

微博情绪分析 项目报告

Yuxin Chen Yimu Wan Yong Xu Shumin Wang

January 2, 2016

Contents

1 介绍	3
1.1 数据集	3
1.2 项目目标	3
1.3 分工与合作	3
1.4 源代码与相关材料	4
2 开发过程	5
2.1 环境搭建与配置	5
2.2 数据预处理	6
2.3 开发	6
2.4 部署	7
2.5 数据可视化	7
3 分析结果及展示	8
3.1 微博量-时间	8
3.2 微博情绪-时间	8
3.3 微博情绪-日期	9

3.4	微博情绪-性别	12
3.5	微博情绪-身份	12
3.6	微博情绪-转发量	13
3.7	微博情绪-波浪线	13
3.8	微博情绪词典	16
3.9	表情的积极程度	16
3.10	微博情绪与股市价格	18
4	性能对比	19
5	总结	20

1 介绍

1.1 数据集

我们采用的是数据堂的数据集，涵盖了2012年全年的3 GB左右的新浪微博数据。数据集含有两个文件，一个是人工标注的每个表情对应的情绪分类，共115个表情，另一个则是抓取的微博。微博使用新浪微博API抓取，为用户发表在2012年的137,981条微博。数据提供的信息包括微博id，用户id，时间戳，评论数，转发数，微博内容，用户性别，用户类别，用户粉丝数，用户关注人数，用户互相关注人数。

1.2 项目目标

项目的主题是微博情绪分析，整个项目整体比较发散我们利用手头的数据，进行一些有意思的数据统计和挖掘操作。

利用1.1一节介绍的含有表情的微博数据集，完成两部分工作。

第一部分是对数据集的挖掘工作，分析微博量与时间、微博情绪与时间、微博情绪与日期、微博情绪与性别、微博情绪与身份、微博情绪与转发量、微博情绪与波浪线、微博情绪与扩散难易程度之间的关系。此外我们还期望能够分析微博和股票价格之间的关系。

第二部分是生成我们自己的极性词典，并给每个表情定义一个积极程度。

上述的工作均已完成，我们同时还针对其中的几个程序进行了性能分析，以体现分布式处理在项目中的应用。

1.3 分工与合作

由于我们一共有四名同学共同完成这个项目，需要进行适当的分工合作。项目分工的大体情况如表1所示。

我们小组采用 *Google Docs* 协同编写文档，文档的内容包括各个程序间接口

姓名	分工
Yuxin Chen	环境搭建、分布式预处理、基本应用、数据可视化、幻灯片、报告撰写
Yimu Wan	环境搭建、单机版预处理、基本应用、表情积极度、幻灯片、报告撰写
Yong Xu	微博分词、极性词典构建、报告撰写
Shumin Wang	基本应用、性能测试

Table 1: 项目分工

路径	说明
./data	数据样例
./applications	本项目对微博数据集进行挖掘的所有代码，以及相对应部署的脚本。
./preprocess	预处理的程序，包括了分词和统计积极与消极表情出现次数的两个程序。
./report	用于存放本报告相关的文件
./result	存放所有的结果文件
./result/charts	包括所有绘制出来的图表，以及用于绘制图标的 Python 程序
./utilities	包括了3个实用工具，它们分别用来从 HDFS 中回收结果、给结果排序以及从符合条件的微博中抽样

Table 2: 项目的文件组织

的定义、任务的安排等。代码的管理通过 *Git* 和 *GitHub* 来完成。此外，我们还通过微信群讨论以及线下开会的方式进行了沟通和交流。

1.4 源代码与相关材料

本项目的所有源代码、实验结果和报告相都已在 GitHub 上开源，请访问：<https://github.com/irmowan/weibo-emotions> 获取。此外，课堂展示所用的幻灯片已上传至 Google Docs，[请访问：https://drive.google.com/file/d/0B6-7ExEnjWkuSkp2Rm44UWtla0k/view?usp=sharing](https://drive.google.com/file/d/0B6-7ExEnjWkuSkp2Rm44UWtla0k/view?usp=sharing) 获取。

项目仓库的文件组织如表2所示。

2 开发过程

2.1 环境搭建与配置

在与王耀辉组交流之后，我们决定使用 Ambari 进行集成式部署。我们配置了以下组件： *HDFS*, *YARN+MapReduce2*, *ZooKeeper*, *Ambari Metrics*, *Hbase*, *Spark*。 *HDFS* 为数据存储提供了支撑， *YARN+MapReduce2* 提供资源管理和 MapReduce 的支持。 *ZooKeeper* 解决分布式应用的数据管理问题。 *Ambari Metrics* 提供集群状态的检测， *Hbase* 支撑了前者的数据存储。 *Spark* 是我们主要直接使用的分布式框架。

