****

网络工程

企业开发实训报告

题目名称：基于Spark的WordCount实现

班 级：网络151

学 号：201508030113

学生姓名：张玉龙

指导教师：刘小明

2018 年 7 月

# 中文摘要

近几年来，大数据机器学习和数据挖掘的并行化算法研究成为大数据领域一个较为重要的研究热点。早几年国内外研究者和业界比较关注的是在 Hadoop平台上的并行化算法设计。然而， HadoopMapReduce 平台由于网络和磁盘读写开销大，难以高效地实现需要大量迭代计算的机器学习并行化算法。Apache Spark 是一个新兴的大数据处理通用引擎，提供了分布式的内存抽象。Spark 最大的特点就是快，可比 Hadoop MapReduce 的处理速度快 100 倍。

本次主要是开发一个基于spark的文本聚类，除了spark环境的搭建，算法实现，自己主要负责数据准备，开发爬虫程序，将河南部分高校的图书馆新闻页爬取，并将爬取到的url,文章标题，文章内容存储到本地的txt文本中。并能根据spark对数据入库前的要求对数据格式进行变换。

**关键词： spark； 网络爬虫； 文章爬取； 文本聚类**

# **目 录**

[中文摘要 I](#_Toc478825369)

**[目 录](#_Toc478825370)** [II](#_Toc478825370)

[第1章 系统开发基础技术 3](#_Toc478825371)

[1.1 实训内容介绍 3](#_Toc478825372)

[1.2相关技术1 3](#_Toc478825373)

[1.2相关技术2 3](#_Toc478825374)

[1.3相关技术3 3](#_Toc478825375)

[第2章 系统分析 5](#_Toc478825376)

[2.1系统需求分析 5](#_Toc478825377)

[2.2系统概要设计 5](#_Toc478825378)

[2.3系统详细设计 5](#_Toc478825379)

[第3章 系统开发 7](#_Toc478825380)

[3.1系统开发环境配置 7](#_Toc478825381)

[3.2 系统开发 7](#_Toc478825382)

[第4章 系统测试 8](#_Toc478825383)

[4.1 测试环境 8](#_Toc478825384)

[4.2功能测试 8](#_Toc478825385)

[4.3测试结果分析 8](#_Toc478825386)

[第5章 结 论 9](#_Toc478825387)

[参考文献 10](#_Toc478825388)

[致 谢 11](#_Toc478825389)

**[附 录](#_Toc478825390)** [12](#_Toc478825390)

[附录: 详细的代码清单 12](#_Toc478825391)

# 第1章 系统开发基础技术

## 1.1 内容简介

### 1.1.1需求内容

实现spark对文本聚类，spark集群对若干条新闻的相似性筛选，在自己主要负责的数据准备方面，爬取高校图书馆新闻，包含文章的url、文章标题、文章正文等，由于网站可能会对爬虫封IP操作，所以需要考虑对策应对，以及有些高校网站需要添加cookie以及其他的一些认证信息，才能用爬虫访问其网页，将爬取到的信息保存到文本中。

### 1.1.2相关技术

1. .爬虫的BeautifulSoup库以及requests库的使用来对网页内容进行爬取。
2. .利用re库，正则表达规则来匹配每条新闻的url
3. .编写repsapce方法来规范文章的格式，避免正文中的换行等格式错误
4. .将数据按格式保存为本地的txt文本

### 1.1.3使用语言

使用的语言是python语言，涉及的3个库有Beautiful Soup库,re库,requests库。

## 1.2Beautiful Soup的使用

### 1.2.1 Beautiful Soup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。

### 1.2.2 Beautiful Soup用法

Beautiful Soup支持Python标准库中的HTML解析器,还支持一些第三方的解析器，如果我们不安装它，则 Python 会使用 Python默认的解析器，lxml 解析器更加强大，速度更快。相关解析器见表1.1所示。

表1.1解析器介绍

|  |  |  |  |
| --- | --- | --- | --- |
| 解析器 | 使用方法 | 优势 | 劣势 |
| Python标准库 | BeautifulSoup(markup, “html.parser”) | Python的内置标准库、执行速度适中、文档容错能力强 | Python 2.7.3 or 3.2.2)前 的版本中文档容错能力差 |
| lxml HTML 解析器 | BeautifulSoup(markup, “lxml”) | 速度快、文档容错能力强 | 需要安装C语言库 |
| lxml XML 解析器 | BeautifulSoup(markup, [“lxml”, “xml”])BeautifulSoup(markup, “xml”) | 速度快、唯一支持XML的解析器 | 需要安装C语言库 |
| html5lib | BeautifulSoup(markup, “html5lib”) | 最好的容错性、以浏览器的方式解析文档、生成HTML5格式的文档 | 速度慢、不依赖外部扩展 |

本程序主要用到的BeautifulSoup中的函数如下：

1. 创建对象：soup = BeautifulSoup(html)

Soup就是利用这个框架创建的一个对象，html就是我们想要爬取的网站的url,后面想要调用框架中的函数就可以直接用Soup针对html进行调用了。

1. 网站爬取：soup.find\_all(tag,(class\_),(limit))

利用find以及find\_all都是对网站结构进行搜索，tag就是爬取位置的标签，class\_就是标签中定义的类名。Limit是一个限制元素，限制标签的搜索次数。当find\_all中设置limit为1时那功能就与find没有差别。

1. 数据获取：strings.strings

爬取到的标签中的内容自动存储为列表，所以想要获取标签内的字符串的话，必须遍历列表然后用strings进行获取标签下的字符串。

如果想要获取标签内写的数据例如href等就需要用get进行获取。

## 1.3Re库以及正则使用

### 1.3.1 正则表达式

正则表达式（通项公式）是用来简洁表达一组字符串的表达式。优势是简洁，一行胜千言。专门被用于字符串匹配，相关语法见表1.2

表1.2 正则公式

|  |  |  |
| --- | --- | --- |
| 操作符 | 说明 | 实例 |
| . | 表示任何单个字符 |  |
| [  ] | 字符集，对单个字符给出取值范围 | [ab]表示a、b，[a-z]表示a到z单个字符 |
| [^  ] | 非字符集，对单个字符给出排除范围 | [^abc]表示非a或b或c的单个字符 |
| \* | 前一个字符0次或无限次扩展 | abc\* 表示ab、abc、abcc、abccc等 |
| + | 前一个字符1次或无限次扩展 | abc+ 表示abc、abcc、abccc等 |
| ? | 前一个字符0次或1次扩展 | abc? 表示ab、abc |
| | | 左右表达式任意一个 | abc|def 表示abc、def |
| {m} | 扩展前一个字符m次 | ab{2}c表示abbc |
| {m,n} | 扩展前一个字符m至n次（含n） | ab{1,2}c表示abc、abbc |
| ^ | 匹配字符串开头 | ^abc表示abc且在一个字符串的开头 |
| $ | 匹配字符串结尾 | abc$表示abc且在一个字符串的结尾 |
| ( ) | 分组标记，内部只能使用| 操作符 | (abc)表示abc，(abc|def)表示abc、def |
| \d | 数字，等价于[0‐9] |  |
| \w | 单词字符，等价于[A‐Za‐z0‐9\_] |  |

举几个例子：

|  |
| --- |
| **正则表达式 对应字符串**  **P(Y|YT|YTH|YTHO)?N 'PN'、'PYN'、'PYTN'、'PYTHN'、'PYTHON'**  **PYTHON+ 'PYTHON'、'PYTHONN'、'PYTHONNN' …**  **PY[TH]ON 'PYTON'、'PYHON'**  **PY[^TH]?ON 'PYON'、'PYaON'、'PYbON'、'PYcON'…**  **PY{:3}N 'PN'、'PYN'、'PYYN'、'PYYYN'</span>** |

### 1.3.2 Re库的使用

Re库是Python的标准库，主要用于字符串匹配。Re库中的主要函数见表1.3

1.3 Re库函数

|  |  |
| --- | --- |
| 函数 | 说明 |
| re.search() | 在一个字符串中搜索匹配正则表达式的第一个位置，返回match对象 |
| re.match() | 从一个字符串的开始位置起匹配正则表达式，返回match对象 |
| re.findall() | 搜索字符串，以列表类型返回全部能匹配的子串 |
| re.split() | 将一个字符串按照正则表达式匹配结果进行分割，返回列表类型 |
| re.finditer() | 搜索字符串，返回一个匹配结果的迭代类型，每个迭代元素是match对象 |
| re.sub() | 在一个字符串中替换所有匹配正则表达式的子串，返回替换后的字符串 |

贪婪匹配：

>>> match = re.search(r'PY.\*N', 'PYANBNCNDN')

>>> match.group(0)

'PYANBNCNDN'

Re库默认采用贪婪匹配，即输出匹配最长的子串。

最小匹配：

>>> match = re.search(r'PY.\*?N', 'PYANBNCNDN')

>>> match.group(0)

'PYAN'

只要长度输出可能不同的，都可以通过在操作符后增加?变成最小匹配。

其他操作符还包括：\*?、+?、??、{m,n}?

