密级：　　保密期限：

北京郵電大學

硕士学位论文

题目：基于中文微博的突发话题检测系统的设计与实现

学　　号：2011111605

姓　　名：宋思奇

专　　业：计算机科学与技术

导　　师：王文东

学　　院：网络技术研究院

2013 年 12 月 29 日

独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：_____ 日期：_____

关于论文使用授权的说明

学位论文作者完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。（保密的学位论文在解密后遵守此规定）

保密论文注释：本学位论文属于保密在__年解密后适用本授权书。非保密论文注释：本学位论文不属于保密范围，适用本授权书。

本人签名：_____ 日期：_____

导师签名：_____ 日期：_____

基于中文微博的突发话题检测系统的设计与实现

摘　要

随着社会化网络服务（Social Network Services, SNS）的不断发展，微博（Microblogging）已经成为了很多国人生活中一个必不可少的组成部分。微博是一种通过单向的关注机制来分享简短即时消息的广播式的社交网络平台。作为一种社会化媒体，它所具有的短文本、实时、社交和媒体特性大大缩短了信息从发布到广泛扩散蔓延的时间，更加有利于信息的快速传播。当前，微博已经成为了网络舆论的主要爆发地和聚集地。有效地检测出微博中的突发话题，不管是对普通用户、商家还是政府部门来说，都有着很强的现实意义。

本论文在总结现有突发话题检测研究成果的基础上，设计并实现了一个基于新浪微博的突发话题检测系统（Emerging Topic Detection System），简称 ETD。它实时地从新浪微博采集用户和微博数据，并尽可能提高数据集的完整性与一致性。它使用了一种新颖的的突发话题检测模型，能够更准确地将大量微博中的突发话题检测出来。最后通过使用一些最新的数据可视化技术，将检测出的突发话题信息在 Web 前端进行展示。

论文首先对突发话题检测的相关理论和技术背景进行了简要介绍；之后详细描述了基于中文微博的突发话题检测系统 ETD 的需求分析，并对整个系统进行了总体设计；接下来描述了系统内部各个子系统的详细设计与实现，包括数据采集子系统、突发话题检测子系统和突发话题可视化子系统；然后，对整个系统进行单元测试和集成测试，表明整个系统达到了预期的设计目标；论文最后对全文进行了总结，对未来的工作进行了展望，并总结了作者在研究生期间的所有工作和成果。

关键词　中文微博 突发话题检测 社交网络 数据采集 数据可视化

DESIGN AND IMPLEMENTATION OF EMERGING TOPIC DETECTION SYSTEM BASED ON CHINESE MICROBLOGGING

ABSTRACT

With the continuous development of SNS, microblogging services have already become a necessary part of daily life of many Chinese people. Microblogging is a broadcast social network platform that people share brief instant messages via a one-way follow mechanism. Being a social medium, its features of short text, real time, sociality and media greatly reduce the time interval between release and spread of information, which favours rapid spread of news. At present, microblogging has become a main aggregation and outbreak site of online public opinions. Effectively detecting emerging topics from microblogging platforms has great practical significance for common users, online businessmen and government departments.

This thesis sums up the existing research findings in the field of emerging topics, and then designs and implements an emerging topic detection system based on Sina Weibo, which is called ETD. The system collects data of users and tweets of Sina Weibo in real time, improves integrity and consistence of the data set via a certain scheduling strategy. It utilizes a novel emerging topic detection model, which can detect emerging topics from mass tweets accurately. Finally it displays information of the detected topics in Web front-end by adopting some latest data visualization technologies.

First, this thesis introduces the related theories and technical background of emerging topic detection. Second, requirement analysis and system design of the emerging topic detection system based on Chinese microblogging are depicted in detail. Third, the detailed design and implementation of each subsystem are highlighted, including data acquisition, emerging topic detection and

visualization. Then unit and integration testing are adopted, which indicates that the entire system achieves the prospective design goals. In the end, the thesis summarizes the whole work of the system, looks forward to future work, and summarizes the work and achievements of the author during the post-graduate period.

KEY WORDS: Chinese microblogging; emerging topic detection; social network; data acquisition; data visualization

目录

第一章 绪论

1.1. 课题背景

微博[1]（Microblogging，微型博客）是一种通过关注机制来分享简短实时消息的广播式的社交网络平台，用户可以通过 Web、Mobile 等各种客户端组建个人社区，以少于 140 字的信息实现即时分享。微博具有社交属性，用户之间通过单向的“关注”关系建立社交网络，无需像 Facebook[2]、人人网[3]那样基于实体世界的真实人际关系，这使得用户之间会更多地基于特定的主题、内容、事件进行互动。同时，微博也具有媒体属性，短文本特性和用户之间的单向关系链条更加有利于信息的快速传播，能够大大缩短信息从发布到扩散蔓延的时间。考虑到社会化媒体拥有的庞大用户数量和社会化网络上信息的指数级扩散速度，微博已经成为快速发布与传播信息的有效平台，突发事件往往会首先在微博上酝酿爆发，然后才在其他传统媒体上继续升温。

作为全球最大的中文微博平台，新浪微博[4]自从 2009 年上线以来，经过几年的蓬勃发展，注册用户规模已经超过 5 亿，每天通过各种终端发布的微博数量超过了 1 亿条，已然成为了最重要的信息聚集和爆发平台。在新浪微博这种信息爆炸的平台之下，如何准确而快捷地获取感兴趣的信息是人们关注的首要问题。由于微博数量太大，与一个话题相关的信息往往会分散在不同的时间和地点出现，仅仅依靠这些孤立的信息，人们很难能够对某些话题或事件进行全面的把握。而突发话题检测技术的出现，使得人们可以将这些分散的信息有效地聚集并组合起来形成话题，从不同侧面完整地刻画一个话题的特征，使用户从宏观上了解一个话题、以及该话题与其它话题之间的关联关系。准确地检测突发话题，还有着很强的现实意义。对于商家来说，透过突发事件，能够更好地了解公众的需求，以便针对不同的人群制定更好的营销策略。对于政府来说，突发话题检测能够帮助政府部门有效地监管网络舆情，提供危机预警的依据，从而有效降低突发事件发生的风险，提高风险的应对能力。此外，研究对新浪微博的突发话题检测，对于学术研究也是很有意义的。将传统的话题检测研究与社交网络的研究相结合，是一个比较新的研究方向。而且这项研究也可以作为社交网络中一些其他研究领域的工作基础，如网络突发事件的传播分析、热点预测、针对话题的意见分析（包括意见领袖识别、社群发现等）、针对话题的社会化搜索等等。

本课题在总结现有突发话题检测研究成果的基础上，设计并实现了一个基于新浪微博的突发话题检测系统（Emerging Topic Detection System），简称 ETD 系统。

1.2. 课题主要研究内容

本课题需要设计并实现一个基于中文微博的突发话题检测系统 ETD，主要研究内容包括：新浪微博的实时数据采集技术、突发话题检测模型和突发话题的可视化技术。

要对新浪微博进行突发话题检测，首先需要采集新浪微博的数据。相比采集、解析并抽取网页内容而言，使用新浪微博提供的 API 进行数据采集的方案更为高效。为了得到更加完备的数据集，需要制定相应的采集控制策略，使用增量采集技术，尽可能地提高微博、用户、用户社会化关系等数据片段的完整性与一致性，以保证更好的实验效果。同时，也需要遵守数据采集的礼貌性，令数据采集能够持续进行。

接下来需要研究的重点内容是基于新浪微博的突发话题检测模型。这里沿用了 TDT 对“话题”的定义——在现实世界里发生的事件及相关事件的组合，它是一组关键词的集合。突发话题则通常涉及到一些公共的突发事件，具有较强的时间突发特性，并遵从产生、发展直到逐渐消亡的完整演变过程。相比而言，时效性不显著的热门话题不在讨论之列。本课题使用了一种新颖的的突发话题检测模型。它定义和引入了一种微博传播价值评价指标，能够有效去除噪声微博；使用了一种基于演化理论[5]（Aging Theory）的突发词检测模型，能够更合理地揭示某个词汇随着话题的兴衰而表现出来的突发度变化趋势；采用了互信息（Mutual Information）模型[6]和图划分算法，能够有效地实现从突发词到突发话题的聚合。比起其他模型或简单的统计和人为标注，它能够更准确地将微博中的突发话题检测出来。

最后，突发话题的可视化技术也是本课题的研究内容之一。数据可视化技术的基本思想是将数据项作为单个图元元素表示，大量的数据集构成数据图像，同时将数据的各个属性值以多维数据的形式表示。它不仅用直观、清晰、有效的方式将数据以及数据间的关系展示出来，而且也为算法模型提供了一种辅助的可交互应用。本课题主要研究的可视化技术为可缩放矢量图形技术，如 SVG、VML 等。这些技术仅仅依赖系统后台提供的数据接口，在前端生成与突发话题数据相关的图形、图表，不仅体现了富客户端应用的理念，也给用户提供了良好的动态呈现和交互效果。

1.3. 主要工作内容

本课题研究中，论文作者主要负责的工作内容如下：

1. 参与前期的技术调研和可行性分析
2. 参与 ETD 系统的需求分析
3. 参与 ETD 系统总体架构的设计和详细设计
4. 负责新浪微博数据的采集工作

5. 负责突发话题检测核心算法的实现与优化
6. 负责搭建突发话题检测可视化展示服务的前端与后台
7. 负责 ETD 系统的部署和测试

1.4. 论文结构

本论文的结构安排如下：

第一章　绪论。介绍了本论文的课题背景、主要研究内容和作者的工作内容等。

第二章　相关技术与理论。讲述突发话题检测的相关理论和技术背景，以及开发过程中在服务器端和 Web 前端开发中用到的相关技术。

第三章　系统设计与研究。首先介绍了 ETD 系统的需求分析，然后介绍突发话题检测模型，讲述了系统的总体设计方案。

第四章　各子系统的设计与实现。详细介绍了 ETD 系统中各子系统——微博数据采集子系统、突发话题检测子系统和突发话题可视化子系统的设计与实现。

第五章　系统测试及功能演示。首先介绍 ETD 系统的测试环境，然后针对各个子系统进行单元测试，接着对整个系统进行集成测试，演示并检验运行效果，最后对测试结果进行分析。

第六章　结束语。对全文进行总结，提出了对未来工作的展望。

第二章　相关技术与理论

2.1. 新浪微博数据采集

2.1.1. 新浪微博 API

新浪微博开放平台[7]提供了开发者 API，让第三方应用能够接入微博数据，从而为双方都创造更大的价值。在第三方应用能够获取并使用用户在新浪微博中的数据之前，新浪微博用户必须对第三方应用进行授权。目前新浪微博使用的是 OAuth 2.0 协议，这种安全、开放而又简单的机制使得用户无需向第三方应用提供微博账号和密码就能完成授权认证。使用 OAuth 2.0 协议进行授权认证的流程图如图 2-1 所示：

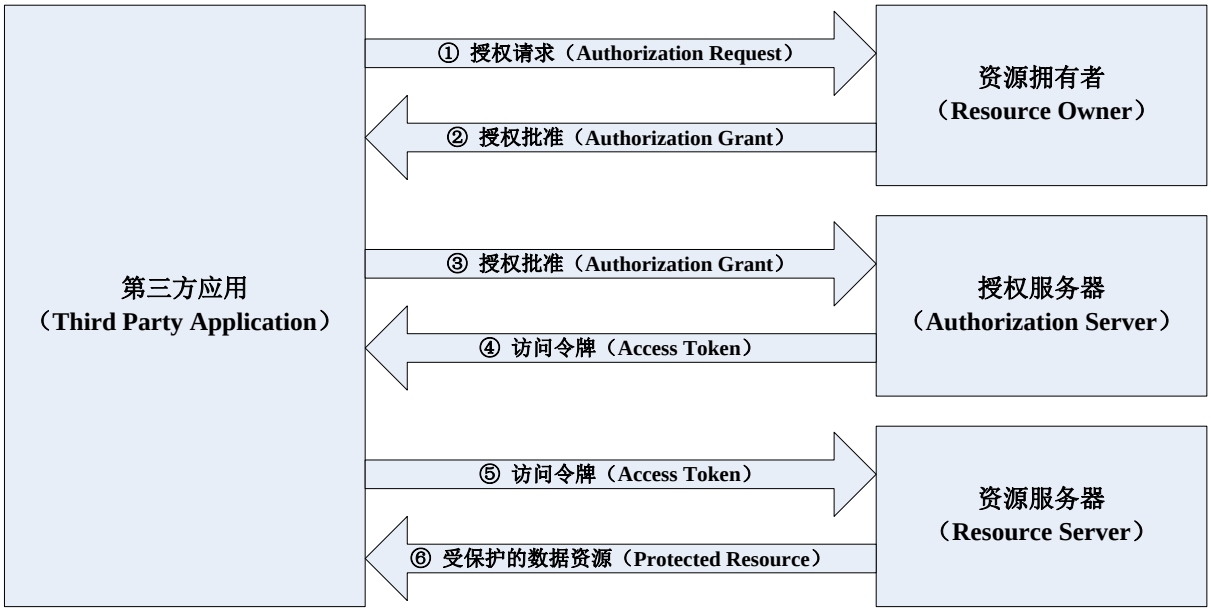

图 2-1 OAuth 2.0 协议的授权流程图

OAuth 2.0 协议有 3 个参与实体，用户、新浪微博和第三方应用。每个第三方应用都有自己的 appKey 和 secretKey。在授权认证时，第三方应用首先进入新浪微博提供的授权页面、或弹出授权框，在用户输入微博账号和密码并点击授权按钮之后，授权请求（Authorization Request）被提交给了资源拥有者（Resource Owner）新浪微博，然后新浪微博返回一个授权批准（Authorization Grant）响应。接着第三方应用使用这个授权批准（Authorization Grant）响应去授权服务器（Authorization Server）去换取一个访问令牌（Access Token）。只有通过使用这个访问令牌（Access Token），第三方应用才能从资源服务器（Resource Server）中获取到新浪微博中受保护的数据资源（Protected Resource），例如用户、微博、社会化关系、搜索、评论、地理位置数据等等。

2.1.2. 微博数据采集方法

对新浪微博的数据进行采集主要有两种方法：解析页面和调用新浪微博 API。

解析页面是最常见的方法，适合采集任何站点的数据。它通过模拟浏览器向网站的服务器发送 HTTP 请求，带上合适的参数，网站就会返回网页，然后解析网页 HTML 中的内容，就能获取所需的数据。随着 Ajax 的盛行，越来越多的动态网页代替了静态网页。在新浪微博的用户微博页面，对于某用户发布的所有微博，需要分 3 批才能加载完该用户的一页微博（45 条/页）。其中，第 1 批的 15 条位于页面初始的 JavaScript 代码中，后 2 批的 30 条会随着用户滚动页面使用 Ajax 传送。因此，还需要发送额外的 Ajax 请求才能获取到完整的数据。由于单个请求返回的响应数据内容并不多，而且对页面 HTML 内容的解析又非常繁琐，数据采集效率并不高。

第二种数据采集方法是调用新浪微博提供的 API。首先需要创建一个第三方应用，获取 appKey 和 secretKey。接着注册一些测试用户，让每个测试用户通过 OAuth 2.0 协议授权创建的应用去使用自己的微博账号和密码，换取 Access Token，然后创建的第三方应用就可以使用这些 Access Token 来调用新浪微博提供的 API 了。尽管使用新浪微博提供的 API 会存在一些限制，但是实测结果表明，不管是程序开发效率还是实际的数据采集效率，它都要高于解析新浪微博页面。

ETD 系统选择使用新浪提供的开发者 API 进行数据采集，并把解析页面和 Ajax 内容作为一个可行的备选方案。

2.2. 突发话题检测

2.2.1. 突发话题检测的相关研究

话题检测与跟踪[8]（Topic Detection and Tracking, TDT）是近年提出的一项信息处理技术，旨在帮助人们应对日益严重的互联网信息爆炸问题。目前的 TDT 研究主要两大类方法：基于文本和基于特征。

基于文本相似度的分类、聚类算法多应用于传统的新闻广播类报道。它将多个文本聚类为不同的集合，然后通过计算热度、媒体权威度、用户关注度等各种指标，挖掘出聚合后的文本中隐含的突发话题，并对突发话题进行持续跟踪。新闻广播类报道的主题相对比较集中，没有过多的噪声干扰，便于将它们分割为不同的话题，因此基于文本的方法特别适合于新闻广播类报道的话题检测。而对于微博这种短文本、噪声大的社会化媒体则不适用。

随着各种新媒体、社区的出现，突发话题检测方法也朝着更加多元化的方向发展，越来越多的方法选择了基于各种特征来挖掘突发话题。爆发模型[9]（Bursty Model）基

于时间序列的统计特性，依据主题中特征词汇的集中到达度来判断一个话题是否满足爆发的特征。爆发模型特别适合于在论坛、博客等应用中进行突发话题检测，因为在这些应用中，文章的主题都是固定的，只要检测下面回复的时间序列特征即可。同时，它也能够发现与突发话题相关的核心用户，便于舆情监控。同样是基于时间序列分析，进化理论[5, 10]（Aging Theory）则更进一步。它将话题的兴衰看作一个生命周期，分为：出生、成长、衰退、消亡四个过程。它借用了营养学的概念，将文档看作一份食物（Food），它包含了一定的营养值（Nutrition），这些营养值可以转化为话题生命周期的能量值（Energy）。相比爆发模型仅关注话题突发的那一时刻，进化理论更关注话题热度的变化趋势。而且在营养值的计算过程中，也更便于结合一些社交网络的社交特性。最后，还有一种动力学模型[11]（Kinetics Model），它借助了 MACD、KDJ 等金融学概念，将话题看作是一个运动中的物体，根据其“动量”的变化状况来检测突发性。

尽管针对突发话题检测的相关研究已经开展了较长的时间，但是现有的话题检测方法并不能很好地适用于微博这一新型社交媒体，因为与其他媒体相比，微博上的话题零碎且不集中，讨论的主题包罗万象，那些能够真正反映社会突发事件的微博很容易被大量闲谈及广告类型的微博所淹没，只有首先从海量的微博数据中分析出微博所属的话题，才能进一步检测其突发性。由于微博刚刚兴起不久，专门针对微博的话题检测研究正处于起步阶段。当前在学术界已经有了一些针对 Twitter[12]的研究，例如 Sayyadi 等人[13]提出了一种从核心词汇构成的网络中利用社群发现算法来检测微博流中隐含的事件信息；Cataldi 等人[14]利用了进化理论对词汇的生命周期做建模，然后对检测出的所有突发词构成的网络进行图划分，将突发词划分到不同的话题集合中去。虽然中文微博和 Twitter 在功能上相近，但是在语言表答习惯、网民使用习惯等方面还是存在着不小的差异。目前对于中文微博的研究还很少，主要用到的方法也以基于语义的微博短文本分类、聚类方法居多[15, 16]。

2.2.2. 微博站点的话题功能

在工业界，各中文微博站点也都对自身系统内部的突发话题或者热门话题进行了检测，并把相关数据整合到了话题页面，分门别类形成榜单。

图 2-2 和 2-3 分别是是新浪微博的“微话题”页面[17]和腾讯微博的热门话题页面[18]。

图 2-2 新浪微博的“微话题”页面

图 2-3 腾讯微博的热门话题页面

微博小编依据系统内统计的词频，归纳出突发话题或者热门话题，用两个井号“#...#”

来标注，然后为每个话题创建一个相关的话题页面，并呈现与之相关的微博，用户可以在这个话题页面发布带有话题标注的微博。除了使用系统推荐的话题，用户也可以使用两个井号“#...#”来标注任何自定义的话题。

这样的做法虽然完全能够满足站点及用户对于话题的使用需求，但是也存在一些问题：第一，准确度低、噪声过大，一些例如“带着微博去旅行”、“好书推荐”之类的话题虽然排到了榜单的前面，但是没有任何实际意义，无法真实反映系统当前的突发热点。第二，这里标注的话题严格来说是一个标签（Tag），而不是一个话题（Topic），所以存在同一个话题被分拆为不同话题的情况，例如“足协杯”和“足协杯决赛”，又如“亚冠决赛”和“广州恒大”。第三，由于能够随意标注自定义话题，标注的话题完全可以与微博内容没有任何关系，例如用户为了淘宝店铺推广的需要，为相关微博打上了公共事件相关的话题标签，以求能够被系统检索，进而让更多的人看到。

在调研了基于中文微博的突发话题检测技术背景之后，本文将在现有突发话题检测研究和系统实现成果的基础上，使用进化理论（Aging Theory）的思想，设计并实现一个基于新浪微博的突发话题检测系统 ETD。

2.3. 系统开发技术

2.3.1. 服务器端开发技术

2.3.1.1. Spring MVC

Spring[19]是一个基于 J2EE 规范的轻量级开源应用开发框架，是为了解决企业级应用开发的复杂性而创建的。为了降低 Java 开发的复杂性，Spring 采取的关键策略有：基于 POJOs（Plain Old Java Objects）实现轻量级和最小侵入性编程，通过依赖注入和面向接口编程实现松耦合，基于切面和惯例进行声明式编程，通过切面和模板减少样板式代码。MVC 是一种复合设计模式，它由模型（Model）、视图（View）和控制器（Controller）三部分组成，目前已经被普遍接受为一种构建 Web 应用的方法。虽然 Spring 集成了多种流行的 MVC 框架，但是 Spring Web Flow 自带了一个强大的 MVC 框架 Spring MVC。

