计算机科学与技术学院 大数据管理与分析 课程实验报告

实验题目：数据分析系统的设计与实现		**学号：**201605130116
日期：2019.6.18	**班级：**2016 级泰山学堂	**姓名：**杜洪超
Email：1503345074@qq.com		

实验目的：

随着 Hadoop 与 Spark 产生的影响越来越深，各种基于 Hadoop 与 Spark 平台的数据分析系统也随之出现。本次实验要求利用之前实验以及所学知识，实现一个基于 Hadoop、Spark 或其他大数据平台的数据分析系统，理解其中的实现细节以及各种算法的原理。

实验软件和硬件环境：

软件环境：

系统：Windows 64 位

软件：openjdk-7-jre,openjdk-7-jdk,java1.7.0_95

Hadoop 2.9.2，spark-2.4.3 python 3.6

Bootstrap 4.0,Vue,jquery-3.2.1

AntV G2,EChart

硬件环境：

CPU: Intel® Core™ i7-5500U CPU @ 2.40GHz × 4

内存：7.7 GiB

实验原理和方法：

基于 skark 大数据分析平台，对 QQ 空间说说进行数据分析，主要分析说说时间分布和内容。通过分析说说发布的时间在不同时间粒度下的分布，可以从中挖掘出用户活跃情况。通过对说说内容进行处理，可以获得用户使用的高频词，分析用户的日常语气。在此基础上，我们可以根据分析得到的用户画像，寻找和某一用户最相似的用户，类似于好友推荐，可以从时间和内容两个维度进行分析，最终加权得到最终结果。

实验步骤：（不要求罗列完整源代码）

1. 安装与配置环境

   安装 hadoop 以及 pyspark，为数据处理做准备。

2. 准备数据集

   爬虫抓取说说信息。实验中仅抓取了好友的说说，实际上可以通过对好友的好友进行二次检索实现抓取大量数据，为了方便处理没有予以实现。

   具体抓取是通过 selenium 实现 web 上 QQ 空间的自动化登陆。成功登陆空间后，通过构造 url 的方法获取好友列表；再依次访问好友空间，拉取说说；只有好友设置为可见的信息能被抓取到，如有限制空间访问权限或者设置空间仅几日内可见的情况，抓取的信息并不完整。

   最终抓取了 193 个好友，将近 90000 条说说，一共 325MB 数据。抓取到的信

息为 json 格式，按好友 QQ 号存放；每条说说包括发布时间，说说内容，评论内容与时间等内容。

3. 数据处理

首先要从数据中提取我们所需要的信息，在这里我们只使用了发布时间和内容，把每个好友的发布说说的时间排序存放到一个单独的文件中，所有说说的内容单独存放到一个文件中。为下一步分析做准备。具体数据处理包括对时间和对内容两部分。

对时间，我们考察若干个不同的维度。基本思想都基于 wordcount，通过统计不同时间段说说的数目来挖掘信息。比如我们统计所有说说在年份上的分布，可以获得用户活跃程度和跨度的基本情况，从中分析出用户的活跃程度及变化趋势。为了更细致的挖掘信息，可以以月为单位，统计每一个月发布的说说数量，能更准确地反应用户的活跃变化趋势。我们还可以统计每个月哪一天，每个周哪一天发布说说数量最多，来探究用户发布说说和日期的相关性。最后，我们可以统计每天那个时段用户最活跃，来判断用户一天之内的活跃情况。

对内容的分析涉及到自然语言处理，使用了 hanlp 自然语言处理库对数据进行了分词，依然利用 wordcount 来统计词频。去掉停用词后的词频数组就构成了用户说说内容的向量表示，通过去除一些小数据，比如出现次数小于 3 的词，我们就能得到一个关于用户说说内容的基本表示。

所有这些数据都被处理为 json 格式，包括在不同时间粒度下的分布向量和词频向量，共同构成了一个用户的信息，以供可视化模块使用。

4. 可视化

使用 Web 界面实现可视化，网页使用了 BootStrap 和 Vue 框架，数据展示采用了 AntV G2 以及 EChart，实现了使用不同图表及词云来可视化数据的目的。

