

Voice Activity Recording (VAR) Tool

General Copyright statement

Voice Activity Recording (VAR) Tool V1.0

Copyright (C) 2008 Biomedical Informatics (BMI), St George's University of London. Citations to the toolkit should be as 'de Lusignan S, Kumarapeli P, Chan T, Pflug B, van Vlymen J, Jones B, Freeman GK. The ALFA (activity Log Files Aggregation) toolkit: a method for precise observation of the consultation. <*J Med Internet Res* 2008; ..>.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

Any redistributions/public demonstration/publication/use of this software should acknowledge the original developers mentioned above as the Copyright holders. It is also encouraged to inform the original developers about such activities in order to assure the consistency among future software versions and to keep potential users up-to-date.

Source codes

Language: Sun Java 1.6

Platform: Microsoft Windows/Linux/Sun Solaris

Src1: var.java

```
package var;
```

```
/**
```

```
*Voice Activity Recording (VAR) Tool V1.0
```

```
*
```

```
*Copyright (C) 2008 Biomedical Informatics (BMI), St George's University of London. Citations to  
*the toolkit should be as 'de Lusignan S, Kumarapeli P, Chan T, Pflug B, van Vlymen J, Jones B,  
*Freeman GK. The ALFA (activity Log Files Aggregation) toolkit: a method for precise observation  
*of the consultation. <J Med Internet Res 2008; ..>.
```

```
*
```

```
*This program is free software: you can redistribute it and/or modify it under the terms of the  
*GNU General Public License as published by the Free Software Foundation, either version 3 of  
*the License, or (at your option) any later version.
```

*This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
*without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
*PURPOSE. See the GNU General Public License for more details.
*
*You should have received a copy of the GNU General Public License along with this program. If
*not, see <<http://www.gnu.org/licenses/>>.
*
*/

```
import java.util.Vector;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.Line;
import javax.sound.sampled.LineUnavailableException;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Container;
import javax.sound.sampled.Mixer;
import java.awt.geom.GeneralPath;
import java.awt.image.BufferedImage;
import javax.swing.event.*;
import javax.swing.JTextArea;
import java.text.DecimalFormat;
import javax.swing.JPanel;
import javax.swing.JFrame;
import javax.swing.JSlider;
import javax.swing.JOptionPane;
import javax.swing.border.TitledBorder;
import javax.sound.sampled.TargetDataLine;
import java.awt.GridLayout;
import java.awt.BorderLayout;

public class var {

    public static void main(String[] args) {
        int count1=0;
        boolean lineFound = false;
        Mixer.Info[] channels = AudioSystem.getMixerInfo();
        JFrame frame1 = new JFrame("Voice Activity Recorder (VAR)");
        frame1.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        Container c = frame1.getContentPane();
        c.setLayout(new GridLayout(0,1));

        JSlider sample = new JSlider(1,10,5);
        sample.setPaintLabels(true);
```

```
sample.setPaintTicks(true);
sample.setSnapToTicks(true);
sample.setMajorTickSpacing(1);
sample.setBorder(new TitledBorder("Sample size:"));
JSlider gain = new JSlider(1,10,5);
gain.setPaintLabels(true);
gain.setPaintTicks(true);
gain.setSnapToTicks(true);
gain.setMajorTickSpacing(1);
gain.setBorder(new TitledBorder("Gain level:"));

JSlider silence = new JSlider(1,10,5);
silence.setPaintLabels(true);
silence.setPaintTicks(true);
silence.setSnapToTicks(true);
silence.setMajorTickSpacing(1);
silence.setBorder(new TitledBorder("Silence level:"));

JPanel setPara = new JPanel(new BorderLayout());
setPara.add(sample, BorderLayout.CENTER);
setPara.add(gain, BorderLayout.SOUTH);
setPara.add(silence, BorderLayout.NORTH);

JPanel topBase = new JPanel(new BorderLayout());
topBase.add( setPara, BorderLayout.CENTER );
JPanel inputBase = new JPanel(new BorderLayout());
inputBase.add( topBase, BorderLayout.NORTH);

JOptionPane.showMessageDialog( c,inputBase,"Voice Activity Recorder
(VAR)",JOptionPane.PLAIN_MESSAGE);

StringBuffer channelInfo = new StringBuffer();
Vector lines = new Vector();
while (count1<channels.length)
{
Mixer mixer = AudioSystem.getMixer(channels[count1]);
try
{
Line[] allTargetLines = mixer.getTargetLines();
Line.Info[] allTLineInfos =
mixer.getTargetLineInfo();
for (int count2=0; count2<allTLineInfos.length; count2++)
{
try
{
```

```

        TargetDataLine line = (TargetDataLine)mixer.getLine(
allTLineInfos[count2] );
        lines.add( line );
        lineFound=true;
    }
    catch(ClassCastException cce)
    {
        channelInfo.append(allTLineInfos[count2]);
        // channelInfo.append(allTLineInfos[count2]);
        //channelInfo.append( "\n\tNote: Is a Data line, " +"but not
a TDL" );
    }
}
}
catch(LineUnavailableException lue)
{
    lue.printStackTrace();
}
count1++;
} // end while >> (count1<channels.length)

Vector allTracePanels = new Vector();
TargetDataLine line = (TargetDataLine)
lines.get(lines.size()-1);
AudioPlotPanel app = new AudioPlotPanel(line);

boolean add = allTracePanels.add(app);

// configure the trace panels and add them to the main container
for(int count2=0; count2<allTracePanels.size(); count2++)
{
    allTracePanels.get(count2);
    app.setSamplingSize( sample.getValue()*32 );
    app.setGain(gain.getValue()); //sGain.getValue()
    app.setsilence(silence.getValue());
    app.start();

    c.add( app );
}

JOptionPane.showMessageDialog(c,new
JTextArea(channelInfo.toString().substring(1),"DataLine
Information",JOptionPane.INFORMATION_MESSAGE);
    frame1.pack();
    frame1.setSize(640,480);
    frame1.setLocation(20,20);

