cowpea
领会和表达复杂世界背后的简单性的能力是一份难得而美妙的礼物 —— Van Jacobson

导航

博客园
首页
联系
订阅XML
管理

< 2019年7月 >

日	一	二	三	四	五	六
30	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

昵称：姜小豆
园龄：6年2个月
粉丝：9
关注：8
+加关注

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类

*nix(4)
API(6)
C
Framework
LeetCode(16)
PHP(2)
Python(17)
算法(1)
网络(2)

Python简单语音识别并响应

起因是一个工作中喜欢说口头禅的同事，昨天老说“你看看你看看 操不操心”。说了几次之后我就在他说完“你看看”后面续上，“操不操心”。往复多次后，我就想，为啥不用Python识别语音并作出响应，正好没弄过语音识别。

1. 语音转文字

> 参考Python语音识别终极指南，吐槽一句：质量太差，是最烂的无审查的机翻。引模块中间都没空格 `importspeech_recognitionassr` 应该是 `import speech_recognition as sr` ； `并创建识一个别器类的例子` 应该是 `并创建一个识别器类的例子` 这块都不仅仅是机翻了吧，怎么会拆了词。但是为了了解API足够了。

语音转文字使用谷歌云平台的语音转文字服务[Google Cloud Speech API](https://cloud.google.com/speech/)，因为是不需要API密钥的。其实是因为有默认密钥：

```
def recognize_google(self, audio_data, key=None, language="en-US",
show_all=False):
    ...
    if key is None: key = "AIzaSyBOti4mM-6x9WDnZIjIeyEU21OpBXqWBgw
    ...
```

通过另外两个函数参数还可以了解到： `lanauage` （指定识别的语言）， `show_all` （False返回识别率最高的一条结果，True返回所有识别结果的 ~~json串~~ 字典数据）

安装 `pip install SpeechRecognition`

1.1 本地语音文件识别测试

```
# coding:utf-8

"""
本地语音文件识别测试
"""
import speech_recognition as sr
import sys

say = '你看看'
r = sr.Recognizer()

# 本地语音测试
harvard = sr.AudioFile(sys.path[0]+'/youseesee.wav')
with harvard as source:
    # 去噪
    r.adjust_for_ambient_noise(source, duration=0.2)
    audio = r.record(source)

# 语音识别
test = r.recognize_google(audio, language="cmn-Hans-CN", show_all=True)
print(test)

# 分析语音
flag = False
for t in test['alternative']:
```

杂项(11)

随笔档案

2019年7月 (1)
2019年6月 (2)
2019年5月 (1)
2019年4月 (2)
2019年2月 (1)
2019年1月 (2)
2018年12月 (4)
2018年11月 (2)
2018年10月 (3)
2018年8月 (4)
2018年7月 (15)
2018年5月 (1)
2018年4月 (4)
2018年3月 (2)
2017年12月 (1)
2017年7月 (1)
2017年6月 (2)
2017年3月 (1)
2016年12月 (3)
2016年11月 (6)
2016年10月 (1)
2016年6月 (1)

积分与排名

积分 - 25402
排名 - 25022

最新评论

1. Re:Python简单语音识别并响应
@谷中天嘻嘻嘻，谢谢鼓励...
--姜小豆
2. Re:Python简单语音识别并响应
很优秀！楼主
--谷中天
3. Re:Echarts关系图-力引导布局
@小斌大人完整项目没有保存。根据你自己的需求，缕清需求慢慢做，肯定可以做出来的...
--姜小豆
4. Re:Echarts关系图-力引导布局
我想要完整项目，最近要用到这方面的功能，能打包发一下我邮箱吗，很感谢
410665086@qq.com
--小斌大人
5. Re:Echarts关系图-力引导布局
能放上github或者发我分享学习一下吗，最近公司用到 谢谢你
410665086@qq.com
--小斌大人

阅读排行榜

1. Echarts关系图-力引导布局(14661)
2. Python-1 试玩OpenCV(10425)
3. Python简单语音识别并响应(4705)
4. 支付宝地铁SDK使用失败记录(4133)
5. 使用jQuery编辑删除页面内容，两种

```
    print(t)
    if say in t['transcript']:
        flag = True
        break
if flag:
    print('Bingo')
```

自己录了一段语音 `youseesee.wav` （内容为轻轻（类似悄悄话，声带不强烈震动）说的*你看看你看看*，持续两秒）。音频文件格式可以是 `WAV/AIFF/FLAC`

