# 光学字符识别

永吉实验高级中学 2019级10班 赵春旭

2021.4

目录

1. 总述
2. 技术性内容
3. 运行环境
4. 软件源代码
5. 代码解读
6. 使用指南

## 一、总述

OCR （Optical Character Recognition，光学字符识别）是指电子设备（例如扫描仪或数码相机）检查纸上打印的字符，通过检测暗、亮的模式确定其形状，然后用字符识别方法将形状翻译成计算机文字的过程；即，针对印刷体字符，采用光学的方式将纸质文档中的文字转换成为黑白点阵的图像文件，并通过识别软件将图像中的文字转换成文本格式，供文字处理软件进一步编辑加工的技术。

本软件通过调用百度智能云提供的通用文字识别接口进行快速文字识别。通过人工智能进行纠错、识别，可进行多场景、多语种、高精度的文字检测与识别服务，批量处理成百上千张图片并在合并后输出。并添加了认证系统以及一言API，密码采用md5哈希散列，有效保护个人信息。

MonianHello为本人使用的网络用户名，本程序使用的部分代码已在Github上开源，储存库中的所有代码均为原创。

<https://github.com/MonianHello/Python-OCR>

<https://github.com/MonianHello/OCR-GUI>

## 二、技术性内容

### 1.运行环境

此软件基于python 3.8开发，并引入了以下扩展库。如直接运行源代码，缺失以下库会导致程序无法运行

|  |  |  |  |
| --- | --- | --- | --- |
| hashlib | ssl | sys | json |
| os | time | threading | getpass |
| Image | urllib | requests | base64 |

封装版本无需安装环境，支持Windows 7 SP1及以上版本

根据当前系统架构(x64 / x86)选择启动版本

### 软件源代码

(见附件)

### 3.代码解读

#### 3.1 一言

在程序初始部分，添加了hitokoto一言函数，此函数可以在连接到互联网时随机返回一条古诗词

开发者中心：<https://developer.hitokoto.cn>

开放源码仓库：<https://github.com/hitokoto-osc>

##### 接口文档

###### 请求参数

| **参数** | **值** | **可选** | **说明** |
| --- | --- | --- | --- |
| c | 见后表 | 是 | 句子类型 |
| encode | 见后表 | 是 | 返回编码 |
| charset | 见后表 | 是 | 字符集 |
| callback | 如：moe | 是 | 调用的异步函数 |
| select | 默认：.hitokoto | 是 | 选择器。配合 encode=js 使用 |
| min\_length | 默认：0 | 是 | 返回句子的最小长度（包含） |
| max\_length | 默认：30 | 是 | 返回句子的最大长度（包含） |

###### 返回编码（参数）

| **参数** | **说明** |
| --- | --- |
| text | 返回纯洁文本 |
| json | 返回格式化后的 JSON 文本 |
| js | 返回指定选择器的同步调用函数。默认选择器为：.hitokoto |
| 其他 | 返回格式化后的 JSON 文本 |

###### 字符集（参数）

| **参数** | **说明** |
| --- | --- |
| utf-8 | 返回 utf-8 编码的内容 |
| gbk | 返回 gbk 编码的内容。不支持与异步函数同用 |
| 其他 | 返回 utf-8 编码的内容 |

#### 3.2认证系统

本程序使用单一账户，密码存储方式为本地数据库。

由于后期会增加多用户管理与MySQL验证服务器，对密码进行了md5+salt散列

如果直接对密码进行散列，那么黑客可以对通过获得这个密码散列值，然后通过查散列值字典（彩虹表）得到密码。这是不安全的，因此本程序在md5散列时添加了salt，有效防止了暴力破解密码。因没有链接到数据库，本地密码的salt为固定值。在实际生产环境中应对每个用户使用单独的salt加盐值的作用  
 为了加强MD5的[安全](http://www.hackline.net/)性（本身是不可逆的），从而加入了新的算法部分即加盐值，加盐值是随机生成的一组字符串，可以包括随机的大小写字母、数字、字符，位数可以根据要求而不一样，使用不同的加盐值产生的最终密文是不一样的。  
 由于使用加盐值以后的密码相当的安全，即便是你获得了其中的salt和最终密文，破解也是一个耗费相当多时间的过程，可以说是破解单纯MD5的好几倍。以下是加盐散列的步骤：   
1).首先我们得到的是明文的hash值  
2).进行计算获取MD5明文hash值  
3).随机生成加盐值并插入  
4).MD5插入加盐值得到的hash  
5).得到最终的密文  
 目前许多公司都已经引入了hash+salt算法，如百度、阿里、腾讯等互联网公司，甚至著名的Linux开源操作系统早已经加入了这种加密模式。不难得知，这种算法在未来应用会更加广泛。

#### 3.3函数库

fetch\_token()

获取token，参数API\_KEYSECRET\_KEY需提前定义

read\_file(image\_path)

读取文件，以用于对其进行base64编码

request(url, data)

对返回json内容解码

get\_file\_path(root\_path,file\_list,dir\_list)

获取该目录下所有的文件名称和目录名称

ocr(path)