以下使我们在环境配置过程中遇到的一些问题，以及相应的解决方案。

- 搭建 HTTP 代理。由于集群所在的环境同学校实验室，因此上外网需要使用学号登陆，而每个学号只能登陆3台主机。此外，学校实验室的内网上国外网站下载安装包时很容易受到人为的干扰。为了解决这两个问题，我们在自己实验室的电脑上提前使用 Squid3 搭建了 HTTP 代理，并在集群上配置以便更高效的搭建并方便日后的使用。
- 设置host。首先要给每台机器设置合理的主机名，此处统一使用 Weibo[1-6].Hadoop 作为集群各机器的主机名。然后配置相关的 hosts 方便通讯 (Ambari 会自动给机器配置)，在自己的电脑上也最好配置一下，这样在后面浏览文件目录、查看日志时会很方便。
- 配置机器间的 SSH 。配置后机器间通过公钥私钥的配对就可以验证身份，而无需在输入密码，这使得像 Ambari 这样的工具得以施展拳脚。
- 关掉 iptables 。根据 Ambari 的提示而关闭，目的是为了避免防火墙阻碍集群内机器间的通讯。
- 同步时间。 Hadoop 集群对时间的同步性有一定的要求，机器间的时间间隔不能过大，这会影响到相互之间的通信。分配到集群之后我们发现集

群各机器之间的时间相差最多已经达到了三十分钟，因此我们搭建了一个(Network Time Protocol,网络时间协议)服务器，用它来为集群的机器进行同步。

- 关掉原来在运行的程序。由于拿到的集群并不是全新的，所以之前留下的程序其实还在运转。这时候端口已经被占用，导致新安装的程序没有办法启动，最后就会导致安装失败。Ambari因此我们在发现这个问题之后先手动关闭了之前的程序。

2.2 数据预处理

为了更好的完成协同工作，我们定义了如表3的接口。

预处理有两个主要的分支，一个是统计带有情绪的表情数量的，另一个是对微博进行分词的。预处理的结果中，微博间由换行符隔开，相邻两个数据域则由制表符隔开。两个程序的数据域0-9定义相同，分词的程序中，域10内是由空格分隔的分词结果，而对于统计的程序，域10是由原始的微博内容组成，之后两个域则是其中出现的带有情绪的表情数量。两个分支很好地解决了开发进度的问题，使得不依赖分词结果的程序能够先完成。

分词过程主要采取jieba分词工具，基于spark python的分词程序比预想中要友好，对于分布式的分词程序来说，只要给集群的所有机器上的python配置好jieba库，然后在程序里直接import jieba即可分词，分布式程序的代码和单机版代码基本一致，并不需要编写其他额外的代码。而且相比于单机版程序，分布式程序更加稳定，对于3G的大文件，单机版的分词程序经常会无故中断，没有运行成功过一次。因此最后我们直接使用分布式分词程序对源文件进行了分词。

2.3 开发

开发中主要调用到了 Spark 中的 `map()`, `reduce()`, `flatMap()` 等方法来 MapRe-

0	1	2	3	4	5	6
微博id	用户id	时间戳	评论数	转发数	性别	关注数
7	8	9	10	-2	-1	
粉丝数	互相关注	身份	微博内容	#积极表情	#消极	

Table 3: 数据接口

duce 操作，使用 `filter()` 来过滤无关信息，使用 `saveAsTextFile()` 来保存结果，使用 `collect()`, `sample()` 等方法来手机结果。此外，还使用到了 `leftOuterJoin()` 等函数，在此不再一一列举。

我们在开发过程中遇到的一些问题：

- Spark 默认不支持覆盖写入数据，所以在每一次写入之前应该先删除此前的结果。
- HDFS 中，删除了结果之后，数据默认仅是移动到 `.Trash` 文件夹下，如果不手动清除，一段时间之后就会占用大量存储空间。
- Hadoop 的状态显示并不准确，经常是状态显示成功但是实际上程序运行到一半已经崩溃了，这时候就需要去确认日志，日志在这里是最可信的。
- Python 环境下 Spark 在序列化一个含有中文的元组（tuple）的时候，没有办法正确的显示，需要提前手动序列化。

