**2015年全国大学生信息安全竞赛**

**作品报告**

**作品名称： 基于微博的舆情分析系统**

**电子邮箱： h1hummer@163.com**

**提交日期： 2015年6月7日**

目录

[摘要 4](#_Toc421484513)

[第1章 作品介绍 5](#_Toc421484514)

[1.1 舆情分析简介 5](#_Toc421484515)

[1.1.1 什么是舆情分析 5](#_Toc421484516)

[1.1.2 舆情分析的两大方法 5](#_Toc421484517)

[1.1.3 舆情分析的注意点 5](#_Toc421484518)

[1.2 开发平台介绍 6](#_Toc421484519)

[1.2.1 Python的基本介绍 6](#_Toc421484520)

[1.2.2 开发工具Pycharm的介绍 6](#_Toc421484521)

[1.2.3 MariaDB简介 7](#_Toc421484522)

[1.3 网络爬虫 7](#_Toc421484523)

[1.3.1 网络爬虫的基本原理 7](#_Toc421484524)

[1.3.2 网络爬虫分类 8](#_Toc421484525)

[1.3.3 微博爬虫的实现策略 8](#_Toc421484526)

[1.4 数据分析基础 8](#_Toc421484527)

[1.4.1 全文检索 9](#_Toc421484528)

[1.4.2 中文分词 9](#_Toc421484529)

[1.4.3 自然语言处理 9](#_Toc421484530)

[1.4.4 常用数据分析算法 9](#_Toc421484531)

[第2章 实现方案 11](#_Toc421484532)

[2.1 总体设计 11](#_Toc421484533)

[2.1.1 系统数据字典 11](#_Toc421484534)

[2.1.2 系统数据流图 12](#_Toc421484535)

[2.2 网络爬虫 12](#_Toc421484536)

[2.2.1 微博爬虫简介 12](#_Toc421484537)

[2.2.2 微博页面分析 13](#_Toc421484538)

[2.2.3 爬虫框架 13](#_Toc421484539)

[2.2.4 数据提取 16](#_Toc421484540)

[2.3 数据存储 16](#_Toc421484541)

[2.3.1 数据库的设计 16](#_Toc421484542)

[2.3.2 数据存取实现 19](#_Toc421484543)

[2.4 数据分析 19](#_Toc421484544)

[2.4.1 中文分词 20](#_Toc421484545)

[2.4.2 自然语言处理 20](#_Toc421484546)

[2.4.3 数据挖掘 20](#_Toc421484547)

[2.4.4 WeiboRank算法 21](#_Toc421484548)

[2.5 分析结果的检索 24](#_Toc421484549)

[2.5.1 全文检索 24](#_Toc421484550)

[2.5.2 检索平台的实现 24](#_Toc421484551)

[第3章 性能测试 27](#_Toc421484552)

[3.1 爬虫稳定性测试 27](#_Toc421484553)

[3.2 爬虫效率测试 28](#_Toc421484554)

[3.3 数据库存取测试 28](#_Toc421484555)

[3.4 分析结果检索测试 29](#_Toc421484556)

[第4章 优势与创新 30](#_Toc421484557)

[4.1 深度定制的爬虫 30](#_Toc421484558)

[4.2 高效的数据提取手段 30](#_Toc421484559)

[4.3 WeiboRank分析算法 30](#_Toc421484560)

[4.4 CLI与Web结合 30](#_Toc421484561)

[4.5 系统的并发实现 31](#_Toc421484562)

[4.6 系统的可移植性 31](#_Toc421484563)

[第5章 总结 32](#_Toc421484564)

[参考文献 33](#_Toc421484565)

# 摘要

微博目前已经成为一种新兴的社会媒体，虽然微博只有区区140个字，但其已经成为广大民众参与舆情言论的重要途径，以至于一条小小的微博经常出现在舆情焦点的浪潮巅峰。针对这种情况，本文提出并实现了一套基于微博的完整的舆情分析系统。整个系统包括从数据收集到存储，再到分析和检索整个完整过程。在舆情的分析上提出了使用基于PageRank的WeiboRank算法来萃取重要的微博，并通过全文检索引擎来根据需要搜索舆情信息。

**关键词：**微博，舆情分析，PageRank算法

# 作品介绍

## 舆情分析简介

### 什么是舆情分析

舆情分析就是根据特定问题的需要，对针对这个问题的舆情进行深层次的思维加工和分析研究，得到相关结论的过程。

网络舆情分析有两个工作重点，一是还原舆情发展过程，找到舆情产生的根源；二是预测，分析出网络舆情的未来走向，再根据预测结果提出应对方案。

网络舆情分析是在关注社会动态，了解民意诉求，并最终为了提高政府服务职能。[[[1]](#endnote-1)]

### 舆情分析的两大方法

1. **内容分析法**

内容分析法是一种对信息内容作客观系统的定量分析的专门方法，其目的是弄清或测验信息中本质性的事实和趋势。提示信息所含有的隐性情报内容，对事物发展做情报预测。

1. **实证分析法**

实证分析法是通过分析大量案例和相关数据后试图得出某些结论的一种常见研究方法。

### 舆情分析的注意点

1. **舆情事件分期与类型界定**

对舆情的分析要明确事件或话题本身所处的阶段，一般分为引发期、酝酿期、发生期、发展期、高潮期、处理期、平息期和反馈期等不同阶段。其次，应该在分析某一舆情热点之前对其进行科学的类型界定。热点事件一般主要分为突发自然灾害事件、生产安全事故、群体性事件、公共卫生事件、公权力形象、司法事件、经济民生事件、社会思潮、境外涉华突发事件等。

1. **舆情分析基本框架**

针对舆情的影响力、民众利益的关切度和对公共部门形象的破坏程度，舆情分析的观点参数设置与数据分析框架应注意反映如下情况：事件发展概况与脉络，民众的主要观点与情绪是什么，民众处于认知、态度表达还是处于行动阶段；引起民众从认知到社会运动、甚至社会骚乱的程度；组织化程度与有无行动计划性；慎用政治化解读，分析有无明确的利益诉求；有没有对现存社会体制构成巨大冲击；被反华势力支持或利用的可能性有多大；行动的对象目标明确与否，其具体指向是哪些部门甚至现存体制等；舆情分析研究人员要成为“非参与性的观察者”，而要防止成为“非观察性的参与者”，既要保持中立，又要敏锐。

## 开发平台介绍

本次系统选用的编程语言为Python，开发工具为PyCharm，操作系统为Windows和Linux。数据库使用MariaDB。

Python语言简单，语法优美，还有大量的第三方优秀的库。同时在爬虫、数据分析领域有着得天独厚的优势，所以选择Python作为我们系统开发的主要语言。因为Python是一门移植性特别好的语言，所以它编写的程序可以在Windows、Linux和Unix下很好的运行。

