软件工程课程设计

社交网络数据收集算法的设计

|  |  |  |
| --- | --- | --- |
| 组号 | 第21组 | |
| 组长 | 姓名：盖云东 | 学号：130104010049 |
| 组员 | 姓名：任志成 | 学号：130104010121 |
| 组员 | 姓名：马剑楠 | 学号：130104010004 |
| 组员 | 姓名：陈海涛 | 学号：130104010045 |

摘要

随着互联网的发展，人们正处于一个信息爆炸的时代。社交网络数据信息量大、主题性强,具有巨大的数据挖掘价值,是互联网大数据的重要组成部分。一些社交平台如Twitter、新浪微博、人人网等,允许用户申请平台数据的采集权限,并提供了相应的API 接口采集数据,通过注册社交平台、申请API授权、调用API 方法等流程获取社交信息数据。但社交平台采集权限的申请比较严格,申请成功后对于数据的采集也有限制。因此,本文采用网络爬虫的方式,利用社交账户模拟登录社交平台,访问社交平台的网页信息,并在爬虫任务执行完毕后,及时返回任务执行结果。相比于过去的信息匮乏，面对现阶段海量的信息数据，对信息的筛选和过滤成为了衡量一个系统好坏的重要指标。本文运用了爬虫和协同过滤算法对网络社交数据进行收集。

关键词：软件工程；社交网络；爬虫；协同过滤算法

目录

[摘要 - 2 -](#_Toc455325061)

[目录 - 3 -](#_Toc455325062)

[课题研究的目的 - 1 -](#_Toc455325063)

[1.1 课题研究背景 - 1 -](#_Toc455325064)

[2 优先抓取策略--PageRank - 2 -](#_Toc455325065)

[2.1 PageRank简介 - 2 -](#_Toc455325066)

[2.2 PageRank流程 - 2 -](#_Toc455325067)

[3 爬虫 - 4 -](#_Toc455325068)

[3.1 爬虫介绍 - 4 -](#_Toc455325069)

[3.1.1爬虫简介 - 4 -](#_Toc455325070)

[3.1.2 工作流程 - 4 -](#_Toc455325071)

[3.1.3 抓取策略介绍 - 5 -](#_Toc455325072)

[3.2 工具介绍 - 6 -](#_Toc455325073)

[3.2.1 Eclipse - 7 -](#_Toc455325074)

[3.2.2 Python语言 - 7 -](#_Toc455325075)

[3.2.3 BeautifulSoup - 7 -](#_Toc455325076)

[3.3 实现 - 8 -](#_Toc455325077)

[3.4 运行结果 - 9 -](#_Toc455325078)

[4 算法部分 - 10 -](#_Toc455325079)

[4.1获取数据的三种途径 - 10 -](#_Toc455325080)

[4.1.1通过新浪微博模拟登录获取数据 - 10 -](#_Toc455325081)

[4.1.2 通过调用微博API接口获取用户微博数据 - 11 -](#_Toc455325082)

[4.2基于用户的协同过滤算法 - 14 -](#_Toc455325083)

[4.2.1集体智慧和协同过滤 - 14 -](#_Toc455325084)

[4.2.2深入协同过滤核心 - 15 -](#_Toc455325085)

[4.3算法实现 - 18 -](#_Toc455325086)

[结论 - 22 -](#_Toc455325087)

[参考文献 - 23 -](#_Toc455325088)