光学字符识别主函数

#### 3.4文件写入

将ocr函数输出的字符存储在列表中，将列表内容遍历写入文件。识别前提供了是否换行的选项，若需要将输出内容换行，则在遍历写入时添加’\n’换行

在识别结束后启动记事本将输出内容打开，方便查阅。

#### 3.5 GUI版本

注：GUI版本尚未完成，存在漏洞

GUI界面使用Pyside2库，后端import OCR.py

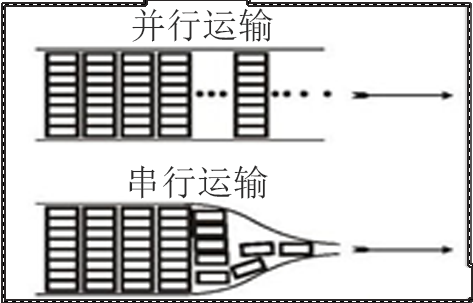
使用QT Designer 设计GUI，并存储在外部main.ui。从文件中加载UI。从 UI 定义中动态创建一个相应的窗口对象，里面的控件对象成为窗口对象的属性。

对信号槽的理解：

某一个对象发生了某个事件，必须让另一个对象做出反应。在Pyside2中，一个对象可以发出一个信号，而另一个对象必须要有接收信号的槽，才能完成一次事件的传递。

在发出信号和接收信号之前，我们还必须连接信号和槽才能生效，如果不连接，那么信号和槽无法对应起来。

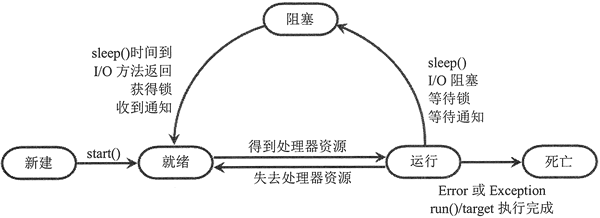
除了信号与槽的链接之外，由于Python的特殊性，在未完成一个事件之前其他事件无法响应，即串行执行任务。



要实现并行执行命令（如一边让进度条移动一边执行识别命令）则需要使用thread来进行多任务操作。为进度条槽函数和光学字符识别主函数创建多个线程，防止线程堵塞。

附：Python生命周期相关参考材料

##### Python线程的生命周期（新建、就绪、运行、阻塞和死亡）



无论是通过 Thread 类直接实例化对象创建线程，还是通过继承自 Thread 类的子类实例化创建线程，新创建的线程在调用 start() 方法之前，不会得到执行，此阶段的线程就处于新建状态。

目前几乎所有的桌面和服务器操作系统，都采用的是抢占式优先级调度策略。即 CPU 会给每一个就绪线程一段固定时间来处理任务，当该时间用完后，系统就会阻止该线程继续使用 CPU 资源，让其他线程获得执行的机会。而对于具体选择那个线程上 CPU，不同的平台采用不同的算法，比如先进先出算法（FIFO）、时间片轮转算法、优先级算法等，每种算法各有优缺点，适用于不同的场景。

对于获得 CPU 调度却未执行完毕的线程，它会转入阻塞状态，待条件成熟之后继续转入就绪状态，重复争取 CPU 资源，直到其执行结束。执行结束的线程将处于死亡状态。

线程执行结束，除了正常执行结束外，如果程序执行过程发生异常（Exception）或者错误（Error），线程也会进入死亡状态。

主线程死亡，并不意味着所有线程全部死亡。也就是说，主线程的死亡，不会影响子线程继续执行；反之也是如此。

对于死亡的线程，无法再调用 start() 方法使其重新启动，否则 Python 解释器将抛出 RuntimeError 异常。

5.使用指南

默认密码：OCRpassword

输入密码时不会在交互窗口里显示，输入密码后回车即可进入

请注意：在编译或运行前请确保填入了正确的ak/sk(API\_KEY / SECRET\_KEY)，否则无法进行识别。

Ak/sk程序内已填入。涉及到隐私问题，如需公开软件源代码，请删除API\_KEY与SECRET\_KEY的值。

首次使用会在C盘根目录下创建 MonianHello\ 文件夹，将图片文件放入文件夹中，软件运行完毕后会在此目录下生成list.txt，存储输出信息。

在运行时请确认互联网连接，未连接到互联网也会导致识别失败。

参赛作品为命令行版本，GUI版本为完成品，仅供参考。

附件：

1. 软件源代码 ocr.py

大小: 7648 字节

MD5: 034719F38D8D0C6BA5857D041A5514C3

SHA1: 98214C06483F36D62C4B5E2F189FFE7C9DB5B26B

CRC32: 7423DBCC

1. 封装文件(64位) OCR-光学字符识别-x64架构.exe

大小: 7832339 字节

MD5: D5A98BBB7DF97826025D5FDBC099C896

SHA1: 092F03084B86AFD8FD22A5B75B67CAFC842E5A98

CRC32: 0C808CD0

1. 封装文件(32位) OCR-光学字符识别-x86架构.exe

大小: 7240157 字节

MD5: 091D141820B2E4FDC9D0515579664D50

SHA1: 9C277DF75A383F02DBBCFB38BEBD3F43AA828450

CRC32: B21CFE6C

1. 私钥/密钥 ak\_sk.txt
2. 示例图片 [Image](dir)
3. 说明文档 说明文档.docx
4. 软件源代码 软件源代码.docx
5. GUI版本 [OCR-GUI](dir)
6. 使用视频 使用帮助.mp4