> `AudioFile` instance given a WAV/AIFF/FLAC audio file

去噪函数 `adjust_for_ambient_noise()` 在音频中取一段噪声（ `duration` 时间范围，默认1s），来优化识别。因为原音频很短，所以这里只取了 0.2s 噪声。

转换函数 `recognize_google()` 的 `lanaguage` 参数范围可以从 Cloud Speech-to-Text API 语言支持 处了解到，「中文、普通话（中国简体）」 为 **cmn-Hans-CN**。

`show_all` 前面有介绍，当上例中该参数为 `False` 时语音识别结果 `test` 输出 `呵呵你看看` ,为 `True` 时输出所有可能的识别结果：

```
{
    'alternative':[
        {
            'transcript':'呵呵你看看',
            'confidence':0.87500638
        },
        {
            'transcript':'呵呵你看'
        },
        {
            'transcript':'哥哥你看'
        },
        {
            'transcript':'哥哥你看看'
        },
        {
            'transcript':'呵呵你看咯'
        }
    ],
    'final':True
}
```

之后分析语音，只是简单找了识别结果是否包含期待值 `你看看` ，找出一个则表示正确识别并匹配，输出Bingo!

上例完整输出为：

```
{'alternative': [{'transcript': '呵呵你看看', 'confidence':
0.87500668}, {'transcript': '呵呵你看'}, {'transcript': '哥哥你看'},
{'transcript': '哥哥你看看'}, {'transcript': '呵呵你看咯'}], 'final':
True}
{'transcript': '呵呵你看看', 'confidence': 0.87500668}
Bingo
```

注：如果发生异常:

```
speech_recognition.RequestError
recognition connection failed: [WinError 10060] 由于连接方在一段时间后没有正确答复或连接的主机没有反应，连接尝试失败。
```

方式(1023)

评论排行榜

1. Echarts关系图-力引导布局(18)
2. LeetCode contest-95[876,877,878](4)
3. 进制闲谈(2)
4. 爬虫学习笔记(2)
5. Python简单语音识别并响应(2)

推荐排行榜

1. 不该迷茫的时候迷茫(1)
2. Python-1 试玩OpenCV(1)
3. Python简单语音识别并响应(1)

是因为（梯子）没有设置全局代理。

1.2 实时语音识别测试

把音频数据来源，从上面的音频文件，改为创建一个麦克风实例，并录音。

需要安装 `pyAudio` ，如果 `pip install pyAudio` 不能安装，可以去Python Extension Packages下载安装。

```
# coding:utf-8

"""
实时语音识别测试
"""
import speech_recognition as sr
import logging
logging.basicConfig(level=logging.DEBUG)

while True:
    r = sr.Recognizer()
    # 麦克风
    mic = sr.Microphone()

    logging.info('录音中...')
    with mic as source:
        r.adjust_for_ambient_noise(source)
        audio = r.listen(source)
    logging.info('录音结束，识别中...')
    test = r.recognize_google(audio, language='cmn-Hans-CN',
show_all=True)
    print(test)
    logging.info('end')
```

`listen()` 函数将监听录音，并等到静音时停止。

> until it encounters `recognizer_instance.pause_threshold` seconds of non-speaking or there is no more audio input.

等到显示录音中，开始说话，沉默后录音结束。试验中说了两次：一次是 `你看看你看看` ，二次是 `你再看看` 。结果打印如下：

```
INFO:root:录音中...
INFO:root:录音结束，识别中...
{'alternative': [{'transcript': '你看看你看看', 'confidence':
0.97500247}], 'final': True}
INFO:root:end
INFO:root:录音中...
INFO:root:录音结束，识别中...
{'alternative': [{'transcript': '你再看看', 'confidence': 0.91089392},
{'transcript': '你在看看'}, {'transcript': '你猜看看'}, {'transcript':
'你再敢看'}, {'transcript': '你在感慨'}], 'final': True}
INFO:root:end
INFO:root:录音中...
```

识别率挺高，（还试过百度的 `baidu-aip` ，因我的音频没识别出来作罢），语音转文字就完成了。

2. 文字转语音

使用 `pyttsx` 模块很简单，python3下为 `pyttsx3` 。

```
import pyttsx3
```

```
engine = pyttsx3.init()
engine.say("风飘荡，雨濛茸，翠条柔弱花头重")
engine.runAndWait()
```

