哈爾濱工業大學

实验报告

实 验（三）

题　　目 命令词识别实验报告

专　　业 计算机科学与技术

学　　号 1160300909

班　　级 1603106

学　　生 张 志 路

指 导 教 师 郑 铁 然

实 验 地 点 G709

实 验 日 期 2018 年 11 月 12 日

计算机科学与技术学院

目 录

一、 设计命令词识别任务

1.1 描述你所设计的命令词识别任务

语音识别并不是一项新兴的技术，但在智能家居领域或许能大展身手。而对于智能家居而言，将语音控制技术规模化应用于智能家居或将成为未来市场发展的新趋势之一。本实验将针对生活中常见的一些家电控制指令进行语音识别。

1.2 列出你的词表

共十个命令词，分别为：开门、关门、开灯、关灯、开空调、关空调、开电视、关电视、烧水、蒸饭。

1.3 介绍语料采集方法和规模

用 Cool Edit 软件录制音频进行采集，每个词录制五遍，其中一遍为模板，其它四遍用来测试。测试样例录制时适当变化音调和音量，以模拟特定说话人不同的发声情况。

二、特征提取

2.1 详细描述你所采用的特征和提取算法

对语音信号提取 mfcc 特征，特征提取算法和方法如下。

(1) mfcc 特征提取的基本流程

① 预加重

预加重处理其实是将语音信号通过一个高通滤波器：$H(z) = 1-uz^{-1}$。

式中的值介于 0.9-1.0 之间，我们通常取 0.97。预加重的目的是提升高频部分，使信号的频谱变得平坦，保持在低频到高频的整个频带中，能用同样的信噪比求频谱。同时，也是为了消除发生过程中声带和嘴唇的效应，来补偿语音信号受到发音系统所抑制的高频部分，也为了突出高频的共振峰。

② 分帧

先将 N 个采样点集合成一个观测单位，称为帧。通常情况下 N 的值为 256 或 512，涵盖的时间约为 20~30ms 左右。为了避免相邻两帧的变化过大，因此会让两

相邻帧之间有一段重叠区域，此重叠区域包含了 M 个取样点，通常 M 的值约为 N 的 1/2 或 1/3。通常语音识别所采用语音信号的采样频率为 8KHz 或 16KHz，以 8KHz 来说，若帧长度为 256 个采样点，则对应的时间长度是 256/8KHz =32ms。

③ 加窗（Hamming Window）

将每一帧乘以汉明窗，以增加帧左端和右端的连续性。假设分帧后的信号为 S(n)，n=0,1,…,N-1，N 为帧的大小，那么乘上汉明窗后为 *S'(n) = S(n)*× $W(n)$，其中 W(n)形式如下：

W(n,a) = (1-a) – a× $cos\frac{2\pi n}{N-1}$ ，*0*≤ n ≤*N-1*

不同的 a 值会产生不同的汉明窗，一般情况下 a 取 0.46。

④ 快速傅里叶变换

由于信号在时域上的变换通常很难看出信号的特性，所以通常将它转换为频域上的能量分布来观察，不同的能量分布，就能代表不同语音的特性。所以在乘上汉明窗后，每帧还必须再经过快速傅里叶变换以得到在频谱上的能量分布。对分帧加窗后的各帧信号进行快速傅里叶变换得到各帧的频谱。并对语音信号的频谱取模平方得到语音信号的功率谱。设语音信号的 DFT 为：

$x_a(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi k/N}$ ，$0 \le k \le N$

式中 x(n)为输入的语音信号，N 表示傅里叶变换的点数。

⑤ 三角带通滤波器

将能量谱通过一组 Mel 尺度的三角形滤波器组，定义一个有 M 个滤波器的滤波器组（滤波器的个数和临界带的个数相近），采用的滤波器为三角滤波器，中心频率为。M 通常取 22-26。各 f(m)之间的间隔随着 m 值的减小而缩小，随着 m 值的增大而增宽。

