第41卷 第6A期
2014年6月

计 算 机 科 学
Computer Science

Vol. 41 No. 6A
June 2014

云计算环境下的一种改进的贝叶斯文本分类算法

张 琳 邵天昊

(南京邮电大学计算机学院 南京 210003)

摘 要 基于云计算的思想运用 MapReduce 模型解决了传统贝叶斯分类算法不适应大规模数据的缺陷,很大程度地提高了分类速度。结合并行化的特点对算法进行了相应的改进,加入了同义词合并和词频过滤等方法,使得向量维数降低,减少了误判。然后对其中特殊的关键词进行加权,增强了分类准确性。最后在 Hadoop 云计算平台上进行了实验,证明了传统的文本分类算法并行化后在 Hadoop 上运行具有较好的加速比,并且改进后的算法能够提高分类精确度。

关键词 云计算,文本分类,并行化,Hadoop

中图法分类号 TP391.1 **文献标识码** A

Improved Bayesian Text Classification Algorithm in Cloud Computing Environment

ZHANG Lin SHAO Tian-hao

(College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract Used the idea of cloud computing, according to MapReduce model to solve the traditional Bayesian classification algorithm suited to large-scale data deficiencies, greatly improved the speed of classification. And the combination of the characteristics of the parallel algorithm was improved accordingly. Adding synonyms and word frequency filtering combined approach allows vector dimensionality reduction, reducing false positives. Wherein the particular keyword was then weighted to enhance the accuracy of classification. Finally, the Hadoop cloud computing platform was experimentally proved that the traditional text classification algorithm after parallelization on Hadoop cloud computing platforms, has better speedup, and the improved algorithm can improve the classification accuracy.

Keywords Cloud computing, Text classification, Parallel, Hadoop

1 引言

随着互联网的发展,大量的数据呈几何级的速度快速增长。据统计,每隔两年,互联网上的数据量就会翻一番。于是“大数据”的概念也就应运而生。随之而来的是大量需要深度分析的数据,其中,Web 文档是最常见也是涵盖范围最广的一类数据。

文本的自动分类处理可以提高信息检索的质量和效率,已经应用于很多领域。贝叶斯[1-4]分类方法在众多概率分类算法中是一种简单有效的方法,并且在某些领域表现出很好的性能。然而该方法对于长篇幅的文档的分类效果不尽人意,并且在处理大规模 Web 文档时会出现诸如训练集生成速度缓慢、机器学习效率低下、文字篇幅超过一定长度之后产生大量误差等问题。

而云计算可以有效地解决计算速度的问题,通过 Map/Reduce[5-8]模型将原有算法并行化,以键/值的方式来分析处理数据集中的记录。只需要将问题分解成可并行操作的子问题,设计 Map 和 Reduce 两个函数,就能运用分布式系统解决问题而不需要考虑实现细节。

在当前大数据的背景下,传统的文本分类方式已经渐渐不适合现在用户的需求。而云计算虽然可以一定程度上解决速率的问题,但是如何合理地运用云计算,充分发挥其性能依旧是一个值得研究的问题。

针对上述情况,本文第 2 节介绍了传统的朴素贝叶斯算法,并对其缺陷和并行化的可行性进行了分析;第 3 节针对并行化的朴素贝叶斯算法进行了改进;第 4 节介绍了如何在云计算平台上通过 Map/Reduce 模型实现该算法;第 5 节通过使用 Hadoop[9-10]云计算平台进行相关实验,证明该方法可以有效地提升大批量文档的分类速度及准确率,并且在该平台中相比于朴素贝叶斯算法,其拥有更快的速度和更高的精度。

2 朴素贝叶斯分类算法及其并行化

2.1 朴素贝叶斯分类方法

贝叶斯分类器是一种典型的基于贝叶斯定理的概率统计分类器。其主要思想是分别计算每个已有类别对于一待分类文档的条件概率,然后将该文档归为条件概率最高的那个类别。其主要分类步骤如下:

(1)分类器的构建

本文受省属高校自然科学基金(13KJB520017),南京邮电大学科研基金(NY213155)资助。

张 琳(1980—),女,博士后,副教授,硕士生导师,主要研究方向为云计算、网络安全、信任、可信计算等,E-mail:zhangl@njupt.edu.cn;邵天昊(1990—),男,硕士生,主要研究方向为云计算、数据挖掘等。

将训练集按类别分组，然后统计每个类包含哪些特征词及每个特征词出现的次数。所有训练集统计保存后分类器准备工作完成。

(2)待分类文档的归类

计算特征词属于每个类别的概率向量$(x_1, x_2, x_3, \cdots, x_n)$。

$$x_k = P(W_k|C_j) = \frac{1+\sum_{l=1}^{|D|} N(W_k, d_l)}{|V|+\sum_{s=1}^{|V|}\sum_{l=1}^{|D|} N(W_s, d_l)} \tag{1}$$

其中，$N(W_k, d_l)$为W_k在d_l中的词频，W_k表示某一特征词，d_l表示C_j类中的某一训练文本，$|V|$为特征词总数，$|D|$为C_j类中的文档数。

为了下文方便说明，这里将公式简化为

$$P(W_k|C_j) = \frac{1+T}{VC+M} \tag{2}$$

其中，$T=\sum_{l=1}^{|D|} N(W_k, d_l)$表示$W_k$在$C_j$类中出现的次数，$VC=|V|$表示总特征词数，$M=\sum_{s=1}^{|V|}\sum_{i=1}^{|D|} N(W_s, d_i)$表示$C_j$类中所有特征词出现的总次数。