课题研究的目的

## 课题研究背景

互联网导致一种全新的人类社会组织和生存模式悄然走进我们，构建了一个超越地球空问之上的、巨大的群体——网络群体，21世纪的人类社会正在逐渐浮现出崭新的形态与特质，网络全球化时代的个人正在聚合为新的社会群体。随着社交网站的兴起，网络社交蓬勃发展，新的互联网热再次升温，有分析人士甚至说，网络社交将缔造人际交往的新模式。互联网的兴起打破了传统的社会交往方式,简单、快捷和无距离的社交体验推动社交网络快速发展,以Facebook、Twitter、微博等为代表的应用吸引了大量活跃网络用户,社交网络信息呈现爆发式的增长。社交网络信息反映了用户的网络行为特征,通过对这些信息的研究,可以实现社会舆论监控、网络营销、股市预测等。社交网络信息的重要价值在于实时性,如何快速、准确、有效地获取目标信息非常重要。但社交网络属于Deep Web 的专有网络,信息量大、主题性强,传统搜索引擎无法索引这些Deep Web 页面,只有通过网站提供的查询接口或登录网站才能访问其信息,这增加了获取社交网络信息的难度。

目前国外有关社交网络数据采集模型的研究较少,对社交网络的研究主要集中在社会网络分析领域。国内社交网络平台的数据采集技术研究有一定成果,如文献提出并实现一种利用新浪微博应用程序接口(Application Programming Interface,API)和网络数据流相结合的方式采集数据,文献利用人人网开发平台提供的API 实现数据采集,并通过WebBrowser 和HttpFox 监测信息交互时的数据包,实现动态获取Ajax 页面信息等。

2 优先抓取策略--PageRank

## 2.1 PageRank简介

PageRank，即网页排名，又称网页级别、Google左侧排名或佩奇排名是Google创始人拉里·佩奇和谢尔盖·布林于1997年构建早期的搜索系统原型时提出的链接分析算法，自从Google在商业上获得空前的成功后，该算法也成为其他搜索引擎和学术界十分关注的计算模型。目前很多重要的链接分析算法都是在PageRank算法基础上衍生出来的。

## 2.2 PageRank流程

首先PageRank的计算充分利用了两个假设：**数量假设**和**质量假设**。步骤如下：  
 **1) 在初始阶段：**网页通过链接关系构建起Web图，每个页面设置相同的PageRank值，通过若干轮的计算，会得到每个页面所获得的最终PageRank值。随着每一轮的计算进行，网页当前的PageRank值会不断得到更新。

2) 在一轮中更新页面PageRank得分的计算方法：在一轮更新页面PageRank得分的计算中，每个页面将其当前的PageRank值平均分配到本页面包含的出链上，这样每个链接即获得了相应的权值。而每个页面将所有指向本页面的入链所传入的权值求和，即可得到新的PageRank得分。当每个页面都获得了更新后的PageRank值，就完成了一轮PageRank计算。

然后，引入阻尼系数p，p的值一般为0.85。

完整公式如下：

IMG_261

IMG_262是被研究的页面，IMG_263是IMG_264链入页面的数量，IMG_265是IMG_266链出页面的数量，而N是所有页面的数量。

使用幂法求PageRank。那公式可以转换为求解IMG_271的值，其中矩阵为 A = q × P + ( 1 一 q) \* IMG_272/N 。 P 为概率转移矩阵，IMG_273为 n 维的全 1 行。

1. 计算P概率转移矩阵：

1) 用矩阵表示连通图

2) 网页链接概率矩阵

3) 概率转移矩阵P

1. 计算A矩阵：

1) 计算 **IMG_284/N**

2) 计算**A矩阵**

1. 循环迭代计算PageRank

3 爬虫

## 3.1 爬虫介绍

### 3.1.1爬虫简介

网络爬虫(又被称为网页蜘蛛，网络机器人，在FOAF社区中间，更经常被称为网页追逐者)，是一种按照一定的规则，自动的抓取万维网信息的程序或者脚本，已被广泛应用于互联网领域。

### 3.1.2 工作流程

网络爬虫的基本工作流程如下：

1. 选取想爬取的网页的URL。
2. 将这些URL放入待爬取URL队列。
3. 根据待爬取队列中的URL获取对应的网页。
4. 分析获取的网页，解析DNS，从中得到数据以及新的URL，并将已经抓取的URL放入已爬取URL队列。
5. 将新的URL放入待爬取URL队列，数据输出，存入本地下载库中。

