摘 要

随着信息技术与互联网的发展以及大数据时代的到来，人们活跃在互联网之上并留下自己的行为数据。数据即为价值，获得人们在互联网上产生的数据是获取有价值数据的基础。网络信息抓取系统就成为一个值得去研究的方向，它是做数据挖掘的基础设施，要分析数据，就得拥有数据，要获得用户数据，就有了网络爬虫。

本系统是一个基于网络信息抓取的数据分析、处理系统。爬虫是一个用来分解Web页面中文本结构并将有用信息持久化到数据库中的工具，它不是一个新兴的技术，平时我们使用的搜索引擎就使用了爬虫技术，或者被称为网页蜘蛛，它会根据一定的算法来爬取符合爬虫协议的整个互联网站点中的数据。在本系统中爬虫作为一个基础服务获取网络中用户产生的数据。

本文首先分析了同类系统的发展与趋势，以及介绍开发本系统的背景与目的，然后介绍了系统实现需要使用到的关键技术，接着阐述了需求、系统实现、详细设计以及功能测试，最后对本项目进行了总结与对该类项目发展的展望与期待。

**关键词**：爬虫、分词、信息摘要、数据可视化

Abstract

With the development of information technology and the Internet and the arrival of large data age, people are active on the Internet and leave their own traces. Data is value, and getting people on the Internet is the basis for obtaining valuable data. Network information crawling system has become a worthwhile to study the direction, it is to do the data mining infrastructure, to analyze the data, you have to have data, to obtain user data, there is a web crawler.

The system is a data analysis and processing system based on network information crawling. The crawler is a tool for breaking down the text structure in a Web page and persisting useful information into the database. It is not an emerging technology. Normally, the search engine we use uses crawler technology, or it is called a web spider. Will be based on a certain algorithm to crawl crawler agreement in accordance with the entire Internet site data. In this system the crawler as a basic service to obtain the network generated by the user data.

This paper first analyzes the development and trend of the same kind of system, and introduces the background and purpose of the development of the system, and then introduces the key technologies needed to be realized by the system. Then, the requirements, system realization, detailed design and function test are expounded. Made a summary and prospects for the development of such projects and expectations.

**Keywords**: crawler, participle, information digest, data visualization

1 绪论

* 1. 项目背景现状以及发展前景

当今是一个信息爆炸的时代，是属于大数据的时代。各大互联网企业都在通过掌握着的用户数据来分析用户的各种社区属性、行为属性，致力于提升产品的使用体验，以及更加智能的问题解决方案。本项目切合当下互联网发展的趋势，先从爬虫入手，通过网络爬虫程序来获取大量的用户数据，然后对数据进行计算，并在多个维度进行数据的分析，最终以可视化的方式展现出来，实现对大量的数据有一个直观的数据感知。

在中国的互联网企业，支付宝它掌握了亿级用户的支付数据，所以它们能通过用户的购买行为支付行为来建立起一个用户的信用体系；滴滴出行，它掌握了人们的出行数据，它能实现对你出行路线的智能规划；微信知道用户的社交关系，能在朋友圈给用户定向投放可能感兴趣的广告。可以看出拥有数据的重要性，以及出现了一种说法就是，未来的“互联网战争”就是争夺用户数据的战争。那么网络爬虫其实就是获取用户数据的一种非常直接的手段。本项目爬取部分微博用户的数据以及微博内容信息进行分析，可以得到这部分用户基本属性形成用户的基本画像，以及通过这部分数据的分析，可以得到各个基本属性在整体数据中的占比或者排名。

提出基于“网络信息抓取以及处理系统设计与实现”基于以下几点的分析：

1. 网络已经离不开人们的生活，当下的web平台内容由用户主导生成，关注网络上用户产生的数据变得尤为的重要。

2. 网络信息抓取往往是与大数据相关，大数据时代已经来临已成为一个不争的事实，大数据作为一种新的资源正在给我们的社会生活带来深远的影响，而这些数据正是出于我们开放的web环境中，所以要得到大量的信息，我们就需要结合网络信息抓取工具。

3. “数据即为价值”这句话在web1.0时代可能体现得还不够明显，在大数据时代下，用户即数据产生源，数据即价值，所以面对大量的数据应该去如何处理获得有用的信息就变得非常的重要，那么本课题中涉及到分词和NLP相关领域，这就变得很有具有研究的意义。

4. 随着社会的发展，我们处理的数据量越来越大，对实现数据可视化的需求也越来越强烈，因此在大数据时代背景下的可视化技术必定会在生产中发挥不可替代的作用，可视化让数据分析更加便捷，实现数据可视化可以方便的提取出有价值的信息。

我们从网络上抓取到数据，然后对数据进行处理，得到我们关心的信息，然后进行可视化，这是一个完整关系非常紧密的流程，目的是为了在大量的用户信息中快速的得到我们想要的关键数据，对用户在网络中的行为进行数据挖掘。

* 1. 系统组成及开发运行环境

本系统由三个部分组成：网络爬虫、数据处理程序、数据可视化平台。

网络爬虫会抓取新浪微博的用户数据，其中包括但不限于微博用户的基本个人信息，微博用户发送的微博信息。抓取的信息来自于一个微博用户的集合。

数据处理程序的作用是在获得了微博用户数据之后对这些数据进行计算，其中包括但不仅限于对数据的摘要提取、关键字提取。最后将每个用户的处理结果放在一个用户数据分析表中。

数据可视化平台是一个Web网站，其采用前后端分离的方式，后端采用Node.js的web开发框架Koa.js提供一个数据查询API服务，前端使用D3.js来实现数据呈现。

系统实现的开发环境具体为：

开发系统：macOS Sierra；

集成开发环境：IntelliJ IDEA CE；

代码编辑器：Visual Studio Code；

系统运行环境基于：CentOS 6.5、Java 8、MySQL 5.6、Node.js 7.9、D3.js 4.0、Koa.js 2.0、现代化浏览器。

本项目使用GitHub来进行代码托管，使用shell脚本来完成本地到线上环境的代码上传、部署、运行。

* 1. 本文组织结构

本文分为七个章节，各个章节的主要内容如下：

第一章：绪论，主要说明了本项目的发展背景和前景，以及对整个项目有个大体阐述。

第二章：关键技术分析，本章主要说明了实现该项目的使用到的具体技术以及对该技术的大致介绍。

第三章：系统需求分析：本章系统性的分析了完成本项目需要实现功能的具体的点，对每个模块的功能进行了详细的阐述。

第四章：系统设计，该章节主要介绍了本项目的架构设计以及数据库设计原型设计等。

第五章：系统实现，本章主要展示了系统最终的实现，各个功能最终的实现结果以及部分运行截图展示。

第六章：系统测试，本章对系统的鲁棒性以及可用性进行了测试，以及阐述了测试在开发过程中扮演的重要角色。

2 关键技术分析

2.1 HTTP相关介绍

1. HTTP协议

HTTP（HyperText Transfer Protocol）是一种使用非常广泛的网络协议，它能获取到Web上如HTML这样的资源，在网络中是交换数据的基础，是一种服务端到客户端的协议。服务端和客户端相互交换数据来进行消息传递，最常见的使用场景是在我们的浏览器中。