Spring MVC 是一个请求驱动型的 Web 框架，一个请求所经历的所有处理流程如图 2-4 所示：

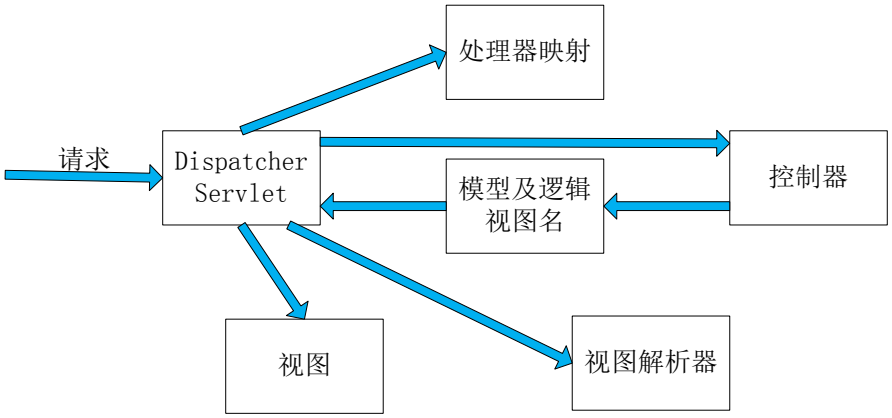

图 2-4 Spring MVC 的请求处理流程

在收到一个请求之后，DispatcherServlet 会根据处理器映射把它分配给对应的控制器，在控制器完成处理后，产生的模型数据会被发给一个视图（根据视图解析器来确定）来呈现输出结果。

ETD 系统采用 Spring MVC 快速搭建灵活、松耦合的 Web 应用服务，有助于将用户界面逻辑与应用逻辑分离。

2.3.1.2. MyBatis 数据持久层框架

MyBatis[20]的前身是 iBatis，是一个半自动化的 Java 数据持久层框架，支持普通 SQL 查询、存储过程以及高级映射。它消除了几乎所有的 JDBC 代码和参数的手工设置以及结果集的检索，使用简单的 XML 或注解进行配置和原始映射，将接口和 Java 的 POJOs（Plain Old Java Objects）映射成数据库中的记录。它的工作流程如下：首先，在程序启动时加载所有配置，并将 SQL 语句的配置信息初始化为一个个 MappedStatement 对象存储在内存中。当调用 Mybatis 提供的 API 时，将请求传递给下层的请求处理层。下层处理层根据传入的 SQL 语句 ID 查找并解析对应的 MappedStatement 对象，得到最终要执行的 SQL 语句和相关参数。然后获取数据库连接，到数据库中执行 SQL 语句，得到执行结果。接着根据 MappedStatement 对象中的结果映射配置对得到的执行结果进行转换处理，以得到最终的处理结果。最后释放数据库连接资源，并将最终的处理结果返回。

MyBatis 使用方便，通过手写核心 SQL 语句和一些简单配置就能实现复杂的数据库读写功能，提高了开发效率以及代码的可扩展性与可维护性，故 ETD 系统采用 MyBatis 完成系统的数据持久化功能。

2.3.2. 浏览器端开发技术

2.3.2.1. jQuery

随着 Web2.0 的兴起，JavaScript 受到越来越多的重视，一系列 JavaScript 库蓬

勃发展起来。从早期的 Prototype、Dojo，再到后来的 jQuery、YUI 和 Ext JS，jQuery 以其独特而优雅的风格，在小型网站的 Web 前端开发中受到了越来越多的追捧。jQuery[21]的核心理念是“write less, do more”。它提供了强大的选择器，方便获取到页面上的各种 DOM 元素包装集；它屏蔽了各种浏览器之间的兼容性问题，封装了很多预定义的对象和实用函数，可以基于这些 DOM 元素包装集进行链式操作；jQuery 提供了丰富的界面效果和可靠的事件机制，并对 Ajax 操作进行了封装，极大地简化了 Web 前端的开发成本，甚至改变了程序员对 JavaScript 的设计思路和编程方式。

2.3.2.2. 数据可视化相关库

数据可视化技术不仅要求用图形、图表等更直观、更清晰、更有效的方式将数据以及数据间的关系展示出来，而且也要给用户提供了良好的动态交互效果。这就需要使用到可缩放矢量图形技术，如 SVG、VML 等。

在众多数据可视化库中，HighCharts[22]和 Raphaël[23]满足了 ETD 系统的数据可视化需求。HighCharts 是一个 JavaScript 图表库，支持的可交互图表类型包括曲线图、区域图、柱状图、饼状图、散状点图和综合图表。而 Raphaël 是一个更加底层的 JavaScript 矢量库，它屏蔽了底层的 SVG、VML 诸多细节，向开发者提供统一的细粒度接口，开发者可以使用这些接口直接操作 DOM 创建出任意复杂度的图像，并基于它进行复杂操作，实现更多个性化的定制。

2.3.3. Web 通信技术

2.3.3.1. Ajax 技术

Ajax[24]（Asynchronous JavaScript and XML）并不是一种单一的技术，而是利用了一系列交互式网页应用技术而形成的有机结合体。它无需插件支持，浏览器只要允许 JavaScript 执行即可；能在不刷新整个页面的情况下更新数据，大大提升了用户体验。对于服务器端来说，请求与响应的异步化减轻了服务器和带宽的负担，提高了 Web 程序的性能。但是 Ajax 也有自身的不足之处，例如它破坏了浏览器前进、后退按钮的正常功能，不利于 SEO 等等。不过随着 HTML5 的普及，越来越多的新技术会逐渐弥补 Ajax 的这些缺憾。总之，它的出现开启了无刷新更新页面的新时代，是 Web 应用开发的一个里程碑。

2.3.3.2. JSON 格式

JSON[25]（JavaScript Object Notation）是一种轻量级的数据交换格式。由于 JSON 是基于 JavaScript 的一个子集，这意味着在 JavaScript 中处理 JSON 数据不需要加载任何特殊的工具包（除了像 IE 6/7 这样的旧版本浏览器不支持 JSON 的解析和序列化，需要加

载 json2.js 库）。相比 XML，JSON 容易阅读、解析速度更快、占用空间更少，C、C++、Java、Python、Ruby 等语言都有相应的 JSON 库支持。JSON 的这些特性使得它成为了前后端之间理想的通信格式，也是未来的流行趋势。

ETD 系统在设计中使用 Ajax 通信方式，制定 JSON 格式的 API，不仅减少了单次数据传输字节流的大小，提升了用户体验，同时也便于扩展，让用户能够感受到更多由 Web 2.0 所带来的新特性和功能。

第三章　系统设计与研究

3.1. 需求分析

3.1.1. 功能性需求

ETD 系统需要实现的主要功能包括 3 个：微博数据采集、突发话题检测和突发话题可视化。

3.1.1.1. 微博数据采集

要检测大量微博数据中的突发话题，首先需要源源不断地从微博站点采集数据。本系统选择了新浪微博作为数据源，采集到的原始数据要求不仅能够作为突发话题检测的基础，也能用于其他社交网络及社会化媒体的研究。

具体用例图如图 3-1 所示：

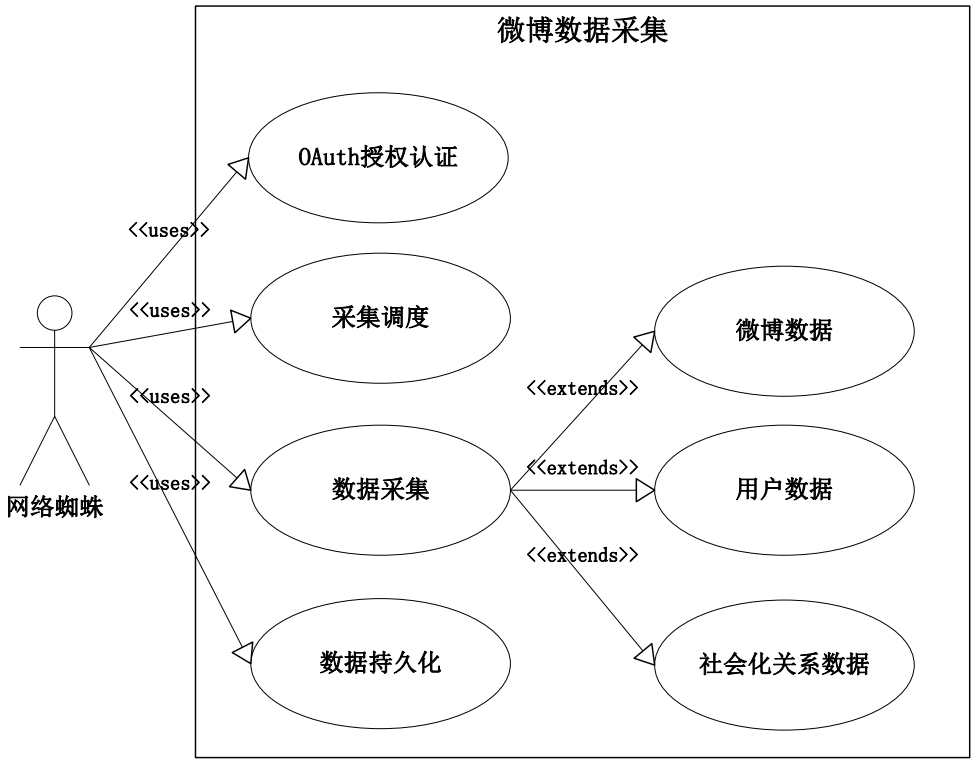

图 3-1 微博数据采集用例图

微博数据采集主要包括以下功能：

- OAuth 授权认证

要使用新浪微博提供的 API 采集微博数据，首先需要在新浪微博第三方平台注册一

个应用，申请一些测试用户，然后让这些测试用户通过 OAuth 2.0 协议授权这个应用去访问新浪微博提供的接口。

- 采集调度

由于新浪微博内的数据量非常大，想要采集整个网络的数据显然是不可能的，只能获取一个较小的数据集。因此，必须制定一个相对较好的采集策略。首先，需要增强数据之间的关联度，以用户为单位顺着关注关系链进行遍历，减少数据缺失的情况，保证数据集的完整性。其次，采用增量采集的策略，不断重访并更新已经访问过的用户内容，防止采集到的数据快照与新浪微博实时内容之间的时间间隔过大，提高整个数据集的新鲜度，保证数据集的一致性。

- 数据采集

对于每个新浪微博用户来说，需要采集的数据有：用户的个人信息数据、发布的微博数据和社会化关系数据。用户微博是必不可少的，只有通过分析微博，才能找出其中的突发词，进而分析出其中的突发话题。而用户的个人信息和社会化关系数据更多的被用于数据清洗，通过计算用户影响力，能够帮助分析出微博的传播价值，进而筛选出那些具有传播价值的微博去进行突发话题检测。同时，用户的社会化关系数据也是作为数据采集的线索而存在的。

- 数据持久化

微博数据采集对数据持久化的要求较高，必须精心制定数据库的读写策略。由于每次从新浪微博采集到的数据量较大，必须快速、高效地对这些大批量的数据进行存储。同时，又能够高效地检索数据库，获取下一个待采集的用户。

3.1.1.2. 突发话题检测

突发话题检测是本系统中最重要的功能，给定一个日期，它能从采集到的海量新浪微博数据中分析出其中蕴含的突发话题信息。

要检测突发话题，首先需要经过数据清洗，根据微博在博主所发布的所有微博中占据的相对显著度和博主自身所具备的传播影响力，保留下那些具有较高传播价值的微博，构成后续待处理的微博集合。然后对这些具有高传播价值的微博进行突发词检测，从中提取出突发发度较高的词汇构成突发词集合。接着通过计算突发词之间的互信息，构建以突发词为节点、突发词之间的语义关系为边的突发词网络。最后根据基于节点相似度的图划分算法，对突发词网络进行图划分，得到每一个的子图对应于一个突发话题。

具体的用例图如图 3-2 所示：

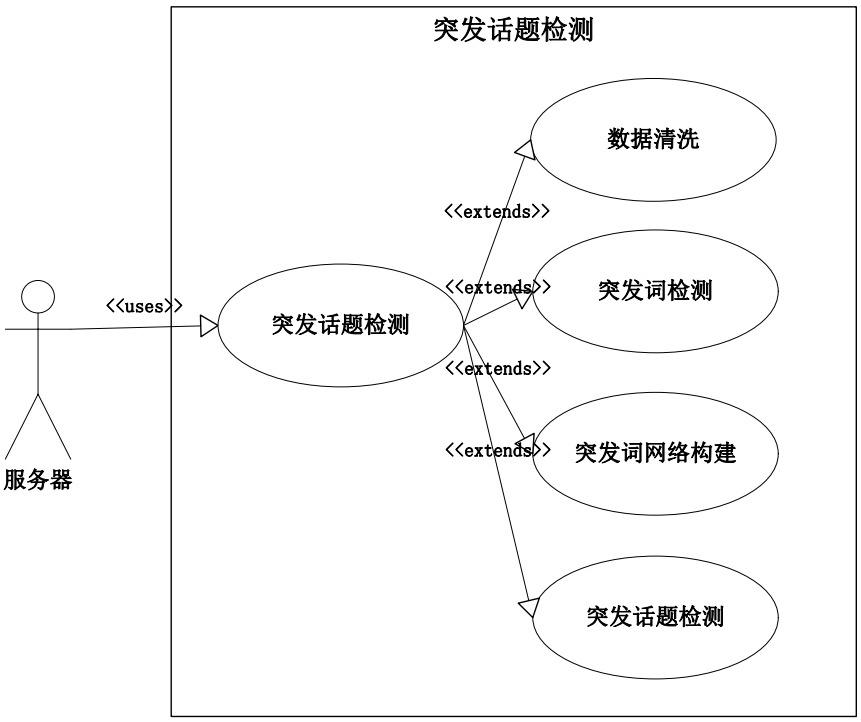

图 3-2 突发话题检测用例图

3.1.1.3. 突发话题可视化

系统将检测到的突发话题以 Web 服务的方式提供给用户进行查阅。用户可以在页面上选择日期，系统会显示所选日期当天从新浪微博中检测到的所有突发话题。用户选择突发话题列表中的某一个突发话题，系统将会展示如下具体信息：

- 话题关键词

由话题核心突发词组成的标签云，字体大小可以反映出词的突发度和突发词之间的关系。

- 话题热度趋势

话题在最近一段时间内的热度变化趋势。

- 地理位置分布

参与突发话题讨论的所有用户在全国各省市自治区的分布。

- 性别比例分布

参与突发话题讨论的所有用户的性别分布。

- 接入终端分布

参与突发话题讨论的所有用户的接入终端分布。

具体的用例图如图 3-3 所示：

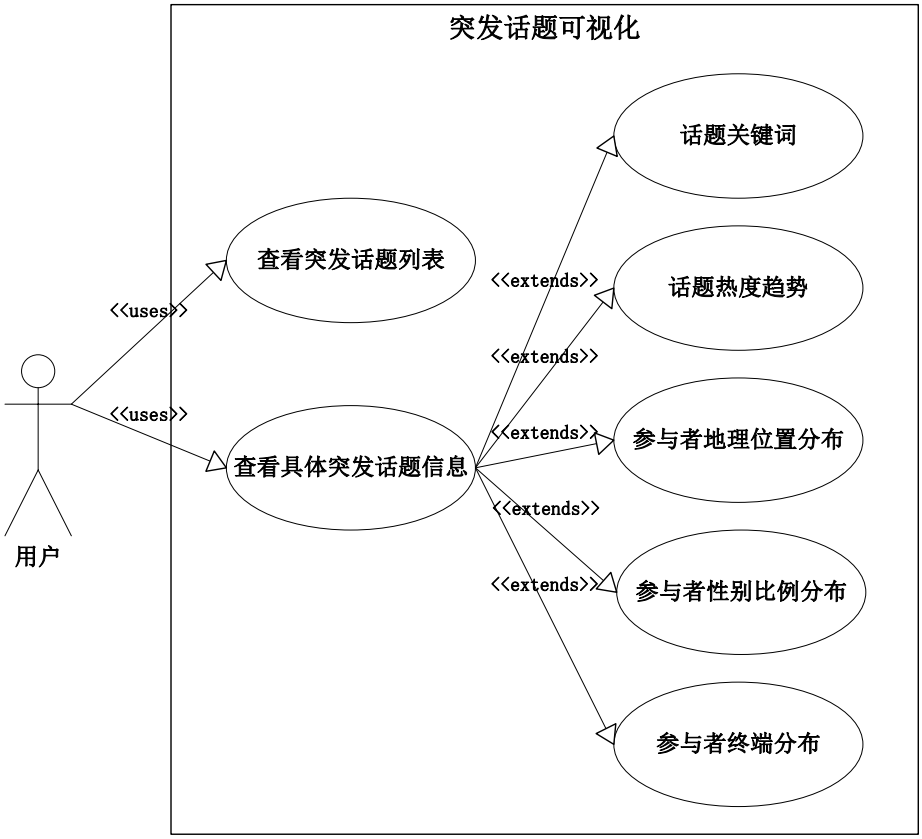

图 3-3 突发话题可视化用例图

3.1.2. 非功能性需求

针对 Web 网站的特点，除了上面提高的功能需求之外，系统还需满足以下非功能需求：

- 性能

系统目前已经积累了数万用户的微博和好友关系信息，并且每天都能新增百万级别的微博、好友关系和新发现的用户。对存储有这样大规模数据的数据库进行操作，必需要确保数据库访问的性能。其次，本文使用突发话题检测模型包括了四个环节，在对海量数据的各种处理过程中，程序的性能也需要得到保证。再次，Web 前端的数据可视化会涉及到了大量的脚本，要减少浏览器执行脚本时占用的系统资源，交互响应要与网站的响应速度一致。

- 稳定性

因为涉及到了对大规模数据的复杂处理，因此必须保证整个系统的稳定性，整个系统应该最大可能的减少崩溃次数，提升容错性。同时，Web 前端页面要稳定运行，在用户进行浏览和操作的时候不能发生页面阻塞或者崩溃。

- 可扩展性

系统在添加新功能或改进已有功能时，整个过程要平滑有序地进行，不仅要确保新

功能的正常运行，也要确保没有改动的功能不受影响，这就需要在对整个系统的各个部件进行设计时要充分考虑可扩展性，比如数据库表设计的兼容性、前后端功能模块的解耦和复用等等。

- 易用性

要确保用户以最小的学习成本来使用系统，用户界面友好，页面布局合理，便于用户交互。提高用户体验，前端在与后台交互时尽量不产生因刷新屏幕而造成的闪烁效果。对各主流浏览器的兼容性较好，在不同浏览器上都能正常显示页面和运行脚本，并支持不同的操作系统。

3.2. 系统设计

3.2.1. 系统架构设计

ETD 系统是一个用户可访问的网站站点，根据上述需求分析中所提到的功能性需求和非功能性需求，整个系统的网络拓扑设计如图 3-4 所示：

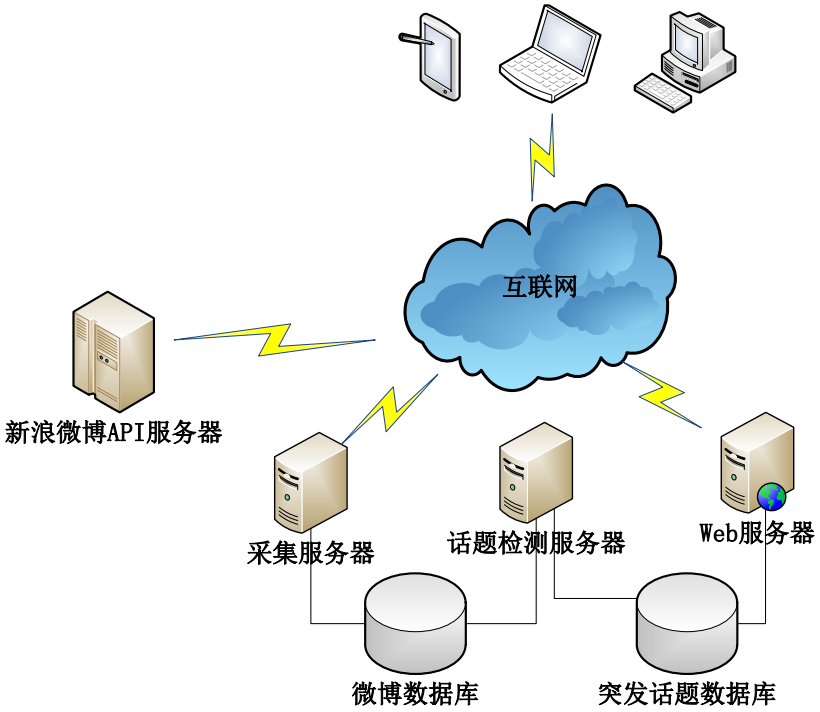

图 3-4 系统网络拓扑图

系统主要包括采集服务器、话题检测服务器、Web 服务器、微博数据库和突发话题数据库。采集服务器负责调用新浪微博提供的 API，实时采集新浪微博中的微博、用户以及用户的社会化关系数据，并将其存储到微博数据库中。话题检测服务器每天定时从微博数据库中读取采集到的原始数据，从中检测出突发话题，将突发话题数据以及一些统计信息存储到突发话题数据库中。当用户通过 Web 浏览器发起查看突发话题请求时，

