基于Spark的大规模社交网络社区发现原型系统

叶小榕[1],邵晴[2]

1. 中国科学技术信息研究所,北京 100038
2. 北龙中网(北京)科技有限责任公司,北京 100190

摘要　为有效发掘大规模社交网络上的用户信息,提高对用户之间关系的深入了解,设计开发了基于Spark的大规模社交网络社区发现原型系统。系统利用ActiveMQ实现对大量用户数据的抓取,使用基于Spark的MLlib提供的朴素贝叶斯算法对用户数据进行清洗,利用Spark的GraphX提供的PageRank算法和MLlib提供的Z-Score算法计算用户排名,最终应用并优化LPA算法,将特征相近、联系较密切的用户快速地划分到同一社区中,为进一步分析利用社区用户数据打下了基础。

关键词　Spark;GraphX;MLlib;社区发现

随着移动互联网的飞速发展,微博、推特等各种社交网站越来越成为人们获取信息、交流思想的重要渠道。为了能快速高效地挖掘、整理、分析和利用这些庞大的用户社交数据,产生了很多用户推荐算法[1],其中,社区发现算法以其算法实现丰富、用途广泛而得到了普遍采用。社区发现算法是将社交网站用户之间的关系描述成关系网络[2],每个节点代表一个用户,节点之间连接的边代表用户之间的关系,通过分析用户之间的交流信息和用户数据,将有相似兴趣、观点的用户挖掘出来,组成多个关系密切、关注交流频繁的社区[3,4],针对不同社区可进行针对性的商品推荐和广告投放、提供推荐搜索和潜在好友等个性化服务。在诸多社区发现算法中,标签传播算法(label propagation algorithm,LPA)具有近似线性的时间复杂度、较快的收敛速度、无需设置任何先验参数、宜于分布式并行计算等优点[5,6],但是这种传统的社区发现算法在迭代时随机选择用户,会导致结果不稳定,影响准确率[7],而且此算法多是基于单机服务器进行计算,在硬件性能、容错机制等方面,无法满足快速增长的计算需求。

针对上述问题,本研究设计开发了基于Spark的大规模社交网络社区发现原型系统。系统采用Spark集群的计算框架,使用多节点服务器增强了硬件性能,通过数据检查点和记录数据更新等方式提供了容错恢复机制,适应了大规模并行计算的社交网络;系统实现了大规模社交网络用户公开数据的抓取,并通过算法清洗僵尸用户,保留真实的用户;系统利用Spark提供的大数据计算平台,在LPA算法进行迭代时,加入了用户的排名参数对算法进行了优化,优先传播高排名用户的标签,避免随机选取,快速形成以高排名用户为中心的稳定的社区结构,初步解决了LPA算法结果不稳定的问题,并将优化后的LPA算法应用到了社区发现原型系统中。通过实验验证,对比应用传统单机版LPA算法的系统,本系统应用的优化后的LPA算法的准确率和稳定性有了提升,系统满足了快速增长的计算需求。

收稿日期:2018-10-09;修回日期:2018-11-20

作者简介:叶小榕,高级工程师,研究方向为计算机软件、数字图书馆,电子信箱:yeelfine@sina.com

引用格式:叶小榕, 邵晴. 基于Spark的大规模社交网络社区发现原型系统[J]. 科技导报, 2018, 36(23): 93-101; doi: 10.3981/j.issn.1000-7857.2018.23.012

1 系统架构

系统大量采用了基于Spark框架组件,Spark框架是由众多子项目构建成的大数据计算平台[8,9]。Spark分为3个层次,其中底层采用Yarn、Mesos等集群资源管理器;中层为Spark核心引擎;上层提供5大功能模块:Spark SQL(结构化数据处理)、MLlib(机器学习)、GraphX(图计算,提供了Pregel接口)、Spark Streaming(实时处理)、SparkR(统计分析)。

系统充分利用了Spark框架提供的大数据计算能力,包括数据抓取模块、数据清洗模块、用户排名模块和社区发现模块4个模块(图1)。(1)数据抓取模块:通过社交网站提供的API、网页爬虫等多种方式,并利用ActiveMQ建立分布式的数据抓取模块,获取公开的用户数据;(2)数据清洗模块:利用Spark中的MLlib提供的朴素贝叶斯算法,对数据抓取模块抓取的网站用户数据进行清洗,剔除僵尸用户,保留真实用户;(3)用户排名模块:利用Spark中GraphX提供的PageRank算法和MLlib提供的Z-Score算法,根据数据抓取模块抓取的用户关联关系和数据清洗模块清洗后的用户数据,计算用户排名;(4)社区发现模块:根据用户数据和用户排名,应用并优化Pregel接口提供的LPA算法,将用户归属到不同的社区,实现社区发现。系统整体架构如图1所示。

