文本复制检测报告单(全文标明引文)

№:ADBD2018R_201805301531272018053015484144017418805 6 检测时间:2018-05-30 15:48:41

检测文献: 53140821_孟政元_计算机科学与技术_网易云音乐用户数据的分析和可视化技术研究与实现

作者: 孟政元

检测范围: 中国学术期刊网络出版总库

中国博士学位论文全文数据库/中国优秀硕士学位论文全文数据库

中国重要会议论文全文数据库

中国重要报纸全文数据库

中国专利全文数据库

图书资源

优先出版文献库

大学生论文联合比对库

互联网资源(包含贴吧等论坛资源)

英文数据库(涵盖期刊、博硕、会议的英文数据以及德国Springer、英国Taylor&Francis 期刊数据库等)

港澳台学术文献库

互联网文档资源

CNKI大成编客-原创作品库

个人比对库

时间范围: 1900-01-01至2018-05-30

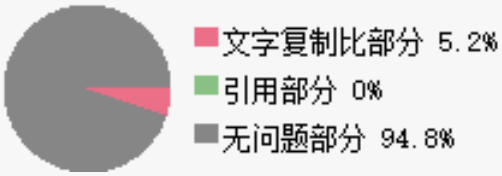
检测结果

总文字复制比：5.2% 跨语言检测结果：0%

去除引用文献复制比：5.2% 去除本人已发表文献复制比：5.2%

单篇最大文字复制比：4.4%（HTTP 代理原理及实现（一） - yokasou的博客 - CSDN博客）

重复字数：	[1149]	总段落数：	[8]
总字数：	[22217]	疑似段落数：	[1]
单篇最大重复字数：	[972]	前部重合字数：	[0]
疑似段落最大重合字数：	[1149]	后部重合字数：	[1149]
疑似段落最小重合字数：	[1149]		



指 标：☐疑似剽窃观点 ☑疑似剽窃文字表述 ☐疑似自我剽窃 ☐疑似整体剽窃 ☐过度引用

表格： 0 公式： 0 疑似文字的图片： 0 脚注与尾注：0

0%（0）	中英文摘要等（总2954字）
0%（0）	第1章绪论（总560字）
0%（0）	第2章数据可视化概述（总3346字）
17.2%（1149）	第3章数据的爬取（总6687字）
0%（0）	第4章数据的结构（总1113字）
0%（0）	第5章数据的存储（总904字）
0%（0）	第6章数据的分析与可视化（总6116字）
0%（0）	第7章结论（总537字）

（注释：无问题部分 文字复制比部分 引用部分）

1. 中英文摘要等

总字数：2954

相似文献列表 文字复制比：0%(0) 疑似剽窃观点：(0)

原文内容 红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

吉林大学学士学位论文（设计）承诺书

本人郑重承诺：所呈交的学士学位毕业论文（设计），是本人在指导教师的指导下，独立进行实验、设计、调研等工作

基础上取得的成果。除文中已经注明引用的内容外，本论文（设计）不包含任何其他个人或集体已经发表或撰写的作品成果。对本人实验或设计中做出重要贡献的个人或集体，均已在文中以明确的方式注明。本人完全意识到本承诺书的法律结果由本人承担。

学士学位论文（设计）作者签名：

2018年5月20日

摘要

网易云音乐是由网易（中国）开发的一款音乐软件，拥有着极其庞大的用于群体。网易云音乐的一大特色和吸引用户的地方是有着比较温和的听众群体，有许多的用户在一些经典的歌曲下面留下了很多能够引起共鸣的评论和小故事，这些东西也会吸引用户之间的相互关注，同时增强了网易云音乐的用户粘性。歌曲的评论数也同时成为了歌曲受欢迎程度的一个参考指标。同时，网易云音乐根据用户的听歌记录，借助于强大的推荐系统，为用户提供了每日歌单的功能，所推荐的歌曲也有很大的可能符合用户的口味。并且，网易云音乐为用户提供了一定的社交功能，用户拥有这自己的个人信息，这些信息从某种程度上来说可以作为用户的个体特征来进行分类和分析。比如，通过分析用户的年龄信息，可以知道用户群体在年龄这个维度上的分布情况；通过分析用户所在地址，可以知道用户群体在地理位置上的分布情况等等。通过这些分析，再结合数据可视化的技术，可以较为清晰的展示和观察用户群体的特征和分布。在整个用户群体中, 存在着许多的关联关系. 这些用户可以随意的互相关注。用户之间相互关注的情况可以用一个很大的有向图来表示, 用户是有向图中的点, 每一个用户对另外一个用户的关注就是有向图的一条边. 这篇论文对部分用户信息进行了采集,分析了用户的个体特征, 用户之间相互关注, 用户之间间接性的相互关联(基于floyd算法)等等在地域上的分布情况，最后进行可视化展现, 使得结果直观明了.

网易云音乐用户数据的分析和可视化技术研究与实现

关键字：数据可视化，网络爬虫，网易云音乐，D3.js

Abstract

Netease Cloud Music is a music software developed by NetEase (China), which has a very large group for use. One of the major features of Netease Cloud Music and the place where it attracts users is a relatively moderate audience group. Many users have left behind a number of classic songs that have attracted a lot of resonating comments and stories. These things will also attract users. The mutual attention between them has enhanced the user stickiness of Netease Cloud Music at the same time. The number of reviews of songs also became a reference indicator of song popularity. At the same time, Netease Cloud Music, based on the user's song recording, uses a powerful recommendation system to provide users with daily song list functions, and the recommended songs are also likely to meet the user's taste. In addition, NetEase Cloud Music provides users with certain social functions. Users own this personal information. To some extent, these information can be categorized and analyzed as the individual characteristics of users. For example, by analyzing the age information of the user, the distribution of the user group in the age can be known. By analyzing the address of the user, the geographical distribution of the user group can be known. Through these analysis, combined with the data visualization technology, you can clearly demonstrate and observe the characteristics and distribution of user groups. In the entire user group, there are many associations. These users can freely pay attention to each other. The situation of mutual concern between users can be represented by a large directed graph, where the user is a point in the directed graph, and the attention of each user to another user is an edge of the directed graph. The user information is collected, the user's individual characteristics are analyzed, the users are concerned with each other, and the indirect correlation between the users (based on the floyd algorithm) and the like are distributed in the region, and finally the visualization is performed, so that the result is intuitive and clear.

