学校代号：10532

学　　号：S11102058

密　　级：普通

湖南大学硕士学位论文

KNN 文本分类及特征加权算法研究

学位申请人姓名：　叶　丹

导师姓名及职称：　杨科华　副教授

培　养　单　位：　信息科学与工程学院

专　业　名　称：　计算机科学与技术

论文提交日期：　2014 年 5 月 14 日

论文答辩日期：　2014 年 5 月 28 日

答辩委员会主席：　骆嘉伟　教授

Research on KNN Text Classification and Term Weighting algorithm

by

YE Dan

B.E.(Hainan University)2011

A thesis submitted in partial satisfaction of the

Requirements for the degree of

Master of Engineering

in

Computer Science and Technology

in the

Graduate School

of

Hunan University

Supervisor

Associate Professor YANG Kehua

May, 2014

湖 南 大 学

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

作者签名：叶丹　　日期：2014年 6 月 4 日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权湖南大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

1、保密□，在_____年解密后适用本授权书。

2、不保密☑。

（请在以上相应方框内打“√”）

作者签名：叶丹　　日期：2014年 6 月 4 日

导师签名：杨科华　　日期：2014年 6 月 4 日

摘 要

随着信息技术和互联网技术的迅速发展，互联网上的数据量呈指数级增长。如何处理如此庞大而急剧增长的海量数据成为信息科学与技术领域所面临的一大挑战。文本分类作为组织和处理海量文本数据的关键技术，可以在较大程度上解决信息的纷繁芜杂问题，帮助用户快速地检索、查询、过滤和利用信息。

本文学习并研究了文本分类及其相关技术，详细介绍了文本分类处理流程中的各个环节，包括：文本预处理、特征选择、特征权重计算、文本分类算法、性能评价。

文本分类算法及特征权重计算是文本分类过程中比较重要的两个问题。特征权重算法的好坏对分类结果的精确度有很大的影响，而分类算法的优劣则直接影响分类效率和分类结果的准确率。本文主要围绕这两个问题进行研究。本文研究内容及创新工作主要体现在以下三个方面：

1、TFIDF (Term Frequency and Inverse Documentation Frequency)加权算法的研究与改进。特征词权重算法对文本分类的精确度有着非常重要的影响，TFIDF 加权方法是 VSM(Vector Space Model)模型下应用最广泛的一种权重算法。传统特征权重算法 TFIDF，忽略了特征词与其他词语之间的语义联系及其在文本集中各个类别间、类内部的分布情况。针对该问题，本文在信息熵与信息增益的基础上，加入词语的语义关联，提出了一种结合语义、信息熵、信息增益的 TFIDF 改进算法(S-TFIDFIGE)。

2、KNN(K-Nearest Neighbor)分类算法的研究与改进。KNN 算法是当前一种主流文本分类算法，因其实现简单、准确率较高而被广泛应用。但是，KNN 算法具有计算复杂度高，分类效率较低的缺陷，限制了其在海量文本分类上的应用。MapReduce 是一个通用性和可扩展性都较强的分布式并行计算模型，能有效地处理海量数据。本文在深入分析了 KNN 分类算法自身特点及 Hadoop MapReduce 编程模型优势的基础上，提出一种基于 MapReduce 并行的 PKNN 算法。

3、设计并进行了相关实验，验证了本文权重改进算法 S-TFIDFIGE 和分类改进算法 PKNN 的可行性和有效性，且对改进后的 S-TFIDFIGE 和 PKNN 算法进行了结合试验。

文章提出的 S-TFIDFIGE 和 PKNN 算法，不仅能提高文本分类的准确率，还可以大幅度减少文本分类的时间，提高文本分类的效率，能适用于处理大规模文本数据分类的实际应用。

关键词：文本分类；TFIDF 算法；语义；KNN 算法；MapReduce

Abstract

With the rapid development of information technology and Internet technology, the text information is increasing exponentially. How to deal with such huge and sharply increasing mass data has become a challenge to the field of information science and technology. Text classification, which is a key technology to organize and manage massive text data, can solve various kinds of problems for information in a large extent. Moreover, it can help users to quickly retrieve, query, filter and use information.

This paper studies and researches the text classification and its related technologies. It focuses on each step during the process of text classification, including text preprocessing, text feature extraction, term weighting algorithm, text classification algorithm and performance evaluation.

Text classification algorithm and term weighting algorithm are the key issues in text classification. Term weighting algorithm has a great effect on the precision rate of classification result, while classification algorithm has direct impact on the efficiency and accuracy of classification. Therefore, this paper mainly focuses on these two problems. The research contents and innovation points in this paper are revealed from the following aspects:

Firstly, this paper focus on the research and improvement of TFIDF algorithm. Term weighting algorithm has a significant influence on the classification results, and the TFIDF is one of the most popular weight algorithms in VSM model. The traditional TFIDF algorithm ignores not only the semantic relation between terms and other feature words, but also the proportion of distribution of terms in categories and between categories of the text datasets. To solve the problem, based on information entropy and information gain, this paper introduces the semantic relation and proposes an improved TFIDF algorithm(S-TFIDFIGE) combining semantics with information entropy and information gain.

Secondly, more attention is paid to the research and improvement of KNN algorithm. KNN algorithm is a widely used algorithm for text classification because of its simple realization and high accuracy. However, since KNN has high computation complexity and low efficiency which limit its application in classification for large quantity of text. MapReduce, a distributed parallel computation model, has strong universal and scalability and can manage mass data

effectively. After making a deep analysis of the characteristics of KNN algorithm and the advantage of Hadoop MapReduce programming model, this paper proposes a PKNN algorithm which based on MapReduce parallel computation.

At last, a series of experiments are designed and complemented to verify the feasibility and validity of improved algorithm S-TFIDFIGE and PKNN, moreover, the improved S-TFIDFIGE algorithm and PKNN algorithm are combined for text classification

By combining the S-TFIDFIGE algorithm and PKNN algorithm proposed in this paper can not only speed up the classification efficiency, but also improve the accuracy of text classification. It can be applied to manage the classification of a large amount of text.

Keywords: text classification; TFIDF algorithm; semantics; KNN algorithm; MapReduce

目 录

插图索引

附表索引

第1章 绪 论

1.1 研究背景及意义

信息技术和互联网技术的高速发展，尤其是20世纪90年代以来，因特网技术发展迅猛，将我们带入了信息爆炸时代。全世界每年出版的图书多达80万种，期刊多达40万种，其它文献信息资料多达400万种，发表的科学论文近500万篇。加利福尼亚伯克利大学的一项研究表明过去3年中全球信息量翻了一番，全世界的信息量以平均每年30%左右的速度飞速增长[1]。Internet上的信息增长速度更是惊人。中国互联网络信息中心发布的《第29次中国互联网络发展状况统计报告》中显示，截至2009年12月31日，我国网页总数为336亿个，年增长率超过100%[2]。中国产业信息网发布的《2013-2018年中国移动互联网市场全景调研及投资趋势预测报告》显示，截至2013年12月，中国网页数量已达1500亿个。

面对如此庞大呈指数级增长的海量信息，如何有效地组织和管理这些信息，并能快速、准确地从中检索用户所需要的信息是当前信息科技领域所面临的一大挑战。文本分类作为组织和处理海量文本数据的关键技术，能够帮助用户快速、准确地定位所需要的信息，在很大程度上解决了信息杂乱问题。因此，自动文本分类作为一项具有重大实用价值的和意义的技术，受到了国内外研究者的广泛关注。

文本分类的任务是根据文本内容，将每篇文本自动归入到一个或多个预先定义好的类别中[3]。在自动文本分类中普遍采用向量空间模型(VSM,Vector Space Model)，即以向量 $D_i(d_{i1},d_{i2},...,d_{in})$ 来表示文本，其中 d_{ij} 表示文本 D_i 的第 j 个特征词的权重，两个文本的相似度通过计算两个向量的余弦值来衡量，特征权重算法的优劣将直接影响到文本分类的精确率[4]。因此，特征权重计算成为人们研究的一个热点。目前较为著名的文本分类方法包括KNN、支持向量机、贝叶斯算法、神经元网络和决策树等。其中KNN分类算法因其过程简单、分类准确率较高等优点被广泛应用。但KNN算法本身是一种懒散型分类算法，几乎所有的计算工作都是在分类阶段进行，具有计算复杂度高，分类效率较低的缺点。因此如何提高KNN分类算法的分类效率成为学术界关注的焦点。

1.2 国内外研究现状综述

1.2.1 文本分类研究现状

国外对于自动文本分类技术的研究开展的较早，可以追溯到20世纪50年代末。1959年，IBM的H.P Luhn对文本分类进行了开创性的研究，首次提出根据

词语在文本中出现的次数和分布来进行分类的思想[5]。随后，许多著名学者如 Maron、Kuhns、Borko 及 Salton 等对文本分类进行了大量的研究。国外对文本分类的研究，大致经历了如下三个阶段。

可行性阶段(1959-1964)：对文本分类的可行性进行相关研究。1960 年，Maron 和 Kuhns 在 ACM 期刊上发表了第一篇有关自动文本分类的论文[6]。1962 年，Harold Borko 等人提出利用因子分析法进行文本分类。

实验性阶段(1965-1974)：对自动文本分类进行相关实验研究。1971 年，Rocchio 提出了利用反馈信息来修正文本向量的权重，得到简单的线性分类器[7]；Salton 则实现了一个文本智能检索系统[8]。

实用化阶段(1975-至今)：自动文本分类被应用到各个领域，如：信息检索、搜索引擎、电子邮件分类系统。最具代表性的有 IBM 开发的文本智能挖掘机，Autonomy 公司开发的 Concept Agents 等。此外，还有一些比较成功的文本分类系统，例如：麻省理工学院研究开发的邮件分类系统，卡内基集团研究开发的 Construe 系统等[9]。

近几年来，国外学者对文本分类技术的研究不断深入，主要体现在文本表示模型、特征选择算法、特征加权算法、文本分类算法等方面。文献[10]在 VSM 基础上，加入了特征词的位置信息及两文本间的内容密切度对其进行改进，提出了一种 E-VSM(Enhanced-Vector Space Model)模型。文献[11]针对朴素贝叶斯分类器的多类别分类情况提出了 MOR (Multi-class Odds Ratio)和 CDM（Class Discriminating Measure）的特征选择方法。文献[12]则对 TFIDF、LSI(Latent Semantic Index)和 Multi-Word 方法进行了对比实验，实验表明 LSI 的分类性能优于其他两种文本表示方法。文献[13]提出了一种使用自适应框架的特征选择方法，同时考虑了特征词在属于该类和不属于该类的文本中的分布情况。文献[14]提出一个采用模糊的相似性为基础的自建特征聚类算法，从而达到降低文本维度的目的。文献[15]通过敏感词在数据库中出现的频率及其分布情况来计算敏感词的 SIF-DIF（sensitive items frequency-inverse database frequency）的值，然后进行排序确定其优先级，最终达到隐藏敏感数据的目的。文献[16]提出了利用 TFIDF 和贝叶斯来预测 Bug 的性质，实现了一个 Bug 挖掘工具，该工具可以通过贝叶斯分类器来预测引入 Bug 的性质。文献[17]提出了一种利用作者和特征词信息对短文本进行分类的方法，如：微博 Twitter。文献[18]提出了一种改进的快速 K-means 文本分类算法。文献[19]则提出了一种快速的自动文本分类算法，降低了文本分类的计算复杂度，提高了文本分类的效率。

国内对自动文本分类的研究开展的相对较晚，始于 80 年代初，大致经历了可行性探讨，辅助分类系统，自动分类系统三个阶段。1981 年，侯汉清教授介绍了国外在信息检索，自动文本分类等方面的研究情况，并对利用计算机进行文本挖

掘进行了探讨[20]。其后，很多单位和企业都投入到文本分类的研究热潮当中，到目前为止，国内已有包括中国科学院、北京大学、清华大学、复旦大学、山西大学、东北大学等多个单位从事文本分类领域的研究，并取得一定的成果，如：中山图书馆的莫少强研究开发的计算机辅助图书分类系统、清华大学吴军研发的自动文本分类系统，东北大学研发的图书馆分类专家系统以及中国科学院开发的智多星中文文本分类器等[21]。

另外，很多学者也对自动文本分类技术进行了大量的研究。黄萱菁、吴立德等提出了一种基于机器学习的，适合多种语言的文本分类模型[22]。李荣陆等将最大熵模型引入到文本分类中，取得了不错的分类效果[23]。张云涛、龚玲等人提出了一种改进的 TFIDF 权重算法，提高了文本分类的精确度[24]。董小国等提出基于句子重要度的特征词权重计算方法[25]。国内对于分类算法的研究也很多，如：田冬阳提出了一种改进 SVM(Support Vector Machines)算法，通过构造多核函数，优化 SVM 参数，并利用改进后的 SVM 构造了构造文本倾向性分类算法[26]。文献[27]通过测试样本的密度，合并高密度区域的样本来减少样本倾斜对 KNN 算法的影响。文献[28]提出了一种基于基于 BP(Back Propagation)网络和 Isomap 算法的垃圾邮件过滤算法。文献[29]利用超边替代策略训练超网络分类模型，提出了一种基于演化超网络的中文文本分类方法。文献[30]则在分布式云平台下实现了贝叶斯分类算法，减少了分类时间，提高了分类效率。

自动文本分类技术经过了 50 多年的发展，已经逐渐趋于成熟，分类精确度也已经达到了较高的水平，但由于文本数据具有高维性以及分类算法的复杂性特点，使得文本分类的效率问题再次成为制约其进一步发展的瓶颈。特别是在网络技术高速发展的今天，文本数据海量增长，如何提高文本分类的分类效率，使其适于处理大规模文本数据成为当今研究的重点。

1.2.2 TFIDF 特征加权算法研究现状

在自动文本分类中普遍采用向量空间模型(VSM)来表示文本，通常需要要对文本进行分词、特征选择和特征权重计算处理，最后形成一个 N 维的空间向量。目前，常用的特征词权重计算方法有布尔加权算法、频度加权算法、平方根加权算法、对数加权算法、熵加权算法和 TFIDF 加权算法等。其中 TFIDF 因其算法思想简单，容易实现，能够获得较高的查全率和查准率而被广泛采用。虽然 TFIDF 特征加权算法拥有诸多的优点，但就 TFIDF 算法本身来说，也存在许多需要改进的地方。

二十世纪九十年代，TFIDF 在文本分类中的应用开始受到国内外研究者的广泛关注，他们针对 TFIDF 存在的缺陷，做了大量的改进工作，这些改进算法大致可以分为以下三类。

(1) 类内、类间分布偏差

传统 TFIDF 算法，将文本集合作为整体来考虑，特别是 IDF 的计算，并未考虑特征词在类间和类内的分布情况。文献[31]把信息论中信息增益应用到文本集合的类别层次上，提出了一种改进的权重公式 *tf*idf*IG* ，用改进的权重公式来衡量词语在文本集合的各个类别中分布比例上的差异，考虑了特征词的类间分布对权重计算的影响。TFIDFIG 算法能较好地反映特征词在类别间的分布情况对特征词权重的影响。但该算法并没有考虑特征词的类内分布对其权重的影响，文献[32]在 TFIDFIG 的基础上，引入了信息熵的概念，提出了一种结合信息增益与信息熵的 TFIDFIGE 算法，进一步考虑了特征词在类内的分布情况对权重计算的影响。文献[33]从信息论的角度出发，在 TFIDF 权重的公式上引入类内、类间因子，提高了文本权重计算的准确度。文献[34]则利用信息论的原理来分析关键词在微博类中的分布情况，提出了一种适用于微博关键字提取的 TFIDF 算法。

(2)语义缺失

传统的特征词权重计算方法，在计算权重的过程中把每个特征词视为孤立的，线性无关的。文献[35]从词语语义相似度角度出发，通过分析相似特征词在文本集中的分布情况对 TFIDF 方法进行了改进。文献[36]针对 TFIDF 词频采用线性处理的不合理性以及难以突出对文本内容起关键性作用的特征的缺点，提出了一种基于“Sigmiod 函数”和“位置因子”的新权重方案。

(3)其它改进

文献[37]针对 TFIDF 在对文件长度标准化时倾向于对短文本的特征词赋予较高的权重，提出一种改进算法 R-tfidf，消除了这种负面影响。文献[38]采用 BNS(Bi-Normal Separation)来代替 IDF，提出了新的权重计算方法。文献[39]提出了一种基于 Hadoop 实现的并行 TFIDF 算法，提高了权重计算的效率。

1.2.3 KNN 文本分类算法研究现状

目前较为著名的文本分类方法有包括 Rocchio 算法、朴素贝叶斯算法、神经网络算法、K 近邻(KNN)、支持向量机(SVM)等。相关研究证明，KNN 算法是 VSM 模型下最好的分类算法之一。KNN 是一种基于实例学习的方法，其主要优点体现在算法较成熟、分类效果稳定、实现方便、支持增量学习，但在 KNN 同样存在着许多不足之处。

