'Enter sociofeedback' button	N.A.	Tap on the touchpad
Enter the Help panel	Click the help icon	Select 'Help' from the menu	Swipe forward
Enter the settings panel	Click the settings icon	Select 'Settings' from the menu	Swipe back
Back to the Welcome panel	N.A.	Select 'Home' from the menu	Swipe down

Table 6. Difference between interaction with Google Glass and mobile phone

The mobile version shares the same site map with the glass version. The mobile version will provide more information about the app, including a description of the two version of the Sociofeedback app and the development team. It will also show the details of low-level features, such as volume in dB and pitch in Hz, in the feedback message field. The interaction and layout of the mobile version is shown in the story board in Figure 28.

Figure 28. Storyboard of the Sociofeedback app on Android mobile phone

V DEVELOPMENT AND IMPLEMENTATION

Based on the user study carried out last year, users preferred to receive the feedback message graphically and the parallel progress bar seems confusing for users. A more dynamic progress bar is created to reflect the volume change. To distinguish pitch and volume, pitch is displayed in a line chart. For research purpose, all the values of the volume and pitch are shown in text as well. The low-level features is also displayed in plain-text. The SVM classifier is implemented in the app.

5.1 IMPLEMENTING THE INTERFACE ON GOOGLE GLASS

When users call the Sociofeedback app by voice comment: "ok glass" > "Sociofeedback", they enter the main activity of the app. The background of the welcome page is an image of the galaxy.

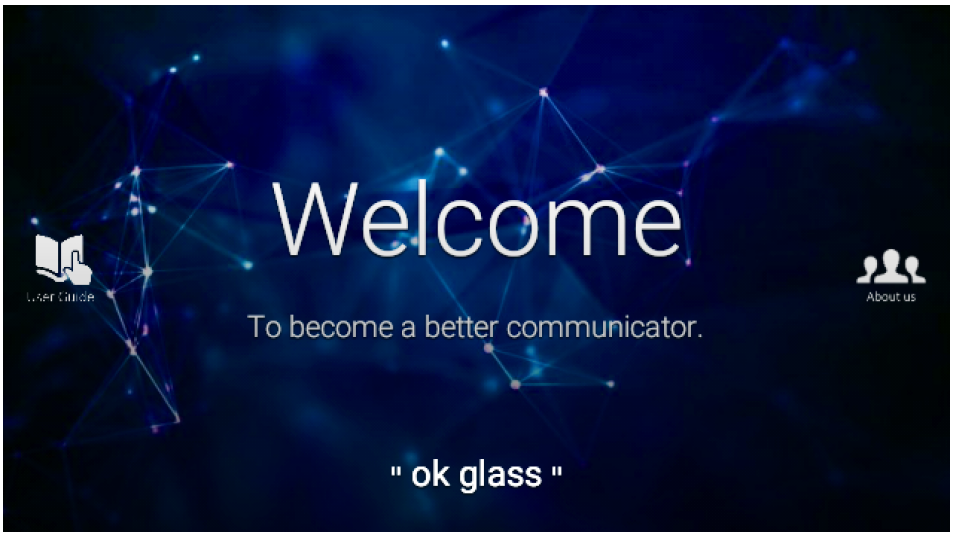


Figure 29. Welcome panel

To notify the user that the app is running robustly, the stars in the image is designed to twinkle just like the real ones. The animation is enabled by fade in and fade out the background image in every 2000 milliseconds. To smooth the transition of this procedure, an alpha blending is applied to the animator. In the Welcome panel, users can swipe forward or back to access the Help panel Settings panel. This is enabled by

```
ImageView background = (ImageView) findViewById(R.id.background);
ObjectAnimator fadeOut = ObjectAnimator.ofFloat(background, "alpha",  1f, .3f);
fadeOut.setDuration(2000);
ObjectAnimator fadeIn = ObjectAnimator.ofFloat(background, "alpha", .3f, 1f);
fadeIn.setDuration(2000);
final AnimatorSet mAnimationSet = new AnimatorSet();
mAnimationSet.play(fadeIn).after(fadeOut);
mAnimationSet.addListener(new AnimatorListenerAdapter() {
    @Override
    public void onAnimationEnd(Animator animation) {
        super.onAnimationEnd(animation);
        mAnimationSet.start();
    }
});
mAnimationSet.start();
```

Set fade in and fade of duration and alpha blending of the animation

Figure 30. Code for the animation of the Welcome panel background

the gesture detector on the Google Glass. Correspondent activities will be called according to users' gestures on the touchpad. Here, swipe forward is denoted as SWIPE_RIGHT and swipe back is denoted as SWIPE_LEFT in gesture detector listener. When users swipe down on the touchpad, they can quit this app. If they tap on the touchpad, they will enter the Feedback panel.

```
private GestureDetector createGestureDetector(Context context) {
    GestureDetector gestureDetector = new GestureDetector(context);
    //Create a base listener for generic gestures
    gestureDetector.setBaseListener( new GestureDetector.BaseListener() {
        @Override
        public boolean onGesture(Gesture gesture) {
            if  (gesture == Gesture.SWIPE_RIGHT) {
                Intent help = new Intent(MainActivity.this, AboutActivity.class);
                startActivity(help);
                return true;
            } else if (gesture == Gesture.SWIPE_LEFT) {
                Intent settings = new Intent(MainActivity.this, SettingsActivity.class);
                startActivity(settings);
                return true;
            } else if (gesture == Gesture.SWIPE_DOWN){
                finish();
                return true;
            } else if (gesture == Gesture.TAP){
                Intent com = new Intent(MainActivity.this, Combination.class);
                startActivity(com);
                return true;
            }
            return false;
        }
    });
    return gestureDetector;
}
```

Check gestures and define next activities

Figure 31. Code for gestures

If the Settings panel is selected, the following items will be shown on the screen. Since the Google Glass does not provide a touchable screen, the selection of features is done on the touchpad. Users can swipe forward or back to select the target checkboxes and click to change the status of the checkboxes. The final status of the checkboxes are recorded in SharedPreference object, so that it can be retrieved when

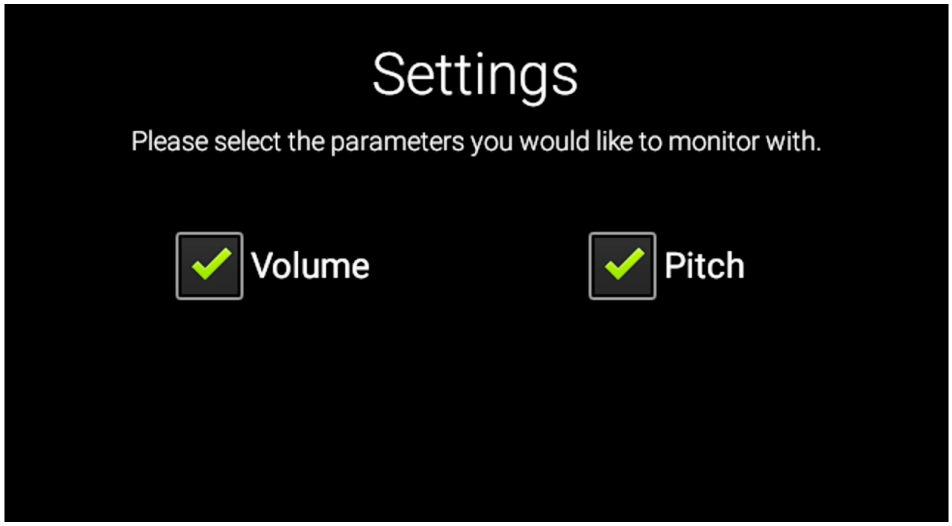


Figure 32. Settings panel

users enter the Feedback panel. When users quit the Settings panel, the status of the checkboxes will be automatically saved and a message will pop up to notify users that the settings has been saved.

Figure 33. Code for storing the checkbox status

```
private GestureDetector createGestureDetector(Context context) {
    GestureDetector gestureDetector = new GestureDetector(context);
    //Create a base listener for generic gestures
    gestureDetector.setBaseListener((gesture) → {
            if (gesture == Gesture.SWIPE_LEFT) {
                if (volumeChk.isFocused()) {
                    volumeChk.setSelected(false);
                    pitchChk.requestFocus();
                } else if (pitchChk.isFocused()) {
                    pitchChk.setSelected(false);
                    speechrateChk.requestFocus();
                } else if (speechrateChk.isEnabled()) {
                    speechrateChk.setSelected(false);
                    volumeChk.requestFocus();
                }
                return true;
            } else if (gesture == Gesture.SWIPE_RIGHT) {
                if (speechrateChk.isEnabled()) {
                    speechrateChk.setSelected(false);
                    volumeChk.requestFocus();
                } else if (pitchChk.isFocused()) {
                    pitchChk.setSelected(false);
                    speechrateChk.requestFocus();
                } else if (volumeChk.isFocused()) {
                    volumeChk.setSelected(false);
                    pitchChk.requestFocus();
                }
                return true;
            }
            return false;
    });

    return gestureDetector;
}

public void onCheckboxClicked(View view) {
    // Is the view now checked?
    boolean checked = ((CheckBox) view).isChecked();
    // Check which checkbox was clicked
    switch(view.getId()) {
        case R.id.checkbox_v:
            isChkV = checked;
            break;
        case R.id.checkbox_p:
            isChkP = checked;
            break;
        case R.id.checkbox_sr:
            isChkSR = checked;
            break;
        // TODO: Veggie sandwich
    }
}

@Override
protected void onStop(){
    super.onStop();
    // We need an Editor object to make preference changes.
    // All objects are from android.context.Context
    SharedPreferences settings = getSharedPreferences(Preferences, 0);
    SharedPreferences.Editor editor = settings.edit();
    editor.putBoolean(STATE_VOLUME, isChkV);
    editor.putBoolean(STATE_PITCH, isChkP);
    editor.putBoolean(STATE_SPEECHRATE, isChkSR);

    // Commit the edits!
    editor.commit();
    Toast.makeText(SettingsActivity.this, "Settings saved", Toast.LENGTH_SHORT).show();
}
```

Store the status of checkboxes in a global variable

Highlight the selected checkboxes based on gestures

Store the final status of the checkboxes when users quit the Settings panel

Notification

The Help panel provides the basic information on how to use this app and

introduction of the development team. Users can quit this panel by swiping down on the touchpad.

Figure 34. Help panel

When users enter the Feedback panel, real-time low-level features will be displayed as follows. Colour of the progress bar will be changed according to the level of loudness. It can give a rough indication of volume control to the user.

Figure 35. Feedback panel

The graphics are shown based on users' settings. The status of the checkboxes in the Settings panel is restored by retrieving values from the SharedPreference object.

```
SharedPreferences settings = getSharedPreferences(Preferences, 0);
Boolean restoredVol = settings.getBoolean(STATE_VOLUME, true);
Boolean restoredPitch = settings.getBoolean(STATE_PITCH, true);
Boolean restoredSR = settings.getBoolean(STATE_SPEECHRATE, true);

if(restoredVol == true){
    volumeLabel.setVisibility(View.VISIBLE);
    textViewV.setVisibility(View.VISIBLE);
    volume.setVisibility(View.VISIBLE);
    textVolume.setVisibility(View.VISIBLE);
    res_textVol.setVisibility(View.VISIBLE);
    lblvolume.setVisibility(View.VISIBLE);
    lblresvol.setVisibility(View.VISIBLE);
    Log.i(TAG, "Show");
} else {
    volumeLabel.setVisibility(View.INVISIBLE);
    textViewV.setVisibility(View.INVISIBLE);
    volume.setVisibility(View.INVISIBLE);
    textVolume.setVisibility(View.INVISIBLE);
    res_textVol.setVisibility(View.INVISIBLE);
    lblvolume.setVisibility(View.INVISIBLE);
    lblresvol.setVisibility(View.INVISIBLE);
    Log.i(TAG, "Not show");
}
```

Set the visibility of the TextViews, progress, and graph according to the restored settings

Figure 36. Code for restoring the checkbox status

5.2 IMPLEMENTING THE INTERFACE ON ANDROID PHONE

5.2.1 THE INTERFACE FOR MONOLOGUE

To retain consistency between the glass and mobile version, the Welcome panel inherit the background to the glass version. The layout is adjusted according to the screen size. An open source UI component is implemented to show the branding

Figure 37. Welcome panel, logo and the code for the animated branding message

message — “To become a better communicator.” The welcome message is replaced by the name of the app. The logo is designed typographically appealing. Several fonts have been tested and the best one is saved as a .PNG image. The logo “Sociofeedback” is displayed via an ImageView. User can access the Help panel and Settings panel by clicking the icons accordingly, and they can proceed to Feedback panel by clicking the transparent “GET STARTED” button. The structure is similar to the glass version.

Again, users can set their preference in Settings panel and find the information of this app in Help panel. More detailed content is added in these two panels, including App Information and introduction to the two versions. Users are required to click the “Done” button to quit the current panel instead of swiping down on the Google Glass. The top bar is created by a LinearLayout with embedded FrameLayout. The rest of the parts are separated by FrameLayout.

Figure 38. Settings panel and Help panel

The feedback panel has gone through several generations before it reaches the final design.