具体流程图如下：

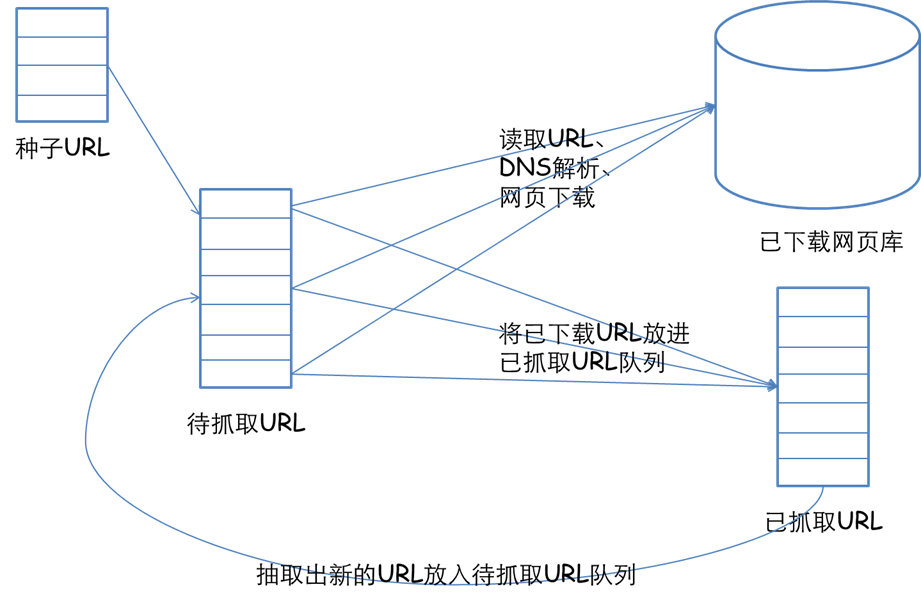


图3-1 爬虫工作流程

### 3.1.3 抓取策略介绍

在爬虫应用中，待爬取URL队列是很重要的一部分。待爬取队列中的URL以怎样的顺序爬取同样很重要，这决定这数据的获取顺序。决定待爬取队列中的URL以怎样的顺序爬取，这就是抓取策略。下面介绍几种常见的抓取策略：

1. 广度优先遍历策略：

广度优先遍历策略是指在抓取过程中，在完成与当前结点直接链接的结点的抓取后，才选择其中一个结点继续向下一层抓取。这种方法的思想是认为在与初始URL在一定链接距离内的网页具有最大的主题相关性。这种方法的缺点是，随着抓取网页数量的增加，大量无关网页会被下载，造成不必要的浪费。

1. 深度优先遍历策略：

深度优先遍历策略是指在抓取开始，选择与当前结点相链接的一个结点，分析其URL，然后在选择其中一个进行抓取，这样重复下去，直到其中一条链路抓取完毕，然后再重新选择一个未抓取的URL进行抓取。显然这种方法的深度决定着抓取的命中率及效率。随着深度的增大，命中率及效率会成倍的下降。

1. 最佳优先遍历策略

最佳优先遍历算法根据一定的网页分析算法，算出网页与初始网页的相关性，相关性高的网页优先抓取，这样，爬虫程序只抓取那些根据计算得到的“有用的”网页，但是这样可能导致在抓取路径上一些相关网页被遗漏。

1. Partial PageRank策略

Partial PageRank策略借鉴了PageRank算法的思想，根据已经下载网页的URL以及待爬取URL队列，计算每个网页的PageRank值，根据值得大小进行抓取排序。为防止部分相关网页因PageRank值小而一直不能被抓取，一般抓取K次后重新计算PageRank值。因为一些新下载的网页没有PageRank值，所以将这个网页的所有入链传递的PageRank值汇总，就够成了这些页面的PageRank值，从而参与排序。