针对KNN算法的不足，人们做了深入研究并提出许多卓有成效的改进方法。这些改进算法大致可以分为以下四类。

(1) 分类速度慢

针对KNN计算复杂度高，分类速度慢，存储开销和计算开销较大这一情况主要有三种方法来提高KNN的文本分类效率。

第一种方法是减小训练集的规模。在实际的数据集中，通常训练集中的样本的分类信息存在冗余，因此可以通过对训练样本进行剪裁来减小计算量。文献[40]提出了一种基于密度约减训练样本的方法：首先，对样本集进行聚类，约减训练样本集中的“噪声”样本；然后将类中相似度高的若干文档合并成一个文档，从而达到减少训练样本的目的。文献[41]则提出了一种基于聚类改进的KNN文本分类算法：该算法通过聚类将训练文本集中若干相似的文本合并成一个中心文档，减少了KNN进行相似度计算的文档数，提高了分类速度。

第二种方法是引入高效的索引方法，减少查找K个最近邻的计算开销。文献[42]提出了TFKNN方法：该方法通过构造一颗SSR树加快搜索待分类文本的K个近邻。文献[43]则提出一种基于特征空间索引的KNN分类算法：该算法把特征词集看成是各个文档的索引，根据待分类文本的特征词在类特征词集中出现的次数进行分类，减少了待分类文本与特征词集比较的时间。

第三种方法是通过提高计算速度来减少计算时间，从而达到提高分类算法效率的目的。文献[44]从并行化算法的角度出发，提出一种分布式KNN分类算法(DKNN算法)，该算法与传统的串行算法相比，能显著地提高分类速度。

(2) 样本分布密度不均

传统KNN的决策规则一个明显的缺点是，当样本分布密度不均匀时，只按照前K个近邻的顺序而忽略了它们的距离，这样容易造成误判，影响分类的性能。文献[45]通过先检测训练集中每个类别的密度，然后在传统文本相似度计算公式的基础上除以训练样本所在类的密度，很好地解决了训练集分布不均的问题。文献[46]介绍了一种基于密度的改进KNN文本分类算法：通过对高密度区域样本进行裁减，使训练集样本分布均匀化来达到减小样本分布不均带来的影响。文献[47]则通过对待测文本的K个近邻样本给出相应的权重：如果该近邻属于小类，就为它分配一个较大的权限，如果该近邻属于大类，就为它分配一个较小的权限，解决了当各类文本分布不均匀时KNN分类器性能下降的问题。

(3) K值的选取

由于KNN算法中几乎所有的计算都发生在分类阶段，而且分类效果很大程度上依赖于K值的选取，K值的选取很重要：K值过小不能充分体现待分类文本的特点；K 值过大则会造成噪声增加而导致分类效果降低。文献[48]提出一种基于并行遗传算法的KNN方法，通过该方法能够得到优化的K值，提高KNN分类精度。文献[49]则提出一种动态获得K值的KNN算法，实验结果表明，动态获取K值的KNN分类算法可以有效地避免训练文本集和测试文本集的大小对分类造成的影响，可以得到较好的分类效果。

(4) 其他改进

除了上述的各种方法外，也有研究者将KNN分类方法和其他算法进行集成，

从而提高了KNN分类器的分类性能。如：文献[50]将贝叶斯算法和KNN分类器进行集成，利用测试文档的后验概率信息对训练文档集的多路静态搜索树进行剪枝，以缩小测试文本K近邻的搜索空间，从而提高KNN算法的效率。文献[51]将SVM和KNN算法进行集成，提出了一种改进的Web文本分类算法。

1.3 本文工作

本文首先对文本分类的相关技术，包括文本预处理、特征选择、特征权重计算、分类算法和分类结果评估方法进行了全面的论述。然后着重探讨了 TFIDF 特征加权算法跟 KNN 分类算法，并进行了相应的改进工作。本文的主要研究内容体现在以下方面：

1、TFIDF 特征加权算法的研究与改进。TFIDF 是目前使用最广泛的加权算法之一，但是该方法存在着一些固有缺陷，如：忽略了特征在类内、类间的分布偏差对权值计算的影响。近年来，研究者们针对其缺陷做了大量的改进工作，但之前的大多工作，把每个特征词视为孤立的，不相关的，很少涉及词汇语义部分的研究。实际上，特征词之间是存在着语义关联的。本文针对这一问题，首先对词语进行语义分析，然后将有语义关联十分密切的词归为一组，再对特征词进行逆文档频率、信息熵、信息增益的计算，提出一种结合语义、信息熵、信息增益的 TFIDF 算法(S-TFIDFIGE)。

2、KNN 算法的研究与改进。KNN 是 VSM(Vector Space Model)模型下最好的分类算法之一，具有算法简单、概念清晰、稳定性好等诸多优点。但 KNN 算法是一种懒散的分类算法，所有计算工作都是在分类阶段完成的，具有计算复杂度高，分类速度慢的缺点，限制了其在海量文本处理方面的应用。因此，如何提高 KNN 的分类速率具有重要的意义。MapReduce 作为通用性与可扩展性都较高的分布式并行计算模型，能有效地处理海量文本数据的相似度计算过程。本文在深入研究了 KNN 算法自身的特点及 MapReduce 编程模型的基础上，提出一种基于 MapReduce 的 PKNN 算法，充分利用了 MapReduce 分布式编程模型的海量数据处理优势，在很大程度上提高了 KNN 的分类速度。

3、验证改进后的 S-TFIDFIGE 和 PKNN 算法的可行性和有效性。本文首先分别针对各个改进算法设计并进行了相关实验，然后将两个算法结合起来进行了分类实验，最后对实验结果进行了详细分析。

1.4 本文结构

本文的结构如下：

第1章：介绍了本课题的研究背景和意义，文本分类、TFIDF及KNN研究现状，

主要研究内容，以及论文的组织结构。

第2章：对文本分类过程中的关键技术进行了研究，包括：文本预处理、文本特征选择、特征权重计算、文本分类算法、性能评估。并对MapReduce分布式并行编程模型及其主要思想，执行过程和模型实现进行了详细的介绍。

第3章：对传统TFIDF加权算法进行了介绍，并对其存在的缺陷进行了详细的分析，针对TFIDF忽略了特征词之间语义关联以及特征词在类内、类间分布的不足，提出了一种结合语义、信息熵和信息增益的TFIDF改进算法S-TFIDFIGE，并给出了算法的详细描述。

第4章：对KNN算法进行了详细介绍，并分析了传统KNN文本分类算法存在的缺陷，指出了KNN算法存在分类效率低，易受类偏斜影响等不足。针对KNN算法计算复杂度高，分类速度较慢的问题，提出了一种基于MapReduce并行化的PKNN分类算法，并给出了该改进算法中涉及的三个函数Map()、Combine()、Reduce()的详细设计。

第5章：设计和实施相关实验，验证了第三、四章所提出的改进后的权值算法S-TFIDFIGE和分类算法PKNN的可行性和有效性，并将改进后的S-TFIDFIGE和PKNN算法进行结合实验。

最后，对本文所完成的工作进行了总结，并对下一步的研究工作进行了展望。

第2章 文本分类和 MapReduce 技术概述

2.1 文本分类技术概述

文本分类是文本挖掘领域的一个重要分支，它的任务是在给定的分类模型下，根据文本内容，将每篇文本自动归入到一个或多个预先定义好的类别中，使文本能够按类别区分开来[52]。从数学的角度来说，文本分类是一个自动映射的过程，是将一个未知类别的文本映射到一个或多个已定义好的类别中。文本分类的基本流程如图2.1所示。

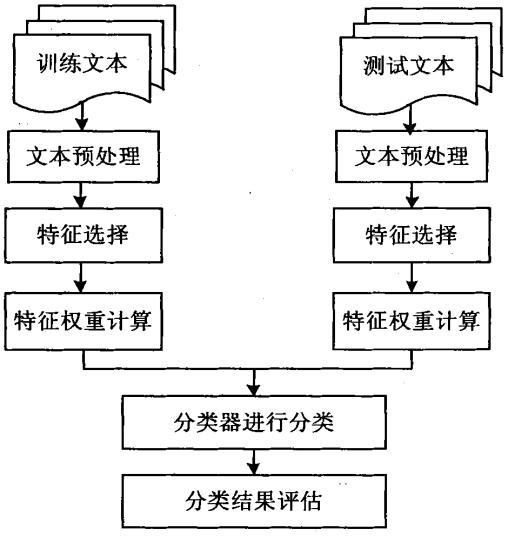

图 2.1 文本分类的基本流程

文本分类过程所涉及的技术很多，按流程来说可分为5个阶段:文本预处理、特征选择、权重计算、分类器的构造与分类、性能评估阶段。文本预处理阶段的主要任务是对原始语料集进行分词处理，如：去除停用词，低频词，标记信息（主要针对网页文本或其他的标记语言文本），单词的前后缀（主要针对英文），并将其表示为计算机可以处理的数据形式。分词后文本的特征词非常多，通常为几万甚至几十万，如果我们选择所有的特征词，那么计算的代价将非常昂贵，而且分类精确度将受到影响。因此，分词后的文本要进行特征选择，并将特征选择后的特征词进行加权处理，从而得到能被机器识别的文本向量。经过预处理后的文本才能进行分类。分类算法是文本分类的核心技术，文本分类的分类算法很多，本章将重点介绍几种有代表性的算法。评估阶段是对文本分类的效果进行评价，通常使用召回率(Recall)、准确率(Precision)、F1值及微平均和宏平均作为评价指标。

2.1.1 文本预处理

目前，计算机还远未达到能够直接“读懂”文本的水平，而且也不可能达到这个水平。所以计算机是无法直接处理任何给定的半结构化和无结构的文本的。要想计算机能够对给定的文本进行训练或者分类，必须对这些文本进行预处理，将它们转换成计算机能够处理的形式。文本预处理主要涉及的技术有：分词处理、停用词处理、文本表示。

(1) 分词处理

分词是进行文本处理的第一步。英文文本的单词与单词之间有空格做分隔符，而中文文本是以字为单位，由一系列连续的汉字构成的汉字串，只有句子间才有诸如逗号、分号、句号之类的标点作为分隔符。但字与字、词与词之间并没有明显的分隔符，因此要实现中文文本分类，必须先采用中文分词技术将连续字序列构成的中文文本分割成若干有意义且能代表该文本的离散的词序列。我们可以用图 2.2 来表示对一个中文文本分词系统的作用。

图 2.2 中文分词系统

目前存在的中文分词方法概括起来可分为三大类：基于字符串匹配的机械分词法、基于统计的分词方法和基于理解的分词方法。其中基于字符串匹配的机械分词法又有三种:由左到右的正向匹配方法；由右到左的逆向匹配方法；最少切分法(使每一句话中切分出的词的数最小)。基于统计的方法认为：词是稳定的字的组合，在上下文中，相邻的字共现的次数越多，就越可能构成一个词，因此字与字相邻共现的概率或频率可以较好的反映他们成词的可信度。基于理解的分词方法的基本思想是在分词的同时进行语义分析、句法分析，利用语义和句法信息来处理歧义现象，达到识别词的效果。它通常包括三个部分：分词子系统、语义句法子系统、总控制部分。

(2) 停用词处理

经过分词处理后的文本，并不是所有的词都对文本分类有帮助。相反，如果将那些对文本分类没有帮助的词语作为特征词，会降低文本分类的精确度，同时还会干扰对分类起作用的关键词。而且去除停用词可以在一定程度上减小特征词的数量。因此，我们需要对分类无帮助的词进行尽可能彻底的过滤处理。

停用词主要是指文本中存在的副词、助词、介词、连词、叹词、量词、数词、代词等。例如，代词：你、我、他；副词：都、很；助词：的、地、得；连词：和、及、或者；叹词：啊、呀。可以看出，这些词在不同文本中出现的词频是非

常大的，如果不剔除，将会对特征词选取造成很大的影响。

在剔除停用词前，一般需要先建立停用词表。目前，我国还没有一套权威的中文停用词表，研究者大多采用手工或者基于统计的停用词选择方法建立他们所需的停用词表。文献[53]在深入研究中文停用词表建立方法的基础上，提出了一种高效的停用词自动选取方法。停用词表建好后，在对应的停用词表中对分词后的每个词进行查找，如果找到，则说明该词是停用词，直接删除。

(3) 文本表示模型

目前，常用的文本表示模型有:布尔模型，概率模型，潜在语义索引模型，向量空间模型。其中使用最广、效果最好的是向量空间模型。下面对这些模型进行详细介绍。

1)布尔模型(Boolean Model)

布尔模型是一种简单而又常用的完全匹配检索模型，它的基本思想是：以关键词出现与否来刻画文本内容。在布尔模型中，每一个文本被表示成一个向量，向量的每一维就是文本的一个特征词，向量中的每一维的取值是一个二值变量，要么为 0，要么为 1。当某一维所对应的特征词在文本中出现时，它的权重值为 1，否则为 0。布尔模型的优点是简单且易于实现，检索速度快。但布尔模型只反映了特征词与文本相关与否，不能反映特征词对文本内容的重要程度，且这种完全匹配的形式容易造成漏检情况的发生，检索精确度不高。

2)概率模型(Probabilistic Model)

概率模型是信息检索领域一个较为成熟的模型，由 Kuhns 和 Maron 在 1960 年提出，并在一些系统应用中取得了非常不错的效果。它以数学理论中的概率排序理论为基础，综合考虑了词频、文本频率、文本长度等因素，把用户查询和文本按照一定的概率关系融合，并在概率空间上通过概率来衡量两个文本的语义相似度。

设文本 d 和用户查询 q 都可用特征词二值向量组$(a_1,a_2,\ldots,a_n)$表示，当特征词 $t_i \in d$ 时，有 $a_i=1$，否则为 $a_i=0$，其相关公式如下所示[21]：

$$sim(d,q) = \sum \log \frac{p_i(1-q_i)}{q_i(1-p_i)} \tag{2.1}$$

公式(2.1)中，

$$p_i = \frac{r_i}{r} \tag{2.2}$$

$$q_i = \frac{n-r_i}{N-r} \tag{2.3}$$

公式(2.2)和(2.3)中，r 表示文本集中与用户查询相关的文本数，r_i 表示 r 个相关文本中包含特征词 t_i 的文本数，N 表示训练文本集中的总文本数，n 表示训练

文本集中包含特征词 t_i 的文本数。

概率模型能够较好地解决文本信息相关性判断的不确定性和查询信息表示的模糊性问题。但是概率模型也存在着一些不足，其存储和计算资源的开销较高，且匹配结果将特征词分为相关和不相关两种状态，处理问题过于简单，未考虑特征词在文本中的出现概率。

3)潜在语义索引模型(Latent Semantic Index)

20 世纪 90 年代，Dumans 和 Funas 等提出了潜在语义索引模型，其基本思想是通过分析某个单词与该单词所处的上下文语境之间的关联关系，从而提取出语义结构，在语义的层次上检索文本，提高文本检索的性能。潜在语义索引模型主要是采用“奇异值分解”方法，去掉较小的奇异值，忽略单词之间细微差别，缩短具有类似用法的单词之间的距离，构造出一个新的语义空间。潜在语义索引模型能够有效地降低文本向量的维数，解决同义词，近义词和多义词的问题。

4)向量空间模型(Vector Space Model)

向量空间模型是由 G. Salton 等人在 1975 年提出的，是目前信息检索领域使用最广泛的文本表示模型[54]。它的基本思想是基于词袋法(bag of word)的思想，不考虑特征词在文本中出现的先后顺序及出现位置，认为特征词在文本中出现的顺序和位置对分类没有影响，文本被看成是由一系列互不相关的特征词组成。下面简单地介绍一下向量空间模型的一些基本概念。

a)文档(Document)：指文本或者文本中的一些段落、句群或者句子，一般指一篇文章。

b)特征词(Feature Term)：文档的内容一般是通过基本语义单位(字、词、短语等)来表示，这些基本的语义单位被统称为文本的特征词。由于词是构成中文文档的最小的语义单位，所以用词作为特征词能够充分表达中文文档的语义。

c)特征词权重(Term Weight)：不同的特征词在文本中的重要程度不同，重要程度则通过权重值大小来体现，对于一篇含有 n 个特征词的文档 d，其第 i 个特征词 t_i 的权重值用 w_i 来表示，则文本 d 可以用向量$(w_1,w_2,...,w_n)$来表示。

d)文本相似度(Similarity)：两个文本 d_1 和 d_2 的内容相关度，通常用文本相似度 $sim\ (d_1, d_2)$来进行度量。在向量空间模型中，文本的相似度可以通过欧氏距离，夹角的余弦值或者向量之间的内积来计算。

在文本分类领域，最经典的文本表示模型是向量空间模型(VSM)。该模型已成为最简便、高效的文本表示模型。所以本文采用向量空间模型来表示文本。

2.1.2 特征选择

经过中文分词和去除停用词处理后，一篇文本被划分成大量的词语，这些词语对刻画文本主题内容的重要程度不同，真正具有代表性的词语只有一小部分，

如果把所有的词语都作为文本特征词，会导致文本向量空间维数过高，对分类算法的时间复杂度和空间复杂度造成很大的影响。大量的实验结果表明，当向量空间的特征维数达到一定值时，可以实现很高的分类性能，之后随着特征维数的增加，分类性能反而会下降。因此，必须对表示文本的特征词进行选择。