Figure 39. Evolution of the Feedback panel

The progress bar for volume will change to blue, yellow, green, and red according to the loudness of the speech. When pitch is detected, the status change to SPEECH DETECTED and the text change to yellow. Otherwise, the text is white. Users can turn on and off the real-time feedback by clicking the button in the middle of the panel. The menu is developed creating resideMenu from a UI library.

```
resideMenu = new ResideMenu(this);
resideMenu.setUse3D(true);
resideMenu.setBackground(R.drawable.background);
resideMenu.attachToActivity(this);
// create menu items;
itemHome     = new ResideMenuItem(this, R.drawable.icon_home,     "Home");
itemSettings = new ResideMenuItem(this, R.drawable.icon_settings, "Settings");
itemAboutUs  = new ResideMenuItem(this, R.drawable.icon_profile,  "About us");

itemHome.setOnClickListener((v) → {
        Intent sociometrics = new Intent(SociometricsActivity.this, MainActivity.class);
        startActivity(sociometrics);
        finish();
});
itemSettings.setOnClickListener((v) → {
        Intent settings = new Intent(SociometricsActivity.this, SettingsActivity.class);
        startActivity(settings);
});
itemAboutUs.setOnClickListener((v) → {
        Intent aboutus = new Intent(SociometricsActivity.this, AboutActivity.class);
        startActivity(aboutus);
});

resideMenu.addMenuItem(itemHome, ResideMenu.DIRECTION_LEFT);
resideMenu.addMenuItem(itemSettings, ResideMenu.DIRECTION_LEFT);
resideMenu.addMenuItem(itemAboutUs, ResideMenu.DIRECTION_LEFT);
resideMenu.setSwipeDirectionDisable(ResideMenu.DIRECTION_RIGHT);

findViewById(R.id.title_bar_left_menu).setOnClickListener((view) → {
        resideMenu.openMenu(ResideMenu.DIRECTION_LEFT);
});
```

Define menu items and the activities linked to them

Define the position of the menu

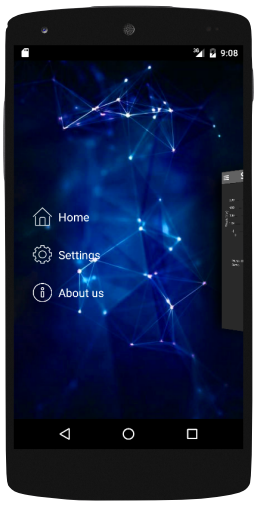


Figure 40. Code for the menu

5.2.2 ADJUSTING THE APPLICATION FOR TWO-PERSON DIALOGS

In order to simulate the situation when such app is used by call center, the app is further designed for displaying the low-level features of two audio inputs — one from the current user and the other from the server. The emulator is tested as the User 1 and the Google Glass is tested as the User 2. Interruption and Mutual Silence percentage are derived from the speech detection result of both users. The users are colour coded so that the user can distinguish his or her low-level features with the other user's easily.

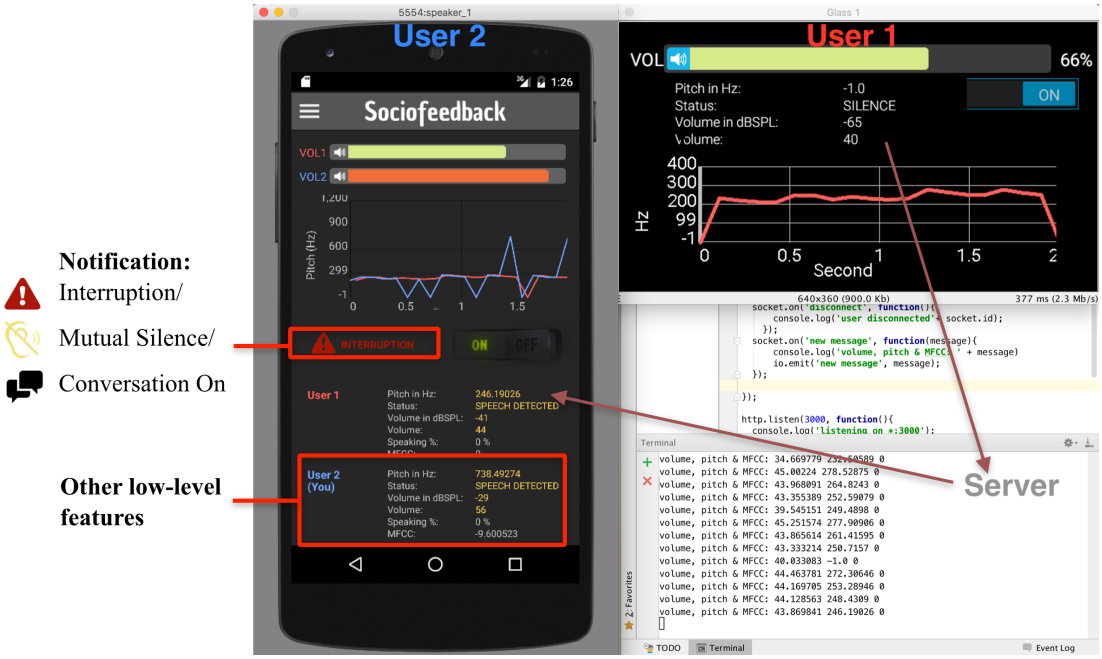


Figure 41. The interface for two users and data transmission path

Volume, pitch and MFCC are the fundamental low-level features detected by the App for single user. Then, the three values are transmitted to the server. The data are sent over Wi-Fi. The server broadcasts the values to the users with an emitter listener. This data transmission scheme relies on Socket.IO, a Javascript library developed by MIT. It is priorly designed for web applications, but a native Android library is created recently [27]. This library is ideal for real-time data transmission between client and server.

User 1

```
String message = Double.toString(Vol) + " " + Float.toString(pitchInHz) + " "
                + Float.toString(finalmfcc_val_float) ;
mSocket.emit("new message", message);
```

Server

```
socket.on('new message', function(message){
    console.log('volume, pitch & MFCC: ' + message)
    io.emit('new message', message);
});
```

User 2

```
private Emitter.Listener onNewMessage = new Emitter.Listener() {
    @Override
    public void call(final Object... args) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                String data = (String)args[0];
                String[] parts = data.split(" ");
                String vol = parts[0];
                String pitch = parts[1];
                String MFCC = parts[2];
```

Figure 42. Code for data transmission

After the User 2 receives the information of the User 1, interruptions and mutual silence can be detected by comparing the pitch of the two users. When both of the pitch 1 and pitch 2 are more than zero, the conversation state is marked as Interruption. On the contrary, if both of them are zero or below zero, the state is marked as Mutual Silence. In the rest of cases, the state is Conversation On. The notification beside the switch will change accordingly.

```
/* Interruption */
private boolean checkInterruption() { return (pitchInHz > 0 && pitchInHz2 > 0); }

/* Mutual Silence */
private boolean checkMutualSilence(){
    return (pitchInHz <= 0 && pitchInHz2 <= 0);
}
```

Figure 43. Code for checking Mutual Silence and Interruption

5.2.3 CLASSIFICATION

5.2.3.1 CLASSIFICATION ON PREVIOUS DATA

In the Sociofeedback app, the SVM developed by National Taiwan University is used to perform classification[26]. In “Real Time Comprehensive Sociometrics for Two Person Dialogs”, 98 dialogs are recorded in 2-channel .WAV files. The team has

extracted 29 features for each participant. We have 196 sessions of features in total. Those values are formatted in excel and used as training data for the classifier. In the paper, there sociofeedbacks are analysed — level of dominance, level of agreement, and level of interest. For each sociofeedback, the levels are 1-low, 2-medium, and 3-high. The data for analysing the level of agreement are used for testing in this project. Since SVM can only produce binary results, majority rule is introduced to finalise the classification result. The model 1 is trained with data classified as 1-low or 2-medium. The model 2 is trained with data classified as 1-low and 3-high. The model 3 is trained with data classified as 2-medium and 3-high. The testing data needs to be tested with the three models and the final result is the majority vote result. If the majority rule does not apply to the case, a random class is assigned to the classification result.

Figure 44. Code for training and testing

The 196 sessions of features are divided into two groups. One group for training and the other group for testing. To fully make use of the data, cross validation technique is employed.

Model 1	Model 2	Model 3	Result
1	1	2	**1**
1	1	3	**1**
1	3	2	**Random**
1	3	3	**3**
2	1	2	**2**
2	1	3	**Random**
2	3	2	**2**
2	3	3	**3**

Table 7. Majority vote

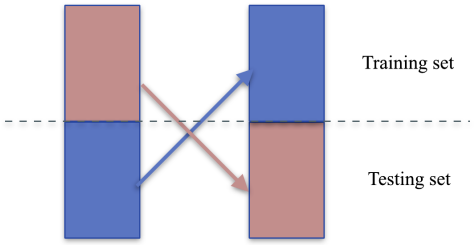


Figure 45. 2-fold Cross validation

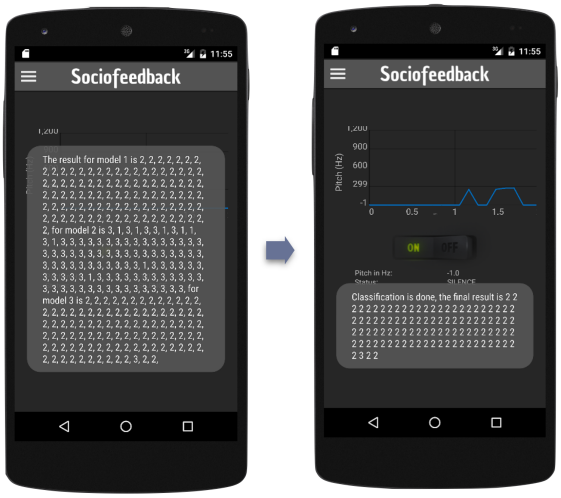


Figure 46. Classification results

5.2.3.2 CLASSIFICATION ON REAL-TIME DATA

Twelves features are derived from volume, pitch, and MFCC as the inputs for classification. They are Interruption, Speaking Percentage, Mutual Silence Percentage, Minimum Volume, Maximum Volume, Entropy of the Volume, Mean Volume, Minimum Pitch, Maximum Pitch, Entropy of the Pitch, Mean Pitch, and MFCC.

The function classificationData() sends the value to the classifier such that the value of the twelves features can feed into this classification function in real-time. The

feedback message is updated based on the standardised values of the past 80 data pairs. The standardisation function is:

$$x_{new} = \frac{x - \mu}{\sigma}$$

Here, the mean and standard deviation come from the previous data. The app is trained with the 196 sessions of data mentioned in session 5.2.3.1.

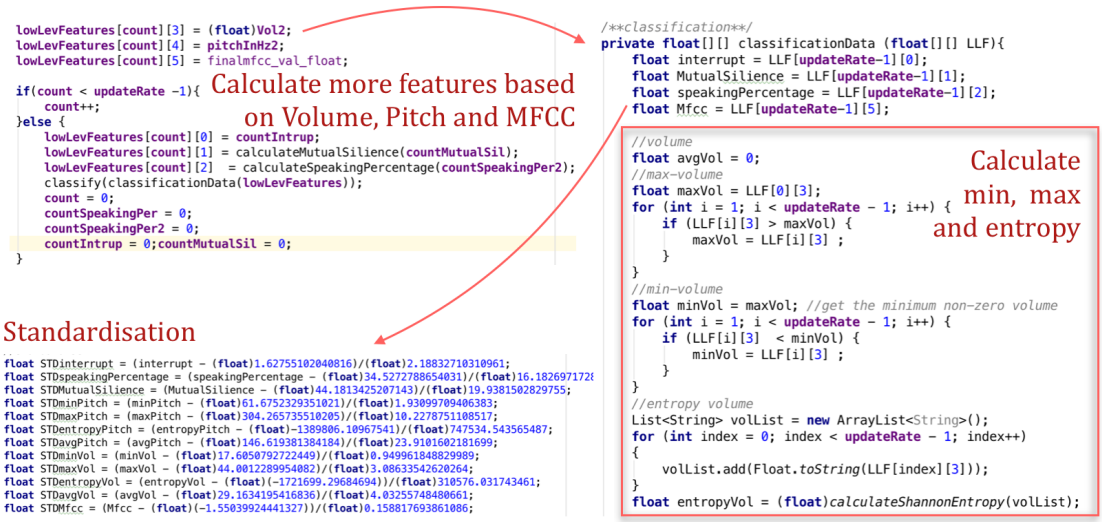


Figure 47. Code for calculating the 12 features

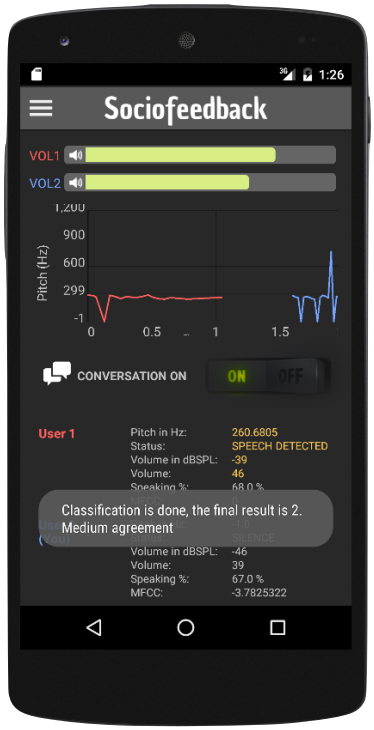


Figure 48. Feedback message

VI TESTING AND ENHANCEMENT

6.1 TEST CASES

A list of test cases has been design to test the functionality of the app. The test cases are set based on the application and functional requirements in session 4.1.1, and the story board in session 4.1.2 and 4.2.

Action	Result (Google Glass)	Result (Android Phone)
Start the Feedback panel from the Welcome panel	Enabled. The graphics show up.	Enabled. The graphics show according to settings.
Change settings from the Welcome panel	Not recommended. Glass goes hot. Sometimes the real-time feedback does not work.	Enabled. Show “Settings saved” and go back to the Welcome panel.
Check help information from the Welcome panel	Enabled. Show the content in correct layout.	Enabled. Show the content in correct layout.
Go back to Welcome panel from the Feedback panel	Enabled. Quit instantaneously.	Enabled. Need to go from the menu.
Change settings from the Feedback panel	Not recommended. Glass goes hot. Glass goes hot. Sometimes the real-time feedback does not work.	Enabled. Show “Settings saved” and go back to the menu when “Done” is clicked.
Check help information from the Feedback panel	Not recommended. Glass goes hot. Glass goes hot. Sometimes the real-time feedback does not work.	Enabled. Show the content and go back to the menu when “Done” is clicked.

Table 8. Test cases

For example, when users want to change their settings, they can access the menu by click the hamburger icon on the top left of the panel. For example, if the user would like to disable the volume part in the Feedback panel, they can go menu -> Settings -> Turn off Volume -> Done -> Swipe back the Feedback panel.