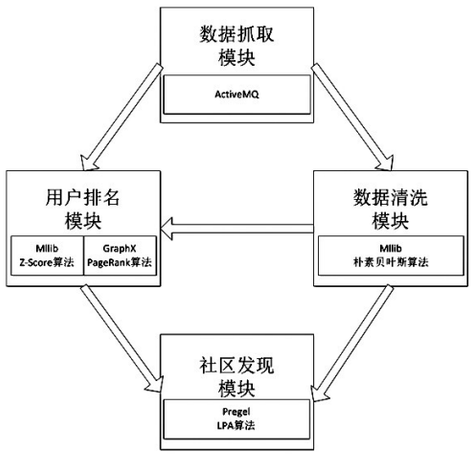

图1 系统整体架构

Fig. 1 System overall architecture

2 数据抓取模块

数据抓取模块通过网站提供的API、网页爬虫等多种方式,并利用ActiveMQ软件高效、可扩展、稳定和安全的消息通信机制,搭建了以ActiveMQ集群服务器和工作节点为核心的分布式数据抓取框架[10],最终获取用户关注的用户ID(后文简称关注ID)、粉丝用户ID(后文简称粉丝ID)、发帖数、收藏数等公开的用户数据。

2.1 分布式数据抓取框架

为提高数据抓取的性能,本模块利用ActiveMQ软件高效、可扩展、稳定和安全的消息通信机制,搭建了以ActiveMQ集群服务器和工作节点为核心的分布式数据抓取框架,分布式数据抓取框架使多个工作节点同时抓取数据,并能使不同工作节点间的抓取程序彼此协调,避免重复抓取和遗漏。

分布式数据抓取框架包括ActiveMQ集群服务器和若干工作节点,其中ActiveMQ集群服务器作为ActiveMQ软件的服务器端,负责维护待抓取的用户ID队列;工作节点作为ActiveMQ的客户端,不断从ActiveMQ集群服务器的队列中接收将要抓取的用户ID,利用下文提供的两种抓取方法获取关注ID、粉丝ID、发帖数、收藏数等用户数据,之后将获取到的关注ID和粉丝ID也放入队列中,作为下一次待抓取的用户ID,这样不断循环往复,实现对用户数据的深度抓取。ActiveMQ数据抓取流程如图2所示。

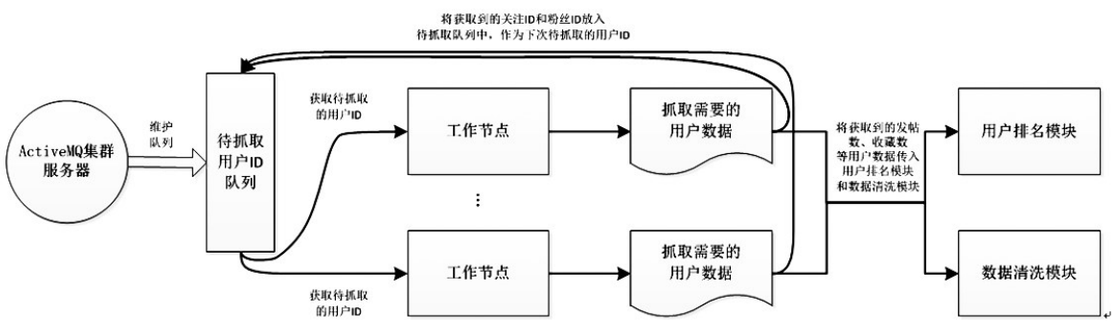

图2 ActiveMQ数据抓取流程

Fig. 2 Flowchart of ActiveMQ data capture

2.2　抓取用户数据方法

数据抓取模块抓取网页采用两种方法:针对提供API接口的社交网站,通过接口获得数据;没有提供接口的网站,通过访问网页源代码,解析出数据。

2.2.1　API接口方式

某些网站,例如推特,提供了便捷的API接口。本模块采用推特官方推荐的开源类库Twitter4J,获取相关数据,根据当前用户ID获取粉丝ID的伪代码:

```
Twitter twitter = (new TwitterFactory(cb.build())).getInstance(); //初始化Twitter接口类
IDS ids = twitter.getFollowersIDs(userID);//调用接口,根据用户ID得到用户的粉丝ID集合
for (long id : ids.getIDs().count) {//循环粉丝ID集合
System.out.println(id+ twitter.showUser(id)); } //得到每个粉丝ID的数据信息
```

2.2.2　直接访问网页方式

针对没有提供API接口的网站,本模块通过Jsoup类库获取用户数据。Jsoup是开源的HTML解析器。本模块应用Jsoup根据用户ID得到网页结构信息,再通过分析网页内id、class、tag等属性,解析出网页中的用户数据,包括粉丝ID、粉丝名称等。

3　数据清洗模块

数据清洗模块采用MLlib中的朴素贝叶斯分类算法,对数据抓取模块抓取的社交网站用户数据进行清洗和过滤,删除僵尸用户,保留真实用户,提高了用户数据的可靠性。

3.1　朴素贝叶斯分类算法

朴素贝叶斯分类算法是基于贝叶斯定理和特征条件独立假设的有监督的分类方法[11]。朴素贝叶斯算法实现简洁,学习与预测的效率较高、速度较快,适合处理大数据,常用于用户类型判断、欺诈检测、垃圾邮件判定、企业信用等级分类、文本分类等[12,13]。朴素贝叶斯算法数学表达式定义[14]如下。

