用微博预测疾病——
基于社交网络数据的流感追踪模型

童凌

2017 年 4 月 25 日

摘要

社交网络产生的数据量不断增大，使得利用其隐含信息进行数据分析和挖掘成为可能。流行性感冒在地区性传播的速度很快，且具有高峰爆发的趋势，且很难对流感爆发的时间和地点进行预测，一旦发病，对公共卫生事业造成的损失巨大。在地区性传染病的监控中，主要依赖于医院的就诊数据。目前只有一部分患者去医院就诊，因此从医院得到的就诊数据具有时效性不强，完整度不高的特点。为了提高监测的及时性与准确性，在社交网络的信息中，根据相关数据以及地理位置信息，对特定病症进行文本分类，构建社交网络数据集。统计社交网络数据集疑似流感病例的数量，统计社交网络信息与实际发病病例的相关度。在保证数据的时效性和准确性的基础上，利用对未来可能发生爆发的地区、爆发规模和爆发时间进行预测。

该预测模型使用新浪微博数据作为数据源，针对流行性感冒进行监控和预测。结合中国疾病预防控制中心给出的传染病的定义，发病模式等信息，归纳出其影响因子结合使用关键字过滤算法以及相关文本分类算法，从海量数据中提取出有效病例，并使用中国疾病预防控制中心的官方数据进行趋势验证。将经过分类算法分类后的数据作为基础，考虑用户社交关系以及地理位置对流感传播的影响，根据历史数据对模型的追踪效果进行评估。

关键词：社交网络，流感，监控，追踪模型

摘要

The amount of data from social network is increasing, which makes it possible to conduct data analysis and mining. In some areas, the spread of infectious disease is rapid and the prediction method is difficult. Once the disease outbreaks, it cause huge losses to public health, affecting the stability and development of the society. The monitoring of regional infectious diseases mainly depends on the hospital visiting data. At present, only a small part of the patients go to the hospital, so the data obtained from the hospital is not timely and complete.

In order to improve the timeliness and accuracy of the monitoring, we classified the text messages into different cases based on geographical information and data from social network. We construct datasets of social network and collect the number of suspected influenza cases, further calculating the correlation between the data from social network and actual cases.

The prediction model uses Sina Weibo data as the source to monitor and forecast the H7N9 avian influenza. Based on the definition of infectious diseases and the information of disease patterns, we conclude the influencing factors. Combined with a keyword filtering algorithm and a number of classification algorithms, we extract valid disease cases from the massive amount of data. Then, we use official data of the Chinese Center for Disease Control to validate the correctness. Based on the classified data, we consider the impact from the social relationships and the geographic location, and evaluate the effectiveness of the prediction model.

keywords: Social Network, H7N9 Flu, Surveillance, Tracking Model

目录

1 绪论

本章回顾了近百年前在美国发生的一场以流感病毒为根源的灾难性疫情，最终蔓延到全世界的故事。文章以此为切入点，同时从身边的生活论述了预防、追踪流感疾病传播和发展的重要性。

1.1 流感——无法预测的威胁

你在生活中经历过这样的情景吗？在乍暖还寒，草长莺飞的春夏时分，外面的天气突然由暖转冷，或是由冷变暖。你听说办公室里的某人打电话请病假了。接着，又有好些同学请假去了医院。城市的大街小巷里，火车和高铁站台上，人们纷纷戴起口罩避免空气飞沫的传播，电视台的频道不断循环播放” 勤洗手，勤换衣” 的公共卫生广告。不幸的是，不知何时开始你也得了流感：发烧、头痛，流鼻涕……最后自己也卧床不起了。

流行性感冒的传染性很强，传播途径多样，常见的途径是通过空气飞沫传播。感染者咳嗽或打喷嚏时，即使流感病毒离开宿主，仍然能在空气中存活 2-8 小时。在此期间与他人接触后就会感染上流感病毒。在 1918-1920 年，全球卫生机构只有很有限的手段应对这场流行病。在 20 世纪期间，全世界各地还发生了至少 5 次其他流感大流行疫情。[5]

流感在世界上发生大流行，主要源于三个方面。一是流感病毒直接感染人类、二是基因重组，即人类流感捕获流感病毒的基因片段后发生基因重组、三是旧病毒株由于人类抗体的失效，再次感染人类。流感病毒分为甲型，乙型和丙型三类，每类病毒含有多种亚型，并在自然界不断演变。尽管医疗机构对于流感的认识不断加深，但现有的流感病毒仍在不断演化、变异。由于流感病毒可能会随时变异，且具有跨多个物种的能力，因此很难预测下次流感将在何时爆发，在何地爆发，以及爆发程度。[4]

除了大流行以外，季节性流感的疾病依然会影响数百万人。可悲的是，大多数人都还没有意识到这种流感疾病有多危险。对幼儿、老年人和具有其它严重症状的患者，感染可能导致严重的并发症、肺炎和死亡。在 21 世纪初期，每年全球约有 300-500 万人罹患重症流感，其中 10% 的患者，即多达 50 万人可能因此死亡。在中国，估计平均每年就有 36000 人死于流感，这比酒精诱发的死亡或交通事故导致的死亡都还要多。每年，全世界约有 5%-10% 的成年人和 25% 的儿童 (图 4) 会患上流感。” 在每年流感季节的高峰期，来我们药房的人有三分之一会询问流感相关

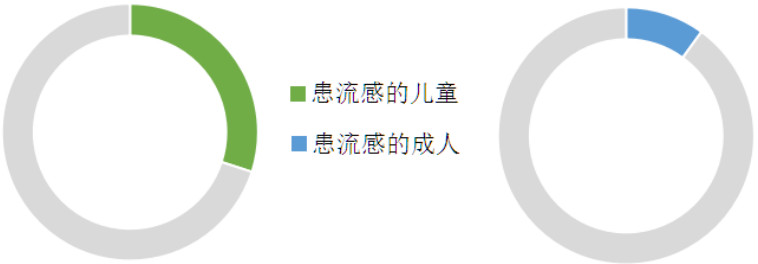


图 4: 每年流感患病人数在全球所有人群中的占比

的症状。”法国药房工作人员 Laurent Rosaz 这样说道。[4]

流感的主要症状是高烧，头痛，肌肉关节疼痛，咽喉疼痛和流鼻涕。对于青少年和成年人来说，流感病毒同样会对个人以及家庭造成损失。一次典型的轻微流感，通常意味着一个人在长达六天的时间里，出现不同程度的工作、学习效率下降。因为流感，中国每年因为生产力下降和病假而造成的损失超过 300 亿元。

1.2 研究背景以及研究意义

H7N9 型流感又被称为禽流感，是一种近年来新发现的传染性流感病毒，近几年来的全球性爆发在世界各国地区造成了严重的威胁。[39] 2013 年 3-4 月，我国各地爆发了大规模的疫情。截至 2013 年 11 月 1 日，新华社报道共有 137 例病例，其中 45 例死亡。[7] 给社会公共卫生带来巨大损失。尽早地预测疾病活动趋势，对预测未来流感的爆发和疾病的防治具有关键作用。而目前针对甲型 H7N9 流感疫情监测的研究较少，特别是使用海量数据预测其流行动态的方法更是罕见。

传统的流行病监控手段通常使用人工方式来收集数据。通常来说主要的监控点在医院。为了监控 H7N9 及其他疾病的发展情况，中国疾病控制预防中心通常在医院就医人群中，对疑似病例进行普查，根据医院得到的信息作为流感发病的调查根据。2013 年我国为了监控 H7N9 的流感疫情，利用全国 13850 个公立医院，收集了数万流感病人的临床数据进行分析和研究。这种方式监测到的数据虽然权威，但需要建立遍布全国的监测网络，且检测——确诊——上报病例的过程具有一定的滞后性。除了使用临床数据进行研究外，其他方式还有通过电话调查，网上普查和通过药房相关药品的销售情况进行估计。显然，这些方法不仅不仅需要复杂的流程，收集的监测数据也严重滞后于流感的发展情况，很难掌握目前疾病在地区性流行的现状，也更加难以完成对未来的疾病预测任务。

Viktor Mayer-Schönberger 在著作《大数据时代：生活、工作与思维的大变革》

中，洞见了大数据对未来产生的深远影响。通过对海量数据进行分析，从中获取有价值的信息的能力，成为社会所需的一种新型能力。同时，作者还提到，如果进行足够深入的信息挖掘，利用大数据背后隐藏的信息，就可以预测未来发生的事件，并对未来事件制定相应的对策。因此，大数据带来的变革将逐渐对我们的生活、工作乃至思维方式的改变做出巨大的影响。因此，探索海量数据背后隐藏的特征信息，对流感相关事件进行挖掘，不失为一种减少流感损失，加强预防的方法。[6]

在大数据预测方面，使用搜索引擎来预测流感的未来发展情况是一种可行的办法 [14]，各大商业机构已对其进行不同的尝试。Google Flu Trends 是 Google 在 2008 年推出的监控流感爆发的系统。[9] 当 Google 搜索引擎中收到大量”Flu”等关键词请求后，对疾控中心的数据进行交叉搜索，评估流感爆发的地区位置和爆发的严重程度。Google Flu Trends 曾经提前 14 天成功预测到一场 H1N1 流感在 2009 年在北美范围的爆发，其精度可以具体到州。由于 Google Flu trends 的预警，相关州县的疾控部门及时采取措施，进入紧急戒备状态，避免了一场大范围的灾难，减少了数百亿美元的损失。百度疾病预测于 2014 年上线，[10] 提供对全国 34 个省区，331 个地级市的 11 种传染性疾病进行未来趋势的预测。从百度提供的资料来看，该产品通过历史数据的规律性数据——如流感具有季节性周期的规律——，同时研究疾病人数与搜索请求数目的相关性来计算预测情况，以搜索情况的数据作为基础，从统计角度验证监控和预测的正确性。

然而，使用搜索引擎来预测仍然具有不足之处。虽然 Google Flu trends 的预测在大部分时期是准确的，但是在 2012 年流感爆发时段，[11]，与官方就诊数据相比，其预测会发生过高估计的情况。《Science》的文章 [13] 认为，其原因是基于一种”big data hubris”的算法优化。例如，对预测模型的过度优化，以及对一些关键词选取的调整——例如，某些疾病的搜索词源于人们在流感季节的关注，而不是真实发病——都会造成其数据的夸大。作为同类型的预测工具，百度疾病预测上线时间较短，在预测疾病数据上还没有体现出明显的优势。

随着社交网络的普及以及移动化设备的广泛使用，现在社交平台已经成为人们生活中不可缺少的一部分。人们可以通过社交网络发布感兴趣的话题，分享生活中的小事，也可以通过社交网络传播自己的情感，通过“点赞”，“转发”或“评论”的方式发表意见。不同的个体之间通过发布的信息作为纽带，形成一张巨大的社交网络。在社交平台中，人们发布的信息量正以惊人的速度增加，用户量也在急剧增长。以新浪微博为例，截至 2016 年 9 月 30 日，微博月活跃人数已经达到 2.97 亿。随着移动通讯网络环境的完善，智能手机的普及，移动互联网渗透到用户生活的方方面面，其用户信息能更加全面完善体现大众随时随地的想法。在博文中，”配

图 + 文字”形式占所有博文的 60% 以上，依然是最主要的博文形式。因此，博文中存在大量文字信息可供挖掘。[37]。当有大量人群在社交网络上发表相关的状态，比如说“感冒了，难受”、“流鼻涕，发烧”等相关症状，某种程度上就能说明流感爆发的可能。社交网络中存在大量与疾病症状有关的数据，构成了一个巨大的数据集。若能根据其症状信息和发布时间、地点中挖掘信息，就能为流行病的发现、监控和预测提供依据。如上所述，社交媒体具有实时性强，覆盖范围广，信息丰富等特点。这些特点为流感疾病的监控提供了相关方法有效的数据支持。

为了提高监测的及时性与准确性，我们可以在社交网络的信息中，根据相关数据以及地理位置信息，对特定病症进行监控。在保证数据的时效性和准确性的基础上，对未来可能发生爆发的地区以及爆发规模进行预测。该预测模型使用新浪微博数据作为数据源，针对 H7N9 进行监控和预测。结合中国疾病预防控制中心给出的传染病的定义，发病模式等信息，归纳出其影响因子。结合多种常用的文本分类算法，将有效病例从海量数据中提取出来，并使用中国疾病预防控制中心的官方数据进行趋势验证。评估文本分类的有效性和准确性，选取最合适的模型，对微博数据进行预测性分类。将经过分类算法分类后的数据作为基础，考虑用户社交关系以及地理位置，对流感传播的影响，根据历史数据对模型的预测效果进行评估。

1.3 本文研究内容以及成果

本文将从几个方面探究社交网络对流行性感冒病例的影响，并归纳出流行病追踪模型。

1. 获取微博数据：编写网络爬虫，抓取社交网络数据并获取疑似流感病例。首先通过关键词过滤的方法，对官方定义的流感病例 [36] 获取其中临床症状的关键字，对抓取的发布信息进行过滤，获得疑似病例的推文及其用户数据。另外，从中国疾控中心 [41] 按不同月份作为统计段，分别获取各种流感病例的官方临床数据。

2. 选取关键词：将所有微博内容切分成词语向量。通过词频统计算法，对所有词语向量进行维度计算，频率计算，按照词频大小选出所有与流感相关的名词，作为候选关键词。

3. 选取疑似病例：分析候选关键词词频及其性质对于流感的影响因素，挑选出 50 个与流感最相关的关键词，在所有微博中选取含有这些关键词的疑似病例。

4. 手工标记数据：选取的疑似病例微博中，从部分微博可以知道确实有人患流感，而部分微博为噪声数据，其中虽然含有关键词，但却没有人患流感。通过人工标记的方式对数据进行分类，作为训练数据的支持。

5. 文本分类模型构建。根据使用文本分类方法，分别使用梯度下降算法、朴素贝叶斯算法、Logistic 回归算法和支持向量机算法，对疑似病例进行分类，比较四种分类方法的结果，选取其查全率和查准率最高的方法，作为模型验证和监控时使用的实验数据。

6. 监控准确度验证。当社交网络数据达到一定规模时，计算实验数据与临床数据的皮尔森相关系数，验证其是否表现出一致性，并计算在各个月份该预测模型的监控表现。

1.4 本文的组织结构

本文共分为七个章节，分别包含内容如下：

第一章介绍了使用“大数据”的广泛使用状况，使用大数据来实现疾病预测的现状以及可能性，阐述了文章的研究内容及其成果，说明了文章的主要组成结构。

第二章为文献综述，说明了目前国内外分别使用搜索引擎关键词和社交网络数据两种不同的数据源来实现疾病监控与预测的方法。并且证明使用社交网络的数据来监控和预测 H7N9 流感是可行的。

第三章定义了相关术语，对文章中需要研究的问题，以及相关专业概念给出了明确的定义。

第四章介绍了整个实验中所用到的所有软件工具。

第五章为作者的工作，是整个疾病监控模型从建立到验证的过程。作者收集所需的数据，使用关键词过滤算法和四种不同的文本分类方法分别提取流感病例，根据得到的反馈结果，选取其中准确率最高的分类方法，并与中国疾控中心的传染病数据在时间维度上进行比较，完成相关性验证。

第六章介绍了在整个实验工作中开发的一些小型工具。包括算法和脚本文件。

第七章对本文进行总结，对课题在研究中的不足进行讨论，对未来进行了展望。

2 研究现状

本章对大数据相关的研究现状，以及近年来使用大数据对流行病进行监测和预测的最新成果进行了总结和综述。

2.1 国内外研究进展

大数据毫无疑问会改变我们现实所处的世界。通过社交网络进行相关事件的预测，早已经成为研究的热点之一。[24] [21] 利用社交网络聚合器来预测相关疾病症状的出现。在文献 [19] 中，作者通过对 Twitter 中的数据进行搜集和分析，发现 Twitter 的发表频率在不同事件频度上均呈现周期性分布，例如，从每天的使用频率来看，人们在白天发表的推文较少，而晚上较多，从而说明了大多数人使用 Twitter 的事件集中在晚上。作者通过文本分类，用户聚类的方式，对用户的工作领域进行预测，并通过该方式，预测不同领域用户的日常作息事件和活动规律。[22] 中，作者利用 Twitter 来预测美国大选的获胜者。收集相关 Twitter 数据，挖掘其中用户发表的推文观点，分析其" 情感度" 来定义选举者受支持的程度，利用回归分析统计法统计出其获胜的概率。在 [33] 中，作者基于对 22 年时间内的事件建立语料库，通过从历史数据中抽取相关模型，对可能发生的近期事件进行预测，并根据当前的热点状况进行相关性分析，预测是否会再次发生故事库中的事件。其准确性得到了验证。其结果以上是使用大数据统计的方式对社交网络活动做出监控预测的研究例子。

在流行病监测方面，国内外目前有两个主要研究方向。第一个方向是以互联网搜索数据作为数据源，挖掘搜索数据背后的信息。[43] [38] [9] [42] 在中国。每年有数亿人次利用搜索引擎查询与疾病有关的名词，因此足够大的样本使得从中反应流行病的情况成为可能。同时与其他数据获取方式相比，从网络搜索信息通常能够及时反应疫情的当期变异，因而从而可以弥补官方数据时效性不强的特点。由于搜索的关键词词频是可以实时统计的，能够和疫情保持完全同步。一些研究也表明互联网搜索信息在流行病预测方面是一种非常有效的方式。[12] [9] [16] [17] 利用网络搜索数据监测的方式更加低成本且快速，准确。因此，搜索数据对于提前对疾病做出预警，对我国的传染病监测以及预防具有重大意义。

然而，仅仅依靠关键词推断疾病的发病率，容易导致预测模型不准确。Google Flu Trends 是 Google 在 2008 年推出的监控流感爆发的系统。Google flu trends 主要依靠搜索关键词信息，利用 Logistic 模型，不断调整发病率与关键词之间的关联

系数。这种方法初期设计时表现很好，可以提前数周乃至数月对流行病进行预测，但是 2013 年《Nature》[11] 发现，Google 在流感高发时段，给出的数据存在夸大行为。例如在 2012 年的流感高发季，Google 预测的疾病感染人数是实际官方记录感染人数的 1.5 倍。文献 [13] 认为, 在关键词搜索方面，其他因素也会同样造成数据的偏差。如在流感高发季节，某些搜索量源于担心感染的公众，而非真实感染的人群。同时，若将过高且不切实际的疫情情况公布给民众，可能会造成社会恐慌，从而造成社会不稳定、谣言盛行等状况。

