**目 录**

[第一章 绪论 1](#_Toc10369)

[1.1 选题背景和意义 1](#_Toc26072)

[1.1.1 研究背景 1](#_Toc1434)

[1.1.2 研究意义 1](#_Toc28788)

[1.2 研究现状 2](#_Toc7481)

[1.3 研究框架和方法 2](#_Toc11234)

[1.3.1 研究框架 2](#_Toc27179)

[1.3.2 研究工具 3](#_Toc13549)

[第二章 意见领袖 4](#_Toc13953)

[2.1 意见领袖的概念 4](#_Toc29738)

[2.2 意见领袖的影响力 4](#_Toc31795)

[2.3 意见领袖的特征 5](#_Toc18559)

[2.4 意见领袖的功能 6](#_Toc21)

[第三章 雪球网意见领袖发现算法分析 7](#_Toc26319)

[3.1 雪球网网络特征 7](#_Toc8542)

[3.2 现有社交网络“微博”的意见领袖发现算法 8](#_Toc8172)

[3.3 PageRank算法 9](#_Toc27198)

[3.4 雪球网意见领袖发现算法 10](#_Toc1863)

[第四章 数据融合算法 12](#_Toc4703)

[4.1 层次分析法 12](#_Toc2037)

[4.1.1 层次分析法定义 12](#_Toc27207)

[4.1.2 层次分析法的优点 12](#_Toc8960)

[4.1.3 层次分析法的局限 12](#_Toc7926)

[4.1.4 层次分析法的算法 13](#_Toc10527)

[第五章 评价算法的合理性 15](#_Toc32561)

[5.1 语义分析 15](#_Toc4935)

[5.2 Tushare财经数据接口包 17](#_Toc30802)

[第六章 数据处理 20](#_Toc30903)

[6.1 采集到的数据 20](#_Toc28328)

[6.2 类关系图 22](#_Toc22122)

[6.3 关注网络 23](#_Toc30106)

[6.3.1 封装对象UserBean.java 23](#_Toc15124)

[6.3.2 与数据库交互Follower.java 23](#_Toc20021)

[6.3.3 封装PRBean.java 24](#_Toc17433)

[6.3.4 关注网络的pageRank算法PageRank.java 24](#_Toc1735)

[6.4 转发网络 25](#_Toc31516)

[6.4.1封装对象RetweetBean 25](#_Toc11444)

[6.4.2 与数据库交互Retweet.java 25](#_Toc24850)

[6.4.3 封装PRBean.java（直接调用关注网络的PRBean类） 25](#_Toc18868)

[6.4.4 转发网络的pageRank算法PageRank.java 25](#_Toc6922)

[6.5 回复/收藏/打赏 26](#_Toc28681)

[6.5.1 封装对象RDFBean.java 26](#_Toc5630)

[6.5.2 分别计算回复数/收藏数/打赏数的加权分数RDF.java 27](#_Toc30072)

[6.6层次分析法 27](#_Toc25198)

[6.6.1 用层次分析法计算权重Index.java 27](#_Toc15119)

[6.7 意见领袖排序 28](#_Toc19861)

[6.7.1 排序算法AllSort.java 28](#_Toc4083)

[6.8 评价 28](#_Toc11636)

[6.8.1 封装对象StockTextBean.java 28](#_Toc28625)

[6.8.2 与数据库交互StockTextDao.java 29](#_Toc15881)

[6.8.3 获得Tushare通联数据HttpUtil.java 29](#_Toc8615)

[6.8.4 判断意见领袖言论的正确性Semtiment.java 29](#_Toc8554)

[第七章 结果分析 31](#_Toc8701)

[7.1 意见领袖 31](#_Toc24994)

[7.2 意见领袖言论正确性 35](#_Toc23497)

[第八章 结论与展望 40](#_Toc2685)

[8.1 全文总结 40](#_Toc3477)

[8.2 研究展望 40](#_Toc25253)

[参考文献 41](#_Toc25752)

[致 谢 42](#_Toc18115)

# 第一章 绪论

## 1.1 选题背景和意义

1.1.1 研究背景

股票市场是股票发行和交易的场所，包括[发行市场](http://baike.baidu.com/view/204052.htm" \t "http://baike.baidu.com/_blank)和[流通市场](http://baike.baidu.com/view/204112.htm" \t "http://baike.baidu.com/_blank)两部分。股份公司通过面向社会发行股票，迅速集中大量资金，实现生产的规模经营；而社会上分散的资金盈余者本着“利益共享、风险共担”的原则投资股份公司，谋求财富的[增值](http://baike.baidu.com/view/391122.htm" \t "http://baike.baidu.com/_blank)。

意见领袖是指在人际传播网络中经常为他人提供信息，同时对他人施加影响的“[活跃分子](http://baike.baidu.com/view/7216108.htm" \t "http://baike.baidu.com/_blank)”，他们在大众传播效果的形成过程中起着重要的中介或过滤的作用，由他们将信息扩散给受众，形成信息传递的两级传播。

作为一名股票投资者，有着两个基本需求。第一，要跟踪关注的股票，查看它们的价格涨跌、看跟它们有关的新闻，偶尔也看几眼分析师报告。第二，对别的投资者在想什么、看什么也很感兴趣，希望和跟自己关注同一家公司的投资者交流，看看他们对公司的看法跟自己有什么不同、他们的信息和思维有没有自己未曾触及的地方。然而，界面塞满花花绿绿的“K线图”的炒股软件，根据编辑判断而不是自己的需求编排的财经新闻网站，挤满了素未谋面、但自称有“内幕消息”的股民的聊天群，以及银行推销的各种号称能满足“个性化需求”的投资理财产品，等等的服务都满足不了个人需求。此时，雪球网出现了，实现了：订制化的信息（公司新闻、公告）和订制化的投资者讨论。当用户访问别人的个人页时，他能迅速看到这名用户最擅长讨论哪些股票；当他浏览一只股票的个股页面时，他能迅速看到关于这家公司最热门、最精华的讨论是什么。

而在这些用户中，一些专业并且靠谱的用户，成为了大“V”，即“意见领袖”。所谓的大“V”本是网络时代的产物，其实在证券行业，大“V”和“老师”一样，粉丝众多，受关注程度高于普通球友。他们所发表的对股票市场的分析言论，往往能影响到很多人，极端者甚至能影响到股票市场的波动。

1.1.2 研究意义

雪球网作为现今流行和强大的金融发声渠道，已经逐渐发展成为一个独立的舆论场，而雪球网意见领袖作为场域中心的重要节点，在现实股票市场与网络中扮演着重要角色，由此引发的影响力不容忽视。本文选择雪球网中最具代表性的意见领袖群体作为研究对象，把最直接的话语权力影响作为研究重点，不但顺应了新兴媒介发展和研究需要，同时也有利于意见领袖理论的延伸和拓展，对于指导意见领袖正确、有效、科学发挥话语权力具有积极作用。具体而言，一方面有利于意见领袖明确自身身份重要性，肩负起构建和谐话语环境、引导实现公众话语权的重任，另一方面，意见领袖要适当倾斜话语权重，平衡话语分配，树立良好影响力榜样，带动公众推动理性交往情景和网络公共领域的建构。

文章主要目的是获得雪球网意见领袖话语权力现状，并对意见领袖的言论与股票市场的波动之间的关系进行分析，期望能在意见领袖的带领下，一般用户能更好地在股票市场获取利益。

## 1.2 研究现状

雪球是一个投资者的社交网络,给投资者提供跨市场、跨品种的数据查询、新闻订阅和互动交流服务,目前已覆盖A股、港股、美股市场。形式比较类似于Twitter、微博，但算是一个金融领域社交网站，目前并没有对金融社交领域的意见领袖发现算法的研究与分析。

国外近些年的研究主要集中在对意见领袖的影响效果上。具体概括为以下三方面：（1）意见领袖的影响程度和作用过程。国外研究者大多采用样本分析法实证考量意见领袖在应用层面的影响，常与其它学科交叉应用，尤以市场领域最多。

1. 意见领袖的识别与评估。在文献[9]中，作者运用数学相关原理，根据社区结构，提取跟帖数据，研究网络社区中的意见领袖。
2. 具体领域内意见领袖的能力与行为探讨。例如文献[10]主要研究影响环境意见领袖行为的潜在因素，目的是通过意见领袖带领公众来保护环境，而对于意见领袖基础理论性的成果相对较少。在国外论著中，意见领袖作为专有词汇，很少直接与传播媒体搭配在一起使用，搜索关键词 opinion leaders 时，大部分指向网络意见领袖，如 Carolin Kaiser 等人。文献[11]则模拟意见领袖传播方式，研究意见领袖的网络社交营销影响。

总之，国外研究者多采用量化方法研究意见领袖的应用价值，研究面广而多，细分下的意见领袖研究不多见，关于意见领袖本体相关研究也不多见。

## 1.3 研究框架和方法

1.3.1 研究框架

论文共分为八个部分，具体如下：

第一章 交代论文研究背景、研究意义、研究现状，介绍雪球网的功能，明确它在金融社交领域占据重要地位，并了解如今对意见领袖研究的情况。

第二章 了解何为意见领袖，为什么要重视意见领袖，意见领袖的特征是什么，意见领袖的功能是什么。

第三章 考虑复杂网络模型，分析雪球网的网络特性，对比Blog网络，问答网络特性，得出金融社交网络更适合无向无权重网络。然后从现有的社交网络“微博”的意见领袖发现算法的研究出发，获得金融社交网络的意见领袖发现算法的启示，关注、转发网络使用PageRank排名算法，回复、收藏、打赏指标使用简单加权算法。

第四章 了解如何使用用层次分析法处理五个指标获得的数据，实现数据融合，从而使所有用户能在统一标准下按影响力分数大小进行排序。

第五章 使用语义分析、文本处理，对用户的回复进行分析，检查回复中是否含有与股票涨跌相关的信息，调用Tushare财经数据接口包，获得用户发表评论后10天内股票的收益情况，看是否与用户言论匹配，从而判断此算法是否合理和有意义。

第六章 整理整个实验过程的数据结构、方法。

第七章 整理最终的数据结果，并进行针对性分析。

第八章 对本文做出总结，以及对未来意见领袖研究领域研究的展望。

1.3.2 研究工具

（1）网络爬虫

网络爬虫是捜索引擎抓取系统的重要组成部分。爬虫的主要目的是将互联网上的网页下载到本地形成一个或联网内容的镜像备份。

利用Python爬虫扒取雪球网上相关信息，如用户ID和他的粉丝ID集合，每条twitter的ID和它对应的用户ID、日期、转发用户ID集合、回复数、收藏数、打赏数等等。

1. 数据库

将以上数据整理成表存储在MySQL的数据库中，方便查看与调用。

1. 编程工具

利用编程工具eclipse，使用编程语言Java，对数据进行分析处理。

# 第二章 意见领袖

## 2.1 意见领袖的概念

意见领袖是传播学中的经典概念之一。20 世纪40 年代，在传播效果研究上，传播学界盛行一种“魔弹”理论，也称为“子弹论”或“皮下注射论”，其核心观点是：传播媒介拥有不可抵抗的强大力量，它们所传递的信息就像子弹打中身体、药剂注入血液一样，可以引起直接的反映，左右人们的态度和意见，直接支配人们的行为。“传播被视为魔弹，它可以毫无阻拦地传递观念、情感、知识和欲望，传播似乎可以把某些东西注人的头脑，就像电流使电灯发亮一样直截了当。”当时针对大众传播的“万能说”、“枪弹论”，美国哥伦比亚大学的传播学者保罗·拉扎斯菲尔德在 1940 年美国总统选举期间，围绕着大众传播的竞选宣传，对选民投票意向的影响进行了一项历时 6个月的实证调查，提出了“意见领袖”和“两级传播”的概念，他认为：对于媒介传播的信息的观点，有部分受众会积极接受，并加以再传播，这些人即为“意见领袖”；而另一部分人则主要依靠与这些“意见领袖”的接触来指导自己的行动。