如此简单即可听到语音朗读了。

3．识别并响应

将上面的组合起来即可完成识别语音并响应了。

- 语音识别转文字
- 文字正则匹配并找出对应的响应文字
- 响应（朗读文字）

```
# coding:utf-8

"""
语音识别并响应。使用谷歌语音服务，不需要KEY（自带测试KEY）。
https://github.com/Uberi/speech_recognition
"""
import speech_recognition as sr
import pyttsx3
import re
import logging
logging.basicConfig(level=logging.DEBUG)

resource = {
    r"(你看看?){1}.*\1": "我不看，再让我看打死你",
    r"(你看看?)+": "你看看你看看，操不操心",
    r"(你.+啥)+": "咋地啦",
    r"(六六六|666)+": "要不说磐石老弟六六六呢？",
    r"(磐|石|老|弟)+": "六六六",
}

engine = pyttsx3.init()

while True:
    r = sr.Recognizer()
    # 麦克风
    mic = sr.Microphone()

    logging.info('录音中...')
    with mic as source:
        r.adjust_for_ambient_noise(source)
        audio = r.listen(source)
    logging.info('录音结束，识别中...')
    test = r.recognize_google(audio, language='cmn-Hans-CN',
show_all=True)

    # 分析语音
    logging.info('分析语音')
    if test:
        flag = False
        message = ''
        for t in test['alternative']:
            logging.debug(t)
            for r, c in resource.items():
                # 用每个识别结果来匹配资源文件key(正则)，正确匹配则存储回答
并退出
                logging.info(r)
                if re.search(r, t['transcript']):
                    flag = True
                    message = c
                    break
```

```
            if flag:
                break
        # 文字转语音
        if message:
            logging.info('bingo....')
            logging.info('say: %s' % message)
            engine.say(message)
            engine.runAndWait()
            logging.info('ok')
    logging.info('end')
```

对应的资源文字为

```
resource = {
    r"(你看看?){1}.*\1": "我不看，再让我看打死你",
    r"(你看看?)+": "你看看你看看，操不操心",
    r"(你.+啥)+": "咋地啦",
    r"(六六六|666)+": "要不说磐石老弟六六六呢？",
    r"(磐|石|老|弟)+": "六六六",
}
```

这里刚好用到正则，其实刚开始没打算用正则，想匹配两次 `你看看` 的时候就想起回溯，就用正则了。

很方便：比如 `磐石老弟` 不好识别，就用 `(磐|石|老|弟)+` 找出一个匹配即可；`你看看你看看` 用回溯 `\1` 。因为匹配时候发现说的快了有时匹配一个看，就用了 `你看看?` 来匹配 `你看` ，其实后面的 `看?` 要不要都可以，但为了说明目的，还是没有去掉。

`(你看看?){1}.*\1` 能匹配

```
你看看你看看
你看看你看
你看你看看看...
```

这样识别率就高了。因为识别结果匹配时候从头往后匹配每个正则，遇到则完成，所以 `(你看看?){1}.*\1` 需放在 `(你看看?)+` 前面。不然语音识别到 `你看看你看看` 就只能触发 `(你看看?)+` 了。

