Hans Journal of Data Mining 数据挖掘, 2018, 8(4), 151-161
Published Online October 2018 in Hans. http://www.hanspub.org/journal/hjdm
https://doi.org/10.12677/hjdm.2018.84017

Hans汉斯

A Comparative Study of Distributed Text Categorization Methods Based on Different Computing Frameworks

Huifeng Tang[1], Wei Guo[2], Le Zhang[2]

[1]Information Engineering University of Strategic Support Force, Zhengzhou Henan
[2]Graduate School, Information Engineering University of Strategic Support Force, Zhengzhou Henan
Email: guowei1533@qq.com

Received: Aug. 15th, 2018; accepted: Aug. 30th, 2018; published: Sep. 6th, 2018

Abstract

In view of the low efficiency of traditional text classification algorithms in the face of increasing mass of text data, a parallel naive Bayes text classifier is designed and implemented on the Spark computing framework, and the implementation process of text classification based on Spark computing framework is introduced. In the experimental stage, the efficiency of text classification is tested under three different computing frameworks, single machine, Map Reduce and Spark, and control experiments are designed under the Spark computing framework using control variables. Experiments show that naive Bayes algorithm in Spark computing framework has high efficiency in dealing with massive text categorization.

Keywords

Text Classification, Computational Framework, Naive Bayes, TF-IDF

基于Spark计算框架的分布式文本分类方法研究

唐慧丰[1]，郭　威[2]，张　乐[2]

[1]战略支援部队信息工程大学，河南 郑州
[2]战略支援部队信息工程大学研究生院，河南 郑州
Email: guowei1533@qq.com

收稿日期：2018年8月15日；录用日期：2018年8月30日；发布日期：2018年9月6日

文章引用: 唐慧丰, 郭威, 张乐. 基于 Spark 计算框架的分布式文本分类方法研究[J]. 数据挖掘, 2018, 8(4): 151-161.
DOI: 10.12677/hjdm.2018.84017

摘　要

针对传统文本分类算法在面对日益增多的海量文本数据时效率低下的问题，论文在Spark计算框架上设计并实现了一种并行化朴素贝叶斯文本分类器，并着重介绍了基于Spark计算框架的文本分类实现过程。实验阶段，分别在单机、Map Reduce和Spark三种不同的计算框架下测试了文本分类的效率，并使用控制变量的方法在Spark计算框架下设计对照实验。实验证明，Spark计算框架下的朴素贝叶斯算法在面对海量文本分类时有着较高的处理效率。

关键词

文本分类，计算框架，朴素贝叶斯，TF-IDF

1. 引言

信息化时代，人类社会每天产生的数据不断增多，达到了 PB 甚至 EB 的数量级。在海量数据中，相当大的一部分是以文本形式存在的半结构化或非结构化数据。为了提高处理海量文本信息时的效率，需要对这些文本信息进行必要的组织和管理。

文本分类技术是指根据文本内容自动将文本划分为一个或多个类别的技术，在自然语言处理领域的许多应用中发挥着关键作用[1]。目前文本分类方法中使用较为广泛的包括 K-最近邻、朴素贝叶斯、决策树等。这些传统文本分类方法在中小规模的数据集上效果较好，但是由于文本分类过程涉及到大量的统计和计算，随着数据量的增大，传统的算法表现出很多不足，最主要的缺点就是分类耗时较高。如何在面对海量数据时快速高效地完成文本分类，对于文本信息的挖掘有着重要的意义。

Google 先后发表的关于 GFS [2]、Map Reduce [3]和 Bigtable [4]的三篇论文为大数据技术奠定了基础，随后 Hadoop 应运而生。Hadoop 不仅提供了 GFS 和 Map Reduce 的开源实现，而且还发展成了包含存储、计算、资源管理等多功能的 Hadoop 生态圈[5]。Hadoop 的 Map Reduce 计算框架出现较早，技术比较成熟，被广泛用于离线数据处理。自 Hadoop 出现至今，其核心组件之一的 Map Reduce 一直都是被广泛使用的并行计算框架，基于 Map Reduce 的文本分类并行化的研究也取得了一定的成果。Spark 作为 Hadoop 生态圈中的新成员，是一个由加州大学伯克利分校推出的基于内存的大数据并行计算框架，它在执行多次迭代计算的任务时，不会将中间结果存储在硬盘上，而是放在内存中，极大地提高了后续过程的计算速度。在执行某些需要迭代计算的任务中，Spark 的速度理论上可以达到 Map Reduce 的近百倍。