特征选择是依据语料的统计结果，过滤文本中那些对文本分类几乎没有贡献的词条，从较低层次的特征集中构造出一个较高层次的新特征集。常用的文本特征选择算法有:频率统计、信息增益、互信息、CHI(Chi-Squared)统计、期望交叉熵[55]等。下面对这几种特征选择算法进行简单介绍。

(1) 频率统计

频率统计包括特征频率(Term Frequency,TF)和文本频率(Document Frequency,DF)两种。TF 是指特征在文档集中出现的次数，DF 是指整个训练数据集中出现某特征词的文本个数。

a)特征频率(TF)

TF 算法认为，特征词在某文本中出现的频率越高，就表明它与该文本的关系越密切，就越能代表该篇文本，词汇的频率对该文本以及该文本所属类别有重要的作用。由于最初的特征词集中存在大量的低频特征词，设定 TF 阈值能够有效地去除大量低频特征词，在很大程度上降低特征维度。因此，TF 主要用在文本标引时直接删除某些低频特征词。

b)文本频率(DF)

DF 是最简单的特征选择算法之一，它的基本思想为：当某特征词的文档频数过低，则表示该词不含或只含有较少的类别信息，甚至可能是噪音数据，应该删除；如果某特征词的文档频数过高，则表示该词对文本分类的贡献不大，也应该删除。DF 不需要依赖类别信息，是一种无监督的特征选择算法，常被集成到文本预处理中用来删除出现次数过少或过多的特征词以提高后续处理的效率。

(2) 信息增益

信息增益(Information Gain,IG)是一种信息熵的评估方法，定义为特征词在文本中出现前和出现后的信息熵的差值。信息增益算法的基本思想是:计算训练集中各特征项的信息增益，删除信息增益值小于阈值 T 的特征项，保留其余特征项作为特征词集。信息增益的计算公式如下[21]:

$$IG(t,C_i)=\sum_{i=1}^{m}(P(C_i|t)\log\frac{P(C_i|t)}{P(C_i)P(t)}+P(C_i,\bar{t})\log\frac{P(C_i,\bar{t})}{P(C_i)P(\bar{t})}) \tag{2.4}$$

公式(2.4)中，$P(C_i)$表示类别C_i在文本集中出现的概率，$P(t)$表示出现特征词 t 的文本在文本集中的概率，$P(\bar{t})$表示不出现特征词 t 的文本在文本集中的概率，$P(C_i|t)$表示包含特征词 t 的文本在类别 C_i 中出现的概率，$P(C_i|\bar{t})$表示不包含特征词 t 的文本在类别 C_i 中出现的概率，m 表示类别数。

信息增益通过特征词对文档分类提供的信息量多少，反映出特征词存在与否对类别预测的影响，在机器学习领域中已经得到了普遍的使用，但是该方法考虑了特征词未出现的情况，导致分类性能较低。

(3) 互信息

互信息(Mutual Information，MI)用于度量两个事件集合之间的相关性。互信息算法的思想为：如果特征词 t 在某一类别中出现的频率较大，而在其它类别中出现的频率较小，那么特征词 t 与这个类别的互信息就较大。互信息的计算方法如下[21]：

$$MI(t,C)=\log\frac{P(C\mid t)}{P(t)P(C)} \tag{2.5}$$

公式(2.5)中，$P(t)$表示包含特征词 t 的文本在文本集中的概率，$P(C)$表示类别 C 在文本集中出现的概率，$P(C\mid t)$表示包含特征词 t 的文本在类别 C 中出现的概率。从公式(2.5)可以看出，t 与 C 的相关度越大，$MI(t,C)$ 的值就会越大；如果 t 与 C 完全不相关，则 $MI(t,C)=0$。

对于多类别的数据集，特征词 t 对于整个数据集的互信息为特征词 t 对于各个类别的互信息的最大值，即：

$$MI(t)=\max_{i=1}^{m} MI(t,C_i) \tag{2.6}$$

互信息特征选择算法在统计语言模型中被广泛采用。从公式(2.6)可以看出，在特征词的 $P(C\mid t)$ 相等的情况下，稀有特征词比普通特征词的 MI 要大，造成了互信息的评估函数更倾向于选取稀有特征词，然而事实上，出现次数较多的词语往往比出现次数较少的词语对分类的贡献更大。

(4) CHI 统计

CHI(Chi-Squared)统计又称为 x^2 统计方法，该方法衡量一个词语与一个类之间的相关程度，它假定特征词 t 和类别 C 之间的关系类似于 x^2 分布。CHI 统计的理论基础是：在给定类别的文本中出现频率较高的特征词与在其他类别的文本中出现频率比较高的特征词，对确定文本的类别属性都起一定的作用。特征词对于某一类别的 CHI 统计值越高，它与该类相关性就越大，隐含的类别信息也就越多。其评估函数如公式(2.7)所示[21]：

$$CHI(t,C_i)=\frac{N\times(AD-BC)^2}{(A+C)\times(B+D)\times(A+B)\times(C+D)} \tag{2.7}$$

公式(2.7)中，A 表示类别 C_i 中出现词条 t 的文本数；B 表示其它类别中出现词条 t 的文本数；C 表示 C_i 类中未出现词条 t 的文本数；D 表示其它类别中未出现词条 t 的文档数；N 表示文本总数。

对于多类别问题，需要分别计算 t 在每个类别上的 CHI 值后取最大值，计算公式如下：

$$CHI(t)=\max_{i=1}^{m} CHI(t,C_i) \tag{2.8}$$

公式(2.8)中，m 表示类别数。在进行特征选择时，把 CHI 值大于特定阈值的特征词保留，小于特定阈值的特征词删除。CHI 统计法的优点在于分类效果比较稳定，准确率也比较高。但是，该方法进行统计时的计算量较大，时间和空间上的代价都比较大，对低频词效果不佳。

(5) 期望交叉熵

期望交叉熵(Expected Cross Entropy，ECE)反应了文本主题类的概率分布和在出现了某个词条的情况下文本主题类的概率分两者之间的距离。特征词 t 的交叉熵越大，则表示它对该文本主题类的分布影响就越大。交叉熵定义如下[56]：

$$ECE(t)=\sum_{i} P(C_i \mid t)\log\frac{P(C_i \mid t)}{P(C_i)} \tag{2.9}$$

公式(2.9)中，$P(C_i)$表示类别 C_i 在文本集中出现的概率，$P(C_i|t)$表示包含特征词 t 的文本在类别 C_i 中出现的概率。与信息增益不同的是，期望交叉熵方法没有考虑特征词未出现的情况，能够获得较高的分类精确度。

2.1.3 特征权重计算

经过文本预处理、特征选择之后，需要给选出的特征词赋予权值。特征词权值是通过特征权重计算来确定的。因此，特征加权算法的好坏直接影响到最终的分类结果[57]。目前，常用的特征词加权算法有布尔权重算法、词频加权算法、平方根权重法、基于熵的权重法和 TFIDF 权重法等[58]。

(1) 布尔权重法

布尔权重(Boolean)法是最简单的权重计算方法之一，权重值根据特征词在文本中是否出现来决定，如果特征词在文本中出现过一次或多次，则该特征词的权重记为 1，否则记为 0。其计算公式如公式(2.10)所示[59]。

$$w_{ik}=\begin{cases}1, tf_{ik}>=1\\ 0, tf_{ik}=0\end{cases} \tag{2.10}$$

布尔权重的计算方法形式简单，易于理解及实现，且速度较快。但是从公式(2.10)，我们可以看出，由于它只限于用 0 或 1 来表示特征词的权值，所以在分类时该权重法求得的权重值提供的信息量不足，不能很好的体现特征词在文本中的重要程度。

(2) 词频加权法

词频(Term Frequency ,TF)加权法是一种最直观的加权算法，它以特征词在文

本中出现的次数作为权重值，其基本思想是：特征词在文本中出现的次数越多，那么它对文本分类的贡献就越大。其计算方法如式(2.11)[56]。

$$w_{ik} = tf_{ik} \tag{2.11}$$

公式(2.11)中，tf_{ik} 为特征词 t_k 在文本 d_i 中出现的频率。词频加权算法易于实现，但是其计算的权重值容易受到文本长度的影响。因为特征词出现频率很可能会随文本长度的增长而变大，所以该算法很可能为那些在文本中出现次数较多，但对分类没有什么贡献的特征词赋予较高的权重值，而给那些对分类有很大帮助出现次数相对较少的特征词赋予较低的权重值。

(3) 平方根权重法

平方根权重法以特征词在文本中出现频数的平方根作为该特征词的权重。若该特征词在文本中不出现，则其权重的值为 0。其计算方法如式(2.12)[59]。

$$w_{ik} = \sqrt{tf_{ik}} \tag{2.12}$$

平方根权重算法认为：用特征词出现频率作为特征词的权重值会导致出现次数较多的特征词的权重值过大，在计算特征词权重时以特征词出现频率的平方根代替特征词出现的频率。

(4) 熵权重法

熵权重算法(Entropy Weighting)是最复杂的特征权重算法之一，它是在信息论的基础上提出来的，是一种非常有效的特征权重计算方法。在熵权重计算方法中，特征词 t_k 在文本 d_i 的权重计算公式如(2.13)所示[60]。

$$w_{ik} = \log(tf_{ik} + 1.0)\left(1 + \frac{1}{\log(N)}\sum_{j=1}^{N}\left[\frac{tf_{jk}}{n_k}\log(\frac{tf_{jk}}{n_k})\right]\right) \tag{2.13}$$

公式(2.13)中，tf_{ik} 为特征词 t_k 在文本 d_i 中出现的频率，N 代表文本集中文本总数，n_k 表示特征词 t_k 在所有文本中出现的总次数，$\frac{1}{\log(N)}\sum_{j=1}^{N}\left[\frac{tf_{jk}}{n_k}\log(\frac{tf_{jk}}{n_k})\right]$ 为 t_k 平均熵。当特征词 t_k 在所有文本中均匀分布时，t_k 的权重值为 1，若特征词 t_k 仅在某一篇文本中出现时，其权重值为 0。

(5) TFIDF 权重法

TFIDF 是目前使用最广泛的加权算法， 关于 TFIDF 算法的介绍与分析将在本文第三章的 3.1，3.2 节进行详细阐述。

2.1.4 文本分类算法

文本分类算法是文本分类系统的核心，目前常用的文本分类方法包括 Rocchio

算法、朴素贝叶斯算法、神经网络算法、KNN 算法和支持向量机等。下面我们将对这几种主流分类算法进行详细的介绍。

(1) Rocchio 算法

Rocchio 在 1971 年首次提出了 Rocchio 算法，被广泛地应用到文本分类领域。其基本思想是：对于某个特定的类，某些词出现属于这个类的可能性就会增加，而另一些词出现属于这个类的可能性就会降低，累计这些正面和负面影响因素，最后由文本分离出的词向量可以得到对于每个类的一个打分，打分越高则属于该类的可能性就越大。

Rocchio算法在训练过程中通过训练集文本为每个文本类创造一个原型向量。原型向量的创造方法为：给定某一文本类 C_i，把训练集中属于类 C_i 的文本向量的分量用正数来表示，把所有不属于类 C_i 的文本向量的分量用负数表示，那么 C_i 原型向量就是所有向量的向量和。

该算法在分类阶段，通过计算每个文本向量与每个文本类的原型向量的距离的大小，并以此距离值来判别文本的所属类别。

Rocchio 算法比较容易实现，分类的速度也较快，但是由于类别的原型向量是正、负样本向量的一种加权平均值，对于那些属于类别原型向量边缘的文本很难区分。

(2) 朴素贝叶斯算法

朴素贝叶斯(NaiveBayes,NB)算法将概率论模型应用到文本分类中，是一种简单有效的分类方法。其基本思想是将训练集中的每个文本看作独立的词语集合，通过贝叶斯全概率公式得到每个词语在不同类中的概率，通过先验概率和类别的条件概率来计算文本 d 对类别 C 的后验概率，以此实现对文本 d 所属类别的判断。根据朴素贝叶斯定理，类 C_i 的后验概率 $P(C_i \mid d_j)$ 计算公式如下[21]：

$$P(C_i \mid d_j) = \frac{P(d_j \mid C_i)P(C_i)}{P(d_j)} \tag{2.14}$$

公式(2.14)中，$P(C_i)$为类别 C_i 的先验概率，其初始定义为类别 C_i 中的文本数 N_i 除以训练集的文本总数 N，$P(d_j \mid C_i)$ 为文本 d_j 属于类别 C_i 的条件概率，其计算公式如下：

$$P(d_j \mid C_i) = \prod_{k=1}^{n} p(w_{jk} \mid C_i) \tag{2.15}$$

公式(2.15)中，n 为文本向量的维数。由于 $P(d_j)$ 对于所有类为常数，所以公式(2.14)可以简化为：

$$P(C_i \mid d_j) = P(d_j \mid C_i)P(C_i) \tag{2.16}$$

对于每一个类别 $C_i(i = 1, 2, \ldots, m)$，都可以计算出 $P(C_i \mid d_j)$，文本的所属类别为后验概率值最大的那个类别。

朴素贝叶斯算法的优点在于易于实现和计算，对数值数据和文本数据的分类效果都较好；但由于算法的前提是假设各特征词之间完全独立。当数据集不满足这种独立性假设时，分类的准确率较低。

(3) 神经网络算法

神经网络(Neural Network)是模仿人脑神经系统的组织结构特性和工作机制而建立的一类非线性预测模型。这种网络一般由多个神经元构成，每个神经元可以有多个连接通路来输入，但只有唯一的一个输出，每个连接通路对应一个连接权系数且该系数是可以调整的。这些互连的神经元构成自适应、非线性的动态系统。常用的神经网络模型有多层感知器、反传网络、自组织映射网络等。图 2.3 为三层 BP(Back Propagation)结构图。

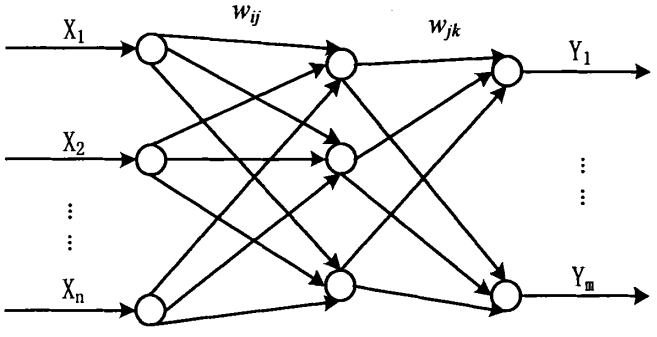

图 2.3 三层 BP 神经网络结构

神经网络算法有很多优点，例如：具有很好的适应性、容错性和鲁棒性，可以存储大量的数据，可并行处理，运行速度快。然而，由于不同结构的神经网络的特征差异比较大，因此对特定结构神经网络的特征研究仍然有待提高和完善。

(4) KNN 算法

KNN 方法是一种基于实例学习的方法，由 Cover 和 Hart 于 1967 年提出，关于 KNN 算法的介绍及分析将在第五章详细阐述。

(5) 支持向量机

支持向量机 SVM(Support Vector Machines)是建立在统计学的 VC 维理论和结构风险最小原理基础上的一种机器学习方法，由 Vapnik 等人率先提出。其基本思想是：利用训练样本集，通过定义的内积函数将输入向量映射到一个更高维的特征空间中，然后在这个空间上面求取最优分类函数(最佳分类超平面)，最后将测试文本向量代入分类函数，根据计算结果来确定测试文本的类别。SVM 的关键在于核函数，采用不同的核函数将得到不同的 SVM 算法。核函数不仅很好地解决

了支持向量机中的非线性分类问题，同样有效地解决了在文本分类中特征选择的非线性问题等。

SVM 是由线性可分情况下的最优分类面发展而来的，其基本概况如图 2.4 来说明。

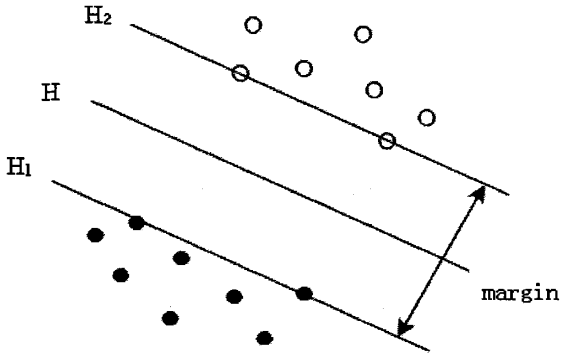

图 2.4 线性可分情况下的最优分类组

图 2.4 中，实心和虚心圆点分别代表两个不同类别的样本，H 为两个类的分类线，H_1， H_2 分别为两个类中离分类线最近的样本且平行于 H 的直线，H_1 和 H_2 之间的距离叫做分类间隔(margin)。所谓最优分类线就是能够将两类样本正确分开(训练错误率为 0)，且使得分类间隔最大的线。

SVM 与传统的机器学习方法相比，具有模型泛化性能好、小样本学习能力强的特点。但是该算法本身的特点决定了算法存在计算量较大、训练时间长的缺陷，特别是当训练集增大时，其计算量也呈线性增长。

