

基于 Spark 与词语相关度的 KNN 文本分类算法

于苹苹¹,倪建成²,韦锦涛²,曹 博¹,姚彬修¹

(1. 曲阜师范大学 信息科学与工程学院,山东 日照 276826;

2. 曲阜师范大学 软件学院,山东 曲阜 273100)

摘 要:针对 K-最近邻(KNN)分类算法在当前大数据背景下分类效率降低、分类效果不理想的问题,提出了一种基于 Spark 框架与词语相关度优化的高效 KNN 文本分类算法。在相似度计算过程中,采用词语相关度将文本词语间的关系考虑在内,对分类算法相似度计算进行优化,从而提高文本分类的准确度;依托 Spark 计算框架的内存处理机制,实现文本分类的并行化,从而提高 KNN 文本分类算法的处理效率,同时在并行化过程中建立类别-距离向量,以进一步加快文本分类的处理速度。实验结果表明,Spark 框架下基于词语相关度的 KNN 文本分类算法在保证分类效果的基础上大大提高了分类效率,较 Hadoop 平台有较好的加速比,可有效地对大数据进行分类处理。

关键词:K-最近邻;词语相关度;Spark;并行化计算

中图分类号:TP391

文献标识码:A

文章编号:1673-629X(2018)03-0087-06

doi:10.3969/j.issn.1673-629X.2018.03.018

KNN Text Classification Based on Word Relatedness and Spark Framework

YU Ping-ping¹, NI Jian-cheng², WEI Jin-tao², CAO Bo¹, YAO Bin-xiu¹

(1. School of Information Science and Engineering, Qufu Normal University, Rizhao 276826, China;

2. School of Software Engineering, Qufu Normal University, Qufu 273100, China)

Abstract: In view of the problem that K-nearest neighbor (KNN) classification algorithm is not satisfactory and inefficient under the big data background, we put forward a highly efficient algorithm of KNN based on Spark framework and word relatedness. In the calculation of the similarity, taking into the relationship between the words account by using the word relatedness, the similarity calculation of the classification algorithm is optimized to improve the accuracy of the text classification. We rely on the in-memory mechanism of Spark to realize the parallelization of text categorization, so as to rise the efficiency of KNN text categorization algorithm. At the same time, the class-distance vector is established to further speed up the processing of text categorization in the calculation. The experiments show that the proposed parallel algorithm could shorten the classification time on the basis of ensuring the classification effect. And it has better speedup, which can effectively classify the big data.

Key words: KNN; word relatedness; Spark; parallel computing

0 引 言

随着 Internet 技术以及社交媒体的发展,文本信息规模越来越大,如何高效地在海量文本信息中挖掘出有价值的信息成为当前的研究热点^[1]。文本分类技术作为文本处理的关键技术,在提高信息检索、利用等方面应用广泛。当前,使用较多的分类算法有朴素贝叶斯、支持向量机(support vector machine, SVM)、K-最近邻(K-nearest neighbor, KNN)等^[2]。由于 KNN 分

类算法具有稳定性强、准确率高等优点,在数据挖掘领域得到了广泛应用^[3]。

近年来,国内外学者对 KNN 分类算法的准确率和分类效率进行了深入研究。在准确率上,文献[4]在传统 KNN 文本分类算法的基础上提出了一种基于关联分析的 KNN 改进算法,能够较好地确定 K 值,降低时间复杂度。文献[5]提出了一种基于 KNN 文本分类的伪装入侵检测方法,使得有区分性的命令权重

收稿日期:2017-03-23

修回日期:2017-07-31

网络出版时间:2017-12-05

基金项目:国家自然科学基金(61402258);山东省本科高校教学改革研究项目(2015M102);校级教学改革研究项目(jg05021*)

作者简介:于苹苹(1991-),女,硕士研究生,CCF 会员(60032G),研究方向为并行与分布式计算、数据挖掘;倪建成,教授,博士,研究方向为分布式计算、机器学习、数据挖掘。