Research and Implementation of Analysis and Visualization Technology of NetEase Cloud Music User Data

Keywords：Data Visualization ， Network Spider ， Netease Cloud Music ，D3.js

目录

2. 第1章绪论

总字数：560

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第1章绪论

1.1 研究的意义

数据可视化是当前与大数据相关的一个重要研究方向，在数据的分析与价值获取中具有重要的意义[1]。数据可视化主要是将纷繁复杂的各种数据转化为各种简洁明了的形式，这些形式可以是图，动画等等。通过数据可视化，数据的可观察性或得了提高，不再是混乱无章的[2]。本文结合网络爬虫技术，通过获取网易云音乐的用户数据，结合当下热门的D3.js数据可视化技术，研究了网易云音乐的用户的各种特征，通过数据可视化来进行各种维度上数据特征的展示与分析。

1.2 内容安排

本文按照数据的处理流程进行行文的安排，后面的章节在前面章节的基础上展开，保证文章的结构严谨，清晰易懂。

第二章简介了本文中要用到的数据可视化结束和D3.js的相关内容及现状。

第三章讲述数据的获取流程，详细说明了如何编写和使用爬虫获取需要的数据。

第四章讲述了数据的结构，便于理解下面将要对数据进行的分析和可视化操作。

第五章讲述了数据的存储方式以及如何对将数据从数据库中取出以便查询和分析。

第六章在前面章节的基础上详细讲述了对数据进行的分析和可视化，将数据从杂乱无章的状态转换成各种简洁明了的图表，并对数据的可视化结果做了简单的分析和比对。

第七章是本文的结论，对本文所做的工作进行总结，给出了明确的分析和结论。

3. 第2章数据可视化概述

总字数：3346

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第2章数据可视化概述

2.1 概述

在近几年的时间内，大数据相关的技术，研究，应用如雨后春笋版出现，数据的价值的挖掘也被提升到了前所未有的高

度[3]。在对过去的各种活动中积累下来的数据的分析过程中，为了更直观的展现出数据的特征，数据之间的关联，数据的潜在价值，数据可视化技术的作用显得越来越重要[4]。通过数据的可视化，数据从杂乱无章的数字和符号变成了直观而又明了的图像，视频等非常容易理解和分析的形式[5]。从这个角度来说，数据可视化在数据的分析和价值的体现过程中起着至关重要的作用[6]。

2.2 基于D3的数据可视化技术

D3代表3个D，分别是Data，Driven，Document(数据驱动文档)。其中的数据和代表即将进行可视化的数据[7]。这里的数据不是原始的数据，而是经过了采集，过滤，分析，整理之后，已经准备好进行可视化的数据[8]。驱动的意思是采用数据来操纵文档，之所以说是数据驱动文档，是因为在D3进行可视化的过程中，数据并不是直接转换成各种图表的，而是通过各种布局方法，对数据的变化进行监听，然后在数据发生变化的时候控制相应的视图进行相应的变化[9]。这个思想与现在前端非常流行的MVC框架异曲同工[10]。同时，D3的代码通过在动作函数结尾返回操作集合，因而可以进行链式操作，使得代码非常的直观明了，可读性非常高[11]。D3的3.x版本采用SVG来作为数据可视化的视图工具，具有非常优异的特点，比如支持XML标准，可以在上面应用CSS样式进行外观的设置和美化，具有完整的DOM事件机制，这些都有利于在可视化的过程中对视图进行操作和修改，同时也给D3在进行可视化过程中的交互提供了极大的便利[12]。

2.3 基于地图可视化在D3中的实现的数据可视化技术

2.3.1 更好的效果

基于地图的可视化能够带来很好的视觉效果, 尤其适合宽屏展示, 很适合人多的热闹气氛[13]。 地图的可视化还能够带给人"高科技感", "未来感"[14]。在本文中,为了更好的展示用户的各个特征在地理空间上的分布情况, 我们借助于D3, 使用基于地图的可视化技术来更加直观明了而又清晰的来展示[15]。

2.3.2 地图数据基于GeoJSON格式

GeoJSON是一种非常简洁的数据格式，一般用于描述地理空间之类的数据信息[16]。但是，GeoJSON并不是一种陌生而全新的数据表达方式，而是在JSON的格式规范上发展而来的，在JSON格式的基础上进行了一些便于表达地理空间信息的增强[17]。GeoJSON是一个普通的JavaScript对象，其约束与普通的JavaScript对象完全相同，但同时又有着一些特有的规范[18]。

GeoJSON是一个普通的对象，这个对象是整个GeoJSON的最外层结构，内部可以无限制嵌套各种用来表达不同个体的结构。这个对象可表示:

- 几何体(Geometry)
- 特征(Feature)
- 特征集合(FeatureCollection)

GeoJSON的最外层结构（即一个普通的JavaScript对象）的内部可以包含很多的子对象, 每一个GeoJSON对象都拥有着一个特殊的type属性, 表示对象所要表示的数据的的类型, type只能是下面之一[19]:

- Point: 点
- MultiPoint: 多点
- LineString: 线
- MultiLineString: 多线
- Polygon: 面
- MultiPolygon: 多面
- GeometryCollection: 几何体集合
- Feature: 特征
- FeatureCollection: 特征集合

举例如下:

点对象:

```
{
"type": "Point",
"coordinates": [-15, 139]
}
```

线对象:

```
{
"type": "LineString",
"coordinates": [[-10, 139], [-17, 138]]
}
```

面对象:

```
{
"type": "Polygon",
```

"coordinates": [[[300, 0], [301, 30], [301, 45], [300, 45], [30, 50]]]