第二个方向是利用社交网络对流行病进行监控和预测。在社交网络事件中，人们发布的信息，互动中均存在大量有关与流行病有关的信息。对其数据进行挖掘，从而归纳出一般规律，也是目前基于社交网络研究的一个重要方面。

文章 [26] 中，作者分析 Twitter 用户发布的信息包含的行为（比如，提取部分关键数据，如数据中显示避免公共聚会，增强个人卫生情况等行为数据作为研究数据）进行流感疫情分析，使用机器学习的方法，使用支持向量机分类器和贝叶斯分类器对四中不同的表达方式进行分类，并挖掘不同类别之间的相关关系。通过与美国疾控中心的流感数据进行对比，发现研究结果与官方数据的相关度非常高。文章 [34] 中，作者使用 Twitter 相关的 API 抓取了与流感病例相关的推文，并使用 SVM 分类器，分类相关流感样例，将得到的结果与 Google Trends 进行比较，得到的结果与 Google Trends 有很高的相关度。根据分类结果说明，在足够大量的数据下，使用 Twitter 用户发布的信息进行监控，结果优于 Google Trends。在文献 [27] 中，作者跟踪最受欢迎的 49 个区的用户的 Twitter 信息，设定 tweets 的流感评分方法，使用关键词加权过滤的方法，计算每条 tweets 的得分。得到的流感发病结果与 2010 年官方流感样疾病的数据表现出高度的线性相关。在国内，文献 [8] 针对新浪微博，利用概率话题模型 (probabilistic topic models) 对主要健康有关的话题在微博上的讨论情况作了详尽细致的分析，结果表明，在微博上发布的与流感有关的讨论信息出现的事件，与流感发生的事件情况具有很高的相关度。同时，与健康有关的话题讨论频率也与同期临床患病的人数高度相关。文献 [18] 中提出，对于社交媒体上与疾病有关的数据并非全都是疑似病例，部分的媒体关注以及发布的消息，使得以前建立的一些模型精度不断下降。作者由此提出了一种自动区分疑似流感病例与其他无关数据的方法，在 2012-2013 年的疾病监控中取得了良好成果。根据以上研究结果，证明了从社交网络中提取用户行为数据来监控用户感染流行病情况的做法是可行的。

流感的预测是社交网络数据挖掘中的难点，也是目前研究中亟待解决的主要问题。在文献 [23] 中，作者以纽约市的社交网络数据作文实验数据，考虑了社交网络

中的朋友关系和地理位置因素在流感传播时的影响，通过对文本进行 SVM 支持向量机的分类学习，监测与疾病有关的 Twitter 信息。在实验过程中，使用动态条件随机模型，对个体的健康状况进行预测。社交网络中，节点与节点之间的关系复杂，若将整个网络的所有个体情况映射出来，是一件成本非常高的事情，但若在复杂网络中，以连接度作为评估节点重要性，用来衡量疾病传播的标准的话，网络中心的节点要比边缘的节点传播更快，因而应该赋予更高的权重。文献 [20] 在校园中随机选取一组个体，监控被选取个体的朋友网络，通过对临床数据进行诊断总结，发现，在社交网络中发现疾病趋势的事件比个体感染流感的事件更早发现。同时通过相关社交媒体数据证明，网络中心的节点在疾病传播速度上不一定边缘节点更快。

在国内流感预测方面，最新的进展来源于百度的疾病预测系统 [10] 自 2014 年 6 月投入使用以来，百度疾病预测系统对全国 34 个省区，331 个地级市的 11 种不同的传染性疾病进行未来趋势的相关预测。由于百度预测的上线时间较短，目前的研究和实际的预测结果仍然未知。因此，最终的预测效果依然有待商榷。类似的预测系统还包括使用 Facebook 和 Twitter 数据的 sick weather。[15] 因此，使用关键词推断流行病的情况依然有不足之处。

目前大多数文献的研究均以 Twitter 作为数据源，均是以英文作为数据源，缺少以中文为主要数据对象的研究。由于中国互联网用户与非中国互联网使用的社交网络软件有明显的区别，其使用的语言也不同，所以其发布的社交网络信息也存在不同特点。因此针对我国社交媒体的预测方法值得探究。

基于以上目前的综合研究现状，本文采用已有的相关算法模型，针对新浪微博社交网络数据，构建数据集，在我国地区级别上，分别研究并构建流感监控、传播和预测的方法。

3 术语解释

为了解释和说明监测 H7N9 流感的一些关键概念，本章主要说明了一些相关医学、计算机领域的术语，以及在传播过程中与传播因素相关的一些名词概念。

3.1 相关术语

- 社交平台：社交平台是一种由计算机程序构建的虚拟平台网络，在平台上，允许不同的参与者之间发表、评论、分析感兴趣的信息，与他人的信息产生互动。定义社交平台具有以下三个特点：

  1. 社交平台是一种基于 Web 2.0 的应用程序。[31]
  2. 社交平台自身不产生信息，由用户发布各种不同介质所承载信息，如文字，图片或者视频等。[35]
  3. 社交平台根据用户的浏览习惯，个人偏好等差异，为不同用户提供差异化服务，并根据用户的反馈，持续提高提供给用户的信息质量。[32]

- H7N9 流感：H7N9 流感是在 2013 年全球首次发现的一种新型病毒，也是与 2013 年率先发现的一种新型传染性流感疾病。主要通过家禽，鸟类的方式传播，典型的临床症状是高热、咳嗽、气促，并快速进展为重症肺炎。[29] [39] 截至目前，尚无证据表明出现了人传播，也未发现病例的临床表现和疾病愈后发生明显改变。[30]

- 发病率：发病率又称为在一定期间内某疾病新发病的频率，常见于流行病学调查。设 P 为发病率，M_c 为观察期间发生的新病例数，M_a 为同期平均人口数，其计算公式为：

  $$P = \frac{M_c}{M_a} \times k$$

  其中 k 的取值为 100%

- 流感样病例：也被称作流感疑似病例，是指是指患有流感的人的典型症状：一般的症状表现是是发烧、头痛、流鼻涕，同时伴有咳嗽或咽痛等症状，同时缺乏其他官方诊断依据。

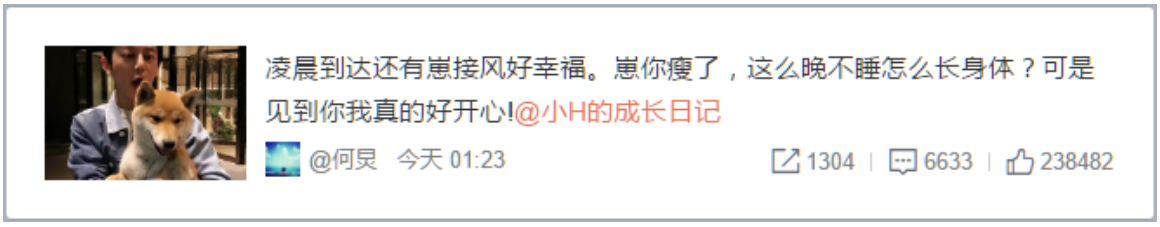


图 5: 微博信息示例

表 1: 微博关键词示例

关键词	微博内容
生病	假装生病没去上学，突然听到爸妈回家声音时的你...
睡觉	好困啊，送完孩子回来睡觉，又要虚度时间了！[委屈]
医院	早上还活蹦乱跳，现在就躺在医院病床上，要戒口只能吃流 -质的东西，可是我好想吃薯片啊。
感冒	现在一感冒，心情就七上八下滴，总爱胡思乱想。如何能克服胡思乱想呢？！[挖鼻][挖鼻][挖鼻]
休息	明天一天的排练哟，抓住休息的尾巴～ [围观][围观] 我在这里:

3.2 问题定义

针对研究社交网络流感的追踪方法时遇到的一系列问题，对研究中出现的关键名词进行定义。

- 微博信息：是指用户在新浪微博社交平台上发表的一条内容，其内容可包含文字、图片、视频、音乐等多媒体信息，以及用户名、发布时间、发布地点、被赞量等附加信息。在本研究中，我们只调查用户发布的文字信息。典型的微博信息如图所示：

- 疾控中心信息：是指中国疾控中心网站 [41] 统计数据中发布的每月法定传染病报告。国家卫生计生委疾病预防控制局在该网站中，公开了全国法定传染病疫情概况，其内容包含甲、乙、丙三类传染病在每月的发病数和死亡数。我们对其数据进行整合统计，以此作为官方数据源。

- 关键词：微博信息中，含有与“流感”等疾病有关的词语，将其称为关键词。一些出现的典型关键词如下表所示：

- 文本分类：在文本信息挖掘和建模时，需要将微博中的信息进行相应分类。在本实验中，文本分类指的是将每条微博区分为两类不同的样本的过程。含有疑似“流感”信息内容的样本，称为正样本；不含有疑似流感信息内容的样本，称为反样本。典型的正样本和反样本的信息如下：

表 2: 正反样本示例

微博内容	样本
嗓子哑的不行，也不知道着凉还是上火，继续拖欠节目 [泪]	正样本
不想说话，头疼的受不了 [生病][生病][生病][生病]	正样本
据说做到这些可以减少百分之九十烦恼，有没有效果，你们试试 [笑 cry]	反样本
好想这列车永远都不要停。真想，不顾一切，去旅行！	反样本

- 数据标注：在本文中，指的是通过机器（或者手动）的方法，正确地从微博信息中区分出正确的正样本和反样本内容，并对其中的正反样本进行标记。例如在表 2 中，通过观察微博内容可以确定，微博信息中是否含有疑似流感信息。在第一条微博内容中，用户患有流感样症状，因此可将其标注为正样本；在第三条微博内容中，用户显然没有流感样症状，因此将其标注为反样本。在“样本”字段中做出标记的这个过程就称为手动数据标注。

- 分类算法：在给定的分类体系下，使用机器学习的相关分类器，根据微博内容自动的确定文本关联的类别。在本文中，即将其分为正类和负类两类不同的数据。从数学角度看，分类算法是一个映射的过程，它将未标明类别的文本映射到已有的类别中，该映射可以是一对一或一对多的映射。常见的分类算法有支持向量机，k 近邻，贝叶斯算法等，本文采用支持向量机、k 近邻和朴素贝叶斯算法进行文本分类，在数据建模一章将会继续介绍。

- 文本分词：分词指的是将一段连续的文字，通过不同的匹配方法，将其拆分成独立词语的一种技术。为了将微博内容以合适的方法映射到高维空间，需要将词语拆分成独立的词语，以提取其特征。在本文中，数据分词指将微博的整句内容划分为一个或者多个独立的词语。

- 相关度：指的是两个事物间存在相互联系的百分比。在本实验中，特指微博中反映出的流感样病例正样本数量与中国疾控中心患病总数的相关系数，采用皮尔森相关系数进行计算。设微博中每月流感样病例正样本的数量为 x, 中国疾控中心的流感患病数据为 y，则相关度计算公式样本共变异数除以 X 的标准差与 Y 的标准差之乘积。其皮尔森相关系数为

$$\sigma_{x,y} = \frac{1}{n-1}(\frac{X_i - \overline{X}}{S_X})(\frac{Y_i - \overline{Y}}{S_Y}) \tag{1}$$

- 文本特征：在本文中，文本特征指的是每条微博信息中与其他微博中能做出显著正负样本区分的词语。比如说，在一条微博数据中若出现“感冒”这个关

键词，那么这条微博是疑似流感样病例的可能性将会大大高于未出现“感冒”这个关键词的微博。那么可以说，“感冒”关键词，是区分正反样本的一个显著的文本特征。其他与流感疾病有关的关键词同样也是显著的文本特征。选取微博数据中合适的文本特征，是数据挖掘中的一个关键问题。

- 训练集、测试集和验证集：在进行数据挖掘时，需要将已有的数据集分为三个不同的集合，分别称为训练集、测试集和验证集。训练集用于建立模型，训练文本分类的特征；验证集用来确定网络结构或者控制模型复杂程度的参数；而测试集评估模型的预测等能力。随机选出一些数据作为模型，从而确定这个规律是否正确。

_id	Content	ID	PubTime
5971721297-M_EaeSNDGik	请不要提醒我，好吗？[眼泪][眼泪][眼泪]	5.97E+09	2016/9/27 19:39
5991041088-M_Efc8XqJtz	10-12毫米大溪地高冷灰墨绿绿珠，被我拍的不像样了	5.99E+09	2016/10/30 9:52
5726404623-M_Eh6lvz7Jo	《漂洋过海来看你》原唱把我唱忧伤，可这位流浪大叔	5.73E+09	2016/11/11 22:47
3125368511-M_EkXfNsJWe	愿你遇到一个成熟的爱人，那个能让你不用在咬着牙逞	3.13E+09	2016/12/7 7:00
2239758733-M_A2B061Ke9	最近身体好差，经常头痛头晕，发烧好左过几日又发烧	2.24E+09	2013/7/31 20:21
2429289144-M_ChK7XohAt	Dégustation“品尝”讲座，主题：奶酪。尝了7个类	2.43E+09	2015/5/14 6:35
2441924490-M_CnBlWOPyZ	世界上最爱我的男人娶了我妈[微笑]和爸爸玩了一下午	2.44E+09	2015/6/21 19:17
2690123423-M_BAcmeFVzB	一定要有下面的对比么！[泪][泪]	2.69E+09	2014/12/10 16:48
1723378513-M_DhxQC2kxk	每年过年都是[泪][泪][泪][泪]脸过敏又呕又烧的[脸	1.72E+09	2016/2/12 22:04
46139572-M_DgWmZdyId	原来这个洗澡的大美妞就是花千骨，我一直以为花是个	46139572	2016/2/8 22:39
5624526886-M_E6YjscTcJ	3块5的手抓饼，里面没有鸡蛋，我的早餐[泪]	5.62E+09	2016/9/6 7:57
1681920200-M_Ad5lHidsr	【网友又曝iPhone缺陷 iPhoen5S容易被压弯】媒体消	1.68E+09	2013/10/8 17:56

图 6: Mongo DB 存储的数据样例

4 相关技术与软件简介

为了总结归纳本次实验用到的相关软件，本章对实验中使用到的所有相关工具进行简介。

4.1 数据库 MongoDB

MongoDB 是一个基于分布式文件存储的数据库。由 C++ 语言编写。旨在为 WEB 应用提供可扩展的高性能数据存储解决方案。MongoDB 是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。其支持的数据结构非常松散，是类似 json 的 bson 格式，因此可以存储比较复杂的数据类型。Mongo 最大的特点是他支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。

在本文中，我们使用 MongoDB 建立并导入与微博有关字段，包括微博发布用户、微博内容、发布时间、微博账号等字段，并利用 MongoDB 数据库来访问和管理所有获取的微博信息。

4.2 编程语言 Python

Python 是一种面向对象的解释型计算机程序设计语言，由荷兰人 Guido van Rossum 于 1989 年发明，第一个公开发行版发行于 1991 年。它是纯粹的自由软件，源代码和解释器 CPython 遵循 GPL 协议。Python 是完全面向对象的语言。函数、模块、数字、字符串都是对象。并且完全支持继承、重载、派生、多继承，有益于

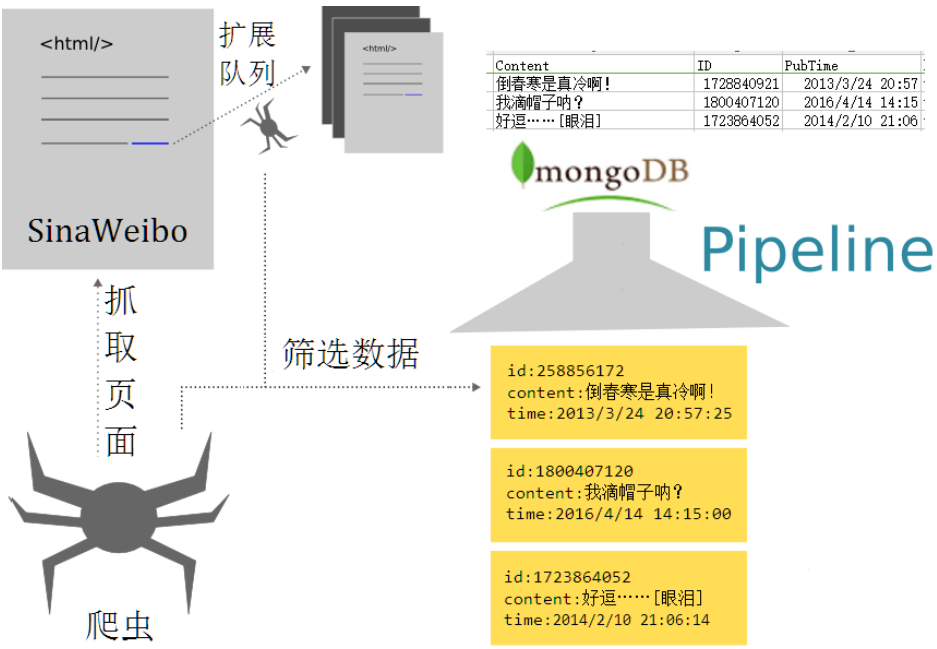


图 7: Python 框架——Scrapy 爬虫示例

增强源代码的复用性。Python 支持重载运算符和动态类型。相对于 Lisp 这种传统的函数式编程语言，Python 对函数式设计只提供了有限的支持。有两个标准库 (functools, itertools) 提供了 Haskell 和 Standard ML 中久经考验的函数式程序设计工具。Python 具有丰富和强大的库，不同的功能被集合到各种不同的应用框架中，对应各种不同的数据处理或编程需求，发挥着灵活的作用。

在实验中，我们实用 Python 中一个简洁实用的框架——Scrapy 框架——用以获取微博源数据 Scrapy 是一个为了爬取网站数据，提取结构性数据而编写的应用框架。可以应用在包括数据挖掘，信息处理或存储历史数据等一系列的程序中。其最初是为了页面抓取 (更确切来说, 网络抓取) 所设计的，也可以应用在获取 API 所返回的数据 (例如 Amazon Associates Web Services) 或者通用的网络爬虫。

