数据挖掘大作业目录

1 总体描述

1.1 开发目的

当前正处在大数据时代，一切以数据说话，而现在每天产生的数据量特别大，那么如何从大量的数据中获得当前所需要的数据？这也是我在研究生阶段比较感兴趣的方向。学习大数据需要解决以下几个问题：1、学习资料来源；2、缺少训练项目。我一直都认为以解决实际需求为目的的学习能让我更快的学习，所以我萌发了这样一个想法，为什么不爬取一些学习资料来供自己学习呢？而且这个也可以训练自己的编写数据挖掘代码的能力。

1.2 系统目标

1. 爬取云栖社区中关于大数据相关的文章；
2. 对爬去的内容进行清洗，去掉无关的内容；
3. 删除带有“window”和“线下”等不符合的学习目的的文章；
4. 删除重复的文章；
5. 将文章信息导出到文件，方便自己阅读使用。

2 开发环境

2.1 开发技术综述

数据挖掘所用的开发环境一般为 linux，对于 linux 新手可以使用 Ubuntu 操作系统。linux 操作系统对开发人员更加友好，而不是 windows 系列操作系统更多的在用户体验上对普通用户友好。linux 操作系统给开发人员提供大量的编程和运行环境，python 环境也不例外。

1 python

python 是一种面向对象的解释型计算机程序设计语言。它包含了一组功能完备的标准库，可以轻松完成很多常见的任务。它的语法特别简单，具有自己特点

即用缩进来定义语句块。python 拥有一个强大的标准库。python 语言的核心只包含数字、字符串、列表、字典、文件等常见类型和函数，而由 python 标准库提供了系统管理、网络通信、文本处理、数据库接口、图形系统、xml 处理等额外的功能。同时 python 社区提供了大量的第三方模块，使用方式与标准库类似。它们的功能覆盖科学计算、web 开发、数据库接口、图形系统多个领域。python 常被用作其他语言与工具之间的“胶水”语言。

2 scrapy

scrapy 是一个为了爬取网站数据，提取结构性数据而编写的应用框架。可以应用在包括数据挖掘，信息处理或存储历史数据等一系列的程序中。scrapy 提供了很多强大的特性来使得爬取更为简单高效，如：对 html、xml 源数据选择及提取的内置支持，提供了 css 选择器以及 xpath 表达式进行处理以及一些帮助函数来使用正则表达式来提取数据；提供交互式 shell 终端，可以方便测试 css 及 xpath 表达式，更方便编写和调试爬虫；通过 feed 导出多格式、多存储后端的内置支持，提供了一系列在 spider 之间共享的可复用的过滤器，对智能处理爬取数据提供了内置支持等等。

2.2 软硬件环境

由于实验机中安装的为 windows 系统，因此使用 VirtualBox 虚拟机安装了 ubuntu 操作系统。具体软硬件环境如表 1 所示。

表 1 软硬件环境定义

名称	详细要求
硬件要求	内存 4g，处理器 4 核*1.9ghz，显存 128mb
运行系统	ubuntu-16.04-64bit
运行环境	python-2.7.12
开发环境	pycharm-2016.2.3
开发框架	scrapy-0.25
辅助工具	pip-9.0.1

2.3 环境安装

1. 安装 pycharm

进入 pycharm 官网下载最新版的 pycharm，当前最新版为 2016.3，官网地址：https://www.jetbrains.com/pycharm/ ，pycharm 在 linux 中安装特别简单，只需要解压，进入 bin 目录下，执行 `sh ./pycharm.sh` 即可。

2. 安装 pip

首先检查是否安装 pip

```
$ pip --version
```

升级 pip

```
$ pip install --upgrade pip
```

3. 安装 lxml

安装 lxml

```
$ sudo apt-get install libxml2-dev libxslt-dev python-dev
```

或者使用 pip 安装

```
$ sudo pip install lxml
```

安装 python-support，下载 python-support_1.0.15_all.deb 并安装

3. 安装 scrapy

安装 scrapy

```
$ pip install scrapy
```

Ubuntu 中提供了 python-scrapy，相较于最新版的 scrapy，该包版本太旧，运行速度较为缓慢。可以使用官方提供的 ubuntu packages。安装方法如下：

1. 把 Scrapy 签名的 GPG 密钥添加到 APT 的钥匙环中:

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 627220E7
```

2. 执行如下命令，创建 `/etc/apt/sources.list.d/scrapy.list` 文件:

```
$ echo 'deb http://archive.scrapy.org/ubuntu scrapy main' | sudo tee /etc/apt/sources.list.d/scrapy.list
```

3. 更新包列表并安装 scrapy-0.25:

```
$ sudo apt-get update && sudo apt-get install scrapy-0.25
```

4. 使用 git

git 是优秀的代码版本控制工具，而 github 是全球最大存放代码的公共仓库，为了方便沟通交流，我也将代码托管在 github 上。

代码地址：https://github.com/htzy/bigdata_data

3 代码结构

首先打开终端，选择工作目录，新建文件夹 bigdata_data

```
$ mkdir bigdata_data
```

进入 bigdata_data

```
$ cd bigdata_data
```

新建项目 bigdata

```
$ scrapy startproject bigdata
```

得到如下内容的 bigdata 目录：

```
bigdata/
    scrapy.cfg

    bigdata/
        __init__.py

        items.py

        pipelines.py

        settings.py

        spiders/
            __init__.py
```

打开 pycharm，打开项目 bigdata。此时基本的代码框架已经形成。

其中 items.py 的作用是保存爬取到的数据的容器，即定义爬取数据时，定

义数据内容具体组成，如下代码指的是数据项由名字、链接和描述组成。

```
import scrapy

class DmozItem(scrapy.Item):
    title = scrapy.Field()
    link = scrapy.Field()
    desc = scrapy.Field()
```

创建一个爬虫，必须继承 scrapy.Spider 类，并定义必要的属性：name、start_urls 和 parse()方法。其中 name 用来给爬虫命名，一般与爬取的网站名一致；start_urls 包含了爬虫在启动时进行爬取的 url 列表，即首先从 url 列表中开始爬取，当爬取到未爬取过的且符合要求的 url 时，添加到待爬取 url 列表中等待下一步爬取；parse()方法是 spider 的一个方法，每次 url 完成下载后生成的 Response 对象会作为参数传递给该函数，在方法里进行解析返回的数据，提取数据即生成 item 以及进一步处理 Request 对象。

爬取命令：

进入项目的根目录，执行下列命令启动爬虫

```
$ scrapy crawl bigdata
```

为了实现系统目标以达到开发目的，在项目基础上新建若干文件，修改后项目结构如图 1。

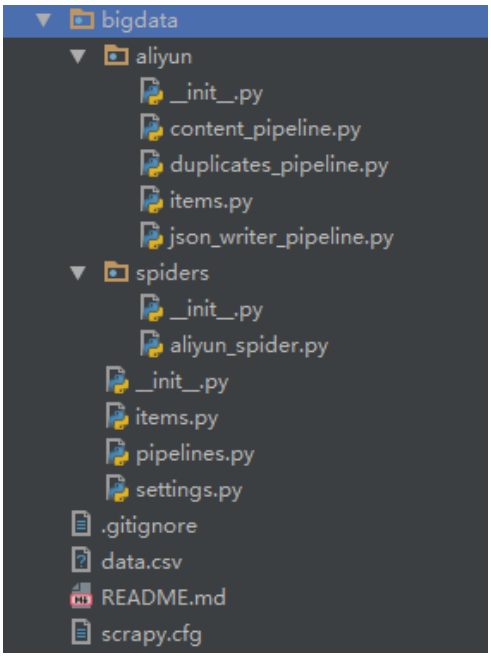

图 1 项目结构图

因为爬取的网站为阿里云栖社区，因此定义一个包：aliyun，将爬取内容相关的代码都存放在该包中，其中包括：内容清洗的 content_pipeline.py，去重的 duplicates_pipeline.py，将爬取的数据导出到 json 文件进行保存方便进一步使用的 json_writer_pipeline.py，数据项 items.py。

在 spiders 爬虫包中定义阿里云爬虫 aliyun_spider.py。将如上结构设置写入到 settings.py 中。这里将忽略默认生成的 items.py 和 pipelines.py。

4 核心功能

4.1 爬取数据

爬取的地址为：https://yq.aliyun.com/，云栖社区为阿里云官方开发者社区，里面的资料较为优质且更新速度快，种类也丰富，其中包括：文本、pdf、图片、视频等资料种类，但是所有需要下载的资源均需要登陆，而模拟登陆阿里云难度较大，暂时将该问题延期处理。这里将重点抓取文本类资源，即文章资料。云栖社区中几乎涵盖当前流行的所有技术，因为当前只需要学习大数据相关的知识，因此利用云栖社区中的搜索功能，并从该页面开始爬取。

搜索地址格式如下：（{keyword}为搜索的关键字，{pageNum}为搜索的页码）
https://yq.aliyun.com/search/articles/?q={keyword}&idex=default&days=&p={pageNum}

我当前想学习的大数据相关的关键词为：docker，maven，scala、spark、hadoop，初始页码均为 1。因此有如下对 start_urls 的定义。