}

2.3.3 借助于地图进行可视化的具体过程

上面我们介绍了用来表示地图具体数据的GeoJSON格式，这里，我们详细说明一下如何将地图数据生成可见的地图，并在生成的地图的基础上进行数据的可视化展示[20]。

由于我们的可视化是采用D3.js进行的，所以需要在网页中实现。

页面的基本结构如下所示：

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Title</title>
</head>
<body>
<svg></svg>
<script>
// js code goes here...
</script>
</body>
</html>
```

首先，在网页中引入D3.js库：

```
<script src="d3.min.js"></script>
```

然后，在body标签的script标签内编写相应的代码进行地图的生成。

```
const svg = d3.select('svg')
const width = svg[0][0].clientWidth
const height = svg[0][0].clientHeight
```

这里，我们通过D3选中页面中的svg元素，获取它的高度和宽度以便进行地图绘制的时候正确的对地图进行缩放。然后，我们通过D3提供插值方法创建一个颜色比例尺（Scale），用不同深浅的颜色来进行不同区域所代表的数值的表示。创建颜色比例尺的方式如下：

```
const white = d3.rgb('#fff')
const red = d3.rgb('#f00')
const interpolate = d3.interpolate(white, red)
```

创建完颜色比例尺之后，我们通过D3提供的布局（Layout）方法，来将地图数据转换成便于绘图的数据。D3不同于其他图表库的地方就在于D3没有提供直接生成图表的任何方法，而是将数据转换成各种便于绘图的数据，比如弧长，角度，距离等等，这个过程就是所谓的布局。这种做法的确是增大了绘图的难度和复杂度，对使用者的要求更高，但是却提供了无与伦比的灵活性和自由[21]。通过将数据转换成便于绘图的数据，使用者可以自由的选择绘图的具体实现方法，无论是用svg还是用canvas都可以，而且可以随心所欲的进行定制，根据自己的需要选择最合适自己的绘图方法。

生成布局的代码如下：

```
const projection = d3.geo.mercator()
.center([107, 31])
.scale(700)
.translate([width/1.8, height/1.6]);
const path = d3.geo.path()
.projection(projection)
```

上面的代码中有连续的函数调用，这种调用方式在JS中一般称为链式调用，由于JS是弱类型动态化的语言，所以函数可以返回任意的类型。通过使得函数返回this，从而使得可以在函数的返回值的基础方继续调用该对象的方法就可以形成上面的写法。这种写法的好处是代码执行效率高，并且非常的清晰，可读性高。

```
svg.selectAll("path")
.data( root.features )
.enter()
.append("path")
.attr("d", path ) //使用地理路径生成器
```

上面的代码中，通过选中Enter集合，将缺少的元素补上（append），然后设置元素的路径（d）属性为path函数的返回

值，就可以通过path这个地理路径生成器完成路径的生成，正确的将地图绘制出来了。

第3章数据的爬取

3.1 原理

通过请求伪造，伪造与正常请求相同的请求头, 调用官方 API. 爬虫通过提供与正常请求相同的参数, 通过post方法向网易云的后台发起请求, 从而获取相应的数据.

通过抓包来分析网易云音乐windows客户端与服务器端的通信方式，从而获取调用官方后台API的具体方式。这样，可以直接通过调用官方的后台API来获取想要的数据。这种爬取数据的方法与传统的爬虫有所不同。传统的爬虫是通过爬取完整的网页，然后从网页的DOM结构中提取出所需要的信息，这种爬取信息的方式爬取的信息非常的多，适合于搜索引擎进行网页的索引和归类，但是并不适合我们的需求。我们需要的信息只是与用户的特征信息相关的那一小部分信息，爬取完整的网页会造成非常大的资源浪费，而且也会减慢爬取的速度。这种直接调用官方后台API爬取数据的方式，可以最大程度的避免爬取不必要的信息，而且可以大幅度加快爬取数据的速度。

3.2 调用官方后台API的具体方法

3.2.1. 访问方式

网易云音乐的后台api的url为`http://music.163.com/weapi/${api_name}?csrf_token=${token}, 通过发起http请求即可访问网易云音乐的后台api，但是后台的api需要加上正确的参数并对参数进行相应的处理，否则会被网易云音乐的后台判定为非法请求，直接屏蔽掉。

3.2.2. 参数的处理

上面已经提到，在对网易云音乐的后台发起请求的时候，必须带上相应的参数并进行正确的处理，否则请求会被当做非法的而自己屏蔽掉。参数的具体处理方法如下(以获取歌曲的评论为例)：

参数名参数作用解释

csrf_token csrf(cross-site request forgery)即跨站请求伪造，这个参数主要是解决一些基于web的攻击，在请求的时候可以设为空字符串（假设客户端之前没有访问过网易云音乐的网站），但不能没有这个参数。

offset 偏移值，在某些api的返回结果中，可能会含有非常长的数据集合，如果全部返回到客户端的话会消耗太多的带宽，导致客户端的卡顿，因此可以设置偏移，从offset说指定的位置开始返回集合中的数据。这个参数需要结合第四个参数limit一起来控制返回的数据。

id 数据的id编号，全局唯一。

limit 在返回的数据集合过长而没有必要全部返回时，可以通过limit参数限制返回的数据的数量，结合offset参数使用可以更加精确的控制返回的结果。比如offset = 200，limit = 40，表示返回从200开始的40个数据。

网易云音乐的后台请求一律采取的是post方法，请求数据放在body部分。将这些参数组成一个对象，如下所示，作为未加密处理的body部分。

const body =

```
{
offset: req.query.offset || 0,
id: ri,
limit: req.query.limit || 20,
csrf_token: ""
};
```

接下来通过相应的方法对请求进行加密处理：用一个16位长度的随机字符串作为加密的秘钥，然后按照下面代码所示的方法进行加密，最后返回一个对象，该对象具有两个属性：params是加密后的数据，encSecKey代表加密用的秘钥，用于服务器端对加密后的数据进行解密。