## 1.4 requests库的使用

### 1.4.1requests库的简介

requests是Python中一个第三方库，基于 urllib，采用 Apache2 Licensed 开源协议的 HTTP 库。它比 urllib 更加方便，可以节约我们大量的工作，完全满足 HTTP 测试需求。

### 1.4.2 requests库的使用

get()函数：

| **参数** | **说明** |
| --- | --- |
| url | 向服务器发送url请求 |
| params | 添加查询参数 |
| \*\*kwargs(headers) | 添加请求头信息 |

使用get()函数向服务器发送HTTP中的get请求

Requests 会自动解码来自服务器的内容。大多数 unicode 字符集都能被无缝地解码。

请求发出后，Requests 会基于 HTTP 头部对响应的编码作出有根据的推测。当你访问r.text 之时，Requests 会使用其推测的文本编码。你可以找出 Requests 使用了什么编码，并且能够使用r.encoding属性来改变它：

Cookie 是在 HTTP 协议下，服务器或脚本可以维护客户工作站上信息的一种方式。Cookie 是由 Web 服务器保存在用户浏览器（客户端）上的小文本文件，它可以包含有关用户的信息。无论何时用户链接到服务器，Web 站点都可以访问 Cookie 信息，目前有些 Cookie 是临时的，有些则是持续的。临时的 Cookie 只在浏览器上保存一段规定的时间，一旦超过规定的时间，该 Cookie 就会被系统清除。

持续的 Cookie 则保存在用户的 Cookie 文件中，下一次用户返回时，仍然可以对它进行调用。在 Cookie 文件中保存 Cookie，有些用户担心 Cookie 中的用户信息被一些别有用心的人窃取，而造成一定的损害。其实，网站以外的用户无法跨过网站来获得 Cookie 信息。如果因为这种担心而屏蔽 Cookie，肯定会因此拒绝访问许多站点页面。因为，当今有许多 Web 站点开发人员使用 Cookie 技术，例如 Session 对象的使用就离不开 Cookie 的支持。

# 第2章 系统分析

## 2.1系统需求分析

功能1：获取整个所要爬取页面的源码，导入urllib.request库，利用其中的urllib.request.urlopen(url1)方法，此时返回的就是url1这个页面的源码。

功能2：解析所获取的源码，并对其进行分析，利用find\_all方法来获取你想要的内容，因为该方法识别的是标签，所以要注意选取的标签要具有唯一性，否则该程序无法继续运行。

功能3：为了更加快捷的获取到每一条信息的url，我们导入了re库，和使用正则方法来匹配所需要的url.

功能4：使用python的bs4库对网站的url、文章标题、文章作者、发布时间、文章正文等信息进行获取，并保存到列表中等待存储以及统计操作，由于考虑到有些网站需要有认证信息，需要对个别网站的爬虫源码进行加头文件处理，并每一条的数据进行爬取然后对本页的数据进行统计并保存。

功能5：爬虫爬取到一项的数据之后，将url、文章标题、文章作者、发布时间、文章正文等信息以追加形式存放，利用 with open方法来向特定的文本中不停的追加数据，直到数据爬取完毕。

## 2.2系统概要设计

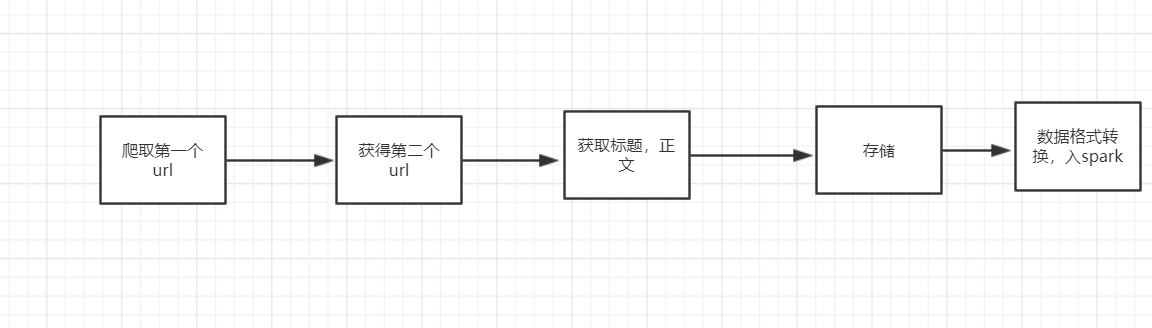
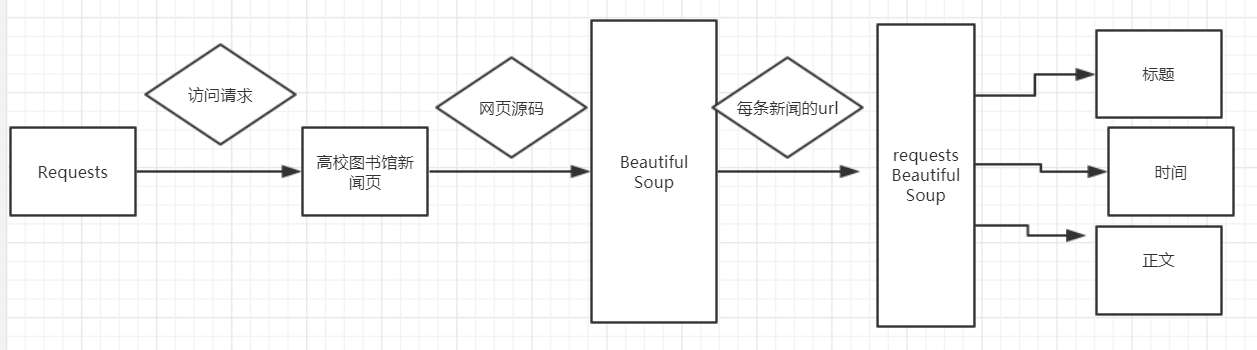


图 2.2

如图2.2所示，本系统首先根据url1对高校的新闻列表进行爬取，获取每一条新闻的url2,再通过requests,BeautifulSoup来对第二次爬取得内容进行获取和解析，来获取特定的信息，title,发出时间，以及正文内容，然后将每一条数据对应起来存放到本地的txt文本中，并依据spark入库时对数据的要求来对数据的格式进行变换。

2.3系统详细设计



1. if \_\_name\_\_=='\_\_main\_\_':
2. for i in range(1,54):
3. main(i)

这个是程序的入口，而range(1,54)是指新闻页的列表页的分页，如图2.3.1所示，在中原工学院的图书馆新闻页的url中，其他部分没有变化，所以可以这样来构造网址来参加循环。