JSON 原始数据 头

保存 复制 全部折叠 全部展开


```
0:
    year:   2015
    count:  43
1:
    year:   2016
    count:  199
2:
    year:   2017
    count:  162
3:
    year:   2018
    count:  15
4:
    year:   2019
    count:  2
```


JSON 原始数据 头

保存 复制 全部折叠 全部展开


```
0:
    year:   2015
    month:  1
    time:   "2015_01"
    count:  0
    id:     0
1:
    year:   2015
    month:  2
    time:   "2015_02"
    count:  0
    id:     1
2:
    year:   2015
    month:  3
    time:   "2015_03"
    count:  0
    id:     2
3:
```


JSON 原始数据 头

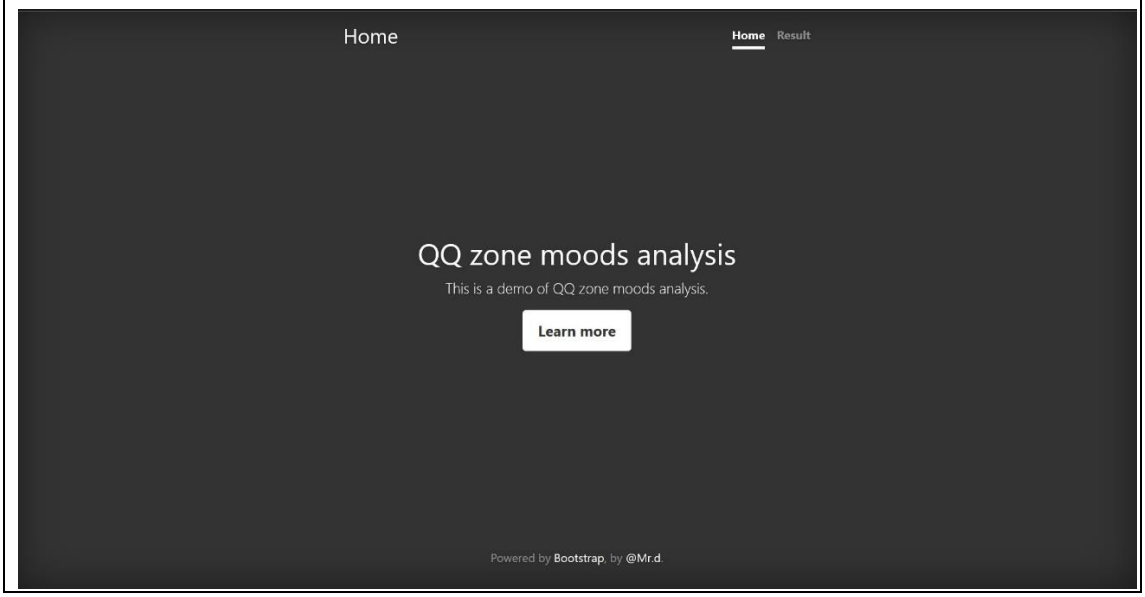
保存 复制 全部折叠 全部展开


```
0:
    hour:   0
    count:  17
1:
    hour:   1
    count:  6
2:
    hour:   2
    count:  6
3:
    hour:   3
    count:  1
4:
    hour:   4
    count:  2
```


JSON 原始数据 头

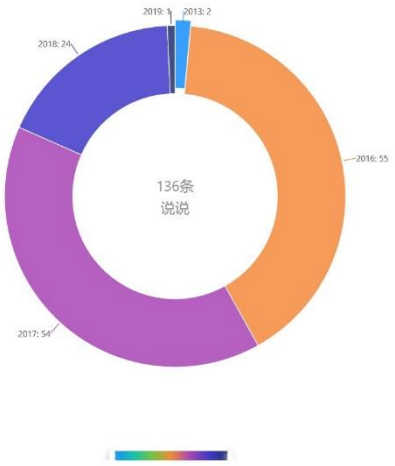
保存 复制 全部折叠 全部展开


```
0:
    name:   "A"
    value:  40
1:
    name:   "App"
    value:  40
2:
    name:   "QQ"
    value:  40
3:
    name:   "h"
    value:  40
4:
    name:   "km"
    value:  90
5:
```

QQ Mood Analysis | 1503345074 | Enter

- Year
- Month
- Day
- Hour
- word cloud
- recommend_time
- recommend_word
- recommend
- reset

