摘要

过去十年来，微博已经全世界范围内最流行的社交方式。在智能手机的帮助下，现在人们可以随时随地对他们身边发生的事发布微博，这带来了数据量的高速增长，而在这些数据中隐藏着一些有价值的知识和信息。近年来对从微博数据中自动检测热门事件的研究越来越多，然而各种算法产生的结果多为干瘪的数据，文字等，缺少一个直观的展示平台。本项目完成了一个用于展现事件检测算法的 Web 应用，通过使用树图，时间线，地图，模拟力的有向图等可视化技术来呈现算法结果。此外，可视化模块还提供了管理界面以让研究者导入数据。

关键词 可视化 事件检测 微博 twitter

Abstract

Social event detection has become one of the hottest research topics with the enormous popularity of social websites. However, most research produce an algorithm with merely textual outcome. A visualization system for event detection and monitoring algorithm is in demand. In this project, diverse representations are used to show the details of the events, including forms of treemap, timeline, evolution graph and map. Besides, this project also provide manage page for researchers to import data.

Keywords event detection monitoring visualization twitter

目录

第1章 项目背景

1.1 项目背景及意义

自从 2006 年 Twitter (Twitter)[1] 公司成立以来，微博（Microblog) 逐渐在世界范围内流行起来，与传统的博客不同的是，微博通常有字数限制，例如 Twitter 和新浪微博都限制每条微博的字数不能超过 140 字。因此用户无法利用微博发表长篇大论，更多的是发布一些简短的见闻或者心情感受。除了文字以外，微博还支持在发布时附上图片或者视频。

现如今，随着智能手机的快速发展以及网络的普及，用户可以随时随地发布微博。在这样的情况下，微博的数量以极快的速度在增加，并且研究发现，这些微博中有许多都与短期内我们日常生活中发生的各种事件有关。这就使得微博成为了一个重要的获取实时新闻的来源。事实上越来越多的年轻人获取新闻的来源不是来自传统的电视，报纸等渠道，而是来自于以微博为代表的社交网络。但是由于微博的信息量太大，当发生某个热门事件时，与其相关的信息可能分布在不同时间段或者不同的交友圈，为了能从全局掌握这些热门事件的信息，近年来提出了话题检测与跟踪（Topic Detection and Tracking）的技术。通过 TDT [2]技术，人们可以从一个更高的角度发现事件的来龙去脉以及事件之间的联系。

从大量的微博数据中找出有用的信息并不是容易，由于微博的“简短”、“口语化”等特性，传统的自然语义处理（NLP）方法无法直接应用到微博数据上，现有的检测微博事件的算法多基于聚类算法。

本项目是澳大利亚昆士兰大学（The University of Queensland，以下简称 UQ）信息科技和电子工程学院数据和知识工程（Data & Knowledge Engineering，以下简称 DKE) 实验室的一个子项目。DKE 实验室正在做一项关于微博事件发现与跟踪的算法研究，本项目起辅助作用，以期达到完成一个集抓取数据，时间检测及跟踪，信息可视化为一体的线上应用的目的。

1.2 项目目标和任务

1.2.1 项目目标

本项目的目标是完成一个基于 B/S 架构的 Web 应用，来对实验室基于 Twitter 的事件检测算法进行可视化展示。该 Web 应用需要从数据库读入算法结果，然后通过 Web 客户端将事件的详细信息，发展情况以可视化技术展现出来，使得用户能直观而生动的了解到指定时间段内的热门事件及发展趋势。由于运行算法的机器与 Web 服务器不是同一台，本项目还需要开发一个管理页面，以将算法产生的文本结果导入到数据库中。该项目需要具备以下特性：

- **高效性**

  事件检测及跟踪需要基于大量的微博的数据，否则无法排除偶然性因素，从而导致算法准确率的降低。为了能及时处理大量的微博数据，要求本项目具备较高的处理效率。

- **可扩展性**

  本项目是为 DKE 实验室的事件检测及跟踪算法的研究而服务，使得算法产生结果不再是干瘪的数字或文本，而是内容丰富而生动形象的图表及动画。考虑到将来算法研究带来的结果的必然变动，要求本项目具备较高的可扩展性。

除了可视化部分，本项目还包括一个事件检测算法的实现，已达到对比作用。但是在可视化部分使用的是实验室一名博士研究生的成果。

1.2.2 任务分解

为了完成以上目标，整个项目可以分成以下两个个任务：

- **算法实现模块**

  前文已经提到，本项目是为实验室事件检测及跟踪算法的研究而服务，算法实现模块将来自于实验室一名博士研究生的科研成果。但是本项目仍然需要实现一个事件检测算法，以用于与实验室成果做对比。该检测算法来自于论文《Towards Effective Event Detection, Tracking and Summarization on Microblog Data》[3]。由于是对论文进行对比，因此需要保证程序编写严格按照论文描述的方法来实现，不能改变论文中描述的算法，否则无法起到对比的作用。

- **可视化模块**

  根据数据特性，设计可视化方案，并基于 Web 将其实现。此模块要求使用图、表、文字等方式尽可能完整而生动地展现事件相关信息。同时，考虑到要呈现的数据量较大，且直接与用户打交道，该模块需要较高的运行效率。

- **开发文档**

  由于毕业之后就要离开 UQ，为了后续接手人员的维护及改进方便，需要编写一份完整的文档，涵盖以上三个模块。这一步也是保证可扩展性的重要步骤。

第2章 项目实施方案

2.1 项目基本流程

项目开发基本流程如下图所示：

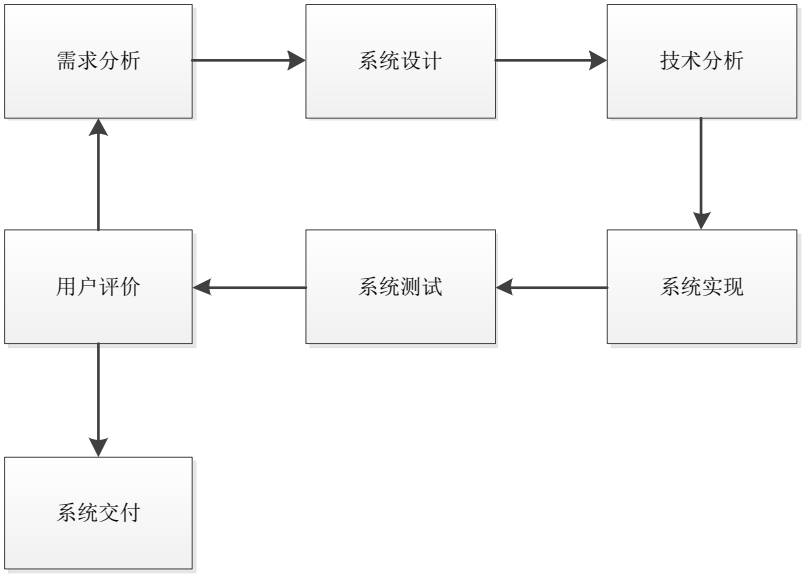

图 2-1 项目基本流程

项目开发基本遵循增量模型，用户，即 UQ 的导师提出需求需要对实验室正在进行的微博事件检测算法研究做一个配套的可视化平台，然而需求并不十分明确。因此，增量模型比较适合本项目的开发。经过需求分析，系统设计，技术分析，系统实现，系统测试等步骤之后，交由用户对系统进行评价并提出改进意见，然后重复上述步骤，直到用户满意为止。

2.2 项目开发计划

本项目从 2013 年 8 月开始，2014 年 5 月基本完成，项目开始时做出的开发计划如下表所示：

时间段	任务
13 年 8 月-9 月	阅读相关论文
13 年 10 月-11 月	完成算法实现模块
13 年 11 月-14 年 1 月	学习网站开发技术， 学习 D3 库的使用
14 年 1 月-3 月	设计可视化方案 完成前台开发
14 年 3 月-4 月	根据需要修改数据抓取模块 完成后台开发 撰写开题报告
14 年 4 月-5 月	部署上线 撰写期中报告
14 年 5 月-6 月	撰写论文，准备答辩

表 2-1 项目进度表

2.3 可行性分析

2.3.1 技术分析

- **算法实现模块**

《Towards Effective Event Detection, Tracking and Summarization on Microblog Data》这篇论文中使用基于聚类的文本分析法来检测事件，其算法并不复杂，可以认为是一个文本处理的程序，完成并不困难，难点是效率能否达到要求。算法中有一步是构建同存矩阵，朴素的算法复杂度达到$O(T^2 * N)$，其中T表示热门词汇的数量，N表示当天微博的数量。而如果用到信息检索领域里的倒排索引并且用哈希来查询关键字，可以将复杂度降低到$O(K^2 + K * N)$。倒排索引这一步可以通过基于 Java 的开源搜索引擎 Lucene 来完成。因此这一模块技术上也可以到达要求，是可行的。

- **可视化模块**

  基于 Web 的可视化模块需要用到许多图表绘制，如果自己从最基础的 Javascript 层开始编写需要花费大量时间在编写、调试与优化上，但是好在开源社区内有不少基于 web 的可视化库，经过仔细比对及分析各个库的异同点、优劣势以及特性，最终决定使用 D3(Data-Driven Documents)[4]库。网站建设使用经典的 WAMP （Windows Apache MySQL PHP)， D3 库示例众多，资料齐全，因此本项目使用的技术都较为成熟，理论上是可行的。

2.3.2 环境和条件分析

通过前期了解分析，本项目对硬件条件要求并不高，DKE 实验室提供一台服务器用于项目的部署，已经足够使用。而项目中较为关键的事件检测及跟踪算法由 DKE 实验室一名科研经验丰富的博士研究生负责完成，得到的事件信息应该较为可靠准确。而本项目实施者在本科期间曾参与多个工程项目，有一定的开发经验。且本项目自 2013 年 8 月就启动，至今一直按照当时制定的进度表稳步推进，时间上也不是问题。

综上所述，本项目可以在规定时间内完成预期任务。

2.4 关键技术分析

本项目主要用到以下两种关键技术，分别分布在算法实现模块以及可视化模块。

1. Lucene 和倒排索引

Lucene[5]是由 Apache 软件基金会支持的一个开源搜索引擎项目，包括 Lucene Core 等几个子项目，Lucene Core 提供基于 Java 的索引、搜索和拼写检查等功能。本项目主要用到 Lucene Core 的索引及搜索功能。Lucene 的索引技术用到了倒排索引，倒排索引技术就是在建立索引时，将每个单词在文本中的位置记录下来，这样当需要检索某个单词时，可以直接根据索引定位到这个单词所在的文本位置。

由于 Lucene 提供的是基于全文的检索，而本项目需要的是对一条微博记录进行检索，这里可以使用虚拟化技术，让 Lucene 认为一条微博就是一个文本，

从而完成对微博索引并检索的任务。

2. Data-Driven Documents

D3 是一个用于对数据进行操作的 Javascript 库，普遍用于基于 Web 的可视化应用。D3 主要用到 SVG 及 CSS 来呈现可视化效果。不同于其他可视化库如 Processing、Raphaël、Protovis， D3 直接建立在 HTML， SVG，CSS 等 web 标准上。D3 还提供渐变效果，使得可视化效果大为增加。SVG 是可缩放向量图形（Scalable Vector Graphics）的缩写，与同为 HTML5.0 中画图两大元素的 Canvas 不同的是， SVG 可以缩放而不损失图像质量，而 Canvas 的优势在于快速渲染，但是对于可视化应用，显然前者更为重要，这也是为什么 D3 被选为本项目的开发库的一个重要原因。

