浙江大学

本科实验报告

课程名称：数字视音频处理

姓　　名：卢涛

学　　院：计算机科学与技术学院

系：计算机科学与技术系

专　　业：计算机科学与技术学院

学　　号：3140102441

指导教师：杨莹春

2016 年 10 月 13 日

浙江大学实验报告

课程名称：数字视音频处理　　实验类型：综合

实验项目名称：Lab2：使用 voicebox 进行说话人识别

学生姓名：卢涛　专业：计算机科学与技术　学号：3140102441

实验地点：曹光彪西楼　　实验日期：2016 年 10 月 16 日

一、 实验目的和要求

本次实验初步介绍如何用 GMM（Gaussian Mixture Model）的方法来进行说话人确认（Speaker Recognition）。

训练说话人模型时，使用第一周正常语速的 6 句

测试的时候，使用第一周其余的语句。

二、 实验内容和原理

配置 Matlab 工具箱

将实验准备中列出的工具箱解压放在一个文件夹下，使用 Matlab 的 File – Set Path 功能中的 Add with subfolders 按钮，将实验所需工具箱及相关添加到 Matlab 的 Path 中并使用 Save 按钮保存配置。

或者，更简单地，将依赖的文件放在和主程序相同的文件夹中即可。

参考代码

参考代码通过 `wavread` 以及 `melcepst` 读取 `.wav` 文件并提取特征 `train_feature`（12 维 MFCC），然后使用 `gmm_estimate` 为说话人训练模型（16 阶 GMM），得到模型的 3 个参数 `[mu, sigma, c]`。最后将被测特征 `test_feature` 和要比对的说话人模型参数传入函数 `lmultigauss`，即得到该被测特征与指定模型的比对得分 `lY`。

代码中的 `[YM,Y] = lmultigauss(x,mus,sigm,c)` 试图计算模型匹配得分，得到 `YM, Y` 两个参数。对 `Y` 求 `mean`，即该段测试语音得分与对应模型的比对得分。

代码训练了 `1` 个说话人的模型，测试了 `13` 组特征，最终 `score` 为得分数组，得分数组的第 `i` 行第 `j` 列的值代表第 `i` 个测试语音与第 `j` 个说话人模型的得