网络出版地址: <http://cnki.net/kcms/detail/61.1450.TP.20171205.0903.012.html>

增大,有利于更准确地表示用户的行为特征。在时间效率方面,Deng 等^[6]使用 K-means 方法对大规模训练集进行聚类裁剪,从而减少相似度的计算,提高分类效率。同时,有研究者将 KNN 算法应用于分布式平台,进一步提高分类效率。如文献[7]将 SVM 分类算法与 KNN 分类算法相结合,利用 Hadoop 云计算平台实现算法并行化。Anchalia 等^[8-9]依托 MapReduce 框架实现了 KNN 分类算法的并行化,缩短了分类时间。MapReduce 框架^[9]利用并行分布式计算模型对数据进行处理,是有效处理大数据集的关键所在。但研究人员在实验中发现 MapReduce 在 Hadoop^[10]中具有限制性^[11]:MapReduce 在执行过程中,每轮作业都需要重新启动,开销过大;同时,通过磁盘对数据进行 I/O 操作,执行效率较低。而 Spark 框架^[12]改善了以上问题,它通过建立弹性分布式数据集,在内存中对数据进行存取操作,加快了数据处理速度。

分类算法中的距离机制直接影响分类效果。传统 KNN 的距离机制仅通过词频计算两样本间的相似度,而在实际分类过程中,文本词语间具有一定相关性。同时,在大数据环境中,数据集不断增大,KNN 分类算法的计算复杂度不断增加,导致运行时间增多。

因此,文中在 KNN 相似度计算过程中引入词语相关度进行特征项加权,同时在 Spark 框架下实现 KNN 文本分类算法的并行化,以提高分类的准确度。

1 相关技术

1.1 KNN 文本分类算法

KNN 分类算法的核心原理为:通过相应距离机制计算待测样本与已知样本的相似度,找出与待测样本相似度最大的 K 个最近邻样本,利用决策函数判断 K 个样本中大多数属于哪个类别,则待分类样本也属于这个类别,并具有这个类别上样本的特性。

余弦相似度计算公式如下:

$$\text{Sim}(D_i, D_j) = \frac{\sum_{k=1}^M w_{ik} * w_{jk}}{\sqrt{(\sum_{k=1}^M w_{ik})^2 * (\sum_{k=1}^M w_{jk})^2}} \quad (1)$$

其中, w_{ik} 表示 D_i 文本中第 k 个特征的权重。

通过相似度计算得到 K 个相似度最大的样本即 K 个最近邻,并利用式(2)计算权重判断最终归类。

$$W(d, C_j) = \sum_{i=1}^K \text{Sim}(d, d_i) y(d_i, C_j) \quad (2)$$

其中, $y(d_i, C_j) = \begin{cases} 1, d_i \in C_j \\ 0, d_i \notin C_j \end{cases}$ 为类别的属性函数。

1.2 词语间相关度

一个词在文档的主题是否具有代表性与该词

条所在一定范围内其他词语在文档中的出现频率(同现频率)是相关的^[13],词语间相关度指两个词语同时出现在一定语言范围内的概率大小。

计算方法如下:

(1)依据文本预处理后特征空间基向量所包含的词条个数定义一个 $N \times N$ 维的词语相关度矩阵,其中 N 为特征空间中的基向量数目,基向量的一个词条对应矩阵中的一维数据,矩阵内各对应值初始为 0。

(2)计算全部段落范围内同时出现的词条对得到段落同现频率。在同一段落中,如果词条 T_i 和词条 T_j 同时出现,则词语相关度矩阵中的对应值加一。

(3)直到所有文档的所有段落都学习后停止。

在得到对应的同现频率后,即可计算相应的词语相关度,词条 T_i 和词条 T_j 基于段落同时出现频率的相关度 RP_{ij} 计算为:

$$RP_{ij} = \frac{PCF_{ij}}{PF_i + PF_j - PCF_{ij}} \quad (3)$$

其中, PF_i 和 PF_j 表示词条 T_i 和 T_j 在文档 d_j 中所有段落范围内的出现频率。

1.3 Spark 框架

Spark 是由 UC Berkeley 开发的一种基于内存的计算框架。Spark 框架由两个主要部分组成:弹性分布式数据集和并行运算。RDD 是 Spark 框架计算中重要的概念,是 Spark 框架的核心和基础。

