华 东 师 范 大 学 软 件 学 院
2013 年软件工程学士学位论文

新浪微博热点话题可视化查询系统

VisualMiner: System for Visually Mining Sina Weibo Trending Topics

姓　　名：张天伦
学　　号：10092510351
班　　级：09 级 3 班
指导教师姓名：钱卫宁
指导教师职称：教授

2013 年 5 月

目　录

摘 要

网民获取信息的渠道正在慢慢向微博转移。一个社会热门话题会在新浪微博上引起广泛讨论。新浪微博支持用户实时搜索和参与话题，但是不能很好地展示话题发展变化的过程。另一方面，Google Trends 通过挖掘和可视化用户搜索的数据，可以直观地呈现一个搜索关键词随时间的变化趋势和在地域上的分布特征。

本文介绍了 VisualMiner，一个新浪微博热点话题的可视化查询系统。我们在海量的微博数据中挖掘一些特征数据，如时间，地点，关键词等，统计话题的讨论数随时间的变化，在不同地域的差异，再借助可视化的方法直观地呈现话题的时间走势和地理分布。除此之外，通过分析微博中出现的情绪词，我们可以挖掘出用户对待某一话题的普遍态度。

我们通过 Sina Weibo API 爬取新浪微博数据，并导入到 HDFS 上，利用 MapReduce 计算框架对原始数据进行分组聚合，再将结果批量上传到 Google App Engine 的数据存储区，最后我们在 Google App Engine 部署 Java Web 应用，借助 Google Chart Tools, Data-Driven Documents 等 JavaScript 库将数据可视化，用户可以在浏览器端查询可视化的数据信息。

在本文中，我们证明我们的应用通过挖掘海量的微博数据，可视化地展示话题的特征，能够为用户全面而直观地关注热点话题提供新的渠道。

关键词：新浪微博，数据挖掘，数据可视化

Abstract

The major media of netizens to get news have been shifting to microblog services. A heated social topic will raise wide discussion on Sina Weibo, China's most popular microblog service. Sina Weibo allows for users to search for a topic and participate in its discussion in realtime. Nonetheless, it lacks the function to provide users with a full view of a topic. On the other hand, Google Trends has been mining and visualizing user's search data, from which a complete picture of a topic over time and across regions could be drawn.

This paper concerns the design and implementation of VisualMiner, which visualizes trends of heated topics on Sina Weibo. By extracting such featuring data as time, location and keywords contained in a piece of weibo, we are able to generate statistics and patterns of a social topic with regard to time and region. With the help of visualization tools, we could present the results to users in a direct and vivid way. Furthermore, we also look into users' emotions towards a topic.

We crawl down data against Sina Weibo API and store them on HDFS, where we group-by and aggregate raw data with MapReduce framework. The processed data are then bulkloaded into Google App Engine's storage layer. Finally, we deploy a Java Web App on top of Google App Engine through which users could view the visualized data. We utilize JavaScript library like Google Chart Tools and Data-Drive Documents to visualize data.

In this paper, we prove that we could bring trends feature to Sina Weibo and present users with a complete picture of a social topic by mining massive data.

Keywords: Sina Weibo, Data Mining, Data Visualization

一、绪论

（一）研究背景

根据中国互联网络信息中心的统计 [1]，截至 2012 年 12 月底，中国有近半数网民在使用微博，比例达到 48.7%，较上一年底增长了 296.0%，相比之下，网络新闻使用率从 2009 年底的 80.1% 下降至 71.5%，网民通过互联网获取新闻信息的渠道正向微博发生转移。

在中文微博产品中，新浪微博[1]最具代表性和影响力。和传统社交网站不同，新浪微博上的社交关系以话题为中心，一个社会热点话题往往能在新浪微博上引起广泛的关注和讨论，例如 2012 伦敦奥运会，江南 style，中国好声音等。和话题相关的微博数量，转发量和评论量会在短期内迅速增加，而且会随着话题的发展保持一段时间。不仅如此，这些数据还带有地域性的特征，如话题发生地附近的相关微博数量显著大于地理距离较远的地方。此外，微博中包含的情绪词可以体现出用户对某一话题的态度和情绪。

微博用户在撰写微博时，可以用 Hashtag [2] 表示一个话题，如“# 雅安地震 #”，新浪微博根据话题的出现次数，可以实时地统计出热点话题; 此外，微话题应用（图 1-1）按照热度（热度由某一话题在近段时间的讨论数及全部讨论数综合计算得出）对话题进行排名，提供话题的简介和相关微博（指出现该话题的微博）。用户可以查询不同时段（一月，一周和 24 小时内），不同地域和不同分类下（如影视，公益等）的热点话题。在现有的新浪微博平台上，用户可以实时地跟踪热点话题并参与讨论，但是并不能了解话题发展的整体趋势和特征，如和话题相关的微博数随时间的变化趋势，在地域上的分布差异，以及微博中的情绪特征等。

不仅如此，文字描述对用户而言乏味枯燥，信息的表达效率较低。与之相比，数据可视化技术 [3] 借助于图形化手段，清晰有效地传达与沟通信息。Google Trends 通过统计用户的搜索词，用可视化的方式显示其随时间

[1]http://weibo.com

图 1-1 新浪微博微话题

Figure 1.1: Sina Weibo Weihuati

的变化趋势（以月为单位）和地域间搜索次数的差异。通过 Google Trends，用户不再需要在大量的文字中寻找自己感兴趣的信息，而是立即能从图像中识别出来。如图 1-2 所示，在 Google Trends 上搜索“地震”，可以很快地在折线的波峰处找到历年来发生的各次地震事件（最为突出的是汶川地震和福岛地震）。而地图上颜色较深的区域也显示出中国和日本是地震的重灾区。

同样地，将微博数据可视化也能帮助用户迅速地提取信息，发现热点话题，关注社会事件。

本文介绍了 VisualMiner，一个新浪微博热点话题的可视化查询系统。系统对新浪微博数据进行挖掘，将话题的信息按照时间，地域，情绪等维度以可视化的方式呈现在网页端，供用户查询检索。用户看到的不再是单调的文字描述，而是形象化的展示，不再局限于热点事件的一个方面，而是事件的全貌。