PyCharm是一个十分优秀，拥有漂亮界面的Python开发IDE。在Python开发者的心中享有极大的声誉，其智能语法提示的水平在同类IDE中首屈一指。而且PyCharm对于Flask web开发框架有着很好的支持。所以开发工具非PyCharm不可。

MariaDB数据库是Mysql的一个分支，它继承了Mysql的有点，同时在存储引擎上开拓创新，是一个存储性能强大的数据库。

### Python的基本介绍

Python，是一种面向对象、直译式的计算机程序语言，具有近二十年的发展历史。它包含了一组功能完备的标准库，能够轻松完成很多常见的任务。它的语法简单，与其它大多数程序设计语言使用大括号不一样，它使用缩进来定义语句块。

Python支持命令式程序设计、面向对象程序设计、函数式编程、面向侧面的程序设计、泛型编程多种编程范式。代码编写方式十分灵活，简洁明了，十分贴近与自然语言。而且

### 开发工具Pycharm的介绍

PyCharm是一种Python IDE，带有一整套可以帮助用户在使用Python语言开发时提高其效率的工具，比如调试、语法高亮、Project管理、代码跳转、智能提示、自动完成、单元测试、版本控制。并且PyCharm的社区版本是免费和开源的，使用其不需要额外的花费和复杂的授权。

### MariaDB简介

MariaDB数据库管理系统是MySQL的一个分支，主要由开源社区在维护，采用GPL授权许可。开发这个分支的原因之一是：甲骨文公司收购了MySQL后，有将MySQL闭源的潜在风险，因此社区采用分支的方式来避开这个风险。

MariaDB的目的是完全兼容MySQL，包括API和命令行，使之能轻松成为MySQL的代替品。在存储引擎方面，10.0.9版起使用XtraDB（名称代号为Aria）来代替MySQL的InnoDB。

#### MariaDB发展

成立于2009年，MySQL之父Michael “Monty” Widenius用他的新项目MariaDB完成了对MySQL的“反戈一击”。开发这个分支的原因之一是：甲骨文公司收购了MySQL后，有将MySQL闭源的潜在风险，因此社区采用分支的方式来避开这个风险。 过去一年中，大型互联网用户以及Linux发行商纷纷抛弃MySQL，转投MariaDB阵营。MariaDB是目前最受关注的MySQL数据库衍生版，也被视为开源数据库MySQL的替代品。

MariaDB虽然被视为MySQL数据库的替代品，但它在扩展功能、存储引擎以及一些新的功能改进方面都强过MySQL。而且从MySQL迁移到MariaDB也是非常简单的。

#### MariaDB特点

MariaDB是一个采用 Maria 存储引擎的MySQL分支版本，是由原来 MySQL 的作者Michael Widenius创办的公司所开发的免费开源的数据库服务器。

相对于MySQL最新的版本5.6来说，在性能、功能、管理、NoSQL扩展方面，MariaDB包含了更丰富的特性。比如微秒的支持、线程池、子查询优化、组提交、进度报告等。

## 网络爬虫

### 网络爬虫的基本原理

网络爬虫源于英文Web Crawler的翻译，它是一个功能强大的自动提取网页的程序，是搜索引擎的重要组成部分。它通常以若干网页作为爬行起点，提取其URL链接（主要是通过分析HTMIL代码，查找网页中链接到其他网页的标签（多为<href>))，放人待抓取的URL队列，重复此过程，当满足系统的某一条件时停止。

### 网络爬虫分类

网络爬虫根据需求的不同分为不同种类：

一种是爬取网页链接，通过url链接得到这个html页面中指定的链接，把这些链接存储起来，再依次以这些链接为源，再次爬取链接指向html页面中的链接……如此层层递归下去，常用的方法是广度优先或者深度优先，根据爬取层次需求不同而选择不同的方法达到最优效果，爬虫的效率优化是一个关键。搜索引擎的第一个步骤就是通过爬虫得到需要索引的链接或数据，存放于数据库，然后对这些数据建立索引，然后定义查询语句，解析查询语句并利用检索器对数据库里的数据进行检索。[[[2]](#endnote-2)]

另一种是爬取数据信息，如文本信息、图片信息等，有时需要做数据分析，通过某种手段来获取数据样本以供后续分析，常用的方法是爬虫获取指定数据样本或利用现有的公共数据库。[2]

本作品的微博爬虫属于第二种类，根据自定义搜索关键字爬取微博信息数据。

### 微博爬虫的实现策略

对于微博网络爬虫原理，其实并不复杂。基本思路是：由关键字指定的url把所有相关的html页面全抓下来（html即为字符串），然后解析html文本（通常是正则表达式），提取微博文本信息，然后把文本信息存储起来。

重点在于对html页面源码结构的分析，不同的html需要不同的解析方法；还有就是长时间爬取可能对IP有影响，有时需要获取代理IP，甚至需要伪装浏览器爬取。（主要是针对像新浪等这些具有反扒功能的网站，新闻网站一般不会有这样的情况）。

对于微博，通常情况下是必须登录才能看到微博信息数据，但是微博有搜索机制，在非登录的情况下不可以直接通过搜索话题来查找相关信息。考虑到某些反爬虫机制，如果一个账号总是爬取信息可能会有些影响（比如被封号）。所以我们注册了几十个账号轮流爬取，并且控制相关的时间，应对微博的反爬虫机制。

## 数据分析基础

数据分析是指通过建立审计分析模型对数据进行核对、检查、复算、判断等操作，将被审计单位数据的现实状态与理想状态进行比较，从而发现审计线索，搜集审计证据的过程。数据分析的目的是把隐没在一大批看来杂乱无章的数据中的信息集中、萃取和提炼出来，以找出所研究对象的内在规律。[[[3]](#endnote-3)]

在实用中，数据分析可帮助人们作出判断，以便采取适当行动。数据分析是组织有目的地收集数据、分析数据，使之成为信息的过程。[3]

基于微博的舆情分析，其本质也是一种数据分析。我们从某几个主要关键字中，在微博上爬取相关的数据，同时存入本地的数据库中，当数据达到一定的数量级之后，对数据进行相关算法分析，得到网络舆情分析的结果。

### 全文检索

全文检索是将存储于数据库中整本书、整篇文章中的任意内容信息查找出来的检索。它可以根据需要获得全文中有关章、节、段、句、词等信息，也就是说类似于给整本书的每个字词添加一个标签，也可以进行各种统计和分析。

在舆情分析系统中，我们将使用全文检索将分析好的数据从数据库中检索出来。

### 中文分词

中文自动分词指的是使用计算机自动对中文文本进行词语的切分，即像英文那样使得中文句子中的词之间有空格以标识。中文自动分词被认为是中文自然语言处理中的一个最基本的环节。

我们本次的数据源目标是新浪微博，内容主要以中文为主。因为中文没有像英文那样以天然的空格隔开，所以需要对句子进行分词，为后面全文检索和自然语言分析奠定基础。

### 自然语言处理

自然语言处理（英语：Natural Language Processing，简称NLP）是人工智能和语言学领域的分支学科。在这此领域中探讨如何处理及运用自然语言；自然语言认知则是指让电脑“懂”人类的语言。[[[4]](#endnote-4)]

