

以豆瓣为例的爬虫说明

在昨天的作业中，有很多人问了我同样的问题，目录，书籍，评论不在同一个页面啊，怎么进行爬取，今天我以豆瓣为例，教大家如何维护一个爬虫的url队列，对不同结构的网页分别解析并获取内容

我们必须知道，同样类型的网页结构是相同的，前端工程师，一般在同类型的页面会使用同一个模板，因此我们对于同一个模板的网页可以使用同一个解析函数,我们将以豆瓣互联网标签下的高排名书籍举例来解决同时爬取不同结构网页

```
import matplotlib.pyplot as plt
import requests
from bs4 import BeautifulSoup
%matplotlib inline
```

```
#建立一个url列表，我们将从其中取url，并根据不同的关键词进行解析
urls_list = []
```

分析网页结构

我们首先来看看我们需要爬取的网页，分别是目录页，书籍信息页，书籍评论页



科技想要什么

[美] 凯文·凯利 / 熊祥 / 中信出版社 / 2011-11 / 58.00元

★★★★★ 8.2 (1737人评价)

在《科技想要什么》一书中，凯文·凯利向我们介绍了一种全新的科技观。他认为，作为整体，科技不是由线路和金属构成的一团乱麻，而是有生命力的自然形成的系统，它的起...



硅谷之谜：《浪潮之巅》续集

吴军 / 人民邮电出版社 / 2015-12-1 / 59.00

★★★★★ 8.5 (912人评价)

这是一本颠覆人们对信息时代的认识、对创新和创业的理解的好书。作者吴军通过介绍硅谷成功的秘诀，揭示了信息时代的特点和方法论。近年来，吴军从技术和管理人员变成...

[在豆瓣购买](#)






失控



作者: [美] 凯文·凯利
出版社: 新星出版社
副标题: 全人类的最终命运和结局
原作名: Out of Control: The New Biology of Machines, Social Systems, and the Economic World
译者: 东西文库
出版年: 2010-12
页数: 707
定价: 88.00元
装帧: 平装
丛书: 东西文库
ISBN: 9787513300711

豆瓣评分

8.7  11876人评价

5星  54.7%
4星  32.5%
3星  10.7%
2星  1.4%
1星  0.8%

想读 在读 读过 评价: ☆☆☆☆☆

[写笔记](#) [写书评](#) [加入购书单](#) [分享到](#)

推荐

失控的书评 (354)

最受欢迎 / 最新发布的 / 只看本版本的评论 / 按评审查看

《失控》中比较严重的误译

 书海回头客 ★★★★★ 2011-05-17 01:59:15

如此优秀的一本书，还是被“东西文库”译得错误百出，弊病非常可惜。今天终于痛下决心，准备利用重读的机会，用英文版电了书对照，陆续挑出这本书中我认为比较严重的翻译错误（和错别字），供译者和各位高手批评指正！申明一下，我下这清单功夫，没有别的总想，只是不认同这种... (337回应)

1326有用 / 27没用

人类啊，放手让机器去搞吧！

 Vince ★★★★★ 2012-09-13 00:47:16

书名《失控》，英文《Out of Control: The New Biology of Machines, Social Systems, and the Economic World》，听起来挺唬人，再加上黑黄色的封面，让人以为又是一本关于90年代的末日论著作。实际上这是一本充满浪漫主义情怀的书。它讲述了随着技术进步放弃对机器的精确控制... (211回应)

210有用 / 5没用

与 Kevin Kelly 对话

目录页

在目录页我们需要获取书籍详细信息的url那么我们可以写出这样一个函数来获取

```

headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/59.0.3071.115 Safari/537.36"}
def index(url):
    """
    此处爬取目录页，返回一个列表，列表中元素包含关键字和url
    """
    try:
        res = requests.get(url, headers=headers)
        soup = BeautifulSoup(res.content.decode("utf-8"), "lxml")
        info_list = soup.find_all("li", {"class": "subject-item"})
        urls = [{"details", info.find("div", {"class": "info"}).h2.a["href"]} for info in
info_list]
        next_url = soup.find("span", {"class": "next"})
        urls.append(("index", "https://book.douban.com"+next_url.a["href"]))
    except Exception:
        return 0, None
    return 1, urls

```

#我么可以在此处查看结果，他会返回不同的值

```
result, urls = index("https://book.douban.com/tag/%E4%BA%92%E8%81%94%E7%BD%91?start=0&type=T")
```

```

from pprint import pprint
#查看结果
pprint(result)
#查看url列表
pprint(urls)