计算测试文本属于每个类的条件概率，选取最大值。

$$P(C_i/d) = \arg\max P(C_i)\prod_{j=1}^{m} P(w_j/C_i) \tag{3}$$

其中，$P(C_i)$为类C_i的先验概率，m为特征项数目。

朴素贝叶斯算法的流程图如图1所示。

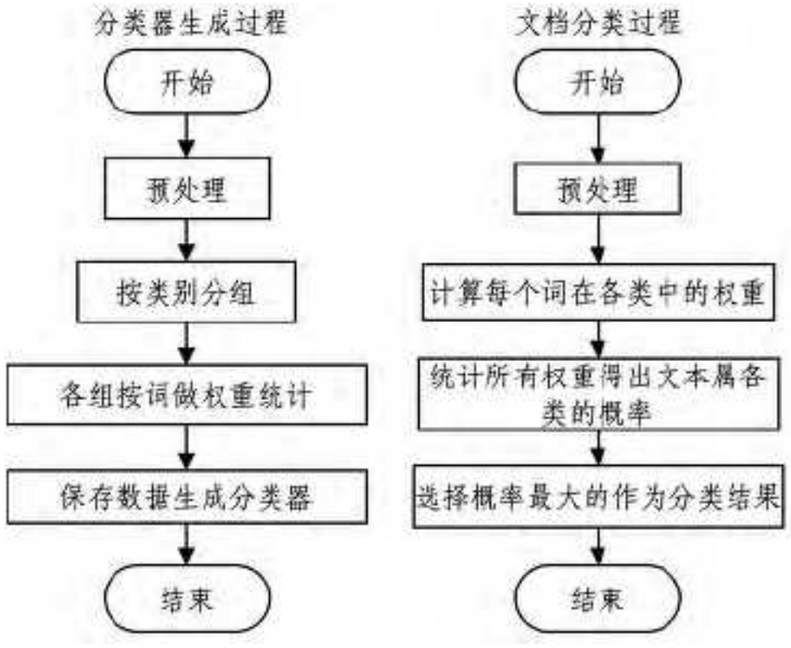

图1 朴素贝叶斯算法流程图

2.2 朴素贝叶斯算法的缺陷

由上述过程可以看出，造成训练集生成速度缓慢，机器学习效率低下，文字篇幅超过一定长度之后会产生大量误差等问题的原因如下：

(1)分类的准确性很大程度上取决于训练集的大小，如果训练集过小，分类结果将会有很大的偶然性。

(2)长篇幅文档会产生大量不能突出其属性的特征词，模糊了分类结果。

(3)不论是大量的训练集学习过程，还是长篇幅文档的分解统计分类工作，都需要占用相当大的计算资源，单台机器已经渐渐不能胜任该任务。

2.3 贝叶斯算法并行化的可行性

分析贝叶斯算法的流程可以发现，无论是分类器的生成过程，还是文档的分类过程，都是由许多组独立的计算叠加而成。因此，其本身就具有拆分和并行计算的可行性。下面对照图1的传统串行贝叶斯分类流程给出并行化后的分类流程图并加以说明。

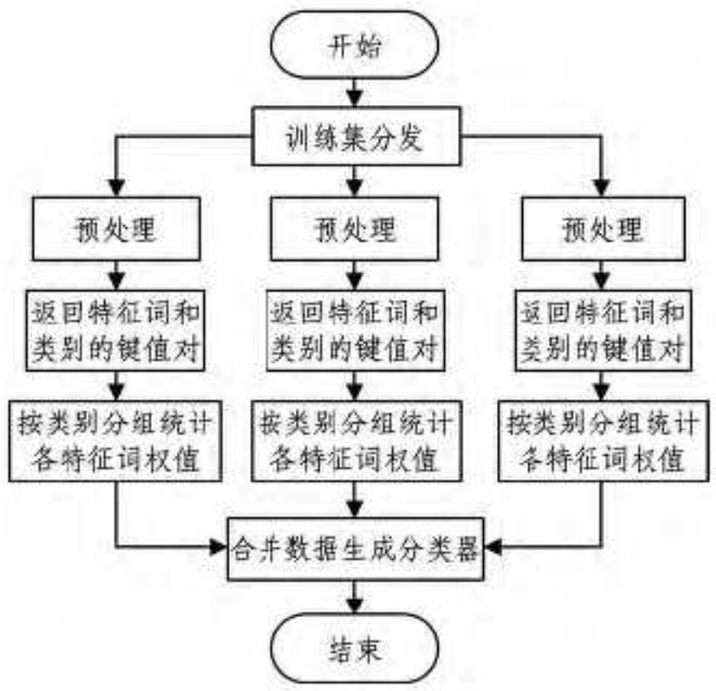

图2 分类器并行生成流程图

(1)分类器生成并行化

每一篇训练文本的学习过程都是相同的并且相互并无联系。因此将串行的一篇接一篇的学习过程并行，把训练集分片后由各个节点分别将训练文本分词统计学习。分片的最小单位为一篇文本，可以根据具体节点数目情况选择合适的分片大小。流程图如图2所示。

(2)文档分类过程并行化

分类的实质是计算文档的每个特征词属于各个类的概率并叠加，最终得到文档属于各个类的概率并取最大值作为分类结果。该过程最耗时的部分在于需要大量$P(w_j/C_i)$的计算，而各个特征词的计算过程是相互独立的，因此将特征词逐一统计的过程并行，各个节点分别完成一部分$P(w_j/C_i)$的计算，最后合并输出得到分类结果。流程图如图3所示。