（二）论文组织

本论文分六个章节，具体章节安排如下：

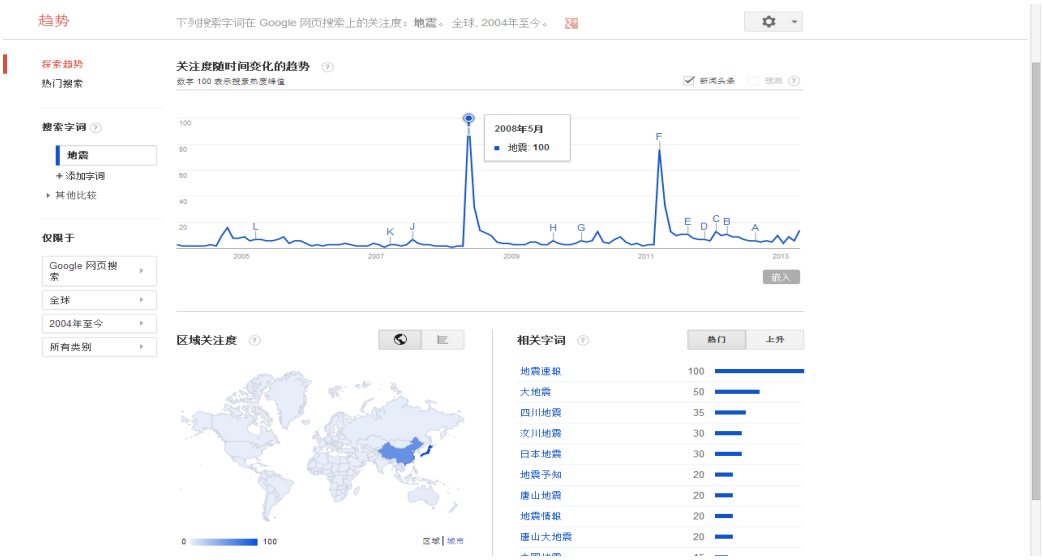

图 1-2 谷歌趋势

Figure 1.2: Google Trends

- 第一章，绪论。介绍论文的研究背景和研究内容
- 第二章，相关工作。介绍系统使用的 Google App Engine，Hadoop 及 jQuery 等相关平台和框架。
- 第三章，系统设计。介绍系统的功能设计，界面设计和架构设计。
- 第四章，系统实现。介绍系统的实现细节，包括数据抓取，数据分析和数据的可视化展示。
- 第五章，实验结果。介绍系统的实现效果。
- 第六章，总结和展望。

二、相关工作

本章主要介绍系统中使用到的相关技术框架：Google App Engine，Hadoop 和 jQuery。

（一）Google App Engine

Google App Engine[4](GAE) 是一个开发、托管网络应用程序的平台，使用 Google 管理的数据中心。在使用 Google App Engine 时，开发者只需上传应用程序，不需要维护任何服务器。Google App Engine 包含以下功能：

- 提供动态网络服务，完全支持常用的网络技术；
- 提供 Datastore 持久存储，并支持查询、排序和事务；
- 提供自动扩展和负载平衡；
- 提供功能完善的本地开发环境，用于在开发者的计算机上模拟 GAE；
- 应用程序可以在 Java 环境和 Python 环境中运行，每种环境提供了标准协议和常用技术以进行网络应用程序开发。

Google App Engine 中数据存储的单位是对象（或实体），实体属于一个种类，具有一组属性。查询可以检索给定种类的实体，按属性值过滤和排序。Google App Engine 提供了两个不同的存储选项，两者的区别在于它们的可用性和一致性保证：

- 主/从数据存储区。使用主 -从复制系统，该系统在将数据写入物理数据中心时异步复制数据，为所有读取和查询都提供了强一致性，其代价是在数据中心出现问题时或进行计划内停机期间，会出现暂时的不可用性。
- High Replication 数据存储区。使用基于 Paxos 算法的系统在各个数据中心之间复制数据。High Replication 针对读取和写入提供了非常高的可用性，其代价是写入时的延迟时间较长。大多数查询是最终一致的。

（二）Hadoop

Hadoop[5] 是一个支持分布式数据密集型计算的软件框架。它主要由分布式存储系统 HDFS 和分布式计算框架 MapReduce 构成。

1. HDFS

HDFS[6] 是一个分布式文件系统，通常由一个存储元数据的 namenode 和多个存储实际数据的 datanode 组成。HDFS 将文件分成固定大小的数据块（默认为 64MB），存储在多台机器上。HDFS 通过给数据块存多个备份（默认为 3，2 个在同一机架上，1 个在不同机架上）实现数据的持久性。HDFS 适合于对大文件的批处理操作，而不适用于低延时的用户交互式应用。

2. MapReduce

MapReduce[7] 是 Google 提出的计算框架，用于大规模数据集的并行计算。MapReduce 作业分为 Map 和 Reduce 两个步骤，开发者只需编写 Map 和 Reduce 函数，专注于算法逻辑，系统会负责资源的管理，任务的分配，提供容错，处理失败。Map 和 Reduce 在键值对结构的数据上操作。Map 函数并行处理输入一个数据域中的键值对，输出另一个数据域的键值对列表。

$$(k_1, v_1) \rightarrow list(k_2, v_2) \tag{2.1}$$

系统根据键对列表进行分组（每个键一个组），并将它们分发到不同的机器。Reduce 函数并行处理每组数据，输出一个数据集合。

$$(k_2, list(v_2)) \rightarrow list(v_3) \tag{2.2}$$

（三）jQuery

jQuery[8] 是一个用于简化客户端对 HTML 进行脚本操作的 JavaScript[9] 库。借助 jQuery，开发者可以方便地操纵 HTML 文档，选择 DOM 元素，处理事件，开发 Ajax[10] 应用。而基于 jQuery 的 jQuery UI[11] 使得 Web 界面的制作更加高效。jQuery 还提供插件，开发者可以将底层的交互，动画

