机器视觉

从 Google 的无人驾驶汽车到可以识别假钞的自动售卖机，机器视觉一直都是一个应用广 泛且具有深远的影响和雄伟的愿景的领域。

我们将重点介绍机器视觉的一个分支：文字识别，介绍如何用一些 Python 库来识别和使用在线图片中的文字。

我们可以很轻松的阅读图片里的文字，但是机器阅读这些图片就会非常困难，利用这种人类用户可以正常读取但是大多数机器人都没法读取的图片，验证码(CAPTCHA)就出现了。验证码读取的难易程度也大不相同，有些验证码比其他的更加难读。

将图像翻译成文字一般被称为 **光学文字识别(Optical Character Recognition, OCR)**。可以实现 OCR 的底层库并不多,目前很多库都是使用共同的几个底层 OCR 库,或者是在上面 进行定制。

Tesseract

在读取和处理图像、图像相关的机器学习以及创建图像等任务中，Python 一直都是非常出色的语言。虽然有很多库可以进行图像处理，但在这里我们只重点介绍：**Tesseract**

Tesseract

Tesseract 是一个 OCR 库,目前由 Google 赞助(Google 也是一家以 OCR 和机器学习技术闻名于世的公司)。Tesseract 是目前公认最优秀、最精确的开源 OCR 系统，除了极高的精确度，Tesseract 也具有很高的灵活性。它可以通过训练识别出任何字体，也可以识别出任何 Unicode 字符。

安装 Tesseract

Windows 系统

下载可执行安装文件：https://code.google.com/p/tesseract-ocr/downloads/list 安装。

Ubuntu Linux 系统

可以通过 apt-get 安装: `$sudo apt-get tesseract-ocr`

Mac OS X 系统

用 Homebrew 可以很方便地安装： `brew install tesseract`

要使用 Tesseract 的功能，比如后面的示例中训练程序识别字母，要先在系统中设置一 个新的环境变量 `$TESSDATA_PREFIX`，让 Tesseract 知道训练的数据文件存储在哪里，然后搞一份 tessdata 数据文件，放到 Tesseract 目录下。

- 在大多数 Linux 系统和 Mac OS X 系统上,你可以这么设置（假设 Tesseract 数据文件目录在/usr/local/share/下）: **$export TESSDATA_PREFIX=/usr/local/share/Tesseract**
- 在 Windows 系统上也类似,你可以通过下面这行命令设置环境变量: **#setx TESSDATA_PREFIX C:\Program Files\Tesseract OCR\Tesseract**

安装 pytesseract

Tesseract 是一个命令行工具，安装之后，要用 tesseract 命令在 Python 的外面运行，但我们可以通过 pip 安装支持 Python 版本的 Tesseract 库：pytesseract

```
pip install pytesseract
```

处理规范文字

处理的大多数文字最好都是比较干净、格式规范的。格式规范的文字通常可以满足一些需求，通常格式规范的文字具有以下特点:

- 使用一个标准字体(不包含手写体、草书,或者十分“花哨的”字体)
- 即使被复印或拍照，字体还是很清晰，没有多余的痕迹或污点
- 排列整齐，没有歪歪斜斜的字
- 没有超出图片范围，也没有残缺不全，或紧紧贴在图片的边缘

文字的一些格式问题在图片预处理时可以进行解决。例如,可以把图片转换成灰度图，调整亮度和对比度，还可以根据需要进行裁剪和旋转（详情需要了解图像与信号处理）等。

文字处理示例 1

通过命令运行 Tesseract，读取文件并把结果写到一个文本文件中:

```
tesseract test.jpg text
```

识别结果很准确,直接输出 JR4

通过 Python 代码实现

```
import pytesseract
from PIL import Image

image = Image.open('test.jpg')
text = pytesseract.image_to_string(image)
print(text)
```

运行结果：

```
JR42
```

文字处理示例 2

很多时候我们在网上会看到这样的图片：

Tesseract 不能完整处理这个图片,最终结果是这样:

PFKT

主要是因为验证码内的多余线条干扰了图片的识别，这个时候我们需要做一下额外的处理，例如转灰度、二值化等操作，从而让图片更加清晰，便于 Tesseract 读取:

```
import pytesseract
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')
image.show()
```

通过一个阈值对前面的“模糊”图片进行过滤的结果

最后可以发现，运行结果已经正确的输出了图片的文字：**PFRT**

从网站图片中抓取文字

用 Tesseract 读取硬盘里图片上的文字,可能不怎么令人兴奋,但当我们把它和网络爬虫组合使用时,就能成为一个强大的工具。

网站上的图片可能并不是故意把文字做得很花哨 (就像餐馆菜单的 JPG 图片上的艺术字),但它们上面的文字对网络爬虫来说就是隐藏起来 了，举个例子：

- 虽然亚马逊的 robots.txt 文件允许抓取网站的产品页面,但是图书的预览页通常不让网络机器人采集。
- 图书的预览页是通过用户触发 Ajax 脚本进行加载的,预览图片隐藏在 div 节点下面;其实,普通的访问者会觉得它们看起来更像是一个 Flash 动画,而不是一个图片文 件。当然,即使我们能获得图片,要把它们读成文字也没那么简单。

- 下面的程序就解决了这个问题:首先导航到托尔斯泰的《战争与和平》的大字号印刷版 1, 打开阅读器,收集图片的 URL 链接,然后下载图片,识别图片,最后打印每个图片的文 字。因为这个程序很复杂,利用了前面几章的多个程序片段,所以我增加了一些注释以让 每段代码的目的更加清晰:

```
import time
from urllib.request import urlretrieve
import subprocess
from selenium import webdriver
#创建新的Selenium driver
driver = webdriver.PhantomJS()

# 用Selenium 试试Firefox 浏览器:
# driver = webdriver.Firefox()

driver.get("http://www.amazon.com/War-Peace-Leo-Nikolayevich-Tolstoy/dp
/1427030200")
# 单击图书预览按钮 driver.find_element_by_id("sitbLogoImg").click() imag
eList = set()
# 等待页面加载完成
time.sleep(5)
# 当向右箭头可以点击时,开始翻页
while "pointer" in driver.find_element_by_id("sitbReaderRightPageTurner
").get_attribute("style"):
    driver.find_element_by_id("sitbReaderRightPageTurner").click()
    time.sleep(2)
    # 获取已加载的新页面(一次可以加载多个页面,但是重复的页面不能加载到集合中)

    pages = driver.find_elements_by_xpath("//div[@class='pageImage']/di
v/img")
    for page in pages:
        image = page.get_attribute("src")
        imageList.add(image)
driver.quit()

# 用Tesseract 处理我们收集的图片URL 链接
for image in sorted(imageList):
    # 保存图片
    urlretrieve(image, "page.jpg")
    p = subprocess.Popen(["tesseract", "page.jpg", "page"], stdout=subp
rocess.PIPE,stderr=subprocess.PIPE)
    f = open("page.txt", "r")
    p.wait()
    print(f.read())
```

和我们前面使用 Tesseract 读取的效果一样，这个程序也会完美地打印书中很多长长的段落，第六页的预览如下所示:

```
6
      "A word of friendly advice, mon
      cher. Be off as soon as you can,
      that's all I have to tell you. Happy
      he who has ears to hear. Good-by,
      my dear fellow. Oh, by the by!" he
      shouted through the doorway after
      Pierre, "is it true that the countess
      has fallen into the clutches of the
      holy fathers of the Society of je-
      sus?"

      Pierre did not answer and left Ros-
      topchin's room more sullen and an-
      gry than he had ever before shown
      himself.
```

但是当文字出现在彩色封面上时，结果就不那么完美了:

```
WEI' nrrd Peace
Len Nlkelayevldu Iolfluy
Readmg shmdd be ax
wlnvame asnossxble Wenfler
an mm m our cram: Llhvary
 - Leo Tmsloy was a Russian rwovelwst
 I and moval phflmopher med lur
 A ms Ideas 01 nonviolenx reswslance m 5 We range     0, "and"
```