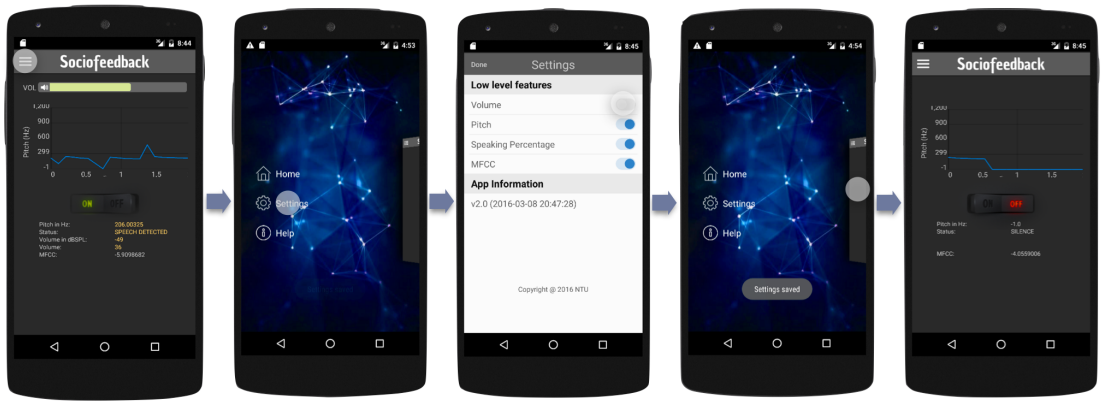


Figure 49. Test case example: Change settings from the Feedback Panel (for single user)

6.2 ENHANCEMENT

6.2.1 IMPROVEMENT OF THE INTERFACE

To maintain consistency across different platform, the welcome panel of the glass version is adjusted. The colour of the graph for pitch is changed to red and a switch is added as shown in Figure 41.

Figure 50. Updated welcome panel on Google Glass

Further more, users are allowed to enter their IP address in the mobile version so that the app does not needs to be re-compiled if the user change their internet connection (see in Figure 51).

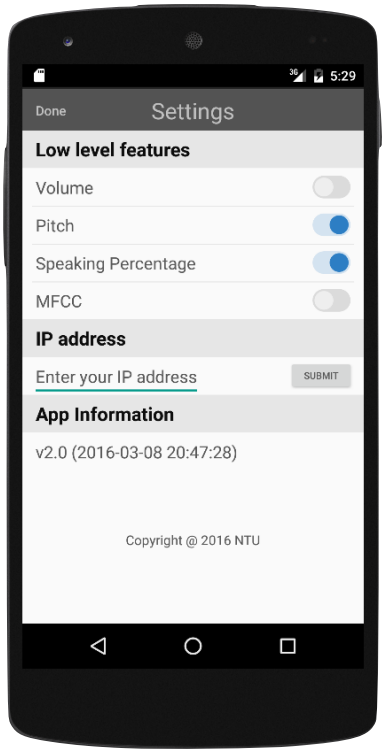


Figure 51. Updated settings panel

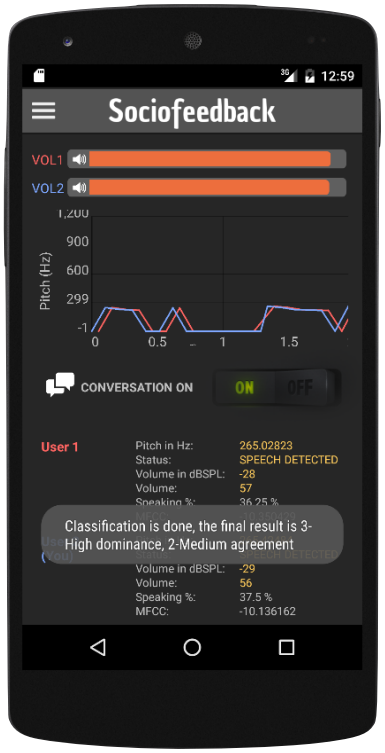


Figure 52. Updated feedback message

6.2.1 IMPROVEMENT OF THE CLASSIFICATION

Other than the level of agreement, the level of dominance is also taken into consideration now. Three extra datasets are added to provide the training data for classify the level of dominance. Those data come from the same dialogs in 5.2.3.1. The facilitator is left out from the training set and the accuracy of classification increases by 7%. Now the feedback message contains the result of two classification result as shown in Figure 52.

VII DISCUSSION

In this paper, a captivating interface is created which allows users to monitor their speech mannerism and social behaviour with minimal cognitive load. We quantified the level of agreement and level of dominance with the low-level speech metrics. The speech metrics and sociometric are displayed both in plain text and in graph.

From the testing result, we can observe that the Google Glass is not capable of reacting to rapid changes or performing complicated computation. Hence, only volume and pitch are extracted and displayed. The phone version for a single user represents the client. Three fundamental features, *Pitch, Volume, and MFCC*, are extracted and sent to the server. The phone version for two users is designed for specialists who work in a call centre or in similar situation. The app in this version will have the full list of features calculated and is able to perform classification.

Features	Google Glass	Android Phone (Single User)	Android Phone (Two users)
Pitch	√	√	√
Min Pitch	×	×	√
Max Pitch	×	×	√
Entropy of the Pitch	×	×	√
Mean Pitch	×	×	√
Volume	√	√	√
Min Volume	×	×	√
Max Volume	×	×	√
Entropy of the Volume	×	×	√
Mean Volume	×	×	√
Speaking%	×	×	√
No. of Interruption	×	×	√
Mutual Silence%	×	×	√
MFCC	×	√	√
Sociofeedback	×	×	√

Features	Google Glass	Android Phone (Single User)	Android Phone (Two users)
Target user	Customer	Customer	Specialist

Table 9. The available features of the speech in different versions of the app

The accuracy of classification on the previous data regarding the level of agreement is 69.38% and for the level of dominance is 67.34%. Most of the classification results for the level of agreement are 2-Medium. This can caused by the lack of training data. Around 60% of the training data are labeled as class 2. Similarly, majority of the classification result for the level of dominance is 3-High. If the dataset is divided into more folds, we could have utilised the data more efficiently. The accuracy could be higher. The real-time classification result shows that the data from the previous project may not be a reliable source for training since the recording devices and algorithms for extracting features are significantly changed. More experiments could have been done to obtain new data for this app. Nevertheless, the classification function is proved operating by comparing the classification result using this app with previous classification result. The accuracy could be improved by applying leave-one-out cross-validation to the classifier.

Compare to the existing Sociofeedback System, the Sociofeedback App is a more unified, portable and comprehensive application for users. Users are not required to possess the knowledge of social communication jargons and programming techniques. They can install the app on their mobile devices without going through complicated configuration. All the functions for data acquisition and analysis are re-written in Android. Since the microphone is embedded in the mobile devices and the data of the speech are transmitted through server, the devices does not have to rely on Wi-Fi and recording devices to perform classification on real-time dialogs. Instead, cellular data could support the data transmission and a mobile device should be enough for all the operations. Hence, users can turn on the sociofeedback wherever they are. For example, if the customer service specialists is on a trip, they can still use this app to improve their communication skills when they are talking to a client

over the phone.

	Sociofeedback System	Sociofeedback App
Interface	Preliminary	Advanced
Devices required	Multiple	The mobile device only
Programming Language	Matlab	Java (Android)
Platforms	Desktop, Vizux Glass, Google Glass	Google Glass, Android phone
Internet requirement	Wi-Fi	Wi-Fi or cellular data
No. of low-level speech metrics to be analysed	29	12
No. of sociometric to be provided	3	2
Portability	No	Yes

Table 9. The difference between existing system and this app

In terms of the functionalities, the Sociofeedback App shows the speech metrics in a more dynamic way and allows user to customise the speech metrics. However, the app does not extracted as many low-level features as the Sociofeedback System. Therefore, the classification result is not as accurate as the Sociofeedback System's. The sociometric is not fully released to users yet. The Sociofeedback system provide sociometric to both ends while the Sociofeedback app only provide it to the specialists. More experiments are required to collect data for the real-time classification. Further research is required to increase the classification accuracy and to interpret the social behaviour based on the classes.

Although, the classification of Sociofeedback app is not fine tuned, the results are promising: the low-level features can be show instantaneously and reliably, guaranteeing the data feed for the real-time sociofeedback. A group of PHD students intend to take over this project to conduct experiments for collecting a more diverse dataset for this app. After the SVM classifiers are tuned with new data, the app is definitely expected to become a handy tool to provide real-time sociofeedback to the users.

VIII CONCLUSION AND RECOMMENDATIONS

The Sociofeedback by Google Glass final year project aims to create a visually appealing and user-friendly interface for processing audio input and provides feedback to the users. High-fidelity diagrams have been created to illustrate the workflow of the application. Typography, colours and layout are meticulously considered. Investigation including volume detection and speaking rate calculation was carried on. The application has been iteratively designed both on the Google Glass and the Android phone. Machine learning algorithms has been explored and implemented on the app.

Over the past year, this project has undertaken deliberate planing and followed the design principles. The three iterations have gone through investigation and modelling, analysis and design, implementation, testing, and evaluation. The application provides adequate information of the usage and allows users to customise the feedback.

In addition to guide users to improve their communication skills in individual presentations or speeches, I further develop this application for conversations. 12 out of 29 features are extracted for classification. The classifier is tested with the data collected for the previous Sociofeedback System and the result is encouraging. However, due to the limited amount of time, we have not conduct experiments to collect new data for the Sociofeedback app. The real-time classification will be released to the users once the data are collected.

In the future, all the features of the speech will be collected from users and be stored in the central server (as shown in Figure 53). After that, the data can be distributed to the specialist. The Sociofeedback app is anticipated to be used in group meetings as well. The application will be able to perform multi-directional communication.

The Sociofeedback app is potentially linked to phone calls. In this case, the app will be capable of analysing recordings so that it is not necessary for the client to install the app on their mobile devices.

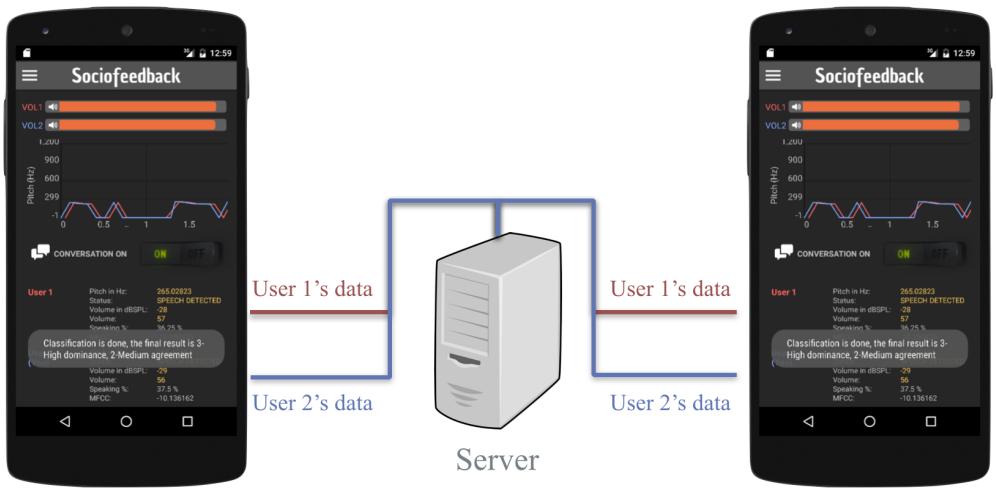


Figure 53. Multi-directional communication

A rating system can be designed to help the development team to collect new data (as shown in Figure 53). The fundamental low-level features — volume, pitch, and MFCC can be stored in a central database together with the rating from the user. Those data can become new training data for the system. Hence, every conversation can be considered as an experiment and every user can become a judge for it. The more data we have, the more accurate the classification result is. This approach can significantly increase the efficiency of data collection.

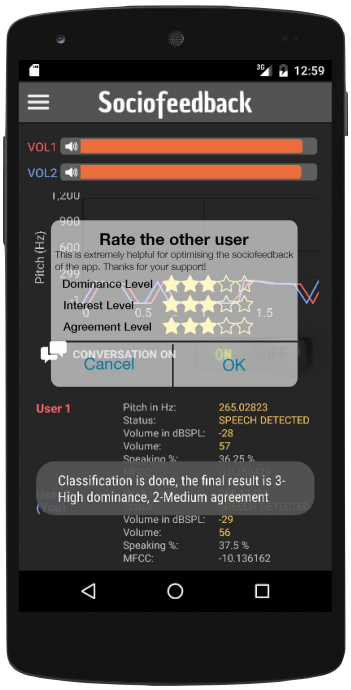


Figure 54. The Sociofeedback app with the rating system

Besides, an extra Performance panel can be created for evaluating the performance of the current user over a period of time. The performance is analysed by comparing the low-level features and the rating received of the current user with other users'. A brief comments can be offered to the current user to indicate how he or she may improve their speech in the future.

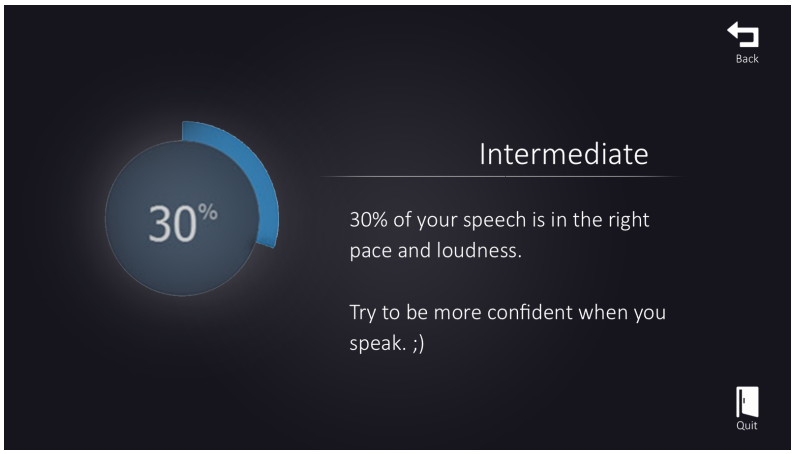


Figure 55. Proposed Performance Panel

The ultimate goal of developing such app is to provide the people who aims to improve their communication skills with their mobile devices. The research team has developed the Sociofeedback System for desktop users and the Sociofeedback app is ideal for Android mobile users and smart glass users. An iOS version and VR version could be developed in the near future. The app can be an accessible and reliable tool for users to check their social behaviour during the interviews, group meetings and presentations.