第3章 在项目中负责的具体工作

3.1 整体结构

整个项目整体数据流图如下：

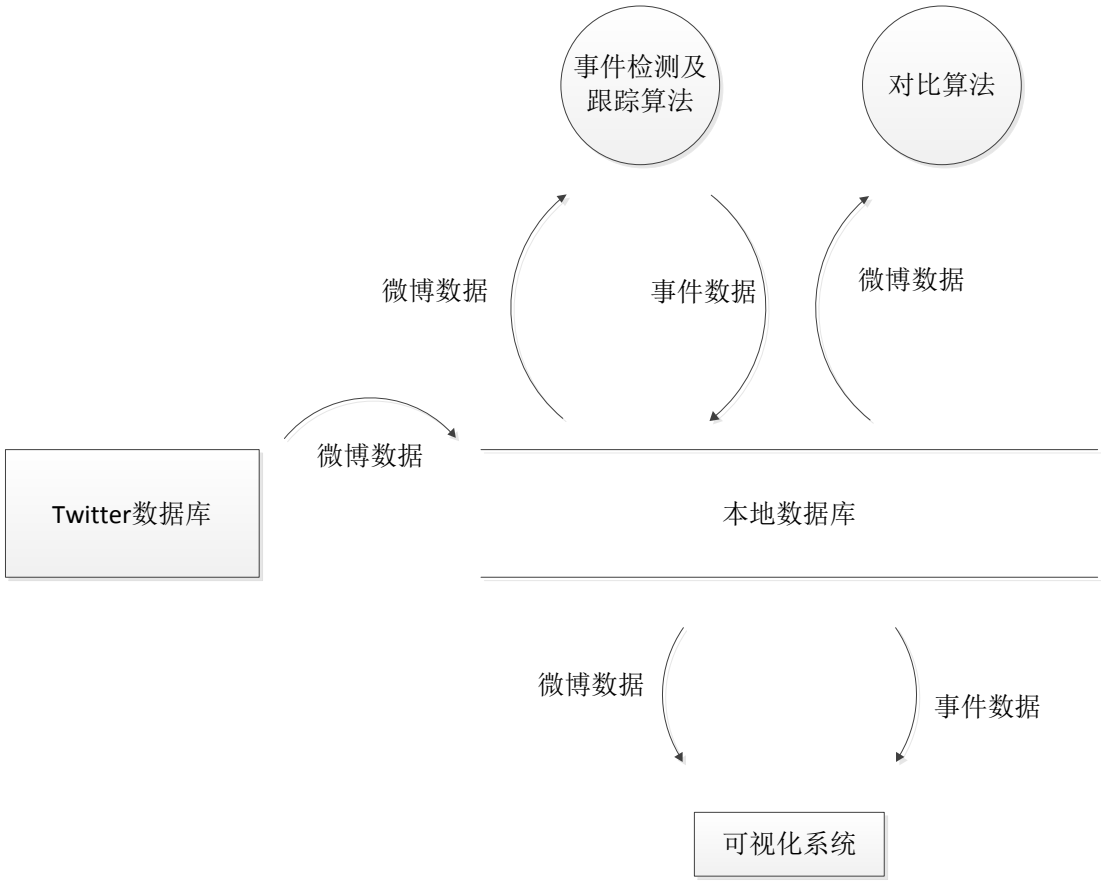

图 3-2 项目整体数据流图

其中，需要本文作者实现的部分为对比算法以及可视化系统，接下来分条叙述。

3.2 对比算法实现

对比算法采用了《Towards Effective Event Detection, Tracking and Summarization on Microblog Data》一文中描述的算法。该部分的程序主要包括三个子程序，其中一个子程序用 C++写成，负责将原始的微博数据根据论文描述

转，筛选热门词汇，计算出每个热门词汇对应的词频，并以词汇/词频的形式输出到文件。第二个程序用 Java 写成，用到了开源搜索引擎 Lucene 的 API，负责对原始的微博数据建立索引，然后读入上一步产生的热门词汇，根据论文描述计算出同存矩阵。最后一个程序是一个小工具，由于 cluto 生成的结果是以行号表示的，不直观，这个程序的目的就是将每个 cluster 中以行号表示的词汇转化成对应的字符串。

3.2.1 第一步：生成热门词汇

该部分程序由 C++写成。包含 Day，Keyword，Tweet 三个类，其 UML 图如下：

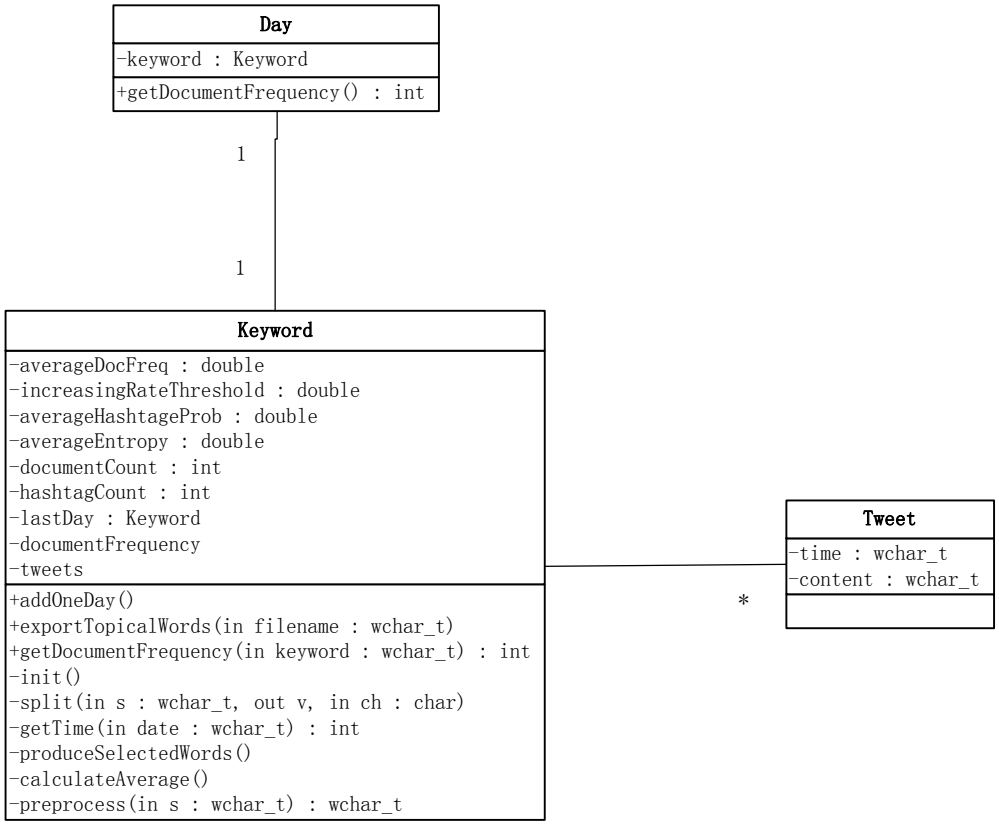

图 3-3 生成热门词汇程序 UML 图

其中 Day 类表示某一天收集到的所有微博，每一个 Day 实例包含一个 Keyword 类型的私有变量。Keyword 类负责从文件读入某一天的所有微博数据，通过论文描述的算法计算出热门词汇，并输出。Tweet 类表示一条微博，包括内容和发布时间两个信息。

程序运行流程如下图所示：

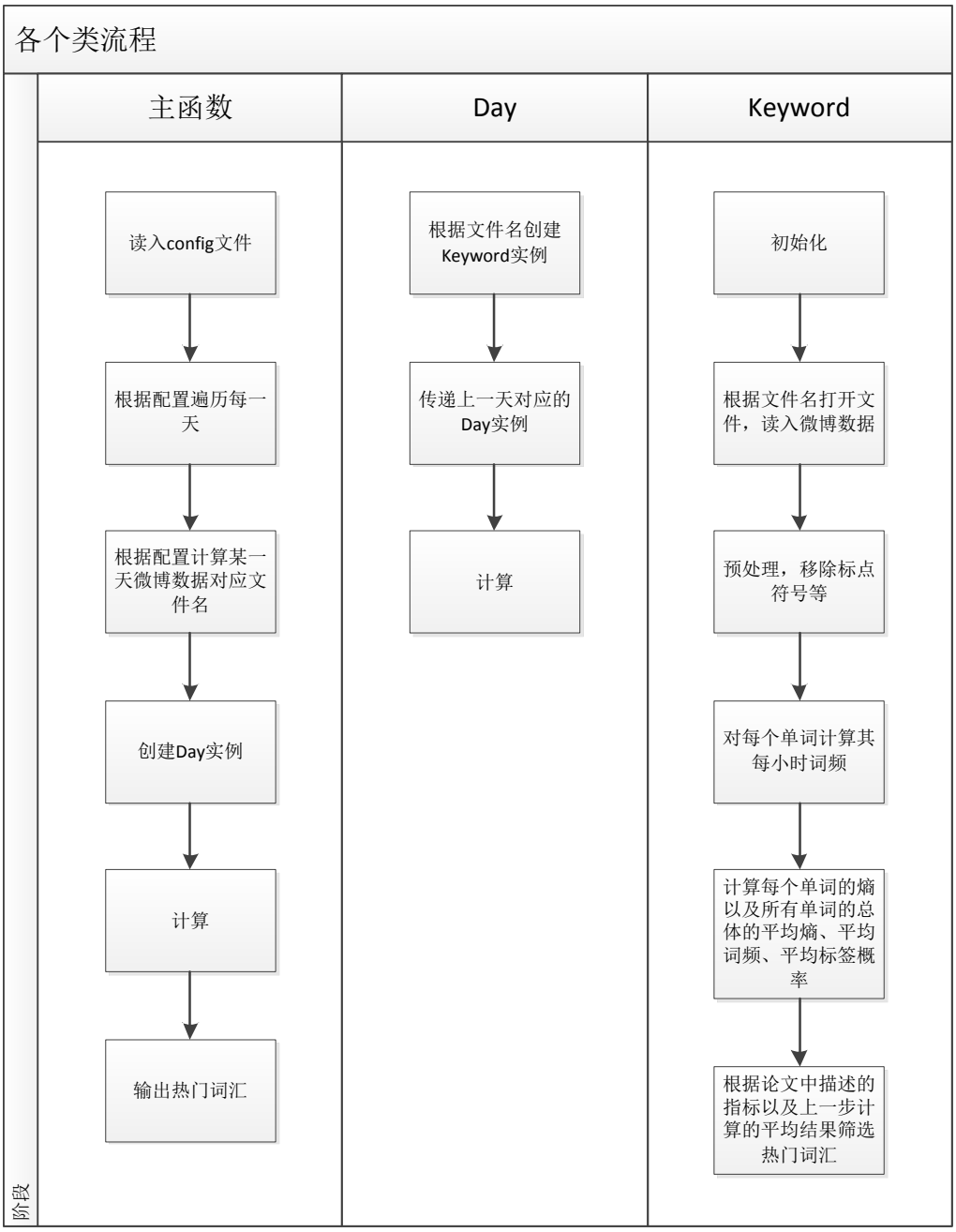

图 3-4 各个类流程图

在主函数中，首先需要打开配置文件，该配置文件需要提供的信息有：

- RootDir：数据所在目录
- StartYear：起始年份
- FinishYear：结束年份

- StartMonth：起始月
- FinishMonth：结束月
- StartDay：起始日
- FinishDay：结束日

格式形如：

StartYear = 2013

StartDay = 01

该配置文件不仅在生成热门词汇这一步需要用到，在第二步计算同存矩阵也要用到。

Keyword 类为主要计算过程发生的类。读入的微博数据经过处理之后会被存入 documentFrequency 这个变量。documentFrequency 变量类型为 string 到 double 数组的 map 映射。其中 string 为某个单词，double 数组大小为 27，分别存放该单词在这一天 24 小时中每小时的词频，当天总词频，熵，出现在“#”的次数。如下图所示：

