

# 法律声明

---

□ 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

□ 课程详情请咨询

■ 微信公众号：大数据分析挖掘

■ 新浪微博：ChinaHadoop



---

# 分布式爬虫

# 大纲

---

- Logger
- Daemon
- Tesseract-OCR
- 图片相似度匹配
- 在线接口的使用

---

# Logger

# Log 系统基本用途

---

- 多线程情况下，debug 调试非常困难
- 错误出现可能有一些随机性
- 性能分析
- 错误记录与分析
- 运行状态的实时监测

# Log 系统设计

---

- 错误级别: Debug, Info, Warning, Error, Fatal
- 日志的来源（通道）:MySQL, Connection, Threading, etc.
- 日志输出位置: File, console, database

# Python 日志系统

---

- **loggers:** 创建日志并指明文件
- **handlers:** 处理器，配置过滤器、输出等
- **filters:** 配置过滤规则
- **formatters:** 配置输出的日志格式

# logging.Filter

控制过滤的规则，重  
写 `filter` 方法来进行  
过滤。构造函数的参  
数通过 `config` 来制  
定和传入

```
class SpiderFilter(logging.Filter):  
  
    def __init__(self, allow=None, disable=None):  
        self.allow_channels = allow  
        self.disable_channels = disable  
  
    def filter(self, record):  
        if self.allow_channels is not None:  
            if record.name in self.allow_channels:  
                allow = True  
            else:  
                allow = False  
        elif self.disable_channels is not None:  
            if record.name in self.disable_channels:  
                allow = False  
            else:  
                allow = True  
        else:  
            allow = False  
        return allow
```



---

# Daemon

# Daemontool

---

Spider always get crash or halt so we need a daemon tool to monitor its status, start it automatically ( can be controlled by master ), especially restart it when it's crashed

daemontools is a collection of tools for managing UNIX services. supervise monitors a service. It starts the service and restarts the service if it dies. Setting up a new service is easy: all supervise needs is a directory with a run script that runs the service.

# Installation

---

<http://cr.yp.to/daemontools/install.html>

## Installation

Create a /package directory:

```
# mkdir -p /package  
# chmod 1755 /package  
# cd /package
```

Download daemontools-0.76.tar.gz into /package. Unpack the package:

```
# wget http://cr.yp.to/daemontools/daemontools-0.76.tar.gz  
# gunzip daemontools-0.76.tar  
# tar -xpf daemontools-0.76.tar  
# rm -f daemontools-0.76.tar
```

# Installation & Config

---

```
# vim src/conf-cc
```

```
append -include /usr/include/errno.h to end of gcc command, like this gcc -O2 -Wimplicit -Wunused -Wcomment -Wchar-subscripts -Wuninitialized -Wshadow -Wcast-qual -Wcast-align -Wwrite-strings -include /usr/include/errno.h
```

set up the daemontools programs:

```
# package/install
```

```
Add csh -cf '/command/svscanboot &' to /etc/rc.local ( a soft link to /etc/rc.d/rc.local )
```

# Create service

---

```
# mkdir /root/spider
```

```
# vi /root/spider/run
```

Add below lines to run file

```
#!/bin/sh
```

```
exec service salt-minion start
```

```
# chmod 1755 /root/spider
```

```
# chmod 755 /root/spider/run
```

```
# ln -s /root/spider /service/spider
```

Once soft link is created (i.e. spider script is added to system service folder), the service is immediately started by daemontool

---

# Tesseract-Ocr

# Pillow

---

Pillow 是一个图像工具包，包含了一个 `Image` 类用来做图像的处理

```
pip install pillow
```

```
from PIL import Image
```

```
def extract_image(html):
```

```
    tree = lxml.html.fromstring(html)
```

```
    img_data = tree.cssselect('div#recaptcha img')[0].get('src')
```

```
    img_data = img_data.partition(',')[0]
```

```
    binary_img_data = img_data.decode('base64')
```

```
    img_data = BytesIO(binary_img_data)
```

```
    img = Image.open(img_data)
```

```
    img.save('test.png')
```

```
    return img
```

# Tesseract-Ocr

---

Tesseract-Ocr 是一个 Google 主导的开源 OCR (Optical Character Recognition) 引擎。Tesseract-Ocr 有很多的 python 开源版本  
pip install pytesseract

```
import pytesseract
```

```
pytesseract.image_to_string(bw)
```



