数据提取与验证码的识别

小小图灵社

整理 by Orash

提取数据

在前面我们已经搞定了怎样获取页面的内容，不过还差一步，这么多杂乱的代码夹杂文字我们怎样把它提取出来整理呢？下面就开始介绍一个十分强大的工具，正则表达式！

正则表达式是对字符串操作的一种逻辑公式，就是用事先定义好的一些特定字符、及这些特定字符的组合，组成一个“规则字符串”，这个“规则字符串”用来表达对字符串的一种过滤逻辑。

正则表达式是用来匹配字符串非常强大的工具，在其他编程语言中同样有正则表达式的概念，Python 同样不例外，利用了正则表达式，我们想要从返回的页面内容提取出我们想要的内容就易如反掌了。

具体内容见附录。

正则表达式相关注解

数量词的贪婪模式与非贪婪模式

正则表达式通常用于在文本中查找匹配的字符串 Python 里数量词默认是贪婪的（在少数语言里也可能是默认非贪婪），总是尝试匹配尽可能多的字符；非贪婪的则相反，总是尝试匹配尽可能少的字符

例如：正则表达式"ab"如果用于查找"abbbc"，将找到"abbb"。而如果使用非贪婪的数量词"ab?"，将找到"a"。

常用方法

- re.match
  - re.match 尝试从字符串的起始位置匹配一个模式，如果不是起始位置匹配成功的话，match()就返回 none
  - 函数语法： re.match(pattern, string, flags=0)
- re.search

    - re.search 扫描整个字符串并返回第一个成功的匹配。
    - 函数语法： re.search(pattern, string, flags=0)
- re.sub
    - re.sub 替换字符串 re.sub(pattern,replace,string)
- re.findall
    - re.findall 查找全部 re.findall(pattern,string,flags=0)

正则表达式修饰符 - 可选标志

正则表达式可以包含一些可选标志修饰符来控制匹配的模式。修饰符被指定为一个可选的标志。多个标志可以通过按位 OR(|) 它们来指定。如 re.I | re.M 被设置成 I 和 M 标志:

修饰符	描述
re.I	使匹配对大小写不敏感
re.L	做本地化识别（locale-aware）匹配
re.M	
re.S	使 . 匹配包括换行在内的所有字符
re.U	根据Unicode字符集解析字符。这个标志影响 \w, \W, \b, \B
re.X	该标志通过给予你更灵活的格式以便你将正则表达式写得更易于理解

Beautiful Soup

Beautiful Soup 的简介

Beautiful Soup 提供一些简单的、python 式的函数用来处理导航、搜索、修改分析树等功能。它是一个工具箱，通过解析文档为用户提供需要抓取的数据，因为简单，所以不需要多少代码就可以写出一个完整的应用程序。

Beautiful Soup 自动将输入文档转换为 Unicode 编码，输出文档转换为 utf-8 编码。你不需要考虑编码方式，除非文档没有指定一个编码方式，这时，Beautiful Soup 就不能自动识别编码方式了。然后，你仅仅需要说明一下原始编码方式就可以了。

Beautiful Soup 已成为和 lxml、html6lib 一样出色的 python 解释器，为用户灵活地提供不同的解析策略或强劲的速度。

创建 Beautiful Soup 对象

```
from bs4 import BeautifulSoup

bs = BeautifulSoup(html,"lxml")
```

四大对象种类

Beautiful Soup 将复杂 HTML 文档转换成一个复杂的树形结构,每个节点都是 Python 对象,所有对象可以归纳为 4 种:

1. Tag
2. NavigableString
3. BeautifulSoup
4. Comment

Xpath

介绍

之前 BeautifulSoup 的用法，这个已经是非常强大的库了，不过还有一些比较流行的解析库，例如 lxml，使用的是 Xpath 语法，同样是效率比较高的解析方法。如果大家对 BeautifulSoup 使用不太习惯的话，可以尝试下 Xpath。

语法

具体请查看 github
https://github.com/littleturings/2021PythonWebCrawler/blob/main/Lecture_notes/%E6%95%B0%E6%8D%AE%E6%8F%90%E5%8F%96%E4%B8%8E%E9%AA%8C%E8%AF%81%E7%A0%81%E7%9A%84%E8%AF%86%E5%88%AB/09.%20%E6%95%B0%E6%8D%AE%E6%8F%90%E5%8F%96-XPath.md

模式	描述
$	匹配字符串的末尾
.	匹配任意字符，除了换行符，当re.DOTALL标记被指定时，则可以匹配包括换行符的任意字符
[...]	用来表示一组字符,单独列出：[amk] 匹配 'a'，'m'或'k'
[^...]	不在[]中的字符：[^abc] 匹配除了a,b,c之外的字符
re*	匹配0个或多个的表达式
^	匹配字符串的开头
re+	匹配1个或多个的表达式
re?	匹配0个或1个由前面的正则表达式定义的片段，非贪婪方式
re{ n}	
re{ n,}	精确匹配n个前面表达式
re{ n, m}	匹配 n 到 m 次由前面的正则表达式定义的片段，贪婪方式
a	b
(re)	G匹配括号内的表达式，也表示一个组
(?-imx)	正则表达式关闭 i, m, 或 x 可选标志。只影响括号中的区域
(?imx)	正则表达式包含三种可选标志：i, m, 或 x 。只影响括号中的区域
(?: re)	类似 (...), 但是不表示一个组
(?imx: re)	在括号中使用i, m, 或 x 可选标志
(?-imx: re)	在括号中不使用i, m, 或 x 可选标志
(?#...)	注释
(?= re)	前向肯定界定符。如果所含正则表达式，以 ... 表示，在当前位置成功匹配时成功，否则失败。但一旦所含表达式已经尝试，匹配引擎根本没有提高；模式的剩余部分还要尝试界定符的右边。
(?! re)	前向否定界定符。与肯定界定符相反；当所含表达式不能在字符串当前位置匹配时成功
(?> re)	匹配的独立模式，省去回溯
\w	匹配字母数字及下划线
\W	匹配非字母数字及下划线
\s	匹配任意空白字符，等价于 [\t\n\r\f].
\S	匹配任意非空字符
\d	匹配任意数字，等价于 [0-9]
\D	匹配任意非数字
\A	匹配字符串开始
\Z	匹配字符串结束，如果是存在换行，只匹配到换行前的结束字符串。c
\z	匹配字符串结束
\G	匹配最后匹配完成的位置
\b	匹配一个单词边界，也就是指单词和空格间的位置。例如， 'er\b' 可以匹配"never" 中的 'er'，但不能匹配 "verb" 中的 'er'
\B	匹配非单词边界。'er\B' 能匹配 "verb" 中的 'er'，但不能匹配 "never" 中的 'er'
\n, \t, 等.	匹配一个换行符。匹配一个制表符。等
\1...\9	匹配第n个分组的内容
\10	匹配第n个分组的内容，如果它经匹配。否则指的是八进制字符码的表达式
[\u4e00-\u9fa5]	中文

JsonPath

详情见 github

多线程爬虫