RDD 是 Spark 框架的核心,是各个集群节点中数据共享的一种有效抽象。Spark 所执行的数据处理是基于 RDD 进行的。RDD 是可并行的数据结构,同时允许用户将数据存储在内存中,方便用户对数据的重复调用,减少了磁盘 I/O 的负载。RDD 是不可变的集合,不能在原有 RDD 上实现内容的修改,只能创建一个新的 RDD:

(1)通过对文件共享系统(HBase、HDFS、Hive)的读取得到 RDD;

(2)通过驱动程序中用作并行计算的 Scala 集合创建;

(3)通过对已存在的 RDD 执行转换操作新建一个 RDD;

(4)通过持久化操作进而转变现有的 RDD。

并行运算包括转换(transform)和动作(action)。转换操作只依据原有的 RDD 定义一个新的 RDD,而不对它进行直接计算;动作是对某个值立即进行计算,并返回结果给驱动程序^[14]。

对 RDD 的控制还可利用缓存和分区操作。将 RDD 缓存在内存中,便于下次计算时的重复利用,减少了磁盘开销。RDD 可根据 key 来指定分区顺序,将数据划分到不同分区中。

2 Spark 框架下基于词语相关度的 KNN 算法

为了提高文本分类效率,分类算法的并行化处理是当前的一大趋势,但通过研究发现,在 KNN 文本分类的一般并行化过程中会降低分类准确率。因此文中在训练样本与待测样本的相似度计算过程中引入词语相关度,提高分类准确度并在 Spark 计算框架下实现并行化,降低运算时间。

2.1 基于词语相关度的相似度计算

通常情况下,传统 KNN 相似度计算方法单纯使用 TF-IDF^[15] 计算相关度,忽略了文本数据本身词语间的关系,易导致文本内容表达的不完整,影响文本分类的准确率^[16]。因此,定义合适的距离机制是分类准确的前提。文中在 KNN 分类算法相似度距离计算中引入词语相关度概念。

基于词语相关度的权值通过该词条词频与该词条在一定语言范围内其他词的平均相关度的乘积而得到,选用段落范围内词语相关度。词条 T_i 基于段落同现频率的词语相关度的加权值公式如下所示:

$$W_{ij} = TF_{ij} * (1 + \rho * (\sum_{k=0, k \neq i}^K PR_{ik}) / k) * IDF_{ij} \quad (4)$$

其中, K 表示该文档中除词条 T_i 外其他词条的个数。

式(4)含义为:基于词语相关度的权重等于原有的 $TF * IDF$ 乘以平均相关度。通过该方法计算的平均相关度结果较小,不利于区分,所以添加区分系数 ρ 。则新的相似度计算公式为:

$$Sim(D_i, D_j) = \frac{\sum_{k=1}^M w_{ik} * w_{jk}}{\sqrt{(\sum_{k=1}^M w_{ik})^2 * (\sum_{k=1}^M w_{jk})^2}} \quad (5)$$

2.2 Spark 框架下基于词语相关度的 KNN 算法

对传统 KNN 文本分类算法的并行化处理会提高分类的效率,但并行化处理通常会降低分类的准确度。因此,文中在 Spark 框架下实现 KNN 算法并行化的同时引入词语相关度对特征项进行加权处理,提高文本分类的准确度和分类效率。

2.2.1 预处理并行化

首先采用中科院分词软件 ICTCLAS2015 对数据集进行分词、去除停用词等预处理。去停词结果以 <key, value>形式给出,并利用 2.1 中提出的基于词语相关度的 TF-IDF 对向量进行计算,得到相应权重。在文本预处理之前,将训练集与测试集提交到 HDFS 中并转化为 RDD 对象,分发给每个节点,Map 函数通过静态方法 UseDistributedCache() 实现对缓存数据的调用。 万方数据

