基于Hadoop平台的海量文本分类的并行化

向小军 高 阳 商 琳 杨育彬
(南京大学计算机科学与技术系 南京 210093)

摘 要 文本分类是信息检索与数据挖掘的研究热点与核心技术,近年来得到了广泛的关注和快速的发展。近来年随着文本数据呈指数增长,要有效地管理这些数据,就必须在分布式环境下用有效的算法来处理这些数据。在Hadoop分布式平台下实现了一简单有效的文本分类算法——TFIDF分类算法,即一种基于向量空间模型的分类算法,它用余弦相似度得到分类结果。在两个数据集上做了实验,结果表明,这一并行化算法在大数据集上很有效并可以在实际领域中得到良好的应用。

关键词 文本分类,并行化,海量数据,Hadoop
中图法分类号 N532 **文献标识码** A

Parallel Text Categorization of Massive Text Based on Hadoop

XIANG Xiao-jun GAO Yang SHANG Lin YANG Yu-bin
(Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

Abstract In recent years, there have been extensive studies and rapid progresses in automatic text categorization, which is one of the hotspots and key techniques in the information retrieval and data mining field. In recent years, as the text data grows exponentially, to effectively manage the large storage of data, we must use efficient algorithm to process it in the distributed environment. In this paper, we implemented a simple and effective text categorization algorithm on hadoop--TFIDF classifier, an algorithm based on vector space model, cosine similarity was applied as the metrics. The experiments on two datasets show that the parallel algorithm is effective on large storage of data and can be applied in practical application field.

Keywords Text categorization, Parallelization, Massive data, Hadoop

1 引言

文本分类(Text Categorization)是指依据文本的内容,由计算机根据某种自动文本分类算法,把文本判断为预先定义好的类别[1]。文本分类是信息存储和信息检索中的重要内容。特别是近年来互联网的飞速发展又给文本分类提供了新的应用平台,这些海量数据为如何有效地进行文本分类带来了巨大的挑战。

近几十年来,一系列的统计学习文本分类方法被提出[3],一些经典的文本分类算法包括Rocchio[4]算法、朴素贝叶斯分类[5]、K-近邻算法(KNN)[6]、决策树分类[7]、支持向量机(SVM)方法[8]。

一般而言,中文文本分类方法可分为两种类型[20]:

第一种类型是基于外延方法的分类方法:它不关心文本的语义,仅仅根据文本的外在特征进行分类。例如上面提到的各种统计方法,它应用了机器学习理论。目前国内的中文文本分类都倾向于这种方法[10]。最常见的方法是基于向量空间模型(Vector Space Module)的方法,该方法的思想是:把文本表征成由特征项构成的向量空间中的一个点,通过计算向量之间的距离,来判定文本之间的相似程度。采用该模型的文本分类方法一般步骤是:先通过对训练语料的学习对每个类建立原型向量作为该类的表征,然后依次计算该向量和各个类的特性向量的距离,选取距离大小符合域值的类别作为该文本所属的最终类别。这种方法简单有效并得到了很多的应用。

第二种类型是基于语义的分类方法:它采用全部或部分理解文本的语义进行分类,但此方法的进一步发展受到了自然语言处理技术的制约。

各种应用领域数据的海量增长以及这些数据往往都分布在不同的地方,使得高性能计算成为了大规模数据挖掘中的一个重要部分,所以需要把并行和分布式计算融合到数据挖掘领域中来。目前对并行数据挖掘已经有很多研究,如文献[14]用的是Multi-Agent系统的技术来进行分类和关联挖掘;文献[15]是对关联规则的挖掘;文献[16]介绍了多核环境

到稿日期:2010-11-24 返修日期:2011-02-23 本文受国家自然科学基金项目(61035003,60875011),科技部国际科技合作计划项目(2010DFA11030),江苏省自然科学基金项目(BK2010054)资助。

向小军(1986-),女,硕士生,主要研究方向为数据挖掘,E-mail:xjxiang.nju@gmail.com;高 阳(1972-),男,博士生导师,主要研究方向为强化学习、智能Agent、智能应用;商 琳(1973-),女,硕士生导师,主要研究方向为人工智能、数据挖掘、粗糙集;杨育彬(1977-),男,硕士生导师,主要研究方向为信息检索、数据挖掘、机器学习。

