CN 43-1258/TP
ISSN 1007-130X

计算机工程与科学
Computer Engineering & Science

第 36 卷第 6 期 2014 年 6 月
Vol. 36, No. 6, Jun. 2014

文章编号:1007-130X(2014)06-1018-05

# 基于 Hadoop 平台的 TFIDF 算法并行化研究*

王静宇[1,2],赵伟燕[2]

(1.北京科技大学计算机与通信工程学院,北京 100083;2.内蒙古科技大学信息工程学院,内蒙古 包头 014010)

**摘　要**:针对大数据集下文本分类算法在单机上训练和测试过程效率低下的问题,提出了基于 Hadoop 分布式平台的 TFIDF 文本分类算法,并给出了算法实现的具体流程。通过 MapReduce 编程模型实现了考虑到词在文档中位置的并行化 TFIDF 文本分类算法,并与传统串行算法进行了对比,同时在单机和集群模式下进行了实验。实验表明,使用并行化的 TFIDF 文本分类算法可实现对海量数据的高速有效分类,并使算法性能得到优化。

**关键词**:文本分类;MapReduce;并行化;TFIDF 算法

**中图分类号**:TP391.1　　**文献标志码**:A

**doi**:10.3969/j.issn.1007-130X.2014.06.004

# Research on parallelizing the TFIDF algorithm based on Hadoop

WANG Jing-yu[1,2], ZHAO Wei-yan[2]

(1. School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083;
2. College of Information Engineering, Inner Mongolia University of Science and Technology, Baotou 014010, China)

**Abstract**: Aiming to improve the efficiency of text classification algorithm on a large data set during the training and testing process, the TFIDF text classification algorithm based on the Hadoop distribution platform is proposed, and its implementation process is given. By using the MapReduce programming model, the parallelized TFIDF text classification algorithm is implemented, which takes the word locations into consideration. Comparative experiments are conducted between the improved TFIDF algorithm and the traditional serial algorithm in both the standalone mode and the cluster mode. The experimental results show that the improved TFIDF text classification algorithm can achieve high-speed mass data classification and optimize performance.

**Key words**: text classification; MapReduce; parallelization; TFIDF algorithm

## 1　引言

随着 Internet 等技术的飞速发展,信息时代数据膨胀的速度远远超过了人工分析的能力,信息处理已经成为人们获取有用信息不可或缺的工具,文本分类技术对实现大数据集信息的有效挖掘至关重要。

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]。文本分类是数据挖掘的关键技术,为了提高分类效率满足人们需求,所以要实现算法并行化。

20 世纪 50 年代,文本挖掘领域的 Luhn H P 专家首先将词频统计的思想用于文本分类,开创了国外对该领域的研究。文本分类经历了人工分类

* **收稿日期**:2012-12-22;**修回日期**:2013-02-25
**基金项目**:国家自然科学基金资助项目(61163025);内蒙古自然科学基金资助项目(2012MS0912);内蒙古教育厅科研资助项目(Njzy12110)
**通信地址**:014010 内蒙古包头市阿尔丁大街 7 号内蒙古科技大学信息与网络中心
**Address**: Center of Information and Network, Inner Mongolia University of Science and Technology, 7 Aerding St, Baotou 014010, Inner Mongolia, P. R. China

时期,20 世纪 90 年代以来进入了基于机器学习的时代。近几十年来,提出了一系列统计学习文本分类方法[2]。国内对文本分类也有很多研究,并构建了语料库。现流行的文本分类算法主要有 $K$ 邻近、神经网络、朴素贝叶斯、支持向量机 SVM 和词频反文档频率 TFIDF(Term Frequency-Inverse Document Frequency)等。开源并行云计算平台 Hadoop 发布后,许多个人及企业开始使用这一平台进行海量数据的挖掘。目前山西财经大学卫洁等人已经实现了"基于 Hadoop 的分布式朴素贝叶斯文本分类",此算法中贝叶斯属性条件的常用计算方法是 TFIDF 算法,未改进 TFIDF 本身对大数据计算的低效性且未考虑词位置信息,增加了算法时间复杂度的同时降低了分类性能。北京交通大学的乔鸿欣实现了基于 MapReduce 的 KNN 分类算法,此算法是懒散的分类算法,每一类文本都要计算它和已知全体文本的相似度,相似度计算量过大,导致了分类的低效性。本文选择可实现任务分割、分类准确度较高且过程简单的 TFIDF 分类算法,同时考虑到词的位置对权重计算的影响,以开源云计算平台 Hadoop 框架为基准,在单机模式和集群模式下研究 TFIDF 算法的并行化并进行实验。实验结果表明,通过考虑词的位置对权重的影响和改变算法的编程模型,实现了算法性能的优化,提高了海量数据的分类效率。