2.1.5 性能评估

自动文本分类从本质上说是一个文本映射的过程，它将测试文本映射到类别集合中的一个或多个类别中去。因此，评价文本分类器的性能可以从映射的速度和映射的准确度两个方面进行。映射的速度取决于分类算法的复杂度，而映射的准确度通过将文本分类结果与人工分类（假设人工分类完全正确）结果相比较，其结果越接近人工分类结果则分类的准确度就越高。因此，使文本分类过程更快速，更准确是文本分类系统的终极目标。

文本分类的速度一般采用分类时间来衡量，文本分类的准确程度常用的评估指标有查准率(P)、查全率(R)、F1 值和宏平均、微平均。为了后面介绍方便，先给出各类别的分类结果判定表。

表 2.1 分类结果判定表

	实际属于该类	实际不属于该类
判定属于该类	a	b
判定不属于该类	c	d

(1) 查准率、查全率和 F1 值

查准率(P)是指利用分类器正确分类的文本数与实际被分到该类的文本数的比例。其计算公式如下[56]：

$$P = \frac{a}{a + b} \tag{2.17}$$

查全率(R)是指利用分类器正确分类的文本数与测试集中该类文本总数的比例。

$$R = \frac{a}{a + c} \tag{2.18}$$

F1 测试值是用来综合评价查准率与查全率的新的评估指标。查准率(P)和查全率(R)这两个指标是互补的，单独提高其中一个指标的值，往往会降低另一个指标的值。一个好的自动文本分类系统应当同时兼顾两者，因此，F1 测试值被提出。其数学公式(2.19)如下：

$$F1 = \frac{2PR}{P + R} \tag{2.19}$$

(2) 微平均和宏平均

前面介绍的查准率(P)、查全率(R)以及 F1 测试值都是对分类器在某一类别上的分类结果进行评估，具有一定的局部性。如何对整个数据集的所有类别的分类结果进行评估，就涉及到下面将要介绍的两种综合评价的标准：微观平均值和宏观平均值。

微观平均值（简称微平均），其计算过程是：先求得各个类别中被正确分类和错误分类的文本数总和，然后再分别计算相应的微平均查准率、微平均查全率和微平均 F1，其公式为[56]：

$$Micro\_P = \frac{\sum_{i=1}^{m} a_i}{\sum_{i=1}^{m} a_i + \sum_{i=1}^{m} b_i} \tag{2.20}$$

$$Micro\_R = \frac{\sum_{i=1}^{m} a_i}{\sum_{i=1}^{m} a_i + \sum_{i=1}^{m} c_i} \tag{2.21}$$

$$Micro\_F1 = \frac{2Micro\_P \times Micro\_R}{Micro\_P + Micro\_R} \tag{2.22}$$

宏观平均值（简称宏平均），其计算过程为：先分别计算文本集中各个类别的查准率、查全率和 F1 值，再分别求得所有类别的查准率、查全率和 F1 值的平均值，其计算公式为[56]：

$$Macro\_P = \frac{\sum_{i=1}^{m} P_i}{m} \tag{2.23}$$

$$Macro\_R = \frac{\sum_{i=1}^{m} R_i}{m} \tag{2.24}$$

$$Macro\_F1 = \frac{2Macro\_P \times Macro\_R}{Macro\_P + Macro\_R} \tag{2.25}$$

微平均和宏平均的区别在于:微平均将文本集中的每个文本视为同等重要，强调小类对整体的影响，而宏平均将文本集中的各个类别视为同等重要，强调大类对整体的影响。

2.2 MapRedcue 概述

随着互联网技术的高速发展，数据海量涌现导致数据处理规模急剧增长，近些年来，高性能计算、并行计算得到了前所未有的发展。在这种背景下，Google 公司于 2004 年提出了一种新的分布式并行编程模型—MapReduce，该模型在很大程度上降低了并行编程的难度，用户在利用 MapReduce 进行大规模数据处理时，可以将精力集中在 Map 和 Reduce 函数的设计和实现上，而不必考虑并行计算中的一些诸如分布式文件系统、任务调度、系统容错、节点通信等复杂问题[61]。因此，MapReduce 已成为云计算平台的主流编程模型，受到国内外研究者的高度关注[62]。

2.2.1 MapReduce 编程模型

MapReduce 模型的主要思想是：将一个大规模数据处理作业分解成大量的可独立运行的 Map 任务，每个 Map 任务交由集群中的 1 个节点（通常为一台计算机）去执行，生成某种格式的中间结果，然后将这些中间结果交给若干个节点（通

常少于 Map 任务节点数）执行 Reduce 任务，得到最终结果。该模型的核心是 Map 和 Reduce 函数的设计，其中 Map 函数负责把输入键值对<key1,value1>转化为中间结果<key2,value2>，Reduce 函数负责将具有相同 key2 值的<key2,value2>进行处理，输出最终结果<key3,value3>，其中数据的转换如表 2.2 所示。

表 2.2 Map 和 Reduce 数据转换

函数	输入	输出	说明
Map	<key1,value1>	List<key2,value2>	1)将小数据集解析成一批<key,value>对，输入 Map 中进行处理 2)每一个输入的<key1,value1>会输出一批中间结果<key2,value2>
Reduce	<key2,List(value2)>	<key3,value3>	1)输入的中间结果<key2,List(value2)>表示一批属于同一个 key2 的 value

2.2.2 MapReduce 的执行过程

MapReduce框架的一大特点是自行处理并行计算中的一些复杂细节，给程序开发者提供了一个十分简洁的并行编程接口。Hadoop MapReduce是运行于HDFS(Hadoop Distributed File System:由一个主节点NameNode和一组从节点DateNode构成)之上的一种并行运算模型，是Google MapReduce的开源实现。最上层包括4个独立的实体：Client（客户）、Jobtracker（作业服务器）、Tasktracker（任务服务器） 和 HDFS。其中，JobTracker负责将作业拆分成多个任务并分发给多个TaskTracker，TaskTracker则负责执行划分后的任务。此外，作业的原始数据输入和最终结果输出则依赖于HDFS。图2.5说明了Hadoop MapReduce框架中作业执行的整个过程，其描述如下：

(1) 作业提交阶段：作业的输入文件被存储在HDFS中，Client程序通过调用MapReduce框架的InputFormat类将输入文件切分成M个数据块(split，默认大小为64MB或128MB)，然后再调用JobClient的submitJob方法向JobTracker提交Job。

(2) 作业划分和分配阶段：JobTracker创建M个Map监控对象和R个Reduce监控对象，M等于数据块的数目，R的值由用户设定。然后，JobTracker通过分析集群内所有TaskTracker的状态，将任务对象（即：MapTask和ReduceTask）提交到空闲的TaskTracker上执行。

(3) Map阶段：每个执行MapTask的TaskTracker从HDFS中读取自己所对应的数据块，并对数据块中的每条数据进行Map操作：根据Map函数将输入数据<key1,value1>转化成中间键值对< key2,value2>。

(4) Shuffle&Sort阶段：Shuffle主要完成混排交换数据，即把中间结果中具有相同key2值的数据尽可能地汇集到同一节点上；而在Sort根据key2的值对Reduce的输入进行分组排序。一般来说，Shuffle和Sort是同时进行的，最后尽可能将中间结果中具有相同key2的数据存储在DataNode的同一个数据分区(Partition)中。

(5) Reduce阶段：Reduce函数遍历对应的中间数据，对每个传入的<key2,value2>键值对执行用户自定义的Reduce函数，输出新的< key3，value3>键值对。

(6) 作业完成：所有的MapTask和ReduceTask都执行完毕后，作业的输出持久保存在HDFS中。

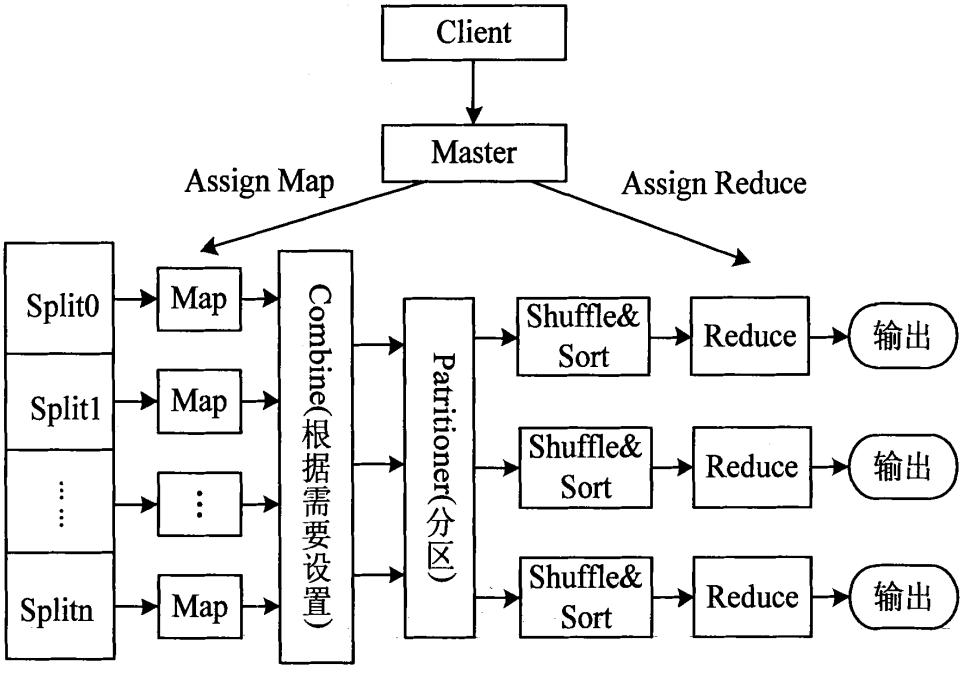

图 2.5 MapReduce 框架的执行过程

2.2.3 MapReduce 模型实现

受Google公司提出的MapReduce启发，国内外研究者在不同的平台上实现了MapReduce，其中最具代表性的包括：Apache的Hadoop，Nokia 研究中心的Disco，Stanford University的Phoenix和香港科技大学的Mars。不同的实现平台具有不同的特点和功能侧重点，几种主要实现平台的比较如表2.3所示。

表 2.3 MapReduce 的实现平台比较

平台名称	难易程度	普及程度	体系结构	功能特点
Hadoop	简单	较高	大型分布式集群上MapReduce 的开源实现框架	充分利用集群的优势进行并行计算，适于开发、运行和处理大规模数据
Disco	简单	较低	分布式移动平台实现的轻量级MapReduce框架	可创建和查询大量<key/value>的索引，适用于大规模数据的分析
Phoenix	简单	较低	共享内存平台上MapReduce的实现	利用共享内存缓冲区实现通信，减小节点间的通讯开销，适用于数据密集型任务处理
Mars	复杂	低	基于图形处理器的MapReduce实现	利用大量 GPU 线程来完成 Map 和 Reduce 操作，适用于图形处理

(1) Hadoop

Hadoop 是由 Apache 公司用 Java 语言开发，并由 Yahoo 资助的云计算平台架构。Hadoop 主要包括两个子项目：Hadoop MapReduce 和 Hadoop HDFS。

Hadoop MapReduce 是一个通过计算机集群对大数据集进行分布式计算的软件框架，由一个 JobTracker 和若干个 TaskTracker 组成。其中，JobTracker 负责作业的划分和分配；TaskTracker 负责划分后任务的执行。

Hadoop HDFS 是一个用于提供高速获取应用程序数据集的分布式文件系统，由一个主节点（NameNode）和多个从节点(DateNode)组成。在 HDFS 内部，NameNode 提供元数据服务，DataNode 负责存储数据。

Hadoop框架的分布式计算和存储都采用的是主/从结构，JobTracker和NameNode的守护进程都运行在主节点上，TaskTracker和DataNode的守护进程都运行在从节点上。为了方便数据的本地计算，TaskTracker一定是运行在DataNode上的，但JobTracker和NameNode则不一定运行在同一台机器上。图2.6是一个典型的Hadoop集群拓扑结构图。

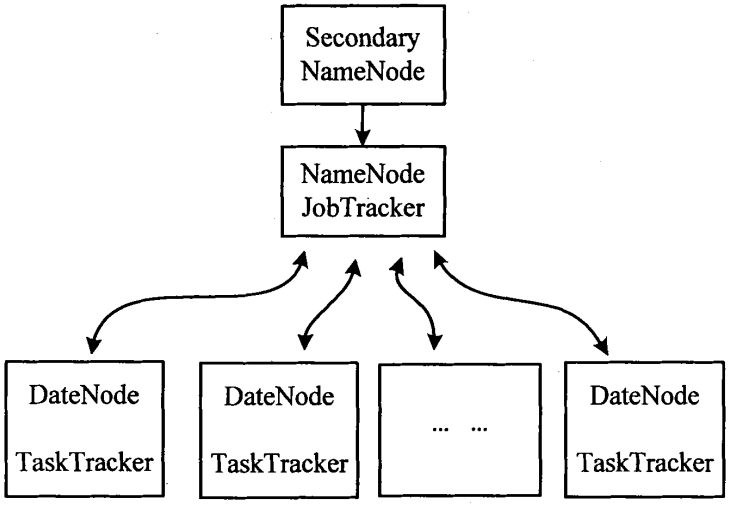

图 2.6 Hadoop 集群拓扑结构

Hadoop 是目前应用最广泛的分布式并行计算编程框架，具有诸多的优点。其最大的优点体现在它的易用性：用户可以在不了解底层分布的情况下，快速地开发出高效的并行应用程序。此外，该平台还具有良好的可扩展性，高效性，可靠性及经济性。但因其执行过程需要将各个 Map 任务节点的中间结果文件传送给 Reduce 任务节点，加大了网络传输开销，因此，不适合用于处理计算时会产生非常大的中间结果文件的程序。

(2) Disco

Disco 由 Nokia 研究中心研发，其核心程序采用 Erlang 语言开发，外部编程接口采用 Python 语言开发。Disco 平台主要由 DDFS(Disco Distributed File System) 和 MapReduce 框架两部分组成。其中，DDFS 是由 1 个 Master 节点和多个 Storage 节点组成，Disco MapReduces 是由一个 Master 服务器和多个 Work 服务器组成。Disco 的总体架构如图 2.7 所示。

Disco MapReduce中，Master和Worker 之间采用轮询机制通讯，使用HTTP方式进行数据传输，Master 服务器主要负责任务分配和调度，并保存相关的原始输入数据、中间结果和最终输出结果，Worker节点则负责Map和Reduce任务的执行。由于Disco的节点间采用轮询机制，时间的设置恰当与否将直接影响到程序的执行效率[63]。目前，Disco主要被应用于大规模数据分析。

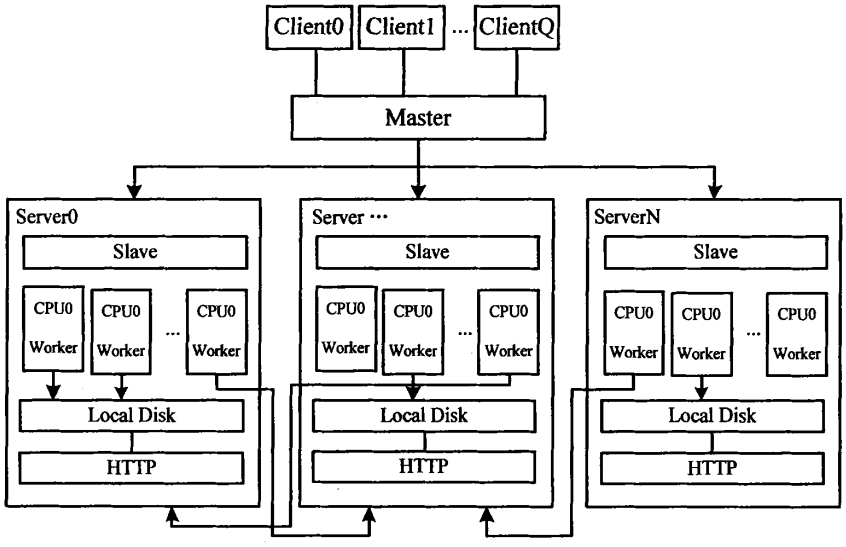

图 2.7 Disco 总体架构

(3) Phoenix

Phoenix是一个在共享内存平台上实现的MapReduce框架，由一个运行时系统和一个对外开放的API接口组成。其运行时系统主要负责管理线程的创建、数据的分区、动态任务的调度、节点容错等。Phoenix MapReduce的执行过程如下：输入数据由分割器分割成若干个数据块后传送给各个Map线程，Map任务执行完毕的中间结果按key值分区，等到所有的Map任务都执行完毕后再动态地分配Reduce任务，最后合并最终的输出结果并存放于内存缓冲区中。应用程序运行时，Phoenix运行时系统的执行过程如图2.8所示。

