20行与200行代码批量获得网站标题-反思平庸与技术精进

作者: LANDGREY ● 创建时间 2018年6月8日 22:43 ● 更新时间 2018年6月9日 01:36
浏览: 873 次. ● 标签: #分享, #思考
您的IP地址: 140.207.23.83

一. 20行代码

临时任务，需要自动化收集大概文本中指定的100多个域名的网站标题(title)。这种以前也做过的简单任务，自己操作起来可以说是驾轻就熟了。

花个两分钟，写个脚本，大概20行代码如下：

```
#!/usr/bin/env python
# coding:utf-8
#
import re
import requests

headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) Chrome/52.0.2743.11


def poc(url):
    try:
        req = requests.get(url, headers=headers, timeout=15)
        title = re.findall("<title>(.*?)</title>", req.text, re.I | re.M)
        if title:
            return url + " " + title[0]
        else:
            return False
    except:
        return False
```

使用POC-T 框架调用以上代码，就可以批量获取网站title了。使用Python脚本+框架的方式，可以简单快速的完成任务。但是实际上，操作起来最多只能获取到大概60%左右的网站标题。剩下的怎么办？还是手工一个个打开网页看标题……

二. 问题所在

批量获得网站Title这个编写脚本的技术工作，自己以前在渗透测试中，也做过很多次。要是有人问我"怎么批量获取网站的标题"，我心里可能会嘲笑问我的人"这么简单的事都不会"。

但是到了真正我来完成批量获取网站标题的任务时，祭出的20行代码却只能获取到60%左右的网站标题，剩下的大量不能正常获取的网站标题，还得人工一个一个的去看。浪费大量的精力去做本来可以很好的利用自动化方式解决的事情，让我感到十分羞愧。

20行代码存在的主要问题，其实自己也大概清楚是怎么回事：

1. 编码处理不当。因为各种环节问题，一些中文标题解码错误，不能正确获得；
2. 不少网站存在首页跳转，程序获取不到跳转地址，自然拿不到标题；
3. 各种其它细节处理与疏于考虑的问题

但自己在批量获取网站标题这个工作上，平庸到可以说是"怎么批量获取网站标题"都不会的人。

最终应该嘲笑的是自己长久以来对平庸技术的满足与追求更加精进技术需求的漠视。

三. 200行代码

知道了问题所在，我就下决心要解决长期以来的对此问题的漠视，做个会"批量获取网站标题"的人。

果然，写出准确率很高的批量获取网站Title的代码，要考虑并处理很多种情况，包括：

1. 编码问题。中文、UTF-8及Unicoded等转换。
2. 全面考虑各种首页跳转的情况。
3. 网站跳转代码的正则匹配表达式的书写考虑要全面。
4. HTTPS协议网站请求相关问题。
5. 匹配网站标题的正则表达式的正确书写。
6. Js代码写网站标题的情况。
7. HTML实体编码的网站标题。
8. 中间包含换行、Tab键等空白字符标题的处理。
9. 网站请求错误重试、延时、网页自动跳转、超时时间、headers等考虑。
10. 获取不到网站Title的原因要显示出来
11. 其它

当然，目前代码可能还是有不完善或者考虑不到的地方，一些较少遇到的情况会人为忽略。比如有的网站首页会以JavaScript 结合Form进行自动跳转

```
<form action="/other_home/" id="form"></form>
<script> document.getElementById("form").submit(); </script>
```

或者Frameset跳转

```
<frameset cols="100%">
        <frame id="frame1" src="/other_home/index.jsp"></frame>
</frameset>
```