意见领袖是两级传播中的枢纽。两级传播论是拉扎斯菲尔德提出的，认为大众传播并不是直接流向一般受众，而是通过意见领袖这个中间环节，再从意见领袖流向一般人群，从而形成“大众传播—意见领袖—一般受众”这样的两级传播模式，通过研究人员比较，意见领袖拥有更多的主观兴趣，他们对媒介的接触频度和接触量都远远高于一般人。

而意见领袖处于“信息二次传播枢纽”。在二次传播中，意见领袖起到十分重要的节点位置，他既是信息的把关人，在众多的信息中选择一部分（或者由于兴趣、或者由于需要）向他人传播，也是信息的加工者，在二次传播的过程中意见领袖往往会加入自己对信息的理解、观点和态度，这些态度倾向也不可避免地影响到信息的接受者。

二级传播假说使人们认识到大众媒介渠道和人际传播这两种渠道在人们信息的获取、观点的形成和具体的行动中扮演的不同角色和作用。简单地说，大众传播的功能主要是告知，而人际渠道则在劝服方面更为重要。在创新的散布过程中，大众传播可以使信息迅速的抵达受众，实现大范围的信息普及。而人际传播使信息接受者对信息的了解更为深入、并且在这种双向的交流过程中，形成一致的意见，通过观念进而影响信息接受者的行为。

## 2.2 意见领袖的影响力

美国社会学家罗杰斯研究认为，大众传播分为信息流和影响流，信息流即媒介信息的传播是一级的，它可以像人们感觉的那样全部直接到达受众，而影响流的传播则是多级的，要经过大众信息的“意见领袖”的中介才能抵达受众。这里说的“意见领袖”是指活跃在人际传播网络中，经常为他人提供信息、观点或建议并对他人施加个人影响的人。

意见领袖即“在将媒介信息传给社会群体的过程中，那些扮演某种有影响力的中介角色。”从种种对意见领袖的描述中可以看出，影响力是意见领袖天然的能力，意见领袖之所以称之为意见领袖，就是因为他的意见对人们的影响。大众传播媒介只对少数人有影响，真正影响人们的是个人之间的接触和面对面的劝说。概念往往先从大众传播媒体流向舆论界的领导人，然后从这些意见领袖流向大众。也就是说，你不能影响意见领袖就不能有效影响大众。

卡兹和拉扎斯，菲尔德也证明了意见领袖在消费、时尚等其他社会生活领域同样具有影响力。

## 2.3 意见领袖的特征

意见领袖广泛存在于人际传播领域，几乎所有人都会受到意见领袖的影响。这些能够影响多数人的少数派，具有其自身的特征。

根据卡兹，拉扎斯菲尔德和罗杰斯所说，意见领袖具有以下基本特征。

(1) 意见领袖与被影响者一般处于平等关系而非上下级关系

意见领袖不等于现实中的“领导”。现实中的领导往往也具有意见领袖的知识丰富、社会地位和能力高于一般人的特征，但是现实中的领导可以成为意见领袖，但不是所有的意见领袖都是领导，也不是所有的领导都能成为意见领袖。而事实是，意见领袖与被影响者一般处于平等关系而非上下级关系，如果传受双方处于上下级关系，那么其作为“领导”而产生的影响是职位差异的权威性而赋予的，而不是作为意见领袖的影响力。与被影响者处于平等关系，对被影响者有种同类人的亲近感觉的意见领袖，这种平等关系的意见领袖一般是我们身边的同学、同事或者朋友，对他们的了解和熟悉使我们更信赖他们，而这些意见领袖的意见更能产生影响。

(2)意见领袖并不集中于特定的群体或阶层，而是均匀分布于社会上任何群体和阶层中

意见领袖的存在范围是非常广泛的，几乎所有有群体的地方都会有意见领袖的存在。因此，意见领袖并不集中于特定的群体或阶层，而是分布于社会的各个群体和阶层之中。除了一些高高在上的专家、明星等意见领袖，意见领袖也存在于在们的日常生活所处的普通群体之中。

（3）意见领袖的影响力一般分为“单一型”和“综合型”

意见领袖的影响力体现在自己最了解、擅长的某一方面或者几个方面，在其他方面，意见领袖就不能够产生影响，而可能作为一个普通的个体受其他领域意见领袖的影响。因此，意见领袖的影响力一般有“单一型”和“综合型”两种。一般来说，在现在都市生活中，意见领袖（特别是某一领域的专家或学者）的影响力通常是单一型的，其影响体现在某一方面。如一个具有大量医学方面知识的人可以对健康等相关问题对人产生影响，而在运动方面则会受到这一方面意见领袖的影响。而在传统社会或农村社会中，更多的存在着具有“综合型”影响力的意见领袖，这些意见领袖通常由某个有声望的家族、或者村里的长者来担任。村里无论发生什么事情都会先去问问这些意见领袖的意见，对当地的农业生产、人际关系、婚姻嫁娶都有着广泛的影响。

## 2.4 意见领袖的功能

（1）对信息的加工和解释

意见领袖通常是接触媒介信息比较频繁的人，较高的信息接收量是成为意见领袖的基础，但是与普通的个体不同，意见领袖对信息并不是单纯的接收，而是基于自己的知识结构、价值判断对信息进行加工与解释，意见领袖基于本身的“选择性注意、选择性理解、选择性记忆”，对接收的信息作出自己独到的判断和解释。

（2） 对信息的扩散和传播

意见的流动就是“同意”的流动。只有认同了这个观点，人们才会传播它。意见领袖并不是对所有接触到的信息都会进行二次传播，只有他们本身认可的信息才会进行二次传播，意见领袖对于认可的信息积极、义务、自发传播，而对于不认可的信息则采取不传播或者批判性传播的态度。

（3） 对信息的支配和引导

意见领袖对于信息进行二次传播，不可避免地带有“说服”的动机，没有人传播的信息的时候不期望得到受者的认同和接受。意见领袖不仅仅可以影响广大网民关注什么，也可以支配人们对于某个话题的态度。这种影响力是意见领袖具备的基本的能力，意见领袖通过自身的说服来对影响人们对于某一问题的态度，进而影响人们的行为。

（4） 对信息的协调和干扰

意见领袖对于其认同的信息进行积极地传播，对于不认同的信息不传播，这样起到信息流动的协调功能。对于不认同的信息意见领袖不仅不去传播，还有可能去批判性传播，传播与信息相对的负面信息，产生负面的效果。意见领袖对于不认同的信息，有着干扰信息传播的作用。

# 第三章 雪球网意见领袖发现算法分析

## 3.1 雪球网网络特征

金融社交网络是社会网络的一种形式，属于复杂网络的研究范畴一。由于网络社区信息传播多元化、匿名性和迅速传播的特点，使得研究者对于网络信息的形成、传播、衰变的追踪、实验与模拟困难霞霞，尽管国内外学者做了大量的研究。但足大部分都是从传播学、社会学以及心理学的某一角度进行研究，分析手段的限制使得大部分研究仍停船在定性阶段。随着复杂网络理论体系的不断完善，人际关系网络的识别及规律研究成为复杂网络中一个重要的研究方向。人际关系网络可看成相关联的人和群体的集合体。而在金融社交网络中，用户注册ID可以作为网络节点，用户之问通过对发帖的正文和回复关系建立，互相关注等方法产生联系。例如，用户B对A的文章进行回复，可以视为A和B发生一次关联。关注和转发这两个操作，最多只能实现一次，所以考虑无权重的网络模型更为合适。

复杂网络根据节点之问是否存在指向关系分为有向网络和无向网络．如图1所示。在人际关系网络中。又可根据权威指向关系．分为正向和反向两种。如图1 b和图1c所示。实际应用中。需要依据网络节点和连接关系的指代意义来进行合理网络选型。

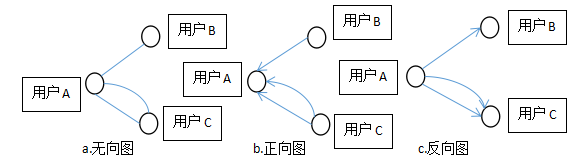


图3-1 网络结构选项

Blog是“自媒体”的代表之一，其核心特征是个人专属表达空间，博主们拥有自己的独立主页。个人Blog从属于社群Blog，社群Blog则属于更大系统的Blog社群。Blog网络通过Blog链接形成。博客内容的客观性成分更多，因此回帖人的行为更多是对发帖人文章的一种态度立场表达，意见领袖身份基本上都是发帖者，可以视为一种正向网络。

问答网络是一种求助网络，例如维基百科、百度知道等。其核心特征是发帖者询问问题．回帖者给予权威性回答，显然意见领袖身份都是回帖者，因此可以视为反向网络。

但是，金融社交网络由于其公共性强，个人性弱，言论方面会有很多的个性化特征和随意性，这就决定其主观成分多于客观成分。思想交锋的激烈程度更多，为形成社会性意见倾向提供了合适的场所。初始帖的作者(俗称楼主)一般是挑起话题的人物；对初始帖进行评论或继续讨论的一系列帖，称为回复帖，起到延续话题的作用，同时具有左右舆论导向的能力。因此，发帖人和回帖人中都可能出现意见领袖。因此，从网络选型上，无向网络更适合金融社交网络。

因此．无向、加权BBS人际网络模型中更能准确地识别出意见领袖。根据上文定性分析，雪球网关注/转发网络模型定义为G=(V，E，W)。V和W表示图G的子集。V中的点代表发帖用户(Vertex或Node)。E表示边集(Edge或Link)，edge<Vi，Vj>∈E，表示节点Vi和Vj之间存在交互关系。V中元素个数和E中元素个数分别成为网络的阶(Order)和边数(Size)。

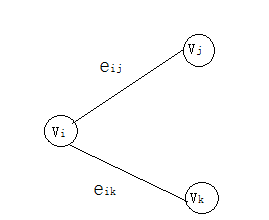


图3-2 无向无权重人际关系网络模型

## 3.2 现有社交网络“微博”的意见领袖发现算法

微博（Weibo），即微型博客（[MicroBlog](http://baike.baidu.com/view/4699059.htm" \t "http://baike.baidu.com/_blank)）的简称，也即是[博客](http://baike.baidu.com/subview/1509/4904688.htm" \t "http://baike.baidu.com/_blank)的一种，是一种通过关注机制分享简短实时信息的广播式的[社交网络](http://baike.baidu.com/subview/1405540/5023504.htm" \t "http://baike.baidu.com/_blank)平台。微博的构成方式跟雪球网十分相似，可以利用关系网络可以计算用户的影响力，从而可以选出影响力最高的用户，即意见领袖，进而进行舆情监测、社会热点监控。

微博的意见领袖发现算法借鉴了网页排名中的算法PageRank ，该算法由 google 创始人拉里·佩奇和谢尔盖·布林发明，随着 google 在商业上的成功而声名鹊起。该算法根据网页之间的链接来确定网页的排名，其核心在于一个假设，质量高的网页所指向的网页的质量必定也高。