1)设$V=(v_1, v_2, \cdots, v_n)$为一个待分类项,$v_n$为$V$的每个特征属性。

2)设$B=(b_1, b_2, \cdots, b_n)$为一个分类集合,$b_n$为每个具体的分类。

3)如果要测试某个V_n属于B集合中的哪个分类b_c,则计算各个特征属性的条件概率估计$P(b_1|V)$,$P(b_2|V)$,$\cdots$,$P(b_n|V)$,然后求出在V发生的条件下,归属于$b_1, b_2, \cdots, b_n$中哪个可能性最大,即

$$P(b_c|V)=\max[P(b_1|V),P(b_2|V),\cdots,P(b_n|V)] \quad (1)$$

4)结合贝叶斯定理,则可以得到朴素贝叶斯分类器的表达式[15]

$$h_{nb}(V)=\arg\max_{b\in V}P(b)\prod_{i=1}^{n}P(V_i|b) \quad (2)$$

3.2　朴素贝叶斯分类算法流程

本模块从数据抓取模块中获得了大量的用户数据,但这些数据中含有僵尸用户的相关信息。本模块利用了朴素贝叶斯算法将这些僵尸用户清洗掉,只保留真实用户,流程分为4个阶段。

1)设定待分类项和分类集合阶段:模块选择了如下6个特征属性作为待分类项(设置真实用户值为1,僵尸用户值为0):

$V=(v_1, v_2, v_3, v_4, v_5, v_6)$

v_1=粉丝ID个数　(粉丝ID个数大于等于10个为1,小于10个为0)

v_2=发帖数　(大于等于10条为1,小于10条为0)

v_3=收藏数　(大于等于10条为1,小于10条为0)

v_4=转帖数　(大于等于10条为1,小于10条为0)

v_5=注册天数　(大于等于10天为1,小于10天为0)

v_6=头像　(有头像为1,无头像为0)

模块设定真实用户和僵尸用户两个分类集合

$$B=(b_1, b_2)$$

2)人工分类阶段:随机选取部分用户数据作为样本数据,按照前一步的特征属性进行人工处理分类,划分出是真实用户还是僵尸用户,从而得到训练样本。

3)训练生成分类器阶段:根据朴素贝叶斯计算公式,代入训练样本,计算出真实用户和僵尸用户这两个类别在训练样本中的出现频率,及每个特征属性对这两个类别的条件概率估计,从而得到可能性最大的条件概率估计,生成分类器。

4)应用阶段:根据朴素贝叶斯公式,使用分类器计算其他的用户数据,判定是否是真实用户。

3.3　利用MLlib实现朴素贝叶斯分类算法

MLlib是基于Spark的可扩展的机器学习库,提供了分类、回归、聚类、协同过滤、降维等学习算法和工具类,能够实现大数据的分布式机器学习[16]。本模块根据上文中的4个阶段,利用MLlib提供的接口,实现朴素贝叶斯算法,伪代码如下:

val data = sc.textFile("data.txt") //从数据抓取模块中读取数据

val parsedData =data.map { line => val parts =line.split(',') //解析数据格式

LabeledPoint(parts(0).toDouble,Vectors.dense(parts(1).split(' ').map(_.toDouble)))}

val trainData = parsedData.selectTraingData().training()//随机选取部分数据作为样本数据进行人工分类

val model =NaiveBayes.train(trainData,lambda = 1.0) //通过训练生成分类器

parsedData.selectTestData().map(p => (model.predict(p.features))).collect().foreach(println) //使用分类器计算其他数据,判定真实用户

经过上述计算后,系统将保留真实用户数据,供用户排名模块使用。

4 用户排名模块

用户排名模块将用户联系的紧密程度和用户的活跃程度,分别用PageRank和Z-Score算法进行量化计算,加权求和,最终得到用户排名。

4.1 PageRank算法

PageRank算法是基于链接分析的排序算法,通过网页之间的链接来计算网页的排名,此算法在论文和期刊排名、网站用户排名中被广泛采用[17,18]。本模块利用PageRank算法,将每个用户排名值定义为PR值,利用用户之间的关注来计算PR值,PR值越高则排名越高[19]。PageRank算法流程如图3所示。

4.2 Z-Score算法

Z-Score算法是数据规范化处理的一种方法,通过计算某个数值与总数的平均数有多少个标准差,将其转化为无量纲、无数量级差异的纯数值,便于不同单位或量级的数据进行比较,Z-Score算法主要用于分数统计、统计决策等领域[20]。本系统计算用户排名时,除了使用用户联系的紧密程度外,还使用发帖数、收藏数、转发数等用户的活跃程度来进行计算[21,22]。为使上述3类数据之间具有可比性,不因为绝对数量的差异而忽略某个数据的影响,本模块利用Z-Score算法,将每个用户的发帖数、收藏数、转发数转化为无单位的Z-Score值,即计算每个用户的发帖数、收藏数、转发数分别与所有用户发帖总数、收藏总数、转发总数的平均数值有