运行识别结果：

语音说了六次：

你看看，你看看你看看，你瞅啥，磐石，666，哈哈哈 （文字为了说明形象化，传输过去只是音频）

```
INFO:root:录音中...
INFO:root:录音结束，识别中...
INFO:root:分析语音
DEBUG:root:{'transcript': '你看看', 'confidence': 0.97500253}
INFO:root:(你看看?){1}.*\1
INFO:root:(你看看?)+
INFO:root:bingo....
WARNING:root:say: 你看看你看看，操不操心
INFO:root:ok
INFO:root:end
------------------------------------------------------------------
--
INFO:root:录音中...
INFO:root:录音结束，识别中...
INFO:root:分析语音
DEBUG:root:{'transcript': '你看看你看看', 'confidence': 0.97500247}
INFO:root:(你看看?){1}.*\1
```

```
INFO:root:bingo....
WARNING:root:say: 我不看，再让我看打死你
INFO:root:ok
INFO:root:end
------------------------------------------------------------------
----
INFO:root:录音中...
INFO:root:录音结束，识别中...
INFO:root:分析语音
DEBUG:root:{'transcript': '你瞅啥', 'confidence': 0.958637}
INFO:root:(你看看?){1}.*\1
INFO:root:(你看看?)+
INFO:root:(你.+啥)+
INFO:root:bingo....
WARNING:root:say: 咋地啦
INFO:root:ok
INFO:root:end
------------------------------------------------------------------
----
INFO:root:录音中...
INFO:root:录音结束，识别中...
INFO:root:分析语音
DEBUG:root:{'transcript': '磐石', 'confidence': 0.80128425}
INFO:root:(你看看?){1}.*\1
INFO:root:(你看看?)+
INFO:root:(你.+啥)+
INFO:root:(六六六|666)+
INFO:root:(磐|石|老|弟)+
INFO:root:bingo....
WARNING:root:say: 六六六
INFO:root:ok
INFO:root:end
------------------------------------------------------------------
----
INFO:root:录音中...
INFO:root:录音结束，识别中...
INFO:root:分析语音
DEBUG:root:{'transcript': '666', 'confidence': 0.91621482}
INFO:root:(你看看?){1}.*\1
INFO:root:(你看看?)+
INFO:root:(你.+啥)+
INFO:root:(六六六|666)+
INFO:root:bingo....
WARNING:root:say: 要不说磐石老弟六六六呢?
INFO:root:ok
INFO:root:end
------------------------------------------------------------------
----
INFO:root:录音中...
INFO:root:录音结束，识别中...
INFO:root:分析语音
DEBUG:root:{'transcript': '哈哈哈', 'confidence': 0.97387952}
INFO:root:(你看看?){1}.*\1
INFO:root:(你看看?)+
INFO:root:(你.+啥)+
INFO:root:(六六六|666)+
INFO:root:(磐|石|老|弟)+
DEBUG:root:{'transcript': '哈哈哈哈'}
INFO:root:(你看看?){1}.*\1
INFO:root:(你看看?)+
INFO:root:(你.+啥)+
INFO:root:(六六六|666)+
INFO:root:(磐|石|老|弟)+
INFO:root:end
```

```
INFO:root:录音中...
```

一共六次，前5次都可以识别并匹配到，第6次测试期待之外的，不响应。`INFO` 为一般输出，`DEBUG` 输出google服务识别到的结果（不是所有结果，第一条匹配则忽略后面识别的多条结果），`WARNING` 输出响应的语音（因为没有录在文章里听不到，所以输出看看说了什么）

分析第一次和最后一次：

第一次，说 `你看看` 识别出来的第一条结果是 `{'transcript': '你看看', 'confidence': 0.97500253}` ，匹配第一条正则 `(你看看?){1}.*\1` 失败，接着匹配第二条 `(你看看?)+` 成功，break正则，并break识别结果 `test['alternative']` 循环。之后语音输出 `你看看你看看，操不操心` 。

```
INFO:root:(你看看?){1}.*\1
INFO:root:(你看看?)+
INFO:root:bingo....
WARNING:root:say: 你看看你看看，操不操心
```

最后一次，说 `哈哈哈` 共识别出来两条结果 `哈哈哈` 和 `哈哈哈哈` ，

```
{'transcript': '哈哈哈', 'confidence': 0.97387952}
{'transcript': '哈哈哈哈'}
```

各自尝试匹配所有正则均以失败告终

```
INFO:root:(你看看?){1}.*\1
INFO:root:(你看看?)+
INFO:root:(你.+啥)+
INFO:root:(六六六|666)+
INFO:root:(磐|石|老|弟)+
```

没有 `bingo` 只有 `end` ，然后本次识别以未响应结束。

到这里 ~~用不到60行代码~~ 就实现了语音识别并响应的功能。（我不喜欢这样说 `XX行代码就实现了XXX功能` ，公众号里网络上各种关于Python文章充斥着这种标题，很令人反感。代码短是Python那些模块写得好，应该感谢的是各位前辈们，而不是沾沾自喜到起噱头标题并吸引一些浮躁的人前来。告诫自己。）

p.s. 写代码两个多小时，写文章大半天，从一团模糊的概念到语义化，也需得经过思考、组织、融合。有待改进的地方，还请多多指教。

分类: Python

« 上一篇：正则表达式整理
» 下一篇：简单封装Redis做缓存

posted on 2018-12-13 10:20 姜小豆 阅读(4705) 评论(2) 编辑 收藏

Comments

#1楼

谷中天

Posted @ 2019-06-07 23:48

很优秀！楼主

支持(0) 反对(0)

#2楼[楼主]
姜小豆
Posted @ 2019-06-13 09:48
@ 谷中天
嘻嘻嘻，谢谢鼓励

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

相关博文：
· python语音识别终极指南
· Python 语音识别
· flask简单的语音识别
· AzureAI服务之语音识别
· 语音识别语音合成