

python验证码识别爆破_python-i春秋验证码识别

[weixin_39781143](#) 于 2020-12-09 23:02:54 发布 53 收藏

文章标签: [python验证码识别爆破](#)

python+机器学习+验证码识别+源码

简单介绍

最近在写某网站的自动注册，在注册的过程中遇到一些问题，如js的执行、验证码的识别等等，今天给大家如何用python通过机器学习的方式实现验证码的识别，这次以i春秋验证码为例进行识别，尽可能的用简单的方式给大家讲解。

使用技术[x] python 2.7 32 位 所需的库 PIL sklearn numpy pandas matplotlib

[x] 数字图像处理基础(中值滤波等)

[x] 机器学习KNN算法

获取验证码headers={

```
"Cookie":"XXXXXXXXXXXXXXXXX",
```

```
"Referer":"https://user.ichunqiu.com/register/index",
```

```
"Upgrade-Insecure-Requests":"1",
```

```
"User-Agent":"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36"
```

```
}
```

简单写一下下载验证码的代码url="https://user.ichunqiu.com/login/verify_image?d=1523697519026"

```
for i in range(0,51):
```

```
with open("./pic/{}.png".format(i),"wb") as f:
```

```
print("{} pic downloading...".format(i))
```

```
f.write(requests.get(url,headers=headers).content)
```

在pic目录下获得50张验证码照片了

验证码处理

ok! 既然获取到验证码了，肯定要对验证码进行一定程度的处理!

像上图这样的二维码，怎么做训练么?! 最起码要给它去掉脸上的雀斑(点点)，修复脸上的刀疤(条纹)，怎么办呢? 难道要寄出神器PS? 不存在的，兄弟! 我们要做的是批量处理，用ps一张张的人工处理，很明显不符合懒人的风格，怎么办?

下面肯定是祭出数字图像处理了，有些同学看到这里就发怵，没关系，这里没有什么技术难度的! 相信我，也相信自己。我突然，发现我有给人灌鸡汤的天赋! 下面正式开始讲解!

1.首先，观察原图像(t2.png)! 即

是不是发现它的色彩十分多呀！在数字图像领域中，我们一般研究灰度图像，那么？问题来了什么是灰度图像呢？简单地说，整幅图片都是灰色的！原图片经过灰度处理是这样的。

是不是都是灰色的呢？但是我们得到的灰度图，有些地方比较浅，有些地方颜色比较深，这是为什么呢？下面是重点内容，都给我听好啦！图像在计算机中是用矩阵进行存储的，可能，你不知道矩阵是什么东西。那少年我推荐你一本《工程数学 线性代数》看完一遍保你收获颇丰！当然啦，实在不想看，可以去百度百科呀，矩阵定义，怎么样我贴心吧？！矩阵中每一个数代表着一个像素！数值的大小反映在图像中就是图像的深浅！值越大，颜色越深！这就说明了，灰度图中颜色深浅的问题，好吧！道理我都懂，怎么转化成灰度图呢？代码如下：

```
# encoding=utf-
8fromPILimportImage,ImageFilterimportmatplotlib.pyplotaspltimportnumpyasnpimportpandasaspdimportmatplotlib
导入相应的库img=Image.open("t2.png")# 载入t2.pngim=img.convert("L")# 这里重点！img图像转化为灰度图像
im、plt.subplot(4,2,1)# 关于plt这个用法在网上有很多教程，我在这里就不一一赘述了
plt.imshow(np.array(im),cmap=cm.gray)
```

2.现在我们得到灰度图像：

观察发现，验证码都是深颜色的，要不把浅颜色的“雀斑”去掉，怎么去掉呢？肯定在矩阵中找一个值(一般称之为阈值)，小于这个值的数都变成零(白色)，大于这个数的值都变成最大值(黑)，咦！灰度图像变成了黑白图像！怎么找这个阈值呢？一般用矩阵的平均数或者中位数，纳尼？什么是中位数，找一个小学课本翻一番。好！那我们分别取平均数和中位数，做一下处理看一下效果！