```
start_urls = [
"https://yq.aliyun.com/search/articles/?q=docker&idex=default&days=&p=1
",
"https://yq.aliyun.com/search/articles/?q=maven&idex=default&days=&p=1"
,
"https://yq.aliyun.com/search/articles/?q=scala&idex=default&days=&p=1"
,
"https://yq.aliyun.com/search/articles/?q=hadoop&idex=default&days=&p=1
",
"https://yq.aliyun.com/search/articles/?q=spark&idex=default&days=&p=1"
]
```

定义爬取的数据项，即 items.py，爬取内容包括：标题、摘要、作者、作者链接、标签、阅读数、点赞数、评论数、时间、链接。在页面中表现如图 2 所示。

图 2 数据项

数据项代码定义如下：

```
from scrapy.item import Item, Field
class SearchItem(Item):
    title = Field()
    abstract = Field()
    author = Field()
    author_link = Field()
    tags = Field()
    viewers = Field()
    like = Field()
    comments = Field()
    time = Field()
    link = Field()
```

当爬取完当前页面中的数据项后爬取下一页的内容。首先检测是否存在下一页，如果存在则生成下一页的链接，并开始抓取下一页的内容，具体代码实现如下：

```
new_url_node = response.xpath('//a[@rel="next"]/@href').extract()
if new_url_node:
    new_url = "https://yq.aliyun.com" + new_url_node[0]
    print new_url
    yield Request(new_url, callback=self.parse)
```

图 3 下一页

4.2 利用 xpath 定位所需内容

利用 xpath 精确定位需要爬取的内容，在这一步做好了，可以节省之后数据清洗工作花费的大量时间。

1. 定位数据项的位置

每一页由 15 项数据项组成，除最后一页可能存在少于 15 项的可能以外，所有需要抓取得数据项都处于带有 `class="media-body _articles-item-body"`的 `div` 标签下，如图 4 所示。

图 4 数据项的位置

因此抓取的代码如下，返回值为包含当前页面所有数据项的 list。

```
response.xpath('//div[@class="media-body _articles-item-body"]')
```

2. 定位标题 title

定义标题属于比较麻烦的步骤了，当前问题可抽象成如下问题：抽取 a 标签下的所有文本内容，其中 a 标签中的格式如下所示：

```
<a>
    这里有很多
    <span>文本</span>
```

```
    信息，但是
    <span>文本</span>
    散布在各个地方。
</a>
```

要求将 a 标签中的所有文本内容，包括 span 标签中的所有文本内容都抽取出来，这里需要注意 a 标签与 span 标签中还存在文本内容，所以不能使用如将所有子节点的文本抽出等方法，这里使用 string(.)函数，它可以能去掉选择器中的所有标签并将所有文本抽出，另外执行 extract()方法将返回 list 数组，而该数组中只包含一个元素，我们这里需要的是字符串而非数组，因此数组的第一个元素即可。抓取代码如下：

```
item['title'] = sel.xpath('h2/div/a').xpath('string(.)').extract()[0]
```

3. 定位摘要 abstract

摘要是浓缩版的文章，虽然当前很多文章的摘要都是系统抽取文章内容的前部分自动生成的，但同样具备该篇文章的参考性。定位抽取摘要内容是所有的内容中最难的部分，当前问题可抽象成如下问题：抽取 div 标签下的有效文本内容，其中 div 标签中的格式如下所示：