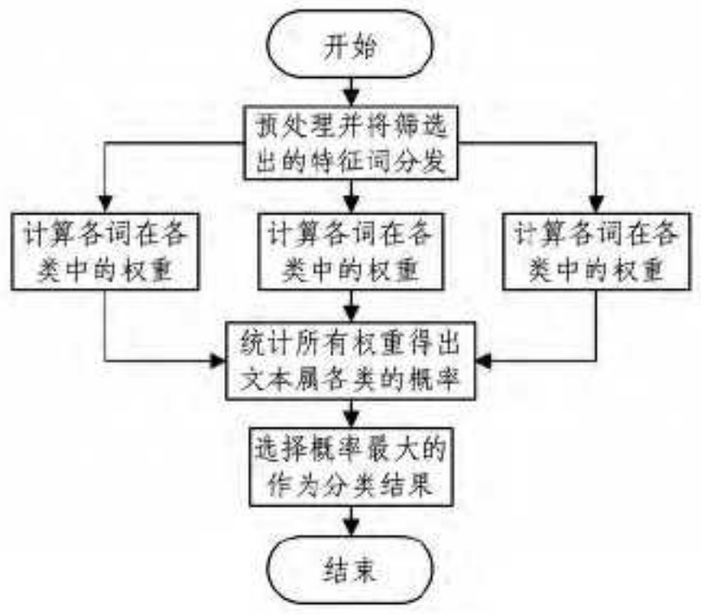

图3 文档分类并行流程图

2.4 基于朴素贝叶斯算法的改进

传统的贝叶斯分类算法的弊端在于，当文本较大时会产生过大的向量空间维数，从而使得分类结果呈离散性变化。因此在云计算出现之前，有着多种提升传统贝叶斯分类算法效果的改进。然而这些改进方法只是在串行情况下增加各种环节来提升分类精度，虽然有着不错的效果，但是对于云计算而言，复杂的分类过程并行化相对于朴素贝叶斯算法也提升了很多。

因此，本文提出了一种适合并行化的提升精度的改进方法，其只需压缩一次 Map/Reduce 过程就可提高分类精度。先用过滤特征词的方法来降低向量维数，再加入核心关键词

的概念以突出核心词向量的作用。

2.4.1 特征词选取及过滤

首先，对于读取到的每个词，过滤掉其中的停用词、虚词、介词、副词等等之后，将其当作特征词记录下来。

然后，把特征词进行同义词过滤，合并同义词的特征向量。

最后过滤掉出现频率过低的一部分特征词，该阈值的设置可以根据具体训练集的大小而定，本文选择的阈值为 8%，即，每 100 个训练文本中出现次数少于 8 次的特征词将会被剔除。

经过上述一系列过滤后，向量维数将会有相当大的降低。

2.4.2 核心关键词

在实际阅读写作当中，人们更加关注标题、摘要、首尾句中出现的词条。因此这里引入了核心关键词的概念，用来强化上述词条在分类中的作用。

在训练时，将标题或摘要等重要位置中的词条单独统计，同样将其结果经过上节所述的过滤，过滤的阈值可以适当降低，如果训练文本数量较少，可以设为 0%。过滤后得到的就是该类的核心关键词，并记录其词频 n。

当某特征词 W_k 属于核心关键词时，将其总词频乘以 $\sqrt[n]{n}$，即 $\sqrt[n]{n}\times\sum_{l=1}^{|D|}N(W_k,d_l)$，然后继续按照朴素贝叶斯算法计算后验概率。

由此加权来突出核心关键词向量的影响，强化分类效果。

3 云计算环境下的分类算法并行化

3.1 分类器的并行化生成

在原有公式基础上加入加权值 $k=\sqrt[n]{n}$，即，

$$P(W_k|C_j)=\frac{1+T\times k}{VC+M} \tag{4}$$

计算任务是完成对上述 4 个变量 T、k、VC、M 的统计，将通过两次计算完成。

将训练文本输入平台，〈key，value〉对应为〈文本类别，文本内容〉。

Map 任务：首先对每个类别建立相应的目录，并且创建相应的子目录，用以统计核心关键词。对所有 Value 调用分词工具（这里选用开源的极易中文分词插件进行改造，在此基础上增加一个过滤器，用来实现特征词过滤和核心关键词筛选的功能）。通过过滤后，每产生一个特征词就执行以下步骤：