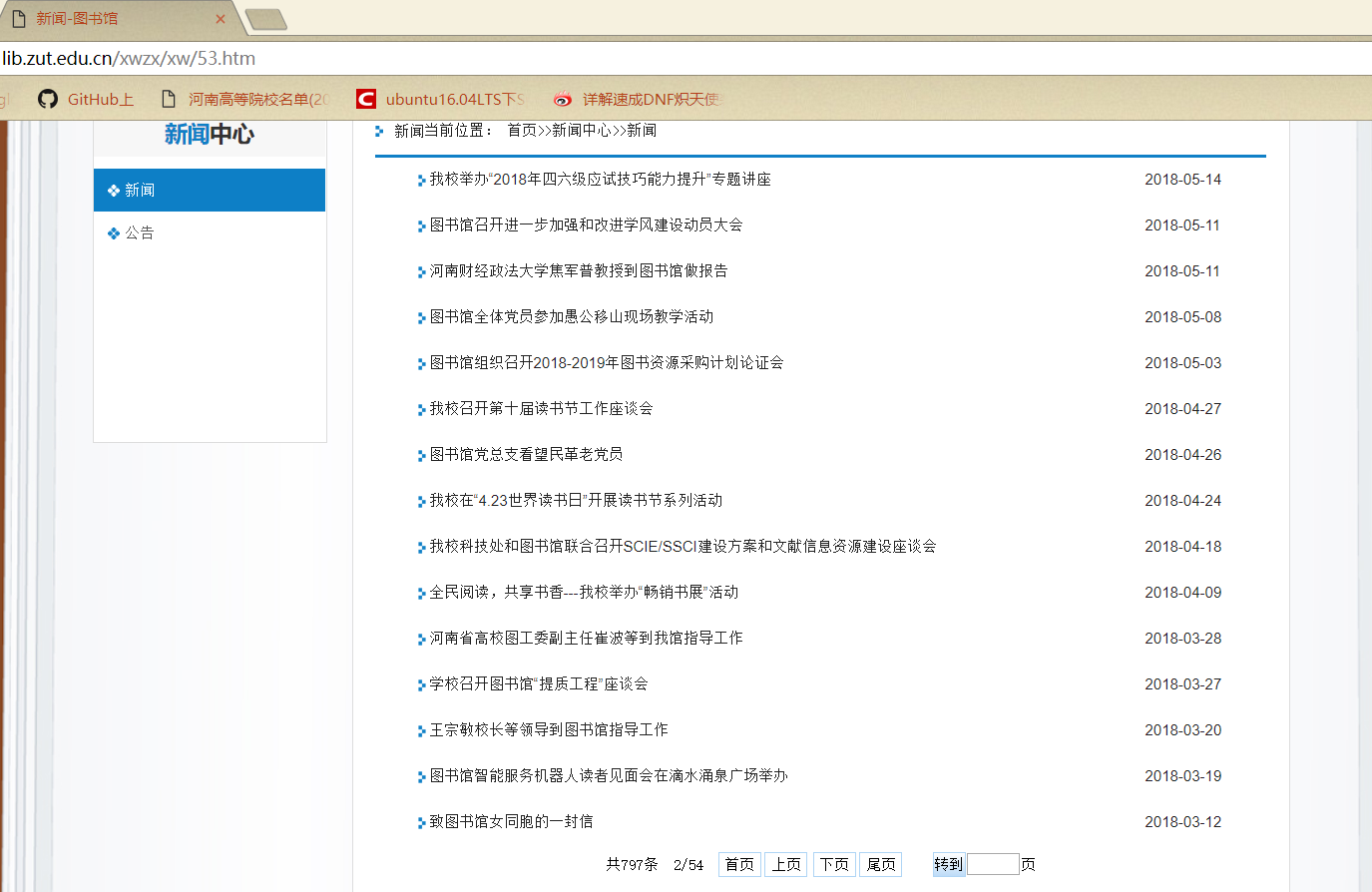


图 2.3.1

def main(id):

url ='http://lib.zut.edu.cn/xwzx/xw/'+str(id)+'.htm'

html = get\_text(url)

这是程序的主函数，在这里构造每一个新闻列表的网址，并运行自己编写的获取新闻内容的方法。

resapce(word):

这个方法是替换原文内容中的空格，换行，特殊字符等等，避免因没有替换而产生的格式错误，换句话说，处理数据时，是以换行来作为标识的，没有替换会导致串行。

html1=urllib.request.urlopen(url1).read().decode('utf-8')

这个方法是将url1整个页面的源码全部获取，Python decode() 方法以 encoding 指定的编码格式解码字符串。默认编码为字符串编码，即为以“utf-8”的编码格式来解码整个网站。

soup1=BeautifulSoup(html1,"html.parser")

用"html.parser"这个解析器来解析刚来获取后的源码。

rel = r'href="../../(.\*?)"'

利用正则来获取自己想要的内容。

title=soup2.find\_all('h1',attrs={'align':'center'})

这个是BeautifulSoup库中自带的方法，根据这个具有唯一性的标签集和属性集来获取自己所要的内容范围，也可以说精确查找。

for t1 in title:

tit=resapce(t1.get\_text())

因为title 返回的是一列表，所以需要写个循环，在运用自己的筛选函数Resapce来将获取的信息进行处理，t1.get\_text()这个方法是只获取其中的文本信息。

with open('zhongyuangongxueyuan.txt','a+', encoding='utf-8') as f:

f.write(tit+'')

这个就是很基本的存储到本地的txt，其实也可以存储到数据库，或者其他格式的文件，只是配置不同，而且若在与程序相同等级的地址中没有这个文件的话，他会自动创建一个，如果存在的话，他会在文本中追加所要存储的信息。

# 第3章 系统开发

本系统使用python语言编写了爬虫，本章介绍python以及java的环境配置。

## 3.1系统开发环境配置

### 3.1.1 python3.5环境配置

1） 打开WEB浏览器访问http://www.python.org/download/

2） 在下载列表中选择Window平台安装包，包格式为：python-XYZ.msi 文件 ， XYZ 为你要安装的版本号。

3） 要使用安装程序 python-XYZ.msi, Windows系统必须支持Microsoft Installer 2.0搭配使用。只要保存安装文件到本地计算机，然后运行它，看看你的机器支持MSI。Windows XP和更高版本已经有MSI，很多老机器也可以安装MSI。

4） 下载后，双击下载包，进入Python安装向导，安装非常简单，你只需要使用默认的设置一直点击"下一步"直到安装完成即可。

### 3.1.2 java1.8.0环境配置

1）打开网址http://www.oracle.com/找到适合的jdk进行安装。

2）进入计算机-系统属性-高级系统设置

3）在"系统变量"中设置3项属性，JAVA\_HOME,PATH,CLASSPATH(大小写无所谓),若已存在则点击"编辑"，不存在则点击"新建"。

变量设置参数如下：

变量名：JAVA\_HOME

变量值：C:\Program Files (x86)\Java\jdk1.8.0\_91(这项要根据自己的实际路径配置)

变量名：CLASSPATH

变量值：.;%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar; (记得前面有个".")

变量名：Path

变量值：%JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin;

## 3.2 系统开发

由于网站存在新闻列表项的url,每一条新闻的url,以及格式的筛选，有的网站还存在需要添加头文件，所以写了4个方法，有的需要添加head。

if \_\_name\_\_=='\_\_main\_\_':

他是程序的入口，调用了main函数，写了一个for循环，来构造url1.