```
function Encrypt(obj)
{
const text = JSON.stringify(obj)
const secKey = createSecretKey(16)
const encText = aesEncrypt(aesEncrypt(text, nonce), secKey)
const encSecKey = rsaEncrypt(secKey, pubKey, modulus)
return {
params: encText,
encSecKey: encSecKey
}
}
```

然后通过post方法发起请求：url为该API对应的url，请求方法为post方法，body的内容就是上面返回的对象经过querystring序列化之后的返回结果。如下所示(注意：为了直观明了，代码有所删减，不能直接运行)：

```
function createWebAPIRequest
(
data,
callback,
errorcallback
)
{
const cryptoreq = Encrypt(data);
const options =
{
body: querystring.stringify
(
{
params: cryptoreq.params,
encSecKey: cryptoreq.encSecKey
}
),
proxy: proxy
};
}
```

Request函数是对功能的进一步封装，对异常进行了自动的处理。在请求成功时，会自动调用回调函数对数据进行进一步的处理，在请求失败时会自动打印出错相关的信息，然后调用请求失败对应的回调函数，以便对当前的异常和错误进行恰当的处理。通过request函数的封装，可以在其他地方减少重复的代码结构，使得其他地方的异常处理更加的简洁，增强可读性。

```
request(options, function(error, res, body)
{
if (error)
{
console.error(error);
errorcallback(error);
}
else
```

{
callback(body, cookie);
}
});

createWebAPIRequest函数是整个爬虫http请求部分的最高层次的封装，直接以必要的参数调用该函数即可得到从服务器端返回的结果。函数的内部会自动对请求参数进行相应的处理，在发生异常时自动进行相应的函数调用和错误打印，在请求成功时对结果进行相应的反序列化以便后续的访问和进一步的处理。

```
createWebAPIRequest
(
"music.163.com",
`/weapi/v1/resource/comments/R_SO_4_${rid}/?csrf_token=`,
"POST",
data,
cookie,
music_req =>
{
res.send(music_req);
},
err => res.status(502).send(err.message)
)
```

注意：代码中用到了ES6标准中的模板字符串和箭头函数。

1. 模板字符串使用反引号 (` `) 来代替普通字符串中的用双引号和单引号。模板字符串可以包含特定语法(${expression})的占位符。占位符中的表达式和周围的文本会一起传递给一个默认函数，该函数负责将所有的部分连接起来，如果一个模板字符串由表达式开头，则该字符串被称为带标签的模板字符串，该表达式通常是一个函数，它会在模板字符串处理后被调用，在输出最终结果前，你都可以通过该函数来对模板字符串进行操作处理。在模版字符串内使用反引号（`）时，需要在它前面加转义符（\）。

2. 箭头函数表达式的语法比函数表达式更短，并且不绑定自己的this，arguments，super或 new.target。这些函数表达式最适合用于非方法函数，并且它们不能用作构造函数。引入箭头函数有两个方面的作用：更简短的函数并且不绑定this。在箭头函数出现之前，每个新定义的函数都有它自己的 this值（在构造函数的情况下是一个新对象，在严格模式的函数调用中为 undefined，如果该函数被称为“对象方法”则为基础对象等）。This被证明是令人厌烦的面向对象风格的编程。箭头函数不会创建自己的this；它使用封闭执行上下文的this值。

3.2.3. 结果

通过上面所述的请求方法，可以比较方便的通过调用网易云音乐后台API的方式进行数据的采集，免去了通过爬取网页来提取内容的麻烦，可以让爬虫以更快的速度进行数据的采集。但是，由于网易云音乐的后台设有严格的反爬虫机制，所以还要通过下面所说的方式来进行规避，防止爬虫被屏蔽。

3.3反爬虫策略

说道爬虫，就不得不提许多的大网站为了防范恶意用户对网站数据的爬取。一般来说，只要是不影响网站的正常功能的爬虫，不恶意盗取用户数据或者恶意刷东西的爬虫，网站的反爬虫机制并不会对爬虫进行封杀。许多的搜索引擎就是靠爬取网站数据来进行网站的索引，但是搜索引擎会严格按照网站的robots.txt中所规定的规则来进行爬取，以免爬取到敏感的或者网站不想要被索引的页面。

由于网易云的后台设置有反爬虫的机制, 所以不能爬取的太快, 否则IP会被封掉. 但是, 由于做分析需要的数据量较大,爬取的速度太慢无法再短时间内怕取到足够的数据. 因此必须采取一定的策略加快数据爬取的速度. 根据网易云的反爬虫机制, 单个IP地址一分钟最多发送20个请求. 因此, 这里采用多个代理并行爬取的方式. 同时, 在http请求头中, 随机化user-agent, 更好的把爬虫伪装成正常用户.

3.4代理程序

在搭建代理的时候, 由于我们的代理是为了隐藏爬虫, 所以必须使用http正向代理中的高匿代理, 否则会被网易云的后台发现异常. 这里使用的是自己使用Node.js编写的一个简单的匿名代理.

代理的原理是客户端先将请求发送到代理服务器, 由代理服务器向目标服务器发起请求, 并将请求结果发回客户端.

代码如下:

```
const PROXY_PORT = 65535
```

这个常量规定了代理程序的运行端口，这里的端口是代理程序作为一个特殊的服务器程序所监听的端口。

```
const http = require('http')
const net = require('net')
```

```
const url = require('url')
```

这三行代码引入了Node.js中将要被下面的程序所用到的三个模块。http模块是Node.js对http常用功能的封装，可以快速的创建http服务器或者发送http请求。net模块是对网络套接字功能的封装，用来快速创建点对点的socket连接，也可以快速创建一个监听在某个网络地址和端口上的监听程序。url模块是一套常用的与url有关的工具集合，实现了常用的url编码，解码，转换，解析等功能。

Web 代理是一种存在于网络中间的实体，提供各式各样的功能。现代网络系统中，Web 代理无处不在。HTTP 代理存在两种形式，分别简单介绍如下：

第一种是 RFC 7230 - HTTP/1.1: Message Syntax and Routing（即修订后的 RFC 2616，HTTP/1.1 协议的第一部分）描述的普通代理。这种代理扮演的是「中间人」角色，对于连接到它的客户端来说，它是服务端；对于要连接的服务端来说，它是客户端。它就负责在两端之间来回传送 HTTP 报文。

用Node.js实现如下：