2.2.2 分类并行化

在 KNN 算法并行化过程中,以 MapReduce 模型为基础,采用 Spark 计算框架来实现算法的并行化。首先将训练集 D 与测试集 T_i 在 HDFS 中读取出来并建立 RDD 对象,然后将训练集分割到相应分片中,并将数据集进行缓存以备重复利用,通过分区处理每个 Map 都将对相近数量的训练集进行处理。

文中并行化算法为了使分类准确,对每个测试样本指定一个 textID 作为 key 值,然后利用 RangePartitioner 函数将测试集分成不同的子集 T_i ,进而通过转换操作 filterByRange 函数将测试集 T_i 读取进每一个 Map 任务中。算法 1 描述了此过程。

算法 1: Spark 框架下基于词语相关度的 KNN 算法。

输入:训练集 D ,测试集 T , K

输出:分类结果

(1)将训练集在 HDFS 中读取并作为 RDD 对象读取进 Map 中;

(2)将测试集在 HDFS 中读取并作为 RDD 对象;

(3)对训练集和测试集进行规范化处理,并缓存;

(4)Map 过程:

使用式(5)计算测试样本与训练样本的距离,并得到每个 Map 中最小的 K 个近邻;

将最小的 K 个近邻建立成类别-距离组成的 CD 向量,并将 CD 定义进 value 值中;

将结果发布给 Reduce。

(5)Reduce 过程:

在 Map 中读取结果;

更新距离 CD_{new} 向量。对于每个测试样本,从最近的邻居开始逐一比较每个邻居的距离值,如果 Map 任务转发来的距离小于当前值,则使用该值更新 Reduce 过程中相应位置的类和距离;否则,继续与下一个值比较;

通过判定公式计算最后分类;

输出结果。

步骤(1)、(2)将训练集与测试集在 HDFS 中读取出来并建立为 RDD 对象,然后对训练集 D 进行分片,同时设置 K 值。为了使分类准确,将测试集作为 RDD 读取但是不与训练集一起分片,而是根据 key 值读取进每个 map 中比较每一个测试样本和所有的训练集。

由于使用余弦相似度来计算样本间的相似度,将两个数据集在步骤(3)中进行规范化处理。同时,将数据集进行缓存以备重复利用。

在 Map 过程中,首先利用式(5)计算测试样本与每个分片中训练样本的相似度距离。通过相似度计算,在每个 Map 任务中都将得到最近的 K 个邻居,将

这 K 个邻居的类别与相应的相似度距离保存为 $\langle \text{class}, \text{distance} \rangle$; CD , 并将 CD 定义进 value 值中, 即 $\langle \text{key}; \text{textID}, \text{value}; \text{CD} \rangle$, 同时根据相似度距离进行升序排列, 如图 1 所示。每进行一次 Map 任务处理, 就启动一次 Reduce 任务, 并将中间结果转发到 Reduce 任务中。

每完成一次 Map 任务, Reduce 就将 CD_{new} 中的距离值与来自 Map 中 CD 的对应距离值进行比较, 直到

得到最终 K 个近邻。所有的值更新完毕后, CD_{new} 将包含所有待测样本的 K 个最近邻的类和距离。最后, 执行 KNN 算法的判定函数确定测试集的所属类别, 并将结果作为 Reduce 阶段的最终结果输出。对于每个测试样本, Reduce 过程将采用 ReduceByKey 根据之前描述的函数聚集 value 。

