

## 摘 要

微博在这短短几年时间内发展迅速，成为了传播信息的一种很重要的载体，仅新浪微博注册用户早已超过 3 亿，每日发博量超过 1 亿条，微博是研究大量舆情信息的最佳场所。微博中如此大的信息量，必然要对信息进行甄选，这是很自然的需求，所以以原始驱动为出发点，从数亿网民之中找到热点话题。监督规范网络行为，净化网络环境，更重要的还能从中获取各类有用信息，进行诸如商业价值(用户兴趣挖掘)，信息传播学(网络拓扑与热点追踪)，以及一些社会学方面的研究。从而能够满足人们的需求，同时对于社会的和谐、网络舆论生态的健康、国家的发展都有重要的现实意义。

微博舆情分析系统就是来实现热点事件的挖掘与分析，本文首先介绍了舆情分析的理论基础和一些相关算法，然后是对整个系统从零到整的完整开发记录，最后是通过本软件分析出微博的热点事件。

**关键词：**微博，舆情分析，向量空间模型，k-means 聚类算法

## **Abstract**

Weibo in this few years has developed rapidly, and become a important tools th at spread information, only the user of sina weibo has more than 300 million, daily s end Weibo are more than 100 million, weibo is the best place to study a lot of public information. The weibo so large amount of information, is bound to the infor mation select ion, it is very natural demand, so to the original drive as a starting point, find h ot topi cs from among hundre ds of millions of Internet users. Supervision and stand ard netw ork behavior, purify network environment, the more important of this can o btain all ki nds of useful information, such as business value (user) interested in mini ng, informat ion and communication (network topology and the hot spot tracking), as well as some sociolo gical research. So that they can meet people's needs, at the sam e time for soci al harmony, ecological health network public opinion, the country's d evelopment has important practical significance.

Microblogging public opinion analysis system is to implement the hot issues mi ning and analysis, firstly ,this paper introduces the basic theory of public opinion ana lysis and some related algorithms, and then introduce how to build the whole system, finally, through the software analysis the microblogging hot issues.

**KeyWords:** Weibo, public opinion analysis, vector space model, k-means

目 录	
第一章 引言 .....	1
1.1. 研究背景及意义 .....	1
1.2. 国内外相关问题研究现状 .....	1
1.3. 论文的理论意义与实用价值 .....	3
第二章 相关技术介绍 .....	4
2.1 开发平台简述 .....	4
2.1.1 Eclipse 简介 .....	4
2.1.2 新浪微博 api .....	4
2.2 OAuth2.0 简介 .....	5
2.3 空间向量模型 .....	6
2.4 文本聚类算法 .....	9
2.5 数据库方案 .....	11
2.5.1 MYSQL 简介 .....	11
2.5.2 MYSQL 背景 .....	11
2.5.3 MYSQL 系统特性 .....	12
第三章 系统设计 .....	13
3.1 热点分析策略 .....	13
3.2 系统总体设计 .....	13
3.3 系统模块详细设计 .....	15
3.3.1 数据获取模块 .....	15
3.3.2 数据分析模块 .....	25
3.4 数据库设计方案 .....	30
第四章 系统实现 .....	32
4.1 数据获取模块实现 .....	32

---

4.1.1 核心代码.....	32
4.1.2 相关时序图.....	33
4.2 数据处理模块实现.....	35
4.2.1 核心代码.....	35
4.2.2 相关时序图.....	36
第五章 系统运行结果.....	39
5.1 源码结构展示.....	39
5.2 数据抓取模块运行展示.....	39
5.3 数据分析模块.....	42
第六章 总结与展望.....	45
6.1 系统主要作用.....	45
6.2 系统难点与解决.....	45
致谢.....	47
参考文献.....	48
附录.....	50

## 第一章 引言

### 1.1. 研究背景及意义

微博在这短短几年时间内发展迅速，成为了群主传播信息的一种很重要的载体，仅新浪微博注册用户早已超过 3 亿，每日发博量超过 1 亿条，微博是研究大量舆情信息的最佳场所。微博中如此大的信息量，必然要对信息进行甄选，这是很自然的需求，所以以原始驱动为出发点，从数亿网民之中找到热点话题。监督规范网络行为，净化网络环境，更重要的还能从中获取各类有用信息，进行诸如商业价值(用户兴趣挖掘)，信息传播学(网络拓扑与热点追踪)，以及一些社会学方面的研究。从而能够满足人们的需求，同时对于社会的和谐、网络舆论生态的健康、国家的发展都有重要的现实意义。

本系统侧重于关于人际关系网络的热点分析，目标是可以迅速找到用户周边圈子内的热点事件，依靠周围用户周边圈子的黏着度来满足用户的信息获取需求。

### 1.2. 国内外相关问题研究现状

舆情挖掘是以话题检测与跟踪的相关技术为理论前提进行发展的。1996 年，为了更好的解决网络背景下事件的提取问题，美国国防高级研究计划局(DARPA)提出了话题检测与跟踪(Topic Detection and Tracking)即TDT 任务。TDT 就是面向多语言文本和语音形式的新闻报道，主要从事报道边界自动识别、锁定和收集突发性新闻话题、跟踪话题发展以及跨语言检测与跟踪等相关任务。为了更好的解决TDT 的相关问题，国内外诸多研究团队提出了许多不同算法，并取得了较为不错的结果。

在国外 TDT 的早期研究中，主要是以Allan[1]和Lam[2]的研究为代表。其中 Allan 等人提出了single-pass 算法，但是因为随着事件的发展，同一事件可能会因为报道侧重点的不同而被定义为不同的事件，因而这种方法会产生很高的误判。Lam 等人提出用新闻簇来代替单一文档作为类，这种方法因为事件的重点分散同样存在很高的误判。

在 TDT 后续的研究中，对命名实体的改进技术是一个主要的方向。Yang[3]等人最早引入了命名实体的概念。他们对命名实体进行分类，直接对地点类的命名实体特征赋予四倍的权重。这种权重计算存在很大的主观因素，使得该算法在大部分环境下不稳定。Giridhar[4]等人随后为了改善这种不稳定性，提出将报道分

为包含全部特征的向量、仅包含NE 特征的向量和排除NE 特征的向量三种向量空间。提出要分情况应用命名实体。但由于分类依旧过于简单，所以在结果的准确率上受到很大的制约。很多研究对文本表示形式进行了改进。Yang[5]等人在划分报道类别的基础上，只选择对类别中最优的报道进行数学描述。并且指出在离线检测的情况下应用组平均的算法（GAC）效果较好；而对于在线式检测则应用单路径聚类（INCR）效果较好。Brants[6]等人在基于原有算法思想的基础上对文档间相似度和增量式TFIDF 模型进行进一步的改进。还有一些研究是对关键词提取方法的改进。比如：Christian Wartena 和Rogier Brussee[7]提出通过提取关键词并对关键词基于不同的相似度量应用k-bisecting 算法进行聚类，并提出使用 Jensen-Shannon 距离来代替余弦相似度量可以得到更好的结果。该改进算法在 Wikipedia 中进行检测显示该聚类与Wikipedia 固有的分类保持高度的一致性。TDT 算法中的一个重要的发展就是语义链的研究，Stocks[8]在基于语义链的思想上提出从两种对文本内容的不同表示中分别提取特征，并结合建立文本表示。Hath[9]等人在构造语义链时结合词典信息（WordNet）和文本的上下文信息，在语义链的文本表示下应用单路径聚类解决事件检测的问题。Ulli Waltinger[10]提出一种新概念的语义链模型，即建立了一种基于在线的应用多语料库的模型，来弥补单一语料库资料有限片面的缺点，在一定程度上改善了词义混淆的现象。相对于国外在该领域的研究，国内的相关研究起步较晚。贾自艳[11]较早的在TDT 中引入了命名实体的方法，他将文本中的信息按人名，地名等特征进行分类标记，并分别赋予不同类别不同的加权系数，在计算文本特征权重时，将特征的词频和所属类别的加权系数的乘积作为最后的权重。清华大学的张阔[12]等人利用2分布对TDT语料中命名实体和话题类别间的关联性进行统计。详细的区分了各种类别的NE 同各类新闻报道的联系。该算法在TDT 数据集上的结果取得了很大的改善。除了对命名实体的改进外，国内很多研究还侧重于改善文本的应用质量。哈尔滨工业大学的洪宇[13]等人提出将话题化为不同的子话题的分治匹配方法，通过句子间相关度的下降坡度识别子话题边界。其后在各子话题间建立话题识别模型，不仅有效地避免了由于子话题互为噪声而产生的歧义，而且对于一篇报道多个话题的现象提出了一种有效地切分方法。在语义链方向，文献[14]提出将文档结构和语义特性结合起来提取重要词语和文档聚类，并且将传统的语义聚类模型进行了改进，对标题，关键词和摘要的相似度进行单独计算并考虑到聚类模型之中。文献[15]的主要贡献是应用DSLM

（dependency structure language model）模型[16]代替传统的一元或二元模型，并将

时间信息添加进模型中来增加新闻与相应话题之间相似度的准确性。其中DSLMM模型克服了TDT中一元和二元语言模型的表述制，它的结构是从数学模型角度进行描述的，并扩展一些简单的语义信息。话题模型层次化和结构化是目前TDT领域重要的研究方向。其中，层次化面向将同一话题下的相关报道组织为宏观到具体的层次体系；结构化则侧重挖掘和表征同一话题的不同侧面。国内尝试建立层次化话题模型的研究来自骆卫华[17]和张阔[18]。前者首先基于时序关系对报道分组，然后进行组内自底向上的层次聚类，最后按时间顺序采用单路径聚类策略合并相关类；后者则面向报道全集建立层次化的索引树，树中第一层节点对应特定话题，而其子树则描述了该话题的层次体系，其建树过程基于输入的报道相对于树中各层次节点是否为新事件进行组织[19]。

国内在该领域还有其它一些较新的研究方向，比如：基于容错集进行话题聚类[20]通过容错集模型（TRSM）有效的减少了在脱机情况下，由“噪音”引起的话题漂移；文献[21]提出了针对互联网环境下论坛网页的方法，通过设置两层选择框架综合考虑文本间的相似度和用户的活跃度，来改善论坛网页中语言非正式带来的困难；文献[22]提出了处理新闻事件的改进算法，文中指出，新闻事件不同于话题的是随着时间持续一段时间内话题会变得没有价值，而新闻事件会转化为有价值的记录。

### 1.3. 论文的理论意义与实用价值

本系统以新浪微博为信息源，以新浪微博的一个用户为起始点，获取与挖掘以该用户为中心的人际网络之间的热点事件，以匹配到用户的最佳兴趣点。不仅仅进行诸如商业价值(用户兴趣挖掘)，信息传播学(网络拓扑与热点追踪)，以及一些社会学方面的研究。从而能够满足人们的需求，同时对于社会的和谐、网络舆论生态的健康、国家的发展都有重要的现实意义。

## 第二章 相关技术介绍

### 2.1 开发平台简述

#### 2.1.1 Eclipse 简介

课题是在 Eclipse 上编译和调试的应用程序，下面简单介绍 Eclipse 信息：

Eclipse 简介：Eclipse 是一个开放源代码的、基于 Java 的可扩展开发平台，对 eclipse 而言，它仅是是一个框架和一组服务，可以通过各种各样的插件和组件来搭建开发环境。

Eclipse 的目标不仅是将其当做 Java IDE 来使用，还要利用其插件开发环境（Plug-in Development Environment, PDE），通过开发各种插件来扩展 Eclipse 的软件开发人员，这使软件开发人员能构建与 Eclipse 环境无缝集成的工具。

Eclipse 最初是替代由 IBM 公司开发的价值四千万美金的商业软件 Visual Age for Java 的下一代 IDE 开发环境，2001 年 11 月交给非营利软件供应商联盟 Eclipse 基金会（Eclipse Foundation）管理。2003 年，Eclipse 3.0 选择 OSGi 服务平台规范为运行时架构。

Eclipse 采用的技术是 IBM 公司开发的（SWT），这种技术基于 Java 的窗口组件，类似 Java 本身提供的 AWT 和 Swing 窗口组件。Eclipse 的用户界面还使用了 GUI 中间层 JFace，这样一来大大简化了基于 SWT 的应用程序的构建。

Eclipse 的插件机制是轻型软件组件化架构。在 RCP 平台上，Eclipse 使用软件人员开发的各种插件来提供所有的附加功能，例如 android 的 ADT 插件，这使得软件开发人员能使用 Eclipse 来编译 android 应用软件，完成软件的编译测试还有调试等工作。还有的插件不仅能支持 JAVA，还可以支持 C/C++（CDT）、Perl、Ruby，Python 和数据库开发。这种插件架构带来非常多的便利，能够支持任意的扩展使其加入到当前的开发环境中，缩短开发周期，节约金钱。

#### 2.1.2 新浪微博 api

新浪微博 API 是新浪为了方便第三方应用接入微博这个大系统而开放的 API，有了这个 API 我就不用重新写爬虫程序获取新浪微博上的大量微博，直接调用新浪微博的 API 完成我的扒取任务。

新浪微博 api 的验证登陆方式是 OAuth2.0 方式，现在对 OAuth2.0 进行简要介



绍。

## 2.2 OAuth2.0 简介

由于需要使用新浪 api 替代网络爬虫来进行数据收集，所以在介绍了新浪 api 之后，我们对新浪 api 的代理接入方法做介绍即 OAuth2.0

### 1) OAuth 前言

OAuth 1.0 已经在 IETF 尘埃落定，编号是 RFC5849，这也标志着 OAuth 已经正式成为互联网标准协议。OAuth 2.0 早已开始讨论和建立的草案。OAuth2.0 很可能是下一代的“用户验证和授权”标准。现在百度开放平台，腾讯开放平台，新浪微博开放平台等大部分的开放平台都是使用的 OAuth 2.0 协议作为支撑。

OAuth（开放授权）是一个开放标准，允许用户让第三方应用访问该用户在某一网站上存储的私密的资源（如照片，视频，联系人列表），而无需将用户名和密码提供给第三方应用。

### 2) OAuth

允许用户提供一个令牌，而不是用户名和密码来访问他们存放在特定服务提供者的数据。每一个令牌授权一个特定的网站（例如，视频编辑网站）在特定的时段（例如，接下来的 2 小时内）内访问特定的资源（例如仅仅是某一相册中的视频）。这样，OAuth 允许用户授权第三方网站访问他们存储在另外的服务提供者上的信息，而不需要分享他们的访问许可或他们数据的所有内容。

OAuth 是 OpenID 的一个补充，但是完全不同的服务。

OAuth 2.0 是 OAuth 协议的下一版本，但不向后兼容 OAuth 1.0。OAuth 2.0 关注客户端开发者的简易性，同时为 Web 应用，桌面应用和手机，和起居室设备提供专门的认证流程。

Facebook 的新的 Graph API 只支持 OAuth 2.0，Google 在 2011 年 3 月亦宣布 Google API 对 OAuth 2.0 的支援。

### 3) 认证和授权过程

在认证和授权的过程中涉及的三方包括：

1、**服务提供方**，用户使用服务提供方来存储受保护的资源，如照片，视频，联系人列表。

2、**用户**，存放在服务提供方的受保护的资源的拥有者。

3、**客户端**，要访问服务提供方资源的第三方应用，通常是网站，如提供照片打印服务的网站。在认证过程之前，客户端要向服务提供者申请客户端标识。

使用 OAuth 进行认证和授权的过程如下所示：

1. 用户访问客户端的网站，想操作用户存放在服务提供方的资源。
2. 客户端向服务提供方请求一个临时令牌。
3. 服务提供方验证客户端的身份后，授予一个临时令牌。
4. 客户端获得临时令牌后，将用户引导至服务提供方的授权页面请求用户授权。在这个过程中将临时令牌和客户端的回调连接发送给服务提供方。
5. 用户在服务提供方的网页上输入用户名和密码，然后授权该客户端访问所请求的资源。
6. 授权成功后，服务提供方引导用户返回客户端的网页。
7. 客户端根据临时令牌从服务提供方那里获取访问令牌。
8. 服务提供方根据临时令牌和用户的授权情况授予客户端访问令牌。
9. 客户端使用获取的访问令牌访问存放在服务提供方上的受保护的资源。

#### 4) 历史回顾

OAuth 1.0 在 2007 年的 12 月底发布并迅速成为工业标准。

2008 年 6 月，发布了 OAuth 1.0 Revision A，这是个稍作修改的修订版本，主要修正一个安全方面的漏洞。

2010 年四月，OAuth 1.0 的终于在 IETF 发布了，协议编号 RFC 5849。

OAuth 2.0 的草案是在 2011 年 5 月初在 IETF 发布的。

OAuth is a security protocol that enables users to grant third-party access to their web resources without sharing their passwords.