为了更直观地了解大规模文本分类在不同计算框架下的计算效率，论文基于 Map Reduce 和 Spark 两种不同的计算框架设计并实现了朴素贝叶斯分类算法，通过对照试验对单机环境以及两种分布式计算框架下的文本分类进行了效率对比。实验证明，基于 Spark 框架的朴素贝叶斯文本分类算法在应对大规模数据时效率较高。

2. 相关技术发展现状

2.1. 文本分类技术

文本分类是一个有监督的学习过程，分为训练过程和分类过程两部分。先对已知类别的训练语料进行学习，得到分类模型，然后利用分类模型对未分类的文本进行分类。

文本分类的过程可以用数学语言描述如下：

假设有一个类别已知的文本集合 $D=\{d_1,d_2,\cdots,d_n\}$ (n 集合中文本总数)，一个文本类别集合 $C=\{c_1,c_2,\cdots,c_m\}$ (m 为类别的个数)，在集合 D 和 C 之间有这样一种关系，用 φ 表示为：

$$\varphi: D \to C$$

该式表示存在一个映射 φ 把 D 中的每个文本 d_i 映射为 C 中的一个或几个类别。文本分类就是通过对训练集文本的学习，来建立这样一个与 φ 近似的映射关系 φ'，这个映射关系就是文本分类器，可以用于未知类别的文本集合的分类。

文本分类的一般流程可以用图 1 来表示。

早在 1957 年，H. P. Luhn [6]发表的一篇有关信息检索的论文中提出了词频统计思想；在上世纪 60 到 80 年代，主要是根据专家提供的知识提炼出合适的规则建立分类器。1981 年，侯汉清[7]教授向国内引入文本分类思想。基于机器学习的自动文本分类方法于 90 年代产生，并且逐渐发展成为主流的分类方法，这种方法是把一些重要的机器学习算法引入到文本分类中。90 年代中期 Joachims 率先将支持向量机引用到文本分类领域，在有限样本的情况下取得了很好的效果。随后，Yiming Yang [8]等人结合决策树算法完成了文本分类，也取得了不错的效果。2001 年，庞剑锋等人[9]把 VSM 向量空间模型运用在中文文本分类；2013 年，张志飞等人[10]采用了基于 LDA 主题模型的方法，对短文本进行分类；2017 年，武建军等人[11]提出了基于互信息的加权朴素贝叶斯分类算法，具有良好的分类效果。

近年来随着计算机和信息技术的发展，数据规模不断扩大，此时单机环境的分类系统已经不能满足需求。面对单机无法处理的大规模数据，对于文本分类并行化的研究逐渐兴起。

江小平等人[12]在 2011 年实现了在 Hadoop 平台上对朴素贝叶斯算法的并行化来进行文本分类；2016 年，长春工业大学的光顺利提出了三个因子改进卡方统计特征选择方法，并在 Spark 平台下实现了朴素贝叶斯分类方法。2017 年，北京交通大学的宋福星[13]设计了 Spark 平台下的基于 BP 神经网络与在线分域特征选择算法(OFFS 算法)相结合的 OFFS-BP 文本分类器。

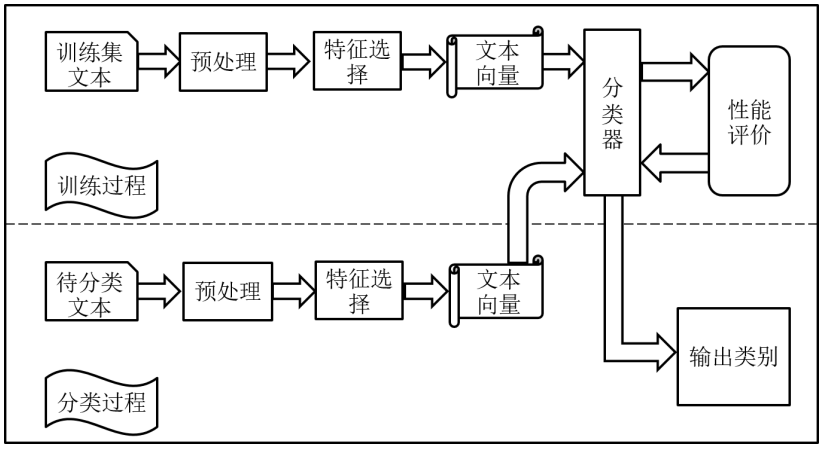

Figure 1. Text classification process
图 1. 文本分类过程

2.2. 分布式计算框架

2.2.1. Hadoop Map Reduce 计算框架

Hadoop Map Reduce 是 Apache 基金会旗下顶级开源项目，属于应用较早、发展较为完善的一种分布式计算框架，现已广泛应用于搜索引擎、推荐系统、日志分析、信息检索等应用，能够有效的解决海量数据的存储、计算以及资源管理调度等问题。Hadoop Map Reduce 可以使用户在不了解分布式底层细节的情况下，开发分布式程序，充分利用集群的资源进行高速运算和存储。