4.3 数据处理语言 R

R 语言起源于 1976 年，其主要目的是作为一门优秀的数据分析语言。随着 R 版本的逐渐成熟与稳定，R 在行业里的应用得到了飞速的发展，在欧美已经成为最主流的数据科学工具和数据分析软件。作为一套完整的数据处理，计算以及操作环境，R 语言可以独立完成数据科学中的几乎所有任务，而且可以完美配合其他工具进行数据交互。具体来说，R 有以下几个优点。

1. 数据分析功能强大：R 语言的函数大部分以扩展包的形式存在，方便管理和扩展。由于代码开源，全世界优秀的程序员，统计学家和生物信息学家加入到 R 社区，为其编写了大量 R 包来扩展其功能。这些 R 包涵盖了各行各业中数据分析的方法，从统计计算到机器学习，从生物信息到自然语言处理，从各种数据库语言接口到高性能计算模型，可以说无所不包。

2. 编程简单：作为一种解释性的高级语言，R 语言的编写非常简洁，仅需要了解一些函数的参数和用法。除此以外，R 能够即时解释输入的程序或命令，用户所见即所得。

3. 跨平台：R 可在多种操作系统下运行。

由于微博是一个较为综合的社交网络平台，其用户发布的信息内容中含有大量不规范的格式。我们使用 R 语言的 dplyr 扩展包，对数据进行进一步的清洗和整理。抓取的数据中在字段含有缺失值，我们将缺失值进行插值填补。对于用户发布的表情符号，以及无意义的各种半角字符，我们全部将其删去。抓取的时间信息中，含有诸如“23 分钟前”，“昨天 13:24”，“1 月 25 日 10:05:09”，“2014 年 1 月 26 日”等不同形式，将其格式进行统一。

4.4 MatLab

Matlab 是一款用于数值计算和绘图的开源软件。Matlab 一精于矩阵运算: 求解联立方程组、计算矩阵特征值和特征向量等等。它被工程人员和科研人员广泛使用来进行各种工业和学术上的数值计算和仿真。例如,NASA 使用它来开发飞行器的对接系统;Jaguar Racing 使用它来可视化和分析从 F1 赛车传来的数据;Sheffield 大学用它来开发用于识别癌细胞的软件。MatLab 这样的软件让编写数值处理程序变得简单, 并提供数据可视化的多种方式。

MatLab 另一个值得称道的优点是其强大的画图功能。MatLab 通过调用 GNUPLOT 来实现非常丰富的画图功能。从数值模型到线性回归，从数据拟合和各种离散数据的分布，所有的画图任务 MatLab 都能胜任。

在本实验中，我们通过 Matlab 自带的 SVM 分类模型，Logistic 训练模型和朴素贝叶斯模型的工具箱，将清洗完毕的微博数据按照指定格式，应用到训练模型中，最终得出对应的分类结果，并自动标注出所有微博的正反样本标记。随后采用随机抽样检测的方式，对标注完毕的数据进行正确性检查，以确保微博信息中正反样本分类的准确性。

开发语言	开发环境	数据库	可视化工具	脚本语言
python 2.7	MacOS Sierra 10.12	Mongodb 3.4.2	Mongobooster 3.3.1	MatLab

表 3: 运行环境

5 疾病监控模型

5.1 数据来源

新浪微博是国内大众首选的社交平台，被誉为中国 Twitter。截至 2016 年 9 月 30 日，微博月活跃人数已经达到 2.97 亿。随着移动通讯网络环境的完善，智能手机的普及，移动互联网渗透到用户生活的方方面面，其用户信息能更加全面完善体现大众随时随地的想法。在博文中，”配图 + 文字”形式占所有博文的百分之 60 以上，依然是最主要的博文形式。因此，博文中存在大量数据可供挖掘。

国家疾控中心的科学卫生数据，作为真实患病病例的官方数据，包含了自 2009 年以来全国的所有传染病统计结果，按照地区、年龄、性别、各种特定疾病的发病人数、发病率、死亡人数、死亡率等分类统计数据。这部分临床数据，用于验证与临床数据与社交网络数据之间的关联性。同时比较临床数据与预测模型预测的相关度，对预测模型的准确度进行评估。

因此本文分别采集新浪微博用户及其推文的相关数据，以及国家疾控中心与流感相关的临床数据作为研究对象，用来作为相应的追踪数据：

5.1.1 采集新浪微博数据

挖掘方法：为了获取足量的研究对象，本文采用 Scrapy——一个成熟的 python 爬虫框架——获取预测模型所需的信息。采集数据运行环境如下：

1. 定义爬虫：为了获得网页中的用户信息，建立一个 Spider 抓取全网数据，定义了下载 URL 的列表、跟踪链接的方案、解析网页内容的方式，编写爬虫。
2. 抓取数据：定义并使用过滤条件，使用爬虫爬取的全网数据，使用正则表达式过并抓取出所有的关键字段。
3. 存储字段：编写过滤器，对关键字段进行验证和去重等处理，连接并插入到数据库中。

表 4: Information 表

字段	说明
id	”用户 ID”作为唯一标识
Birthday	用户出生日期
Province	用户所在省份
City	用户所在城市
Gender	性别
NickName	微博昵称
Num-Fans	粉丝数量
Follow-Fans	关注他人数量
Num-Tweets	发布微博数量
Signature	用户个性签名

表 5: Tweets 表

字段	说明
id-tweets	”用户 ID-微博 ID”作为唯一标识
Coordinates	经纬度坐标
Comments	评论数量
Content	微博内容
ID	用户 ID
Like	被点赞的数量
PubTime	发布时间
Tools	发布平台
Transfer	被转发数量

在 MongoDB 数据库中设置 Information、Tweets 两张表，手动建立对应字段，按照表格中定义的字段进行采集。采集完毕后，典型的微博示例数据如下：

5.1.2 采集国家疾控中心的数据

自 2009 年以来，国家疾控中心网在其数据统计栏目中，每月定期公开最近的月度传染病发疫情报告，报告中包含月度全国所有传染病数据，按照地区、年龄、性别、各种特定疾病的发病人数、发病率、死亡人数、死亡率等统计数据。

这部分临床数据，可以用于验证与临床数据与社交网络数据之间的关联性。同时比较临床数据与预测模型预测的相关度，对预测模型的准确度进行评估。

因此，我们从国家疾控中心，按月份和年份两个粒度采集现有的公开数据。设置一张流感发病数据表，字段如下：

截至 2017 年 2 月 25 日，我们共采集 7,836 名用户发表的 1,953,132 条数据。在采集到的数据中，有 213,328 条数据为 2012 年及其以前的微博数据，由于时间间

表 6: 爬虫采集的微博数据内容示例

微博内容	发布时间	疾病信息
老公出差快三个星期了，一个人在家默默发烧中 -_-待周末酣睡两天吧！	2016/9/27 19:13	有
最近各种事情缠在一起，实在解不开。人生总有那个几个阶段，是必须挺过去的。私事太杂，公事又忙不过来。头痛……	2012/7/16 09:30	无
【北京新确诊一名 H7N9 禽流感患者】：北京市卫生局昨天表示，一名男性患者被新确诊为人感染 H7N9 禽流感病例，发病前曾食用过鸽子，目前在地坛医院接受.. 详情：	2014/1/25 09:56	有
【儿童感冒咳嗽食疗大全】宝宝感冒了老咳嗽，睡也睡不好，妈妈看了心疼死，药又不敢给宝宝乱吃，那么，这份食疗大全就可以帮上忙了，妈妈们赶紧收藏吧！	2017/1/27 00:15	无

表 7: Case 表

字段	说明
Time	发病时间
City	地区
Incidents	发病数
IncidentsRate	发病率
Motality	死亡数
MotalityRate	死亡率

表 8: 2013-2017 年度流感统计数据

年份	2013	2014	2015	2016	2017
收集的微博条数	167,933	340,285	302,369	723,648	205,569
流感发病人数	130,410	218,207	198,706	305,703	30,109
流感死亡人数	18	46	15	60	6

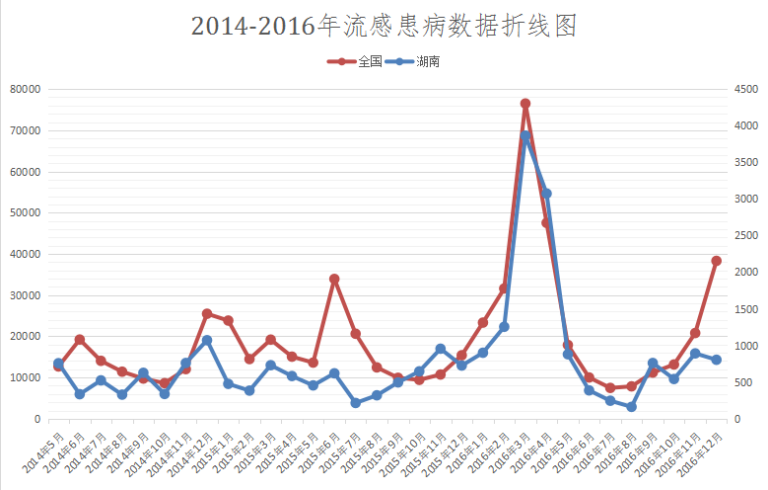


图 8: 湖南与全国流感患病病例统计

隔太久而不作研究。同时，手动采集中国疾控中心自 2013 年起至 2017 年 1 月的每月数据，并将所有微博数据按年份分类，统计如下。

与此同时，为了验证中国流感数据在国家维度和省份维度上是否呈现相同的趋势，选取中国国家疾控中心和湖南省疾控中心的传染病疫情，绘制三年来的病例统计折线图，统计如下。由于湖南省流感患病数据与中国疾控中心统计的患病数据相关度约为 0.87，二者表现出高度相关，故接受该假设。

5.2 数据预处理

由于微博是一个较为综合的社交网络平台，其用户发布的信息内容中含有大量不规范的格式。为了得到高质量的微博相关数据，并进行进一步处理，对于抓取得到的微博信息数据，主要进行以下操作：

- 去重：部分抓取的微博数据为重复数据，我们根据 id-tweets，即” 用户 ID-微博 ID 为唯一标识” 以此作为主键进行去重，删除重复数据。

- 格式统一：在抓取发布时间这个字段时，微博采用了不同的表示方法：对于最近发布的内容，抓取到的时间信息以“今天 hh 时 mm 分”表示；对于 2017 年发布的内容，时间信息以“MM 月 DD 日 HH 时 MM 分”表示，而对于前几年发布的内容，其信息以“YYYY-MM-DD HH:MM:SS”表示。为了进一步按时段统计其微博数据信息，编写脚本，将不同的时间格式统一调整为“YYYY-MM-DD HH:MM:SS”，以此作为标准格式。

- 缺失值处理：在接近 195 万微博数据，有 189 条社交网络数据存在内容缺失，针对这一部分内容，我们将其手动删去

- 离散化：由于微博发布的内容在不同时间段上并不呈现平均分布的特点，将微博内容信息按照月份为维度统计数量，以此作为与中国疾控中心的官方数据的对比基础。

5.3 微博特征信息提取

在微博文字信息中，选取有效的微博特征信息，是构建有效文本分类器的前提。分析各种特征的有效性并选出最有代表性的特征，是模式识别的关键一步。本实验针对每一条微博内容，在微博内容上进行分析，进一步分析并提取相应的微博特征向量。

由于微博信息内容多为连续或离散的句子，要想将其处理为机器可识别的特征向量，一个有效的办法是通过中文词法分析，提取出其特征向量。也就是说，特征向量是中文信息处理的基础与关键。为了提取来源数据中的结构化数据，使用 R 语言包中的文本分词工具。

由于微博中文数据与通常的英文不同，其结构没有像英文一样由空格分开，所以中文文本分词工具一般都涉及到词语切分。RwordSeg 是一套在 R 语言环境下，高效且快速的中文分词工具。RwordSeg 包采用隐马尔科夫模型为基础，可完成词性标注、命名实体识别、新词识别等功能，同时支持用户词典。因此实验采用进行该词法分析系统进行分词。

由于微博信息中存在大量停用词，即不包含任何实际含义的助词、语气词，如”的，了，哦，啊，噢，吗，得” 等，和标点符号，表情等数据，我们将其含有停词的数据删去。经过分词后的微博信息内容示例如下：

最终，将 1,953,132 条微博数据分别分割为词语集合。去除停用词后，对微博中出现的每个单词的词频进行排序，并将其根据不同词性进行分类。我们将其分为

表 9: 经过分词后的微博内容示例

排序	微博内容
1	心理, 箴言, 现实, 是, 污浊, 的, 河流, 要, 想, 接受, 污浊, 的, 河流, 而, 自身, 不, 被, 污染, 我们, 必须, 成为, 大海
2	趣味, 测试, 下图, 中, 第一眼, 看到, 的, 三个, 词, 就是, 对, 你, 的, 描述, 哟, 应该, 蛮准, 的
3	一个, 成熟, 的, 人, 应该, 多点, 宽容, 与, 理解, 对于, 过去, 不, 理解, 的, 以后, 会, 理解, 小时候, 不, 理解, 的, 长大, 了, 会, 理解, 长大, 了, 不, 理解, 的, 年老, 了, 会, 理解, 无论如何, 也, 理解, 不了, 的, 呢, 只要, 人家, 没, 违法, 我们, 就要, 理解, 其, 存在, 的, 合理性, 并, 容许, 别人, 以, 其, 喜欢, 的, 方式, 自由, 存在, 理解, 别人, 就是, 理解, 自己
4	做, 你, 自己, 喜欢, 的, 事, 哪怕, 别人, 都, 笑, 你, 傻, 你, 也, 完全, 可以, 全身心, 地去, 做, 只要, 你, 喜欢, 只要, 做, 这件, 事会, 让, 你, 快乐, 只有, 你, 的, 快乐, 才, 是, 你, 该, 在乎, 的, 别怕, 别人, 说, 什么, 除了, 帮, 你, 快乐, 的话, 你, 都, 可以, 不, 听, 这样, 你, 的, 心, 就, 简单, 了, 事, 就, 简单, 了, 你, 的, 快乐, 就, 多, 了, 哈哈
5	我, 是, 要, 为, 人民, 服务, 但, 不, 可能, 为, 所有, 的, 人, 所有, 的, 民, 服务, 我要, 为, 我, 的, 亲人, 友人, 熟人, 美人, 这样, 的, 民, 服务, 光, 这些, 就够, 我, 忙活, 的, 了, 哪, 还有, 精力, 再, 为, 其它, 的, 人, 其它, 的, 民, 服务, 呢

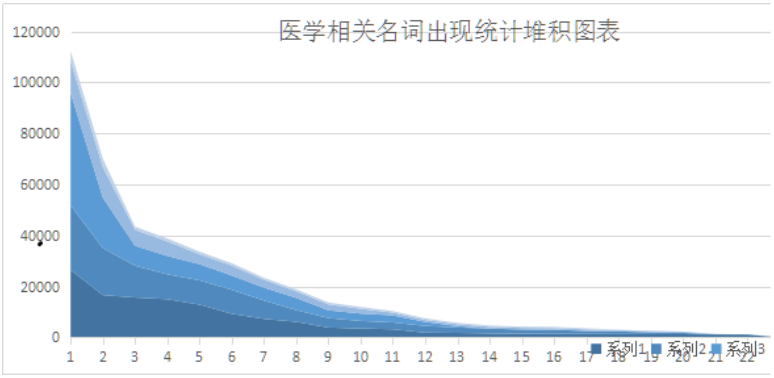


图 9: 湖南与全国流感患病病例统计

五种不同的词性，分别为医学名词、身体感觉、个人情绪、相关动作和疾病症状。并按照各个分类的词频从高到低排序，结果显示：在微博上出现最多的词语如下表所示：

经统计，在所有出现上表关键词的与流感可能有关的微博数据中，至少有 93.3% 的微博中出现了医学名词、身体感觉、个人情绪、相关动作和疾病症状五类症状中的任意一个表格中的名词。也就是说，93.3% 的微博疾病信息均含有下表中的任意关键词。我们采取下表中的数据，在原始数据集中采取关键词过滤算法，筛选出 462,579 条数据，算法如下：

在微博信息中经过去重后，剩余 270,236 条数据，以上数据均含有下表中相应关键词，作为进一步处理的基础。

表 10: 医学名词、身体感觉、个人情绪、相关动作和疾病症状关键词分类

排名	医学名词	频次	医学名词	频次	身体感觉	次数	情绪	频次	动作	频次	症状	频次
1	空气质量	26501	院长	1117	冷	25298	泪	44202	运动	12408	感冒	3885
2	健康	16620	肠胃	1104	累	18412	哭	19830	睡觉	11985	失眠	3165
3	安全	15636	黑眼圈	1056	疼	12469	抓狂	7838	伤害	6259	发烧	1219
4	药	14992	病毒	1040	吐	9768	晕	7212	休息	5672	咳嗽	1129
5	身体	12868	伤痛	1007	寒	9523	烦恼	6324	受伤	3885	头疼	1015
6	症	9278	免疫力	911	温暖	9373	担心	5639	治疗	3852	头痛	951
7	环境	7287	护士	839	痛苦	7180	难过	5051	照顾	3078	发热	764
8	医院	6132	嗓子	804	生病	4577	害怕	4796	熬夜	2582	糖尿病	754
9	死亡	3876	鼻涕	777	紧张	3723	难受	3029	流泪	2306	出血	742
10	维生素	3492	肠道	756	睡不着	2955	崩溃	2916	做梦	1872	心脏病	710
11	患者	3184	水肿	714	不适	2680	大哭	2803	减轻	1038	腹泻	607
12	医疗	1872	脸色	636	泪流满面	2637	折磨	1576	吸烟	769	呕吐	601
13	血管	1741	气血	622	恶心	2152	煎熬	552	吃药	762	贫血	505
14	医学	1573	冷空气	584	疲劳	1868			救人	717	痛经	501
15	保健	1455	病情	576	无力	1607			延缓	682	吐血	481
16	身体健康	1389	大姨妈	539	寒冷	1598			杀菌	657	肚子疼	466
17	病人	1366	疫苗	492	心痛	1158			患有	596	抑郁症	441
18	高温	1314	子宫	483	抑郁	1026			消毒	468	结核病	198
19	血压	1259	医师	455	瞌睡	744			救治	453	肺结核	135
20	中药	1233	急性	427	太冷	541			注射	429	无力感	73
21	基因	1167	诊所	171	苍白无力	60			病倒	92	h7n9	6
22	抗氧化	1138	抗病	150	四肢无力	28					失眠病	3
23					有气无力	20						