2.4 部署

我们为每一个程序编写了部署的脚本，并首先在小数据上测试，测试通过后再进一步部署到整个集群上，大大提高了开发效率。

2.5 数据可视化

数据可视化使用 `iPython` 配合 `matplotlib` 来完成，所有相关的代码均存放于 `./result/charts` 中。

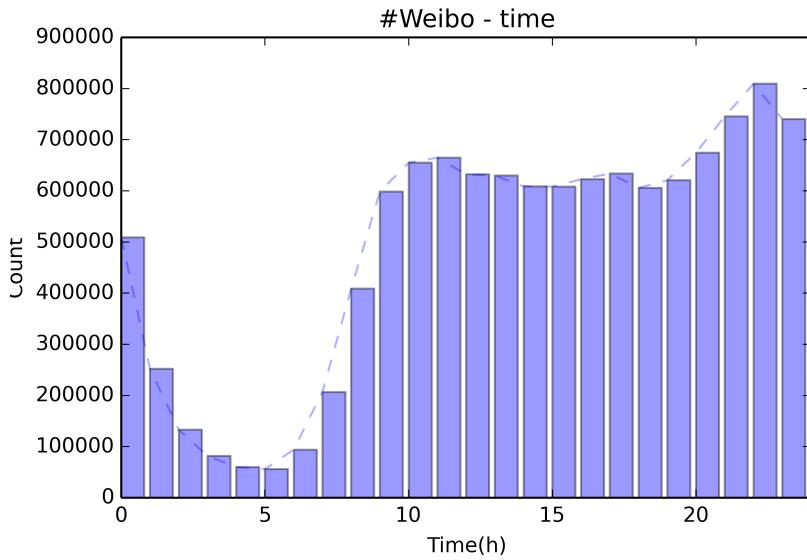


Figure 1: 微博量-时间

3 分析结果及展示

3.1 微博量-时间

在这一小节中我们统计了微博量与时间的关系。程序实现中，首先将每一条微博 map 为 $(HH:MM, emotion)$ 的 key-value pair，其中 HH:MM 表示小时和分钟数，emotion 表示情绪（这里复用了3.2节中的程序）。再对 map 好的 key-value pair 根据键的值来做 reduce。

图1展示了以小时为间隔的统计结果。数据显示，9时-24时微博量相对稳定，高峰出现在22时-23时这一区间内。23时之后微博量开始下降，低谷在5时-6时出现，呈现一个明显的U形，符合大部分人的作息习惯。

3.2 微博情绪-时间

在这一节中我们试图通过统计来揭示微博情绪随时间的变化情况。程序实现中，map 和 reduce 操作在3.1节中已经介绍，这里补充说明 $(HH:MM, emotion)$

的 key-value pair 中 emotion 的计算方法。计算方法如下：

$$emotion = \begin{cases} 1 & npos > 0 \text{ and } nneg = 0 \\ -1 & npos = 0 \text{ and } nneg > 0 \\ 2 & nneg > 0 \text{ and } npos \geq nneg \\ -2 & npos > 0 \text{ and } nneg > npos \\ 0 & \text{otherwise} \end{cases}$$

公式中的 $npos$, $nneg$ 为预处理后统计出来的两个数值, 代表微博中包含的正负表情个数。公式的结果中, 1和-1代表这条微博具有明显的情绪倾向, 0代表微博中不包含含有情绪的表情 (或者表情不在数据集的情感词典中出现), 2和-2代表这条微博的情绪有歧义, 但更偏向积极或更偏向消极。

图2反映了有明确情绪的微博和总微博数量之间的关系。没有明确情绪的微博由上述 $emotion$ 为 0和±2的微博组成。图3反映了积极微博和消极微博数量上的关系, 可以明显的看出, 情感积极的微博总是占据大多数的。为了更好的体现微博整体情感的变化趋势, 我们使用 $PN\ ratio$ 来衡量整个微博的情感。

$$PN\ ratio = \frac{\#\text{positive weibo}}{\#\text{negative weibo}}$$

图4反映了 $PN\ ratio$ 与时间的关系。有意思的是, 整体的图形与微博量-时间的图形非常相似, 这也就意味着, 在深夜的时候, 消极微博占的比重更大。全天来看, 4时开始整体开始往积极方向发展, 而到了15时之后, 则开始往消极方向发展。

3.3 微博情绪-日期

这一小节我们分析每日微博情绪 (使用 $PN\ ratio$ 衡量) 波动, 并以此检验衡量标准是否有效。从[1]中作者的实验来看, 好的衡量标准应该能够准确的反映