等动作抽象成模块化的控件，从而构建出强大的动态网页和网络应用。此外，jQuery 还提供了多浏览器的支持。

三、系统设计

本章主要介绍系统的功能设计，界面设计和架构设计。

（一）功能设计

用户可以在主页上浏览一个按讨论数排序的话题列表。我们将一个话题的讨论数定义为微博中出现该话题的微博数。我们为每个话题定义了正则表达式，当一条微博中的字符串匹配了话题的正则表达式，就认为它出现了该话题，一条微博中多次匹配只计一次。

用户可以浏览一个话题的下列信息：

- 话题的简介；
- 出现该话题的热门微博（指评论数和转发量总和较多的微博）；
- 该话题讨论数随时间变化（以天为单位）的折线图；
- 该话题讨论数在各省分布的地图；
- 该话题中出现的情绪词云图；
- 该话题的情绪特征柱状图；

情绪词是我们参考心理学家的相关研究（Ekman 六类基本情绪 [12], GPOMS 心境量表 [13] 及中文情绪形容词检测表 [14]）定义的词汇，如开心，失意，生气等。我们统计某话题下情绪词出现的次数，包括一条微博中出现的所有情绪词。我们将这些情绪词分为快乐，悲伤，愤怒，恐惧，惊奇，厌恶和其他这七类情绪特征，并展示情绪特征的出现次数。

（二）界面设计

用户通过 Web 界面访问我们的系统。界面总体上分成左右两栏：左侧为导航栏，右侧为内容栏。

在首页上，导航栏显示的是热门话题和话题分类选项及搜索框。内容栏显示的是根据话题讨论数排序的话题列表。列表中的一项包括话题名称和话题的讨论数。内容栏右下角提供向前向后翻页按钮。如图 3-1 所示。

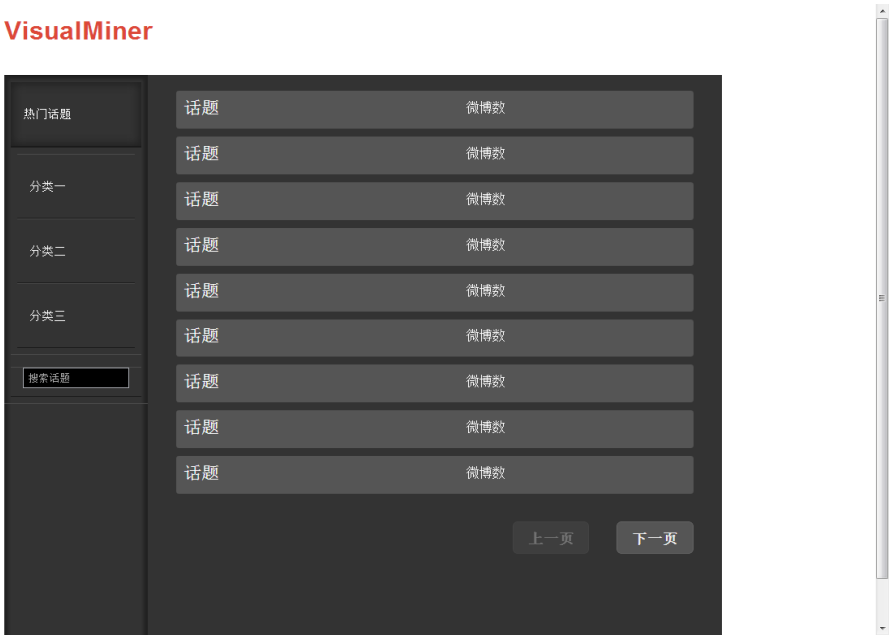

图 3-1 主页

Figure 3.1: Home Page

在用户点击某个话题后，界面跳转到该话题的页面。此时内容栏分为上下两部分，上面固定显示话题的介绍，下面是可视化栏。导航栏提供热门话题，事件概况，情绪特征，地域特征，变化趋势 5 个选项。点击热门话题，回到话题列表页面。点击其余选项，可视化栏显示相应页面。默认展示的是事件概况页面。

事件概况页面上显示出现该话题的热门微博。如图 3-2 所示。

在导航栏上点击情绪特征，可视化栏进入情绪特征页面。该页面左侧为情绪特征柱状图，右侧为情绪词云图。如图 3-3 所示。

在导航栏上点击地域特征，可视化栏进入地域特征页面。该页面显示按省划分的地图。一个区域填充色的深浅表示该区域出现该话题的微博数。当用户将鼠标移到一个区域上时，可以显示具体数值。如图 3-4 所示。