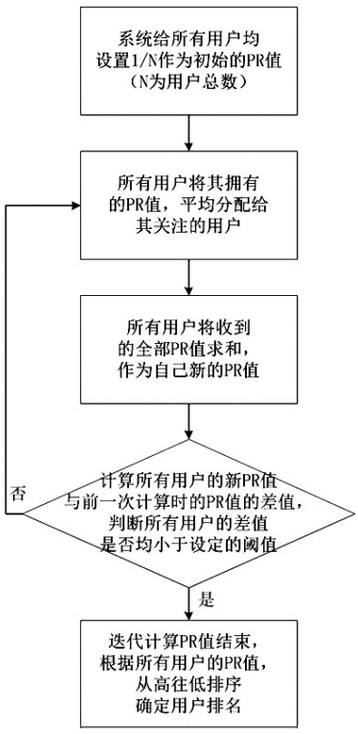

图3 PageRank算法流程

Fig. 3 Flowchart of PageRank algorithm

多少个标准差,从而得到每个用户上述3个数据的标准化数值,用来进行排名比较。Z-Score计算公式为

$$z=\frac{x-u}{\sigma} \tag{3}$$

其中,u为总体平均值;$x-u$为离均差;σ为总体标准偏差。

4.3 PageRank和Z-Score算法模型

4.3.1 利用GraphX实现PageRank算法模型

GraphX是基于Spark的分布式图计算框架,融合了图并行计算以及数据并行计算的优势,提供了一整套可扩展的图算法工具包[23,24],支持PageRank、三角形计数、最大连通图等6种经典的图算法[25,26]。

本模块调用了GraphX提供的PageRank算法的API接口[27,28],伪代码如下:

val graphx = GraphLoader.edgeListFile(sc, "relations.txt") //导入用户之间关注关系的文件relations.txt

val setPrs=graphx.pageRank(threshold)//调用GraphX的PageRank接口,参数threshold为迭代计算PR值的差值时设定的阈值

经过以上计算,得到每个用户ID和其PR值对应关系的集合,集合的样式为Set <用户ID,PR值>。

4.3.2 利用MLlib实现Z-Score算法模型

本模块调用了MLlib提供的Z-Score算法接口,伪代码如下:

val features=MLUtils.loadLibSVMFile(sc, "numbers.txt") //导入记录用户发帖数、收藏数、转发数的文件numbers.txt

val scaler=new StandardScaler().fit(features) //初始化MLlib的Z-Score接口

val setZscore=scaler.transform(features).collect() //调用Z-Score进行计算

经过以上计算,得到每个用户ID和其发帖数、收藏数、转发数的Z-Score值对应关系的集合,集合的样式为Set <用户ID, <PostsZScore, FavoriteZScore, ForwardZScore>>。

4.3.3 利用PageRank和Z-Score计算用户排名

用户排名影响因素很多,本系统的用户排名模块既考虑了用户联系的紧密程度,将用户之间的关注作为主要参考因素,也考虑了用户的活跃程度,引入发帖数、收藏数和转帖数作为次要参考因素。本模块将PageRank和Z-Score的计算结果按用户ID加权求和,得到每个用户ID对应的排名值:

Rank(userID)=PRR(userID)+d*(PostsZScore(userID)+FavoriteZScore(userID)+ForwardZScore(userID)) (4)

式中,PR为用户ID的PR值,d为权重参数,PostsZScore、FavoriteZScore和ForwardZScore分别为发帖数、收藏数、转发数的Z-Score值,Rank为用户的排名值。

本模块将用户按排名值进行排序,引入社区发现模块计算。

5 社区发现模块

社区发现模块根据数据清洗模块提供的用户数据,应用了基于Spark的LPA算法,并在算法中加入了从用户排名模块中得到的用户排名,对算法进行了优化,从而将用户更加稳定的划分到不同的社区中,实现了社区发现原型系统。

5.1 优化标签传播算法

标签传播算法(label propagation algorithm,LPA)是经典社区发现算法,适用于大规模并行计算的社交网络[29]。其基本思想是,用户所属的社区由它关联用户所属于的社区决定,如果关联的用户多数属于某个社区,那么该用户自身也属于这个社区[30-32]。但是LPA算法在进行标签选择迭代时随机选择用户,在多次计算时,社区个数会在一定范围内上下波动,导致计算结果不够稳定[33]。

本模块一方面采用Spark集群提高了算法性能,另一方面在迭代时引入用户排名,减少了随机性,使算法收敛更快、计算时间更少、结果也更加稳定,从而优化了LPA算法。优化后的LPA算法,在初始时为每个用户赋予标签(一般选取用户ID直接作为初始的标签),用于表示该用户所属的社区;随后循环所有用户,每个用户在所有关联用户中寻找出现次数最多的那个标签,作为自己新的标签,如果标签的出现次数相同,则选取用户排名高的标签,改变了原算法随机选择用户造成结果不稳定的情况,这一迭代过程,直至所有用户的标签不再变化,或迭代次数达到设定的阈值(通常阈值在10~20次之间)时为止;经过以上过程,所有具有相同标签的用户就划分到同一社区里。优化后的LPA算法流程如图4所示。