在舆情分析中，数据源的分类和情感分析是很重要的。分类可以用于汇总和推荐舆情信息，而情感分析可以为舆情的走向提供预测信息。所以自然语言处理是舆情分析的必要步骤。

### 常用数据分析算法

1. **C4.5**

C4.5算法是机器学习算法中的一种分类决策树算法,其核心算法是ID3算法。C4.5算法有如下优点：产生的分类规则易于理解，准确率较高。

1. **PageRank**

PageRank是Google算法的重要内容。PageRank根据网站的外部链接和内部链接的数量和质量俩衡量网站的价值。

1. **CART: 分类与回归树**

CART, Classification and Regression Trees。 在分类树下面有两个关键的思想。第一个是关于递归地划分自变量空间的想法；第二个想法是用验证数据进行剪枝。

微博的用户、博文和评论间也暗含着类似于网页间连接的关系，可以使用类似于PageRank的算法进行内容的提取。

# 实现方案

本项目目的是为了打造一个基于微博的舆情分析系统，项目命名为Xenon，来源于元素周期表的54号元素氙（Xe）。本项目将提供从数据收集存储，到处理分析，以及检索展示的完整解决方案。目前项目完全开源，遵循MIT开源协议，托管在Github上，地址为<https://github.com/KureFM/Xenon>。

## 总体设计

Xenon主体使用Python开发，一共分为5个模块，分别是爬虫、数据存储、数据分析、数据检索和调度引擎。

各个模块分工良好，完全可以完全独立运行。其中前4个模块主要是提供函数方法接口，最终由调度引擎进行调度。各个模块通过统一的数据model进行数据交换，使得开发工作可以独立进行。

各个模块对应的文件如表 2‑1所示。

|  |  |
| --- | --- |
| 爬虫 | crawler.py |
| 数据存储 | data.py |
| 数据分析 | analysis.py |
| 数据检索 | retrieval.py |
| 调度引擎 | xenon.py |
| 数据模型 | models.py |
| 通用函数 | common.py |
| 系统异常 | errors.py |

表 2‑1 Xenon各模块文件

### 系统数据字典

数据字典最重要的作用是作为分析阶段的工具。任何字典最重要的用途都是供人查询对不了解的条目的解释，在结构化分析中，数据字典的作用是给数据流图上每个成分加以定义和说明。换句话说，数据流图上所有的成分的定义和解释的文字集合就是数据字典，而且在数据字典中建立的一组严密一致的定义很有助于改进分析员和用户的通信。

数据字典的定义是软件的核心部分，它直接影响了数据模型的编写以及数据库表的创建，所以给出数据字典的描述是十分重要的。本系统的数据字典如图 2‑1所示

博文=内容+时间+转发列表+评论列表+点赞列表

评论=内容+时间+点赞列表

用户=用户信息+关注列表+粉丝列表

图 2‑1 微博数据模型

### 系统数据流图

数据流图是描述系统中数据流程的一种图形工具，它标志了一个系统的逻辑输入和逻辑输出，以及把逻辑输入转换逻辑输出所需的加工处理。它能够清晰地描述系统各部分的协作过程。本系统的数据流图如图 2‑2所示。

爬虫

数据库

分析引擎

检索引擎

调度引擎

保存数据

查找重复

读取数据

保存分析结果

保存索引

读取数据

调度

调度

调度

图 2‑2 系统数据流图

## 网络爬虫

### 微博爬虫简介

本系统采用深度定制的网络爬虫来收集微博的信息，由于微博对爬虫的封锁日益严重，现在使用普通爬虫已经无法获取到完整的微博内容。为此，我们针对新浪微博专门定制了爬虫，可以高效完整地爬取微博的信息。

### 微博页面分析

为了快捷高效的地提取微博页面的内容，我们选择了移动版的网页微博来入手。首先是移动版的登陆逻辑简单，而且信息熵较高。经过分析比较，包含同等信息的页面，桌面版的大小是移动版的5到10倍。

微博的页面是完全动态的，数据完全依靠Ajax进行加载，即时是第一页的HTML页面，也是将json数据放在script标签中。这一特点虽然看起来棘手，但是对于定制爬虫反而是好消息，只要分析清楚Ajax请求的内容，数据处理起来比HTML要简单很多。

### 爬虫框架

根据面向对象的思想，将爬虫由上到下分为微博爬虫类（WeiboCrawler）、微博登陆类（WeiboLogin）和HTTP操作类（HttpOperation）。其基本UML图如图 2‑3所示。