Map Reduce 计算框架的核心思想是“分而治之”，就是把较大的任务分解成若干小任务，然后分发到集群的各个计算节点上分别进行计算，最后将结果汇总。整个过程可分为 Map 和 Reduce 两个阶段。在 Map 阶段会将待处理的原始数据按照一定原则分割成数据块，然后分发到各个计算节点上进行计算；Reduce 阶段会把 Map 阶段得到的结果进行汇总，合并成最终的结果。

2.2.2. Spark 计算框架

Spark 是加州大学伯克利分校的 AMP 实验室所开发的类 Hadoop Map Reduce 通用并行计算框架。Spark 既具有与 Map Reduce 相同的可扩展性、容错性、兼容性，同时与 Map Reduce 相比具有以下几点优势：

- 中间输出和结果存储在内存中，不再需要读写 HDFS，极大地提高了任务的执行速度。
- 使用 RDD (Resilient Distributed Datasets)作为分布式索引来对数据进行分区和处理，能够减少机器之间的数据混合(Data Shuffling)。
- Map Reduce 在数据 Shuffle 时每次都花费大量时间来排序，而 Spark 任务在 Shuffle 中不是所有情况都需要排序。

因此，Spark 能更好地适用于数据挖掘与机器学习领域许多需要迭代的算法。Spark 的发展历程如表 1 所示。Spark 的发展速度非常之迅速，目前，Spark 已经成为代码活跃度最高的大数据处理框架。Spark 官方维护运营公司 Databricks 已经组织并举办了从 2013 年到 2017 年的五次 Spark Summit 技术峰会。Spark 的应用领域正在不断扩大，会逐渐应用到更多行业的各类应用场景中。

3. 文本分类算法流程

3.1. 文本预处理

文本预处理是在正式处理文本数据之前，对原始文本进行的一系列必要的操作，以方便后续的处理。对文本分类而言，任何算法都不能在一整篇文本上执行，必须要经过文本预处理，完成分词、去除停用词等步骤，才能将一整篇文本转化为一组特征项，输入到文本分类算法中去。

Table 1. Spark’s development course
表 1. Spark 的发展历程

时间	发展
2009 年	Spark 由 UC Berkeley 的 AMPLab 研发成功
2010 年 3 月	Spark 正式开源
2011 年	AMP 实验室开始在 Spark 上面开发高级组件
2013 年 6 月	加入 Apache 基金项目
2014 年 2 月	升级为 Apache 基金的顶级项目
2014 年 5 月	Spark 1.0 发布
2016 年 6 月	Spark 2.0 发布
2018 年 2 月	最新版本 Spark 2.3.0 发布

3.1.1. 分词

英文文本的单词之间以空格分隔，在分词时可以直接按空格标点切分，难度相对较低；中文的词与词、短语与短语之间并不存在明确的分隔界限，难度相对较高。无论是中文还是英文，分词的结果精准与否，对文本后续的特征选择和模型训练过程有非常大的影响。因此，选择一种快速且精准的分词算法非常重要。

目前比较流行的分词工具有：IK Analyzer、Jieba、Ansj、Paoding Analyzer、NLPIR、盘古分词等。其中 IK Analyzer 实现相对简单，分词速度快，并且支持多语言分词，支持用户扩展词典。因此论文采用 IK Analyzer 实现文本的分词操作。

3.1.2. 去除停用词

停用词是指在文中大量出现，但几乎不携带任何信息的词，包括冠词、介词、代词等，如中文的“的”、“啊”，英文的“the”、“a”等。这些词的大量存在对文本分析用处极低，但却大大增加了计算量，因此要在预处理时将它们发现并删除。IK Analyzer 分词工具支持在配置文件里配置自己的停词表，极大地降低了去停用词的计算量。