下的数据挖掘；文献[17,18]都是介绍序列模式的并行挖掘。

目前国内外对文本分类的并行技术研究不是很多。在前几年，一些人引入网格计算来进行文本分类的并行挖掘。文献[17]用向量空间模型在4个结点的网格计算环境下进行并行分类，在这一并行模型中，不是每一个过程都能够并行化，测试数据是分块进行的，并行性体现得不够充分和灵活。

国外的并行化文本分类研究有文献[10,11,13]等；文献[10]结合了期望最大化(Expectation Maximization)算法和朴素贝叶斯算法，用于标记样本不是很多但是未标记的样本很多的数据，但是引入期望最大化算法训练那些大量的未标记文档很耗时，所以引入并行化。并行化用的是SPMD(Single Program Multiple Data)模型，即每个结点有一份源程序的拷贝，数据分布在各个处理器的局部内存里，处理器之间通过MPI(Message-Passing Interface)通信。但是这种模型缺乏灵活性，磁盘I/O效率低。Reynaldo Gil-García[13]在2007年提出了两种近邻算法的并行化，即主从机制和管道式技术。并行环境是一些简便的PC机组成的集群环境，由通信网络连在一起。但是近邻算法本身决定了它的耗时性。

以上列出的文章都有一些局限性，缺乏广泛推广和实际应用的能力，特别缺乏对海量文本数据的挖掘，其分布式平台并没有体现出什么优势。自从Hadoop平台[12]发布以来，很多企业都开始用这一平台进行海量数据的挖掘。Hadoop，一个分布式系统基础架构，由Apache基金会开发，采用MapReduce的编程模型。用户可以在不了解分布式底层细节的情况下，开发分布式程序。它充分利用集群的威力高速运算和存储。Hadoop平台的优势集中体现在：针对读写速度慢的问题，可以采用分布式存储的方式(HDFS)提高读写速度；针对硬件故障可以通过存储冗余数据的方式来解决；针对如何正确地把分布在不同存储地点的数据加以整合分析，可以通过MapReduce以键值对的形式抽象出磁盘的读写来进行整合。

本文针对一个经典的向量空间模型方法TFIDF算法——一种基于TFIDF的Rocchio算法[2]，把它并行化，我们选择这一算法的动机在于它的快速高效性，在MapReduce模型下易于实现，速度远远优于其他的算法。我们在SogouCS[22]这一大数据集上做了实验，在时间性能上取得了不错的效果，在仅有6个结点的集群环境下，能在数十秒内取得分类结果。为了比较并行性和测试算法的有效性，同时用另外一个相对较小的数据集TanCorp-12[14]做了实验。充分说明了这一算法时间性能上的优越性。

本文第2节详细介绍我们采用的TFIDF分类算法；第3节介绍对这一算法进行并行化的平台和过程；第4节利用实际的数据集做了一些分析；最后是对全文的简单总结。

2 TFIDF算法

文本分类的目标是把文档归类到一些固定的预定义的类别中。假定每一篇文档都只属于一个类别，本文所用的分类方法是1997年Thorsten Joachims[2]提出的TFIDF分类算法。这种算法是一种基于向量空间模型的算法。向量模型结构简单、便捷，与其他的模型相比较，即使向量模型不是最优的，其性能也相当好，并且这种模型易于并行化。

2.1 数据表示

在描述一个模型之前，通常都要把原始数据表示成适合于这个模型和分类任务的形式。在文本分类任务中，原始数据通常都是一篇篇完整的文档。在信息检索领域，一个词通常是一个好的表示单元，一些研究表明，它们出现的顺序通常对很多任务影响很小。所以就导出了一种在信息检索领域很广泛的表示，即bags-of-words的表示形式。

Bags-of-words的表示形式等价于机器学习中特征-值的表示。每个不同的词代表一个特征，与之相应的值是这个词在文档中出现的次数。每篇文档都表示成向量的形式。