HttpOperation主要是提供一些常用的HTTP操作，比起get，post方法，并为保持会话信息提供保障。

WeiboLogin包含全部的微博登陆逻辑，若日后登陆的方式有所改变，直接修改这里便能继续使用

WeiboCrawler则是微博爬虫的主要功能，包括搜索功能和获取单一的微博或者用户信息。

整个爬虫的流程图如图 2‑4所示

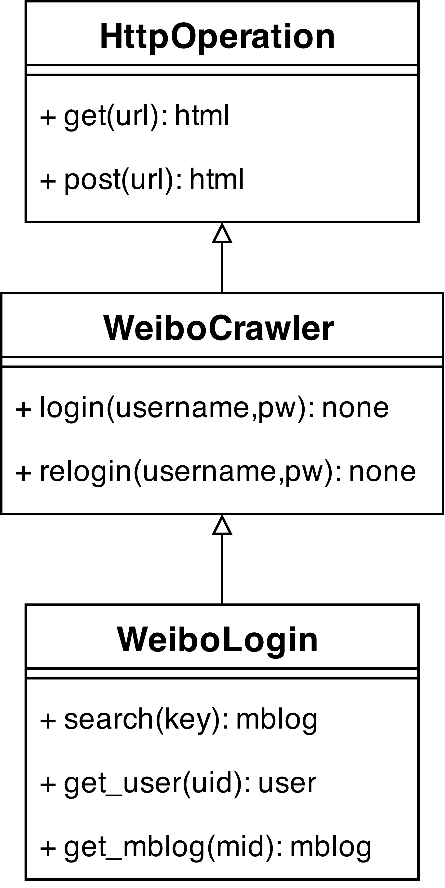


图 2‑3 爬虫UML类图

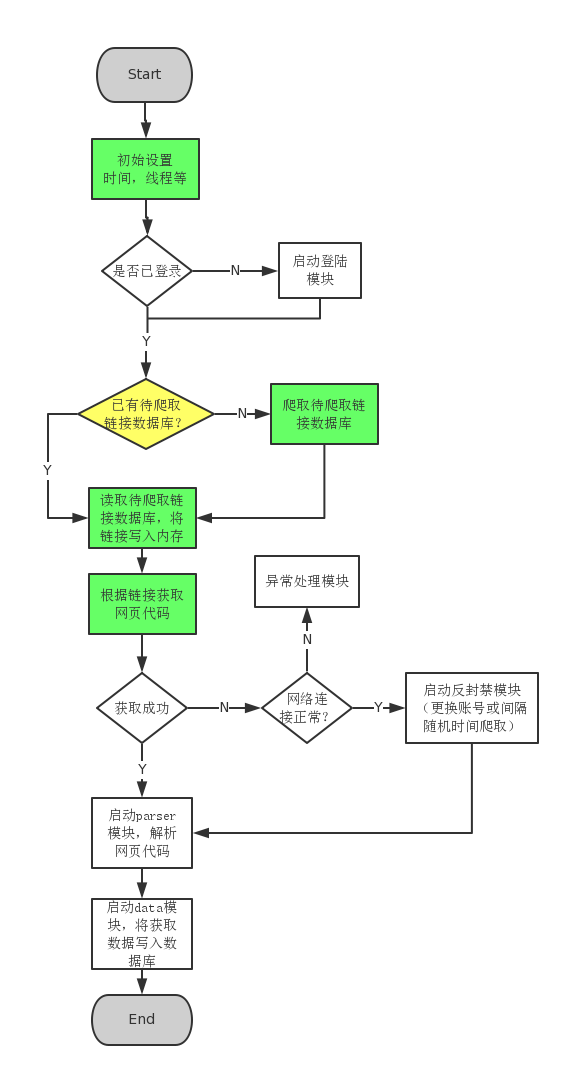


图 2‑4 爬虫流程图

### 数据提取

获取到页面html代码后，接着就是要提取其中的数据了。Ajax请求返回的数据可以使用json模块解析就能获取数据，但为了兼容html中的json数据，我们使用了正则表达式来提取。实际测试一条编写良好的正则表达式并不比json（其实json模块内部也是使用正则表达式解析）慢，而且容错能力更强。

## 数据存储

爬虫的数据存储使用统一的MariaDB数据库来进行存储，使用统一的数据库可以方便的将爬虫扩展成分布式爬虫，为数据收集提高保障。

### 数据库的设计

数据库设计的首要任务是分析数据对象的结构和关系并给出E-R图，其中数据对象的分析已经在2.1.1系统数据字典中做了说明，根据数据字典可以得出如所示的数据库E-R图。

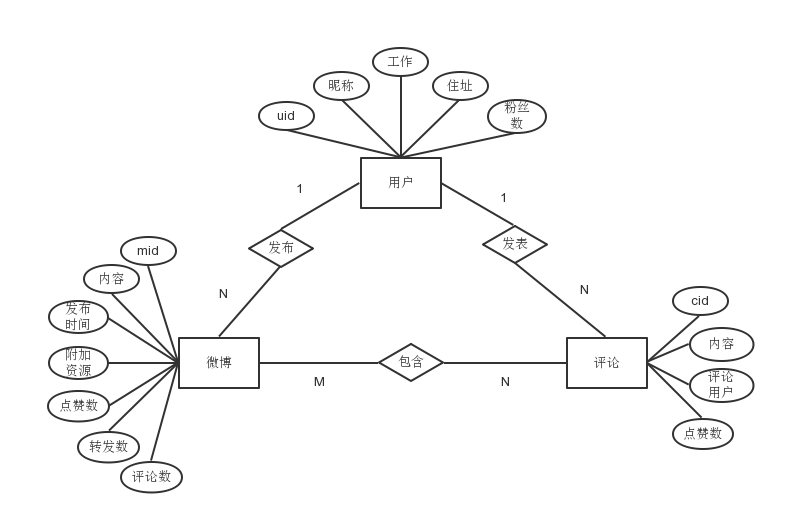


图 2‑5 数据库E-R图

参考E-R图，结合相应数据库平台，可以得出具体数据表的设计。表的具体设计如表 2‑2到表 2‑11所示。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段 | 类型 | 空 |  | 默认 | 注释 |
| uid (主键) | bigint(20) | 否 |  |  | 微博用户ID |
| nickname | varchar(30) | 否 |  |  | 昵称 |
| job | varchar(50) | 是 |  | NULL | 工作 |
| address | varchar(50) | 是 |  | NULL | 地址 |
| fans\_num | int(10) | 是 |  | NULL | 粉丝数 |

表 2‑2 Users（微博用户实体）关系表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 空 | 默认 | 注释 |
| mid *(主键)* | bigint(20) | 否 |  | 微博编号ID |
| content | varchar(300) | 否 |  | 内容 |
| posttime | datetime | 是 | NULL | 发布时间 |
| time\_stamp | datetime | 是 | NULL | 时间戳，用于更新比较 |
| extra | varchar(1000) | 是 | NULL | 附加资源 |
| forward\_num | int(10) | 否 |  | 转发数 |
| comment\_num | int(10) | 否 |  | 评论数 |
| like\_num | int(10) | 否 |  | 点赞数 |

表 2‑3 Weibo（一条微博的实体）表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 空 | 默认 | 注释 |
| cid *(主键)* | bigint(20) | 否 |  | 评论编号ID |
| content | varchar(300) | 否 |  | 内容 |
| suid | bigint(20) | 是 | NULL | 评论用户 |
| like\_num | int(10) | 是 | NULL | 点赞数 |

表 2‑4 Comments（微博评论实体）表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 空 | 默认 | 注释 |
| NoID *(主键)* | int(10) | 否 |  | 逻辑主键 |
| uid | bigint(20) | 否 |  | 用户编号ID |
| mid | bigint(20) | 否 |  | 微博编号ID |

表 2‑5 user\_weibo（用户-微博）关系表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 空 | 默认 | 注释 |
| NoID *(主键)* | int(10) | 否 |  | 逻辑主键 |
| mid | bigint(20) | 否 |  | 微博编号ID |
| cid | bigint(20) | 否 |  | 评论编号ID |

表 2‑6 weibo\_comment（微博-评论）关系表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 空 | 默认 | 注释 |
| NoID *(主键)* | int(10) | 否 |  | 逻辑主键 |
| uid | bigint(20) | 否 |  | 用户编号ID |
| cid | bigint(20) | 否 |  | 评论编号ID |

表 2‑7 user\_comment（用户-评论）关系表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 空 | 默认 | 注释 |
| NoID *(主键)* | int(10) | 否 |  | 逻辑主键 |
| uidA | bigint(20) | 否 |  | 用户A编号ID |
| uidB | bigint(20) | 否 |  | 用户B编号ID |

表 2‑8 follow（用户关注，用户uidA关注用户uidB）表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 空 | 默认 | 注释 |
| NoID *(主键)* | int(10) | 否 |  | 逻辑主键 |
| midA | bigint(20) | 否 |  | 微博A编号ID |
| midB | bigint(20) | 否 |  | 微博B编号ID |