## 2 TFIDF 算法原理

TFIDF 是一种用于资讯检索与资讯探勘的常用加权技术。在一给定的文档中,词频(TF)是指某一具体给定的词语在这个文档中出现的次数。对于在某一特定文档里的词语 $t_i$,其词频可以表示为:

$$tf_{i,j} = \frac{m_{i,j}}{\sum_k m_{k,j}} \tag{1}$$

其中,$m_{i,j}$ 是该词在文档 $d_j$ 中出现的次数,分母 $\sum_k m_{k,j}$ 是在文档 $d_j$ 中所有字词出现次数之和。为了防止它偏向长文件,这个数一般需要进行归一化。

逆向文件频率(IDF)即对一个词语普遍重要性的度量。某一特定词语的 IDF 值,可由总文件数除以包含该词语的文件的数目,再将二者之商取对数得到,公式表示如下 :

$$idf_i = \log \frac{|M|}{|\{j:t_i \in d_j\}|} \tag{2}$$

其中,$|M|$ 表示的是语料库中文件总的数目,分母 $|\{j:t_i \in d_j\}|$ 表示包含词语 $t_i$ 的文件数目。

由式(1)和式(2)可得到单词的权重公式为[3]:

$$tfidf_{i,j} = tf_{i,j} \times idf_i \tag{3}$$

考虑到实际情况中词的位置不同,对文章类别的判断的贡献不同,根据以下信息对公式做出修正。设 $L_i$ 中存放的是某一词在文本中的位置信息;$\sum_{i,j} L_i$ 为文档总长度;$R$ 为加权系数,我们根据词的不同位置设定的加权系数如表 1 所示。

**Table 1 Weighted coefficient setting**

**表 1 加权系数的设定**

| 位置 | $\frac{L_i}{\sum_{i,j} L_i}$ | $R$ |
|---|---|---|
| 标题、关键词 | 0~0.05 | 0.28 |
| 摘要 | 0.05~0.2 | 0.26 |
| 首段 | 0.2~0.3 | 0.18 |
| 中间内容 | 0.3~0.8 | 0.10 |
| 尾段 | 0.8~1.0 | 0.18 |

则得到考虑词的位置的权重计算的修正公式为:

$$tfidf^*_{i,j} = R \times tf_{i,j} \times idf_i \tag{4}$$

当某一词语在特定文档中出现次数比较多,且该词语在整个文档集合中出现次数相对少时,就得到了高的 TFIDF 值。即 TFIDF 的计算倾向于过滤掉常见词语,保留相对来说比较重要的词语。

TFIDF 算法用特征项组成的向量来表示文档,同一类所有文档向量的组合便得到此类文档的特征向量 $\boldsymbol{c}_j$ 。利用本算法测试一篇文档所属的类别,是通过计算此文档的向量与各类特征向量的相似度距离 $sim(\boldsymbol{d}_i, \boldsymbol{c}_j)$,并将文档归属于相似度距离最大的类向量所属的类。TFIDF 算法原理中主要考虑三个要素:词的加权技术、文档长度归一化、相似度的选择。本算法选择的都是最常用的几种度量,其中词语的权重用的是 TFIDF,文档长度的归一用的是欧氏向量长度,相似度用的是余弦相似度[4]。

文档 TFIDF 向量化后,建立每一类文档的原形向量 $\boldsymbol{c}_j$,如下式:

$$\boldsymbol{c}_j = \alpha \frac{1}{|\boldsymbol{c}_j|} \sum_{\boldsymbol{d} \in \boldsymbol{c}_j} \frac{\boldsymbol{d}}{\|\boldsymbol{d}\|} - \beta \frac{1}{|\boldsymbol{D} - |\boldsymbol{c}_j||} \sum_{\boldsymbol{d} \in \boldsymbol{D} - |\boldsymbol{c}_j|} \frac{\boldsymbol{d}}{\|\boldsymbol{d}\|} \tag{5}$$