分，分数高则表示更匹配。

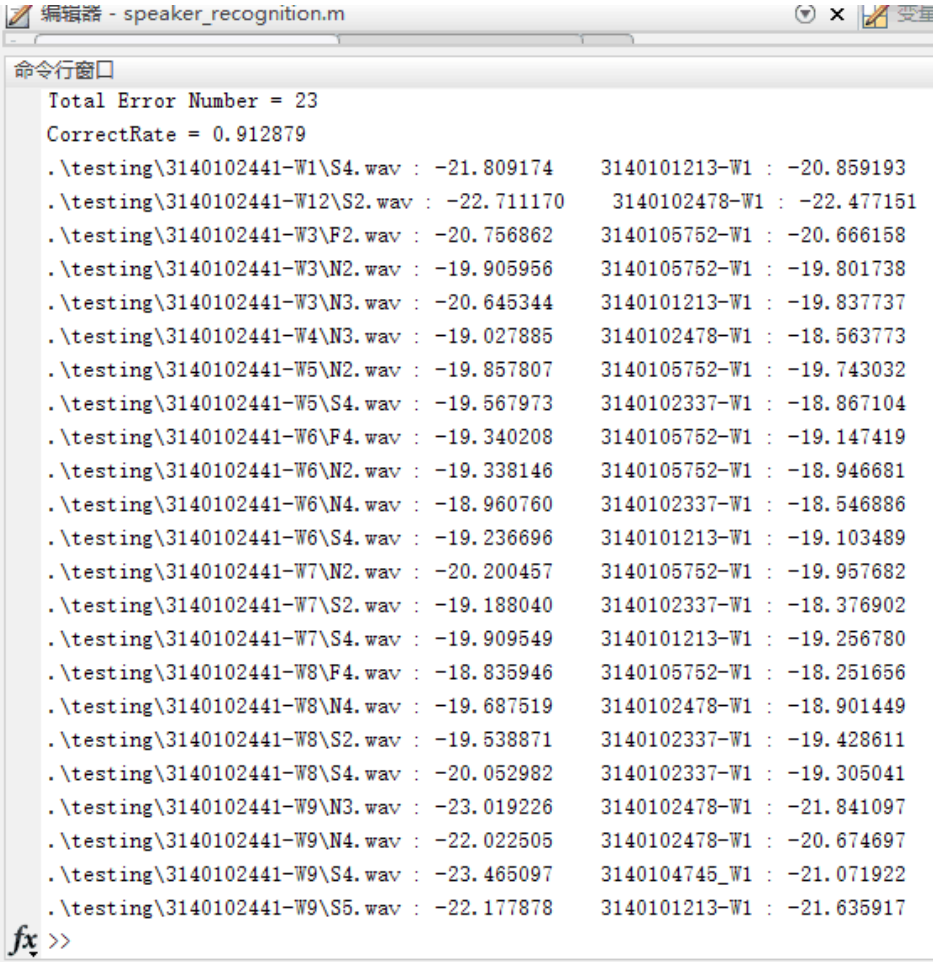
参考代码：

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%本代码适用于单人的说话人确认
clc;
clear all;
close all;
MFCC_size=12;%mfcc 的维数
GMMM_component=16;%GMM component 个数

mu_model=zeros(MFCC_size,GMMM_component);%高斯模型 分量 均值
sigma_model=zeros(MFCC_size,GMMM_component);%高斯模型 分量 方差
weight_model=zeros(GMMM_component);%高斯模型 分量 权重

train_file_path='.\training\1\1_';%模型训练文件路径
num_train=6;%目标说话人模型训练文件的个数
test_file_path='.\testing\';%测试文件路径
num_test=1;%测试情况下朗读的次数
num_uttenance=6;%测试情况下每次朗读的句子的总数

all_train_feature=[];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%train model
%使用 1_1～1_6 训练
for i=1:num_train
    train_file=[train_file_path num2str(i) '.wav'];
    [wav_data ,fs]=wavread(train_file);
    train_feature=melcepst(wav_data ,fs);
    all_train_feature=[all_train_feature;train_feature];

end
[mu_model,sigma_model,weight_model]=gmm_estimate(all_train_feature',
GMMM_component);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%test
for i=1:num_test
    for j=1:num_uttenance
        test_file=[test_file_path num2str(i) '_' num2str(j) '.wav'];
        [wav_data ,fs]=wavread(test_file);
        test_feature=melcepst(wav_data ,fs);
        [lYM, lY] = lmultigauss(test_feature', mu_model, sigma_model,
weight_model);
```

```
        score(i) = mean(lY);
        fprintf('Test: %d_%d  score:%f\n',i,j,score(i));
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%result
[max_score,max_id]=max(score);
[min_score,min_id]=min(score);
fprintf('Max score:%f\nMin score:%f\n',max_score,min_score);
```

三、 实验过程和数据记录

将参考代码进行一定的修改之后，按照实验步骤进行训练与测试，结果如下：



测试的语句一共有 264 句，其中误识的数目为 23 句，正确率为 91.29%

四、 实验结果分析

定性分析:

1 、由上图可以看出，错误的语句主要集中在前九周，后几周基本是没有什么错误的，这个很大可能是因为前九周的录音，我是在四周内完成的，基本每周录两次，录完了九周之后，剩下的录音是隔了一段时间之后再录的，而刚好那段寝室比较安静，而且前九周的录音，寝室噪音比较大，并且那时候我的嗓子也不太好，所以导致了上述结果。

2、仔细听了错误的音频之后，将其与正确识别的音频做了定性的比较，大致上的感受是错误的音频带有较大的噪音，并且音调偏低，音量偏小，声音比较浑浊。

用 praat 进行定量分析：

1、 综合分析