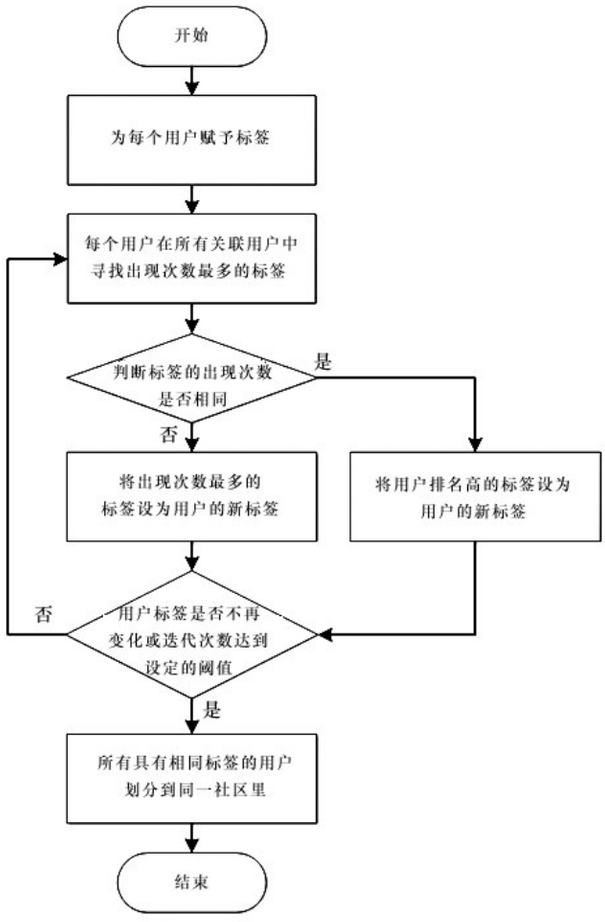

图4 LPA算法流程

Fig. 4 Flowchart of LPA algorithm

5.2 实现优化的LPA算法

GraphX的Pregel对图遍历、最短路径、各种社区发现算法等提供了非常灵活的API[34]。本模块应用并修改了Pregel接口,来实现优化的LPA算法。

本模块调用了Pregel接口的sendMessage、mergeMessage、vertexProgram等3个函数:sendMessage函数,将用户自己的标签信息发送给关联用户;mergeMessage函数,处理关联用户发送过来的标签消息,将相同的标签进行合并;vertexProgram函数,根据用户排名返回个数最多的标签。伪代码如下:

```
val lpaGraph = GraphLoader.edgeListFile(sc, "relations.txt").mapVertices{case(labelId,_)=>labelId} //导入用户之间关注关系的文件relations.txt,并为每个用户赋予标签
def sendMessage(): Iterator={ //向关联用户发送标签信息
    Iterator((srcId, dstAttr), (dstId, srcAttr )) }
def mergeMessage() : Map={ //处理收到的标签消息,并将相同的标签进行合并
    (count1.keySet ++ count2.keySet).map }
def vertexProgram(): vertexId = {
    val msgMaxByLabel = message.getMax() //在所有关联用户中取出出现次数最多的标签
    if (msgMaxByLable.count>1){ //判断标签的出现次数是否相同,如果相同,将用户排名高的标签设为用户的新标签
        vertexId = msgMaxByLable.selectByUserIdOrder()
    } else { vertexId = msgMaxByLable.get(0)}} //如果不同,将出现次数最多的标签设为用户的新标签
//加载用户数据,设定迭代次数阈值,并调用前文的3个函数
val myLPA=Pregel(lpaGraph, maxIterations = maxSteps)(
        vprog = vertexProgram
        sendMsg = sendMessage
        mergeMsg = mergeMessage)
val label=myLPA.run(lpaGraph, maxSteps)//执行运算得到结果,所有具有相同标签的用户划分到同一社区里,最终得到划分好的社区。
```

6 结果与分析

基于Spark的大规模社交网络社区发现算法的原型系统,已经开始在测试平台运行,该系统由10台服务器组成,其中3台部署了数据抓取模块,7台安装了Spark集群,搭建了数据清洗模块、用户排名模块、社区发现模块。服务器的CPU分别为16核到32核的至强处理器,内存为32~64 GB,操作系统为CentOS7.1;软件包括activeMQ5.15.0、Hadoop2.6.4、Spark2.2.0。本系统的具体环境配置见表1。