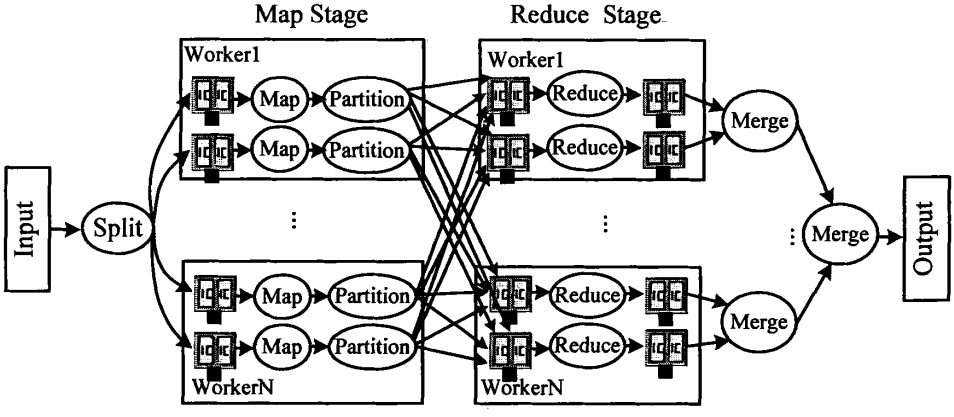

图 2.8 Phoenix 工作流程

Phoenix中的Map和Reduce任务采用线程实现，任务间的通信利用共享内存缓冲区实现，避免了因数据拷贝产生的通信开销，适于处理数据密集型的任务。但

Phoenix也存在缺乏高效的错误发现机制、无法自动执行迭代计算等不足。

(4) Mars

Mars是由香港科技大学的He Bingsheng等在GPU(Graphic Processing Units)上设计并实现的MapReduce系统[64]。与其它MapReduce框架类似，Mars也包括了Map和Reduce两个阶段，GPU上Mars的基本工作流程如图2.9所示。

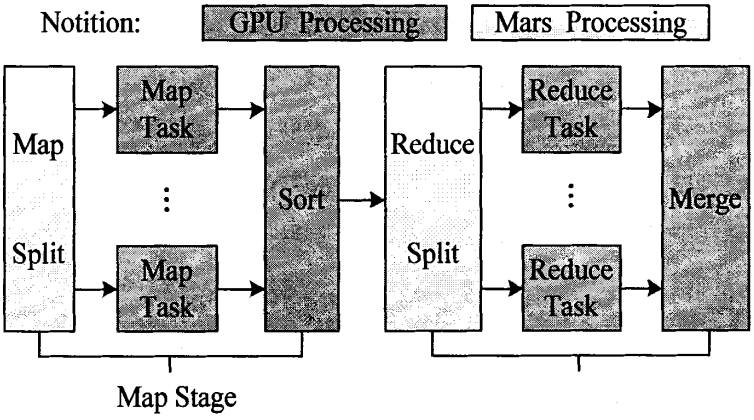

图 2.9 GPU 上 Mars 的工作流程

在每个阶段开始之前，Mars对线程配置进行初始化，包括：GPU中线程组的个数和各个线程组中线程的数量。在运行时，Mars在GPU内创建大量的线程，然后将任务均匀分配给各个线程，每个线程负责一个Map或Reduce任务的执行。由于目前GPU线程并不支持动态调度，因而Mars也未能实现动态的负载均衡。另外，Mars的编程接口是专门为图形处理而设计的，对一般用户来说，编程难度较大。

2.3 小结

本章首先对文本分类过程中一些关键技术进行了全面的介绍，主要包括文本预处理、特征选择、特征权重计算、文本分类算法、性能评价。然后详细介绍了MapReduce并行编程模型及其主要思想、执行过程和一些主要的模型实现。

第3章 TFIDF 加权算法改进

特征权重计算是在文本预处理、特征选择之后的环节。TFIDF 是使用最广泛的特征加权算法之一，本章将对 TFIDF 算法进行简单介绍，并对其存在的缺陷进行全面分析，最后针对其缺陷，给出本文改进的 TFIDF 算法：结合语义、信息熵、信息增益的 S-TFIDFIGE 算法。

3.1 TFIDF 算法简介

1973 年，Salton 在文献[65]中首次提出了 TFIDF (Term Frequency and Inverse Documentation Frequency)算法， TFIDF 体现的主要思想为：一个特征词在特定的文本中出现的频率越高，说明它的类别区分能力越强，应该赋予较高的权值；一个特征词在文本集中出现的范围越广，说明它的类别区分能力越低，应该赋予较低的权值。其计算公式为[66]：

$$w_{ik} = tf_{ik} \times idf_{ik} \tag{3.1}$$

公式(3.1)中，tf_{ik} 为特征词 t_k 在文本 d_i 中出现的频率；idf_{ik} 表示特征词 t_k 的逆文档频率，它认为特征词在文本集中出现的范围越广，该特征词就越不重要，通常用下面的公式计算：

$$idf_{ik} = \log(\frac{N}{n_k} + \alpha) \tag{3.2}$$

公式(3.2)中，N 表示文本集中的总文本数；n_k 表示包含特征词 t_k 的文本总数，α 是一个可调节的参数，通常取 0.01。考虑到文本长度对特征词的权重值的影响，通常会用公式(3.3)对权重值进行归一化处理，将权重值规范在[0,1]间。

$$w_{ik} = tf_{ik} \times idf_{ik} = \frac{tf_{ik} \times \log(N/n_k + 0.01)}{\sqrt{\sum_{j=1}^{N}\left(tf_{jk}^{\ 2} \times \log(N/n_k + 0.01)^2\right)}} \tag{3.3}$$

TFIDF 是目前使用最广泛的加权算法，该算法的优点在于算法思想简单，容易实现，既考虑了特征词的词频，又考虑了特征词在整个文本集中的分布。

3.2 TFIDF 缺陷分析

TFIDF 算法被广泛应用于文本特征权重计算，它以词频 TF(Term Frequency)和反文档频数 IDF(Inverse Documentation Frequency)的乘积作为文档特征词的权

值。虽然 TFIDF 特征加权算法拥有诸多的优点，但就 TFIDF 算法本身来说，也存在许多需要改进的地方。

首先，IDF 反映的主要思想是：如果包含特征词 t 的文本数越少，则说明特征词 t 的类别区分能力越强，IDF 的结果就越大。反之，则说明特征词 t 的类别区分能力越小，IDF 的结果就越小。从 IDF 的计算公式(3.2)可以看出，IDF 仅仅考虑了特征词在整个文本集中的分布情况，并没有考虑特征词在各个类之间以及类别内部的分布情况。

考察特征词的类间分布情况：如果类 C_i 中包含特征词 t 的文本数为 m，而其它类包含特征词 t 的文本数为 n，则包含特征词 t 的总文本数 $n_t = m + n$，当 m 值大的时候，n_t 也大，按照公式(3.2)得到的 IDF 的值会变小，得到的权重值偏小，则表示该特征词 t 类别区分能力不强。但是实际上，m 值越大，词语 t 在类 C_i 的文本中出现的越频繁，说明特征词 t 能够很好地代表类 C_i 类中的文本特征，应该赋予较高的权重。另一方面，虽然包含 t 的文本数 m 较小，但如果 t 均匀分布在各个类间，这样的特征词的类别区分能力很弱，应该赋予较小的权重，可按照传统的 IDF 公式计算的 IDF 值却很大，最终得到的权重值就偏大。

进一步分析特征词在类内的分布：如果特征词 t 在类 C_i 的各个文本中均匀分布，则表示 t 可以很好的代表该类别的文本特征，具有较强的分类能力，应该赋予较高的权重。相反，如果一个特征词只在类 C_i 中的几篇文本中出现，则表明该特征词的代表性较差，应该赋予较低的权重。但 IDF 的计算公式中无法反映和处理该情况。

其次，TFIDF 加权算法在计算过程中把每个特征词视为孤立的，毫不相干的。但实际上，特征词之间是存在着语义关联的。比如语义十分相近的两个词“文本”跟“文档”在传统的 TFIDF 算法中被视为两个毫不相干的特征词。假如有两篇文章非常相似，其中有一篇文章里频繁的用到“文本”这个词，而另一篇文章中频繁的使用“文档”这个词。那么由传统的 TFIDF 方法得到的结果可能极为不准确。

再次，从 TF*IDF 的计算公式中可以看出词频 TF 是影响该词语在一篇文本中权重的一个直接因素。而文本长度是影响词频 TF 的重要因素，一般来说一篇长文本中某词语出现的频率要比一篇短文本的高，这将导致短文本中的特征词的权重普遍偏小，影响到最终的分类结果。

此外，TFIDF 加权算法还存在未考虑特征词的出现位置、特征词的长度、数据集偏斜等问题。

3.3 本文改进算法

在以上的改进算法中，引入信息增益与信息熵的 TFIDFIGE 算法取得了不错的分类效果[32]。但该算法忽略了特征词之间的语义联系，没有考虑与特征词语义

相近的词语在文本集中、类间、类内的分布情况，影响了权值计算的精确性。针对这一问题，本文首先对特征词进行语义分析，将与特征词相似度大于某一阈值的词语归到该特征词的相似特征词组中，再进一步分析 IDF、信息熵与信息增益，提出了一种新的改进算法：S-TFIDFIGE。

3.3.1 相关知识

在对本文改进算法 S-TFIDFIGE 进行详细介绍之前，先介绍一下该算法中涉及到的几个概念的定义。

(1)信息熵

1850年，德国物理学家Rudolf Clausius首次提出了熵的概念，它用来表示一种能量在空间中分布的均匀程度，该能量分布得越均匀，熵的值就越大。1948年，Claude Elwood Shannon发表了《通信的数学原理》，将熵的概念引入到信息论中。在信息论中，信息的基本作用就是消除事物的不确定性，因此信息携带信息量的大小，可以通过事物不确定性减少的多少来表示。熵在信息论中的定义如下：

定义 3.1：设存在 n 个相同概率的消息，每个消息的概率 $p(x)=1/n$，则一个消息传递的信息量为：

$$I(X)=lb\frac{1}{p(X)}=-lb\big(p(X)\big)=lb\big(n\big) \tag{3.4}$$

定义 3.2：对于给定的概率分布 $P=(p_1, p_2, ..., p_n)$，则该分布传递的信息量即 P 的熵为：

$$I(P)=-\big(p_1*lbp_1+p_2*lbp_2+...+p_n*lbp_n\big)=-\sum_{k=1}^{n}p_k*lbp_k \tag{3.5}$$

当 $p_k=0$ 时，$p_k*lbp_k=0$。当 $p_1=p_2=...=p_n$ 时，$I(P)=1$。熵的公式表明，概率分布越均匀，其所携带的信息量越大。

(2)信息增益

在文本分类中，信息增益用来衡量特征词为分类系统带来信息的多少，带来的信息越多，该特征词就越重要。对一个特征词来说，分类系统在观察到它之前和观察到它后信息量将发生变化，而这前后信息量的差值就是这个特征词给分类系统带来的信息量，这里的信息量，就是前面介绍的信息熵。信息增益的详细定义如下。

定义 3.3：假设 $I(X)$表示一个随机文档落入某个类的概率空间的熵，$I(X|y)$表示观察到 y 后，文档落入某个类的概率的空间熵，即观察到 y 后对分类的不确定程度。这种不确定程度减少的量就是信息增益，表示为：

$$IG(X/y)=I(X)-I(X/y) \tag{3.6}$$

由定义 3.3 可知信息增益表示词语 y 对分类的作用，即：词语 y 所能提供的分类信息量。

3.3.2 词语相似度计算

词语相似度在不同的应用中有不同的含义，本文所说的词语相似度指两个词语在文本中可以互相替换的程度，它是一个[0,1]之间的数值，相似度越大，表示两个词语可替换的程度越大。本文中词语相似度的计算是以“知网”为语义本体的。“概念”与“义原”是知网中的两个最基本的概念，概念是对词汇语义的一种描述，每一个词汇可以表达为几个概念；义原是描述概念的最基本单位。文献[67] 对知网的结构做了比较详尽的描述，下面给出基于《知网》词汇语义相似度计算的具体方法。

词语 $W_1(S_{11},S_{12},\ldots,S_{1n})$ 有 n 个概念（义项），$W_2(S_{21},S_{22},\ldots,S_{2m})$ 有 m 个概念（义项），W_1 和 W_2 的相似度为各个概念的相似度之最大值，计算公式如下：

$$Sim(W_1,W_2)=\max_{i=1\ldots n,j=1\ldots m} Sim(S_{1i},S_{2j}) \tag{3.7}$$

公式(3.7)中 $Sim(S_{1i},S_{2j})$ 代表两个概念相似度，其计算公式如下：

$$Sim(S_1,S_2)=\sum_{i=1}^{4}\beta_i\prod_{j=1}^{i}Sim_j(S_1,S_2) \tag{3.8}$$

公式(3.8)中，$Sim_1(S_1,S_2)$ 代表两个概念的第一独立义原描述式的相似度，$Sim_2(S_1,S_2)$ 代表两个概念的除第一独立义原以外的其他独立义原描述式的相似度，$Sim_3(S_1,S_2)$ 代表两个概念的关系义原描述式的相似度，$Sim_4(S_1,S_2)$ 代表两个概念的符号义原描述式的相似度。$\beta_i(1\le i\le 4)$ 是可调节的参数，且有：$\beta_1+\beta_2+\beta_3+\beta_4=1$，$\beta_1\ge\beta_2\ge\beta_3\ge\beta_4$，通常 β_1 的取值在 0.5 以上。所有的概念最终都是用义原来表示，两个义原相似度计算公式如下：

$$Sim(p_1,p_2)=\frac{\alpha}{d+\alpha} \tag{3.9}$$

公式(3.9)中，p_1，p_2 表示两个义原，d 代表 p_1 和 p_2 在义原层次体中的路径长度，是一个正整数，α 是一个可调节的参数。

3.3.3 S-TFIDFIGE 算法

针对传统TFIDF算法忽略了特征词在类内、类间的分布情况，TFIDFIGE算法从信息论的角度出发，把信息增益引入文本集合的类别间，并把信息熵引入文本集合的各个类内部，根据训练文本集合的类别信息熵和文本类别中特征词的条件熵之间信息量的增益值以及特征词在类内的信息分布熵值来确定该特征词在文本

分类中所能提供的信息量，TFIDFIGE算法权重，计算公式如下[32]：

$$w_{ik} = tf_{ik}(d_i) \times idf(t_k) \times IG(C, t_k) \times E_{ic}(t_k) \tag{3.10}$$

其中，

$$\begin{aligned} IG(C, t_k) &= E(C) - E(C / t_k) \\ &= -\sum_{i=1}^{m} p(C_i) \times lb(p(C_i)) + \sum_{i=1}^{m} p(C_i / t_k) \end{aligned} \tag{3.11}$$

$$E_{ic}(t_k) = -\sum_{j=1}^{n} \frac{tf(t_k, d_j)}{tf(t_k, C_i)} lb \frac{f(t_k, d_j)}{tf(t_k, C_i)} \tag{3.12}$$

公式(3.10)(3.11)(3.12)中，w_{ik}为特征词t_k在文本d_i中权重，$tf_{ik}(d_i)$为t_k在d_i中出现的次数，$idf(t_k)$为t_k的反文本频数，$IG(C, t_k)$表示t_k的类间分布信息增益，E_{ic}表示t_k的类内分布熵，$p(C_i/t_k)$表示t_k在类别C_i中出现的概率，$tf(t_k, C_i)$表示t_k在类别C_i出现的次数。

TFIDFIGE算法从理论上很好的反应了特征词在各个类别之间以及类内部的分布对权值计算结果的影响，但它在计算过程中把各个特征词视为完全独立的，相互间不存在任何联系，忽略了特征词之间了的语义联系。

事实上，词与词之间是存在语义关系的，许多词互为同义词、近义词，这些词在文本中是可以互相替换的，而单独计算词语权重的方法将导致特征词的权重计算结果存在一定的偏差。例如，一个词在某类中经常出现，而与这个词语义相近的特征词在其他类中也频繁出现，则表明该特征词不能很好的代表该类别，分类能力弱，不能有效地用来分类，应该赋予小的权重值，但因其TF值较大，则通过TFIDFIGE算出的权重值却偏大。同时存在另外一些词，在某个类中出现的频率并不是很高，但与它语义相近的其他词，在该类别中频繁出现，那么该特征词的代表性较好，应该赋予较大的权值，但因其TF较小，由TFIDFIGE得到的权值则偏小。

在文本中通常存在这样三类词：字面不同但含义相同（同义词），一个词语包含多种意思（一词多义），上下文之间隐含着某种语义关系。在文本中特征词的这种关系体现为词汇间的语义关联，这种现象发生的主要原因是词汇层面(字面表示)和概念层面(词汇意思本身)的差别。如果忽略特征词间的这种语义关联，容易导致最终得到的特征词权值不能很好地刻画特征词的重要程度。

针对上述问题，本文从语义的角度出发，首先对特征词进行语义相似度计算，将与特征词 T 语义十分相近的词语归入 T 的相似特征组中，然后再进行 IDF、信息熵、信息增益的计算，提出了一种结合语义、信息熵、信息增益的 S-TFIDFIGE

