本节我们首先来尝试识别最简单的一种验证码，图形验证码，这种验证码出现的最早，现在也很常见，一般是四位字母或者数字组成的，例如中国知网的注册页面就有类似的验证码，链接为：[http://my.cnki.net/elibregister/commonRegister.aspx](http://my.cnki.net/elibregister/commonRegister.aspx" \o ")，页面如图 8-1 所示：



图 8-1 知网注册页面

表单的最后一项就是图形验证码，我们必须完全输入正确图中的字符才可以完成注册。

### 1. 本节目标

本节我们就以知网的验证码为例，讲解一下利用 OCR 技术识别此种图形验证码的方法。

### 2. 准备工作

识别图形验证码需要的库有 Tesserocr，如果没有安装可以参考第一章的安装说明。

### 3. 获取验证码

为了便于实验，我们先将验证码的图片保存到本地，以供测试。

打开开发者工具，找到验证码元素，可以看到这是一张图片，它的 src 属性是 CheckCode.aspx，在这里我们直接将这个链接打开：[http://my.cnki.net/elibregister/CheckCode.aspx](http://my.cnki.net/elibregister/CheckCode.aspx" \o ")，就可以看到一个验证码，直接右键保存下来即可，将名称命名为 code.jpg，如图 8-2 所示：

IMG_257

图 8-2 验证码

这样我们就可以得到一张验证码图片供下面测试识别使用了。

### 4. 识别测试

接下来我们新建一个项目，将验证码图片放到项目根目录下，用 Tesserocr 库来识别一下该验证码试试，代码如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | import tesserocr  from PIL import Image    image = Image.open('code.jpg')  result = tesserocr.image\_to\_text(image)  print(result) |

在这里我们首先新建了一个 Image 对象，然后调用了 Tesserocr 的 image\_to\_text() 方法，传入该 Image 对象即可完成识别，实现过程非常简单，识别结果如下：

|  |  |
| --- | --- |
| 1 | JR42 |

另外 Tesserocr 还有一个更加简单的方法直接将图片文件转为字符串可以达到同样的效果，代码如下：

|  |  |
| --- | --- |
| 1  2 | import tesserocr  print(tesserocr.file\_to\_text('image.png')) |

不过经测试此种方法的识别效果不如上一种方法好。

### 5. 验证码处理

如上的图片识别基本没有难度，只是新建一个 Image 对象，然后调用 image\_to\_text() 方法即可得出图片的识别结果。

接下来我们换一个验证码试一下，命名为 code2.jpg，如图 8-3 所示：

IMG_258

图 8-3 验证码

重新用下面的代码测试一下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | import tesserocr  from PIL import Image    image = Image.open('code2.jpg')  result = tesserocr.image\_to\_text(image)  print(result) |

这时可以看到如下输出结果：

|  |  |
| --- | --- |
| 1 | FFKT |

发现这次识别和实际的结果有所偏差，这是因为验证码内的多余线条干扰了图片的识别。

对于这种情况，我们还需要做一下额外的处理，如转灰度、二值化等操作。

我们可以利用 Image 对象的 convert() 方法参数传入 L 即可将图片转化为灰度图像，代码如下：

|  |  |
| --- | --- |
| 1  2 | image = image.convert('L')  image.show() |

传入 1 即可将图片进行二值化处理：

|  |  |
| --- | --- |
| 1  2 | image = image.convert('1')  image.show() |

另外我们还可以指定二值化的阈值，上面的方法采用的是默认阈值127，不过我们不能用原图直接转化，可以先转为灰度图像，然后再指定二值化阈值转化，代码如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | image = image.convert('L')  threshold = 80  table = []  **for** i **in** range(256):  **if** i < threshold:          table.append(0)  **else**:          table.append(1)    image = image.point(table, '1')  image.show() |

在这里我们指定了一个变量 threshold 代表二值化阈值，阈值设置为 80，处理之后我们看一下结果，如图 8-4 所示：

IMG_259

图 8-4 处理结果

经过处理之后我们发现原来的验证码中的线条已经被去除了，而且整个验证码变得黑白分明，这时重新识别验证码，代码如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | import tesserocr  from PIL import Image    image = Image.open('code2.jpg')    image = image.convert('L')  threshold = 127  table = []  **for** i **in** range(256):  **if** i < threshold:          table.append(0)  **else**:          table.append(1)    image = image.point(table, '1')  result = tesserocr.image\_to\_text(image)  print(result) |

即可发现运行结果变成了：

|  |  |
| --- | --- |
| 1 | PFRT |

识别正确。

可见对于一些有干扰的图片，我们做一些灰度和二值化处理，会提高其识别正确率。

### 6. 本节代码

本节代码地址为：[https://github.com/Python3WebSpider/CrackImageCode](https://github.com/Python3WebSpider/CrackImageCode" \o ")。

### 7. 结语

本节我们了解了利用 Tesserocr 识别验证码的过程，对于简单的图形验证码我们可以直接用它来得到结果，如果要提高识别的准确度还可以对验证码图片做一下预处理。