表 2‑9 forward（微博转发，微博midA转发微博midB）表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 空 | 默认 | 注释 |
| NoID *(主键)* | int(10) | 否 |  | 逻辑主键 |
| mid | bigint(20) | 否 |  | 微博编号ID |
| uid | bigint(20) | 否 |  | 点赞用户ID |

表 2‑10 weibo\_PL（微博点赞）表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 空 | 默认 | 注释 |
| NoID *(主键)* | int(10) | 否 |  | 逻辑主键 |
| cid | bigint(20) | 否 |  | 评论编号ID |
| uid | bigint(20) | 否 |  | 用户编号ID |

表 2‑11 comment\_PL（评论点赞表）表

### 数据存取实现

本系统的数据主要是在MariaDB上实现的，MariaDB是Mysql的一个分支，它继承和发扬了Mysql的优点，语法和Mysql保持一致。在Python中，我们主要通过一个第三方的库Mysqldb（<https://pypi.python.org/pypi/MySQL-python/1.2.5>）实现与数据库的存储交互。

在本系统里，我们主要是对Mysqldb原有的接口函数进行封装，方便进行调用。

在data.py中定义了一个MysqlConnector()的函数，它返回的是一个Mysqldb的connect对象。所有对数据库的操作都是基于这个对象进行操作的。

在已经有connect对象的基础上，就可以对数据库进行查询，插入，修改，删除等等的操作。

因为我们主要是通过爬虫爬取数据，然后再把数据存入数据库中。在整个系统里面，插入操作是最为频繁的，为了同一接口，在data.py编写了一个insert\_basic\_function(sql, args)函数，对数据插入进行操作。

对于查询，因为每个模块需要的数据不一样，没有办法同一接口，所以在使用到数据库数据的时候在自行定制。

## 数据分析

数据分析是本系统的重点内容，数据分析部分包括分词、自然语言处理、数据挖掘。

### 中文分词

在分词这一模块，我们使用的是“结巴”分词。“结巴”中文分词是Github上的一个开源项目（<https://github.com/fxsjy/jieba>），由于其能够准确的分词，且高效、可拓展性强，受到广大开发者的青眯。

其支持以下三种分词模式，可以满足不同的需求，在使用上非常的灵活：

**精确模式**，试图将句子最精确地切开，适合文本分析；

**全模式**，把句子中所有的可以成词的词语都扫描出来, 速度非常快，但是不能解决歧义；

**搜索引擎模式**，在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。

同时“结巴”分词还支持繁体分词和添加自定义词典等高级功能，是Python下一款相当不错的中文分词解决方案。我们主要用“结巴”分词来为自然语言处理和全文检索提供帮助。

### 自然语言处理

我们在自然语言处理中选用的是NLTK（Natural Language Toolkit），其中最重要的原因是这是基于Python的自然语言处理工具，可以和整个项目无缝衔接。

NLTK创建于 2001年，最初是宾州大学计算机与信息科学系计算语言学课程的一部分。从那以后，在数十名贡献者的帮助下不断发展壮大。如今，它已被几十所大学的课程所采纳，并作为许多研究项目的基础。[[[5]](#endnote-5)]

其使用简单且参考资料多，学习上手很快，是现阶段合理的自然语言解决方案。

在系统中主要被用来进行情感分析和微博内容分类。

### 数据挖掘

数据挖掘是指从大量数据或者数据库中提取有用的信息。在微博舆情分析中，应用数据挖掘，可以将隐藏的舆情信息从海量的微博中提取出来。[[[6]](#endnote-6)]

如果使用过微博，不难发现那些有重要的信息微博往往都有大量的转发、评论和点赞。转发、评论和点赞这些就是微博中数据对象间的关系。关系越多，微博的越重要。根据这点，很容易让人想起Google的PageRank算法，根据网页的反向链接来计算页面的排名。同理，可以构造出一个WeiboRank来对微博进行排序

### WeiboRank算法

#### 简介

参考继承PageRank的思想，结合新浪微博（后面简称微博）的数据和关系，分析计算出能对微博进行排序的WeiboRank算法。

#### PageRank的思想

PageRank通过网络浩瀚的超链接关系来确定一个页面的等级。Google把从A页面到B页面的链接解释为A页面给B页面投票，Google根据投票来源（甚至来源的来源，即链接到A页面的页面）和投票目标的等级来决定新的等级。简单的说，一个高等级的页面可以使其他低等级页面的等级提升。[[[7]](#endnote-7)]

PageRank让链接来“投票”，一个页面的“得票数”由所有链向它的页面的重要性来决定，到一个页面的超链接相当于对该页投一票。一个页面的PageRank是由所有链向它的页面（“链入页面”）的重要性经过递归算法得到的。一个有较多链入的页面会有较高的等级，相反如果一个页面没有任何链入页面，那么它没有等级。[7]

#### WeiboRank的讨论对象

和PageRank讨论的对象不同，微博不仅仅是一个网页，其中细分成了博文、评论和用户。各种数据间的关系也不是反向链接，而是根据各种数据模型间的关系，分为转发、评论、点赞和关注等。

在具体使用过微博，并进行细致分析后不难得出，微博一共有3种数据模型，分别是博文、评论和用户。其数据字典如图 2‑1所示。

这些数据的共有种有意义的关系，这9种关系如图 2‑6所示，详细的描述如下（以用户B为主体进行描述）：

1. 关注：用户B关注用户A。
2. 粉丝：用户A关注用户B，从用户B的角度来看，用户A是用户B的粉丝。
3. 发表微博：用户B发表微博B
4. 发表评论：用户B发表评论B。
5. 点赞（对微博）：用户A对微博B点赞。
6. 点赞（对评论）：用户A对评论B点赞。
7. 转发：微博A转发微博B。
8. 评论（对微博）：评论A评论微博B。
9. 评论（对评论）：评论C评论评论B。

用户A

微博A

评论A

用户B

微博B

评论B

评论C

图 2‑6 微博模型间关系

#### 从PageRank到WeiboRank的转换

从更深一层来看，PageRank的Rank值计算是根据页面间的关系进行的。在网页中这种关系表现为网页与网页间的链接关系，而到了微博中，就变成了博文、评论和用户中各种复杂的关系。而页面就变成了微博、评论和用户。

由于微博中已经不是单一的数据模型，所以Rank的计算类型也不再是单一的PageRank。而是根据数据模型分为了MblogRank、CommentRank和UserRank。

PageRank把从A页面到B页面的链接解释为A页面给B页面投票，根据这个思想，可以认为图 2‑6中由数据模型A到数据模B的关系认为是A给B投票。于是可以得出这3个Rank计算时使用的关系。

MblogRank（博文Rank，简称MBR）使用关系③、⑤、⑦和⑧。

CommentRank（评论Rank，简称CR）使用关系④、⑥和⑨。

UserRank（用户Rank，简称UR）使用关系②。

而关系①就像网页中的正向链接，并不是反映当前对象的排名好坏，而是用来计算链接到别的页面的Rank值，以避免多次投票的情况出现。所以关系①是用于辅以计算UserRank。