图 1 给出了 KNN 分类并行化处理流程。

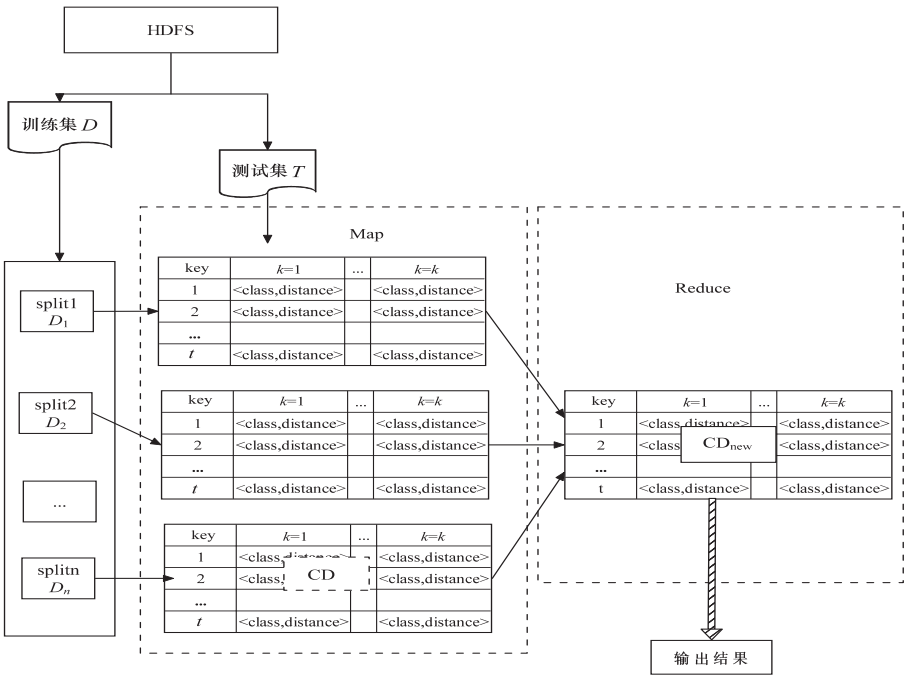


图 1 Spark 框架下 KNN 算法并行化过程

3 实验

3.1 实验环境与数据集

在 Hadoop 平台上部署了 Spark 计算框架, 通过 Vcenter 创建了 6 台虚拟机, 其中包含 1 个 Master 节点和 5 个 Slave 节点。虚拟机中操作系统均为 Ubuntu 14.04.3-Server-amd64 版本, Hadoop 版本为 2.6.0, Spark 版本为 1.4.0, Java 开发包版本为 jdk1.7.0_79, 程序开发工具为 Eclipse Mars.1 Release (4.5.1)。其中 Master 节点担任 NameNode 角色, 主要负责管理与调用并维护着所有文件的元数据, Slave 节点担任 DataNode 角色, 根据 NameNode 的调用检索和处理数据。

数据集采用 Sogou 实验室提供的分类语料库, 选择语料库中整理过的搜狐新闻网站新闻语料以及对应的分类信息。实验中采用的文本为教育、互联网、财经、军事、旅游、体育、文化、健康、招聘等 20 大类, 每个类别中有 2 000 篇文本, 共 40 000 篇文档。逐个在每个类别中随机选取 500 篇文本组成训练集。为了确保实验的充分性和有效性, 将剩余文本划分为不同大小, 组成不同规模的测试集, 如表 1 所示。

表 1 测试集

序号	规模/篇	说明
测试集 D_1	1 000	20 类, 从数据集每个类别中各随机抽取 50 篇
测试集 D_2	5 000	20 类, 从数据集每个类别中随机抽取一定数量, 一个类别最多抽取 500 篇
测试集 D_3	10 000	10 类, 从相应数据集中各抽取 1 000 篇
测试集 D_4	20 000	20 类, 从数据集中相应类别各抽取 500 篇
测试集 D_5	30 000	完整测试集

3.2 评价指标

选用分类方法中常用的准确率 (Pr) 和查全率 (Re)。准确率用于表示分类的正确性, 即检测出分类文档中正确分类的文档所占的比率; 查全率表示分类的完整性, 即所有应分类的文档中被正确分类的文档所占的比率, 公式如下:

$$\text{Pr}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i}$$
(6)

$$\text{Re}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i}$$
(7)