## 3.2 工具介绍

本爬虫的实现以及运行都是基于Eclipse的，Eclipse安装Pydev插件后即配置好Python语言的开发环境，在解析模块还使用了BeautifulSoup插件。下面介绍主要用到的这三种工具：

### 3.2.1 Eclipse

Eclipse是一个开源的、基于Java的可扩展开发平台。就其本身而言，它只是一个框架和一组服务，用于通过插件组件构建开发环境。本文通过安装Pydev插件构建Python语言的开发环境。

### 3.2.2 Python语言

Python是一种面向对象、解释型计算机程序设计语言，由Guido van Rossum于1989年发明，第一个公开发行版发行于1991年。Python的设计哲学是“优雅、明确、简单”。其特点是易读、易维护以及强制用空白符作为语句缩进，使得不好的编程习惯(例如if语句下一行不向右缩进)都不能通过编译。

Python具有丰富和强大的库，它常被称为胶水语言，能够把用其它语言制作的各种模块(尤其是C\C++)很轻松的联结在一起。需要注意的是在使用某些扩展类库的时候可能需要考虑平台问题，某些可能不提供跨平台的实现。Python具有良好的可扩展性。

Python还可以以交互模式运行，比如主流操作系统Linux、Mac、Windows都可以直接在命令模式下直接运行Python交互环境。直接下达操作指令即可实现交互操作。

Python目前已经公布了Python3.x版本，本爬虫使用Python27编写。

### 3.2.3 BeautifulSoup

BeautifulSoup是Python的一个插件，用于解析HTML和XML，是替代正则表达式的利器，本爬虫使用BeautifulSoup4。

BeautifulSoup会产生一个DOM树，尽可能和源文档的含义内容一致，这种措施通常能够满足搜集数据的需求。BeautifulSoup下载地址为：<http://www.crummy.com/software/BeautifulSoup> 安装过程如下：

(1)下载完成后解压缩。

(2)打开命令提示符，将工作目录切换到BS的目录下。

(3)运行setup.py build。

(4)运行setup.py install。

(5)输入Python，进入Python模块，然后输入from bs4 import BeautifulSoup检测是否成功。

## 3.3 实现

主要含有主调度函数、URL管理、HTML下载器、HTML解析器、HTML输出这五个模块。系统结构图如下：

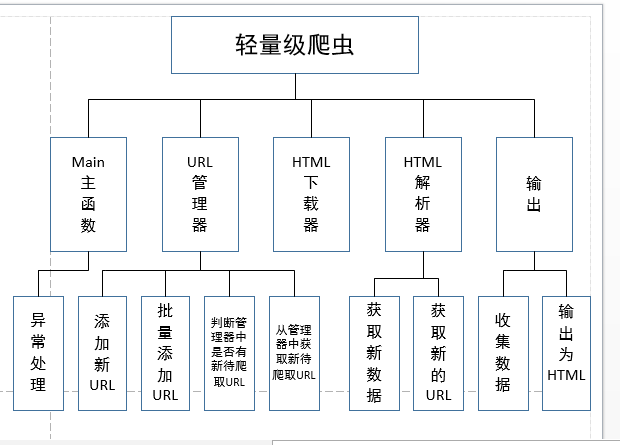


图3-2 爬虫功能结构图

下面是主要的调度函数：

from baike\_spider import url\_manager, html\_downloader, html\_parser,html\_outputer

class **SpiderMain**(object):

def **\_\_init\_\_**(*self*):

*self*.urls = url\_manager.UrlManager()

*self*.downloader = html\_downloader.HtmlDownloader()

*self*.parser = html\_parser.HtmlParser()

*self*.outputer = html\_outputer.HtmlOutputer()

def **craw**(*self*, root\_url):

count = 1

*self*.urls.add\_new\_url(root\_url)

while *self*.urls.has\_new\_url():

try:

new\_url = *self*.urls.get\_new\_url()

print *'craw %d : %s'* %(count,new\_url)

html\_cout = *self*.downloader.download(new\_url)

new\_urls, new\_data = *self*.parser.parse(new\_url,html\_cout)