上述的关系中，有些是一对一关系，有些则是一对多关系。为了避免一对多关系造成的重复投票，必须对这种关系进行合理的修正。在PageRank中在遇到于一个页面链接到多个页面情况时，将当前页面的PageRank平均分给这些链接，避免了多次投票的出现也保证了投票的公平性。在WeiboRank中处理方式也是基本是一样的。

在微博中转发和评论都是一对一的关系，无需进行额外修正。但是关注（被关注）、发表评论、发表微博和点赞都是一对多关系，必修进行修正防止重复投票。修正的方法是将用户的UserRank按一定规则平均分给这些关系。之所以按一定规则，是因为考虑到用户在做出关注、发表微博、发表评论和点赞的决定时的概率是不一样的，所以必须按一定的比例进行分配。

#### WeiboRank的计算方法

根据上述内容，本节从数学的角度给出WeiboRank的计算方法。PageRank的算法给出的是基于所有网页的矩阵计算方法。一次计算可以得出所有页面的Rank值。但是以目前手头的数据收集水平和机器计算水平，以及微博中的复杂关系，直接对单一的数据模型计算Rank值会更简便。

在微博中分别用mid、uid和cid表示博文、用户和评论的唯一标识符。

某个标识为UID的用户的关注数量用表示，所有微博的数量用表示，评论的数量用表示，点赞的数量用表示。

关注、发表微博、发表评论和点赞的权重为、、和。

于是可得用户做出这些操作的Rank基数为

是标识为UID的用户的粉丝列表。

是标识为MID的微博的评论列表。

是标识XID的点赞列表，其中XID可以是MID或者CID。

因为MblogRank和CommentRank的计算都需要用到UserRank，所以先给出UserRank的计算方法。UR的值就是该用户所以粉丝的关注者Rank的总和。

同时，根据图 2‑6的关系图，可以得出MblogRank和CommentRank的计算方法。

## 分析结果的检索

### 全文检索

Lucene是一套用于全文检索和搜寻的开放源码程式库，由Apache软件基金会支持和提供。Lucene提供了一个简单却强大的应用程序界面，能够做全文索引和搜寻，在Java开发环境里Lucene是一个成熟的免费开放源代码工具；就其本身而论，Lucene是现在并且是这几年，最受欢迎的免费Java资讯检索程式库。

PyLucene是Python对搜索引擎Lucene 的封装，可以通过Python方便的调用Lucene的API。Lucene是全文搜索的模块，可以很方便的嵌入到应用程序中。对于Python的应用程序可以通过PyLucene使程序具备全文搜索的能力。

### 检索平台的实现

本系统的检索平台主要基于Web实现。我们摒弃了常规的GUI图形界面，用Web展示分析的数据，更加具有美观性，同时具有更加好的操控性。

在Python里，开发Web有着得天独厚的优势。这是因为在Python诞生后的这么多年，有很多的优秀的Web框架。他们继承了Python优美简洁的语法特性，对于网页的开发非常容易上手，其中最有代表性的有Flask，Django，web.py，Pylons，Tornado等等。它们在网页的开发中有着极大的声誉。在这里我们选择了Flask。

同时，为了使得检索平台更加的美观，我们使用了Bootstrap对Web进行了渲染，使得Web页面可以在不同浏览器上提供美观一致的显示效果。

#### Web平台的优势

**跨平台**，我们的整个系统都是用Python编写的，目的就是为了实现多平台兼容。而不同平台的GUI实现不同，且开发过程繁杂。虽然Python可以用于编写GUI，但写起来很牵强而且不美观（或者说要达到美观相当困难）。而使用Web平台代替GUI，就可以在任何有浏览器的系统上运行，只要使用的是同一个Web框架，就能实现“一次编写随处可用”。

**提供一致性体验**，编写好的HTML页面和逻辑无需进行任何改动，只需对不同的平台适配CSS即可完成跨平台的开发，同时可以保证各平台使用体验的一致性。借助类似于Bootstrap之类的CSS/HTML框架，可以轻松地为给平台提供一致的体验。

**降低部署难度**，目前很多服务器为了减少不必要的内存消耗，并没有配备GUI界面，即时完成了浩大的GUI开发，在这些系统上也无法运行。而使用Web平台则可以部署在这些CLI的服务器上，用有GUI界面的PC机去访问Web页面进行数据检索。

#### Flask介绍

Flask 是一个 Python 实现的 Web 开发微框架。“微”(micro) 表示你并不需要把整个 Web 应用塞进单个 Python 文件（虽然确实可以 ），也不意味着 Flask 在功能上有所欠缺。微框架中的“微”意味着 Flask 旨在保持核心简单而易于扩展。Flask 不会替你做出太多决策——比如使用何种数据库。而那些 Flask 所选择的——比如使用何种模板引擎——则很容易替换。除此之外的一切都由可由开发者自行决定。

#### Bootstrap介绍

Bootstrap是一组用于制作网站及网络应用程序的工具，里面包含有HTML、层叠样式表（CSS）及JavaScript的框架，提供字体排印、表单、按钮、导航及其他各种元件，并提供了Javascript扩充套件。对于同时要供不同上网装置及浏览器，如一般电脑、手机及平板电脑等浏览的网站及网络应用程序，Bootstrap提供了CSS媒体查询（Media Query），节省网站制作者分别制作不同版本的时间及不便。

使用Bootstrap，可以实现一次开发适配所有浏览器的愿景，减少开发时间。

#### 具体实现

具体的思想是：首先通过Pylucene检索引擎对MariaDB数据库的数据建立索引，让后我们通过使用关键字，调用Pylucene引擎对数据库的数据进行提取，并在Web平台进行展示。同时，为了使得检索平台更加的美观，我们使用了Bootstrap对Web进行了渲染，使得人机交互的体验更好。检索平台的工作流程图如图 2‑7所示。