核心代码：

if \_\_name\_\_=='\_\_main\_\_':

for i in range(1,54):

main(i)

Main 函数调用的是具体获取和处理新闻数据

核心代码：

html2=urllib.request.urlopen(url2).read().decode('utf-8')

soup2=BeautifulSoup(html2,"html.parser")

title=soup2.find\_all('h1',attrs={'align':'center'})

for t1 in title:

tit=resapce(t1.get\_text())

print(tit)

with open('zhongyuangongxueyuan.txt','a+', encoding='utf-8') as f:

f.write(tit+'')

在main函数中调用了 resapce(word):方法

主要是替代格式问题。

主要代码：

def resapce(word):

return word.replace('\n','').replace('\r','').replace('\t','').replace(' ','').replace('\xa0','').replace('&nbsp;','')

有的网站需要添加头文件，否则没有权限访问

核心代码：

headers = {

'Accept': 'text / html, application / xhtml + xml, application / xml;q = 0.9, image / webp, image / apng, \* / \*;q = 0.8',

'Accept - Encoding': 'gzip, deflate',

'Accept - Language': 'zh-CN,zh;q=0.9,en;q=0.8',

'Cache - Control': 'max - age = 0',

'Connection': 'keep - alive',

'Cookie': 'td\_cookie=3615363546; JSESSIONID=291525C93525A077BFB1706F222B51D8',

'Host': 'lib.haust.edu.cn',

'Upgrade - Insecure - Requests': ' 1',

'User - Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3423.2 Safari/537.36',

}

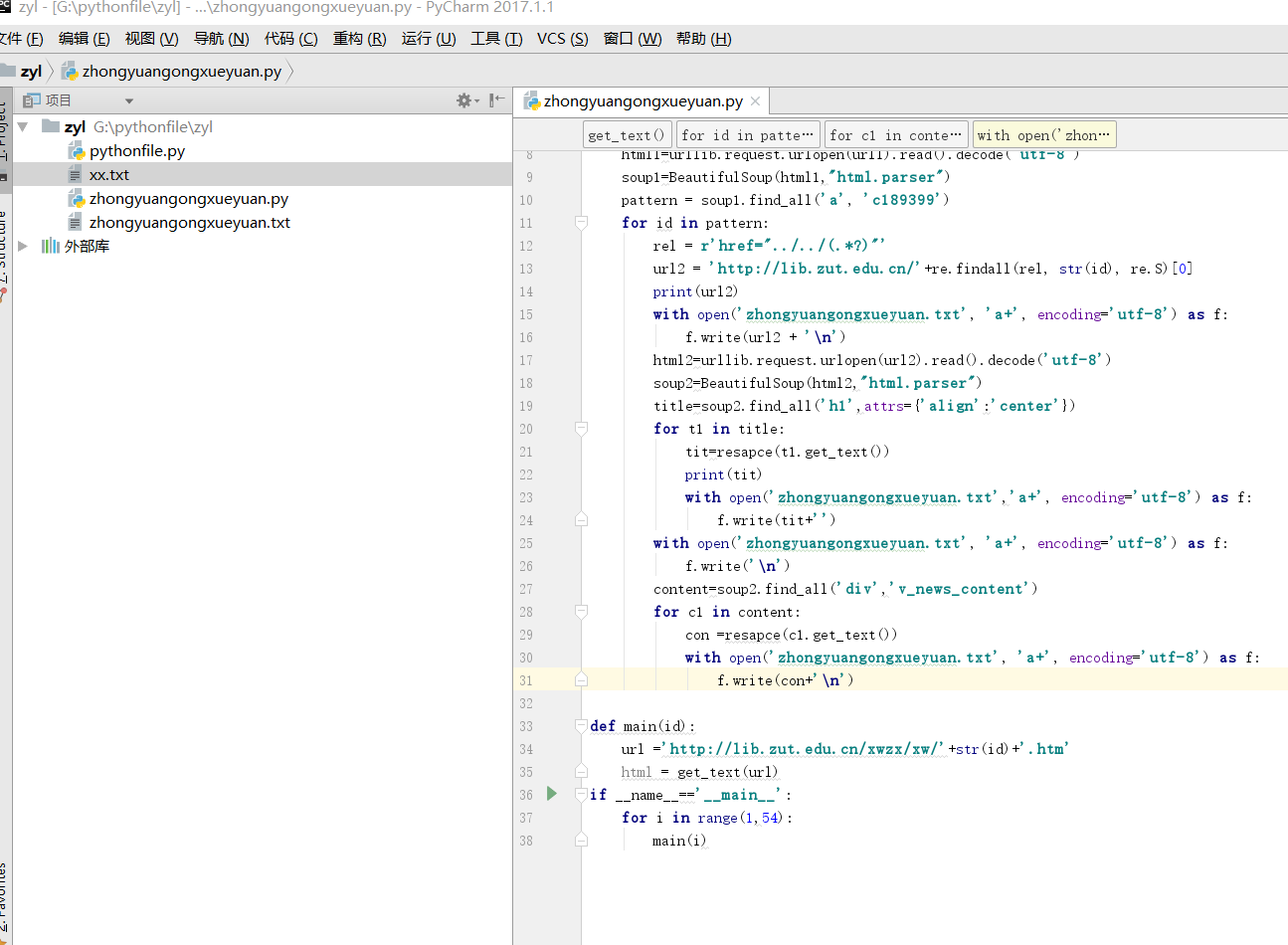
req = urllib.request.Request(url1, headers=headers)

# 第4章 系统测试

## 4.1 测试环境

Python3.5：编写爬虫系统

Pycharm编译器



## 4.2功能测试

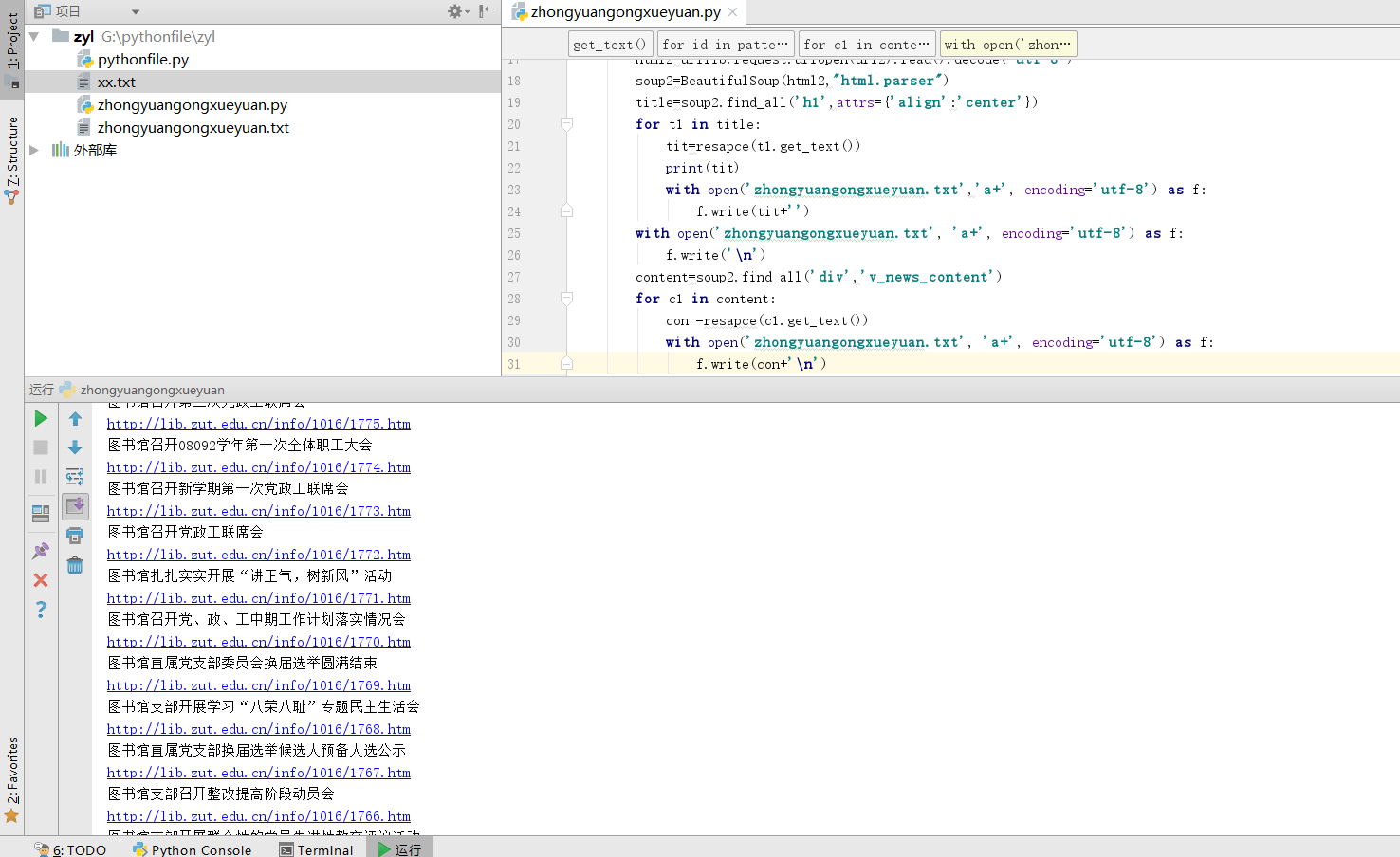


图4.2.2

如图4.2.2，将爬取得url,title在控制台打印出来。



图 4.2.3

如图4.2.3，数据爬取保存成功。

## 4.3测试结果分析

爬虫一开始爬取的时候，有时候程序无法运行，也没有语法错误，通过查阅，发现在find\_all方法中，参数标签选的不过完美，换句话说就是没有满足唯一性，导致爬取得数据太多，无法运行；也遇到过爬取得数量远远少于实际数量，通过分析发现是网页结构不同，url的信息会成为一个列表，因为只取了第一个索引的数据，导致数据遗漏，加了一个for循环解决了；有的网站拒绝链接，通过分析网页信息将headers加上后，解决此问题，每一个高校的网页设计都是不尽相同的，需要具体的网页具体分析。