2.2 TFIDF算法

TFIDF算法是基于相关反馈算法的向量空间检索模型。模型中有3个要考虑的因素：词的加权技术、文档长度的归一化、相似度的选择。这个算法选择的都是最通用的几种度量，其中词语的权重用的是TFIDF[9]，文档长度用的是欧式向量长度，相似度用的是余弦相似度。

首先，对每一个文档进行TFIDF向量化，然后为每一类文档C_j建立一个原型向量$\vec{c_j}$。

$$\vec{c_j}=\alpha\frac{1}{|C_j|}\sum_{\vec{d}\in C_j}\frac{\vec{d}}{\|\vec{d}\|}-\beta\frac{1}{|D-C_j|}\sum_{\vec{d}\in D-C_j}\frac{\vec{d}}{\|\vec{d}\|} \tag{1}$$

式中，α和β是用来调节正例和负例的相对影响程度的参数，$|C_j|$是类别j中的文档的个数，$\|\vec{d}\|$是$\vec{d}$的欧式长度，当$\vec{c_j}$中出现负数的时候要把它设置为0。这样做的结果是增大了只在c_j中出现的索引词的权重，减小了在类别j中出现同时也在其他类别中出现的索引词的权重，所以权值大的索引词能更好地代替类别j的特征。

这个原型向量的结果集中，每个类别一个向量，代表了学习到的模型。利用这个模型就可以对新的文档d'进行分类。首先要对这个新文档利用TFIDF权值进行向量化，用$\vec{d'}$表示；然后计算代表每一个类别的原型向量$\vec{c_j}$与$\vec{d'}$的余弦相似度，取最大的相似度值所对应的类别为d'的测试结果。

$$H_{TFIDF}(d')=\arg\max_{c_j\in C}\cos(\vec{c_j},\vec{d'})=\arg\max_{c_j\in C}\frac{\vec{c_j}\cdot\vec{d'}}{\|\vec{c_j}\|\cdot\|\vec{d'}\|} \tag{2}$$

3 Hadoop平台下海量文本分类的并行化实现

3.1 Hadoop分布式平台介绍

我们所用的并行化环境是Hadoop-0.20.2平台。Hadoop是Apache下的一个开源软件，它最早是作为一个开源搜索引擎项目Nutch的基础平台而开发的。随着项目的进展，Hadoop被作为一个单独的开源项目进行开发。Hadoop作为一个开源的软件平台使得编写和运行用于处理海量数据的应用程序更加容易。

Hadoop框架中最核心的设计就是MapReduce和HDFS。HDFS是Hadoop分布式文件系统Hadoop Distributed File System的缩写，为分布式计算存储提供了底层支持。HDFS和现有的分布式文件系统有很多共同点。但同时，它和其他的分布式文件系统的区别也是很明显的。HDFS是一个高度容错性的系统，适合部署在廉价的机器上。HDFS能提供高吞吐量的数据访问，非常适合大规模数据集上的应用，适合部署在廉价的机器上。HDFS能提供高吞吐量的数据访问，非常适合大规模数据集上的应用。

MapReduce是一种简化的分布式编程模式，可使程序自动分布到一个由普通机器组成的超大集群上并发执行。就如

同 java 程序员可以不考虑内存泄露一样，MapReduce 的 runtime 系统会解决输入数据的分布细节，跨越机器集群的程序执行调度，处理机器的失效，并且管理机器之间的通讯请求。Map 就是将一个任务分解成为多个任务，reduce 就是将分解后多任务处理的结果汇总起来，得出最后的分析结果。在分布式系统中，机器集群可以看作硬件资源池，它将并行的任务拆分，然后交由每一个空闲机器资源去处理，能够极大地提高计算效率。同时这种资源无关性对计算集群的扩展无疑提供了最好的设计保证。任务分解处理以后，就需要将处理以后的结果再汇总起来，这就是 reduce 要做的工作。图 1 展示了 MapReduce 的工作模式，map 负责分解任务，reduce 负责将分解的任务进行合并[23]。Hadoop 并行环境取决于文件的大小和数量、处理的复杂度以及群集机器的数量、相连的带宽，当以上四者并不大时，Hadoop 优势并不明显。Hadoop 适合于处理单机无法处理的海量数据集。