根据 PageRank 的思想，可以得到微博上影响力的假设：影响力高的用户关注的用户的影响力必定也高。将用户看成是 PageRank 中的网页，将关注关系看做是网页中的链接关系。从而，可以根据 PageRank 的算法流程得到在微博关注网络上的影响力计算算法：

(1) 赋予所有用户相同的影响力权重;

(2) 将每个用户的影响力权重按照其关注的人数等量分配; 1/n

(3) 对每个用户来说，其影响力等于其粉丝分配给他的权重之和;

(4) 第 2 步和第 3 步迭代，直到权重不再发生大的变化为止。

如果只是基于关系网络的话，那么很容易就造成，粉丝数目多的人影响力必然会很高。这样就导致有些用户去购买一些僵尸粉就可以达到很高的影响力了。这样的算法显然是不能应对实际情况的，因为还有太多的信息没有用到。

当然，用户之间的回复关系、转发关系、@关系均可以构成网络，它们也有相应的假设:

（1）影响力越高的用户回复的微博的影响力越高，从而使该微博主人的影响力变高。

（2）影响力越高的用户转发的微博的影响力越高，从而使该微博原创作者的影响力变高。

（3）影响力越高的用户倾向于在其微博中@影响力高的用户。

这样就又得到了转发网络、回复网络、@网络三种网络，借鉴 PageRank 算法，可以得到另外的三种影响力结果。将它们与关系网络的影响力结果进行融合，就可以最终的影响力结果了。

## 3.3 PageRank算法

Google网页PageRank模型如图3-1所示，网页1被3个箭头所指向，假定共计得到了100分。然后通过链接平均将这1130分均分给它所指向的两个网页，网页2和网页3。平均分配是因为点击两个链接的可能性是相等的。由指定关系得到一定的分数，通过计算每个网页得到分数的多少评价网页重要性的方法，就是PageRank的基本设计方法。

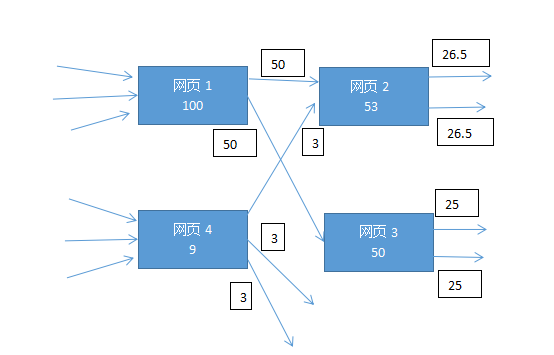


图3-3 PageRank计算方法

网页是彼此相连的，所以分数会不断传递下去，但PageRank的计算是收敛的，如图3-2所示，通过对三个网页分别计算它们的PageRank的一个迭代收敛的例子。

由图3-2可以看到这3个网页的分数之和无论如何循环总是1，这种特性在数学上称为“不变分布”，这也是PageRank计算收敛的原理。

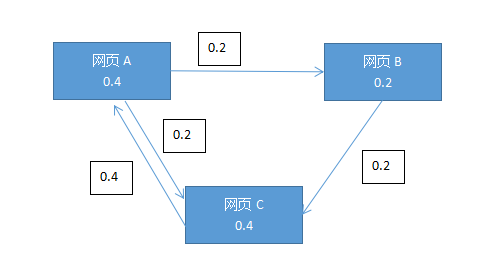


图3-4 PageRank的迭代收敛例图

 （3-1）

（1）：网页A的PageRank值。

（2）：网页存在指向A的链接，并且网页在上一次迭代时的PageRank值。

（3）：网页的外链数量。

（4）d:阻尼系数，0<d<1,表示在随机模型中网页将自身d的份额的PageRank值平均分给每个外链。（d一般为0.85）

PageRank算法的优点在于它对互联网上的网页给出了一个全局的重要性排序，并且算法的计算过程是可以离线完成的，这样有利于迅速响应用户的请求。不过，其缺点在于主题无关性，没有区分页面内的导航链接、广告链接和功能链接等，容易对广告页面有过高评价；另外，PageRank算法的另一弊端是，旧的页面等级会比新页面高，因为新页面，即使是非常好的页面，也不会有很多链接，除非他是一个站点的子站点。这就是PageRank需要多项算法结合的原因。

## 3.4 雪球网意见领袖发现算法

雪球网与微博的网络构成特别相似。用户可以随意关注另一个用户，查看该用户有关信息及发帖信息。对于每一个帖子，用户可以选择无视，也可以选择转发、回复、收藏、打赏。所以对于雪球网意见领袖发现算法的分析，可以参考微博的PageRank算法。

1. 关注网络：影响力高的用户关注的用户影响力必定也高。

（2）转发网络：影响力越高的用户转发的帖子的影响力越高，从而使该原创作者的影响力越大。

原本对于回复网络、收藏网络、打赏网络，也可采取类似的假设，运用PageRank算法进行运算。但是由于数据采集不全面的原因，只能采集到每个帖子的回复数、收藏数、打赏数，而无法获得究竟是那些用户进行了回复、收藏、打赏等操作，所以对于这些数据，采取的处理方法是，简单的加权处理。

1. 回复数：用户所有帖子加权平均回复数越大，用户影响力越大。

（4）收藏数：用户所有帖子加权平均收藏数越大，用户影响力越大。

（5）打赏数：用户所有帖子加权平均赞助数越大，用户影响力越大。

针对回复这一指标来说，假设对于某个用户，总共发了N个帖子，他所发每一个的帖子地位均等，即均为1/N，但是每个帖子权重Wi不一样，权重分别为每个帖子的回复数。同理，收藏与打赏。

一个用户的影响力=  （3-2）

# 第四章 数据融合算法

在以上的研究中，对于每个用户，已经分别算出了关注PR值，转发PR值，回复加权值，收藏加权值，打赏加权值，但是这五个结果之间不是简单的一比一的关系，必须计算出他们各自的权重，再进行相加。这需要用到数据融合算法“层次分析法”。

## 4.1 层次分析法

4.1.1 层次分析法定义

层次分析法(Analytic Hierarchy Process,简称AHP)是美国运筹学家、匹兹堡大学T. L. Saaty教授在20世纪70年代初期提出的， AHP是对定性问题进行定量分析的一种简便、灵活而又实用的多准则决策方法。它的特点是把复杂问题中的各种因素通过划分为相互联系的有序层次，使之条理化，根据对一定客观现实的主观判断结构（主要是两两比较）把专家意见和分析者的客观判断结果直接而有效地结合起来，将一层次元素两两比较的重要性进行定量描述。而后，利用[数学方法](http://baike.baidu.com/view/96884.htm" \t "http://baike.baidu.com/_blank)计算反映每一层次元素的相对重要性次序的权值，通过所有层次之间的总排序计算所有元素的相对权重并进行排序。

4.1.2 层次分析法的优点

（1）系统性

将对象视作系统，按照分解、比较、判断、综合的思维方式进行决策。成为成为继机理分析、统计分析之后发展起来的系统分析的重要工具；

1. 实用性

定性与定量相结合，能处理许多用传统的最优化技术无法着手的实际问题，应用范围很广，同时，这种方法使得决策者与决策分析者能够相互沟通，决策者甚至可以直接应用它，这就增加了决策的有效性；

1. 简洁性

计算简便，结果明确，具有中等文化程度的人即可以了解层次分析法的基本原理并掌握该法的基本步骤，容易被决策者了解和掌握。便于决策者直接了解和掌握。

4.1.3 层次分析法的局限

1. 囿旧

只能从原有的方案中优选一个出来，没有办法得出更好的新方案；

1. 粗略

该法中的比较、判断以及结果的计算过程都是粗糙的，不适用于精度较高的问题；

1. 主观

从建立层次结构模型到给出成对比较矩阵，人主观因素对整个过程的影响很大，这就使得结果难以让所有的决策者接受。当然采取专家群体判断的办法是克服这个缺点的一种途径。

4.1.4 层次分析法的算法

（1）构建四个指标对比矩阵

顾名思义，就是这四个指标两两相互比较，形成一个4阶矩阵。那么按照什么标准进行比较呢，这里有一个度量表：

表4-1 指标比较度量表

|  |  |
| --- | --- |
| Aij | 意义 |
| 1 | 元素 i 与元素 j 对上一层次因素的重要性相同 |
| 3 | 元素 i 比元素 j 略重要 |
| 5 | 元素 i 比元素 j 重要 |
| 7 | 元素 i 比元素 j 重要得多 |
| 9 | 元素 i 比元素 j 的极其重要 |
| 2n, n=1,2,3,4 | 元素 i 与 j 的重要性介于aij = 2n-1与aij = 2n+1 之间 |

反之，则

 （4-1）

那么现在可以按照以上的量表，对以上四个指标进行两两评判，得到一个矩阵A。

假设某一矩阵如下

表4-2 矩阵A

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | A1 | A2 | A3 | A4 |
| A1 | 1 | 9 | 7 | 5 |
| A2 | 1/9 | 1 | 1/3 | 1/5 |
| A3 | 1/7 | 3 | 1 | 1/3 |
| A4 | 1/5 | 5 | 3 | 1 |

（2）计算矩阵的特征向量和指标权重

A. 对矩阵的各列求和。

表4-3 矩阵A各列求和

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | A1 | A2 | A3 | A4 |
| A1 | 1 | 9 | 7 | 5 |
| A2 | 1/9 | 1 | 1/3 | 1/5 |
| A3 | 1/7 | 3 | 1 | 1/3 |
| A4 | 1/5 | 5 | 3 | 1 |
| SUM | 1.454 | 18.000 | 11.333 | 6.533 |

B. 对每一列进行归一化处理。

 （4-2）

其中，的值为各列的和，就是用各列的元素除以列的和，得到的结果为一个新的矩阵B。

表4-4 矩阵B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| B | B1 | B2 | B3 | B4 |
| B1 | 0.688 | 0.500 | 0.618 | 0.765 |
| B2 | 0.076 | 0.056 | 0.029 | 0.031 |
| B3 | 0.098 | 0.167 | 0.088 | 0.051 |
| B4 | 0.138 | 0.278 | 0.265 | 0.153 |
| SUM | 1.000 | 1.001 | 1.000 | 1.000 |

C. 对每一行进行求和，即得出特征向量。

表4-5 矩阵B各行求和

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| B | B1 | B2 | B3 | B4 | SUM |
| B1 | 0.688 | 0.500 | 0.618 | 0.765 | 2.571 |
| B2 | 0.076 | 0.056 | 0.029 | 0.031 | 0.192 |
| B3 | 0.098 | 0.167 | 0.088 | 0.051 | 0.404 |
| B4 | 0.138 | 0.278 | 0.265 | 0.153 | 0.834 |
| SUM | 1.000 | 1.001 | 1.000 | 1.000 | 4.001 |

D. 计算指标的权重：对特征向量进行归一化处理。（其含义同对矩阵归一化处理）

 （4-3）

表4-6 权重

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| B | B1 | B2 | B3 | B4 | SUM | W |
| B1 | 0.688 | 0.500 | 0.618 | 0.765 | 2.571 | 0.643 |
| B2 | 0.076 | 0.056 | 0.029 | 0.031 | 0.192 | 0.048 |
| B3 | 0.098 | 0.167 | 0.088 | 0.051 | 0.404 | 0.101 |
| B4 | 0.138 | 0.278 | 0.265 | 0.153 | 0.834 | 0.208 |
| SUM | 1.000 | 1.001 | 1.000 | 1.000 | 4.001 | 1.000 |