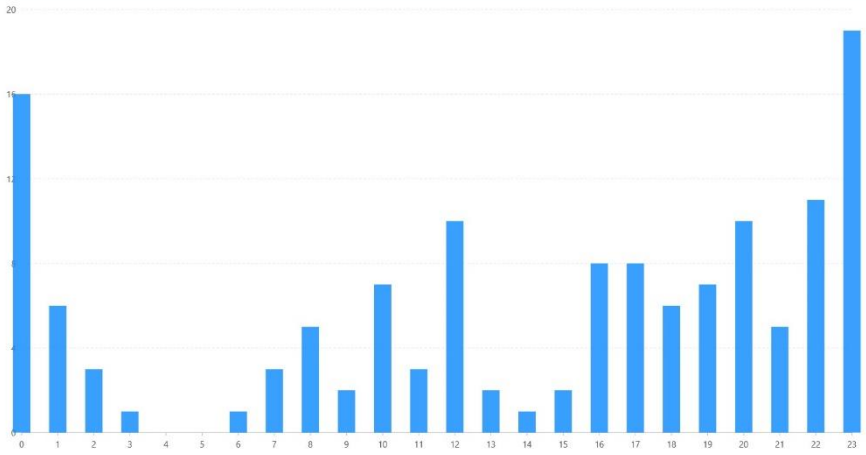


QQ Mood Analysis | 1503345074 | Enter

- Year
- Month
- Day
- Hour
- word cloud
- recommend_time
- recommend_word
- recommend
- reset