图 3-5 每个单词需要保存的数据及位置，共 27 项

3.2.2 第二步：计算同存矩阵

该部分代码由 Java 写成，包括三个类：Configure，IndexerByLine，Searcher。它们的 UML 图如下：

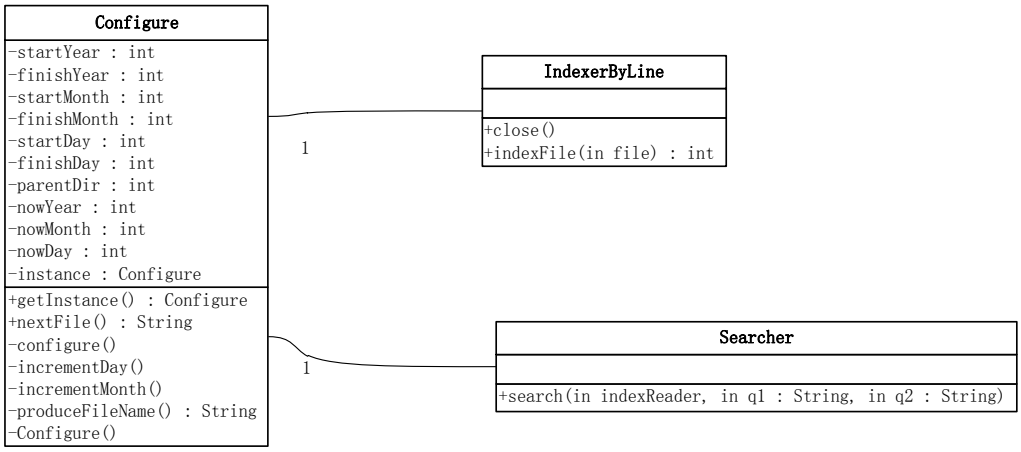

图 3-6 计算同存矩阵程序 UML 图

- Configure 类：负责读取配置文件保存配置信息。与上一小节中的 config 步骤一样，Config 类从配置文件中读入数据所在目录、起始年份、结束年份、起始月、结束月、起始日、结束日等信息。另外，类内部保存有当前日期，使用时只需调用 nextFile 方法就能不断的得到第二天微博数据对应的文件名。nextFile 方法通过调用 incrementDay 方法更新内部保存的当前日期，从而完成功能实现。
- IndexerByLine 类：负责对微博数据文件建立索引。与一般的搜索引擎应用不同的是，由于每个文件内包含多条微博，而需要建立索引的是每条微博而不是每个文件，因此需要按行来建立索引。indexFile 方法读入文件，并调用 Lucene 库 Document.add 方法格式化每一条微博，最后通过 indexWriter 输出索引文件。
- Searcher 类：负责根据索引查找给定单词。search 方法最多可以接受两个单词作为参数，并根据索引搜索出同时包含这两个单词的微博的条数。
- 主函数：主函数调用 Configure 类读入配置参数，然后对参数规定时间段内的对应文件，通过调用 IndexerByLine 建立索引，然后读入上一小节程序输出的热门词汇，调用 Searcher 类对热门词汇查找同时包含某两个热门词汇的微博数，存入其成员变量 int[][] graph 中，最后以 Cluto 可以接受的形式输出同存矩阵。

3.2.3 第三步：调用 Cluto 工具集

这一步通过调用论文中提到的 Cluto 工具集，将第二步生成的同存矩阵作为输入，得到一个表示聚类结果。

3.2.4 第四步：格式化 Cluto 输出结果

由于第三步中 Cluto 的输出结果形式为，每行一个数字 n，表示这一行对应的单词属于 id 为 n 的一个簇。这样的形式难以让人类理解，因此这一步的目的就在于将 Cluto 输出结果转化为易于理解的形式，即每行多个单词，表示这些单词同属于一个簇。

这部分程序较为简单，流程如下：

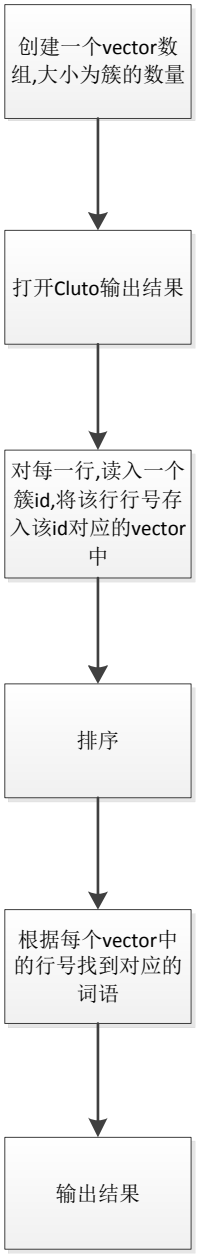

图 3-7 格式化流程

3.3 可视化模块实现

可视化模块使用基于 MVC 架构的 PHP 框架 Yii[6]实现。该模块主要包括总览，事件详情，管理等三部分，该模块业务结构如下图所示：

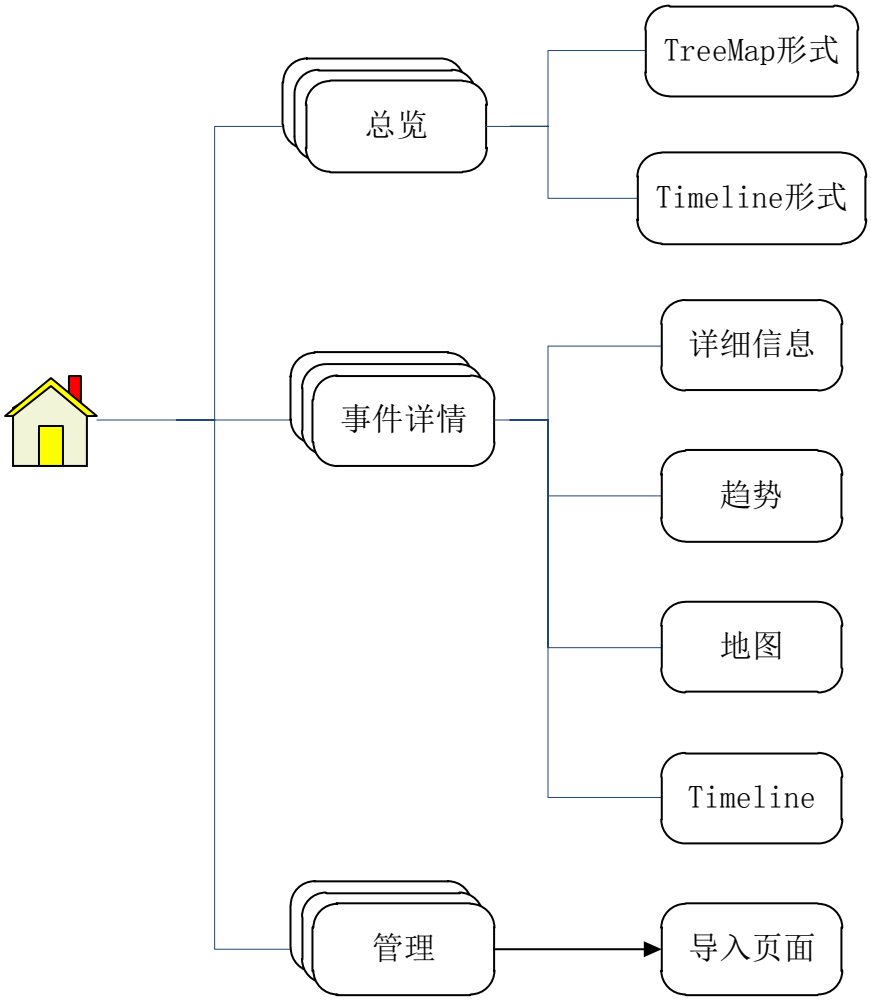

图 3-8 可视化模块总体结构

从代码角度看，其中每个页面都是一个视图，拥有与其对应的控制器。除此之外还有一个 Ajax 控制器，用于处理异步通信消息，但是 Ajax 控制器没有用于显示的 Web 页面。代码文件的组织结构如下：

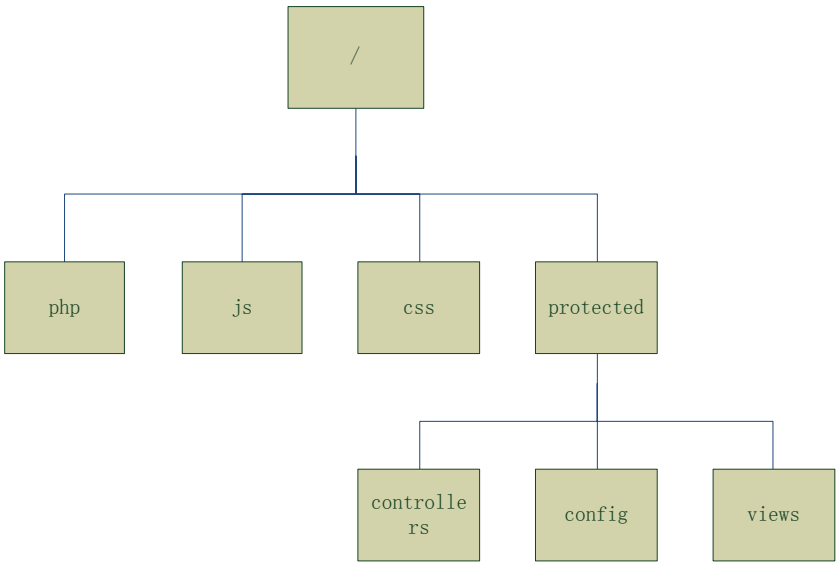

图 3-9 可视化模块代码组织结构

其中，php 文件夹存放工具类，包括 DataFormatter 类以及 Importer 类。js 文件夹存放需要用到的前端 javascript 文件，包括 bootstrap、jquery、timelinejs、d3、wordcloud 等库以及自己写的一些 js 文件。css 文件夹存放与库配套的样式文件。Protected 文件夹是 Yii 的默认结构，存放大多数系统运行相关的文件。主要用到其中的 controllers、config、views 三个文件夹，分别存放控制器、配置文件、视图。

3.3.1 数据库设计

可视化模块数据库采用 MySQL。共有以下几张主要表格:

1. events：表示一个事件的基本表，拥有字段如下：

字段名	说明	属性
Dataset_id	数据集名字	Primary Key
Event_id	属于哪个事件	Primary Key
start_time	事件开始时间	
end_time	时间结束时间	
num_of_tweets	包含的微博数量	
most_freq_word	词频最高的单词	
second_freq_word	词频次高的单词	
third_freq_word	词频第三高的单词	
summary	事件摘要	
merge_from_events	从哪些事件合并而来	
merge_to_event	合并到哪个事件	
split_into_events	分裂成哪些事件	
split_from_event	从哪个事件分裂得到	
disjoint_root_id	并查集的根	
image_url	代表事件的一张图片	

表 3-2 Events 表字段及说明

2. events_keywords：用于存储每个事件的所有关键字，拥有以下字段：

字段名	说明	属性
id	唯一性	Primary Key，自增
Dataset_id	表示数据集	
Event_id	属于哪个事件	
Keyword	关键字	
Term_frequency	词频	

表 3-3 events_keywords 表字段及说明

3. events_tweets：用于存储每个事件的所有微博，拥有字段如下

字段名	说明	属性
Id	唯一性	Primary Key，自增
Dataset_id	数据及名字	
Eventid	属于哪个事件	
Tweetid	微博的 id	

表 3-4 events_tweets 表字段及说明

4. tweets_g20_aug：基本数据表，存储所有从 Twitter 服务器抓到的数据，主要字段如下：

字段名	说明	属性
Id	唯一性	Primary Key
originalText	微博内容	
CreatedAt	创建时间	
RetweetedNb	转发数	
Country	作者国家	
Tweetid	微博唯一 id	
userScreeenName	作者名字	
IsOriginal	是否原创	
OriginalSourceTweetID	原始微博 id	
imageURL	包含的图片的地址	

表 3-5 tweets_g20_aug 主要字段及说明

5. country_name：由于 FusionTable 中的国家名与 tweets_g20_aug 中的国家名略有不同，需要这张表充当翻译。所有字段如下：

字段名	说明	属性
Id	唯一性	Primary Key，自增
Name_in_tweets	在微博表中的国家名	
Name_in_Fusiontable	在 FusionTable 中的国家名	

表 3-6 country_name 表字段及说明

3.3.2 页面细节

可视化部分，包括图 3-8 中的总览和事件详情两部分，每个页面大多遵循以下数据流流程：

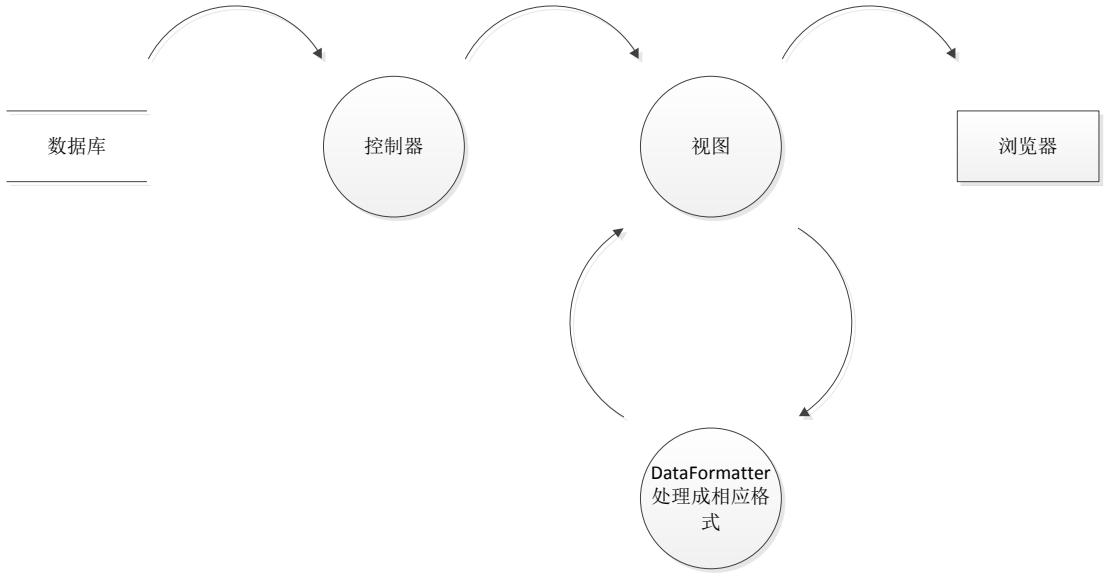

图 3-10 可视化部分数据流图

项目实施过程中并没有严格按照 MVC 的结构来进行编码，而是通过一个 DataFormatter 类来对数据库中的 raw 数据进行格式化。接下来分页面进行叙述。

3.3.2.1 TreeMap

树图（TreeMap）本来是一种对树状数据进行可视化的工具，在本项目中树图用来表示事件总览。每一个表示事件的矩形大小由其热门程度决定。

树图通过调用 D3 库 d3.layout.treemap 实现，数据由控制器从数据库中取得并交由视图来渲染。视图渲染的过程中调用 DataFormatter 类来将数据库获得的 raw 数据格式化为 json 格式，并传递给 d3.layout.treemap 进行绘制。

D3.layout.treemap 接受的 Json 格式见辅助工具小节的 DataFormatter 部分。

绘制完 TreeMap 之后，还要通过 D3 库对每个矩形添加描述事件信息的文字以及热度。然后还要对文字添加悬浮提示效果，使得用户将鼠标移动到文字上后获得更加详细的事件信息。

由于 D3 的 API 限制，在 TreeMap 所在的 DIV 元素中添加一个悬浮提示框会导致提示框被矩形截断，从而无法看清悬浮窗内容的问题。为了解决这个问题，本页面在 TreeMap 所在的 DIV 外创造一个悬浮提示框，当鼠标移动到事件文字信息上时，取得计算鼠标位置以及当前矩形的大小、位置，根据这些信息重新计算悬浮提示框的位置，并填充该矩形表示事件的内容。

3.3.2.2 Timeline

Timeline 在本模块中有两个地方会被用到：

1. 用于表示事件总览。即所有事件都会呈现在时间轴上。
2. 用于表示某个事件及其相关事件。即只有选中事件和与它相关的事件会呈现在时间轴上。

Timeline 的实现用到了 TimelineJS[7]。TimelineJS 是一个方便好用的基于 Javascript 的开源 timeline 实现。同样，数据从数据库经由控制器传递到视图，然后通过调用 DataFormatter 格式化为 Json 格式，Json 格式见 TimelineJS 的文档。

3.3.2.3 详细信息

该页面用于呈现某个事件包括的微博、该事件经过算法计算出来的关键词汇、以及微博中包含的图片三类信息。页面布局分为左右两部分，左边为

微博，右边上半部分是关键词组成的词云，下半部分是图片。这样布局是考虑到微博数量比较多，如果全部上下布局，不管是微博部分在上还是微博部分在下都会导致用户无法第一眼将三部分都收入视线。

- 微博部分：该事件的微博按照转发数量降序排列。每条微博的呈现方式通过 Twitter 提供的嵌入式微博的方式实现，虽然数据抓取部分已经将微博的内容等信息全部抓取到了本地数据库，但是这样做的好处是视觉效果与 Twitter 一致，并且支持转发、收藏、喜欢、关注等功能。但是缺点是访问 Twitter 服务器会导致微博内容会略微延迟零点几秒才出现。考虑到微博数据量可能很大，但一般来说用户只关心最具代表性的几条,因此这部分显示的微博数上限为 100 条。
- 词云部分：按照词频大小令该事件的关键词汇以不同的字体大小呈现。这部分用到了 wordcloud2.js[8]，一个开源的词云实现。关键字数据从数据库经由控制器传递到视图，然后调用 DataFormatter 格式化为数组格式，传递给 wordcloud 函数。
- 图片部分：显示转发数最高的附带图片的微博的图片。由于大小限制，设置为最多显示十张图片。图片以幻灯片的形式放映，每隔一段时间会自动跳到下一张。

3.3.2.4 趋势

该页面用于呈现某个事件的发展过程，包括关联事件（拥有分裂关系或者合并关系的事件），热度发展两个部分。

- 关联事件：这部分通过 d3 的 Force-directed 图来表现选中事件及其相关事件。相关事件的定义是，如果事件 A 经由分裂得到 B，C，或者事件 A 与事件 D 合并得到事件 E,则事件 ABCDE 互为相关。并且相关性还具有传递性的特征，即如果 A 与 B 相关，B 与 C 相关，则 A 与 C 也相关。考虑到系统效率，事件的相关性的实现采用并查集的方法来实现，并将根直接存入数据库。这样显示时可以直接在常熟时间内得到所有相关事件。而需要做的是在导入数据的时候计算事件之间的依赖关系。Force-Directed 图中，每个圆表示一个事件，半径与事件热度正相关，圆与圆之间用带箭头的直线表示事件的发

展过程。直线的颜色与指出的圆颜色一致，便于分辨。

- 热度：热度图用来展现该事件起始时间段内微博发布时间的趋势。该部分的实现用到了 d3.svg.area。数据由数据库经控制器传递到视图，然后调用 DataFormatter 的 formatHotness 函数格式化为 json 格式，最后根据数据绘制成图形。

3.3.2.5 地图

地图部分用于展示该事件在不同国家都有多少人在关注，并且当用户点击某个国家时，能跳转到事件详细信息页面，但是此时只显示用户点击的国家的用户的微博，这样便于用户了解到该国用户对这个事件的看法。

这部分一开始是用 D3 以及 TopoJSON 来实现，并且做出了原型，但是后来考虑到虽然目前来说这样的实现已经足够展示所有地理信息（由于抓取到的微博只有约 5%拥有经纬信息，大部分微博的地理信息只能通过用户的简介精确到国家这一层），但是将来可能会有更为丰富的地理信息需要展示，因此最终决定将这部分换为 Google Map API[9]来实现。

大部分 Google Map API 都需要提供经纬度来确定地理位置，但是这恰恰是数据中欠缺的。这部分最终找到了 Fusion Table 来对国家的地理轮廓进行着色，颜色深度根据该国用户的活跃度变化。但是由于 Google Map API 的问题，传递颜色参数时最多只能设置四个，这导致只有四个国家有颜色。解决方案是将所有国家按照用户活跃度分为四个组，对每个组内的国家渲染同样的颜色。

Fusion Table 的国家轮廓信息来自位于 Google Doc 的表格，链接如下：

https://www.google.com/fusiontables/DataSource?dsrcid=423734

使用 Fusion Table 时，需要填入一串编码以指定使用哪张表格，上述表格的编码是 1foc3xO9DyfSIF6ofvN0kp2bxSfSeKog5FbdWdQ。

3.3.2.6 导入

导入页面用于导入新事件，考虑到研究者可能会用多种算法跑出多套数据，本项目在设计数据库时添加了 dataset_id 字段，管理员可以选择添加一个数据集，并赋予名字，然后导入数据到这个数据集中，并且可以选择可视化模块显示的数据来自于哪个数据。这样数据库中可以存在多套数据而