OAuth 是个安全相关的协议，作用在于，使用户授权第三方的应用程序访问用户的 web 资源，并且不需要向第三方应用程序透露自己的密码。

OAuth 2.0 是个全新的协议，并且不对之前的版本做向后兼容，然而，OAuth 2.0 保留了与之前版本 OAuth 相同的整体架构。

这个草案是围绕着 OAuth2.0 的需求和目标，历经了长达一年的讨论，讨论的参与者来自业界的各个知名公司，包括 Yahoo!, Facebook, Salesforce, Microsoft, Twitter, Deutsche Telekom, Intuit, Mozilla, and Google。

## 2.3 空间向量模型

向量空间模型（VSM: Vector Space Model）由 Salton 等人于 20 世纪 70 年代提出，并成功地应用于著名的 SMART 文本检索系统。

VSM 概念简单，把对文本内容的处理简化为向量空间中的向量运算，并且它

以空间上的相似度表达语义的相似度，直观易懂。当文档被表示为文档空间的向量，就可以通过计算向量之间的相似性来度量文档间的相似性。文本处理中最常用的相似性度量方式是余弦距离。

$M$  个无序特征项  $t_i$ ，词根/词/短语/其他每个文档  $d_j$  可以用特征项向量来表示  $(a_{1j}, a_{2j}, \dots, a_{Mj})$  权重计算， $N$  个训练文档  $AM \times N = (a_{ij})$  文档相似度比较 1) Cosine 计算，余弦计算的好处是，正好是一个介于 0 到 1 的数，如果向量一致就是 1，如果正交就是 0，符合相似度百分比的特性，余弦的计算方法为，向量内积/各个向量的模的乘积。2) 内积计算，直接计算内积，计算强度低，但是误差大。

向量空间模型（或词组向量模型）是一个应用于信息过滤，信息撷取，索引以及评估相关性的代数模型。SMART 是首个使用这个模型的信息检索系统。

文件（语料）被视为索引词（关键词）形成的多次元向量空间，索引词的集合通常为文件中至少出现过一次的词组。

搜寻时，输入的检索词也被转换成类似于文件的向量，这个模型假设，文件和搜寻词的相关程度，可以经由比较每个文件(向量)和检索词（向量）的夹角偏差程度而得知。

实际上，计算夹角向量之间的余弦比直接计算夹角容易：

余弦为零表示检索词向量垂直于文件向量，即没有符合，也就是说该文件不含此检索词。

通过上述的向量空间模型，文本数据就转换成了计算机可以处理的结构化数据，两个文档之间的相似性问题转变成了两个向量之间的相似性问题。

本文中使用的空间向量模型主要用于下面 2 个目的

1. 使用向量表示微博文档
2. 借以来判断 2 条微博的相似性

下面详细介绍如何实现

### 1. 向量表示文档

首先，我们知道任何一条微博都是由单词构成的，我们在最初之时就构建了一个包含全部文档集合的词典，只要文档集合中出现了的单词，我们的词典之中就会有。于是我们根据词典里面单词的个数  $N$  构建出一个  $N$  维的向量  $\langle a_1, a_2, a_3, \dots, a_n \rangle$ ，对于每一个文档都有这样的转换：

文档  $A$  对应的向量  $A = \langle a_1, a_2, a_3, \dots, a_n \rangle$ ；若  $a_i \neq 0$ ，则表示  $a_i$  这个单词在这个文档中出现，反之则不出现，这样我们就将文档向量化了。

### 2. 相似性计算

余弦距离经常被用在文本相似性比较中。余弦结果为一个0到1的数，1表示向量一致，0则表示正交，符合相似性百分比的特性。不同文档长度的归一化是通过计算向量内积与文档向量的长度的比值实现的，即前提是忽略文档向量长度的影响。

比如：比较的两个文档是

$$D = (d_1, d_2, d_3, \dots, d_n)$$

$$Q = (q_1, q_2, q_3, \dots, q_n)$$

则他们之间的距离是

$$\text{Cos}(D, Q) = \frac{\sum_{i=1}^n (d_i \times q_i)}{\sqrt{\sum_{i=1}^n (d_i)^2 \sum_{i=1}^n (q_i)^2}}$$

但是要是仅仅由余弦来判定距离那是有一定的误差的，原因如下：

在构建向量的时候单词之间的权重是不一样的，我们直接按照单词出现或者不出现来构建向量是与原本的向量有一定的误差，所以我们还需要按照权值给文档向量化。

这里我们使用TF—IDF (Term Frequency—Inverse Document Frequency) 权重计算相结合的方法

TF-IDF (term frequency-inverse document frequency) 是一种用于资讯检索与资讯探勘的常用加权技术。TF-IDF 是一种统计方法，用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。TF-IDF 加权的各种形式常被搜索引擎应用，作为文件与用户查询之间相关程度的度量或评级。除了 TF-IDF 以外，因特网上的搜索引擎还会使用基于链接分析的评级方法，以确定文件在搜寻结果中出现的顺序。

TFIDF 的主要思想是：如果某个词或短语在一篇文章中出现的频率 TF 高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。TFIDF 实际上是：TF \* IDF，TF 词频(Term Frequency)，IDF 反文档频率(Inverse Document Frequency)。TF 表示词条在文档 d 中出现的频率。IDF 的主要思想是：如果包含词条 t 的文档越少，也就是 n 越小，IDF 越大，则说明词条 t 具有很好的类别区分能力。如果某一类文档 C 中包含词条 t 的文档数为 m，而其

它类包含  $t$  的文档总数为  $k$ ，显然所有包含  $t$  的文档数  $n=m+k$ ，当  $m$  大的时候， $n$  也大，按照 **IDF** 公式得到的 **IDF** 的值会小，就说明该词条  $t$  类别区分能力不强。但是实际上，如果一个词条在一个类的文档中频繁出现，则说明该词条能够很好代表这个类的文本的特征，这样的词条应该给它们赋予较高的权重，并选来作为该类文本的特征词以区别与其它类文档。这就是 **IDF** 的不足之处。

**TFIDF** 的原理是：在一份给定的文件里，

词频 (term frequency, **TF**) 指的是某一个给定的词语在该文件中出现的次数。这个数字通常会被归一化，以防止它偏向长的文件。(同一个词语在长文件里可能会比短文件有更高的词频，而不管该词语重要与否。)

逆向文件频率 (inverse document frequency, **IDF**) 是一个词语普遍重要性的度量。某一特定词语的 **IDF**，可以由总文件数目除以包含该词语之文件的数目，再将得到的商取对数得到。

某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的 **TF-IDF**。因此，**TF-IDF** 倾向于保留文档中较为特别的词语，过滤常用词。

## 2.4 文本聚类算法

文本聚类 (Text clustering) 文档聚类主要是依据著名的聚类假设：同类的文档相似度较大，而不同类的文档相似度较小。作为一种无监督的机器学习方法，聚类由于不需要训练过程，以及不需要预先对文档手工标注类别，因此具有一定的灵活性和较高的自动化处理能力，已经成为对文本信息进行有效地组织、摘要和导航的重要手段，为越来越多的研究人员所关注。

文本聚类的算法主要有如下几类：

### 1. 划分法 (partitioning methods)

给定一个有  $N$  个元组或者纪录的数据集，分裂法将构造  $K$  个分组，每一个分组就代表一个聚类， $K < N$ 。而且这  $K$  个分组满足下列条件：

- (1) 每一个分组至少包含一个数据纪录；
- (2) 每一个数据纪录属于且仅属于一个分组（注意：这个要求在某些模糊聚类算法中可以放宽）；

对于给定的  $K$ ，算法首先给出一个初始的分组方法，以后通过反复迭代的方法改变分组，使得每一次改进之后的分组方案都较前一次好，而所谓好的标准就是：同一分组中的记录越近越好，而不同分组中的纪录越远越好。

使用这个基本思想的算法有：K-MEANS 算法、K-MEDOIDS 算法、CLARANS 算法；

## 2. 层次法 (hierarchical methods)

这种方法对给定的数据集进行层次似的分解，直到某种条件满足为止。具体又可分为“自底向上”和“自顶向下”两种方案。例如在“自底向上”方案中，初始时每一个数据纪录都组成一个单独的组，在接下来的迭代中，它把那些相互邻近的组合成一个组，直到所有的记录组成一个分组或者某个条件满足为止。代表算法有：BIRCH 算法、CURE 算法、CHAMELEON 算法等；

## 3. 基于密度的方法 (density-based methods)

基于密度的方法与其它方法的一个根本区别是：它不是基于各种各样的距离的，而是基于密度的。这样就能克服基于距离的算法只能发现“类圆形”的聚类的缺点。这个方法的指导思想就是，只要一个区域中的点的密度大过某个阈值，就把它加到与之相近的聚类中去。代表算法有：DBSCAN 算法、OPTICS 算法、DENCLUE 算法等；

## 4. 基于网格的方法 (grid-based methods)

这种方法首先将数据空间划分成为有限个单元 (cell) 的网格结构，所有的处理都是以单个的单元为对象的。这么处理的一个突出的优点就是处理速度很快，通常这是与目标数据库中记录的个数无关的，它只与把数据空间分为多少个单元有关。代表算法有：STING 算法、CLIQUE 算法、WAVE-CLUSTER 算法；

## 5. 基于模型的方法 (model-based methods)

基于模型的方法给每一个聚类假定一个模型，然后去寻找一个能很好的满足这个模型的数据集。这样一个模型可能是数据点在空间中的密度分布函数或者其它。它的一个潜在的假定就是：目标数据集是由一系列的概率分布所决定的。通常有两种尝试方向：统计的方案和神经网络的方案；

这里我们使用的 k-means 算法进行聚类：

k-means 描述如下：

K-means 算法是很典型的基于距离的聚类算法，采用距离作为相似性的评价指标，即认为两个对象的距离越近，其相似度就越大。该算法认为簇是由距离靠近的对象组成的，因此把得到紧凑且独立的簇作为最终目标。k 个初始类聚类中心点的选取对聚类结果具有较大的

影响，因为在该算法第一步中是随机的选取任意  $k$  个对象作为初始聚类的中心，初始地代表一个簇。该算法在每次迭代中对数据集中剩余的每个对象，根据其与其他各个簇中心的距离将每个对象重新赋给最近的簇。当考察完所有数据对象后，一次迭代运算完成，新的聚类中心被计算出来。如果在一次迭代前后， $J$  的值没有发生变化，说明算法已经收敛。

算法过程如下：

- 1) 从  $N$  个文档随机选取  $K$  个文档作为质心
- 2) 对剩余的每个文档测量其到每个质心的距离，并把它归到最近的质心的类
- 3) 重新计算已经得到的各个类的质心
- 4) 迭代 2~3 步直至新的质心与原质心相等或小于指定阈值，算法结束

具体如下：

输入： $k$ ,  $data[n]$ ;

- (1) 选择  $k$  个初始中心点，例如  $c[0]=data[0], \dots, c[k-1]=data[k-1]$ ;
- (2) 对于  $data[0] \dots data[n]$ ，分别与  $c[0] \dots c[k-1]$  比较，假定与  $c[i]$  差值最少，就标记为  $i$ ;
- (3) 对于所有标记为  $i$  点，重新计算  $c[i]=\{\text{所有标记为 } i \text{ 的 } data[j] \text{ 之和}\} / \text{标记为 } i \text{ 的个数}$ ;
- (4) 重复 (2) (3)，直到所有  $c[i]$  值的变化小于给定阈值。

## 2.5 数据库方案

该系统的数据库服务器采用比较适合中型数据量的 mysql。

### 2.5.1 MYSQL 简介

MySQL 是一个开放源码的小型关联式数据库管理系统，开发者为瑞典 MySQL AB 公司。目前 MySQL 被广泛地应用在 Internet 上的中小型网站中。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，许多中小型网站为了降低网站总体拥有成本而选择了 MySQL 作为网站数据库。

### 2.5.2 MYSQL 背景

MySQL 最初的开发者的意图是用 mSQL 和他们自己的快速低级例程 (ISAM) 去连接表格。经过一些测试后，开发者得出结论：mSQL 并没有他们需要的那么快和灵

活。这导致了一个使用几乎和 mSQL 一样的 API 接口的用于他们的数据库的新的 SQL 接口的产生，这样，这个<sup>[1]</sup>API 被设计成允许为用于 mSQL 而写的第三方代码更容易移植到 MySQL。

MySQL 这个名字是怎么来的已经不清楚了。基本指南和大量的库和工具带有前缀“my”已经有 10 年以上，而且不管怎样，MySQL AB 创始人之一 Michael Widenius 的女儿也叫 My。这两个到底是哪一个给出了 MySQL 这个名字至今依然是个迷，包括开发者在内也不知道。

MySQL 的海豚标志的名字叫“sakila”，代表速度、力量、精确，它是由 MySQL AB 的创始人从用户在“海豚命名”的竞赛中建议的大量的名字表中选出的。获胜的名字是由来自非洲斯威士兰的开源软件开发者 Ambrose Twebaze 提供。根据 Ambrose 所说，Sakila 来自一种叫 SiSwati 的斯威士兰方言，也是在 Ambrose 的家乡乌干达附近的坦桑尼亚的 Arusha 的一个小镇的名字。