其中,$\alpha$ 和 $\beta$ 是调节正例、负例相对影响程度的参数,$|\boldsymbol{c}_j|$ 是属于类别 j 的文档的个数,$\|\boldsymbol{d}\|$ 为 $\boldsymbol{d}$ 的欧氏长度,$D$ 为文档总数。

学习到的模型是由一个个代表各个类的原型向量组成的。利用此模型就可以对新文档 $d'$ 进行分类。首先要对新文档 $d'$ 利用 TFIDF 权值进行向量化,用 $\boldsymbol{d}'$ 表示;$\boldsymbol{c}$ 表示原型向量集合,然后分别计算代表每一个类别的原型向量 $\boldsymbol{c}_j$ 与 $\boldsymbol{d}'$ 的余弦相似度;最后取最大的相似度值所对应的类别为 $\boldsymbol{d}'$ 的归属类别,即测试结果[7]。其中余弦相似度计算公式如下所示:

$$H_{\mathrm{TFIDF}}(\boldsymbol{d}') = \arg\max_{\boldsymbol{c}_j\in\boldsymbol{c}}\cos(\boldsymbol{c}_j,\boldsymbol{d}) = \arg\max_{\boldsymbol{c}_j\in\boldsymbol{c}}\frac{\boldsymbol{c}_j\boldsymbol{d}'}{\|\boldsymbol{c}_j\|\cdot\|\boldsymbol{d}'\|} \quad (6)$$

TFIDF 算法是有监督的文本分类算法,它的训练集是已标记的文档,它对训练集规模很敏感,随着训练集规模的增大,分类精度显著提高[5]。

# 3 Hadoop 框架下的 TFIDF 算法

## 3.1 Hadoop 分布式并行计算平台

Hadoop 是一个能够对大数据进行分布式处理的框架,实现了 Google 的 MapReduce 编程模型和架构,能够把应用程序分割成许多小的工作单元,并且把这些单元放到任何集群节点上执行[6]。MapReduce 模型的计算流程如图 1 所示。

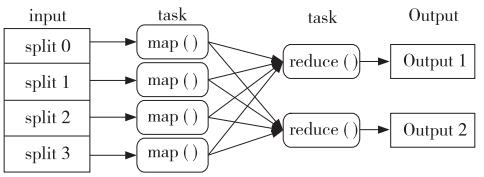



Figure 1 Calculation model of MapReduce

图 1 MapReduce 计算模型

Hadoop 主要由分布式文件系统(HDFS)和 MapReduce 编程模型构成。分布式文件系统主要负责各节点上的数据的存储,并实现了高吞吐的数据读写。MapReduce 计算模型主要由 Map 和 Reduce 两个函数组成,它们是由用户自定义实现的,其功能是按一定的映射规则将输入的〈$key$,$value$〉对转换成另一个或一批〈$key$,$value$〉对输出[7]。其中 HDFS 和 MapReduce 的关系如图 2 所示。

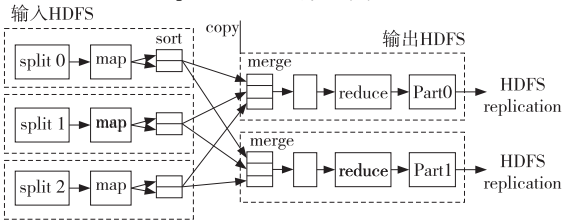



Figure 2 Relationship between HDFS and MapReduce

图 2 HDFS 和 MapReduce 的关系图

## 3.2 MapReduce 下 TFIDF 算法的处理流程

Hadoop 分布式计算的核心思想就是任务的分割及并行运行。从 TFIDF 的计算公式可看出,它非常适合分布式计算求解。单词词频 TF 只与它所在文档的单词总数及它在此文档中出现的次数有关。因此,可以通过分割数据,并行统计文档中的单词词频 TF,加快计算速度。得到单词词频 TF 后,单词权重 TFIDF 的计算取决于包含此单词的文档个数(因为文档总数是一个常量)。因此,只要能确定包含此单词的文档个数,即能以并行计算的方式实现 TFIDF 的求解。改进的计算 TFIDF 处理流程如图 3 所示,主要包括以下三步:

**步骤 1** 统计每份文档中单词的出现次数并记录单词位置信息;

**步骤 2** 统计文档单词词频 TF 和各个文档长度;

**步骤 3** 计算单词的 TFIDF 值:$tfidf^* = R\times n/N\times\log(D/d)$,其中 $R$ 为加权系数,$N$ 为一个文档总词数,$n$ 为在一个文档中出现特定词语的次数,$D$ 为文档集内文档数,$d$ 为出现某特定词的文档数目。

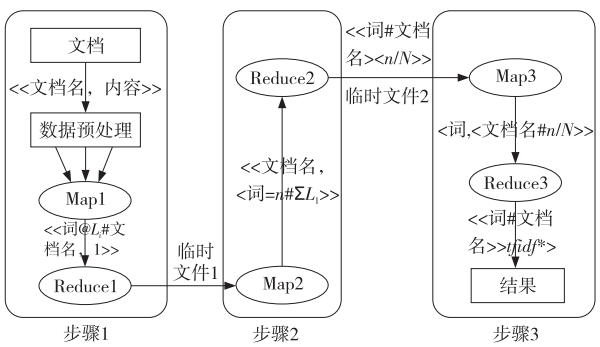



Figure 3 Processing flow of TFIDF algorithm by MapReduce

图 3 MapReduce 下 TFIDF 算法的处理流程

## 3.3 实验与分析

实验选择 Sogou 文本分类语料库[8]作为数据集,并进行以下实验。文本分类语料库来源于 Sohu 新闻网站保存的大量经过编辑手工整理与分类的新闻语料与对应的分类信息。其分类体系包括几十个分类节点,主要包括以下几类:汽车、财经、IT、健康、体育、旅游、教育、招聘、文化、军事。网页规模约为十万篇文档。分布式环境包括五个节点,一个 Master,四个 Slave。操作系统为 Ubuntu 12.04,内存 4 GB。集群环境的配置如下:HDFS 的配置容量为 1.72 TB,五个节点的 HDFS 配置如表 2 所示。

**Table 2 Configuration capacity of HDFS**

**表 2 HDFS 配置容量**

| | HDFS 容量/GB | Map 任务数 | Reduce 任务数 |
|---|---|---|---|
| Master | 318.75 | 0 | 0 |
| Slave1 | 302.75 | 2 | 2 |
| Slave2 | 302.75 | 2 | 2 |
| Slave3 | 302.75 | 2 | 2 |
| Slave4 | 302.75 | 2 | 2 |

(1) 单机环境下 TFIDF 算法对比。

考虑词的位置并且使用 MapReduce 编程的 TFIDF 算法和传统 TFIDF 算法进行对比实验,实验结果如图 4 所示。

由图 4 可以看出,当数据量较小时,MapReduce 下 TFIDF 算法和传统 TFIDF 算法性能差距并不明显。由于 Hadoop 对数据进行的是分块处理,并且默认数据块大小为 64 MB,所以当存在很多小数据文件时,运行速度反而降低了,因此对于小数据集 Hadoop 的优越性体现得并不明显。但是,随着数据集增大,传统算法所需要的时间急剧增长,应用了 Hadoop 框架的 TFIDF 算法所需要的时间只是呈线性增加,表现出了一定的优越性。

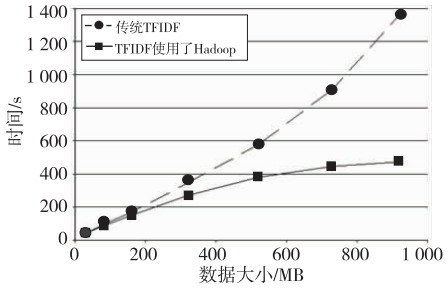



Figure 4 Contrast between the improved TFIDF algorithm and traditional TFIDF algorithm