3.2. 文本特征选择

经过预处理后，原始文本中的长句被转换成一组组特征项。如果文档较长，必然会存在特征项的维度过高的情况，并且噪声项也可能会存在，这些都给后续的计算增加了难度[14]。为了提高分类速度和精度，应把在各个类别都比较常见的词语删除，只保留关键词语，这个过程称为文本特征选择。经过文本特征选择，筛选出针对每一类的特征项集合，大大降低特征项的维度，减少计算量。

3.3. 文本表示

文本表示是指将非结构化的文本表示成一种便于计算机理解和处理的数学模型，比如布尔模型、概率统计模型以及向量空间模型。目前比较常用的是向量空间模型 VSM (Vector Space Model)。VSM 可以将文本表示成特征向量的形式。这种特征向量的形式符合大多数机器学习算法的输入，为算法中的数学计算提供便利。但是 VSM 文档特征向量维数众多，在高维向量中每个特征项有不同的权重。因此，在表示文本时需要计算每个特征项的权重，并按权重来进行排序，选取权重较高的特征项作为最佳特征。

文本中的特征权重有多种计算方法，应用比较普遍的就是 TF-IDF 加权算法。其中的 TF (Term Frequency)为特征项频率，即该特征项在某篇文本中出现的次数，反映特征项在文本内部的分布情况，计算方式如下：

$$TF_{w_{ij}} = \frac{n_{w_{ij}}}{\sum_k n_{w_{kj}}}$$

其中 w_{ij} 表示第 j 篇文本中的第 i 个特征项，$n_{w_{ij}}$ 表示该特征项在文本 d_j 中出现的次数，$\sum_k n_{w_{kj}}$ 表示文本 d_j 中所有特征项出现次数之和。

IDF (Inverse Document Frequency)为特征项的逆向文本频率，反映了特征词在整个文本集合的分布情况。其定义如下：

$$IDF_{w_i} = \log \frac{M}{\sum_j \left| j : w_i \in d_j \right|}$$

其中 M 表示语料库中的文本总数，$\sum_j \left| j : w_i \in d_j \right|$ 表示包含词语的文本数目总和。

特征权重计算公式如下：

$$TFIDF_{w_{ij}} = TF_{w_{ij}} \times IDF_{w_{ij}}$$

从 TFIDF 的经典公式中可以发现，如果一个特征项具有很高的词频 TF，并且只是高频出现在一篇或几篇文章中，也就说明该特征项的 IDF 很高，那么该特征项的权重就会高于特征集中的其他项，具有很好的区分性和代表性，适合用来分类。

3.4. 分类方法选择

论文使用的文本分类方法是经典的朴素贝叶斯分类方法，它是由 Maron 和 Huhns 于 1960 年提出的一种基于概率模型的分类方法，后来被 Lewis 引入信息检索和文本分类领域。朴素贝叶斯算法以贝叶斯定理和特征条件独立假设为基础，先通过总结经验获得先验概率，然后根据贝叶斯公式计算相应的后验概率，得到该对象属于某一类别的概率，然后用最大后验概率的类别选为最终对象的类别。

朴素贝叶斯应用到文本分类时，基本描述如下：设有数据集 D，类别数量为 N，即 D 表示为 $C_1, C_2, \cdots, C_N$；数据集中的每个样本有 n 个属性，第 i 个属性是 $X_i = \{1, 2, \cdots, n\}$。对于一个给定的待分类样本 X 和类别 $C_i (1 \le i \le N)$，朴素贝叶斯分类算法使用最大的后验概率 $P(C_i \mid X)$ 来预测 X 所属的类别。

根据贝叶斯定理，文本 X 属于类别 C_i 的后验概率计算如下：

$$P(C_i \mid X) = \frac{P(X \mid C_i) P(C_i)}{P(X)}$$

先验概率计算如下：

$$P(C_i) = n_i / N$$

其中，n_i 为属于类别 C_i 的训练样本的数量，N 为训练样本的总数。基于特征条件独立假设，条件概率计算如下：

$$P(X \mid C_i) = P(X_1 \mid C_i) P(X_2 \mid C_i) \cdots P(X_n \mid C_i) = \prod_{j=1}^{n} P(X_j \mid C_i)$$