2008 年 1 月 16 号 MySQL AB 被 Sun 公司收购。而 2009 年，SUN 又被 Oracle 收购。就这样如同一个轮回，MySQL 成为了 Oracle 公司的另一个数据库项目。

### 2.5.3 MYSQL 系统特性

1. 使用 C 和 C++ 编写，并使用了多种编译器进行测试，保证源代码的可移植性
2. 支持 AIX、FreeBSD、HP-UX、Linux、Mac OS、NovellNetware、OpenBSD、OS/2 Wrap、Solaris、Windows 等多种操作系统
3. 为多种编程语言提供了 API。这些编程语言包括 C、C++、Python、Java、Perl、PHP、Eiffel、Ruby 和 Tcl 等。
4. 支持多线程，充分利用 CPU 资源
5. 优化的 SQL 查询算法，有效地提高查询速度
6. 既能够作为一个单独的应用程序应用在客户端服务器网络环境中，也能够作为一个库而嵌入到其他的软件中。
7. 提供多语言支持，常见的编码如中文的 GB 2312、BIG5，日文的 Shift\_JIS 等都可以用作数据表名和数据列名。
8. 提供 TCP/IP、ODBC 和 JDBC 等多种数据库连接途径。
9. 提供用于管理、检查、优化数据库操作的管理工具。
10. 支持大型的数据库。可以处理拥有上千万条记录的大型数据库。
11. 支持多种存储引擎。



## 第三章 系统设计

### 3.1 热点分析策略

本系统是通过根据入口用户进行分析，力求得到入口用户最喜爱的热点话题，新浪微博的一般用户都是信息接收者，而新浪微博的大量加 V 的明星用户有很多，他们是信息的产生者，会有巨量的粉丝关注，并且有大量的粉丝对其每条微博进行评论。

我们可以进行假设，若是一条微博有巨量的评论数，无论是好评还是差评，那么它的关注度一定是很高的，可以从另一个方面反映出大家对这条微博所描述的事件的兴趣度，所以本系统的分析策略如下图描述

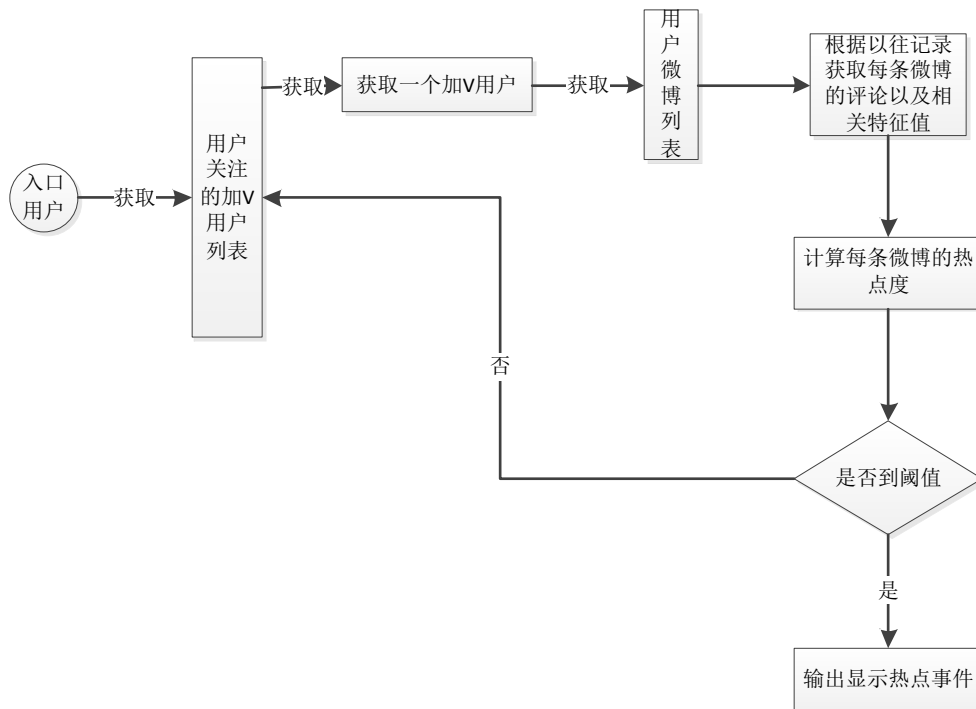


图 3-1 策略描述

### 3.2 系统总体设计

该系统重点在于数据的获取与分析，并未强调运算速度，故采用单机开发，所以本系统是 PC 上的一个应用程序。

总体设计框图如图 3-2 所示

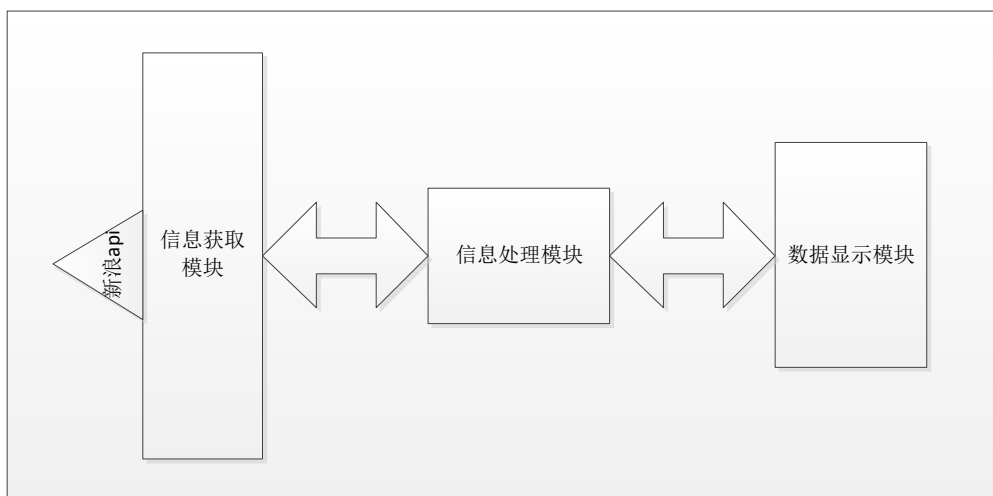


图 3-2 系统总体设计框图

如图所示该系统总分为 3 个模块：

- 信息获取模块：该模块从微博上获取大量相关数据，并且将这些数据进行简单的处理，建立倒排索引。
- 信息处理模块：该模块主要将信息获取模块处理过的数据进行深加工，进行聚类，以及热点评价。
- 数据显示模块：该模块将信息处理模块处理好的数据显示出来，供用户使用

### 3.3 系统模块详细设计

#### 3.3.1 数据获取模块

该模块从微博上获取大量相关数据，并且将这些数据进行简单的处理，最终建立倒排索引，插入数据库之中。内部模块如图 3-3

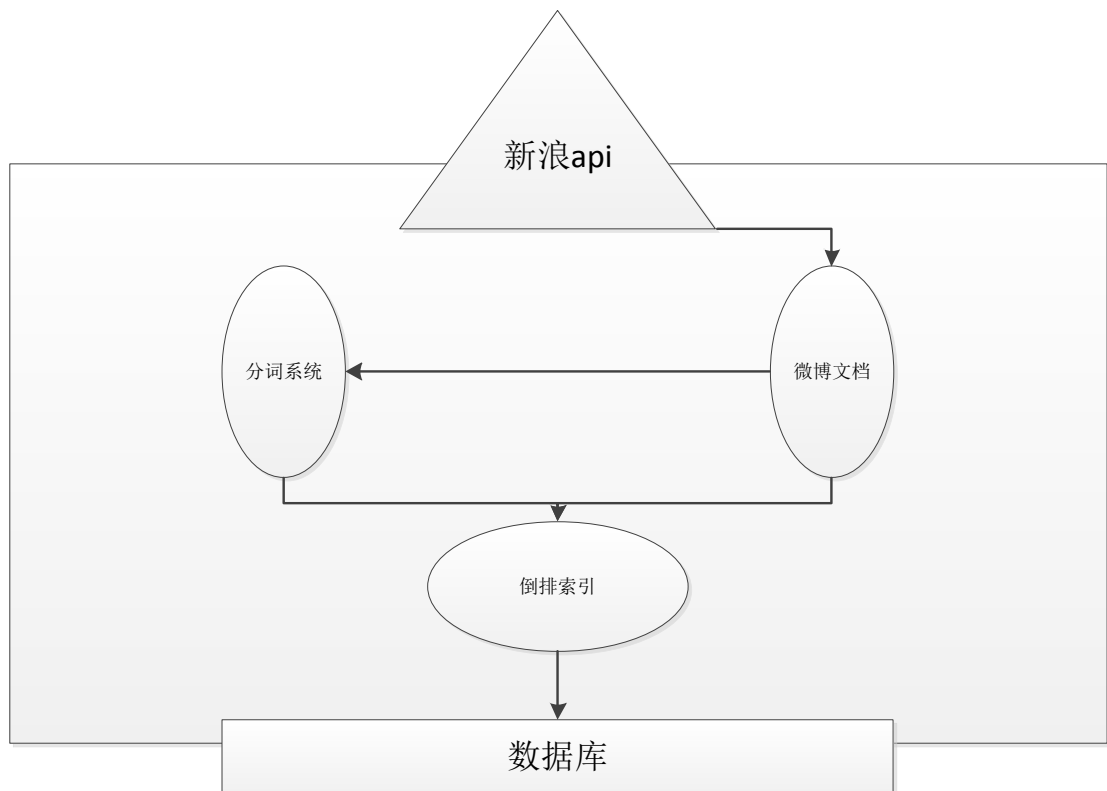


图 3-3 数据获取模块

- 新浪 api：获取数据源的模块，借用新浪微博的开放平台。
- 微博文档：暂时存放获取数据。
- 分词系统：建立词典专用。
- 倒排索引：处理好数据，以便后面数据分析用

##### 3.3.1.1 分词系统

微博是一条一条的文档，我们需要提取其特征指纹，我们才能将大量微博聚类，提取特征指纹，本文是使用空间向量模型，这意味着我们需要将微博划分为

大量的单个的单词，这里本系统采用 iK-analyzer 分词器。

IK Analyzer 是一个开源的，基于 java 语言开发的轻量级的中文分词工具包。从 2006 年 12 月推出 1.0 版开始，IKAnalyzer 已经推出了 4 个大版本。最初，它是以开源项目 Luence 为应用主体的，结合词典分词和文法分析算法的中文分词组件。从 3.0 版本开始，IK 发展为面向 Java 的公用分词组件，独立于 Lucene 项目，同时提供了对 Lucene 的默认优化实现。在 2012 版本中，IK 实现了简单的分词歧义排除算法，标志着 IK 分词器从单纯的词典分词向模拟语义分词衍化。

IK Analyzer 2012 特性:

1.采用了特有的“正向迭代最细粒度切分算法“，支持细粒度和智能分词两种切分模式；

2.在系统环境: Core2 i7 3.4G 双核, 4G 内存, window 7 64 位, Sun JDK 1.6\_29 64 位 普通 pc 环境测试, IK2012 具有 160 万字/秒 (3000KB/S) 的高速处理能力。

3.2012 版本的智能分词模式支持简单的分词排歧义处理和数量词合并输出。

4.采用了多子处理器分析模式, 支持: 英文字母、数字、中文词汇等分词处理, 兼容韩文、日文字符

5.优化的词典存储, 更小的内存占用。支持用户词典扩展定义。特别的, 在 2012 版本, 词典支持中文, 英文, 数字混合词语。

## 分词原理

主要有三种方法:

### 1) 基于词典匹配的分词方法

基于字典匹配的分词方法按照一定的匹配策略将输入的字符串与机器字典词条进行匹配, 如果在词典中找到当前字符串, 则成功匹配输出识别的词汇。按照匹配操作的扫描方向不同, 词典匹配分词方法又可以具体分为正向匹配和逆向匹配, 以及结合了两者的双向匹配算法; 按照不同长度有限匹配的情况, 可以分为最大 (最长) 匹配和最小 (最短) 匹配; 按照是否与词性标注过程相结合, 又可以分为单纯分词方法和分词与词性标注相结合的方法。集中常见的词典分词方法如下:

① 正向最大匹配

② 逆向最大匹配

③ 最少切分

实际引用中上述方法经常组合使用, 达到最好的效果, 从而衍生出了

结合正向最大匹配方法和逆向最大匹配的双向匹配分词法。由于中文分词最大的问题是歧义处理，结合中文语言的自身特色，经常采用逆向匹配的切分算法，处理的精度高于正向匹配，产生的切分歧义现象也少。

## 2) 基于语义理解的分词

基于语义理解的分词方法是模拟人脑对语言 and 句子的理解，达到识别词汇单元的效果。基本模式是把分词、句法、语义分析并行进行，利用句法和语义信息来处理分词的歧义。

一般结构中通常包括分词子系统、句法语义子系统、调度系统。在调度系统的协调下，分词子系统可以获得有关词、句子等的句法和语义信息，模拟人脑对句子的理解过程。基于语义理解的分词方法需要使用大量的语言知识和信息。

## 3) 基于词频统计的分词

这种做法基于人们对中文词语的直接感受。通常词是稳定的词的组合，因此在中文文章的上下文中，相邻的字搭配出现的频率越多，就越有可能形成一个固定的词。根据  $n$  元语法知识可以知道，字与字相邻同时出现的频率或概率能够较好的反应成词的可信度。

基于词频统计的分词方法只需要对语料中的字组频率进行统计，不需要切分词典，因而又叫做无词典分词法或者统计分词法。

### 3.3.1.2 倒排索引

倒排索引[23]源于实际应用中需要根据属性的值来查找记录。这种索引表中的每一项都包括一个属性值和具有该属性值的各记录的地址。由于不是由记录来确定属性值，而是由属性值来确定记录的位置，因而称为倒排索引(inverted index)。带有倒排索引的文件我们称为倒排索引文件，简称倒排文件(inverted file)。

在关系数据库系统里，索引是检索数据最有效率的方式，。但对于搜索引擎，它并不能满足其特殊要求：

1) 海量数据：搜索引擎面对的是海量数据，像 Google，百度这样大型的商业搜索引擎索引都是亿级甚至几千的网页数量，面对如此海量数据，使得数据库系统很难有效的管理。

2) 数据操作简单：搜索引擎使用的数据操作简单，一般而言，只需要增、删、改、查几个功能，而且数据都有特定的格式，可以针对这些应用设计出简单高效

的应用程序。而一般的数据库系统则支持大而全的功能，同时损失了速度和空间。最后，搜索引擎面临大量的用户检索需求，这要求搜索引擎在检索程序的设计上要分秒必争，尽可能的将大运算量的工作在索引建立时完成，使检索运算尽可能的少。一般的数据库系统很难承受如此大量的用户请求，而且在检索响应时间和检索并发度上都不及我们专门设计的索引系统。