只显示其中一套。

导入页面可以导入事件基本信息，分裂合并关系，事件包含的微博。

由于 PHP 限制，上传文件的大小必须小于 10M。

除了通过上传的方式导入，还可以通过直接从服务器本地读入文件来导入，这样做的好处是可以通过将文件拷贝到服务器上从而使导入过程更快更方便，但是缺陷是需要相应权限。

另外导入页面还提供了导入一个事件所有相关事件的功能，即指定文件路径、数据集名字以及某个事件的 id，系统会遍历分裂合并信息并导入该事件的相关事件。

3.3.2.7 管理

管理页面用于设置使用哪个数据集，以及触发检查数据库内部统一性。

导入数据仅仅是导入基本数据，为了可视化展示，事件之间的依赖关系还需要计算，内部统一性就是这个确保这些依赖关系正确。

需要计算的内容有：

1. 以转发数最多的微博的内容填充到 events 表的 summary 字段。
2. 以转发数最多且包含图片的微博的图片地址填充到 events 表的 image_URL 字段。
3. 根据 merge_from_events，merge_to_event，split_into_events，split_from_event 字段，使用并查集算法，对每个事件计算一个根事件，并将根事件的 eventid 填充到 disjoint_root_id 字段。

3.3.3 辅助工具

3.3.3.1 Ajax

Ajax 部分用于异步通信，主要在以下几个地方用到：

1. TreeMap 中鼠标悬浮于文字上，通过 Ajax 控制器返回该事件的详细信息。
2. 趋势页面中，当用户点击代表某个事件的圆时，更新右侧微博为选中事件转发数最高的微博。

严格来说 Ajax 部分也是网站页面的其中一个，但是不同的是，在渲染网页是通过调用 renderPartial 函数而不是 render 函数，使得得到的页面

不附带头信息、脚注信息等附加信息。

3.3.3.2 DataFormatter

DataFormatter 是一个用于对数据库传来的数据进行格式化以供其他 API 调用的工具，主要包括以下几个方法：

方法名	参数内容	返回形式	输出结果用途
generateEvents	事件及其分裂合并关系	JSON	输出将用于趋势页面中 Forced-Directed 图
generateOverviewEvents	事件	JSON	输出将用于 TreeMap 显示
formatOverviewTimeline	事件	JSON	输出将用于 Timeline
formatHotness	时间	JSON	输出将用于热度图

表 3-7 DataFormatter 主要函数

其中，generateEvents 方法返回的 JSON 格式如下：

```
{
  "name" : "root",
  "children" : [
    {
      "name" : event_id,
      "children" : [
        {
          "size" : popularity,
          "eid" : event_id,
          "gid" : group_id,
          "name" : top_3_keywords,
          "start_time" : date
        }
      ]
    }
  ]
}
```

3.3.3.3 Importer

Importer 是用来导入数据以及确保数据内部统一性的类，包括以下几个方法：

方法名	功能
importEventsTweets	导入事件包含的微博
importMergeSplit	导入合并分裂关系
importEvent	导入事件基本信息
makeInternallyConsistent	确保内部数据统一
importSeriesEvents	指定一个事件,导入所有与其相关的事件及微博

表 3-8 Importer 方法及功能

如下图所示，事件之间有直接关系也有间接关系，importSeriesEvents 方法中有个 layer_threshold 参数，用于指定寻找相关事件时遍历的层数。layer_threshold 越高则理论上会将越多的间接关联的事件也归为相关事件。事实上当不指定遍历层数时，目前数据集里的 8000 个事件会有接近 4000 个事件将被认为互为相关，因此需要指定遍历层数。

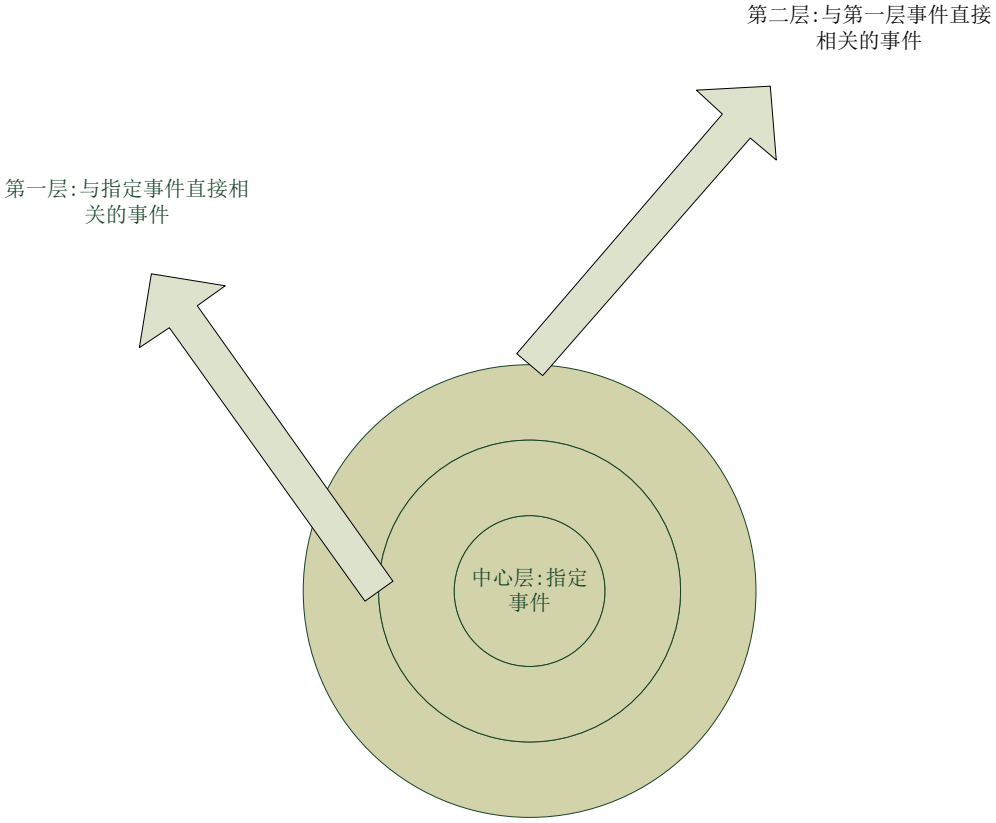

图 3-11 遍历层数为两层的遍历过程示意图

3.4 可视化模块运行截图

3.4.1 主页

首先打开主页 TopicVisualization/index.php，服务器会返回页面展示数据库里近期所有事件，如下图：

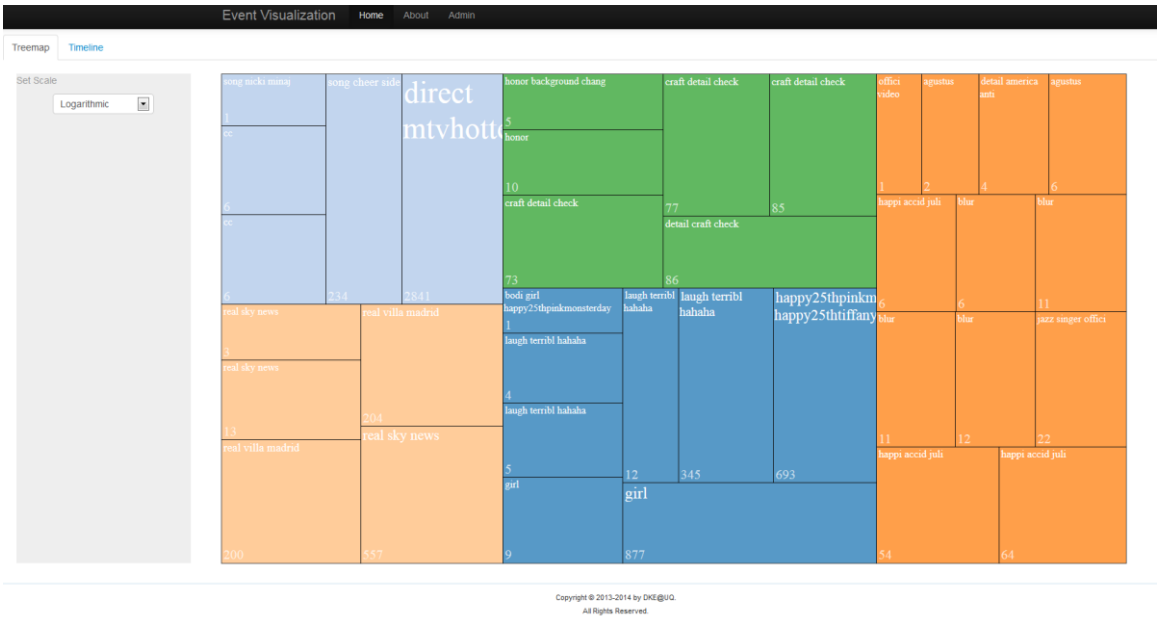

图 3-12 所有事件总览

左侧是一些控制选项，用来控制左侧 treemap 的表现形式，目前只有设置比例函数一项。默认是 log 关系，即代表每个事件的方块的大小与该事件的热门程度呈现 log 相关性。将来可以增加选择时间段的功能，即设置一个时间段，只展现该时间段内发生的时间。

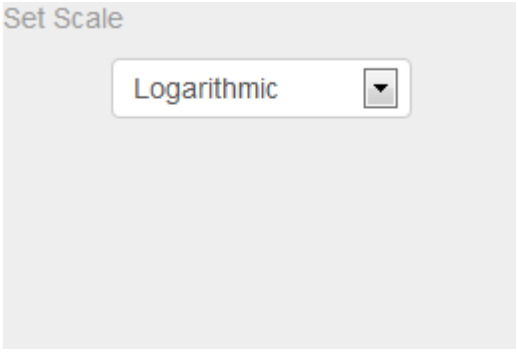

图 3-13 控制选项栏

右边是主体部分，用 treemap 来展现所有事件的简要信息，每个矩形代

表一个事件，矩形面积由该事件的热门程度以及比例函数决定。矩形的背景色按照事件所在组着色，相同组的事件拥有相同的背景。每个方块中的文字包括两部分信息：三个频率最高的关键字组成的标题以及代表热门程度的一个数字。点击标题将会跳转到该事件的详情页面。

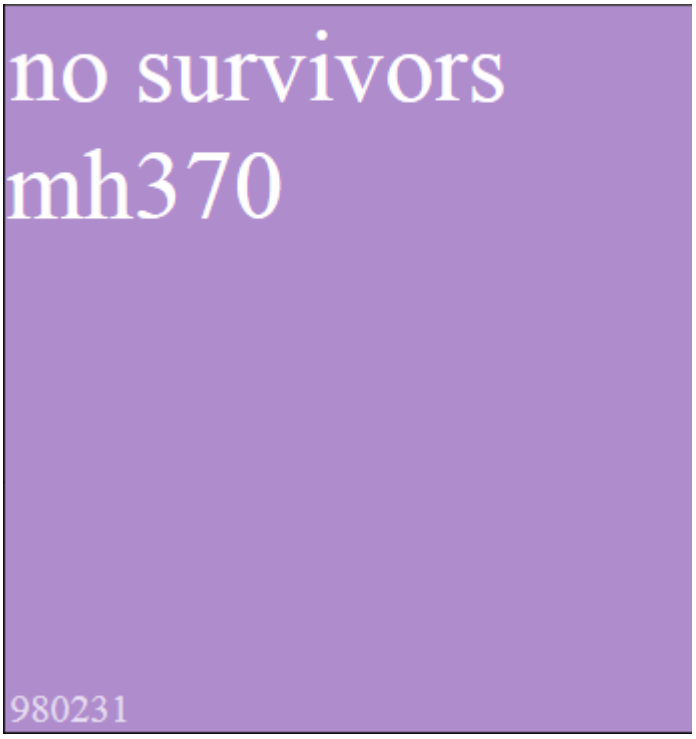

图 3-14 代表一个事件的矩形

当鼠标移动到某个矩形上方时，页面会出现一个提示框，显示该事件的标题以及摘要。由于目前的算法并没有实现摘要，摘要部分的内容将会以改事件转发数最多的 tweet 的内容代替。