Web 服务器接收并处理相应的 HTTP 请求，从突发话题数据库中读取相应的数据，以 JSON 格式返回响应；最终用户侧的浏览器将 Web 服务器返回的 JSON 数据渲染为与突发话题相关的各种图形、图表呈现给用户。

由于系统规模和开发条件的限制，目前将采集服务器、微博数据库部署到一台服务器主机上，将话题检测服务器、Web 服务器、突发话题数据库部署到另一台服务器主机上。将来，随着业务和数据处理规模的扩大，可以对这个架构进行进一步的改进。可以使用 Sharding 技术对数据库进行分片存储管理，把每个服务器都扩展成集群，然后使用 Nginx 对其进行负载均衡的处理。

3.2.2. 突发话题检测模型

突发话题检测模型包含四个组成部分：

①数据清洗。

由于新浪微博具有短文本、实时、社交性、媒体性等多种属性，从新浪微博采集获取到的所有微博中往往会存在大量的噪声，即一些无主题或琐碎主题相关的微博。例如某著名的女演员曾经发布的一条微博“今天的天气真不错!”在瞬间就获得了大量的转发和评论，看似得到了粉丝们的追捧，但是这条微博的存在对于挖掘社会关注的突发话题是没有任何帮助的，反而会带来话题漂移问题，并且会降低检测的性能。这些噪声微博在所有微博中占据了相当大的比重，必须设法在实施突发话题检测之前将它们清除掉。这里需要使用一个微博的传播价值模型，只有那些具有较高传播价值的微博才有可能蕴含社会突发话题，也才具有较高的挖掘价值。传播价值取决于两个方面的因素：一是该微博在博主所发布的所有微博中占据的相对显著度，它有效规避了名人效应的存在，相对显著度越高的微博越有可能包含重要的信息；二是博主自身所具备的影响力，它取决于社交网络的结构以及用户之间的交互关系，决定了微博在社交网络中的传播速度、范围以及时间长短。

②突发词检测。

经过第一步的数据清洗之后，只那些具有高传播价值的微博被保留了下来，构成后续待处理的微博集合。这一步检测出的所有突发词都来自于这些具有较高传播价值的微博。因为突发词检测这一步的处理对象是词汇，所以首先必须使用分词技术，把微博切分为一个个独立的词汇，并过滤掉其中的特殊符号和停用词，作为突发词检测的候选词汇集合。传统的话题检测方法在检测突发词时往往仅考虑了特征值在某个时间窗口内的集中到达度，而通常都忽略了特征值在时间序列上的动态变化趋势，这种方法虽然简单高效，但是实际上只是检测出了特征值的热度而不是突发度。为了利用特征值在时间序列上的动态变化趋势，本文采用的方法受到了进化理论（Aging Theory）的启发，利用能量值动态跟踪词汇突发度的演化周期——出生、成长、衰退、消亡。具体的做法为：

根据这些词汇在微博中出现的频率和微博的传播价值，首先计算出词汇在时间窗口内的营养值，然后再根据词汇当前的营养值以及该营养值在之前一段时间内的变化趋势，计算词汇在时间窗口内的能量值。最后对所有的词汇按照能量值大小进行排序，挑选出排序靠前的词汇作为突发词。

③突发词网络构建。

目前已经检测出了在某个时间段内出现的突发词集合，接下来需要发现隐藏在突发词集合中的突发话题。由于话题被视为一组具有相互之间强语义关联的词汇集合，因此在检测出该时间段内的所有突发词后，首先需要计算突发词之间的语义关联关系。互信息（Mutual Information，MI）是一个计算语言学中用于量度两个随机变量之间统计相关性大小的指标，常用于特征提取。因此这里突发词之间语义关联关系被定义为了互信息。若两个突发词在同一条微博中共同出现的次数越多，这两个突发词之间的互信息值越大，反之则互信息值越小。然后以所有的突发词为节点、突发词之间的互信息大小为边的权重构成一个无向加权图。最后将那些权重太低的边去掉，只保留一部分关联关系较大的边，构成最终的突发词网络。

④突发话题识别。

在构建好突发词网络之后，突发话题被看作是突发词网络中的子图。突发话题识别包含以下步骤：首先，根据突发词之间的相关性，即互信息，计算各突发词两两之间的相似度。由于突发词之间的相关性具有对称性及传递性，因此在度量两个突发词节点之间的相似度时，不仅要考虑直接连接两个节点的边的权重，还需要考虑这两个突发词节点的共同邻居节点所传递过来的相关性。然后，对于每个突发词节点，对它的所有邻居节点按照相似度进行排序，挑选相似度最大的邻居节点作为最大邻居。最后，根据基于节点相似度的图划分算法，依靠最大邻居节点对突发词网络进行图划分，每一个得到的子图所包含的突发词集合对应于一个突发话题，突发话题的能量值是它所包含的所有突发词的能量值之和。

3.2.3. 数据库设计

根据之前进行的需求分析和系统架构设计，实现 ETD 系统需要构建两个数据库——微博数据库和突发话题数据库。根据新浪微博 API 和突发话题检测模型，按照如下的方案进行数据库设计。

3.2.3.1. 微博数据库

从新浪微博采集到的原始数据都被存储在微博数据库 Sinaweibo 中，涉及 3 个表：用户信息表 users、用户关系表 user_relation、微博表 statuses。

用户信息表 users 如表 3-1 所示，用于存储新浪微博中注册用户的个人信息。

表 3-1 用户信息表

字段	说明	类型	主键	非空
user_id	用户 ID	bigint	√	√
screen_name	用户昵称	varchar(50)		
name	友好显示名称	varchar(50)		
province	用户所在省级编号	varchar(4)		
city	用户所在市级编号	varchar(4)		
location	用户所在地	varchar(200)		
description	用户个人描述	varchar(200)		
url	用户个人主页地址	varchar(50)		
profile_image_url	用户小头像地址（50*50 像素）	varchar(100)		
domain	用户个性化域名	varchar(50)		
gender	性别：m 为男，f 为女，n 为未知	char(1)		
followers_count	粉丝数	int		
friends_count	关注数	int		
statuses_count	微博数	int		
favourites_count	收藏数	int		
created_at	用户创建（注册）时间	datetime		
following	是否已关注	bit		
verified	是否是加 V 用户	bit		
verified_type	认证类型	int		
allow_all_act_msg	是否允许所有人给我发私信	bit		
allow_all_comment	是否允许所有人对我的微博进行评论	bit		
follow_me	此用户是否关注我	bit		
avatar_large	用户大头像地址	varchar(100)		
online_status	用户的在线状态：0 为不在线，1 为在线	int		
bi_followers_count	用户的互粉数	int		
remark	用户备注信息,只有查询用户关系时才返回	varchar(50)		
lang	用户当前的语言版本	varchar(50)		
verified_reason	认证信息	varchar(200)		
weihao	用户的微号	varchar(50)		
geo_enabled	是否允许标识用户的地理位置	bit		
statuses_frequency	用户当前发布微博的频率（按天计算）	real		
visited	标记该用户是否正在被一个采集客户端采集：0 为未正在采集，1 为正在采集，2 为该用户不存在	int		√
iteration	访问次数	int		√
update_time	上一次更新的时间	datetime		√

用户关系表 user_relation 如表 3-2 所示,用于存储新浪微博中用户之间的关注关系。

表 3-2 用户关系表

字段	说明	类型	主键	非空
source_user_id	源用户 ID	bigint	√	√
target_user_id	目标用户 ID	bigint	√	√
relation_state	关系是否存在，1 存在，0 已经不存在	int		
iteration	访问次数	int		√
update_time	上一次更新的时间	datetime		√

微博表 statuses 如表 3-3 所示，用于存储新浪微博中用户发布的微博以及与微博相关的信息。

表 3-3 微博表

字段	说明	类型	主键	非空
status_id	微博 ID	bigint	√	√
user_id	用户 ID	bigint		
created_at	微博创建时间	datetime		
mid	微博 MID（经 MD5 加密）	bigint		
content	微博内容	varchar(500)		
source_url	微博来源 URL	varchar(200)		
source_name	微博来源名称	varchar(200)		
favorited	是否已收藏	bit		
truncated	是否被截断	bit		
in_replay_to_status_id	回复的微博 ID	bigint		
in_replay_to_user_id	回复的用户 ID	bigint		
in_replay_to_screen_name	回复的用户昵称	varchar(50)		
thumbnail_pic	缩略图片地址	varchar(500)		
bmiddle_pic	中等尺寸图片地址	varchar(500)		
original_pic	原始图片地址	varchar(500)		
retweeted_status_id	转发的原微博 ID	bigint		
geo_type	地理位置类型	varchar(50)		
geo_coordinates_x	地理位置纬度	real		
geo_coordinates_y	地理位置经度	real		
reposts_count	转发数	int		
comments_count	评论数	int		
mlevel	（暂未支持）	int		
iteration	访问次数	int		√
update_time	上一次更新的时间	datetime		√

3.2.3.2. 突发话题数据库

突发话题检测以及可视化产生的数据以及一些中间结果都被存储在突发话题数据

库 Topic-detection 中。

用户表 user 如表 3-4 所示，用于存储每一天所产生的微博所对应的博主的相关数据，主要包括一些截止到当天的统计信息和用户影响力计算相关的信息。

表 3-4 用户表

字段	说明	类型	主键	非空
user_id	用户 ID	bigint	√	√
date	日期	datetime	√	√
all_post	微博总数	int		
all_comment	所有微博所获得的评论总数	int		
all_repost	所有微博所获得的转发总数	int		
follower	粉丝数	int		
ave_post	平均每天所发布的微博条数	real		
activity_degree	用户活跃度	real		
ppv	用户信息传播能力	real		
ir	用户影响力	real		

微博表 weibo 如表 3-5 所示，用于存储每一天所发布的微博的相关数据，主要包括与计算微博传播价值相关的信息。

表 3-5 微博表

字段	含义	类型	主键	非空
status_id	微博 ID	bigint	√	√
date	日期	datetime	√	√
user_id	用户 ID	bigint		
content	微博内容	int		
comment	评论数	int		
repost	转发数	int		
length	长度	int		
url	是否包含 url 链接	bit		
pic	是否包含图片	bit		
rc	微博的相对显著度	real		
iq	微博的信息质量	real		
pv	微博的传播价值	real		

用户表 user 和微博表 weibo 用于数据清洗部分，最终依靠微博表 weibo 中的 pv 字段确定具有传播价值的微博，作为后面突发词检测的输入。

词汇表 term_map 如表 3-6 所示，用于存储通过分词从微博中拆分出的所有词汇，

不含特殊符号和停用词。

表 3-6 词汇表

字段	含义	类型	主键	非空
term_id	词汇 ID（自增）	bigint	√	√
word	词汇	varchar(50)		

词汇词频表 term_tf 如表 3-7 所示，用于存储词汇在微博中的词频、归一化词频和基于类别的词汇权重（CBTW, Category Based Term Weight）等信息。

表 3-7 词汇词频表

字段	含义	类型	主键	非空
term_id	词汇 ID	bigint	√	√
status_id	微博 ID	bigint	√	√
date	日期	datetime	√	√
tf	词汇在微博中的词频	real		
ntf	词汇在微博中的归一化词频	real		
weight	词汇在微博中的基于类别的词汇权重	real		

词汇能量表 term_energy 如表 3-8 所示，用于记录每一天词汇的营养值、能量值等信息，is_emerging 字段标识该词是否为突发词。

表 3-8 词汇能量值表

字段	含义	类型	主键	非空
term_id	词汇 ID	bigint	√	√
date	日期	datetime	√	√
occinyes	在该时间窗口内（当天）至少包含过一次该词汇的微博数	int		
occoutyes	在该时间窗口外（前 7 天）至少包含过一次该词汇的微博数	int		
occinno	在该时间窗口内（当天）没有包含过该词汇的微博数	int		
nutrition	词汇的营养值	real		
energy	词汇的能量值	real		
is_emerging	是否为突发词	bit		

词汇表 term_map、词汇词频表 term_tf、词汇能量表 term_energy 用于突发词检测部分。

突发词互信息表 term_mutual_information 如表 3-9 所示，用于记录每一天检测出的所有突发词两两之间的语义关联关系强度，该强度值由互信息来度量。

表 3-9 突发词互信息表

字段	含义	类型	主键	非空
term1_id	突发词 1 的 ID	bigint	√	√
term2_id	突发词 2 的 ID	bigint	√	√
date	日期	datetime	√	√
t1yest2yes	时间窗口内，两个突发词都同时至少出现过 1 次的微博数	int		
t1yest2no	时间窗口内，突发词 1 至少出现过 1 次而突发词 2 没有出现过的微博数	int		
t1not2yes	时间窗口内，突发词 2 至少出现过 1 次而突发词 1 没有出现过的微博数	int		
mutual_information	两个突发词之间的互信息	real		

突发词拓扑表 graph 如表 3-10 所示，其中的边是突发词互信息表 term_mutual_information 中的边在通过互信息筛选之后的子集，用于记录由突发词构成的无向加权图的相关信息，包括两个突发词之间的互信息以及相似度。

表 3-10 突发词拓扑表

字段	含义	类型	主键	非空
term1_id	突发词 1 的 ID	bigint	√	√
term2_id	突发词 2 的 ID	bigint	√	√
date	日期	datetime	√	√
weight	突发词 1 与 2 之间的权重，即互信息	real		
similarity	突发词 1 与 2 之间的相似度	real		

突发词互信息表 term_mutual_information、突发词拓扑表 graph 用于突发词网络构建部分。

突发词拓扑节点表 graph_term 如表 3-11 所示，用于记录突发词网络中的各个节点的相关信息。其中，max_neighbor 字段标识最大邻居节点，被用于突发话题检测。

表 3-11 突发词拓扑节点表

字段	含义	类型	主键	非空
term_id	突发词 ID	bigint	√	√
date	日期	datetime	√	√
max_neighbor	与该突发词相似度最大的突发词 ID	bigint		
topic_id	突发词所属的话题 ID	bigint		
nutrition	突发词的营养值	int		
energy	突发词的能量值	int		

突发话题表 topic 如表 3-12 所示，记录了系统所检测出的所有突发话题，包含的信息有突发话题的基本信息和统计信息。突发话题的基本信息有话题的描述、营养值、能量值话题所包含的突发词，突发话题的统计信息有包含的所有突发词、话题热度趋势、参与者的地理位置分布、性别比例分布和接入终端分布。

表 3-12 突发话题表

字段	含义	类型	主键	非空
topic_id	话题 ID	bigint	√	√
date	日期	datetime	√	√
topic_description	话题描述	int		
topic_terms	话题包含的突发词信息	int		
topic_nutrition	话题的营养值	int		
topic_energy	话题的能量值	int		
trend	话题的热度趋势	real		
province_distribution	参与话题的地理位置分布	real		
sex_distribution	参与话题的性别比例分布	real		
terminal_distribution	参与话题的接入终端分布	real		

突发词拓扑表 graph、突发词拓扑节点表 graph_term、突发话题表 topic 用于突发话题检测部分。突发话题表 topic 中也包含了一些突发话题的统计信息，用于突发话题的可视化展示。

3.2.4. 开发与运行环境

系统的开发环境如下：

- 后台编程语言：Java（JDK 1.7）
- 集成开发环境：Eclipse Indigo Release
- 服务器端框架：Spring 3.1.1、MyBatis 3.1.1
- 打包与构建工具：Maven 3.0.5
- 中文分词器：ICTCLAS 2013
- Web 前端编程语言：HTML、CSS、JavaScript
- Web 前端调试工具：Chrome 开发工具
- 版本控制：svn

系统的微博数据采集服务部署在 Windows 服务器上：

- 服务器型号：戴尔 PowerEdge R510
- 操作系统：Windows Server 2003
- 数据库：SQLServer 2008 R2 Enterprise

突发话题检测和可视化服务运行在 Linux 操作系统的轻量级 Jetty 服务上，保证了系统的安全性，同时也提升系统的性能和可扩展性。Web 前端部署在了和服务器端同样的环境中，广泛支持各种主流浏览器。具体的运行环境如下：

- 服务器型号：戴尔 PowerEdge 2950
- 操作系统：Ubuntu 10.04
- 内核版本：Linux 2.6.32-21-generic
- 数据库：MySQL 5.1
- Servlet 容器：Jetty 8.1.12
- 浏览器：世界主流浏览器（包括 Chrome、Firefox、Opera、Safari、IE 7+等）以及一些国产的壳浏览器（包括搜狗、360、遨游等）

第四章　各子系统的设计与实现

根据上一章的需求分析和系统设计，将 ETD 系统划分为三个子系统，分别是微博数据采集子系统、突发话题检测子系统和突发话题可视化子系统。下面介绍各子系统的设计与实现。

4.1. 微博数据采集子系统

微博数据采集主要受到两方面的限制：一个是新浪微博 API 的对 IP、应用和用户的访问频率限制，二是程序自身的性能瓶颈。为了突破这些限制，微博数据采集子系统的设计采用了分布式的架构，其架构框图如图 4-1 所示：

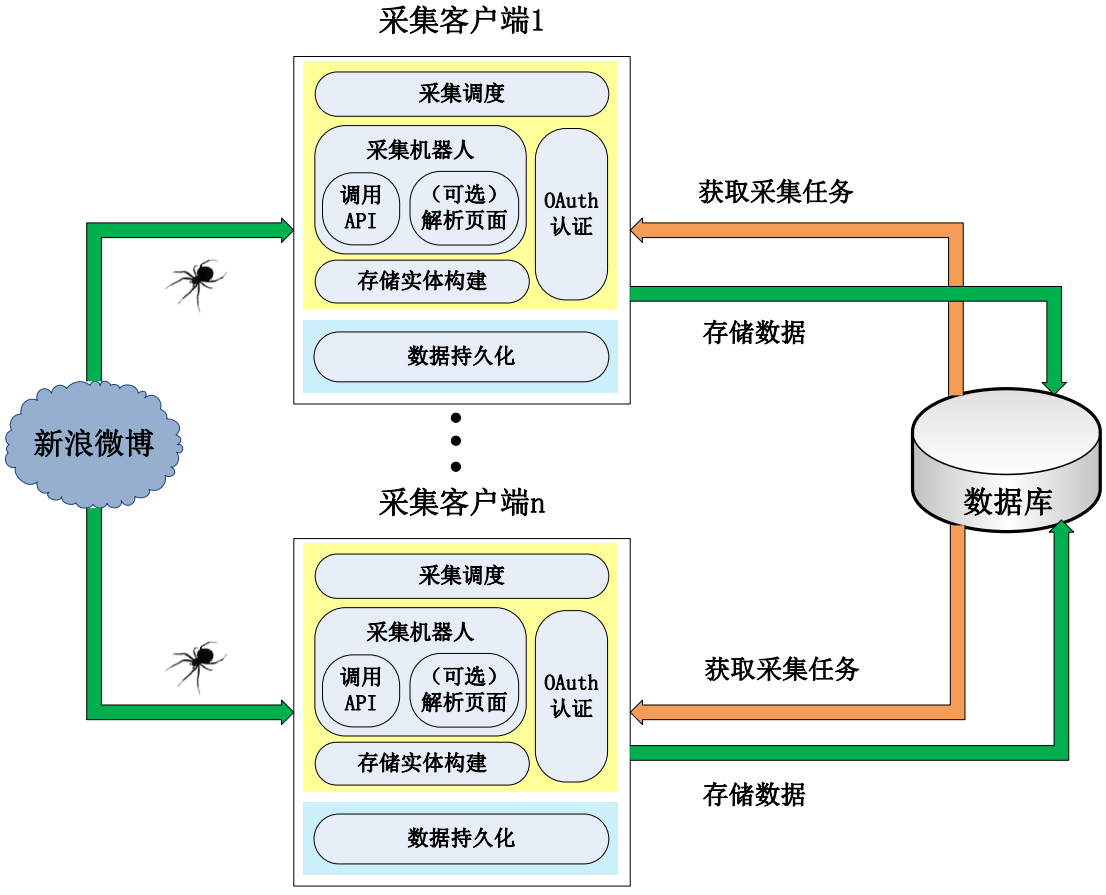

图 4-1 微博数据采集子系统架构框图

整个微博数据采集子系统可以部署 n 个独立的采集客户端进程，每个采集客户端都是以一个在新浪微博注册的第三方应用而存在的，具有自己的 appKey 和 secretKey。在每个采集客户端内，都让一些测试用户通过 OAuth 2.0 协议去授权这个第三方应用去使用自己的微博账号和密码，换取 Access Token。然后采集客户端使用这些 Access Token