表 11: 选取部分关键词后的流感有关的词语分布以及频率

排名	医学名词	频次	身体感觉	频次	情绪	频次	动作	频次	症状	频次
1	空气质量	26501	冷	25298	泪	44202	运动	12408	感冒	3885
2	健康	16620	累	18412	哭	19830	睡觉	11985	失眠	3165
3	安全	15636	疼	12469	抓狂	7838	伤害	6259	发烧	1219
4	药	14992	吐	9768	晕	7212	休息	5672	咳嗽	1129
5	身体	12868	寒	9523	烦恼	6324	受伤	3885	头疼	1015
6	症	9278	温暖	9373	担心	5639	治疗	3852	头痛	951
7	环境	7287	痛苦	7180	难过	5051	照顾	3078	发热	764
8	医院	6132	生病	4577	害怕	4796	熬夜	2582	糖尿病	754
9	死亡	3876	紧张	3723	难受	3029	流泪	2306	出血	742
10	维生素	3492	睡不着	2955	崩溃	2916	做梦	1872	心脏病	710
11	患者	3184	不适	2680	大哭	2803	减轻	1038	腹泻	607
12	医疗	1872	泪流满面	2637	折磨	1576	吸烟	769	呕吐	601
13	血管	1741	恶心	2152	煎熬	552	吃药	762	贫血	505
14	医学	1573	疲劳	1868			救人	717	痛经	501
15	保健	1455	无力	1607			延缓	682	吐血	481
16	身体健康	1389	寒冷	1598			杀菌	657	肚子疼	466

Algorithm 1 使用关键词过滤方法提取与流感有关的微博信息

Require: 原始 1,953,132 条微博内容数据集 S;
表 11 中所有关键词构成的词语集合 P;
Ensure: 所有含有表 11 中任意一个关键词的所有微博信息

```
for i = 0 → length(P) do
    for j = 0 → length(S) do
        if P_i ∈ S_j then
            将 S_j 写入文件
        end if
    end for
end for
return S
```

经过关键词过滤后的数据集为疑似病例数据集，无法确认其病例与临床数据是否精确吻合。如果两者吻合，那么来自微博的流感疑似病例可以在某种程度上反映相关患病情况。由于社交媒体上与疾病有关的数据并非全都是疑似病例，部分的媒体关注以及发布的消息，如" 注意，春天为流感爆发高峰季" 为噪声数据，需要对数据进行进一步处理，从而更准确地提取出流感病例。

5.4 提取流感病例的方法

使用分词工具和上述算法提取出有效数据后，得到的微博结果均包含一个或多个具有流感症状的关键词。但是包含关键词的过滤结果，并不一定都是感染流感的用户。部分微博账号为医学机构账号，虽然内容中包含关键词，但发布的信息为非流感症状信息。如“春季来临，预防感冒的十个妙招”等信息。通过关键词过滤的方法，这些样例也会被归纳到流感疑似信息的数据集中。为了提高数据的质量，去除数据集中无效信息，我们在文本分词和关键词过滤的基础上，对过滤关键词后的结果进行文本分类，以确认真实病例和疑似病例的区别。采用的三种方法分别为 K 最近邻分类算法，SVM 支持向量机分类算法和朴素贝叶斯算法，对三种算法的分类结果进行比较，选取分类准确度最高的算法得到的数据，表示监测结果的数据，并以此作为预测算法的原始数据。

上述三种分类算法均需要构建训练数据集。为了提高训练效果，对数据源中部分用户发布的信息进行人工标注，分为“流感病例”和“非流感病例”两类，作为正样本和反样本。训练数据源中 30% 的数据后，对剩余的数据进行自动分类，进行病例的提取。

使用该分词工具，对 195w 数据进行分词，共得到 294141 个不同的词语，进行词频统计，对微博中出现的每个单词的词频进行排序，并将其根据不同词性进行分类。我们将其分为五种不同的词性，分别为医学名词、身体感觉、个人情绪、相关动作和疾病症状。并按照各个分类的词频从高到低排序，结果显示：在微博上出现最多的词语如下：

使用的典型的提取流感病例分类方法如下：

5.4.1 朴素贝叶斯算法

朴素贝叶斯是构建分类器的简单技术：为相应问题分配类标签的模型，表示为特征值的向量，其中类标签是从有限集中绘制的。训练这样的分类器并不是一个单一的算法，而是基于一个共同原理的一系列算法：所有朴素的贝叶斯分类器假定特定特征的值与任何其他特征的值无关，给定类变量。对于某些类型的概率模型，可以在受监督的学习环境中非常有效地训练朴素贝叶斯分类器。在许多实际应用中，朴素贝叶斯模型的参数估计采用最大似然法; 换句话说，可以使用朴素的贝叶斯模型，而不接受贝叶斯概率或使用任何贝叶斯方法。

朴素贝叶斯的一个优点是，它只需要少量的训练数据即可达到较高的分类准确度。尽管这个算法采用了朴素的设计和非常简单的假设，但朴素贝叶斯分类器在许多复杂的现实世界中都表现得非常出色。在 2004 年，对贝叶斯分类问题的分析表明，对于朴素贝叶斯分类器的显着效果，存在相关理论依据。[25]

。将微博内容看作独立的单词集合，通过训练集，由贝叶斯分类器得到每个单词在不同类的概率大小，构造出特征向量，将其输入贝叶斯模型, 由参数 θ 决定贝叶斯算法假设文档由混合模型产生, 每个类别对应混合模型的一个分量。实现的算法表示如下：

5.4.2 支持向量机算法

在机器学习中，支持向量机 [28]（SVMs，也支持向量网络）是具有相关学习算法的监督学习模型，对需要分类和回归分析的数据进行分析。当给定一组训练示例时（即微博数据），每个训练示例被标记为属于两个类别中的正样本或者反样本，并将新的微博内容标记为正样本或者反样本，成为非概率二进制线性分类器。在数学上，SVM 模型是将样本看做空间中的点的映射，对正反样本来说，在空间上尽可能保持最大化间距，然后将新的例子映射到相同的空间中，并且根据它们所在的方位预测属于正样本或反样本。

Algorithm 2 使用朴素贝叶斯算法进行病例提取

Require: 文本分类微博数据集 C 文本微博训练集 D
Ensure: 各个类别 c 的特征向量 V，各类别条件概率 $condprob$ 和先验概率 $prior$

$V \leftarrow ExtractVocabulary$ 将微博信息映射到高维空间
$N \leftarrow CountWords(D)$ 计算文档单词量
for $c \in C$ **do**
    $N_c \leftarrow CountWordsInClass(D, c)$ 计算训练文档 D 中 c 的词频
    $prior[c][/c] \leftarrow N_c/N$ 计算类别 c 的先验概率
end for
$Textc \leftarrow CombinetoStrings(D, c)$ 将类别 c 下的文档合并成字符串
for $t \in V$ **do**
    $Tct \leftarrow CountWords(Textc, t)$ 计算 c 类别下，单词 t 的出现次数
end for
for $t \in V$ **do**
    $condprob[t][c][/c] \leftarrow \frac{Tct \times N_c}{N}$ 计算条件概率 $P\{t|c\}$
end for
return $V, Prior, condprob$

Algorithm 3 使用朴素贝叶斯算法提取测试集的概率算法

Require: 分类集合 C，特征集合 V，先验概率 $prior$
条件概率 $condprob$, 测试集 d
Ensure: 条件概率最大的类别 Max

除了执行线性分类之外，SVM 还可以采用不同的核函数 (Kernel) 执行非线性分类，将其输入隐含地映射到高维特征空间中。

当数据没有被标记时，监督学习是不可能的，并且需要一种无监督的学习方法，它试图找到数据到组的自然聚类，然后将新的数据映射到这些形成的组。提供对支持向量机的改进的聚类算法被称为支持向量聚类 [2]，并且当数据未被标记时或者当只有一些数据被标记为分类的预处理时经常被用于工业应用中通过。

由于在 SVM 算法模型中，对文本分类属于线性不可分问题，因此我们将文本中的内容分别映射到高维空间，并应用特征向量来表示数据。将上文中计算的词频作为特征向量 C，计算每一条语句中各个特征的权值。将训练集和测试集中的微博内容信息，分别映射到高维空间，在特征空间上，对相应数据进行和训练，并根据训练结果得出正反样本的分类结果。

经过文本标注后，每条文本将其处理为 $< label >< key, value >$ 其中 $< label >$ 为该文本的标注结果（正样本或反样本），key 表示文本特征（即分词后的单词词频），value 表示该语句在相应文本特征上的权重。经过迭代训练后，我们得到较稳定的收敛模型，并对测试集的特征向量放到训练集上，进行相应测试。使用 SVM

算法进行病例提取的相关算法如下：

Algorithm 4 使用 SVM 算法进行病例提取

Require: 特征向量矩阵 C，微博数据集 D
Ensure: 微博数据集 D 中的所属类别
1: $V \leftarrow ExtractVocabulary(D)$ 将微博信息映射到高维空间
2: **for** each $c \in C$ **do**
3: $W \leftarrow margin(c)$ 计算各个类别的支持向量
4: **end for**
5: 初始化类别距离为无穷大 $Dist\{t|c\}$
6: **for** each $t \in V$ **do**
7: **for** each $w \in W$ **do**
8: **if** $Dist\{t, w|c\} \leq Dist\{t|c\}$ **then**
9: $Dist\{t, w|c\} = Dist\{t|c\}$
10: **end if**
11: **end for**$map.put\{t, c\}$
12: **end for**
13: **return** $map\{t, c\}$

5.4.3 Logistic 分类算法

Logistic 模型又称为，逻辑回归模型，其通常与因变量的回归模型相关。通常来说逻辑函数，我们使用多项式变量设定初始值，并将其带入相应特征向量中，然后初始化每一项多项式的参数，以训练出最优化的回归系数。Sigmoid 函数通常表示如下：

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

其输入为

$$z = \sum_{i=0}^{n} w_i x_i \tag{3}$$

其中 w_i 为待训练的参数，x_i 为特征向量的分量。采用优化函数的相应算法，我们可以求出 Logistic 函数在给定数据集下的最优参数，并利用其函数对微博进行预测分类。在算法中，我们根据向量矩阵的特征，设定多项式参数，并使用相应的回归算法最小化价值函数，计算 $h_\theta(x)$ 的函数值。将 $h_\theta(x) \leq 0$ 的样本标记为负样本，将 $h_\theta(x) > 0$ 的样本标记为正样本。

Algorithm 5 使用 Logistic 算法进行病例分类

Require: 特征向量矩阵 C，微博数据集 D
Ensure: 微博数据集 D 中的所属类别
1: $Initialize h_\theta(x)$ 初始化逻辑函数
2: $InitializeCostFunction$ 初始化价值函数
3: $FindSigmoid(x)$ 找到最佳的分类回归系数
4: **for** each $c \in C$ **do**
5: $Calculate h_\theta(c)$ 计算各个类别逻辑斯蒂函数值
6: **if** $h_\theta(c) \leq 0$ **then**
7: $List \leftarrow c$ 将该微博内容标记为正样本
8: **end if**
9: **end for**
10: **return** List

5.5 分类结果算法比较

实验时，从经过人工标注的 44947 条数据源中，选取 25% 的样本作为测试集，75% 样本作为训练集，经过交叉训练后，得到较稳定的模型，测试数据训练在四种模型上的准确率如下表所示。

在微博分类问题中，为了衡量其预测算法与真实病例的匹配效果，我们采用”查准率”(Recall),” 查全率”(Prcision) 和 F1 值 (F1-value) 三个机器学习的分类参数作为分类模型的性能度量标准。对于二分类问题，我们将样例根据其真实类别与分类器预测的类别组合，分为四种不同的情况：

- 真正例 (True Positive)：微博内容中含有流感信息，且分类器正确地将其识别为正样本的情况。
- 假正例 (False Positive)：微博内容中没有流感信息，分类器错误地将其识别为正样本的情况。
- 假反例 (Frue Negative)：微博内容中含有流感信息，分类器错误地将其识别为负样本的情况。
- 真反例 (Talse Negative)：微博内容中不含流感信息，分类器正确地将其识别为负样本的情况。

其混淆矩阵可定义如下：

表 12: 分类结果混淆矩阵

真实情况	分类结果	
	正例	反例
正例	TP(真正例)	FN(假反例)
反例	FP(假正例)	TN(真反例)

表 13: 四种文本分类模型训练值对比

Bayes	Precision	Recall	F1	Logistic	Precision	Recall	F1	Support
Positive	0.85	0.88	0.86	Positive	0.86	0.94	0.90	1019
Negative	0.78	0.72	0.75	Negative	0.88	0.74	0.80	590
Avg/total	0.82	0.82	0.82	Avg/total	0.87	0.86	0.86	1609
Linear SVM	Precision	Recall	F1	Gradient Desc	Precision	Recall	F1	Support
Positive	0.90	0.90	0.90	Positive	0.92	0.92	0.91	1019
Negative	0.83	0.83	0.83	Negative	0.83	0.86	0.80	590
Avg/total	0.87	0.88	0.87	Avg/total	0.89	0.89	0.89	1609

查全率 R 和查准率 P 分别定义为

$$R = \frac{TP}{TP + FN} \tag{4}$$

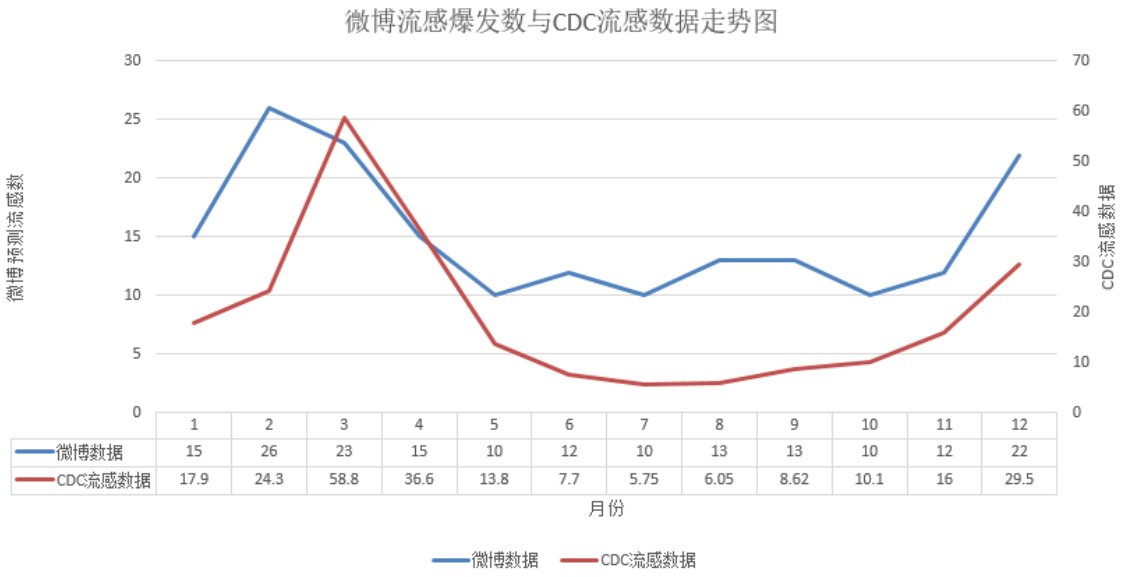
$$P = \frac{TP}{TP + FP} \tag{5}$$

为了综合考虑分类器在样本总体上的查全率和查准率的性能反应情况，通常采用 F1 值衡量二分类模型精确度。设 N 为被检测的样例总数，则 F1 值的度量方法为：

$$F_1 = \frac{2 \times P \times R}{P + R} = \frac{2 \times TP}{N + TP - TN} \tag{6}$$

在使用相同训练数据的情况下，四种分类算法得到的查全率和查准率，以及 F1 值如下表所示：

通过该表可知，使用 SVM 分类算法的准确度较高，达到了 0.87。而在四种方法中，查全率和查准率这两个相关参数上，梯度下降分类算法的准确度较好，接近 0.92。使用综合评估 F1 值的方法进行计算，得到 F1 值在梯度下降分类算法上的值最高。因此，使用该分类模型进行后续微博的病例数据提取，将其划分为正样本的结果作为病例数据，验证社交网络数据的有效性。



	1	2	3	4	5	6	7	8	9	10	11	12
微博数据	15	26	23	15	10	12	10	13	13	10	12	22
CDC流感数据	17.9	24.3	58.8	36.6	13.8	7.7	5.75	6.05	8.62	10.1	16	29.5

图 10: 微博流感数据与中国疾控中心流感数据走势图

5.6 与中国疾病控制中心的数据验证

为了证明在微博数据上提取的流感疑似病例的有效性，使用中国疾病控制中心的流行病疫情数据与经过分类筛选后的数据进行同步验证，得到二者按月份统计，在不同时间段的有效性病例图趋势如以下折线图所示。

通过二者计算皮尔森相关系数 ϵ 约等于 0.7078，可以得出结论: 来自社交网络的流感发病率数据与国家疾控中心的数据，拥有很高的相关度。基于以上实验和二者数据变量的关联度验证，证明了二者数据具有关联关系。

5.7 结论

社交网络中存在大量与疾病症状有关的数据，构成了一个巨大的数据集。社交媒体具有实时性强，覆盖返回广，信息丰富等特点。这些特点为 H7N9 疾病的监控提供了相关方法有效的数据支持。根据其症状信息和发布时间挖掘相关信息，为寻找其中反映相应流感症状的监控和预测方法提供了有效依据。

由于临床数据在统计相关流感病例时，具有一定的滞后性，且延迟过程期间就有可能由于疫情爆发，不能很好地对疫情进行预防。而社交网络的数据没有延迟，而且获取方法更加容易，因此基于社交网络的流感疫情监控方法与传统的真实病例监测方法相比，具有更高的时效性。在微博信息数据量足够大的情况下，可以在一定程度上反应相应的流感疫情。

6 实用工具的开发

6.1 新浪微博爬虫

为了获取足量的研究对象，本文采用 Scrapy——一个成熟的 python 爬虫框架——抓取数据 [40]