同时为了综合评价分类效果, 使用 F_1 值:

$$F_1 = \frac{Pr * Re * 2}{Pr + Re}$$

(8)

选用加速比对 Spark 并行化框架进行评价:

$$Speedup(g) = \frac{\text{在一个节点上使用的时间}}{\text{在 } n \text{ 个节点上使用的时间}}$$

(9)

3.3 实验结果与分析

实验首先基于词语相关度对 KNN 算法的影响进行验证。以准确率、查全率以及 F_1 值为指标,在单一节点上对比基于词语相关度算法、距离加权 KNN 算法以及传统 KNN 算法的性能,如表 2 所示。

表 2 三种算法性能比较

测试集	传统 KNN			距离加权 KNN			基于词语相关度		
	Pr/%	Re/%	F_1	Pr/%	Re/%	F_1	Pr/%	Re/%	F_1
D_1	80.8	79.1	79.94	83.4	79.6	81.74	83.3	79.7	81.46
D_2	82.1	79.6	80.83	85.3	82.1	84.38	85.1	81.4	83.21
D_3	88.4	83.3	85.77	91.2	84.4	87.67	92	84.7	88.2
D_4	90.2	85.7	87.89	92.1	86.1	89.46	93.7	87.2	90.33
D_5	91.6	87.8	89.66	94.8	88.2	91.38	96.5	89.1	92.65
Avg	86.62	83.1	84.82	89.98	84.08	86.93	90.12	84.42	87.17

由表 2 可知,在串行状态下,基于词语相关度的算法较传统 KNN 算法在准确率上提高了 2.5~4.9 个百分点。而与距离加权 KNN 算法相比,在小数据集上两者分类准确率基本相似,但在较大数据集上,基于词语相关度的 KNN 算法的准确率随数据集的增大而逐渐提高。在查全率方面,文中算法较传统 KNN 算法与距离加权都有所提高,平均查全率提高了 1.3 和 0.34 个百分点。在综合指标 F_1 中,可以看出文中算法分类效果更佳,较传统 KNN 算法与加权 KNN 算法平均值各提高了 2.35 和 0.24 个百分点。因此,在串行状态下,基于词语相关度算法可以有效提高 KNN 文本算法的分类效果。

为了验证文中算法是否能够有效弥补并行化过程中分区操作带来的分类准确率下降的缺陷,对传统 KNN 算法、基于 Spark 框架的传统 KNN 算法(3 个节点)以及 Spark 框架下基于词语相关度优化的 KNN 算法(3 个节点)在不同测试集下进行了比较,结果如表 3 所示。

表 3 并行化算法性能比较

测试集	传统 KNN 算法		基于 Spark 框架的传统 KNN 算法		Spark 下基于词语相关度的 KNN 算法	
	P_c /%	T /s	P_c /%	T /s	P_c /%	T /s
D_1	80.8	267	78.4	64	81.8	57
D_2	82.1	372	80.3	71	83.8	62
D_3	88.4	596	86.2	82	90.1	74
D_4	90.2	1 276	87.7	117	91.9	93
D_5	91.6	2 265	89.8	123	93.2	104
Avg	86.62	795.4	84.48	-	88.16	-

由表 3 可知,在运行时间方面,Spark 框架下基于词语相关度的 KNN 算法运行时间较传统 KNN 算法缩短了 5%~26%,与 Spark 框架下的传统 KNN 算法运行时间基本一致。表明 Spark 框架下的并行化 KNN 文本分类算法能够实现大规模文本数据的分类。而在准确率上,Spark 框架下的传统 KNN 算法的准确度较串行状态传统 KNN 算法平均降低了 2.1%,这是由于通常情况下的并行化算法要对数据集进行分区操作,会降低分类的准确度。而 Spark 框架下基于词语相关度的 KNN 文本分类算法较串行状态下的传统 KNN 分类算法准确率平均提高了 1.56 个百分点,较 Spark 框架下的传统 KNN 算法提高了 3.68 个百分点。这是由于使用词语相关度对相似度进行了优化,弥补了由并行化导致的准确度的影响。因此,文中利用词语相关度建立的距离机制有效改善了通常情况下并行化分区操作对准确度的影响。