*self*.urls.add\_new\_urls(new\_urls)

*self*.outputer.collect\_data(new\_data)

if count == 20:

break

count = count + 1

except:

print *'carw failed'*

*self*.outputer.output\_html()

if \_\_name\_\_==*"\_\_main\_\_"*:

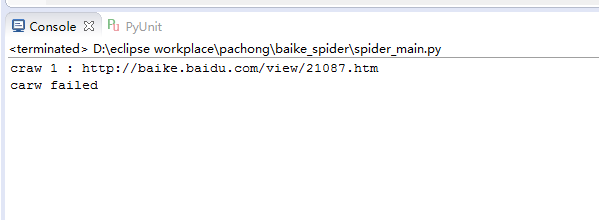
root\_url = *"http://baike.baidu.com/view/21087.htm"*

obj\_spider = SpiderMain()

obj\_spider.craw(root\_url)

## 3.4 运行结果

本次测试爬取百度百科中Python的20条相关词条。



由于时间以及技术有限，尚不能完美运行，爬虫在爬取到主页面后就会停止运行，错误还没有找到，以后会尽量完善。

4 算法部分

4.1获取数据的三种途径

### 4.1.1通过新浪微博模拟登录获取数据

模拟登录功能的实现方式是读取包含账号信息的配置文件，模拟网页登录新浪微博的过程，获取访问新浪微博页面时所需要的有效认证信息，即需要保存在本地的cookie信息。程序向新浪服务器发送经过加密的用户名(username)和密码(password)，服务器从传递的URL参数中提取字符串并解密得到原用户名和密码。

其中对username和password的加密是模拟登录过程中的关键步骤。对username进行Base64编码得到用户名的加密结果。但password的加密过程比较复杂。首先利用HttpClient对象访问新浪服务器获取服务器时间(servertime)、一个随机生成的字符串(nonce)2个参数。然后利用新浪服务器给出的pubkey和rsakv值创建RSA算法公钥(key)。将servertime，nonce和password按序拼接成新的字符串message，使用key对message进行RSA加密并将加密结果转化为十六进制，得到password的加密结果。将加密后的用户名和密码一起作为请求通行证的URL请求的报头信息传递给新浪服务器，新浪服务器经过验证无误后，返回登录成功信息，HttpClient保存有效的Cookies值。

模拟登录成功后，程序向服务端请求数据获取任务，否则结束本次任务。

### 4.1.2 通过调用微博API接口获取用户微博数据

利用微博系统的SDK获取数据的工作流程，调用API需要通过用户身份的鉴权，目前使用OAuth鉴权。其优点是数据准确快捷，很少因微博平台的变化而变化，但缺点是接口提供商限制的因素多且不稳定。

(1)新浪微博开放平台

目前，新浪微博提供了相应的API供本系统使用，通过调用这些API，本系统可以获取如微博内容、用户关系、用户信息、用户标签等信息。要使用这些API，需要先创建一个自己的应用，然后进行认证，认证完成后即可使用API获取新浪微博的数据了。

但是现在新浪微博加大了对API接口的限制，只有公司用户才可以申请微服务应用(里面有访问数据的接口)，这对个人数据采集的实践造成较大困难。

(2)微博JAVA SDK

微博JAVA SDK是新浪为JAVA开发者提供的专门用于访问新浪微博的接口，通过微博JAVA SDK，第三方微博应用可以很方便的访问微博提供的API接口，获取到需要的信息。微博JAVA SDK的整体架构包含如下几个：

src:weibo4j封装了大部分接口，包括授权，timeline，用户等常见接口 ；

weibo4j.http包含了https请求的封装；

weibo4j.model包含了user，status等实体类 ；

weibo4j.org.json 完成json解析的各种方法；

weibo4j.util 包含了请求api接口时候所需要的工具类；

examples：里面包含了封装接口的各个测试。

(3)微博应用创建

首先使用自己的新浪微博账号进行登陆，如果没有的话，就要先注册，然后再登陆到新浪的微博开放平台。成功登陆以后，首先要填写个人的基本信息，填写了个人信息后，就可以获得开发者的身份，这时就可以创建应用了。