建立朴素贝叶斯分类器有两种方法，其一是多项式模型，另一个是多元伯努利模型。这两种模型的区别是对 $P(X \mid C_i)$ 的计算，而多项式模型要优于多元伯努利模型。因此论文在对条件概率进行贝叶斯估计时采用的是多项式模型，计算方法如下面公式所示。公式中分子和分母分别加 1 和加$|V|$进行拉普拉斯平滑。

$$P(X_j \mid C_i) = \frac{1 + TF(X_j, C_i)}{|V| + \prod_{j=1}^{|V|} TF(X_j, C_i)}$$

4. 基于 Spark 的朴素贝叶斯算法设计与实现

朴素贝叶斯算法是一种基于统计的算法，基于特征条件独立假设，该算法具有较高的分类准确率[15]。但其传统串行的实现方式由于时间复杂度较高，只局限于中小规模的数据集，在处理海量数据时该算法的分类效率大大降低。为克服传统算法无法高效处理海量数据的缺点，论文研究了在 Spark 并行计算框架的基础上朴素贝叶斯算法的并行化实现，从而来提升海量数据的文本分类速度。

4.1. 数据预处理

实验将原始文本集存储在 HDFS 上，文本集中有若干分类目录，目录下包含属于此类的若干文档。

在文本预处理阶段，先对文本进行格式化，将文本转化成统一的格式，然后再进行分词和去除停用词等操作。

预处理流程如下：

- 通过读取目录的方式将文本集读入，分别用不同的 int 型数值作为每一类的类标号 label；
- 利用中文分词工具对文本集进行分词；
- 按照贝叶斯模型的输入要求转换输入格式：每个类别形成一个新文本，新文本中每行存放一篇文本，每篇文本的格式为<label, (word_1, word_2,⋯, word_n)>，即类别名和文档分词后的词组成的集合；
- 预处理完成后，会在新目录 transformed Train 下生成与类别数相同的多个文本，这些文本以类标号 label 命名，如 0.txt~19.txt。将预处理过后的数据上传至 HDFS 中。

4.2. 文本向量化

原始数据在预处理之后，其形式还是文本，不能直接输入到分类器中，因此需要先通过文本特征抽取，将文本形式转换为向量形式。

常用的特征提取的方法有 TF-IDF、Word2Vec 以及 Count Vectorizer 等，论文使用的是 TF-IDF 算法，该算法原理和实现较为简单，用以评估一个字词对于一个文件集或一个语料库中的其中一份文件的重要程度。

假设用 t 表示词语，d 表示文本，D 表示语料库。词频 $TF(t, d)$是词语 t 在文档 d 中出现的次数。文件频率 $DF(t, D)$是包含词语 t 的文档的个数。IDF 的计算公式如下：

$$IDF(t,D)=\log\frac{|D|+1}{DF(t,D)+1}$$

其中，$|D|$ 是语料库中的文档总数，为了避免分母为 0 进行加 1 平滑。

TF-IDF 度量值表示如下：

$$TFIDF(t,d,D)=TF(t,d)\cdot IDF(t,D)$$

在 Spark 计算框架下，TF-IDF 被分成 TF 和 IDF 两部分进行计算。

TF：将输入的文本转换成固定长度的特征向量，通过应用 hash 函数将原始特征映射到 index，即 index 就是单词的哈希值，value 是根据映射的 index 计算的单词频数。

IDF：作用于一个数据集并产生一个 IDF 模型，该 IDF 模型接收由上一步产生的特征向量，计算每个词在语料库中出现的文本数目。

文本向量化的流程如图 2 所示。

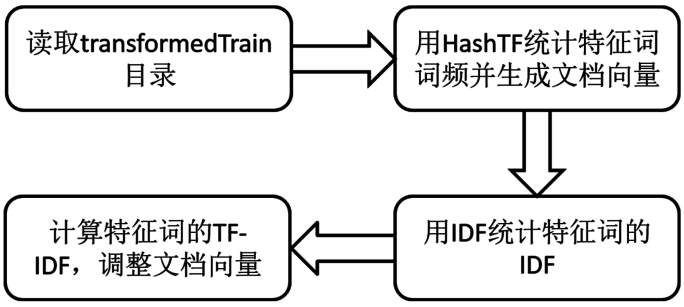

Figure 2. Flow chart of text Vectorization
图 2. 文本向量化流程图

4.3. 模型训练

Spark MLlib 模型训练的输入数据的格式是 RDD(label,features)。训练完成后，生成朴素贝叶斯分类模型。具体步骤如下：

- 准备训练数据 trainDataRdd，训练数据的格式是 RDD(label,features)；
- 调用类伴生对象 NaiveBayes 的静态 train 方法，根据输入参数，初始化 NaiveBayes 类，然后开始训练模型。训练完成后，生成一个贝叶斯分类模型。