```

```

1
[('details', 'https://book.douban.com/subject/5375620/'),
 ('details', 'https://book.douban.com/subject/6709783/'),
 ('details', 'https://book.douban.com/subject/26838557/'),
 ('details', 'https://book.douban.com/subject/6021440/'),
 ('details', 'https://book.douban.com/subject/26306686/'),
 ('details', 'https://book.douban.com/subject/26658379/'),
 ('details', 'https://book.douban.com/subject/26873486/'),
 ('details', 'https://book.douban.com/subject/26297606/'),
 ('details', 'https://book.douban.com/subject/26582822/'),
 ('details', 'https://book.douban.com/subject/26929955/'),
 ('details', 'https://book.douban.com/subject/5914587/'),
 ('details', 'https://book.douban.com/subject/10828002/'),
 ('details', 'https://book.douban.com/subject/25752043/'),
 ('details', 'https://book.douban.com/subject/20429677/'),
 ('details', 'https://book.douban.com/subject/10945606/'),
 ('details', 'https://book.douban.com/subject/1919072/'),
 ('details', 'https://book.douban.com/subject/25808226/'),
 ('details', 'https://book.douban.com/subject/26541801/'),
 ('details', 'https://book.douban.com/subject/6965746/'),
 ('details', 'https://book.douban.com/subject/26665230/'),
 ('index', 'https://book.douban.com/tag/互联网?start=20&type=T')]

```

可以发现列表中包含了不同关键字的网址

图书信息页

图书详情页这一页我们不单单要获取内容存储，还要获取url加入列表中，都把它返回就行了

```
target_url = "https://book.douban.com/subject/5375620/"
infos = {}
def details(url):
    headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36"}
    res = requests.get(url, headers = headers)
    soup = BeautifulSoup(res.content.decode("utf-8"), "lxml")
    infos = {
        "title": soup.find("span", {"property": "v:itemreviewed"}).get_text(),
        "rate": soup.find("strong", {"class": "rating_num"}).get_text(),
        "intro": soup.find_all("div", {"class": "intro"})[0].get_text(),
        "author_intro": soup.find_all("div", {"class": "intro"})[1].get_text()
    }
    next_url = ("short_evaluate", soup.find("div", {"class": "mod-hd"}).find("span", {"class": "pl"}).a["href"])
    return 1, infos, next_url
```

```
result, infos, url = details(target_url)
```

短评页

```
def write_evaluate(file, evaluate_list):
    with open(file, "w", encoding="utf-8") as f:
        f.write("\r\n".join(evaluate_list))
```

```
def short_evaluate(url):
    try:
        headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36"}
        res = requests.get(url, headers = headers)
        soup = BeautifulSoup(res.content.decode("utf-8"), "lxml")
        short_evaluate = [short_eva.find("p", {"class": "comment-content"}).get_text() for short_eva in soup.find("div", {"id": "comments"}).find_all("div", {"class": "comment"})]
        return 1, short_evaluate
    except Exception:
        print("短评"+"在抓取"+url+"时错误")
        return 0, None
```

```
result, short_eva = short_evaluate("https://book.douban.com/subject/5375620/comments/")
```

```
short_eva
```

['反正我在硅谷的时候都没听说KK是谁，倒是回国后发现大家都读过失控。神神叨叨的，随便翻开一章时常会觉得哇好精彩；不过神神叨叨的，盖上册的最后一页后我也没法复述这书到底说了什么...'，

'为什么我完全读不下去，只有我一个人有这样的感觉吗？'，

'整整七百页更象是一个科学边缘的异端分子随手写的生活小杂文，kk四处引经据典无非就是想说明“科技现在也被我们变成一个复杂的生命体啦”。如此混乱的结构竟还有恁多人热烈捧场，这是借群体行为艺术来阐释书名深意么'，

'一本被及其过誉了的书，我决定不受舆论干扰，**follow my heart**的给个差评。\\n背后的团队很会炒作，成功的进行了教科书般的营销。\\n这是一本成功的畅销书，但若定位成无论其他任何类型，科普也好、哲学也好、互联网也好、社会科学也好，都是失败的。\\n全书看似有逻辑，实际上很多都是在凑字数，充满了冗长的自我重复；大段的引用他人观点，整体思想上并没有一个富有说服力的逻辑脉络作为支撑；罗列看似高深的术语和名词，貌似理论著作，其实在骗外行，当然作者自己也不是专家；喜欢举一些个案的例子，尤其是那些游离在主流科学以外的人，以迎合读者的口味。\\n'，

'三星是我的主观偏见了，客观来论应该能上四。但我只是想说：1 不要为了证明自己的观点而去学习；2 不要写自己不懂的事情。'，

'所有权而不是使用权，屏幕，去中心化，人工智能，未来二十年！凯文凯利。它的言论关键不在他的基本框架和原理而在于具体的实现和焦点。过去读失控那本书得到的错误结论原因在此。计算机智能从机械和计算角度思考完全不可能超越人类智能，但是从生物进化和热力学突变角度思考那么超越人工智能完全是指日可待的事情。'，