```
function request(cReq, cRes)
{
const pReq = http.request(options, function(pRes)
{
cRes.writeHead(pRes.statusCode, pRes.headers)
pRes.pipe(cRes);
})
cReq.pipe(pReq)
}
```

第二种是 Tunneling TCP based protocols through Web proxy servers（通过 Web 代理服务器用隧道方式传输基于 TCP 的协议）描述的隧道代理。它通过 HTTP 协议正文部分（Body）完成通讯，以 HTTP 的方式实现任意基于 TCP 的应用层协议代理。这种代理使用 HTTP 的 CONNECT 方法建立连接，但 CONNECT 最开始并不是 RFC 2616 - HTTP/1.1 的一部分，直到 2014 年发布的 HTTP/1.1 修订版中，才增加了对 CONNECT 及隧道代理的描述，详见 RFC 7231 - HTTP/1.1: Semantics and Content。实际上这种代理早就被广泛实现。

用Node.js实现如下：

```
function connect(cReq, cSock)
{
const pSock = net.connect(u.port, u.hostname, function()
{
cSock.write('HTTP/1.1 200 Connection Established\r\n\r\n');
pSock.pipe(cSock);
})
cSock.pipe(pSock);
}
```

基于上面的两代理方式，我们创建一个web服务器来将两种代理方式进行整合。首先创建一个http服务器，监听来自客户端的请求，当客户端发来request请求时，使用第一种代理方式代理客户端的请求；当客户端发来connect请求时，自动使用第二种代理方式代理客户端的请求。这样，就创建了一个完整的代理程序。

用Node.js实现如下：

```
http.createServer()
.on('request', request)
.on('connect', connect)
.on('error', function(err)
{
console.error(err)
})
.listen(PROXY_PORT, function()
{
console.log(`proxy server online: http://localhost:${PROXY_PORT}`)
});
```

为了保证程序运行的稳定性，我们使用PM2来控制代理程序的请求。PM2是一个应用非常广泛的进程管理器，用来对服务器端对程序进行管理，比如自动整理程序的日志，在程序异常崩溃退出后自动重启，自动的负载均衡等，开机启动等。在程序部署完成且已经设置为开机启动之后，我们将程序所在主机的硬盘做成镜像，然后部署到多个服务器上，组成代理服务器集

群，加快爬取数据的速度。

3.5 Node.js基于事件循环的异步非阻塞并发机制

由于需要爬取的数据量较大, 传统的单线程同步程序耗时非常长, 所以必须采取并发机制. 这里采用的是Node.js基于事件循环的异步非阻塞并发机制. 在使用并发机制的时候, 需要使用互斥锁来确保程序有序运行.

3.6 并发锁

程序中的并发锁一共有4把，分别控制了爬取数据的四个方面。

程序的4个并发锁分别分装在两个对象中，__processing和__processed。这两个对象内部各有四个属性，分别是follow，followed，playlist，detail。

程序运行中锁的值的变化如下：

首先，锁的初始值均为false。

当某个并发线程（这里及线面说提到的线程是不存在的，因为Node.js是单线程并发，这里只是为了方便解释，详细的并发机制详见3.5章节）开始处理某个用户时，会选择一个__processed为false，__processing为false的个体对象，这个个体对象当前没有被其他的线程所占据，也没有被处理过。并且，选择了这个对象之后要将该对象的__processing值设置为true，防止其他的并发线程选择处理这个个体。

当前个体被处理完成之后，负责处理这个个体的并发线程负责将__processing和__processed的值都设置为false，表示当前个体已经被处理过了，且没有任何一个线程正在处理这个个体。

通过这个简单的锁机制，可以保证多个线程在并发的过程中允许的进行数据的处理，避免出现多个线程处理一个个体或者某个个体被重复处理的情况。

3.7 异常处理

在程序的运行过程中，网络可能会出错导致有些个体的处理处于失败的状态。这时候需要将锁进行手动的处理，进行一些修改，然后重新启动处理程序，对这些状态异常的个体重新进行处理。

指　标
疑似剽窃文字表述

1. 箭头函数表达式的语法比函数表达式更短，并且不绑定自己的this，arguments，super或 new.target。这些函数表达式最适合用于非方法函数，并且它们不能用作构造函数。引入箭头函数
2. Web 代理是一种存在于网络中间的实体，提供各式各样的功能。现代网络系统中，Web 代理无处不在。
3. 这种代理扮演的是「中间人」角色，对于连接到它的客户端来说，它是服务端；对于要连接的服务端来说，它是客户端。它就负责在两端之间来回传送 HTTP 报文。
4. 这种代理使用 HTTP 的 CONNECT 方法建立连接，但 CONNECT 最开始并不是 RFC 2616 - HTTP/1.1 的一部分，直到 2014 年发布的 HTTP/1.1 修订版中，才增加了对 CONNECT 及隧道代理的描述，详见 RFC 7231 - HTTP/1.1: Semantics and Content。实际上这种代理早就被广泛实现。

5. 第4章数据的结构

总字数：1113

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第4章数据的结构

4.1 JSON格式与JSON文件

JSON（JavaScript Object Notation）是一种用来表示数据结构的格式，其规范与JavaScript的对象字面量非常相似，非常适合用来表示格式化的对象数据。将JSON对象序列化之后以utf-8编码保存在文件中就是JSON文件。由于JSON格式简单，方便存储和传输，解析难度小，编码和解码方式优异，因而广泛应用在当今的前端与后端的数据交互中。

在本文中，爬虫爬取到的数据就是JSON字符串，通过反序列化解析之后即可获取从后端返回的用户数据。

4.2 User的结构

用户的个人profile存储在follow这个collection中, 主要的结构如下:

```
{
"_id":"5a51ba300604d374f28af6fb",
"profile":
{
"userId":363516402,
"gender":0,
"birthday":-2209017600000,
"city":220100
```

},
"__processing":
{
"follow":false,
"followed":false,
"playlist":false,
"detail":false
},
"__processed":
{
"follow":true,
"followed":true,
"playlist":false,
"detail":true
}
}
各个字段的含义如下:
1. _id(数据库主键)
2. profile.userId(用户id)
3. profile.gender(用户性别, 0为男性, 1为女性)
4. profile.birthday(用户的出生日期)
5. profile.city(用户所在的城市)
6. __processing(互斥锁, 用于并发)
7. __processed(特殊标识，表示这个用户是否已经处理过了)
4.3 Follow的结构
用户之间相互关注的信息存储在follow这个collection中, 主要的结构如下:
{
"_id":"5a4b488c954454ebecda66cb",
"from":"test",
"to":"test1",
"updatedAt":1515168662545
}
各个字段的含义如下:
1. _id(数据库主键)
2. from(关注别人的那个人的userId)
3. to(被关注的那个人的userId)
4. updatedAt(这条记录的插入时间)

6. 第5章数据的存储

总字数：904

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第5章数据的存储

5.1 数据库的选择

数据存储在自己搭建的MongoDB数据库中, 共五个collection, 总计N万条用户数据. MongoDB数据库是一种结构自由, 无schema的文档数据库, 便于存储这种对象化的用户数据.

5.2 数据库的搭建

本文中所用到的MongoDB数据库搭建在ubuntu上。

首先，通过apt安装：

apt update

apt install mongodb

然后设置为开机启动，通过systemd来控制和管理：

systemctl enable mongodb

systemctl start mongodb

至此，数据库的搭建就完成了。

5.3 数据库的管理

MongoDB数据库的操作语言是JavaScript。它的数据以collection为集合进行组织和存储，且某个数据库或者集合集合在使用之前不必提前创建，使用到的时候会自动创建。本文用到的数据存储在CloudMusic这个数据库内。我们只需要创建具有这个数据库权限的用户即可。

5.4 数据库的连接

数据库的连接方式如下：

```
const url = `mongodb://username:password@domain.com:port/CloudMusic?authMechanism=SCRAM-SHA-1&authSource=CloudMusic`
const mongodb = require('mongodb')
const MongoClient = mongodb.MongoClient
const getConnection = async function()
{
return (await MongoClient.connect(url, {poolSize: 200})).db('CloudMusic')
}
```

调用getConnection即可获取数据库的连接。

5.5 数据库的优化

由于数据库内存储的数据非常多，查询起来非常的耗时。所以，需要对数据库建立索引，针对常用的查询条件进行索引的建立，加快查询的速度。

7. 第6章数据的分析与可视化

总字数：6116

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第6章数据的分析与可视化

6.1 用户的总体分布情况

用饼状图来表示比例。

6.1.1 性别分布情况

在表示用户个人信息的数据结构中, profile.gender表示用户的性别(0代表未知, 1代表男性, 2代表女性)。

样本所包含的用户总数: 232433。

查询性别未知的用户的数量: {'profile.gender': 0}, 结果为 39466。

查询男性用户的的数量: {'profile.gender': 1}, 结果为 103419。

查询女性用户的的数量: {'profile.gender': 2}, 结果为 89548。

性别不明男性用户数量女性用户数量

39466 102419 89548

采用d3.js可视化之后如图4-1所示：

6.1.2 年龄分布情况

查询70后的用户数量: {'profile.birthday': {$gte: 0, $lt: 315532800000}}。

查询80后的用户数量: {'profile.birthday': {$gte: 315532800000, $lt: 631152000000}}。

查询90后的用户数量: {'profile.birthday': {$gte: 631152000000, $lt: 946684800000}}。

查询00后的用户数量: {'profile.birthday': {$gte: 946684800000, $lt: 1262304000000}}。

查询10后的用户数量: {'profile.birthday': {$gte: 1262304000000}}。

样本总人数: 88260。

年龄段人数

70后 849

80后 10873

90后 53504

00后 20019

10后 3015

采用d3.js可视化之后如图4-2所示：

6.2 用户在不同省份的分布
用中国地图来表示不同省份的分布情况
6.2.1 数量分布情况
对于每一个省份，以它的编号作为查询条件来查询符合条件的人数，从而获取全国所有的身份的用户人数，代码如下：

```
const result = {}
for(const gb in regions)
{
const num = await user.find
(
{
'profile.province': Number(gb)
}
).count()
result[regions[gb]] = num
}
```

结果如下表所示:

省份	用户数量
北京	11766
天津	2781
河北	5923
山西	3889
内蒙古	3526
辽宁	5265
吉林	2901
黑龙江	3730
上海	7317
江苏	14033
浙江	11514
安徽	7044
福建	5835
江西	5772
山东	11301
河南	10174
湖北	8216
湖南	7733
广东	23985
广西	4246
海南	1146
重庆	4387
四川	10242
贵州	2656
云南	5013
西藏	516
陕西	6274
甘肃	2798
青海	917
宁夏	969
新疆	5598
台湾	635
香港	832
澳门	196

对每个省份的用户人数使用D3.js可视化之后结果如图4-3所示:
对每个省份的用户密度(用户数 / 省份面积)使用D3.js可视化之后结果如图4-4所示:

6.2.2 性别分布情况

以省份的编号和用户性别来作为查询条件进行查询, 得到的不同性别的用户在全国各个省份的分布情况.

代码如下:

```
for(const gb in regions)
{
const female = await user.find
(
{
'profile.province': Number(gb),
'profile.gender': 2
}
).count()
const male = await user.find
(
{
'profile.province': Number(gb),
'profile.gender': 1
}
).count()
}
```

结果如下:

省份男性用户数量女性用户数量

北京 5513 4058

天津 1249 1005

河北 2696 2197

山西 1688 1522

内蒙古 1581 1357

辽宁 2462 1951

吉林 1324 1092

黑龙江 1654 1467

上海 3223 2641

江苏 6319 5136

浙江 4848 4184

安徽 3233 2671

福建 2582 2065

江西 2547 2188

山东 5046 4204

河南 4523 3837

湖北 3729 3027

湖南 3314 3250

广东 11343 8195

广西 1956 1546

海南 574 383

重庆 1904 1725

四川 4558 3971

贵州 1195 1046

云南 2288 2005

西藏 232 200

陕西 2908 2433

甘肃 1274 1104

青海 438 357

宁夏 389 403

新疆 2696 2058

台湾 274 298
香港 382 358
澳门 96 76
使用d3.js进行可视化展示之后的结果如下:
男性用户在全国的密度分布, 如图4-5所示
女性用户在全国的密度分布, 如图4-6所示
女性用户在所在的省份的用户中的比率在全国的分布, 如图4-7所示
6.2.3 年龄分布情况
查询代码如下:

```
const result = {}
for(const gb in regions)
{
const age70 = await user.find
(
{
'profile.province': Number(gb),
'profile.birthday': {$gte: 0, $lt: 315532800000}
}
).count()
```

其他年龄用户数量的查询和上面类似，只需要修改相应的birthday的数值即可查询不同年龄的用户。
结果如下:
省份 70后 80后 90后 00后 10后
北京 132 1167 2849 2849 271
天津 11 153 756 756 26
河北 26 308 1388 1388 50
山西 12 174 948 948 23
内蒙古 7 153 891 891 21
辽宁 22 324 1392 1392 53
吉林 14 146 746 746 23
黑龙江 23 175 969 969 30
上海 58 553 1981 1981 127
江苏 57 656 3820 3820 139
浙江 42 578 2749 2749 145
安徽 15 245 1919 1919 65
福建 21 306 1266 1266 78
江西 9 199 1342 1342 64
山东 29 568 3034 3034 109
河南 22 409 2383 2383 93
湖北 24 391 2206 2206 117
湖南 24 301 1830 1830 98
广东 75 1246 5387 5387 301
广西 14 211 891 891 53
海南 3 54 254 254 17
重庆 18 202 1211 1211 54
四川 34 451 2686 2686 137
贵州 8 94 625 625 28
云南 11 189 1293 1293 61
西藏 2 28 152 152 7
陕西 16 275 1759 1759 39
甘肃 6 111 746 746 27
青海 3 28 199 199 10
宁夏 2 33 224 224 11
新疆 21 266 1582 1582 83

台湾 13 47 152 152 13
香港 5 99 286 286 38
澳门 1 35 65 65 9
使用d3.js进行可视化展示之后的结果如下:
70后在全国的分布, 如图4-8所示
80后在全国的分布, 如图4-9所示
90后在全国的分布, 如图4-10所示
00后在全国的分布, 如图4-11所示
10后在全国的分布, 如图4-12所示
6.3 用户之间相互关注的情况的对比分析
使用弦图来进行展示
6.3.1 男女之间互相关注的数量的对比
查询男性用户对女性用户的关注数量: {'from.gender': 1, 'to.gender': 2}
查询男性用户对男性用户的关注数量: {'from.gender': 1, 'to.gender': 1}
查询女性用户对女性用户的关注数量: {'from.gender': 2, 'to.gender': 2}
查询女性用户对男性用户的关注数量: {'from.gender': 2, 'to.gender': 1}
具体代码如下:

```
const male2male = await $follow_sex.find
(
{
'from.gender': 1,
'to.gender': 1,
}
).count()
const male2female = await $follow_sex.find
(
{
'from.gender': 1,
'to.gender': 2,
}
).count()
```

其他情况下的相互关注数量的查询与上面类似，只需要修改相应的gender的数值即可查询不同性别之间的用户相互关注的数量。
查询后, 结果如下表所示:
关注男性女性
男性 1312487 927842
女性 633642 478432
使用d3.js可视化之后结果如图4-13所示:
6.3.2 不同省份的用户之间相互关注数量的对比
不同省份的用户之间相互关注数量的对比的查询代码具体如下:

```
for(let name in regions){
for(let name_inner in regions){
const num = await $follow_sex.find({
'from.province': regions[name],
'to.province': regions[name_inner],
}).count()}
```

查询结果较为复杂, 这里不列表展示.
使用d3.js可视化之后结果如图4-14所示:
6.3.3 不同年龄的用户的互相关注数量对比
查询不同年龄的用户互相关注数量对比的具体代码如下:

```
const year70 = 0
const year80 = 315532800000
const year90 = 631152000000
```

```
const year00 = 946684800000
const year10 = 1262304000000
const year20 = 1562304000000
const years = [year70, year80, year90, year00, year10, year20]
```

这里规定几个与特定时间点有关的常量，year70是1970年的开始时间，一般作为其实时间点，将后面的时间表示为从它开始走过的毫秒数。Year80是从1970年到1980年的毫秒数，后面的几个常量以此类推。

这样做的好处是便于时间的比较和存储。

查询方式如下：