5. 实验与分析

5.1. 实验环境

实验环境是由 3 台配置相同的计算机组成的集群，一台计算机作为 Master 节点，另外两台作为 Slave 节点，每台计算机的硬件配置均为内存 8 GB，硬盘 500 GB，操作系统采用 ubuntu-14.04.1-server-amd64 版本。软件采用的 Hadoop 版本为 2.7.1，Java 版本为 1.8.0，Spark 版本为 2.1.0，Scala 版本为 2.11.7，使用的 Spark 开发环境是 IntelliJ IDEA，Maven 版本为 3.3.9。

集群中各节点的 IP 地址如表 2 所示。

节点部署情况如图 3 所示。Master 运行 Namenode 节点、Master 进程和 Worker 进程，Slave 运行 DataNode 和 Worker 进程。

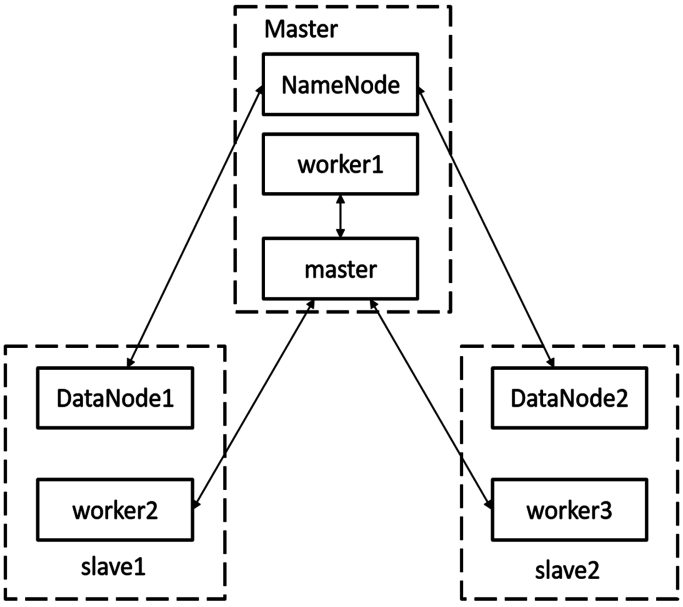

Figure 3. Node deployment schematic
图 3. 节点部署示意图

Table 2. Host-IP address corresponding table
表 2. 主机-IP 地址对应表

主机名	IP 地址
Master	192.168.0.100
Slave1	192.168.0.101
Slave2	192.168.0.102

5.2. 实验语料

论文使用复旦大学自然语言处理实验室的文本分类语料库作为实验数据集，其中训练语料共 9804 篇文档，包含教育、环境、军事、政治、体育等 20 个不同类别的文本集合，共计约 133 MB；测试语料 9833 篇文档，共计约 135 MB。论文在原始数据的基础上将测试语料分别复制到 10 倍，20 倍，30 倍的规模，用以测试不同计算框架在面对大规模数据时的运行效果。

5.3. 实验设计及结果分析

5.3.1. 实验一

为了对比不同计算框架运行文本分类程序的效率，论文中使用同一种分类算法，在单机、Map Reduce、Spark 三种计算框架下对相同数据集进行分类效率的对比实验，分别记录运行时间和分类准确率。

- 单机环境

在单机环境下编写并调试了朴素贝叶斯算法。在使用训练语料进行模型训练过后，使用 10 倍规模(1350 MB)的测试数据的数据集为输入，进行了多次测试，记录运行时间和分类准确率。具体试验记录如表 3 所示。计算均值后得出平均运行时间是 2,873,036 ms，准确率是 89.29%。

- Map Reduce 计算框架

在 eclipse 上把单机版朴素贝叶斯算法进行分布式改写，使其符合 Map Reduce 计算框架的运行模式，将原始数据集放至 HDFS 上作为输入，启动 Hadoop 集群运行程序，记录运行时间和分类准确率，如表 4 所示。由运行结果可以得知，相同的数据集在 Map Reduce 下的运行时间是 141,912 ms，准确率是 88.16%。

- Spark 计算框架

在 IDEA 中编写并调试了朴素贝叶斯算法的 Spark 版本。对原始数据集在 Spark 平台上进行文本分类，将写好的程序打成 jar 包用 spark-submit 提交给集群运行，记录运行时间和分类准确率。由运行结果可以看到，相同的数据集在 Spark 上面的运行时间是 69,583 ms，准确率是 87.03% (表 5)。