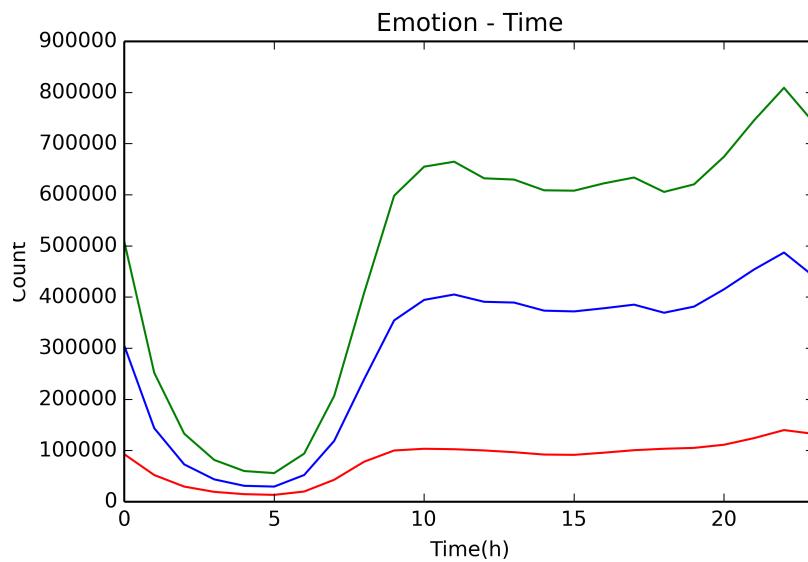


Figure 2: 有明确微博和总微博数量的关系。图中绿色线表示所有微博量，蓝色为积极情绪，红色为消极情绪。

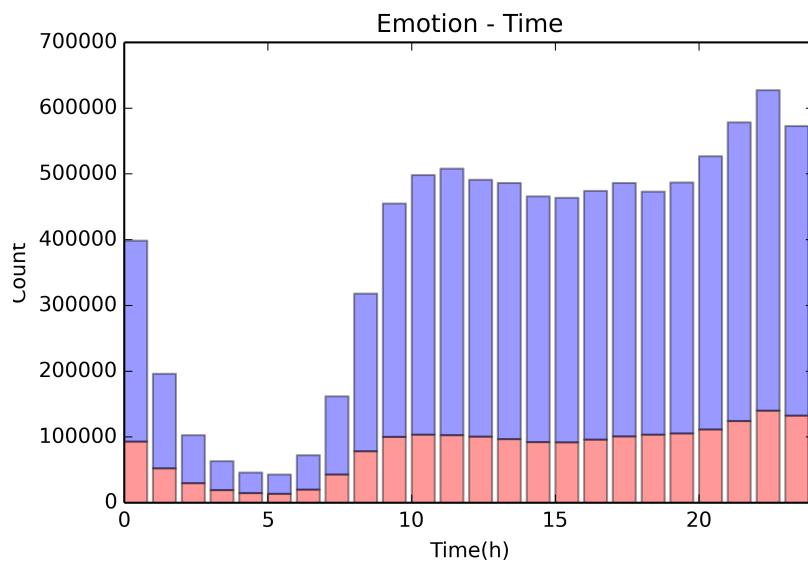


Figure 3: 微博情绪-时间（不包括没有明确情绪的微博）。图中蓝色代表积极情绪，红色代表消极情绪，蓝色柱堆叠于红色柱上方。

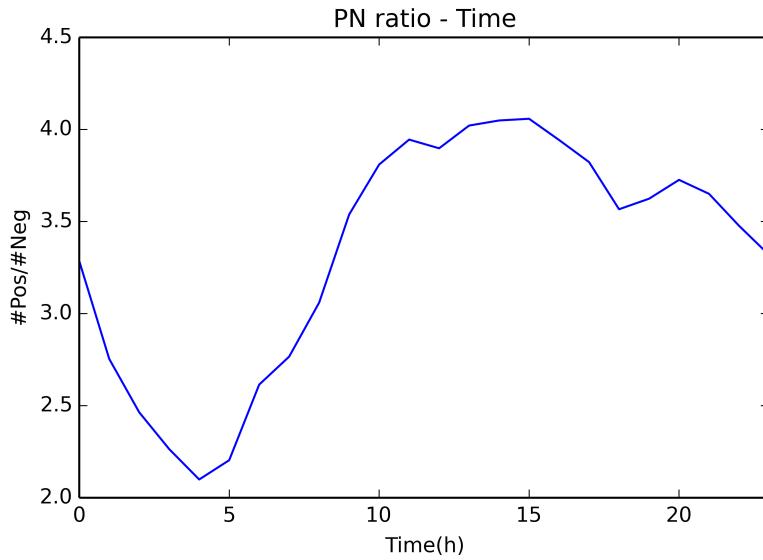


Figure 4: 积极与消极微博比例和时间的关系。

重大公众事件的发生。因此我们在实验之前，预期 $PN\ ratio$ 在春节等节日应该达到最大，在5·12等纪念日时达到低谷。

实验结果符合我们的预期， $PN\ ratio$ 能够如实体现重大公众事件发生后大众心理的变化。积极情绪中：2012年01月01日，元旦，微博情绪达到了全年最高峰， $PN\ ratio$ 为5.87。2012年01月22日，除夕， $PN\ ratio$ 达到了另一个极大值5.76。随后的大年初一，该值仍维持较高的水平，达到5.51。到了2012年01月28日，春节7天假期结束，上班族不得不重返工作岗位，微博相应的出现了较多的消极情绪，这一天 $PN\ ratio$ 为2.96。全年还有两个明显的峰值，一个是在2012年02月14日的情人节，另一个则是在2012年06月01日的六一儿童节。

消极情绪方面：2012年05月12日，汶川大地震四周年，微博上包括名人、机构在内的众多网友纷纷发微博表示悼念， $PN\ ratio$ 为2.29。此外，2012年07月22日和2012年07月23日两天达到了全年的最低谷， $PN\ ratio$ 分别为2.09和2.19。经过采样分析，主要与两件事情有关，一是2012年07月22日重创北京的大暴雨，二是2011年7月23日发生的甬台温铁路列车追尾事故。两件事情的叠加，使得微