三角滤波器的频率响应定义为:

$$H_m(k)=\begin{cases}0 & k<f(m-1)\\ \frac{2(k-f(m-1))}{(f(m+1)-f(m-1))(f(m)-f(m-1))} & f(m-1)\le k\le f(m)\\ \frac{2(f(m+1)-k)}{(f(m+1)-f(m-1))(f(m)-f(m-1))} & f(m)\le k\le f(m+1)\\ 0 & k\ge f(m+1)\end{cases}$$

式中：$\sum_{m=0}^{M-1} H_m(k) = 1$

三角带通滤波器有两个主要目的：

对频谱进行平滑化，并消除谐波的作用，突显原先语音的共振峰。因此一段语音的音调或音高，是不会呈现在 MFCC 参数内，换句话说，以 MFCC 为特征的语音辨识系统，并不会受到输入语音的音调不同而有所影响。

此外，还可以降低运算量。

⑥ 计算每个滤波器组输出的对数能量为

$s(m) = ln(\sum_{k=0}^{N-1} |X_a(k)|^2 H_m(k))$ ， $0 \leq m \leq M$

⑦ 经离散余弦变换（DCT）得到 MFCC 系数

$C(n) = \sum_{m=0}^{N-1} s(m) \cos \frac{\pi n(m-0.5)}{M}$ ， $n=1,2,\ldots,L$

将对数能量带入离散余弦变换，求出 L 阶的 Mel-scale Cepstrum 参数。L 阶指 MFCC 系数阶数，通常取 12-16。这里 M 是三角滤波器个数。

⑧ 对数能量

此外，一帧的音量（即能量），也是语音的重要特征，而且非常容易计算。因此，通常再加上一帧的对数能量（定义：一帧内信号的平方和，再取以 10 为底的对数值，再乘以 10）使得每一帧基本的语音特征就多了一维，包括一个对数能量和剩下的倒频谱参数。

注：若要加入其它语音特征以测试识别率，也可以在此阶段加入，这些常用的其它语音特征包含音高、过零率以及共振峰等。

⑨ 动态查分参数的提取（包括一阶差分和二阶差分）

标准的倒谱参数 MFCC 只反映了语音参数的静态特性，语音的动态特性可以用这些静态特征的差分谱来描述。实验证明：把动、静态特征结合起来才能有效提高系统的识别性能。差分参数的计算可以采用下面的公式：

$$d_t = \begin{cases} C_{t+1} - C_t & t > K \\ \dfrac{\sum_{k=1}^{K} k(C_{t+k} - C_{t-k})}{\sqrt{2\sum_{k=1}^{K} k^2}} & \text{其他} \\ C_t - C_{t-1} & t \geq Q - K \end{cases}$$

式中，d_t 表示第 t 个一阶差分；C_t 表示第 t 个倒谱系数；Q 表示倒谱系数的阶数；K 表示一阶导数的时间差，可取 1 或 2。将上式中结果再代入就可以得到二阶差分的参数。

总结：因此，MFCC 的全部组成其实是由 N 维 MFCC 参数（N/3MFCC 系数+N/3 一阶差分参数+N/3 二阶差分参数）和帧能量（此项可根据需求替换）组成。