- 结果分析

基于 Spark 计算框架的朴素贝叶斯算法比单机上的速度快得多，大约是单机的 35.7 倍，是 Map Reduce 计算框架的 2 倍。很明显，在处理大规模数据集时，单机环境已经很难升任。实验数据充分显现了 Spark 并行计算框架相比单机环境和 Map Reduce 框架在处理速度上的优越性。

Table 3. Experimental results under single machine environment
表 3. 单机环境下实验结果

	1	2	3	4	5
时间(ms)	2,703,370	2,815,270	3,125,703	2,954,320	2,766,520

Table 4. Experimental results in Hadoop map reduce environment
表 4. Hadoop Map Reduce 环境下实验结果

	1	2	3
时间(ms)	145,710	154,323	125,703

Table 5. Experimental results under spark environment
表 5. Spark 环境下实验结果

	1	2	3
时间(ms)	75,710	64,320	68,720

Spark 计算框架和 Hadoop Map Reduc 计算框架相比，运行时间缩短大概一半。至于 Spark 的运行速度没有达到理论上那么高，可能存在以下几个原因：一是由于迭代计算过程各节点间的通讯占用了时间；二是实验所用数据量还不够大，不能充分发挥 Spark 计算框架的优势；三是 Spark 计算框架对硬件要求较高，当内存较小时也可能存在计算速度较慢的情况。

以上是基础性实验，通过对比一定规模的数据集在单机、Map Reduce 框架和 Spark 框架下的运行速度，验证了 Spark 在提高文本分类速度上的优势。为了更显著地体现 Spark 的良好性能，在下面两组实验中采用控制变量法，以“数据量”和“计算节点数”为变量，分别在 Spark 计算框架下做了两组实验，并分析不同变量对于两种不同计算框架性能的影响。

5.3.2. 实验二

实验二在 Spark 平台上较大规模的数据集进行测试，Spark 的计算节点数设置为 3 个。将原始数据的 10 倍规模、20 倍规模和 30 倍规模记为数据集 1、数据集 2、数据集 3。每个数据集分别在两种不同的计算框架进行三次试验，计算平均运行时间，并将结果用折线图的形式直观地表示，分析数据量的增大对运行速度的影响。三个数据集的平均运行时间表 6 所示。以数据集 1 的处理速度为基准，Spark 在处理数据集 2 和数据集 3 时，运算速度分别提高了 5%和 18%。实验证明 Spark 在进行文本分类时，数据规模越大优势越明显。

5.3.3. 实验三

实验三使用相同的数据集在不同节点数量的 Spark 集群下运行，变量是集群中参与计算的节点数目，数据采用数据集 3。分 4 组实验，第一组有 1 个 master 和 1 个 worker，第二组是 1 个 master 和 2 个 worker，第三组是 1 个 master 和 3 个 worker，第三组是 1 个 master 和 4 个 worker。分别记录每组的运行时间，并将结果转化为速度以折线图的形式表示，分析节点数增加对运行速度的影响。第一组速度假设为 1，各组速度比较如图 4 所示。

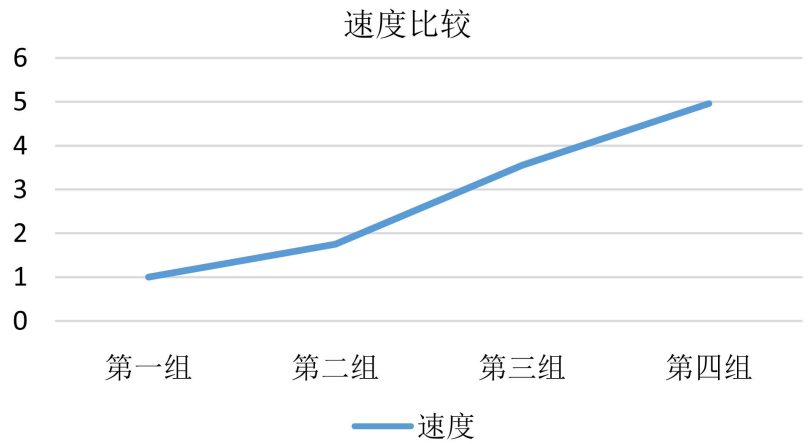

Figure 4. The influence of the number of calculated nodes on the speed of operation
图 4. 不同计算节点数对运行速度的影响