创建应用时，除了填写应用的基本信息，包括名称、用途、应用网址以外，最重要的是应用的类型。站内应用可以获得更高的权限和更高的API调用频率。

创建完站内应用后，应用将会获得一个App Key和一个Secret Key，这两个Key是以后会用到的。其中App Key是应用在开放平台中的身份标志，每次应用要调用API时，都需要提供App Key和使用人账号，这样方便新浪统计某个应用的使用频率，这样既能保证某个应用不能频繁调用API，避免出现恶意攻击新浪的事件，另一方面，新浪也可以通过使用频率来进行热门应用的推荐。而App Secret是新浪微博分配给某个应用的密码，这是为了保证其真实性，这个是唯一的，这样可以防止其他应用冒充某个热门应用。

(4)微博内容获取

总体的大致流程如图4-1。



图4-1 通过微博API接口获取数据大致流程

在采集过程中，首先需要的是获取新浪微博的身份验证，即申请Access Token码，从而获取对新浪微博开发的权限。进行身份验证以后，就可以开始对新浪微博进行开发了。在调用API接口Timeline获得授权用户所有的微博信息以后，就可以通过输入关键词对微博信息进行关键词匹配，如果匹配成功，就将微博信息中的id和微博内容显示在框架并以txt文件的形式保存下来。由于新浪微博对API接口调用的次数和频率有限制，一旦达到限制， 系统将等待几分钟后才能进行采集。

4.2基于用户的协同过滤算法

### 4.2.1集体智慧和协同过滤

集体智慧是指在大量的人群的行为和数据中收集答案，帮助你对整个人群得到统计意义上的结论，这些结论是我们在单个个体上无法得到的，它往往是某种趋势或者人群中共性的部分。

协同过滤是利用集体智慧的一个典型方法。要理解什么是协同过滤 (Collaborative Filtering, 简称 CF)，首先想一个简单的问题，如果你现在想看个电影，但你不知道具体看哪部，你会怎么做？大部分的人会问问周围的朋友，看看最近有什么好看的电影推荐， 而我们一般更倾向于从口味比较类似的朋友那里得到推荐。这就是协同过滤的核心思想。

协同过滤一般是在海量的用户中发掘出一小部分和你品位比较类似的，在协同过滤中，这些用户成为邻居，然后根据他们喜欢的其他东西组织成一个排序的目录作为推荐给你。

如图4-2所示，基于用户的协同过滤算法示意图。

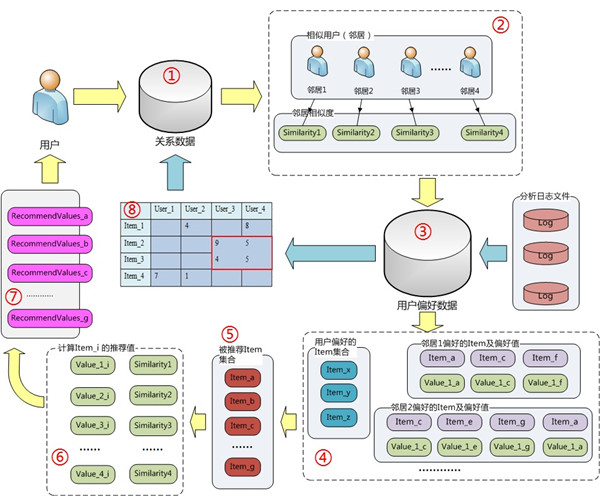


图4-2 基于用户的协同过滤算法示意图

### 4.2.2深入协同过滤核心

首先，要实现协同过滤，需要一下几个步骤：收集用户偏好；找到相似的用户或物品；计算推荐。

(1)收集用户偏好

从用户的行为和偏好中发现规律，并基于此进行推荐，所以如何收集用户的偏好信息成为系统推荐效果最基础的决定因素。用户有很多种方式向系统提供自己的偏好信息，比如：评分，投票，转发，点赞，评论，收藏，点击流，页面停留时间等等。

