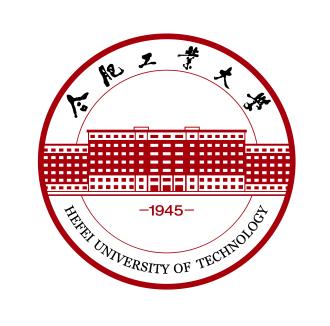
|  |  |
| --- | --- |
| **学 号：** | **密 级： 公开** |



**Hefei University of Technology**

**本科毕业设计（论文）**

**UNDERGRADUATE THESIS**



**类 型：** 设计

**题 目：**基于爬虫的网络信息可视化分析平台

**专业名称：** 软件工程

**入校年份：**

**学生姓名：**

**指导教师：**

**学院名称：**

**完成时间：**

合 肥 工 业 大 学

**本科毕业设计（论文）**

**基于爬虫的网络信息可视化分析平台**

学生姓名：

学生学号：

指导教师：

专业名称：

学院名称：

2021年 05月

**A Dissertation Submitted for the Degree of Bachelor**

**Web Information Visualization Analysis Platform Based on Crawler**

By

Hefei University of Technology

Hefei, Anhui, P.R.China

May, 2021

**毕业设计（论文）独创性声明**

本人郑重声明：所呈交的毕业设计（论文）是本人在指导教师指导下进行独立研究工作所取得的成果。据我所知，除了文中特别加以标注和致谢的内容外，设计（论文）中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 合肥工业大学 或其他教育机构的学位或证书而使用过的材料。对本文成果做出贡献的个人和集体，本人已在设计（论文）中作了明确的说明，并表示谢意。

毕业设计（论文）中表达的观点纯属作者本人观点，与合肥工业大学无关。

毕业设计（论文）作者签名：　　　　签名日期：　2021年　06月　04日

**毕业设计（论文）版权使用授权书**

本学位论文作者完全了解 合肥工业大学 有关保留、使用毕业设计（论文）的规定，即：除保密期内的涉密设计（论文）外，学校有权保存并向国家有关部门或机构送交设计（论文）的复印件和电子光盘，允许设计（论文）被查阅或借阅。本人授权 合肥工业大学 可以将本毕业设计（论文）的全部或部分内容编入有关数据库，允许采用影印、缩印或扫描等复制手段保存、汇编毕业设计（论文）。

（保密的毕业设计（论文）在解密后适用本授权书）

|  |  |
| --- | --- |
| 学位论文作者签名： | 指导教师签名： |
| 签名日期：　　2021年 06月 04日 | 签名日期：　　2021年 06月 04日 |
|  |  |

**摘 要**

随着互联网的流行，社交媒体开始如雨后春笋般涌出，人们也开始越来越关注一些社会热点信息，希望通过一些媒体信息发布平台了解一些事件的发展，但是网络信息庞大且繁杂，很难让我们直观地看到一些社会热点的发展状况，因此网络信息可视化平台应运而生。

本论文就设计并实现了一个基于爬虫的网络信息可视化分析平台，运用爬虫技术爬取指定目标网站的实时热点信息并进行存储和分析，利用可视化技术将分析结果进行直观的展示。该平台的数据获取是基于Scrapy异步爬虫实现，数据抓取速度快，可以满足大量数据的需求，数据处理使用jieba分词再基于SnowNLP实现情感分析，后端基于Django，前端可视化界面则是基于Vue、Element-ui和ECharts实现，数据库则是使用MongoDB。

通过该平台，用户可以方便地了解多个媒体信息发布平台的热点信息及其发展趋势，并可以自定义提交需求。这样用户就不用再苦恼于网络上繁杂的信息，可以更清楚直观地得到自己需要的信息。

**关键词：**网络信息；热点信息；可视化；数据获取；爬虫；

**ABSTRACT**

With the popularity of the Internet, social media has sprung up like mushrooms, and people have begun to pay more and more attention to some social hot information, hoping to understand the development of some events through some media information publishing platforms, but the network information is huge and complicated, and it is difficult to see the development status of some social hot spots directly, so the network information visualization platform came into being.

This dissertation designs and implements a crawler-based visual analysis platform for network information. It uses crawler technology to crawl the real-time hotspot information of the specified target website, stores and analyzes it, and uses visualization technology to visually display the analysis results. The data acquisition of the platform is based on the Scrapy asynchronous crawler, which is fast and can meet the needs of large amounts of data. The data processing uses jieba word segmentation and then realizes sentiment analysis based on SnowNLP. The backend is based on Django, and the front-end visualization interface is based on Vue,Element-ui and ECharts is implemented, and the database uses MongoDB.

Through this platform, users can easily understand the hot information and development trends of multiple media information publishing platforms, and can customize submission requirements.In this way, users no longer have to worry about the complicated information on the network, and can get the information they need more clearly and directly.

**KEYWORDS:** Network information; hot information; visualization; data acquisition；crawler

**目 录**

[1 绪论 1](#_Toc1071240336)

[1.1 课题背景和意义 1](#_Toc1980914351)

[1.2 国内外研究现状 1](#_Toc788517816)

[1.2.1 文本分析研究现状 1](#_Toc497347875)

[1.2.2可视化研究现状 2](#_Toc919381001)

[1.3 课题内容 2](#_Toc891643642)

[1.3 论文结构 3](#_Toc713802328)

[2 项目相关技术综述 4](#_Toc1032074554)

[2.1 Scrapy爬虫 4](#_Toc851612259)

[2.2 jieba分词和SnowNLP 6](#_Toc68729758)

[2.3 MongoDB数据库 6](#_Toc1942324267)

[2.4 Django 7](#_Toc745037422)

[2.5 Vue和Element-ui 7](#_Toc2014289544)

[2.6 ECharts 7](#_Toc1232154700)

[3 系统设计 9](#_Toc639234879)

[3.1 需求分析和总体系统架构 9](#_Toc1907409059)

[3.1.1 需求分析 9](#_Toc188172197)

[3.1.2 功能模块 9](#_Toc1514186595)

[3.1.3 系统架构 10](#_Toc1252885215)

[3.2 爬虫设计 11](#_Toc1164649670)

[3.2.1爬虫流程 11](#_Toc2101044932)

[3.2.2目标网站 12](#_Toc1188564503)

[3.2.3 爬虫配置 12](#_Toc310717527)

[3.2.4 管道配置 13](#_Toc1696730432)

[3.3 数据处理设计 13](#_Toc513022111)

[3.3.1 总体处理流程 14](#_Toc215776872)

[3.3.2 关键字统计 14](#_Toc1609491568)

[3.3.3 情感分析 14](#_Toc1020765764)

[3.4 后端设计 15](#_Toc1910823312)

[3.4.1 数据库设计 15](#_Toc1736947546)

[3.4.2 接口设计 19](#_Toc2132191951)

[3.5 前端可视化页面设计 20](#_Toc690502968)

[3.5.1 页面设计 20](#_Toc281754788)

[3.5.2 请求接口设计 20](#_Toc251280281)

[4 系统实现和成果展示 21](#_Toc1314832765)

[4.1 爬虫实现 21](#_Toc787553725)

[4.2 数据处理实现 22](#_Toc1473739614)

[4.3 后端实现 22](#_Toc65308000)

[4.4可视化前端实现 24](#_Toc267412383)

[4.5成果展示 25](#_Toc1864131557)

[4.5.1 热点排行信息查看 25](#_Toc820152416)

[4.5.2 可视化信息查看 27](#_Toc1751609266)

[4.5.3 爬虫配置 30](#_Toc1591100586)

[5 总结与展望 32](#_Toc1161176458)

[5.1 总结 32](#_Toc1708829317)

[5.2 展望 32](#_Toc1995519488)

[参考文献 34](#_Toc1443919617)

**[致谢](#_Toc1391791819)** [35](#_Toc1391791819)

**插图清单**

[图2. 1 平台工作流程 4](#_Toc294175882)

图2. 2 Scrapy工作流程 [6](#_Toc294175882)

图3. 1 功能需求结构图 [10](#_Toc294175882)

图3. 2 平台系统架构图 [10](#_Toc294175882)

图3. 3 爬虫工作流程 [11](#_Toc294175882)

图3. 4 微博和知乎数据爬取流程 [12](#_Toc294175882)

图3. 5 微博爬取详细流程 [13](#_Toc294175882)

图3. 6 知乎爬取详细流程 [13](#_Toc294175882)

图3. 7 数据处理流程 [14](#_Toc294175882)

图4. 1 爬虫项目结构 [21](#_Toc294175882)

图4. 2 数据处理项目结构 [22](#_Toc294175882)

图4. 3 后端项目结构 [23](#_Toc294175882)

图4. 4 前端项目结构 [24](#_Toc294175882)

图4. 5 热点信息展示 [25](#_Toc294175882)

图4. 6 分页跳转 [26](#_Toc294175882)

[图4. 7 热点趋势变化 26](#_Toc294175882)

[图4. 8 可视化信息队列 27](#_Toc294175882)

[图4. 9 详细微博信息 28](#_Toc294175882)

[图4. 10 批次切换 28](#_Toc294175882)

[图4. 11 词云图 29](#_Toc294175882)

[图4. 12 情感分析图 29](#_Toc294175882)

[图4. 13 情感演变 30](#_Toc294175882)

[图4. 14 设置详情界面 30](#_Toc294175882)

[图4. 15 设置编辑图 31](#_Toc294175882)

[图4. 16 设置更新后 31](#_Toc294175882)

**表格清单**

表3. 1 CountHot表 [15](#_Toc294175882)

表3. 2 WeiboHot表 [16](#_Toc294175882)

表3. 3 WeiboHots表 [16](#_Toc294175882)

表3. 4 ZhihuHot表 [16](#_Toc294175882)

表3. 5 ZhihuHots表 [17](#_Toc294175882)

表3. 6 ResCount表 [17](#_Toc294175882)

表3. 7 ResSpirit表 [17](#_Toc294175882)

表3. 8 WeiboAnalyse表 [18](#_Toc294175882)

表3. 9 WeiboAnalyses表 [18](#_Toc294175882)

表3. 10 Config表 [19](#_Toc294175882)

表3. 11 parse模块接口 [19](#_Toc294175882)

表3. 12 config模块接口 [20](#_Toc294175882)

**1 绪论**

**1.1** 课题背景和意义

随着互联网的迅速发展，互联网已经成为了人们传播和获取各种信息的主要手段。截至2020年12月，我国网民规模达9.89亿，手机网民规模达9.86亿，互联网普及率达70.4%。其中，40岁以下网民超过50%，学生网民最多，占比为21.0%[1]。