Table 6. The average running time of the three groups of experiments
表 6. 三组实验平均运行时间

	数据集 1	数据集 2	数据集 3
时间(ms)	69,583	132,533	175,523

从运行速度来看，相同的数据集进行测试，随着集群数量的增加，文本分类的运算速度呈增加趋势。而且通过速度增长的趋势可以看出，集群的加速比也在不断提高。从实验结果得出，Spark 在处理朴素贝叶斯分类算法上具有良好的加速比，增加 Spark 上的 worker 节点数目能够明显加快文本分类并行化的速度。

6. 总结

针对当前日益增多的海量文本数据，论文在朴素贝叶斯分类算法的基础上，利用 scala 编程语言和 Spark MLlib 算法库，设计并实现了基于 Spark 的朴素贝叶斯文本分类器。在实验环节，用控制变量法对不同计算框架的运行速度进行比较，并对 Spark 平台下文本分类的效率进行测试。实验表明 Spark 能显著提高海量文本的处理效率和处理能力，并且具有良好的可扩展性。

虽然实验结果比较理想，但论文的研究仍然存在一些不足之处。比如实验数据量不够大，算法和试验的细节优化做得还不够好，在接下来的研究中会不断完善。

参考文献

[1] 罗元帅. 基于随机森林和 Spark 的并行文本分类算法研究[D]: [硕士学位论文]. 成都: 西南交通大学, 2016.

[2] Ghemawat, S., Gobioff, H. and Leung, S. (2003) File and Storage Systems: The Google File System. *ACM Symposium on Operating Systems Principles*, Bolton Landing, 19-22 October 2003, Vol. 37, 29-43. https://doi.org/10.1145/1165389.945450

[3] Dean, J. and Ghemawat, S. (2004) Map Reduce: Simplified Data Processing on Large Clusters. *Proceedings of Operating Systems Design and Implementation* (*OSDI*), **51**, 107-113.

[4] Chang, F., Dean, J., Ghemawat, S., *et al*. (2006) Big Table: A Distributed Storage System for Structured Data. *Proceedings of Use Nix Symposium on Operating Systems Design & Implementation*, **26**, 205-218.

[5] 光顺利. 基于 Spark 的文本分类的研究[D]: [硕士学位论文]. 长春: 长春工业大学, 2016.

[6] Luhn, H.P. (1959) Auto-Encoding of Documents for Information Retrieval Systems. Modem Trends in Documentation. Pergamon Press, New York.

[7] 侯汉清. 分类法的发展趋势简论[M]. 北京: 中国人民出版社, 1981.

[8] Yang, Y. (1999) An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval*, **1**, 69-90. https://doi.org/10.1023/A:1009982220290

[9] 庞剑锋, 卜东波. 基于向量空间模型的文本自动分类系统的研究与实现[J]. 计算机应用研究, 2001, 18(9): 23-26.

[10] 张志飞, 苗夺谦, 高灿. 基于 LDA 主题模型的短文本分类方法[J]. 计算机应用, 2013, 33(6): 1587-1590.

[11] 武建军, 李昌兵. 基于互信息的加权朴素贝叶斯文本分类算法[J]. 计算机系统应用, 2017, 26(7): 178-182.

[12] 江小平, 李成华, 向文, 张新访. 云计算环境下朴素贝叶斯文本分类算法的实现[J]. 计算机应用, 2011, 31(9): 2551-2554.

[13] 宋福星. 基于 Spark 的超大文本分类方法的设计与实现[D]: [硕士学位论文]. 北京: 北京交通大学, 2017.

[14] 贺鸣, 孙建军, 成颖. 基于朴素贝叶斯的文本分类研究综述[J]. 情报科学, 2016, 34(7): 147-154.

[15] 李方, 刘琼荪. 基于改进属性加权的朴素贝叶斯分类模型[J]. 计算机工程与应用, 2010, 46(4): 132-133.

知网检索的两种方式：

1. 打开知网页面 http://kns.cnki.net/kns/brief/result.aspx?dbPrefix=WWJD
   下拉列表框选择：[ISSN]，输入期刊 ISSN：2163-145X，即可查询
2. 打开知网首页 http://cnki.net/
   左侧“国际文献总库”进入，输入文章标题，即可查询

投稿请点击：http://www.hanspub.org/Submission.aspx
期刊邮箱：hjdm@hanspub.org