# 第5章 结 论

本次实训实现了部分高校新闻的爬取，对于新闻的标题，Url，作者，日期，以及正文的爬取以及数据存入本地基本完成，在虚拟机中部署spark环境和spark集群，以及实现新闻的相似度分类聚合

通过本次实训学习了爬虫技术，正则匹配技术，python中bs4库、Re库以及requests库的使用，python存储到本地的方法，通过自己动手也完成了在虚拟机上spark集群的分布，算法的研究以及数据的转换。实践是检验真理的唯一标准，刚接触到课题的时候还觉得比较困难，但自己通过搜索资料，询问老师和同学让我学到了不少知识，也基本完成了本次实训的所有任务。这些学到的知识不仅仅用于完成本次实训，对于以后进一步的发展也打下了良好的基础。

# 参考文献

[1] Bill Lubanovic. Python语言及其应用 [M] 北京：人民邮电出版社，2015.01

[2] 小甲鱼. 零基础入门学习Python［M］. 清华大学出版社，2016.11

[3] 谌卫军、王浩娟. Java程序设计 [M] . 清华大学出版社

[4] Richard Lawson. 用Python写网络爬虫[M] 北京：人民邮电出版社，2016.08

# 致 谢

首先，要感谢我的指导老师，老师从百忙之中抽空指导我如何着手去完成实现，推荐我使用的语言，从实训的选题，前期准备到最终完成，夏老师都十分尊重我的想法和意见，同时为我解答了许多疑惑。再次感谢老师利用宝贵时间为我答疑解惑。

其次，我要感谢我的伙伴，当我在编写代码遇到困难的时候，他总能为我排查问题，并教会我一些技巧。并给我许多改善代码的中肯意见。

最后，要感谢实验室为我们提供了一个良好的实训环境，可以让我们安安静静的学习自己想要学习的知识，顺顺利利的完成我们的实训任务。

# **附 录**

## 附录: 详细的代码清单

### 爬虫代码：

import urllib.request

from bs4 import BeautifulSoup

import re

def resapce(word):

return word.replace('\n','').replace('\r','').replace('\t','').replace(' ','').replace('\xa0','').replace('&nbsp;','')

def get\_text(url1):

html1=urllib.request.urlopen(url1).read().decode('utf-8')

soup1=BeautifulSoup(html1,"html.parser")

pattern = soup1.find\_all('a', 'c189399')

for id in pattern:

rel = r'href="../../(.\*?)"'

url2 = 'http://lib.zut.edu.cn/'+re.findall(rel, str(id), re.S)[0]

print(url2)

with open('zhongyuangongxueyuan.txt', 'a+', encoding='utf-8') as f:

f.write(url2 + '\n')

html2=urllib.request.urlopen(url2).read().decode('utf-8')

soup2=BeautifulSoup(html2,"html.parser")

title=soup2.find\_all('h1',attrs={'align':'center'})

for t1 in title:

tit=resapce(t1.get\_text())

print(tit)

with open('zhongyuangongxueyuan.txt','a+', encoding='utf-8') as f:

f.write(tit+'')

with open('zhongyuangongxueyuan.txt', 'a+', encoding='utf-8') as f:

f.write('\n')

content=soup2.find\_all('div','v\_news\_content')

for c1 in content:

con =resapce(c1.get\_text())

with open('zhongyuangongxueyuan.txt', 'a+', encoding='utf-8') as f:

f.write(con+'\n')

def main(id):

url ='http://lib.zut.edu.cn/xwzx/xw/'+str(id)+'.htm'

html = get\_text(url)

if \_\_name\_\_=='\_\_main\_\_':

for i in range(1,54):

main(i)

### 环境的搭建：

Centos7安装spark-2.3.1

安装过程

1. 安装centos 64位 需要内存空间4G 其他的配置就根据个人的需求来配置了
2. 需要先下载spark的压缩包，不过在
3. 选择spark-2.3.1-bin-hadoop2.7,该版本spark需要Scala2.11，和hadoop2.7+
4. 需要先安装wget(下载文件的命令） #yum -y install wget（非必需内容，纯属为了下载软件包方便）

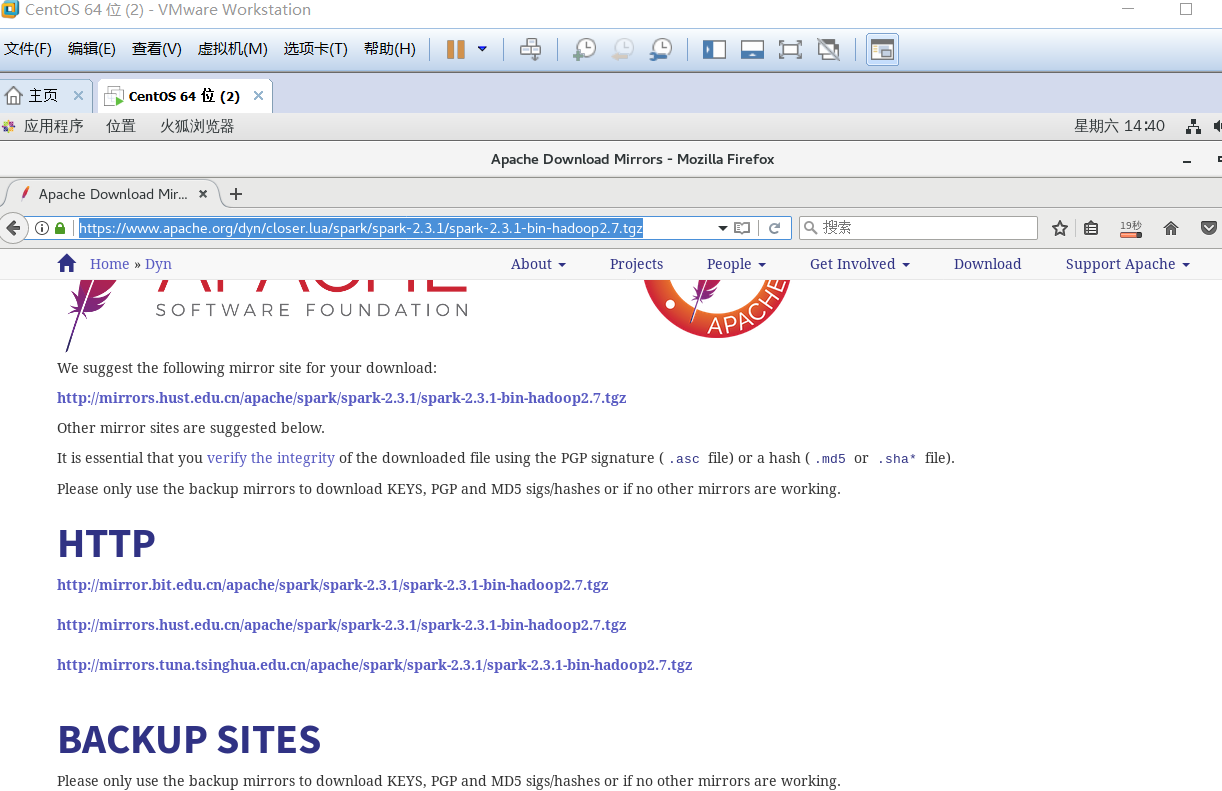


1. 下载spark-2.3.1.tgz

打开centos中自带的浏览器，打开

<https://www.apache.org/dyn/closer.lua/spark/spark-2.3.1/spark-2.3.1-bin-hadoop2.7.tgz>