疫情背景下，社交媒体的兴起为公众对热点事件的表达和分享提供了便捷化的主题选择和内容自定义途径，以微博、知乎为代表的社交信息发布媒体中的高现势性和高时序性评论数据作为舆情信息基底，有效支撑了演化态势预测、舆论监控引导及事件预警防控等主题研究[2]。与此同时，面对各大社交媒体网站中爆炸的相关数据信息，如何进行合理的数据提取、清洗和分析显得十分重要，尤其是面对不同社交媒体，信息的展示方式也会有所不同，如何实现对这些信息的精确化研究也十分关键。

近些年来，随着爬虫技术进步，越来越多的爬虫解决方案，让人们可以更加快捷方便地获取到指定的网络数据，同时，基于文本语义的研究也越来越成熟，让人们可以更多地获取到包含在大量网络信息中的语义信息。但是目前，从数据获取到存储再到分析及其可视化这一工程化的平台却很少。

因此，本论文设计并实现了一个基于爬虫的网络信息可视化平台，使用爬虫技术实现对主流社交媒体平台网络信息的抓取，再通过一些数据处理方法实现数据的处理和分析并结构化存储，然后利用前端技术实现可视化展示。整个平台包括爬虫配置、数据管理、可视化展示等多个结构化模块，可以实现对网络热点信息的获取和展示。

**1.2** 国内外研究现状

1.2.1 文本分析研究现状

从20世纪80年代开始，主要经历了可行性探讨的初级阶段，辅助分类系统的过渡阶段和自动分类系统的发展阶段，目前文本分类技术慢慢被应用到各个领域中去，包括新闻、购物、社交媒体等。另外，英文文本分类和中文文本分类还是有不少差别的，首先，英文文本的单词是通过空格来隔开的，而中文文本则没有空格来区分，要想读取并分析中文文本，首先就需要对其进行分词。另外，相较于语义分析，英文文本更倾向于句法分析，而中文文本则恰好相反，更倾向于语义分析，这使得对中文文本的处理变得更为困难[3]。

目前，基于中文的文本分析方法主要有两种，一种是基于统计学的文本分析策略，通过一些基于机器学习的算法，如朴素贝叶斯、Logistic Regression、Support Vector Machines等实现文本的分类；另一种是基于深度学习的文本特征提取和分类方法，通过卷积神经网络进行文本特征提取，可以大大降低文本特征的提取难度，并且可以提高模型的训练效率，更清楚准确地表达语义信息[4]，如Long-Short Term Memory等。

**1.2.2可视化研究现状**

最近几年以来以人工智能为代表的热潮,非常大地促进了大数据环境下人工智能的发展,使得数据的使用趋向于平民化。 这也使得了数据可视化与机器学习的相关理论进行进一步的交互融合[5]。

目前，国内的许多大学也针对数据可视化建立了相关的研究团队，如北京大学可视化与可视化分析研究组、浙江大学可视化分析研究组等，同时一些互联网公司也开发了很多用于可视化的框架，如百度的ECharts、蚂蚁金服的AntV等。这些都不断推动着可视化研究的发展，也开始让更多人开始重视可视化技术。

随着数据可视化工具研究的发展，国内许多学者也都撰写了关于大数据可视化的总结论文。通过分析支持大数据可视化分析的基本理论，包括支持认知理论的分析过程，信息可视化理论，人机交互的用户界面，并在此基础上讨论的大数据信息可视化技术应用于主流：面向文本的、网络、时间和空间、多维可视化技术。通过这些可视化技术，可以极大地帮助我们了解数据的各方面信息，同时与人工智能等技术的结合，实现智能科学和可视化的交融，可以更加高效地提升可视化的效率，完善可视化信息的表达。

**1.3 课题内容**

本课题的主要目标是实现对一些主流信息发布网站数据的爬取，获取实时的信息数据，同时对网络信息进行分析聚类然后进行可视化的直观展示，更快捷直观的了解热点问题，并且可以通过定向数据爬取，更具体的了解某些热点信息的发展趋势。

**1.3 论文结构**

论文共分为 5 章，具体安排如下：

第 1 章简要介绍毕业设计课题的背景和意义，以及国内外研究现状，同时还介绍了毕业设计论文的主要结构；

第 2 章简要介绍毕业设计中各个项目所用到的相关技术；

第 3 章主要介绍平台系统各个项目的详细设计；

第 4 章主要介绍平台系统的实现和最终成果展示；

第 5 章简单总结全文所展开的毕业设计工作。

**2 项目相关技术综述**

本项目主要由四大模块组成，首先是用于网络数据获取的爬虫模块，使用到目前比较流行的异步爬虫框架Scrapy，然后是对数据进行清洗和分析的模块，最后是用于平台管理的后端和可视化展示的前端模块，其中后端基于Django，前端基于Vue，可视化使用Echarts，具体平台工作流程如下图2.1：

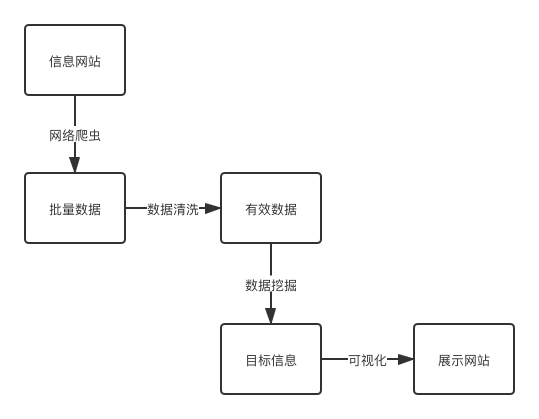


图 2. 1 平台工作流程

**2.1** Scrapy爬虫

Scrapy是一套异步处理框架，是纯python实现的爬虫框架，其基于Twisted网络引擎，支持很多传输层和网络层协议。Scrapy框架中集成了很多常见爬虫的配置，用户不需要再编写和封装很多重复的流程式代码，只要专注于业务处理模块就可以轻松地实现一个爬虫，用来抓取网页内容或者其它各种数据。

Scrapy框架主要由五大部分组成，它们分别是调度器、下载器、爬虫和实体管道、Scrapy引擎。下面我们分别介绍各个部分的作用。

**(1) 调度器（Scheduler）**：

调度器，相当于一个URL（抓取网页的网址）的优先队列，由它来控制下一步要抓取的网址是什么，同时去除重复的网址，防止造成重复抓取。并且用户可以根据自己的需求定制调度器，实现更加灵活的爬取。

**(2) 下载器（Downloader）：**

下载器，是所有组件中需求最大的部分，它根据调度器指派的URL高速地下载网络上的资源。Scrapy下载器代码不复杂，但效率很高，就是因为Scrapy下载器是建立在twisted这个高效的异步网络引擎上的，可以灵活的调节并发量。除此之外。通过Downloader中间件，用户也可以实现对爬虫请求的封装，规避一些反爬虫措施，同时也可以集成Selenium实现动态渲染数据的爬取。

**(3) 爬虫（Spider）:**

爬虫，是用户最需要关注的部分，因为用户爬虫的业务逻辑主要在这个部分进行实现。用户需要在这个部分定制自己的爬虫，可以选择正则表达、Xpath或者其它的网页选择器，用于从指定的网页中提取自己需要的信息，然后封装成自定义的实体（Item）。 用户也可以从中提取出链接，然后加入到调度器中，让Scrapy继续抓取下一个页面。

**(4) 实体管道（Item Pipeline）：**

实体管道，用于处理爬虫（spider）提取的实体，也就是在Item中定义的数据实体。主要的功能是实现实体持久化、验证实体的有效性以及清除一些不需要的信息。同时，用户也可以利用管道实现数据库的连接和数据保存等功能。

**(5) Scrapy引擎（Scrapy Engine）：**

Scrapy引擎是整个框架的核心。它用来控制调度器、下载器、爬虫等，来保证整个项目的有序运行。实际上，Scrapy引擎就相当于计算机的CPU，它控制着整个项目的流程，保证每个爬虫的运转。

Scrapy框架的每个部分都是一个组件，相互配合保证项目运行，其具体的工作流程如下图2.2：

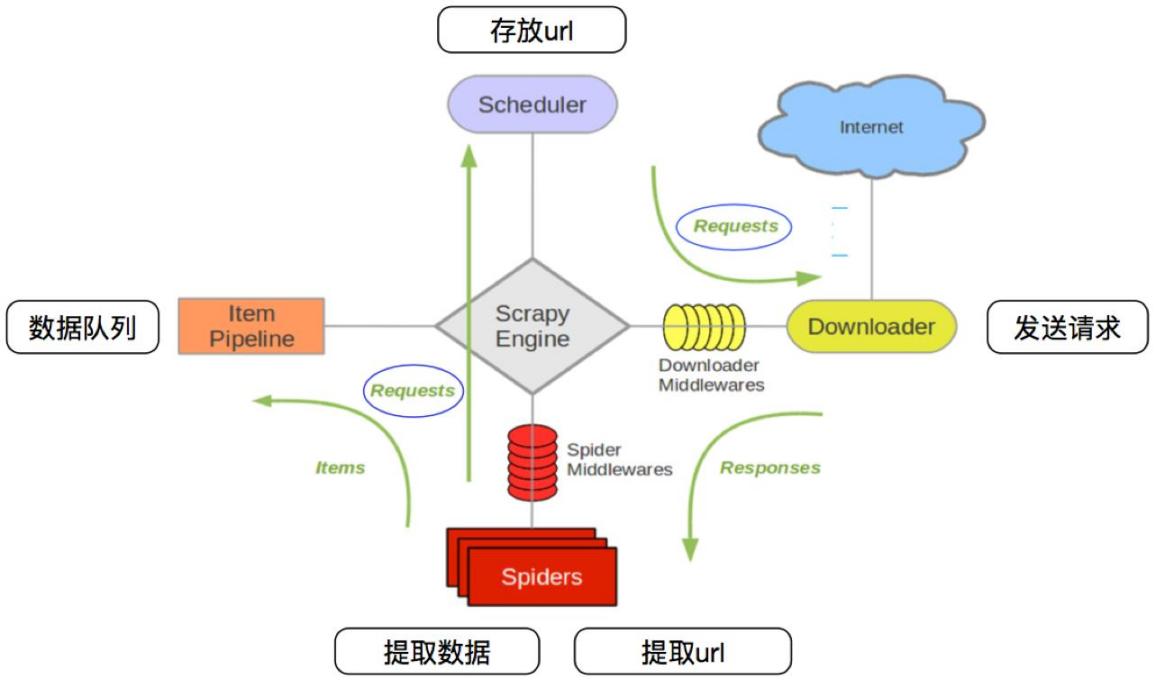


图 2. 2 Scrapy工作流程

**2.2 jieba分词和SnowNLP**

因为本论文中数据来源主要是国内的一些社交媒体网站，所以文本主要面向中文。而jieba分词是一款非常流行中文开源分词包，具有高性能、准确率、可扩展性等特点。除了分词功能，该框架还有词性标注、关键词抽取等功能，并且支持自定义词典，可以针对不同的场景，自定义合适的词典，使分词的效果更好。