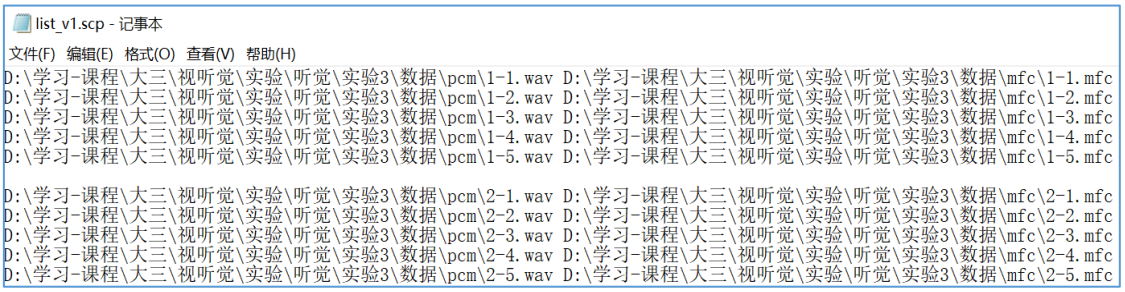
由此可知，39 维的 MFCC 参数即由 13 维 MFCC 系数、13 维一阶差分参数和 13 维二阶差分参数组成。

(2) mfcc 特征提取的方法

① HTK 工具包

配置好 list_v1.scp 文件后，运行“hcopy -A -D -T 1 -C tr_wav.cfg -S list_v1.scp”命令即可。

list_v1.scp 文件配置部分截图如下。



读取.mfc 文件的代码如下。

```
'''''读取.mfc 二进制文件'''
def readmfcc(filename):
    fid = open(filename, 'rb')
    nframes,frate,nbytes,feakind = struct.unpack('>IIHH',fid.read(12))
    '''''
    nframes: number of frames 采样点数
    frate: frame rate in 100 nano-seconds unit
    nbytes: number of bytes per feature value
    feakind: 9 is USER
    '''
    ndim = nbytes//4  # feature dimension(4 bytes per value) 39 维度
    mfcc0 = np.empty((nframes,ndim))
    for i in range(nframes):
        for j in range(ndim):
            mf = fid.read(4)
            c = struct.unpack('>f',mf)
            mfcc0[i][j] = c[0]
    fid.close()
    return mfcc0
```