```
<div>
    \t\t\n\n 这里有很多
    <span>文本</span>
    信息，但是
    <span>文本</span>
    散步在各个地方。\t\t\n\n
    <a>查看全文</a>
</div>
```

要求将 div 标签下的“这里有很多文本信息，但是文本散步在各个地方。”抽出，这里涉及三个问题：抽取 span 中的文本；去掉最后的 a 标签，因为“查看全文”是无效内容；去掉多余的制表符和换行符。即需要去除最后一个子节点 a，但子节点之间又包含文本内容，最后去掉前后的制表符和换行符。解决思路为：保留 div 标签下的所有文本内容和 span 标签下的文本内容，最后将得到的 list 转为字符串再去掉前后的制表符和换行符。抓取代码如下：

```
item['abstract'] = \
''.join(sel.xpath('div[@class="_search-articles-content"]/text()'
'|'
'div[@class="_search-articles-content"]//span/text()').extract()).stri
p()
```

4. 定位节点获得文本内容

定位作者与定位阅读数、点赞数和评论数问题类似，均为定位节点并获得文本内容，此类问题较为简单，抽取方法均类似，但需主要的是，这里同样需要将 list 转为字符串，方便后面处理。

```
    item['author']=sel.xpath('div[@class="infos"]/span/a/text()').extrac
 t()[0]
    item['viewers']=sel.xpath('div[@class="actions
clearfix"]/span[1]/text() ').extract()[0]
    item['like']  =  sel.xpath('div[@class="actions  clearfix"]/span[2]/
    text()').extract()[0]
    item['comments'] = sel.xpath('div[@class="actions clearfix"]/span[3]/
    text()').extract()[0]
```

5. 定位标签 tags

定位标签 tags 同样比较简单，但需注意 tag 可能存在，也可能只有一个或者存在多个，也因此定位上文中的信息时，不能使用 div 的次序来定位，否则会因 tags 缺少导致不存在该 div 时，div 次序错误而抓不到信息的异常情况。另外在使用'//'时，需要指定前节点否则将遍历整个 html 文档，当抓取到所有的标签内容时，使用','做为 tag 的分隔符，拼接字符串，具体抓取代码如下：

```
    item['tags']=",".join(sel.xpath('div[@class="_groups-tags"]//a/text(
)').extract())
```

6. 定位节点获得属性内容

定位作者链接、时间和链接问题类似，需要取出节点某个属性的属性值，使用'@'即可。

```
    item['author_link']=sel.xpath('div[@class="infos"]/span/a/@href').ex
tract()[0]
    item['time'] = sel.xpath('h2/span/time/@datetime').extract()[0]
    item['link'] = sel.xpath('h2/div/a/@href').extract()[0]
```

4.3 设置 pipeline

在 settings 中设置所有的 pipeline 以及 pipeline 的值，该值范围为 0-1000，该值决定 pipeline 的执行次序，即从小到大执行。

当前有三个 pipeline，分别为 cotent_pipeline“删除内容不符”、duplicates_pipeline“删除重复”和 json_writer_pipeline“以 json 格式导出到文件”。pipeline 的执行顺序为：content、duplicates 和 json_writer，所以 settings 中对 pipeline 的设置为：

```
ITEM_PIPELINES = {
    'bigdata.aliyun.duplicates_pipeline.DuplicatesPipeline': 100,
    'bigdata.aliyun.content_pipeline.ContentPipeline': 300,
    'bigdata.aliyun.json_writer_pipeline.JsonWriterPipeline': 800
}
```

1. 删除内容不符的文章

因为当前在 linux 环境下学习大数据，因此不需要 window 系列的内容，另外云栖社区中关于线下活动报名的文章也不是需要爬取的内容，因此“线下”、“预告”和“报名地址”关键字也排除在外。利用 re 中的正则搜索方法搜索“标题”和“摘要”，当出现上述关键字时则忽略该数据项，这里用到的正则表达式为：u'window|线下|预告|报名地址'。另外忽略该数据项需要使用到 DropItem 类，否则会抛出异常。具体代码如下：