平均数效果如下：

中位数效果如下：

那么肯定是用平均数做呀！代码如下！

```
# encoding=utf-
8fromPILimportImage,ImageFilterimportmatplotlib.pyplotaspltimportnumpyasnpimportpandasaspdimportmatplotlib
im转化为im_z矩阵print(im_z.mean())# im_z的平均值print(np.median(im_z))# im_z的中位数
plt.subplot(4,2,1)plt.imshow(np.array(im),cmap=cm.gray)plt.subplot(4,2,2)plt.imshow(img)im_b=im.point(lambda
用im_z的平均值197转化为黑白图像im_bplt.subplot(4,2,3)plt.imshow(im_b)# 显示
im_bim_a=im.point(lambda i:i>220,mode='1')# 用im_z的平均值220转化为黑白图像
im_aplt.subplot(4,2,4)plt.imshow(im_a)# 显示im_a
```

3.得到黑白图像了，可还是有“雀斑”怎么办，用上神奇的中值滤波啦！定义，一般都是在灰度图上进行中值滤波，我也是突然想看一下，在黑白图像的处理结果怎么样(当然，这里的中值只能是零或一啦)，结果还不错！

在这里要说的是，中值滤波这个概念，比较容易理解，我这个小菜鸡，概括的不怎么样！只能委屈大家看一下优秀的博客啦。处理代码如下

```
# encoding=utf-
8fromPILimportImage,ImageFilterimportmatplotlib.pyplotaspltimportnumpyasnpimportpandasaspdimportmatplotlib
下面是核心代码#####im_f=im_b.filter(ImageFilter.MedianFilter(size=3))# 用 im_b图像进行中值滤波，掩模用的3×3的模板plt.subplot(4,2,5)plt.imshow(im_f)# 显示 im_fplt.show()
```

4.得到干净的验证码，这样就结束了么？no!no!no! 紧接着，要把单个的字符分割出来！怎么分割来？下面我会给出代码，在代码中给出解释

```
# encoding=utf-8
from PIL import Image, ImageFilter
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import matplotlib

下面是核心代码#####
im = im_fprint(im.size)
a = np.array(im) # im转化为a矩阵
pd.DataFrame(a.sum(axis=0)).plot.line() # 画出每列的像素累计值
plt.imshow(a, cmap='gray') # 画出图像
split_lines = [7, 25, 44, 60, 78] # 经过调整过的分割线的合理间距
vlines = [plt.axvline(i, color='r') for i in split_lines] # 画出分割线
plt.show()
```

得到图像如下

啧啧！我都有点佩服我自己呢！下面要把字符一个个的分割出来！同时要把上边和下边的黑线去掉！

怎么办呢？talk is cheap, show my code!

```
# encoding=utf-8
from PIL import Image, ImageFilter
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import matplotlib

a = np.array(im)
pd.DataFrame(a.sum(axis=0)).plot.line() # 画出每列的像素累计值
plt.imshow(a, cmap='gray') # 画出图像
split_lines = [7, 25, 44, 60, 78]
vlines = [plt.axvline(i, color='r') for i in split_lines] # 画出分割线
plt.show() # im.crop()#####核心代码#####
y_min = 5
y_max = 35 # 设置获取图像的高和宽
ims = []
for x_min, x_max in zip(split_lines[:-1], split_lines[1:]):
    im = im.crop([x_min, y_min, x_max, y_max]).save(str(c) + '.png') # crop()函数是截取指定图像！ # save保存图像！
    c = c + 1
for i in range(1, 5):
    file_name = "{}.png".format(i)
    plt.subplot(4, 2, i)
    im = Image.open(file_name).convert("1") # im=img.filter(ImageFilter.MedianFilter(size=3))
    plt.imshow(im) # 显示截取的图像！
plt.show()
```