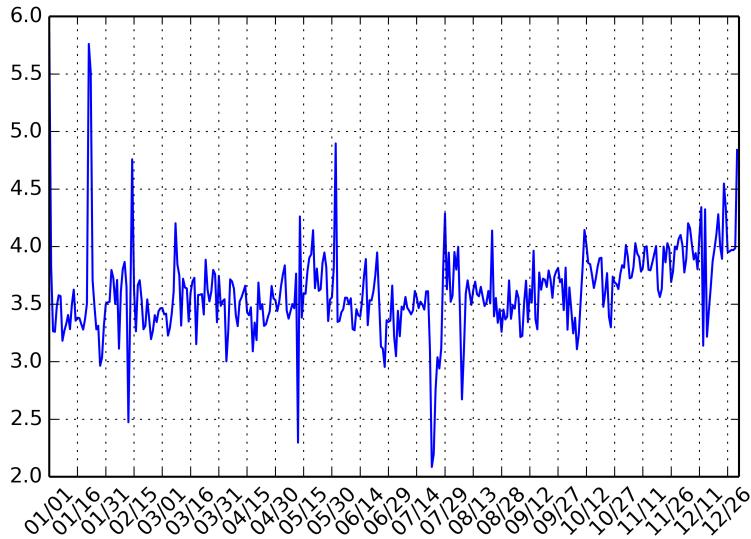


Figure 5: 2012年日微博情绪波动。

博上出现了更多悼念和问责的声音，最终把情绪拉向了全年的最低谷。初次之外，2012年01月28日还出现了一次低谷， $PN\ ratio$ 为2.96，经过分析，与惠特尼·休斯顿去世有关。通过数据还可以类似的找出其于公众事件的关联，在此不再赘述。

3.4 微博情绪-性别

这一小节分析微博情绪在男女之中是否明显的差别。

从数据中看，积极微博的占比在男女性中不相上下，而女性的消极微博占比要多出约2个百分点。

3.5 微博情绪-身份

从图7中蓝色部分（积极的微博）可以看出，最积极的是学校的微博，其次是企业和名人所发。原来预计政府应该在这一项中有最高的比重，但是可以看到，由于政府的微博有很大一部分被划分为无表情一类，推测可能的原因是政

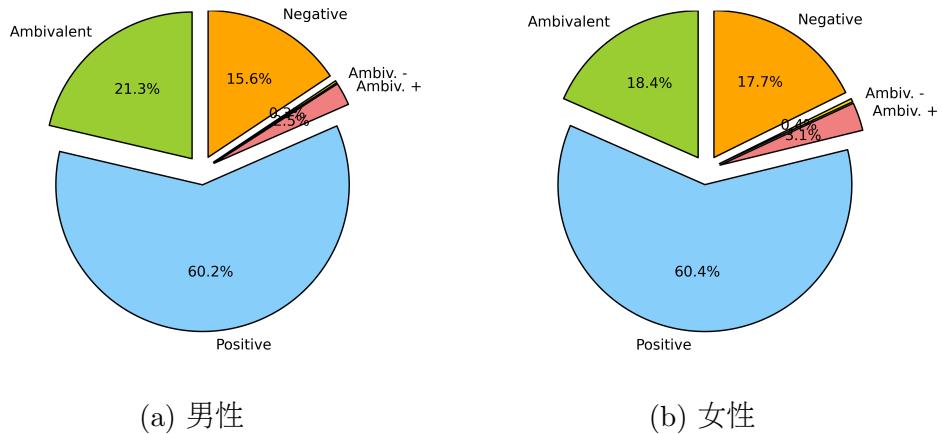


Figure 6: 各种微博情绪在不同性别人群里的分布。（图中蓝色代表积极情绪的微博，绿色代表无情绪，橙色代表消极情绪，剩余两类为有歧义的微博）

府使用的表情更偏向那些没有明显感情色彩的表情。另一方面，从消极占比来看，政府的消极微博占比则相较总体情况要少了很多，由此可以判断政府的微博还是比大部分的用户要更加积极的。除了政府，学校、名人、企业和媒体相对来说负面情绪要少一些，而负面情绪在微博会员这一类别中表现的更强烈。

3.6 微博情绪-转发量

为了检验什么样的情绪更容易在微博上被传播，我们过滤出含有转发符号（“//”）的微博。从图8中可以明显的比较出，积极情绪的微博更容易被转发。

3.7 微博情绪-波浪线

根据经验，人们在发布积极情绪的微博时，更容易加上波浪线（~），比如“么么哒~~”、“好漂亮~~”等。为了检验这一经验是否符合实际情况，我们对此进行了统计。如图9所示，数据显示，含有波浪线的微博积极情占比从60.3%提高到了65.6%，而消极情绪的占比降低了6.4%，符合我们日常经验。