在HTTP协议中没有规定它或者它支持的层，所以HTTP可以在任何的互联网协议上以及其他网络上进行实现。它假定下层协议可以提供可靠的传输，所以能够提供这种保证的协议都可以被它使用，因此HTTP协议的底层协议由TCP进行实现。HTTP的本质是简单的、可扩展的、无状态和有会话的。其中HTTP header让协议变得易于扩展，只要client与server端达成header的语义上的一致，新的功能就很容易的被加进来。HTTP是被设计为无状态的，在一个HTTP连接中两个不同的请求是没有任何关系的，这样就会存在一个问题，client在持续与server交互的过程中，server无法记录client的状态和操作，但是通常我们可以通过header扩展cookie来解决这个问题，每次server与client交互的过程中都带上cookie这样使得交互有会话，使之能交换信息，在我们的爬虫程序中就使用到了cookie，使每次我们的HTTP请求都会带上一个用户的cookie来告诉服务器，我们是已经有“身份”的，这样server才会同意把正确完整的数据返回给我们。

1. HTTP请求与响应

HTTP请求是对服务器资源的访问，一个HTTP Request主要包括了请求行、请求头、请求正文。请求的第一行是“方法 URL 协议/版本”并以回车结束，比如：

GET /index.html HTTP/1.1

这代表了这是一个GET类型的请求，请求服务端根目录下的index.html资源使用的是HTTP1.1版本，现在HTTP最新的版本是HTTP2，但是目前互联网中使用最广泛的HTTP的版本仍然是1.1。在HTTP1.1协议中规定了八种方法来指明以怎样的方式来操作指定的资源，其中使用得最为广泛的是GET、POST请求，一般GET请求用于向资源发出“显示”动作，主要目的是读取server端的数据，而POST主要的动作是向服务端提交数据，这在Web页面的表单提交中使用广泛。

HTTP响应即为在收到HTTP请求之后server端对其作出的回应，它与HTTP请求是类似的也由三个部分组成，分别是状态行、消息报头、响应正文。状态行由协议以及协议版本、状态码、状态描述构成。一个普通的HTTP响应状态行格式如下：

HTTP1.1 200 OK(CRLF)

其含义上面的介绍已经提到部分，其中CRLF是回车换行符，200是一个HTTP请求成功的响应状态码，表示这次请求被服务器接受并且正确的返回，HTTP状态码在HTTP响应中非常重要，是判断请求状态的重要依据，常见的状态码有：

1xx：消息 - 表示请求已经被服务器接受正在处理。

2xx：成功 - 请求被服务器接受并且正确处理。

3xx：重定向 - 资源可能被转移，需要后续操作才能完成。

4xx：客户端错误 – 客户端请求错误或者请求无法实现。

5xx：服务器错误 – 服务器未能实现合法的请求。

响应头主要包含了服务器端的一些描述以及对客户端的一些设置，响应正文就是服务器返回给客户端的资源。

2.2 爬虫搜索算法介绍

在爬虫下载网络中的资源时使用最多的算法是与图论相关的算法，其中使用最多的就是BFS（Breath-First Search）广度优先搜索与DFS（Depth-First Search）深度优先搜索，在通过一个页面作为入口快速的进行层次遍历时我们经常使用BFS。

广度优先搜索它是先从根节点开始然后沿着树的宽度遍历树的节点，在本课题中就可以把用户关系模型抽象成为一颗N叉树，爬虫需要遍历这个N叉树使用的算法就是BFS，它访问终止的条件就是树中所有的节点均被访问到。

图2.1就是一颗多叉树，对其进行BFS遍历的顺序就为树中每个节点的顺序。

2.3 反爬虫以及相关应对策略

目前很多网站为了保护自己用户的数据，都做了很多反爬虫的相关工作，本课题中的爬虫目标站点就做了反爬虫设置。其中反爬虫常用的手段就是设置同一个用户的HTTP(S)请求频率，但是由于HTTP是无状态的，所以判断是否是同一用户或者爬虫程序的方式就有多种。常见的是判断每次请求的HTTP请求头中的User-Agent，这是用户访问该站点的代理，通常是我们使用的浏览器，以及判断用户带上的cookie，一般cookie中携带了标识用户的信息，这样就能在服务器端验证是否是同一用户在请求，还有就是通过IP地址来判断。通常情况下做的比较完善的反爬虫会更具这些甚至更多条件来综合判断一个请求是否是机器人或者正常的用户请求。

那么反爬虫的应对策略入口即为反爬虫的策略，为了不让服务器认为我们是爬虫程序，我们可以把请求的User-Agent换成常规浏览器或者搜索引擎的User-Agent，然后不停的更换cookie值，如果需要验证用户身份，那么还需要登录用户的cookie值，以及使用代理IP，维护一个Proxy IP池，每次连接都随机一个IP，这样就可以提升我们的爬取效率而不被站点轻易的放入黑名单。

2.4 Node平台以及Koa.js

Node或者Node.js指的是一个开放源代码、跨平台的JavaScript语言执行环境，由Node基金会维护，它采用Google的V8引擎作为JavaScript的执行引擎。事件驱动非阻塞I/O使其成为高性能Web服务器中的一员。Node.js以单线程运行，异步I/O让其不仅能支撑起高并发的请求，还能避免了多线程额外的资源开销以及线程上下文切换时的性能损耗。但是异步的编码方式让很多开发者不能很好地适应其程序执行顺序，以及异常、错误的捕获。

Node.js官方推荐的Web框架是Express，Koa是一个类似的Express的web开发框架，并且Koa的作者为Express的原班人马，它的主要特点是使用了ES2015的Generator函数，并对架构进行了重新设计，换句话说Koa其实是和Express拥有一样的内部架构，但是Koa使用了新版的ECMAScript标准来实现。在本项目中使用的Koa版本是2.0，其特点是使用了最新的async/await来实现之前的Generator，这可以让开发者更加方便以书写同步代码的方式来编写异步代码。