系统测试运行期间,数据抓取模块通过公开API得到了约11万的社交网络用户的信息,包括用户ID、关注ID、粉丝ID、发帖数、收藏数等。

数据清洗模块从中随机选取了1万条数据作为朴素贝叶斯训练数据,由训练数据训练出分类器,其余10万条作为测试数据,经过MLlib运算,分类器准确率为82.71%;通过朴素贝叶斯分类器,在10万测试数据中判定约92%的用户为真实用户。用户排名模块应用PageRank和Z-Score算法,计算出用户的排名。利用此排名,社区发现模块优化了LPA算法,将约9万用户归属到约150个社区,实现了社区发现。图5为社区发现计算结果的截图(隐去了ID中间几位以保护隐私)。

为验证优化后LPA算法的性能,本系统与传统单机版LPA系统、未优化的集群LPA系统,分别在准确率和结果稳定性上进行了多次测试对比,其中平均准确率是指多次测试时LPA准确率的平均值,结果稳定性指多次测试时社区个数波动的区间,结果见表2。

表1 系统的环境配置

Table 1 System environment configuration

服务器ID	测试服务器IP地址	硬件参数	操作系统	部署软件	节点类型
1	192.168.1.100	CPU:16核,内存:32GB	CentOS7.1	ActiveMQ5.15.0	ActiveMQ集群服务器
2-3	192.168.1.101-102	CPU:16核,内存:32GB	CentOS7.1	ActiveMQ5.15.0	ActiveMQ的工作节点
4	192.168.1.200	CPU:32核,内存:64GB	CentOS7.1	Hadoop2.6.4, Spark2.2.0	Spark的master
5-7	192.168.1.201-203	CPU:32核,内存:64GB	CentOS7.1	Hadoop2.6.4, Spark2.2.0	Spark的slave
8-10	192.168.1.204-206	CPU:16核,内存:32GB	CentOS7.1	Hadoop2.6.4, Spark2.2.0	Spark的slave

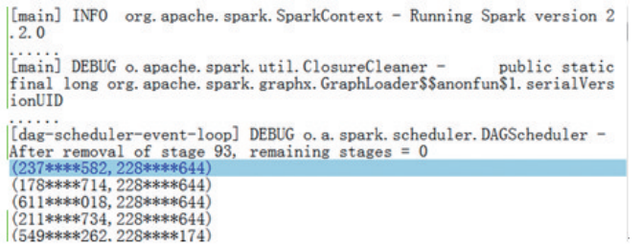

图5　社区发现计算结果截图

Fig. 5　Screenshot of community detection results

表2　对比结果

Table 2　Comparison results

性能	传统单机版 LPA	优化前集群 LPA	优化后集群 LPA
平均准确率	80.4%	80.5%	83.1%
结果稳定性	131~173	128~170	141~158

经过实验对比,可以发现在平均准确率方面,传统单机版LPA算法和优化前集群LPA算法两个版本结果近似,而优化后的版本准确率提升了约3%,因为引入了用户排名,高排名用户的标签将优于低排名用户的标签进行传播,这样形成了更多的以高排名用户为中心的社区,更加贴合实际,准确率更高。在结果稳定性方面,单机版LPA和优化前集群LPA的社区个数波动范围较大,结果不稳定,而优化后集群LPA的社区个数波动范围明显缩小,说明优化后的LPA算法结果更加稳定,原因也是由于用户排名的引入,避免了随机选取用户,从而提高了稳定性。

7　结论

本原型系统利用Spark集群较好地实现了大规模设计网络下社区发现算法,并通过用户排名对LPA算法做了优化。下一步,本原型系统将继续完善,如增加算法结果的可视化展现、提高发现僵尸用户的准确性等。

参考文献(References)

[1] 杨文杰, 周志刚, 雷欢, 等. 基于GraphX的社交网络用户推荐算法研究[J]. 自动化与信息工程, 2018, 39(1): 27-31.
Yang Wenjie, Zhou Zhigang, Lei Huan, et al. Research on recommendation algorithm for social network users based on GraphX[J]. Automation & Information Engineering, 2018, 39(1): 27-31.

[2] 林友芳, 王天宇, 唐锐, 等. 一种有效的社会网络社区发现模型和算法[J]. 计算机研究与发展, 2012, 49(2): 337-345.
Lin Youfang, Wang Tianyu, Tang Rui, et, al. An effective model and algorithm for community detection in social networks[J]. Journal of Computer Research and Development, 2012, 49(2): 337-345.

[3] 李镇. 基于Spark的大规模社交网络社区发现算法设计与实现[D]. 扬州: 扬州大学, 2015.
Li Zhen. Design and implementation of community detection algorithm based on spark[D]. Yangzhou: Yangzhou University, 2015.

[4] 梁晋, 梁吉业, 赵兴旺. 一种面向大规模社会网络的社区发现算法[J]. 南京大学学报, 2016, 52(1): 159-166.
Liang Jin, Liang Jiye, Zhao Xingwang. A community detection algorithm for large social network[J]. Journal of Nanjing University, 2016, 52(1): 159-166.

[5] Zhu Xiaojin, Ghahramani Zoubin. Learning from labeled and unlabeled data with label propagation[R]. Pittsburgh: Carnegie Mellon University, 2002.