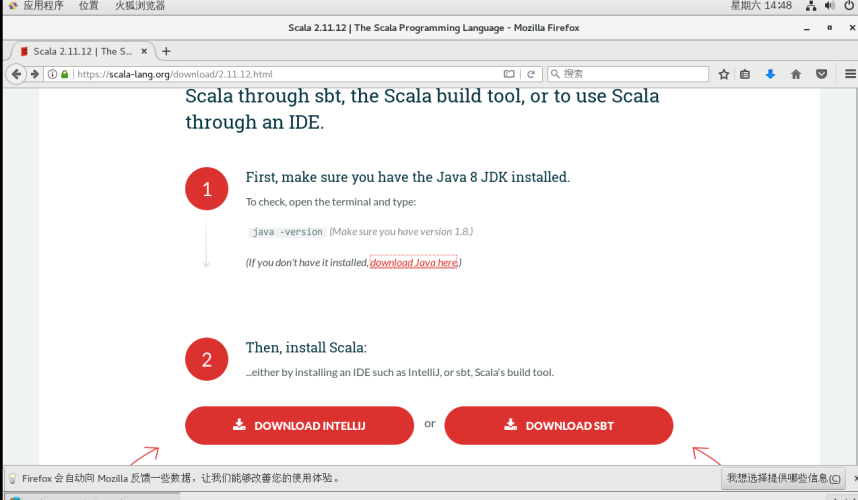
选择第一个压缩包，进行下载。



1. 选择spark-2.3.1-bin-hadoop2.7,该版本spark需要Scala2.11，和hadoop2.7+，所以我们还需要去[下载Scala2.11.12]

<https://scala-lang.org/download/2.11.12.html>

打开后，需要本机装有jdk8



下载jdk8\_64\_linux

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

还需要hadoop2.7+：

<http://mirrors.tuna.tsinghua.edu.cn/apache/hadoop/common/hadoop-2.7.6/>

把安装包都下好，准备解压。

7 . 安装jdk8（centos7中会有openjdk,此时需要先卸载openjdk，安装sun的jdk)

具体方法可以参见:

https://blog.csdn.net/u010533843/article/details/54989154(转载）

将压缩包解压

1 $ cd /home/cmfchina

2 $ tar -zxvf jdk-8u131-linux-x64.tar.gz

再建立一个java文件夹，把解压后的文件移动到那里面

1 mkdir /root/java

2 mv /root/cmfchina/jdk1.8.0\_171 /root/java

修改环境变量

Vim /etc/profile

在文件的最后加上

export JAVA\_HOME=/root/java/jdk1.8.0\_171

export JRE\_HOME=JRE\_HOME=${JAVA\_HOME}/jre

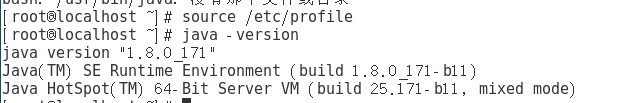
export CLASSPATH=.:${JAVA\_HOME}/lib:${JRE\_HOME}/lib:$CLASSPATH

export JAVA\_PATH=${JAVA\_HOME}/bin:${JRE\_HOME}/bin

export PATH=$PATH:${JAVA\_PATH}

然后source /etc/profile/ 通过命令source /etc/profile让profile文件立即生效

输入java -version来查看jdk版本



安装成功。

1. 解压下载好的3个安装包

mkdir /usr/spark

tar -zxvf scala-2.11.12.tgz -C /usr/spark/

tar -zxvf spark-2.3.1-bin-hadoop2.7\_3.tgz -C /usr/spark/

tar -zxvf hadoop-2.7.6.tar.gz -C /usr/spark/



解压成功，接下来设置环境变量。

gedit /etc/profile

除了刚才的jdk环境还需要加上

#scala env

export SCALA\_HOME=/usr/spark/scala-2.11.12

export PATH=$PATH:$SCALA\_HOME/bin

#spark env

export SPARK\_HOME=/usr/spark/spark-2.3.1-bin-hadoop2.7

export PATH=$PATH:$SPARK\_HOME/bin

#hadoop env

export HADOOP\_HOME=/usr/spark/hadoop-2.7.6

export PATH=$PATH:HADOOP\_HOME/bin

注意：使用source /etc/profile 命令使其生效

1. 设置hadoop配置文件

（1）新建以下目录

cd /usr/spark

mkdir hadoop

mkdir hadoop/hdfs

mkdir hadoop/hdfs/data

mkdir hadoop/hdfs/name

mkdir hadoop/temp

1. 修改/usr/spark/hadoop-2.7.6/etc/hadoop/文件夹下的多个配置文件

a.修改core-site.xml

Cd /usr/spark/hadoop-2.7.6/etc/hadoop/

然后vim core-site.xml

<configuration>

<property>

<name>fs.default.name</name>

<value>hdfs://master:9000</value>

<description>HDFS的URI，文件系统://namenode标识:端口号</description>

</property>

<property>

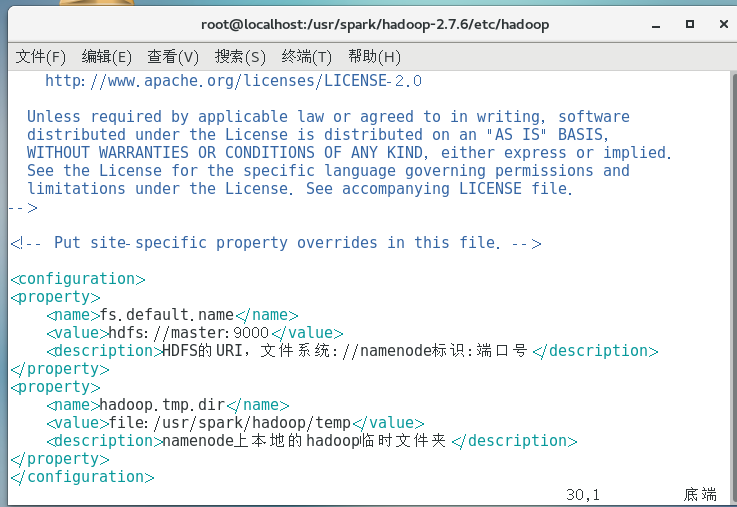
<name>hadoop.tmp.dir</name>

<value>file:/usr/spark/hadoop/temp</value>

<description>namenode上本地的hadoop临时文件夹</description>

</property>

</configuration>



b.修改hdfs-site.xml

```

<configuration>

<property>

<name>dfs.name.dir</name>

<value>file:/usr/spark/hadoop/hdfs/name</value>

<description>namenode上存储hdfs名字空间元数据 </description>

</property>

<property>

<name>dfs.data.dir</name>

<value>file:/usr/spark/hadoop/hdfs/data</value>

<description>datanode上数据块的物理存储位置</description>

</property>

<property>

<name>dfs.replication</name>

<value>1</value>

<description>副本个数，配置默认是3,应小于datanode机器数量</description>

</property>

<property>

<name>dfs.namenode.secondary.http-address</name>

<value>master:9001</value>

</property>

</configuration>

```

c.修改mapred-site.xml

```

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

<property>

<name>mapreduce.jobhistory.address</name>

<value>master:10020</value>

</property>

<property>

<name>mapreduce.jobhistory.webapp.address</name>

<value>master:19888</value>

</property>

```

d.修改yarn-site.xml

```

<configuration>

<!-- Site specific YARN configuration properties -->

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

<property>

<name>yarn.resourcemanager.address</name>

<value>master:8032</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>master:8030</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>master:8031</value>

</property>

<property>

<name>yarn.resourcemanager.admin.address</name>

<value>master:8033</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address</name>

<value>master:8088</value>

</property>

</configuration>

e.接着将hadoop-env.sh,mapred-env.sh,yarn-env.sh中的JAVA\_HOME都修改为我们配置好的java路径（根据个人jdk设置而定）

比如，我的是`export JAVA\_HOME=/root/java/jdk1.8.0\_171

` 

####4、配置hadoop节点信息

修改/usr/spark/hadoop-2.7.6/etc/hadoop/下的slaves文件，添加节点，本次预计使用三个slave节点,内容如下：