'这人到底怎么火的？神马玩意啊这是。'，

'什么破玩意 读了前几页实在受不了 '，

'技术，联姻人造（机械化）与天生（生物化），并非所有重要的事情都能事先计划好。面对大型任务，我们需要领会到如何通过“去中心化”并借助“最少的规则”来完成。无为而治，自然而然，要成长为新的物种，就要经历所有你不会再扮演的角色。'，

'这个翻译对于这本书真是让人不忍直视，单看某个词，感觉大概差不多，连成句一看要么味同爵蜡索然无味，要么不知所云，翻译的极为生硬，文绉绉的，毫无美感。**The natural flux**翻译成“自然的流变”。这个“流变”是个毛啊.....'，

'终于读完了。这本书被捧过高了。写得东一榔头西一棒子，而且每个主题写得都不深。在20年前可能有高瞻远瞩的时代意义，但是现在读没太大意思。'，

'把他能想起来的都拖出来扯了一遍。失控的是作者的思考能力吧→_→ '，

'一篇文章就可以说明白的思想内容，不必写成一本书：，最终用户群有群体效应，会产生自己的一种方向。'，

'题目将**out of control**翻译为失控，其实讲的是如何将控制变得更加智能，更加具有适应性、鲁棒性和可进化性。700多页的篇幅看起来非常杂乱，其实复杂理论、涌现、智能算法等等反复出现，真的没必要认真读完的一本书。'，

'如果可以打6星，我会毫不犹豫！目前正在啃，握笔的手忍不住的抖动！'，

'若干年之后读依然不过时，核心理念并不复杂。'，

'作者是在94年就能预见到人际网络时代和云架构的牛人，这本书写得非常平实易懂。'，

'思考的人，不用看；不过脑的，看了等于白看。这本近700页书对我而言最大的意义是终于可以开始看那本文艺的书了'，

'作者很不错，一个基督徒的科学观，也是一个对自由经济致敬的人。'，

'严格说我读了这本书的1/6左右。重点读了第3章和第11章。KK关于“我们如何彼此连接”影响了“**what we make, how we make it, how we decide what to make**”的论述相当精彩，很多小段子，完全猜不到是90年代中的出版物。但正如读书会大家讨论的那样，KK的专栏作家味道实在太浓，写关于科学的东西却没有用科学的方法去写。至于细部的槽怎么吐，请询豆瓣ID:pkuent.....']

组装

我们将上面写的三个部分组装在一起实现一个简单的不同类型网页的抓取

```

infoslist = []
eva_list = []
def fetch():
    key,url = urls_list.pop()
    print("开始抓取网页",url,"类型为",key)
    if(key=="index"):
        result,urls = index(url)
        if result == 1:
            print("抓取目录"+url+"成功")
            urls_list.extend(urls)
        else:
            print("抓取目录"+url+"发生了错误")
    elif(key=="details"):
        result,infos,urls = details(url)
        if result == 1:
            infoslist.append(infos)
            urls_list.append(url)
            print("抓取书籍详情",url,"成功")
        else:
            print("抓取书籍详情时出错，目标网页为",url)
    else:
        result,short_evas = short_evaluate(url)
        if result == 1:
            eva_list.extend(short_evas)
            print("抓取评论",url,"成功")
        else:
            print("抓取书籍评论时出错，目标网页为",url)

```

```

urls_list.append(("index","https://book.douban.com/tag/互联网"))

```

```

import time
import random
while urls_list:
    time.sleep(random.randint(2,3))
    fetch()

```

开始抓取网页 https://book.douban.com/tag/互联网 类型为 index

抓取目录https://book.douban.com/tag/互联网成功

开始抓取网页 https://book.douban.com/tag/互联网?start=20&type=T 类型为 index

抓取目录https://book.douban.com/tag/互联网?start=20&type=T成功

具体如何储存则由你自己

继续改进你的代码

上面的所有代码完成了抓取和解析的操作，可以将解析和抓取分开，实现两个类Fetcher，和Parser，之所以分开是因为抓取这一步骤可以使用多线程加快进度，同时对于抓取较多页面时，必须使用代理，分开有助于维护。

上面代码并没有存储的操作，要知道存储操作的变化性挺大，在爬取大型网站时需要使用数据库来存储，我们可以将整个过程通过实现saver这个类来实现。

这样一个爬虫架构就出现了，由fetcher，parser，saver三个类，他们之间使用三个队列来交互，爬取队列，解析队列，存储队列。

一点小小的建议:可以实现类似flask route的装饰器来增加框架的易用性

如上面的if就可以写成类似下列的形式

```
@parser.key("index")
def index():
    .....
```

如何实现类似的装饰器，请查看该教程[flask route这不是魔法](#)