```
import re
from scrapy.exceptions import DropItem
class ContentPipeline(object):
    def process_item(self, item, spider):
        if re.search(u'window|线下|预告|报名地址', item['title'], re.I):
            print "ignore this item"
            raise DropItem("Contains word that you don't want: %s" %
item['title'])
        elif re.search(u'window|线下|预告|报名地址', item['abstract'],
re.I):
            print "ignore this item"
            raise DropItem("Contains word that you don't want: %s" %
item['abstract'])
        else:
            return item
```

2. 删除重复的文章

搜索的关键字都是与大数据相关的关键字，因此搜索的文章很有可能重复，必须要去掉重复的内容，解决方法很简单，唯一区分文章的不是文章名，而是文章的链接，因此只需要将文章的链接存放到 set 中，每一次检查该 set 中是否存在该链接，如果有，则删除该文章，如果没有，则将链接添加到 set 中，具体代码如下：

```
from scrapy.exceptions import DropItem
class DuplicatesPipeline(object):
    def __init__(self):
        self.ids_seen = set()

    def process_item(self, item, spider):
        if item['link'] in self.ids_seen:
            raise DropItem("Duplicate item found:%s" % item)
        else:
            self.ids_seen.add(item['link'])
            return item
```

3. 以 json 格式导出到文件

将爬取结果输出到文件是为了方便阅读或进一步处理的必要措施，但在使用时必须注意文件的编码问题，因为开发环境在 ubuntu 下，在 window 系统中打开可能有误，因此需指定读文件的编码方式，并对中文做特殊处理。具体代码如下：

```
import json
import codecs
class JsonWriterPipeline(object):
    def __init__(self):
        self.file = codecs.open('item.jl', 'w', encoding='utf-8')

    def process_item(self, item, spider):
        line = json.dumps(dict(item)) + "\n"
        self.file.write(line.decode('unicode_escape'))
        return item
```

5 运行效果

运行爬虫即在终端中输入如下命令即可。

```
$ scrapy crawl aliyun
```

爬取结果：

```
{
 'downloader/request_bytes': 109867,
 'downloader/request_count': 225,
 'downloader/request_method_count/GET': 225,
 'downloader/response_bytes': 2225319,
 'downloader/response_count': 225,
 'downloader/response_status_count/200': 221,
 'downloader/response_status_count/403': 4,
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2016, 12, 12, 1, 8, 38, 70293),
 'item_dropped_count': 652,
 'item_dropped_reasons_count/DropItem': 652,
 'item_scraped_count': 2636,
 'log_count/DEBUG': 2865,
 'log_count/INFO': 3,
 'log_count/WARNING': 652,
 'request_depth_max': 50,
 'response_received_count': 225,
 'scheduler/dequeued': 224,
 'scheduler/dequeued/memory': 224,
 'scheduler/enqueued': 224,
 'scheduler/enqueued/memory': 224,
 'start_time': datetime.datetime(2016, 12, 12, 1, 7, 51, 955930)
}
```

可以从结果中看出共爬取到 2636 个数据项，共删除掉 652 个数据项。

其中保存到 json 文件中的每个数据项，格式如下：

```
{
    "like": "0 人赞",
    "tags": "docker,数据库,配置,docker-network,docker-book",
    "abstract": "Docker 有三个特定的功能，以帮助建立与环境无关的系统： 只读文件系统 环境变量注入 存储卷 处理卷是一个大主题，为了学习前两个功能，在本文的其余部分将改变对示例的需求。 WordPress 使用一个名为 MySQL 的数据库程序
```

来存储大部分数据，所以先确保运行 WordPress 的容器是只读文件系统，是一个好主意。 1 只读文件系统 使用只读文件系统产生以下两个积极效果。首先，你对容器不能更...",
"author_link": "/users/1597953772496508",
"comments": "0 人评论",
"link": "/articles/66130",
"author": "博文视点",
"time": "2016-12-09T16:14:12+08:00",
"title": "用 Docker 构建与环境无关的系统",
"viewers": "29 人浏览"
}

参考

安装 lxml：http://lxml.de/installation.html
Ubuntu 软件包：http://scrapy-chs.readthedocs.io/zh_CN/latest/topics/ubuntu.html
python-support: https://launchpad.net/ubuntu/+source/python-support
xpath 布尔表达式：https://msdn.microsoft.com/zh-cn/library/ms256081(v=vs.120).aspx
xpath 小结：
http://dtt4725.github.io/blog/2014/03/10/xpath%E5%AE%9E%E7%94%A8%E7%9A%84%E5%B0%8F%E7%BB%93/
scrapy 中文文档：http://scrapy-chs.readthedocs.io/zh_CN/1.0/intro/tutorial.html