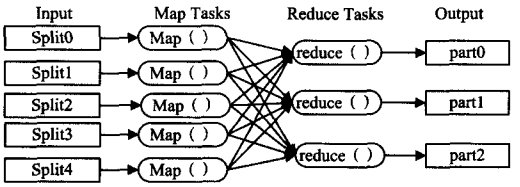


图 1　Mapreduce 工作模式

我们的分布式环境包含 6 个结点(node-1，node-2，node-3，node-4，node-5，node-6)，操作系统为 Ubuntu10. 04，内存 4G，8 核的处理器 Intel(R) Core™ i7 CPU 860@2. 80GHz。1 个 namenode，5 个 datanode。集群环境的配置如下：hdfs 的配置容量为 2. 41TB。6 个结点的 HDFS 配置如表 1 所列。堆空间为 888. 94MB。

表 1　HDFS 配置容量

	node-1	node-2	node-3	node-4	node-5	node-6
HDFS capacity	418. 12GB	418. 12GB	408. 95GB	408. 95GB	408. 95GB	408. 95GB
Map task capacity	0	3	3	3	3	3
Reduce task capacity	0	3	3	3	3	3

3.2　MapReduce 并行化实现过程

首先对文档进行预处理，文件均处理成 Hadoop 里定义的顺序文件(sequence file)，这是为了读取方便，第一步进行分词；第二步把每一行处理成〈文档 id＋"\t"＋标记，文档内容〉的形式；第三步进行 tfidf 向量化预处理，每个文档对应的向量借鉴了 mahout 开源项目[23]里面的 VectorWritable 数据类型，这大大方便了处理和节约内存。计算 tfidf 的过程如图 2 所示。

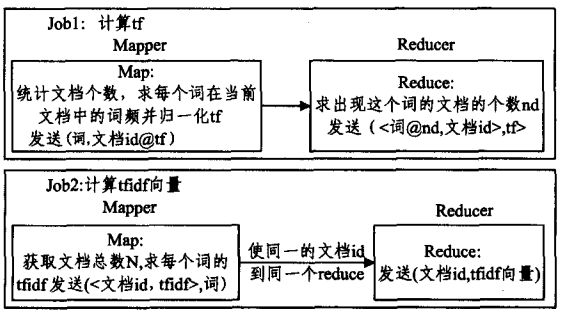


图 2　tfidf 向量化的 MapReduce 过程

用 MapReduce 编程模式并行化 IFIDF 分类算法时，在训练阶段和测试阶段分别需要一个 MapReduce 任务。在训练阶段，map 并行地发出每一个向量，key 设置为类标，则同一个类的向量会自动发送到同一个 reduce 进行累加，在 reduce 执行完毕后有一个 cleanup 函数，可以由 reduce 计算的结果计算出每一类文档的原型向量。在测试阶段，首先在 Map 的 map 执行之前的 setup 函数里读取训练模型，在 map 函数里面，读取每一个测试文档的向量，计算它与以上模型中的原型向量的相似度，得到测试结果。整个过程如图 3 所示。算法伪代码如图 4 所示。