# 第五章 评价算法的合理性

以上的研究中，已经在一个统一标准下，将所有用户的分数计算出来，接下来，直接将用户按分数从高到低排序，排名靠前的即为雪球网意见领袖。

那么该如何评价通过这一算法得出的意见领袖是否具有现实性，即这一算法是否具有合理性呢？可以通过研究这些意见领袖关于股票市场发表的言论的正确率来判断，具体来说，就是意见领袖发表了某一股票会涨/跌的消息，在一段时间内，这只股票涨/跌了，说明他的言论是正确的。这就涉及到语义分析的技术，还有获取股票市场数据的Tushare数据接口包的使用。

## 5.1 语义分析

[语义分析](http://baike.baidu.com/view/487035.htm" \t "http://baike.baidu.com/_blank)是[编译](http://baike.baidu.com/view/69568.htm" \t "http://baike.baidu.com/_blank)过程的一个逻辑阶段， 语义分析的任务是对结构上正确的[源程序](http://baike.baidu.com/view/546605.htm" \t "http://baike.baidu.com/_blank)进行上下文有关性质的审查，进行类型审查。语义分析是审查[源程序](http://baike.baidu.com/view/546605.htm" \t "http://baike.baidu.com/_blank)有无语义错误，为[代码生成](http://baike.baidu.com/view/8477718.htm" \t "http://baike.baidu.com/_blank)阶段收集类型信息。比如语义分析的一个工作是进行类型审查，审查每个算符是否具有语言规范允许的运算对象，当不符合语言规范时，[编译程序](http://baike.baidu.com/view/454895.htm" \t "http://baike.baidu.com/_blank)应报告错误。如有的[编译程序](http://baike.baidu.com/view/454895.htm" \t "http://baike.baidu.com/_blank)要对实数用作[数组](http://baike.baidu.com/view/209670.htm" \t "http://baike.baidu.com/_blank)下标的情况报告错误。又比如某些某些程序规定运算对象可被强制，那么当二目运算施于一[整型](http://baike.baidu.com/view/1311503.htm" \t "http://baike.baidu.com/_blank)和一实型[对象](http://baike.baidu.com/subview/2387/10666278.htm" \t "http://baike.baidu.com/_blank)时，编译程序应将整型转换为实型而不能认为是[源程序](http://baike.baidu.com/view/546605.htm" \t "http://baike.baidu.com/_blank)的错误。

在讲文本语义分析之前，我们先说下文本基本处理，因为它构成了语义分析的基础。

拿到一段文本后，通常情况下，首先要做分词。分词的方法一般有如下几种：

1. 基于字符串匹配的分词方法

此方法按照不同的扫描方式，逐个查找词库进行分词。根据扫描方式可细分为：正向最大匹配，反向最大匹配，双向最大匹配，最小切分(即最短路径)；总之就是各种不同的启发规则。

1. 全切分方法

它首先切分出与词库匹配的所有可能的词，再运用统计语言模型决定最优的切分结果。它的优点在于可以解决分词中的歧义问题。下图是一个示例，对于文本串“南京市长江大桥”，首先进行词条检索(一般用Trie存储)，找到匹配的所有词条（南京，市，长江，大桥，南京市，长江大桥，市长，江大桥，江大，桥），以词网格(word lattices)形式表示，接着做路径搜索，基于统计语言模型(例如n-gram)找到最优路径，最后可能还需要命名实体识别。下图中“南京市 长江 大桥”的语言模型得分，即P(南京市，长江，大桥)最高，则为最优切分。

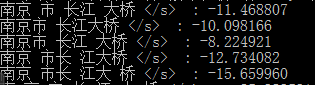


图5-1 “南京市长江大桥”语言模型得分

1. 由字构词的分词方法

可以理解为字的分类问题，也就是自然语言处理中的sequence labeling问题，通常做法里利用HMM，MAXENT，MEMM，CRF等预测文本串每个字的tag[62]，譬如B，E，I，S，这四个tag分别表示：beginning, inside, ending, single，也就是一个词的开始，中间，结束，以及单个字的词。 例如“南京市长江大桥”的标注结果可能为：“南(B)京(I)市(E)长(B)江(E)大(B)桥(E)”。由于CRF既可以像最大熵模型一样加各种领域feature，又避免了HMM的齐次马尔科夫假设，所以基于CRF的分词目前是效果最好的。

对文本分词后，接下来需要对分词后的每个term计算一个权重，重要的term应该给与更高的权重。

Term Weighting的打分公式一般由三部分组成：local，global和normalization 。即

 (5-1)  
 其中L\_{i,j}是term i在document j中的local weight，G\_i是term i的global weight，N\_j是document j的归一化因子。  
常见的local，global，normalization weight公式有

表5-1 Local weight formulas

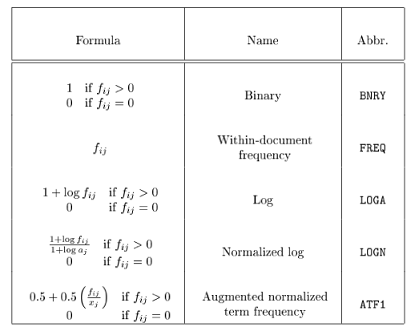


表5-2 Global weight formulas

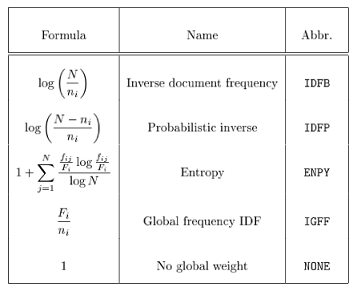
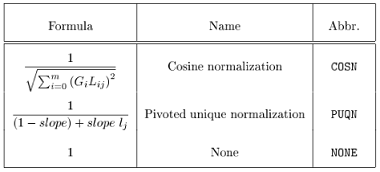


表5-3 Normalization factors



Tf-Idf是一种最常见的term weighting方法。在上面的公式体系里，Tf-Idf的local weight是FREQ，glocal weight是IDFB，normalization是None。tf是词频，表示这个词出现的次数。df是文档频率，表示这个词在多少个文档中出现。TD表示总文档数。idf则是逆文档频率

 （5-2）

## 5.2 Tushare财经数据接口包

TuShare是一个免费、开源的python财经数据接口包。主要实现对股票等金融数据从数据采集、清洗加工 到 数据存储的过程，能够为金融分析人员提供快速、整洁、和多样的便于分析的数据，为他们在数据获取方面极大地减轻工作量，使他们更加专注于策略和模型的研究与实现上。考虑到Python pandas包在金融量化分析中体现出的优势，TuShare返回的绝大部分的数据格式都是pandas DataFrame类型，非常便于用pandas/NumPy/Matplotlib进行数据分析和可视化。

[通联数据](https://m.datayes.com/" \t "http://tushare.waditu.com/_blank)（DataYes）是国内目前最大的开放金融数据平台，整合了包括股票、基金、期货、期权和港股方面的全品类金融数据。从数据的多样性、质量性和稳定性的角度考虑，遵从TuShare一贯的开放、简单易用的特点，TuShare为用户集成了绝大部分通联数据的接口。

Data API 返回HTTP JSON和CSV格式数据， 任何语言都可以直接调用HTTP API， 以获得数据。由于API使用安全 性高的https协议传输，程序访问必须使用token作为请求头，token可以在数据开放平台中获取。

获取到凭证后，就可以使用其来调用目标API，调用API时，只需将所需参数按照统一规范拼装一个正确的URL，通过https请求，就能够拿到需要的数据。API一次调用，最多返回10万条记录，因为基于https协议传输的API，一次性传输太大的内容，网络传输过程中可能出错。希望通过输入参数的合理设置，将一次返回的数据量，控制在可以接受的范围。

每一次调用都会得到对应的返回，当HTTP状态码是200时，表示调用成功。根据用户的选择，分别会返回json或者csv格式的内容。 对于json的请求方式，返回格式形如{“retCode”:1,“retMsg”:“Success”,“data”:“数据内容”}。

"retCode"表示查询操作的状态代码, "retMsg"表示查询操作的返回消息。 以下是状态代码和消息的对应关系：

表5-4 状态代码和消息的对应关系

|  |  |  |
| --- | --- | --- |
| retCode | retMsg | 描述 |
| 1 | Success | 成功且有数据 |
| -1 | No Data Returned | 无数据返回 |
| -2 | Illegal Request Parameter | 输入参数不合法 |
| -3 | Service Suspend | 系统暂停服务 |
| -4 | Server Error | 系统错误 |
| -5 | Server Busy | 系统繁忙 |

相关的API为获取一只股票历史上某一时间段的因子数据

getStockFactorsDateRange

/api/market/getStockFactorsDateRange.json?field=ticker,tradeDate,pe&secID=&ticker=000001&beginDate=20150101&endDate=20150701

表5-5 所需的请求字段

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 必填 | 描述 |
| secID | String | 2选1 | 一只股票ID |
| ticket | String | 2选1 | 一只股票交易代码 |
| beginDate | Date | 否 | 开始日期 |
| endDate | Date | 否 | 结束日期 |
| field | String | 是 | 可选参数，用“,”分隔，默认为空，返回全部字段，不选即为默认值。 |

本算法应用的返回字段，即得到10天内股票的涨跌情况。

表5-6 返回字段

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| REVS10 | Double | 股票的10日收益，属于超卖超卖型因子。 |

# 第六章 数据处理

本章详细解析了金融领域意见领袖算法，使用Python网络爬虫工具获得雪球网数据，并存储在数据库MySQL中，使用Java编程语言处理其中相关数据，实现找到意见领袖的目的，同时研究意见领袖的言论与股市波动之间的关系。

## 6.1 采集到的数据

表6-1 雪球网用户信息表xueqiu\_user

|  |  |  |
| --- | --- | --- |
| 名 | 类型 | 说明 |
| userID | varchar(15) | 用户ID |
| userName | varchar(40) | 用户名字 |
| userProfile | varchar(40) | 用户配置文件 |
| userGender | varchar(40) | 用户性别 |
| userProvince | varchar(40) | 用户所在省份 |
| userDescription | varchar(255) | 用户签名 |
| userVerified | varchar(40) | 用户是否已验证 |
| userVerifiedDescription | varchar(40) | 用户验证签名 |
| userDiscussStocks | varchar(40) | 用户讨论的股票相关信息 |
| userTweetsCount | int | 用户发布帖子的总数 |
| userTweet | mediumtext | 用户所有帖子的ID |
| userFollowersCount | int | 用户粉丝总数 |
| userFollowers | mediumtext | 用户所有粉丝的ID |
| userAttentionCount | int | 用户关注其他用户的总数 |
| userAttention | mediumtext | 用户所关注其他用户的ID |
| userStocksCount | int | 用户投资的股票总数 |
| userStocks | mediumtext | 用户投资的股票的ID和名字 |
| userGroupsCount | int | 用户群组个数 |
| userGroups | mediumtext | 用户群组ID |
| updateTime | timestamp | 用户信息更新时间 |