倒排列表用来记录有哪些文档包含了某个单词。一般在文档集合里会有很多文档包含某个单词，每个文档会记录文档编号（DocID），单词在这个文档中出现的次数（TF）及单词在文档中哪些位置出现过等信息，这样与一个文档相关的信息被称做倒排索引项（Posting），包含这个单词的一系列倒排索引项形成了列表结构，这就是某个单词对应的倒排列表。右图是倒排列表的示意图，在文档集合中出现过的所有单词及其对应的倒排列表组成了倒排索引。

在实际的搜索引擎系统中，并不存储倒排索引项中的实际文档编号，而是代之以文档编号差值（D-Gap）。文档编号差值是倒排列表中相邻的两个倒排索引项文档编号的差值，一般在索引构建过程中，可以保证倒排列表中后面出现的文档编号大于之前出现的文档编号，所以文档编号差值总是大于 0 的整数。如图 2 所示的例子中，原始的 3 个文档编号分别是 187、196 和 199，通过编号差值计算，在实际存储的时候就转化成了：187、9、3。

之所以要对文档编号进行差值计算，主要原因是为了更好地对数据进行压缩，原始文档编号一般都是大数值，通过差值计算，就有效地将大数值转换为了小数值，而这有助于增加数据的压缩率。

现将相关术语描述如下：

倒排索引（英语：Inverted index），也常被称为反向索引、置入档案或反向档案，是一种索引方法，被用来存储在全文搜索下某个单词在一个文档或者一组文档中的存储位置的映射。它是文档检索系统中最常用的数据结构。通过倒排索引，可以根据单词快速获取包含这个单词的文档列表。倒排索引主要由两个部分组成：“单词词典”和“倒排文件”。

倒排索引有两种不同的反向索引形式：

一条记录的水平反向索引（或者反向档案索引）包含每个引用单词的文档的列表。

一个单词的水平反向索引（或者完全反向索引）又包含每个单词在一个文档中的位置。

后者的形式提供了更多的兼容性（比如短语搜索），但是需要更多的时间和空间来创建。

现代搜索引起的索引都是基于倒排索引。相比“签名文件”、“后缀树”等索引结构，“倒排索引”是实现单词到文档映射关系的最佳实现方式和最有效的索引结构。

构建倒排索引主要有“合并法”和“简单法”：

#### 简单法

索引的构建相当于从正排表到倒排表的建立过程。当我们分析完网页时，得到的是以网页为主码的索引表。当索引建立完成后，应得到倒排表，具体流程如图所示：

流程描述如下：

- 1) 将文档分析称单词 term 标记，
- 2) 使用 hash 去重单词 term
- 3) 对单词生成倒排列表

倒排列表就是文档编号 DocID，没有包含其他的信息（如词频，单词位置等），这就是简单的索引。

这个简单索引功能可以用于小数据，例如索引几千个文档。然而它有两点限制：

- 1) 需要有足够的内存来存储倒排表，对于搜索引擎来说，都是 G 级别数据，特别是当规模不断扩大时，我们根本不可能提供这么多的内存。
- 2) 算法是顺序执行，不便于并行处理。

#### 合并法

归并法, 即每次将内存中数据写入磁盘时, 包括词典在内的所有中间结果信息都被写入磁盘, 这样内存所有内容都可以被清空, 后续建立索引可以使用全部的定额内存。

#### 归并索引

合并流程：

- 1) 页面分析，生成临时倒排数据索引 A，B，当临时倒排数据索引 A，B 占满内存后，将内存索引 A，B 写入临时文件生成临时倒排文件，
- 2) 对生成的多个临时倒排文件，执行多路归并，输出得到最终的倒排文件 (inverted file)。

索引创建过程中的页面分析，特别是中文分词为主要时间开销。算法的第二步相对很快。这样创建算法的优化集中在中文分词效率上。

倒排索引建立起之后，有时候会根据需要进行更新，于是就出现了更新的决策问题，有如下几个主要方法：

更新策略有四种：完全重建、再合并策略、原地更新策略以及混合策略。

完全重建策略：当新增文档到达一定数量，将新增文档和原先的老文档整合，然后利用静态索引创建方法对所有文档重建索引，新索引建立完成后老索引会被遗弃。此法代价高，但是主流商业搜索引擎一般是采用此方式来维护索引的更新。

再合并策略：当新增文档进入系统，解析文档，之后更新内存中维护的临时索引，文档中出现的每个单词，在其倒排表列表末尾追加倒排表列表项；一旦临时索引将指定内存消耗光，即进行一次索引合并，这里需要倒排文件里的倒排列表存放顺序已经按照索引单词字典顺序由低到高排序，这样直接顺序扫描合并即可。其缺点是：因为要生成新的倒排索引文件，所以对老索引中的很多单词，尽管其在倒排列表并未发生任何变化，也需要将其从老索引中取出来并写入新索引中，这样对磁盘消耗是没必要的。

原地更新策略：试图改进再合并策略，在原地合并倒排表，这需要提前分配一定的空间给未来插入，如果提前分配的空间不够了需要迁移。实际显示，其索引更新的效率比再合并策略要低。

混合策略：出发点是能够结合不同索引更新策略的长处，将不同索引更新策略混合，以形成更高效的方法。



## 3.3.1.3 模块内流程图

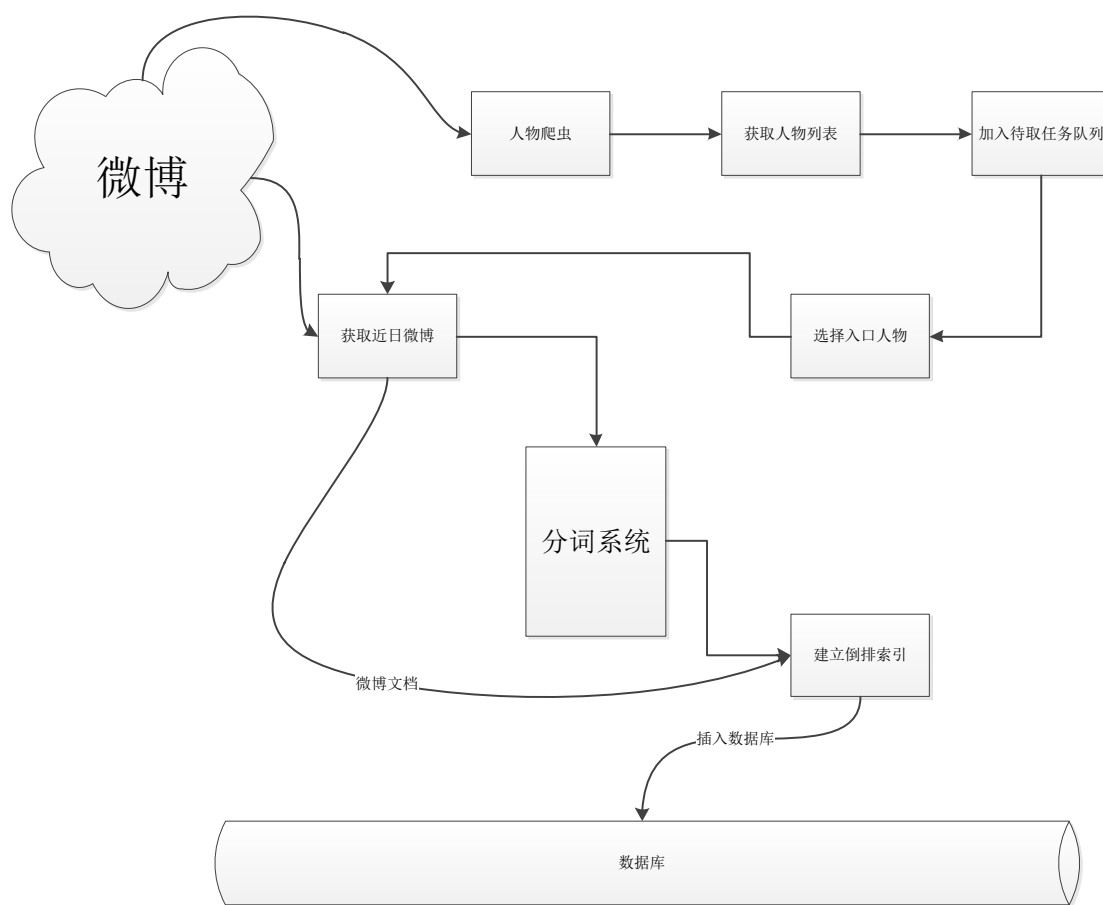


图 3-4 数据获取模块内部流程图

- ① 人物爬虫：以用户为单位，获取该用户的粉丝以及该用户的近期微博。
- ② 人物列表：分为待处理与已处理，待处理用户会被放入待取任务队列。
- ③ 加入待取任务队列：这里存放下一个需要获取的用户微博的对象。
- ④ 获取近日微博：将待取任务队列中的一个用户提取出，并且获得他近日的微博，然后将其加入已处理用户集。
- ⑤ 分词系统：将即时获得的微博经过分词系统，将新词加入到词典，旧词则更新 IDF。
- ⑥ 建立倒排索引：根据词典以及微博建立倒排索引，最终加入到数据库中，供下一个模块使用。



DBHandler：这个封装了对数据库的一系列操作

类名	成员函数	详解
DBHandler	convertCharset	更改字符串的字符集
	getCon	得到一个数据库的连接
	connectDB	连接数据库
	closeDB	关闭数据库连接
	addDoc	增加一个文档到数据库
	addLexicon	在单词库中增加一个单词
	isLexExis	检查这个单词是否已存在单词库

表 3-2 DBHandler 详解

Doc：这个是微博文档对象类

类名	成员函数	详解
Doc	setDoc	设置文档内容
	getDocCon	得到文档内容
	getDocID	得到文档 ID

表 3-3 Doc 详解

Global：这个类是存放全局共享数据的

类名	成员变量	详解
Global	Access_token	OAuth2.0 协议中的 token 号
	uIDString	其实用户 id 号
	noMeanStrings	无意义文档过滤匹配字
	usedIDlist	已经处理过的用户 id
	dBer	全局的数据库访问单体
	Queue	用户队列
	Split	分词系统实例
	docList	文档库

表 3-4 Global 详解

Liexicon: 这个类是词典类, 记录单词以及他出现的文档 ID 以及 IDF

类名	成员函数	详解
Liexocon	setID	设置单词 id 号
	getID	得到单词 id 号
	setLexicon	设置单词内容
	getLexicon	得到单词内容
	computeIDF	计算该单词 IDF 值
	updatedocF	更新该单词所出现过的文档数

表 3-5 Liexicon 详解

People: 这个是人物类, 主要包含了其用户 ID, 主要作用是获取微博以及粉丝的 ID

类名	成员函数	详解
People	getID	得到用户 ID 号
	getFriends	得到该用户所关注的所有朋友 ID 号
	getDoc	得到该用户近期发出的微博

表 3-6 People 详解

PeopleQueue: 这个是人物队列类

类名	成员函数	详解
PeopleQueue	needBreak	判断是否需要终端
	getBreak	得到断点一边继续执行
	SetBreak	设置断点
	getMaxQueueLen	得到队列最大长度
	putIn	塞入队列
	getOut	取出