图 4 MapReduce 下的 TFIDF 算法和传统 TFIDF 算法的对比

(2)单机和分布式环境下的对比。

取数据集的 80%作为训练集,其余的 20%作为测试数据集进行实验,结果如表 3 所示。

**Table 3 Running time in single mode**

**表 3 单机模式下运行时间** s

| 数据集 | 训练时间 | 测试时间 |
|---|---|---|
| 文本分类语料库 | 1 225 | 242 |

Hadoop 下实现 MapReduce 编程,在五个节点上 Sogou 文本分类语料库数据集的运行时间如表 4 所示。

(3)不同节点数下的对应运行时间。

图 5a 和图 5b 分别显示了 Sogou 文本分类语料库,随着节点数目由 1 增加到 5 时的训练时间和测试时间曲线。

**Table 4 Running time in cluster environment**

**表 4 集群环境下运行时间** s

| 数据集 | 训练时间 | 测试时间 |
|---|---|---|
| 文本分类语料库 | 423 s | 76 s |

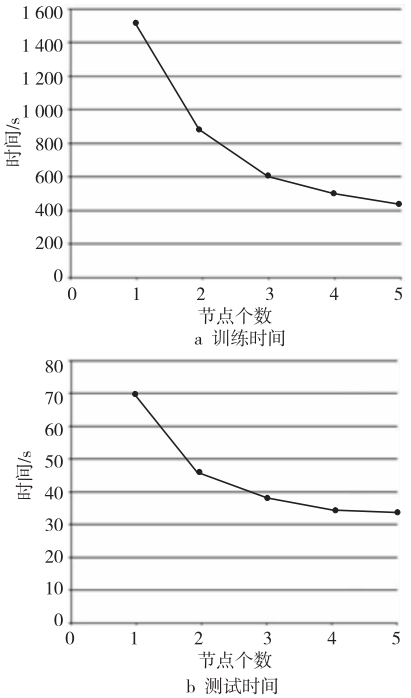



Figure 5 Training time and testing time

图 5 训练时间与测试时间

(4)不同数据量对应的加速比和时间复杂度分析。

由实验数据可以看到,加速比[9]如表 5 所示:当数据量较小(<200 MB)时,加速比并不理想,这说明了主节点任务分配、资源调度负载均衡等机器之间的通信消耗的时间比分布式计算带来的增速还要大;但是,随着数据的增大,集群环境运行效率表现出了很大的优势,加速比有了显著提升,并且接近 4,符合用于衡量并行系统的标准-效率(efficiency),$Ep=Sp/P$,其中 $Sp$ 为加速比,$P$ 为节点的个数。

**Table 5 Corresponding acceleration ratio of different data quantity**

**表 5 不同数据量对应的加速比**

| 节点数和加速比 | 数据大小/MB | | | | | | |
|---|---|---|---|---|---|---|---|
| | 200 | 500 | 1 000 | 1 200 | 1 500 | 1 800 | 2 000 |
| 单机 | 218 | 497 | 1 405 | 1 758 | 2 118 | 2 502 | 3 102 |
| 四个从节点 | 132 | 197 | 442 | 486 | 562 | 651 | 804 |
| 加速比 | 1.652 | 2.523 | 3.179 | 3.617 | 3.769 | 3.843 | 3.858 |
| 加速比平均值 | 3.206 | | | | | | |

同时由算法原理可知,传统 TFIDF 算法的时间复杂度为 $O(\log|M|)$,当数据集达到一定范围时,改进的并行算法的时间复杂度为 $O((\log|M|)/N)$,其中 $M$ 为文档数目,$N$ 为节点个数。

(5)传统算法与改进算法分类结果对比。

选择手工整理分类的 Sogou 文本分类语料库进行实验,在以下 10 个类别中每类选择 2000 篇文档为实验数据,分别统计传统算法和考虑词位置信息的改进算法的查全率和查准率,通过多次实验,结果如图 6 所示。可以看到除个别类别外,改进的算法的查全率和查准率都高于传统算法的,提高了分类性能。

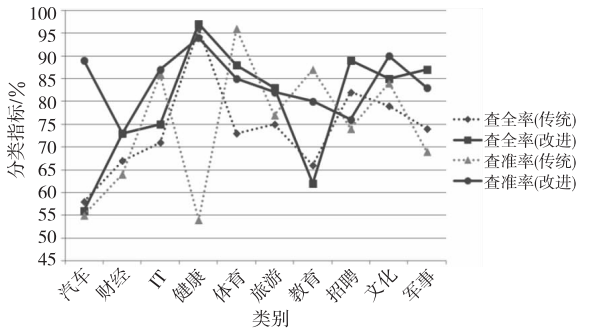



Figure 6 Classification results contrast diagram between the traditional algorithm and improved algorithm