在导航栏上点击变化趋势，可视化栏进入变化趋势页面。该页面显示对话题的讨论数随时间变化的情况，用折线图表示，当用户将鼠标移到一个

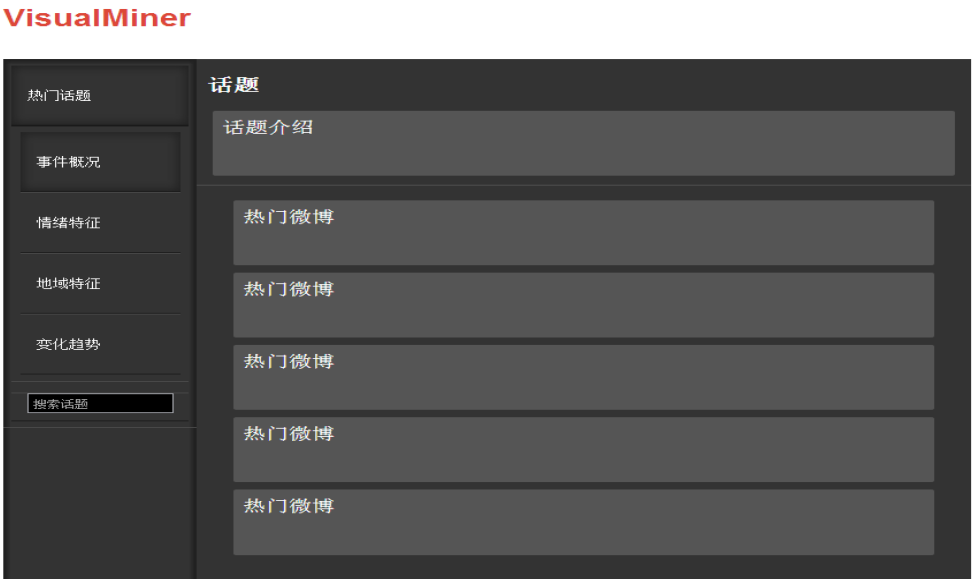

图 3-2 事件概况

Figure 3.2: overview

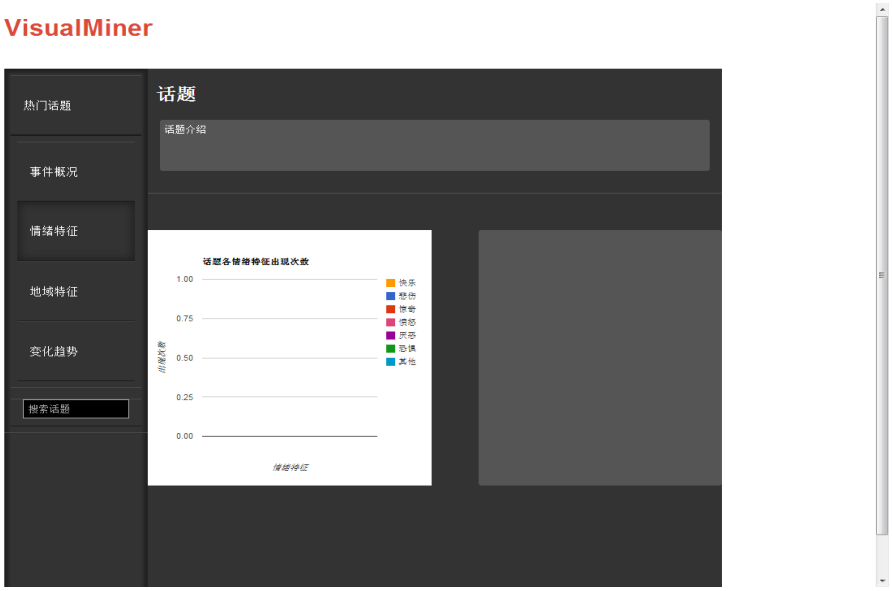

图 3-3 情绪特征

Figure 3.3: emotion

点上时可以显示具体数值。如图 3-5 所示。

以上所有的界面都在一个 HTML 页面上，由 jQuery 实现动态页面跳转。

图 3-4 地域特征

Figure 3.4: location

图 3-5 变化趋势

Figure 3.5: trend

（三）架构设计

系统架构如图 3-6 所示，总体上分成三个模块：数据获取模块，数据分析模块和数据呈现模块。

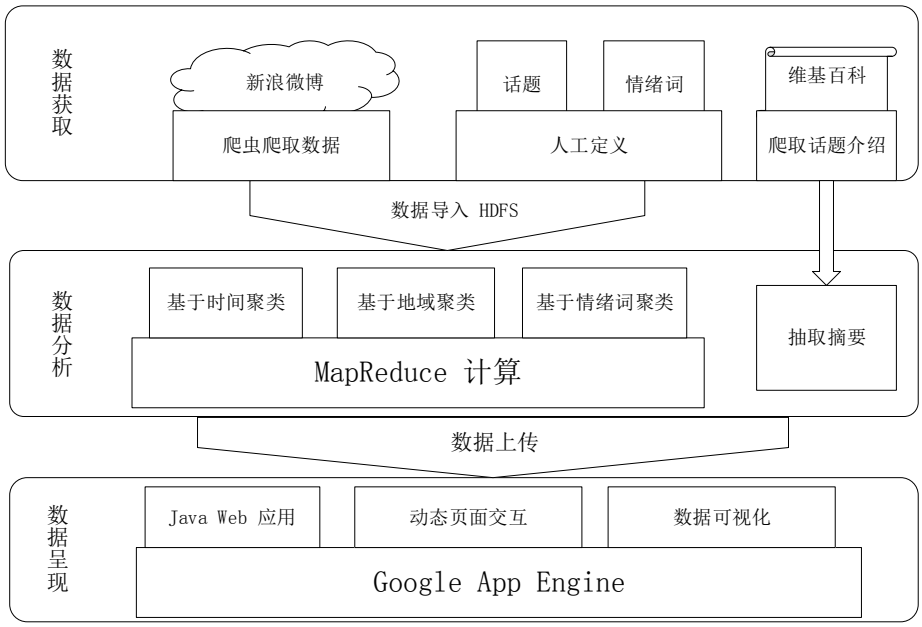

图 3-6 系统架构

Figure 3.6: system architecture

1. **数据获取模块**

数据获取模块分为三个部分：

- 第一部分，设计爬虫爬取新浪微博数据。
- 第二部分，根据曾在新浪微博上引起广泛关注的话题定义话题列表，基于心理学家的相关研究定义情绪词和情绪特征。
- 第三部分，通过 MediaWiki API 从维基百科爬取话题的介绍页面。

我们将前两部分数据导入 HDFS，用于下一阶段的 MapReduce 分析，第三部分的数据用于下一阶段抽取话题的简介。

2. **数据分析模块**

数据分析模块分为两个部分：

- 第一部分，我们应用 MapReduce 计算框架分析微博数据，在时间，地域，情绪等维度上对微博数据进行分组聚合。
- 第二部分，我们从爬取的维基百科页面中抽取第一段话，用作话题的简介。

我们将两部分计算结果上传到 Google App Engine 的数据存储区。我们选用 High Replication 存储区存储数据，虽然其写入时的延时较高，但我们的系统在批量导入数据后只需要响应读操作，而它提供的高可靠性，保证不会因为机器崩溃导致用户无法访问，影响用户体验。我们的查询是基于键（话题）的，而不是基于关系的，所以我们没有选择传统的关系型数据库[15]。

3. 数据呈现模块

我们在 Google App Engine 上构建 Java Web 应用，利用 jQuery 实现动态用户界面和 Ajax，调用 JavaScript 库（Google Chart Tools[16]，Data-Driven Documents[17] 等）可视化数据。