2.5 Java与MySQL简介

Java是一门使用非常广泛的程序设计语言，它拥有跨平台、面向对象泛型支持等特点。他的编程风格类似于C++或者C#，它并不是完全的编译类型的语言，它首先是将源代码编译成字节码，然后依赖不同平台如Windows/Linux上的虚拟机来解释这些字节码进而执行程序，所以它的执行过程是先编译后解释运行。这比起其他编译-链接-运行的编程语言来讲在执行效率上稍有逊色，但是随着JVM的不断改进，Java的执行效率已经得到大幅度的改善。Java语言目前主要用于Java Web开发以及移动平台Android应用程序的开发，它同时也能开发桌面应用程序。

MySQL(社区版)是一个开放源代码的，关系数据库管理系统，之前被Sun公司收购，后来Oracle收购了Sun公司，所以目前它是Oracle旗下的一款产品。MySQL因为开源、性能好、使用成本低很快成为了一个非常受欢迎的数据库管理系统。他被广泛的应用在中小型的网站中，随着MySQL的不断优化它目前已经能接受住更大规模网站的考验，在Google、Facebook已经被广泛的得到应用。

2.6 Web数据可视化与D3.js

1. Web数据可视化

数据可视化领域的起源可以追溯到上世纪五十年代的计算机图形学早起，人们利用计算机创建了首批图表。在如今的大数据互联网时代，用户每天在互联网上产生的海量数据，如果我们想对这些海量的数据进行分析，挖掘可用价值，那么直接面对这些冷冰冰的数据，让我们感到很难下手，如果将数据可视化起来，让数据以丰富的图表进行展示，那么我们将会更好更方便的发现有用的数据，更加有利于分析数据之间的关联。

数据可视化的实现方式有多种，最常见的是一些办公软件的实现，比如office系列软件中就能做出一些漂亮的图表，这在我们日常的办公中经常会用到，但是当我们面临海量的数据信息，结构复杂的数据时，这些办公软件的实现数据可视化就变得尤为的困难甚至无法实现。那么使用编程的方式以W3C开放标准使用相关Web技术，就能使数据在Web页面中能很好的展示出来。最为传统的方式是通过HTML与CSS制作Web页面进行数据展示，需要交互的地方可以使用JavaScript，但是这都不是数据可视化在Web页面中的最佳实践。新兴的Web标准与HTML5的出现，让数据驱动文档变得尤为的轻松起来。

SVG（Scalable Vector Graphics）是一种基于可扩展的标记语言，用于描述矢量的图形格式。SVG是W3C标准，目前在现代浏览器中均被部分实现。Canvas在Web数据可视化中也经常用到，它最早是由Apple公司引入至Webkit中，后来在Google的Chrome中也被实现，它由一个可绘制的地区在HTML中定义高度宽度，JavaScript代码可以访问到该区域，通过一套完整的API来实现在该区域的绘图功能。

1. D3.js

D3（Data-Driven Documents）即为数据驱动文档，是一个使用JavaScript显示动态图形的库，兼容W3C标准，使用了SVG、Canvas、CSS标准。在Web数据可视化类库中比较知名的还有Echarts、Highcharts，但是D3与其不同的是其高度定制化以及其专业性，D3.js面向海量数据处理时优势明显，其缺点就是使用门槛较高。

2.7 本章小结

本章主要围绕本项目所使用到的关键技术进行阐述以及简单的介绍，通过这些简要的分析也体现了在项目技术选型时为什么要使用该项技术。

3 系统需求分析

3.1 系统需求概述

本系统的最终目的是为了获取新浪微博大量的用户数据，包括但不限于个人详细信息，个人发送或者转发的微博内容，然后持久化数据到自己的服务器中。获取数据的入口为一个用户，然后对其关注列表进行层次遍历。在获取到一定量的用户数据之后对系统服务器中的数据进行处理，不仅要实现微博用户个人信息的梳理还要对用户所发送微博数据进行摘要提取，得到全量微博数据内容的摘要，形成一个简单的用户画像和全量数据的统计概况。在处理完数据之后，还需要对其数据进行可视化，需要一个可视化系统来查看分析我们得到的关键数据。

3.2 爬虫需求分析

1. 功能需求

1）数据量及内容需求

既然要做大数据的分析，那么一定要具有足够的数据量。

用户关注列表数据应当超过100万条记录，用户关注列表应当是以一个用户作为入口，抓取其关注用户的列表，然后再根据这些用户继续抓取关注列表，其中能对其进行层次分析，比如第一个用户应当是第一层次，后续关注列表中的用户应当是第二层，第二层中的所用用户的关注列表应该是第三层，以此类推，通过这些用户形成一个用户集合，其中有会部分数据是重合的，但是重合数据会处于不同的关注层次中，这个用户集合应该是后续爬虫爬取用户信息以及发送微博的基础。

用户详细信息记录应当超过10万条，爬虫应当抓取超过10万用户的基本信息，包括但不限于以下信息：

1. 微博用户名
2. 用户性别
3. 用户的所在地域
4. 个人简介
5. 信用记录
6. 用户受教育程度

爬取微博信息的数据来源应该是在这个10万用户之中所发送或转发的微博，但是应该是选择性的获取其中的这些信息，比如应该忽略发送微博数较多的用户，参考值1000，当不符合爬取条件时爬虫应当主动跳过，防止单用户长期占用爬虫资源导致整个系统性能低下并且获取到数据的价值低下。

微博数据是非重要的数据，我们至少应该拥有如下功能：

1. 知道该微博是哪一个用户发送的。
2. 在微博网站中可以查看到这条微博的url地址。
3. 该微博的发送时间。
4. 微博信息的来源，比如是哪一个终端设备发送的。
5. 发送微博的内容。
6. 判断该微博是否是转发的。
7. 如果不是该用户自己原创的微博应该获取原始微博的内容。

2）爬取方式

需要对微博用户数据进行按层次的爬取，如按照一个用户的关注列表作为爬虫程序的入口，然后对关注列表的用户的关注列表进行爬取，这样的爬取方式类似于N叉树。爬虫模块的用例图如3.1所示。

1. 非功能性需求

1）系统鲁棒性与稳定性

爬虫程序应该按照规划的爬取路径去获取用户的信息，并且能正确的解析服务器返回的数据，爬虫程序应当稳定的运行在服务器，只有爬虫到设定的爬取临界值或者人为终止时，信息抓取才应当停止，否则系统应该正常的运行，不应当被反爬虫或受外界干扰如高延迟而终止系统的运行，整个过程应当是可靠的。如果在抓取过程中某条记录抓取失败，应当被日志所记录，所以爬虫系统应该还具有完备的日志系统，用于分析记录整个爬取过程以及用于错误调试等。