SnowNLP是国内的一个python类库，可以方便的处理中文文本内容，是受到了TextBlob的启发而写的。SnowNLP主要可以进行中文分词、提取文本关键词、提取摘要、情感分析、词性标注、文本分类、转换拼音、繁体转简体、分割句子等。其中中文分词算法是基于Character-Based Generative Model，文本分类原理是朴素贝叶斯，提取关键字的原理是TextRank，提取摘要的原理是TextRank。相比于jieba分词，SnowNLP可以针对分词结果进行情感分析和文本分类，并且可以基于自己的数据进行模型训练，提高识别的准确度。

**2.3** MongoDB数据库

MongoDB是一个开源、无模式的、高性能的并且基于分布式文件存储的文档型数据库，其由C++语言编写，是NoSQL数据库的一种。

相对于传统的关系型数据库，MongoDB数据库面向集合存储，易存储对象类型的数据，并且内置GridFS，支持大容量的存储，非常适合分布式系统，同时对于非索引字段的查询，MongoDB的性能更加优越。

**2.4 Django**

Django 是基于Python 编写的一个开源Web 应用框架。使用Django，不需要编写很多的代码就可以开发出功能比较全面的网站，并且Django有很多的第三方支持库，让Django的功能也更加的丰富。Django本身基于MTV模型，即 Model（模型）+Template（模板）+ View（视图）的开发设计模式，与传统的MVC模型相比，MTV的视图层则用来处理请求和数据，然后与模型层通信，取代了MVC中控制层的作用。使用MTV开发模型可以极大地方便后期对网站的更新迭代，同时也可以简化很多重复程序的编写。

因为平台数据库使用MongoDB这一数据库，Django原生的ORM映射不支持，所以需要使用第三方的库Djongo，使用这一库，可以使用Django原生的数据库映射方式实现文档结构数据的增删查改，极大的提高了数据库维护的便利性和后端接口开发的效率。

**2.5 Vue和Element-ui**

Vue是一套用于构建用户界面的渐进式框架。与其它的一些大型框架不同的是，Vue被设计为可以自底向上逐层应用，先在基础层搭建好框架，再逐层添加新的功能。

Vue基于MVVM模型，实现了View和Model的双向绑定，同时也实现了数据和视图的分离，通过数据变化驱动视图变化。因此Vue的核心也在视图层，十分易于上手，同时还便于与第三方库或既有项目整合，大大提升开发效率。另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue也完全能够为复杂的单页应用提供驱动。

对于前端框架Vue，其内置的一些组件样式十分单一，但是如果要自己编写组件就会很麻烦，不仅要自己编写模板和样式，还要自己添加事件，处理数据，所以饿了么推出了基于Vue的组件库，也就是Element-ui，其提供了丰富的前端组件，并且可以在Vue项目中十分方便的使用，极大地方便了我们前端界面的构建，提高了

开发效率。

**2.6 ECharts**

ECharts，一个使用JavaScript实现的开源可视化库，最初由百度的前端团队开发，后面捐赠给Apache基金会。ECharts可以流畅地运行在PC和各种移动设备上，并且兼容当前绝大部分浏览器，如IE8/9/10/11、Chrome、Firefox、Safari等，其底层依赖于轻量级的Canvas类库ZRender，基于BSD开源协议，提供了直观、生动、交互丰富，可高度定制化的数据可视化图表。除了已经内置的一些功能丰富的图表，ECharts 还提供了自定义接口，用户可以根据接口自定义自己需要的可视化图表。在ECharts官网还提供了很多图表样例和教程，对新手十分友好。

**3 系统设计**

3.1 需求分析和总体系统架构

**3.1.1 需求分析**

基于爬虫的网络信息可视化平台，主要用于网络热点信息的爬取、数据清洗和分析以及可视化展示，通过平台后台可以实现热点信息获取、数据爬取任务管理以及可视化结果查看等操作。综上所述，本平台应包含如下几个功能：

**（1）数据爬取：**可以根据爬虫任务队列爬取指定网站的热点信息数据，并在后台启动定时任务，定时爬取指定信息内容并存储到数据库中。

**（2）数据清洗：**针对爬取的数据，去除不需要的数据内容，如网页节点信息、图片URL、各种特殊字符等等。

**（3）数据分析和存储：**针对批量爬取的大量数据进行适当的处理分析，然后将分析结果结构化保存到数据库中。

**（4）爬虫任务管理和简单配置：**用户可以通过平台管理界面进行爬虫任务队列管理，管理需要爬取的任务，并根据已经爬取的内容进行动态管理，同时也可以根据平台管理界面查看爬虫运行状态，并进行基础的爬虫配置管理，如cookies、headers等。

**（5）可视化结果查看：**用户可以通过该平台实时的查看当前各个媒体平台的热点信息，并可以针对具体的热点信息，查看可视化分析的结果。

**3.1.2 功能模块**

根据需求分析，将本系统分为四大功能模块，分别是爬虫模块、数据处理模块、后端接口模块、前端可视化展示模块，其功能需求如下图3.1：

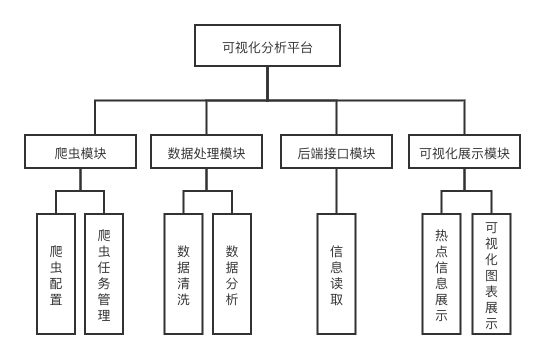


图3. 1 功能需求结构图

**3.1.3 系统架构**

基于以上需求分析和功能模块，可以将整个系统划分为四大部分，分别为爬虫、数据处理、后端系统、前端系统，每个部分单独创建项目，然后通过定时任务进行爬虫任务和数据处理的启动，然后前后端系统实现与数据库的数据交换。以下为系统架构图3.2：

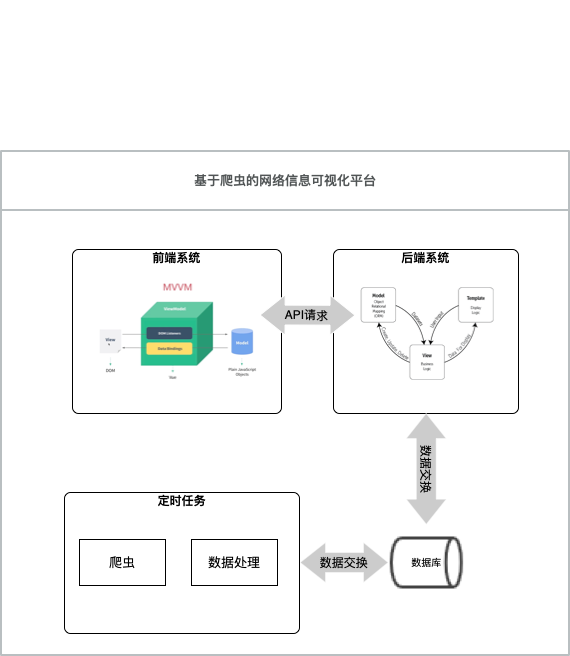


图3. 2 平台系统架构图

**3.2 爬虫设计**

爬虫模块的主要任务是根据数据库中存储的爬虫任务队列和设置的目标网站进行批量的自动化爬取。同时，针对不同的目标网站，要制定不同的爬取策略和爬取配置，并将结果结构化的存储到数据库中。

**3.2.1爬虫流程**

本项目的爬虫是在Scrapy爬虫框架的基础上进行构建的，所以爬虫设计是基于Scrapy框架，主要流程描述如下：

● 调度器先把Request对象发送到Scrapy引擎，然后通过下载中间件将请求送到下载器；

● 下载器发送URl请求，获取响应结果后通过下载中间件再发送到Scrapy引擎，然后通过爬虫中间件送到爬虫；

● 爬虫从响应结果中提取新的URL地址，然后封装成Request对象，再通过爬虫中间件送到Scrapy引擎，再送到调度器；

● 爬虫提取数据后通过Scrapy引擎再发送到管道

● 管道进行数据的处理和保存

详细流程如下图3.3：

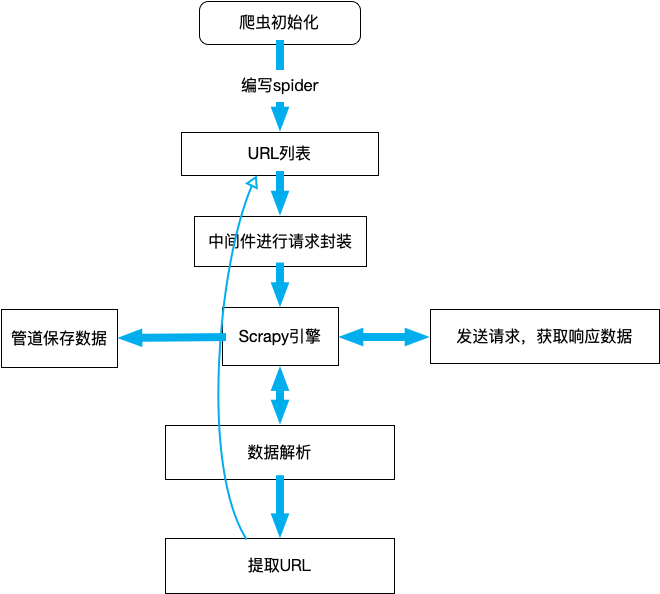


图3. 3 爬虫工作流程

**3.2.2目标网站**

本论文中爬虫的目标网站主要有两个，一个是微博，一个是知乎，这两个都是目前比较主流的媒体信息发布平台。对于微博，信息的主要呈现方式是通过博文和评论，所以我们爬虫的设计要针对关键字搜索的多个博文进行爬取，再针对相应博文下的评论进行爬取，最后再整合提交数据到管道中；而对于知乎，信息主要呈现方式是问题和相对应的回答以及回答下的评论，所以对于爬虫的设计要相对应的改变，增加爬虫爬取的深度，下面是微博爬虫和知乎爬虫的工作流程图3.4：