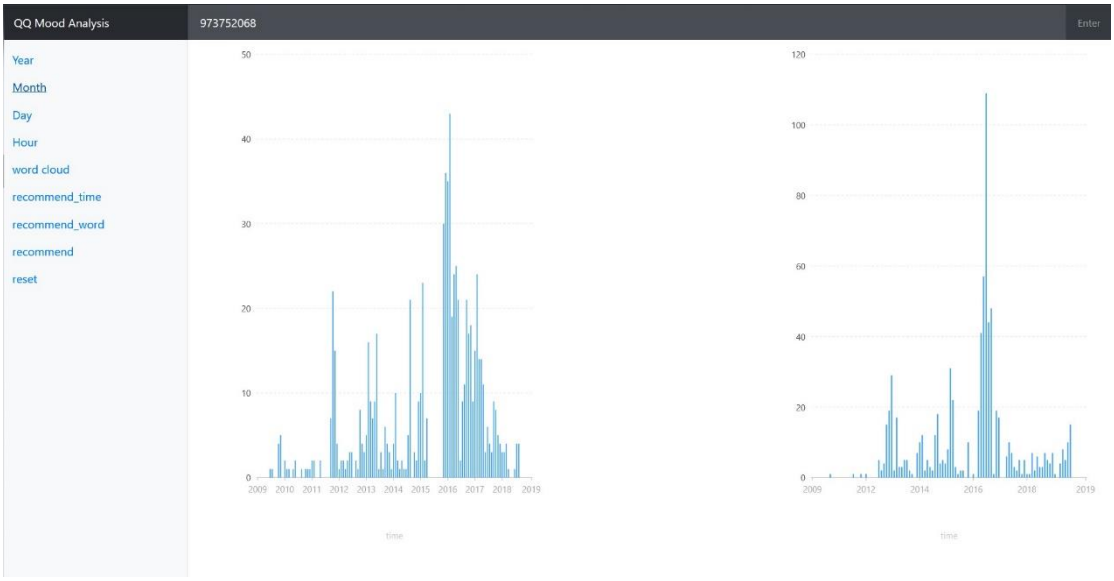
QQ Mood Analysis | 1503345074 | Enter

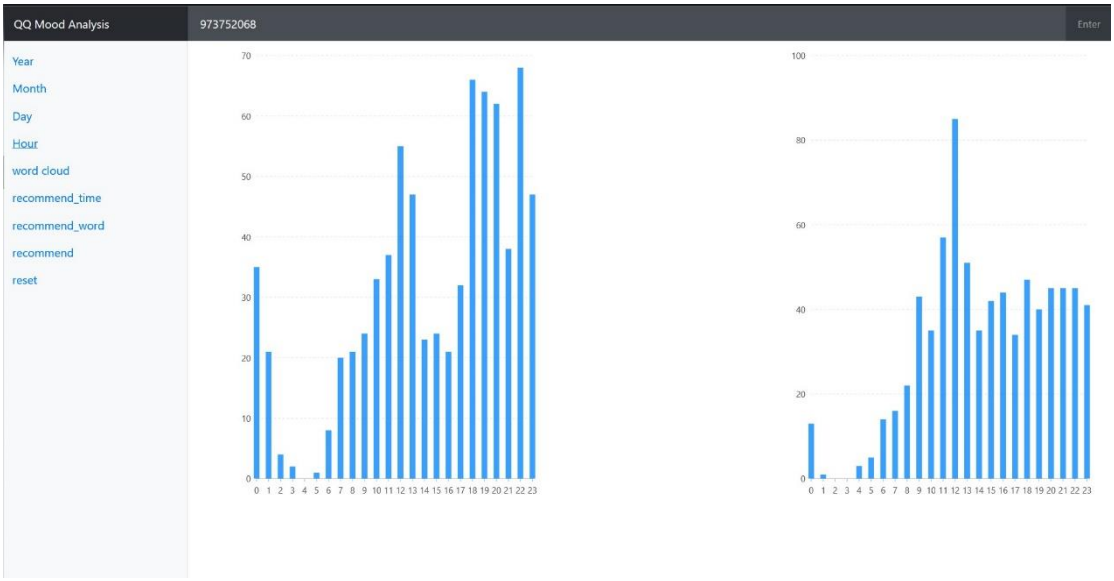
- Year
- Month
- Day
- Hour
- word cloud
- recommend_time
- recommend_word
- recommend
- reset



5. 数据高级处理：相似好友推荐

我们有了针对用户的向量化画像，就可以进行相似度分析，找出和某个用户行为最相似的用户，可以实现类似好友推荐的目的。因为我们的数据已经向量化，因此可以方便地使用各种向量距离或系数来分析相似度，具体实现中我们一共寻找了 18 中不同的计算方式，包括 Pearson 相关系数，欧式距离，余弦距离，Jaccard 距离，Tanimoto 距离，Manhattan 距离，兰式距离，Chebyshev 距离，Hausdorff 距离等等。我们以 Pearson 相关系数为例，此系数用于描述向量之间的相似程度，最终结果在[-1,1]之间，绝对值越大相关性越强；但 Pearson 相关系数与量纲有关，因此我们对其结果进行一定的纠正，计算两个向量的 1 范数的倍数关系，把最终结果绝对值除以此系数作为最终分数，得分最高的相似度最高。最终实现相似好友推荐。下图是某个用户和其最相似用户的比照。





我们可以通过时间和内容相似度分析结果加权作为最终结果显示，也可以单独使用一种评价方式。在时间分析上，我们有年为单位，月为单位以及每天每小时三个评价手段，可以使用参数控制权重。

结论分析与体会：

QQ 空间说说分析属于短文本分析，具有内容短，口语化强，价值不高等特性，因此一般分析价值不高。但基于时间的相似度分析具有较好的准确度和实际意义。爬取数据还抓取了评论信息，但却没有抓到点赞信息，点赞信息分析可用作关系网分析，可留作以后的研究目标。评论信息相比说说内容更短也更难分析，因此没有进行处理。数据分析过程用到的主要技术非常简单，主要是对数据进行分类统计，在数据量非常大的情况下，spark 平台非常适合这种计算。

就实验过程中遇到和出现的问题，你是如何解决和处理的，自拟 1－3 道问答题：

1. QQ 空间对爬虫不友好，抓取数据时难以完全自动化，多次尝试有可能被限制登陆，需要人工辅助登陆。
2. 前端框架入门有一定门框，没有接触过前端需要花较长时间摸索。

3. 可视化工具同理，虽然只是简单使用但仍然不够灵活，没有相应的前端经验只能用较为简单粗暴的方法实现
4. 相似度分析方法很多，原定目标都实现在前端，但界面不友好，且前端压力大，最终在后台处理数据，前端只调用结果。