调用新浪微博提供的 API 去完成对新浪微博的数据采集工作，最后把这些数据存储到系统的数据库中。

为了确保整个微博数据采集子系统能够有更好的中央控制，同时保证客户端功能的简单，采集客户端中不保存任何待采集的用户队列，而是每次都从数据库中获取。各采集客户端之间对用户的互斥访问由 users 表的 visited 字段（见表 3-1）进行控制：0 代表当前没有客户端正在采集该用户的数据，1 代表某个采集客户端正在采集该用户的数据，2 代表该用户在新浪微博系统中不存在。每个客户端在选择下一个待采集的用户时，都要首先将其 visited 字段赋值为 1，在采集完成之后才将其重新清 0。

下面对采集客户端中的各个模块进行详细介绍。

4.1.1. 采集调度模块

采集调度模块负责调度微博采集客户端的采集任务。

为了提高数据的完整性与一致性，以用户为中心采集用户的个人信息、关注与粉丝列表、发布的微博。为了便于控制数据采集的频率，保持设计的简洁，所有采集过程在采集客户端的主线程内顺序进行，而不把它们分为不同的线程单独采集。采集策略为广度优先，同时不限制用户关系采集的深度。

微博数据采集子系统在刚刚启动时，首先进行累积式（Batch）采集，只采集那些新发现的用户的数据。当采集了一定数量的新浪微博数据之后，整个采集过程就需要向增量式（Incremental）转变，在不断采集新用户数据的同时，也要定期重访并更新那些已经采集过的用户数据。通过更多地进行数据集的日常维护与即时更新，提高并维持整个数据集的新鲜度（Freshness）。

由于批量存储的数据量较大，在存储新用户 ID、用户的社会化关系和用户微博时所耗费的时间较长。因此，每当需要存储数据库时，就开辟一个新的数据存储线程负责把采集到的数据存储到数据库中，而在它存储数据的过程中，数据采集主线程还能继续采集新的数据。这样的做法提高了程序运行的效率。

经过上述对访问策略和存储控制的讨论，一个微博采集客户端的基本运行流程被设计如下：

①设置种子用户 ID 到数据库中；

②用户通过 OAuth 2.0 协议进行授权，生成一组可用的 Access Token；

③设置重访相关参数；

④获取待采集的用户信息（新用户 & 重访）；

⑤采集用户的数据，包括个人信息、社会化关系（关注和粉丝列表）、微博，存入微博数据库 Sinaweibo；

⑥重复第④、⑤两步，不断采集新的用户；

⑦每隔不到一天的时间（如 23 个小时）重复第②步，重新生成新的 Access Token 集合，并更新重访相关参数。

微博采集客户端更加详细的运行流程如图 4-2 所示：

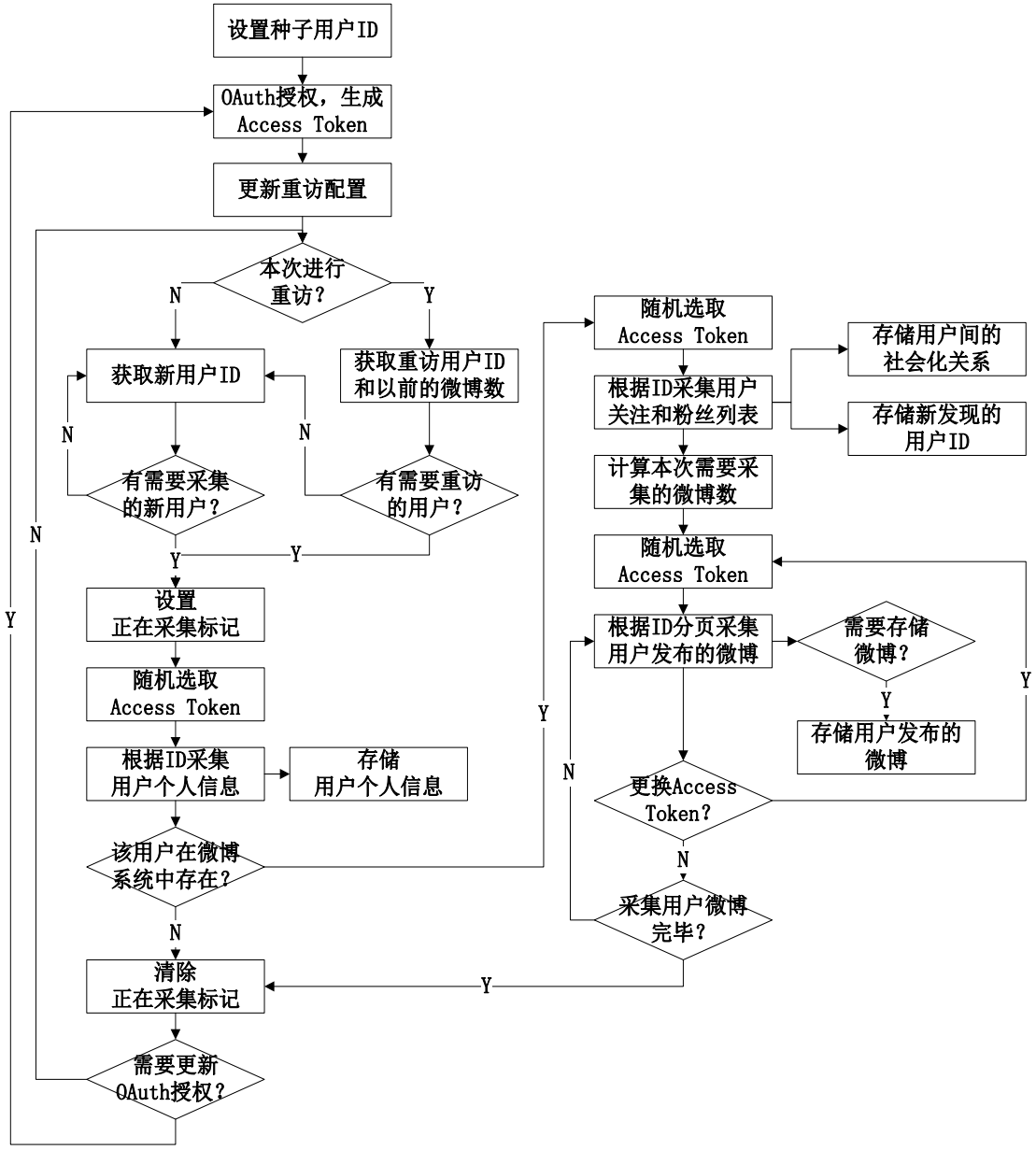

图 4-2 微博采集客户端的运行流程图

下一个待采集的新用户的选择条件是，在 users 表（见表 3-1）中的 iteration 为 0、visited 为 0 并且 update_time 最小（即被发现的最早）。

对于重访，需要配置以下参数：

- revisit_ratio：重访占所有访问的比重
- min_statuses_frequency：重访用户的最小发布微博的频率
- min_days_since_last_visit：最小重访时间间隔
- 系统会定期扫描配置文件 revisit.properties 来更新重访配置。

下一个待采集的重访用户的选择条件是，users 表中的 iteration 大于 0、visited 为 0、statuses_frequency 大于最小发布微博的频率 min_statuses_frequency、上次访问时间 update_time 距离现在大于最小重访时间间隔 min_days_since_last_visit 并且 update_time 最小。

4.1.2. OAuth 授权认证模块

OAuth 授权认证模块为微博数据采集提供可用的 Access Token，是通过 API 进行微博数据采集的基础构件。

首先定义两个配置文件 userinfo.txt 和 config.properties。

- userinfo.txt

配置若干个在新浪微博注册的测试用户的信息，包括用户名和密码。

- config.properties

配置第三方应用及新浪微博 OAuth 的相关信息，包括第三方应用的 appKey、secretKey 和认证之后的回调页面，以及 OAuth 的授权地址和获取 Access Token 的地址。

数据采集子系统在启动时，授权认证模块首先会读取所有的配置信息，然后在数据采集的过程中提供 2 个公有方法供采集调度模块调用：

- AccessToken.generate()

使用所有测试用户的账户信息对这个第三方应用进行批量授权，获取 Access Token。它封装了整个 OAuth 2.0 授权认证的过程。在系统初始化的时候需要使用这个方法。由于每个 Access Token 最多只能有 1 天的有效期，因此每隔不到一天左右的时间就需要重新调用这个方法去获取最新的 Access Token。

在该方法的实现中，使用到的具体 API 如表 4-1 所示：

表 4-1 使用 OAuth 2.0 协议授权认证过程中使用到的 API

API	说明
https://api.weibo.com/oauth2/authorize	用户授权认证
https://api.weibo.com/oauth2/access_token	获取 Access Token

- AccessToken.setOneAccessToken()

在所有可用的 Access Token 中随机选择一个供数据采集使用。由于每个用户的 Access Token 使用次数有限，因此必须通过这个方法来平均分配每个 Access Token 的使用，以防止访问次数超过上限。

4.1.3. 数据采集模块

本系统只采集新浪微博的用户个人信息、用户微博和用户社会化关系数据。数据采集模块的接口定义定义如图 4-3 所示：

```
public interface WeiboInter {
    // 根据用户 ID 获取用户信息
    public User getUserById(String userId);
    // 根据用户 ID 获取用户的粉丝 ID 列表
    public String[] getFollowersIdsByUserId(String userId);
    // 根据用户 ID 获取用户的关注 ID 列表
    public String[] getFolloweesIdsByUserId(String userId);
    // 根据用户 ID 获取用户发布的微博列表
    public List<Status> getStatusesByUserId(String userId);
}
```

图 4-3 数据采集模块的接口定义

当前是通过调用新浪微博 API 来实现这个接口的，使用到的具体 API 如表 4-2 所示：

表 4-2 数据采集过程中使用到的 API

API	说明
https://api.weibo.com/2/users/show	获取用户的个人信息
https://api.weibo.com/2/statuses/user_timeline	获取用户发布的微博
https://api.weibo.com/2/friendships/friends	获取用户的关注列表
https://api.weibo.com/2/friendships/followers	获取用户的粉丝列表

使用一个合法的 Access Token 调用这些 API，新浪微博就会返回相应的 JSON 格式的数据。以获取李开复的个人信息为例，新浪微博 API 返回的数据如图 4-4 所示：

```
{
    "id": 1197161814,
    "name": "李开复",
    "location": "北京  东城区",
    "description": "创新工场 CEO，媒体联系：press@chuangxin.com",
    "profile_image_url": "http://tp3.sinaimg.cn/1197161814/50/1290146312/1",
    "followers_count": 51759259,
    "friends_count": 556,
    "statuses_count": 13779,
    "favourites_count": 878,
    "created_at": "Fri Aug 28 16:35:46 +0800 2009",
    /* 其他数据项 */
}
```

图 4-4 调用 2/users/show 接口返回的数据示例

需要注意的是，对于获取用户的个人信息、关注列表和粉丝列表，都只需要调用一次 API 就可以了；而对于获取用户发布的微博，则需要调用若干次 API 分页返回。因此在每获取一定页数的微博之后，需要更换 Access Token，防止用户访问次数被耗尽；同时把这些微博先批量存储，防止最后统一批量存储微博时因数据量过大而导致数据库在短时间内读写性能降低。

有时，因为不合理的调用或者一些其他因素，新浪微博 API 偶尔也会抛出异常，本系统主要处理的异常如表 4-3 所示：

表 4-3 异常处理

异常码	说明	处理方法
-1	网络连接断开	采集暂停一段时间后再进行尝试。只要检查并排除了网络连接故障，采集即可自动恢复
10022	IP 请求次数超过上限	采集暂停一段时间
10023	用户请求次数超过上限	更换 Access Token
20003	用户不存在	通知采集调度模块，在数据库中标记此用户不存在，停止继续采集这个用户的社会化关系和微博，换下一个用户进行采集

由于这个数据采集模块的接口是可插拔的，因此假如在某段时间因为一些不可抗拒的因素导致新浪微博 API 失效，还可以使用解析页面和 Ajax 等其他方法来实现此接口。

4.1.4. 数据持久化模块

数据持久化模块主要完成两项工作：获取下一个待采集的用户 ID 和存储采集到的数据。由于采集的数据量较大，数据库的容量和运算负荷都很大，因此在对数据库进行读写操作时采用了如下策略和方法：

- 使用连接池

数据采集客户端会经常调用数据持久化模块存储存取数据，如果每次操作数据库之前都创建一个新的数据库连接，在使用完之后再销毁这个链接，势必造成巨大的浪费。因此，本系统引入了一个数据库连接池。系统在启动时先初始化了 30 个数据库连接放入连接池中，每当当需要操作数据库时，连接池管理器便从连接池中中取出一个空闲的连接进行使用，使用完之后再把它释放回连接池等待下一次使用。当连接池中的连接不够用时，连接管理器还可创建更多新的数据库连接放入连接池中。在使用了数据库连接池之后，创建和销毁数据库连接的开销被大大地降低了。

- 利用索引

一个数据采集客户端在获取下一个待采集新用户的条件为访问次数 iteration 为 0，visited 为 0，然后按照上次更新时间 update_time 字段的增序排列选择第一个用户 ID；

而获取待采集重返用户的条件为访问次数 iteration 大于 0，visited 为 0，用户发布微博频率 statuses_frequency 大于某个阈值，上次更新时间 update_time 距离现在大于某个最小时间间隔，然后按照上次更新时间 update_time 字段的增序排列选择第一个用户 ID。由于迅速积累了成百上千万的用户，为了快速找到下一个待采集的用户，需要在用户信息表 users（见表 3-1）上建立非聚集联合索引（update_time, iteration, statuses_frequency）。

- 使用事务

在根据用户 ID 获取用户的关注列表和粉丝列表时，新浪微博会返回很多新的用户 ID；在根据用户 ID 获取用户发布的微博时，新浪微博也会返回很多新的微博。在这些用户 ID、用户之间的关注关系和用户微博中，有些记录已经存在于数据库中了，而有些则是新的，因此在插入数据库时需要使用数据库的事务（Transaction）功能，在记录存在时更新，而在记录不存在时将其插入。具体的 SQL 语句模式为如图 4-5 所示：

```
begin tran;
update 语句
if @@rowcount=0
insert  语句
commit tran;
```

图 4-5 插入数据库时的事务使用模式

- 批量存储

除了根据用户 ID 获取用户个人信息每次只有一条数据记录之外，根据用户 ID 获取用户的社会化关系和微博时，都会返回很多新用户 ID、用户之间的关注关系和用户微博。为了提高数据库访问性能，不能一条一条地将它们插入数据库，必须采用批量的方式进行存储。

在进行数据的批量存储之前，首先要进行存储实体的构建。虽然新浪微博提供的 SDK 对原始的 JSON 数据进行了一些处理，把它转换为 Java 语言的类对象，但是这些对象的成员与之前在第三章中设计的数据库表结构并不完全一致。因此在存储数据库之前，首先要把它们转换为系统自定义的实体（Bean）类。本微博数据采集子系统定义的存储实体类如表 4-4 所示：

表 4-4 存储实体类

存储实体类	说明
UserIdBean	用户 ID 实体类
UserBean	用户实体类
StatusBean	微博实体类
UserRelationBean	用户社会化关系实体类

各实体类对象的数据结构与微博数据库中各表的字段结构相对应。数据在被转换为

实体对象之后，就可以被插入到数据库中了。

4.2. 突发话题检测子系统

突发话题检测子系统运行的时间窗口在当前被设定为了 1 天。通过配置 Java 的 Quartz 定时调度模块，它会在每天的 0 点启动，从微博数据库 Sinaweibo 中读取微博数据采集子系统采集的原始数据，分析出其中蕴含的突发话题，并对其做一些简单的统计，然后将这些数据存储到突发话题数据库 Topic-detection 中。数据流图如图 4-6 所示：

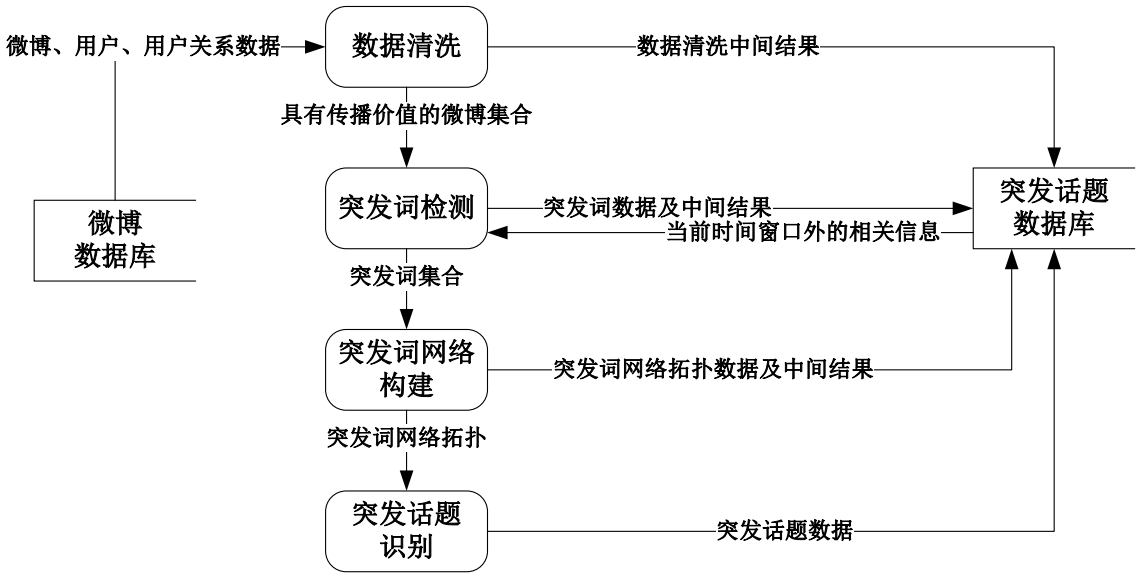

图 4-6 突发话题检测子系统数据流图

为了提高处理效率，整个数据流会不间断地流经突发话题检测的各个环节进行处理，每一步过程中的产生的中间结果都被保存到数据流的核心数据结构中，待进入下一步处理使用。

突发话题检测子系统数据流里面用到的核心数据结构如表 4-5 所示：

表 4-5 突发词检测核心数据结构

数据结构	说明
List<Map<String, Object>> userList	用户列表
List< Map<String, Object>> relationList	用户关系列表
List<Map<String, Object>> statusList	微博列表
List<Map<String, Object>> termStatusList	词汇-微博列表
List<Map<String, Object>> termList	词汇列表
List<Map<String, Object>> emergingList	突发词列表
List<Map<String, Object>> graphList	突发词网络列表
List<Set<Long>> topicList	突发话题列表

尽量少地读写数据库，仅在过程中把需要存储的数据（包括一些有用的中间结果）存储到突发话题数据库中。

下面就突发话题检测的各个处理流程，暨子系统的各个功能模块做具体的介绍。

4.2.1. 数据清洗模块

突发话题检测的第一步是数据清洗。首先从微博数据库 Sinaweibo 中读取用户、用户社会化关系、微博数据，分别组成用户列表 userList、用户关系列表 relationList、微博列表 statusList。然后计算微博列表 statusList 中每条微博的传播价值（Promulgating Value, PV）。微博传播价值的计算过程如图 4-7 所示：

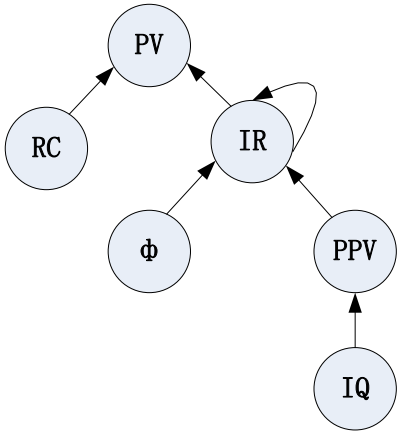

图 4-7 微博传播价值的计算过程

微博 m 的传播价值 $PV(m)$ 取决于两个方面的因素：一是该微博在用户 u 所发布的所有微博中占据的相对显著度 $RC(m,u)$，二是博主自身所具备的影响力 $IR(u)$。计算方法如式（4-1）所示：

$$PV(m) = RC(m,u) * Ln(1 + IR(u)) \tag{4-1}$$

微博 m 的相对显著度 $RC(m,u)$ 对用户 u 发布的微博 m 的转发数 $repost(m,u)$ 和评论数 $comment(m,u)$ 做了归一化处理，能够反映微博 m 在用户 u 发布的所有微博中的重要程度，如式（4-2）所示：

$$RC(m,u) = \alpha * \frac{|repost(m,u)|}{\sum_{m' \in M(u)} |repost(m',u)|} + \beta * \frac{|comment(m,u)|}{\sum_{m' \in M(u)} |comment(m',u)|} \tag{4-2}$$

而用户 u 的影响力 $IR(u)$ 主要包含两方面因素：用户的活跃度 $\phi(u)$ 和用户的信息传播能力 $PPV(u)$。这里采用了类似 PageRank 的思想，体现了活跃用户的影响力更多取决

自身的信息传播能力，而非活跃用户的影响力则更多地来自于粉丝的贡献。如式（4-3）所示：

$$IR(u)=(1-\phi(u))*\frac{\sum_{u'\in follower(u)}IR(u')}{|followee(u')|}+\phi(u)*PPV(u) \tag{4-3}$$