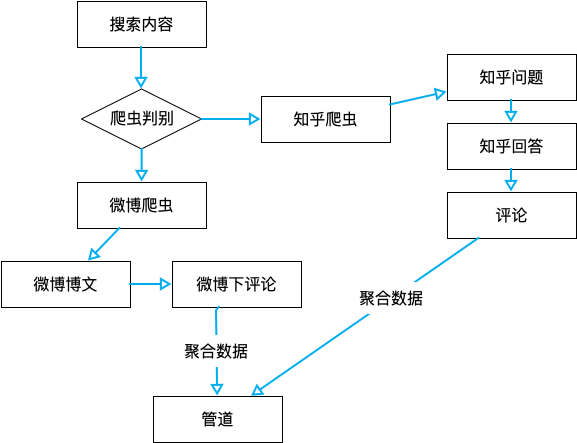


图3. 4 微博和知乎数据爬取流程

**3.2.3 爬虫配置**

目前，各大网站针对爬虫都有一定反爬策略，所以我们需要针对不同网站进行一定的爬虫设置，在爬虫引擎根据URL发出请求前我们需要在中间价中对请求进行封装，比如引入Selenium、添加headers、添加cookies、添加验证token等。

针对不同的目标网站，我们也要进行不同的爬虫配置，比如微博，它的反爬关键在于模糊URL，无法通过指定URL进行微博UID的获取，所以我们要先利用中间件通过Selenium获取相对应的微博UID，然后再根据具体的UID构造评论的URL，所以，需要在Middleware中引入Selenium，并且在Spider中实现评论URL构造，具体流程如下图3.5：

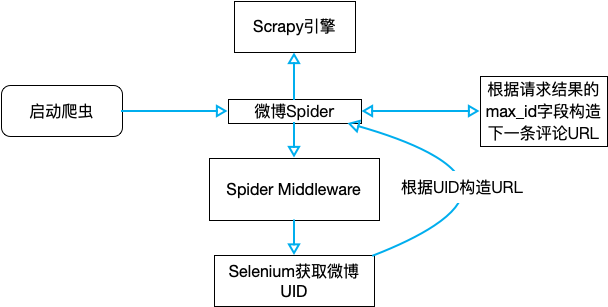


图3. 5 微博爬取详细流程

对于知乎，现在使用的是x-zse-86的加密方案，在spider发送请求前，我们需要根据搜索内容构造URL，并通过JS逆向实现对URL的加密，然后在headers中添加验证字段，知乎后台验证成功后才会将数据发送过来，详细流程如下图3.6:

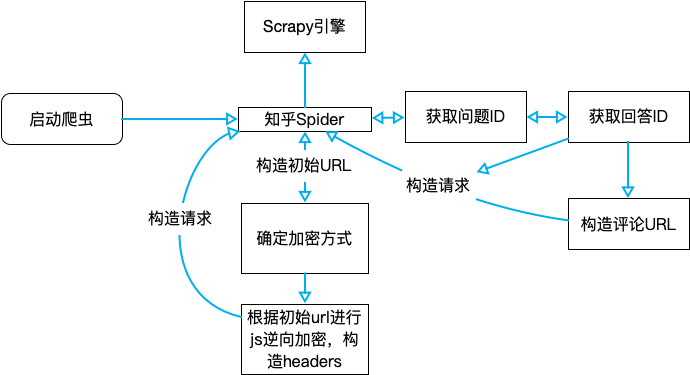


图3. 6 知乎爬取详细流程

**3.2.4 管道配置**

爬虫爬取的数据需要进行一定的处理，并将其存储到MongoDB数据库中，所以需要在Scrapy的Pipeline中配置数据库的连接和结果的存储方式，并实现根据不同的爬虫，向不同的集合中存储数据。

**3.3 数据处理设计**

**3.3.1 总体处理流程**

数据处理模块的主要任务是读取数据库中指定内容和轮次的数据，然后进行简单的数据处理，接着使用基于统计学的分析方法，对大量的评论数据进行分析，并将结果结构化的存储到数据库中，用于后面可视化的展示，总体的流程如下图3.7：

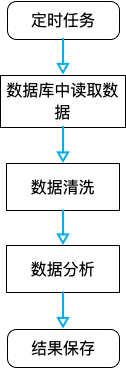


图3. 7 数据处理流程

**3.3.2 关键字统计**

从数据库中读出来的数据大多是评论，有短评论和长评论。对于长评论，为了统计关键字，我们需要先进行句子划分，然后再进行简单句分词，最后再根据分词结果，使用Collections中的Counter进行词频率统计，再和相关的数据进行整合。

**3.3.3 情感分析**

对文本进行统计学的情感分析，基于SnowNLP实现，其情感预测的基本模型是贝叶斯模型。对于有两个类别和*c*1的分类问题来说，其特征为*w*1⋯*w*n，特征之间是相互独立的，属于类别*c*2的贝叶斯模型的基本过程为：

/private/var/folders/5n/3wc8v5z11z940lqnc2fc4ky00000gn/T/com.kingsoft.wpsoffice.mac/wpsoffice.tylwfjwpsoffice （3.1）

其中：

/private/var/folders/5n/3wc8v5z11z940lqnc2fc4ky00000gn/T/com.kingsoft.wpsoffice.mac/wpsoffice.Sswahpwpsoffice （3.2）

先根据训练的数据集中的正负样本进行模型训练，得到一个情感分类器，然后再针对要分析的数据进行相应的分词、去停用词操作，再通过分类器预测情感结果。和上面词频统计一样，需要针对长评论进行语句划分，然后再针对每一句进行情感预测，最后结果取平均值，与其他相关数据进行整合。

**3.4 后端设计**

后端基于Django进行实现，Django基于MTV模式，对于Model层，需要先在model中定义好ORM映射的字段和数据库表的一些配置，这样当App初始化时就会自动在数据库中生成相对应的表，因为Django的原生ORM不支持MongoDB，所以需要使用第三方库Djongo实现ORM映射。后端接口也要根据前端需要的数据进行设计，主要设计两个APP，一个是Config用于爬虫管理，一个是Parse用于可视化信息传输。

**3.4.1 数据库设计**

传统的关系型数据库，在model里面可以通过表之间一对一、一对多、多对多的关系进行存储，但是对于MongoDB这一关系型数据库，数据都是结构化的对象存储，所以我们需要使用Djongo中的EmbeddedField和ArrayField来实现集合的嵌套存储，主要设计的表如下：

**（1）热点信息表**

CountHot表用来记录当前各个网站所爬取热点信息的个数，用于后期的统计和从热点表中读取最新的热点信息。其具体如下表3.1所示：

表3. 1 CountHot表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 备注 |
| \_id | ObjectField | id |
| parse\_type | CharField | 类型 |
| count | IntegerField | 计数 |

WeiboHot表用来记录单条热点信息，WeiboHots用于记录多条热点信息，后者包含前者，基于Djongo的ArrayField用来实现热点内容聚合存储。其具体如下表3.1和3.2所示：

表3. 2 WeiboHot表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 备注 |
| ranking | CharField | 排名 |
| content | TextField | 内容 |
| count | CharField | 热度 |
| desc | CharField | 热度描述 |

表3. 3 WeiboHots表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 备注 |
| \_id | ObjectField | id |
| time | CharField | 爬取时间 |
| count | IntegerField | 排行计数 |
| hots | ArrayField | 多个排行信息 |

ZhihuHot表用来记录单条热点信息，ZhihuHots用于记录多条热点信息，后者包含前者。其具体如下表3.4和3.5所示：

表3. 4 ZhihuHot表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 备注 |
| ranking | CharField | 排名 |
| content | TextField | 内容 |
| count | CharField | 热度 |

表3. 5 ZhihuHots表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 备注 |
| \_id | ObjectField | id |
| time | CharField | 爬取时间 |
| **count** | IntegerField | 排行计数 |
| **hots** | ArrayField | 多个排行信息 |

**（2）分析结果表**

主要用来记录情感分析结果的表，需要几个表的聚合，主要记录每个关键字对应多条微博评论的分析结果，包括词频统计、感情正负结果统计。

其中ResCount用来记录词频统计结果，ResSpirit用来记录情感分析结果，其具体如下表3.6和3.7所示：

表3. 6 ResCount表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 备注 |
| name | CharField | 关键词名称 |
| value | IntegerField | 关键词数目 |

表3. 7 ResSpirit表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 备注 |
| positive | IntegerField | 正向情感数目 |
| negative | IntegerField | 负向情感数目 |

WeiboAnalyse用来记录单条微博分析结果，WeiboAnalyses用来记录相同关键字多条微博的分析结果，其具体如下表3.8和3.9所示：

表3. 8 WeiboAnalyse表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 备注 |
| mid | CharField | 微博标识号 |
| content | TextField | 微博内容 |
| comments\_count | IntegerField | 评论数 |
| res\_count | ArrayField | 词频统计结果 |
| res\_spirit | ArrayField | 情感分析结果 |

表3. 9 WeiboAnalyses表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 备注 |
| \_id | ObjectField | id |
| search\_content | CharField | 搜索内容 |
| search\_patch | IntegerField | 搜索轮次 |
| parse\_time | CharField | 爬取时间 |
| weibos | ArrayField | 多条微博分析结果 |

**（3）爬虫配置表**

主要用于记录爬虫的一些配置信息，如headers、cookies、爬虫类型等。其具体如下表3.10所示：

表3. 10 Config表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 备注 |
| \_id | ObjectField | id |
| parse\_type | CharField | 爬虫类型 |
| headers | TextField | 爬虫headers |
| cookies | TextField | 爬虫cookies |
| status | CharField | 配置生效状态 |

**3.4.2 接口设计**

根据前面划分的两个APP， 每个APP里面再针对具体的业务功能进行接口的设计，主要的后端接口如下：

**（1）parse模块**

parse模块主要接口包括热点信息获取、爬虫队列获取、搜索内容获取、分析结果获取、可视化结果获取等。具体如下表3.11所示

表3. 11 parse模块接口

|  |  |  |  |
| --- | --- | --- | --- |
| 请求方式 | URL | 参数 | 备注 |
| get | /parse/hots | hot\_type（热点类型）、count（热点批次） | 获取热点信息 |
| get | /parse/hotschange | date（日期） | 查询的点变化信息 |
| get | /parse/search | search\_type（搜索类型） | 获取搜索列表 |
| get | /parse/getsearch | content（搜索内容）、search\_type（类型） | 获取搜索内容 |
| get | /parse/detail | content、patch、detail\_type | 获取分析结果 |
| get | /parse/detailchange | content、mid、time | 获取可视化结果 |