最终脚本owt.py，将近200行代码如下：

```
#!/usr/bin/env python
# coding:utf-8
# Build By LandGrey
#
import re
import os
import ssl
import sys
import socket
import requests
import argparse
import HTMLParser
from requests.adapters import HTTPAdapter
from multiprocessing.dummy import Pool as ThreadPool


try:
    requests.packages.urllib3.disable_warnings()
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    pass
else:
    ssl._create_default_https_context = _create_unverified_https_context


def out_format(url, information):
    for char in ('\r', '\n', '\t'):
        information = information.replace(char, "")
    try:
        message = u"{target:50} {information}".format(target=url, information=in
    except:
        try:
            message = "{target:50} {information}".format(target=url, information=
        except:
            message = "{target:50} {information}".format(target=url, information=
    try:
        print(message)
    except UnicodeError:
        print("{target:50} {information}".format(target=url, information="PrintU


def html_decoder(html_entries):
    try:
        hp = HTMLParser.HTMLParser()
        return hp.unescape(html_entries)
    except Exception as e:
        return html_entries


def match_title(content):
    title = re.findall("document\.title[\s]*=[\s]*['\"](.*?)['\"]", content, re.
```

```
    if title and len(title) >= 1:
        return title[0]
    else:
        title = re.findall("<title.*?>(.*?)</title>", content, re.I | re.M | re.S
        if title and len(title) >= 1:
            return title[0]
        else:
            return False


def page_decode(url, html_content):
    raw_content = html_content
    try:
        html_content = raw_content.decode("utf-8")
    except UnicodeError:
        try:
            html_content = raw_content.decode("gbk")
        except UnicodeError:
            try:
                html_content = raw_content.decode("gb2312")
            except UnicodeError:
                try:
                    html_content = raw_content.decode("big5")
                except:
                    return out_format(url, "DecodeHtmlError")
    return html_content


def get_title(url):
    origin = url
    if "://" not in url:
        url = "http://" + url.strip()
    url = url.rstrip("/") + "/"
    # First Try Obtain WebSite Title
    try:
        s = requests.Session()
        s.mount('http://', HTTPAdapter(max_retries=1))
        s.mount('https://', HTTPAdapter(max_retries=1))
        req = s.get(url, headers=headers, verify=False, allow_redirects=True, tim
        html_content = req.content
        req.close()
    except requests.ConnectionError:
        return out_format(origin, "ConnectError")
    except requests.Timeout:
        return out_format(origin, "RequestTimeout")
    except socket.timeout:
        return out_format(origin, "SocketTimeout")
    except requests.RequestException:
        return out_format(origin, "RequestException")
    except Exception as e:
        return out_format(origin, "OtherException")
    html_content = page_decode(url, html_content)
```