表 3-7 PeopleQueue 详解

splitSystem: 这个是分词系统类，封装了 iK-analyzer 的相关操作，方便开发

类名	成员函数	详解
splitSystem	setDoc	设置需要被分词的文档
	beginSplit	执行分词系统过程

表 3-8 splitSystem 详解

### 3.3.2 数据分析模块

根据数据获取模块建立的倒排索引，依据聚类算法以及评分获得最热点事件，形成最终结果。该内部模块如下图

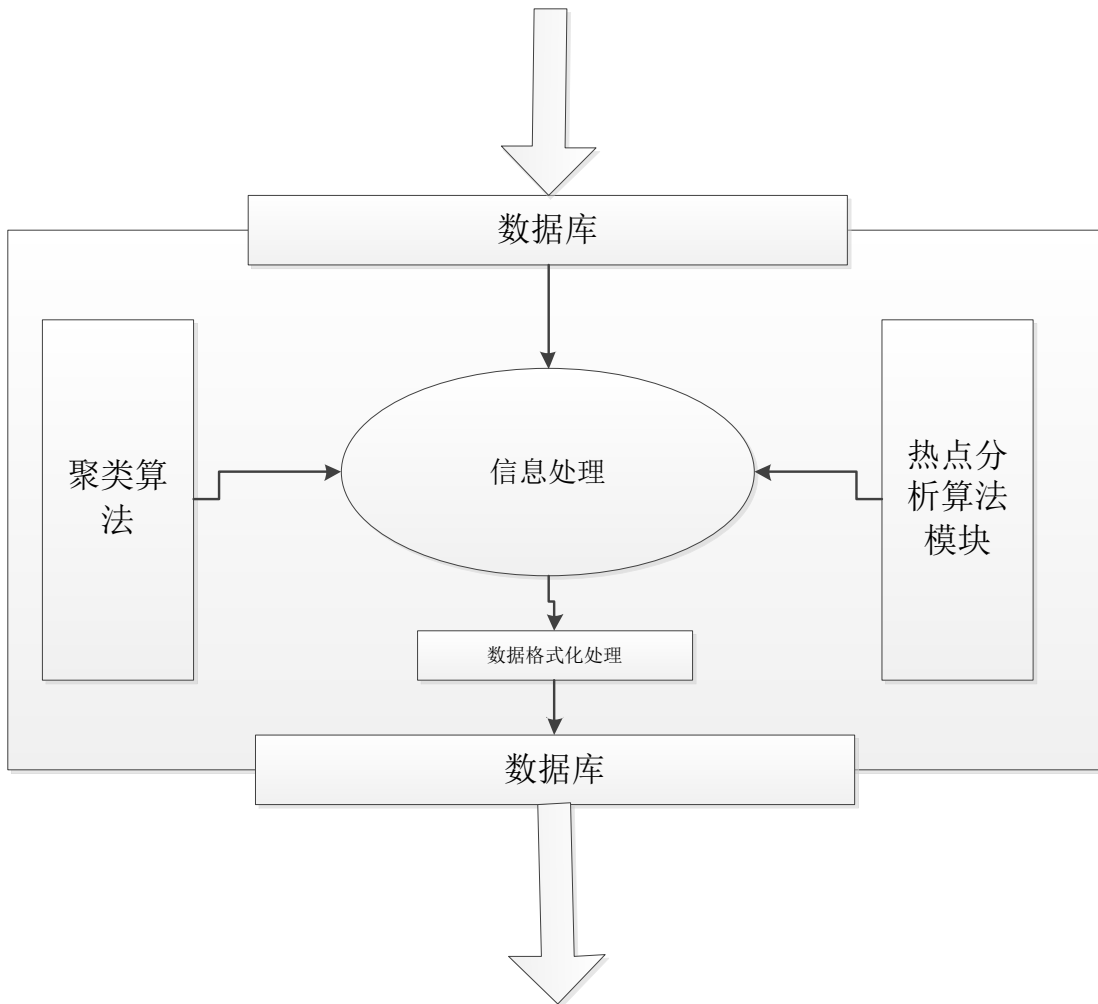


图 3-9 数据分析模块

### 3.3.2.1 模块类图设计

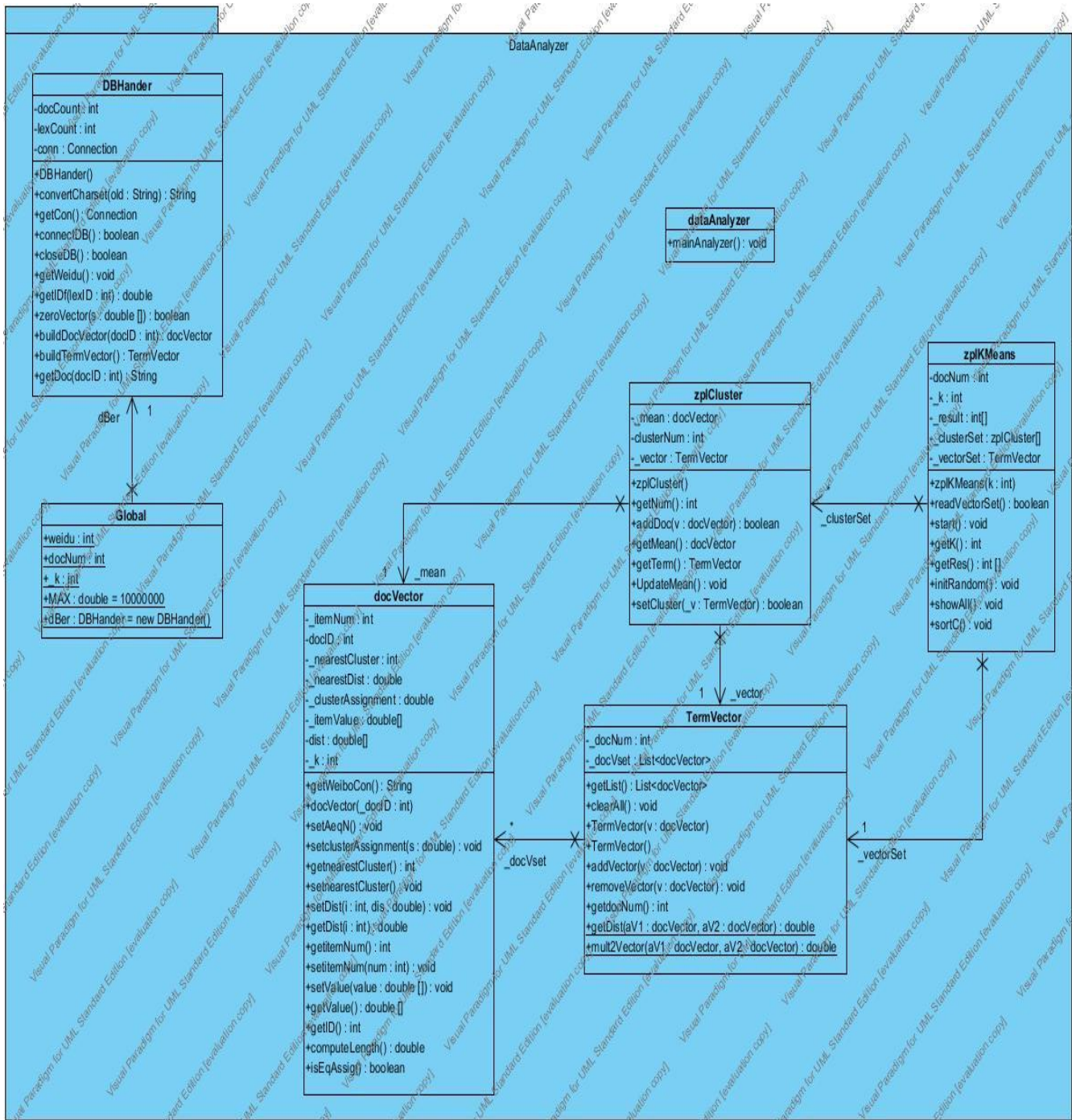


图 3-10 类图设计

**dataAnalyzer:** 该类是本模块的主运行类。

类名	成员函数	详解
dataAnalyzer	mainAnalyzer	该模块程序主入口

表 3-9 dataAnalyzer 详解

**DBHandler:** 该类是专门是对读取数据库的一些操作进行的封装

类名	成员函数	详解
DBHandler	convertCharset	转换字符集
	getCon	得到一个数据库连接
	connectDB	连接数据库
	closeDB	关闭数据库连接
	getWeidu	得到需要向量化的维度
	getIDF	得到某个单词的 IDF 值
	zeroVector	判断一个向量是否是零向量
	buildDocVector	将 Doc 库中所有的文档全部向量化
	buildTermVector	将全部文档向量化并且存入一个数据结构
	getDoc	得到一个文档的内容

表 3-9 DBHandler 详解

**docVector:** 该类是将一条微博进行向量化之后的向量类

类名	成员函数	详解
docVector	getWeiboCon	得到微博内容
	setAeqN	将猜测的聚类值认为是最近的距离
	setclusterAssignment	设置聚类猜测值
	getnearestCluster	得到最近的聚类编号
	setnearestCluster	设置最近的聚类编号
	setDist	设置与中心点的距离
	getDist	得到与中心点的距离
	getitemNum	得到维度
	setitemNum	设置维度
	setValue	设置向量值
	getValue	得到向量值
	getID	得到文档 ID 号
	computeLength	计算该向量长度
	isEqAssig	最近距离是否与猜测值相等

表 3-10 docVector 详解

**Global:** 该类是全局共享数据

类名	成员变量	详解
Global	Weidu	需要向量化的维度, 即单词库中的数目
	docNum	文档库中所有的文档数目
	dBer	数据库处理器的实例
	_k	被划分的聚类中心数目

表 3-11 Global 详解



**TermVector:** 存放向量集类

类名	成员函数	详解
TermVector	getList	得到向量化后的文档的列表
	clearAll	清楚实例中所有的数据
	addVector	增加一个向量化文档到该向量集合中
	removeVector	移除一个向量化的文档
	getdocNum	得到已有的文档数目
	getDist	得到 2 个向量的距离, 即余弦角度
	mult2Vector	2 个向量相乘

表 3-12 TermVector 详解

**zplCluster:** 划分出来的聚类

类名	成员函数	详解
zplCluster	getNum	该聚类的文档数目
	addDoc	将一个文档加入到该聚类中
	getMean	得到该聚类的中心值
	getTerm	得到向量集
	UpdateMean	新加入文档之后需要重新计算中心值
	setCluster	设置一个向量集来存储需要的向量

表 3-13 zplCluster 详解

zplKMean: K-means 算法的实现类

类名	成员函数	详解
zplKMean	readVector	从数据库之中读出文档并且向量化
	Start	进行 k-means 聚类算法
	getK	得到聚类中心数目
	getRes	得到最后的结果
	initRandom	随机计算确定的中心点
	showAll	将计算结果写入 excel 文档中
	sortC	对结果进行排序

表 3-14 zplKMean 详解

### 3.4 数据库设计方案

本数据库主要是存放词典、微博文档以及倒排索引。

根据需要设计出下面几张表

- 1) Doc 表，存放微博文档

Fields	Indexes	Foreign Keys	Triggers	Options	Comment	SQL Preview
Name	Type	Length	Decimals	Allow Null		
docID	int	11	0	<input type="checkbox"/>		1
docText	text	0	0	<input checked="" type="checkbox"/>		

图 3-12 doc 表

- 2) Lexicon 表，存放单词词典

Name	Type	Length	Decimals	Allow Null	
docF	int	11	0	<input checked="" type="checkbox"/>	
IDF	text	0	0	<input checked="" type="checkbox"/>	
lexiconID	int	11	0	<input type="checkbox"/>	1
lexicon	text	0	0	<input checked="" type="checkbox"/>	

图 3-13 lexicon 表

- 3) Postingitem 表，存放倒排索引


Name	Type	Length	Decimals	Allow Null	
tf	int	11	0	<input checked="" type="checkbox"/>	
id	int	11	0	<input type="checkbox"/>	 1
lexiconID	int	11	0	<input checked="" type="checkbox"/>	
docID	int	11	0	<input checked="" type="checkbox"/>	

图 3-14 postingitem 表

4) Usedlist 表，存放已经使用过的用户


Fields	Indexes	Foreign Keys	Triggers	Options	Comment	SQL Preview	
Name	Type	Length	Decimals	Allow Null			
id	int	11	0	<input type="checkbox"/>		 1	
userid	text	0	0	<input checked="" type="checkbox"/>			

图 3-15 usedlist 表

## 第四章 系统实现

### 4.1 数据获取模块实现

#### 4.1.1 核心代码

dataMining 类: (更多代码见附件)

```
package DataMining;
```

```
//这个是数据获取模块的主程序
```

```
public class dataMining {
```

```
    public boolean mainDataMining(){
```

```
        initDB();
```

```
        //初始化微博与人物队列
```

```
        if(loginWeibo())
```

```
        {
```

```
            process();
```

```
        }
```

```
        else {
```

```
            return false;
```

```
        }
```

```
        return false;
```

```
    }
```

```
//登陆微博并且初始化人物队列
```

```
public boolean loginWeibo(){
```

```
    People tmp = null;
```

```
    int size = Global.usedIDList.size();
```

```
    if(size == 0){
```

```
        tmp = new People(Global.uIDString);
```

```
    }
```

```
    else {
```

```
        tmp = new People(Global.usedIDList.get(size -1));
```

```
    }
```

```

        tmp.getFriends(Global.queue);
        return true;
    }

    public boolean process(){

        People tmp;
        //处理数据，建立倒排索引
        while(Global.queue.getCount() != 0){
            tmp = Global.queue.getOut();
            if(tmp == null)
                return false;
            //数据处理
            if(tmp.getDoc() == false)
                return false;
            tmp.getFriends(Global.queue);
            Global.usedIDList.add(tmp.getID());
            Global.dBer.insertUsedList(tmp.getID());
        }
        return true;
    }

    public boolean initDB(){
        Global.dBer.connectDB();
        Global.dBer.initUsedList();
        Global.dBer.initValue();
        return false;
    }

}

```

#### 4.1.2 相关时序图

时序图如下：

1) 初始化

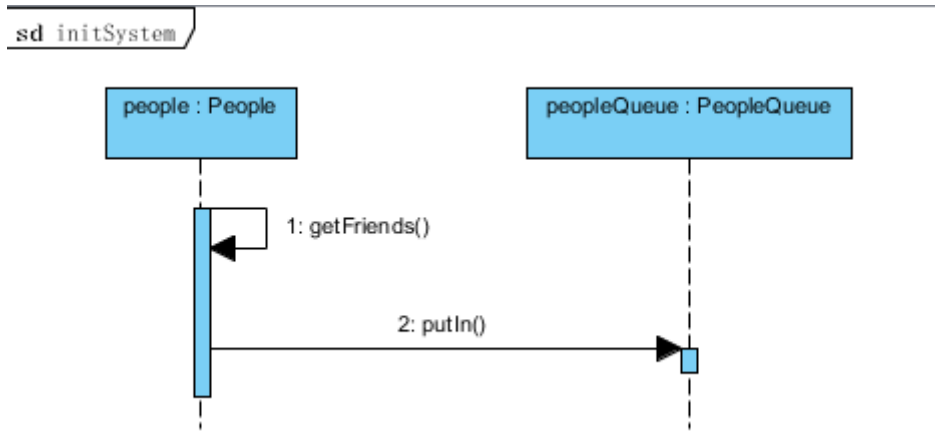


图 4-1 模块初始化

从首个用户开始，抓取用户粉丝，塞入待获取任务队列。

## 2) 数据获取

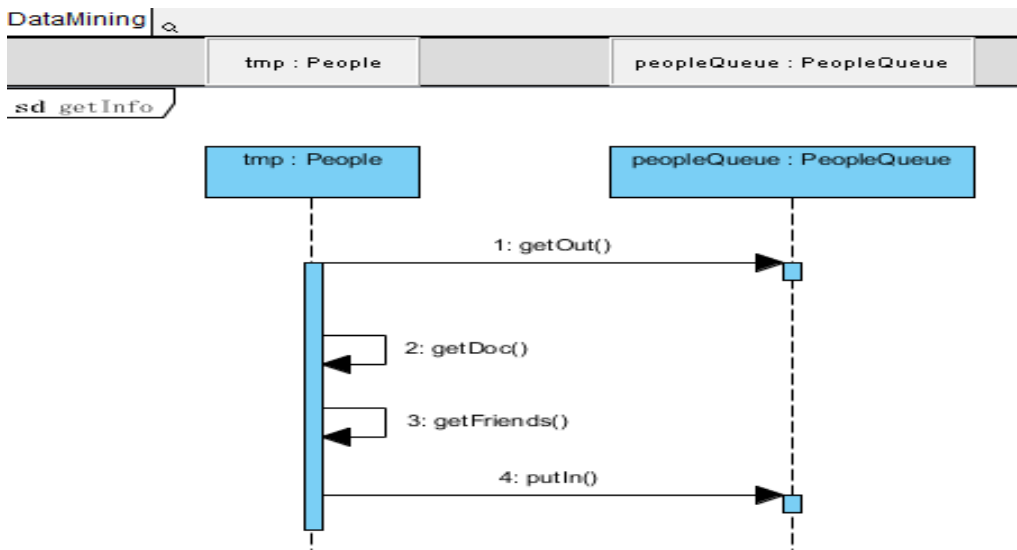


图 4-2 数据获取

先从队列之中得到一个待抓取用户，然后在得到该用户的近期微博，获得微博之后，进入其他类进行处理，处理完后，获取该用户的粉丝，然后又加入到待抓取队列之中，等待下次的处理。