使用 Scrapy 编写 Python 爬虫分为以下几个步骤：

1. 构建 http 请求

   为了向服务器发起访问，构造 HTTP 请求，获取相应 Cookie。由于微博主页面 (http://weibo.com.cn) 加载了大量 JavaScript 脚本，使用 Ajax 异步请求获取数据，所以不仅编写爬虫困难，而且爬取相应的数据需要更长的时间，手机微博 (http://weibo.cn) 站点没有 javascript 相关代码，只有纯 HTML 页面，爬取起来相对容易，所以我们爬取微博手机 web 端的页面。我们构造的 http 请求各部分如下：

   为了避免新浪微博的反爬虫机制，在构造每个 cookie 时各自使用不同的账号密码登录。指定 https://login.sina.com.cn/sso/login.php 为登录页面，构造不同的 user-agents，模仿不同的手机设备登录。构建线程代理池，每次从代理中随机抽取一个 ip 来登录。通过以上方式，来达到模拟人工登录的效果。

2. 抓取全网数据

   登录成功后，首先遍历用户发表的所有推文，抓取所有与推文有关的 html 网页，遍历完毕某个 id 用户的所有文章后，对其关注者和粉丝进一步进行搜索。算法表示如下：

3. 数据过滤和清洗

   根据获得的个人信息页面，微博内容页面，粉丝列表和关注者列表，使用正则表达式将符合要求的数据 (见上表) 进行过滤，编写数据库命令，写入数据库中。

   与该研究无关的字段，以及出现数据缺失的字段均为无效字段，在数据清洗时应删去。与当前时间相隔 5 年以上，时间太长的字段不是研究的主要对象，在数据清洗时也应删去。

Algorithm 6 抓取微博数据集数据

Require: 任意一个微博账户 id
Ensure: 微博数据集 S
新建一个队列 $List$
$List \leftarrow id$
while $List \neq NULL$ **do**
    构造 Http 请求，访问 $List$ 中第一个 id 的首页
    抓取其微博信息 $info$
    $S \leftarrow info$
    获取其所有粉丝的 ID_1，所有关注者的 ID_2
    $List \leftarrow ID_1$
    $List \leftarrow ID_2$
    $Pop(id)$
end while
    return S;

6.2 文本分词工具

在微博文本分词过程中，由于大部分微博信息为中文，其结构没有像英文一样由空格分开，所以中文文本分词工具一般都涉及到词语切分。RwordSeg 是一套在 R 语言环境下，高效且快速的中文分词工具。RwordSeg 包采用隐马尔科夫模型，可用于人名、地名识别、关键词提取等相关领域，在试验中我们采用其库函数，完成相关分词功能，伪代码如下：

Algorithm 7 使用 R 语言相关包工具完成文本分词

Require: $RwordSeg$ 分词工具
    微博数据集合 S
    停用词词语集 P
Ensure: 分词后的微博数据集合 S'
加载 RwordSeg 库
读入文本，将文本转换为可以分词的字符
剔除微博数据集 S 中的 URL 链接，停用词 P
将分词结果转换为向量
统计所有词语出现的词频
排序并输出结果

7 总结与展望

本章对实验相关的流程和实验作了总结，并讨论了在实验过程中的不足之处。

7.1 本文讨论与总结

为了使用新浪微博的社交网络数据来追踪流感疫情的发展模型，我们在实验中编写网络爬虫，抓取大量社交网络数据。通过统计词频，选取与流感可能相关的词语，并切分为词语向量，通过词频统计算法进行统计。在所有微博数据中过滤出含有上述词语的微博内容，划分训练集和验证集，对部分数据进行人工标注正样本和反样本（即从微博信息中是否能看出相关流感信息），并分别采用朴素贝叶斯算法、支持向量机算法、Logistic 回归算法和随机梯度下降算法，构建文本分类模型。统计并选取其查全率和查准率最高的方法，作为模型和验证时的相关数据。在社交网络数据达到一定规模时，计算社交网络数据和中国疾控中心二者的关于流感的相关性数据，绘制折线图，统计二者的相关性。结论显示，社交网络数据和临床数据吻合度很高，因此，微博数据的有效性和高时效性为流感的预测和传播提供了有效的数据支持。

本文采用基于新浪微博的相关数据，成功构建了流感疫情的追踪模型，为流感的监控和预测提供了一种切实有效的手段。一方面，建立该模型有助于及时发现流感爆发疫情，从而采取措施进行控制，除此以外，当某些地区可能出现大规模疫情时，该实验提供的追踪模型可以对其进行有效干预。整个实验流程相应的流程图如下所示：

7.2 实验的不足之处

微博内容数量问题：在实现相关分类时，虽然抓取了超过 2,000,000 条微博数据，但与流感疾病相关的微博内容仍然不足。由于新浪微博的反爬虫机制和硬件设备的限制，无法做到对更大规模数据的爬取。若能实现对千万级别以上的数据分析，微博中反应流感的情况可能会与 CDC 的统计数据具有更高的相关性。

关键词过滤的选取问题：在关键词过滤的过程中，本次试验首先从 294141 组词语中，挑选出 124 个与健康有关的词语。由于该过程为手工挑选的过程，没有一个统一的算法完成这个，不同的人有各自的挑选标准，从而造成挑选出的关键词语不同，从而影响最终的正样本数量，进一步影响结果。在文献 [8] 中，作者使用了概率主题模型算法来寻找与健康相关的词语。作为发现在大规模文档中隐藏主题结

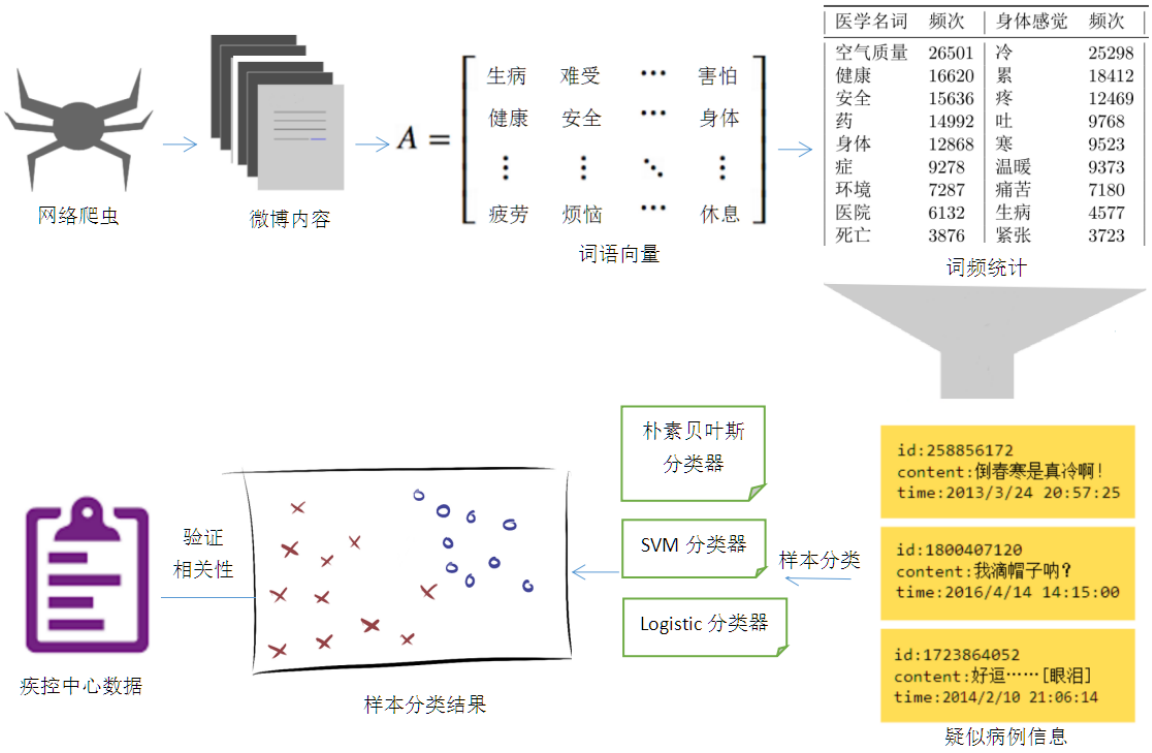


图 11: 实验流程图

构的有力工具，这个算法定义主题 (Topic) 是固定的词语的概率分布，并使用随机算法统计每个词语可能出现的概率。主题建模的中心问题就是利用看到的文档推断出隐藏的主题结构，具有更高的科学性，当样本数据越大，其统计出的模型概率越准确。因此，若实验采用概率主题模型算法来寻找与健康有关的词语，可能会筛选出不同的关键词，其寻找出的流感样症状微博数据也会更加接近真实微博样本数据。

分词精度问题：中文是一种十分复杂的语言，让计算机理解中文语言更是困难。在英文行文中，单词与单词之间存在天然的分隔符——空格；而中文在词语级别中没有任何明显的分隔符。因此，到目前为止，中文分词的精度依然是自然语言处理过程中一个巨大的障碍。因此，若拥有更加成熟，精度更高的分词算法，就能够在关键词上出现更少的误判，从而提高最终结果的精度。

模型训练的样本误差问题：在实验的模型训练中，我们使用大量微博数据，选择四种合适的分类模型，通过学习训练的过程，得到相应模型的参数，让最终的模型能够最好的预测场景，供预测，分类等。而在模型训练过程中不可避免地会出现过拟合和欠拟合现象，从而导致有部分数据在正反样本的分类中出现错误。这种出现少量错误的情况，叫做模型训练中出现的误差 (Bias)，为了修正相应的模型误差，

通常使用交叉验证、重新取样等相应的方法，在更大规模的数据集上也有助于减少错误。

手动标记数据的误差问题：在本实验中，为了训练相应数据，我们给定的集合中标记了近 40000 条带标记的微博信息，区分正反样本，作为原始训练样本。然而，在大量的数据规模上，很难保证手动标记的每一条微博信息是完全准确的。

7.3 未来展望

本文主要使用临床数据对社交媒体上的病例进行验证，并未涉及互联网搜索数据的对比。由于在流行病传播过程中，具有多种复杂的影响因素，如个人体质不同造成的免疫力不同，由于季节性动物迁徙造成特定地区的爆发等，需要继续对算法进行加强研究。另外，由于中国疾控中心提供的数据没有在地区上实现更细粒度的划分，在地区性流感统计上还没有可行的验证性数据，因此目前该模型未能监测到特定地区性流感的爆发性传播。由于临床数据量较少，模型预测的精度和及时程度依然具有提高空间。

除此以外，该模型无法做到对未来的流感传播情况和可能爆发的地点进行预测。根据在流感追踪模型中已获得的微博正样本数据，若能更加充分挖掘微博中的内容信息，如微博圈子的粉丝、关注者、以及地理位置的信息，并能利用其相关特征，结合现有的相关模型进行预测，提出有效的流感预测算法，将会更加有助于及时预报可能传播的疫情，为流感预测提供更有效的手段。

希望未来能够有所突破，在流感的监控和预测方面做出更加有效、准确的实时性系统，为国家公共卫生事业提供有力的支持。

8 参考文献

参考文献

[1] Barry, John M. (2004). The Great Influenza: The Epic Story of the Greatest Plague in History. Viking Penguin. ISBN 978-0-670-89473-4.

[2] MacCallum, W.G. (1919). "Pathology of the pneumonia following influenza". JAMA-Journal of the American Medical Association. 72 (10): 720–723.

[3] Victor Vaughan: A Biography of the Pioneering Bacteriologist, 1851-1929 By Richard Adler

[4] http://www.who.int/topics/influenza/zh/ ,Influenza, World Health Organization. Retrieved at April, 5th, 2017

[5] https://www.hhs.gov/ U.S.Department of Health and Human Services, Retrieved at April, 3th, 2017

[6] Espinal, Yadira. "Viktor Mayer-Schonberger and Kenneth Cukier, Big Data: A Revolution That Will Transform How We Live, Work and Think." International Journal of Communication 7.0 (2013).

[7] "H7N9 Update - October 28, 2013". Centers for Disease Control and Prevention. October 28, 2013. Retrieved 1 November 2013.

[8] Exploring Health Topics in Chinese Social Media: An Analysis of Sina Weibo. 2014.

[9] Ginsberg J, Mohebbi M H, Patel R S, et al. Detecting influenza epidemics using search engine query data.[J]. Nature, 2008, 457(7232):1012-1014.

[10] http://trend.baidu.com [EB/OL]. 2017.3

[11] Butler D. When Google got flu wrong.[J]. Nature, 2013, 494(7436):155-6.

[12] Eysenbach G. Infodemiology: tracking flu-related searches on the web for syndromic surveillance.[J]. 2006, 244:244-248.

[13] Lazer D, Kennedy R, King G, et al. Big data. The parable of Google Flu: traps in big data analysis.[J]. Science, 2014, 343(6176):1203.

[14] Rech J. Discovering trends in software engineering with google trend[J]. Acm Sigsoft Software Engineering Notes, 2007, 32(2):1-2.

[15] Caduff, Carlo. "Sick Weather Ahead: On Data-Mining, Crowd-Sourcing and White Noise." The Cambridge Journal of Anthropology 32.1 (2014): 32-46.

[16] Hulth A, Rydevik G, Linde A, et al. Web queries as a source for syndromic surveillance.[J]. PLOS ONE, 2009, 4(2).

[17] Bao, J-X., et al. "Gonorrhea incidence forecasting research based on Baidu search data." 2013 International Conference on Management Science and Engineering. 2013.

[18] Broniatowski DA, Paul MJ, Dredze M (2013) National and Local Influenza Surveillance through Twitter: An Analysis of the 2012-2013 Influenza Epidemic. PLoS ONE 8(12): e83672.

[19] [Xia, Rongze, et al. "Mining users' activity on large Twitter text data." Proceedings of the Fifth International Conference on Internet Multimedia Computing and Service. ACM, 2013.]

[20] Christakis NA, Fowler JH Social Network Sensors for Early Detection of Conatagious Outbreaks, PLoS ONE5 (9).

[21] Dodge, Graham, James Sajor, and Michael Belt. "Social networking aggregator to track illnesses." U.S. Patent Application No. 14/123,923.

[22] Gayoavello D. A Meta-Analysis of State-of-the-Art Electoral Prediction From Twitter Data[J]. Social Science Computer Review, 2013, 31(6): 649-679.

[23] Adam Sadilek, Henry Kautz and Vincent Silenzio, C.2013. Predicting disease transmission from geo-tagged micro-blog data. In Proceedings of Canada, August, 2012.

[24] Filipa Peleja, João, João Magalhães, "Ranking Linked-Entities in a Sentiment Graph", Web Intelligence (WI) and Intelligent Agent Technologies (IAT) 2014 IEEE/WIC/ACM International Joint Conferences on, vol. 2, pp. 118-125, 2014.

[25] Zhang, Harry. The Optimality of Naive Bayes (PDF). FLAIRS2004 conference.

[26] Collier, Nigel, Nguyen Truong Son, and Ngoc Mai Nguyen. "OMG U got flu? Analysis of shared health messages for bio-surveillance." Journal of biomedical semantics 2.5 (2011): S9.

[27] Lampos V, De Bie T, Cristianini N, et al. Flu Detector - Tracking Epidemics on Twitter[J]. Spring, 2010: 599-602.

[28] Cortes, C.; Vapnik, V. (1995). "Support-vector networks". Machine Learning. 20 (3): 273–297. doi:10.1007/BF00994018.

[29] 11.Yang Y, Guo F, Zhao W, Gu Q, Huang M, Cao Q, et al. Novel avian-origin influenza A (H7N9) in critically ill patients in China. Crit Care Med. 2015;43(2):339-45.

[30] Zhou L, Ren R, Yang L, Bao C, Jiabing W, Wang D, et al. Sudden increase in human infection with avian influenza A(H7N9) virus in China, September- December 2016. Western Pac Surveill Response J. 2017;9(1).

[31] Obar, Jonathan A.; Wildman, Steve (2015). "Social media definition and the governance challenge: An introduction to the special issue". Telecommunications policy. 39 (9): 745–750.

[32] boyd, d.m.; Ellison, N.B. (2007). "Social Network Sites: Definition, History, and Scholarship". Journal of computer-mediated communication. 13 (1): 210–230.

[33] Radinsky K, Horvitz E. Mining the web to predict future events[C]. web search and data mining, 2013: 255-264.

[34] Aramaki E, Maskawa S, Morita M, et al. Twitter catches the flu: detecting influenza epidemics using Twitter[C]. empirical methods in natural language processing, 2011: 1568-1576.

[35] Kaplan Andreas M., Haenlein Michael (2010). "Users of the world, unite! The challenges and opportunities of social media" (PDF). Business Horizons. 53 (1): 61.

[36] http://www.acha.org/ILI-Project/ILI-case-definition-CDC.pdf

[37] 2016 年微博用户发展报告 [EB/OL] http://data.weibo.com/report .2016.12.

[38] 王晶晶, 邹远强, 彭友松, 李肯立, 蒋太交. 基于百度指数的登革热疫情预测研究 [J]. 计算机应用与软件,2016,(07):42-46+78.

[39] 宋蕊, 成军. 认识甲型 H7N9 禽流感 [J]. 首都医科大学学报,2013,(03):475-478.

[40] 赵本本, 殷旭东, 王伟. 基于 Scrapy 的 GitHub 数据爬虫 [J]. 电子技术与软件工程,2016,(06):199-202.

[41] 中国疾病预防控制中心 [EB/OL] http://www.chinacdc.cn 2017.3.

[42] [1] 鲁力, 邹远强, 彭友松, 李肯立, 蒋太交. 百度指数和微指数在中国流感监测中的比较分析 [J]. 计算机应用研究,2016,(02):392-395.

[43] 李秀婷. 基于互联网搜索数据的中国流感监测 [A]. 中国系统工程学会. 社会经济发展转型与系统工程——中国系统工程学会第 17 届学术年会论文集 [C]. 中国系统工程学会:,2012:10.