```
'from.birthday':
{
$gte: years[index],
$lt: years[index + 1]
},
'to.birthday':
{
$gte: years[index_inner],
$lt: years[index_inner + 1]
},
```

查询结果较为复杂, 这里不列表展示。

使用d3.js可视化之后结果如图4-15所示:

6.4 采用floyd算法计算用户之间的距离

6.4.1 用有向图展示用户之间的关联关系

为了直观的展现用户之间的关联, 我们用有向图来描述这种关系.

我们用有向图中的节点来代表用户实体, 用有向图中的边来代表用户之间的相互关注, 这样, 用户之间的关联关系就表示成了一个巨大的有向图(见图4-16), 我们下面的分析也是基于这个有向图来做的.

在这里, 用户之间的距离是指在由用户节点和关注边所构成的有向图中, 某个用户节点与另外一个用户节点之间的最短距离.

6.4.2 计算用户的关联性

在代表用户关联的有向图中, 我们通过floyd算法来计算任意两点之间的最短距离

6.4.3 矩阵的存储

由于矩阵比较大, 存储在数据库中不方便进行运算, 运算时临时生成又比较耗时, 因此我们把矩阵序列化之后以JSON的形式存储在文件中, 运算的时候从文件中读入并且进行反序列化以及初始化.

6.4.4 矩阵的运算

从数据库中根据用户之间相互关注的信息, 生成表示有向图的矩阵, 下面所进行的运算都是基于该矩阵来进行的.

首先生成有向图的节点, 该过程中对有向图的边进行遍历是要去除重复的节点。

然后对节点进行处理, 便于编号和索引, 方便下面所要进行的运算。

```
const list = []
vertex.forEach(function(e)
{
console.log(e.index)
list.push(e.userId)
})
```

然后遍历有向图的边来生成有向图的矩阵。

最后根据生面生成的有向图来运用floyd算法计算出距离矩阵和路径矩阵。

至此, 有向图的运算已经完成, 我们下面的分析都是基于该运算结果来进行的.

6.4.5 对矩阵的运算结果进行分析

在前面矩阵的运算结果的基础上, 对矩阵进行分析。

有向图共有7586个节点, 10000条有向边。

有7586 * 7586 = 57547396对节点。

可以联通的节点有2457173对。

因此任意一个节点到另外一个节点有路径的概率为2457173 / 57547396 = 4.3%

最长的路径长度为7。

由此可以看出网易云音乐用户之间的关联度极高。

8. 第7章结论

总字数：537

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第7章结论

在本文中, 通过对网易云音乐用户数据的爬取, 处理, 分析以及可视化, 同时借助于比较新颖的基于地图的可视化技术, 比较直观明了的展示了网易云音乐用户在几个常见的维度以及在地理空间上的一些特征.

主要的结论有:

1. 男性用户比较多, 但是与女性用户的比例差距并不大(男女比例为: 44:39)

2. 90后用户占据了用户的大多数(61%), 其次是00后(23%), 再其次是80后(13%), 10后和70后只占据了很少的一部分, 因此可以看出网易云音乐的用户群体非常的年轻化.

3. 从地理空间的角度来看, 无论是用户总人数, 不同年龄段的用户还是男女用户人数在沿海的省份的人数和密度都比内陆地区要高

4. 从地理空间的角度来看, 除了台湾和宁夏之外, 所有的身份都是男性用户较多, 女性用户较少

5. 从用户互相关注的的维度来看, 最为活跃的是90后的用户.

与此同时, 本文通过对网易云音乐部分用户之间相互关注的情况的分析计算, 可以看出网易云音乐的用户之间具有极其强烈的关联性.

参考文献

[1] 李希娟. 大数据时代下的数据可视化研究[D]. 河北大学, 2014.
[2] 刘勘, 周晓峥, 周洞汝. 数据可视化的研究与发展[J]. 计算机工程, 2002, 28(8):1-2.
[3] 陈建军, 于志强, 朱昀. 数据可视化技术及其应用[J]. 红外与激光工程, 2001, 30(5):339-342.
[4] 石昊苏, 韩丽娜. 数据可视化技术及其应用展望[C]// 全国自动化新技术学术交流会会议论文集(一). 2005.
[5] 韩子良, 毕妤. 数据可视化在数据挖掘中的应用[J]. 计算机应用与软件, 2003, 20(11):71-73.
[6] 吴猛. 基于Web的数据可视化技术初探[J]. 福建电脑, 2007(12):58-59.
[7] 涂聪. 大数据时代背景下的数据可视化应用研究[J]. 电子制作, 2013,47(5x):118-118.
[8] 赵聪. 可视化库D3.js的应用研究[J]. 信息技术与信息化, 2015(2):107-109.
[9] 王媛媛, 丁毅, 孙媛媛,等. 数据可视化技术的实现方法研究[J]. 现代电子技术, 2007, 30(4):71-74.
[10] 任永功, 于戈. 数据可视化技术的研究与进展[J]. 计算机科学, 2004, 31(12):92-96.
[11] 黄志澄. 给数据以形象给信息以智能数据可视化技术及其应用展望[J]. 计算机安全, 1999(6):3-9.
[12] 张浩, 郭灿. 数据可视化技术应用趋势与分类研究[J]. 软件导刊, 2012, 11(5):169-172.
[13] TomSoukup, IanDavidson. 可视化数据挖掘:数据可视化和数据挖掘的技术与工具[M]. 电子工业出版社, 2004.
[14] 王维江, 张俊霞. 数据可视化技术研究的新进展[J]. 计算机时代, 2002(5):4-6.
[15] JulieSteele, NoahIliinsky. 数据可视化之美[M]. 机械工业出版社, 2011.
[16] 陈明. 大数据可视化分析[J]. 计算机教育, 2015(5):94-97.
[17] 王媛媛. B/S模式下数据可视化研究及其在商业智能中的应用[D]. 河北工业大学, 2007.

[18] 王子毅, 张春海. 基于ECharts的数据可视化分析组件设计实现[J]. 微型机与应用, 2016, 35(14):46-48.
[19] 高科. 基于HTML5的数据可视化实现方法研究[J]. 科技传播, 2013(1):218-219.
[20] 张秋侠. 基于数据挖掘技术的数据可视化分析的研究与实现[D]. 哈尔滨理工大学, 2004.
[21] 邵荣. 基于SVG的数据可视化的研究与应用[J]. 电脑开发与应用, 2009, 22(4):56-58.

致谢

感谢我的父母，家人，朋友，老师在论文编写期间对我的支持和帮助。徐昊老师，王康平老师在此篇论文的编写上给予了我莫大的帮助，在此致以由衷的谢意。