用户活跃度 $\phi(u)$ 考察了用户 u 的粉丝数 $Follower(u)$ 和微博数 $post(u)$，并相对于全网内的最大值做了归一化处理，如式（4-4）所示：

$$\phi(u)=\alpha*\frac{Ln(|follower(u)|)}{Ln(\max_{u'\in V}|follower(u')|)}+\beta*\frac{Ln(post(u))}{Ln(\max_{u'\in V}post(u'))} \tag{4-4}$$

用户的 u 信息传播能力 $PPV(u)$ 可以由用户所发布的所有微博的信息质量反映，如式（4-5）所示：

$$PPV(u)=\sqrt{\frac{\sum_{m\in M(u)}(IQ(u,m))^2}{|M(u)|}} \tag{4-5}$$

其中，对于一条用户 u 所发布的微博 m，它的信息质量可由该微博的转发数 $|repost(m)|$、评论数 $|comment(m)|$、微博长度 $length(m)$、是否包含图片 $pic(m)$、是否包含外链 $url(m)$ 以及该用户的粉丝数 $|follower(u)|$ 来综合衡量，如式（4-6）所示：

$$IQ(u,m)=\left[\frac{|repost(m)|+|comment(m)|+1}{|follower(u)|+1}\right]^{\left(\frac{1}{\frac{length(m)}{140}+pic(m)+url(m)}\right)} \tag{4-6}$$

其中，用户的活跃度 $\phi(u)$、信息传播能力 $PPV(u)$ 和影响力 $IR(u)$ 被添加到了用户列表 userList 的每个对应列表项中，而微博的信息质量 $IQ(u,m)$、相对显著度 $RC(u,m)$ 和传播价值 $PV(m)$ 被添加到了微博列表 statusList 的每个对应列表项中，最后分别将这两个列表中的数据存储到突发话题数据库 Topic-detection 的 user 表（见表 3-4）和 weibo 表（见表 3-5）中。

4.2.2. 突发词检测模块

经过前面的数据清洗模块处理之后，需要在那些具有高传播价值的微博中检测出其中的所有突发词。处理过程依次为：预处理、营养值计算、能量值计算、突发词判别。

4.2.2.1. 预处理

从微博列表 statusList 中选取传播价值（PV 值）大于 0 的微博作为待处理的微博集合。首先通过配置微博黑名单，过滤掉一些没有意义的微博，例如“此微博已被删除”、“此微博已被原作者删除”等。然后使用分词工具 ICTCLAS 2013[26]，对每条微博进行分词处理。分词之后，一条微博就变成了一组由词汇以及它的词性所构成的集合。

接着对得到的词汇进行过滤：

第一步，根据词性进行过滤。当前的词性白名单配置为：名词、动词、形容词、字符串。其中，留下字符串是为了在除中文之外，能够保留更多的非中文信息，例如 iPhone 等等；同时，也需要删除其中代表 URL 的字符串。

第二步，在上一步的基础上，去掉停用词，例如“比如”、“不但”等等。这些词只有在句子的具体语境中才能体现出作用，而在单独存在时没有明确的含义，而且它们的大量存在会对突发词检测的结果造成干扰，因此需要将它们去掉。

最后，对于每个词汇，计算它在包含它的每条微博中的出现的词频和归一化词频。归一化词频的计算如式（4-7）所示：

$$ntf_{t,j} = \frac{tf\left(t_k^t, post_j^t\right)}{\max\left[tf\left(post_j^t\right)\right]} \tag{4-7}$$

其中，$ntf_{t,j}$ 是词汇 t 在在包含了它的第 j 条微博 $post_j^t$ 的归一化词频。

在经过这样一系列预处理之后，微博列表 statusList 就被转化为了一个词汇-微博列表 termStatusList 和一个词汇列表 termList。把词汇列表中的词汇批量存储到 Topic-detection 数据库的 term_map 表（见表 3-6）中，获取数据库中自增的词汇 ID。

4.2.2.2. 营养值计算

营养值反映了词汇随时间的热度趋势。词汇从微博中获得营养，它的计算取决于两方面的因素：每条包含它的微博本身的传播价值和词汇在微博中的基于类别的权重（Category-based Term Weight, CBTW）大小，如式（4-8）所示：

$$nutrition_t^k = \sum_{post_j^t \in POST^k} w_{t,j} * PV\left(post_j^t\right) \tag{4-8}$$

其中，词汇在微博中的基于类别的权重大小的计算方法如式（4-9）所示：

$$w_{t,j} = ntf_{t,j} \cdot \log\left(1+\frac{A}{B+1}\right)\log\left(1+\frac{A}{C+1}\right) \tag{4-9}$$

式中的 A 表示在该时间窗口内至少包含过一次该词汇的微博数，B 表示在该时间窗口外至少包含过一次该词汇的微博数，C 表示在该时间窗口内没有包含过该词汇的微博数。在本系统中，时间窗口被设定为了 1 天，时间窗口之外指的是时间窗口的前 7 天这

个时间段。

上面计算出的词频、归一化词频和基于类别的权重信息的被添加到了词汇-微博列表 termStatusList 的每个对应列表项中。

4.2.2.3. 能量值计算

词汇的能量值反映的是词汇热度的变化趋势，即考虑多个连续时间窗口内营养值的变换情况。在具体计算是，某个词汇 t 的能量值是两部分的叠加：它在当前时间窗口内的营养值、之前 s 个时间窗口分别与当前时间窗口营养值之差的加权和，如式（4-10）所示：

$$energy_t^k = nutrition_t^k + \sum_{x=k-s}^{k-1}\left(\frac{nutrition_t^k - nutrition_t^x}{k-x}\right) \quad (4\text{-}10)$$

在本系统中，时间窗口被设定为了 1 天， s 被设置为了 7。

计算出的词汇的营养值和能量值信息都被添加到了词汇列表 termList 的每个对应列表项中。

4.2.2.4. 突发词判别

遍历词汇列表 termList，基于每个词汇的能量值进行突发词判别，具体方法如下：

a) 去掉所有能量值为负数的词汇；

b) 将能量值为正的所有词汇按照降序排列；

c) 计算该序列的能量值的均值 μ 和标准差 σ，并将能量值大于 $\mu+3\sigma$ 的词汇选作突发词，形成突发词列表 emergingTermList。

经过突发词检测之后，将最终的词汇-微博列表 termStatusList 和词汇列表 termList 中的数据批量存储到 Topic-detection 数据库的 term_tf（见表 3-7）和 term_energy 表（见表 3-8）中。

4.2.3. 突发词网络构建模块

检测出该时间窗口内的所有突发词后，需要构建以突发词为节点，突发词之间的语义关系为边的突发词网络。

突发词网络构建是基于突发词列表 emergingTermList 进行的，把最终形成的网络拓扑信息存储到突发词网络列表 graphList 中。由于本系统中用词与词之间的互信息大小作为边的权重，所以要实现这个方法，首先需要计算互信息。两个突发词 et_i^k 和 et_j^k 的之间的互信息的计算方法如式（4-11）所示：

$$MI\left(et_i^k, et_j^k\right) \approx \log_2\left(\frac{A\times N}{(A+C)\times(A+B)+1}+1\right) \quad (4\text{-}11)$$

式中的 N 表示该时间窗口内的微博总数，A 表示在这个时间窗口内两个突发词共同出现过的微博条数，B、C 分别表示在这个时间窗口内其中一个突发词出现过而另一个突发词没有出现过的微博条数。计算完所有突发词节点之间的互信息，即边的权重之后，把互信息存储到 Topic-detection 数据库的 term_mutual_information 表（见表 3-9）中。

此时如果不对边进行筛选，得到的将是一个全连接的无向图，这个图的规模将会非常大，运算开销很重。所以，将那些权重太低的边去掉，留下一部分关联关系较大的边。在本系统中，阈值被定为了所有边权重均值的一半，大于该阈值的边被留下来，形成了最终的突发词网络。将突发词网络列表 graphList 存储到 Topic-detection 数据库的 graph 表（见表 3-10）中。

4.2.4. 突发话题识别模块

突发话题对应于突发词网络中的子图。子图内部的突发词节点之间的相关性较高，而不同子图内突发词节点之间的相关性交低。突发话题识别功能具体的实现步骤如下：

第一步，基于突发词网络列表 graphList，计算突发词节点之间的相似度。在计算时不仅要考虑直接连接两个节点的边的权重，还需要考虑这两个突发词节点的共同邻居节点所传递过来的相关性，如式（4-12）所示：

$$similarity(i,j)=\frac{w_{ij}}{W}+\sum_{v\in\Gamma(i)\cap\Gamma(j)}\frac{w_{iv}\times w_{vj}}{W^2} \quad (4\text{-}12)$$

其中，w_{ij} 表示突发词节点 v_i 和 v_j 之间的互信息权重，W 代表突发词网络中所有边的互信息权重之和。

第二步，对于突发词列表 emergingTermList 中的每个突发词节点，从突发词网络列表 graphList 中找到它的所有邻居节点，按照相似度进行排序，挑选相似度最大的邻居节点作为它的最大邻居节点。

第三步，基于突发词列表 emergingTermList 中的最大邻居节点信息，使用图划分算法，将它们划分到不同的子图中，即生成突发话题列表 topicList。对于每个突发话题，其中营养值最高的前 3 个突发词连接起来构成默认的话题名称。

最后，对每个识别出的突发话题做一些简单的统计，包括话题热度（话题所包含的突发词营养值之和）趋势、参与者地理位置分布、接入终端分布、性别比例分布等，并把每个统计的结果转换为 JSON 格式。

在上述所有步骤都完成之后，分别将识别出的突发话题信息以及最大邻居节点等中间结果存储到 Topic-detection 数据库的 topic 表（见表 3-12）和 graph_term 表（见表 3-11）。

上述第三步中使用的基于节点相似度的图划分算法如图 4-8 所示：

输入：突发词网络 G

输出：子图集合 $\{g^1, g^2, ...g^n\}$

初始化：$i=0$

算法过程：

FOR 突发词网络 G 中的每个突发词节点 v_i

获取与它相似度最大的邻居节点 v_m

IF 最大邻居节点 v_m 存在

IF v_i 和 v_m 两个节点都不属于任何一个子图 $g^k(1 \le k \le i)$

创建新的子图 g^{i+1}，并添加 v_i 和 v_m 两个节点

$i=i+1$

IF v_i 和 v_m 两个节点中的一个属于一个子图 $g^k(1 \le k \le i)$，而另一个不属于任何子图

添加另一个节点到子图 g^k

IF v_i 和属于一个子图 $g^j(1 \le j \le i)$，而 v_m 属于另一个子图 $g^k(1 \le k \le i)$

合并子图 g^k 到 g^j，删除子图 g^k

$i=i-1$

ELSE IF v_i 不属于任何一个子图 $g^k(1 \le k \le i)$

创建新的子图 g^{i+1}，并添加节点 v_i

$i=i+1$

END

图 4-8 基于节点相似度的图划分算法

4.3. 突发话题可视化子系统

突发话题可视化子系统为突发话题检测的结果提供了 Web 展示服务，用户可以通过浏览器进行查看。整个子系统使用 Spring MVC 和 MyBatis 框架搭建，使用 Maven 打包、构建，部署在 Linux 服务器下的 Jetty Web 容器中。

4.3.1. Web 后台服务器

突发话题可视化子系统的后台设计体现了两种思想：分层架构与 MVC 模式。

分层架构是对复杂软件的一种纵向切分，将各种代码以其完成的使命作为依据来分割，每一层次中完成同一类型的操作，层次之间向下依赖，以降低软件的复杂度，提高可维护性。

本系统从上到下被划分为了 4 层，依次为表现层、控制器层、业务逻辑层和数据持久化层，如图 4-9 所示：

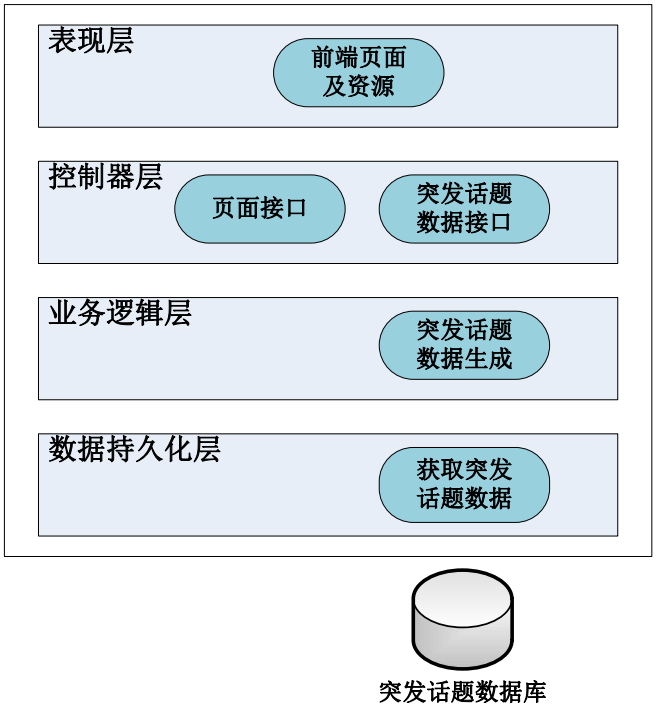

图 4-9 后台分层架构

- 表现层

表现层提供了前端页面，以及页面所依赖的资源，包括图片、样式表、JavaScript 脚本组件等。

- 控制器层

控制器层为前端提供了在 Web 前端通过浏览器访问后台服务的接口，它截取用户在浏览器端发出的页面或数据请求，交给业务逻辑层处理，并将结果返回给前端浏览器。为了降低页面前端和后台服务的耦合度，将页面中的可视化渲染工作留给前端完成，突发话题可视化子系统的后台只向 Web 前端提供两个访问接口：

① /emerging

提供突发话题检测页面。

② /emerging_topics.json

这是一个只返回 JSON 数据的 Restful 风格的 API，根据传入 date 参数获取这一天

的突发话题数据。如果没有传入 date 参数，则默认返回的是昨日的突发话题数据。接口返回数据格式示例如图 4-10 所示：

```
[
    {
        "info": { "id": 7, "rank": 1, "name": "h7n9 禽流感" },
        "trend": {
            "trendList": [ 0, 1.7687, 2.1782, /*  其他热度趋势数据  */ ],
            "dateList": [ "2013-03-30", "2013-03-31", "2013-04-01", /*  其他日期  */ ]
        },
        "tagList": [
            { "text": " h7n9", "weight": 0.1832792736, "term_id": 17438},
            { "text": "禽流感", "weight": 1.1633168659, "term_id": 32471},
            /*  其他突发词数据  */
        ],
        "terminalData": [
            { "app": "新浪微博", "num": 1804 },
            { "app": "专业版微博  ", "num": 1125 },
            /*  其他接入终端数据  */
        ],
        "sexData": { "female": 2157, "male": 5281 },
        "provinceData": {
            "data": [
                { "name": "北京", "value": 3400 },
                { "name": "广东", "value": 890 },
                /*  其他省市自治区数据  */
            ],
            "info": "参与人数"
        }
    },
    /*  其他突发话题数据  */
]
```

图 4-10 /emerging_topics.json 接口返回的数据格式示例

在使用 Spring MVC 的具体实现中，控制器层的类需要使用@Controller 进行注解，并使用@Autowired 注解注入业务逻辑层类的实例。

- 业务逻辑层

业务逻辑层处理来自数据持久化层的原始数据，将数据按照接口格式的要求进行拼接，然后返回。

在使用 Spring MVC 的具体实现中，业务逻辑层的类需要使用@Service 进行注解，并使用@Autowired 注解注入数据持久化层类的实例。

- 数据持久化层

数据持久化层负责系统与数据库之间的操作。

由于使用了 MyBatis，只需定义数据持久化层的接口，并使用@Repository 进行注解，而具体的 SQL 语句在 XML 文件中配置。在程序启动时 MyBatis 会加载所有配置，并将 SQL 语句的配置信息初始化为一个个 MappedStatement 对象存储在内存中，供业务逻辑层使用。

由于突发话题检测子系统把最终检测出的突发话题以及一些 JSON 格式的统计信息存入到了突发话题数据库 Topic-detection 的 topic 表（见表 3-12）中，因此只要将它们读出即可。具体的 SQL 配置如图 4-11 所示：

```
<mapper namespace="cn.bupt.bnrc.mining.weibo.topic.repository.mysql.FrontEndMapper">
    <select id="getEmergingTopicData" resultType="map">
        select * from table_topic where date=#{date} order by topic_energy desc
    </select>
</mapper>
```

图 4-11 SQL 语句配置

系统后台的 4 层架构同样也符合 MVC 的设计理念。MVC 由模型（Model）、视图（View）和控制器（Controller）三部分组成。每当用户和系统产生交互时，控制器（Controller）截获请求，事件处理器被触发；控制器（Controller）从模型（Model）中请求数据，并将其交给视图（View）；最后视图（View）将数据呈现给用户。在 MVC 的三个组成部分中，视图（View）和模型（Model）往往是比较独立的，而控制器（Controller）是连接两者的桥梁，它们更像是一种横向切分。

通过对比系统后台的分层架构与 MVC 设计模式，可以发现它们之间的关联为：表现层可以对应为视图（View），它包括了用户看到并与之交互的页面，以及页面中需要的一些资源。业务逻辑层与数据持久化层两层加起来构成了模型（Model），即后台提供的数据以及相应的处理逻辑。控制器层与控制器（Controller）对应，是用户与系统之间联系的纽带，接受用户的输入并调用模型和视图去完成用户的需求。

4.3.2. Web 前端展示

4.3.2.1. 前端目录结构

系统的前端的资源文件目录按照类别的不同分为如下几类：

- lib：存放使用到的库的源码
- js：JavaScript 脚本文件，负责页面的动态生成、交互和与后台服务器的通信
- css：样式表文件，负责页面结构的显示样式
- img：页面所需要的图片文件

根目录下的文件包括：

- index.jsp：网站首页，包括页面的 HTML 布局结构和与后台的 JSP 接口
- logo.ico：站点小图标

4.3.2.2. 页面功能实现

基于中文微博的突发话题检测系统只有一个页面，将检测到的突发话题以 Web 服务的方式提供给用户进行查阅。页面布局设计如图 4-12 所示：

图 4-12 页面布局

用户打开页面浏览突发话题信息的序列图如图 4-13 所示：

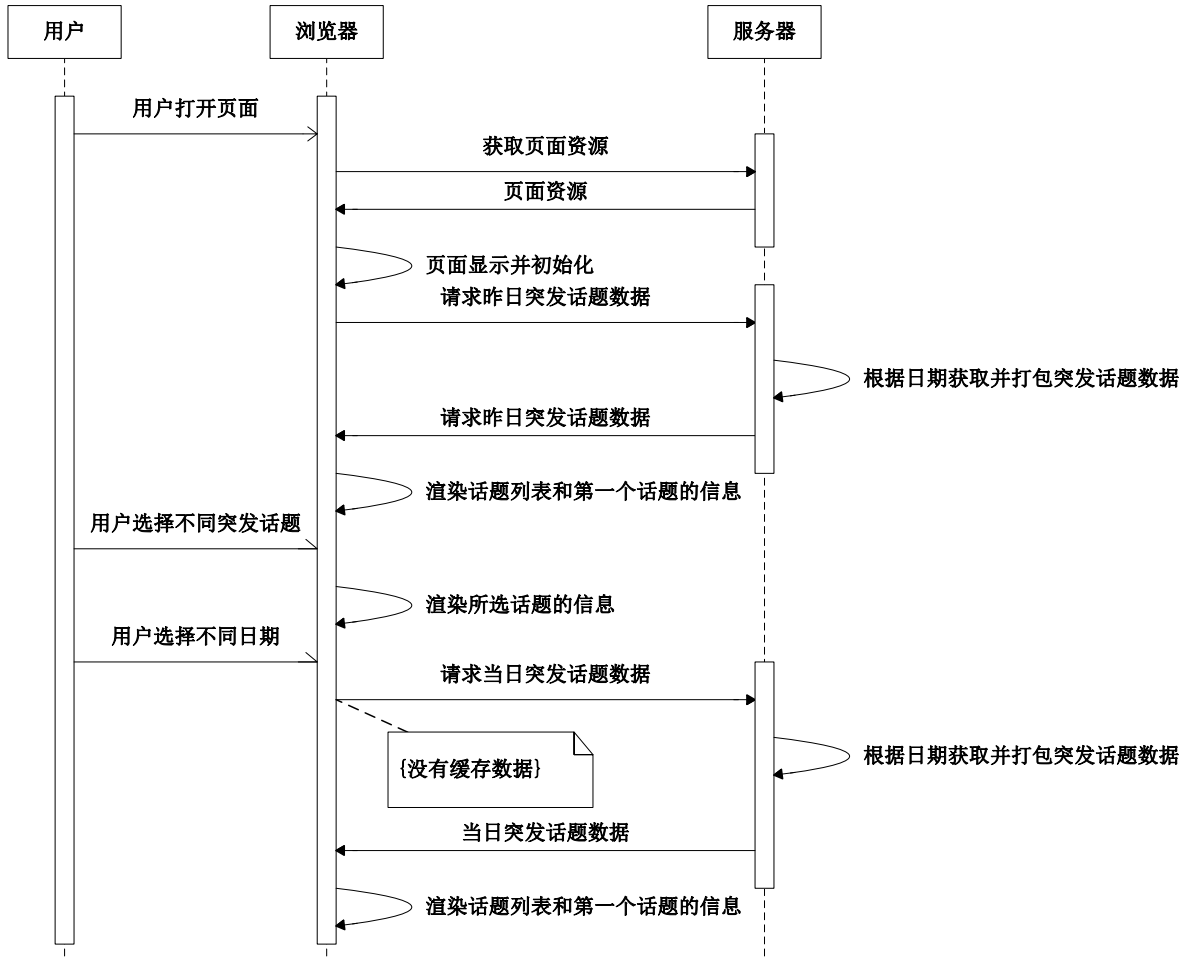

图 4-13 用户打开页面浏览突发话题信息的序列图

由于系统后台只提供了两个接口，一个返回整个页面，另一个根据日期返回这一天的突发话题数据。因此用户在浏览器中打开页面之后，页面会向后台服务器再发起一个 Ajax 请求，获取昨日的突发话题列表，并渲染突发话题列表和第一个话题的具体信息，包括：