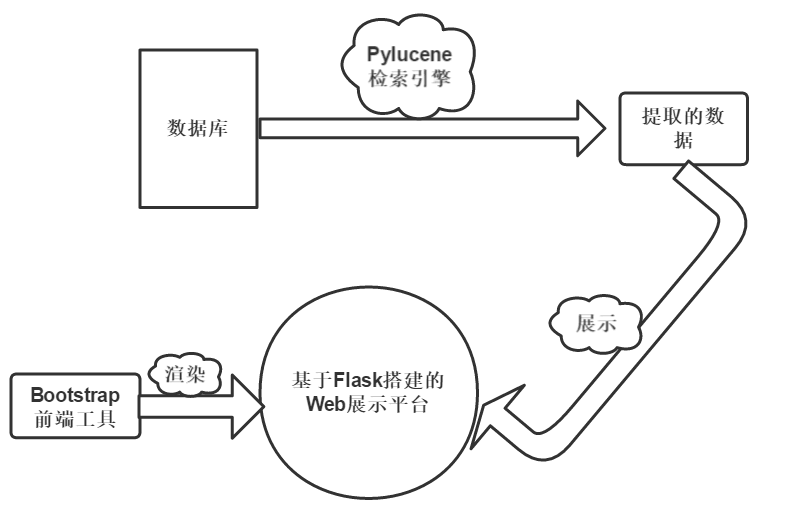


图 2‑7 检索平台工作流程

# 性能测试

## 爬虫稳定性测试

设定启动任务，从早上开始（如图 3‑1）到晚上（如图 3‑2），爬虫依旧没有出现间断。由于学校晚上断网断电，只能测试从早到晚。

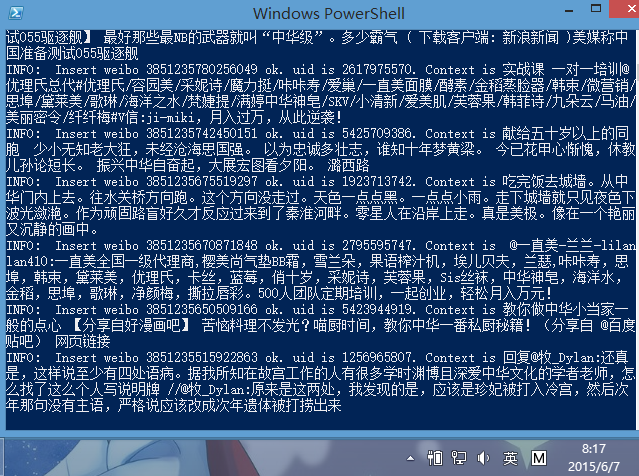


图 3‑1 爬虫早上开始工作

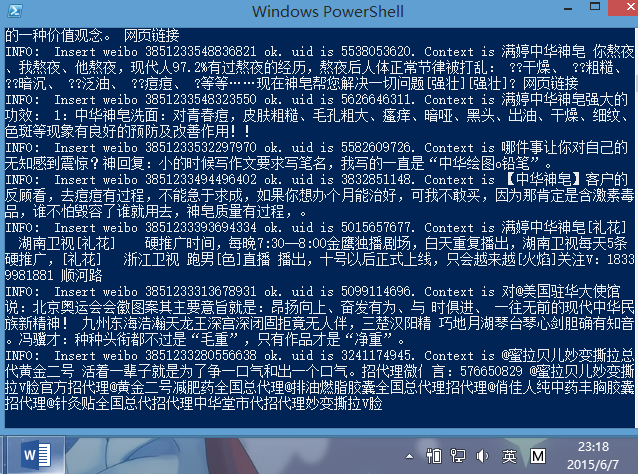


图 3‑2 爬虫工作到晚上

## 爬虫效率测试

由于新浪微博的封锁机制，无法大流量持续进行数据爬取，特别是要严格限制每个账号的访问频率。经过测试，每个账号30秒访问一次是比较保险的频率。为了弥补访问评论的限制，我们使用了人海战术——多账号来弥补这个缺陷。我们使用了接300个账号来达到每秒10次的访问频率。

那么每小时可以收集越36000条数据，考虑到网络拥塞等问题，有效率95%左右，每小时依旧可以收集到34000条数据。按每天有效运行18个小时计算，至少可以获得60万条数据。半个月下来就可以收集到上千万条数据，基本可以满足分析需求。

根据实际运行测试，一天的成果为608293条数据，基本达到理想需求。

## 数据库存取测试

在用循环对数据库进行压力存储测试，我们可一看到数据库中不断存入了数据，如图 3‑3所示。



图 3‑3 数据插入过程

根据我们的时间的计算，在10分钟内，插入了约5万条数据，与目前爬虫的效率比起来，在很长一段时间内基本够用。在执行插入操作时，不会有插入重复，插入数据错误的情况发生。对此，我们采取了一系列的措施，使得数据库没有出错。稳定性很好。

## 分析结果检索测试

系统使用Web平台对数据进行检索展示，主要基于Flask技术搭建web，Pylucene检索引擎建立索引查询。具体的展示界面如图 3‑4所示。

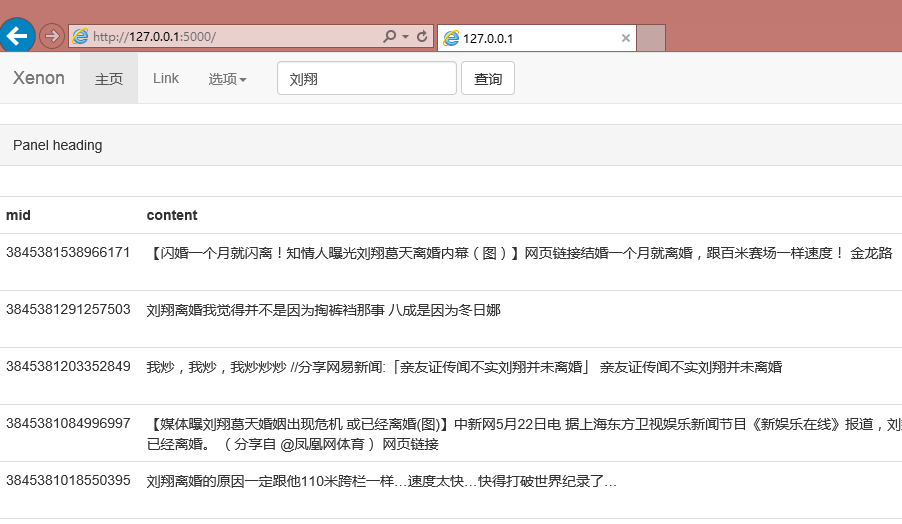


图 3‑4 检索结果

在浏览器中输入Web的地址，即可对数据进行查询操作。方便，简洁，效率高。

# 优势与创新

## 深度定制的爬虫

近段时间，新浪微博对于微博爬虫做出的限制越来越大。我们曾经尝试着使用一些开源的微博爬虫框架对微博进行爬取。结果都不如人意。这些爬虫（如Scrapy）在发布的时候的确能够爬取到一定的内容。

但是，微博的反爬虫机制一直在变化更新，很多的框架都失去了基本的功能。对此，我们决定自己编写了一个框架，从登陆到会话保持，从关键字搜索到内容提取，从绕过反爬虫机制到效率提升，我们都做了一系列的研究和总结。在提取前人的框架的优点和精华的同时，我们也加入了很多的自己独创的智能爬虫机制，如Mobile端爬取，超时延时处理。