9 致谢

大学四年学习时光已经接近尾声，在此我想对我的母校，我的父母、亲人们，我的老师和同学们表达我由衷的谢意。感谢我的家人对我大学四年学习的默默支持;感谢南华大学给了我四年深造的机会，让我能继续学习和提高；感谢各位老师和同学们四年来的关心和鼓励。老师们课堂上的激情洋溢，课堂下的谆谆教诲；同学们在学习中的认真热情，生活上的热心主动，所有这些都让我的四年充满了感动。

这次毕业论文设计我得到了很多老师和同学的帮助，其中我的论文指导老师罗凌云教授对我的关心和支持尤为重要。每次遇到难题，罗老师的指导总是让我受益匪浅。导师渊博的专业知识，严谨的治学态度对我影响深远，这不禁使我树立了远大的学术目标、掌握了基本的研究方法，还让我明白了许多待人接物与为人处事的道理。在毕业设计的每个阶段，从选题到查阅资料，论文提纲的确定，中期论文的修改，后期论文格式调整等各个环节中都给予了我悉心的指导。这几个月以来，罗老师不仅在学业上给我以精心指导，同时还在思想给我以无微不至的关怀，在此谨向罗老师致以诚挚的谢意和崇高的敬意。

同时，本篇毕业设计的写作也得到了赵鑫同学的热情帮助。感谢在整个毕业设计期间和我密切合作的同学，和曾经在各个方面给予过我帮助的伙伴们，在此，我再一次真诚地向帮助过我的老师和同学表示感谢!

Appendices

Scrapy 爬虫 -Python 框架的部分代码

Cookies.py

```
# encoding=utf-8
import json
import base64
import requests

myWeiBo = [
# {'no': 'jiadieyuso3319@163.com', 'psw': 'a123456'},
# {'no': 'shudieful3618@163.com', 'psw': 'a123456'},
    {'no':'kakale_3@163.com', 'psw':'a456123'},
    {'no':'ningyyyy_happy@163.com', 'psw':'js456123'},
    {'no':'kiss_baby-520@163.com', 'psw':'c456123'},
    {'no':'jiegjieg00fan@163.com', 'psw':'a456123'},
    {'no':'keny_com@163.com', 'psw':'c456123'},
    {'no':'kevin_likai@163.com', 'psw':'c456123'},
    {'no':'kg_wak@163.com', 'psw':'a    456123'},
    {'no':'kpkp_8899@163.com', 'psw':'c456123'},
    {'no':'gzq_0324@163.com', 'psw':'a456123'},
]

def getCookies(weibo):
    """ 获取Cookies """
    cookies = []
    loginURL = r'https://login.sina.com.cn/sso/login.php?client=ssologin
                                            .js(v1.4.15)'
    for elem in weibo:
        account = elem['no']
        password = elem['psw']
        username = base64.b64encode(account.encode('utf-8')).decode('utf
                                                -8')
        postData = {
            "entry": "sso",
            "gateway": "1",
            "from": "null",
            "savestate": "30",
            "useticket": "0",
```

```
            "pagerefer": "",
            "vsnf": "1",
            "su": username,
            "service": "sso",
            "sp": password,
            "sr": "1440*900",
            "encoding": "UTF-8",
            "cdult": "3",
            "domain": "sina.com.cn",
            "prelt": "0",
            "returntype": "TEXT",
        }
        session = requests.Session()
        r = session.post(loginURL, data=postData)
        jsonStr = r.content.decode('gbk')
        info = json.loads(jsonStr)
        if info["retcode"] == "0":
            print "Get Cookie Success!( Account:%s )" % account
            cookie = session.cookies.get_dict()
            cookies.append(cookie)
        else:
            print "Failed!( Reason:%s )" % info['reason']
    return cookies

cookies = getCookies(myWeiBo)
print "Get Cookies Finish!( Num:%d)" % len(cookies)
```

items.py

```
# encoding=utf-8

from scrapy import Item, Field

class InformationItem(Item):
    """ 个人信息 """
    _id = Field()  # 用户ID
    NickName = Field()  # 昵称
    Gender = Field()  # 性别
    Province = Field()  # 所在省
    City = Field()  # 所在城市
    Signature = Field()  # 个性签名
```

```
    Birthday = Field()  # 生日
    Num_Tweets = Field()  # 微博数
    Num_Follows = Field()  # 关注数
    Num_Fans = Field()  # 粉丝数
    Sex_Orientation = Field()  # 性取向
    Marriage = Field()  # 婚姻状况
    URL = Field()  # 首页链接

class TweetsItem(Item):
    """ 微博信息 """
    _id = Field()  # 用户ID微博-ID
    ID = Field()  # 用户ID
    Content = Field()  # 微博内容
    PubTime = Field()  # 发表时间
    Co_oridinates = Field()  # 定位坐标
    Tools = Field()  # 发表工具平台/
    Like = Field()  # 点赞数
    Comment = Field()  # 评论数
    Transfer = Field()  # 转载数

class FollowsItem(Item):
    """ 关注人列表 """
    _id = Field()  # 用户ID
    follows = Field()  # 关注

class FansItem(Item):
    """ 粉丝列表 """
    _id = Field()  # 用户ID
    fans = Field()  # 粉丝
```

Pipelines.py

```
# encoding=utf-8
import pymongo
from items import InformationItem, TweetsItem, FollowsItem, FansItem

class MongoDBPipeline(object):
    def __init__(self):
        clinet = pymongo.MongoClient("localhost", 27017)
        db = clinet["SinaNew"]
```

```
        self.Information = db["Information"]
        self.Tweets = db["Tweets"]
        self.Follows = db["Follows"]
        self.Fans = db["Fans"]

    def process_item(self, item, spider):
        """ 判断的类型，并作相应的处理，再入数据库item """
        if isinstance(item, InformationItem):
            try:
                self.Information.insert(dict(item))
            except Exception:
                pass
        elif isinstance(item, TweetsItem):
            try:
                self.Tweets.insert(dict(item))
            except Exception:
                pass
        elif isinstance(item, FollowsItem):
            followsItems = dict(item)
            follows = followsItems.pop("follows")
            for i in range(len(follows)):
                followsItems[str(i + 1)] = follows[i]
            try:
                self.Follows.insert(followsItems)
            except Exception:
                pass
        elif isinstance(item, FansItem):
            fansItems = dict(item)
            fans = fansItems.pop("fans")
            for i in range(len(fans)):
                fansItems[str(i + 1)] = fans[i]
            try:
                self.Fans.insert(fansItems)
            except Exception:
                pass
        return item
```

Middleware.py

```
# encoding=utf-8
import random
```

```
from cookies import cookies
from user_agents import agents

class UserAgentMiddleware(object):
    """ 换User-Agent """

    def process_request(self, request, spider):
        agent = random.choice(agents)
        request.headers["User-Agent"] = agent

class CookiesMiddleware(object):
    """ 换Cookie """

    def process_request(self, request, spider):
        cookie = random.choice(cookies)
        request.cookies = cookie
```

Settings.py

```
# encoding=utf-8
BOT_NAME = 'Sina_spider1'

SPIDER_MODULES = ['Sina_spider1.spiders']
NEWSPIDER_MODULE = 'Sina_spider1.spiders'

DOWNLOADER_MIDDLEWARES = {
    "Sina_spider1.middleware.UserAgentMiddleware": 401,
    "Sina_spider1.middleware.CookiesMiddleware": 402,
}

ITEM_PIPELINES = {
    'Sina_spider1.pipelines.MongoDBPipleline': 300,
}

DOWNLOAD_DELAY = 0.01  # 间隔时间
# CONCURRENT_ITEMS = 1000
# CONCURRENT_REQUESTS = 100
# REDIRECT_ENABLED = False
# CONCURRENT_REQUESTS_PER_DOMAIN = 100
# CONCURRENT_REQUESTS_PER_IP = 0
# CONCURRENT_REQUESTS_PER_SPIDER=100
```

```
# DNSCACHE_ENABLED = True
LOG_LEVEL = 'INFO'     # 日志级别
# CONCURRENT_REQUESTS = 70
```

User-agents.py

```
# encoding=utf-8

agents = [
    "Mozilla/5.0 (Linux; U; Android 2.3.6; en-us; Nexus S Build/GRK39F)
                                        AppleWebKit/533.1 (KHTML, like
                                        Gecko) Version/4.0 Mobile Safari
                                        /533.1",
    "Avant Browser/1.2.789rel1 (http://www.avantbrowser.com)",
    "Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit/532.5 (
                                        KHTML, like Gecko) Chrome/4.0.
                                        249.0 Safari/532.5",
    "Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US) AppleWebKit/532.9 (
                                        KHTML, like Gecko) Chrome/5.0.
                                        310.0 Safari/532.9",
    "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/534.7 (
                                        KHTML, like Gecko) Chrome/7.0.
                                        514.0 Safari/534.7",
    "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/534.14
                                        (KHTML, like Gecko) Chrome/9.0.
                                        601.0 Safari/534.14",
    "Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit/534.14
                                        (KHTML, like Gecko) Chrome/10.0.
                                        601.0 Safari/534.14",
    "Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit/534.20
                                        (KHTML, like Gecko) Chrome/11.0.
                                        672.2 Safari/534.20",
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/534.27 (KHTML, like
                                         Gecko) Chrome/12.0.712.0 Safari
                                        /534.27",
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML, like
                                        Gecko) Chrome/13.0.782.24 Safari
                                        /535.1",
    "Mozilla/5.0 (Windows NT 6.0) AppleWebKit/535.2 (KHTML, like Gecko)
                                        Chrome/15.0.874.120 Safari/535.2
                                        ",
```

```
"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.7 (KHTML, like
                                    Gecko) Chrome/16.0.912.36 Safari
                                    /535.7",
"Mozilla/5.0 (Windows; U; Windows NT 6.0 x64; en-US; rv:1.9pre)
                                    Gecko/2008072421 Minefield/3.0.
                                    2pre",
"Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.10) Gecko/
                                    2009042316 Firefox/3.0.10",
"Mozilla/5.0 (Windows; U; Windows NT 6.0; en-GB; rv:1.9.0.11) Gecko/
                                    2009060215 Firefox/3.0.11 (.NET
                                    CLR 3.5.30729)",
"Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.1.6) Gecko/
                                    20091201 Firefox/3.5.6 GTB5",
"Mozilla/5.0 (Windows; U; Windows NT 5.1; tr; rv:1.9.2.8) Gecko/
                                    20100722 Firefox/3.6.8 ( .NET
                                    CLR 3.5.30729; .NET4.0E)",
"Mozilla/5.0 (Windows NT 6.1; rv:2.0.1) Gecko/20100101 Firefox/4.0.1
                                    ",
"Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:2.0.1) Gecko/20100101
                                    Firefox/4.0.1",
"Mozilla/5.0 (Windows NT 5.1; rv:5.0) Gecko/20100101 Firefox/5.0",
"Mozilla/5.0 (Windows NT 6.1; WOW64; rv:6.0a2) Gecko/20110622
                                    Firefox/6.0a2",
"Mozilla/5.0 (Windows NT 6.1; WOW64; rv:7.0.1) Gecko/20100101
                                    Firefox/7.0.1",
"Mozilla/5.0 (Windows NT 6.1; WOW64; rv:2.0b4pre) Gecko/20100815
                                    Minefield/4.0b4pre",
"Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0 )",
"Mozilla/4.0 (compatible; MSIE 5.5; Windows 98; Win 9x 4.90)",
"Mozilla/5.0 (Windows; U; Windows XP) Gecko MultiZilla/1.6.1.0a",
"Mozilla/2.02E (Win95; U)",
"Mozilla/3.01Gold (Win95; I)",
"Mozilla/4.8 [en] (Windows NT 5.1; U)",
"Mozilla/5.0 (Windows; U; Win98; en-US; rv:1.4) Gecko Netscape/7.1 (
                                    ax)",
"HTC_Dream Mozilla/5.0 (Linux; U; Android 1.5; en-ca; Build/CUPCAKE)
                                     AppleWebKit/528.5  (KHTML, like
                                     Gecko) Version/3.1.2 Mobile
                                    Safari/525.20.1",
"Mozilla/5.0 (hp-tablet; Linux; hpwOS/3.0.2; U; de-DE) AppleWebKit/
                                    534.6 (KHTML, like Gecko)
```

```
                                    wOSBrowser/234.40.1 Safari/534.6
                                     TouchPad/1.0",
"Mozilla/5.0 (Linux; U; Android 1.5; en-us; sdk Build/CUPCAKE)
                                    AppleWebkit/528.5  (KHTML, like
                                    Gecko) Version/3.1.2 Mobile
                                    Safari/525.20.1",
"Mozilla/5.0 (Linux; U; Android 2.1; en-us; Nexus One Build/ERD62)
                                    AppleWebKit/530.17 (KHTML, like
                                    Gecko) Version/4.0 Mobile Safari
                                    /530.17",
"Mozilla/5.0 (Linux; U; Android 2.2; en-us; Nexus One Build/FRF91)
                                    AppleWebKit/533.1 (KHTML, like
                                    Gecko) Version/4.0 Mobile Safari
                                    /533.1",
"Mozilla/5.0 (Linux; U; Android 1.5; en-us; htc_bahamas Build/CRB17)
                                     AppleWebKit/528.5  (KHTML, like
                                     Gecko) Version/3.1.2 Mobile
                                    Safari/525.20.1",
"Mozilla/5.0 (Linux; U; Android 2.1-update1; de-de; HTC Desire 1.19.
                                    161.5 Build/ERE27) AppleWebKit/
                                    530.17 (KHTML, like Gecko)
                                    Version/4.0 Mobile Safari/530.17
                                    ",
"Mozilla/5.0 (Linux; U; Android 2.2; en-us; Sprint APA9292KT Build/
                                    FRF91) AppleWebKit/533.1 (KHTML,
                                     like Gecko) Version/4.0 Mobile
                                    Safari/533.1",
"Mozilla/5.0 (Linux; U; Android 1.5; de-ch; HTC Hero Build/CUPCAKE)
                                    AppleWebKit/528.5  (KHTML, like
                                    Gecko) Version/3.1.2 Mobile
                                    Safari/525.20.1",
"Mozilla/5.0 (Linux; U; Android 2.2; en-us; ADR6300 Build/FRF91)
                                    AppleWebKit/533.1 (KHTML, like
                                    Gecko) Version/4.0 Mobile Safari
                                    /533.1",
"Mozilla/5.0 (Linux; U; Android 2.1; en-us; HTC Legend Build/cupcake
                                    ) AppleWebKit/530.17 (KHTML,
                                    like Gecko) Version/4.0 Mobile
                                    Safari/530.17",
"Mozilla/5.0 (Linux; U; Android 1.5; de-de; HTC Magic Build/PLAT-
                                    RC33) AppleWebKit/528.5  (KHTML,
```

```
                                    like Gecko) Version/3.1.2
                                    Mobile Safari/525.20.1 FirePHP/0
                                    .3",
"Mozilla/5.0 (Linux; U; Android 1.6; en-us; HTC_TATTOO_A3288 Build/
                                    DRC79) AppleWebKit/528.5  (KHTML
                                    , like Gecko) Version/3.1.2
                                    Mobile Safari/525.20.1",
"Mozilla/5.0 (Linux; U; Android 1.0; en-us; dream) AppleWebKit/525.
                                    10  (KHTML, like Gecko) Version/
                                    3.0.4 Mobile Safari/523.12.2",
"Mozilla/5.0 (Linux; U; Android 1.5; en-us; T-Mobile G1 Build/CRB43)
                                     AppleWebKit/528.5  (KHTML, like
                                     Gecko) Version/3.1.2 Mobile
                                    Safari 525.20.1",
"Mozilla/5.0 (Linux; U; Android 1.5; en-gb; T-Mobile_G2_Touch Build/
                                    CUPCAKE) AppleWebKit/528.5  (
                                    KHTML, like Gecko) Version/3.1.2
                                     Mobile Safari/525.20.1",
"Mozilla/5.0 (Linux; U; Android 2.0; en-us; Droid Build/ESD20)
                                    AppleWebKit/530.17 (KHTML, like
                                    Gecko) Version/4.0 Mobile Safari
                                    /530.17",
"Mozilla/5.0 (Linux; U; Android 2.2; en-us; Droid Build/FRG22D)
                                    AppleWebKit/533.1 (KHTML, like
                                    Gecko) Version/4.0 Mobile Safari
                                    /533.1",
"Mozilla/5.0 (Linux; U; Android 2.0; en-us; Milestone Build/
                                    SHOLS_U2_01.03.1) AppleWebKit/
                                    530.17 (KHTML, like Gecko)
                                    Version/4.0 Mobile Safari/530.17
                                    ",
"Mozilla/5.0 (Linux; U; Android 2.0.1; de-de; Milestone Build/
                                    SHOLS_U2_01.14.0) AppleWebKit/
                                    530.17 (KHTML, like Gecko)
                                    Version/4.0 Mobile Safari/530.17
                                    ",
"Mozilla/5.0 (Linux; U; Android 3.0; en-us; Xoom Build/HRI39)
                                    AppleWebKit/525.10  (KHTML, like
                                     Gecko) Version/3.0.4 Mobile
                                    Safari/523.12.2",
"Mozilla/5.0 (Linux; U; Android 0.5; en-us) AppleWebKit/522  (KHTML,
```

```
                                        like Gecko) Safari/419.3",
    "Mozilla/5.0 (Linux; U; Android 1.1; en-gb; dream) AppleWebKit/525.
                                        10  (KHTML, like Gecko) Version/
                                        3.0.4 Mobile Safari/523.12.2",
    "Mozilla/5.0 (Linux; U; Android 2.0; en-us; Droid Build/ESD20)
                                        AppleWebKit/530.17 (KHTML, like
                                        Gecko) Version/4.0 Mobile Safari
                                        /530.17",
    "Mozilla/5.0 (Linux; U; Android 2.1; en-us; Nexus One Build/ERD62)
                                        AppleWebKit/530.17 (KHTML, like
                                        Gecko) Version/4.0 Mobile Safari
                                        /530.17",
    "Mozilla/5.0 (Linux; U; Android 2.2; en-us; Sprint APA9292KT Build/
                                        FRF91) AppleWebKit/533.1 (KHTML,
                                        like Gecko) Version/4.0 Mobile
                                        Safari/533.1",
    "Mozilla/5.0 (Linux; U; Android 2.2; en-us; ADR6300 Build/FRF91)
                                        AppleWebKit/533.1 (KHTML, like
                                        Gecko) Version/4.0 Mobile Safari
                                        /533.1",
    "Mozilla/5.0 (Linux; U; Android 2.2; en-ca; GT-P1000M Build/FROYO)
                                        AppleWebKit/533.1 (KHTML, like
                                        Gecko) Version/4.0 Mobile Safari
                                        /533.1",
    "Mozilla/5.0 (Linux; U; Android 3.0.1; fr-fr; A500 Build/HRI66)
                                        AppleWebKit/534.13 (KHTML, like
                                        Gecko) Version/4.0 Safari/534.13
                                        ",
    "Mozilla/5.0 (Linux; U; Android 3.0; en-us; Xoom Build/HRI39)
                                        AppleWebKit/525.10  (KHTML, like
                                        Gecko) Version/3.0.4 Mobile
                                        Safari/523.12.2",
    "Mozilla/5.0 (Linux; U; Android 1.6; es-es; SonyEricssonX10i Build/
                                        R1FA016) AppleWebKit/528.5  (
                                        KHTML, like Gecko) Version/3.1.2
                                        Mobile Safari/525.20.1",
    "Mozilla/5.0 (Linux; U; Android 1.6; en-us; SonyEricssonX10i Build/
                                        R1AA056) AppleWebKit/528.5  (
                                        KHTML, like Gecko) Version/3.1.2
                                        Mobile Safari/525.20.1",
]
```