IX REFERENCE

[1] Elisabeth G., *Guidelines for Public Speaking*, Baruch College 2006

[2] Pentland, A.S.: *Socially aware, computation and communication*, Computer, vol. 38, no. 3, pp. 3340 (2005)

[3] Sampling. [Online]. http://medlibrary.org/medwiki/Sampling_(signal_processing)

[4] Indri Hartanto, "Sociofeedback by Vuzix Glasses", 2014

[5] Jeff Tang, *Voice and Audio*, *Beginning Google Glass Development*, 2014

[6] Tahir Y., Dauwels S., Dauwels J., Thalmann D., Thalmann N.M., Rasheed U., "Real Time Comprehensive Sociometrics for Two Person Dialogs", Institute for Media Innovation, Nanyang Technological University, Singapore, 2013.

[7] CDP Glossary of Technical Terms. [Online]. http://www.moz.ac.at/sem/lehre/lib/cdp/cdpr5/html/glosstec.html

[8] AudioRecord, Android Developer - Android, The world's most popular mobile platform.[Online].http://developer.android.com/reference/android/media/AudioRecord.html

[9] Martin C., Sensing Voice Amplitude with Android, DoEpicCoding. [Online] http://www.doepiccoding.com/blog/?p=195

[10] The Physics Classroom, Pitch and Frequency. [Online] http://www.physicsclassroom.com/class/sound/Lesson-2/Pitch-and-Frequency

[11] Android NDK, Android Developer - Android, The world's most popular mobile platform. [Online] http://developer.android.com/tools/sdk/ndk/index.html

[12] Google Developers, Voice Input. [Online]. https://developers.google.com/glass/develop/gdk/voice

[13] Android Studio [Online] http://developer.android.com/sdk/index.html#Requirements

[14] Screen resolution of Google Glass [Online] http://blogs.forrester.com/julie_ask/13-05-20-google_glass_what_ebusiness_professionals_need_to_know_this_early

[15] Skills You Need, What is communication [Online] http://www.skillsyouneed.com/general/what-is-communication.html

[16]Kevin A. Lee, *Introduction to Software Build and Release Management,* IBM Press 2006

[17] UC Berkeley, *Sound Programming* [Online] http://husk.eecs.berkeley.edu/courses/cs160-sp14/index.php/Sound_Programming

[18] Dave Marshall, *Digitisation of Sound* [Online] https://www.cs.cf.ac.uk/Dave/Multimedia/node145.html

[19]Microsoft, Differences Between PCM/ADPCM Wave Files Explained [Online] https://support.microsoft.com/en-us/kb/89879

[20] Presonus, Digital Audio Basics: Sample Rate and Bit Depth [Online] http://www.presonus.com/news/articles/sample-rate-and-bit-depth

[21] Anssi Klapuri, "Introduction to Music Transcription", in *Signal Processing Methods for Music Transcription*, edited by Anssi Klapuri and Manuel Davy, 1–20 (New York: Springer, 2006): p. 8. ISBN 978-0-387-30667-4.

[22] Roy D. Patterson, Etienne Gaudrain, and Thomas C. Walters (2010). "The Perception of Family and Register in Musical Tones". In Mari Riess Jones, Richard R. Fay, and Arthur N. Popper. *Music Perception*. Springer. pp. 37–38.
ISBN 978-1-4419-6113-6.

[23] TarsosDSP [Online] https://github.com/JorenSix/TarsosDSP

[24] Hall, M.A., “Correlation-based feature selection for machine learning”, The university of Waikato, 1999.

[25] Yu L., Liu H., “Feature selection for high-dimensional data: A fast correlation-based filter solution”, Proceeding of Machine Learning-International Workshop Then Conference, vol. 20, p.856, 2003.

[26] Hsu C.W., Chang C.C., Lin C.J., “A Practical Guide to Support Vector Classification ”, Department of Computer Science, National Taiwan University, 2010.

[27] Kanazawa N., “Native Socket.IO and Android” [Online] http://socket.io/blog/native-socket-io-and-android/

APPENDIX A - PROJECT PLAN

WEEK	2015				
	AUG	SEP	OCT	NOV	DEC
Monthly Objectives	Set up project scope and understand the benefits of Sociofeedback	Understand and implement speech recording function on Google Glass	Determine the volume and frequency of the speech	Link the speech mannerisms with feedback	Acquire speaking rate from the speech
Week 1		☐ Install the previous FYP App on Google Glass and learn the code. ☐ Learn open-source Google Glass Applications	☐ Analyse the raw data by Matlab and continue working on FFT	☐ Display volume using tool bar and roughly define the threshold for feedback	☐ Conduct research on speech recognition system on Google glass
Week 2		☐ Install the Google Glass emulator ☐ Program and test the audio recording function on Google Glass	☐ Make assumptions for the feedback and test the output on Google Glass ☐ Display the necessary parameters - Volume	☐ Understand the threshold for different types of feedbacks	☐ Implement the speech recognition function on Google glass
Week 3	☐ Clarify the project scopes	☐ Extract raw data from the audio clip	☐ Display the level of loudness as a number on the Glass screen panel	☐ Optimize the Application to ensure that the feedback is real-time	☐ Display the speaking rate on Google Glass
Week 4	☐ Settle down the FYP objectives and understand the previous FYP	☐ Try different methods to get frequency from the audio data array	☐ Optimize the App	☐ Adjust the interface to display the feedback intuitively	☐ Adjust the interface for displaying speaking rate
Deadline		**☐ 14/9/2015 Project plan and strategy**		**9/11/2015 Submission of Interim Report**	

WEEK	2016				
	JAN	FEB	MAR	APR	MAY
Monthly Objectives	Design a new interface for the Sociofeedback app on Google Glass	Design the interface for Android mobile phone	Apply classifier on the app and adjust the app for two person	☐ Establish TCP/IP connection to realise real-time data transmission	Presentation
Week 1	☐ List out applicational requirement and functional requirement ☐ Create wireframe for the interface	☐ Create wireframe and story board for the phone version	☐ Understand SVM ☐ Re-write the Gradle file to enable plugging in native library	☐ Modify the final report according to the result ☐ Improve classification accuracy	☐ Keep rehearsing the presentation and ensure the App is well functioning
Week 2	☐ Create storyboard for the interface ☐ Design prototype	☐ Try several layout and colours for the feedback panel	☐ Feed real-time data to the classifier	☐ Continue test the App ☐ Prepare the demo	☐ Presentation
Week 3	☐ Implement the new design on Google Glass	☐ Adjust the layout of the panels for the phone version	☐ Adjust the interface for two person ☐ Test the app with 2-channel audio input	☐ Work on presentation ☐ Create a video for introducing the app	
Week 4	☐ Refine the design and create settings and help panel	☐ Create a menu to link the panels	☐ Prepare and submit draft reports	☐ Rehearse the presentation	
Deadline			**☐ 3/24/2016 FYP Draft Report**	**4/8/2016 FYP Final Report** **4/11/2016 - 4/15/2016 Demo**	**5/9/2016 - 5/11/2016 Oral Presentation**

APPENDIX B - PROJECT CODE

JAVA

MainActivity.java

```
package com.example.user.pitchvolfeb;
/**
 * Created by Sailin on 01/2/16.
 */
import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.animation.AnimatorSet;
import android.animation.ObjectAnimator;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ImageView;

public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.main);

        ImageView background = (ImageView) findViewById(R.id.background);
        ImageButton settings = (ImageButton) findViewById(R.id.settings);
        settings.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Intent settings = new Intent(MainActivity.this, SettingsActivity.class);
                startActivity(settings);
```

```
        }
    });
    ImageButton aboutUs = (ImageButton) findViewById(R.id.aboutus);
    aboutUs.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            Intent aboutUs = new Intent(MainActivity.this, AboutActivity.class);
            startActivity(aboutUs);
        }
    });
    Button start = (Button) findViewById(R.id.start);
    start.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            Intent com = new Intent(MainActivity.this, SociometricsActivity.class);
            startActivity(com);
        }
    });

    //"To become a better communicator"
    SecretTextView secretTextView;
    ObjectAnimator fadeOut = ObjectAnimator.ofFloat(background, "alpha", 1f, .
3f);
    fadeOut.setDuration(2000);
    ObjectAnimator fadeIn = ObjectAnimator.ofFloat(background, "alpha", .3f, 1f);
    fadeIn.setDuration(2000);
    final AnimatorSet mAnimationSet = new AnimatorSet();
    mAnimationSet.play(fadeIn).after(fadeOut);
    mAnimationSet.addListener(new AnimatorListenerAdapter() {
        @Override
        public void onAnimationEnd(Animator animation) {
            super.onAnimationEnd(animation);
```

```
            mAnimationSet.start();
        }
    });
    mAnimationSet.start();

    secretTextView = (SecretTextView)findViewById(R.id.description);
    secretTextView.setDuration(3000);

    secretTextView.show();    // fade in
    secretTextView.hide();    // fade out
    secretTextView.toggle();
  }
}
```