```
Amplitude:
   Minimum: -0.0928955078 Pascal
   Maximum: 0.11895752 Pascal
   Mean: -6.29383555e-007 Pascal
   Root-mean-square: 0.0145754969 Pascal
Total energy: 0.000915505645 Pascal² sec (energy in air: 2.28876411e-006 Joule/m²)
Mean power (intensity) in air: 5.31112774e-007 Watt/m² = 57.25 dB
Standard deviation in channel 1: 0.0145757083 Pascal
```

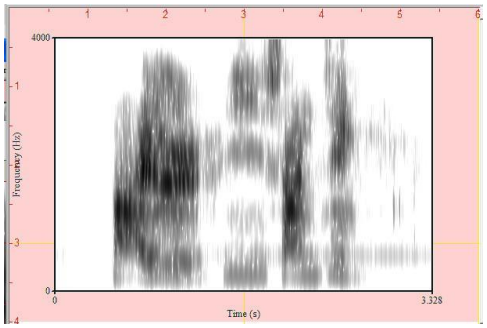
主要对振幅和能量进行量化的分析，我统计了作为模型的 6 句话的振幅和能量以及出错语句的平均振幅和能量，结果如下：

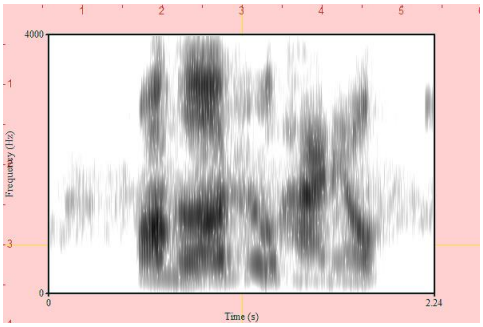
模型：root-mean-square : 0.03478963 pascal Mean power in air : 64.40 dB

出错语句：root-mean-square : 0.01844833 pascal Mean power in air : 57.11347826 dB

从中，可以看出，出错语句的振幅以及能量明显要比模型的低，说明录制的时候语调偏低，音量偏小。

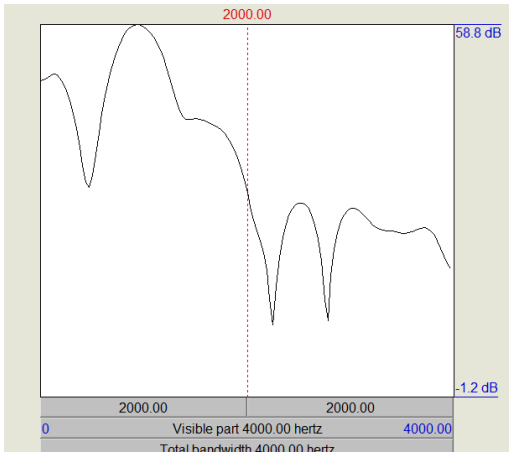
2、 语图分析

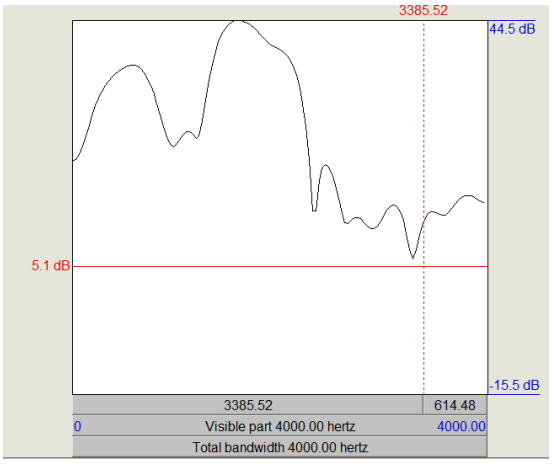




左边的是模型语句的语图，右图是出错语句的语图，从语图中可以看出，两者的差别并不是很大。