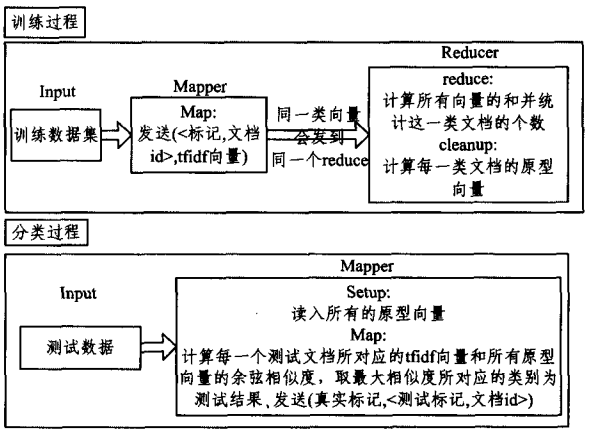


图 3　Map/Reduce 化的训练和测试过程

```
class Mapper
  method map(〈label,docid〉,tfidfVector)
    write(〈label,docid〉,tfidfVector);
class Reducer
  Matrix matrix;//每一行对应一个类别的向量和
  int i=0;
  String[] labels;
  Vector totalVec;//记录所有训练文档的向量和
  int fileNum[]={0};//记录每一类文档的总数
  int totalFile=0;
  method reduce(label,[〈docid,tfidfVector〉,…]
    label[i]=label;
    for (val:values)
      totalVec=totalVec+val;
      matrix[i]=matrix[i]. plus(val);
      fileNum[i]++;
      totalFile++;
      i++;
    method cleanup()
    Vector vector;
      for(int i=0;i<matrix. rows;i++)
  vector=ALPHA * matrix[i]/fileNum[i]－BETA * (totalVec－
matrix[i])/(totalFile－fileNum);
        write(label[i],vector);
```

(a)训练过程

```
class Mapper
  Matrix matrix;//用来存储训练模型
  String[] labels;
  method setup()
    int i=0;
```

```
SequenceFile. Reader reader=new SequenceFile.
Reader("trainingModel");
while(reader. next(key. value)
  matrix. assignRow(i,value);
  labels[i]=key;
method map((label,docid),tfidfVector)
double max=0;//最大余弦相似度
String testLabel;//测试类标
for(int i=0;i<ma. numRows;i++)
  double cos=cos-sim(tfidfVector,ma. getRow(i);
  if(max<cos)
    max=cos;
    testLabel=labels[i];
  write(label,(testLabel,docid));
```

(b)测试过程

图 4　训练过程和测试过程伪代码

4　实验及其结果

我们把这个基于 Hadoop 平台下的 TFIDF 分类并行化算法在两个中文数据集上做了实验。第一个数据集是 TanCorp-12[21]，即谭松波等人搜集的 14150 篇文档，29.5M，包含 12 个类别，其分布如表 2 所列；第二个数据集是 SogouCS[22]，来自搜狐新闻 08 年 1 月－6 月期间奥运、体育、IT、国内、国际等 18 个频道，提供 URL 和正文信息，3.33G，包含 15 个类别，2101582 篇文档，其分布如表 3 所列。

表 2　TanCorp-12 数据集分布

类别	财经	地域	电脑	房产	教育	科技
文档数	819	150	2943	935	808	1040
类别	汽车	人才	体育	卫生	艺术	娱乐
文档数	590	608	2805	1406	546	1500

表 3　SogouCS 数据集分布

类别	2008	auto	business	career	cul
文档数	98074	32731	344559	94	22281
类别	health	house	it	learning	mil. news
文档数	32518	216819	65421	31529	14683
类别	news	sports	travel	women	yule
文档数	466825	446681	43634	87319	198414

我们在单机和分布式环境下都做了实验，训练数据都是一样的，抽出其中的 90%作为训练集，剩下的 10%作为测试集。在 Hadoop 平台下的实验从 1 个结点逐一增加到 5 个结点。在单机算法上，这两个数据集的运行时间如表 4 所列。

表 4　单机算法上的运行时间

数据集	训练时间	测试时间
TanCorp	10s	4s
SogouCS	1125s	110s

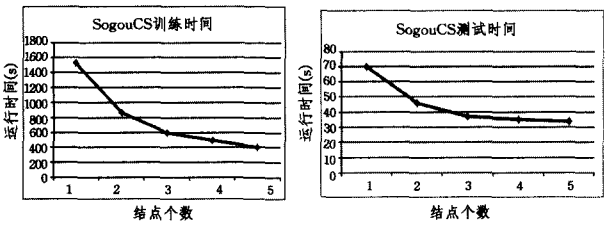
在 Hadoop 平台上实现并行化以后，在 5 个结点上这两个数据集的运行时间如表 5 所列。

