# 验证码识别

##### ****验证码基础知识****

**验证码设计的目的就是区分“人”和“机器”，所以它在被设计的时候，主要的目的就是防止被采用就是自动识别出来。也叫反自动化技术，可以防止恶意破解密码、刷票、论坛灌水，有效防止黑客对网站、应用程序以暴力破解方式进行不断的身份验证。**

**验证码是很多网站通行的方式，利用比较简易的方式实现了这个功能，同时验证码的样子也会尽量千奇百怪，让机器不能够识别。所以说作为破解者，是处于劣势地位的**

**所以本项目在旨在的问题就是针对市面上各种主流验证码的识别，并运用到在暴力破解密码的过程中的识别**

**各类验证码收集和分类**

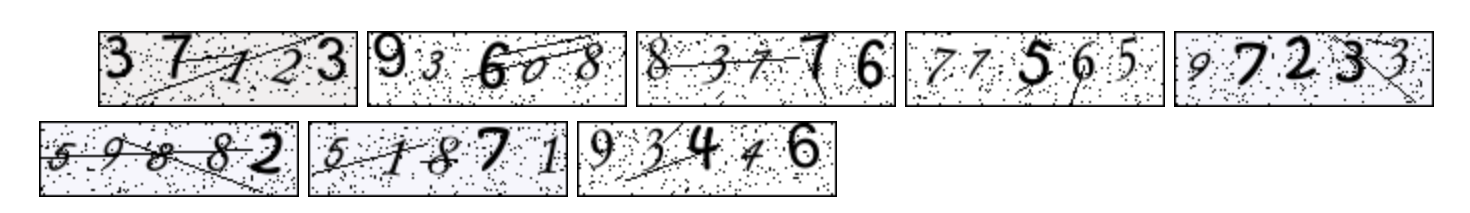
**在做这个验证码识别的功能之前,首先要知晓现在主流的验证码是什么样的**

****第一类****

**1、都是数字**

**2、主要是点干扰，线干扰**

**3、位置基础差不多都是水平的**



****第二类****

**1、前景和背景的颜色是非常接近的**

**2、背景上有线干扰**

**3、位置是波浪形的**

**4 、字母有粘连**



****第三类****

**1、彩色图像，每个字符都不一样；**

**2、背景上白色彩色干扰；**

**3、位置基本是水平的；**

**4 、字母有粘连。**



****第四类****

**1、彩色图像，梯度颜色，其中干扰线和字母前景非常相似；**

**2、三种字体；**

**3、位置不定，不是波浪；**

**4、字母有粘连，当相比之下不是很严重。**



****第五类****

**1、彩色图像，背景随机图片，非常复杂，可以干扰识别，比较难处理**

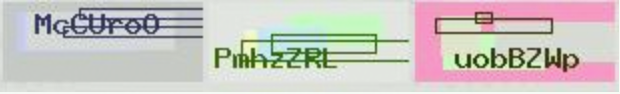
**2、文字可以说设计的非常透明**

**3、文字粘连**



****剩余的其他类型****

**总之各种验证码五花八门，有手写体的，以噪点作为干扰的，大概如下**







**功能的实现**

**首先目前主流的三种处理验证码的相关算法如下：**

**1）图像处理基本算法，比如opencv提供的一系列，此外包括blob分析等**

**2）神经网络相关算法，依然基于opencv提供的svm,annmlp等**

**3）OCR识别打包算法，比如tesseract-ocr**

**本项目采用的是第三种算法，也就是Tessera**

****第一步-验证码的定位****

****思路****

**这的实现思路通过是python用selenium库里webdriver将浏览器最大化，截取整个网页，并保存定位验证码（通过id或class元素实现定位），获取验证码的x,y坐标轴，以及长宽，形成我们需要获取的位置坐标，打开之前保存的网页截图，从中再通过PIL库的截取验证码**