四、系统实现

本章介绍系统的实现细节，包括数据获取，数据分析和数据呈现。

（一）数据获取

1. 爬取微博数据

新浪微博每天会产生一亿多条微博 [1]，我们不可能爬取所有的微博数据，因此需要设计一个爬虫可以高效地抽取一个样本数据。我们使用了 Haixin Ma 等人 [18] 开发的爬虫从新浪微博抓取数据。爬虫使用 Sina Weibo API 抓取用户档案，社交关系和微博。抓取的过程如图 4-1 所示。

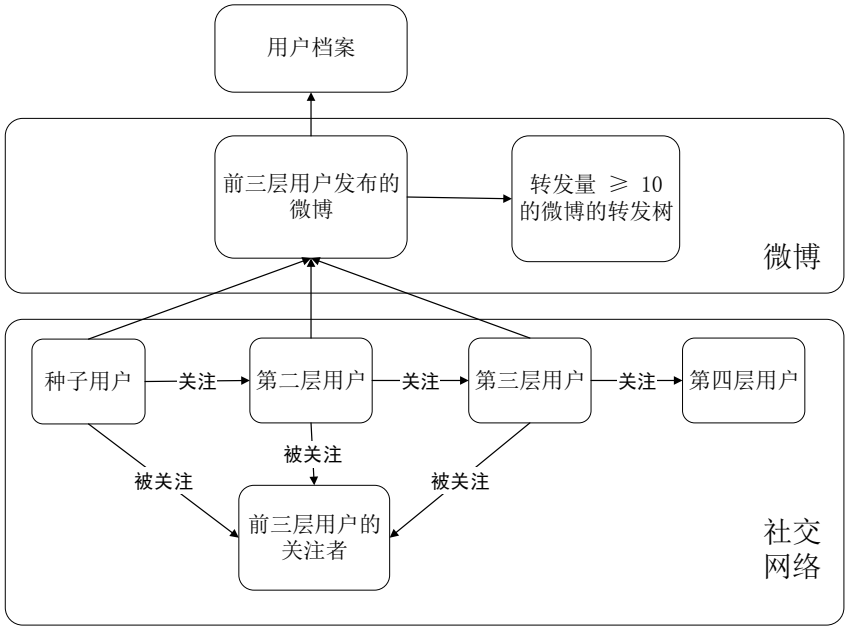

图 4-1 数据抓取

Figure 4.1: crawling data

爬虫从 32 个种子用户出发，以广度优先的方式通过关注关系寻找下层用户，收集了前三层用户的微博，及转发量大于 10 的微博的转发树。

2. 爬取维基百科页面

维基百科提供了 MediaWiki API 支持应用程序爬取维基百科页面，我们使用 HTTP POST[19] 动作，以话题名为参数，爬取话题在维基百科上的介绍页面，返回的数据为 HTML 格式。

（二）数据分析

1. MapReduce 计算

我们抓取的微博数据以 JSON （JavaScript Object Notation）[20] 格式储存，（这里为了方便展示，我们将微博的各属性分行显示，实际上一行数据含有多条这样的微博，此外这里隐去了无关的部分）。

```
{
  "created_at": "2010-04-10",
        "text": 谷歌退出了中国"",
        "user": {
          "province": "41",
          "city": "1",
        },
        "reposts_count": 10,
        "comments_count": 20,
}
```

created_at 表示微博发布的时间；text 表示微博的正文；user 表示微博用户的信息；其中包括用户的位置信息：province（省）和 city（市）；reposts_count 和 comments_count 表示微博的转发量和评论量。在解析过程中，我们将其以字符串的形式读入，然后反序列化为一个 JSON 对象，这样我们可以调用 JSON 对象暴露的方法获取上述属性值。

我们为每一话题定义了对应的正则表达式。如果一条微博中含有能匹配某个正则表达式的字符串，则表示它出现了该话题。如一个话题是“谷

歌退出中国”，对应的正则表达式为“(Goolge|google| 谷歌).* 退出.* 中国”，则一条含有“Google 正式退出了中国”的微博中就出现了“谷歌退出中国”这一话题。

我们应用 MapReduce 计算框架在时间，地域，情绪等维度上对数据进行分组和聚合。首先我们通过伪代码解释 Map 和 Reduce 的过程。regexToTopicMap 是一个从正则表达式映射到话题的哈希表。

Algorithm 1 map(key, value, context)

```
1: tweetList ← value.deserialize()
2: for all tweet in tweetList do
3:    text ← tweet.getText()
4:    count ← tweet.getCount()
5:    for all regex in regexToTopicMap do
6:       if tweet matches regex then
7:          topic ← regexToTopicMap.get(regex)
8:          context.write(topic, count)
9:       end if
10:   end for
11: end for
```

在 Map 阶段，我们根据话题对微博进行分组。Map 阶段的输入以行为单位，key 是该行在文件中的偏移量，value 是该行的数据，在这里是包含多条微博的字符串。我们将所有这样的字符串反序列化成 JSON 对象。之后，我们从中获取 text （微博正文）和 count （统计量，如 reposts_count，comments_count 等）。下面我们遍历 regexToTopicMap 的所有键，寻找能够匹配 text 的正则表达式。如果匹配成功，则输出话题和统计量的键值对。MapReduce 框架会自动根据话题对 Map 输出进行分组，作为 Reduce 的输入。

在 Reduce 阶段，我们对各组话题的统计数据进行聚合。Reduce 阶段的输入以话题为单位，key 是某一话题，valueList 是出现了这一话题的所有微博的统计量列表。在这个例子中，我们将统计量相加进行求和操作，当然，我们也可以求得最大值，平均值等其他统计结果。

Algorithm 2 reduce(key, valueList, context)

1: $sum \leftarrow 0$
2: **for all** $value$ in $valueList$ **do**
3: $\quad sum \leftarrow sum + value$
4: **end for**
5: $context.write(key, sum)$

下面我们具体解释对 MapReduce 框架的应用过程。根据功能设计，我们需要统计话题的讨论数。由于 Reduce 阶段输入的 valueList 对应一个话题，其中的一个 value 表示话题出现了一次，那么它的长度就是话题的讨论数。在对不同维度进行聚合操作时，我们只需要修改 Map 输出的键值。在分析话题讨论数随时间的变化趋势时，我们取出 created_at 字段，将它和话题拼接在一起作为 Map 输出的键（用'\t' 隔开）。而在分析地域分布时，则是取出 province 字段，统计一个话题在一个省的讨论数。源数据的 province 是一个编号，新浪微博提供了编号和省份名的映射表，包括各省，直辖市，港澳台，海外和其它。为了地图显示方便，我们忽略海外和其它的数据。对于和情绪词有关的统计，我们需要增加一个循环，遍历情绪词表，寻找微博正文中的完全匹配。我们将话题，情绪词和情绪特征的拼接词作为 Map 输出的键。