SociometricsActivity.java

```
package com.example.user.pitchvolfeb;
/**
 * Created by Sailin and Priya on 14/2/16.
 * References: https://github.com/SpecialCyCi/AndroidResideMenu
 * Image source: https://dribbble.com/shots/343597-Switch-button
 */

import android.Manifest;
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.os.Bundle;
import android.os.Environment;
```

```
import android.support.v4.app.ActivityCompat;
import android.util.Log;
import android.view.View;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import com.akexorcist.roundcornerprogressbar.IconRoundCornerProgressBar;
import com.github.nkzawa.emitter.Emitter;
import com.github.nkzawa.socketio.client.IO;
import com.github.nkzawa.socketio.client.Socket;
import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.GridLabelRenderer;
import com.jjoe64.graphview.Viewport;
import com.jjoe64.graphview.series.DataPoint;
import com.jjoe64.graphview.series.LineGraphSeries;

import java.net.URISyntaxException;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Random;

import be.tarsos.dsp.AudioDispatcher;
import be.tarsos.dsp.AudioEvent;
import be.tarsos.dsp.AudioProcessor;
import be.tarsos.dsp.io.android.AudioDispatcherFactory;
import be.tarsos.dsp.mfcc.MFCC;
```

```
import be.tarsos.dsp.pitch.PitchDetectionHandler;
import be.tarsos.dsp.pitch.PitchDetectionResult;
import be.tarsos.dsp.pitch.PitchProcessor;

public class SociometricsActivity extends Activity  {

    private static final String TAG = "feedback";
    static final String STATE_VOLUME = "volume selection";
    static final String STATE_PITCH = "pitch selection";
    static final String STATE_SPEECHRATE = "speech rate selection";
    private static final String Preferences = "setting";

    private TextView volumeLabel;
    private TextView volumeLabel2;
    private IconRoundCornerProgressBar volume;
    private IconRoundCornerProgressBar volume2;

    private TextView lblvolume;
    private TextView textVolume;
    private TextView lblresvol;
    private TextView res_textVol;
    private TextView lblpitch;
    private TextView textPitch;
    private TextView lblrespitch;
    private TextView res_textPitch;
    private TextView lblresspeackingPer;
    private TextView res_speakingPer;
    private TextView lblmfcc;
    private TextView res_mfcc;

    private TextView lblvolume2;
```

```
private TextView textVolume2;
private TextView lblresvol2;
private TextView res_textVol2;
private TextView lblpitch2;
private TextView textPitch2;
private TextView lblrespitch2;
private TextView res_textPitch2;
private TextView lblresspeackingPer2;
private TextView res_speakingPer2;
private TextView lblmfcc2;
private TextView res_mfcc2;

private GraphView graph;
private LineGraphSeries<DataPoint> series_aud;
private LineGraphSeries<DataPoint> series_aud2;
private static final Random RANDOM = new Random();
private int lastX = 0;

private ResideMenu resideMenu;
private ResideMenuItem itemHome;
private ResideMenuItem itemAboutUs;
private ResideMenuItem itemSettings;

private ImageButton Btnfeedback;
private ImageView ImgConvState;
private TextView TextConvState;
private boolean mStartFeedback = true;
private AudioDispatcher dispatcher2;

//low level features
private double Vol;
```

```
    private float pitchInHz;
    private int countSpeakingPer = 0;
    private int countMutualSil = 0;
    private int countIntrup = 0;
    private double Vol2;
    private float pitchInHz2;
    private int countSpeakingPer2 = 0;
    private int count = 0;
    private int updateRate = 80; // the larger the lower update rate - for feedback
message;
    private int noOfLLF = 6; //number of low level features : volume (x4: mean, max,
min, entropy), pitch (x4), speaking%, MFCC, Interruption count, Mutual silence %
    private float[][] lowLevFeatures = new float[updateRate][noOfLLF];

    //MFCC centerfrequency
    private int bufferSize = 1024;
    private int sampleRate = 8000;
    private MFCC mfcc = new MFCC(bufferSize, sampleRate);

    private Socket mSocket;
    {
        try {
            mSocket = IO.socket("http://172.20.208.27:3000");  //LWN:172.22.185.215
HOME: 172.20.208.27
        } catch (URISyntaxException e) {}
    }

    //read&write permission
    //Storage Permissions
    private static final int REQUEST_EXTERNAL_STORAGE = 1;
    private static String[] PERMISSIONS_STORAGE = {
```

```
            Manifest.permission.READ_EXTERNAL_STORAGE,
            Manifest.permission.WRITE_EXTERNAL_STORAGE
    };

    //for classification
    private native int trainClassifierNative(String trainingFile, int kernelType,
                                             int cost, float gamma, int isProb, String modelFile);
    private native int doClassificationNative(float values[][], int indices[][],
                                              int isProb, String modelFile, int labels[], double probs[]);

    static {
        System.loadLibrary("signal");
    }


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sociometrics);
        verifyStoragePermissions(this);

        //Volume
//        textViewV=(TextView)findViewById(R.id.textViewV);
        volumeLabel=(TextView)findViewById(R.id.volume);
        volume = (IconRoundCornerProgressBar) findViewById(R.id.progressBarV);
        lblvolume = (TextView) findViewById(R.id.lbl_tvdBlevel);
        textVolume = (TextView) findViewById(R.id.tvdBlevel);
        lblresvol = (TextView) findViewById(R.id.lbl_res_tvdBlevel);
        res_textVol = (TextView) findViewById(R.id.result_tvdBlevel);
        lblresspeackingPer = (TextView)findViewById(R.id.lbl_res_speakingPer);
        res_speakingPer = (TextView) findViewById(R.id.result_speakingPer);
```

```
lblmfcc = (TextView) findViewById(R.id.lbl_res_mfcc);
res_mfcc = (TextView) findViewById(R.id.result_mfcc);


volumeLabel2=(TextView)findViewById(R.id.volume2);
volume2 = (IconRoundCornerProgressBar) findViewById(R.id.progressBarV2);
lblvolume2 = (TextView) findViewById(R.id.lbl_tvdBlevel2);
textVolume2 = (TextView) findViewById(R.id.tvdBlevel2);
lblresvol2 = (TextView) findViewById(R.id.lbl_res_tvdBlevel2);
res_textVol2 = (TextView) findViewById(R.id.result_tvdBlevel2);
lblmfcc2 = (TextView) findViewById(R.id.lbl_res_mfcc2);
res_mfcc2 = (TextView) findViewById(R.id.result_mfcc2);


//Pitch
lblpitch = (TextView) findViewById(R.id.lbl_tvMessage);
textPitch = (TextView) findViewById(R.id.tvMessage);
lblrespitch = (TextView) findViewById(R.id.lbl_result_txtview);
res_textPitch = (TextView) findViewById(R.id.result_txtview);

lblpitch2 = (TextView) findViewById(R.id.lbl_tvMessage2);
textPitch2 = (TextView) findViewById(R.id.tvMessage2);


//Speaking percentage
lblresspeackingPer2 = (TextView)findViewById(R.id.lbl_res_speakingPer2);
res_speakingPer2 = (TextView) findViewById(R.id.result_speakingPer2);

lblrespitch2 = (TextView) findViewById(R.id.lbl_result_txtview2);
res_textPitch2 = (TextView) findViewById(R.id.result_txtview2);
```

```
//Graph
graph = (GraphView) findViewById(R.id.graph);
graph.setTitleColor(Color.WHITE);

//data to plot
series_aud = new LineGraphSeries<DataPoint>();
series_aud2 = new LineGraphSeries<DataPoint>();
graph.addSeries(series_aud);
graph.addSeries(series_aud2);
series_aud.setColor(Color.parseColor("#ff6666"));
series_aud2.setColor(Color.parseColor("#80aaff"));
// customize - viewport
Viewport viewport = graph.getViewport();
viewport.setYAxisBoundsManual(true);
viewport.setMinY(-1);
viewport.setMaxY(1200);
viewport.setScrollable(true);

GridLabelRenderer labelGraph = graph.getGridLabelRenderer();
labelGraph.setHorizontalAxisTitle("Time (s)");
labelGraph.setVerticalAxisTitle("Pitch (Hz)");
labelGraph.setTextSize(6);
labelGraph.setVerticalAxisTitleColor(Color.parseColor("#C0C0C0"));
labelGraph.setHorizontalAxisTitleColor(Color.parseColor("#C0C0C0"));
labelGraph.setGridColor(Color.parseColor("#C0C0C0"));
labelGraph.setGridStyle(GridLabelRenderer.GridStyle.BOTH);
labelGraph.setHorizontalLabelsColor(Color.parseColor("#C0C0C0"));
labelGraph.setVerticalLabelsColor(Color.parseColor("#C0C0C0"));
labelGraph.setHorizontalAxisTitleTextSize(6);
labelGraph.setHorizontalAxisTitleTextSize(6);
```

```
        //conversation state and start button
        Btnfeedback = (ImageButton) findViewById(R.id.startfeedback);
        Btnfeedback.setOnClickListener(clicker);
        ImgConvState = (ImageView) findViewById(R.id.imgConvState);
        TextConvState = (TextView) findViewById(R.id.textConvState);

        // attach to current activity;
        resideMenu = new ResideMenu(this);
        resideMenu.setUse3D(true);
        resideMenu.setBackground(R.drawable.background);
        resideMenu.attachToActivity(this);
        // create menu items;
        itemHome     = new ResideMenuItem(this, R.drawable.icon_home,     "Home");
        itemSettings = new ResideMenuItem(this, R.drawable.icon_settings, "Settings");
        itemAboutUs  = new ResideMenuItem(this, R.drawable.icon_profile,  "About
us");

        itemHome.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Intent sociometrics = new Intent(SociometricsActivity.this,
MainActivity.class);
                startActivity(sociometrics);
                finish();
            }
        });
        itemSettings.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Intent settings = new Intent(SociometricsActivity.this,
SettingsActivity.class);
                startActivity(settings);
```

```
        }
    });
    itemAboutUs.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            Intent aboutus = new Intent(SociometricsActivity.this, AboutActivity.class);
            startActivity(aboutus);
        }
    });

    resideMenu.addMenuItem(itemHome, ResideMenu.DIRECTION_LEFT);
    resideMenu.addMenuItem(itemSettings, ResideMenu.DIRECTION_LEFT);
    resideMenu.addMenuItem(itemAboutUs, ResideMenu.DIRECTION_LEFT);
    resideMenu.setSwipeDirectionDisable(ResideMenu.DIRECTION_RIGHT);

    findViewById(R.id.title_bar_left_menu).setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View view) {
            resideMenu.openMenu(ResideMenu.DIRECTION_LEFT);
        }
    });


}

@Override
protected void onResume() {
    super.onResume();
    SharedPreferences settings = getSharedPreferences(Preferences, 0);
    Boolean restoredVol = settings.getBoolean(STATE_VOLUME, true);
    Boolean restoredPitch = settings.getBoolean(STATE_PITCH, true);
```

```
Boolean restoredSR = settings.getBoolean(STATE_SPEECHRATE, true);
Log.i(TAG, restoredVol.toString());


//respond according to settings
if(restoredVol == true){
    volumeLabel.setVisibility(View.VISIBLE);
    volume.setVisibility(View.VISIBLE);
    textVolume.setVisibility(View.VISIBLE);
    res_textVol.setVisibility(View.VISIBLE);
    lblvolume.setVisibility(View.VISIBLE);
    lblresvol.setVisibility(View.VISIBLE);

    volumeLabel2.setVisibility(View.VISIBLE);
    volume2.setVisibility(View.VISIBLE);
    textVolume2.setVisibility(View.VISIBLE);
    res_textVol2.setVisibility(View.VISIBLE);
    lblvolume2.setVisibility(View.VISIBLE);
    lblresvol2.setVisibility(View.VISIBLE);
} else {
    volumeLabel.setVisibility(View.INVISIBLE);
    volume.setVisibility(View.INVISIBLE);
    textVolume.setVisibility(View.INVISIBLE);
    res_textVol.setVisibility(View.INVISIBLE);
    lblvolume.setVisibility(View.INVISIBLE);
    lblresvol.setVisibility(View.INVISIBLE);

    volumeLabel2.setVisibility(View.INVISIBLE);
    volume2.setVisibility(View.INVISIBLE);
    textVolume2.setVisibility(View.INVISIBLE);
    res_textVol2.setVisibility(View.INVISIBLE);
```

```
        lblvolume2.setVisibility(View.INVISIBLE);
        lblresvol2.setVisibility(View.INVISIBLE);
    }

    if(restoredPitch == true){
        textPitch.setVisibility(View.VISIBLE);
        res_textPitch.setVisibility(View.VISIBLE);
        lblpitch.setVisibility(View.VISIBLE);
        lblrespitch.setVisibility(View.VISIBLE);
        graph.setVisibility(View.VISIBLE);

        textPitch2.setVisibility(View.VISIBLE);
        res_textPitch2.setVisibility(View.VISIBLE);
        lblpitch2.setVisibility(View.VISIBLE);
        lblrespitch2.setVisibility(View.VISIBLE);
    }else{
        textPitch.setVisibility(View.INVISIBLE);
        res_textPitch.setVisibility(View.INVISIBLE);
        lblpitch.setVisibility(View.INVISIBLE);
        lblrespitch.setVisibility(View.INVISIBLE);
        graph.setVisibility(View.INVISIBLE);

        textPitch2.setVisibility(View.INVISIBLE);
        res_textPitch2.setVisibility(View.INVISIBLE);
        lblpitch2.setVisibility(View.INVISIBLE);
        lblrespitch2.setVisibility(View.INVISIBLE);
    }


}
```

```
//onclicklistener for btnFeedback
View.OnClickListener clicker = new View.OnClickListener() {
    @Override
    public void onClick(View arg0) {
        onFeedback(mStartFeedback);
        mStartFeedback = !mStartFeedback;
    }
};


private void onFeedback(boolean start) {
    if (start) {
        Btnfeedback.setImageResource(R.drawable.on);
        startFeedback();
        mSocket.on("new message", onNewMessage);
        mSocket.connect();
        train();

    } else {
        Btnfeedback.setImageResource(R.drawable.off);
        stopFeedback();
    }
}

private void stopFeedback(){
    dispatcher2.stop();
    dispatcher2 = null;

    mSocket.disconnect();
    mSocket.off("new message", onNewMessage);
```

```
    }

    private void startFeedback(){
        //real-time - speaker 2 (You)
        dispatcher2 = AudioDispatcherFactory.fromDefaultMicrophone(sampleRate,
bufferSize, 0);
        PitchDetectionHandler pdh2 = new PitchDetectionHandler() {
            @Override
            public void handlePitch(PitchDetectionResult result, AudioEvent e) {
                //Calculating MFCC
                float[] mfcc_buffer;
                float[] mfcc_val;
                float sum=0;
                mfcc_buffer = e.getFloatBuffer();
                float bin[] = mfcc.magnitudeSpectrum(mfcc_buffer);
                float fbank[] = mfcc.melFilter(bin, mfcc.getCenterFrequencies());
                float f[] = mfcc.nonLinearTransformation(fbank);
                mfcc_val = mfcc.cepCoefficients(f);

                float min_mfcc = mfcc_val[0];
                //for display purpose,  we take the minimum MFCC
                for (int i = 1; i < mfcc_val.length; i++) {
                    if (mfcc_val[i] < min_mfcc) {
                        min_mfcc = mfcc_val[i];
                    }
                }
                final float finalmfcc_val_float = min_mfcc;

                //Calculating SPL value in dB
                final Double dbSPLValue2 = calculate(e.getFloatBuffer());
                Vol2 = dbSPLValue2 + 85.0;  //+70.0
```

```
            //Getting Pitch Frequency in Hz
            pitchInHz2 = result.getPitch();
            //Log.d(TAG, "Volume: " + String.valueOf(Vol2) + "; pitch:" +
String.valueOf(pitchInHz2));

            runOnUiThread(new Runnable() {

               @Override
               public void run() {

                  //update pitch
                  textPitch2.setText("" + pitchInHz2);
                  if (pitchInHz2 > 0) {
                     res_textPitch2.setText("SPEECH DETECTED");
                     textPitch2.setTextColor(Color.parseColor("#FFCC66"));
                     res_textPitch2.setTextColor(Color.parseColor("#FFCC66"));
                     textVolume2.setTextColor(Color.parseColor("#FFCC66"));
                     res_textVol2.setTextColor(Color.parseColor("#FFCC66"));
                     countSpeakingPer2++;

                  } else if (pitchInHz2 == -1) {
                     res_textPitch2.setText("SILENCE");
                     textPitch2.setTextColor(Color.parseColor("#C0C0C0"));
                     res_textPitch2.setTextColor(Color.parseColor("#C0C0C0"));
                     textVolume2.setTextColor(Color.parseColor("#C0C0C0"));
                     res_textVol2.setTextColor(Color.parseColor("#C0C0C0"));
                     pitchInHz2 = 0;
                  }

                  //update conversation state
```

```
if (checkInterruption()){
   countIntrup++;
   ImgConvState.setImageResource(R.drawable.interruption);
   TextConvState.setText("INTERRUPTION");
   TextConvState.setTextColor(Color.parseColor("#a71d15"));
} else if(checkMutualSilence()){
   countMutualSil++;
   ImgConvState.setImageResource(R.drawable.silence);
   TextConvState.setText("MUTUAL SILIENCE");
   TextConvState.setTextColor(Color.parseColor("#efe48a"));
} else {
   ImgConvState.setImageResource(R.drawable.conversation);
   TextConvState.setText("CONVERSATION ON");
   TextConvState.setTextColor(Color.parseColor("#c0c0c0"));
}

//update volume
textVolume2.setText(String.valueOf(Math.round(dbSPLValue2)));
res_textVol2.setText(String.valueOf(Math.round(Vol2)));
if(Vol2 >= 0 && Vol2 <= 14){
   volume2.setProgressColor(Color.parseColor("#8dcdc1"));
}else if(Vol2 > 15 && Vol2 <= 25){
   volume2.setProgressColor(Color.parseColor("#fff5c3"));
}else if(Vol2 > 25 && Vol2 <= 50){
   volume2.setProgressColor(Color.parseColor("#d3e397"));
}else if(Vol2 > 50){
   volume2.setProgressColor(Color.parseColor("#eb6e44"));
}
volume2.setProgress((float) (Vol2 / 60 * 100));
//draw pitch
series_aud2.appendData(new DataPoint(lastX++, pitchInHz2), true,
```

```
20);
                //display MFCC
                res_mfcc2.setText(Float.toString(finalmfcc_val_float));

                //store data for classification

                lowLevFeatures[count][3] = (float)Vol2;
                lowLevFeatures[count][4] = pitchInHz2;
                lowLevFeatures[count][5] = finalmfcc_val_float;

                if(count < updateRate -1){
                   count++;
                }else {
                   lowLevFeatures[count][0] = countIntrup;
                   lowLevFeatures[count][1] =
calculateMutualSilience(countMutualSil);
                   lowLevFeatures[count][2]  =
calculateSpeakingPercentage(countSpeakingPer2);

res_speakingPer.setText(Double.toString(calculateSpeakingPercentage(countSpeakin
gPer)) + " %");
                   res_speakingPer2.setText(Double.toString(lowLevFeatures[count]
[2]) + " %");
                   //classificationData(lowLevFeatures);
                   classify(classificationData(lowLevFeatures));
                   count = 0;
                   countSpeakingPer = 0;
                   countSpeakingPer2 = 0;
                   countIntrup = 0;countMutualSil = 0;

                }
```

```
                }


            });
        }
    };

    //real-time input - speaker 2 (You)
    AudioProcessor p2 = new
PitchProcessor(PitchProcessor.PitchEstimationAlgorithm.FFT_YIN, sampleRate,
bufferSize, pdh2);
    dispatcher2.addAudioProcessor(p2);
    new Thread(dispatcher2, "Audio Dispatcher - real-time").start();

}

/* calculate volume in dB  */
private Double calculate(float[] floatBuffer) {
    double power = 0.0D;
    for (float element : floatBuffer) {
        power += element * element;
    }
    double value = Math.pow(power, 0.5)/ floatBuffer.length;;
    return 20.0 * Math.log10(value);
}

/* Interruption */
private boolean checkInterruption(){
    return (pitchInHz > 0 && pitchInHz2 > 0);
```

```
}

/* Mutual Silence */
private boolean checkMutualSilence(){
    return (pitchInHz <= 0 && pitchInHz2 <= 0);
}

private float calculateMutualSilience(int cntMutualSil){
    float mutualSilPer;
    mutualSilPer = (float)cntMutualSil/updateRate * 100;  //in number
    DecimalFormat df = new DecimalFormat("#.##");
    mutualSilPer = Float.valueOf(df.format(mutualSilPer));
    return mutualSilPer;
}

/* Speaking Percentage */
private float calculateSpeakingPercentage(int  cntSpeakPer){
    float speakingPer;
    speakingPer = (float)cntSpeakPer/updateRate * 100;  //in100%
    DecimalFormat df = new DecimalFormat("#.##");
    speakingPer = Float.valueOf(df.format(speakingPer));
    return speakingPer;
}

/* entropy */
public static double calculateShannonEntropy(List<String> values) {
    Map<String, Integer> map = new HashMap<String, Integer>();
    // count the occurrences of each value
    for (String sequence : values) {
        if (!map.containsKey(sequence)) {
            map.put(sequence, 0);
```

```
        }
        map.put(sequence, map.get(sequence) + 1);
    }
    // calculate the entropy
    double result = 0.0;
    for (String sequence : map.keySet()) {
        double frequency = (double) map.get(sequence) / values.size();
        result -= frequency * (Math.log(frequency) / Math.log(2));
    }
    return result;
}
```

```
/** real-time - speaker 1 (from Google Glass/another user)
    receive data from the server **/
private Emitter.Listener onNewMessage = new Emitter.Listener() {
    @Override
    public void call(final Object... args) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                String data = (String)args[0];
                String[] parts = data.split(" ");
                String vol = parts[0];
                String pitch = parts[1];
                String MFCC = parts[2];
```

```
Vol = Double.valueOf(vol);
Double dbSPLValue = Vol - 85.0;
pitchInHz = Float.valueOf(pitch);

textPitch.setText("" + pitchInHz);
if (pitchInHz > 0) {
   res_textPitch.setText("SPEECH DETECTED");
   textPitch.setTextColor(Color.parseColor("#FFCC66"));
   res_textPitch.setTextColor(Color.parseColor("#FFCC66"));
   textVolume.setTextColor(Color.parseColor("#FFCC66"));
   res_textVol.setTextColor(Color.parseColor("#FFCC66"));
   countSpeakingPer++;
} else if (pitchInHz == -1) {
   res_textPitch.setText("SILENCE");
   textPitch.setTextColor(Color.parseColor("#C0C0C0"));
   res_textPitch.setTextColor(Color.parseColor("#C0C0C0"));
   textVolume.setTextColor(Color.parseColor("#C0C0C0"));
   res_textVol.setTextColor(Color.parseColor("#C0C0C0"));
}


textVolume.setText(String.valueOf(Math.round(dbSPLValue)));
res_textVol.setText(String.valueOf(Math.round(Vol)));
if (Vol >= 0 && Vol <= 14) {
   volume.setProgressColor(Color.parseColor("#8dcdc1"));
} else if (Vol > 15 && Vol <= 25) {
   volume.setProgressColor(Color.parseColor("#fff5c3"));
} else if (Vol > 25 && Vol <= 50) {
   volume.setProgressColor(Color.parseColor("#d3e397"));
} else if (Vol > 50) {
   volume.setProgressColor(Color.parseColor("#eb6e44"));
```

```
                }
                volume.setProgress((float) (Vol / 60 * 100));
                series_aud.appendData(new DataPoint(lastX++, pitchInHz), true, 20);
                res_mfcc.setText(MFCC);

            }
        });
    }
};

//disconnect socket
@Override
public void onDestroy() {
    super.onDestroy();

    mSocket.disconnect();
    mSocket.off("new message", onNewMessage);
}

/**classification**/
private float[][] classificationData (float[][] LLF){
    float interrupt = LLF[updateRate-1][0];
    float MutualSilience = LLF[updateRate-1][1];
    float speakingPercentage = LLF[updateRate-1][2];
    float Mfcc = LLF[updateRate-1][5];

    //volume
    float avgVol = 0;
    //max-volume
    float maxVol = LLF[0][3];
    for (int i = 1; i < updateRate - 1; i++) {
```

```
    if (LLF[i][3] > maxVol) {
        maxVol = LLF[i][3] ;
    }
}
//min-volume
float minVol = maxVol; //get the minimum non-zero volume
for (int i = 1; i < updateRate - 1; i++) {
    if (LLF[i][3]  < minVol) {
        minVol = LLF[i][3] ;
    }
}
//entropy volume
List<String> volList = new ArrayList<String>();
for (int index = 0; index < updateRate - 1; index++)
{
    volList.add(Float.toString(LLF[index][3]));
}
float entropyVol = (float)calculateShannonEntropy(volList);

//pitch
float avgPitch = 0;
//max-pitch
float maxPitch = LLF[0][4];
for (int i = 1; i < updateRate - 1; i++) {
    if (LLF[i][4] > maxPitch) {
        maxPitch = LLF[i][4];
    }
}
//min-pitch
float minPitch = maxPitch; //get the minimum non-zero pitch
for (int i = 1; i < updateRate - 1; i++) {
```

```
            if (LLF[i][4] < minPitch && LLF[i][4] > 0) {
                minPitch = LLF[i][4];
            }
        }
        //entropy-pitch
        List<String> PitchList = new ArrayList<String>();
        for (int index = 0; index < updateRate - 1; index++)
        {
            volList.add(Float.toString(LLF[index][4]));
        }
        float entropyPitch = (float)calculateShannonEntropy(volList);
        //avg-pitch&vol
        for (int i = 0; i < updateRate -1; i++){
            avgVol = avgVol + LLF[i][3];
            avgPitch = avgPitch + LLF[i][4];
        }

        avgVol = (float) ((avgVol/updateRate)/60.0); //normalized average volume
        avgPitch = (float)((avgPitch/updateRate)/500.0); //normalized average pitch

        //standardise
        float STDinterrupt = (interrupt - (float)1.62755102040816)/
(float)2.18832710310961;
        float STDspeakingPercentage = (speakingPercentage -
(float)34.5272788654031)/(float)16.1826971728663;
        float STDMutualSilience = (MutualSilience - (float)44.1813425207143)/
(float)19.9381502829755;
        float STDminPitch = (minPitch - (float)61.6752329351021)/
(float)1.93099709406383;
        float STDmaxPitch = (maxPitch - (float)304.265735510205)/
(float)10.2278751108517;
```

```
        float STDentropyPitch = (entropyPitch - (float)-1389806.10967541)/
(float)747534.543565487;
        float STDavgPitch = (avgPitch - (float)146.619381384184)/
(float)23.9101602181699;
        float STDminVol = (minVol - (float)17.6050792722449)/
(float)0.949961848829989;
        float STDmaxVol = (maxVol - (float)44.0012289954082)/
(float)3.08633542620264;
        float STDentropyVol = (entropyVol - (float)(-1721699.29684694))/
(float)310576.031743461;
        float STDavgVol = (avgVol - (float)29.1634195416836)/
(float)4.03255748480661;
        float STDMfcc = (Mfcc - (float)(-1.55039924441327))/
(float)0.158817693861086;


        float[][] lowLevFeaturesResult = {{STDinterrupt,STDspeakingPercentage,
STDMutualSilience,STDminPitch, STDmaxPitch, STDentropyPitch, STDavgPitch,
STDminVol, STDmaxVol, STDentropyVol, STDavgVol, STDMfcc}};
        String msg =  "Interrupt: " + String.valueOf(interrupt)
                    + ", speaking percentage: " + String.valueOf(speakingPercentage)
                    + ", mutual silience: " + String.valueOf(MutualSilience)
                    + ", min pitch: " + String.valueOf(minPitch)
                    + ", max pitch: " + String.valueOf(maxPitch)
                    + ", entropy pitch: " + String.valueOf(entropyPitch)
                    + ", mean pitch: " + String.valueOf(avgPitch)
                    + ", min vol: " + String.valueOf(minVol)
                    + ", max vol: " + String.valueOf(maxVol)
                    + ", entropy vol: " + String.valueOf(entropyVol)
                    + ", mean vol: " + String.valueOf(avgVol)
                    + ", mfcc: " + String.valueOf(Mfcc) ;
```

```
        //Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();
        return lowLevFeaturesResult;
    }


    private void train() {
        // Svm training
        int kernelType = 2; // Radial basis function
        int cost = 1; // Cost - C 4
        int isProb = 0;
        float gamma = 0.1f; // Gamma -r 0.25

        //for level of dominance
        String trainingFileLoc1 = Environment.getExternalStorageDirectory()+"/
training_set1";
        String modelFileLoc1 = Environment.getExternalStorageDirectory()+"/model1";
        String trainingFileLoc2 = Environment.getExternalStorageDirectory()+"/
training_set2";
        String modelFileLoc2 = Environment.getExternalStorageDirectory()+"/model2";
        String trainingFileLoc3 = Environment.getExternalStorageDirectory()+"/
training_set3";
        String modelFileLoc3 = Environment.getExternalStorageDirectory()+"/model3";

        //for level of agreement
        String trainingFileLoc4 = Environment.getExternalStorageDirectory()+"/
training_set4";
        String modelFileLoc4 = Environment.getExternalStorageDirectory()+"/model4";
        String trainingFileLoc5 = Environment.getExternalStorageDirectory()+"/
training_set5";
        String modelFileLoc5 = Environment.getExternalStorageDirectory()+"/model5";
        String trainingFileLoc6 = Environment.getExternalStorageDirectory()+"/
```

```
training_set6";
        String modelFileLoc6 = Environment.getExternalStorageDirectory()+"/model6";


        if (    trainClassifierNative(trainingFileLoc1, kernelType, cost, gamma, isProb,
modelFileLoc1) == -1 ||
                trainClassifierNative(trainingFileLoc2, kernelType, cost, gamma, isProb,
modelFileLoc2) == -1 ||
                trainClassifierNative(trainingFileLoc3, kernelType, cost, gamma, isProb,
modelFileLoc3) == -1 ||
                trainClassifierNative(trainingFileLoc4, kernelType, cost, gamma, isProb,
modelFileLoc4) == -1 ||
                trainClassifierNative(trainingFileLoc5, kernelType, cost, gamma, isProb,
modelFileLoc5) == -1 ||
                trainClassifierNative(trainingFileLoc6, kernelType, cost, gamma, isProb,
modelFileLoc6) == -1) {
            Log.d(TAG, "training err");
            finish();
        }
        Toast.makeText(this, "Training is done", Toast.LENGTH_SHORT).show();
    }

    /**
     * classify generate labels for features.
     * Return:
     *   -1: Error
     *   0: Correct
     */
    public int callSVM(float values[][], int indices[][], int groundTruth[], int isProb,
String modelFile,
                       int labels[], double probs[]) {
```

```
// SVM type
//Log.d(TAG, "callSVM");
final int C_SVC = 0;
final int NU_SVC = 1;
final int ONE_CLASS_SVM = 2;
final int EPSILON_SVR = 3;
final int NU_SVR = 4;

// For accuracy calculation
int correct = 0;
int total = 0;
float error = 0;
float sump = 0, sumt = 0, sumpp = 0, sumtt = 0, sumpt = 0;
float MSE, SCC, accuracy;

int num = values.length;
int svm_type = C_SVC;
if (num != indices.length)
   return -1;
// If isProb is true, you need to pass in a real double array for probability array
int r = doClassificationNative(values, indices, isProb, modelFile, labels, probs);

// Calculate accuracy
if (groundTruth != null) {
   if (groundTruth.length != indices.length) {
      Log.d(TAG, "Ground Truth.length != indices.length");
      return -1;

   }
   for (int i = 0; i < num; i++) {
      int predict_label = labels[i];
```

```
            int target_label = groundTruth[i];
            if(predict_label == target_label)
                ++correct;
            error += (predict_label-target_label)*(predict_label-target_label);
            sump += predict_label;
            sumt += target_label;
            sumpp += predict_label*predict_label;
            sumtt += target_label*target_label;
            sumpt += predict_label*target_label;
            ++total;
        }

        if (svm_type==NU_SVR || svm_type==EPSILON_SVR)
        {
            MSE = error/total; // Mean square error
            SCC = ((total*sumpt-sump*sumt)*(total*sumpt-sump*sumt)) /
((total*sumpp-sump*sump)*(total*sumtt-sumt*sumt)); // Squared correlation
coefficient
        }
        accuracy = (float)correct/total*100;
        Log.d(TAG, "Classification accuracy is " + accuracy);
    }

    return r;
  }

  private void classify(float values[][]) {
    // Svm classification
    int[][] indices = {
        {1,2,3,4,5,6,7,8,9,10,11,12}
    };
```

```
        int[] groundTruth = null;
        int[] labels1 = new int[1];
        int[] labels2 = new int[1];
        int[] labels3 = new int[1];
        int[] labels4 = new int[1];
        int[] labels5 = new int[1];
        int[] labels6 = new int[1];
        double[] probs = new double[1];
        int label_dom = 0;
        int label_agr = 0;
        int isProb = 0; // Not probability prediction
        String modelFileLoc1 = Environment.getExternalStorageDirectory()+"/model1";
        String modelFileLoc2 = Environment.getExternalStorageDirectory()+"/model2";
        String modelFileLoc3 = Environment.getExternalStorageDirectory()+"/model3";
        String modelFileLoc4 = Environment.getExternalStorageDirectory()+"/model4";
        String modelFileLoc5 = Environment.getExternalStorageDirectory()+"/model5";
        String modelFileLoc6 = Environment.getExternalStorageDirectory()+"/model6";
        //String modelFileLoc1 =
Environment.getExternalStorageDirectory().getAbsolutePath()+"/storage/extSdCard/
sociofeedback/model1";
//      String modelFileLoc1 = Environment.getExternalStorageDirectory()+
File.separator + "/sociofeedback/model1";
//      String modelFileLoc2 = Environment.getExternalStorageDirectory()+
File.separator + "/sociofeedback/model2";
//      String modelFileLoc3 = Environment.getExternalStorageDirectory()+
File.separator + "/sociofeedback/model3";
        if (    callSVM(values, indices, groundTruth, isProb, modelFileLoc1, labels1,
probs) != 0
                ||callSVM(values, indices, groundTruth, isProb, modelFileLoc2, labels2,
probs) != 0
                ||callSVM(values, indices, groundTruth, isProb, modelFileLoc3, labels3,
```

```
probs) != 0
            ||callSVM(values, indices, groundTruth, isProb, modelFileLoc4, labels4,
probs) != 0
            ||callSVM(values, indices, groundTruth, isProb, modelFileLoc5, labels5,
probs) != 0
            ||callSVM(values, indices, groundTruth, isProb, modelFileLoc6, labels6,
probs) != 0) {
         Log.d(TAG, "Classification is incorrect");
      }
      else {
         String m1 = "",m2 = "", m3 = "", m4 = "",m5 = "", m6 = "", m = "", mm = "";
         Random rand = new Random();
         int value = rand.nextInt(50);
         for (int l : labels1)  //   -> 1v2
            m1 += l + "; ";
         for (int l : labels2)  //   -> 1v3
            m2 += l + "; ";
         for (int l : labels3)  //   -> 2v3
            m3 += l + "; ";

         for (int l : labels4)  //   -> 1v2
            m4 += l + "; ";
         for (int l : labels5)  //   -> 1v3
            m5 += l + "; ";
         for (int l : labels6)  //   -> 2v3
            m6 += l + "; ";

         for (int i = 0; i < labels1.length; i++) {
            if (labels1[i] == labels2[i]) {
               m += labels1[i] ;//+ ", ";
               label_dom = labels1[i];
```

```
    }else if (labels1[i] == labels3[i]){
        m += labels1[i] ;//+ ", ";
        label_dom = labels1[i];
    }else if (labels3[i] == labels2[i]){
        m += labels2[i] ; // + ", ";
        label_dom = labels2[i];
    }else {
        label_dom = rand.nextInt(3)+1;
        //Integer.toString(label_dom);
    }
}

for (int i = 0; i < labels4.length; i++) {
    if (labels4[i] == labels5[i]) {
        mm += labels4[i] ;//+ ", ";
        label_agr = labels4[i];
    }else if (labels4[i] == labels6[i]){
        mm += labels4[i] ;//+ ", ";
        label_agr = labels4[i];
    }else if (labels5[i] == labels6[i]){
        mm += labels5[i] ; // + ", ";
        label_agr = labels5[i];
    }else {
        label_agr = rand.nextInt(3)+1;
        //Integer.toString(label_agr);
    }
}


if(label_dom == 1){
    m += "-Low dominance";
```

```
        }else if(label_dom == 2){
            m += "-Medium dominance";
        }else if (label_dom == 3){
            m += "-High dominance";
        }

        if(label_agr == 1){
            mm += "-Low agreement";
        }else if(label_agr == 2){
            mm += "-Medium agreement";
        }else if (label_agr == 3){
            mm += "-High agreement";
        }
        Toast.makeText(this, "Classification is done, the final result is " + m + ", " +
mm, Toast.LENGTH_LONG).show();

    }
  }


  private static void verifyStoragePermissions(Activity activity) {
    // Check if we have write permission
    int permission = ActivityCompat.checkSelfPermission(activity,
Manifest.permission.WRITE_EXTERNAL_STORAGE);

    if (permission != PackageManager.PERMISSION_GRANTED) {
      // We don't have permission so prompt the user
      ActivityCompat.requestPermissions(
          activity,
          PERMISSIONS_STORAGE,
          REQUEST_EXTERNAL_STORAGE
```

```
            );
        }
    }
}
```