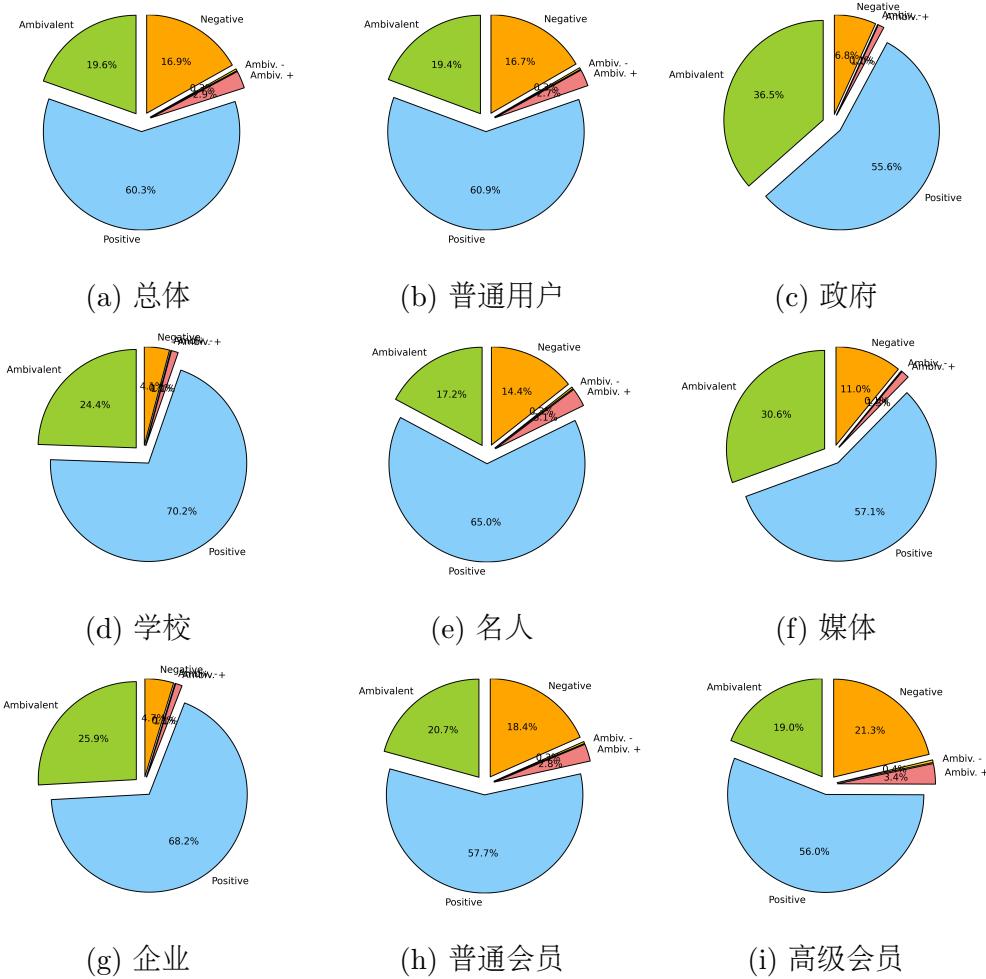


Figure 7: 不同类别用户的微博的情绪分布。

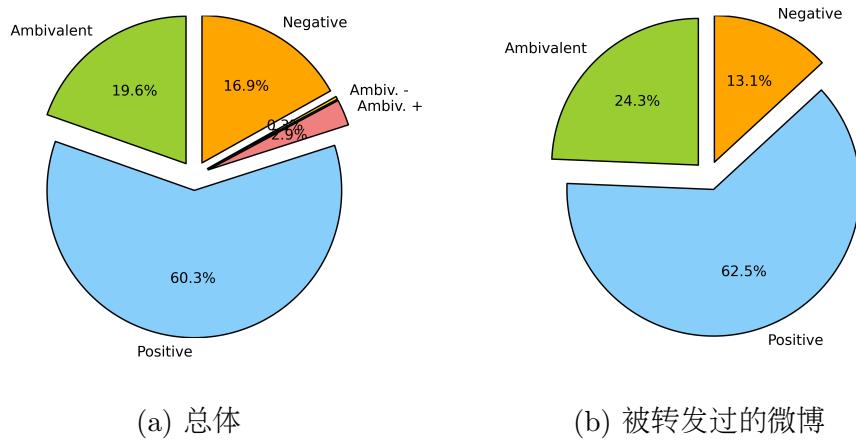


Figure 8: 被转发过的微博的情绪分布。

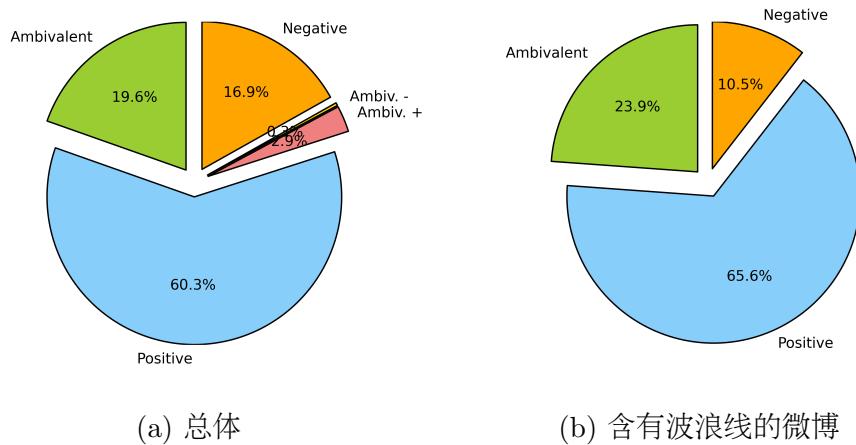


Figure 9: 含有波浪线的微博情绪分布情况。

积极		消极		中性	
词	积极度	词	积极度	词	积极度
新婚	0.87	他妈的	0.20	楼盘	0.50
爱	0.89	艹	0.24	毕业生	0.50
天天开心	0.89	骗	0.39	高管	0.48
非常感谢	0.86	MLGB	0.34	美国	0.53
好样	0.85	睡眠不足	0.29	组图	0.50
情人节	0.71	哭泣	0.34	研究所	0.58