## 3) 建立倒排索引

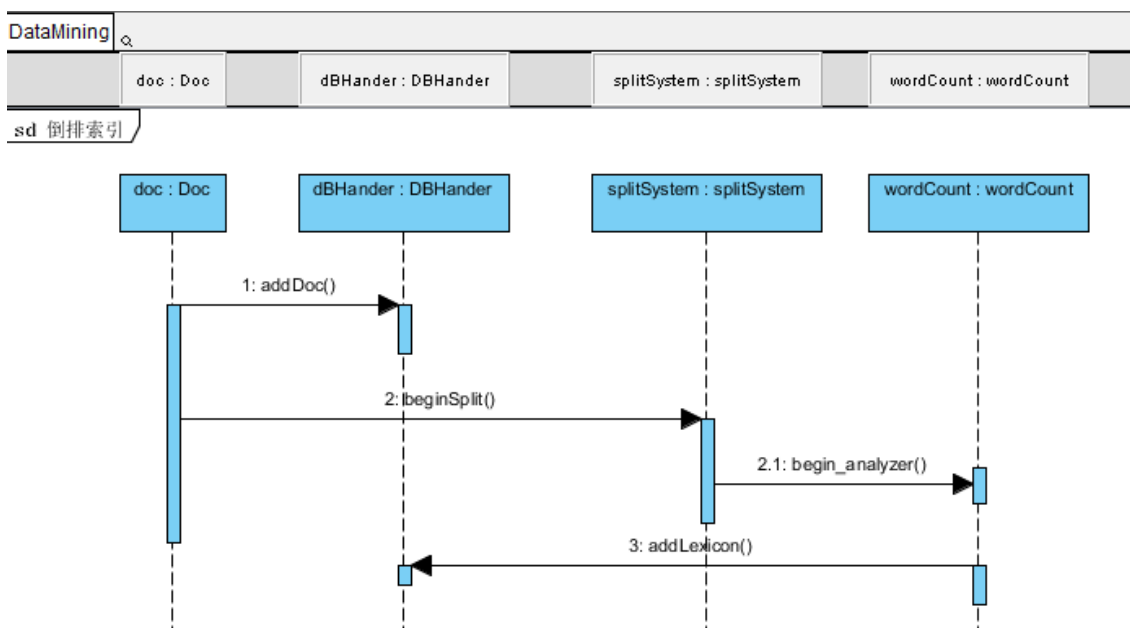


图 4-3 建立倒排索引

获取到一个用户的微博之后，首先将这个微博加入到数据库之中，然后对这条微博进行分词，将分完的单词加入到词典之中。这样子就完成了倒排索引的建立。

## 4.2 数据处理模块实现

### 4.2.1 核心代码

Dataanalyzer 类: (更多代码见附件)

```

package DataAnalyzer;

public class dataAnalyzer {
    public void mainAnalyzer(){
        Global.dBer.connectDB();
        zplKMeans kMeans;
        kMeans = new zplKMeans(20);
        kMeans.readVectorSet();
        kMeans.initRandom();
        kMeans.start();
        kMeans.showAll();
        Global.dBer.closeDB();
    }
}
  
```

## 4.2.2 相关时序图

### ① 数据分析

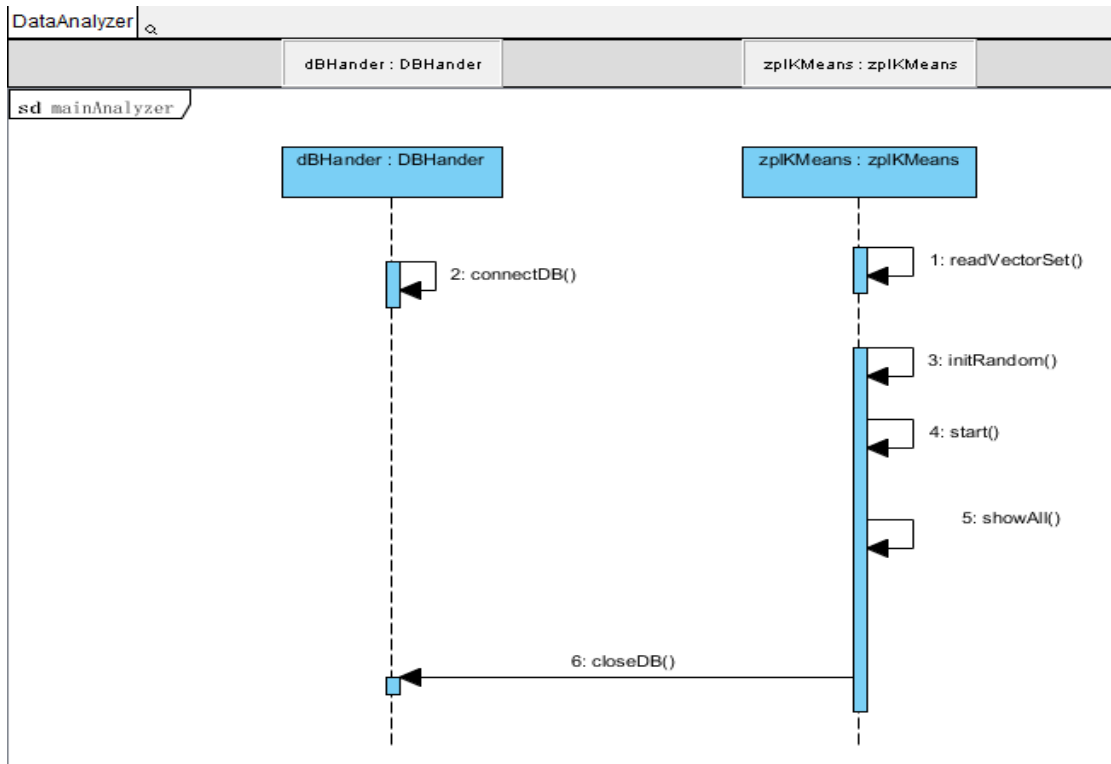


图 4-4 数据分析时序图

步骤如下：

- 1) 连接数据库
- 2) 将数据库中的全部文档向量化，将一个向量化方法如下：
  - a) 读取整个单词库，获得所有单词的数目  $n$ ，创建一个  $n$  维的向量  $v$  ( $a_1, a_2, \dots, a_n$ )。
  - b) 遍历该文档，若出现第  $i$  号单词，则将  $v$  的第  $i$  个位置改成该单词的  $TF \cdot IDF$  的值。
  - c) 遍历完后，则  $v$  为该文本的向量化
- 3) 从全部向量化之后，从向量化集中随机提取出几个向量作为基本中心点
- 4) 进行  $k$ -means 聚类算法
- 5) 显示结果



## 6) 关闭数据库连接

## ② 建立向量集

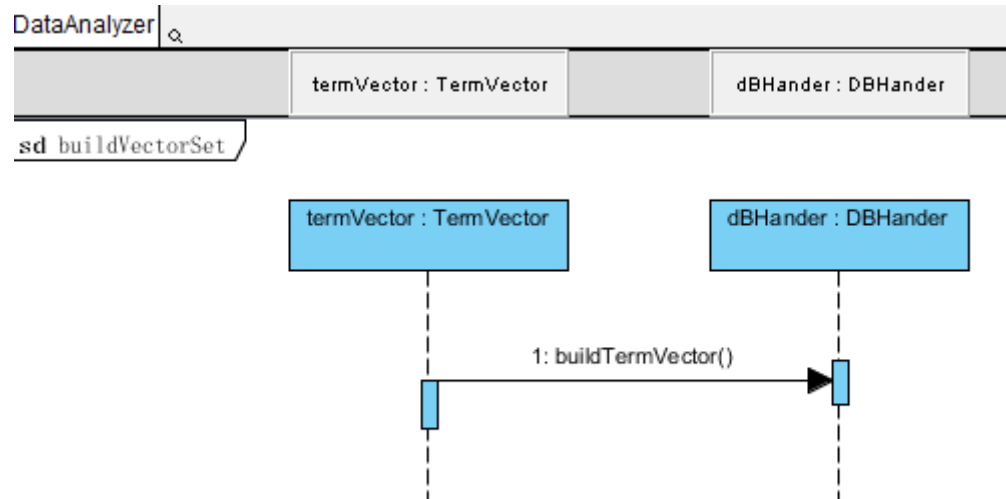


图 4-5 建立向量集时序图

## ③ K-means 算法

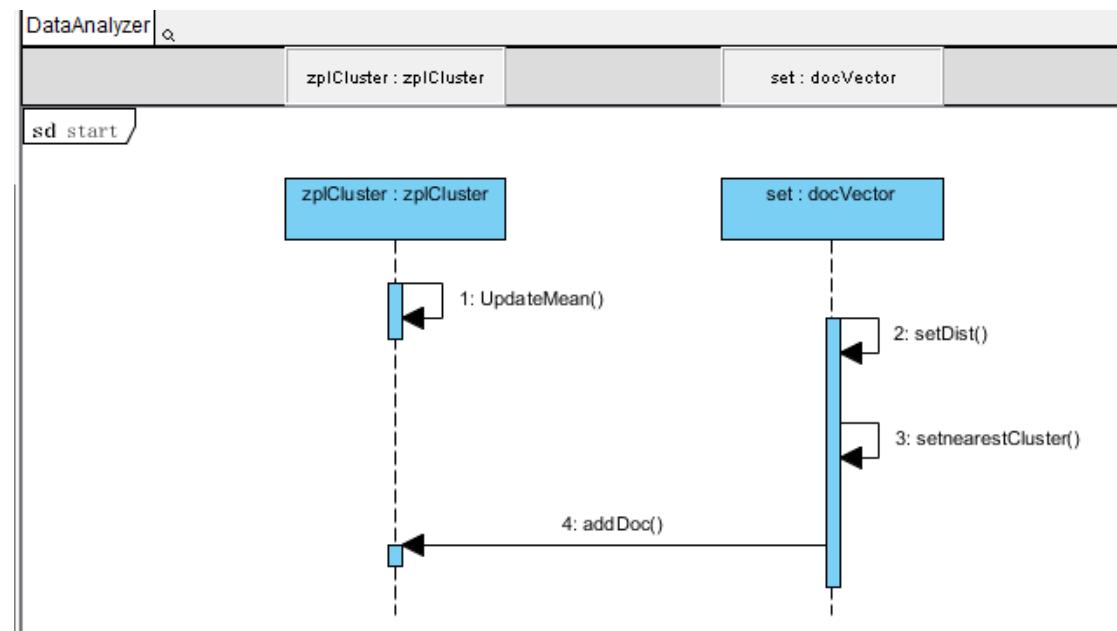


图 4-6 k-means 时序图

#### ④ 状态图

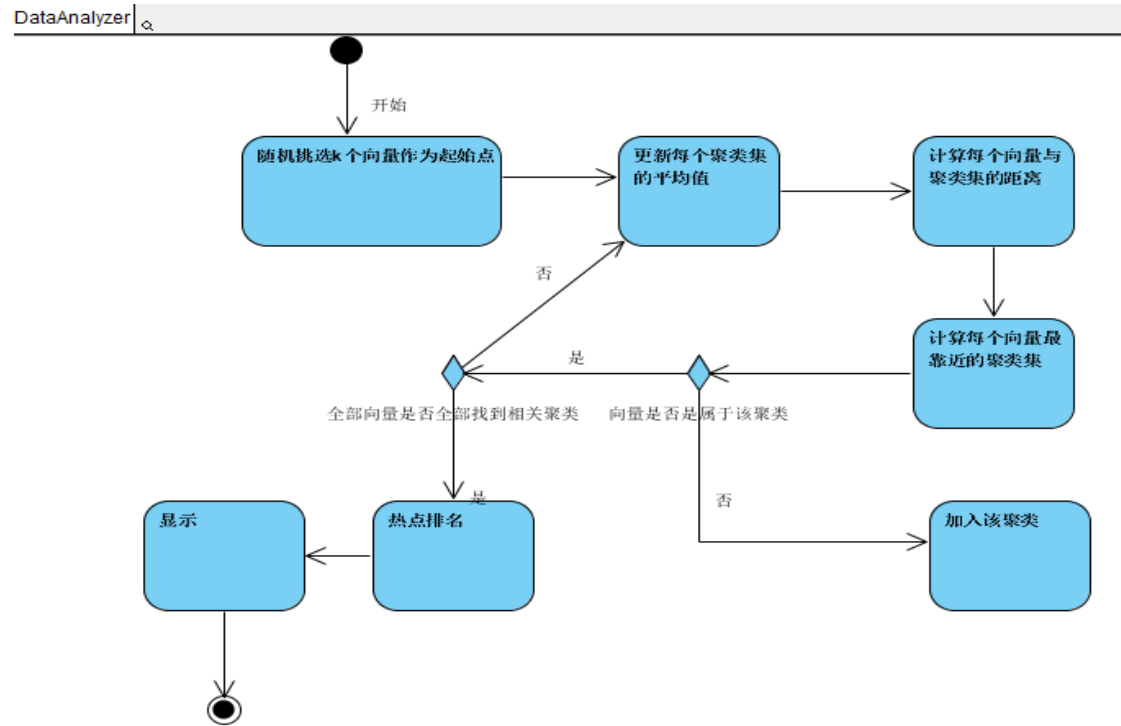


图 4-7 数据分析状态图

## 第五章 系统运行结果

### 5.1 源码结构展示

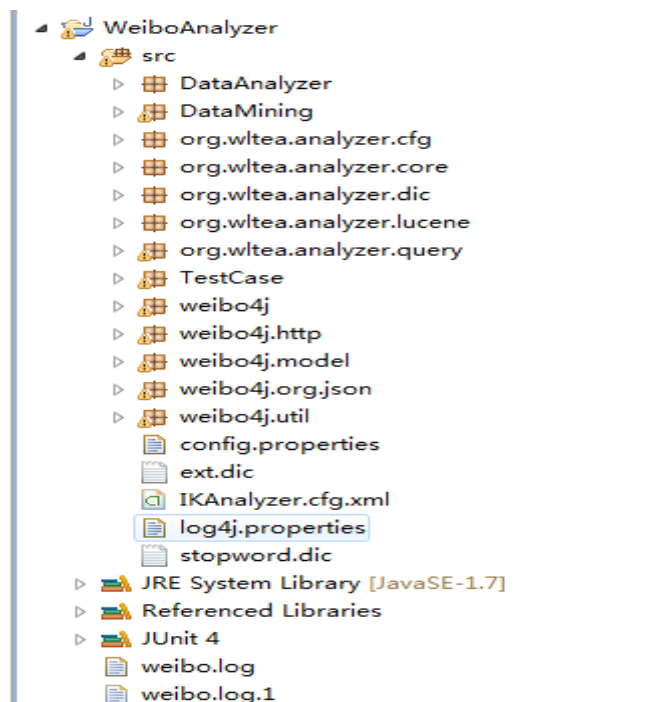


图 5-1 系统源码结构图

- 1) DataMining 包  
数据挖掘模块
- 2) DataAnalyzer  
数据分析模块
- 3) Org.wltea 包  
Ik-analyzer 中文分词器源码
- 4) Weibo4j 包  
新浪微博 api 源码
- 5) Testcase 包  
开发测试用例