用户行为都会多于一种，那么如何组合这些不同的用户行为呢 ？基本上有如下两种方式。

(1)将不同的行为分组

一般可以分为点赞和评论、转发，然后基于不同的用户行为，计算不同用户或者物品的相似度。

(2)不同行为产生的用户喜好对它们进行加权

对不同行为产生的用户喜好进行加权，然后求出用户对物品的总体喜好。

好了，当我们收集好用户的行为数据后，还要对数据进行预处理，最核心的工作就是减噪和归一化。

减噪： 因为用户数据在使用过程中可能存在大量噪音和误操作，所以需要过滤掉这些噪音。

归一化：不同行为数据的取值相差可能很好，例如用户的查看数据肯定比购买数据大得多。通过归一化，才能使数据更加准确。

(2)找到相似的用户或物品

当已经对用户行为进行分析得到用户喜好后，我们可以根据用户喜好计算相似用户和物品，然后基于相似用户或者物品进行推荐。

关于相似度的计算，现有的几种基本方法都是基于向量(Vector)的，其实也就是计算两个向量的距离，距离越近相似度越大。在推荐的场景中，在用户 - 物品偏好的二维矩阵中，我们可以将一个用户对所有物品的偏好作为一个向量来计算用户之间的相似度，或者将所有用户对某个物品的偏好作为一个向量来计算物品之间的相似度。下面我们详细介绍几种常用的相似度计算方法：

欧几里德距离(Euclidean Distance)

最初用于计算欧几里德空间中两个点的距离，假设 x，y 是 n 维空间的两个点，它们之间的欧几里德距离是：



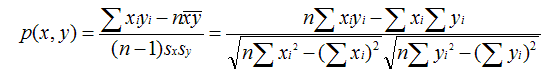
可以看出，当 n=2 时，欧几里德距离就是平面上两个点的距离。

当用欧几里德距离表示相似度，一般采用以下公式进行转换：距离越小，相似度越大。



皮尔逊相关系数(Pearson Correlation Coefficient)

皮尔逊相关系数一般用于计算两个定距变量间联系的紧密程度，它的取值在 [-1，+1] 之间。当取值大于0时表示两个变量是正相关的，即一个变量的值越大，另一个变量的值也会越大；当取值小于0时表示两个变量是负相关的，即一个变量的值越大，另一个变量的值反而会越小。



sx, sy是 x 和 y 的样品标准偏差。

### 4.3算法实现

/\*\*

\* @return double

\* @param pm1

\* @param pm2

\* @Description获取两个用户之间的皮尔逊相似度,相关系数的绝对值越大,相关度越大

\*/

public static double getUserSimilar(Map<String, Integer> pm1, Map<String, Integer> pm2) {

int n = 0;// 数量n

int sxy = 0;// Σxy=x1\*y1+x2\*y2+....xn\*yn

int sx = 0;// Σx=x1+x2+....xn

int sy = 0;// Σy=y1+y2+...yn

int sx2 = 0;// Σx2=(x1)2+(x2)2+....(xn)2

int sy2 = 0;// Σy2=(y1)2+(y2)2+....(yn)2

for (Entry<String, Integer> pme : pm1.entrySet()) {

String key = pme.getKey();

Integer x = pme.getValue();

Integer y = pm2.get(key);

if (x != null && y != null) {

n++;

sxy += x \* y;

sx += x;

sy += y;

sx2 += Math.pow(x, 2);

sy2 += Math.pow(y, 2);