表 5　完全分布式环境下的运行时间

数据集	训练时间	测试时间
TanCorp	28s	22s
SogouCS	403s	34s

从表 5 中可见，对于小数据集在分布式环境下面取得的效果不好，比单机更耗时，小数据集效果不好的原因是：Hadoop 对数据是分块处理的，默认是 64M 分为一个数据块，如果存在大量小数据文件(例如 2～3M 一个的文件)，这样的小数据文件远远不到一个数据块的大小就要按一个数据块来进行处理。这样处理带来的后果有两个：1)存储大量小文件占据存储空间，致使存储效率不高检索速度也比大文件慢。2)在进行 MapReduce 运算时这样的小文件消费计算能力，默认是按块来分配 Map 任务的。所以我们的算法就是针对单机上无法处理的大数据集而设计的，文件的大小要超过默认数据块的大小，才会有优势体现出来。下面给出更多关于 SogouCS 数据集运行的结果。

图 5(a)和图 5(b)分别显示了 SogouCS 的训练时间和测试时间曲线图，图 6(a)和图 6(b)分别显示了 SogouCS 的训练时间和测试时间的加速比。运行环境中的结点从 1 逐渐增至 5。



(a)训练时间　　(b)测试时间

图 5　并行算法的运行时间

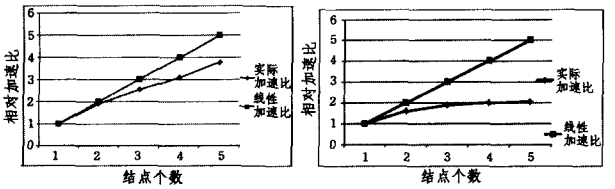


图 6　与图 5 对应的加速比曲线

为了验证算法的高效性，我们对比了 KNN 分类的效果，在 MapReduce 化的 KNN 算法中，我们用同样的 SogouCS 数据作为训练集，4.4M 作为测试数据，运行时间如图 7 所示。可以看出，KNN 算法的效率远远不如 TFIDF 分类算法，KNN 算法的运行时间远远长于 TFIDF 分类算法，几乎是 TFIDF 的 10 倍。

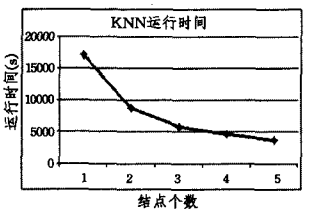


图 7　KNN 算法的运行时间

经观察和对比，可以发现 TFIDF 算法的高效性。在仅有 6 个结点的集群环境下，用大小适量的数据做实验，能够较快地得出实验结果，训练时间 411s，测试时间 35s。

实验表明，数据块的实际大小必须超过 HDFS 的默认块大小，才能呈现出较好的加速比。并且数据块的个数最好与分布式环境能同时处理的块的个数相当，才能达到最好的效果。必须要同时满足数据块大于 64M 和数据块的个数大于

集群环境能同时处理的块数，并行性才会表现得比较好。

结束语 我们在 Hadoop 这一并行环境运行下实现了一种经典的基于向量空间模型的文本分类算法——TFIDF 分类算法。这种并行化的结果在与集群环境能处理的相对合适的数据集上取得了不错的效果，与理想的线性加速比相当，能够很快地训练出模型和取得分类结果。实验表明，Hadoop 环境在大数据集上的分类远远优于单机算法，这一结果正好弥补了在单机上无法完成的海量数据的挖掘。

下一步我们将从文本分类的角度探索更加实际的应用，情感分析是文本分类的一种特殊应用，并且有一定的挑战性，我们将从这个方面出发，找到更加合理的应用。另外，海量数据集的处理是我们研究的重点，未来我们立志于把海量数据处理做成一种服务，以满足广大用户的需求。

参 考 文 献

[1] Sebastiani F. Text Categorization[Z]. Encyclopedia of Database Technologies and Applications. 2005:683-687

[2] Joachims T. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization[C]// Proceedings of the Fourteenth International Conference on Machine Learning. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1997

[3] Yang Y. An Evaluation of Statistical Approaches to Text Categorization[J]. Journal of Information Retrieval, 1999, 1(1/2): 67-88

[4] Rocchio J J Jr. Relevance Feedback in Information Retrieval [M]. Salton G, ed. The SMART Retrieval System: Experiments in Automatic Document Processing. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971: 313-323