**我们先随便找一个带有验证码的网站，如下**



**我们所要截取的图片就是箭头指向处，前提就是电脑安装selenium和PIL 的python库**

**具体实现代码如下**

**Huiduhua.py**

**# encoding:utf-8**

**from PIL import Image**

**from selenium import webdriver**

**url = 'http://www.isaipu.net/manage/login'**

**driver = webdriver.Chrome()**

**driver.maximize\_window() # 将浏览器最大化**

**driver.get(url)**

**# 截取当前网页并放到D盘下命名为printscreen，该网页有我们需要的验证码**

**driver.save\_screenshot('./img/printscreen.png')**

**imgelement = driver.find\_element\_by\_id('captchaImage') # 定位验证码**

**location = imgelement.location # 获取验证码x,y轴坐标**

**print(location)**

**size = imgelement.size # 获取验证码的长宽**

**print(size)**

**rangle = (int(location['x']+880), int(location['y']+350), int(location['x'] + size['width']+970),**

**int(location['y'] + size['height']+365)) # 写成我们需要截取的位置坐标**

**i = Image.open("./img/printscreen.png") # 打开截图**

**frame4 = i.crop(rangle) # 使用Image的crop函数，从截图中再次截取我们需要的区域**

**frame4=frame4.convert('RGB')**

**frame4.save('./img/save.jpg') # 保存我们接下来的验证码图片 进行打码**

**driver.close()**

**效果**



**可以看到成功截取网页的验证码**

****第二步-对验证码的简化****

**初步得到一个验证码后我们需要对他进行一个简化，验证码内容由阿拉伯数字和小写字母组成，长度为4个字符，字符间偶有重叠，背景噪声与验证码字体明显不同，验证码字符为相近颜色。至此想到对验证码图片像素进行统计分析**

**步骤如下**

**1)遍历每个像素点的RGB值，统计每个颜色出现的次数；**

**2)为体现像素间的差距对RGB进行加权处理，查看加权后的颜色分布；**

**3)通过查看颜色分布设定去掉背景噪声的阈值；**

**4)将保留下的像素填充到与原图尺寸一样的白色背景的图片上；**

**5)去除边框，最后对所得图像灰度化，二值化。**

**实现代码**

**from PIL import Image**

**import matplotlib.pyplot as plt**

**import os**

**#字典转list**

**def dict2list(dic:dict):**

**keys = dic.keys()**

**vals = dic.values()**

**1st = [(key,val) for key, val in zip(keys,vals)]**

**return lst**

**code = Image.open("save.jpg")#打开图片转成RGB格式**

**# 分析图像颜色分布**

**newImg = Image.new("RGB",img.size,(255,255,255, 255))**

**w,h = code.size**

**a={}**

**# 统计颜值出现的次数**

**for i in range(w):**

**for j in range(h):**

**r,g,b = code.getpixel((i,j))**

**t = r\*255\*255 + g\*255+b**

**if t in a.keys():**

**a[t]=a[t]+1**

**else:**

**a[t]=1**

**1 = sorted(dict2list(a), key=lambda x:x[1], reverse=True)**

**dic = {}**

**# 画图显示颜色分布**

**plt.figure(1, figsize=(20,5))**

**plt.scatter(dic.keys(),dic.values(),linewidth=1)**

**plt.show( )**

**# 新增与原图一样大小的图片，用白色填充背景**

**code\_denose = Image.new("RGB", code.size,(255,255,255,255))**

**for i in range(w):**

**for j in range(h):**

**if(i==0 or i== W or 1==W-1 or j==0 or j==h or j==h-1):**

**rmg. putpixe1((i,j), (255,255,255))**

**r,g,b = img.getpixel((i,j))**

**t=r\*255\*255+g\*255+b**

**if(t<5000000): #设定阈值为5000000**

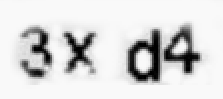
**code\_ denose.putpixe1((1,j), (r,g,b))**

**if(i==0 or i== W or i==W-1 or j==0 or j==h or j==h-1):#去除边框**

**code\_denose.putpixel1((1,j),(255, 255255)**

**rmg.convert('LA').convert('1').save(code\_denose)#灰度化、二值化**

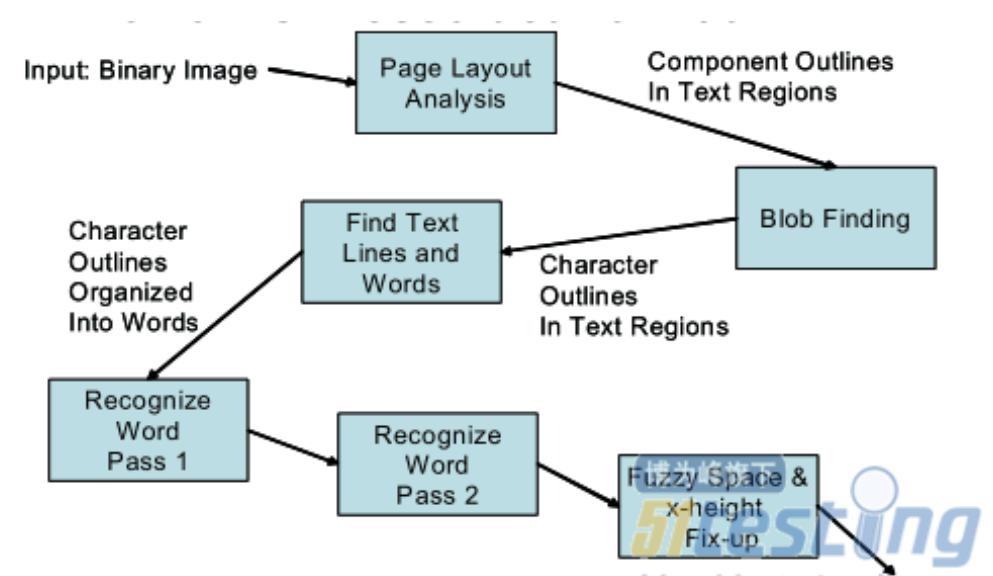
**效果如下**

 **转变**

****第三步-对验证码的识别****

**这里也是最关键的一步，这里我们利用到的是OCR即光学字符识别，是指通过电子设备扫描纸上的打印的字符，然后翻译成计算机文字的过程。也就是说通过输入图片，经过识别引擎，去识别图片上的文字。Tesseract是一种适用于各种操作系统的光学字符识别引擎，本功能通过该识别引擎对预处理后的验证码进行识别。**

**步骤大概如下**

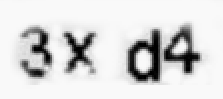


**1)连通区域分析，检测出字符区域区域以及子轮廓。在此阶段轮廓线集成为块区域。**

**2)由字符轮廓和块区域得出文本行，以及通过空格识别出单词。固定字宽文本通过字符单元分割出单个字符，对比例文本通过一定的间隔和模糊间隔来分割。**

**3)依次对每个单词进行分析，采用自适应分类器，分类器有学习能力，先分析且满足条件的单词也作为训练样本，所以后面的字符识别更准确。**

**首先对单个验证码图片进行识别，Windows系统进入命令行模式，进入验证码图片所在的目录，运行tesseract name.png textname（其中name.png要被识别的验证码图片文件名，textname为识别后的验证码字符存放的文件名，默认格式为TXT文件）。随机选取两个，识别结果如下：**

**识别的结果为3xd4**

**具体实现代码在文件夹中 yzm.py**