得到的图像结果如下！

5.还记得咱们在第一步中获取的50张图片么？没错！把他们都分割出来吧！整理一下代码！ # encoding=utf-8

```
from sklearn.neighbors import KNeighborsClassifier as KNN

from sklearn.externals import joblib

from PIL import Image, ImageFilter

import numpy as np

def imgprocess(name):

# 图片分割成单个字符！

path = "./pic/{}.png".format(str(name))

img = Image.open(path).convert("L")

th = np.array(img).mean()

im_b = img.point(lambda i: i > th, mode='1')

im_f = im_b.filter(ImageFilter.MedianFilter(size=3))
```

```

split_lines = [7, 25, 43, 61, 79]

y_min = 5

y_max = 35

ims = []

c = 1

for x_min, x_max in zip(split_lines[:-1], split_lines[1:]):

im_f.crop([x_min, y_min, x_max, y_max]).save('./pic2later/{}-{}.png'.format(str(name),str(c)))

c = c + 1

for i in range(1,51):

print("process {} pic".format(i))

imgprocess(i)

```

机器学习之KNN

1.关于knn算法呢，我以前做过一些笔记，我po上去了
大家记得看一下！

2.相应的代码及注解# encoding=utf-8

```

from sklearn.neighbors import KNeighborsClassifier as KNN

from sklearn.externals import joblib

from PIL import Image,ImageFilter

import numpy as np

def imgprocess(name):

path="./pic/{}.png".format(str(name))

img=Image.open(path).convert("L")

th=np.array(img).mean()

im_b = img.point(lambda i: i > th, mode='1')

im_f= im_b.filter(ImageFilter.MedianFilter(size=3))

split_lines = [7, 25, 43, 61, 79]

y_min = 5

y_max = 35

ims = []

c = 1

for x_min, x_max in zip(split_lines[:-1], split_lines[1:]):

```

```

im_f.crop([x_min, y_min, x_max, y_max]).save('./pic2later/{}-{}.png'.format(str(name),str(c)))

c = c + 1

for i in range(1,51):

print("process {} pic".format(i))

imgprocess(i)

"""

#####下面是核心代码#####

"""

def Y():

# 获取字符的值！ 这些验证码是我一个一个肉眼写出来的！

with open("./pic/reslut.txt") as f:

Y=list(f.read().replace("\n",""))

return Y

def getX():

# 获取X的值！

path="./pic2later/{}-{}.png"

X=[]

for i in range(1,51):

for c in range(1,5):

img=Image.open(path.format(str(i),str(c)))

ls = np.array(img).tolist()

xx = []

for l in ls:

for x in l:

xx.append(x)

X.append(xx)

return X

def train():

# 用knn模型进行训练

knn = KNN()

knn.fit(getX(), Y())

```

```

joblib.dump(knn, "./model.pkl")

# 保存结果！

train()

写到这里了，干脆把预测代码也写一下吧！跟训练代码差不多！# encoding=utf8

from sklearn.neighbors import KNeighborsClassifier as KNN

from sklearn.externals import joblib

from PIL import Image, ImageFilter

import numpy as np

MODEL_PATH="./model.pkl"

def getX(file_name):

X=[]

img=Image.open(file_name).convert("L")

th=np.array(img).mean()

im_b = img.point(lambda i: i > th, mode='1')

im_f = im_b.filter(ImageFilter.MedianFilter(size=3))

split_lines = [7, 25, 43, 61, 79]

y_min = 5

y_max = 35

for x_min, x_max in zip(split_lines[:-1], split_lines[1:]):

ls=np.array(im_f.crop([x_min, y_min, x_max, y_max])).tolist()

xx=[]

for l in ls:

for x in l:

xx.append(x)

X.append(xx)

return X

def predict(file_path):

# 预测

knn=joblib.load(MODEL_PATH)

X=getX(file_path)

Y=knn.predict(X)

```

```
return "".join(Y)
```

```
print(predict("./pic/1.png"))
```

总结

1.预测结果可能不准确，原因是因为只有50张验证码进行训练

2.懒得对它进行评估了，你们随意就好

3.验证码识别的一般套路: 灰度化、图像处理、二值化、选算法、训练、评估调整参数、预测，当然，我在这里二值化与处理的顺序换了一下，灵活处理哈！

代码：

在下载代码之前先说几句！自己先运行一下train()函数重新生成model.pkl，在进行预测！至于为什么，32位与64位是不兼容的！最后！厚着脸皮！腆着脸！说要！要点赞！要回复哟！嘤嘤嘤！有什么问题可以，在评论区提问哟！

密码在这里！密码在这里！密码在这里！密码在这里！密码在这里！