图 6 传统算法与改进算法分类结果对比图

## 4 结束语

本文通过引入加权系数考虑到词位置不同对权重的影响不同并在 Hadoop 平台下利用 MapReduce 编程,对传统的 TFIDF 文本分类算法进行了性能优化,并进行了传统算法和改进算法、单机和分布式环境、不同节点数下的对应的运行时间三组对比实验,同时通过文本分类的查全率和查准率性能指标、并行算法加速比、时间复杂度等参数,验证了考虑词语位置信息并通过 MapReduce 的 TFIDF 文本分类算法可以取得更为高效和准确的分类结果,很好地解决了在单机下无法完成的海量数据高效挖掘的问题。但是,由于算法本身的一些缺陷,比如类内类间偏差对权重的影响和不同词的切分方法对分类的影响等,将进一步改进算法,同时增加节点数和各节点配置,进行不同数据集的相关实验,不断完善算法性能和集群配置,提高海量数据的挖掘能力。

## 参考文献:

[1] Sebastiani F. Text categorization[EB/Ol]. [2005-12-15]. http://nmis.isti.cnr.it/sebastiani/publications/TM05.pdf.

[2] Yang Y. An evaluation of statistical approaches to text categorization[J]. Journal of Information Retrieval, 1999, 1(1/2):67-68.

[3] Xie Xin-jun, He Zhi-jun. Design and implementation of a unique form workflow system[J]. Computer Engineering, 1988, 24(9):53-55. (in Chinese)

[4] Xiang Xiao-jun, Gao Yang, Shang Lin, et al. Parallel text categorization of massive texr based on Hadoop[J]. Computer Science, 2011. (in Chinese)

[5] Wang Yu. Reseach on text classification algorithms based TFIDF [D]. Zhengzhou: Zhengzhou University, 2006. (in Chinese)

[6] Liu Peng. Actual combat Hadoop-open the shortcut to cloud computing[M]. Beijing: Publishing House of Electronics Industry, 2011. (in Chinese)

[7] Li Bin. Improve of TF-IDF algorithm based on Hadoop[J]. Microcomputer & Its Applications, 2012, 31(7): 14-16. (in Chinese)

[8] Text categorization corpus[EB/OL]. [2012-09-15]. http://www.sogou.com/labs/dl/c.html. (in Chinese)

[9] Xue Yi-bo, Wang Jian-zhong. The study of speedup on parallel processing system[J]. Computer Engineering & Design, 1994, 16(1): 12-16. (in Chinese)

## 附中文参考文献:

[3] 谢鑫军, 何志均. 一种单一表单工作流系统的设计和实现[J]. 计算机工程, 1988, 24(9): 53-55.

[4] 向小军, 高阳, 商琳, 等. 基于 Hadoop 平台的海量文本分类的并行化[J]. 计算机科学, 2011, 38(10): 184-188.

[5] 王宇. 基于 TFIDF 的文本分类算法研究. [D]. 郑州: 郑州大学, 2006.

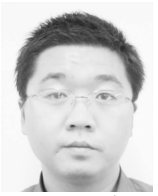
[6] 刘鹏. 实战 Hadoop-开启通向云计算的捷径[M]. 北京: 电子工业出版社, 2011.

[7] 李彬. 基于 Hadoop 框架的 TF- IDF 算法改进[J]. 微型机与应用, 2012, 31(7): 14-16.

[8] Sogou 文本分类语料库[EB/OL]. [2012-09-15]. http://www.sogou.com/labs/dl/c.html.

[9] 薛一波, 王建中. 并行处理中加速比的研究[J]. 计算机工程与设计, 1994, 16(1): 12-16.

## 作者简介:



**王静宇**(1976 -),男,河南开封人,博士,副教授,CCF 会员(E200026132M),研究方向为云计算和信息安全。**E-mail**: btu_wjy@126.com

**WANG Jing-yu**, born in 1976, PhD, associate professor, CCF member (E200026132M), his research interests include cloud computing, and information security.