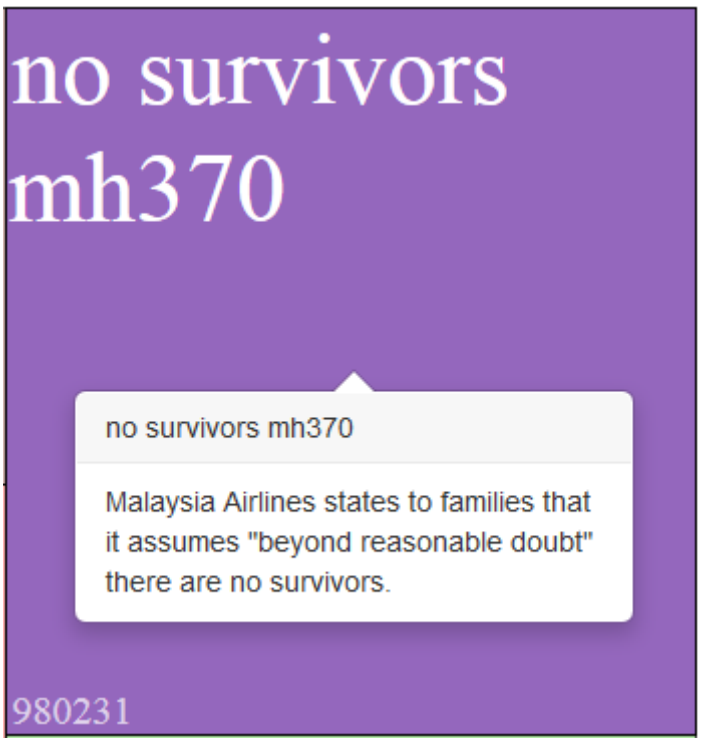

图 3-15 鼠标移动到矩形上方出现提示框

在页面的上方是一个导航栏，除了以 treemap 形式展现所有事件，还可以用 timeline 的形式展现。

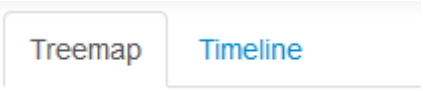

图 3-16 导航栏

3.4.2 Timeline

点击图 3-16 中的 Timeline 跳转到 TopicVisualization/index.php?r=timeline 页面：

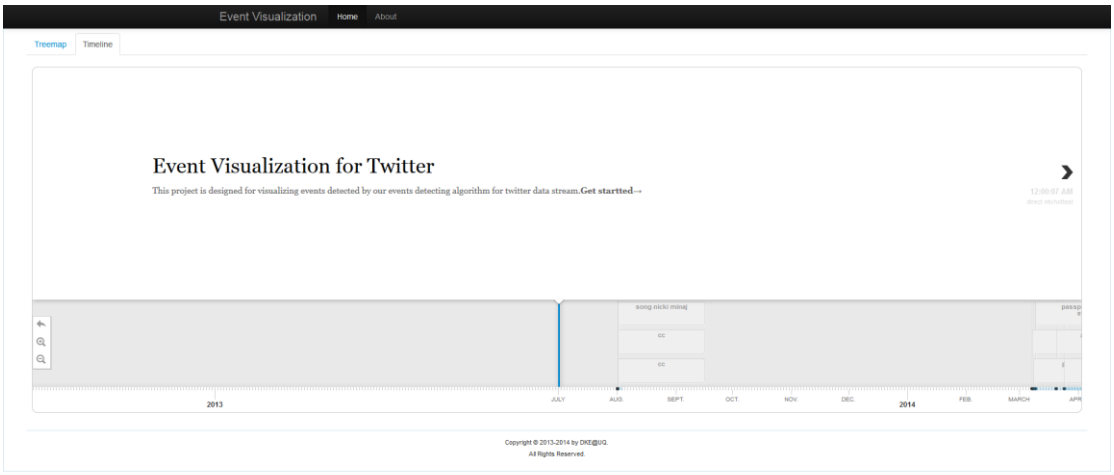

图 3-17 Timeline 界面

Timeline 界面将所有事件按照它们的起止时间排列，但是一开始显示的并不是任何一个事件，而是一个概述。点击右方的箭头开始按照时间顺序浏览事件。

下方是一个时间轴，时间轴上分布着各个事件，以三个频率最高的关键字作为标题表征。时间轴支持进行拖动，缩放等操作。

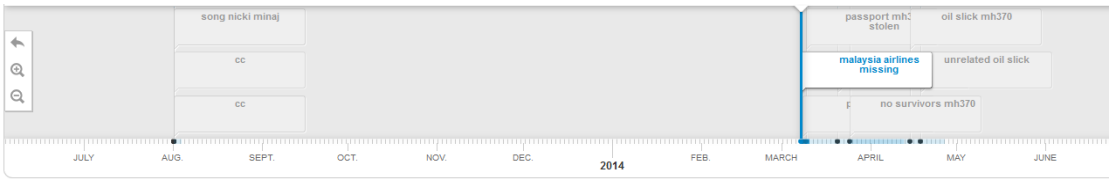

图 3-18 时间轴

点击时间轴上的事件或者点击左右侧的箭头可以查看某个事件的概述，包括标题，摘要，起止时间，多媒体（图片或视频）以及查看详情的链接。

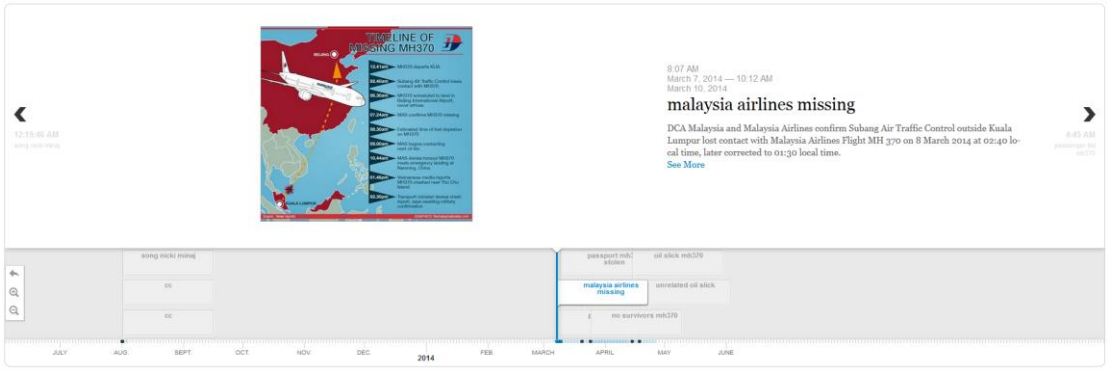

图 3-19 某个事件的概述

3.4.3 事件详细信息

点击 treemap 中的标题或者 Timeline 中的“See More”，都可以跳转到该事件的详情页面，地址形式如：

TopicVisualization/index.php?r=selectevent/event/index/252

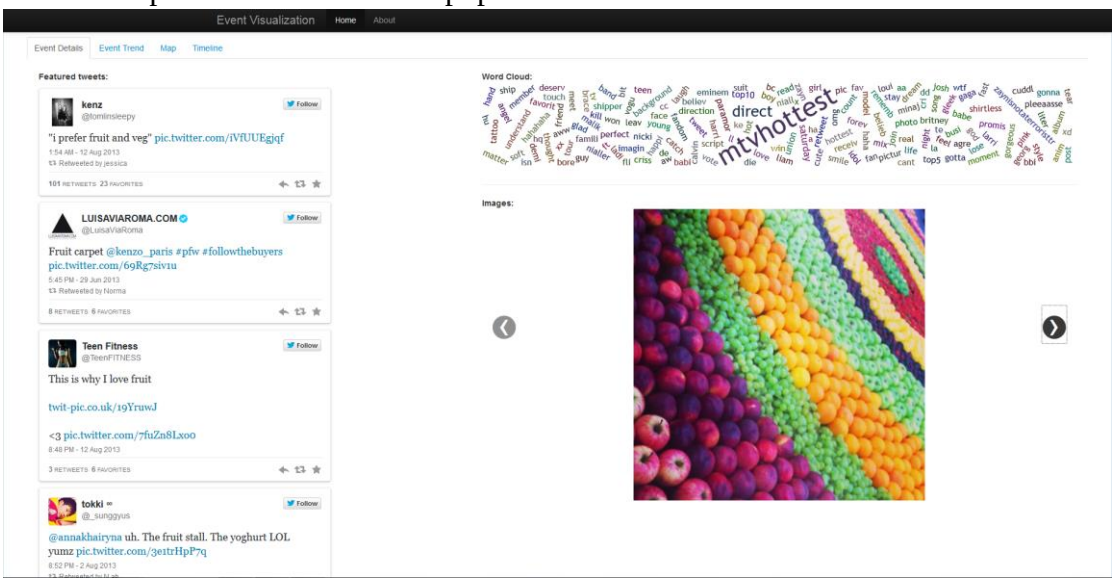

图 3-20 事件详情

事件详情页面也有一个导航栏，包括四项：详情，趋势，地图，时间线。

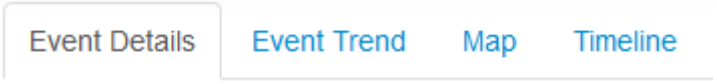

图 3-21 事件详情的导航栏

下面分别叙述：

- 详情

Event Details 如图 3-20 所示，左侧是该事件包含的 tweets，按照转发数降序排列，上限为 100 条。由于这部分内容是由 twitter 提供的 API 抓取，可以直接进行转发，关注，喜欢等操作。