**（2）config模块**

config模块主要接口包含了配置读取和配置修改功能，具体如下表3.12所示：

表3. 12 config模块接口

|  |  |  |  |
| --- | --- | --- | --- |
| 请求方式 | URL | 参数 | 备注 |
| get | /config/request | 无 | 获取配置信息 |
| post | /parse/request | 配置信息 | 修改配置信息 |

**3.5 前端可视化页面设计**

**3.5.1 页面设计**

根据业务功能，主要分为三大页面，一个是爬虫配置页面，负责查看和修改爬虫配置；一个是可视化信息查看页面，负责查看当前以及爬取的热点数据及其可视化信息；还有一部分是查看各个平台实时的热点信息排行。

对于前端界面，主要分为三大块，上半部分为导航栏，负责管理登录和通知，界面左侧为菜单栏，负责页面的跳转，右侧为主体展示界面，负责展示各种需要的信息。

**3.5.2 请求接口设计**

前端请求使用Axios进行请求封装，根据后端接口进行设计，为了抽取配置，在api文件夹中统一请求格式并封装成函数，在具体的业务逻辑中直接调用。

**4 系统实现和成果展示**

**4.1 爬虫实现**

爬虫项目项目结构如下图4.1：



图4. 1 爬虫项目结构

爬虫模块是基于Scrapy框架进行构建，在spider中针对不同的爬虫任务创建爬虫，如hot（热点信息爬取）、weibo（微博信息爬虫）、zhihu（知乎信息爬虫），然后根据具体任务的爬虫设计编写爬虫业务代码，然后在middleware中封装请求和添加Selenium模块实现动态数据抓取，最后在pipeline配置数据库连接并针对不同爬虫实现数据整合和保存。

爬虫模块的核心任务在于业务逻辑的编写，尤其是针对一些网站的反爬措施，需要进行灵活的变更，比如对于知乎网站，要解决js逆向的问题，就必须根据搜索关键字进行加密，所以在爬虫的业务逻辑中需要针对加密方式编写相应的加密代码，然后配合爬虫进行数据获取。

**4.2 数据处理实现**

数据处理项目自己构建，项目结构如下图4.2：

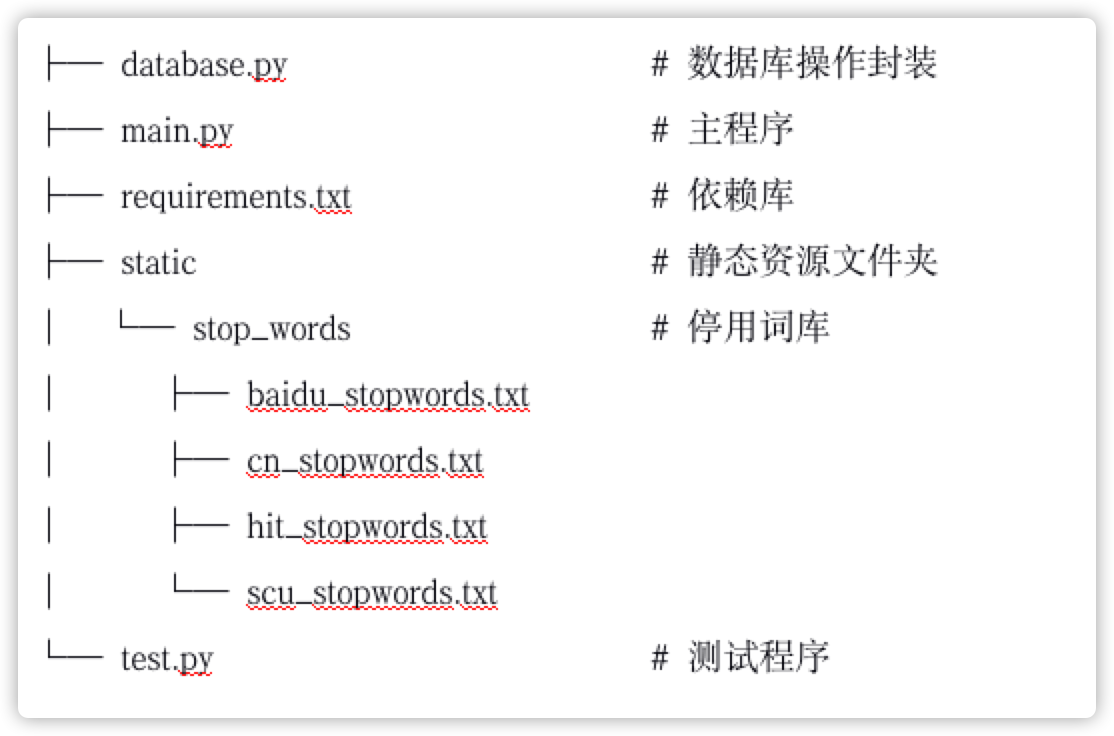


图4. 2 数据处理项目结构

数据处理模块的实现示基于jieba分词库和SnowNLP，首先要选取合适的停用词库，提高分词的准确率， 这里选取了多个常用词库并进行筛选合并，最终整合了一个较为全面的停用词库。对于词频统计，是在分词的基础上，使用python的collections模块的Couter进行统计；对于情感分析，利用SnowNLP的情感分析模块实现。

**4.3 后端实现**

后端项目结构如下图4.3：

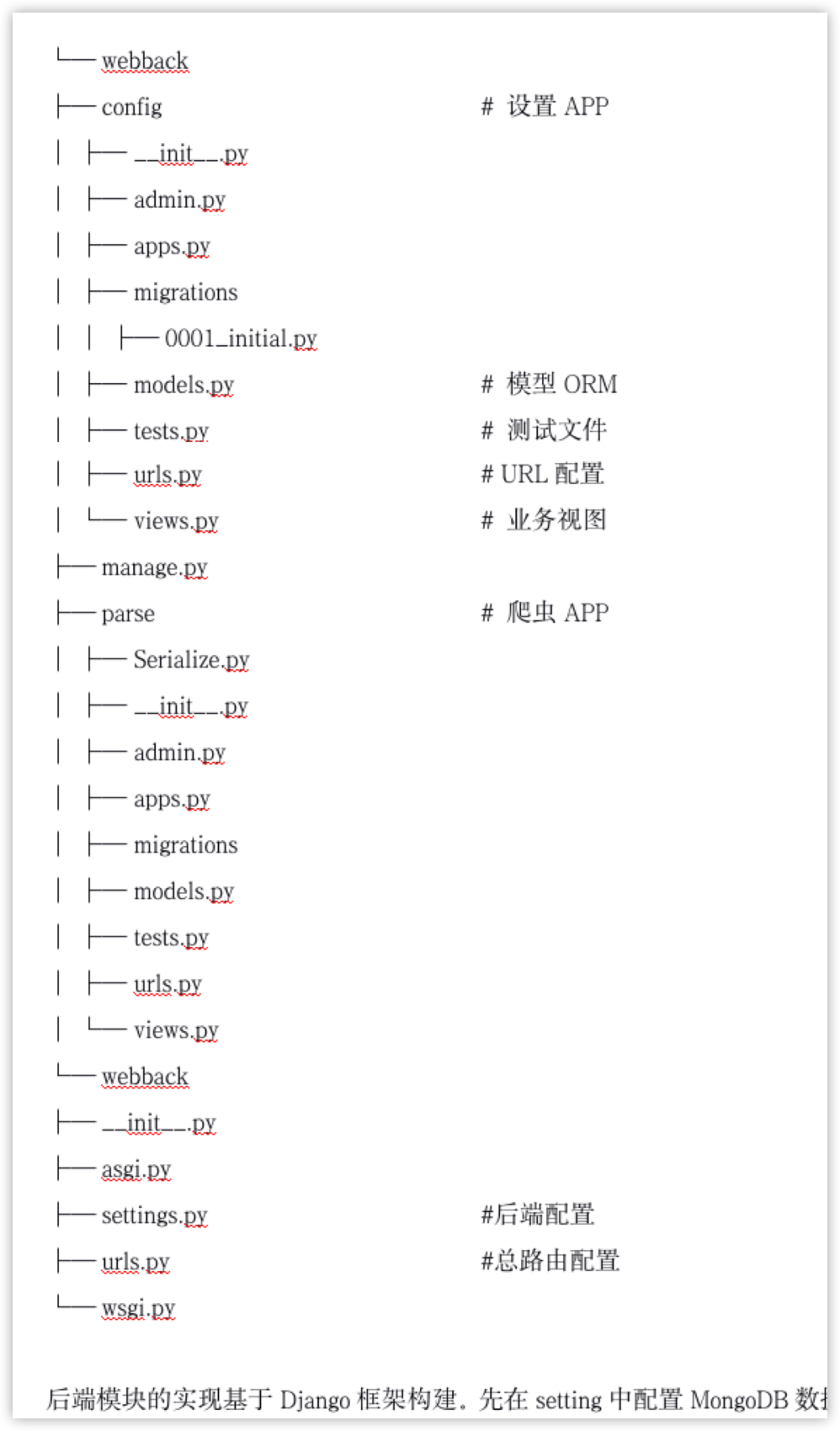


图4. 3 后端项目结构

后端模块的实现基于Django框架构建。先在setting中配置MongoDB数据库，然后根据后端设计的模块创建parse和config两个app。接着在不同的app中根据数据库设计的表进行model编写实现数据库的映射，再根据api进行路由编写，最后再根据api数据需求编写view业务处理代码。