(1) 输出中间结果$\langle(C_j,w_i),1\rangle$；

(2) 输出中间结果$\langle(C_j,\text{count}),1\rangle$；

(3) 输出中间结果$\langle w_i,1\rangle$；

(4) 如果读取到 core 标志就输出中间结果$\langle(C_j,w_i,\text{core}),1\rangle$，其中 core 表示核心关键词标志。

Combiner 函数：该函数的功能是将 map 函数产生的中间结果先行在本机上合并，以减少各节点向网络上发送的数据量，减轻网络负担。

(1) 对$\langle(C_jw_i),1\rangle$集，如 key 值相同则求和，输出$\langle(C_j,w_i),n\rangle$，$n$ 代表求和结果。

(2) 对$\langle(C_j,\text{count}),1\rangle$集，如 key 值相同则求和，输出$\langle(C_j,\text{count}),m\rangle$，$m$ 代表求和结果。

(3) 对$\langle(C_j,w_i,\text{core}),1\rangle$集，如 key 值相同则求和，输出$\langle(C_j,w_i,\text{core}),\text{num}\rangle$，num 代表求和结果。

(4) 对$\langle w_i,1\rangle$集中 key 值相同的直接删除，仅输出剩下的键值对。

Reduce 任务：统计从各节点接收到的中间结果，得到所需变量。其步骤如下：

(1) 对$\langle(C_j,w_i),n\rangle$集，如 key 值相同则求和，即可得某一特征词 w_i 在某类文档中出现的次数 T。

(2) 对$\langle(C_j,w_i,\text{core}),\text{num}\rangle$集，如 key 值相同则求和，即得到某一核心关键词的词频 n。

(3) 对$\langle(C_j,\text{count}),m\rangle$集，如 key 值相同则求和（这里 count 仅表示一个计数变量，所以每个特征词所产生的该中间结果的 key 都是一样的），即得到某类中所有特征词的总次数 M。

(4) 对$\langle w_i,1\rangle$集，删除其中 key 值相同的，统计剩余键值对的数量，得到整个训练中的特征词词汇总数 VC。

至此，4 个变量 T、k、VC、M 的统计完成，训练集学习完毕，分类模型建立。

3.2 测试文本分类的并行化

对测试文本进行同样的分词过滤，将各个特征词 w_i 输入分类模型中，〈key，value〉对应为〈测试文本编号，w_i〉。

Map 任务：对输入的〈测试文本编号，w_i〉，从分类模型中取出相应参数 T、k、VC、M，计算其条件概率 $P=P(W_k|C_j)=\frac{1+T\times k}{VC+M}$，并输出中间结果$\langle(\text{测试文本编号},C_j),P\rangle$。

Reduce 任务：对$\langle(\text{测试文本编号},C_j),P\rangle$集，key 值相同的求 P 的积，得到该文档对于各个类别的后验概率 $P(C_i/d)$，并且取出最大值作为分类结果。

4 云平台上的实验结果及分析

测试环境为局域网，其由 Hadoop 平台组成，计算机配置为 CPU i5，内存 8G，Linux 操作系统，其中，1 台是主节点服务器，其他 9 台是子节点服务器，配置的集群结构如图 4 所示。