2）性能要求

整个爬虫程序应当尽量的少占用服务器的资源，系统要在内存少于1G，带宽小于1M的环境中正常运行。

3）部署与运行

爬虫系统在本地开发，部署到服务器上应该是很便捷的，能直接发布到服务器上运行而不应该是在服务器端编译。

3.3 数据处理需求分析

数据统计应当是在数据量准备充分之后进行的，当爬虫程序获取完目标数据之后然后使用数据处理系统来进行离线计算，并把最后处理过的数据放入数据库中与对应的用户关联起来。

1. 功能性需求

本系统主要的功能是用于计算爬虫程序获取的数据，它应当完成如下的功能指标：

1. 对微博用户基本信息中的所在区域进行处理，提取出用户所在的省份，并放入新的用户关系表中。
2. 对用户的受教育程度进行统计，通过用户的学校来统计受教育程度，分为大学、中学、小学、未知，将处理结果存放在新的表中。
3. 对用户所发微博的来源进行统计，并获取发送来源最多的作为用户发送微博的主要来源存放入数据库。
4. 用户发送微博的内容进行摘要提取，包括转发时自己发送的内容以及源微博内容，提取关键字不少于3个。

数据处理用例图如3.2所示。

1. 非功能性需求

本系统应当具有良好的稳定性及健壮性，要求以尽量短的时间内完成对数据的计算与处理，并尽少占用服务器的资源，具备完整的日志系统来记录整个计算过程，如果错误或者有异常抛出应当可以通过日志文件来进行错误定位、分析。

3.4 数据可视化平台需分析

数据可视化平台应当是一个可以在线访问的站点，它可以对爬虫程序和数据处理程序最后的计算结果进行友好的展现。它还应该具有良好的第三方接入API，来共享这些数据，并且具备良好的兼容性与可交互性。

1. Web后端

1）功能型需求

Web后端应该是一个具有数据查询分析功能的Web程序，它能对爬虫数据以及处理系统的数据进行访问。

2）功能性API

后端应该向前端提供数据查询的API，其应当包含以下功能：

1. 输入某一个已经被爬取过的微博用户用户名，返回该用户的个人详细信息，以及处理系统计算后的微博信息摘要。
2. 能知道具体以为用户在那个时间段内在微博上比较活跃，比如在那个时间段发送的微博数据比较多。
3. 能对全部用户的微博来源进行分析，可以统计出使用终端设备的排行榜。
4. 统计所有用户的学校，并得出出现频率前十的学校名称。
5. 能统计微博用户的所在区域。
6. 统计全量数据的性别。
7. 用户所在的公司。
8. 用户的学历情况分为大学、中学、小学、未知。
9. 全量用户数据中微博给用户的标签情况。
10. 全量用户中信用情况的占比。

除了能处理已经抓取到的数据之外，Web后端还应该能对微博用户数据进行在线抓取，如给定一个用户名称或ID，能在线请求微博的API并将数据返回给前端。

3）接口通用性

所有的接口均可以被第三方网站调用实现对数据的共享，所有接口应该使用常见的数据格式，比如返回格式为JSON或者XML的形式便于前端解析。

4）非功能型需求

所有接口应该具有良好的设计风格，错误调用提示以及应该具有快速响应等特点。

1. Web前端
2. 功能型需求

Web前端的主要功能是实现后端接口返回数据的可视化，实现让用户能更好的阅读数据，获取有用价值。前端通过后端API返回的数据对数据进行以图表的形式展现，可视化平台进入后应当展现包括以下数据：

1. 整个爬虫获取到的全部数据概况
2. 用户性别统计图
3. 用户区域统计图
4. 用户所在的公司统计图
5. 用户学历占比统计图
6. 用户学校排名统计图
7. 用户微博标签统计图
8. 用户阳光信用情况统计图

可视化平台首页默认操作用例图如3.3所示。

除了能展现爬虫处理好的数据，前端应该还具备一个搜索用户数据的功能，给定一个微博用户名或者ID，如果该用户在本地系统具备数据，就将系统的数据返回给前端，并形成用户的基本画像，包括用户的基本信息，用户发送微博处理后的结果摘要提取，活动时间报表，如果本地没有该微博用户的数据，后端应该在线请求微博API获取用户的基本数据然后进行展现，其用例图如3.4所示。

2）非功能能型需

前端展示的图表信息应该是可交互的，而不是一张用数据生成的图片，用户可以更改一些筛选条件或者根据一定的规则来重新渲染图表，用例图如3.5所示。

3.5 本章小结

本章主要对整个系统实现所需求信息采集、数据处理、可视化模块的功能进行详细的阐述，以及对部分数据量有一个固化的指标，介绍了本系统实现后应该具有的所有功能。

4 系统设计

4.1 系统设计原则

整个系统应该严格围绕设计原则来进行设计，通过对本系统的整体需求和现有的软硬件资源制定了以下的系统设计原则。

1. 可用性

系统的可用性是作为最为重要的原则，整个系统应该对照系统需求提供完整的服务。

1. 高效性

爬虫获取数据需要发送大量的网络请求，这很考验服务器以及爬虫程序的性能，在反爬虫以及服务器硬件的限制下，应该尽量提高数据抓取的效率。

1. 稳定性

数据采集操作是一个长时间的集网络I/O、文件I/O、数据操作的后台任务，系统不应该轻易的崩溃或者被阻塞从而影响后续的采集工作，所以系统应该具有很好的稳定性。

1. 自动化

系统会部署在不同的几个机器上，可能是在开发环境的机器或者服务器，整个系统部署运行应该是很便捷的，所以整个系统应该易于移植，实现高度自动化的部署、运行。

1. 可扩展

系统各个模块之间应该具有较低的耦合度，并且在不会影响现有模块功能的情况下可以很容易的实现新特性扩展。

4.2 系统架构设计

本系统最终运行在一个小型的服务器集群之上，其中爬虫代码分布在多个VPS主机上，数据处理系统和数据库系统位于一个单节点之上，Web服务位于单一的节点，系统总体架构如图4.1所示