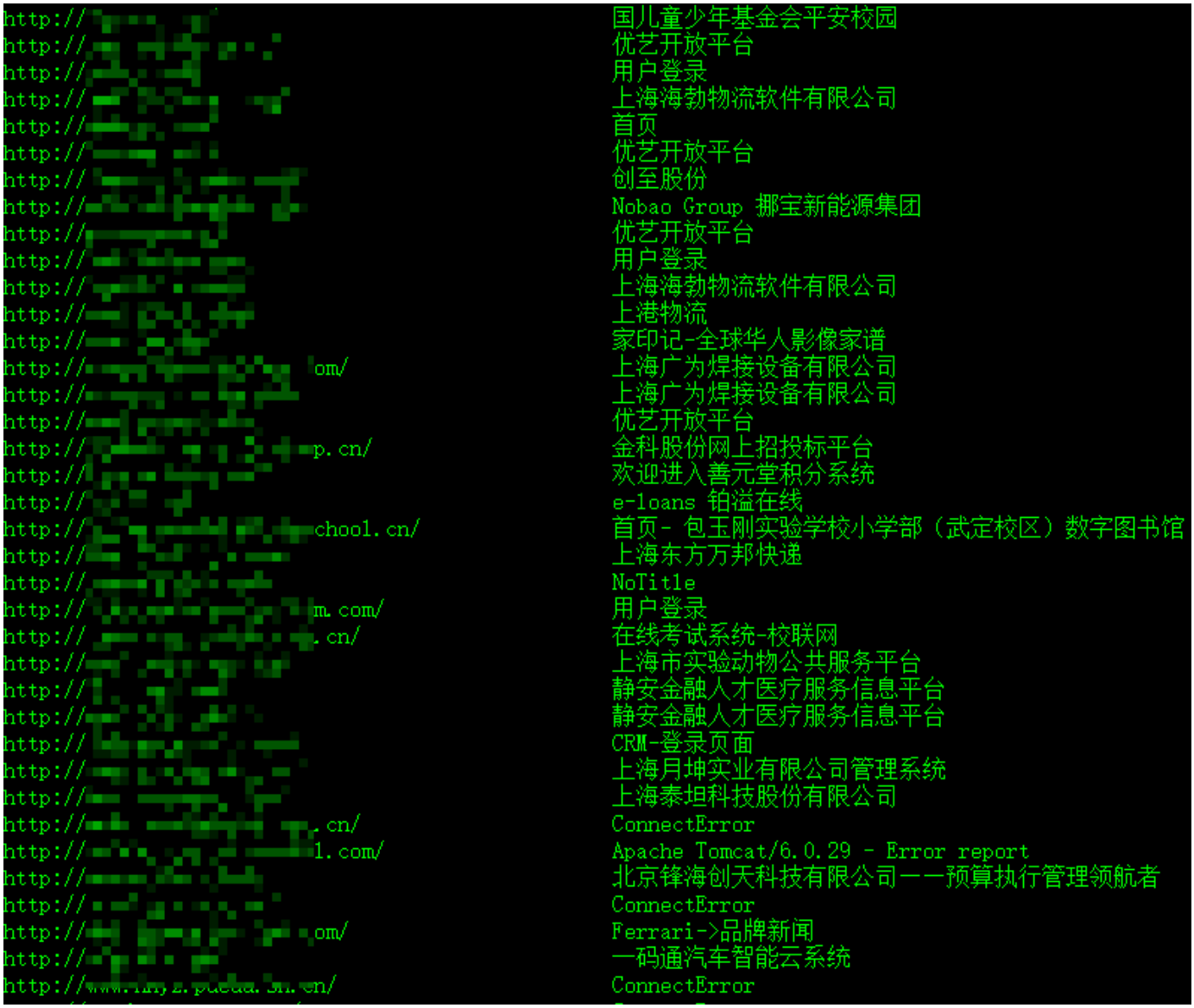
```
    if html_content:
        title = match_title(html_content)
    else:
        exit(0)
    try:
        if title:
            if re.findall("\$#\d{3,};", title):
                title = html_decoder(title)
            return out_format(origin, title)
    except Exception as e:
        return out_format(origin, "FirstTitleError")
    # Find Jump URL
    for pattern in patterns:
        jump = re.findall(pattern, html_content, re.I | re.M)
        if len(jump) == 1:
            if "://" in jump[0]:
                url = jump[0]
            else:
                url += jump[0]
            break
    # Second Try Obtain WebSite Title
    try:
        s = requests.Session()
        s.mount('http://', HTTPAdapter(max_retries=1))
        s.mount('https://', HTTPAdapter(max_retries=1))
        req = s.get(url, headers=headers, verify=False, timeout=15)
        html_content = req.content
        req.close()
    except requests.ConnectionError:
        return out_format(origin, "ConnectError")
    except requests.Timeout:
        return out_format(origin, "RequestTimeout")
    except socket.timeout:
        return out_format(origin, "SocketTimeout")
    except requests.RequestException:
        return out_format(origin, "RequestException")
    except Exception as e:
        return out_format(origin, "OtherException")
    html_content = page_decode(url, html_content)
    if html_content:
        title = match_title(html_content)
    else:
        exit(0)
    try:
        if title:
            if re.findall("[$#]\d{3,};", title):
                title = html_decoder(title)
            return out_format(origin, title)
        else:
            return out_format(origin, "NoTitle")
    except Exception as e:
        return out_format(origin, "SecondTitleError")
```

```
if __name__ == "__main__":
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) Chrome/52.0.2743
        "Accept-Language": "zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3",
        "Accept-Encoding": "gzip, deflate",
        "Connection": "close",
    }

    patterns = (
        '<meta[\s]*http-equiv[\s]*=[\s]*[\'"]refresh[\'"][\s]*content[\s]*=[\s]*
        'window.location[\s]*=[\s]*[\'"](.*?)[\'"][\s]*;',
        'window.location.href[\s]*=[\s]*[\'"](.*?)[\'"][\s]*;',
        'window.location.replace[\s]*\([\'"](.*?)[\'"]\)[\s]*;',
        'window.navigate[\s]*\([\'"](.*?)[\'"]\)',
        'location.href[\s]*=[\s]*[\'"](.*?)[\'"]',
    )

    urls = []
    results = []
    parser = argparse.ArgumentParser(prog='owt.py', description="Obtain WebSite T
    parser.add_argument("-t", dest='target', default='urls.txt', help="target wit
    parser.add_argument("-x", dest='threads', default=4, type=int, help="number o
    if len(sys.argv) == 1:
        sys.argv.append('-h')
    args = parser.parse_args()
    target = args.target
    threads = args.threads
    if os.path.isfile(target):
        with open(target, 'r') as f:
            for line in f.readlines():
                urls.append(line.strip())
        try:
            pool = ThreadPool(threads)
            pool.map(get_title, urls)
            pool.close()
            pool.join()
        except KeyboardInterrupt:
            exit("[*] User abort")
    else:
        if "://" not in target:
            target = "http://" + target
        get_title(target)
```

测试效果，有标题的网站基本都能够正确获得：



四. 心得

在前人的基础上，对技术不断的总结和精益求精，才能获得个人真正长足的进步。

行百里者半九十，做出60%的成绩可能需要1份努力就够了，而追求90%以上的成绩则可能需要10倍的努力才能获得，而不是看起来仅仅只差30%，差半分努力而已。

平庸和精进技术的差别，相差更多的，是对技术精益求精的不懈追求，而不是泛泛而谈，浅尝辄止。

<