用 Scikit-learn 训练样本 HandleData.py

```
# user/bin/python
# -*- encoding:utf-8 -*-

import pandas as pd
import sys
import csv
import codecs
reload(sys)
sys.setdefaultencoding('utf-8')  # 设置系统编码为UTF-8前面需要记
                                   得,reload(sys)
f_path=[u'G:\\Design\\keywords-cleaned\\cleanData\\cough.csv',
        u'G:\\Design\\keywords-cleaned\\cleanData\\fare.csv',
        u'G:\\Design\\keywords-cleaned\\cleanData\\fashao.csv',
        u'G:\\Design\\keywords-cleaned\\cleanData\\ganmao.csv',
        u'G:\\Design\\keywords-cleaned\\cleanData\\H7N9.csv',
        u'G:\\Design\\keywords-cleaned\\cleanData\\toutong.csv']
f_all_path=u'G:\\Design\\keywords-cleaned\\cleanData\\alldata.csv'
f_allcleaned_path=u'G:\\Design\\keywords-cleaned\\cleanData\\
                                   alldataCleaned.csv'
def merge():
    # 将个表合并成一个表6
    f_cough = pd.read_csv(f_path[0])
    f_fare = pd.read_csv(f_path[1])
    f_fashao = pd.read_csv(f_path[2])
    f_ganmao = pd.read_csv(f_path[3])
    f_H7N9 = pd.read_csv(f_path[4])
    f_toutong = pd.read_csv(f_path[5])

    f_write=open(f_all_path,'w')
    fileheader = ['label', 'Content', '_id', 'ID', 'PubTime']
    writer = csv.DictWriter(f_write, fileheader)
    writer.writeheader()
    # 获取每个表的长度
    f_cough_len=f_cough.values.shape[0]
    f_fare_len = f_fare.values.shape[0]
    f_fashao_len = f_fashao.values.shape[0]
    f_toutong_len = f_toutong.values.shape[0]
    f_ganmao_len = f_ganmao.values.shape[0]
```

```
    f_H7N9_len = f_H7N9.values.shape[0]

    for row in range(f_cough_len):
        writer.writerow(
            {'label': f_cough['label'][row], 'Content': f_cough['Content
                                                ''][row],
             '_id': f_cough['_id'][row], 'ID': f_cough['ID'][row],
             'PubTime': f_cough['PubTime'][row]})

    for row in range(f_fare_len):
        writer.writerow(
            {'label': f_fare['label'][row], 'Content': f_fare['Content']
                                                [row],
             '_id': f_fare['_id'][row], 'ID': f_fare['ID'][row],
             'PubTime': f_fare['PubTime'][row]})
    for row in range(f_fashao_len):
        writer.writerow(
            {'label': f_fashao['label'][row], 'Content': f_fashao['
                                                Content'][row],
             '_id': f_fashao['_id'][row], 'ID': f_fashao['ID'][row],
             'PubTime': f_fashao['PubTime'][row]})
    for row in range(f_ganmao_len):
        writer.writerow(
            {'label': f_ganmao['label'][row], 'Content': f_ganmao['
                                                Content'][row],
             '_id': f_ganmao['_id'][row], 'ID': f_ganmao['ID'][row],
             'PubTime': f_ganmao['PubTime'][row]})
    for row in range(f_toutong_len):
        writer.writerow(
            {'label': f_toutong['label'][row], 'Content': f_toutong['
                                                Content'][row],
             '_id': f_toutong['_id'][row], 'ID': f_toutong['ID'][row],
             'PubTime': f_toutong['PubTime'][row]})
    for row in range(f_H7N9_len):
        writer.writerow(
            {'label': f_H7N9['label'][row], 'Content': f_H7N9['Content']
                                                [row],
             '_id': f_H7N9['_id'][row], 'ID': f_H7N9['ID'][row],
             'PubTime': f_H7N9['PubTime'][row]})
    f_write.close()
# 移除重复的数据
```

```
def removeRp():
    alldata=pd.read_csv(f_all_path)
    print alldata.values.shape

    f_write = open(f_allcleaned_path, 'w')
    fileheader = ['label', 'Content', '_id', 'ID', 'PubTime']
    writer = csv.DictWriter(f_write, fileheader)
    writer.writeheader()

    all_cleaned_data=alldata.drop_duplicates(['Content','PubTime'])
    print all_cleaned_data.values.shape

    all_cleaned_data.to_csv('G:\\Design\\keywords-cleaned\\cleanData\\
                                        alldataclened.csv')

def test():
    data = pd.read_excel('G:\\Design\\keywords-cleaned\\cleanData\\test.
                                        xlsx')
    data=data.drop_duplicates(['name'])
    print data
    new_data=data.reset_index(drop=True)
    new_data=pd.DataFrame(new_data)
    for i in range(new_data.values.shape[0]):
        print new_data['name'][i]
if __name__ == '__main__':
    # merge()
    removeRp()
    # test()
```

```
# user/bin/python
# _*_ coding:UTF-8_*_
import pandas as pd
import pickle
from sklearn.cross_validation import train_test_split
import Models
# 获取中文停词
def getStopWords(filepath):
    stop_words=[]
    with open(filepath,'r') as f:
        line=f.readline()
        while line:
```

```
            if not line:
                break
            line=unicode(line)
            line=line.strip('\n')
            stop_words.append(line)
            line=f.readline()
    return stop_words
# 保存模型
def saveModeltodisk(modelpath,model):
    with open(modelpath,'w') as f:
        pickle.dump(model,f)
# 获取模型
def getModelfromdisk(modelpath):
    with open(modelpath,'r') as f:
        model=pickle.load(f)
    return model

# 导入分完词的数据返回所有数据集,
def getData(filepath):
    lines = []
    data = pd.read_csv(filepath)  # data is a dataframe
    x_data = data['Content']
    y_data = data['label']
    return x_data, y_data

# 将所有数据划分为训练数据和测试数据
def getsplitedData(filepath):
    x_data,y_data=getData(filepath)
    x_data=Models.getVecData(x_data)
    x_train,x_test,y_train,y_test=train_test_split(x_data,y_data,
                                        test_size=0.25,random_state=33)
    return x_train,x_test,y_train,y_test
```

```
# user/bin/python
# _*_ coding:UTF-8_*_
import sys
import Utils
import Models
from sklearn.metrics import classification_report
reload(sys)
sys.setdefaultencoding('utf-8')
```

```
# 训练分类器
def trainClassification(model,x_train,y_train):
    model.fit(x_train,y_train)
    return model
def predict(model,x_test,y_test):
    y_predict=model.predict(x_test)
    print 'Accurancy of navie bayes is :', model.score(x_test, y_test)
    print classification_report(y_test, y_predict, target_names=['
                                        positive','negative'])
if __name__ == '__main__':
    filepath = 'C:\\Users\\admin\\Desktop\\Dissertation\\ZX-Model'
    x_train, x_test, y_train, y_test = Utils.getsplitedData(filepath)
    print x_train.shape, x_test.shape
    mnb_model=Models.getMNB_model()
    svm_model=Models.getSVM_model(linearModel=True)
    mnb_model=trainClassification(mnb_model,x_train,y_train)
    svm_model = trainClassification(svm_model, x_train, y_train)
    predict(mnb_model,x_test,y_test)
    predict(svm_model,x_test,y_test)
```

```
# user/bin/python
# _*_ coding:UTF-8_*_

from sklearn.feature_extraction.text import CountVectorizer,
                                    TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC,SVC

# 将是数据转换成矩阵
def getVecData(x_data):
    count_vec = CountVectorizer(decode_error='ignore', stop_words='
                                        english', analyzer='word')
    tf_idf_vec = TfidfTransformer()
    x_data = count_vec.fit_transform(x_data)
    x_data = tf_idf_vec.fit_transform(x_data)
    # # calculate the sparse of x_train
    print 'The average feature sparsity is {0:.3f}%' \
        .format(x_data.nnz / float(x_data.shape[0] * x_data.shape[1]) *
                                            100)
    return x_data
```

```
def getMNB_model():
    model=MultinomialNB()
    return model

def getSVM_model(linearModel=True,C=1.0,kernel='rbf'):
    if linearModel:
        model=LinearSVC()
    else:
        model=SVC(C=C,kernel=kernel)
    return model
```

表 14: 部分正样本数据

ID	PubTime	Content
1657537077	2016/12/31 22:31	嗓子, 不好, 说话, 沙哑, 咳嗽, 唱
1870660163	2016/12/31 21:51	感冒
5971643145	2016/12/31 14:55	铁打, 身子, 感冒
5971642955	2016/12/31 9:43	星星, 好像, 感冒
5458890745	2016/12/31 7:35	一夜, 宿醉, 加, 头痛, 差点, 折腾, 死
5991039686	2016/12/31 1:46	好好, 吃饭, 免疫力, 急剧下降, 低血糖, 头晕眼花, 拉肚子, 反胃, 想, 吐, 感冒, 估计, 大姨妈, 综合症
5971641743	2016/12/30 6:19	奥特曼, 冰雪, 榴莲, 冰激凌, 一如既往, 好吃, 感冒, 咳嗽, 阻止不了
6003490856	2016/12/30 5:55	感冒, 咳嗽, 需, 一粒, 蒜
1770522272	2016/12/28 20:40	感冒, 冻, 研究, 发现, 一生, 平均, 感冒, 次, 每次, 平均, 持续, 天, 一辈子, 感冒, 症状, 持续时间, 平均, 年, 感冒, 可不是, 冻, 简单, 意想不到, 原因, 缠, 生命, 时报, 帮, 揪出, 致病, 元凶,↓,↓,↓
5626662577	2016/12/28 16:39	感冒, 真的, 厉害, 小心, 感冒, 年末, 特别, 健康
1865943284	2016/12/27 19:00	腰疼, 发烧, 双重, 攻击, 漫漫长夜, 熬过去
5971642955	2016/12/27 15:37	感冒, 发烧, 喉咙痛
2096479903	2016/12/27 11:07	分不清, 感冒, 流行, 感冒, 吃错药, 咳咳, 冷空气, 刚刚, 袭, 赶上, 新一波, 冬季, 感冒, 有人, 劝, 多喝水, 不用, 吃药, 痊愈, 有人, 说, 吃药, 抑制, 感冒, 咋办, 分清, 感冒, 流行, 感冒, 关键, 戳, 视频,↓,↓, 全方位, 感冒, 好好
5972691708	2016/12/25 20:30	好不容易, 休息, 明天, 发烧, 逛街
5995758103	2016/12/25 18:59	剩, 咳嗽, 好好, 养
5995758102	2016/12/25 4:04	感冒, 真心, 难受, 听说, 感冒, 真的
2348287675	2016/12/24 17:26	圣诞节, 在家, 感冒,ssq
5726404623	2016/12/23 23:40	不用, 花, 一分钱, 轻松, 治, 感冒, 鼻塞, 感冒, 久, 愈, 朋友, 视频, 中, 小女孩, 岁, 分钟, 鼻子, 通, 感冒, 完, 感冒, 高发, 季节, 马给, 身边, 喜欢, 进, 微博看, 视频
5972690399	2016/12/22 16:39	天晴, 感冒
5971641627	2016/12/22 10:22	世上, 妈妈, 感冒, 老妈, 午休, 睡觉, 孩子, 老妈, 甩手掌柜, 碎觉
6003490856	2016/12/22 6:01	冬天, 感冒, 教, 一招, 瞬间, 止咳
1956817932	2016/12/21 19:39	出国前, 收到, 贴心, 圣诞礼物, 花, 宝宝, 咳嗽
5890216995	2016/12/21 1:53	偏头痛
5971641743	2016/12/21 0:19	感冒, 太难受, 耳鸣, 鼻塞, 加, 鼻涕, 横流, 两眼, 水汪汪, 郁闷, 郁闷, 心情, 沉重, 人待会
5972689455	2016/12/20 23:54	头痛, 想, 哭
5971641700	2016/12/20 22:32	发个, 烧, 不想, 感冒
5995758103	2016/12/19 20:46	感冒, 鼻塞, 天知道, 难受
2919236233	2016/12/19 13:00	好苦, 做人, 好苦, 活着, 好苦
5159551076	2016/12/19 12:35	本周, 金市
1802633291	2016/12/18 17:12	感冒, 今晚, 生活, 手记, 停, 改, 直播, 连线, 来聊, 这周, 生活, 不想, 原因, 停滞, 工作, 同意, 扣, 放送, 老歌
3979934337	2016/12/17 21:37	佳, 咳嗽, 好多天
3312977743	2016/12/15 16:20	咳嗽, 路, 陈, 师傅, 教, 小窍门, 缓解, 咳嗽, 超级, 简单, 三步, 鸡蛋, 白砂糖, 鸡蛋, 打到, 碗, 里, 糖, 均匀, 撒, 上锅, 蒸熟, 味道, 不错, 快点, 试试
5531241012	2016/12/15 14:58	感冒, 不好
3807381112	2016/12/15 12:14	感冒, 出门, 戴, 口罩
1833961724	2016/12/14 19:33	头痛, 冷, 恢复力, 变差
5770314536	2016/12/14 13:40	感冒, 真不知道, 感冒
5572657545	2016/12/13 21:40	孙祥, 回应, 住院, 重感冒, 身体, 大碍
1346896144	2016/12/13 20:50	昨天, 喝, 感冒, 戒酒
3219654444	2016/12/13 9:05	话, 只能, 梗, 喉咙, 烂往, 心底
6003490856	2016/12/13 5:56	咳嗽, 克星, 在家, 跑, 医院, 太, 实用
6003490856	2016/12/12 18:55	一分钱, 不花, 对付, 咳嗽, 顽疾, 见效, 太, 有用
5151217211	2016/12/12 15:21	休息时间, 看病, 理发, 逛街, 之自, 拍,gay
3171944275	2016/12/12 8:42	小雨, 温度, 度, 气温, 降温, 出门, 小伙伴, 伞, 一不小心, 头, 洗, 感冒
2671775731	2016/12/11 10:13	感冒, 咳嗽, 越来越, 纷纷, 中标, 小病, 难受, 感冒, 一回, 感觉, 身体, 掏空, 咳嗽, 咳得, 抓狂
5972691372	2016/12/9 3:35	六个月, 娘俩, 纷纷, 感冒, 咳嗽, 点, 终于, 睡下, 快快
5770314734	2016/12/8 22:24	来到, 拉萨, 感冒, 全好, 高原, 忌讳, 同行, 耿姐, 兴奋, 淡然, 拉萨, 机场, 开往, 市内, 路上, 满目, 沧凉, 山, 光秃秃, 山顶, 云彩, 问, 朋友, 想法, 耿姐, 抢, 说, 想, 云彩, 扯下来, 天, 高云, 密, 高山, 之颠, 真有, 伸手, 攀天, 欲望
5930221915	2016/12/8 21:10	感冒, 太, 猖狂
5329650869	2016/12/7 15:40	感冒, 想, 吃, 大甜
5042996073	2016/12/6 16:38	感冒, 难受, 头疼
1676280571	2016/12/3 3:08	偏头痛, 演出, 前, 头痛, 欲裂, 演出, 时, 状态, 出奇, 伴随, 头疼, 演完, 这场, 感受, 热情, 爱, 台上台下, 走心, 石家庄
1956817932	2016/12/2 20:38	名字, 超级, 好看, 鸡皮疙瘩, 几次, 泪目, 感冒, 嗓子疼, 默默, 回家, 喝药, 想要, 生病, 周末
2634154091	2016/11/30 22:27	腿毛, 哥, 公公, 打电话, 感冒, 咳嗽
2053374713	2016/11/29 16:44	嗓子疼
5875958974	2016/11/27 22:44	气冷, 各位朋友, 多加, 衣服, 感冒, 照顾, 先, 照顾
5125965954	2016/11/25 0:54	一眼, 扫过去, 一口, 正要, 咽下, 米饭, 咽不下, 哽在, 喉咙, 里, 剧烈地, 咳嗽
2301626242	2016/11/21 20:23	宝宝, 低烧, 妈妈, 留心, 原因, 低烧, 指, 体温, 长期, 低烧, 指, 持续, 发烧, 两周, 包括, 间断, 发烧, 宝宝, 长期, 低烧, 原因, 导致
1298140275	2016/11/21 11:35	這次, 感冒, 更惨, 現在, 說, 話, 无聲音
1298140275	2016/11/20 18:29	感冒, 嚴重, 天, 天, 不過, 還是, 寘貝們, 丨, 起來, 直播, 喔, 雖然, 氣色, 超差, 强壯
5972690523	2016/11/19 7:53	一种, 深度, 中毒, 感觉, 喉咙, 毛毛, 呛, 下午, 真的, 够, 快点, 吃, 好吃
2609906283	2016/11/18 19:50	咳嗽, 嗓子, 发炎, 低烧, 难受, 晓得, 能抗
5142742807	2016/11/16 21:42	一路, 感冒, 魔, 怔
3312977743	2016/11/16 18:32	咳嗽, 总, 不好, 听听, 美国, 权威, 医生, 生命, 时报
5834626445	2016/11/15 17:30	遇见, 生命, 中该, 偶然, 教会, 走, 地方, 经历, 我该, 经历, 事, 遇见, 我该, 遇见
3210743781	2016/11/14 19:23	日子, 硬生生, 过成, 无时不刻, 头疼, 浑身, 疼, 三天, 感冒, 五天, 发烧, 睡着, 开心
5515510661	2016/11/10 9:25	昨晚, 更新
5153227793	2016/11/10 8:22	喉咙, 发炎
5446697209	2016/11/9 17:43	几天, 双十, 一忙, 上火, 喉咙痛, 发烧
5759828712	2016/11/7 5:51	习惯, 买来, 奶粉, 不冲, 喜欢, 手指, 醮, 舔, 奶粉, 好像, 变味, 淡, 一点, 算了, 感冒, 味觉, 迟钝, 说, 罢, 白白的, 手指, 奶粉罐, 抽出来, 放到, 嘴里, 吮吸, 收到, 前男友, 来信, 想, 融为一体, 骨灰, 白色
6065489741	2016/11/7 4:18	感冒, 做, 件, 事, 找, 一张, 床, 休息, 补充, 维, 吃, 黑, 巧克力, 止咳, 打开, 加湿器, 吃, 流质食物, 缓解, 鼻塞, 喝, 毫升, 水, 远离, 乳制品, 服用, 非处方药, 耐心, 天, 自行, 消失, 美国威斯康星大学, 发布, 感冒, 白皮书
1881670810	2016/11/6 20:25	感冒, 不停, 出去玩