```

localhost

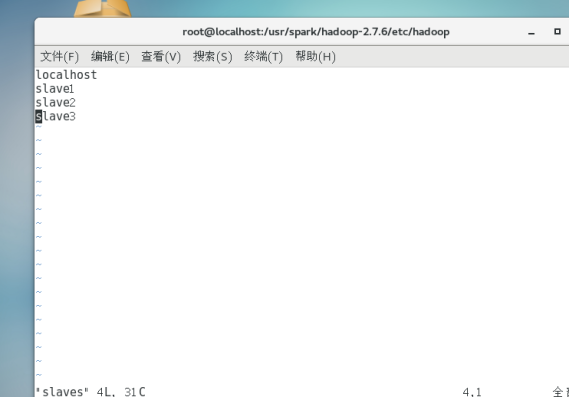
slave1

slave2

slave3

```

其中2和3是我的节点，10.133.43.83 控制和1是10.133.4.223



修改主机hosts文件，内容如下：

![](http://chuantu.biz/t6/336/1530256065x-1404729680.bmp)

修改主机名称gedit /etc/hostname

```

gedit /etc/hostname

在做的过程中，发现校网的ip不适用此次实验，因为在校网这个局域网中，两台主机是无法互相访问的，此时我们的解决办法是采用实验室的网络，不过由于DHCP自动分配，每次链接的ip地址都不同，应及时修改其配置文件。

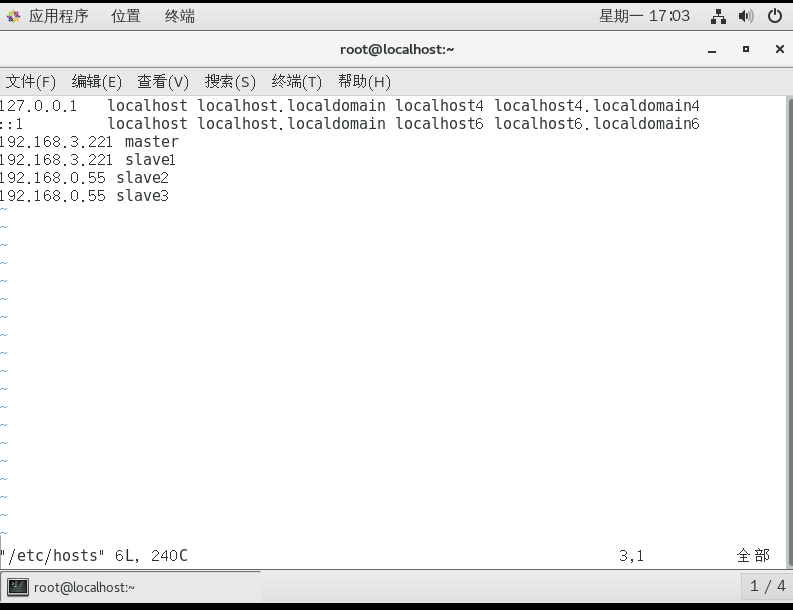
中心节点192.168.3.221 我的节点192.168.0.55

此时修改的部分为

Hostname文件中删除一些东西

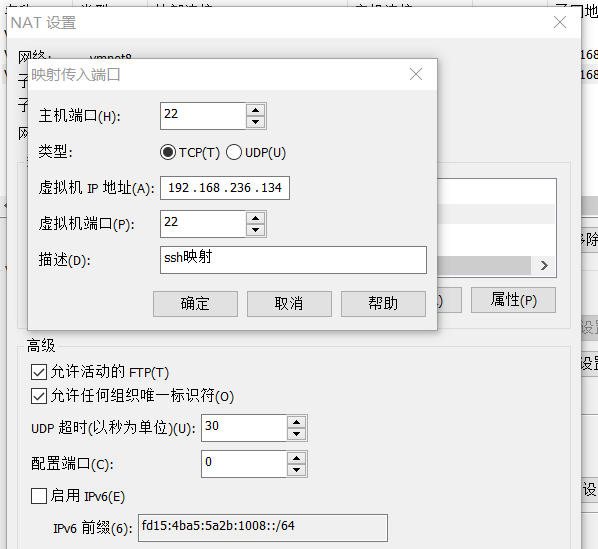


只需要保留自己节点的名称。

在hosts文件中，更改ip地址

此时需要对虚拟机进行与window端口映射

可以参考https://blog.csdn.net/u010947098/article/details/54614756



此时虚拟机与window共享一个IP地址。

需要在终端启动一下ssh,命令为

第一步

#查看本机是否安装SSH软件包

[root@localhost ~]# rpm -qa | grep ssh

openssh-server-6.6.1p1-12.el7\_1.x86\_64

openssh-clients-6.6.1p1-12.el7\_1.x86\_64

libssh2-1.4.3-8.el7.x86\_64

openssh-6.6.1p1-12.el7\_1.x86\_64

#如果没有，则需要安装

[root@localhost /]# yum install openssh-server

第二步

#开启 SSH 服务

[root@localhost ~]# service sshd start

Redirecting to /bin/systemctl start sshd.service

#查看TCP 22端口是否打开

[root@localhost ~]# netstat -ntpl | grep 22

tcp 0 0 0.0.0.0:22 0.0.0.0:\* LISTEN 17816/sshd

tcp6 0 0 :::22 :::\* LISTEN 17816/sshd

第三步

#接下来便可使用终端仿真程序（例如putty）去登陆远程主机

如果你在客户端不能连接SSH服务的话，那可能是防火墙的原因，终端命令行中输入 iptables -nL 来看是否开放了ssh tcp 22 端口：

[root@localhost ~]# iptables -nL

你可以将防火墙中的规则条目清除掉：

[root@localhost ~]# iptables -F

5、配置ssh无密码登录

(1)ubuntu自带ssh-client，我们还需要ssh-server

apt-get install openssh-server

(2)启动ssh服务

/etc/init.d/ssh start

(3)查看sshd是否启动

ps -e | grep ssh

(4)root账户默认不允许登录ssh，修改权限

`gedit /etc/ssh/sshd\_config`

找到Authentication，修改PermitRootLogin yes，保存

```

/etc/init.d/ssh restart

```

(5)ssh登录本地

`ssh localhost`

SSH首次登录会有提示，直接输入yes即可，这时是需要密码的

(6)生成秘钥

`ssh-keygen -t rsa`

之后一直按Enter键，默认将秘钥保存在.shh/id\_rsa文件中

(7)RSA公钥加入授权文件

cd .ssh

cp id\_rsa.pub authorized\_keys

(8)重新登录，实现免密码登录localhost

6、配置master免密码登录slave1

这里使用scp命令，可自行搜索相关信息，配置确保slave1节点已安装ssh-server

(1)将master根目录下密码复制到slave1的根目录下

root@master:~# scp ~/.ssh/id\_rsa.pub root@slave1:~/.ssh

(2)在slave1节点将RSA公钥加入授权文件

root@slave:~# cp .ssh/id\_rsa.pub authorized\_keys

如有多台计算机，重复以上操作即可

(3)在master节点上ssh登录slave1,实现免密码登录

ssh slave1

7、配置slave1节点的hadoop，同master节点，复制过去即可

8、Hadoop运行

(1)格式化分布式文件系统，在master节点下

cd /usr/local/hadoop/hadoop-2.7.3

bin/hadoop namenode -format

(2)启动hadoop守护进程

sbin/start-all.sh

(3)检测启动情况

jps

这时在master节点可以看到NameNode，SecondaryNameNode，ResourceManager

在slave1节点可以看到DataNode，NodeManager，因为我也将master节地点添加为slaves，所以也能看到类似信息

(4)停止hadoop进程

sbin/stop-all.sh

1. 修改spark配置文件

1)$SPARK\_HOME/conf/spark-env.sh

cp spark-env.sh.template spark-env.sh

添加以下内容:

export SCALA\_HOME=/usr/local/scala/scala-2.12.1

export JAVA\_HOME=/usr/local/java/jdk1.8.0\_111

export SPARK\_MASTER\_IP=master

export SPARK\_WORKER\_MEMORY=512m

export HADOOP\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop

2)$SPARK\_HOME/conf/slaves

这是在那个路径下再进行配置

cp slaves.template slaves

添加以下内容:

master

slave1

### 算法代码：

import org.ansj.splitWord.analysis.BaseAnalysis

import org.apache.spark.graphx.Graph

import org.apache.spark.mllib.feature.{HashingTF, IDF}

import org.apache.spark.{SparkConf, SparkContext}

import org.apache.spark.mllib.linalg.{SparseVector => SV}

object groupNews {

def main(args: Array[String]): Unit = {

//相似度阈值

val sim = 0.5

val sparkconfig = new SparkConf().setAppName("DocSim").setMaster("local[2]")

val ctx = new SparkContext(sparkconfig)

val path = "/home/fenglulu/java/news/henanshifandaxue.txt"