由此，从纵向上看，系统架构分为三层。第一层是微博的服务，该层是我们获取数据的来源，可以是微博的API或者网页，该层不需要本系统去实现，但是需要去分析微博提供的服务，给爬虫程序提供入口url格式等；第二层是云服务层，我们的爬虫程序、数据处理系统、web后端都会托管在几个VPS上，这些是我们系统运行的环境之一；第三层是用户层，包括开发人员以及用户，开发人员通过部署脚本将程序部署到云端运行，用户通过浏览器访问我们系统提供的服务。从横向看，系统云端也被切分为三层，整个VPS集群都会部署爬虫程序，其中C2作为主要的服务器提供Web访问服务以及作为数据处理中心，在C2周围的服务器都只作为爬虫服务器，并将采集到的数据传回C2做计算。

4.2.1 爬虫系统设计

爬虫的整体架构分为三模块，分别是底层工具层、数据访问控制层、爬虫逻辑层，爬虫逻辑层通过调用底层工具层和数据访问控制层来实现网络请求、数据解析、数据储存等操作。其架构图如4.2所示。

下面详细说明各个层次的设计细节：

1. 底层工具

底层工具是一个最基础模块，它的主要功能是为上层模块服务。它主要包含了日志模块、内容解析模块、数据库模块、网络模块三个子模块。

1）日志模块

它作为整个爬虫系统的日志操作核心，提供不同等级的日志输出包括信息、警告、错误，直接反应为不同的输出格式，可用于分析爬虫的爬取状态以及记录爬取行为和错误追踪，爬虫访问日志都需要使用该模块将日志信息格式化输出。

2）内容解析模块

内容解析模块主要用于解析网页或者JSON中的文本内容，通过一段文本的输入得到具体的一个实体对象实现对网页信息的提取，如图4.3所示。

3）网络模块

该模块用于发送网络请求，为爬虫逻辑模块提供服务，HttpRequest工具类提供发送GET、POST方法，并且对方法进行重载，支持以匿名或者非匿名的方式发送该请求，这里的发送方式是一个非常重要的点，如果以匿名的方式发送HTTP请求那么这个请求将不会带上用户的cookie，也就是说微博的服务端并不知道用户是谁，非匿名那么就会带上cookie，在爬取微博关注列表的时候我们应该使用非匿名的形式，因为微博查看关注列表需要用户登录。这样非匿名的爬取方式会受到微博反爬虫的影响，因为服务端可以知道具体用户是谁，就能直接限制该用户的请求频率而不需要在乎是否为同一IP。在匿名模式下，不会携带cookie，爬取微博用户信息的时候就会用到，这样我们就可以配合代理IP池来应对反爬虫策略。HttpRequest不仅仅是设置了是否能携带cookie，还做了其他反爬虫的应对策略设置。每次通过HttpRequest的请求的User-Agent都将会是随机的，在底层工具模块中一个Weibo工具类中存放了常见的User-Agent。不仅如此，HttpRequest还制定了请求出错或者被反爬虫限制后的应对策略，它将控制HTTP请求发起的速率、线程休眠或者重新发起上次失败的请求。网络模块工作流程图如4.4所示。

4）数据库模块

另外一个模块是关于数据库操作的，ConnectionFactory是一个获取数据库连接的工厂类，里面包含了Connection的单例实现，所有的爬虫数据操作都会使用这个底层工具来获取数据库的连接。

1. 数据访问控制层

数据访问控制层包含了用户数据模型的抽象，其中会包含用户、微博的实体，每一个实体的实现都和微博用户、微博内容的数据结构相对应。DAO（Data Access Object）提供了数据访问对象，其中包含了各个实体对象访问数据库的方法，可以很方便的将一个用户或者微博实体插入到数据库中，如图4.5所示。

1. 爬虫逻辑层

该层是对爬虫数据集采逻辑的控制，包括用户关系爬取模块、用户基本信息爬取模块、微博内容爬取模块。

1）用户关系爬取

爬取用户关系可以通过爬取用户的关注列表，通过关注列表可以获取这些用户的微博ID，获取微博用户ID是后续爬虫工作的关键，因为后续的微博用户信息微博用户发送的微博信息都要基于用户ID的API或者页面去抓取。获取微博用户ID的一种手段是通过微博的用户名去获取微博对应的ID，然后通过该ID获取该用户的关注列表，这是一个JSON格式的数据，包含了其关注列表的用户ID，这样我们通过循环就能获取指定微博ID用户的全部关注列表的微博用户ID。

通过广度遍历的方式我们就能以一个用户作为入口按层次去获取用户的ID，这样就能获得我们爬虫的用户集合，它的形式为ID-用户名的形式。爬取流程图如4.6所示。

2）用户基本信息模块爬取

通过爬取用户的ID，获得了一个ID数据集合，就可以遍历这个集合就能通过微博个人信息页面的URL或者API来获取用户基本信息。

3）微博内容爬取模块

同理在获取到用户ID集合后，就能通过用户ID获取用户发送微博的列表，然后使用网络模块和内容解析模块进行数据采集。

4.2.2数据处理系统设计

数据处理系统包括用户信息处理和微博信息处理两个子模块，其中处理的方式主要包括人工分析数据并使用程序进行批量处理，以及使用Ansj开源的分词处理工具来进行文本摘要的提取。

1. 用户信息处理

爬虫抓取完用户数据之后，这些数据还不能很好的将其归类，需要二次处理，将用户所在的省份以及教育程度和微博信息来源最多的设备等提取出来，其功能结构图如4.7所示。

1. 微博信息处理

将爬虫获取到的微博信息进行遍历，使用Ansj进行中文分词，提取微博内容摘要信息，如果是转发的微博还需要对源微博进行摘要提取，工作流程图如图4.8所示。

4.2.3 可视化平台系统设计

可视化平台系统主要对处理系统处理好的数据进行查询然后做前端展现，其采用前后端分离的方式进行设计。

1. Web后端

后端系统以Node技术平台为基础，Koa提供一个Web服务器，其系统功能模块结构如图4.9。

1. Web前端

前端是一个单页应用完全进行模块化设计，JavaScript脚本采用AMD规范，并使用require.js来进行异步加载，每个图形报表为一个模块，使用art-template作为前端模板引擎，并使用tmod.js来对模板进行预编译。其模块结构图如4.10所示。

4.3 数据库设计

本课题使用的数据库为MySQL，数据库名为weibo\_portrait，使用的字符集为UTF-8且使用支持事务处理的InnoDB存储引擎。

1. user\_ref表

user\_ref表的作用是存储了用户之间的关系，以及作为爬虫程序的入口。里面存储了爬虫程序需要抓取信息的全部用户。

表 4-1 用户关系表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段编号 | 字段名称 | 字段类型 | 中文名称 | 约束关系 | 备注 |
| 1 | id | bigint | 主键 | 唯一 | 记录id |
| 2 | uid | bigint | 微博uid | 非空 | 微博ID |
| 3 | username | varchar(5) | 微博用户名 | 非空 | 微博昵称 |
| 4 | depth | tinyint | 深度 | 非空 | 用户所在所在的层次 |