SettingsActivity.java

```
package com.example.user.pitchvolfeb;

/**
 * Created by Sailin on 11/2/16.
 */

import android.app.Activity;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import com.kyleduo.switchbutton.SwitchButton;


public class SettingsActivity extends Activity implements View.OnClickListener{
    private SwitchButton volumeChk, pitchChk, speechrateChk;
    private static boolean isChkV, isChkP, isChkSR;

    static final String STATE_VOLUME = "volume selection";
    static final String STATE_PITCH = "pitch selection";
    static final String STATE_SPEECHRATE = "speech rate selection";
    private static final String Preferences = "setting";
```

```
private Button BtnDone;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_settings);

    volumeChk = (SwitchButton) findViewById(R.id.checkbox_v);
    pitchChk = (SwitchButton) findViewById(R.id.checkbox_p);
    speechrateChk = (SwitchButton) findViewById(R.id.checkbox_sr);



    SharedPreferences settings = getSharedPreferences(Preferences, 0);
    Boolean restoredVol = settings.getBoolean(STATE_VOLUME, true);
    Boolean restoredPitch = settings.getBoolean(STATE_PITCH, true);
    Boolean restoredSR = settings.getBoolean(STATE_SPEECHRATE, true);

    volumeChk.setChecked(restoredVol);
    pitchChk.setChecked(restoredPitch);
    speechrateChk.setChecked(restoredSR);

    BtnDone = (Button)findViewById(R.id.done);
    BtnDone.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            onStop();
            finish();
        }
    });

}
```

```
    @Override
    protected void onStop(){
        super.onStop();

        // We need an Editor object to make preference changes.
        // All objects are from android.context.Context
        SharedPreferences settings = getSharedPreferences(Preferences, 0);
        SharedPreferences.Editor editor = settings.edit();
        editor.putBoolean(STATE_VOLUME, isChkV);
        editor.putBoolean(STATE_PITCH, isChkP);
        editor.putBoolean(STATE_SPEECHRATE, isChkSR);

        // Commit the edits!
        editor.commit();
        Toast.makeText(SettingsActivity.this, "Settings saved",
Toast.LENGTH_SHORT).show();
    }


    @Override
    public void onClick(View v){
        int id = v.getId();
        switch (id){
            case R.id.checkbox_v:
                volumeChk.setChecked(!volumeChk.isChecked());
                volumeChk.toggle();
                isChkV=!isChkV;
                break;
            case R.id.checkbox_p:
                pitchChk.setChecked(!pitchChk.isChecked());
                pitchChk.toggle();
```

```
            isChkP=!isChkP;
            break;
        case R.id.checkbox_sr:
            speechrateChk.setChecked(!speechrateChk.isChecked());
            speechrateChk.toggle();
            isChkSR=!isChkSR;
            break;
        default:break;

    }
}
/*
public void onCheckboxClicked(View view) {
    // Is the view now checked?
    boolean checked = ((CheckBox) view).isChecked();

    // Check which checkbox was clicked
    switch(view.getId()) {
        case R.id.checkbox_v:
            if (checked) {
                // enable volume detection
                isChkV = true;
            }else {
                // disable volume detection
                isChkV = false;
            }
            break;
        case R.id.checkbox_p:
            if (checked) {
                // enable pitch detection
                isChkP = true;
```

```
            }else{
                // disable pitch detection
                isChkP = false;
            }
            break;
        case R.id.checkbox_sr:
            if (checked) {
                // enable speech rate detection
                isChkSR = true;
            }else {
                // disable speech rate detection
                isChkSR = false;
            }
            break;
        // TODO: Veggie sandwich
    }
  }
  */
}
```