如果想把文字加工成普通人可以看懂的效果，还需要花很多时间去处理。

比如，通过给 Tesseract 提供大量已知的文字与图片映射集，经过训练 Tesseract 就可以“学会”识别同一种字体，而且可以达到极高的精确率和准确率，甚至可以忽略图片中文字的背景色和相对位置等问题。

案例：尝试对知乎网验证码进行处理：

许多流行的内容管理系统即使加了验证码模块，其众所周知的注册页面也经常会遭到网络 机器人的垃圾注册。

那么，这些网络机器人究，竟是怎么做的呢?既然我们已经，可以成功地识别出保存在电脑上 的验证码了，那么如何才能实现一个全能的网络机器人呢?

大多数网站生成的验证码图片都具有以下属性。

- 它们是服务器端的程序动态生成的图片。验证码图片的 src 属性可能和普通图片不太一 样，比如 `<img src="WebForm.aspx?id=8AP85CQKE9TJ">`，但是可以和其他图片一样进行 下载和处理。
- 图片的答案存储在服务器端的数据库里。

- 很多验证码都有时间限制，如果你太长时间没解决就会失效。
- 常用的处理方法就是，首先把验证码图片下载到硬盘里，清理干净，然后用 Tesseract 处理 图片，最后返回符合网站要求的识别结果。

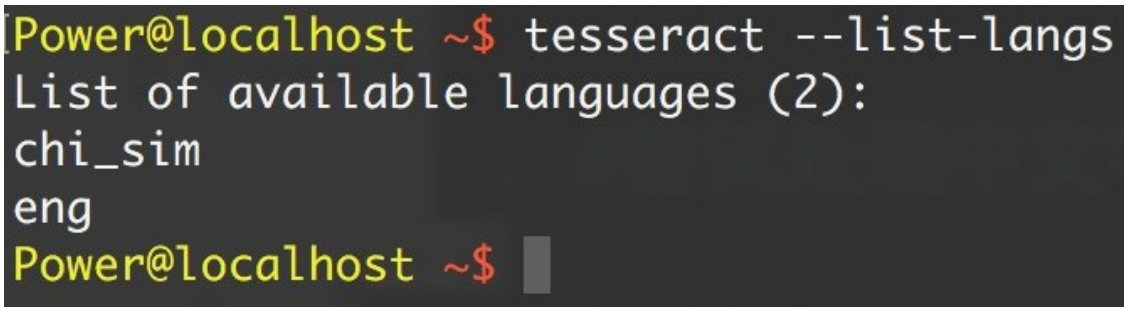
```
# -*- coding:utf-8 -*-

import requests
import time
import pytesseract
from PIL import Image
from bs4 import BeautifulSoup

def captcha(data):
    with open('captcha.jpg','wb') as fp:
        fp.write(data)
    time.sleep(1)
    image = Image.open("captcha.jpg")
    text = pytesseract.image_to_string(image)
    print "机器识别后的验证码为：" + text
    command = raw_input("请输入 Y 表示同意使用，按其他键自行重新输入：")
    if (command == "Y" or command == "y"):
        return text
    else:
        return raw_input('输入验证码：')

def zhihuLogin(username,password):

    # 构建一个保存 Cookie 值的 session 对象
    sessiona = requests.Session()
    headers = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:47.0) Gecko/20100101 Firefox/47.0'}

    # 先获取页面信息，找到需要 POST 的数据（并且已记录当前页面的 Cookie）
    html = sessiona.get('https://www.zhihu.com/#signin', headers=header
s).content

    # 找到 name 属性值为 _xsrf 的 input 标签，取出 value 里的值
    _xsrf = BeautifulSoup(html ,'lxml').find('input', attrs={'name':'_x
srf'}).get('value')

    # 取出验证码，r 后面的值是 Unix 时间戳,time.time()
    captcha_url = 'https://www.zhihu.com/captcha.gif?r=%d&type=login' %
 (time.time() * 1000)
    response = sessiona.get(captcha_url, headers = headers)


    data = {
```

```
        "_xsrf":_xsrf,
        "email":username,
        "password":password,
        "remember_me":True,
        "captcha": captcha(response.content)
    }

    response = sessiona.post('https://www.zhihu.com/login/email', data
= data, headers=headers)
    print response.text

    response = sessiona.get('https://www.zhihu.com/people/maozhaojun/ac
tivities', headers=headers)
    print response.text


if __name__ == "__main__":
    #username = raw_input("username")
    #password = raw_input("password")
    zhihuLogin('xxxx@qq.com','ALAxxxxIME')
```