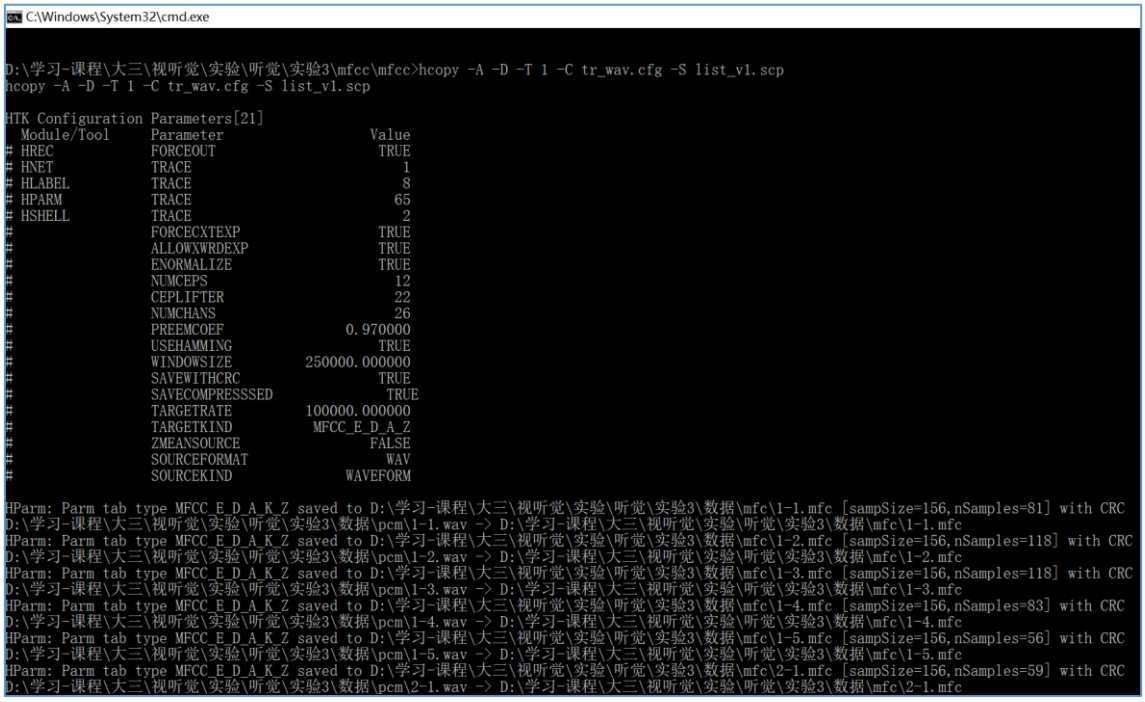
② python 库

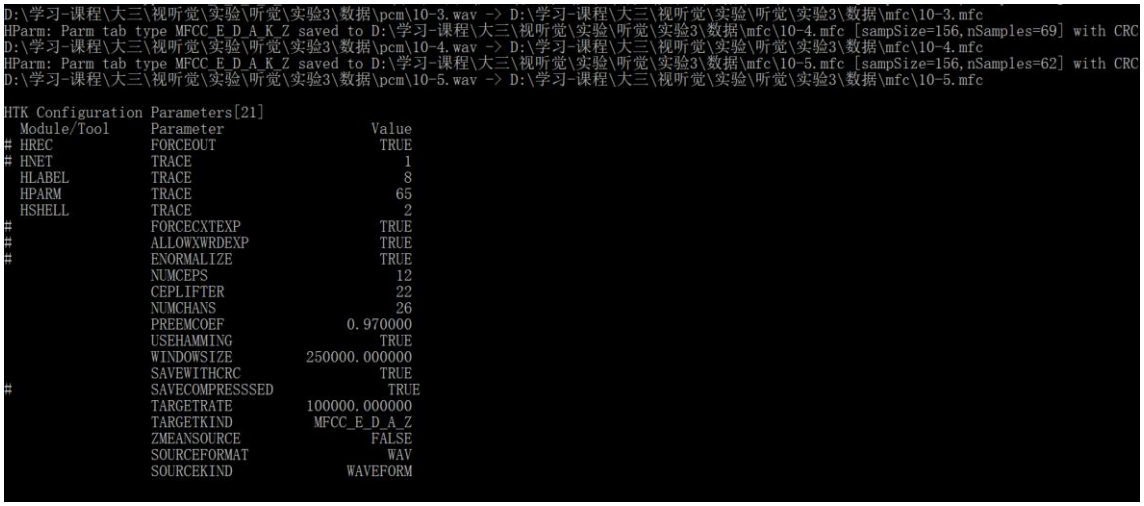
利用 librosa 库，提取 mfcc 特征，调用代码如下。

```
'''''利用 librosa 库，提取 mfcc 特征'''
def mfcc(filename):
    y, sr = librosa.load(filename, sr=None) # Load a wav file
    mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=39) # extract mfcc feature
    return mfccs.transpose()
```