Table 4: 生成的微博情绪词典样例。

3.8 微博情绪词典

我们可以首先利用表情和表情极性词典统计得到的每条微博的情绪，来生成一个描述单词的极性词典，具体做法如下：首先进行分词，这里使用了结巴开源库；之后计算每个词出现的次数和出现在积极微博中的次数，将两者相除，就可以得到一个 $[0, 1]$ 之间的值来衡量这个词的极性。

$$\text{词 } word \text{ 的积极程度} = \frac{word \text{ 出现在积极微博中的次数}}{word \text{ 出现在所有微博中的次数}}$$

在这里，我们定义如果一个词的积极程度大于0.6，那么就认为其是积极的，介于0.6和0.4之间的是无明显情绪的，小于0.4的是消极的。表4列出了最后得到的一些结果。

3.9 表情的积极程度

数据集提供的表情仅用三种值来划分它的极性（积极、消极和无情绪），在此，我们进行了一些统计，得到了每个表情的积极程度。具体做法如下：首先对微博进行分词，分词方法同3.8一节；之后使用3.8一节得到的极性词典，来计算每一条微博的积极程度，用 $[-1, 1]$ 之间的实数来表示；最后，对每个表情计

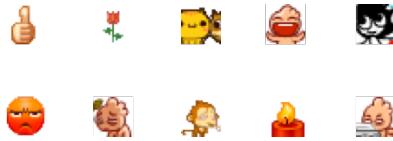


Figure 10: 积极程度最高和最低的5个表情，第一行为最积极的表情，第二行为最消极的表情，从左至右程度依次递减。

积极		消极	
表情	积极程度	表情	积极程度
[太开心]	0.887	[白眼]	-0.769
[圣诞袜]	0.905	[哼]	-0.748
[给力]	0.924	[怒]	-0.855
[来]	0.853	[悲伤]	-0.785
[礼物]	0.879	[鄙视]	-0.782
[可爱]	0.898	[委屈]	-0.710

Table 5: 表情的积极程度样例

算其积极程度。 使用下式来计算每一条微博的情绪:

$$\text{微博的情绪} = \frac{\sum \text{所有出现过的单词的积极程度}}{\#\text{所有出现过的单词}}$$

使用下式来计算每个表情的积极程度:

$$\text{表情的积极程度} = \frac{\sum \text{包含该单词的微博的情绪}}{\#\text{包含该单词的微博}}$$

表5展示了一部分结果。图10是最积极和最消极的表情。

对于这样的结果，我们还能进一步进行迭代，根据新得到的表情权重，再次给微博打上新的标签，此时一条微博不再像原来那样只使用二元的情绪分类，而可以像表情积极程度一样，使用实数化的积极程度。这样，可以多次进行迭代，至微博情绪和表情情绪都基本收敛，能够得到更加准确的结果。

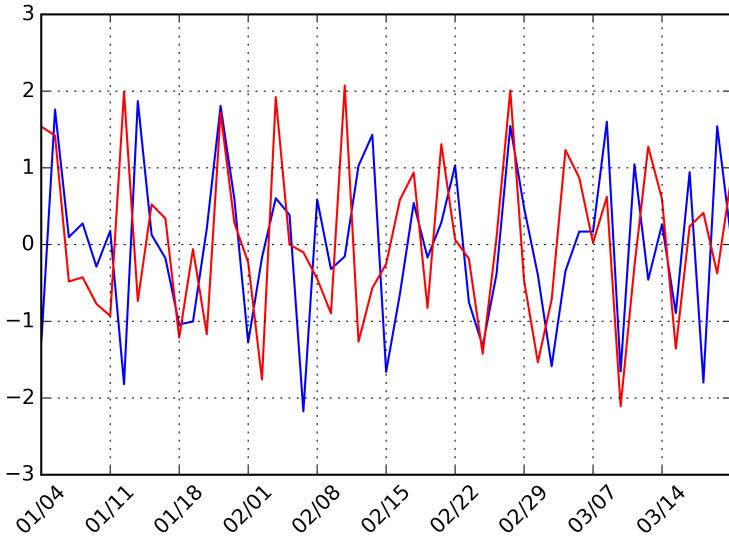


Figure 11: 微博情绪-上证指数。已将上证指数往左移动3个单位。

3.10 微博情绪与股市价格

我们将3.3一节提到的 *PN ratio* 和 上证指数进行比较。如图11所示，红色为相邻两个交易日股市价格之差对应的 *Z-scores*，蓝色为相邻两日微博情绪之差对应的 *Z-scores*。 *Z-scores*的定义如下：