}

}

// p=(Σxy-Σx\*Σy/n)/Math.sqrt((Σx2-(Σx)2/n)(Σy2-(Σy)2/n));

double sd = sxy - sx \* sy / n;

double sm = Math.sqrt((sx2 - Math.pow(sx, 2) / n) \* (sy2 - Math.pow(sy, 2) / n));

return Math.abs(sm == 0 ? 1 : sd / sm);

}

/\*\*

\* @return double

\* @param pm1

\* @param pm2

\* @Description获取两个用户之间的欧几里得距离,距离越小越好

\*/

public static double getEuclidDistance(Map<String, Integer> pm1, Map<String, Integer> pm2) {

double totalscore = 0.0;

for (Entry<String, Integer> test : pm1.entrySet()) {

String key = test.getKey();

Integer a1 = pm1.get(key);

Integer b1 = pm2.get(key);

if (a1 != null && b1 != null) {

double a = Math.pow(a1 - b1, 2);

totalscore += Math.abs(a);

}

}

return Math.sqrt(totalscore);

}

/\*\*

\* @return String

\* @param simUserObjMap

\* @param simUserSimMap

\*@Description根据相关系数得到推荐物品

\*/

public static String getRecommend(Map<String, Map<String, Integer>> simUserObjMap, Map<String, Double> simUserSimMap) {

Map<String, Double> objScoreMap = new HashMap<String, Double>();

for (Entry<String, Map<String, Integer>> simUserEn : simUserObjMap.entrySet()) {

String user = simUserEn.getKey();

double sim = simUserSimMap.get(user);

for (Entry<String, Integer> simObjEn : simUserEn.getValue().entrySet()) {

double objScore = sim \* simObjEn.getValue();

String objName = simObjEn.getKey();

if (objScoreMap.get(objName) == null) {

objScoreMap.put(objName, objScore);

}else {

double totalScore = objScoreMap.get(objName);

objScoreMap.put(objName, totalScore + objScore);

}

}

}

List<Entry<String, Double>> enList = new ArrayList<Entry<String, Double>>(objScoreMap.entrySet());

Collections.sort(enList, new Comparator<Map.Entry<String, Double>>() {

public int compare(Map.Entry<String, Double> o1, Map.Entry<String, Double> o2) {

Double a = o1.getValue() - o2.getValue();

if (a == 0) {

return 0;

}

else if (a > 0)

{

return 1;

}

Else

{

return -1;

}

}

});

return enList.get(enList.size() - 1).getKey();

}

}

简单测试结果：

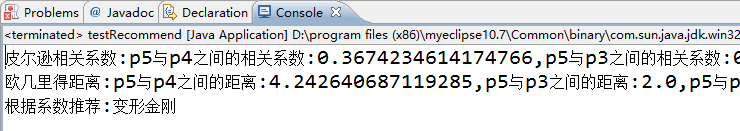


图4-3 结果示意图

结论

这一次我们的题目是新课题，对于本科生的我们真的难度很大。社交网络是当今互联网研究前沿。

最新技术一直是专家学者保持着。网上的资料很少。而我们又没有这方面的基础。因此对我们做这次课程设计造成很多困难。

不过，通过这次课程设计实践，我们了解了网络爬虫的原理流程，细致研究了基于用户的协同过滤算法，让我们对社交网络数据采集又有了新的认识。

数据挖掘是大数据时代最常提及的。若我们以后从事这方面工作，这次课程设计为我们提供了良好的开端。

参考文献

[1] 黄延炜，刘嘉勇.新浪微博数据获取技术研究[J].信息安全与通信保密,2013,(6);71-73.

[2] 邓夏玮.基于社交网络的用户行为研究[D].北京;北京交通大学,2012.

[3] Mark Lutz.Python学习手册(第3版),2009.

[4] 罗刚.自己动手写网络爬虫,2010.