算法，S-TFIDFIGE 算法流程图如图 3.1 所示。

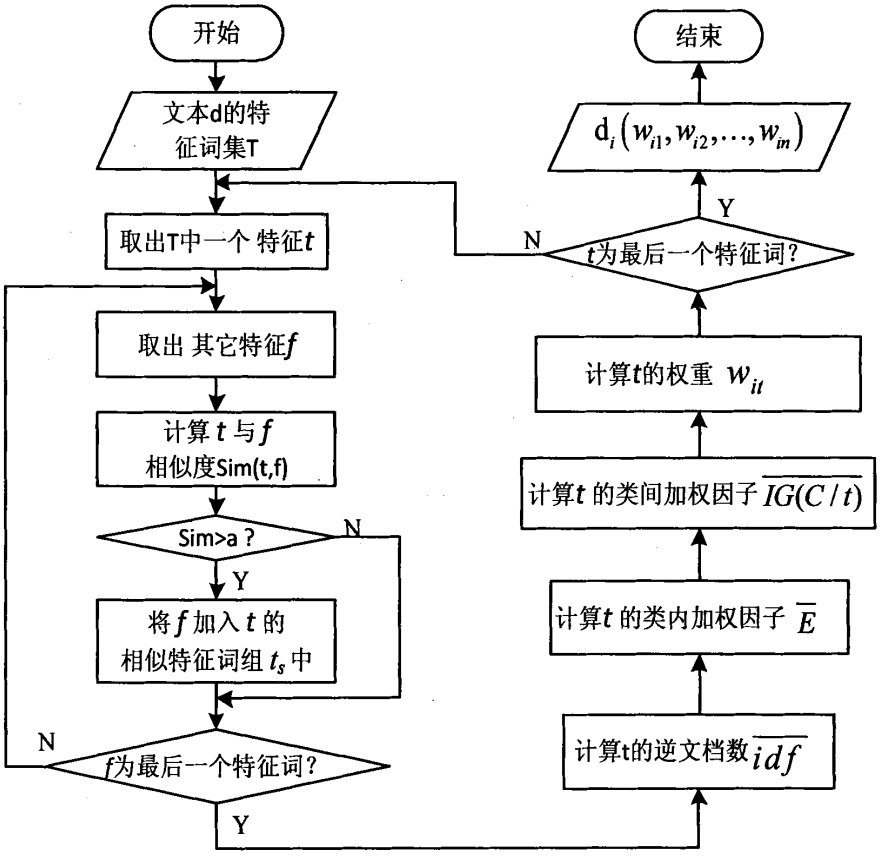

图 3.1　S-TFIDFIGE 算法流程图

改进后的 S-TFIDFIGE 算法描述如下：

输入：文本集经过预处理和特征选择后的特征词集

输出：文本的特征向量

步骤 1　对于特征词集中的特征词 t，根据公式(3.7)计算 t 与特征词集中其他词语的语义相似度，将与特征词 t 语义相似度 $>\alpha$（α 为一阈值，依据本文中 α 取 0.8）的特征词归到 t 的相似特征词组中，并把 t 的相似特征组中的词称为 t 的相似特征词，统计 t 的相似特征词的个数 m；

步骤 2　计算特征词 t 的反文档频数，计算公式如下：

$$\overline{idf} = lb\frac{N}{\overline{n}} \tag{3.13}$$

$$\overline{n} = \frac{n_t + \sum_{r=1}^{m} n_i}{m+1} \tag{3.14}$$

公式(3.13) (3.14)中，N 为文档集中的总文本数，$\overline{n}$ 为出现特征词 t 及其相似特征词的文本数之和的平均数，n_t 为出现特征词 t 的文本数。由公式(3.10)计算的反文档频数，考虑了特征词 t 及其相似特征词在文本集中的分布情况，计算结果更加贴近实际。

步骤 3　计算类内信息熵加权因子：

$$\overline{E}=-\sum_{i=1}^{n}\frac{tf(t_g,d_j)}{tf\left(t_g,C_K\right)}lb\frac{tf\left(t_g,d_j\right)}{tf\left(t_g,C_K\right)} \tag{3.15}$$

公式(3.15)中，$tf\left(t_g,d_j\right)$ 表示特征词 t 及其相似特征词在类别 C_K 中第 j 个文本中出现的频数总和，$tf\left(t_g,C_K\right)$ 表示特征词 t 及其相似特征词在类别 C_K 中出现的频率总和。计算公式如下：

$$tf\left(t_g,d_j\right)=tf_j+\sum_{r=1}^{m}tf_{rj} \tag{3.16}$$

$$tf\left(t_g,C_K\right)=\sum_{i=1}^{n}\left(tf_i+\sum_{r=1}^{m}tf_{ri}\right) \tag{3.17}$$

公式(3.16) (3.17)中，n 表示 C_K 类中的总文本数，tf_j 表示特征词 t 在 C_K 中的第 j 个文本中出现的次数，tf_{ri} 表示 t 的相似特征词组中第 r 个词在 C_K 中的第 i 个文本中出现的次数。由信息熵加权因子的定义可知，当特征词 t 及 t 的相似特征词在类中均匀分布，其信息熵 $\overline{E}$ 值取最大值 1，分类能力最强；当特征词 t 及 t 的相似特征词仅在类中的某个文本中出现时，其信息熵 $\overline{E}$ 的值取最小值 0，分类能力最弱。因此，$\overline{E}$ 能够很好地反映特征词 t 及其相似特征词在类内的分布情况，并且其值与分类能力呈正比。

步骤 4　计算内间信息增益加权因子：

$$\overline{IG(C,t)}=E(C)-\overline{E(C/t)} \tag{3.18}$$

其中，

$$E(C)=-\sum_{i=1}^{u}p(C_i)*lb\left(p(C_i)\right) \tag{3.19}$$

$$\overline{E(C/t)}=-\sum_{i=1}^{u}\left[\left(p(C_i/t)+\sum_{r=1}^{m}p(C_i/t_r)\right)*lb\left(p(C_i/t)+\sum_{r=1}^{m}p(C_i/t_r)\right)\right] \tag{3.20}$$

公式(3.18) (3.19) (3.20)中，C 代表文本集合，$p(C_i)$ 表示类别 C_i 的概率，u 为文档类别数，$p(C_i/t)$ 表示特征词 t 在类 C_i 中出现的概率。$p\left(C_i/t_j\right)$ 表示 t 的相似特

征词组中第 j 个词在类 C_i 中出现的概率。本文将特征词 t 及 t 的相似特征词看做一个整体，做为计算对象，再一同计算信息增益值，最终对文档分类提供的信息量就会变大，对文档分类提供的信息更加全面。

步骤 5　计算特征词 t 的权重值，计算公式如下：

$$w_{it} = tf(d_i) * \overline{idf} * \overline{E} * \overline{IG(C,t)} \tag{3.21}$$

其中，w_{it} 为特征词 t 在文本 d_i 中的权重，$tf(d_i)$ 为特征词 t 在文本 d_i 中的频数。

步骤 6　重复步骤 1 至 5，直到文本的所有特征的权重都计算完成，得到文本的特征向量 $(w_{i1}, w_{i2}, \ldots, w_{in})$，其中 n 为特征词向量的维度，w_{in} 表示文本 d_i 的第 n 个特征词的权重。

由以上的算法描述可以看出，结合语义、信息熵、信息增益的权值算法 S-TFIDFIGE 不仅考虑了特征词 T 在整个文本集及类间、类内中的分布情况，还考虑到与其语义相近的词的分布情况，使得最后得到的刻画文本的特征向量表更加准确。

3.4 小结

本章首先对传统 TFIDF 加权算法进行了简单介绍，并对传统 TFIDF 算法存在的缺陷进行了详细的分析。最后针对基于信息熵与信息增益的 TFIDF 算法（TFIDFIGE）忽略了特征词之间的语义关联的缺陷，将词与词之间的语义关联这一重要因素引入特征词权重算法中，提出了一种结合语义、信息熵、信息增益的 TFIDF 改进算法（S-TFIDFIGE），对于 S-TFIDFIGE 算法的有效性将在第五章通过相关实验进行验证。

第4章 KNN 分类算法改进

在权重计算完成后，每个文本被表示成一个n维的空间向量，下一步的工作就是运用分类算法构造分类器来对文本进行分类。分类算法是文本分类系统的核心，分类算法的好坏直接影响到最终的分类效果。KNN是一种有监督的基于实例的文本分类算法，大量研究证明KNN分类算法是向量空间模型(VSM)下最好的分类算法之一。本章将对传统KNN算法进行简单的介绍，然后详细分析其存在的缺陷，最后针对其存在的缺陷，给出本文改进方法：基于MapRecudce的PKNN算法。

4.1 KNN 算法简介

KNN 算法由 Cover 和 Hart 于 1967 年提出，该算法的基本思想为：给定一个待分类文本，计算该文本与训练样本集中每个文本的相似度，找出训练文本集中与该测试文本距离最近(最相似)的 k 个文本，最后根据测试文本的这 K 个近邻所属的类别判定该测试文本所属的类别。其算法流程图如图 4.1 所示。

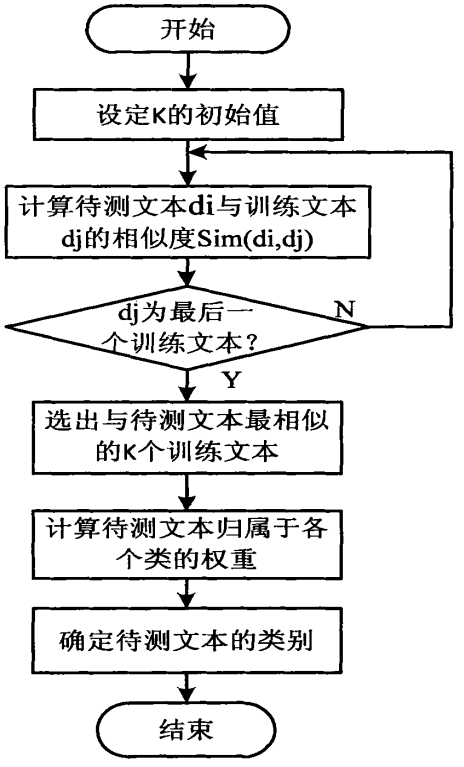

图 4.1 KNN 算法流程图

KNN 算法详细描述如下：

(1)设定 K 的初始值；

(2)根据公式(2.20)计算待分类文本与所有训练文本的文本相似度[21]：

$$Sim\left(d_i, d_j\right) = \frac{\sum_{k=1}^{n} w_{ik} \times w_{jk}}{\sqrt{\sum_{k=1}^{n} w_{ik}^2} \times \sqrt{\sum_{k=1}^{n} w_{jk}^2}} \tag{4.1}$$

公式(4.1)中：d_i为待分类文本向量，d_j为训练集中某一训练文本向量；n为文本向量空间的维数；w_k为文本向量的第k维。

(3)根据计算出来的相似度值，在训练集中选取与待分类文本相似度最高 K 个训练文本。

(4)根据选取的 K 个训练文本，来计算每类的权重，计算方法如下[21]：

$$p\left(x, c_j\right) = \sum_{d_i \in KNN} Sim\left(x, d_i\right) y\left(d_i, c_j\right) \tag{4.2}$$

公式(4.2)中，x 为待分类文本向量，$Sim(x,d_i)$为相似度计算公式，请参考公式(4.1)，$y(d_i, C_j)$为类别属性函数，即：如果d_i属于类C_j，那么该函数值取1，否则取0。

(5)比较各个类的权重，选出权重值最大的类，把该类的类别标识赋给待分类文本，即确定了该文本的所属类别。

KNN 方法是一种基于实例学习的方法，其主要优点体现在算法较成熟、分类效果稳定、实现方便、支持增量学习。

4.2 KNN 缺陷分析

KNN 算法被广泛应用于文本分类，有着算法简单、分类精确度较高等诸多的优点，但 KNN 算法同样存在着不少缺陷。

首先，因为KNN算法是一种懒散型的分类算法，几乎所有的计算都在分类阶段进行。其训练过程只是简单的将训练文本进行预处理并表示成向量形式，而分类阶段，需要将计算待分类文本与训练文本集中的所有文本的相似度，然后选出最相近的K个文本，当数据规模增大时，其计算开销也将跟着增大的。

其次，KNN分类器容易受类偏斜问题的影响。在文本分类中，训练集中的类偏斜现象普遍存在，所谓类偏斜是指各个类别的样本在数量上存在很大的差距。当训练集中的样本存在类偏斜问题时，大样本的类别在分类过程将占据优势，影响最终分类结果的正确性。例如，如A类的样本容量很大，而B类样本容量很小时，有可能导致当输入原本属于B类的新样本d时，该样本的K个邻居中A类的样本数比B类的样本数多，分类器最终将d误判为A类。

此外，在KNN算法中，K值的选取对分类的效率跟精确度有很大影响：K值取的太小，待分类文本的特点将得不到充分的体现；K 值取得太大，将会造成噪

声的增加而导致最终分类效果降低。在实际文本分类中，K值往往都是根据以往经验大致估算来确定的。这种估算的方法往往难以得到最佳的K值，造成了KNN分类算法精确度的下降。

4.3 基于 MapReduce 的 PKNN 算法

由4.2节的分析，我们知道KNN是一种惰性分类算法，在分类过程中，需要计算待分类文本与训练集中所有训练文本的相似度，而文本数据通常是非常高维的，达几百甚至几千，随着数据量的增加，计算量也变得十分庞大，最终导致KNN算法分类时间过长，分类效率较低，限制了其在海量文本数据处理的应用。因此，如何提高KNN算法的分类效率成为当今的研究热点。本文研究工作主要致力于在保持分类精确度的前提下如何提升传统KNN算法的分类速度。

MapReduce 作为一个分布式并行编程模型，为用户提供了一个十分简单和易用的编程接口，用户在使用过程中，只需要像编写串行程序一样来实现 Map、Reduce 等函数即可以实现一个分布式程序，而不必关心一些并行编程的复杂工作，如数据切分、节点间的通信、节点失效等。Hadoop MapReduce 是 Google MapReduce 的开源实现，是一个易于处理大规模数据集并行计算的框架。MapReduce 的自身设计的特点决定它适合处理以下问题：1)大规模逻辑简单的数据运算任务，且该任务容易分割成操作相同的子任务；2)原本就存放在分布式集群上的数据，例如：某些检索系统的日志文件；3)很少甚至不需要人工干预的批处理工作。

KNN算法分类过程中，最大的计算任务是文本间相似度的计算：每个待测文本需要与所有的训练文本进行相似度计算，而文本向量又具有高维性，使得计算任务繁重，影响了KNN算法的分类效率。而文本相似度的计算过程是基于两两文本向量间的余弦值计算（见公式4.1），其逻辑处理是十分简单的，且这些两两文本对相似度的计算很容易被分成若干个MapRedcue的子任务，因此MapReduce非常适合用来处理KNN分类算法过程中文本相似性计算的工作。

在深入分析了KNN分类过程的特点以及MapReduce的优势之后，本文选择采用Hadoop MapRedcue来实现KNN算法的并行化。Hadoop MapReduce执行过程中，需要将各个Map任务节点的中间结果文件传送给Reduce任务节点，网络传输开销较大，不适合用于处理计算时产生的中间结果文件非常大的程序。因此我们的一大难点是如何减少中间结果文件的大小。我们将通过以下两种措施来控制中间结果的大小。

首先，除了必要的Map和Reduce函数外，增设一个Combine函数。仔细分析KNN算法，每个Map结点会产生 xy（x为Map节点中测试文本的数量，y为Map结点中训练文本的数量）个key/value中间键值对，Reducer从各个Map节点中读取数

据，并从具有相同key值的y个key/value键值对，取出相似度最大的k个近邻，再根据这k个近邻判别文本所属类别。从上述描述可以明显看出，Reducer只需取出相似度最大的k个近邻，那么每个Map只需要输出本地的k个近邻即可。因此，我们可以在每个Map操作后面增设计一个combine()函数，Combine函数负责对从每个key值的y个value中取出最大的k个value值。当x=100000，y=10000的时候，没有增设combine()函数的时候，每个Map节点输出的中间键值对为10亿（100000*10000），而增设了combine()函数后，每个Map节点输出的中间键值对为20万（100000*20），由此可见combine()函数可以大大减少中间结果的输出。

其次，对测试文本数据进行切分。加入Combine函数之后，每个Map节点输出的中间结果的大小将由Map节点上的测试文本数决定，例如：Map节点上存储了10000个测试文本的key/value数据，那么Map节点的最终输出将是10000*k个key/value。很多现有的分布式KNN算法如DKNN算法采取对训练文本数据进行分割，将整个测试文本数据副本分发到各个Map节点，当测试文本数据较大的时候，网络传输开销是非常大的。而事实上，我们提出用MapReduce来改进KNN算法，就是为了解决其处理海量文本的分类问题，待测试文本集往往比训练文本集大的多，所以传统的分布式KNN算法在数据处理上显然是不合理的。对测试集进行分割的方法产生的中间数据将是传统分布式KNN算法的 $1/m$ （m为Map节点的个数）。

综上所述，本文改进算法的基本思路为：JobTracker节点对输入的测试集数据进行划分并将训练集数据装入Distributed cache，每个执行MapTask的TaskTracker从HDFS和Distributed cache中读取自己所对应的数据块和训练集数据，然后对数据块中的每个测试文本向量启动一次Map计算过程，通过map函数完成测试文本向量跟训练文本向量相似度的计算，Combine函数对map计算的中间结果进行排序，并取出每个测试文本的前K个近邻作为Reduce节点的输入，Reduce节点根据每个测试文本的前K个近邻分别计算每个类别的权重，并把权重最大的类的类标识赋给测试文本，输出最终结果。该算法设计主要包括三个函数，分别是Map、Combine和Reduce。下面对这三个函数设计进行详细阐述。