//加载过滤数据(id,title,content)

val input = ctx.textFile(path).map(x => x.split("====").toSeq).filter(x => x.length == 3 && x(1).length > 1 && x(2).length > 1)

//分词(id,title,[words])

val splitWord = input.map(x => (x(0), x(1), BaseAnalysis.parse(x(2)).toStringWithOutNature(" ").split(" ").toSeq))

//聚类初始化 计算文章向量(id,(id,content,title))

val init\_rdd = input.map(a => {

(a(0).toLong, a)

})

init\_rdd.cache()

//计算TF-IDF特征值

val hashingTF = new HashingTF(Math.pow(2, 18).toInt)

//计算TF

val newSTF = splitWord.map(x => (x.\_1, hashingTF.transform(x.\_3))

)

newSTF.cache()

//构建idf model

val idf = new IDF().fit(newSTF.values)

//将tf向量转换成tf-idf向量

val newsIDF = newSTF.mapValues(v => idf.transform(v)).map(a => (a.\_1, a.\_2.toSparse))

newsIDF.take(10).foreach(x => println(x))

//构建hashmap索引 ,特征排序取前10个

val indexArray\_pairs = newsIDF.map(a => {

val indices = a.\_2.indices

val values = a.\_2.values

val result = indices.zip(values).sortBy(-\_.\_2).take(10).map(\_.\_1)

(a.\_1, result)

})

//(id,[特征ID])

indexArray\_pairs.cache()

indexArray\_pairs.take(10).foreach(x => println(x.\_1 + " " + x.\_2.toSeq))

//倒排序索引 (词ID,[文章ID])

val index\_idf\_pairs = indexArray\_pairs.flatMap(a => a.\_2.map(x => (x, a.\_1))).groupByKey()

index\_idf\_pairs.take(10).foreach(x => println(x.\_1 + " " + x.\_2.toSeq))

//倒排序

val b\_content = index\_idf\_pairs.collect.toMap

//广播全局变量

val b\_index\_idf\_pairs = ctx.broadcast(b\_content)

//广播TF-IDF特征

val b\_idf\_parirs = ctx.broadcast(newsIDF.collect.toMap)

//相似度计算 indexArray\_pairs(id,[特征ID]) b\_index\_idf\_pairs( 124971 CompactBuffer(21520885, 21520803, 21521903, 21521361, 21524603))

val docSims = indexArray\_pairs.flatMap(a => {

//将包含特征的所有文章ID

var ids: List[Long] = List()

//存放文章对应的特征

var idfs: List[(Long, SV)] = List()

//遍历特征，通过倒排序索引取包含特征的所有文章,除去自身

a.\_2.foreach(b => {

ids = ids ++ b\_index\_idf\_pairs.value.get(b).get.filter(x => (!x.equals(a.\_1))).map(x => x.toLong).toList

})

//b\_idf\_parirs(tf-idf特征),遍边文章，获取对应的TF-IDF特征

ids.foreach(b => {

idfs = idfs ++ List((b, b\_idf\_parirs.value.get(b.toString).get))

})

//获取当前文章TF-IDF特征

val sv1 = b\_idf\_parirs.value.get(a.\_1).get

import breeze.linalg.\_

//构建当前文章TF-IDF特征向量

val bsv1 = new SparseVector[Double](sv1.indices, sv1.values, sv1.size)

//遍历相关文章

val result = idfs.map {

case (id2, idf2) =>

val sv2 = idf2.asInstanceOf[SV]

//对应相关文章的特征向量

val bsv2 = new SparseVector[Double](sv2.indices, sv2.values, sv2.size)

//计算余弦值

val cosSim = bsv1.dot(bsv2) / (norm(bsv1) \* norm(bsv2))

(a.\_1, id2, cosSim)

}

// 文章1，文章2，相似度

result.filter(a => a.\_3 >= sim)

})

docSims.take(10).foreach(x => println(x))

//取出所有，有相似度的文章

val vertexrdd = docSims.map(a => {

(a.\_2.toLong, a.\_1.toLong)

})

//构建图

val graph = Graph.fromEdgeTuples(vertexrdd, 1)

val graphots = Graph.graphToGraphOps(graph).connectedComponents().vertices

//聚类初始化 计算文章向量 init\_rdd(id,(id,content,title))

init\_rdd.join(graphots).take(10).foreach(x => println(x))

val simrdd = init\_rdd.join(graphots).map(a => {

(a.\_2.\_2, (a.\_2.\_1, a.\_1))

})

val simrddtop = simrdd.groupByKey().filter(a => a.\_2.size >= 6).sortBy(-\_.\_2.size).take(50)

val simrdd2 = ctx.parallelize(simrddtop, 18)

simrdd2.take(10).foreach(x => {

val titles = x.\_2.map(x => x.\_1(1)).toArray

//选取事件主题名

val title = mostSimilartyTitle(titles)

println("事件---------------------" + title)

println(x.\_1)

x.\_2.foreach(x => println(x.\_2 + " " + x.\_1(0) + " " + x.\_1(1)))

})

}

/\*\*

\* 相似度比对 最短编辑距离

\* @param s

\* @param t

\* @return

\*/

def ld(s: String, t: String): Int = {

var sLen: Int = s.length

var tLen: Int = t.length

var cost: Int = 0

var d = Array.ofDim[Int](sLen + 1, tLen + 1)

var ch1: Char = 0

var ch2: Char = 0

if (sLen == 0)

tLen

if (tLen == 0)

sLen

for (i <- 0 to sLen) {

d(i)(0) = i

}

for (i <- 0 to tLen) {

d(0)(i) = i

}

for (i <- 1 to sLen) {

ch1 = s.charAt(i - 1)

for (j <- 1 to tLen) {

ch2 = t.charAt(j - 1)

if (ch1 == ch2) {

cost = 0

} else {

cost = 1

}

d(i)(j) = Math.min(Math.min(d(i - 1)(j) + 1, d(i)(j - 1) + 1), d(i - 1)(j - 1) + cost)

}

}

return d(sLen)(tLen)

}

/\*\*

\*

\* @param src

\* @param tar

\* @return

\*/

def similarity(src: String, tar: String): Double = {

val a: Int = ld(src, tar)

1 - a / (Math.max(src.length, tar.length) \* 1.0)

}

/\*\*

\* 选出一组字符串 中相似度最高的

\* @param strs

\* @return

\*/

def mostSimilartyTitle(strs: Array[String]): String = {

var map: Map[String, Double] = Map()

for (i <- 0 until strs.length) {

for (j <- i + 1 until strs.length) {

var similar = similarity(strs(i), strs(j))

if (map.contains(strs(i)))

map += (strs(i) -> (map.get(strs(i)).get + similar))

else

map += (strs(i) -> similar)

if (map.contains(strs(j)))

map += (strs(j) -> (map.get(strs(j)).get + similar))

else

map += (strs(j) -> similar)

}

} //end of for

if (map.size > 0)

map.toSeq.sortWith(\_.\_2 > \_.\_2)(0).\_1

else

""

}

}