- 由话题核心突发词组成的标签云
- 话题在最近一段时间内的热度变化趋势
- 参与突发话题讨论的所有用户在全国各省市自治区的分布
- 参与突发话题讨论的所有用户的性别比例
- 参与突发话题讨论的所有用户的接入终端分布

用户可以在这一天范围内切换显示不同的突发话题的信息。当用户选择其他日期时，浏览器会向后台服务器发起 Ajax 请求，根据所选日期获取那一天的突发话题数据。为了减少请求数量，前端会把所有通过 Ajax 获取到的数据都缓存起来。只有在没有命中缓存时，才会向后台服务器发起 Ajax 请求。

页面的 JavaScript 脚本架构采用了命名空间的管理方法，不仅保护了全局命名空间，提高稳定性；也方便多人模块化开发。根命名空间为 Weibo，它的子命名空间有：

- namespace：提供命名空间的注册管理功能
- App：提供页面的初始化与交互功能
- View：包含了页面中全部的可视化组件
- Cache：提供了对数据和视图的缓存

4.3.2.3. 可视化组件实现

除去突发话题列表组件，页面中有 5 个可视化组件，用于展示突发话题的信息，分别是地理位置分布组件、性别比例分布组件、标签云组件、话题热度趋势组件、接入终端分布组件。

为了提高组件的模块化和可重用性，各组件都采取了面向对象的实现模式，如图 4-14 所示：

```
(function(Weibo, $) {
    // 构造函数，定义组件的私有成员
    function XxxView($container, options) {
        var defaultSettings = { /* 组件默认配置参数 */ };
        this.$container = $container; // 组件在页面中要被放置到的 jQuery 包装集容器
        this.settings = $.extend(defaultSettings, options || {}); // 自定义配置参数
    }
    // 扩展组件原型，提供组件的公有成员
    $.extend(XxxView.prototype, {
        reset: function() { // 组件重置方法
            this.$container.empty();
        },
        render: function(xxxData) { // 组件渲染方法
            // 方法定义…
        },
        // 组件的其他公共成员定义
    });
    // 注册命名空间，并暴露给全局
    Weibo.namespace('Weibo.View.XxxView');
    Weibo.View.XxxView = XxxView;
}) (Weibo, $);
```

图 4-14 可视化组件的实现模式

各可视化组件的调用方法如图 4-15 所示：

```
var xxxData = { /* 来自后台的 JSON 数据 */ };
var $container = $('someSelectorExpr'); //组件在页面中要被放置到的 jQuery 包装集容器
var options = { /* 组件自定义配置参数（可选） */ };
var xxxView = new Weibo.View.XxxView($container, options); // 创建一个新的可视化组件对象
xxxView.render(xxxData);
```

图 4-15 可视化组件的使用方法

对于可视化组件的 render 方法，在具体实现时，标签云组件使用了 jQuery-UI 库提供的 jQCloud 插件，如图 4-16 所示：

图 4-16 标签云组件

话题热度趋势组件使用了 HighCharts 库提供的曲线图功能实现，如图 4-17 所示：

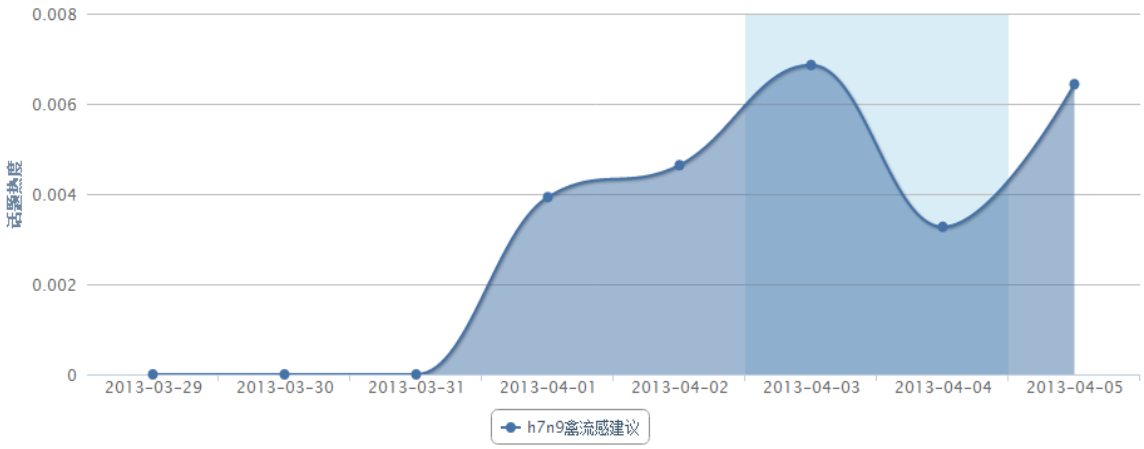

图 4-17 话题热度趋势组件

接入终端分布组件使用了 HighCharts 库提供的饼图功能实现，如图 4-18 所示：

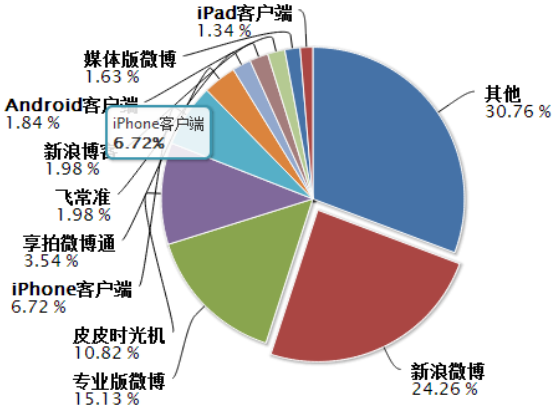

图 4-18 接入终端分布组件

而性别比例分布组件和地理位置分布组件由于图形较为特殊，需要使用 Raphaël 库提供的底层矢量图形操作 API 进行个性化定制。

性别比例分布组件如图 4-19 所示：

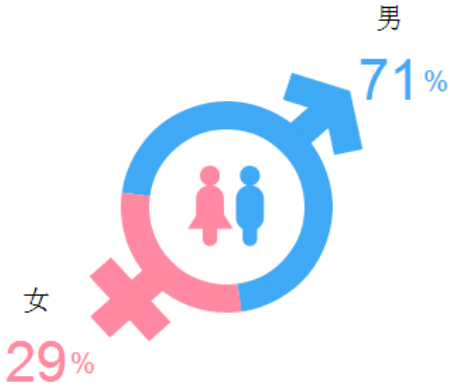

图 4-19 性别比例分布组件

地理位置分布组件如图 4-20 所示：

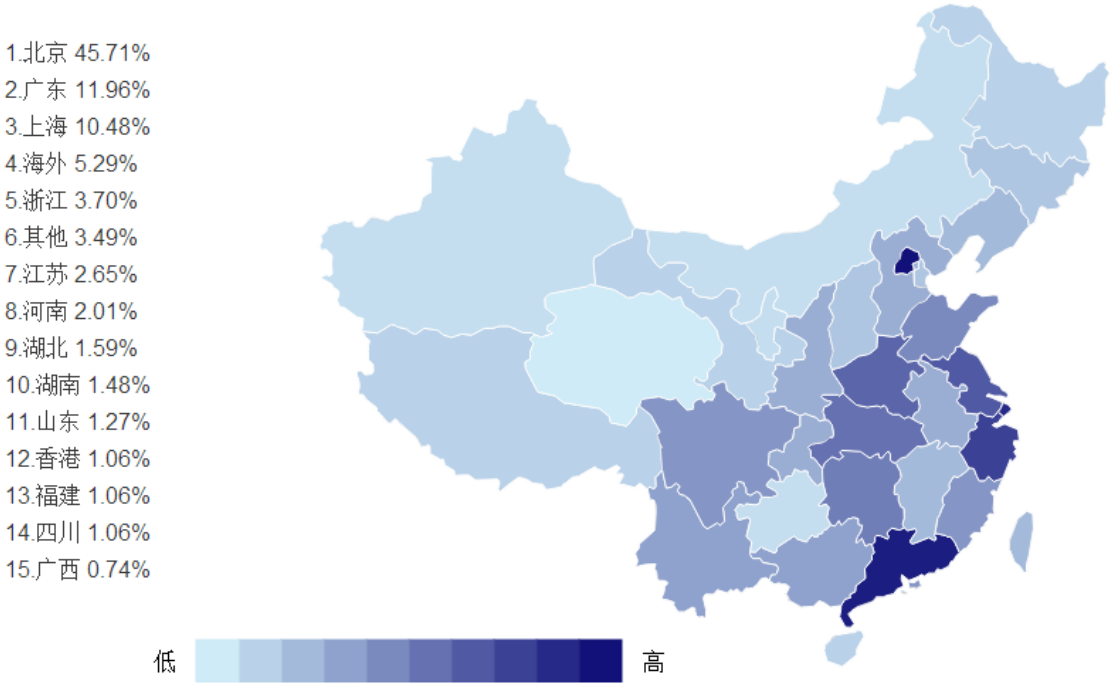

图 4-20 地理位置分布组件

在地里位置分布组建的实现过程中，渲染每个省市自治区的图形区域都特别消耗 CPU。由于浏览器引擎是单线程的，因此大量的渲染必然导致 UI 的阻塞。因此，采取了分片技术，以 5 到 10 个为一组、多组之间异步渲染，不仅保证了渲染的稳定性，同时也防止 UI 阻塞的出现。

第五章　系统测试及功能演示

本章详细描述 ETD 系统的测试工作，首先介绍系统的测试环境，然后针对各个子系统进行单元测试，接着对整个系统进行集成测试，演示并检验运行效果，最后对测试结果进行分析。

5.1. 测试环境

ETD 系统的测试环境拓扑搭建如图 5-1 所示：

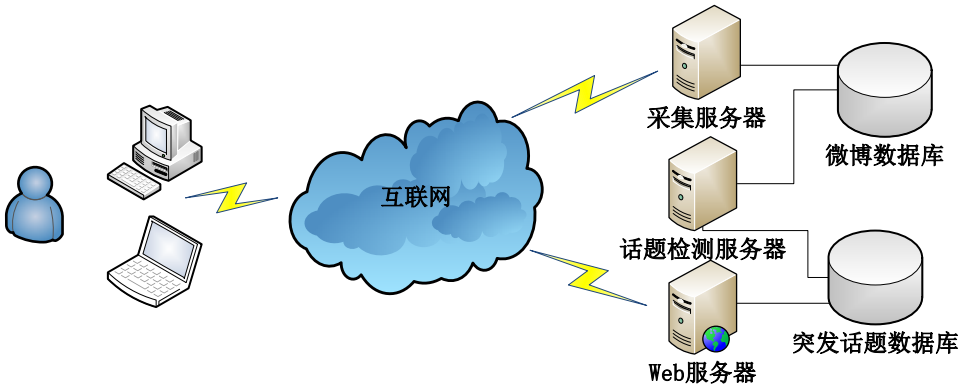

图 5-1 测试环境拓扑

其中，采集服务器、微博数据库部署在一台安装有 Windows Server 2003 系统的服务器上，服务器型号为戴尔 PowerEdge R510，数据库版本为 SQLServer 2008 R2 Enterprise。而话题检测服务器、Web 服务器、突发话题数据库部署到另一台安装有 Ubuntu 10.04 的服务器上，内核版本为 Linux 2.6.32-21-generic，数据库为 MySQL 5.1，Servlet 容器为 Jetty 8.1.12，系统的打包构建工具为 Maven 3.0.5。用户可以在 PC 端通过 Web 方式浏览突发话题检测的结果，测试涵盖的浏览器包括世界主流浏览器（Chrome、Firefox、Opera、Safari、IE 7+等）以及一些国产的壳浏览器（搜狗、360、遨游等）。

5.2. 单元测试

针对第四章中描述的各个子系统及其中的各功能模块的设计与实现，分别进行单元测试。对于后端 Java 程序，采用 JUnit 测试组件进行单元测试。对于前端程序，为了保证本系统对浏览器的兼容性，轮流使用各世界主流浏览器以及一些国产的壳浏览器进行测试；在测试之前要注意清空浏览器缓存，避免因使用旧的缓存文件而导致测试结果不符合预期。

5.2.1. 微博数据采集子系统

5.2.1.1. 采集调度模块

- 功能性测试

表 5-1 采集调度功能性测试

用例编号	测试用例	预测结果	测试结果
1-1-1	整体采集调度	首先设置种子用户，然后能够顺着用户的关注关系不断采集每个用户的个人信息、社会化关系和微博数据，存储数据库	符合
1-1-2	Access Token 和新浪 API 的使用调度	控制 Access Token 的使用、更新和新浪微博 API 的调用频率，可持续地采集数据	符合
1-1-3	重访控制	能够根据配置的参数控制重访并更新已经采集过的用户数据	符合
1-1-4	存储控制	每采集一批数据后，新建一个线程存储数据	符合

- 容错性测试

表 5-2 采集调度容错性测试

用例编号	测试用例	预测结果	测试结果
1-1-5	用户不存在	在数据库中标记此用户不存在，换下一个用户进行采集	符合
1-1-6	用户微博页数过多	在每获取一定页数的微博之后，更换 Access Token，批量存储已经采集到的微博	符合

- 稳定性测试

在采集调度模块的调度下，微博数据采集子系统能够不间断地稳定运行一个多月以上。但是在多个采集客户端并存的情况下，微博数据采集子系统的性能有所下降。

5.2.1.2. OAuth 授权认证模块

- 功能性测试

表 5-3 OAuth 授权认证功能性测试

用例编号	测试用例	预测结果	测试结果
1-2-1	批量生成 Access Token	对每个测试账号都进行 OAuth 授权认证，生成一组可用的 Access Token	符合
1-2-2	随机获取一个 Access Token	获取的 Access Token 在大多情况下能直接使用，并且每个被使用的次数比较平均	符合

批量生成 Access Token 的相关日志如图 5-2 所示：

```
INFO  2013-04-19 12:43:46,408 - [main]bnrc.weibo.crawler.util.AccessToken - 用户授权中...
INFO  2013-04-19 12:43:46,545 - [main]bnrc.weibo.crawler.util.AccessToken - 第1个access_token: 2.00n6e6qBIySZSD28ff03b4a7gQVi9B
INFO  2013-04-19 12:43:46,795 - [main]bnrc.weibo.crawler.util.AccessToken - 第2个access_token: 2.00ODc5HDIySZSD849c540b598piM4B
INFO  2013-04-19 12:43:46,893 - [main]bnrc.weibo.crawler.util.AccessToken - 第3个access_token: 2.00SnSwHDIySZSD80169079df0XPEUe
INFO  2013-04-19 12:43:46,987 - [main]bnrc.weibo.crawler.util.AccessToken - 第4个access_token: 2.00YFUEIDIySZSDaf3dec094fPRbmCC
INFO  2013-04-19 12:43:47,100 - [main]bnrc.weibo.crawler.util.AccessToken - 第5个access_token: 2.00s4oGIDIySZSDf4fe90455535HneC
INFO  2013-04-19 12:43:47,206 - [main]bnrc.weibo.crawler.util.AccessToken - 第6个access_token: 2.00C4z9IDIySZSDb9247122fdcp1qeC
INFO  2013-04-19 12:43:47,297 - [main]bnrc.weibo.crawler.util.AccessToken - 第7个access_token: 2.00ahz9IDIySZSD1cebb1d6be4B1u7C
INFO  2013-04-19 12:43:47,411 - [main]bnrc.weibo.crawler.util.AccessToken - 第8个access_token: 2.00WDATIDIySZSDe4fd034ee09yw2LE
INFO  2013-04-19 12:43:47,506 - [main]bnrc.weibo.crawler.util.AccessToken - 第9个access_token: 2.00GFq9IDIySZSD20d6f3ffe8tzjTuC
INFO  2013-04-19 12:43:47,626 - [main]bnrc.weibo.crawler.util.AccessToken - 第10个access_token: 2.009xy9IDIySZSD9778937fb9dxnVtD
```

图 5-2 生成 Access Token 的相关日志

- 容错性测试

表 5-4 OAuth 授权认证容错性测试

用例编号	测试用例	预测结果	测试结果
1-2-3	生成 Access Token 失败	试图重新再次进行 OAuth 授权认证，直到生成一个可用的 Access Token 为止	符合

- 稳定性测试

在网络连接的畅通的情况下，OAuth 授权认证模块的稳定性很大程序上取决于新浪微博 OAuth 2.0 系统的稳定性。只要对方稳定，OAuth 授权认证模块就能稳定工作。

5.2.1.3. 数据采集模块

- 功能性测试

表 5-5 数据采集功能性测试

用例编号	测试用例	预测结果	测试结果
1-3-1	采集用户个人信息	根据用户 ID 获取到其个人信息的 JSON 数据	符合
1-3-2	采集用户粉丝 ID 列表	根据用户 ID 获取到其粉丝列表 ID 的 JSON 数据	符合
1-3-3	采集用户关注 ID 列表	根据用户 ID 获取到其关注列表 ID 的 JSON 数据	符合
1-3-4	批量采集用户发布的微博	根据用户 ID 分页获取到其发布的微博的 JSON 数据	符合

图 5-3 展示了采集一个新用户（ID 为 1733106087）数据的相关日志。

```
INFO  2013-04-19 13:33:01,954 - [main]bnrc.weibo.crawler.crawl.Crawler - ★★★★★正在爬取新用户1733106087★★★★★
INFO  2013-04-19 13:33:01,954 - [main]bnrc.weibo.crawler.crawl.Crawler - 正在爬取用户1733106087的个人信息
INFO  2013-04-19 13:33:07,164 - [main]bnrc.weibo.crawler.crawl.Crawler - 正在爬取用户1733106087的社会化关系信息
INFO  2013-04-19 13:33:12,325 - [main]bnrc.weibo.crawler.crawl.Crawler - 正在爬取用户1733106087的微博信息
INFO  2013-04-19 13:33:12,325 - [main]bnrc.weibo.crawler.crawl.Crawler - 原微博数: 0, 现微博数: 521
INFO  2013-04-19 13:33:12,325 - [main]bnrc.weibo.crawler.util.WeiboInterImpl - 正在爬取微博第1页
INFO  2013-04-19 13:33:17,600 - [main]bnrc.weibo.crawler.util.WeiboInterImpl - 正在爬取微博第2页
ERROR 2013-04-19 13:33:21,147 - [Thread-2830]bnrc.weibo.crawler.database.DBOperation - 插入用户用户社会化关系错误，重新插入!
INFO  2013-04-19 13:33:22,806 - [main]bnrc.weibo.crawler.util.WeiboInterImpl - 正在爬取微博第3页
ERROR 2013-04-19 13:33:24,402 - [Thread-2829]bnrc.weibo.crawler.database.DBOperation - 插入用户id错误，重新插入!
INFO  2013-04-19 13:33:28,106 - [main]bnrc.weibo.crawler.util.WeiboInterImpl - 正在爬取微博第4页
INFO  2013-04-19 13:33:33,312 - [main]bnrc.weibo.crawler.util.WeiboInterImpl - 正在爬取微博第5页
INFO  2013-04-19 13:33:38,663 - [main]bnrc.weibo.crawler.util.WeiboInterImpl - 正在爬取微博第6页
```

图 5-3 采集新用户数据过程中产生的相关日志

在不断采集新用户数据的同时，也会定期重新访问那些已经采集过了用户数据，以

便保持本地数据集整体的新鲜度（Freshness）。图 5-4 展示了重访更新一个旧用户（ID 为 1664613160）数据的相关日志：

```
INFO  2013-04-19 13:21:03,545 - [main]bnrc.weibo.crawler.crawl.Crawler - ☆☆☆☆☆正在重访用户1664613160，已访问1次☆☆☆☆☆
INFO  2013-04-19 13:21:03,545 - [main]bnrc.weibo.crawler.crawl.Crawler - 正在爬取用户1664613160的个人信息
INFO  2013-04-19 13:21:08,797 - [main]bnrc.weibo.crawler.crawl.Crawler - 正在爬取用户1664613160的微博信息
INFO  2013-04-19 13:21:08,797 - [main]bnrc.weibo.crawler.crawl.Crawler - 原微博数：746，现微博数：747
INFO  2013-04-19 13:21:08,797 - [main]bnrc.weibo.crawler.util.WeiboInterImpl - 正在爬取微博第1页
```