### 5.2 数据抓取模块运行展示

- 1) 运行测试用例 testMining

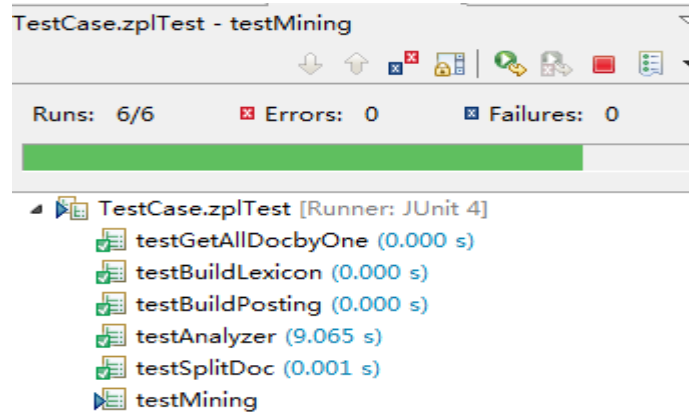


图 5-2 测试用例 testMining

## 2) 数据库中数据

Doc 表

docID	docText
1	希望有人成立一个不能吃葱花的教派，就像不能吃猪肉的伊斯兰
2	明明不想笑，笑亦无味...
3	你只看到我的凶残却没看到我的可爱;你有你的偏见,我有我的个性
4	04月30日“哇啦一下”：很不错 离家很近 环境也不错 就是放壳
5	【生活大百科】常吃一点蒜，消毒又保健；多食一点醋，不用上
6	好内都没试训甘爽啦。[哈哈][哈哈][哈哈]
7	太极品了这画的，拔插头谁能忍！！「转」
8	服了自己了，考试之前又这样！作为理科女好悲伤啊！就知道会
9	感觉好无助'事情真的是难以预测'我感觉我真的好无助'胃好疼'我
10	爱情中最伤感的时刻是后期的冷淡。一个曾经爱过你的人，忽然
11	【激励自己的9句话】1，人之所以能，是相信能；2，拿望远镜
12	我还记得小时候，老爸常带我去自然博物馆，我总是磕磕绊绊地
13	【反恐精英OL】CSol福利4制服控的诱惑——爱丽丝(女仆装)
14	【沁园春·上班】上班路上，千里车流，万里人潮。望大街内外
15	中午去食堂吃饭，选了茄子和花菜，阿姨竟然特大方地从花菜下
16	【AM年鉴Volume-16之设计师Geoffrey Bradfield】来自美国
17	我朋友是个传奇迷，吃饭的是后他问我们→_→也玩传奇私服？狂
18	吓死我了！你个衰人！干嘛不说明白那是办公室电话！[泪][泪][泪]
19	泰来 今天(5月3日)天气：多云转晴，10℃~19℃，北风4-5级，
20	患得患失，每天都害怕
21	天气好凄凉，心.....
22	哈哈~~the best day,every day

图 5-3 doc 表

## 3) Lexicon 表

docF	IDF	lexiconID	lexicon
21	1.5995356	1	希望
9	5.5297473	2	有人
3	6.7223924	3	成立
57	0.4484696	4	一个
53	0.6789701	5	不
3	6.7223924	6	能吃
3	6.7223924	7	葱花
3	6.7223924	8	教派
9	-0.3479969	9	就像
3	6.7223924	10	猪肉
3	6.7223924	11	伊斯兰教
19	0.8959286	12	一样
5	6.1380738	13	今后
3	6.7223924	14	问到
9	0.1374299	15	为什么
3	6.7223924	16	挑出来
19	1.6443899	17	时候
7	5.9453105	18	就能
3	6.7223924	19	高贵
3	6.7223924	20	冷艳
3	6.7223924	21	回答
33	1.2884958	22	因为

图 5-4 lexicon 表

## 4) Usedlist 表

id	useid
2263	30445709
2264	26846915
2265	15839426
2266	29194172
2267	28037927
2268	28047145
2269	29202098
2270	27897558
2271	28078485
2272	29197893
2273	27896647
2274	15839429
2275	29200269
2276	28086975
2277	28088162
2278	29200735
2279	26655033
2280	23913962
2281	21200450

图 5-5 usedlist 表

## 5) Postingitem 表

tf	id	lexiconID	docID
1	40340	28	2
1	40341	29	2
1	40342	30	3
2	40343	31	3
1	40344	32	3
1	40345	33	3
1	40346	34	3
1	40347	35	3
1	40348	36	3
1	40349	37	3
1	40350	38	3
1	40351	39	3
1	40352	40	3
2	40353	41	3
1	40354	42	3
1	40355	43	3
1	40356	44	3
1	40357	45	3
2	40358	46	3
1	40359	47	3
1	40360	48	3
1	40361	49	3

图 5-6 postingitem 表

## 5.3 数据分析模块

### 1) 运行测试用例 testAnalyzer

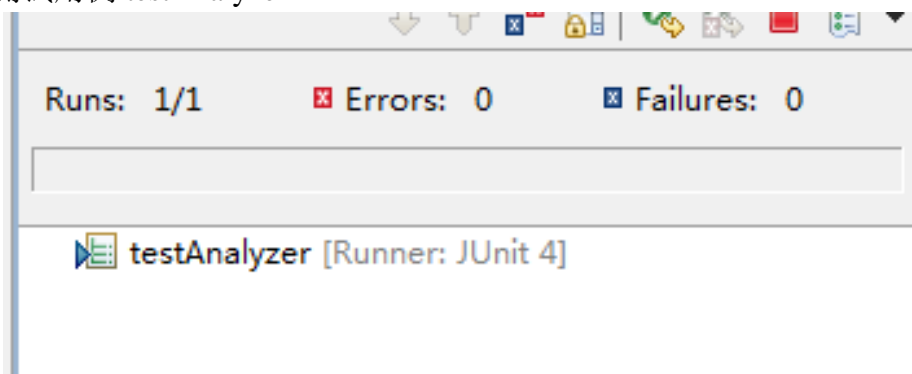


图 5-7 运行测试用例 testAnalyzer

### 2) 调试运行结果

```

doc:反洗脑!计生委告诉我们:中国人口增长过快,中国资源不够分,中国人喜欢生孩子,少生能促共富。如果你认为,这些话有一定道理,就是被洗脑了!因
doc:【专栏:中国的适度人口是五到八亿?】- 在有关生育政策调整的争论中,常见的一个反对理由是中国应该把人口降到五至八亿,甚至三亿,但这不过是?
doc:『馬英九答大陸網友計生提問』人口從來不是負擔,有新生人口才能促進國家進步、繁榮,不論貧窮或富有家庭,生育都是基本人權。-摘自小馬哥辦公室
doc:【计划生育的惊人代价】大家都知道,中国人为限制生育付出了惨重的代价,人口结构老化,人权被侵犯,男女比例失调,失独家庭大增,国际声誉受损等
doc: #观点#【水资源够吗——全面放开生育与自然资源】中国人均水资源占有量在全球处于中等水平,水资源并不比世界大部分国家更缺乏。为什么中国的局部
doc:【专栏:应放开二胎还是全面放开生育?】- 放开二胎是远远不够的,中国需要的是立即全面无条件放开生育,并在生育率再次下滑到更替水平之际,像/
doc:【计生委官员:通过增加生育冲淡老龄化是饮鸩止渴】国家计生委党组成员杨玉学称计划生育不是老龄化主要因素,生活富裕、医疗水平提高都使老龄人口:
doc:【拆分计生委之一】将卫生部的职责、人口计生委的计划生育管理和服务职责整合,组建国家卫生和计划生育委员会,同时将人口计生委的研究拟订人口发
972549 DEBUG [2013-05-23 20:35:22] Request:
972549 DEBUG [2013-05-23 20:35:22] GET:https://api.weibo.com/2/friendships/friends.json
972549 DEBUG [2013-05-23 20:35:22] Authorization: OAuth2 2.00GLy1AC3oj17E9d85327844CQtJFB
972549 DEBUG [2013-05-23 20:35:22] API-RemoteIP: 192.168.1.104
983505 DEBUG [2013-05-23 20:35:33] Response:
983505 DEBUG [2013-05-23 20:35:33] https StatusCode:200
983505 DEBUG [2013-05-23 20:35:33] Server:nginx/1.2.0
983505 DEBUG [2013-05-23 20:35:33] Date:Thu, 23 May 2013 12:35:24 GMT
983505 DEBUG [2013-05-23 20:35:33] Content-Type:application/json;charset=UTF-8
983505 DEBUG [2013-05-23 20:35:33] Content-Length:46482
983505 DEBUG [2013-05-23 20:35:33] Connection:keep-alive
983505 DEBUG [2013-05-23 20:35:33] Api-Server-IP:10.75.24.204
983505 DEBUG [2013-05-23 20:35:33] Vary:Accept-Encoding
983505 DEBUG [2013-05-23 20:35:33] X-Varnish:1698901994
983505 DEBUG [2013-05-23 20:35:33] Age:0

```

图 5-8 调试运行结果

### 3) 将最终结果输入到 excel 中

近20年来我致力于扶贫工作，投入家庭储蓄百多万元。帮助了成千上万个穷人。可是最近不断打电话骚扰我的都是语言粗鲁的低收入人士。我得罪他们的是“廉租房不该有私人厕所”，“不要为钓鱼岛开战死人”等主张。他们不明白自己的利益到底在哪儿，喜欢听灌米汤式的宣传。这真是我们国家的危险所在。谢谢网友阅读、传播我的博客文章，欢迎大家在这里评论，提出自己的见解。——看到茅于軾的博文《战争与和平——人民需要觉醒》有感而发的评论。

<http://t.cn/zHAV6AU>

【微软智能机和平板占游戏总收入15%】微软发布数据显示，在游戏收入方面，主机仍是最主要的动力，但移动和平板部分也在快速的增长，目前已经占据每年游戏收入的15%以上。意味着智能机和平板游戏每年收入100亿美元左右。相当于掌机和社交游戏收入的总和还多。<http://t.cn/zH4zUOt>

【社交媒体用户昵称研究】昵称为中英文结合——力求精致优质生活的小资女；纯英文昵称——IT技术宅男遍布；昵称中带横杠——微博活跃用户偏爱的昵称；长昵称——经常半夜出没晒美食报复社会；短昵称——微博认证用户级别普遍较高。来自：@凯络中国

拥毛分子抬着大标语“打倒汉奸卖国贼茅于軾”在马路上游行。警察不管。如果说某个中央领导是汉奸卖国贼，警察一定会管。这就是中国的人权状况。普通人并不能得到人权的有效保护。朋友们劝我起诉他们。但是我不想用对立的方法。我只是说：你们这样做有损于你们事业的正义性。符合正义的事要用正义的手。

【今日微博：谣言与辟谣】谣言一：1801裸女自杀。@记者贤云 辟谣：当事人于昨晚自己前往酒吧拿回当时遗落手机，目前情绪较稳定。谣言二：一个电话解救8个被拐儿童。@长沙颜家文 辟谣：经过警方核实，这不是一起贩卖小孩案件，系8名藏族妇女带自家小孩出行。点评：在微博上消费他人，自己也可能被消费。

#读报#【南都：@茅于軾：很多糊涂的人还想走过去的路】我觉得他们（骂我的人）很可悲，太糊涂了，不知道自己的利益在哪里。为他们办事的人，他们却以为是敌人。谁会怀念文革那个时代呢？农民吃不饱饿死，工人几十年不加工资，知识分子、老干部挨批挨斗，谁得到了好处呢？<http://t.cn/zTH97gA>

【皮尤调查：青少年不喜欢成人加入Facebook】77%上网少年使用Facebook，尽管七成的青少年在Facebook上和父母是好友，但他们不喜欢越来越多的成人加入Facebook。<http://t.cn/zH4iHYP>

【北京现“鬼来电”骚扰电话 接通有婴儿哭尖叫声】近日，一个被称为“鬼来电”的陌生电话引起北京市民热议。市民称如果接通该号码，电话里会传出“婴儿哭泣”、“女人尖叫”等怪声，但回拨过去却显示为空号。警方建议，如果市民接到此电话，可在接电话的地点报警。<http://t.cn/zHLEIMV>

毛泽东时代没有贪官污吏，没有贫富差距（大家都是穷人），没有赌博娼妓，没有吸毒贩毒。但是农民挨饿到饿死，工人几十年不加工资穷得养不了家，富人被抄家挨打甚至被打死，干部被斗。只有他一个人高高在上，不顾别人死活。可叹无知的群盲，不知道自己的利益在哪里。赶紧开放对文革和毛泽东的客观评价。

对我骚扰的事我已报警。北京和我要去的外地派出所都加强了保安工作。感谢关心我的诸君。我不希望形成对立，希望大家都讲理，追求一个和谐友好的社会，大家都要反对暴力。暴力是一副毒药，大家都吃苦头。现在个别地方暴力拆迁，暴力打胎，暴力抓人，不跟你讲理，正在腐蚀我们的社会。文明社会才是方向。

#凤凰精英苑#【独家：茅于軾谈“拥毛派”及对毛泽东看法的转变】茅于軾认为“拥毛派”对毛泽东的作为不了解，希望他们了解历史真相。面对“拥毛派”的恶言以对，他会“以直抱怨”。谈及“拥毛派”和“反毛派”的辩论共识，他认为，“大家都希望中国能好起来”。<http://t.cn/zHbLgHT> @茅于軾

大饥荒时河南饿死人294万占人口总数的6.1%，排行居全国第六位，算是重灾区。但是河南人对毛泽东的感情特别深。可见感情不是据于事实的。如果有人掌握了控制人们感情的方法就可以为所欲为。但是如果一旦人们改变了思想方法，情况就会大变。胡锦涛特别强调科学发展观，就是要基于事实符合科学的思想方法。

图 5-9 最终运行结果



## 第六章 总结与展望

### 6.1 系统主要作用

本系统在互联网的背景下，建立了对大量本地生活信息处理、分析的系统，实现了如下几点：

- 1) 通过该系统，我首先学习了网络爬虫的原理以及搜索引擎相关知识，最后通过新浪微博的 api 对网络数据模块进行了简化，并且通过学习、模仿搜索引擎建立倒排文件，对数据进行了初步处理，方便以后操作。
- 2) 中文字符串匹配：该系统之中，不仅仅采用了开源的中文分词器，而且在中文分词器之外，自己还做了过滤模块，将无用微博如：“抱歉，该微博以被....”将其滤去；还有相应的无用单词，如“你”“我”“他”“的”“之”等。该过滤模块使用正则表达式来进行实现该功能。
- 3) 热点分析策略：本系统中的热点分析策略，主要是依靠 k-means 聚类算法对大量微博进行聚类，经过我的反复试验，将 k 值设为 20 效果最佳。
- 4) 最终实现了一款微博舆情热点分析系统