[6] 张俊丽, 常艳丽, 师文. 标签传播算法理论及其应用研究综述[J]. 计算机应用研究, 2013, 30(1): 21-25.
Zhang Junli, Chang yanli, Shi Wen. Overview on label propagation algorithm and applications[J]. Application Research of Computers, 2013, 30(1): 21-25.

[7] 张素琪, 高星, 霍士杰, 等. 基于速度优化和社区偏向的标签传播算法[J]. 数据分析与知识发现, 2018, 2(3): 60-69.
Zhang Suqi, Gao Xing, Huo Shijie, et al. A label propagation algorithm based on speed optimization and community preference [J]. Data Analysis and Knowledge Discovery, 2018, 2(3): 60-69.

[8] 胡俊, 胡贤德, 程家兴. 基于Spark的大数据混合计算模型[J]. 计算机系统应用, 2015, 24(4): 214-218.
Hu jun, Hu Xiande, Cheng Jiaxing. Big data hybrid computing mode based on Spark[J]. Computer Systems & Applications, 2015, 24(4): 214-218.

[9] 杨天晴, 王津, 杨旭涛, 等. 一种Spark环境下的高效率大规模图数据处理机制[J]. 计算机应用研究, 2016, 33(12): 3730-3747.
Yang Tianqing, Wang Jing, Yang Xutao, et al. High efficiency large-scale graph data processing mechanism in environment of Spark[J]. Application Research of Computers, 2016, 33(12): 3730-3747.

[10] 戴俊, 朱晓民. 基于ActiveMQ的异步消息总线的设计与实现[J]. 计算机系统应用, 2010, 19(8): 254-257.
Dai Jun, Zhu Xiaomin. Design and implementation of an asynchronous message bus based on ActiveMQ[J]. Computer Systems & Applications, 2010, 19(8): 254-257.

[11] Domingos P M, Pazzani M J. On the optimality of the simple

bayesian classifier under zero-one loss[J]. Machine Learning, 1997, 29(2): 103-130.

[12] 陈湘辉. 基于朴素贝叶斯算法的社交网络数据挖掘技术研究[J]. 计算机测量与控制, 2017, 25(6): 199-202.
Chen Xianghui. Social networks data mining technology research based on naive bayes algorithm[J]. Computer Measurement & Control, 2017, 25(6): 199-202.

[13] 张东亮, 董礼. 基于改进的朴素贝叶斯算法在垃圾短信过滤中的研究[J]. 计算机测量与控制, 2012, 20(2): 526-528.
Zhang Dongliang, Dong Li. Research of sms spam filtering based on optimized naïve Bayesian algorithm[J]. Henan Science, 2012, 20(2): 526-528.

[14] 刘磊, 陈兴蜀, 尹学渊, 等. 基于特征加权朴素贝叶斯分类算法的网络用户识别[J]. 计算机应用, 2011, 31(12): 3268-3270.
Liu Lei, Chen Xingshu, Yin Xueyuan, et al. Network user identification based feature weighting naive bayesian classification algorithm[J]. Journal of Computer Applications, 2011, 31(12): 3268-3270.

[15] 周志华. 机器学习[M]. 北京: 清华大学出版社, 2016: 150-151.
Zhou Zhihua. Machine learning[M]. Beijing: Tsinghua University Press, 2016: 150-151.

[16] 李彦广. 基于Spark+MLlib分布式学习算法的研究[J]. 商洛学院学报, 2015, 29(2): 16-19.
Li Yanguang. Research on distribution learning algorithm based on Spark + Mllib[J]. Journal of Shangluo University, 2015, 29(2): 16-19.

[17] 宫秀文, 张佩云. 基于PageRank的社交网络影响最大化传播模型与算法研究[J]. 计算机科学, 2013, 40(6A): 136-140.
Gong Xiuwen, Zhang Peiyun. Research on propagation model and algorithm for influence maximization in social network based on pagerank[J]. Computer Science, 2013, 40(6A): 136-140.

[18] 原野, 李晨, 田丽华. 面向微博的PageRank算法的改进与应用[J]. 计算机应用与软件, 2017, 34(3): 31-37.
Yuan Ye, Li Chen, Tian Lihua. Improvement and application of PageRank algorithm for micro-blog[J]. Computer Applications and Software, 2017, 34(3): 31-37.

[19] 王鹏, 汪振, 李松江, 等. 基于用户行为的改进PageRank影响力算法[J]. 计算机工程, 2017, 43(12): 155-159.
Wang Peng, Wang Zhen, Li Songjiang, et al. Improved pagerank influence algorithm based on user behavior[J]. Computer Engineering, 2017, 43(12): 155-159.

[20] 王天吉, 朱艳辉, 李飞. 一种基于Z-score的微博文本情感分类方法[J]. 信息与电脑, 2018(6): 40-42.
Wang Tianji, Zhu Yanhui, Li Fei. A method of emotional classification on microblog text based on Z-score[J]. China Computer & Communication, 2018(6): 40-42.