4.3.1 Map 函数设计

Map函数的任务是读取对应测试集数据块和训练集数据，计算测试文本向量跟训练文本向量的相似度，形成键值对的映射。具体做法是首先调用内置Split函数按行读取测试集数据并将其转换成特定格式的文件，然后对测试集中的每个测试文本，计算其与每个训练文本的相似度，最后将结果存入context集合。输入数据<key，value>对的形式为<测试文本ID，测试文本向量>；输出数据<key，value>对的形式为<测试文本ID，<训练文本类标识，两文本相似度>>，map过程流程图

如图4.2所示，函数实现伪代码如下。

算法 4.1 Map 函数的实现过程

Input:

<key，value> 其中 key 为测试文本 ID，value 为测试文本向量

Output:

<key，value>，其中 key 为测试文本 ID，value 为<训练文本类别标识，两文本相似度>的复合键

```
void map(id, di)
{
    for each dj in D(Training Set) do   /*遍历所有的训练文本 dj*/
        c = FindCategory(dj);           /*取出训练文本类别标识c*/
        s = calculate:Sim(di,dj);       /*根据式(4.1)计算测试文本di与训练文本dj的相似度*/
        emit(id,<c,s>);                 /*输出运算后的中间结果*/
    end for
}
```

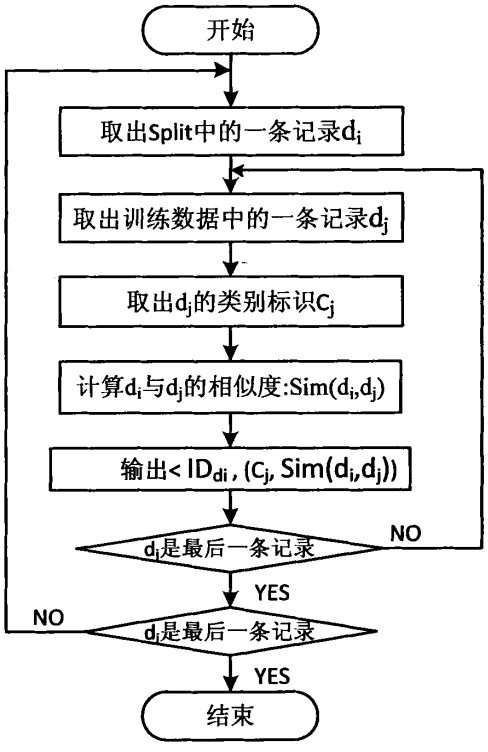

图 4.2 Map 过程流程图

4.3.2 Combine 函数设计

当本地Map节点的Map函数执行完后，对于任意一个测试文本d都包含了y(训练集中文本的数目)个计算结果。如果此时不加处理地将这些计算结果全部提交到Reducer节点去计算，不仅造成网络传输开销的浪费，并且加重了Reducer节点的负担。为了最大限度地节约通信及计算资源，本文设计了Combine函数，具体步骤为：首先读取Map函数的输出键值对，将具有相同key值（即同一个测试文本）的记录归组并将其value加入tempList(变长数组)中，并对根据value中的Sim(di,dj)值进行排序，并取前K个记录（最近邻），作为Reduce节点的输入。Combine过程流程图如图4.3所示，其函数实现伪代码如下。

算法 4.2 Combine 函数的实现过程

Input:

<key，List(value)>，其中 key2 为测试文本 ID，value 为<训练文本类别标识，两文本相似度>的复合键

Output:

<key，List(value)>，其中 key 为测试文本 ID，value 为<训练文本类别标识，两文本相似度>的复合键

```
void combine(id, list(<c,s>))
{
    for each <c,s> in list          /*遍历每个测试文本对应的所有 value:<c,s>值*/
        tempList.add(<c,s>); /*将每个 value 值加到一个 tempList 变长数组中*/
        Sort(tempList) by s;        /*对 tempList 成员依据 s 进行排序*/
        For i=K to tempList.length
            Remove tempList[i];     /*移除该测试文本的K个近邻以外的数据*/
        end for
        emit(id,tempList);          /*输出与该测试文本最相似的K个近邻*/
    end for
}
```

4.3.3 Reduce 函数设计

Reduce函数将combine函数的输出作为输入，并对他们进行相应的规约操作。它的主要的任务是根据每个测试文本的K个近邻，分别计算每类的权重，并将权重最大的类的类别赋给测试文本。其中输入数据<key，value>对的形式为<测试样本的ID，vector(训练文本类标识，相似度值)>，输出数据<key，value>对的形式为<测试文本的ID，测试文本的类标识>。Reduce过程流程图如图4.4所示，其函数实现伪代码如下。

算法 4.3 Reduce 函数的实现过程

Input:

<key，List(value)>，其中 key 为测试文本 ID，value 为<训练文本类别标识，两文本相似度>的复合键

Output:

<key，value>，其中 key 为测试文本 ID，value 为测试文本类别标识

```
void reduce(id, list(<c,s>))
{
    for each <c,s> in list
        calculate weight p(C);   /*根据 k 个近邻及公式（25）计算每个类的权重
*/
    end for
    MaxP = max(p(Ck));          /*求得权重最大的类 ck*/
    emit(id, Ck);               /*输出该测试文本 ID 及其所属类别*/
}
```

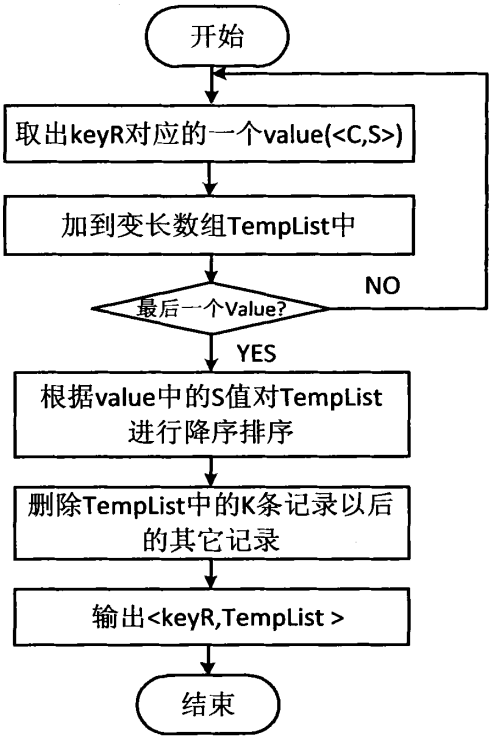

图 4.3 Combine 过程流程图

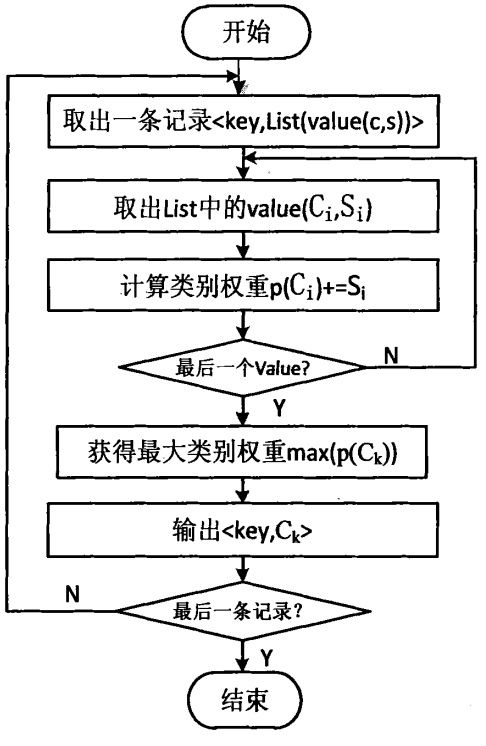

图 4.4 Reduce 过程流程图

PKNN 总的算法流程图如图 4.5 所示，其中 Map 过程，Combine 过程，Reduce 过程的详细流程图请参考图 4.2，4.3，4.4。

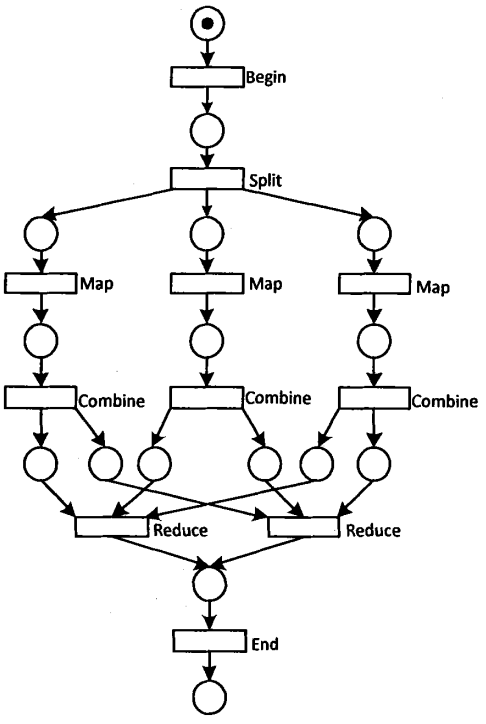

图 4.5 PKNN 算法流程图

图4.6为PKNN算法在MapReduce模型上的并行实现过程中的数据流图。由Reduce产生的结果最终以<测试文本ID，测试文本类别标识>的形式进行存储。

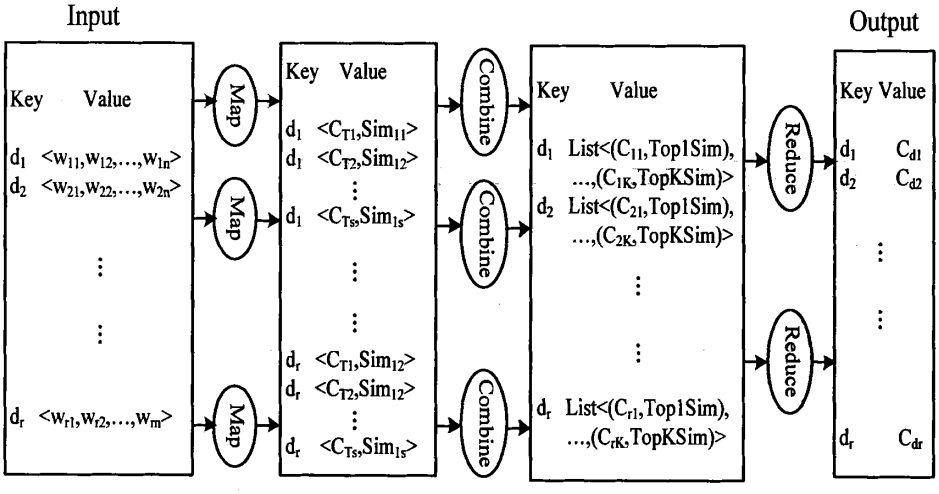

图 4.6 PKNN 算法数据流图

4.4 小结

本章首先对 KNN 文本分类算法进行了简单的介绍，然后详细分析了 KNN 算法存在的不足，并对现有的改进算法进行了介绍。最后在深入分析 KNN 算法自

身特点与 MapReduce 编程模型的基础上，提出一种基于 MapReduce 的并行化 PKNN 算法，并给出了改进算法涉及的三个函数 Map()、Combine()、Reduce()的详细设计。

本章提出的基于 MapRedcue 的 PKNN 算法，在 Map、Reduce 函数的基础上增设了 Combine 函数。此外，对输入数据处理上采用对大规模测试文本数据进行切分，大大削减了 Map 节点执行后的中间结果，节省了网络传输开销，使得本算法更适合处理海量文本数据的分类问题。

第5章 实验设计及结果分析

特征加权算法跟分类器的构造是文本分类过程中比较关键的两个环节。本文第3章对传统TFIDF特征加权算法进行了改进，第4章则对KNN文本分类算法进行了改进。本章我们进行相关实验来验证改进后的特征加权算法(S-TFIDFIGE)及文本分类算法(PKNN)的可行性及有效性。

5.1 实验环境

本文实验环境分为单机环境和集群环境。集群环境是基于云平台改进的KNN算法(DKNN)及本文基于MapReduce的KNN算法（PKNN）的实现平台。单机环境是本实验中其他相关算法的实现平台。

单机环境下PC机的配置如下：

CPU:2.3GHz，内存：2G ，操作系统Ubuntu Server 12.04，Java开发工具包：JDK1.6。

集群环境使用11台PC搭建：其中1台PC作为NameNode，其余10台PC作为DataNode，其中每台PC机配置如下：

CPU:2.3GHz，内存:2G ，操作系统：Ubuntu Server 12.04，Java开发工具包：JDK1.6，云平台：Hadoop1.2.1。

5.2 实验数据

本实验的数据使用搜狗实验室提供的文本分类语料库，该语料库是来自搜狐新闻2008年1至6月期间国内、体育、社会、娱乐等18个频道的数据，语料库大小达3.4G，包括15个类别，2101582篇文档，其具体分布如表5.1所列。本实验选取了新闻、体育、商务、娱乐、房产、财经、女性和IT八个大类做为实验数据源。

表 5.1 SogouCS 文本集分布

类别	新闻	娱乐	体育	财经	IT
文档数（篇）	466825	198414	446681	98074	65421
类别	商务	房产	女性	旅游	汽车
文档数（篇）	344559	216819	87319	43634	32731
类别	健康	教育	文化	军事	就业
文档数（篇）	32518	31529	22281	14683	94

5.3 实验及结果分析

5.3.1 改进后的 S-TFIDFIGE 算法实验

(1) 实验步骤

① 从 SogouCS 中的新闻、体育、商务、娱乐、房产、财经、女性和 IT 八个类中各随机抽取 1000 篇文本作为训练文本，1000 篇文本作为测试文本；

② 采用中科院 ICTCLAS 系统对文本集进行分词处理，得到候选特征词集。

③ 计算每个候选特征词的信息增益值，选取前 1000 个词作为特征词。

④ 对于训练集中的文本，采用本文提出的 S-TFIDFIGE 算法计算特征词权重，将所有训练文本表示成向量形式。

⑤ 对于测试集中的文本，事先并不知道类别，则用传统的 TFIDF 算法计算权重，并把所有测试文本也表示成向量形式。

⑥ 最后采用 KNN（K 值为 20）分类器对测试集中文本的进行分类实验。

⑦ 实验结果评估。本文实验结果采用文本分类中常用的查准率(P)、查全率(R)、F1 值，评估标准详细介绍请参考 2.1.5 节。

(2) 实验结果分析

实验分别采用传统 TFIDF 算法、基于信息增益的 TFIDFIG 算法、基于信息增益与信息熵的 TFIDFIGE 算法及本文改进后的 S-TFIDFIGE 算法来计算特征词的权重，然后采用 KNN 分类器对测试集文本进行分类。表 5.2、表 5.3 和表 5.4 分别对应四种特征权重算法分类结果的查准率、查全率、F1 值。

表 5.2 特征权重算法的查准率

	TFIDF	TFIDFIG	TFIDFIGE	S-TFIDFIGE
新闻	0.7426	0.7653	0.7768	0.8096
娱乐	0.8957	0.9178	0.9244	0.9489
体育	0.8234	0.8391	0.8663	0.8786
财经	0.7929	0.8364	0.8717	0.8783
IT	0.8331	0.8587	0.8846	0.9068
商务	0.9389	0.9434	0.9445	0.9696
女性	0.6725	0.7067	0.7353	0.7686
房产	0.6569	0.6742	0.7084	0.7585
Macro_P	0.7945	0.8177	0.8390	0.8624

从表 5.2 可以看出，S-TFIDFIGE 算法的分类准确率在各个类别上均有相应的提高，尤其是房产类。S-TFIDFIGE 算法的宏平均查准率与传统的 TFIDF 算法相

比，提高了 6.79%。与 TFIDFIG 算法相比，提高了 4.47%。与 TFIDFIGE 算法相比提高了 2.34%。说明改进后的 S-TFIDFIGE 分类准确性较高。

表 5.3 特征权重算法的查全率

	TFIDF	TFIDFIG	TFIDFIGE	S-TFIDFIGE
新闻	0.7122	0.7501	0.7353	0.7889
娱乐	0.9833	0.9807	0.9215	0.9264
体育	0.8097	0.8249	0.8663	0.8768
财经	0.9228	0.9135	0.9343	0.9561
IT	0.6502	0.7284	0.7931	0.8873
商务	0.8854	0.8988	0.9213	0.9387
女性	0.8321	0.8246	0.7962	0.8078
房产	0.6821	0.7343	0.8687	0.8691
Macro_R	0.8097	0.8319	0.8546	0.8814

表 5.3 显示，在查全率方面，改进后的 S-TFIDFIGE 算法也都优于 TFIDF、TFIDFIG 和 TFIDFIGE 算法，其宏平均查准率同比其他三种算法分别提高了 7.17%、4.95%、2.68%，说明本文提出的 S-TFIDFIGE 具有较好的稳定性。

表 5.4 特征权重算法的 F1 值

	TFIDF	TFIDFIG	TFIDFIGE	S-TFIDFIGE
新闻	0.7271	0.7576	0.7554	0.7991
娱乐	0.8165	0.8319	0.8762	0.8777
体育	0.9375	0.9482	0.9229	0.9375
财经	0.8529	0.8732	0.9019	0.9554
IT	0.7304	0.7882	0.8364	0.8969
商务	0.8871	0.9158	0.9328	0.9441
女性	0.7438	0.7611	0.7645	0.7771
房产	0.6693	0.703	0.7804	0.81
Macro_F1	0.7956	0.8224	0.8464	0.8747