1. user表

user表存储用户的个人基本信息。

表4-2 用户信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段编号 | 字段名称 | 字段类型 | 中文名称 | 约束关系 | 备注 |
| 1 | id | int | 主键 | 唯一 | 记录id |
| 2 | uid | bigint | 用户id | 非空 | 微博id |
| 3 | avatar | varchar(100) | 用户头像url | 默认 | 微博用户头像url地址 |
| 4 | username | varchar(50) | 用户名 | 非空 | 微博用户名 |
| 5 | home\_page | varchar(200) | 主页 | 默认 | 用户个人主页url |
| 6 | gender | int | 性别 | 默认 | / |
| 7 | address | varchar(100) | 地址 | 默认 | / |
| 8 | introduction | varchar(200) | 简介 | 默认 | 用户自我简介 |
| 9 | company | varchar(100) | 公司 | 默认 | 用户所在公司 |
| 10 | school | varchar(100) | 学校 | 默认 | 用户教育情况 |
| 11 | blog | varchar(100) | 博客 | 默认 | 用户博客url地址 |
| 12 | level | tinyint | 等级 | 默认 | 用户微博的等级 |
| 13 | credit | varchar(50) | 信用 | 默认 | 微博用户的阳光信用 |
| 14 | reg\_time | bigint | 注册时间 | 默认 | 微博用户的注册时间 |
| 15 | follow\_count | int | 关注人数 | 非空 | 用户关注的总人数 |
| 16 | fans\_count | int | 粉丝数 | 非空 | 用户拥有的粉丝数 |
| 17 | birthday | bigint | 生日 | 默认 | 用户生日 |
| 18 | tweet\_count | int | 微博数 | 默认 | 用户所发表的微博数 |
| 19 | job | varchar(50) | 职业 | 默认 | / |
| 20 | tags | varchar(200) | 标签 | 默认 | 用户的标签 |
| 21 | last\_modify | bigint | 最后时间 | 默认 | 最后修改该记录的时间 |

注：在该表的设计中时间都使用了bigint的数据类型来存储是为了存储一个时间戳。

1. proxy表

该表存储了爬虫所需要的代理ip以及端口号，这张表形成了程序中的IP池，为爬虫提供代理资源。

表 4-3 proxy表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段编号 | 字段名称 | 字段类型 | 中文名称 | 约束关系 | 备注 |
| 1 | id | int | 主键 | 唯一 | / |
| 2 | ip | varchar(16) | 代理ip | 默认 | / |
| 3 | port | int | 代理端口 | 默认 | / |

1. user\_deal表

该表用于存储user表中数据处理后的结果。

表 4-4 user\_deal表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段编号 | 字段名称 | 字段类型 | 中文名称 | 约束关系 | 备注 |
| 1 | id | int | 主键 | 唯一 | 记录数 |
| 2 | uid | bigint | 微博uid | 默认 | 微博用户ID |
| 3 | province | varchar(10) | 省份名 | 默认 | 微博用户地址 |
| 4 | education | varchar(10) | 受教育程度 | 默认 | 更具用户学校分析后的结果 |
| 5 | source | varchar(100) | 微博来源 | 默认 | 发送微博来源数量最多的设备 |

1. tweets\_deal表

该表用于存储微博内容处理后的结果。

表 4-5 tweets\_deal表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段编号 | 字段名称 | 字段类型 | 中文名称 | 约束关系 | 备注 |
| 1 | id | int | 主键 | 唯一 | 记录数 |
| 2 | uid | bigint | 微博uid | 默认 | 微博用户ID |
| 3 | tid | bigint | 微博ID | 默认 | 对应tweets表中id |
| 4 | text | varchar(500) | 微博内容 | 默认 | 微博记录的微博内容 |
| 5 | retext | varchar(500) | 源微博内容 | 默认 | 转发微博的源微博内容 |

4.4 原型设计

1. 可视化平台用户搜索界面原型如图4.9所示。
2. 根据某一维度数据统计图表展示原型如图4.10所示。

4.5 本章小结

本章项目阐述了整个系统的架构设计以及各个模块的架构设计，每个端使用到的技术栈或者第三方库，通过功能模块图展示了模块之间的关系以及通过流程图展示模块以怎样的方式进行工作，然后设计了系统数据存储所要的表，最后对可视化平台做了原型设计。

5 系统实现

系统实现是项目开发过程中非常重要的一步，它是对项目需求和系统设计的具体贯彻，本项目前端基于D3.js来实现数据可视化，爬虫基于Java编程语言来实现信息提取、处理，Web服务端使用Node.js的Koa框架来实现。

5.1爬虫以及处理系统实现

1. 目录结构

爬虫与处理系统的整体目录结构如下：

├── file

├── out

├── release.ssh

└── src

其中第一个文件夹file为资源目录，里面存放了代理ip地址与端口对应关系的proxy.txt，程序可以通过该文件将代理ip导入至数据库，从而在程序中形成ip池，还存放了文件操作类测试用例生成的测试文件。out文件夹是程序编译打包后的.jar文件生成目录，src是源代码目录，release.ssh是部署脚本，通过命令行将打包后的程序部署到不同的服务器上。

源文件目录结构如下：

├── com

│   └── jiavan

│   └── weibo

│   ├── crawler

│   ├── dao

│   ├── entity

│   ├── portrait

│   ├── test

│   └── util

└── config.properties

1）crawler包：存放了爬虫的爬取逻辑代码，用户数据抓取、微博内容抓取的实现。

2）dao与entity包：微博用户、微博内容等实体以及通过实体对象访问数据库的代码实现。

3）portrait包：通过爬虫抓取后数据进行处理的系统实现，包括使用第三方库Ansj对微博内容进行分词、摘要提取。

4）test包：单元测试。

5）util包：包含底层工具类的实现，比如HttpRequest.java提供了一系列发起网络请求的方法以及Log.java实现不同级别的格式化日志输出。

6）config.properties：存放了数据库连接的配置文件。

1. 代码实现

由于程序实现代码比较多，下面展示了其中比较重要模块的类或者函数的结构，以及重要逻辑的实现。

1. HttpRequest

图5.1为底层工具类HttpRequest.java的主要方法，系统中的每个爬虫模块都会调用下面其中之一的方法来请求URL获取网页内容并解析。

表5-1展示了sendGet方法的部分实现片段，这段代码是该方法发送请求的核心，它基于Java原生API的HttpURLConnection类，实现发送一个HTTP请求，并设置请求头和超时时间，其中trustAllHttpsCertificates方法是用于跳过HTTPS验证，微博的部分接口是HTTPS类型的，但是其证书不受JDK的信任，如果不做处理Java将会抛出异常，所以这里我们让不受JDK信任的HTTPS请求也能成功发送并获得返回结果。