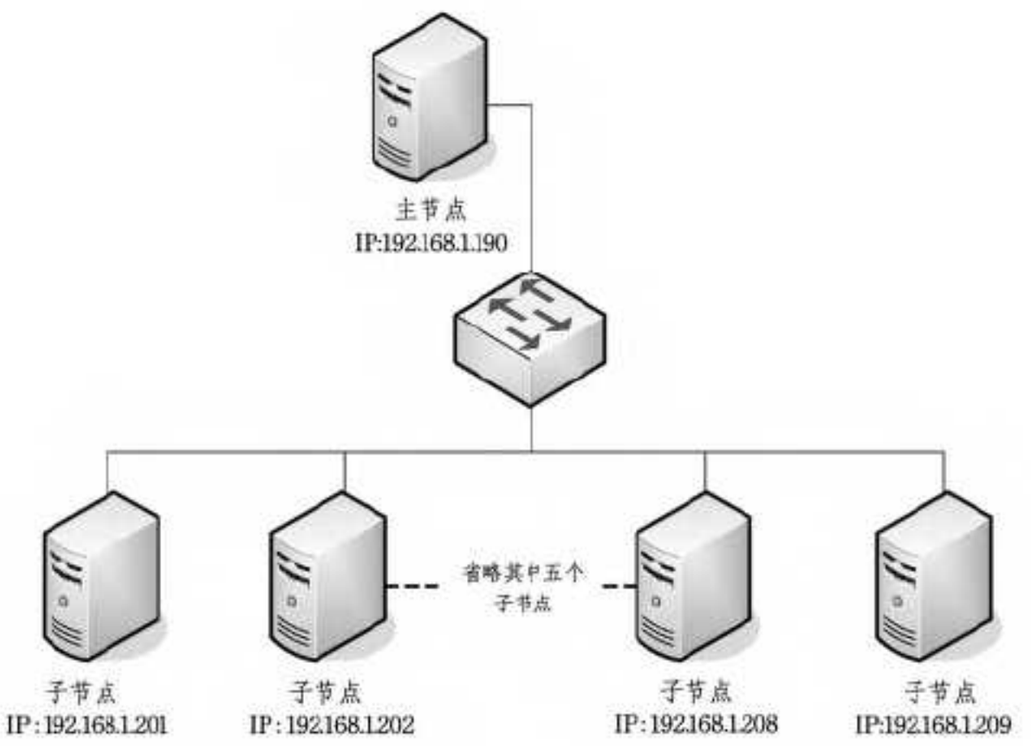

图 4 集群结构图

实验数据采用搜狗实验室提供的互联网语料库资源 SougouC进行训练和测试，一共 10 个类别：汽车、财经、IT、健康、体育、旅游、教育、招聘、文化、军事。

(1) 加速性能比较

分别选择 1、3、5、7、10 个节点对上述数据集进行交叉分类测试实验。采用朴素贝叶斯分类方法和改进后的分类方法分别进行实验，结果如图 5 所示。

可以发现随着节点数的增加，对于相同规模数据的处理时间有了明显的缩短，并且改进后的方法拥有更快的计算速度。说明云计算平台的并行计算方法确实可以大幅度地提高大批量文档分类的效率。

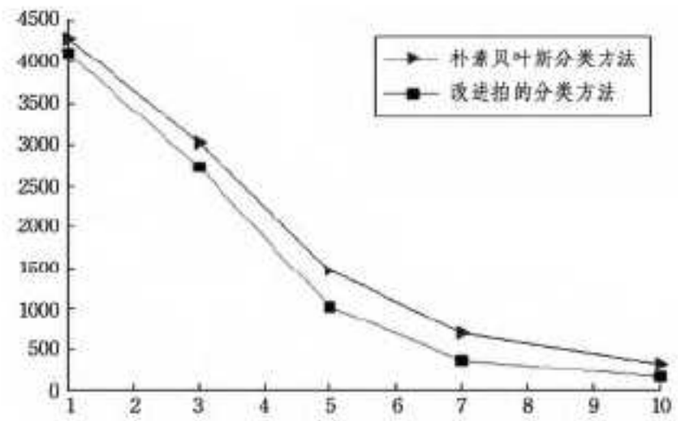

图 5　计算时间对比统计图

仔细分析实验数据可以发现，随着节点数的增加，改进后的方法相对于朴素贝叶斯方法其速度提升越发明显，这是因为通过过滤合并特征词的方法虽然可以有效地降低向量维数，加快计算的速度，但是核心词的统计也会稍微加重计算的负担。因此在单节点的情况下，两种方法的运行速度并没有太大的区别。

然而核心词的统计计算量只占总计算量的很小一部分，随着节点数的增加，该影响会逐渐变小，因此，可以看见两种方法的速度之差在逐渐扩大。直到节点数增加到一定程度使得分类的计算时间变得相当短，两者的差距才逐渐缩小。