2.2 给出特征提取部分运行结果的截图

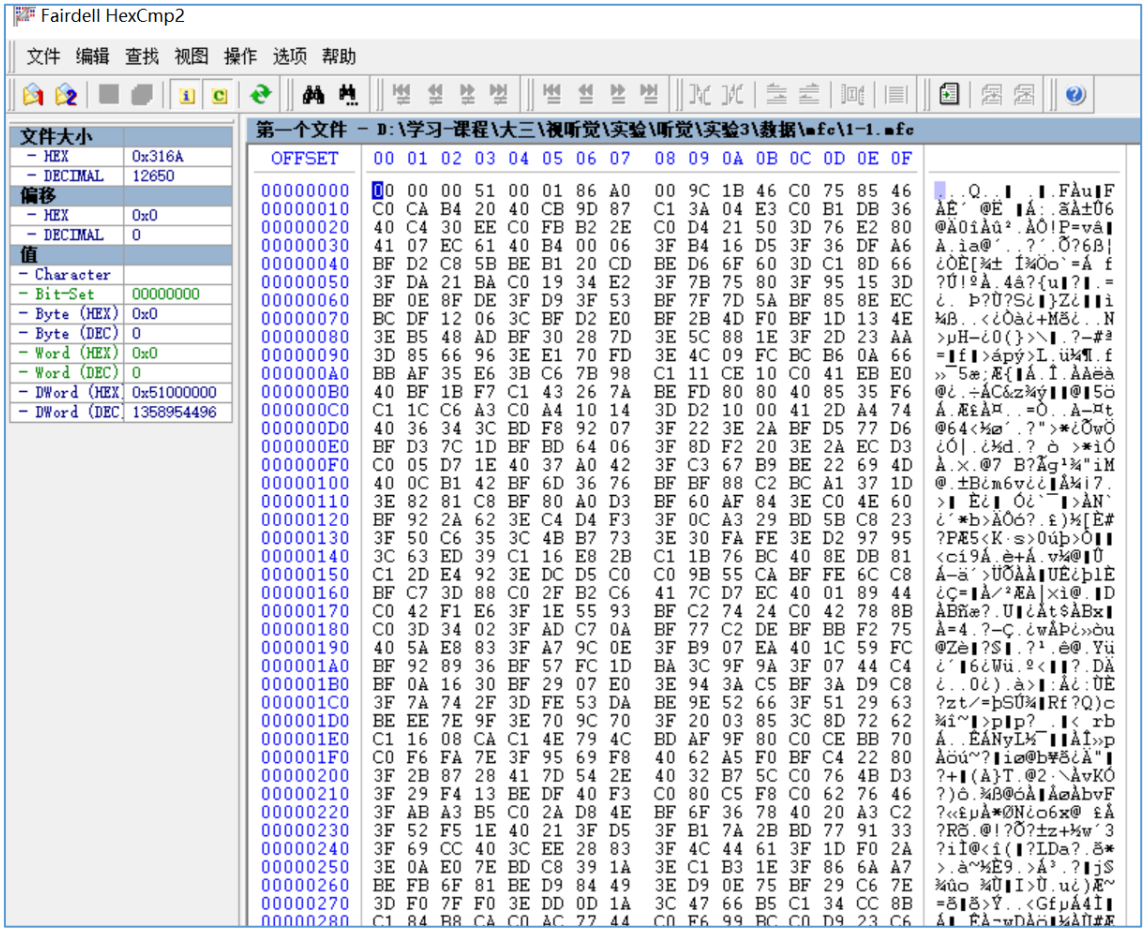
下面给出利用 HTK 工具包进行特征提取的部分截图。





2.3 给出特征文件内容的截图

下面给出利用 HTK 工具包进行特征提取的部分文件内容的截图。



三、 DTW 计算

3.1 给出 DTW 算法，标明所采用的开发工具

(1) 开发工具

编程语言：python 3.6.4

编程工具：PyCharm 5.0.3、Anaconda3-5.1.0

操作系统：Windows 10 64-bit

(2) DTW 算法

算法代码如下。

```
'''''
dtw 算法
wavedata1: mfc 文件 1
wavedata2: mfc 文件 2
'''
def dtw(wavedata1, wavedata2):
    # 初始化
    len1 = len(wavedata1)
    len2 = len(wavedata2)
    D = np.empty((len1+1,len2+1))  # 记录代价
    P = np.empty((len1+1,len2+1,2))  # 记录路径
    D[0][0] = 0
    P[0][0] = [0, 0]
    for i in range(1,len1+1):
        D[i][0] = sys.maxsize
    for j in range(1,len2+1):
        D[0][j] = sys.maxsize
    # 计算
    for i in range(1,len1+1):
        for j in range(1,len2+1):
            d = euclidean(wavedata1[i-1], wavedata2[j-1])  # 欧氏距离
            a1 = 2*d + D[i-1][j-1]
            a2 = d + D[i-1][j]
            a3 = d + D[i][j-1]
            minD = min(a1, a2, a3)
            if minD == a1:
                P[i][j] = [i-1, j-1]
            elif minD == a2:
                P[i][j] = [i-1, j]
            else:
                P[i][j] = [i, j-1]
            D[i][j] = minD
    # 回溯路径
    w = 0
    m = P[len1][len2]
    while True:
        pm = P[int(m[0])][int(m[1])]
        if pm[0]+1 == m[0] and pm[1]+1 == m[0]:
            w += 2
        else:
            w += 1
        if pm[0] == 0 and pm[1] == 0:
            break
        m = pm
    return D[len1][len2]/w

'''''欧氏距离'''
def euclidean(a, b):
    return np.sqrt(np.sum(np.square(a-b)))
```