### 6.2 系统难点与解决

事件挖掘是一个发展很广阔的领域，而热点事件挖掘是这个领域之中较复杂的过程。在实现该系统之中，还是遇到一些问题，下面是遇到的问题以及解决方案：

#### 1) 新浪微博 api 调用限制

由于新浪对开放平台的应用有不同的测试等级，不同的等级可调用 api 的次数限制不同，由于我是测试用户，一个小时只能调用 150 次新浪微博 api。

对于该问题，我的解决方案如下：

使用断点策略，若是在一次数据获取中达到了新浪 api 调用限制，我将会把断点保存入数据库，待限制时间取消之后，我就可以继续从断点执行。

#### 2) 中文字符集

从微博上获取的微博内容，好像大多数是 utf-8，但是我将微博插入数据库之后，数据库之中显示的是乱码，甚至有些微博插入数据库中直接会报错。

对于该问题，我的解决方案如下：

我将每条微博都无论是什么字符编码都进行了 utf-8 编码的转换，且我将系统容错性大大提升，若是某条微博真的不能插入数据库之中，系统将会把该条微博放弃。

### 3) 广告用户的微博

由于有广告微博的存在，有时候会使分析到的数据有误，比如：某条微博转发次数很高，系统不能判断这个是广告微博用户到处转发，还是真正用户都喜欢这条微博，故判断难度增加。

对于该问题，我的解决方案如下：

肉眼观看最终结果，自己剔除。

## 致谢

经过近两个多月的努力，我顺利完成了这篇论文。我衷心地感谢各位领导和老师对我的关心和帮助，特别是我的指导老师——曹晟老师，一次又一次的评阅我的论文，并提出了许多宝贵的意见和建议，使我进一步熟悉和掌握了系统开发的流程和方法，最终顺利地完成了本系统的开发。同时也感谢四年来各科任课老师的教导，您们授予的宝贵知识为我以后的学习、工作奠定了基础。

感谢在百忙之中参加论文评审和答辩的各位专家！

## 参考文献

- [1] Allan J, Papka R, Lavrenko V. On-Line New Event Detection and Tracking [J]. Proceedings of SIGIR '98:21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval [C]. New York: ACM Press, 1998, 37-45
- [2] Lam W.Meng H.Wong Ket al. Using contextual analysis for news event detection[J]. International Journal on Intelligent Systems.2001,16(4):525-546
- [3] Y Yang, J Carbonell, C Jin. Topic-conditioned novelty detection[J]. In:Hand D,etal.Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining[C]. New York: ACM Press, 2002,688-693
- [4] G Kumaran and J Allan. Text classification and named entities for new event detection [J]. In Proceedings of the SIGIR Conference on Research and Development in Information Retrieval [C]. Sheffield, South Yorkshire: ACM,2004, 297-304
- [5] Y Yang, T Pierce, J Carbonell. A study on Retrospective and On-Line Event detection [J]. Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval [C]. 1998,CMU, USA: ACM, 28-36
- [6] T Brants, F Chen, and A Farahat. A system for new event detection [J]. In Proceedings of the 26th SIGIR conference on Research and development in information retrieval [C], 2003.
- [7] Christian Wartena, Rogier Brussee. Topic Detection by Clustering Keywords[C].2007.
- [8] Nicola Stokes, Paula Hatch, Joe Carthy. Lexical Semantic Relatedness and Online New Event Detection [J]. Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval [C]. Greece: ACM, 2000, 324-325
- [9] Hatch P, Stokes N, Carthy J. Topic detection, a new application for lexical chaining [J]. British Computer Society IRSG 2000 [C]. Cambridge: British Computer Society , 2000, 94 - 103
- [10]Ulli Waltinger, Alexander Mehler, and Maik Stührenberg. 2008b. An integrated model of lexical chaining: Application, resources and its format.[J] In Angelika Storrer, Alexander Geyken, Alexander Siebert, and Kay-Michael Würzner, editors, Proceedings of KONVENS 2008 -- Ergänzungsband Textressourcen und lexikalisches Wissen[C], pages 59—70
- [11]贾自艳,何清,张俊海等. 一种基于动态进化模型的事件探测和追踪算法[J].计算机研究与发展.2004,41(7):1273 -1280
- [12]张阔, 李娟子, 吴刚等.基于词元再评估的新事件检测模型[J].软件学报.2008.4:817-828
- [13]洪宇, 张宇, 范基礼等.基于话题分治匹配的新事件检测[J].计算机学报.2008.4:688-695
- [14]Hei-Chia Wang, Tian-Hsiang Huang, Jiunn-Liang Guo. Journal Article Topic Detection Based on Semantic Features. [C]Springer-Verlag Berlin Heidelberg,2009.
- [15]JingQiu, LeJian Liao, XiuJie Dong. Topic Detection and Tracking for Chinese News Web

Pages.[C]2008.

[16]Changki Lee, Gary Geunbae Lee, Myunggil Jang. Dependency structure language model for topic detection and tracking.[J] Information Processing and Management, Vol. 43, No. 5, Sep. 2007, pp. 1249-1259.

[17]骆卫华, 于满泉, 许洪波, 王斌, 程学旗. 基于多策略优化的分治多层次聚类算法的话题发现研究[J]. 中文信息学报. 2006,20(1): 29-36

[18]Zhang Kuo, Li Juan Zi, Wu Gang. New Event Detection Based on Indexing-tree and Named Entity[A]. Sigir2007[C]. ACM: Amsterdam, 2007.

[19]洪宇, 张宇, 刘挺, 等. 话题检测与跟踪的评测及研究综述[J]. 中文信息学报. 2007,21(6):71-87

[20]Yonghui Wu, Yuxin Ding, Xiaolong Wang, Jun Xu. On-line Hot Topic Recommendation Using Tolerance Rough Set Based Topic Clustering[J]Journal of Computers, Vol. 5, No. 4, April 2010.

[21]Mingliang Zhu, Weiming Hu, Ou Wu. Topic Detection and Tracking for Threaded Discussion Communities.[C]2008.

[22]Canhui Wang, Min Zhang, Shaoping Ma. Automatic Online News Issue Construction in Web Environment.[C]BeiJing 2008 April

[23]张俊林 电子工业出版社 2012 年 1 月 《这就是搜索引擎-核心技术详解》 第三章

## 附录

### 1. 数据抓取模块代码

**dataMining.java**

**package** DataMining;

//这个是数据获取模块的主程序

**public class** dataMining {

**public boolean** mainDataMining(){

        initDB();

        //初始化微博与人物队列

**if**(loginWeibo())

        {

            process();

        }

**else** {

**return false**;

        }

**return false**;

    }

//登陆微博并且初始化人物队列

**public boolean** loginWeibo(){

    People tmp = **null**;

**int** size = Global.usedIDList.size();

**if**(size == 0){

        tmp = **new** People(Global.uIDString);

    }

**else** {

        tmp = **new** People(Global.usedIDList.get(size -1));

    }

    tmp.getFriends(Global.queue);

**return true**;

}

**public boolean** process(){

```

    People tmp;
    //处理数据，建立倒排索引
    while(Global.queue.getCount() != 0){
        tmp = Global.queue.getOut();
        if(tmp == null)
            return false;
        //数据处理
        if(tmp.getDoc() == false)
            return false;
        tmp.getFriends(Global.queue);
        Global.usedIDList.add(tmp.getID());
        Global.dBer.insertUsedList(tmp.getID());
    }
    return true;
}

public boolean initDB(){
    Global.dBer.connectDB();
    Global.dBer.initUsedList();
    Global.dBer.initValue();
    return false;
}

}

2. 数据分析模块
dataAnalyzer.java
package DataAnalyzer;

public class dataAnalyzer {
    public void mainAnalyzer(){
        Global.dBer.connectDB();
        zplKMeans kMeans;

        kMeans = new zplKMeans(20);
        kMeans.readVectorSet();
        kMeans.initRandom();
        kMeans.start();
        kMeans.showAll();
        Global.dBer.closeDB();
    }
}

```

```
}  
zplCluster.java  
package DataAnalyzer;  
  
public class zplCluster {  
    private docVector _mean;  
    private int clusterNum;  
    private TermVector _vector;  
  
    public zplCluster(){  
        _mean = new docVector(-1);  
        _vector = null;  
        clusterNum = 0;  
    }  
    public int getNum(){  
        return clusterNum;  
    }  
    public boolean addDoc(docVector v){  
        if(v !=null){  
            _vector.addVector(v);  
            clusterNum ++;  
            return true;  
        }  
        return false;  
    }  
    public docVector getMean(){  
        return _mean;  
    }  
    public TermVector getTerm(){  
        return _vector;  
    }  
    public void UpdateMean() {  
  
        int weidu = Global.weidu;  
        double[] value = new double[weidu];  
        for(int i = 0; i< weidu;i++){  
            double tmp = 0;  
            for(docVector v:_vector.getList()){  
                tmp += v.getValue()[i];  
            }  
        }  
    }  
}
```



```

        }
        value[i]= tmp/(double)weidu;
    }
    _mean.setValue(value);
    try{
        if(_mean.computeLength() ==0)
            throw new Exception();

    }catch (Exception e) {
        // TODO: handle exception
        System.out.println("list len:"+_vector.getList().size());
        if(_vector.getList().size() == 0){
            System.out.println("++++");
        }
    }
}

}

public boolean setCluster(TermVector _v) {
    if(_v != null){
        _vector = _v;
        System.out.println("list cluster len:"+_v.getList().size());
        UpdateMean();
        return true;
    }
    return false;
}

}

}

zplKMeans.java
package DataAnalyzer;
import java.io.File;
import java.io.IOException;
import java.util.Random;

import jxl.Workbook;
import jxl.write.Label;
import jxl.write.WritableSheet;
import jxl.write.WritableWorkbook;

```

```

import jxl.write.WriteException;
import jxl.write.biff.RowsExceededException;

public class zplKMeans {
    private int docNum;
    private int _k;
    private int[] _result;
    private zplCluster[] _clusterSet;
    //原始数据
    private TermVector _vectorSet;

    public zplKMeans(int k) {
        Global._k = k;
        docNum = Global.docNum;
        _k = k;
        _clusterSet = new zplCluster[_k];
        _result = new int[_k];

        for(int i =0; i<_k;i++){
            _clusterSet[i]= new zplCluster();
            _result[i]= i;
        }
    }

    public boolean readVectorSet(){
        _vectorSet = Global.dBer.buildTermVector();
        if(_vectorSet == null)
            return false;
        docNum = Global.docNum;
        return true;
    }

    public void start() {
        int it = 0;
        while(true){
            System.out.println("times:"+it);
            it++;
            //重新计算每个聚类的均值
            for(int i= 0; i<_k;i++){
                _clusterSet[i].UpdateMean();

                System.out.println("mean:"+_clusterSet[i].getMean().computeLength());
            }
        }
    }
}

```

```

        if(_clusterSet[i].getMean().computeLength() == 0){
            System.out.println("id:"+i);
            return;
        }
    }

    //计算每个数据和每个聚类的距离
    for(docVector set:_vectorSet.getList()){
        for(int j=0;j<_k;j++){
            double dist = TermVector.getDist(set,
            _clusterSet[j].getMean());
            set.setDist(j, dist);
        }
    }

    //计算每个数据离那个聚类最近
    for(docVector set:_vectorSet.getList()){
        set.setnearestCluster();
    }

    //4、比较每个数据最近的聚类是否就是它所属的聚类
    //如果全相等表示所有的点已经是最佳距离了
    int k = 0;
    for(docVector set:_vectorSet.getList()){
        if(set.isEqAssig())
            k++;
    }
    if (k == _vectorSet.getdocNum())
        break;

    //5、否则需要重新调整资料点和群聚类的关系，调整完毕后再重新开始循环；
    //需要修改每个聚类的成员和表示某个数据属于哪个聚类的变量
    //清空完
    for(int i= 0; i<_k;i++){
        _clusterSet[i].getTerm().clearAll();
    }
    for(docVector set:_vectorSet.getList()){
        _clusterSet[set.getnearestCluster()].addDoc(set);
        set.setAeqN();
    }
}

}

```

```
public int getK() {
    return _k;
}

public int[] getRes() {
    return _result;
}

public void initRandom() {
    int[] intRet = new int[_k];
    for(int i =0; i<_k;i++){
        intRet[i]= -1;
    }
    int intRd = 0; //存放随机数
    int count = 0; //记录生成的随机数个数
    int flag = 0; //是否已经生成过标志
    while(count<_k){
        Random rdm = new Random(System.currentTimeMillis());
        intRd = Math.abs(rdm.nextInt())%docNum+1;
        for(int i=0; i<_k;i++){
            if(intRet[i] == intRd){
                flag = 1;
                count --;
                break;
            }
        }
        if(flag == 0){
            intRet[count] = intRd;
            docVector vector = Global.dBer.buildDocVector(intRd);
            if(vector == null){
                count --;
            }
        }
        else {
            TermVector termVector = new TermVector(vector);
            if(_clusterSet[count].setCluster(termVector) == false){
                count --;
            }
        }
        else {
            _vectorSet.removeVector(vector);
        }
    }
}
```

```

        System.out.println("IDnum:"+intRd+";content:"+vector.getWeiboCon());
    }
}

}
count++;
flag = 0;
}

}

public void showAll(){
    sortC();
    int j = 1;
    //打开文件
    WritableWorkbook book = null;
    try {
        book = Workbook.createWorkbook(new File("D://result.xls"));
        //生成名为“第一页”的工作表，参数0表示这是第一页
        WritableSheet sheet = book.createSheet("第一页", 0);
        for(int i=_k-1;i>0;i--){
            String cotString;
            int size = _clusterSet[_result[i]].getTerm().getList().size();
            if(size > 10){
                cotString =
_clusterSet[_result[i]].getTerm().getList().get(0).getWeiboCon();
                System.out.println("time:"+size);
                System.out.println("text:"+cotString+"
ID:"+_clusterSet[_result[i]].getTerm().getList().get(0).getID());

                //在Label对象的构造子中指名单元格位置是第j列第i行(j,i)以及单元格
                内容为str[j]

                Label labe = new Label(0,j,String(j));
                Label label = new Label(1,j,cotString);
                //将定义好的单元格添加到工作表中
                try {
                    sheet.addCell(labe);
                    sheet.addCell(label);
                } catch (RowsExceededException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }
    }
}

```

```

        } catch (WriteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        j++;
    }
}
book.write();
try {
    book.close();
} catch (WriteException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

/*      int size = _clusterSet[_result[_k-1]].getNum();
    for(int i = 0; i< size;i++){
        String cotString =
_clusterSet[_result[_k-1]].getTerm().getList().get(i).getWeiboCon();

        System.out.println("text:"+cotString+"
ID:"+_clusterSet[_result[_k-1]].getTerm().getList().get(i).getID());
    }
    */

}

private String String(int j) {
    // TODO Auto-generated method stub
    return null;
}

public void sortC(){
    int flag = 0;
    for(int j =_k-1; j>0;j--){
        for(int i =0; i< j;i++){
            if(_clusterSet[_result[i]].getNum() >

```

```
_clusterSet[_result[i+1]].getNum()){  
    int t = _result[i];  
    _result[i]= _result[i+1];  
    _result[i+1] = t;  
    flag = 1;  
}  
}  
if(flag == 0)  
    break;  
}  
  
}  
}
```