在 4 个节点的情况下,Spark 框架与 Hadoop 平台中算法在 5 个不同规模数据集的运行时间如图 2 所示。

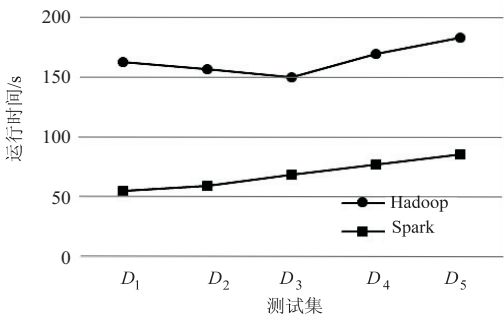


图 2 两种架构运行时间比较

由图 2 可知,Hadoop 在处理小数据集时,运行时间较长,这是由于 Hadoop 平台处理过程中磁盘 I/O 计算时花费时间较多,在处理小样本时较为明显。而 Spark 框架将数据存储在内存中不需要磁盘 I/O,因此在各规模数据集下运行时间平缓增长并优于 Hadoop。

加速比主要用于衡量一个系统的扩展性能。当采用 D_4 数据集时,Spark 和 Hadoop 在不同节点下的加速比比较如图 3 所示。

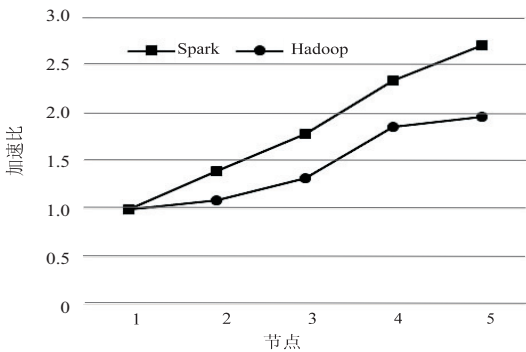


图 3 两种架构加速比比较

由图 3 可知,加速比在 Spark 框架下随节点增加呈线性增长趋势,由此可知随着节点的增加能更好地提高分类处理效率,这说明 Spark 框架在处理 KNN 分类算法上具有较好的加速比。并且由此可知,节点数不断增加时 Spark 的加速比优势将会更为凸显。因此,Spark 优于 Hadoop 平台具有较好的加速比,能够高效地实现大数据集的处理。

4 结束语

针对 KNN 分类算法在当前大数据环境下的分类问题,结合词语相关度对常用的 KNN 分类相似度进行优化,并在 Spark 框架下实现算法的并行化,提高分类效率。实验结果表明,文中提出的并行化算法在保证分类准确率的情况下,较传统 KNN 算法在时间效率上有明显提高。但该算法没有考虑相似度中其他属性值的影响,分类效果仍有可提高的空间。

参考文献:

- [1] 王小林,陆骆勇,邵伟鹏. 基于信息熵的新的词语相似度算法研究[J]. 计算机技术与发展,2015,25(9):119-122.
- [2] 苏金树,张博锋,徐 昕. 基于机器学习的文本分类技术研究进展[J]. 软件学报,2006,17(9):1848-1859.
- [3] 王邦军,李凡长,张 莉,等. 基于改进协方差特征的李-KNN 分类算法[J]. 模式识别与人工智能,2014,27(2):173-178.
- [4] 范恒亮,成卫青. 一种基于关联分析的 KNN 文本分类方法[J]. 计算机技术与发展,2014,24(6):71-74.
- [5] 王秀丽. 基于 K 最近邻文本分类的伪装入侵检测[J]. 小型微型计算机系统,2014,35(12):2650-2654.
- [6] ZHANG L,ZHANG C J,XU Q Y,et al. Weighted-KNN and its application on UCI[C]//Proceedings of the 2015 IEEE international conference on information and automation. [s. l.];

IEEE,2015:1748-1750.