AboutActivity.java

```
package com.example.user.pitchvolfeb;

/**
 * Created by indrihartanto on 11/2/15.
 */

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
```

```
public class AboutActivity extends Activity {
    private Button BtnDone;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_about);

        BtnDone = (Button)findViewById(R.id.done);
        BtnDone.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                finish();
            }
        });
    }
}
```

LAYOUT

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000000">



    <ImageView
        android:id="@+id/background"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
```

```
    android:src="@drawable/background" />


  <ImageView
    android:id="@+id/title"
    android:layout_marginTop="175dp"
    android:layout_width="match_parent"
    android:layout_height="48dp"
    android:src="@drawable/title2"
    />
  <com.example.user.pitchvolfeb.SecretTextView
    android:id="@+id/description"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="270dp"
    android:layout_centerHorizontal="true"
    android:text="To become a better communicator."
    android:textSize="16sp"
    android:typeface="sans"
    android:textColor="@android:color/white"
    />


  <Button
    android:id="@+id/start"
    android:layout_width="220dp"
    android:layout_height="40dp"
    android:text="Get Started"
    android:textColor="#4d4d4d"
    android:typeface="sans"
    android:textSize="15sp"
    android:layout_marginTop="40dp"
```

```
    android:background="#cccccc"
    android:alpha="0.30"
    android:layout_below="@+id/description"
    android:layout_centerHorizontal="true" />

<ImageButton
    android:id="@+id/settings"
    android:layout_width="35dp"
    android:layout_height="35dp"
    android:adjustViewBounds="true"
    android:scaleType="centerCrop"
    android:background="@android:color/transparent"
    android:text="Settings"
    android:src="@drawable/icon_settings"
    android:layout_alignTop="@+id/aboutus"
    android:layout_alignRight="@+id/start"
    android:layout_alignEnd="@+id/start" />

<ImageButton
    android:id="@+id/aboutus"
    android:layout_width="35dp"
    android:layout_height="35dp"
    android:adjustViewBounds="true"
    android:scaleType="centerCrop"
    android:src="@drawable/icon_profile"
    android:background="@android:color/transparent"
    android:text="About Us"
    android:layout_marginTop="33dp"
    android:layout_below="@+id/start"
    android:layout_alignLeft="@+id/start"
    android:layout_alignStart="@+id/start" />
```

```
</RelativeLayout>
```

activity_sociometrics.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#262626">


    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/layout_top">

        <ImageView
            android:layout_width="match_parent"
            android:layout_height="2dp"
            android:background="#262626"/>

        <FrameLayout
            android:layout_width="fill_parent"
            android:layout_height="46dp"
            android:background="#535353">

            <Button
                android:layout_width="25dp"
                android:layout_height="25dp"
```

```
            android:background="@drawable/titlebar_menu_selector"
            android:id="@+id/title_bar_left_menu"
            android:layout_gravity="left|center_vertical"
            android:layout_marginLeft="10dp"/>

        <ImageView
            android:layout_marginTop="10dp"
            android:layout_centerHorizontal="true"
            android:layout_centerVertical="true"
            android:layout_width="match_parent"
            android:layout_height="32dp"
            android:src="@drawable/title2"
            />

    </FrameLayout>

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="2dp"
        android:background="#1a1a1a"/>
</LinearLayout>

<ImageButton
    android:id="@+id/startfeedback"
    android:layout_width="250dp"
    android:layout_height="100dp"
    android:layout_marginTop="270dp"
    android:layout_marginLeft="180dp"
    android:src="@drawable/off"

    />
```

```
<ImageView
    android:id="@+id/imgConvState"
    android:layout_width="30dp"
    android:layout_height="30dp"
    android:layout_marginTop="300dp"
    android:layout_marginLeft="25dp"
    android:src="@drawable/conversation"/>

<TextView
    android:id="@+id/textConvState"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="308dp"
    android:layout_marginLeft="62dp"
    android:text="CONVERSATION STATE"
    android:textColor="#C0C0C0"
    android:textSize="13dp"
    android:textStyle="bold"/>

<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="40dp">

    <TextView
        android:id="@+id/volume"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="VOL1"
        android:layout_marginBottom="28dp"
```

```
    android:textColor="#ff6666"
    android:layout_marginTop="27dp"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
    />

<com.akexorcist.roundcornerprogressbar.IconRoundCornerProgressBar
    android:id="@+id/progressBarV"
    android:layout_width="295dp"
    android:layout_height="20dp"
    android:layout_marginBottom="28dp"
    android:layout_marginTop="26dp"
    android:layout_marginLeft="48dp"
    android:layout_marginRight="10dp"
    app:rcBackgroundPadding="2dp"
    app:rcIconPadding="3dp"
    app:rcIconSize="21dp"
    app:rcIconSrc="@mipmap/ic_volume"
    app:rcMax="100"
    app:rcProgress="90"
    app:rcRadius="5dp" />

<TextView
    android:id="@+id/volume2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="VOL2"
    android:layout_marginBottom="28dp"
    android:textColor="#80aaff"
    android:layout_marginTop="57dp"
    android:layout_marginLeft="10dp"
```

```
        android:layout_marginRight="10dp"
        />

    <com.akexorcist.roundcornerprogressbar.IconRoundCornerProgressBar
        android:id="@+id/progressBarV2"
        android:layout_width="295dp"
        android:layout_height="20dp"
        android:layout_marginBottom="28dp"
        android:layout_marginTop="56dp"
        android:layout_marginLeft="48dp"
        android:layout_marginRight="10dp"
        app:rcBackgroundPadding="2dp"
        app:rcIconPadding="3dp"
        app:rcIconSize="21dp"
        app:rcIconSrc="@mipmap/ic_volume"
        app:rcMax="100"
        app:rcProgress="90"
        app:rcRadius="5dp" />

    <!--TextView
        android:id="@+id/textViewV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceSmall"
        android:text="0%"
        android:layout_marginTop="20dp"
        android:layout_marginLeft="350dp"
        android:layout_alignBottom="@+id/volume" /-->
</FrameLayout>
```

```
<com.jjoe64.graphview.GraphView
    android:id="@+id/graph"
    android:layout_width="wrap_content"
    android:layout_height="150dp"
    android:layout_marginTop="130dp"
    android:layout_marginLeft="10dp"
    android:layout_centerHorizontal="true"
    android:layout_marginRight="20dp"
    />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="User 1"
    android:textColor="#ff6666"
    android:textStyle="bold"
    android:layout_marginTop="370dp"
    android:layout_marginLeft="20dp"/>

<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="320dp"
    android:layout_marginLeft="100dp"
    android:id="@+id/layout_content">


    <TextView
    android:id="@+id/lbl_tvMessage"
    android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
android:text="Pitch in Hz:"
android:textColor="#C0C0C0"
android:textSize="12dp"
android:layout_marginBottom="28dp"
android:layout_marginTop="50dp"
android:layout_marginLeft="20dp"
android:layout_marginRight="10dp"
/>

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/tvMessage"
android:text="0 Hz"
android:textSize="12dp"
android:layout_marginLeft="130dp"
android:layout_marginTop="50dp"
android:textColor="#C0C0C0"/>


<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Status:"
android:textSize="12dp"
android:id="@+id/lbl_result_txtview"
android:layout_marginLeft="20dp"
android:layout_marginTop="65dp"
android:textColor="#C0C0C0"/>
```

```
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/result_txtview"
android:text="silence"
android:textSize="12dp"
android:layout_marginTop="65dp"
android:layout_marginLeft="130dp"
android:textColor="#C0C0C0"/>

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Volume in dBSPL:"
android:textSize="12dp"
android:id="@+id/lbl_tvdBlevel"
android:layout_marginTop="80dp"
android:layout_marginLeft="20dp"
android:textColor="#C0C0C0"/>

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/tvdBlevel"
android:text="0 dB"
android:textSize="12dp"
android:layout_marginLeft="130dp"
android:layout_marginTop="80dp"
android:textColor="#C0C0C0"/>

<TextView
```