图 5-4 重访更新用户数据过程中产生的相关日志

- 容错性测试

表 5-6 数据采集容错性测试

用例编号	测试用例	预测结果	测试结果
1-3-5	网络连接断开	采集暂停一段时间后再进行尝试	符合
1-3-6	IP 请求次数超过上限	采集暂停一段时间	符合
1-3-7	用户请求次数超过上限	更换 Access Token	符合
1-3-8	用户不存在	通知采集调度模块，换下一个用户	符合

如果遇到用户（ID 为 1822542811）不存在的情况，则更换下一个用户进行采集，如图 5-5 所示：

```
INFO  2013-04-19 18:06:02,728 - [main]bnrc.weibo.crawler.crawl.Crawler - ★★★★★正在爬取新用户1822542811★★★★★
INFO  2013-04-19 18:06:02,728 - [main]bnrc.weibo.crawler.crawl.Crawler - 正在爬取用户1822542811的个人信息
ERROR 2013-04-19 18:06:02,853 - [main]bnrc.weibo.crawler.util.WeiboInterImpl - 爬取用户信息过程中出错
ERROR 2013-04-19 18:06:02,853 - [main]bnrc.weibo.crawler.util.WeiboInterImpl - Error Code: 20003 : User does not exists!
INFO  2013-04-19 18:14:26,963 - [main]bnrc.weibo.crawler.crawl.Crawler - ★★★★★正在爬取新用户1822862863★★★★★
```

图 5-5 用户不存在的相关日志

- 稳定性测试

只要新浪微博 API 能够稳定的提供数据，数据采集模块就能够稳定运行。

5.2.1.4. 数据持久化模块

- 功能性测试

表 5-7 数据存储功能性测试

用例编号	测试用例	预测结果	测试结果
1-4-1	获取待采集的新用户	从数据库中获取待采集的新用户 ID，它被发现时间距离现在最远	符合
1-4-2	获取待采集的重访用户	从数据库中获取待采集的重访用户 ID，它发布微博频率大于某个阈值，上次更新时间小于某个时间间隔，且距离现在最远	符合
1-4-3	存储用户个人信息	存储用户个人信息，若用户个人信息已存在则进行更新	符合
1-4-4	批量存储用户 ID	批量存储用户 ID，忽略已经存在的用户 ID	符合

（续上表）

1-4-5	批量存储用户关系	批量存储用户关注关系，若已存在则更新	符合
1-4-6	批量存储微博	批量存储用户发布的微博，若已存在则更新	符合

以上面提到的 ID 为 1733106087 的用户为例，他的个人信息、关注关系以及发布的所有信息分别被存储到了微博数据库 Sinaweibo 中的用户信息表 users、用户关系表 user_relation 和微博表 statuses 中，分别如图 5-6、5-7、5-8 所示：

	user_id	screen_name	name	province	city	location	description
1	1733106087	关注北京出租车拒载	关注北京出租车拒载	11	1	北京 东城区	关注北京出租车拒载现象，为北京市和谐出行贡献微薄的力

图 5-6 用户 1733106087 的个人信息

source_user_id	target_user_id	relation_state	iteration
1733106087	1055483684	1	0
1733106087	1074857631	1	0
1733106087	1110091354	1	0
1733106087	1110277741	1	0
1733106087	1193848022	1	0
1733106087	1195031270	1	0
1733106087	1195432503	1	0
1733106087	1240186262	1	0
1733106087	1247325967	1	0
1733106087	1248740475	1	0

图 5-7 用户 1733106087 的部分社会化关注关系

	status_id	user_id	created_at	mid	content
1	3504418970383928	1733106087	2012-10-23 22:20:54.000	3504418970383928	【北京好司机】不过照片放仪表盘上可不太安全啊。//@六月李
2	3504417439062447	1733106087	2012-10-23 22:14:48.000	3504417439062447	出租车行驶超过15公里以后，每公里会加收50%的返程费，因此以
3	3504029088860186	1733106087	2012-10-22 20:31:36.000	3504029088860186	现场实拍国贸桥下拉活儿的黑车，及等候中不打表的正规出租，
4	3503990819699588	1733106087	2012-10-22 17:59:33.000	3503990819699588	【出租好司机】京B-P3400的司机师傅帮助博友及时抵达北京台，
5	3503990362483411	1733106087	2012-10-22 17:57:46.000	3503990362483411	下车索取发票是为了遗失物品时联系出租公司和司机的好习惯。
6	3503988496135035	1733106087	2012-10-22 17:50:19.000	3503988496135035	请反复拨打交通委24小时便民电话68351150或68351570投诉。并
7	3503985081966190	1733106087	2012-10-22 17:36:47.000	3503985081966190	京BQ2684 请遗失黑色触屏手机的朋友们注意以下信息。@平安北
8	3503719217613756	1733106087	2012-10-22 00:00:19.000	3503719217613756	鼓励多投诉，有些事儿不能只发发牢骚就过去的。 //@鲁尔Edwa
9	3503718907353807	1733106087	2012-10-21 23:59:06.000	3503718907353807	北京南站需要出租车~@交通北京

图 5-8 用户 1733106087 发布的部分微博

- 容错性测试

表 5-8 数据存储容错性测试

用例编号	测试用例	预测结果	测试结果
1-4-7	没有获取到待重访的用户 ID	通知采集调度模块，转而去获取下一个待采集的新用户 ID	符合
1-4-8	存储数据失败	不断尝试存储，直到存储成功为止	符合

- 稳定性测试

当只有一个采集客户端运行时，数据持久化模块能够非常稳定地运行。再增加一个采集客户端时，稳定性也能够得到保证。但是当使用 3 个以上采集客户端同时读写数据库时，稳定性会降低。

5.2.2. 突发话题检测子系统

以2013年4月20日的数据为例，进行突发话题检测子系统中各个模块的单元测试。

5.2.2.1. 数据清洗模块

- 功能性测试

表 5-9 数据清洗功能性测试

用例编号	测试用例	预测结果	测试结果
2-1-1	微博信息质量计算	能够正确计算每条微博的信息质量	符合
2-1-2	用户的信息传播能力计算	能够正确计算每个用户的信息传播能力	符合
2-1-3	用户活跃度计算	能够正确计算每个用户的活跃度	符合
2-1-4	用户的影响力计算	能够正确计算每个用户的影响力	符合
2-1-5	微博的相对显著度计算	能够正确计算每条微博的相对显著度	符合
2-1-6	微博的传播价值计算	能够正确计算每条微博的传播价值	符合

- 容错性测试

表 5-10 数据清洗容错性测试

用例编号	测试用例	预测结果	测试结果
2-1-7	计算微博的相对显著度分母为0	能够正确计算微博的相对显著度	符合
2-1-8	计算用户的影响力分母为0	能够正确计算用户的影响力	符合

- 稳定性测试

数据清洗模块的计算量较大，尤其是计算用户影响力时使用到了类似 PageRank 的算法，消耗了大量内存，稳定性还不够高。需要增加 JVM 分配的堆栈大小。

采集到的在 2013 年 4 月 20 日这一天发布的微博总数有 12,2682 条，主题分散，并且其中绝大部分都是一些噪声微博，其中一部分如图 5-9 所示：

status_id	user_id	content
3568954024988382	1851524785	女人的心情，三分天注定，七分靠shopping！！
3568954024988548	1802393212	如果我放弃，不是因为我输了，而是因为我懂了。
3568954028605712	1734530730	【晚安，河南】来郑州可体验一周四季：周一卫衣，周二衬衣，周三短袖，周四马上毛衣，周五必须穿大衣，周...
3568954029026363	2316925143	生活中有很多的感悟，我们想到，但却说不出来！而阿狸真的说到心 坎里了
3568954029184275	2395641294	哈哈 笑喷了！这神父太油菜了！[花心]
3568954029376053	2166429772	全球首家丰胸秘籍专题博，教你如何安全健康的丰胸！
3568954029376539	1668595852	你要么找一个能让你过上好日子的男人，要么找一个你甘心情愿为他吃苦的男人。
3568954029376587	2522328274	[泪]终于有人开通阿狸类微 博了，每天分享说到心 坎里的话，好喜 欢！

图 5-9 2013 年 4 月 20 日发布的部分微博

通过对微博进行数据清洗，从几十万条微博中筛选出了 6190 条具有较高传播价值的微博，其中一部分如图 5-10 所示：

status_id	user_id	content	vop
3569126935162802	1904228041	#地震快讯#中国地震台网正式测定：04月20日11时40分在四川省雅安市天全县、芦山县交界（北纬30.1度，东经...	2.839603E-08
3569126972756352	1720962692	【最新数字：死亡超过37人！】据央视消息，雅安市市委书记徐孟加介绍，死亡超过37人，伤者超过600人。另...	1.599266E-07
3569127010509860	2715357317	#关注雅安7级地震#新浪籍州呼吁网友们将微博头像更换成绿丝带，一起为震区人民祈福！[绿丝带]	4.018766E-06
3569127073985197	1411201145	关注雅安地震：震中芦山县，目前公布7级。又是龙门山脉！记得汶川救灾时，曾走过雅安-芦山-宝兴，去小金、...	1.770712E-06
3569127073985482	2803301701	【雅安地震已致41人遇难】人民日报记者王明峰消息，截至目前，通过无线电台了解，芦山县死亡28人、伤307...	1.279254E-06
3569127089974611	1663937380	#雅安地震#【四川雅安地震已致28人死亡 伤亡超500人】据新华社成都4月20日消息，记者从前方指挥部了解到...	1.709289E-07
3569127094169379	1726375767	易宝支付即将启动网络捐款平台，请大家关注支持。	3.868878E-07
3569127161286790	2386024584	11.35分 雅安地震灾情汇总信息 地震速报客户端安卓下载地址 官网下载 http://t.cn/zT2y0Qg 微盘下载 http://t.cn/z...	1.930045E-05
3569127232599474	1757353251	完整版#地震自救互救知识#>>>包涵了自救互救，止血搬运等简单易操作的知识。前往灾区的记者、专业或非专...	2.279025E-08

图 5-10 经过数据清洗之后保留下的 2013 年 4 月 20 日部分微博

5.2.2.2. 突发词检测模块

- 功能性测试

表 5-11 突发词检测功能性测试

用例编号	测试用例	预测结果	测试结果
2-2-1	分词	能够把一条微博就变成一组由词汇以及它的词性组成的集合	符合
2-2-2	预处理	根据词性和停用词进行过滤，并计算词频和归一化词频	符合
2-2-3	基于类别的词汇权重	能够正确计算每个词汇在微博中的基于类别的词汇权重	符合
2-2-4	营养值计算	能够正确计算每个词汇在时间窗口内的营养值	符合
2-2-5	能量值计算	能够正确计算每个词汇在时间窗口内的能量值	符合
2-2-6	突发词判别	能够判别时间窗口内的突发词	符合

- 容错性测试

表 5-12 突发词检测容错性测试

用例编号	测试用例	预测结果	测试结果
2-2-7	能量值计算没有获取到时间窗口外某天的营养值	忽略那天的营养值对能量值计算的影响	符合
2-2-8	计算基于类别的词汇权重时分母为 0	能够正确计算词汇在微博中的基于类别的权重大小	符合

- 稳定性测试

分词工具 ICTCLAS 的高稳定性保证了整个突发词检测模块的稳定性。由于计算量

较大，同样需要增加 JVM 分配的堆栈大小。

在 6190 条具有较高传播价值的微博中，找到了 5 个突发词，如图 5-11 所示：

term_id	word	nutrition	energy	is_emerging
222	地震	.0028444887942520	.0102198033507852	1
13906	四川	.0004513366649582	.0016215776299075	1
19014	救援	.0002474404906149	.0008890171288888	1
23690	灾区	.0005205440688260	.0009543307841097	1
74681	雅安	.0067605696450999	.0081126835480628	1

图 5-11 检测出的 2013 年 4 月 20 日的突发词

5.2.2.3. 突发词网络构建模块

- 功能性测试

表 5-13 突发词网络构建功能性测试

用例编号	测试用例	预测结果	测试结果
2-3-1	互信息计算	能够正确计算突发词之间的互信息	符合
2-3-2	突发词网络边的筛选	能够去掉那些权重（即互信息）较小的边	符合

- 容错性测试

表 5-14 突发词网络构建容错性测试

用例编号	测试用例	预测结果	测试结果
2-3-3	当前时间窗口没有突发词	不进行突发词网络构建	符合
2-3-4	计算互信息时分母为 0	能够正确计算突发词之间的互信息	符合

- 稳定性测试

在经过突发词检测后，只保留了较少的突发词进行突发词网络构建，稳定性较高。

5.2.2.4. 突发话题识别模块

- 功能性测试

表 5-15 突发话题识别功能性测试

用例编号	测试用例	预测结果	测试结果
2-4-1	相似度计算	能够正确计算突发词节点之间的相似度	符合
2-4-2	最大邻居节点发现	能够根据相似度找到节点的最大邻居节点	符合
2-4-3	图划分算法	能够根据最大邻居节点，对突发词网络进行划分，每个划分代表一个话题	符合

（续上表）

用例编号	测试用例	预测结果	测试结果
2-4-4	突发话题热度趋势统计	能够正确统计突发话题热度（即话题所包含的突发词营养值之和）趋势，结果为 JSON 格式	符合
2-4-5	突发话题参与者地理位置分布统计	能够正确统计话题参与者地理位置分布，结果为 JSON 格式	符合
2-4-6	突发话题参与者性别比例分布统计	能够正确统计话题参与者性别比例分布，结果为 JSON 格式	符合
2-4-7	突发话题参与者接入终端分布统计	能够正确统计话题参与者接入终端分布，结果为 JSON 格式	符合

- 容错性测试

表 5-16 突发话题识别容错性测试

用例编号	测试用例	预测结果	测试结果
2-4-8	当前时间窗口没有突发词	不进行突发话题识别	符合
2-4-9	当前时间窗口只有 1 个突发词	1 个突发词直接构成一个话题	符合
2-4-10	某个节点没有最大邻居节点	这个节点单独构成一个突发话题	符合

- 稳定性测试

在经过突发词检测后，只保留了较少的突发词构建了突发词网络，由于考虑到了各方面的情况，图划分算法的稳定性较高。统计功能涉及到了大量计算，稳定性还需加强。

对上面检测到的 2014 年 4 月 20 日的 5 个突发词进行网络构建之后，拓扑中的节点和边的信息分别如图 5-12 和 5-13 所示：

term_id	date	max_neighbor	topic_id	nutrition	energy
222	2013-04-20	13906	35	.0028444887942520	.0102198033507852
13906	2013-04-20	23690	35	.0004513366649582	.0016215776299075
19014	2013-04-20	23690	35	.0002474404906149	.0008890171288888
23690	2013-04-20	19014	35	.0005205440688260	.0009543307841097
74681	2013-04-20	13906	35	.0067605696450999	.0081126835480628

图 5-12 2013 年 4 月 20 日的突发词网络中的节点信息

term1_id	term2_id	date	weight	similarity
222	13906	2013-04-20	1.541746243359580	.6158581653836959
222	19014	2013-04-20	1.256998911912549	.5947566716233100
222	23690	2013-04-20	1.324244856051729	.6072157500008960
222	74681	2013-04-20	1.438937255583419	.6060327721860550
13906	19014	2013-04-20	1.351396673772260	.6134646288442260
13906	23690	2013-04-20	1.503652547690220	.6289321792266889
13906	74681	2013-04-20	1.576517342551879	.6254430507469189
19014	23690	2013-04-20	1.766939812632460	.6309423834642770
19014	74681	2013-04-20	1.362515215810979	.6078390374972030
23690	74681	2013-04-20	1.418555818553910	.6198436623851190

图 5-13 2013 年 4 月 20 日的突发词网络中的边信息

对其进行基于节点相似度的图划分算法，识别出了一个突发话题，并进行了一些简单的统计，如图 5-14 所示：

topic...	date	topic_description	topic_terms	topic_nutriti...	topic_energy
35	2013-04-20	雅安地震灾区	<TEXT>	.010824379663	.043428848793

trend	province_distribution	sex_distribution	terminal_distribution
{"trendList":[9.1196	{"data":[{"name":"北京","value":68},{"	{"female":47,"male"	[{"app":"新浪微博","num":47},

图 5-14 识别出的 2013 年 4 月 20 日的突发话题

5.2.3. 突发话题可视化子系统

由于突发话题可视化子系统为整个 ETD 系统提供了在 Web 前端的展示，因此下面只列出了这部分的测试用例，而测试结果在 5.3 集成测试与功能演示中一起展示。

5.2.3.1. Web 后台服务器

- 功能性测试

表 5-17 Web 后台服务器功能性测试

用例编号	测试用例	预测结果	测试结果
3-1-1	页面接口	能够提供突发话题检测系统的前端展示页面	符合
3-1-2	数据接口	能够根据日期提供这一天的突发话题数据，如果没有传入日期则返回昨天的突发话题数据，数据格式为 JSON	符合

- 容错性测试

表 5-18 Web 后台服务器容错性测试

用例编号	测试用例	预测结果	测试结果
3-1-3	当日没有突发话题数据	为前端返回空的 JSON 数据，由前端进行判断	符合

- 稳定性测试

由于突发话题子系统已经提前计算好了突发话题的统计结果，突发话题子系统的后台服务只做简单的数据拼接，因此在并发请求较多的情况下，稳定性仍然能够得到保证。

5.2.3.2. Web 前端展示

- 功能性测试

表 5-19 Web 前端展示功能性测试

用例编号	测试用例	预测结果	测试结果
3-2-1	突发话题列表组件渲染	能够根据后台提供的接口正确渲染突发话题列表组件	符合
3-2-2	标签云组件渲染	能够根据后台提供的接口正确渲染标签云组件	符合
3-2-3	话题热度趋势组件渲染	能够根据后台提供的接口正确渲染话题热度趋势组件	符合
3-2-4	接入终端分布组件渲染	能够根据后台提供的接口正确渲染接入终端分布组件	符合
3-2-5	地理位置分布组件渲染	能够根据后台提供的接口正确渲染地理位置分布组件	符合
3-2-6	性别比例分布组件渲染	能够根据后台提供的接口正确渲染性别比例分布组件	符合
3-2-7	用户切换突发话题列表中的话题	能够根据用户所选的话题，正确更新页面各可视化组件的内容显示	符合
3-2-8	用户选择日期	从后台获取这一天的突发话题数据，更新突发话题列表组件，同时根据第一个突发话题的数据渲染页面各可视化组件的内容	符合

- 容错性测试

表 5-20 Web 前端展示容错性测试

用例编号	测试用例	预测结果	测试结果
3-2-9	从后台获取突发话题数据失败	在页面上给出提示“获取数据错误!”	符合
3-2-10	当日没有突发话题数据	在页面上给出提示“这一天没有检测到突发话题!”	符合
3-2-11	参与某话题的用户中没有男性或女性	性别比例分布组件中不显示相应的男性或女性图标	符合

- 稳定性测试

Web 前端页面中虽然使用了非常消耗 CPU 计算资源的可缩放矢量图形技术，但是通过不断优化 JavaScript 代码，很好地提高了前端脚本运行的性能，页面即使是运行在 IE 7 这种老旧的浏览器上也能保持较高的稳定性。

5.3. 集成测试与功能演示

系统的集成测试验证各子系统之间的协调工作情况。

ETD 系统在启动时，随机选取了 10 个种子用户，以便尽可能保证采集到的数据片段能较为全面地反映由不同背景的用户所组成的社会化网络的真实情况。然后微博数据采集子系统开始对新浪微博的用户个人信息、用户社会化关系和微博数据进行不间断地采集。在每天凌晨的 0 点时，突发话题子系统启动，获取前一天发布的所有微博进行突发话题检测，并对检测到的突发话题做一些简单的统计。最后突发话题可视化子系统会通过 Web 的方式向用户展现这些突发话题信息。

用户打开浏览器，在地址栏中输入 http://10.108.107.173:8080/weibo-mining/emerging，浏览器会默认显示前一天的突发话题，如图 5-15 所示：

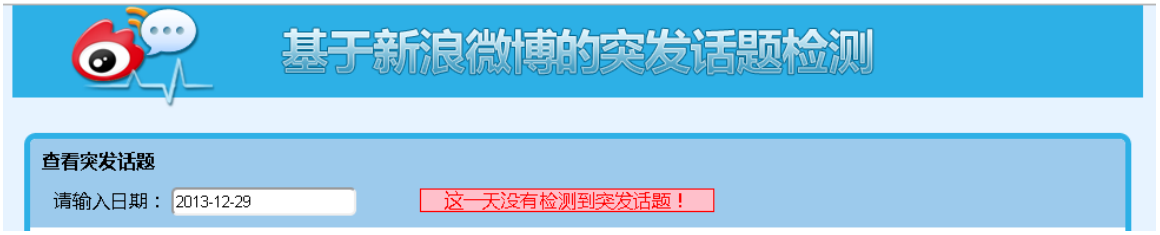

图 5-15 2013 年 12 月 29 日的突发话题展示

这一天没有检测出突发话题，系统会显示相应的提示信息。2013 年比较重要的突发事件有 4 月 20 日的四川雅安地震，可以选择那天查看一下系统检测出的突发话题。单击日期选择框，在弹出的日期控件中选择 2013 年 4 月 20 日，如图 5-16 所示：