(2) 分类效果比较

由已有研究成果[2]可知朴素贝叶斯分类方法的分类总识别度为 86.1%，这里不再赘述。改进后分类方法的分类总识别度为 91.2%。分类的结果如表 1 所列。表明了改进后的方法可以在一定程度上提高分类的精度。

表 1　改进后的分类实验结果

分类预测	汽车	体育	健康	旅游	IT	招聘	教育	文化	经济	军事	召回率/%
汽车	7843	3	10	21	43	2	14	7	54	3	98.0
体育	7	7723	15	29	37	56	28	80	11	8	96.5
健康	27	16	7185	77	124	56	325	117	6	67	89.8
旅游	31	24	19	7417	66	78	68	147	114	36	92.7
IT	5	12	77	147	7389	54	214	4	76	22	92.4
招聘	7	5	141	22	76	6997	445	128	142	37	87.5
教育	2	15	263	7	101	24	7551	9	14	14	94.4
文化	44	16	347	112	76	54	976	6123	185	67	76.5
经济	201	3	17	137	303	38	179	44	7012	66	87.7
军事	18	5	14	28	45	22	54	93	24	7697	96.2
分类精度/%	95.8	98.7	88.8	92.8	89.5	94.8	76.6	90.7	91.8	96.0	

结束语　本文算法在朴素贝叶斯分类算法的基础上根据分布式计算的特点进行了改进，并部署在 Hadoop 云计算平台上，进行了相应的测试和完善。实验表明：相对于朴素贝叶斯方法，改进后的方法可以有效地提高分类算法在应对海量数据时的准确性和运算速度。

参 考 文 献

[1] Jing Y S, Pavlovic V, Rehg J M. Boosted Bayesian network classifiers[J]. Machine Learning, 2008, 73(2): 155-184

[2] Webb G I, Boughton J R, Zheng F, et al. Learning by extrapolation from marginal to full-multivariate probability distributions: Decreasingly naive Bayesian classification[J]. Machine Learning, 2012, 86(2): 233-272

[3] Tillman R E. Structure learning with independent non-identically distributed data[C]// Proceedings of the 26th Annual International Conference on Machine Learning. New York, 2009: 1041-1048

[4] Su J, Zhang H, Ling C X, et al. Discriminative parameter learning for Bayesian networks[C]// Proceedings of the 25th International Conference on Machine Learning(ICML 2008). Helsinki, Finland, 2008: 1014-1023

[5] Ekanayake J, Li H, Zhang B, et al. Twister: A runtime for iterative MapReduce[C]// Proceedings of the 19th ACM International Symposium onHigh Performance Distributed Computing. Chicago, Illinois, USA, 2010: 810-818

[6] Dean J, Ghemawat S. Mapreduce: Smiplified data processingon large clusters[C]// Proceedings of the 6th Symposium onOperating System Design and Implementation. SanFrancisco, California, USA: USENIX Association, 2004: 137-150

[7] Thusoo A, Sarma J S, Jain N, et al. Hive: A warehousing solution over a map-reduce framework[C]// Proceedings of the Conference on Very Large Databases (VLDB. 09). Lyon, France, 2009: 1626-1629

[8] Dean J, Ghemawat S. Map/Reduce advantages overparallel databases include storage-systemindependenceand fine-grain fault tolerance for large jobs[J]. Communi-cations of the ACM, 2010, 53(1): 72-77

[9] Dittrich J, Quiane-Ruiz J-A, Jindal A, et al. Hadoop++: Making a yellow elephant run like a cheetah(without it evennoticing)[J]. Proceedings of the VLDB Endowment, 2010, 3(1): 518-529

[10] Bu Y, Howe B, Balazinska M, et al. HaLoop: Efficient itera-tive data processing on large clusters[J]. Proceedings of theVLDB Endowment, 2010, 3(1): 285-296