处理中文字符

如果手头上有中文的训练数据，也可以尝试对中文进行识别。

命令：`tesseract --list-langs` 可以查看当前支持的语言，`chi_sim` 表示支持简体中文。



那么在使用时候，可以指定某个语言来进行识别，如：

```
tesseract -l chi_sim paixu.png paixu
```

模板:排序算法

查 · 论 · 编	排序算法 [隐藏]
理论	计算复杂性理论 · 大O符号 · 全序关系 · 列表 · 稳定性 · 比较排序 · 自适应排序 · 排序网络 · 整数排序
交换排序	冒泡排序 · 鸡尾酒排序 · 奇偶排序 · 梳排序 · 侏儒排序 · 快速排序 · 臭皮匠排序 · Bogo排序
选择排序	选择排序 · 堆排序 · 平滑排序 · 笛卡尔树排序 · 锦标赛排序 · 圃排序
插入排序	插入排序 · 希尔排序 · Splay排序 · 二叉查找树排序 · 图书馆排序 · 耐心排序
归并排序	归并排序 · 梯级归并排序 · 振荡归并排序 · 多相归并排序 · 串列排序
分布排序	美国旗帜排序 · 珠排序 · 桶排序 · 爆炸排序 · 计数排序 · 鸽巢排序 · 相邻图排序 · 基数排序 · 闪电排序 · 插值排序
并发排序	双调排序器 · Batcher归并网络 · 再两排序网络
混合排序	区块排序 · Tim排序 · 内省排序 · Spread排序 · J排序
其他	拓扑排序 · 煎饼排序 · 意粉排序

表现在程序里，则可以这么写：

```
# -*- coding:utf-8 -*-

from PIL import Image
import subprocess

def cleanFile(filePath):
    image = Image.open(filePath)

    # 调用系统的tesseract 命令, 对图片进行OCR 中文识别
    subprocess.call(["tesseract", "-l", "chi_sim", filePath, "paixu"])

    # 打开文件读取结果
    with open("paixu.txt", 'r') as f:
        print(f.read())

if __name__ == "__main__":
    cleanFile("paixu.png")
```

结果如下：

Tesseract Open Source OCR Engine v3.05.00 with Leptonica
模板:排序算法

查 - 论 - 编 排序算法 臆翻
理论 计算复杂性理论 - 大〇符号 - 全序关系 - 列表 - 稳定性 - 比较排序 - 自适应排序 - 排序网络 - 整数排序

交换排序 冒泡排序 - 鸡尾酒排序 - 奇偶排序 - 梳排序 - 侏儒排序 - 快速排序 - 臭皮匠排序 - Bogo排序

选择排序 选择排序 - 堆排序 - 平滑排序 - 笛卡尔树排序 - 锦标赛排序 - -排序

撞入排序 插入排序 - 希尔排序 - Sploy排序 - 二叉查找树排序 - 图书馆排序 - 耐心排序

归并排序 归并排序 - 梯级归并排序 - 振荡归并排序 - 多相归并排序 - 串列排序

分布排序 美国旗帜排序 - 珠排序 - 桶排序 - 爆炸排序 - 计数排序 - 鸽巢排序 - 相邻图排序 - 基数排序 - 闪电排序 - 插值排序

并发排序 双调排序器 - BOtCher归并网络 - 两两排序网络

混台排序 区块排序 - fm排序 - 内省排序 - Spread排序 - 腓序
其他 拓扑排序 - 煎饼排序 - 意粉排序