表5-1 sendGet函数实现片段

|  |
| --- |
| URL connectUrl = new URL(url); // Skip https trustAllHttpsCertificates(); HttpsURLConnection.setDefaultHostnameVerifier(hostnameVerifier); connection = (HttpURLConnection) connectUrl.openConnection(); Log.i("Start fetching " + url); for (String key : requestHeaders.keySet()) {  connection.setRequestProperty(key, requestHeaders.get(key)); } // set connect timeout 5s connection.setConnectTimeout(Weibo.CONNECTION\_TIME\_OUT); connection.setReadTimeout(Weibo.CONNECTION\_TIME\_OUT); connection.connect(); |

1. TweetInfo

TweetInfo.java是在crawler包下的一个类，它主要实现了微博内容信息的抓取，该类主要包括如图5.2中的方法。

其中getTweetsByUid函数用于通过一个用户ID来获取该用户的微博信息，并且传入一个布尔类型的参数来指定抓取的方式是通过代理的形式还是直接在当前服务器发起一个HTTP请求，表5-2是该函数实现的主要片段。

表 5-2 getTweetsByUid实现片段

|  |
| --- |
| ArrayList<Tweet> tweets = new ArrayList<>(); int pageNo = 1; while (true) {  String url = Weibo.getTweetListUrl(uid, pageNo++);  String content = isProxy ? HttpRequest.getByProxy(url, Proxy.getProxyPool()) : HttpRequest.get(url, true);  try {  // 解析数据代码段  } catch (Exception e) {} return tweets; |

1. TweetDeal

TweetDeal.java实现了对微博内容的处理，通过使用开源的Ansj库来实现对文本的分词摘要提取，从而获取微博内容的摘要信息，获取摘要信息的函数名为getKeyWords，该方法需要传入2个参数，一个是需要摘要提取的文本信息，这里应该传入的是微博的内容信息，第二个参数是指定提取多少个关键词，具体代码实现如表5-3所示。

表 5-3 getKeyWords方法实现

|  |
| --- |
| public static String getKeyWords(String text, int keywordNum) {  if (text == null || text.equals("")) {  return "";  }  KeyWordComputer kwc = new KeyWordComputer(keywordNum);  Collection<Keyword> keywords = kwc.computeArticleTfidf(text);  String result = "";  int i = 0;  for (Keyword keyword : keywords) {  result += keyword.getName();  if (i < keywords.size() - 1) {  result += ",";  }  i++;  }  return result; } |

1. 系统运行截图

爬虫程序作为一个命令行程序运行在服务器，并将数据采集日志输出到文件系统中，图5-3展示了爬虫采集数据的日志信息。

5.2 Web服务端实现

1. 目录结构

├── index.js

├── node\_modules

├── package.json

├── routes

├── static

└── utils

其中index.js是Web程序启动的入口，node\_modules是所需要的node.js模块，package.json是本项目的配置文件，routes是后端路由，static是静态资源目录也是前端代码存放目录，utils是工具模块库，其中包含了异步操作数据库模块。

1. 代码实现

后端最为主要的功能是提供API服务，其实现代码为api.js，其中实现了包括如图5.4中的API。

表5-4截取了部分Web后端的的代码，来自api.js，其实现了获取用户性别统计概况的接口，将会以JSON的格式将数据返回给前端。

表5-4 获取用户性别分布接口实现

|  |
| --- |
| api.get('/getUserGender.json', async (ctx) => {  let sql = 'select gender, count(gender) as count from user group by gender order by count desc';  let groups = await query(sql);  sql = 'select count(\*) as count from user';  let total = await query(sql);  total = total[0].count;  groups.forEach((item) => {  item.rate = item.count / total;  });  ctx.body = {  total,  groups  };  }); |

5.3 数据可视化实现

1. 目录结构

├── assets

├── index.html

├── script

├── style

├── test

└── tpl

数据可视化平台是一个单页应用，其中index.html是整个应用的访问入口，assets是资源文件夹用户存放如d3所需的一些数据，script为脚本存放目录，style是样式存放目录，test为单元测试目录，tpl是模板存放目录。

1. 代码实现

可视化图形主要由d3-mod下的js模块生成，首先前端先向服务器请求对应区域的API数据，然后通过d3.js来驱动数据生成SVG，由于每个绘制模块的代码量都比较多，这里拿生成用户性别占比图表的getGender.js绘制流程来讲，其具体用数据驱动图表的流程如图5.5所示。

1. 系统运行截图

图5-6展示了搜索用户基本信息模块的实现效果，用户通过输入框输入一个微博用户昵称，后端通过抓取用户数据返回给前端展示。

图5-7展示了抓取的用户数据中毕业高校分布柱状图，其显示的数据为在抓取的全部用户数据中，对用户的毕业院校进行统计，获取全部院校并且统计每一个院校的人数，该图展示的是人数前10 的高校名称，由此我们可以看到在用户集合中人数前三的学校分别是北京大学、清华大学、中国传媒大学。

图5-8展示了抓取的用户数据中性别占比情况。从中我们可以看出男性占比比较大，已经超过了一半。

图5-9展示了用户所处的地理位置占比和排名情况。在我们抓取的用户集合中，已经分布到了中国的每一个省份直辖市以及自治州和港澳台，其中人数占比最多的是北京市，然后是上海和广东，我们可以看到沿海城市或者内陆比较发达的城市人数的占比会比较靠前。

图5-10展示了用户阳光信用的占比情况。在拥有信用记录的用户中占比最大的是信用良好，然后是信用极好，信用较低的人占比极低，说明在整个用户集合中，用户的信用情况还是比较良好的。

5.4 本章小结

本章通过对爬虫以及数据处理模块、Web数据可视化平台系统实现中的代码目录结构、部分代码、以及最终实现效果图进行分析和展示，贯彻执行了系统设计的原则和系统需求的实现。

6 系统测试

6.1 系统测试概述

系统测试以粒度最大，多采用黑盒测试的方法与计算机硬件、数据、程序等元素组合起来一起测试，以验证是否符合项目需求以及系统设计。其目的是为了通过与需求规格的比较发现系统与需求不符合或者有矛盾的地方，从而提出更加完善的解决方案，它的任务是尽可能的找出系统在运作中可能会出现的问题，验证各个模块是否能正常工作，提高系统的稳定性和可靠性，最终完成整个系统的交付。