在保证提取到一定数量的微博的同时，还兼顾到了效率，同时保证微博账号不被封杀。相对于大多数的微博爬虫开源项目来说，我们的爬虫是稳定的，高效的。

## 高效的数据提取手段

为了能够高效的提取微博上的数据，我们抛弃了Python的常用的一些第三方的文本解析库。虽然这些库好用，但是都普遍存在一个问题，那就是速度。这些第三方的库，为了能够兼容众多的文本环境，所以有很多的判断机制在里面，这样一来，提取文本的速度就慢了。对于小部分的文本，可能看不出问题。但是一旦到了内容众多的时候，它的劣势就凸显出来。为了解决这个问题，我们决定自己编写正则表达式。为此，我们建立起了一套规范，便于迅速的编写和调用相应的正则表达式。由于我们的正则表达式是自己根据具体的文本内容环境深度定制的，所以我们的提取效率很高，根据我们的测试，比第三方的库快了1000倍以上。

## WeiboRank分析算法

WeiboRank根据微博的具体情况，结合了PageRank的思想，实现了微博中用户、博文和评论3种数据模型的排序。该算法可以从大量的数据中发现重要的数据，再结合全文检索，可以实现精确的舆情信息获取。

## CLI与Web结合

命令行与Web界面相结合的交互系统。我们摒弃了传统的GUI界面，使用了CLI与Web相结合的方式对系统进行操作。

系统的主题运行于CLI上，可以方便地部署在任何计算机上。同时使用Web进行分析数据的检索，可以对任何安装有浏览器的计算机提供一致性的检索体验。

这种结合方式减少了部署难度，同时还提升了用户体验。

## 系统的并发实现

我们的系统大部分是IO密集型操作，CPU大部分时间都是在等待网络数据传输和数据库数据传输，十分地浪费资源。

为了能够快速，有效的提取微博上的数据，我们采用了多线程并发的形式进行爬虫。我们知道，近段时间，新浪微博对于爬虫的限制是越来越大，大有屏蔽一切网络爬虫的意思。所欲，对于我们的爬虫来说，速度上有了一定的限制。但是我们的目的是为了获取一定数量的微博文本信息。虽然爬虫速度受到限制，我们可以一次启动多个爬虫，对同一关键字进行爬取。在获得同等数量的微博文本内容的同时，减少爬虫的时间，提高了效率。

数据读写方面，我们也是用了多线程的方式，同时给线程加锁，避免了数据读写的冲突，极大的提高了系统的IO效率。

## 系统的可移植性

众所周知，Python是一门跨平台的语言。Python编写的程序，不用修改过多的代码，便能够在Windows、Linux和MacOS上无缝的移植运行。本系统开始构建的时候，就考虑到了移植性的问题。所以代码中所调用的API，都尽可能的考虑到了不同的系统，尽可能在兼容性上做到完美。对于调用的第三方的库，我们也都考虑了在内，对于可能会出现错误的地方，我们都做了异常处理。一切的努力，都是为了更好的跨平台运行。

事实上，我们的付出达到了应该有的目的。经过我们的测试，本系统能够在Windows，Linux和MacOS完美的运行。

# 总结

新浪微博自从2009年推出开始，截止2014年9月30日，微博月活跃用户数已经达到1.67亿人。这是一个非常庞大的数字，而在这庞大的基数下，微博的数量更为可观。毋庸置疑，我们现在正处在一个伟大的时代——互联网时代。在这个时代，我们的可以在网络上交友，互动，分享自己的生活，倾述自己的情感。发表对于热门事件的看法。小人物的观点，也能够在微博上被无限的放大，广泛的传播。这个时代不同于过去的年代，以往只能够通过报纸、传闻听取到名人大家的观点，现在是一个言论自由，言论平等的时代。我们在网络上发表的话语观点，借助这互联网的无穷无尽的力量，广泛的传播在这个世界的每一个角落。

每天，人们在网络上发表的动态十分的多。在这个大数据的互联网时代，有效的提取数据，高效准确的对数据进行分析显现得特别的有意义。通过数据来对未知事物进行预测，这是一件伟大的举动。这让人充满着激情，每当想到自己正在做的是这么美妙的一件事，倍感兴奋。

本系统主要是基于微博，对相关主题的微博进行一定数量的采样，存入本地数据库。随后使用自己编写的WeiboRank算法对微博文本数据进行分析整合，得出微博舆情分析的结果。

为了实现这一目的，我们主要采用了微博爬虫，数据库存储，舆情分析算法，检索展示平台实现了这一个系统。我们把这个系统称为Xenon。为了能够保证系统能够正常高效的运行，我们做了很多的努力。这几年，新浪微博在国内一家独大，同时微博也意识到了微博数据的作用。微博开始启用反爬虫机制，尽可能的避免第三方获取到这些有价值的微博数据。我们在编写Xenon系统的过程中，微博更改了登陆认证的机制。一时间，网上的关于微博登陆的方法全部失效。这对于我们的打击十分的大。但是后来，我们决定尝试从手机端进行尝试，采取“农村包围城市”的策略，成功的登陆了微博。

随后便是如何有效的和反爬虫机制进行对抗，这是一个有趣的过程。具体实现在代码注释中有详细说明。接下来便是数据的存储，本着学习的态度，我们决定使用CentOS+MariaDB+phpMyadmin 作为数据存储环境，因为其开源免费的特性，我们能够从中学习到很多的知识。

本系统的另一个重点是舆情分析算法，我们在提取微博数据提取到的数据都是在算法中使用到的。为了实现这一算法，我们参考了大量的书籍和网络上的资料。通过参考其他的算法，取长补短。终于完成了这一算法。虽然称不上特别完美，但是毕竟是我们自己设计的。我们对此感到自豪。

对于数据的检索展示平台，我们采用了Web展示。同时在前端上也对Bootstrap学习了一番，并将其用于项目中。一切的选择都是本着学习的态度。所以，在编写程序的过程中，我们接触到了很多以前从未了解的技术。我们认为，这正是这次比赛的目的：在完成作品的过程中，不断的学习，不断的完善。

在此，感谢老师的耐心指导。同时，在这几个月里，我们学习到了很多的东西，受益匪浅。

# 参考文献

1. []许鑫, 章成志. 互联网舆情分析及应用研究[J]. 情报科学, 2008 (8): 1194-1200. [↑](#endnote-ref-1)
2. []徐远超, 刘江华, 刘丽珍, 等. 基于 Web 的网络爬虫的设计与实现[J]. 微计算机信息, 2007 (21): 119-121. [↑](#endnote-ref-2)
3. []范金城, 梅长林. 数据分析[M]. 科学出版社, 2002. [↑](#endnote-ref-3)
4. [] wiki. 自然语言处理[EB/OL]. http://zh.wikipedia.org/wiki/自然语言处理, 2015-06-05/2015-06-07. [↑](#endnote-ref-4)
5. [] Bird S, Klein E, Loper E. Natural language processing with Python[M]. " O'Reilly Media, Inc.", 2009. [↑](#endnote-ref-5)
6. []邵峰晶, 于忠清. 数据挖掘原理与算法[M]. DynoMedia Inc., 2003.

   MLA [↑](#endnote-ref-6)
7. [] wiki. PageRank[EB/OL]. http://zh.wikipedia.org/wiki/PageRank, 2015-05-29/2015-06-07. [↑](#endnote-ref-7)