```

```
        frame1.setVisible(true);
}/// end of main

}

class AudioPlotPanel extends JPanel implements Runnable
{

    TargetDataLine line;
    byte[] b;
    BufferedImage bImage1;
    BufferedImage bImage2;
    int offset;
    int channels;
    int frameSize;
    long lastGainChange = 0;
    long startTime;
    static int MAX_GAIN = 5;
    int gain;
    int silence;
    static int sample;

    int gradientScaling = -1;
    DecimalFormat df;
    int maxLevel = 1;

    AudioPlotPanel(TargetDataLine line)
    {
        df = new DecimalFormat("0.00000");
        this.line = line;
        try
        {
            line.open();
            line.start();
            AudioFormat af = line.getFormat();
            channels = af.getChannels();
            frameSize = af.getFrameSize();
        }
        catch(LineUnavailableException lue)
        {

        }

    }

    public void start()
    {
```

```
Thread t = new Thread(this);
startTime = System.currentTimeMillis();
t.start();
}

public void setSamplingSize(int sample)
{
    b = new byte[sample*frameSize];
    AudioPlotPanel.sample = sample;
}

public void setGain(int gain)
{
    this.gain = gain;
    lastGainChange = System.currentTimeMillis();
}

public void setsilence(int silence)
{
    this.silence = silence;
}

public void run()
{
    while(line!=null)
    {
        line.read(b,0,sample*frameSize);
        repaint();
        try
        {
            Thread.sleep(1000/60);
        }
        catch(InterruptedException ie)
        {
        }
    }
} // end while >> (line!=null)
} //end method >> public void run()

// draw the oscilloscope line
public void paintComponent(Graphics panel1)
{
    Graphics2D g1, g2;
    gradientScaling = (gradientScaling<1 ?getWidth()/10 : gradientScaling);
    if
(bImage1==null||bImage1.getWidth()!=getWidth()||bImage1.getHeight()!=getHeight() )
```

```
{
bImage1 = (BufferedImage)createImage(this.getWidth(),this.getHeight());
g1 = bImage1.createGraphics();
g1.setColor(Color.BLACK);
g1.fillRect(0,0,this.getWidth(), this.getHeight());
bImage2 = (BufferedImage)createImage(this.getWidth(),this.getHeight());
}

g1 = bImage1.createGraphics();
g1.drawImage(bImage2,0,0,this);

GeneralPath[] gp;
//gp = new GeneralPath[ 1 ];

int i2=0;
//gp[i2] = new GeneralPath(GeneralPath.WIND_EVEN_ODD,sample);
//gp[i2].moveTo(0,0);
double[] lastSignalSize = null;
double level = 0;
int count = 0;
for (int count1=0; count1<(int)(sample); count1++)
{
byte[] frameSample = new byte[frameSize];
for (int count2=0; count2<frameSize; count2++)
{
frameSample[count2] = b[(count1*4)+count2];
}

double total = 0;

for (int count2=0; count2<channels; count2++)
{
double signalChannelSize = frameToSignedDoubles( frameSample )[count2];
total += signalChannelSize;
level += Math.abs(signalChannelSize);
count++;
}

double average = total/channels;
if (lastSignalSize==null)
{
lastSignalSize = new double[1];
lastSignalSize[0] = average;
}
}
```

```
g2 = bImage2.createGraphics();
g2.fillRect(0,0,this.getWidth(), this.getHeight());
panel1.drawImage(bImage1,0,0,this);
double averageLevel = level/count;

//show the text for parameters
panel1.setColor(Color.CYAN);
panel1.drawString( "Voice Activity Recorder - pushpa@BMI",
getWidth()*(10/100),getHeight()- 10 );
panel1.setColor(Color.YELLOW);
panel1.drawString( "Average level: " + df.format(averageLevel), getWidth()-
200,getHeight()- 10 );
panel1.drawString( "Level: " + df.format(level), getWidth()-200,getHeight()- 30 );
panel1.setColor(Color.WHITE);
panel1.drawString( "Count: " + count, getWidth()-200,getHeight()- 50 );
panel1.drawString( "Sampling size: " + sample, getWidth()-200,getHeight()- 70 );
panel1.drawString( "Channels: " + channels, getWidth()-200,getHeight()- 90 );
panel1.drawString( "Gain level: " + gain, getWidth()-200,getHeight()- 110 );
panel1.drawString( "Silence level: " + silence, getWidth()-200,getHeight()- 130 );
}

public double[] frameToSignedDoubles(byte[] b)
{
double[] d = new double[channels];
for (int cc = 0; cc < channels; cc++)
{
d[cc] = (b[cc*2+1]*256 + (b[cc*2] & 0xFF))/32678.0;
}
return d;
} //end method >> public double[] frameToSignedDoubles(byte[] b)
}
```