6.2 测试与分析

1. 可用性测试

可用性测试是保障系统功能正常使用和软件质量的基本要求，是对系统可用性的整体评估，如果能通过可用性测试的测试用例，那么说明系统已经达到满足基本使用的要求。表6-1描述了系统可用性测试的测试用例。

表6-1 可用性测试测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例编号 | 01 | | 系统版本号 | | v0.2.0 |
| 测试环境 | 服务器环境：内存1GB，CentOS 6.5，MySQL 5.5，Java 8，Node.js 7.9，1M带宽  浏览器环境：Chrome 58.0.3029.110 (64-bit) | | | | |
| 前置条件 | 网络连接正常并且稳定 | | | | |
| 测试步骤 | 1. 启动用户爬虫任务 nohup java –jar portrait.jar userinfo & 2. 启动微博爬虫任务 nohup java –jar portrait.jar tweets & 3. 运行数据处理程序。 4. 启动后端服务器。 5. 打开浏览器输入首页访问地址。 6. 随机点击一个数据图表查看按钮。 7. 在搜索用户的输入框查询一个微博存在的用户昵称。 | | | | |
| 输入数据 | 微博用户名 | | | | |
| 预期输出 | 前端以图形的方式正确展示接口返回来的数据 | | | | |
| 实际输出 | 与预期输出相符合 | | | | |
| 问题描述 | 页面在加载的过程中有跳帧的情况发生 | | | | |
| 设计人 | 贾正权 | 设计日期 | | 2017/5/22 | |
| 测试人 | 贾正权 | 测试日期 | | 2017/5/22 | |
| 再次测试人 | 贾正权 | 测试日期 | | 2017/5/23 | |
| 问题修改摘要 | 优化脚本加载过程，减少页面重绘，调整系统界面 | | | | |

1. 功能测试

表6-2描述了用户信息抓取模块的功能测试用例，该模块实现了以一个微博用户ID集合为基础抓取用户基本信息的功能。

表6-2 用户信息抓取模块功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例编号 | 02 | | 系统版本号 | | v0.2.1 |
| 测试环境 | 服务器：内存1GB，CentOS 6.5，MySQL 5.5，Java 8，Node.js 7.9，1M带宽 | | | | |
| 前置条件 | 网络连接正常并且稳定 | | | | |
| 测试步骤 | 命令运行nohup java –jar portrait.jar userinfo & | | | | |
| 输入数据 | 无 | | | | |
| 预期输出 | 爬虫运行一天后数据库user表中应该至少包含3万条用户基本信息记录。 | | | | |
| 实际输出 | 与预期输出相符合。 | | | | |
| 设计人 | 贾正权 | 设计日期 | | 2017/5/24 | |
| 测试人 | 贾正权 | 测试日期 | | 2017/5/24 | |
| 问题修改摘要 | 信息采集正常无需修改 | | | | |

表6-3描述了获取用户性别占比接口的功能测试用例。

表 6-3 获取用户性别占比功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例编号 | 03 | | 系统版本号 | | v0.2.1 |
| 测试环境 | 浏览器环境：Chrome 58.0.3029.110 (64-bit) | | | | |
| 前置条件 | 网络连接正常并且稳定 | | | | |
| 测试步骤 | 打开浏览器在地址栏输入<http://127.0.0.1:3000/api/getUserGender.json> | | | | |
| 输入数据 | 无 | | | | |
| 预期输出 | 页面返回用户集合中性别占比的JSON格式数据 | | | | |
| 实际输出 | { total: 52205, groups: [ { gender: 1, count: 28862, rate: 0.5528589215592377 }, { gender: 2, count: 17311, rate: 0.3315965903649076 }, { gender: 0, count: 6032, rate: 0.11554448807585481 } ] }  与预期输出相符合 | | | | |
| 设计人 | 贾正权 | 设计日期 | | 2017/5/24 | |
| 测试人 | 贾正权 | 测试日期 | | 2017/5/24 | |
| 问题修改摘要 | 接口测试正常无需修改 | | | | |

6.3 本章小结

本章首先阐述了系统测试的概念以及重要性，然后分别从可用性测试和功能性测试两个方面进行了测试。通过测试找出了系统中存在的一些设计细节上的问题，并在后续的版本中得到了修改。

7 总结

本文首先分析了在大数据时代数据在互联网中的实际应用，然后阐述了数据处理和数据可视化的意义，结合自己的软件开发经验，设计了一个从微博用户信息抓取以及处理到数据可视化的系统。

通过对系统需求的详细分析以及系统设计，对每个模块具体功能的实现。其中信息抓取以及处理部分使用了比较传统的Java技术栈来实现整个系统，对比Web端使用新兴的Node技术栈。让我认识到传统与新技术的差别和各方的优劣。以Java实现的爬虫系统为代表，在整个爬虫系统架构设计和实现的过程中，都有很多可选的方案已经被成熟的运用，但是最终实现起来都相对繁琐，开发的周期比较长没有Node.js更加快捷。Web后端以最新的ECMAScript标准来实现整个开发过程，让我觉得整个开发过程是非常敏捷和高效的，但是在处理问题上没传统技术那么多的解决方案，以及稳定性方面还需要大量的实践去验证。

因为时间和技术能力以及硬件资源的关系，本系统还有许多可以值得提升的地方。比如爬虫的信息采集效率，在本系统中通过3个服务器去抓取数据，但是由于微博反爬虫限制，只能不断优化抓取算法来提高效率。之前使用了HTTP代理但是随着微博服务升级为HTTPS，没有其代理服务器资源只能减缓抓取速率。在处理文本的时候使用了开源的库来进行摘要提取，精确度还可以进一步提升，自身在这些方面了解以及学习的技术还不够，以后应该多关注机器学习、数据挖掘、算法等方面的知识。

经过本次毕业设计让我从最开始的需求分析到最后的系统实现，让我在大学学习到的软件工程、程序开发等课程的知识得到了一次综合性的应用。以前在学校或者公司实习过程中更多的是参与到程序实现的这一环节。通过这次整体的软件系统设计也让我认识到了在开发前期需求分析的重要性，同时提高了自己对需求分析、开发流程把控、开发时长评估、系统测试的水平，也让我认识到自身的一些缺点。在以后的工作学习中我应当保持初心，持续学习新的知识，不断提升自身技术水平，迈向职业规划的新台阶。