表6-2 雪球网帖子信息表xueqiu\_tweet

|  |  |  |
| --- | --- | --- |
| 名 | 类型 | 说明 |
| tweetID | varchar(15) | 帖子ID |
| tweetUserID | varchar(15) | 帖子对应的用户ID |
| tweetTime | timestamp | 帖子发布的时间 |
| tweetTitle | varchar(255) | 帖子题目 |
| tweetText | mediumtext | 帖子内容 |
| tweetRetweetID | varchar(15) | 转发本帖子的帖子ID |
| tweetRetweetUserID | varchar(15) | 转发本帖子的用户ID |
| tweetReplyCount | int | 帖子被回复数 |
| tweetRetweetCount | int | 帖子被转发数 |
| tweetDonateCount | int | 帖子被打赏数 |
| tweetFavCount | int | 帖子被收藏数 |
| tweetSource | varchar(40) | 帖子来源网站 |
| updateTime | timestamp | 帖子数据更新时间 |
| tweetSentiment | tinyint | 帖子情感值 |

表6-3 股票与帖子的对应关系表stock\_tweet

|  |  |  |
| --- | --- | --- |
| 名 | 类型 | 说明 |
| tweetID | varchar(15) | 帖子ID |
| stockID | varchar(15) | 股票交易代码 |
| updateTime | timestamp | 数据更新时间 |

## 6.2 类关系图

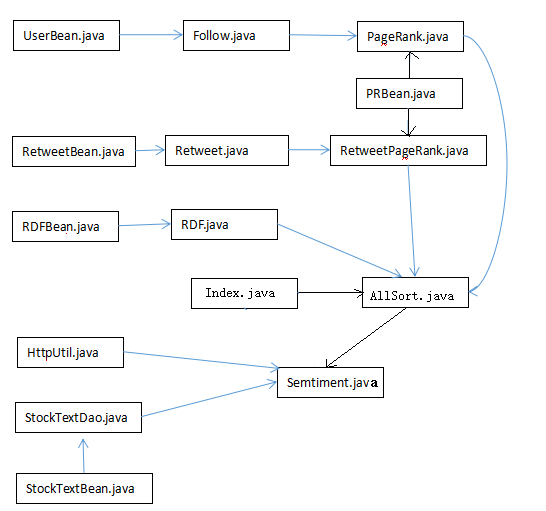


图6-1 类关系图

类之间的关系说明：

1. 类UserBean封装用户ID和粉丝ID，类Follow调用类UserBean，用于获得数据库内满足条件的所有用户ID和对应粉丝 ID。
2. 类PRBean封装用户ID，临时PageRank分数和最终PageRank分数，类PageRank调用类Follow和类PRBean,将满足条件的用户进行PageRank循环计算，每次计算临时存储在PRBean中，直到最终收敛于某个阈值，得出他们所有人各自的最终PageRank关注分数。
3. 类RetweetBean封装用户ID和转发用户ID，类Retweet调用类RetweetBean，用于获得数据库内满足条件的所有用户ID和转发用户ID。
4. 类PageRank调用类Retweet和类PRBean,将满足条件的用户进行PageRank循环计算，每次计算临时存储在PRBean中，直到最终收敛于某个阈值，得出他们所有人各自的最终PageRank转发分数。
5. 类RDFBean封装用户ID，发帖总数，回复分数，收藏分数和打赏分数，类RDF调用类RDFBean，用于获得数据库内满足条件的所有用户ID，发帖总数，回复数，收藏数和打赏数，然后分别求出三个指标的平均加权分数。
6. 类Index使用层次分析法，获得五个指标的权重。
7. 类AllSort调用了类Index和类PageRank，类RetweetPageRank，类RDF，进行加权计算，算出所有用户的总分数，并进行从高到低排序，排名靠前的即是意见领袖。
8. 类StockTextBean封装了股票交易代码，发帖文字和发帖时间，类StockTextDao调用StockTextBean获得特定用户所有帖子的股票交易代码，发帖文字和对应的发帖时间。
9. 类HttpUtil用于联通Tushare，通过API获得某天10天后某支股票的涨跌情况。
10. 类Semtiment调用类AllSort，获得所有总分数不为零的用户，调用类StockTextDao获得用户所有帖子的交易代码，发帖文字和发帖时间，对比发帖文字是否包含正面/负面的情感词语，然后调用类HttpUtil获得这些股票在发帖10天后的涨跌情况。通过比较文字与股市变化，计算这些用户言论的正确率。

## 6.3 关注网络

6.3.1 封装对象UserBean.java

作用：将用户ID，和粉丝ID数组封装起来，方便调用。

私有变量:

1. private String userID;
2. private String[] followerID;

公有方法:

（1）public String getUserID();

（2）public void setUserID(String userID;

（3）public String[] getFollowerID();

（4）public void setFollowerID(String[] followerID);

（5）public String toString();

6.3.2 与数据库交互Follower.java

公有方法：

1. public Connection getConnection();

作用：与数据库获得连接。

（2）public List<UserBean> getFollowers();

作用：获得满足条件的用户ID，和粉丝ID集合。因为数据库里存储的粉丝ID字符串形式如“[111111,222222,333333]”这种类型，所以必须将字符串进行处理，先按“,”分割，然后去掉第一个字符串的第一个字符“[”，去掉最后一个字符串的最后一个字符“]”。

6.3.3 封装PRBean.java

作用：将用户ID，暂存的PageRank分数，最终的PageRank分数封装成对象。

私有变量：

1. private String UserID;
2. private double tmpPR;
3. private double PR;

公有方法：

1. public String getUserID();
2. public void setUserID(String userID);
3. public double getTmpPR();
4. public void setTmpPR(double tmpPR);
5. public double getPR();
6. public void setPR(double PR);
7. public String toString();

6.3.4 关注网络的pageRank算法PageRank.java

私有变量：

1. private Follower follow = new Follower();

作用：方便调用数据库的用户ID，粉丝ID

1. private Hashtable<String, UserBean> hashBean = new Hashtable<String, UserBean>();

作用：将用户ID和对象UserBean（内含相同的用户ID，及对应的粉丝ID数组）构成一张哈希表，方便查询使用。

（3) private Hashtable<String, PRBean> hashPR = new Hashtable<String, PRBean>();

作用：将用户ID和对象PRBean（内含相同的用户ID，暂存用户的PageRank分数，最终的PageRank分数）构成一张哈希表，方便查询使用。

公有方法：

1. public void initPR();

作用：通过getFollowers()函数获得的UserBean对象集合，遍历所有符合条件的用户ID，将对应的UserBean对象放置在哈希表hashBean中。同时每个用户ID建立一个对应的PRBean对象，使对象的暂存的PageRank分数和最终的PageRank分数都初始化为1.0，并把对象放置到哈希表hashPR中。

1. public void calculatePR();

作用：遍历哈希表hashPR，对于每个用户，将暂存PageRank分数初始化为0，同时在哈希表hashBean找到对应的UserBean对象，获得粉丝ID集合。遍历粉丝ID，通过Array.getLength(Array)获得粉丝的粉丝数，同时通过ID在哈希表hashPR中获得PageRank分数，则用户暂存的PageRank分数为，所有粉丝PageRank分数除以他们各自的粉丝数，再相加。

（3）public boolean updatePR()；

作用：遍历哈希表hashPR，对于每个用户，如果暂存的PageRank分数与最终PageRank分数的绝对值之差小于0.001，则停止所有计算，返回true，并将暂存的PageRannk分数赋予最终PageRank分数，否则则返回false。

（4）public Hashtable<String, PRBean> getFollowPR();

作用：调用以上函数，先初始化两个哈希表，设置布尔值flag，初始化为false，循环调用pageRank.calculatePR()函数和flag = pageRank.updatePR()函数，直到flag等于true时停止循环，返回哈希表hashPR。

## 6.4 转发网络

6.4.1封装对象RetweetBean

作用：将用户ID，和转发用户ID封装起来，方便调用。（因为观察到数据库里转发用户最多只有一个，所以没用数组形式）。

私有变量：

1. private String userID;
2. private String retweetUserID;

成员变量

（1）public String getUserID();

（2）public void setUserID(String userID);

（3）public String getRetweetUserID();

（4）public void setRetweetUserID(String retweetUserID);

（5）public String toString();

6.4.2 与数据库交互Retweet.java

公有方法：

（1）public Connection getConnection();

作用：与数据库获得连接。

1. public List<RetweetBean> getRetweet();

作用：获得满足条件的用户ID，和粉丝ID集合。

6.4.3 封装PRBean.java（直接调用关注网络的PRBean类）

6.4.4 转发网络的pageRank算法PageRank.java

私有变量：

（1）private Retweet retweetDao = new Retweet();

作用：方便调用数据库的用户ID，转发用户ID

1. private Hashtable<String, RetweetBean> hashBean

= new Hashtable<String, RetweetBean>();

作用：将用户ID和对象RetweetBean（内含相同的用户ID，及对应的用户ID）构成一张哈希表，方便查询使用。

1. private Hashtable<String, PRBean> hashPR

= new Hashtable<String, PRBean>();

作用：将用户ID和对象PRBean（内含相同的用户ID，暂存用户的PageRank分数，最终的PageRank分数）构成一张哈希表，方便查询使用。

公有方法：

（1）public void initPR();

作用：通过getFollowers()函数获得的TweetBean对象集合，遍历所有符合条件的用户ID，将对应的TweetBean对象放置在哈希表hashBean中。同时每个用户ID建立一个对应的PRBean对象，使对象的暂存的PageRank分数和最终的PageRank分数都初始化为1.0，并把对象放置到哈希表hashPR中。

（2）public void calculatePR();

作用：遍历哈希表hashPR，对于每个用户，将暂存PageRank分数初始化为0，同时在哈希表hashBean找到对应的TweetBean对象，获得转发用户ID。若返回的是0，说明没被转发，若返回的是-1，说明帖子被删了。同时通过转发用户ID在哈希表hashPR中获得PageRank分数，则用户暂存的PageRank分数为，所有转发用户PageRank分数相加。

（3）public boolean updatePR()；

作用：遍历哈希表hashPR，对于每个用户，如果暂存的PageRank分数与最终PageRank分数的绝对值之差小于0.001，则停止所有计算，返回true，并将暂存的PageRannk分数赋予最终PageRank分数，否则则返回false。

（4）public Hashtable<String, PRBean> getFollowPR();

作用：调用以上函数，先初始化两个哈希表，设置布尔值flag，初始化为false，循环调用pageRank.calculatePR()函数和flag = pageRank.updatePR()函数，直到flag等于true时停止循环，返回哈希表hashPR。

## 6.5 回复/收藏/打赏

6.5.1 封装对象RDFBean.java

作用：将用户ID，发帖总数，回复指标的分数，收藏指标的分数，打赏指标的分数封装成对象。

私有变量：

（1）private String userID;

（2）private int count;

（3）private double replyValue;

（4）private double donateValue;

（5）private double favoriteValue;

公有方法：

（1）public RDFBean();

（2）public String getUserID();

（3）public void setUserID(String userID);  
（4）public int getCount()；

（5）public void setCount(int count)；

（6）public double getReplyValue()；

（7）public void setReplyValue(double replyValue)；

（8）public double getDonateValue()；

（9）public void setDonateValue(double donateValue)；

（10）public double getFavoriteValue()；

（11）public void setFavoriteValue(double favoriteValue)

（12）public String toString()；

6.5.2 分别计算回复数/收藏数/打赏数的加权分数RDF.java

私有变量：

（1）private Hashtable<String, RDFBean> hashRDF = new Hashtable<String, RDFBean>();

作用：将用户ID和对象（内含相同的用户ID，及对应的用户ID，总发帖数，回复/收藏/打赏指标分数）构成一张哈希表，方便查询使用。

公有方法：

（1）public Connection getConnection();