# 识别过程

大量验证码都是添加了干扰元素的，因此第一步要找出噪声并去除掉

TeEW

<http://www.bjhjyd.gov.cn/>



北京市小客车指标调控管理信息系统

倡导绿色出行  
共建绿色北京

用户登录

个人用户 ☐ 非营运车 ☒ 营运车

手机号:  没有手机号的用编码代替

密 码:  忘记密码

验证码:  xheN

我要登录

# 找出验证码的色彩

对色彩像素进行统计

```
pixdata = img.load()
colors = {}
# 统计字符颜色像素情况
for y in range(img.size[1]):
    for x in range(img.size[0]):
        if colors.has_key(pixdata[x,y]):
            colors[pixdata[x, y]] += 1
        else:
            colors[pixdata[x,y]] = 1

# 排名第一的是背景色，第二的是主要颜色
colors = sorted(colors.items(), key=lambda d:d[1], reverse=True)
```

```
((240, 240, 240), 1996) - 排名第一的是背景色
((51, 153, 0), 645) - 排名第二的是验证码字体颜色
((241, 244, 237), 168),
((192, 168, 185), 37),
((161, 250, 53), 1)
```

# 去噪

---

把验证码色彩设置为黑色，其余颜色设置为白色



```
significant = colors[1][0]
for y in range(img.size[1]):
    for x in range(img.size[0]):
        if pixdata[x,y] != significant:
            pixdata[x,y] = (255,255,255)
        else:
            pixdata[x, y] = (0,0,0)
```

# 调用 TesseractOcr 进行识别

---

```
word = pytesseract.image_to_string(img, lang='eng', config='ocr.conf')
```

lang 指定识别的语言

config 指定配置文件，我们设置了有效字符仅包含A~Za~z0~9

tessedit\_char\_whitelist

abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ12  
34567890

一共 12 个验证码，识别正确了4个，正确率 33%

---

# 图片匹配

# 标准字体的图片

---

考虑下面这种类型的图片



用 Tesseract-Ocr 完全不能识别，干扰信息太多，而且干扰笔画的色彩与验证码一样

仔细观察，这些字体都比较标准，我们可以考虑用图片相似度匹配的方式来识别

# 标准字体的图片

---

考虑下面这种类型的图片



用 Tesseract-Ocr 完全不能识别，干扰信息太多，而且干扰笔画的色彩与验证码一样

仔细观察，这些字体都比较标准，我们可以考虑用图片相似度匹配的方式来识别

# 图片匹配

- 把所有的图片找出来，裁剪并拼接成如下的样子

0123456789abcdefghijklmnopqrstuvwxyz

- 把验证码图片中文字部分剪裁出来



- 把验证码图片转化为黑白，设定一个阈值200，小于200的处理为白色

1bda2

- 将参考字体与验证码每个字体比对，计算它们的距离，计算方式为每个像素的色彩差之和

$$distance = \sum_{i=0}^n p_i - l_i$$



# 图片匹配

- 把所有的图片找出来，裁剪并拼接成如下的样子

0123456789abcdefghijklmnopqrstuvwxyz

- 把验证码图片中文字部分剪裁出来



- 把验证码图片转化为黑白，设定一个阈值200，小于200的处理为白色

1bda2

- 将参考字体与验证码每个字体比对，计算它们的距离，计算方式为每个像素的色彩差之和

$$distance = \sum_{i=0}^n p_i - l_i$$

# 在线人工服务

---

将图片发送到注册的在线服务，由它们人工判别并返回

```
data = {  
    'action': 'usercaptchaupload',  
    'apikey': api_key,  
    'file-upload-01': img_data.encode('base64'),  
    'base64': '1',  
    'selfsolve': '1',  
    'maxtimeout': str(self.timeout)  
}  
encoded_data = urllib.urlencode(data)  
request = urllib2.Request(self.url, encoded_data)  
response = urllib2.urlopen(request)  
result = response.read()
```

# 疑问

---

□ 问题答疑：<http://www.xxwenda.com/>

■ 可邀请老师或者其他回答问题

# 联系我们

---

## 小象学院：互联网新技术在线教育领航者

- 微信公众号：大数据分析挖掘
- 新浪微博：ChinaHadoop