表 15: 部分反样本样例数据

ID	PubTime	Content
2569015175	2016/4/11 17:25	美女, 客人, 做, 韩式, 永久, 眉毛, 样子, 过段时间, 整体, 新长, 眉毛, 柔和, 自然
1625293545	2016/4/11 17:23	中国社科院, 半数, 网友, 月薪, 社科院, 文化, 研究, 中心, 组织, 编写, 文化, 蓝皮书, 指出, 我国, 互联网, 网民, 学历, 高中, 中专, 技校, 占, 大专, 占, 本科, 占, 互联网, 用户, 收入, 统计, 收入, 元, 占, 元, 占, 元, 占, 考研, 吃, 住, 优惠,
2327270763	2016/4/11 17:19	立春, 分享, 清人, 咬春诗, 暖律, 潜, 催腊底, 春, 登筵, 生菜, 记, 芳辰, 灵根属, 土含, 冰脆, 细缕, 堆盘, 切玉匀, 佐酒, 暗香, 生, 匕, 柊, 加餐, 清, 响动, 牙唇, 帝城, 节物, 乡园, 味, 取次, 关心, 白发, 新, 吃, 春饼, 称, 咬春, 咬, 字, 心情
5389681635	2016/4/11 17:18	硬货, 欧树, 万能, 油
1719140934	2016/4/11 17:15	运动, 闪光, 三星, 实战, 体育, 摄影,galaxy,edge
2730623002	2016/4/11 17:14	丰宁, 滕氏, 布糊画, 列入, 国家, 非遗, 名录, 滕氏, 布糊画, 丰宁, 民间艺术, 家, 滕腾, 发明, 集, 绘画, 雕塑, 堆绣, 唐卡, 剪纸, 工艺, 新画种, 大成, 用料, 讲究, 色彩, 绚丽, 操作, 细腻, 画面, 逼真, 取材, 画作, 成, 须, 经绘样, 分解, 制版
3775048831	2016/4/11 17:12	跟着, 大疆, 创新, 来到, 世界, 拉力, 锦标赛, 瑞典, 站, 现场, 寒冷, 卡尔斯塔德, 感受, 世界, 顶级, 车手, 冰雪, 丛林中, 疾速, 飞奔
3901435221	2016/4/11 17:11	ig,update,enjoy,long,weekend,na,ka,april,pm
5237640081	2016/4/11 17:10	截拳道, 基础, 功法, 石, 天龙
1676317545	2016/4/11 17:10	校马, 开, 跑, 拉开, 校庆, 月, 特色, 活动, 序幕
1676317545	2016/4/11 17:06	月, 日, 泰国, 公主, 诗琳通, 访问, 清华大学, 清华大学, 校长, 邱勇, 工字, 厅, 会见, 诗琳通, 公主, 一行, 诗琳通, 公主, 当天, 出席, 清华大学, 举办, 梁思成, 佛光寺, 暨, 建筑, 遗产, 乡村, 研究, 国际, 学术, 研讨会
1970798547	2016/4/11 17:04	瑞士, 航空
1676317545	2016/4/11 17:02	月, 日, 清华大学, 次, 科研工作, 讨论会, 闭幕, 校长, 邱勇, 院士, 第十七次, 科研工作, 讨论会, 意义, 成果, 清华大学, 深化, 科研, 体制, 机制, 改革, 若干意见, 这是, 纲领性, 指导性, 意义, 文件, 第一次, 提出, 清华, 科研, 引领, 性, 目标
1970798547	2016/4/11 17:02	海航, 全球, 买买, 买
3143107950	2016/4/11 17:01	公司, 挺, 上课, 挺
5609885403	2016/4/11 17:00	爆款, 水蜜桃, 手机, 实拍, 图到, 超甜, 多汁, 桃子, 放软, 香甜, 想, 吃, 想, 做代, 李, 加徽星,wechat,guoyou
1676317545	2016/4/11 16:59	月, 日, 上午, 清华大学, 外国, 语言, 文学系, 周年, 系庆, 启动, 暨, 发展, 座谈会, 丙, 会议室, 清华大学, 外文系, 成立, 年, 初名, 西洋, 文学系, 年, 易名, 外国语, 文学系, 简称, 外文系, 经历, 几次, 调整, 年, 更名, 外国, 语言, 文学系
1642333010	2016/4/11 16:58	打破, 公牛队, 胜, 记录, 勇士队, 球员, 格林, 说, 八卦, 乔丹, 全明星, 告诉, 赢, 纪录, 打破, 纪录, 破, 我会, 生气, 我会, 责怪, 乔丹, 逼装, 漂亮, 或者说, 超越, 记录, 自信, 装, 逼
3193700390	2016/4/11 16:58	史上, 便宜, 手机, 支架, 成本低, 手快, 超, 实用, 不来学, 问, 手模, 出场费, 秒, 拍, 作品
2730623002	2016/4/11 16:58	无锡, 古运河, 京杭大运河, 历经, 多年, 具有活力, 缺少, 黄金水道, 横贯, 无锡市, 古运河, 段, 吴桥, 经西水墩, 南门, 至清, 名桥, 这长, 约, 公里, 河段, 最具, 江南水乡, 风情
3822595309	2016/4/11 16:58	语文课, 老师, 作文题目, 老师, 小明, 写道, 爱, 英语老师, 上课, 没到, 胸先, 下课, 走, 屁股, 老师, 滚回去, 家长
1705511313	2016/4/11 16:55	丑, 先, 睡, 我美, 直播, 轻氧, 酵素, 联合, 创始人, 小雪, 速来, 围观, 点此,yukii
1237058801	2016/4/11 16:53	设计师, 简单, 线条, 画, 世纪, 现代主义, 住宅, 建筑, 有种, 简装版, 赶脚,michael,kamp,cad
3833022500	2016/4/11 16:53	实力, 解套, 行情, 演绎, 老秦, 总能, 一击, 命中, 市场, 脉搏, 致使, 创造财富, 信, 大可, 拉过来, 秦, 对质, 建议, 回头, 老秦, 今日, 所发, 博文, 内容, 回落, 触及, 新低, 欧盘, 进一步, 回落, 微博, 见证, 明白, 拿下, 利润, 何难
2327270763	2016/4/11 16:50	传统, 美食, 海宁, 长安, 宴球, 宴球, 长安, 一道, 名菜, 写, 进, 长安镇, 志, 当家的, 宾客, 一道, 见面礼, 宴球, 圆, 缠含, 大团圆, 含意, 宴球, 原料, 鱼, 白鲢, 花, 鲢, 草鱼, 三种, 选用, 斤, 白鲢鱼, 最佳, 鱼肉, 细嫩, 活泛, 长安, 宴球, 鱼
1299491507	2016/4/11 16:49	生日快乐, 世界, 独一无二
1417678504	2016/4/11 16:35	约, 桃花, 别错过, 电影, 后续, 今天下午
1773568137	2016/4/11 16:35	太平花, 谷
1773568137	2016/4/11 16:33	湾里区, 第一, 中学
5453121809	2016/4/11 16:32	美丑, 睡, 直播, 醉, 直播
1833784983	2016/4/11 16:31	来到, 乐迷, 优惠, 商城, 超多, 给力, 活动, 外, 线下, 店, 福利, 爆棚, 月, 日千店, 生态, 狂欢, 月, 日, 购买,lepar
2730623002	2016/4/11 16:31	辛弃疾, 汉宫春, 立春, 春已, 归来, 看美人, 头上, 袅袅, 春幡, 无端, 风雨, 未肯, 收尽余寒, 年, 时, 燕子, 料, 今宵, 梦到, 西园, 浑, 未办, 黄, 柑荐, 酒, 传青, 韭堆, 盘, 笑, 东风, 熏, 梅染, 柳, 没些, 闲, 闲时, 镜里, 朱颜, 清愁, 问, 何人,
2708473010	2016/4/11 16:30	碾压, 辨析, 力, 拍照, 当年, 赢, 辨析, 力和超, 弱光, 表现, 白平衡, 拍照, 体验, 改进, 面对, 重点, 提升, 弱光, 表现, 旗舰, 手机, 最强, 夜视仪, 三星, 战平, 超越, 喵, 喵,edge,lg,lg,edge,lg
1659896173	2016/4/11 16:29	誰, 說, 吃, 午餐
3822595309	2016/4/11 16:28	老婆, 互换, 身份, 爸爸, 妈妈, 中午, 六岁, 儿子, 回来, 厨房, 说, 爸爸, 一会, 回来, 洗手, 开饭, 老婆, 开门, 儿子, 一声, 哭, 妈妈, 爸爸, 疯
1773928521	2016/4/11 16:27	宝宝, 远视眼, 玩具, 放, 特别, 使劲, 调节, 眼睛, 看得见, 时间, 久, 宝宝, 眼睛, 较长时间, 地向, 旋转, 发展, 成, 内斜视, 正确, 方法, 玩具, 悬挂, 围栏, 更换, 玩具, 位置, 方向, 玩具, 逗, 宝宝, 玩具, 放在, 眼睛, 太近, 地方, 会影
2327270763	2016/4/11 16:27	金色, 新疆, 天马, 细毛羊, 提起, 新疆, 想到, 新疆, 细毛羊, 天马, 新疆, 自古以来, 良马, 地方, 汉武帝, 乌孙今, 伊犁, 马, 称为, 天马, 后凉, 创建人, 吕光, 赞誉, 龟兹, 马为, 天骥龙麟, 新疆, 细毛羊, 珍贵, 毛, 细度, 强度, 羊毛, 密度, 高
2663387043	2016/4/11 16:26	富贵, 牡丹
2615149943	2016/4/11 16:25	金饰, 美, 一点, 店, 金饰, 现货, 无需, 等待, 购买, 请, 上海, 店, 徐汇区, 新乐, 路, 号, 驾仕堂,ch,weixin
2615149943	2016/4/11 16:24	帅图, 款式, 货, 看好, 资讯, 购买, 请, 上海, 店, 徐汇区, 新乐, 路, 号, 驾仕堂,weixin
2615149943	2016/4/11 16:22	藍寶石, 現貨, 无需, 等待, 购买, 请, 上海, 店, 徐汇区, 新乐, 路, 号, 驾仕堂,baby,fat,weixin
2615149943	2016/4/11 16:22	新款, 帽子, 补货, 已到, 店, 购买, 请, 上海, 店, 徐汇区, 新乐, 路, 号, 驾仕堂,weixin
1648362363	2016/4/11 16:20	感谢, 参与, 后援会, 活动, 芋亲, 请贴, 版头, 设计, 加精, 亲, 贴, 前缀, 设计者, 私信, 后援会, 姓名, 地址, 电话, 贴, 寄, 礼物, 咯, 举办, 同乐会, 明年, 周年, 搞次,id,ps
2301626242	2016/4/11 16:20	孕期, 晒太阳, 春日, 暖阳, 怎能, 辜负, 阳光明媚, 春天, 太阳, 暖暖的, 刺眼, 毒辣, 大人, 孩子, 外面, 享受, 阳光, 爱抚, 孕妈咪, 怎能, 错过
2615149943	2016/4/11 16:19	两款, 金, 手链, 补货, 已到, 店, 无需, 等待, 购买, 请, 上海, 店, 徐汇区, 新乐, 路, 号, 驾仕堂,weixin
2874933414	2016/4/11 16:13	王者, 爹来, 快速, 丢掉, 烟头, 爹, 干, 纷争, 早已, 看透, 输赢, 稚嫩, 脸上, 几个, 巴掌, 印, 年少, 轻狂, 抽烟, 王, 抽烟, 爹, 拳头, 陪, 身边
1119831235	2016/4/11 16:12	超, 美的, 叶子, 镂空, 宽版, 手镯, 精致, 优雅, 做工, 繁复, 美丽, 尤物, 错过
3193700390	2016/4/11 16:12	号, 外号, 外, 姐, 卡通, 形象, 投票, 出炉, 一周, 投票, 软萌, 姐, 出炉, 刘海, 圆脸, 身怀绝技, 戳, 链接, 查看
2327270763	2016/4/11 16:09	诀别书, 春华, 竞芳, 五色, 凌素, 琴尚, 御, 新声, 代故, 锦水, 鸳, 汉宫, 有水, 彼物, 新, 嗟, 世之人, 瞢, 淫, 不悟, 朱弦断, 明镜, 缺, 朝露, 晞, 芳时, 歇, 白头, 吟, 伤离别, 努力, 加餐, 勿念, 妾, 锦水, 汤汤, 君长, 诀, 据传说, 司马相如, 卻
1688208774	2016/4/11 16:08	打开, 迷家官, 网会, 随机, 跳转, 页面, 网页, 效果, 信, 没准,ssd,jinseiyarinaoshi,com
1998321847	2016/4/11 16:07	一款, 名为, 虚拟现实, 恤, 看透, 人体, 穿, 秒, 变, 人体, 解剖, 标本, 神奇, 恤, 黑, 科技, 利用, 技术, 一种, 场景, 化, 展示, 失望, 至少, 未来, 医院, 检查, 身体,virtuali,tee,vr
5883111505	2016/4/11 16:07	自信, 千万别, 打击
3833070073	2016/4/11 16:05	旅行, 前, 做, 项
2249195655	2016/4/11 16:03	岁, 南昌, 玩, 一圈, 滕王阁, 夜景, 特别, 美, 体验, 中国, 摩天轮, 超, 开心
2203060760	2016/4/11 16:03	未画, 完, 待续, 搁笔, 忙
2730623002	2016/4/11 16:02	过年, 食材, 选用, 法则, 小炒, 肉用, 臀, 做, 肉圆, 用前, 夹心, 煨, 肉用, 硬短勒, 炒, 鱼片, 青鱼, 季鱼, 做, 鱼松, 用鱼, 鲤鱼, 蒸鸡用, 雏鸡, 煨, 鸡用, 骟, 鸡, 取, 鸡汁, 老鸡, 鸡用, 雌, 嫩, 鸭用, 雄才, 肥, 莼菜, 用头, 芹韭用, 根, 皆, 之
5609885403	2016/4/11 16:00	大樱桃, 新货, 手机, 实拍, 图来, 新鲜, 水果, 想, 吃, 想, 做代, 李, 加徽星,wechat,guoyou
1802546037	2016/4/11 16:00	申论, 范文, 跟着, 正确, 脚步, 佳作, 范文, 不看, 难受
3822595309	2016/4/11 15:58	一只, 老鼠, 愁眉苦脸, 猫, 天神, 同情, 遭遇, 施法, 一只, 猫, 老鼠, 猫, 害怕, 狗, 天神, 狗, 老虎, 天神, 就让, 做, 老虎, 担心, 遇上, 猎人, 天神, 只好, 变回, 老鼠, 天神, 说, 做, 帮不了, 拥有, 老鼠, 胆
1593835893	2016/4/11 15:57	通告, 补妆, 一镜, 神情, 杀青
3737387213	2016/4/11 15:48	内参, 收评, 数据, 好于, 预期, 股指, 高位, 震荡, 周一, 公布, 经济, 数据, 好于, 市场, 预期, 通胀, 忧虑, 暂时, 缓解, 股指, 高开高, 走, 全天, 上涨, 点, 量, 放大, 亿左右, 个股, 呈现, 两极, 分, 发自
2078244997	2016/4/11 15:47	配音, 时, 发现自己, 眼神, 内心, 潜台词, 说
1688208774	2016/4/11 15:47	两位, 主角, 当上, 歌姬, 飞行员, 拜托, 起码, 标准, 流程, 走, 两边, 钦定, 太, 敷衍,ep
2761951265	2016/4/11 15:45	犯困, 一杯, 冰, 咖啡
5722080925	2016/4/11 15:44	香港, 艾丝, 美养发, 头发, 里, 长痘痘, 原因, 头发, 里, 长痘痘, 原因, 内分泌, 失调, 皮脂腺, 分泌, 旺盛, 毛孔, 堵塞, 头上, 长痘, 头发, 粘, 灰尘, 细菌, 易, 发生, 感染, 炎症, 导致, 发痒, 疼痛, 症状
1625293545	2016/4/11 15:43	就业率, 十大, 研究生, 院校, 排行榜
2730623002	2016/4/11 15:42	靖西, 米, 饺, 靖西, 米, 饺, 广西, 靖西县, 特色, 风味, 美食, 特有, 香米, 浸泡, 人工, 水磨, 碾压, 蒸熟, 适量, 面粉, 制成, 饺子皮, 内陷, 包以, 猪肉, 莲藕, 凉薯, 等料, 馅, 放到, 蒸笼, 蒸熟, 饺子皮, 韧而嫩, 特有, 淡淡的, 香味, 糯而, 不
2327270763	2016/4/11 15:42	少年, 识, 愁滋味, 爱, 上层, 楼, 爱, 上层, 楼, 赋, 新词, 强说, 愁, 而今, 识尽, 愁滋味, 欲说还休, 欲说还休, 道, 天凉, 好个秋, 辛弃疾, 丑, 奴儿, 书, 博山, 道, 中壁
2761951265	2016/4/11 15:41	减肥, 每块, 肉, 脾气
3316646582	2016/4/11 15:40	事情, 只不过, 说, 东西, 就让, 愁眉, 现实, 说