作用：与数据库获得连接。

（2）public List<RDFBean> getCountValue();

作用：获得满足条件的用户ID，和他们各自的发帖数。

（3）public void getEachRDF(RDFBean bean);

作用：获得某一用户满足条件的所有帖子各自的回复数/收藏数/打赏数。对于回复指标来说，所有帖子原始分数为1/总帖子数，权重分别为各自的回复数，将原始分数乘以权重，再想加，就是回复指标分数。同理收藏指标，打赏指标。然后将分数添加到对象RDFBean中，最后添加到哈希表hashRDF对应位置。

（4）public Hashtable<String, RDFBean> getRDFValue();

作用：调用以上函数，获得所有用户各自总发帖数，再对每个用户计算分数，更新哈希表，最后返回哈希表hashRDF。

## 6.6层次分析法

6.6.1 用层次分析法计算权重Index.java  
私有变量：

（1）private double[][] A\_index =

{

{1, 3, 7, 7, 7,},

{1/3, 1, 5, 5, 5,},

{1/7, 1/5, 1, 1, 1,},

{1/7, 1/5, 1, 1, 1,},

{1/7, 1/5, 1, 1, 1,} };

作用：比较五个指标，关注PageRank分数，转发PageRank粉丝，回复指标分数，打赏指标分数，收藏指标分数，两两之间的重要性。假设关注比转发重要3倍，比回复/打赏/收藏重要7倍，转发比回复/打赏/收藏重要5倍，回复/打赏/收藏重要性相同。

1. public double[] getColSum();

作用：对矩阵A\_index的各项求和。

1. public double[][] getNewIndex();

作用：对每一列进行归一化处理，即用各列元素除以列的和，得到新矩阵B\_index。

（4）public double[] getRowSum();

作用：对每一行进行求和，即得出特征向量。

（5）public double[] getWeight();

作用：对特征向量进行归一化处理，即用各行的和除以总指标数5，得到五个指标的权重数组。

## 6.7 意见领袖排序

6.7.1 排序算法AllSort.java

私有变量：

1. private Hashtable<String, Double> sortHash = new Hashtable<String, Double>();

作用：将用户ID和对应的总分数放在哈希表sortHash，方便查找使用。

公有方法：

（2）public Hashtable<String, Double> sort();

作用：获得关注网络哈希表followerHash，转发网络哈希表retweetHash，回复/收藏/打赏指标哈希表rdfHash，按用户ID，找到对应的分数，同时，获得通过层次分析法获得的权重数。将权重和对应分数相乘，再相加，然后乘以100（得到百分比形式），最终得到的就是用户的总的分数，放入哈希表sortHash对应的位置。然后将哈希表sortHash按分数从高到低进行排序，返回最终的哈希表。

## 6.8 评价

6.8.1 封装对象StockTextBean.java

作用：将股票交易代码，发帖文字，发帖时间封装起来，方便调用。

私有变量：

（1）private String stockID;

（2）private Timestamp tweetTime;

（3）private String tweetText;

公有变量：

（1）public String getStockID();

（2）public void setStockID(String stockID);

（3）public String getTweetText();

（4）public void setTweetText(String tweetText);

（5）public Timestamp getTweetTime();

（6）public void setTweetTime(Timestamp tweetTime);

（7）public String toString();

6.8.2 与数据库交互StockTextDao.java

公有方法：

（1）public Connection getConnection();

作用：与数据库获得连接。

1. public List<StockTextBean> getTweetText(String tweetUserID);

作用：获得某一特定用户的所有帖子对应的股票交易代码。

6.8.3 获得Tushare通联数据HttpUtil.java

私有变量：

1. private static CloseableHttpClient httpClient = createHttpsClient();

作用：创建http\_client，准备与Tushare接口连接。

1. private static final String ACCESS\_TOKEN =

"b03b7e5229797ed661ce6e88364e3dbff1442aaa271ef7569096b378b74cf7de";

作用：在Tushare中注册获得的认证token，才有获得Tushare数据的权限。

公有方法：

1. public static CloseableHttpClient createHttpsClient();

作用：创建http\_client。

1. public int isUp(String stockID, String beginDate, String endDate) throws Exception;

作用：根据api store页面上实际的api url来发送get请求，获取数据，调用的API是通过股票交易代码，查询开始时间，查询结束时间，去获得股票十日内收益。返回的是json格式，所以需要进行解析。若收益大于零，则返回1，收益小于零，返回-1，若无该股票，则返回-2。

6.8.4 判断意见领袖言论的正确性Semtiment.java

私有变量：

1. private static final String basePath = "E:/Javaworkspace";

作用：因为函数里涉及到文件绝对路径，设置工程根路径basePath，方便移植代码的时候使用。

公有方法：

1. public boolean isContain(String text, String fileName) throws IOException;

作用：使用情感词典（多个文件，里面每行为一个词语，语义为正面或者负面），查询某个字符串是否在词典中。

1. public static void main(String[] args) throws Exception;

作用：找到按总分数排序出来的意见领袖对应的帖子言论，帖子日期，及帖子对应的股票代码。查询帖子言论是正面还是负面，同时通过帖子日期，股票代码查询在帖子发布十天后，股票是涨还是跌。若帖子是正面且股票涨了，则言论是正确的，该意见领袖是有意义的，股票言论数加一，正确言论数加一，若帖子是负面的且股票跌了，则言论是错误的，只股票言论数加一，则该意见领袖是无意义的。最终求得每个用户的言论正确率。