四、 计算正确率

4.1 正确率

正确检出的语料文件的个数：

正确率= 100 %

说明如下：

① 利用 HTK 工具包进行 39 维 mfcc 特征提取时，正确率= 95 %

② 利用 python 中的 librosa 库进行 mfcc 特征提取时，不同维数的正确率差异较大，列表如下。

维数	正确率
6	0.95
9	0.975
12	0.975
15	1
18	0.975
21	0.95
39	0.8

由表可知，当利用 python 中的 librosa 库进行 mfcc 特征提取时，提取 15 维 mfcc 特征较好，此时正确率为百分之百。

五、 总结

5.1 请总结本次实验的收获

(1) 通过此次实验，我进一步熟悉了 DTW 算法相关的理论知识，对语音识别和尤其是命令词识别有了更深的理解。

(2) 了解了 mfcc 特征提取的算法与方法。

(3) 初步了解了 HTK 工具包的使用。

(4) 设计了命令词识别任务，成功进行了离线命令词识别。经过对程序的改进，成功完成了在线命令词识别，随录制随识别。

5.2 请给出对本次实验内容的建议

无。

5.3 介绍一下你开展了哪些扩展尝试，效果如何？

扩展尝试：将经过实验验证的算法，转化为能实时采集，在线检测的命令词识别系统。

(1) 增加一个录音模块，该模块实现录音和播放功能，并将录音保存至文件，主要代码如下。

```
'''''录音'''
def my_record(filename):
    pa = PyAudio()
    stream = pa.open(format=paInt16, channels=2, rate=framerate, input=True, frames_per_buffer=NUM_SAMPLES)
    my_buf = []
    for i in range(0, int(framerate/NUM_SAMPLES*RECORD_SECONDS)):#控制录音时间
        string_audio_data = stream.read(NUM_SAMPLES)
        my_buf.append(string_audio_data)
        sys.stdout.write('. ')
        sys.stdout.flush()
    save_wave_file(filename,my_buf)
    stream.close()

'''''播放录音'''
def play(filename):
    wf = wave.open(filename,'rb')
    p = PyAudio()
    stream = p.open(format=p.get_format_from_width(wf.getsampwidth()), channels=wf.getnchannels(), rate=wf.getframerate(), output=True)
    while True:
        data = wf.readframes(NUM_SAMPLES)
        if data == b'':
            break
        stream.write(data)
    stream.close()
    p.terminate()
```

(2) 此外，对语音识别进行容错改进，即设置一个 DTW 值门限，当计算出的最小 DTW 值仍大于该门限时，系统认为该语音表示的词不在语料库中。部分代码如下。

```
if min < 120:  # 门限
    print(filename+'文件识别结果为:'+wordlist[flag-1]+'\n')
else:
    print('抱歉，未能识别出结果，请重新录制识别\n')
```