2. 抽取话题简介

在数据获取模块，我们从维基百科爬取了话题的 HTML 页面，但话题的内容不是展示的重点，我们只需要抽取话题简介。维基百科通常会在第一段正文给出一个词条的简介。如图 4-2 的红框部分。

这对应到 HTML，就是第一组“ " 和“ ”包含的内容，我们将这段内容取出，去除其中的 HTML 标签（如“”等），就生成了话题的简介。

3. 数据存储

Google App Engine 的 Datastore 数据存储区提供了批量上传导入数据（CSV 或者 XML 文件）的功能。数据存储区是非关系型的，它的存储单位

图 4-2 维基百科

Figure 4.2: wikipedia

是实体（Entity），表 4-1 是 Datastore 与关系型数据库（RDBMS）类似概念的比较。

表 4-1 Datastore 与 RDBMS 的比较

Table 4.1: Datastore vs. RDBMS

Datastore	**RDBMS**
Kind	Table
Entity	Row
Property	Column
Entity key	Primary key

在上传数据之前，需要先在配置文件中定义一个实体的种类（Kind）和属性（Property）。例如我们要上传话题的讨论数，评论量和转发量，可以定义一个 topic_count 种类，实体的属性包括 topic （话题），tweets_count （讨论数），comments_count （评论数）和 repost_count （转发数）。Google App

Engine 提供了在网页端查询数据的服务，图 4-3 即是数据存储区中的数据格式，它还包括存储区自动生成的 Entity key，作为实体的唯一标识。

ID/Name	comments_count	repost_count	topic	tweets_count
id=40011	307408	789503	微博反腐	12899
id=40012	25423	58066	小悦悦事件	2446
id=40013	200157	517066	重庆打黑事件	12243
id=40014	361600	489555	公知约架	41808
id=40015	22900	41391	少年被充气泵塞肛门穿孔	2189
id=40016	173521	613770	台湾塑化剂	10748
id=40017	14009	23952	扶老人被诬告	817
id=40018	35539	133334	双汇瘦肉精	3036
id=40019	266155	644193	3B大战	31668
id=40020	114959	593524	微公益	16565
id=41011	66987	273639	微博打拐及解救乞讨儿童	10327
id=41012	215191	622104	日本地震	15385
id=41013	345570	849221	周克华抢劫案	20758
id=41014	11979	25292	申纪兰	609
id=41015	181	788	上海地铁10号线	38
id=41016	915885	2509344	文革	89389
id=41017	49013	134377	活熊取胆	2372
id=41018	11157	21327	狼爸教育	1435
id=41019	27680	96209	钱云会	2247
id=41020	286997	547594	重庆不雅视频	6111

图 4-3 GAE 数据存储

Figure 4.3: GAE Datastore Entity

（三）数据呈现

我们构建了一个基于 Google App Engine 的 Java Web 应用。在客户端，我们借助 jQuery 操纵 HTML 文档，实现动画效果，监听事件，因此只需要一个页面即能呈现各组数据，而不需要跳转。另外，我们直接调用 jQuery 的 Ajax 方法完成客户端和服务器端的数据传输。

如图 4-4 所示，当我们点击页面触发事件时，jQuery 的 Ajax 方法从客户端发起一个 XmlHttpRequest 请求，在服务器端由相应的 Java Servlet[21] 处理请求（如从 High Replication 存储区读取某个话题的微博数），返回序列化后的结果（如 JSON 字符串），触发 jQuery 重新绘制页面。

我们使用了基于 JavaScript 的可视化工具 Google Chart Tools 和 Data-Driven Documents （D3）。

在绘制折线图，柱状图和地域图时，我们加载 Google Chart Tools 的 corechart 包（包括 linechart，columnchart，geochart），使用 DataTable[1] 对象

[1] https://developers.google.com/chart/interactive/docs/reference

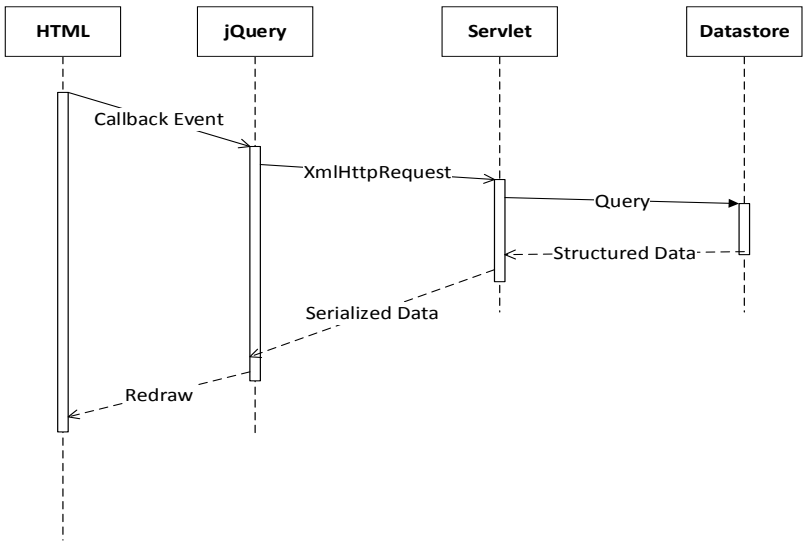

图 4-4 网页交互

Figure 4.4: interaction

将返回的 JSON （嵌套）结构转化为图表（行列）结构。

在绘制情绪词云图时，我们调用了基于 D3 的 d3-cloud[2] JavaScript 词汇云图实现。情绪词云图的基本思路是根据情绪词的出现次数决定其在云图中显示字体的大小，这样用户一眼就能辨识出话题中的主要情绪。我们设计了一个简单而显示效果不错的函数（4.1）来实现这一点，其中 size 是字体大小，count 是情绪词出现次数，max 是所有 count 的最大值。

$$size = count/max * 90 + 10 \tag{4.1}$$

同时，我们将相同情绪特征的情绪词用同一种颜色表示，并且和柱状图中对应情绪特征的颜色一致。这样用户可以很方便地通过情绪特征找到它下面的情绪词，更细粒度地了解话题中的情绪。