图 3-22 tweet 信息

右上部分是 Word Cloud，包含所有关键字，字体大小关系与它们的词频大小关系一致。

图 3-23 词云

右下部分是该事件包含的 tweets 附带的图片，按照对应的 tweets 的转发数降序排列，上限十张。

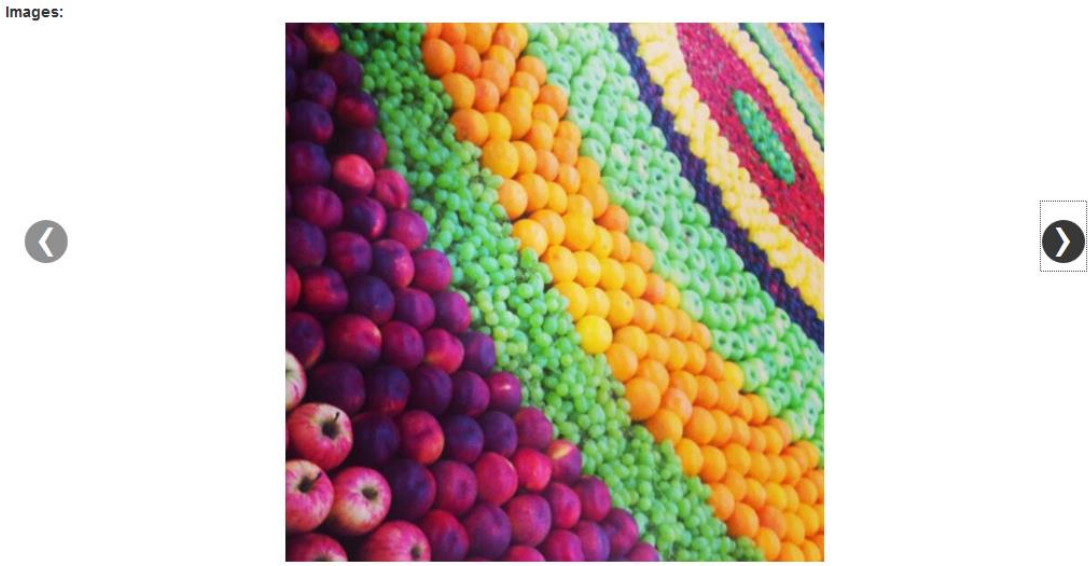

图 3-24 详情的图片部分

- 趋势

点击图 3-21 的导航栏中的 trend，进入事件趋势页面，该页面主要表现事件的时间信息，如发展趋势。地址形式为：

TopicVisualization/index.php?r=selecttrend/trend/index/252

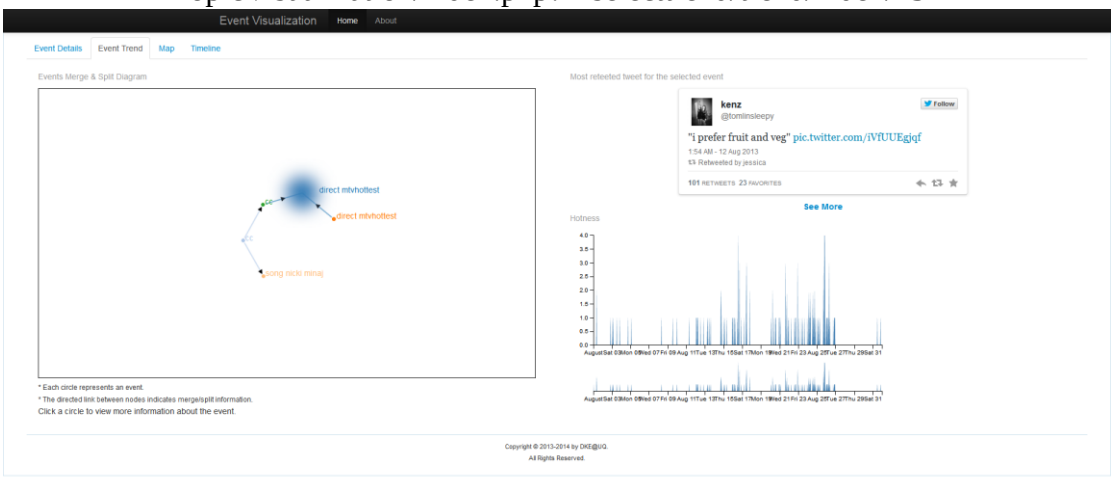

图 3-25 事件趋势页面

左侧是 force-directed 图，每个圆表示一个事件，半径越大热门程度越高。a->b 表示 a 事件分裂或者融合为 b 事件。将鼠标移动到圆上会出现提示框，

显示该事件的摘要信息。这部分图像支持鼠标滚轮的放大缩小以及鼠标拖动功能。

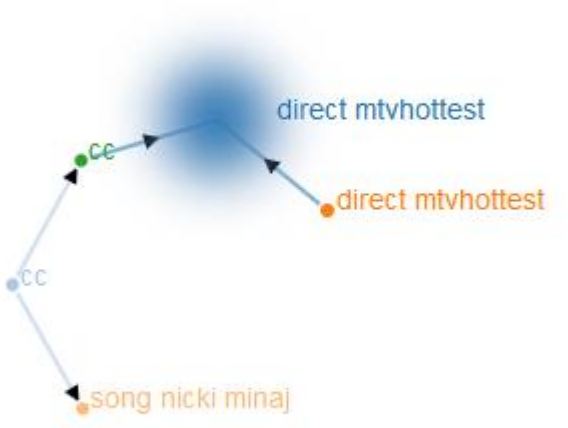

图 3-26 趋势信息的 forced-directed 图

右边上方时选中的事件的转发数最多的 tweet，在 forced-directed 图中选中不同的园会相应的改变这部分信息。

图 3-27 选中事件转发数最多的 tweet

右边下方是热度图，表现这个事件的所有 tweet 的发布时间。

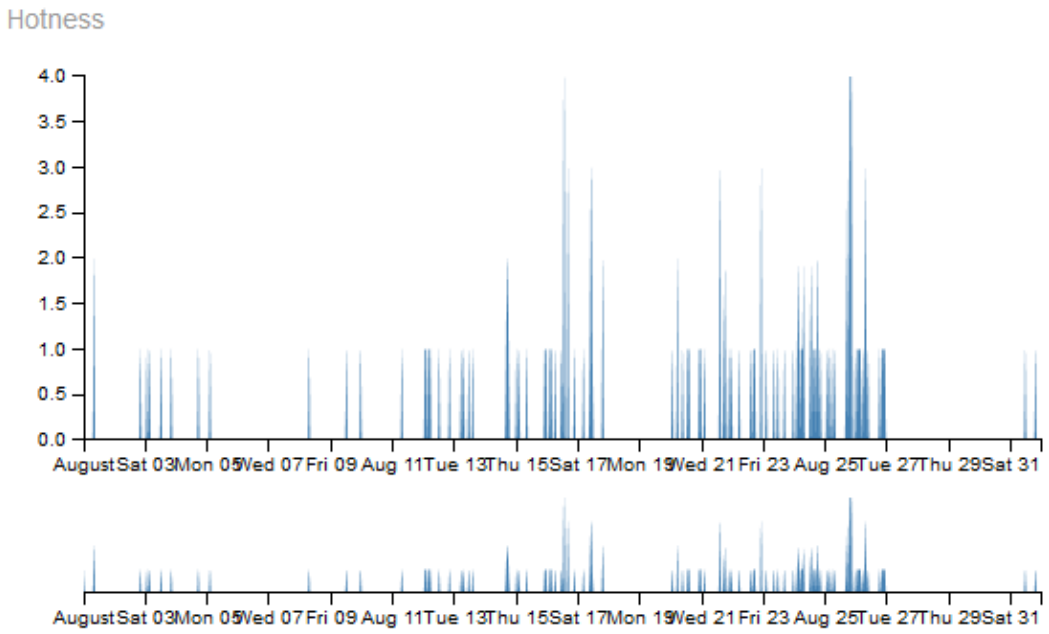

图 3-28 热度图

热度图分上下两部分，下半部分是整体的热度，可以用鼠标选择部分时间段，则上半部分会显示选中时间段内的热度信息。

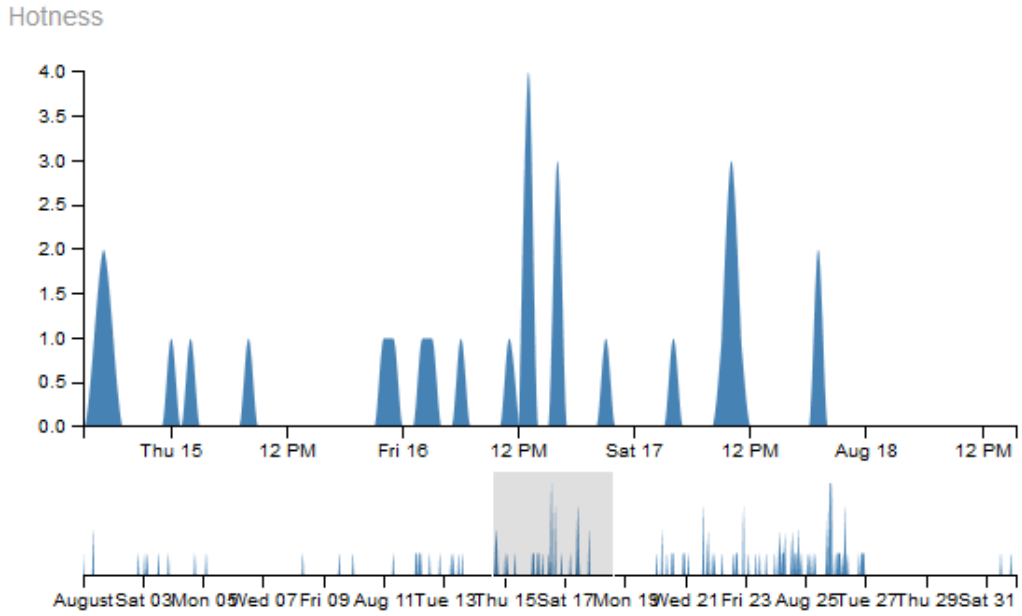

图 3-29 选中部分时间段的效果图

- 地图

  地图部分采用 Google Map API 来绘制，链接形式为

TopicVisualization/index.php?r=selectmap/map/index/252

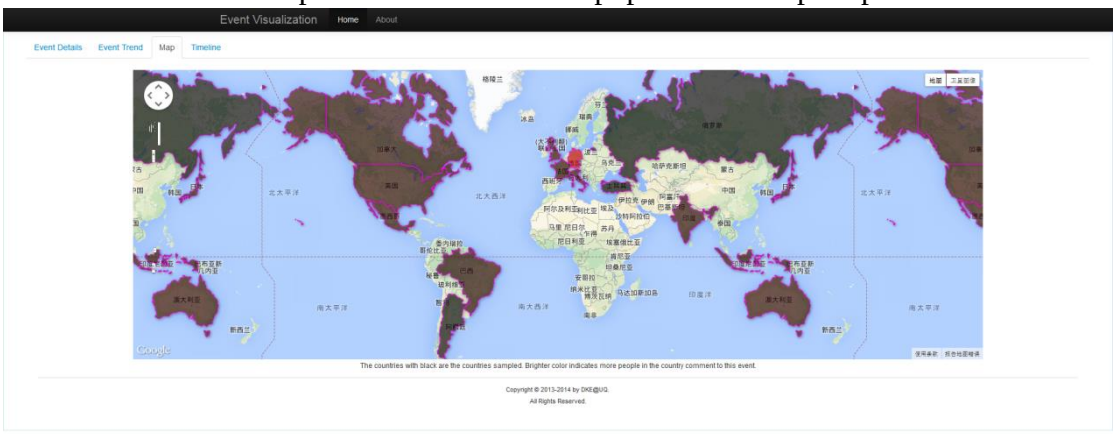

图 3-30 地图页面

背景色着色的国家表示数据库内已有的国家，目前数据库内的微博数据来自阿根廷、澳大利亚、巴西、加拿大、法国、德国、印度、印度尼西亚、意大利、日本、韩国、墨西哥、俄罗斯、土耳其、英国、美国。根据不同国家 twitter 用户对该事件发布的 tweets 的多少，分别对各个国家进行着色，黑色表示无数据，颜色越偏红表示 tweets 越多。由于 Google Map API 限制，无法对每个国家都着不同颜色，因此所有国家被分为四组，地图上也最多会有四种颜色。