**4.4可视化前端实现**

前端项目结构如下图4.4：

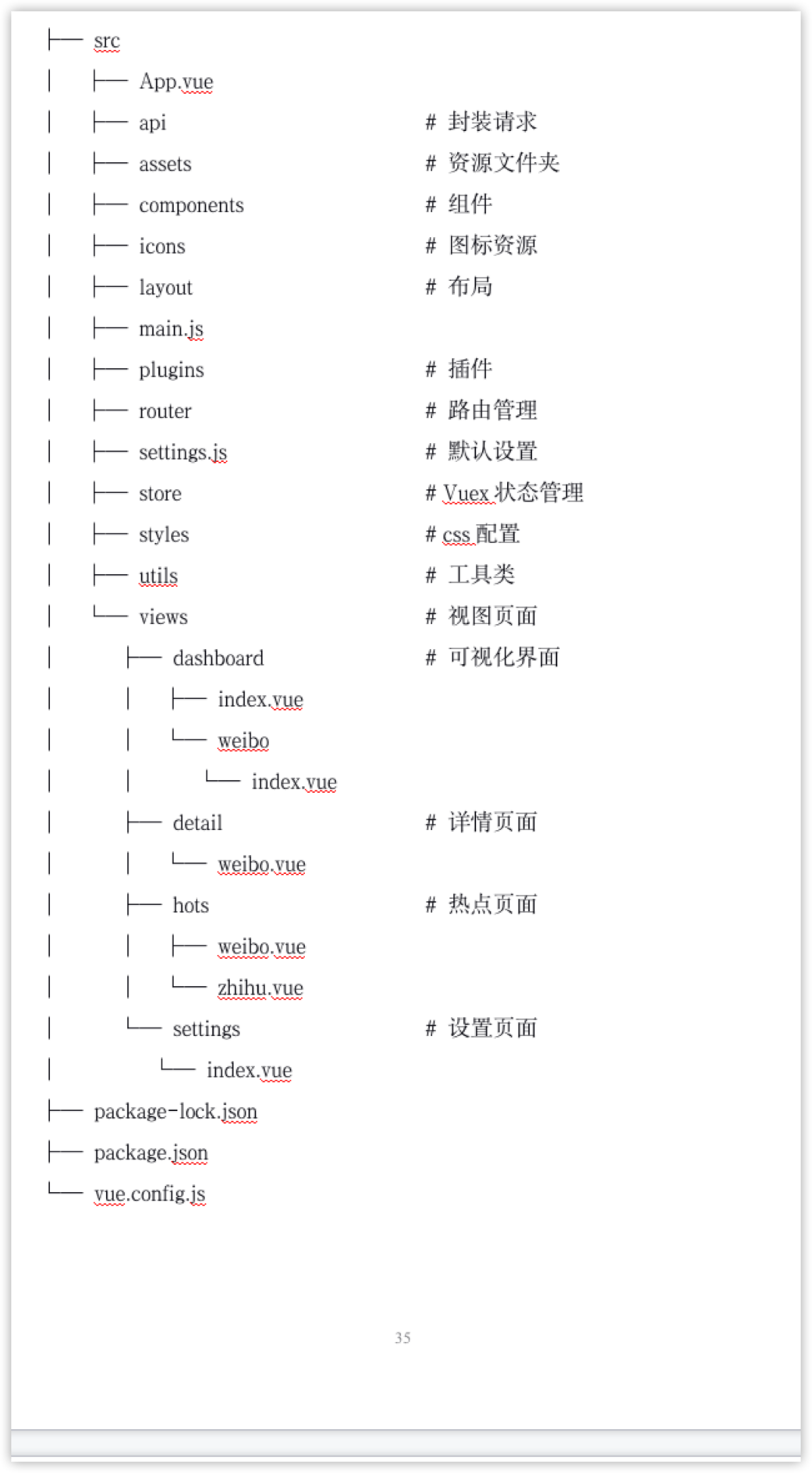


图4. 4 前端项目结构

前端模块的实现基于Vue框架，并引入Element-ui和Echarts。其中在api中主要编写对应后端接口的axios请求；在components中主要编写了一些常用的组件，比如一些图表组件；在views中编写对应的页面，主要有dashboard、hots、detail、settings四个页面；最后在router中配置页面路由用来管理页面跳转。

**4.5成果展示**

本平台可展示的内容主要是可视化的前端界面，主要内容有热点排行的查看、可视化信息的展示、爬虫配置三大功能。界面主体分为三大部分，左侧菜单栏可以用来切换功能界面，上侧是页面导航和登录退出功能栏，右侧是主体展示部分。

**4.5.1 热点排行信息查看**

进入平台后，左边菜单栏的第一行功能就是热点排行，点击后默认进入微博排行查看，可以通过下滑菜单进行切换到知乎，展示热点信息如下图4.5：

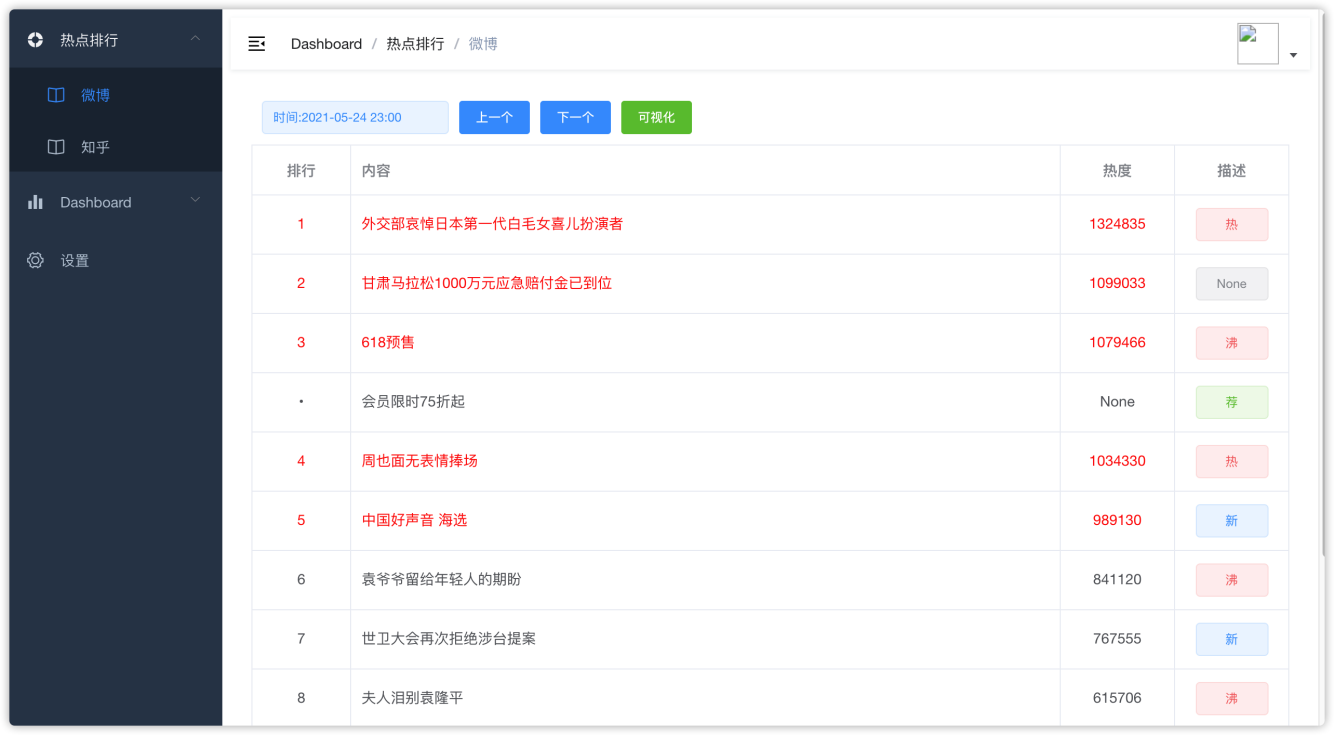


图4. 5 热点信息展示

可以看到热点信息包括排行、内容、热度和对应的描述信息，其中重要的热点信息会被标红，描述信息也会根据内容改变颜色。

往下滑动可以进行分页跳转，并且跳转后自动回到第一条，如下图4.6：



图4. 6 分页跳转

热点信息排行上面会有更新时间和切换按钮，可以切换显示上一时间段和下一时间段的热点信息排行。 因为后台爬虫的定时时间为1小时，所以切换的时间间隔也为1小时，默认进入的是最新时间段的热点信息。

热点信息上部最右侧有可视化功能的按钮，点击后会跳转到热度变化的可视化界面，通过图表展示最近一段时间热点信息的变化情况，具体如下图4.7：



图4. 7 热点趋势变化

主要反映的热点信息前三名的变化趋势，鼠标移动到相应时间段时会具体显示热点信息的具体内容。

**4.5.2 可视化信息查看**

通过菜单栏可以切换到可视化信息查看界面，默认进入微博的页面，可以通过下滑菜单栏进行切换。首先进入的页面为当前可视化任务队列，在这个界面可以查看目前爬虫工作进度和对队列进行管理，展示信息如下图4.8：

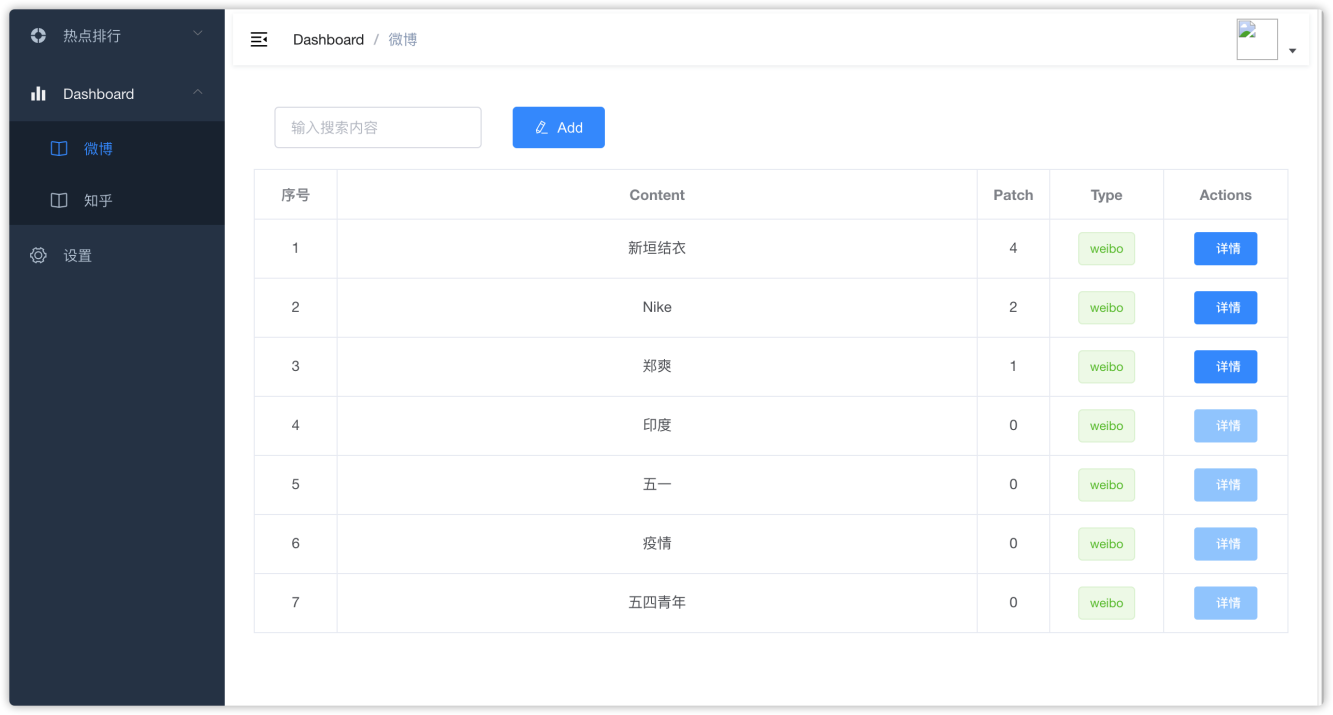


图4. 8 可视化信息队列

在当前页面可以通过上部的输入框添加搜索信息到队列中，后台爬虫会从队列中读取关键字，然后到对应的网站爬取相关数据并进行分析存储，当爬虫调用后，对应信息的patch会加1，然后详情的按钮会被激活，表示用户可以查看网站分析的结果。