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Volume:"
android:textSize="12dp"
android:id="@+id/lbl_res_tvdBlevel"
android:layout_marginLeft="20dp"
android:layout_marginTop="95dp"
android:textColor="#C0C0C0"/>

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/result_tvdBlevel"
android:text="0"
android:textSize="12dp"
android:layout_marginLeft="130dp"
android:layout_marginTop="95dp"
android:textColor="#C0C0C0"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Speaking %:"
    android:textSize="12dp"
    android:id="@+id/lbl_res_speakingPer"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="110dp"
    android:textColor="#C0C0C0"/>

<TextView
    android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:id="@+id/result_speakingPer"
        android:text="0 %"
        android:textSize="12dp"
        android:layout_marginLeft="130dp"
        android:layout_marginTop="110dp"
        android:textColor="#C0C0C0"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="MFCC:"
        android:textSize="12dp"
        android:id="@+id/lbl_res_mfcc"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="125dp"
        android:textColor="#C0C0C0"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/result_mfcc"
        android:text="0"
        android:textSize="12dp"
        android:layout_marginLeft="130dp"
        android:layout_marginTop="125dp"
        android:textColor="#C0C0C0"/>


</FrameLayout>
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="User 2"
    android:textColor="#80aaff"
    android:textStyle="bold"
    android:layout_marginTop="470dp"
    android:layout_marginLeft="20dp"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="(You)"
    android:textColor="#80aaff"
    android:textStyle="bold"
    android:layout_marginTop="485dp"
    android:layout_marginLeft="20dp"/>

<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="420dp"
    android:layout_marginLeft="100dp"
    android:id="@+id/layout_content2">


    <TextView
        android:id="@+id/lbl_tvMessage2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Pitch in Hz:"
```

```
    android:textColor="#C0C0C0"
    android:textSize="12dp"
    android:layout_marginBottom="28dp"
    android:layout_marginTop="50dp"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="10dp"
    />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/tvMessage2"
    android:text="0 Hz"
    android:textSize="12dp"
    android:layout_marginLeft="130dp"
    android:layout_marginTop="50dp"
    android:textColor="#C0C0C0"/>


<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Status:"
    android:textSize="12dp"
    android:id="@+id/lbl_result_txtview2"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="65dp"
    android:textColor="#C0C0C0"/>

<TextView
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:id="@+id/result_txtview2"
    android:text="silence"
    android:textSize="12dp"
    android:layout_marginTop="65dp"
    android:layout_marginLeft="130dp"
    android:textColor="#C0C0C0"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Volume in dBSPL:"
    android:textSize="12dp"
    android:id="@+id/lbl_tvdBlevel2"
    android:layout_marginTop="80dp"
    android:layout_marginLeft="20dp"
    android:textColor="#C0C0C0"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/tvdBlevel2"
    android:text="0 dB"
    android:textSize="12dp"
    android:layout_marginLeft="130dp"
    android:layout_marginTop="80dp"
    android:textColor="#C0C0C0"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:text="Volume:"
    android:textSize="12dp"
    android:id="@+id/lbl_res_tvdBlevel2"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="95dp"
    android:textColor="#C0C0C0"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/result_tvdBlevel2"
    android:text="0"
    android:textSize="12dp"
    android:layout_marginLeft="130dp"
    android:layout_marginTop="95dp"
    android:textColor="#C0C0C0"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Speaking %:"
    android:textSize="12dp"
    android:id="@+id/lbl_res_speakingPer2"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="110dp"
    android:textColor="#C0C0C0"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/result_speakingPer2"
```

```
    android:text="0 %"
    android:textSize="12dp"
    android:layout_marginLeft="130dp"
    android:layout_marginTop="110dp"
    android:textColor="#C0C0C0"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="MFCC:"
    android:textSize="12dp"
    android:id="@+id/lbl_res_mfcc2"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="125dp"
    android:textColor="#C0C0C0"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/result_mfcc2"
    android:text="0"
    android:textSize="12dp"
    android:layout_marginLeft="130dp"
    android:layout_marginTop="125dp"
    android:textColor="#C0C0C0"/>
<!--TextView
    android:id="@+id/text2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Client-side result"
    android:textColor="#C0C0C0"
```

```
        android:layout_marginTop="115dp"
        android:textSize="16dp"/-->
    </FrameLayout>
</RelativeLayout>
```

activity_settings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SociometricsActivity"

    >

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/layout_top">

        <ImageView
            android:layout_width="match_parent"
            android:layout_height="2dp"
            android:background="#262626"/>


        <FrameLayout
            android:layout_width="fill_parent"
            android:layout_height="46dp"
```

```
        android:background="#535353">
        <TextView
            android:id="@+id/settings"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Settings"
            android:layout_gravity="center_horizontal|center_vertical"
            android:textColor="#C0C0C0"
            android:textSize="24dp"
            />

        <Button
            android:id="@+id/done"
            android:layout_width="40dp"
            android:layout_height="25dp"
            android:background="@android:color/transparent"
            android:text="Done"
            android:textAllCaps="false"
            android:textColor="#C0C0C0"
            android:layout_gravity="left|center_vertical"
            android:layout_marginLeft="10dp"/>


    </FrameLayout>

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="2dp"
        android:background="#1a1a1a"/>
</LinearLayout>
```

```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="46dp"
    android:id="@+id/layout_general">

    <FrameLayout
        android:layout_width="fill_parent"
        android:layout_height="40dp"
        android:background="#e6e6e6">
        <TextView
            android:id="@+id/General"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="14dp"
            android:text="Low level features"
            android:layout_gravity="left|center_vertical"
            android:textColor="#0d0d0d"
            android:textStyle="bold"
            android:textSize="20dp"
            />
    </FrameLayout>

</LinearLayout>


<FrameLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
```

```
android:layout_height="200dp"
android:layout_marginTop="86dp"
android:id="@+id/layout_generalsettings">



<TextView
    android:id="@+id/volume"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:layout_marginLeft="14dp"
    android:text="Volume"
    android:textSize="18dp"
    android:textColor="#595959"
    />

<com.kyleduo.switchbutton.SwitchButton
    android:id="@+id/checkbox_v"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:layout_marginLeft="14dp"
    android:layout_gravity="right"
    android:layout_marginRight="14dp"
    android:onClick="onClick"
    />

<ImageView
    android:layout_width="340dp"
    android:layout_height="1dp"
```

```
    android:layout_marginTop="40dp"
    android:background="#e6e6e6"
    android:layout_gravity="center_horizontal"/>



<TextView
    android:id="@+id/pitch"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="48dp"
    android:layout_marginLeft="14dp"
    android:text="Pitch"
    android:textSize="18dp"
    android:textColor="#595959"
    />

<com.kyleduo.switchbutton.SwitchButton
    android:id="@+id/checkbox_p"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="48dp"
    android:layout_marginLeft="14dp"
    android:layout_gravity="right"
    android:layout_marginRight="14dp"
    android:onClick="onClick"
    />

<ImageView
    android:layout_width="340dp"
```

```
        android:layout_height="1dp"
        android:layout_marginTop="80dp"
        android:background="#e6e6e6"
        android:layout_gravity="center_horizontal"/>



    <TextView
        android:id="@+id/speechrate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="88dp"
        android:layout_marginLeft="14dp"
        android:text="Speaking Percentage"
        android:textSize="18dp"
        android:textColor="#595959"
        />

    <com.kyleduo.switchbutton.SwitchButton
        android:id="@+id/checkbox_sr"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="88dp"
        android:layout_marginLeft="14dp"
        android:layout_gravity="right"
        android:layout_marginRight="14dp"
        android:onClick="onClick"/>


</FrameLayout>
```

```
<FrameLayout
    android:layout_width="fill_parent"
    android:layout_height="40dp"
    android:layout_marginTop="206dp"
    android:background="#e6e6e6">
    <TextView
        android:id="@+id/AppInfo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="14dp"
        android:text="App Information"
        android:layout_gravity="left|center_vertical"
        android:textColor="#0d0d0d"
        android:textStyle="bold"
        android:textSize="20dp"
        />
</FrameLayout>

<TextView
    android:id="@+id/appinfo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="254dp"
    android:layout_marginLeft="14dp"
    android:text="v2.0 (2016-03-08 20:47:28)"
    android:textSize="18dp"
    android:textColor="#595959"
    />
```

```
    <TextView
        android:id="@+id/copyright"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="470dp"
        android:layout_centerHorizontal="true"
        android:text="Copyright @ 2016 NTU"
        android:textSize="14dp"
        android:textColor="#595959"
        />
</RelativeLayout>
```

activity_about.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SociometricsActivity"

    >

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/layout_top">

        <ImageView
```

```
    android:layout_width="match_parent"
    android:layout_height="2dp"
    android:background="#262626"/>


<FrameLayout
    android:layout_width="fill_parent"
    android:layout_height="46dp"
    android:background="#535353">
    <TextView
        android:id="@+id/settings"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Help"
        android:layout_gravity="center_horizontal|center_vertical"
        android:textColor="#C0C0C0"
        android:textSize="24dp"
        />

    <Button
        android:id="@+id/done"
        android:layout_width="40dp"
        android:layout_height="25dp"
        android:background="@android:color/transparent"
        android:text="Done"
        android:textAllCaps="false"
        android:textColor="#C0C0C0"
        android:layout_gravity="left|center_vertical"
        android:layout_marginLeft="10dp"/>
```

```
    </FrameLayout>

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="2dp"
        android:background="#1a1a1a"/>
</LinearLayout>


<LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="46dp"
    android:id="@+id/layout_general">

    <FrameLayout
        android:layout_width="fill_parent"
        android:layout_height="40dp"
        android:background="#e6e6e6">
        <TextView
            android:id="@+id/General"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="14dp"
            android:text="Guide to the app"
            android:layout_gravity="left|center_vertical"
            android:textColor="#0d0d0d"
            android:textStyle="bold"
            android:textSize="20dp"
            />
```

```
    </FrameLayout>

</LinearLayout>


<FrameLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="240dp"
    android:layout_marginTop="86dp"
    android:id="@+id/layout_generalsettings">



    <TextView
        android:id="@+id/mobile"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:layout_marginLeft="14dp"
        android:text="Mobile version"
        android:textSize="18dp"
        android:textColor="#595959"
        />

    <TextView
        android:id="@+id/mobileDes"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="33dp"
        android:layout_marginLeft="14dp"
```

```
        android:text="The new version of the app brings a more personalised
experience and operate independently on the phone.Swipe right to access the menu or
tab the menu icon on the top left of the phone. Select the low level features you would
like to monitor in settings."
        android:textSize="12dp"
        android:textColor="#595959"
        />

    <ImageView
        android:layout_width="340dp"
        android:layout_height="1dp"
        android:layout_marginTop="120dp"
        android:background="#e6e6e6"
        android:layout_gravity="center_horizontal"/>


    <TextView
        android:id="@+id/glass"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="128dp"
        android:layout_marginLeft="14dp"
        android:text="Glass version"
        android:textSize="18dp"
        android:textColor="#595959"
        />

    <TextView
        android:id="@+id/glassDes"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
    android:layout_marginTop="153dp"
    android:layout_marginLeft="14dp"
    android:text="This app also runs on Google Glass. "
    android:textSize="12dp"
    android:textColor="#595959"
    />

<ImageView
    android:id="@+id/imgGlass"
    android:layout_width="215dp"
    android:layout_height="133dp"
    android:layout_marginTop="132dp"
    android:src="@drawable/googleglass"
     />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Swipe forward: About us"
    android:textSize="10dp"
    android:layout_marginTop="180dp"
    android:layout_marginLeft="230dp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Swipe back:      Settings"
    android:textSize="10dp"
    android:layout_marginTop="190dp"
    android:layout_marginLeft="230dp" />
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Swipe down:     Quit"
        android:textSize="10dp"
        android:layout_marginTop="205dp"
        android:layout_marginLeft="230dp"/>

</FrameLayout>


<FrameLayout
    android:layout_width="fill_parent"
    android:layout_height="40dp"
    android:layout_marginTop="316dp"
    android:background="#e6e6e6">
    <TextView
        android:id="@+id/AppInfo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="14dp"
        android:text="About us"
        android:layout_gravity="left|center_vertical"
        android:textColor="#0d0d0d"
        android:textStyle="bold"
        android:textSize="20dp"
        />
</FrameLayout>

<TextView
    android:id="@+id/appinfo"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="364dp"
        android:layout_marginLeft="14dp"
        android:text="This app is developed by a FYP student and a master student from NTU, supervised by Assoc Prof Justin Dauwels. Previously, sociofeedback is obtained using Matlab. Check out our demo on Youtube: https://www.youtube.com/watch?v=3uT9O3MqUOg. "
        android:textSize="12dp"
        android:textColor="#595959"
        />
    <TextView
        android:id="@+id/copyright"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="470dp"
        android:layout_centerHorizontal="true"
        android:text="Copyright @ 2016 NTU"
        android:textSize="14dp"
        android:textColor="#595959"
        />
</RelativeLayout>
```

SERVER

index.js

```
/**
 * Created by sailin.
 */

var app = require('express')();
var http = require('http').Server(app);
```

```
var io = require('socket.io')(http);


app.get('/', function(req, res){
  res.sendFile(__dirname + '/index.html');
});

io.on('connection', function(socket){
  console.log('a user connected' + socket.id);
  socket.on('disconnect', function(){
      console.log('user disconnected'+ socket.id);
    });
  socket.on('new message', function(message){
      console.log('volume, pitch & MFCC: ' + message)
      io.emit('new message', message);
  });

});

http.listen(3000, function(){
  console.log('listening on *:3000');
});
```