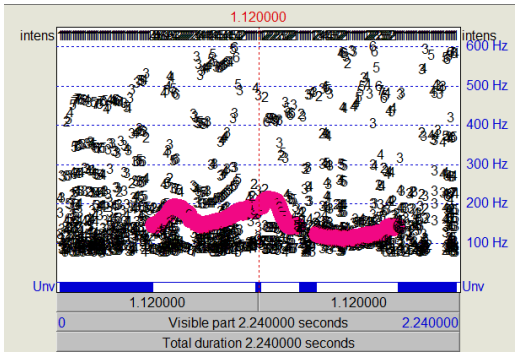
3、 二维频谱分析

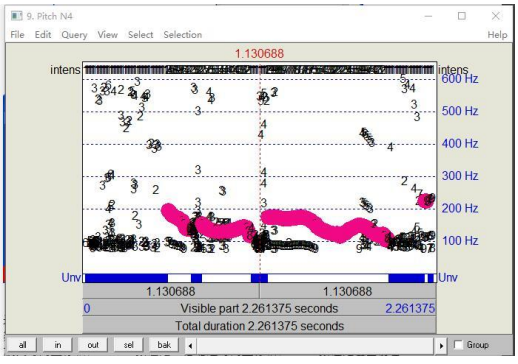




从二维频谱图中可以看出，出错语句的最高能量值为 44.5db，明显低于模型的 58.8db，在频率方面，可以看出，出错语句峰值的频率略微高于模型的峰值语句。

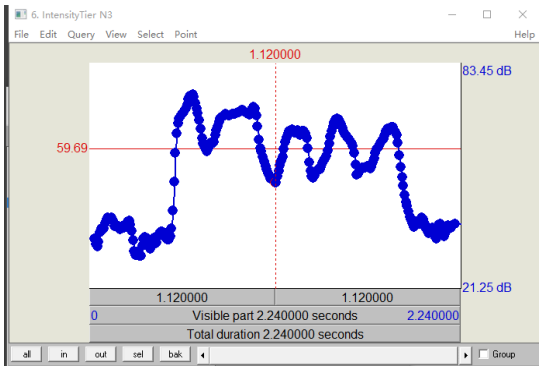
4、 基频分析

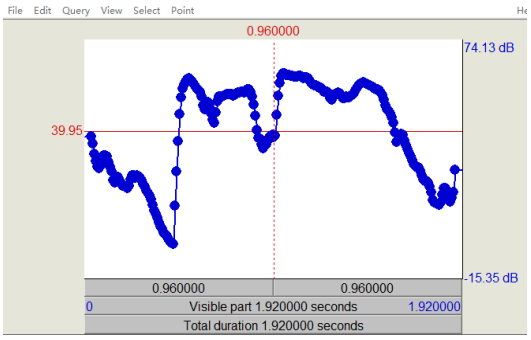




从基频图上，我们可以看出来，两者的差别不是很大，主要是出错语句的频率不连续，中间出现了间断，而模型语句的基频图一直是连续的，这可能是因为在录制出错语句的时候，中间停顿了一下而造成的不连续。

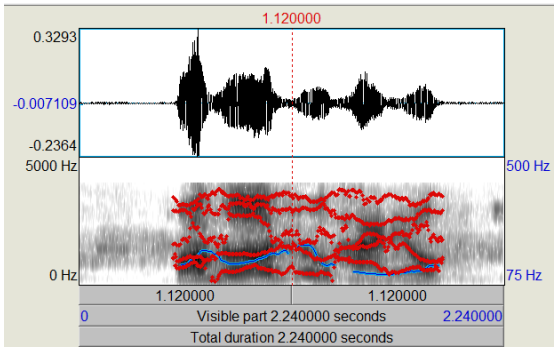
5、 强度分析

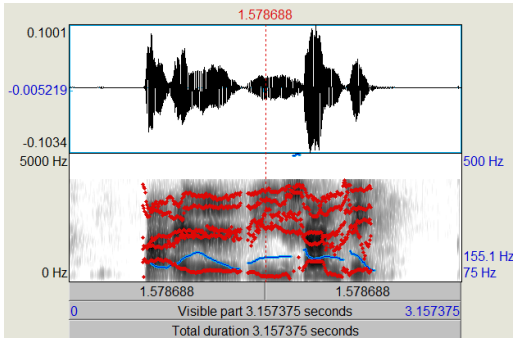




从强度图中可以看出，出错语句的平均强度为 59.2dB，低于模型语句的 64.1dB，这也符合我们之前的分析结果。

6、 共振峰





从共振峰的峰值频率可以看出，出错语句的频率明显要低于模型语句，这也说明，在录制出错语句的时候，声音比较低沉。

总结分析：

从定性和定量的角度综合来看，出错语句的振幅和强度明显要低于模型语句，导致音调偏低，音量偏小，还有一点就是，出错语句里面，噪声比较大，

设计算法时可以从这两方面着手改良。

五、 算法设计

首先，先用 matlab 对音频进行噪声处理。利用 matlab 添加 wav 音频文件，并在文件上添加用 randn 产生均值为 0 方差为 1 的正态分布白噪声，再用 IIR、FIR 滤波器滤波。

代码如下:

```
%读取信号的 matlab 程序
[x,Fs,bits]=wavread('mail.wav');%读出信号，采样率和采样位数。
%x=x(:,1);                          %这里语音信号是双声道，如只取单声道作分析，可加 x=x(:,1)或 x=x(:,2)
sound(x,Fs,bits);                  %对原始信号的声音进行回放
X=fft(x,4096);                     %对信号函数进行快速离散傅里叶变换分析
magX=abs(X);                       %求幅值
angX=angle(X);                     %求相位
figure(1)                          %画图
subplot(221);plot(x);title('原始信号波形');
subplot(222);plot(X); title('原始信号频谱');
subplot(223);plot(magX);title('原始信号幅值');
subplot(224);plot(angX);title('原始信号相位');
%加噪 matlab 程序
fs=4096;                           %采样频率取值
f=fs*(0:511)/1024;                 %对采样频率进行取值
t=0:1/1400:(size(x)-1)/1400;       %所加噪声信号的点数调整到与原始信号相同
d=0.004*randn(size(x));            %用 randn 产生均值为 0 方差为 1 的正态分布白噪声
```

```
x1=x+d;                                    %叠加噪声
sound(x1,4096);                            %播放加噪声后的语音信号
y1=fft(x,4096);
y2=fft(x1,4096);                           %对加噪信号函数进行快速离散傅里叶变换分析
figure(2)                                  %画图
plot(t,x1)
title('加噪后的信号');
xlabel('time n');
ylabel('幅值  n');
figure(3)
subplot(2,1,1);plot(f,abs(y1(1:512)));title('原始信号频谱');
xlabel('Hz');ylabel('幅值');
subplot(2,1,2);plot(f,abs(y2(1:512)));title('加噪后的信号频谱');
xlabel('Hz');ylabel('幅值');
%去噪 matlab 程序
N=10;wc=0.3;
[b,a]=butter(N,wc);                        %设计 N 为 10 阶的低通滤波器，wc 为它的 0.3dB 边缘频率，向量 b 和 a 分别表示系统函数的分子、分母多项式的系数
X=fft(x);
figure(4)                                  %画图
subplot(321);plot(x);title('滤波前信号的波形');
subplot(322);plot(X);title('滤波前信号的频谱');
y=filter(b,a,x1);                          %采用数字滤波器对数据进行滤波
Y=fft(y);
subplot(323);plot(y);title('IIR 滤波后信号的波形');
subplot(324);plot(Y);title('IIR 滤波后信号的频谱');
sound(y)                                   %IIR 滤波后的恢复信号声音回放
z=fftfilt(b,x1);                           %基于 FFT 的重叠相加法对数据进行滤波
```

```
Z=fft(z);
subplot(325);plot(z);title('FIR 滤波后信号的波形');
subplot(326);plot(Z);title('FIR 滤波后信号的频谱');
sound(z)                                %IIR 滤波后的恢复信号声音回放
```

处理好噪音之后，由于之前的步骤分析出来的结果是出错音频的振幅和强度明显低于模型音频，因此，我改进的算法是在 GMM 算法的基础上，降低振幅和强度的权重，重新进行训练与测试。代码如下：

```
function [Alpha, Mu, Sigma] = GMM_EM(Data, Alpha0, Mu0, Sigma0)
%% EM 迭代停止条件
loglik_threshold = 1e-10;
%% 初始化参数
[dim, N] = size(Data);
M = size(Mu0,2);
loglik_old = -realmax;
nbStep = 0;

Mu = Mu0;
Sigma = Sigma0;
Alpha = Alpha0;
Epsilon = 0.0001;
while (nbStep < 1200)
 nbStep = nbStep+1;
 %% E-步骤 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 for i=1:M
  % PDF of each point
  Pxi(:,i) = GaussPDF(Data, Mu(:,i), Sigma(:,:,i));
 end