用户可以通过详情按钮跳转到相对应的微博详情页，然后里面有对应微博的详细统计信息，包括微博标识号mid，微博内容，分析批次，具体展示信息如下图所示4.9：



图4. 9 详细微博信息

展示信息上部有分析结果的时间，对应的右侧有批次切换按钮，用户可以通过这切换显示不同时间段的分析结果，具体如下图4.10：



图4. 10 批次切换

在每一条结果的最右边有三个功能按钮，可以对每一条微博展开分析结果，主要包括词云和情感分析饼图，具体如下图所示4.11和4.12：



图4. 11 词云图

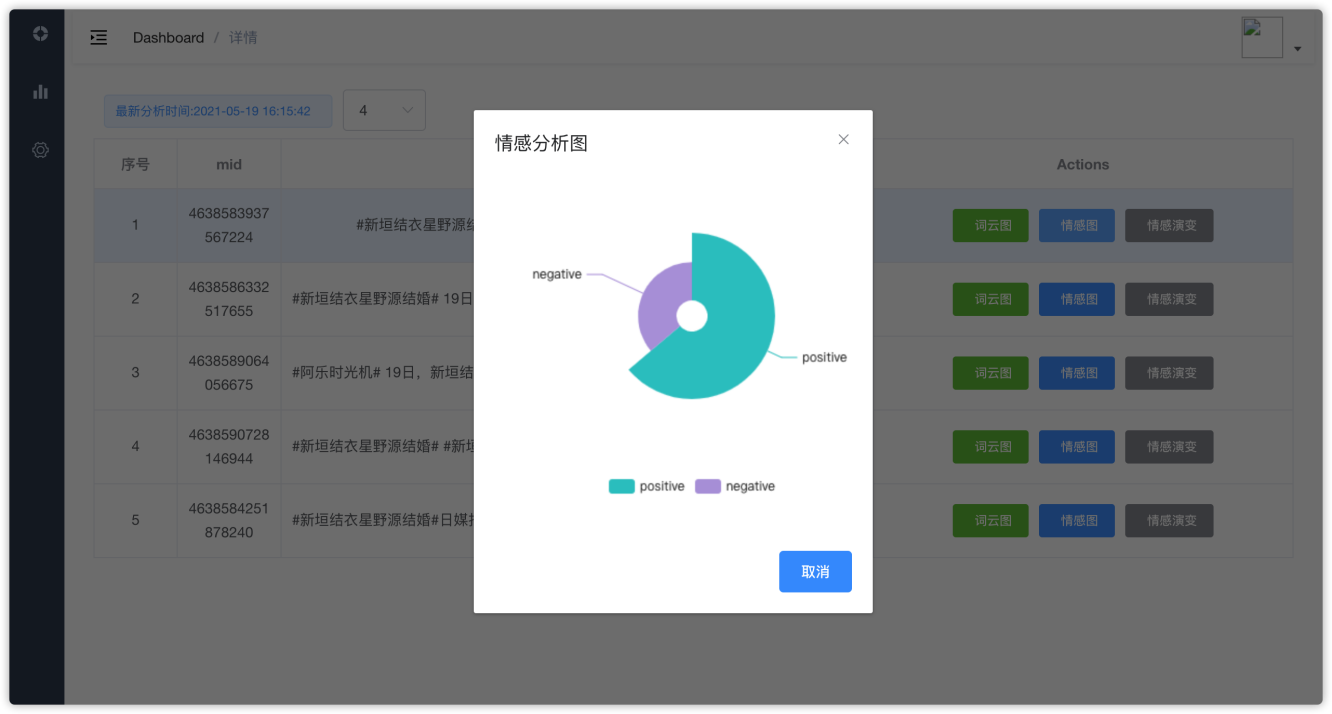


图4. 12 情感分析图

词云图主要展示对当前微博下评论关键词的统计结果，当词越大的时候表示在评论中该关键词出现的频率约大。情感分析饼图主要展示评论中积极评论和消极评论的分布情况。

除此之外，列表每一行都有情感演变的按钮，点击后会在列表下面展示情感演变趋势，每点击一行列表图表都会发生相应的变化。具体如下图4.13：

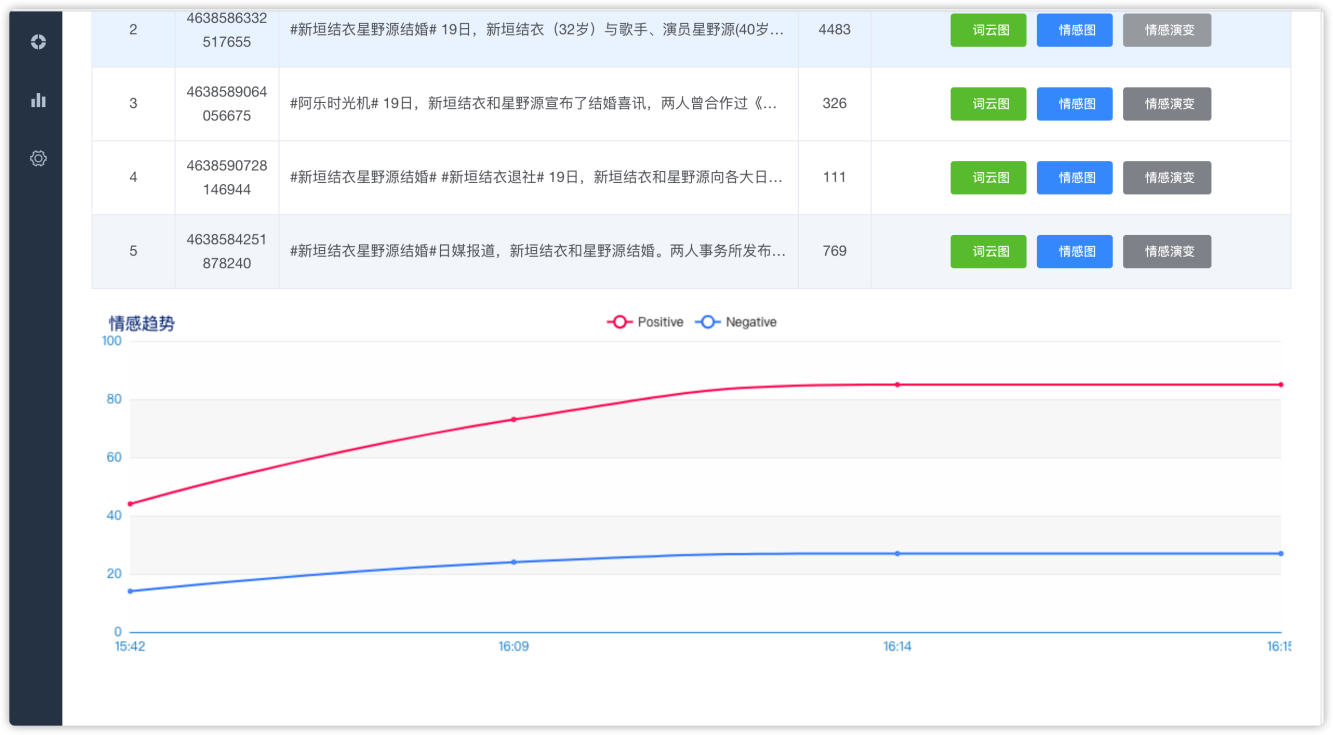


图4. 13 情感演变

每点击一行，下方的图表都可以相应的动态变换，可以让我们更直观的看到每条微博的情感演变情况。

**4.5.3 爬虫配置**

通过菜单栏可以跳转到爬虫设置页面，在这里可以查看不同爬虫配置的工作状态，并可以相对应的修改，具体如下图4.14、4.15和4.16：

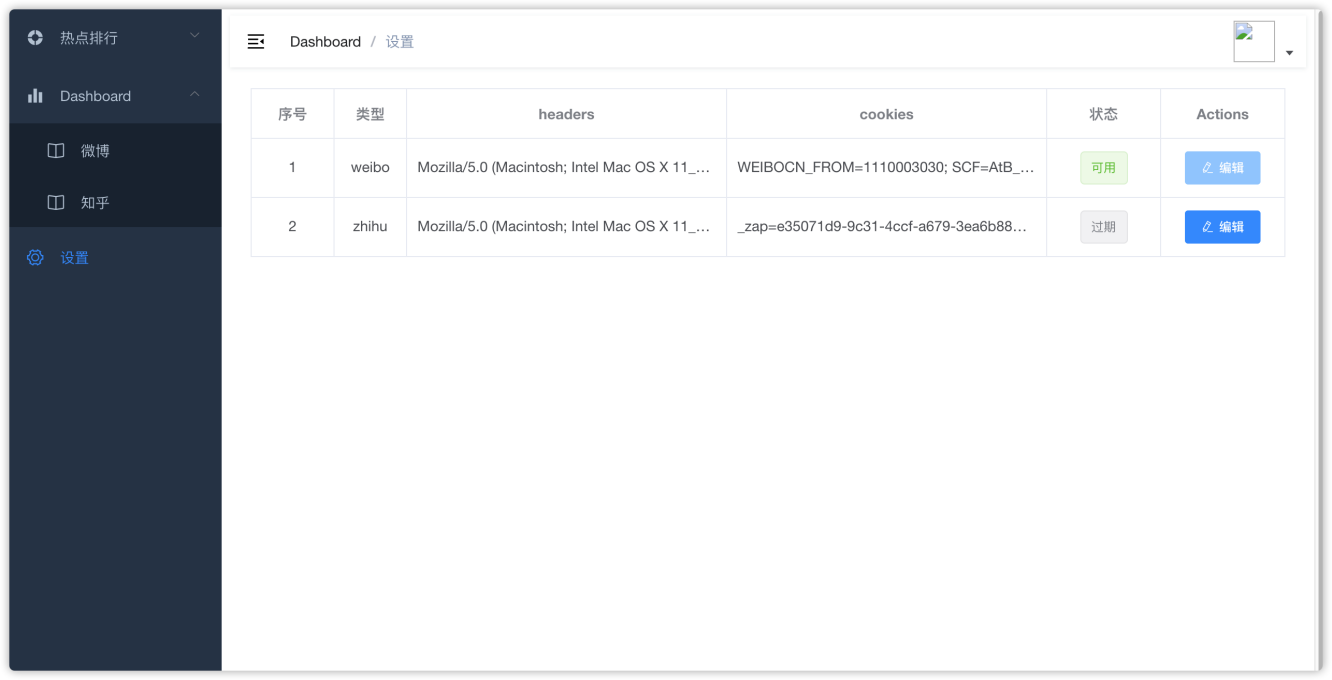


图4. 14 设置详情界面

当配置过期时，用户可以通过右侧的编辑按钮编辑详细的配置，具体编辑界面如下图所示：



图4. 15 设置编辑图

设置修改后，后台需要进行验证后才能使用，所以会显示待验证状态，具体如下图所示：

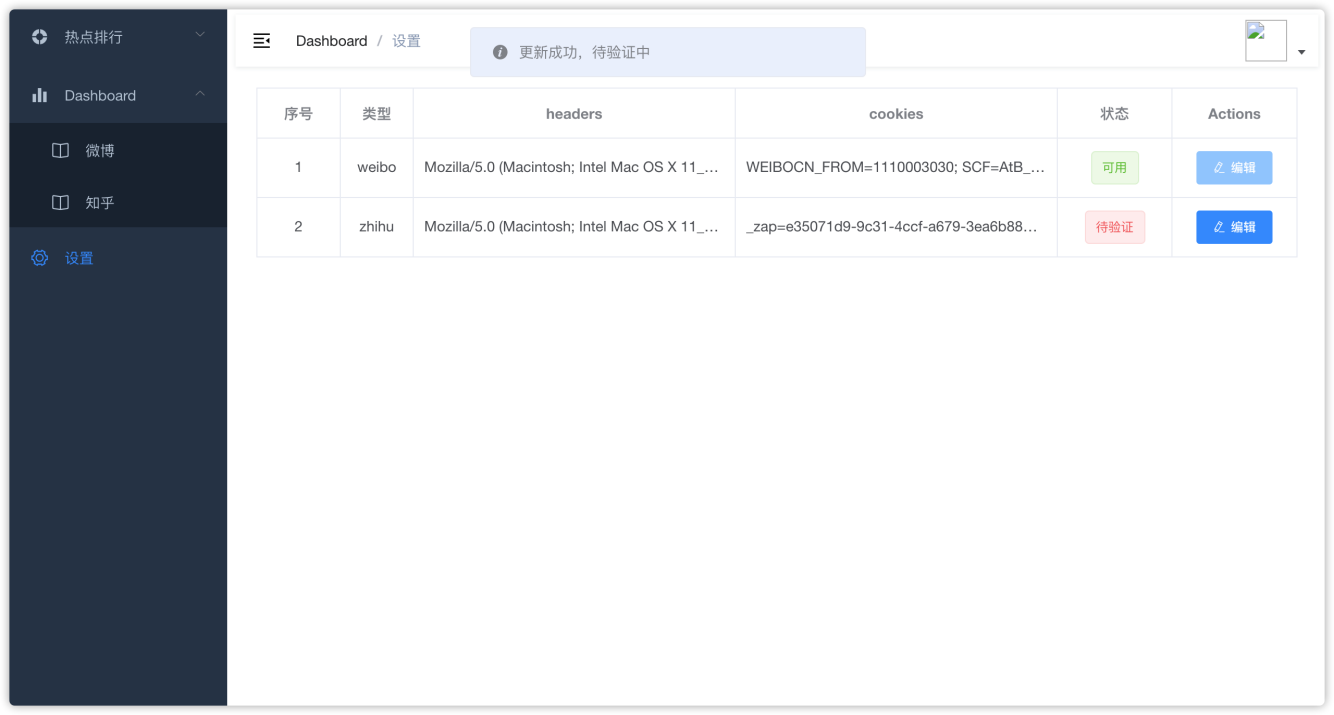


图4. 16 设置更新后

**5 总结与展望**

**5.1** 总结

本文设计了一个基于爬虫的网络信息可视化分析平台，用于对一些主流媒体网站热点信息的分析和可视化。整个平台包括网络信息的获取、数据的处理分析、可视化界面展示等一系列工程化的任务，可以将主流媒体网站的实时热点信息更直观的进行展示，同时也可以根据用户的需求，进行指定热点信息的抓取、分析和展示，让用户更详细的了解热点信息的发展和变化，同时该平台集成了对多个网站信息的可视化展示，让用户可以更加方便的浏览不同媒体渠道的热点信息，除此之外，用户通过该平台还可以查看过去一段时间内的热点信息，更清晰的了解到相关信息的发展变化过程。

在设计的过程中，先调研了很多主流媒体发布平台，如知乎、微博、今日头条等，了解这些网站的信息展示方式，然后又查询了相关网站信息的获取方式，最后选择了目前使用频率最多的微博和知乎。对于数据获取方式，又查询了很多爬虫相关的文献，又考虑到数据量的问题，最后选择了使用异步爬取框架Scrapy。对于数据处理方面，自己阅读了很多关于数据分析的相关文献，最后也结合预期可视化的结果选取了合适的算法。对于可视化平台，也利用设计软件，简单设计了预期的可视化界面。最后，自己又针对各个模块，详细设计了平台实现方案。

在具体实现过程中，先针对每个模块进行项目构建，然后再具体针对每个项目，进行相关任务的开发，每完成一个功能后都会进行相应的测试，并记录问题。在开发的过程中，自己也遇到了很多需求变更的环节，比如数据抓取的字段变化、可视化展示界面重构等，自己也都在变更前进行了调研。

总体上，最后该平台实现较为完整，数据获取、数据处理、可视化展示模块等功能都进行了实现，较好的完成了课题的需求。

**5.2** 展望

本次设计完成的基于爬虫的网络信息可视化分析平台较为完整，基本的爬虫、数据处理、可视化展示等功能基本已经实现，但是在许多方面仍有不足，在此，进行反思和展望。