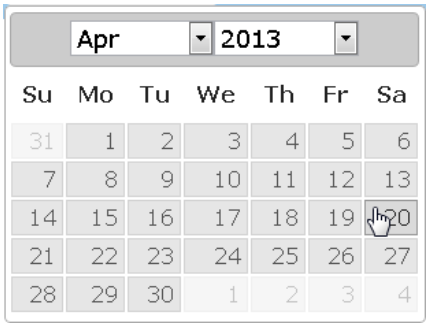

图 5-16 日期选择

这一天检测出的突发话题只有 1 个——“雅安地震灾区”，如图 5-17 所示：

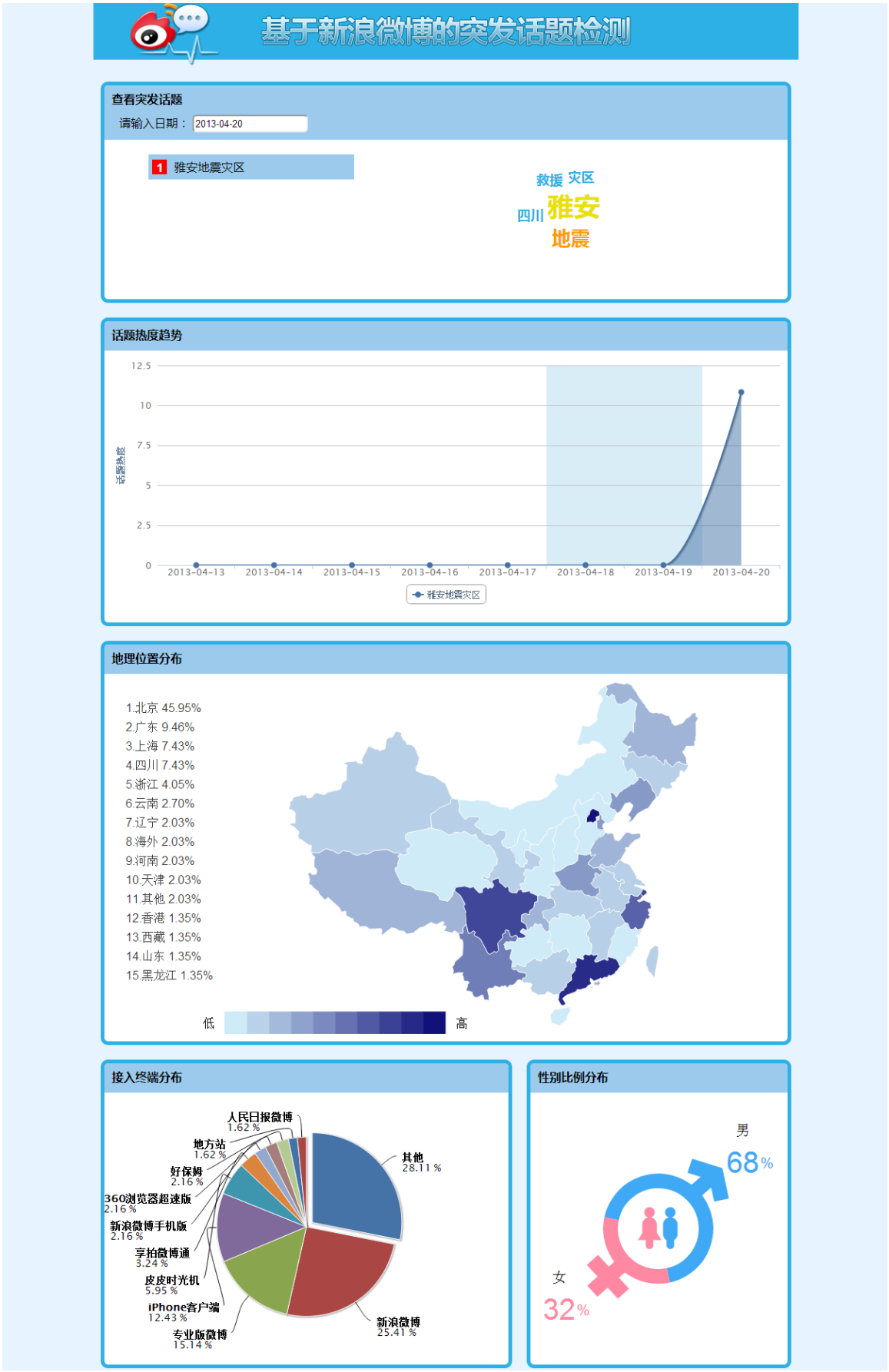

图 5-17 2013 年 4 月 20 日的突发话题展示

话题“雅安地震灾区”的核心关键词为雅安、地震、灾区、四川、救援。页面中还展示了关于这个突发话题的一些统计信息的可视化展现，包括话题热度趋势、参与者的地理位置分布、接入终端分布和性别比例分布。可见，由于地震的突发性，话题热度从4月20日开始暴涨，而之前几天却没有任何相关的讨论。在用户方面，来自北上广和四川的用户对此突发话题的讨论更加热烈，用户的接入终端以Web、桌面应用为主，男性比女性更加关注此类公共突发事件。

2013年4月20日只有一个突发话题。如果在日期选择框中选择2013年4月9日，则显示这一天检测出了多个话题，单击页面左侧话题列表中的不同话题，可以切换显示不同话题的可视化信息，如图5-18所示（统计信息略）：

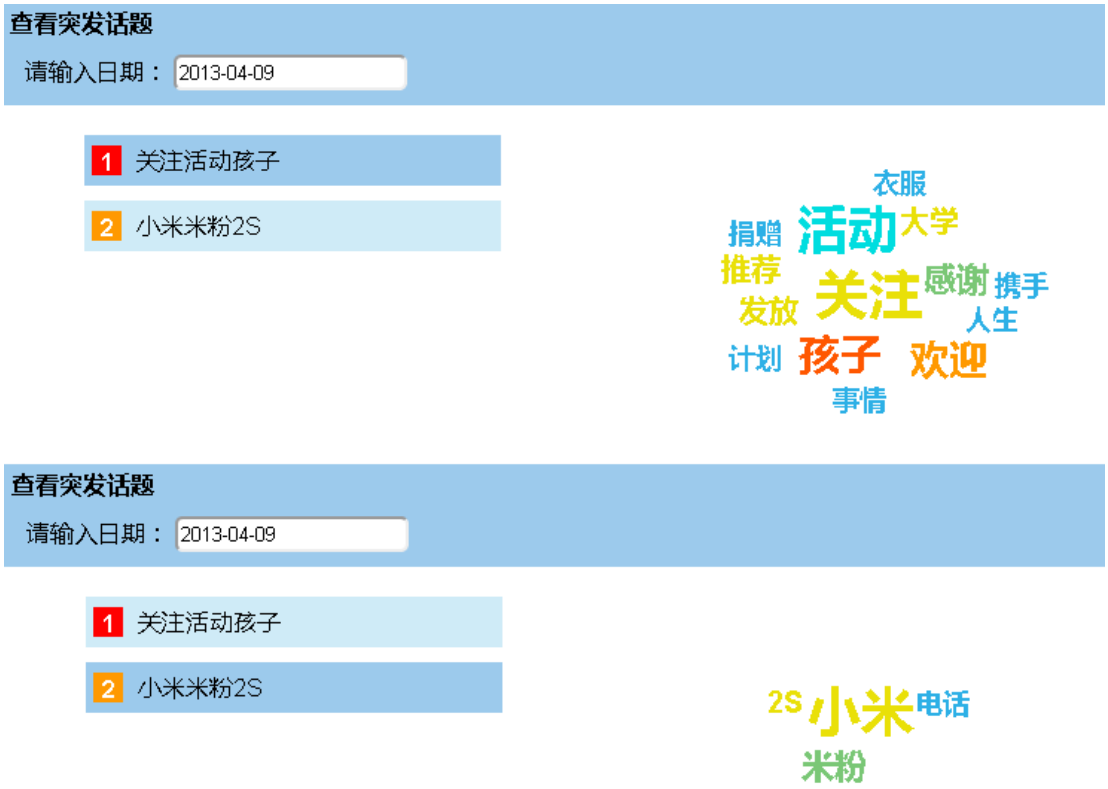

图5-18 话题切换显示

5.4. 测试结果分析

通过对各个子系统的每个模块的单元测试，可以看出ETD系统中的各个子系统已经基本实现了需求分析中所提出的需求。而通过对系统进行集成测试，说明ETD系统已经可以在服务器端进行部署，并投入到线上供用户使用。

下面以2013年4月检测出的部分突发话题（如图5-19所示）做测试结果分析。

2013-04-03	禽流感h7n9
2013-04-05	h7n9禽流感建议
2013-04-06	日本博鳌朝鲜
2013-04-07	论坛博鳌亚洲
2013-04-08	首相英国娘子
2013-04-08	撒切尔夫人
2013-04-08	中国安全技术
2013-04-09	小米米粉2S
2013-04-09	关注活动孩子
2013-04-11	地址投票发起
2013-04-11	中国公司企业
2013-04-11	上海机会获得
2013-04-12	歌手歌王
2013-04-12	快乐生日
2013-04-13	感染病例确诊
2013-04-14	古城凤凰
2013-04-14	成功购买活动
2013-04-14	工作北京上海
2013-04-15	中国胡耀邦耀邦
2013-04-15	活动获得机会
2013-04-16	活动复旦获得
2013-04-16	马拉松比赛
2013-04-16	波士顿爆炸中国
2013-04-17	工作信息招聘
2013-04-17	中国市场问题
2013-04-17	使用移动合作
2013-04-18	邮箱停止
2013-04-18	手机中兴计划
2013-04-18	中国关注用户
2013-04-19	上海主题管家
2013-04-19	时代影响力美国
2013-04-19	中国最强音媒体
2013-04-20	雅安地震灾区

图 5-19 2013 年 4 月检测出的部分突发话题

从 2013 年 4 月份检测到的部分突发话题可以看出，ETD 系统已经具备了基本的话题检测功能，能够较为准确地从采集到的新浪微博数据中检测出一些突发话题。但是，目前 ETD 系统只是提供了一个基于中文微博的突发话题检测的基本框架，模型中的很多算法都有继续改进的空间：

在数据采集方面，采集到的数据片段还不够完整，尚不能全面反应出新浪微博整个系统的特征，例如 4 月份比较火爆的“中国星跳跃”就没有被系统检测出来。所以，系统需要有更多的用户数据积累。

在数据清洗方面，如果只想得到关于公共事件的微博，目前仅筛选具有较高传播价值的微博显然是不够的，例如上面的 ID 为 19 的话题“成功购买活动”和 ID 为 20 的话题“工作北京上海”就不是我们想要找的突发话题。可见，有必要引入分类机制来获得更好的数据清洗结果。

在突发词检测方面，有时会漏掉一些词，造成突发话题关键词的不完整，例如 ID 为 29 的话题“邮箱停止”，就少了关键词“雅虎”。所以，需要有更好的突发词检测算法。

在突发词网络构建和突发话题识别方面，偶尔会出现一个话题被错误划分为两个的问题，例如 ID 为 24 的话题“马拉松比赛”和 ID 为 25 的话题“波士顿爆炸中国”都指的是波士顿爆炸案；偶尔也会出现两个话题没有划分开的情形，例如 ID 为 17 的话题“感染病例确诊”中含有突发词“科比”。因此，基于节点相似度的图划分算法还有待改进。

第六章　结束语

6.1. 全文总结

本论文设计并实现了一个基于中文微博的突发话题检测系统 ETD。整个系统实时地采集来自新浪微博的用户和微博的相关数据，以一天为时间窗口进行突发话题检测，用户可以通过浏览器查看到每天从微博数据中检测到的突发话题数据。

本论文的创新之处主要有两点：

1、使用了一种新颖的的突发话题检测模型。定义和引入了一种微博传播价值评价指标，同时兼顾了微博在用户发布的所有微博中的相对显著度和用户影响力对于微博传播产生的影响，避免了海量噪声数据带来的话题漂移问题；使用了一种基于演化理论（Aging Theory）的突发词检测模型，与爆发模型（Bursty Model）相比，它能够更合理地揭示某个词汇随着话题的兴衰而表现出来的突发度变化趋势；基于互信息（Mutual Information）模型，借鉴社群发现算法，采用了一种基于节点相似度的图划分模型，在不需要事先指定聚类个数的前提下能够实现话题的有效聚合。比起其他模型或简单的统计和人为标注，它能够更准确地将微博中的突发话题检测出来。

2、能够对新浪微博数据进行实时的采集和突发话题检测，并使用可视化的方法展现。使用新浪微博提供的 API，实时获取最新的新浪微博内容。以用户为单位顺着用户之间的关注关系进行广度遍历，采取增量采集调度策略，提高了数据片段的完整性与一致性。将突发话题检测模型与数据采集进行实时的对接，实时地检测出最新的突发话题，而非仅仅是做离线分析。最后通过使用一些最新的可缩放矢量图形技术，以图形、图表等方式对检测出的突发话题信息在 Web 前端进行展示，为用户提供了良好的动态呈现效果和交互效果。

论文首先对突发话题检测的相关理论和技术背景以及前后端开发技术进行介绍；之后详细描述了基于中文微博的突发话题检测系统 ETD 的需求分析，并对整个系统进行了总体设计；接下来描述了系统内部各个子系统的详细设计与实现；最后，通过单元测试和集成测试对系统进行了验证和分析，表明整个系统的运行结果达到了预期的设计目标。

6.2. 未来工作展望

在未来的工作中，还需要从以下几个方面来改进本文中实现的 ETD 系统：

- 扩展对新浪微博进行数据采集的规模，并解决由此带来的来自于采集、计算、

存储等方面的瓶颈。

- 实现在使用新浪微博 API 之外的备用数据采集方案。
- 对于突发话题检测，采用开源软件 Hadoop 对采集到的海量微博进行分布式处理，提高整个系统的处理性能和稳定性。
- 改进突发话题检测模型，使得数据清洗、突发词检测、突发词网络构建、突发话题识别等各个算法的表现更优。

6.3. 研究生期间工作

在研究生期间，论文作者参与的工作主要有：

一、参加企业合作项目“A Content-aware Social Search System”

- 负责系统的设计页面布局和脚本架构
- 负责注册、登录、问题、首页、设置等页面的基本呈现和交互功能
- 负责系统上线后的推广和运营

二、参加企业合作项目“A Content-aware Mobile Social Search System”

- 负责 4 人项目团队的协调工作
- 参与前期的技术调研和可行性分析
- 参与系统的需求分析
- 参与系统总体架构的设计和详细设计
- 负责系统后台数据采集模块设计与实现
- 负责系统的部署和版本控制工作

三、参加北京邮电大学青年科研创新计划专项“移动互联网实时交互应用的社会化行为分析研究”

- 负责 3 人项目团队的协调工作
- 参与前期的技术调研和可行性分析
- 参与系统的需求分析
- 参与系统总体架构的设计和详细设计
- 负责新浪微博数据的采集工作
- 负责突发话题检测核心算法的实现与优化
- 负责搭建突发话题检测可视化展示服务的前端与后台
- 负责系统的部署和测试
- 负责系统的部署和版本控制工作

四、参加企业合作项目“社会化网络数据挖掘技术服务项目”

- 参与社交网络和可穿戴式设备领域的专利调研
- 负责专利技术交底书的撰写

五、学术输出

- 以第一作者身份向 EI 国际会议 CPSCom 发表学术论文一篇，题目是《Leading Users Detecting Model in Professional Community Question Answering Services.》，EI 收录，检索号：20140717310439
- 以第一作者身份向 EI 国际期刊 JCIS 发表学术论文一篇，题目是《Modeling Leading Users in Professional Social-Network Based Community Question Answering Services》，已录用，待 2014 年 4 月份出版，待 EI 检索
- 以第三作者（学生排名第二）身份向 SCI 国内期刊中国通信发表学术论文一篇，题目是《Ranking Potential Reply-Providers in Community Question Answering System》，SCI 收录，检索号：240NN

参考文献

[1] 微博介绍 http://baike.baidu.com/subview/1567099/11036874.htm

[2] Facebook 网站 http://www.facebook.com/

[3] 人人网 http://www.renren.com/

[4] 新浪微博 http://www.weibo.com/

[5] Chen C C, Chen Y T, Sun Y, et al. Life cycle modeling of news events using aging theor. Machine Learning: ECML, 2003: 47-59.

[6] 互信息模型 http://en.wikipedia.org/wiki/Mutual_information

[7] 新浪微博开放平台 http://open.weibo.com/

[8] TDT 介绍 http://wiki.mbalib.com/wiki/TDT

[9] Kleinberg J. Bursty and hierarchical structure in streams. Data Mining and Knowledge Discovery, 7(4), 2003, pp. 373-397.

[10] Chen K Y, Luesukprasert L, Chou S. Chou. S. Hot topic extraction based on timeline analysis and multidimensional sentence modeling. Knowledge and Data Engineering, 19(8), 2007, pp. 1016-1025.

[11] He D, Parker D S. Topic dynamics: an alternative model of bursts in streams of topics. In Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, pp. 443-452.

[12] Twitter 网站 http://www.twitter.com/

[13] Sayyadi H, Hurst M, Maykov A. Event Detection and Tracking in Social Streams. ICWSM, 2009.

[14] Cataldi M, Di Caro L, Schifanella C. Emerging Topic Detection on Twitter Based on Temporal and Social Terms Evaluation. In proceeding of the 10th International Workshop on Multimedia Data Mining, ACM, 2010: 4.

[15] 邱云飞, 程亮. 微博突发话题检测方法研究. 计算机工程, 38(9), 2012.

[16] 郑斐然, 苗夺谦, 张志飞等. 一种中文微博新闻话题检测的方法. 计算机科学, 39(1), 2012, pp.138-141.

[17] 新浪微博的“微话题”页面 http://huati.weibo.com/

[18] 腾讯微博的热门话题页面 http://k.t.qq.com/p/topic/

[19] Craig Walls. Spring 实战（第 3 版）. 人民邮电出版社, 2013

[20] MyBatis 介绍 http://en.wikipedia.org/wiki/MyBatis

[21] jQuery 介绍 http://en.wikipedia.org/wiki/JQuery

[22] HighCharts 官方网站 http://www.highcharts.com/

[23] Raphaël 官方网站 http://raphaeljs.com/

[24] Ajax 介绍 http://en.wikipedia.org/wiki/Ajax_(programming)

[25] JSON 介绍 http://www.json.org/json-zh.html

[26] ICTCLAS 分词工具官网 http://ictclas.nlpir.org/

致 谢

时光荏苒，岁月如梭。两年半的研究生时光即将结束。在此，我要感谢所有在这段时间里给予过我指导和帮助的人。

我首先要感谢我的导师王文东教授。在硕士研究生生涯期间，无论是在学习方面还是生活方面，王老师都给予了我很大的帮助。王老师虽然工作繁忙，但仍然经常指导我的工作，与我探讨学术问题，他严谨的治学态度和认真的工作风格给我留下了深刻的印象，在此谨向王老师致以最真挚的感谢。

同时，我还要感谢龚向阳教授和阙喜戎副教授，他们在科研项目上给了我很多的支持和帮助，对我的项目工作和论文写作提出了宝贵的建议和意见，在这里我要向龚老师和阙老师表示由衷的感谢。

感谢田野师兄、郭亮师姐和韩闻文师姐，他们在学术研究方面对我的帮助和启发很大，在此向他们表示感谢。

感谢所有项目组的同学，特别是黄水桂、刘跃杰等同学。我们在项目开发和学术研究中团结一致、密切配合，顺利地完成了课题任务。从他们身上我学到了很多，并且也结下了深厚的友谊。

此外，我还要感谢我的家人，特别是我的父母，他们的关心和鼓励是我一生中最宝贵的财富，没有他们的支持，就没有我今天的成绩。

最后，非常感谢诸位专家教授们在百忙之中评审我的学位论文，并给予指导。

作者攻读学位期间发表的学术论文目录

[1] Siqi Song, Ye Tian, Wenwen Han, Xirong Que, Wendong Wang. Leading Users Detecting Model in Professional Community Question Answering Services. IEEE Cyber, Physical and Social Computing (CPSCom), 2013, pp.1302-1307.（EI 收录，第一作者，检索号：20140717310439，论文署名单位为北京邮电大学）

[2] Siqi Song, Wendong Wang, Xirong Que, Ye Tian, Wenwen Han. Modeling Leading Users in Professional Social-Network Based Community Question Answering Services. Journal of Computational Information Systems (JCIS), vol. 10, no. 8.（EI 收录，第一作者，录用待发表，论文署名单位为北京邮电大学）

[3] Wenwen Han, Xirong Que, Siqi Song, Ye Tian, Wendong Wang. Ranking Potential Reply-Providers in Community Question Answering System. China Communications 2013, vol. 10, no. 10, pp. 125-136.（SCI 收录，第三作者，检索号：240NN，论文署名单位为北京邮电大学）