- [7] 李正杰,黄 刚. 基于 Hadoop 平台的 SVM_KNN 分类算法的研究[J]. 计算机技术与发展,2016,26(3):75-79.
- [8] LU S P,TONG W Q,CHEN Z J. Implementation of the KNN algorithm based on Hadoop[C]//2015 international conference on smart and sustainable city and big data. Shanghai, China:IET,2015:123-126.
- [9] ANCHALIA P P,ROY K. The k-Nearest neighbor algorithm using MapReduce paradigm[C]//Proceedings of the 2014 5th international conference on ISMS. [s. l.]; IEEE,2014:513-518.
- [10] DEAN J,GHEMAWAT S. MapReduce:simplified data processing on large clusters[J]. Communications of ACM,2008,51(1):107-113.
- [11] GHEMAWAT S. The Google file system[J]. ACM SIGOPS Operating Systems Review,2003,37(5):29-43.
- [12] GROLINGER K,HAYES M,HIGASHINO W A,et al. Challenges for MapReduce in big data[C]//Proceedings of the IEEE world congress on services. Anchorage, AK: IEEE,2014:182-189.
- [13] ZAHARIA M,CHOWDHURY M,DAS T,et al. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing[C]//Proceedings of the 9th USENIX conference on networked systems design and implementation. Berkeley:USENIX Association,2012:141-146.
- [14] 楼华锋. 面向文本聚类的语义加权研究[D]. 上海:上海交通大学,2010.
- [15] ZAHARIA M,CHOWDHURY M,FRANKLIN M J,et al. Spark:cluster computing with working sets[C]//USENIX conference on hot topics in cloud computing. Boston, MA: USENIX Association,2010:1765-1773.
- [16] 梁喜涛,顾 磊. 中文分词与词性标注研究[J]. 计算机技术与发展,2015,25(2):175-180.

(上接第 86 页)

- [7] 顾亦然,蒋璐璐. 一种改进无线传感器网络的 DV-Hop 定位算法[J]. 计算机技术与发展,2012,22(12):109-112.
- [8] 潘琢金,刘文春,罗 振,等. 无线传感器网络 DV-Hop 定位算法的改进[J]. 计算机工程与设计,2016,37(7):1701-1704.
- [9] 王小辉,李圣普,吕海莲. 基于布谷鸟算法的 WSN 节点定位研究[J]. 计算机技术与发展,2014,24(12):208-211.
- [10] ZHOU Gongqian,YANG Lujing,LIU Zhong. Wireless sensor network node localization based on error bound DV-Hop algorithm[C]//28th Chinese control and decision conference. [s. l.]; IEEE,2016:2390-2396.
- [11] ZHANG Ying,ZHU Zhuling. A novel DV-Hop method for localization of network nodes[C]//Proceeding of the 35th Chi-

nese control conference. [s. l.];[s. n.],2016:8346-8351.

- [12] 刘士兴,黄俊杰,刘宏银,等. 基于多通信半径的加权 DV-Hop 的定位算法[J]. 传感技术学报,2015,28(6):883-887.
- [13] WANG Ying,FANG Zhiyi,CHEN Lin. A new type of weighted DV-Hop algorithm based on correction factor in WSNs[J]. Journal of Communications,2014,9(9):699-705.
- [14] ZHENG Jiuhe, QIAN Huanyan, WANG Lie. An improved DV-Hop positioning algorithm for wireless sensor network[C]//IEEE international conference on progress in information and computing. [s. l.]; IEEE,2015:492-497.
- [15] 侯志伟,包理群,安丽霞. 基于残差加权的三维 DV-Hop 改进 WSN 定位算法[J]. 计算机应用与软件,2016,33(1):112-115.