[2]https://github.com/jasondavies/d3-cloud

五、系统测试

本章介绍系统的测试结果。

（一）测试设置

系统的测试设置如下：

- 源数据：新浪微博从 2012 年 3 月到 2013 年 1 月底的数据，共 200 GB。
- Hadoop 集群配置：1 个 master 节点（32G RAM, 2 CPU / 8 Core 和 7 个 slave 节点（16G RAM，2 CPU / 8 core）。每个数据块存两份。
- 测试平台：Google Chrome 26 和 Mozilla Firefox 20
- Google App Engine 配置：读操作和写操作每 24 小时 5 万次上限。

（二）测试结果

我们选择了一些较为典型的页面来展示我们的测试结果。如图 5-1，主页上是话题列表，我们用数字和进度条表示话题的讨论数。

图 5-1 主页效果

Figure 5.1: homepage

如图 5-2 是在主页上点击“玉树地震”后进入其事件概况页面，包括话题的简介及评论数和转发数和位居前 5 的微博，我们保留了微博的作者和内容。通过这些信息，用户可以清楚地了解一个话题的内容和新浪微博上的主流观点。

图 5-2 事件概况效果

Figure 5.2: overview page

图 5-3 显示的是新浪微博上关于“上海 11.15 大火”话题的情绪特征页面。通过柱状图，我们可以明显看到“悲伤”（蓝色）是主导情绪，这和人们一般对灾难的反应相符。排在第二的“快乐”（快乐）似乎有悖常理，进一步查询右边的情绪词云图可以发现这里“快乐”的含义是“相信”，“美丽”，“共勉”这些在伤痛中互相勉励，共度难关的词汇。结合情绪特征和情绪词，我们可以推测出人们对这场不幸的主要心态是悲伤中不失信心。

图 5-4 显示的是新浪微博上关于“上海 11.15 大火”话题的地域分布页面。图上深色区域表示话题讨论数较多的地域。我们可以发现话题的讨论集中在了话题的发生地（“上海”）。注意这仅仅作为参考，不能排除其它地

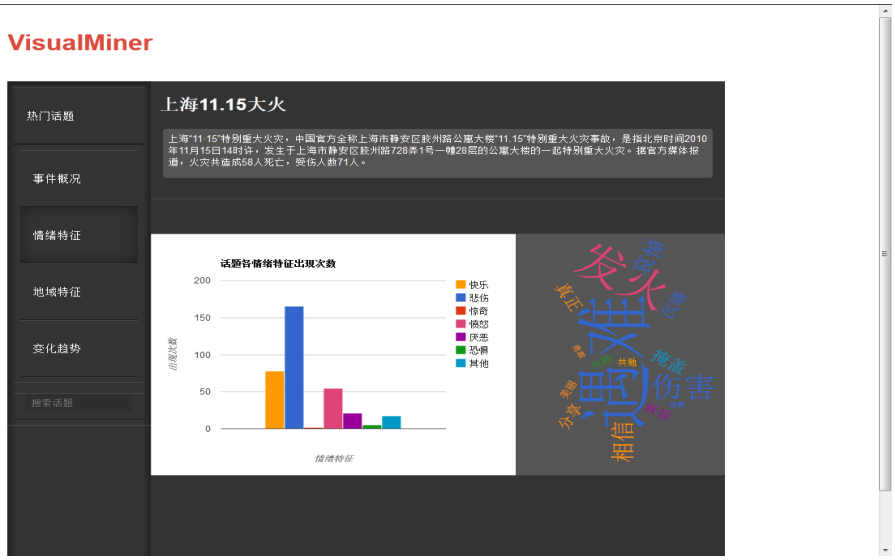

图 5-3 情绪特征效果

Figure 5.3: emotion page

方的微博用户刚好对该话题感兴趣，也不是所有的话题都能呈现出地域差异。

图 5-4 地域特征效果

Figure 5.4: location page

图 5-5 显示的是新浪微博上关于“伦敦奥运会”话题的变化趋势页面。有关“伦敦奥运会”的讨论数在 2012 年 8 月显著上升，随后回落。而那段时间正是伦敦奥运会举办的时间。通过变化趋势图，用户能立即找到话题的起止时间。

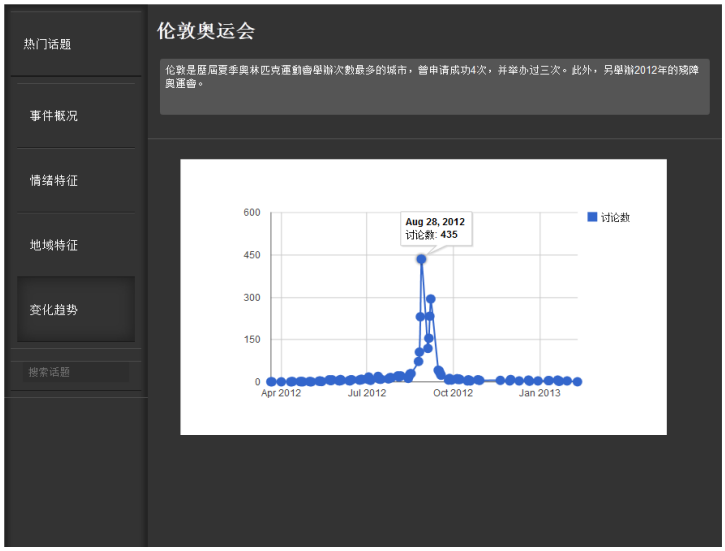

图 5-5 变化趋势效果

Figure 5.5: trend page

六、总结和展望

本文介绍了 VisualMiner，一个新浪微博热点话题的可视化查询系统。系统通过爬取新浪微博数据，对一个话题在时间，地域和情绪等维度上进行分组聚合，将结果数据以可视化的方式呈现给用户。用户可以直观地了解话题的讨论数随时间的变化趋势，在地域上的分布情况以及话题中的情绪特征。