# 第七章 结果分析

## 7.1 意见领袖

以上算法的数据测试中，帖子ID限制在26870000与26880000之间，用户ID限制在5000000000与6000000000之间。

在进行排序操作后，数据如下：

图7-1 算法得出的意见领袖

|  |  |  |  |
| --- | --- | --- | --- |
| userID | score | userID | score |
| 5819606767 | 374.71192142696805 | 5199849053 | 211.45524593596846 |
| 5190345050 | 147.38132178098977 | 5734150735 | 127.13045081907369 |
| 5826452318 | 86.380843786201 | 5457080143 | 86.10318739576657 |
| 5472442560 | 84.73051007249381 | 5480875446 | 82.28330722175323 |
| 5158708788 | 80.38241497058738 | 5271260195 | 76.9758220113558 |
| 5548624395 | 74.37412930021945 | 5091204599 | 73.0 |
| 5376236220 | 56.19754564312153 | 5063348038 | 51.89241503795131 |
| 5267456690 | 46.257759898905576 | 5052657219 | 40.816868245887605 |
| 5649506512 | 39.16044638527519 | 5749132507 | 38.14047276613307 |
| 5405184747 | 37.0564125199763 | 5057887409 | 35.67662220343038 |
| 5057887409 | 35.67662220343038 | 5805831020 | 35.54509052522607 |
| 5545011370 | 34.587292223366894 | 5467872602 | 34.0 |
| 5974914861 | 33.15854217220316 | 5340328453 | 32.99999999999999 |
| 5514308902 | 32.8789140023917 | 5244250711 | 32.875 |
| 5185865531 | 32.51534438863697 | 5033249022 | 31.88221472701792 |
| 5514474822 | 30.485605824429207 | 5511195936 | 30.013707037643204 |
| 5410863005 | 29.92364462106159 | 5765429529 | 29.92364462106159 |
| 5721600122 | 29.345088798573553 | 5187960654 | 29.215033039647576 |
| 5800488532 | 28.999999999999993 | 5874242620 | 28.999999999999993 |
| 5462126276 | 28.999999999999993 | 5768850874 | 28.999999999999993 |
| 5502895643 | 28.719329499676032 | 5296621426 | 28.516273185884838 |
| 5104102858 | 28.398299900393432 | 5629459650 | 27.545612992562855 |
| 5262091929 | 27.4453947368421 | 5916818864 | 27.12673102669232 |
| 5152081633 | 27.121130030959744 | 5432738147 | 26.999999999999996 |
| 5160111270 | 26.866113154719486 | 5252177556 | 26.666943029890234 |
| 5648255828 | 26.555397916978684 | 5059969504 | 26.43181818181818 |
| 5024698911 | 26.372082333519508 | 5582784865 | 26.353496812221294 |
| 5793686612 | 26.317696633603067 | 5238646294 | 26.270063716111654 |
| 5963263464 | 25.760329440284785 | 5063748394 | 25.63636363636363 |
| 5410800119 | 25.603623188405795 | 5951114167 | 25.521833041447746 |
| 5899108858 | 25.40872608042842 | 5951480610 | 25.394923388671724 |
| 5601404303 | 25.30143540669856 | 5180885411 | 25.30143540669856 |
| 5336369061 | 25.11590477177116 | 5894505703 | 25.07954545454545 |
| 5288693117 | 25.07954545454545 | 5272978393 | 25.055248758145325 |
| 5724929197 | 25.05516637478108 | 5291597824 | 25.05516637478108 |
| 5035790932 | 25.026515151515145 | 5732670814 | 25.02277252208309 |
| 5481335926 | 25.013171978248895 | 5667609925 | 25.013171978248895 |
| 5461237762 | 25.013171978248895 | 5513217170 | 25.013171978248895 |
| 5453117137 | 25.013094799524094 | 5504830612 | 25.012740825730873 |
| 5205526918 | 25.010331156604877 | 5807584892 | 25.010329440284785 |
| 5132528889 | 25.00792459590999 | 5992204306 | 25.007328137722457 |
| 5346281959 | 25.006590669139733 | 5952472219 | 25.006590669139733 |
| 5441695769 | 25.006590669139733 | 5433029961 | 25.005600995732568 |
| 5988544051 | 25.005600995732568 | 5032359409 | 25.005600995732568 |
| 5710011705 | 25.005600995732568 | 5128859884 | 25.004759386567947 |
| 5857787366 | 25.004010593518284 | 5280009877 | 25.003914002391696 |
| 5624808117 | 25.003914002391696 | 5525751336 | 25.00383328262853 |
| 5441278387 | 25.003004206224528 | 5514804700 | 25.001807851239665 |
| 5012499006 | 25.001512350865397 | 5673164838 | 25.001282631211364 |
| 5833114470 | 25.00055456828598 | 5843000265 | 25.00033066585167 |
| 5451142347 | 25.00008238336424 | 5862823850 | 25.00003979870698 |
| 5096958202 | 24.999999999999993 | 5268034968 | 24.999999999999993 |
| 5268034968 | 24.999999999999993 | 5405384597 | 24.999999999999993 |
| 5152187219 | 24.999999999999993 | 5632367028 | 24.999999999999993 |
| 5201984462 | 24.999999999999993 | 5733332710 | 24.999999999999993 |
| 5537434717 | 24.999999999999993 | 5968175953 | 24.999999999999993 |
| 5248615563 | 24.999999999999993 | 5128391587 | 24.999999999999993 |
| 5125410266 | 24.999999999999993 | 5409020661 | 24.999999999999993 |
| 5785584799 | 24.999999999999993 | 5817788557 | 24.999999999999993 |
| 5294835973 | 24.999999999999993 | 5917612179 | 24.999999999999993 |
| 5651253930 | 24.999999999999993 | 5803348748 | 24.999999999999993 |
| 5970513239 | 24.999999999999993 | 5646044030 | 24.999999999999993 |
| 5281012338 | 24.999999999999993 | 5488248500 | 24.999999999999993 |
| 5874637063 | 24.999999999999993 | 5808642033 | 24.999999999999993 |
| 5718081953 | 24.999999999999993 | 5451741619 | 24.999999999999993 |
| 5527104759 | 24.999999999999993 | 5546561524 | 24.999999999999993 |
| 5744241972 | 24.999999999999993 | 5115533166 | 24.999999999999993 |
| 5611699915 | 24.999999999999993 | 5080940479 | 24.999999999999993 |
| 5709347307 | 24.999999999999993 | 5707193182 | 24.999999999999993 |
| 5473155687 | 24.999999999999993 | 5032341502 | 24.999999999999993 |
| 5101431346 | 24.999999999999993 | 5295364539 | 24.999999999999993 |
| 5821075679 | 24.999999999999993 | 5734485050 | 24.999999999999993 |
| 5382443951 | 24.999999999999993 | 5095641265 | 24.999999999999993 |
| 5415988175 | 24.999999999999993 | 5313910285 | 24.999999999999993 |
| 5953147688 | 24.999999999999993 | 5585173588 | 24.999999999999993 |
| 5626337241 | 24.999999999999993 | 5140235080 | 24.999999999999993 |
| 5713434364 | 24.999999999999993 | 5130104430 | 24.999999999999993 |
| 5777374933 | 24.999999999999993 | 5445037801 | 24.999999999999993 |
| 5024638800 | 24.999999999999993 | 5797919155 | 24.999999999999993 |
| 5947124810 | 24.999999999999993 | 5918892813 | 24.999999999999993 |
| 5796830148 | 24.999999999999993 | 5183251762 | 24.999999999999993 |
| 5120173134 | 24.999999999999993 | 5737691045 | 24.999999999999993 |
| 5261404261 | 24.999999999999993 | 5083516853 | 24.999999999999993 |
| 5193237839 | 24.999999999999993 | 5640736425 | 24.999999999999993 |
| 5481874464 | 24.999999999999993 | 5276203315 | 24.999999999999993 |
| 5981692957 | 24.999999999999993 | 5347538444 | 24.999999999999993 |
| 5396549043 | 24.999999999999993 | 5882213731 | 24.999999999999993 |
| 5191897631 | 24.999999999999993 | 5300614907 | 24.999999999999993 |
| 5294840098 | 24.999999999999993 | 5197388655 | 24.999999999999993 |
| 5443864223 | 24.999999999999993 | 5037055221 | 24.999999999999993 |
| 5552147023 | 24.999999999999993 | 5505211378 | 24.999999999999993 |
| 5083530802 | 24.999999999999993 | 5865113994 | 24.999999999999993 |
| 5726339050 | 24.999999999999993 | 5416383900 | 24.999999999999993 |
| 5055530142 | 24.999999999999993 | 5166645132 | 24.999999999999993 |
| 5101275860 | 24.999999999999993 | 5542298233 | 24.999999999999993 |
| 5620633691 | 24.999999999999993 | 5547165819 | 24.999999999999993 |
| 5291187801 | 24.999999999999993 | 5965993913 | 24.999999999999993 |
| 5651680658 | 24.999999999999993 | 5018340208 | 24.999999999999993 |
| 5997838742 | 24.999999999999993 | 5316361588 | 24.999999999999993 |
| 5723594170 | 24.999999999999993 | 5983244005 | 24.999999999999993 |
| 5802617203 | 24.999999999999993 | 5124492789 | 24.999999999999993 |
| 5730139862 | 24.999999999999993 | 5408814853 | 24.999999999999993 |
| 5342935520 | 24.999999999999993 | 5573078520 | 24.999999999999993 |
| 5077443304 | 24.999999999999993 | 5069000105 | 24.999999999999993 |
| 5422471790 | 24.999999999999993 | 5180553061 | 24.999999999999993 |
| 5527643886 | 24.999999999999993 | 5181851531 | 24.999999999999993 |
| 5759740437 | 24.999999999999993 | 5165239739 | 24.999999999999993 |
| 5692096934 | 24.999999999999993 | 5064079225 | 24.999999999999993 |
| 5413139871 | 24.999999999999993 | 5646074137 | 24.999999999999993 |
| 5301982656 | 24.999999999999993 | 5787948939 | 24.999999999999993 |
| 5471550962 | 24.999999999999993 | 5501926621 | 24.999999999999993 |
| 5752785218 | 24.999999999999993 | 5123753147 | 24.999999999999993 |
| 5005421381 | 24.999999999999993 | 5090290671 | 24.999999999999993 |
| 5291366970 | 24.999999999999993 | 5001890332 | 24.999999999999993 |
| 5586143754 | 24.999999999999993 | 5637576956 | 24.999999999999993 |
| 5073412012 | 24.999999999999993 | 5259391775 | 24.999999999999993 |
| 5380426438 | 24.999999999999993 | 5333943894 | 24.999999999999993 |
| 5295749860 | 24.999999999999993 | 5498831835 | 24.999999999999993 |
| 5036676738 | 24.999999999999993 | 5379052410 | 24.999999999999993 |
| 5847727331 | 24.999999999999993 | 5472720074 | 24.999999999999993 |
| 5266858642 | 24.999999999999993 | 5068768895 | 24.999999999999993 |
| 5479010936 | 24.999999999999993 | 5982403381 | 24.999999999999993 |
| 5430663750 | 24.999999999999993 | 5600296722 | 24.999999999999993 |
| 5767838029 | 24.999999999999993 | 5725953131 | 24.999999999999993 |
| 5456829275 | 24.999999999999993 | 5709211187 | 24.999999999999993 |
| 5540358800 | 24.999999999999993 | 5732987306 | 24.999999999999993 |
| 5988856708 | 24.999999999999993 | 5748155519 | 24.999999999999993 |
| 5493303656 | 24.999999999999993 | 5340027087 | 24.999999999999993 |
| 5220950334 | 24.999999999999993 | 5596919465 | 24.999999999999993 |
| 5915845886 | 24.999999999999993 | 5708177124 | 24.999999999999993 |
| 5499787935 | 24.999999999999993 | 5919547419 | 24.999999999999993 |
| 5397638031 | 24.999999999999993 | 5778798258 | 24.999999999999993 |
| 5038904405 | 24.999999999999993 | 5398379877 | 24.999999999999993 |
| 5244017343 | 24.999999999999993 | 5219410805 | 24.999999999999993 |
| 5417501550 | 24.999999999999993 | 5948212779 | 24.999999999999993 |
| 5097824359 | 24.999999999999993 | 5450927815 | 24.999999999999993 |
| 5415206042 | 24.999999999999993 | 5005661483 | 24.999999999999993 |
| 5308487817 | 24.999999999999993 | 5275991663 | 24.999999999999993 |
| 5413540758 | 24.999999999999993 | 5801180936 | 24.999999999999993 |
| 5270239324 | 24.999999999999993 | 5691590978 | 24.999999999999993 |
| 5964333589 | 24.999999999999993 | 5059564892 | 24.999999999999993 |
| 5469326511 | 24.999999999999993 | 5981847708 | 24.999999999999993 |
| 5919924805 | 24.999999999999993 | 5298943444 | 14.298279743974385 |
| 5139317791 | 7.952490774907751 | 5440061191 | 2.6250000000000004 |
| 5244267392 | 0.3818181818181819 | 5464937347 | 0.0953523244017538 |
| 5432742787 | 0.0 | 5985963680 | 0.0 |
| 5290954115 | 0.0 | 5116681160 | 0.0 |
| 5709091034 | 0.0 | 5882258293 | 0.0 |
| 5704604271 | 0.0 | 5714266879 | 0.0 |
| 5529439712 | 0.0 | 5257360480 | 0.0 |
| 5332122919 | 0.0 | 5943184518 | 0.0 |
| 5058989565 | 0.0 | 5253345864 | 0.0 |
| 5261573082 | 0.0 | 5130688316 | 0.0 |
| 5806880843 | 0.0 | 5868369968 | 0.0 |
| 5230845010 | 0.0 |  |  |