点击某个国家，会出现提示框，显示该国家有多少 tweets，并给了一个显示该国家用户 tweets 的链接。

图 3-31 地图提示框

点击该链接会跳转到事件详情页面，然而不同的是此时显示的是该国家

用户的 tweets 和图片。

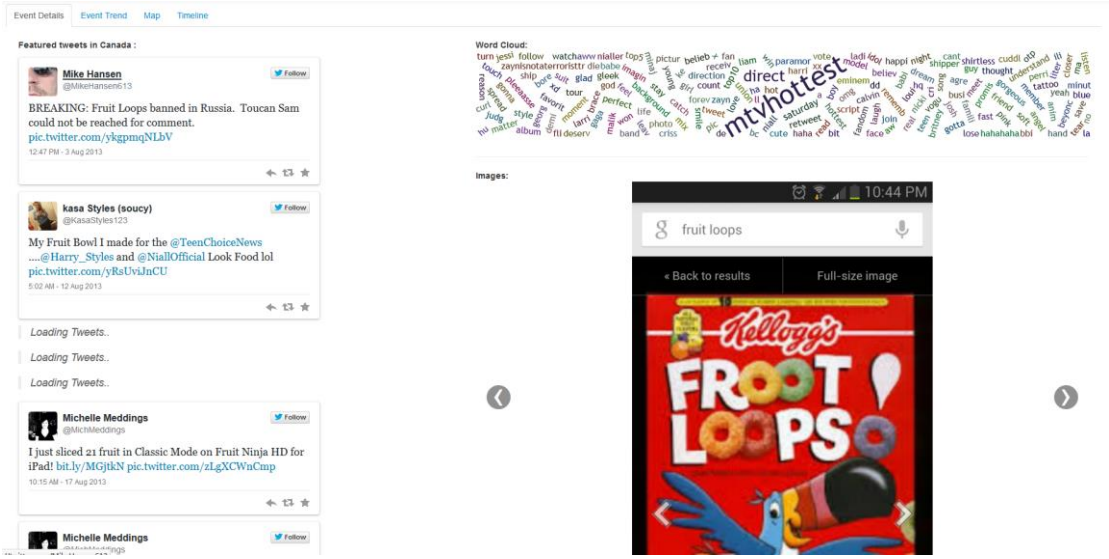

图 3-32 示例：加拿大用户的相关微博

- Timeline

图 3-21 最后的链接也是一个 Timeline，但是显示的是该事件以及其相关事件的时间信息。地址形式为：

TopicVisualization/index.php?r=timeline/event/eid/1

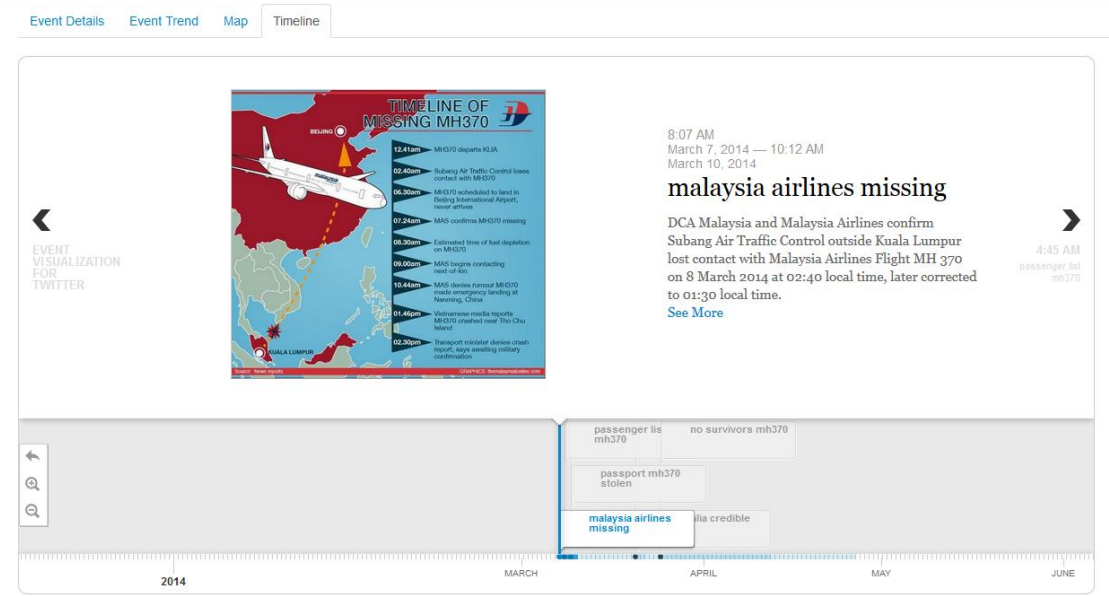

图 3-33 某个事件及其相关事件的 Timeline

3.4.4 管理界面

通过点击网站上方的 Admin 可以访问到管理界面，管理界面有两部分：

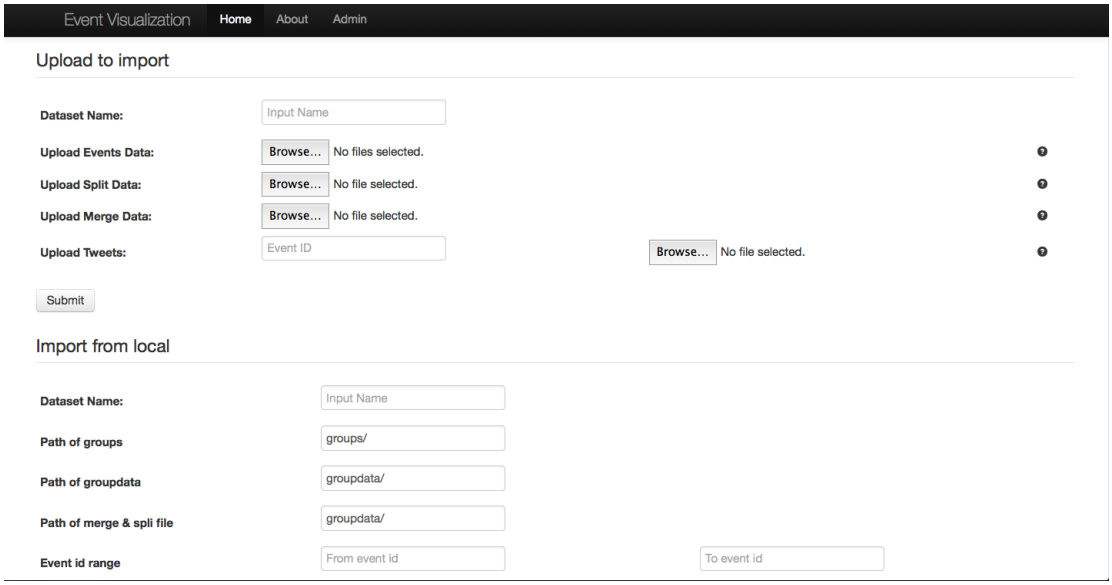

图 3-34 导入部分示意图

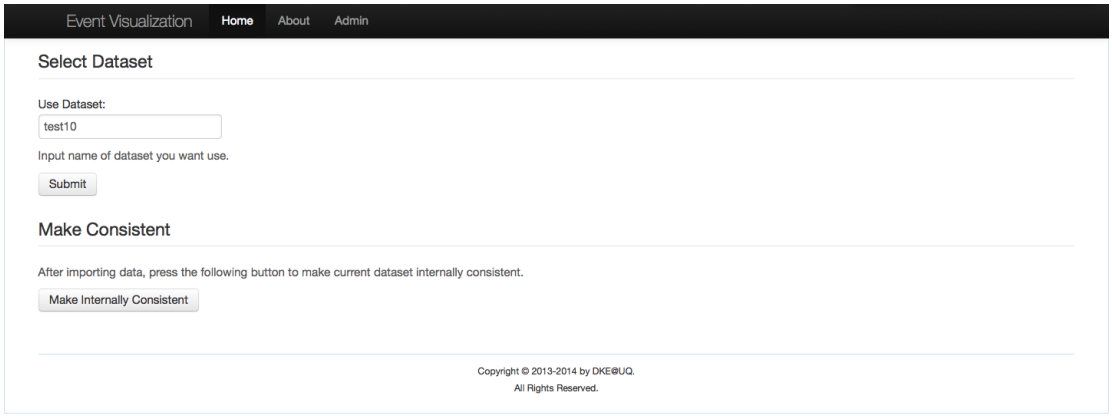

图 3-35 管理部分示意图

其中，导入部分根据提示指定数据集名字，上传相应格式的的文件即可。也可以通过从服务器本地直接导入。管理部分有两个作用，一是选择想使用的数据集，二是在上传完后点击“Make Internally Consistent”按钮确保数据库内部数据统一。

3.4.5 关于

最后是整个网站的 About 页面。通过点击网站最上方的 about 可以访问。目

前该页面显示本项目用到的开源工具。

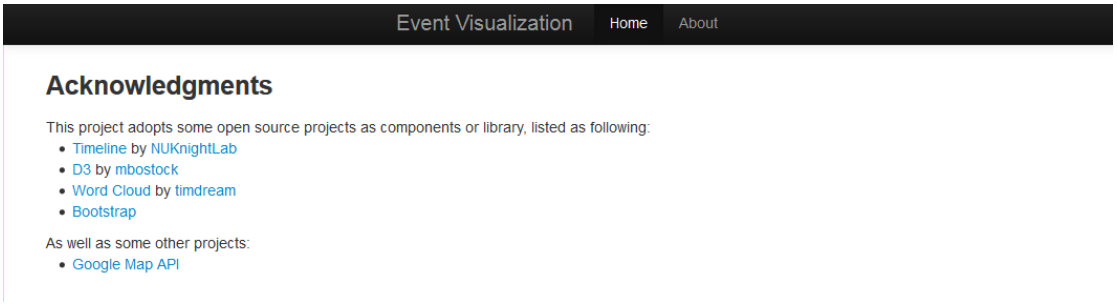

图 3-36 关于页面

第4章 项目成果

4.1 成果概述

本项目源文件共计 6000 余行代码，主要包含两项成果：

1. 一个微博事件发现的对比算法实现
2. 一个用于展示事件的 Web 应用

其中对比算法的实现为实验室博士研究生的理论研究提供了结果对比，可视化模块为目前以及将来对于微博内容的时间发现及跟踪研究提供了一个可视化平台，将算法结果生成的干瘪文本转化为生动形象的图表信息。可视化模块还实现了多数据集并存，选择数据集显示的功能，更方便后续研究者添加实验结果数据。

4.2 总结和展望

目前为止本项目已经完成了应该有的基本内容，然而还有不少可以提高的地方：

1. 由于美工经验不足，可视化模块部分页面显得不够美观，有待改善。
2. 部分页面功能还可以更细一些，比如增加一些选项，使得用户可以调节显示界面。
3. 显示效果还可以做得更绚丽一些。
4. 当数据量很大的时候，加载页面会明显变慢，需要对数据库查询进行优化。

参考文献

[1] http://www.twitter.com

[2] Information Technology Laboratory. Topic detection and tracking evaluation.

[3] Rui Long, Haofen Wang, Yuqiang Chen, Ou Jin, and Yong Yu. Towards effective event detection, tracking and summarization on Microblog data. In Web-Age Information Management, pages 652-663. Springer, 2011.

[4] http://d3js.org/

[5] http://lucene.apache.org/

[6] http://www.yiiframework.com/

[7] http://timeline.knightlab.com/

[8] http://timdream.org/wordcloud2.js/

[9] https://developers.google.com/maps/