虽然如此，系统在架构和功能等方面均有很大的改进空间。

在架构方面，虽然使用 MapReduce 框架编写程序可以方便地对海量的微博数据进行并行计算，但是直接用 SQL 语言表达更加直观高效。另一方面，MapReduce 适合于离线分析，不适合于即时的用户交互，这限制了系统的功能。我们考虑在下一步尝试 Google 的 BigQuery[22]，它是 Google 的列存储系统 Dremel[23] 的公开版本。BigQuery 支持类 SQL 的查询语言，支持嵌套的数据格式（如 JSON），可以即时分析海量数据，这些特性很符合我们的需求。此外，它还支持将 Google App Engine 上存储的数据导入 BigQuery，方便了数据的迁移。

在功能方面，目前我们预先定义了话题，对抓取的微博数据进行离线分析。在下一步的工作中，我们希望能够定时（如每天）抓取新浪微博上提及热点话题的数据，由用户输入话题，使用 BigQuery 在线分析微博数据，用可视化技术实时地向用户呈现一个话题近期的变化趋势（如与前一天相比的变化量），或是当前各地域该话题的讨论情况。此外，基于关键词的情绪分析十分粗糙（例如“不快乐”虽然包含“快乐”一词，表达的却是完全相反的情绪），我们希望能够对微博进行语义分析，挖掘其中的情绪特征。

我们希望在未来的工作中在这些方面继续探索，不断完善我们的系统。

参考文献

[1] 中国互联网络信息中心. 第 29 次中国互联网络发展状况统计报告 [R]. Technical report, 中国互联网络信息中心, Jan 2012.

[2] Wikipedia. Hashtag[E]. http://en.wikipedia.org/wiki/Hashtag, Apr 2013.

[3] William S. Clevelanda and Robert McGilla. Graphical perception: Theory, experimentation, and application to the development of graphical methods[J]. *Journal of the American Statistical Association*, 79(387):531--554, 1984.

[4] Dan Sanderson. *Programming Google App Engine[M]*, pages 129--244. O'Reilly Media, USA, Oct 2012.

[5] Tom White. *Hadoop: The Definitive Guide[M]*, pages 17--80. O'Reilly Media, USA, May 2012.

[6] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system[C]. In *MSST*, Incline Village, NV, 2010.

[7] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters[C]. In *OSDI*, San Francisco, CA, USA, 2004.

[8] David Flanagan. *jQuery Pocket Reference[M]*, pages 1--150. O'Reilly Media, USA, Dec 2010.

[9] Douglas Crockford. *JavaScript: The Good Parts[M]*, pages 1--147. O'Reilly Media, USA, Aug 2008.

[10] Anthony T. Holdener III. *Ajax: The Definitive Guide[M]*, pages 112--331. O'Reilly Media, USA, Jan 2008.

[11] Eric Sarrion. *jQuery UI[M]*, pages 1--213. O'Reilly Media, USA, Mar 2012.

[12] Paul Ekman and Wallace V Friesen. Constants across cultures in the face and emotion[J]. *Journal of Personality and Social Psychology*, 17(2):124--129, Feb 1971.

[13] J. Bollen, H. Mao, and X.-J Zeng. Twitter mood predicts the stock market[J]. *Journal of Computational Science*, 2(1):1--8, 2011.

[14] 钟杰, 钱铭怡. 中文情绪形容词检测表的编制与信效度研究 [J]. 《中国临床心理学杂志》, 第 1 期:9--13 页, 2005 年.

[15] Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. *Database System Concepts, Sixth Edition[M]*, pages 257--426. McGraw-Hill, USA, 2010.

[16] Google. Google Chart Tools[E]. https://developers.google.com/chart/, Mar 2012.

[17] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3: Data-driven documents[J]. *IEEE Transactions on Visualization and Computer Graphics archive*, 17(12):2301--2309, Dec 2011.

[18] Haixin Ma, Weining Qian, Fan Xia, Xiaofeng He, Jun Xu, and Aoying Zhou. Towards modeling popularity of microblogs[J]. *Frontiers of Computer Science*, 7(2):171--184, Apr 2013.

[19] David Gourley, Brian Totty, Marjorie Sayer, Anshu Aggarwal, and Sailu Reddy. *HTTP: The Definitive Guide[M]*, pages 102--413. O'Reilly Media, USA, Sep 2002.

[20] Douglas Crockford. The application/json Media Type for JavaScript Object Notation (JSON)[E]. http://www.ietf.org/rfc/rfc4627.txt?number=4627, July 2006.

[21] Jason Hunter. *Java Servlet Programming, 2nd Edition[M]*, pages 110--300. O'Reilly Media, USA, Apr 2001.

[22] Google. Google BigQuery[E]. https://developers.google.com/bigquery/, Jan 2013.

[23] Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, and Theo Vassilakis. Dremel: Interactive analysis of web-scale datasets[C]. In *VLDB*, Singapore, 2010.

[24] Scott Murray. *Interactive Data Visualization for the Web[M]*, pages 1--110. O'Reilly Media, USA, Mar 2013.

致谢

首先，我要声明 VisualMiner 不是由我一个人实现的系统，而是由钱卫宁老师指导，夏帆学长，马海欣学姐，叶励晶同学和我共同参与的项目，系统中使用的 Hadoop 实验集群由大数据中心提供，实验期间得到了集群管理员张磊学长的帮助，因此，我向以上所有老师和同学表示由衷的感谢。

我要特别感谢论文指导老师钱老师。从听他的操作系统课开始，钱老师渊博的知识和孜孜不倦的学习态度一直激励着我，使我对学术研究产生了浓厚的兴趣。我要特别感谢马海欣学姐。她为本文的撰写提供了诸多思路，我在论文中也引用了她之前的工作。我要特别感谢我的室友们，怀念我们一起为各自目标奋斗的日子。

我要感谢社区（StackExchange, Google Group）里的程序员们，他们耐心地解答了我的问题，分享自己的经验，是我的榜样。我要感谢我的家人和朋友们，他们是我永远的堡垒。