分析：上图出现的所有用户均是在五大指标分数里最少有一个分数不是为零，虽然有些用户最终分数为零，是因为权重计算的结果。未显示的所有用户则所有指标分数都为零。

## 7.2 意见领袖言论正确性

对比意见领袖发表言论后十天的股票走势，是否跟言论相符合。下表只列出言论正确率大于零的用户及对应的分数。

图7-2 意见领袖言论正确率与分数之间的关系

|  |  |  |  |
| --- | --- | --- | --- |
| userID | 发布与股票相关的言论数 | 言论正确率（百分比形式） | score |
| 5432742787 | 1 | 100.0 | 0.0 |
| 5298943444 | 9 | 88.88888888888889 | 14.298279743974385 |
| 5290954115 | 1 | 100.0 | 0.0 |
| 5709091034 | 1 | 100.0 | 0.0 |
| 5244267392 | 3 | 100.0 | 0.3818181818181819 |
| 5868369968 | 2 | 100.0 | 0.0 |
| 5096958202 | 1 | 100.0 | 24.999999999999993 |
| 5480875446 | 3 | 100.0 | 82.28330722175323 |
| 5632367028 | 3 | 100.0 | 24.999999999999993 |
| 5733332710 | 1 | 100.0 | 24.999999999999993 |
| 5125410266 | 2 | 100.0 | 24.999999999999993 |
| 5974914861 | 1 | 100.0 | 33.15854217220316 |
| 5104102858 | 3 | 100.0 | 28.398299900393432 |
| 5035790932 | 1 | 100.0 | 25.026515151515145 |
| 5052657219 | 4 | 100.0 | 40.816868245887605 |
| 5294835973 | 1 | 100.0 | 24.999999999999993 |
| 5917612179 | 1 | 100.0 | 24.999999999999993 |
| 5262091929 | 2 | 100.0 | 27.4453947368421 |
| 5651253930 | 22 | 90.9090909090909 | 24.999999999999993 |
| 5152081633 | 3 | 66.66666666666666 | 27.121130030959744 |
| 5063348038 | 2 | 100.0 | 51.89241503795131 |
| 5451741619 | 1 | 100.0 | 24.999999999999993 |
| 5546561524 | 1 | 100.0 | 24.999999999999993 |
| 5744241972 | 2 | 100.0 | 24.999999999999993 |
| 5199849053 | 7 | 100.0 | 211.45524593596846 |
| 5185865531 | 9 | 100.0 | 32.51534438863697 |
| 5707193182 | 2 | 100.0 | 24.999999999999993 |
| 5473155687 | 2 | 100.0 | 24.999999999999993 |
| 5032341502 | 1 | 100.0 | 24.999999999999993 |
| 5649506512 | 70 | 91.42857142857143 | 39.16044638527519 |
| 5313910285 | 2 | 100.0 | 24.999999999999993 |
| 5244250711 | 2 | 100.0 | 32.875 |
| 5953147688 | 20 | 95.0 | 24.999999999999993 |
| 5585173588 | 1 | 100.0 | 24.999999999999993 |
| 5059969504 | 1 | 100.0 | 26.43181818181818 |
| 5238646294 | 1 | 100.0 | 26.270063716111654 |
| 5376236220 | 9 | 66.66666666666666 | 56.19754564312153 |
| 5140235080 | 10 | 100.0 | 24.999999999999993 |
| 5713434364 | 1 | 100.0 | 24.999999999999993 |
| 5280009877 | 15 | 93.33333333333333 | 25.003914002391696 |
| 5445037801 | 2 | 100.0 | 24.999999999999993 |
| 5410800119 | 10 | 80.0 | 25.603623188405795 |
| 5918892813 | 1 | 100.0 | 24.999999999999993 |
| 5629459650 | 2 | 100.0 | 27.545612992562855 |
| 5120173134 | 1 | 100.0 | 24.999999999999993 |
| 5193237839 | 1 | 100.0 | 24.999999999999993 |
| 5481874464 | 1 | 100.0 | 24.999999999999993 |
| 5276203315 | 6 | 100.0 | 24.999999999999993 |
| 5749132507 | 6 | 100.0 | 38.14047276613307 |
| 5347538444 | 2 | 100.0 | 24.999999999999993 |
| 5033249022 | 2 | 100.0 | 31.88221472701792 |
| 5819606767 | 5 | 80.0 | 374.71192142696805 |
| 5667609925 | 13 | 92.3076923076923 | 25.013171978248895 |
| 5882213731 | 9 | 88.88888888888889 | 24.999999999999993 |
| 5300614907 | 1 | 100.0 | 24.999999999999993 |
| 5552147023 | 7 | 100.0 | 24.999999999999993 |
| 5807584892 | 4 | 100.0 | 25.010329440284785 |
| 5916818864 | 2 | 50.0 | 27.12673102669232 |
| 5083530802 | 2 | 100.0 | 24.999999999999993 |
| 5416383900 | 2 | 100.0 | 24.999999999999993 |
| 5166645132 | 1 | 100.0 | 24.999999999999993 |
| 5158708788 | 23 | 86.95652173913044 | 80.38241497058738 |
| 5547165819 | 5 | 100.0 | 24.999999999999993 |
| 5651680658 | 2 | 100.0 | 24.999999999999993 |
| 5288693117 | 7 | 100.0 | 25.07954545454545 |
| 5467872602 | 8 | 100.0 | 34.0 |
| 5857787366 | 1 | 100.0 | 25.004010593518284 |
| 5316361588 | 11 | 81.81818181818183 | 24.999999999999993 |
| 5951480610 | 3 | 100.0 | 25.394923388671724 |
| 5408814853 | 3 | 100.0 | 24.999999999999993 |
| 5336369061 | 1 | 100.0 | 25.11590477177116 |
| 5077443304 | 6 | 83.33333333333334 | 25.11590477177116 |
| 5180553061 | 2 | 100.0 | 24.999999999999993 |
| 5181851531 | 1 | 100.0 | 24.999999999999993 |
| 5165239739 | 1 | 100.0 | 24.999999999999993 |
| 5413139871 | 2 | 50.0 | 24.999999999999993 |
| 5032359409 | 1 | 100.0 | 25.005600995732568 |
| 5646074137 | 1 | 100.0 | 24.999999999999993 |
| 5710011705 | 2 | 100.0 | 25.005600995732568 |
| 5514474822 | 11 | 54.54545454545454 | 30.485605824429207 |
| 5267456690 | 9 | 77.77777777777779 | 46.257759898905576 |
| 5752785218 | 1 | 100.0 | 24.999999999999993 |
| 5005421381 | 1 | 100.0 | 24.999999999999993 |
| 5090290671 | 1 | 100.0 | 24.999999999999993 |
| 5502895643 | 4 | 100.0 | 28.719329499676032 |
| 5291366970 | 1 | 100.0 | 24.999999999999993 |
| 5586143754 | 4 | 75.0 | 24.999999999999993 |
| 5190345050 | 205 | 88.78048780487805 | 147.38132178098977 |
| 5295749860 | 1 | 100.0 | 24.999999999999993 |
| 5833114470 | 7 | 85.71428571428571 | 25.00055456828598 |
| 5805831020 | 1 | 100.0 | 35.54509052522607 |
| 5847727331 | 1 | 100.0 | 24.999999999999993 |
| 5511195936 | 5 | 100.0 | 30.013707037643204 |
| 5768850874 | 4 | 75.0 | 28.999999999999993 |
| 5982403381 | 1 | 100.0 | 24.999999999999993 |
| 5441695769 | 2 | 100.0 | 25.006590669139733 |
| 5725953131 | 2 | 100.0 | 24.999999999999993 |
| 5540358800 | 1 | 100.0 | 24.999999999999993 |
| 5296621426 | 3 | 100.0 | 28.516273185884838 |
| 5340027087 | 1 | 100.0 | 24.999999999999993 |
| 5499787935 | 1 | 100.0 | 24.999999999999993 |
| 5624808117 | 2 | 100.0 | 25.003914002391696 |
| 5765429529 | 2 | 50.0 | 29.92364462106159 |
| 5673164838 | 5 | 80.0 | 25.001282631211364 |
| 5513217170 | 1 | 20.0 | 25.013171978248895 |
| 5038904405 | 1 | 100.0 | 24.999999999999993 |
| 5398379877 | 1 | 100.0 | 24.999999999999993 |
| 5244017343 | 2 | 100.0 | 24.999999999999993 |
| 5453117137 | 1 | 100.0 | 25.013094799524094 |
| 5415206042 | 1 | 100.0 | 24.999999999999993 |
| 5005661483 | 1 | 100.0 | 24.999999999999993 |
| 5308487817 | 1 | 100.0 | 24.999999999999993 |
| 5275991663 | 1 | 100.0 | 24.999999999999993 |
| 5413540758 | 1 | 100.0 | 24.999999999999993 |
| 5514804700 | 4 | 100.0 | 25.001807851239665 |
| 5545011370 | 25 | 84.0 | 34.587292223366894 |
| 5504830612 | 35 | 88.57142857142857 | 25.012740825730873 |
| 5059564892 | 1 | 100.0 | 24.999999999999993 |

分析：由表可知，除去正确率等于零的意见领袖外，其他意见领袖的言论正确率都大于百分之五十，且大多数在百分之八十以上。相对应的，他们的分数绝大多数在24.999999999999993以上。另外，分数明显高的意见领袖，发表言论的次数相对更多，而且正确率能保持在百分之五十以上，进而可以推测分数越高的意见领袖，对股票市场的走势把握得越精准。

# 第八章 结论与展望

## 8.1 全文总结

在对本课题的研究与开发过程中，具体做了以下几个方面的工作：

（1）首先研究了金融社交的概念，和雪球网这一网站在金融方面提供的服务。

（2）研究了意见领袖的基本概念，意见领袖的重要性，特征以及它的功能。回顾了过去对意见领袖的研究，发现大多处于量化级别，具体细化研究比较少，更不用提金融社交领域的意见领袖发现算法了。

（3）对比现有社交网络，抽象出雪球网的网络模型为无向无权重网络。并通过研究最为相似的社交网络“微博”的意见领袖发现算法，得到了雪球网意见领袖发现算法的启示：关注、转发网络使用PageRank排名算法，回复、收藏、打赏指标使用简单加权算法。同时，着重研究了网页排名算法PageRank的思想和使用方法。

（4）研究了用于数据融合中确定权重的层次分析法，使所有用户能在统一标准下按影响力分数大小进行排序。

（5）研究语义分析和文本处理方法，分析意见领袖的言论中是否含有与股票涨跌相关的信息。再调用Tushare财经数据接口包，获得用户发表评论后10天内股票的收益情况，看是否与用户言论匹配，从而判断此算法是否合理和有意义。

## 8.2 研究展望

本课题对“意见领袖的发现”进行了深入地研究，设计的关于金融社交领域的意见领袖发现算法是具有现实意义的。通过这一算法计算出意见领袖，关注他们对于股票市场的分析，股票投资会有更加有目的性和方向性。然而，这一研究还有许多值得继续研究改进的地方，未来的研究可以在以下方面深入。

1. 对于每个帖子的回复、收藏、打赏指标，可以分别构成各自的网络，分别使用PageRank排名算法进行处理，再进行数据融合。
2. 对于回复指标来说，因为每个人可以多次回复，所以形成的网络应该是无向有权重的。
3. 每个用户都可以建立属于自己的群组，所以可以考虑建立群组越多，组内人员越多，该用户的影响力越大。

# 参考文献

[1] 肖宇,许炜,夏霖.网络社区中的意见领袖特征分析.计算机工程与科学.33(1).2010,1:150-157.

[2] 金迪,马衍民.PageRank算法的分析及实现.经路技术协作信息.18(1001).2009,8:118.

[3] 刘志明,刘鲁.微波网络舆情中的意见领袖.系统工程.29(6).2011,6:9-16.

[4] 邓雪,李家铭,曾浩健等.层次分析法权重计算方法分析及其应用研究.数学的实践与认识.42(7). 2012,4:93-100

[5] 赵悦阳.PageRank算法与HITS算法比较研究.医学信息学杂志.32(2).2011,3:57-60.

[6] 马文文,魏文晗,邓一贵.基于隐含语义分析的微博话题发现方法.计算机工程与应用.50(1). 2014,2:97-100.

[7] Belle Tseng.Identifying opinion leaders in the blogosphere.ResearchGate.108(115). 2007,1:971-974.

[8] J Wang,X Jia,L Zhang.Identifying and Evaluating the Internet Opinion Leader Community Through k-clique Clustering.Jouranl of Computer.8(9).2013,5:120-124.

[9] KE Dalrymple.Following the Leader:Using Opinion Leaders in Environmental

Strategic Communication.Society & Natural Resources.26(12).2013,7:1438-1453.

[10] [C Kaiser](http://xueshu.baidu.com/s?wd=author:(Carolin%20Kaiser)%20Institute%20of%20Information%20Systems,%20University%20of%20Erlangen-Nuremberg,%20Lange%20Gasse%2020,%2090403,%20Nuremberg,%20Germany&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight=person" \t "http://xueshu.baidu.com/_blank),[J Kröckel](http://xueshu.baidu.com/s?wd=author:(Johannes%20Kr%C3%B6ckel)%20Institute%20of%20Information%20Systems,%20University%20of%20Erlangen-Nuremberg,%20Lange%20Gasse%2020,%2090403,%20Nuremberg,%20Germany&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight=person" \t "http://xueshu.baidu.com/_blank),[F Bodendorf](http://xueshu.baidu.com/s?wd=author:(Freimut%20Bodendorf)%20Institute%20of%20Information%20Systems,%20University%20of%20Erlangen-Nuremberg,%20Lange%20Gasse%2020,%2090403,%20Nuremberg,%20Germany&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight=person" \t "http://xueshu.baidu.com/_blank).[Simulating the spread of opinions in online social networks when targeting opinion leaders](http://xueshu.baidu.com/s?wd=paperuri:(095aa99b8f2a5b0604f21684b967a5fd)&filter=sc_long_sign&tn=SE_xueshusource_2kduw22v&sc_vurl=http://link.springer.com/10.1007/s10257-012-0210-z&ie=utf-8&sc_us=7989655049687380047" \t "http://xueshu.baidu.com/_blank).Information Systems and e-Business Management.11(4).2013,11:597-621.

[11] 付永利.网络意见领袖影响力研究.[学位论文].河南大学：2007.

[12] 王婧.微博意见领袖话语权利研究 -- 以新浪微博为例.[学位论文].渤海大学研究生学院: 2011.

# 致 谢

本论文的工作是在我的导师张熙教授的悉心指导下完成的，张熙教授严谨的治学态度和科学的工作方法给了我极大的帮助和影响，在此衷心感谢老师对我的关心和指导。

在撰写论文期间，石嘉伟学长和王迪学长对我论文中的研究工作给予了热情的帮助，在此向他们表达我的感激之情。

另外也感谢父母，他们的理解和支持是我能够在学校专心完成我的学业。