[5] Tzeras K, Hartmann S. Automatic Indexing Based on Bayesian Inference Networks[C]// Proc. 16th ACM Int. SIGIR Conference. 1993: 22-34

[6] Masand B, Lino G, Waltz D. Classifying News Stories Using Memory Based Reasoning[C]// 15th ACM SIGIR Conference. 1992: 59-65

[7] Apte C, Damerau F, Weiss S. Automated Learning of Decision Rules for Text Categorization[J]. ACM Trans. on Information Systems, 1994, 12(3): 233-251

[8] Joachims T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features[C]// Proc. 10th European Conference on Machine Learning (ECML). 1998: 137-142

[9] Salton G, Buckley C. Term Weighting Approaches in Automatic Text Retrieval[J]. Information Processing and Management, 1988, 24(5): 513-523

[10] Kruengkrai C, Jaruskulchai C. A Parallel Learning Algorithm for Text Classification[C]// Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. 2002: 201-206

[11] Lertnattee V, Theeramunkong T. Parallel Text Categorization for Multi-dimensional Data[C]// K. M. Liew, et al., eds. PDCAT 2004, LNCS 3320. 2004: 38-41

[12] Hadoop T W. The Definitive Guide[M]. YAHOO! Press, 2009

[13] Gil-García R, Badía-Contelles J M, Pons-Porrata A. Parallel Nearest Neighbour algorithms for Text Categorization[C]// EURO-PAR 2007 Parallel Processing, Lecture Notes in Computer Science. 2007: 328-337

[14] Khan D. CAKE-Classifying, Associating & Knowledge Discovery An Approach for Distributed Data Mining (DDM) Using PArallel Data Mining Agents (PADMAs)[Z]. Web Intelligence and Intelligent Agent Technology, 2008

[15] Paul S, Saravanan D V. Hash Partitioned Apriori in Parallel and Distributed Data Mining Environment with Dynamic Data Allocation Approach[C]// ICCSIT '08 Proceedings of the 2008 International Conference on Computer Science and Information Technology. 2008

[16] Qiu Xiao-hong, Fox G, Yuan Hua-peng, et al. Parallel Data Mining on Multicore Clusters[C]// Seventh International Conference on GCC'08. 2008: 41-49

[17] Wang Jin-lin, Chen Xi, Zhou Ke-fa, et al. Parallel Research of Sequential Pattern Data Mining Algorithm[C]// 2008 International Conference on Computer Science and Software Engineering. 2008

[18] Wang Jin-lin, Chen Xi, Zhou Ke-fa. Research on a Scalable Parallel Data Mining Algorithm[C]// 2009 Fifth International Joint Conference on INC, IMS and IDC. 2009: 888-893

[19] 王鄂，李铭. 云计算下的海量数据挖掘研究[J]. 现代计算机，2009

[20] 高洁，吉根林. 文本分类技术研究[J]. 计算机应用研究，2004

[21] 谭松波，王月粉. 中文文本分类语料库-TanCorpV1.0[OL]. http://www.searchforum.org.cn/tansongbo/corpus.htm

[22] 搜狐新闻数据(SogouCS)[OL]. http://www.sogou.com/labs/dl/cs.html

[23] http://mahout.apache.org/

(上接第 173 页)

[22] Cuff J A, Barton G J. Evaluation and improvement of multiple sequence methods for protein secondary structure prediction [J]. Proteins, 1999, 34(4): 508-19

[23] Cuff J, Clamp M, Siddiqui A, et al. JPRED: A consensus secondary structure prediction server[J]. Bioinformatics, 1998, 14: 892-893

[24] Montgomerie S, Cruz J A, Shrivastava S, et al. Proteus2: a web server for comprehensive protein structure prediction and structure-based annotation[J]. Nucleic Acids Res, 2008, 36: 202-209

[25] http://bioinf.cs.ucl.ac.uk/psipred/psiform.html

[26] http://www.compbio.dundee.ac.uk/www-jpred/

[27] http://cubic.bioc.columbia.edu/predictprotein/

[28] http://compbio.soe.ucsc.edu/SAM_T08/T08-query.html

[29] http://wks16338.biology.ualberta.ca/proteus2/