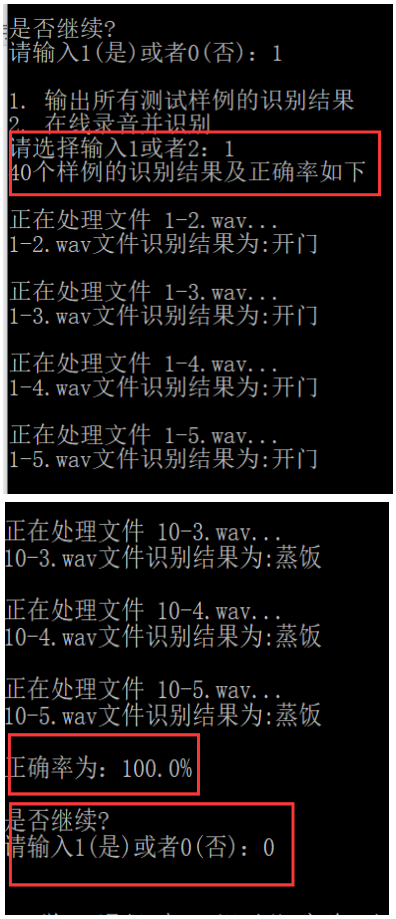
(3) 主模块实现简单的界面，运行结果部分截图如下。

在线识别结果如下。

```
D:\学习-课程\大三\视听觉\实验\听觉\实验3\code\code_v1>ipython main.py
正在载入模板...
载入模板完成!

1. 输出所有测试样例的识别结果
2. 在线录音并识别
请选择输入1或者2: 2
请输入保存文件名（例如"wave"），回车后将开始录音，录音将持续3秒: guanmen
. . . . . .
. . . . . .
. . . . . 文件已保存在 online/data/guanmen.wav
是否播放录音?
请输入1(是)或者0(否): 1

是否开始语音识别?
请输入1(是)或者0(否): 1
正在处理文件 guanmen...
dtw = 76.2034327984
guanmen文件识别结果为:关门


是否继续?
请输入1(是)或者0(否): 1

1. 输出所有测试样例的识别结果
2. 在线录音并识别
请选择输入1或者2: 2
请输入保存文件名（例如"wave"），回车后将开始录音，录音将持续3秒: kaikongtiao
. . . . . .
. . . . . .
. . . . . 文件已保存在 online/data/kaikongtiao.wav
是否播放录音?
请输入1(是)或者0(否): 0

是否开始语音识别?
请输入1(是)或者0(否): 1
正在处理文件 kaikongtiao...
dtw = 83.3169577463
kaikongtiao文件识别结果为:开空调
```

```
1. 输出所有测试样例的识别结果
2. 在线录音并识别
请选择输入1或者2: 2
请输入保存文件名（例如"wave"），回车后将开始录音，录音将持续3秒: guandeng
. . . . . .
. . . . . .
. . . . . 文件已保存在 online/data/guandeng.wav
是否播放录音?
请输入1(是)或者0(否): 0

是否开始语音识别?
请输入1(是)或者0(否): 1
正在处理文件 guandeng...
dtw = 73.6983630039
guandeng文件识别结果为:关灯


是否继续?
请输入1(是)或者0(否): 1

1. 输出所有测试样例的识别结果
2. 在线录音并识别
请选择输入1或者2: 2
请输入保存文件名（例如"wave"），回车后将开始录音，录音将持续3秒: aaaaa
. . . . . .
. . . . . .
. . . . . 文件已保存在 online/data/aaaaa.wav
是否播放录音?
请输入1(是)或者0(否): 0

是否开始语音识别?
请输入1(是)或者0(否): 1
正在处理文件 aaaaa...
dtw = 139.823562759
抱歉，未能识别出结果，请重新录制识别
```

已有测试样例的部分输出结果及正确率如下。



综上所述，增加扩展后，实时语音识别的效果基本与已有测试样例的识别效果一致，在特定人特定词识别的情况下，正确率可达 100 %。

参考文献：

[1] 韩纪庆，张磊，郑铁然.语音信号处理[M].北京：清华大学出版社，2013.