[21] 郑永广, 岳昆, 尹子都, 等. 大规模社交网络中高效的关键用户选取方法[J]. 计算机应用, 2017, 37(11): 3101-3106.
Zheng Yongguang, Yue Kun, Yin Zidu, et al. Efficient approach for selecting key users in large-scale social networks [J]. Journal of Computer Applications, 2017, 37(11): 3101-3106.

[22] 文馨, 陈能成, 肖长江. 基于Spark Graph X和社交网络大数据的用户影响力分析[J]. 计算机应用研究, 2018, 35(3): 830-834.
Wen Xin, Chen Nengcheng, Xiao Changjiang. Analysis of user influence based on social network big data and Spark Graph X[J]. Application Research of Computers, 2018, 35(3): 830-834.

[23] 王虹旭, 吴斌, 刘旸. 基于Spark的并行图数据分析系统[J]. 计算机科学与探索, 2015, 9(9): 1066-1074.
Wang Hongxu, Wu bin, Liu Yang. Parallel graph data analysis system based on Spark[J]. Journal of Frontiers of Computer Science and Technology, 2015, 9(9): 1066-1074.

[24] 段剑峰. 基于Spark的大规模图数据并行计算研究[J]. 现代计算机, 2016(7): 44-46.
Duan Jianfeng. Research on large-scale graph parallel computing based on Spark[J]. Modern Computer, 2016(7): 44-46.

[25] 孙海. Spark的图计算框架: Graph X[J]. 现代计算机, 2017(9): 120-127.
Sun Hai. Spark's graph calculation framework: Graph X[J]. Modern Computer, 2017(9): 120-127.

[26] 陈虹君. Spark框架的Graph X算法研究[J]. 电脑知识与技术, 2015, 11(1): 75-77.
Chen Hongjun. Research on Graph X algorithms in Spark framework[J]. Computer Knowledge and Technology, 2015, 11(1): 75-77.

[27] 宋宝燕, 张永普, 单晓欢. Spark-Graph X框架下的大规模加权图最短路径查询[J]. 辽宁大学学报, 2017, 44(4): 289-293.
Song Baoyan, Zhang Yongpu, Shan Xiaohuan. A shortest path method on large-scale graph based on Spark-Graph X[J]. Journal of Liaoning University, 2017, 44(4): 289-293.

[28] 张陶, 于炯, 廖彬, 等. 基于Graph X的传球网络构建及分析研究[J]. 计算机研究与发展, 2016, 53(12): 2729-2752.
Zhang Tao, Yu Jiong, Liao Bin, et al. The construction and analysis of pass network graph based on Graph X[J]. Journal of Computer Research and Development, 2016, 53(12): 2729-2752.

[29] 崔印昌. 基于Spark的社会网络分析系统的设计与实现[D]. 北京: 北京邮电大学, 2017.
Cui Yinchang. Design and implementation of social analysis system based on Spark[D]. Beijing University of Posts and

Telecommunications, 2017.

[30] Newman M E J, Girvan M. Finding and evaluating community structure in networks[J]. Physical Review E, 2004, 69(2): 026113.

[31] Usha Nandini Raghavan, Reka Albert, Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks[J]. Physical Review E, 2007, 76(3): 036106.

[32] Michael J Barber, Clark J W. Detecting network communities by propagating labels under constraints[J]. Physial Review E, 2009, 80(2 Pt 2): 026129

[33] 赵卓翔, 王轶彤, 田家堂, 等. 社会网络中基于标签传播的社区发现新算法[J]. 计算机研究与发展, 2011, 48(S3): 8-15.
Zhao Zhuoxiang, Wang Yitong, Tian Jiatang, et al. A novel algorighm for community discovery in social networks based on label propagation[J]. Journal of Computer Research and Development, 2011, 48(S3): 8-15.

[34] 艾川, 陈彬, 刘亮, 等. 基于Pregel的大规模网络传播仿真算法设计及实现[J], 中国科学: 信息科学, 2018, 48(7): 932-946.
Ai Chuan, Chen Bin, Liu Liang, et al. Design and implementation of large-scale network propagation simulation method inspired by Pregel mechanism[J]. Scientia Sinica(Informationis), 2018, 48(7): 932-946.

A large scale social networking community detection prototype system based on Spark

YE Xiaorong[1], SHAO Qing[2]

1. Institute of Scientific and Technical Information of China, Beijing 100038, China
2. KNET Co., Ltd., Beijing 100190, China

Abstract In order to effectively explore the user information in large-scale social networks and improve the understanding of the relationship between users, a community detection prototype system based on Spark is designed and developed. The ActiveMQ is used to acquire a large amount of the user data, taking advantage of the naive Bayesian algorithm provided by Spark-based MLlib to clean the user data, and using the PageRank algorithm provided by Spark-based GraphX and the Z-Score algorithm provided by MLlib to calculate the user ranking. In the prototype system, the LPA algorithm is finally used and optimized, to group the users of similar features and close ties into the same community quickly, as a foundation for further analysis and utilization of the community user data.

Keywords Spark; GraphX; MLlib; community detection

（责任编辑　陈广仁）