 % 计算后验概率 beta(i|x)
 Pix_tmp = repmat(Alpha,[N 1]).*Pxi;
 Pix = Pix_tmp ./ (repmat(sum(Pix_tmp,2),[1 M])+realmin);
 Beta = sum(Pix);
 %% M-步骤 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 for i=1:M
  % 更新权值
  Alpha(i) = Beta(i) / N;
  % 更新均值
  Mu(:,i) = Data*Pix(:,i) / Beta(i);
  % 更新方差
```

```
    Data_tmp1 = Data - repmat(Mu(:,i),1,N);
    Sigma(:,:,i) = (repmat(Pix(:,i)',dim, 1) .* Data_tmp1*Data_tmp1') / Beta(i);
    %% Add a tiny variance to avoid numerical instability
    Sigma(:,:,i) = Sigma(:,:,i) + 1E-5.*diag(ones(dim,1));
  end

%  %% Stopping criterion 1 %%%%%%%%%%%%%%%%%%%%
%  for i=1:M
    %Compute the new probability p(x|i)
%    Pxi(:,i) = GaussPDF(Data, Mu(:,i), Sigma(i));
%  end
  %Compute the log likelihood
%  F = Pxi*Alpha';
%  F(find(F<realmin)) = realmin;
%  loglik = mean(log(F));
  %Stop the process depending on the increase of the log likelihood
%  if abs((loglik/loglik_old)-1) < loglik_threshold
%    break;
%  end
%  loglik_old = loglik;

  %% Stopping criterion 2 %%%%%%%%%%%%%%%%%%%%
  v = [sum(abs(Mu - Mu0)), abs(Alpha - Alpha0)];
  s = abs(Sigma-Sigma0);
  v2 = 0;
  for i=1:M
    v2 = v2 + det(s(:,:,i));
  end

  if ((sum(v) + v2) < Epsilon)
    break;
  end
  Mu0 = Mu;
  Sigma0 = Sigma;
  Alpha0 = Alpha;
end
nbStep
```

取代之前的 GMM 文件，采用改进后的算法之后，重新进行测试，结果如下：

```
Total Error Number = 10
CorrectRate = 0.9621
.\testing\3140102441-W1\F2.wav : -20.263955    3140102337-W1 : -19.739520
.\testing\3140102441-W1\S4.wav : -21.728014    3140101213-W1 : -20.589658
.\testing\3140102441-W3\F3.wav : -21.895034    3140105752-W1 : -21.761460
.\testing\3140102441-W3\N2.wav : -19.937711    3140105752-W1 : -19.450921
.\testing\3140102441-W3\N3.wav : -21.624903    3140103542-W1 : -20.523568
.\testing\3140102441-W4\N3.wav : -18.728392    3140102478-W1 : -18.666144
.\testing\3140102441-W5\N2.wav : -19.672688    3140102337-W1 : -19.656031
.\testing\3140102441-W5\S4.wav : -19.622291    3140102337-W1 : -18.849593
.\testing\3140102441-W6\F4.wav : -19.689147    3140105752-W1 : -19.205180
.\testing\3140102441-W6\F6.wav : -19.892929    3140101213-W1 : -19.720980
```

可以看出，错误语句减少到了 10 句，准确率提高到了 96.12%，说明，设计的改进算法还是起到了作用。

六、 讨论与心得

这个大程，整体上难度不大。难的地方主要有两点，一点是一开始老师提供的代码是有错误的，然后我直接复制代码过来运行，一直都提示存在错误，但是，我对 matlab 代码又不是很熟悉，所以我花了很长时间去搞懂 matlab 的语法， 并且把参考代码好好地研究了一下，才发现了代码的错误所在。代码能跑之后， 测试结果和结果分析这一块，就没有那么难了。

第二个难点就是算法的设计。我是基于之前分析出来的错误音频的特点，结合网上搜素的 GMM 优化算法，调整了算法的参数权重分配，才设计出的新的改进算法，从结果上来看，准确率提高了不少，改进算法还是可以的，不过这个算法的局限性在于，只是针对了我自己的情况，而不适用于他人，因为每个人的错误音频与模型音频的差别都不一样。

总的来说，这个实验让我对语音识别的过程有了大致的了解， 并且对 GMM 算法有了自己的认识，另外， 还掌握了 praat 软件的使用，收获还是非常大的。