$$Z_{X_t} = \frac{X_t - \bar{x}(X_{t\pm k})}{\sigma(X_{t\pm k})}$$

式中 $\bar{x}(X_{t\pm k})$ 表示了以t为中心，前后k个单位时间的均值； $\sigma(X_{t\pm k})$ 则是这段时间里的标准差。

图4截取了第一季度的数据，可以看到有几个小的区间两者有比较强的相关性。对这个区间进行Granger因果分析，对应1-3个单位滞后的p值为0.26, 0.50, 0.40，并没有办法说明两者的因果关系。参考[1]的实验，二元情绪同样没有达到很好的效果，而应该对情绪进行进一步的细化，使用GPOMS中“平静”这一类别的词语来分析才能达到最好的效果。由于中文情感词典资源的不足[?]，再加上时间的限制，我们在这个项目中很遗憾没有办法继续进一步地分析。

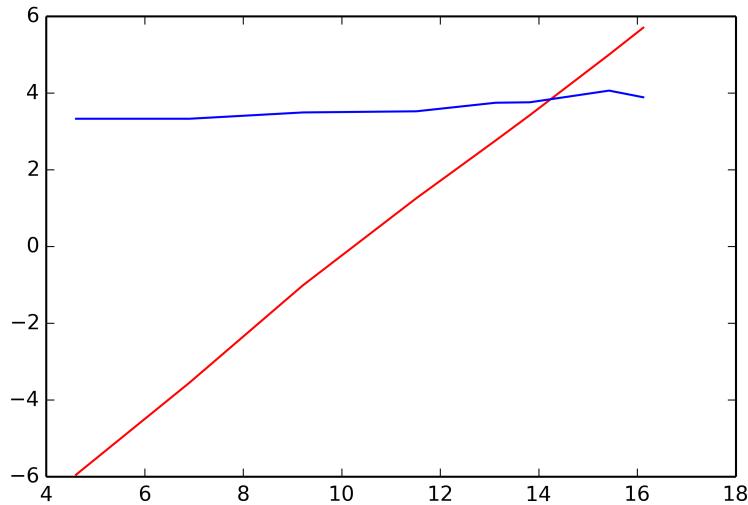


Figure 12: 单机版与分布式版本性能-数据量对比。

4 性能对比

本次性能对比主要是对比单机程序与分布式程序在不同数据集下的效率。

本次性能对比我们使用对微博情绪分类的程序作为测试对象。目的是对比分布式程序和单机程序在不同数据集大小下的表现。

其中，单机程序在服务器上单机运行，分布式程序在6个和单机服务器配置相同的服务器上运行。

两个程序运行时间随数据集增长如图12所示，从图中我们可以看出：

1. 对于分布式程序，无论数据集多大，初期的任务调度，集群间机器通信，任务分配都花了一定的时间，差不多是28s
2. 对于单机程序，运行时间和数据集的大小基本成线性增长。（图中我们对横轴和纵轴都取了log），单机程序是用python写的，没有写多线程，因此该脚本是以单线程运行的，因此所耗时间应该是和数据大小成线性增长的，这符合我们的预期。

3. 从图上的对比我们可以看出数据条目数超过1000000条时，分布式程序所耗时间就快要小于单机程序了，这里分布式计算的优势就已经体现出来了。
4. 对于分布式程序，他的真正进行mapreduce计算的执行时间并不是随着数据量的大小成正比的。这有可能是因为我们executor的数量并不是固定的。对于大数据，会有更多的executor来执行我们所需的操作。分布式的优点也会更加明显。
5. 对于分布式程序，对于小数据的情况，实际的运行的时间并不是非常稳定，两次相同的任务时间差可能会有十几秒。相比于单机程序，分布式程序仍然具有无法匹敌的速度。总的来说，对于大数据文本处理，Spark具有很大的优势，仅仅是它的map功能就可以解决大部分可以并行的文本处理操作，比较推荐在生产环境中实践和使用。

5 总结

这个项目由四人合作，其中大家基本都是第一次使用Spark分布式环境处理分析数据。总计大约花了两周左右的时间，一周部署和熟悉环境，一周撰写源代码，分析数据。

从开始的部署环境摸不着头脑，然后一个个坑踩过来，一个个error的解决让大家对分布式环境的机理更加熟悉。再往后熟悉集群上的各类操作，使用Python脚本写代码，Python作为脚本语言的便捷性也体现得非常好，不用做特别的环境配置，就可以直接使用。

项目也体现了大数据分析的核心思想。虽然3 G的文本数据不算非常大，但是由于极性词典的使用，有些应用的算法复杂度也达到了 $O(MN)$ 的级别，需要思考一定程度的优化(例如根据分词后的词语匹配词典而不应该遍历词典匹配微博中的词)。有些程序随着Part数的变多，时间也在增长，但增长又不是简单的线性关系，这在性能对比部分有所说明。

通过这个课程项目，我们接触了较大规模的数据，了解了分布式开发工具的大致原理和基本使用，在分布式环境下做了一些基于MapReduce的数据处理，得到了不少有意思的结果，总体上收获很多。

参考文献

- [1] Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011.