首先，对于目标网站的数据获取，目前只调研并实现了微博和知乎两个网站的数据获取，并且数据格式较为单一，主要面向中文文本，对于一些图片和英语文本只是简单的进行过滤，并没有进行抓取分析。后期，可以针对相应的问题，调研一些其它信息展示形式的目标网站，比如像Bilibili、网易云等不一样展示形式的社交媒体网站，分析类型也可以不单单面向评论，也可以针对实时弹幕等。

其次，对于文本情感分析的算法方面，目前主要使用的基于统计学的机器学习算法，准确率较低，后面可以尝试其它的一些基础算法，并且也可以针对具体的场景使用一些深度学习方法，比如LSTM，用来提高文本分类的准确率。

最后，对于可视化的展示，目前图表的内容较为单一，后面可以针对性的丰富可视化的内容，让网站内容更加丰富。

**参考文献**

1. 第47次《中国互联网络发展状况统计报告》发布[J].新闻世界,2021(03):96.
2. 张琛，马祥元，周扬，郭仁忠．基于用户情感变化的新冠疫情舆情演变分析 [J/OL]．地球信息科学学报，2020-09-30。
3. 姚芳. 基于python的中文文本分类研究[D] 华中科技大学，2016
4. 曹鲁慧,邓玉香,陈通,李钊.一种基于深度学习的中文文本特征提取与分类方法[J].山东科学,2019,32(06):106-111.
5. 孙东伟,刘滨.数据可视化研究与技术实现[J].产业与科技论坛,2020,19(19):40-42.
6. 陈小燕, 干丽萍, 郭文平. 大数据可视化工具比较及应用[J]. 计算机教育, 2018, 000(006):97-102.
7. 郑戟明，柳青．Echarts在数据可视化课程中的应用[J].电脑知识与技术，2020(16): 9-11.
8. 潘平. 基于网络系统集成的大数据分析及应用软件[J]. 信息与电脑(理论版), 2018, 418(24):107-108.
9. 董少林,李钟慎.采用Scrapy分布式爬虫技术的微博热点舆情信息获取与分析[J].电脑与信息技术,2020,28(05):23-26.
10. 罗骁. 基于分布式爬虫的用户评论分析系统[D] 南京邮电大学，2018.12
11. 刘洪浩.基于深度学习的COVID-19疫情期间网民情绪分析[J].软件,2020,41(09):185-188.
12. 罗骁. 基于分布式爬虫的用户评论分析系统[D].南京邮电大学,2018.
13. 张林兵，郭强，吴行斌，梁耀洲等. 基于多维行为分析的用户聚类方法研究[J] 电子科技大学学报 2020: 49-2
14. Huijie Li. Research on Educational Donation Mechanism Based on Data Mining and Economics[J]. Advances in Social Sciences,2016,2(10).
15. Oladejo Bolanle,Babajide Olanrewaju. A Framework for News Media Knowledge Discovery Based on Text Mining Techniques[J]. Journal of Social Science and Humanities,2020,2(8).

**致谢**

本毕业设计（论文）是在软件学院史骏老师的悉心指导下完成。在这过程中，无论是开题、课题实施、论文编写，史骏老师都给了自己很多支持和帮助。尤其是在自己实习和考研期间，老师也经常联系我，给自己很多意见，帮助解决了很多问题。在此我向他表示深深的感谢和敬意。除此之外，也要感谢软件学院这几年来的培养，让自己在这四年里学到了很多专业的知识，让自己的学习能力、实践能力都得到了很大的提升，真的是他们的谆谆教诲才成就了现在的我。

时光荏苒，转眼间大学四年就这样过去了，随着论文的落笔，自己大学的学习生涯也即将走向尾声，写到致谢部分也真的是感慨良多，在自己大学这四年间，走过了很多的路，认识了很多的人，也学到了很多东西，相信这些时光的碎片自己一定会好好珍藏，以后也会成为自己美好的回忆。

最后，感谢所有这些年来帮助过自己的那些“可爱的人”。

作者：李浩

2021年 05 月 22日