对于综合评估指标 F1 值，由表 5.4 显示的数据可以看出 TFIDF、TFIDFIG、TFIDFIGE 和 S-TFIDFIGE 算法的宏平均 F1 值分别为 79.56%、82.24%、84.64%、87.47%。这说明在四种权重算法中，TFIDFIG 算法优于传统的 TFIDF 算法，TFIDFIGE 算法则优于前两种算法。而本文提出 S-TFIDFIGE 算法的分类效果最佳。

综上所述，改进后的 S-TFIDFIGE 算法的准确性、稳定性和分类效果都优于其他三种算法，这证明改进后的 S-TFIDFIGE 算法与理论上的论证是一致的，特征词之间的语义关联对权重值有不可忽视的影响，将这个因素引入权重公式中对最终的分类效果有明显的提高。

5.3.2 改进后的 PKNN 算法实验

通过第四章的介绍与分析，我们知道本文提出的PKNN算法适用于处理大规模测试文本数据的分类问题。为了验证PKNN算法的有效性。本文设计了3套方案，各方案中的数据来源采取从八个类的的每类中随机抽取等比例的文本数作为实验数据集，文本的预处理环节同5.3.1节。

(1)方案设计

方案1：将本文改进算法PNN与其他串行算法的分类效果进行对比。

数据集大小分配如表5.5所示，分别用传统的KNN，基于聚类改进的KNN(CKNN)算法（将训练集中每个类的4篇文档合并为一个中心文档），及本文改进算法PKNN进行实验。采用分类精确度与分类时间两个常用指标作为评价标准。

表 5.5 A 组实验数据集

数据集编号	训练样本(篇)	测试样本(篇)
A1	8000	80000
A2	10000	100000
A3	12000	200000
A4	16000	400000

方案2：将本文改进算法（PKNN）与其他分布式KNN算法(DKNN)的分类效果进行对比。数据集大小分配如表5.6所示，分别用DKNN及本文改进算法PKNN进行实验。采用分类运行时间作为评价标准。

表 5.6 B 组实验数据集

数据集编号	训练样本/篇	测试样本/篇
B1	10000	10000
B2	10000	100000
B3	10000	400000
B4	10000	800000
B5	10000	1000000
B6	10000	1500000

方案3：验证本文改进算法（PKNN）的可扩展性。

该部分实验通过改变节点的个数，来测试三组大小不同的数据的分类时间。数据集的分配如表5.7所示。

表 5.7　C 组实验数据集

数据集编号	训练样本(篇)	测试样本(篇)
C1	10000	100000
C2	10000	300000
C3	10000	500000

(2) 实验结果分析

1)根据方案1，统计了各个数据集在用KNN、CKNN和改进的PKNN算法进行分类后的分类精确度跟分类时间如图5.1，5.2所示。

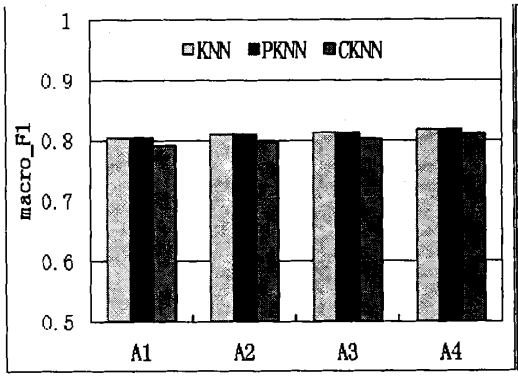

图 5.1　分类精确度对比

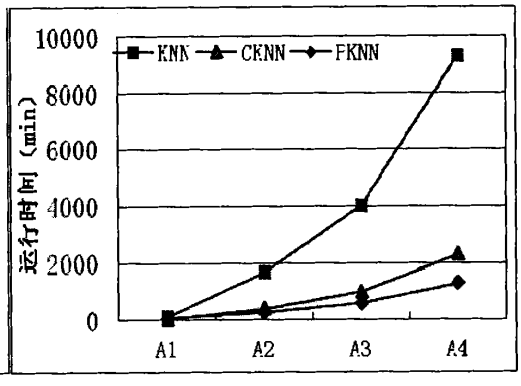

图 5.2　分类时间对比

从图5.1和图5.2可以看出，采用聚类算法将类中若干相似的文档合并成一个中心文档的CKNN方法，减少了参与相似度计算文本的规模，提高了文本分类的速度，但其分类精确度受到了影响，有所下降。而本文基于MapReduce改进的PKNN算法在保持较高分类效率的同时，其分类精确度与传统的KNN算法保持一致。

另外，在数据较小的时候，PKNN算法优势并不明显，这是因为在Hadoop环境下，存在对数据的分割以及节点间的数据传输，会消耗一定的时间，随着数据规模的增大，加速的效果更加明显。

2)根据方案2，统计分布式改进算法DKNN和本文改进的PKNN算法对数据集B1、B2、B3、B4、B5、B6进行分类时所用的时间，如图5.3所示。

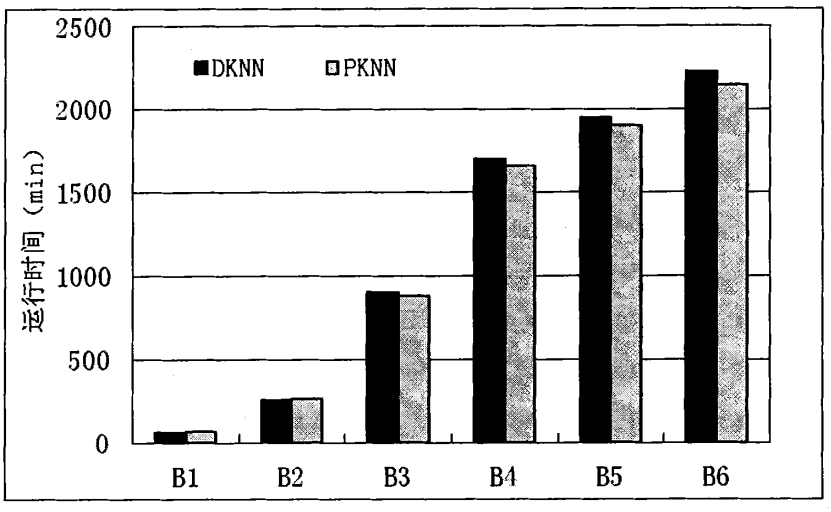

图 5.3 DKNN 跟 PKNN 分类时间对比

由图5.3可知，在训练样本与测试样本规模相当，且不是很大的时候，PKNN算法跟DKNN算法分类效率差别并不是很大。当测试集的规模增大的时候，PKNN算法优势逐渐显示出来，分类时间明显小于DKNN算法，这与4.3节的理论分析是一致的，说明本文改进算法更加适合用来对大规模测试文本进行分类。

3)根据方案3，将节点个数由4逐渐增加到10，记录各个数据集在不同节点数下的分类时间，实验结果如图5.4所示。

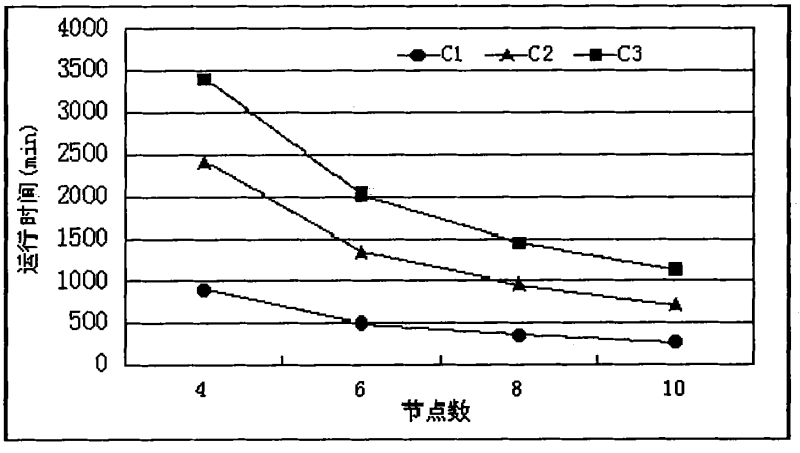

图 5.4 PKNN 运行时间变化图

由图5.4可以看出，各数据集的分类运行时间随着结点的增多而减小，说明本文改进算法PKNN可以通过增加Hadoop集群环境中的节点个数来减少文本分类时间，证明本文改进算法具有良好的可扩展性。

综上所述，本文提出的PKNN算法可以在保持分类精确度不变的情况下，大大缩短文本分类的时间，提高文本分类的效率。同时，该算法具有良好的可扩展

性，可以通过增加集群中的节点数来减少算法运行时间，适于用来处理海量文本数据的分类问题。

5.3.3 S-TFIDFIGE 和 PKNN 算法结合实验

通过前面三四章的分析我们知道，S-TFIDFIGE 加权算法主要影响文本分类的精确度，而 PKNN 算法主要影响文本分类过程的分类时间。本实将验证改进的 S-TFIDFIGE 算法和 PKNN 算法结合后的分类效果。

各方案中的数据采取，从 SogouCS 中的商务、娱乐、财经、女性、IT、新闻、体育和旅游八个类中各随机抽取 1250 篇（共 10000）文本作为训练文本，25000（共 200000 篇）篇文本作为测试文本；文本的预处理环节同 5.3.1。

(1) 方案设计：对比方案 1 和方案 2 的实验效果。

方案1：采用传统的TFIDF算法和KNN算法来计算特征权重和进行分类；

方案2：采用改进后的S-TFIDFIGE算法和PKNN算法来计算特征权重和进行分类；

(2) 实验结果分析

本实验的方案 1 和方案 2 中的 K 值均取 20，实验结果采用宏平均查准率(Macro_P)、宏平均查全率(Macro_R)、宏平均 F1 值(Macro_F1)和分类时间对分类结果进行评估。表 5.8 为方案 1 和方案 2 的实验分类结果。

表 5.8 两种方案实验结果对比

方案	Macro_P	Macro_R	Macro_F1	分类耗时(min)
1. TFIDF + KNN	80.08%	81.45%	80.76%	3356
2. S-TFIDFIGE + PKNN	86.39%	88.12%	87.25%	481

由表 5.8 的实验结果可以看出，方案 2 比方案 1 在综合评估指标 Macro_F1 上大约提高了 6.49%，且分类速度提高了近 7 倍。由此可见，使用本文改进后的权重算法 S-TFIDFIGE 跟分类算法 PKNN 能够在提高文本分类精确率的同时，大幅度提升文本的分类速度。

5.4 小结

本章通过设计和实施相关实验来验证第三、四章所提出的改进后的 S-TFIDFIGE 权值计算方法和 PKNN 分类算法的可行性和有效性。

首先，设计了 S-TFIDFIGE 算法的对比实验。分别使用 TFIDF、TFIDFIG、TFIDFIGE 和本文改进算法 S-TFIDFIGE 计算权重，利用 KNN 分类器进行分类，

对比最终的分类精确度。实验结果表明，本文改进算法 S-TFIDFIGE 较之其他权重算法，能够获得更高的分类精确度。

然后，通过设计三组实验来验证 PKNN 算法的有效性。首先，将本文改进算法 PKNN 与传统 KNN 算法，基于聚类改进的 CKNN 算法进行对比，实验结果显示 PKNN 算法能够在提高分类的速度的同时保持分类的精确度。其次，将 PKNN 算法与分布式 DKNN 算法进行对比，实验结果表明 PKNN 算法对大规模测试集的分类效率优于 DKNN 算法。最后，通过改变 Hadoop 集群中的结点数来观察各个数据集的分类时间变化，实验结果证明 PKNN 算法具有很好的扩展性。

最后，将本文改进后的权重算法 S-TFIDFIGE 和分类算法 PKNN 进行结合。实验结果表明，改进后的 S-TFIDFIGE 和 PKNN 算法能够很好的结合，提高文本分类的效率跟准确率。

结 论

随着互联网的快速普及，目前个人、家庭、企业、政府都开始利用网上电子资源或者向互联网发送大量的电子文档，如电子邮件、电子图书、电子新闻等等。由于互联网上的电子文档数量急剧膨胀，导致用户在查询所需要的信息时，出现了大量的无用信息。无用信息常常比有用信息的数量大几倍、几十倍甚至几百倍，用户需要通过一个有效的工具过滤掉那些繁杂的无用的信息。文本分类作为组织和处理海量文本数据的关键技术，能够帮助用户快速、准确地定位所需要的信息，在很大程度上解决了信息杂乱问题，是一项具有重大实用价值和意义的课题。本文重点研究了文本分类关键技术中的权重计算跟分类算法，所做的工作主要体现在以下几个方面：

1、介绍了文本分类的研究背景及意义，分析了文本分类的国内外研究现状以及 TFIDF 算法、KNN 算法的研究现状。探讨了中文文本分类的一些关键技术，主要包括文本预处理、特征选择、特征权重计算、分类算法、性能评价。此外，简单介绍了分布式并行计算编程模型 MapReduce，重点介绍了 MapReduce 模型的基本思想及其优缺点，执行过程和模型实现。

2、详细的分析了传统TFIDF权重算法存在的缺陷，并介绍了国内外学者针对TFIDF算法的固有缺陷进行的一系列相关研究。接着，针对传统TFIDF算法忽略了特征词与其他词语之间的语义联系及其在文本集中各个类别间、类内部的分布情况，从语义的角度出发，提出了一种结合语义、信息熵、信息增益的TFIDF改进算法（S-TFIDFIGE）。

3、详细分析了 KNN 算法存在的不足，并对现有的改进算法进行了介绍。然后，针对 KNN 算法计算复杂度大，分类效率较低的问题，提出一种基于 MapReduce 并行化的 PKNN 算法，充分利用了分布式编程模型 MapReduce 的海量数据处理优势，并给出了该改进算法涉及的三个函数 Map()、Combine()和 Reduce()的详细设计。

4、对改进后的S-TFIDF、PKNN进行了相关实验。实验内容主要包括：

1）设计了 S-TFIDFIGE 算法的对比实验。

将本文改进的加权算法 S-TFIDFIGE 与传统 TFIDF、基于信息增益的 TFIDFIG、基于信息熵和信息增益的 TFIDFIGE 进行了对比。

2）设计三组实验来验证 PKNN 算法的有效性。

第一组实验对比本文改进算法 PKNN 与其他串行改进算法的分类效果。

第二组实验对比了本文改进算法 PKNN 与并行改进算法的分类效率。

第三组实验通过改变集群中的结点数来观察各个数据集在 PKNN 算法下分类

的时间变化。

3）将改进后的权重算法 S-TFIDFIGE 和分类算法 PKNN 进行结合，并将传统 TFIDF 跟 KNN 算法进行结合，然后将两组方案进行实验对比。

实验结果表明：改进后的加权算法 S-TFIDFIGE 的分类精确度比其他三种加权算法都要高。改进后的 PKNN 分类算法能够在保持分类精确度不变的情况下，大大缩短文本分类的时间，具有较好的可扩展性。S-TFIDFIGE 和 PKNN 算法结合后，能够同时提高文本分类的效率和准确率，适合处理大规模文本的分类问题。

本文虽然取得了一定的研究成果，完成了本研究课题的预期目标，但仍然还有许多值得完善、改进和探索的地方。文本分类是一个极具研究意义和挑战的研究课题，本文仅研究了其中的特征权重计算和分类算法的一些很小的方面，对于文本分类的其他方面的研究还有待日后进一步深入探讨。本文下一步的研究工作将主要放在：

1、进一步探索在中文文本分类中特征词相似度所蕴含的语义特点，对特征词相似度计算方法进行改进，从而提高特征词相似度计算的准确性。中文文本在分词处理、时态、词性标注、单复数处理和语法句法分析等方面，有别于其它语言。虽然，目前对计算机语言学的研究已经取得了一定的进展，但其仍面临着统一的、大规模的语料库和开放的、标准的、分类文本集等基础建设的缺乏，以及语义、句法分析等方面的技术难题。本文结合语义的 S-TFIDFIGE 权重计算方法建立在特征词语义相似度计算的基础上，因此如何更好计算特征词之间的相似度，提高特征词间相似度计算的准确性，对本文改进的 S-TFIDFIGE 权重计算方法有着重大的意义。

3、文本分类其他过程的并行化。本文仅实现了KNN文本分类算法的并行化，后续的工作将探索利用MapReduce对文本分类中其他过程进行并行化处理，如对文本预处理过程的并行化，文本特征选择算法的并行化，特征权重计算的并行化等，进一步提高文本分类整个过程的效率。

4、其他文本分类关键技术的研究。本文仅从分类器和权重计算两个关键技术出发，对文本分类做了一些相关工作，但对文本分类来说其他技术的研究也同样很重要，比如特征选择，也会影响文本分类的结果。因此，下一步的工作可以对其他文本分类关键技术进行研究，进一步提高文本分类的性能。

5、多标签分类问题的研究。多标签分类就是给一个待测文本分配一个或多个类别。目前常用的做法有：1)对每个类别训练得到一个二元分类器，分类时把所有判别为“是”的类别都判为待测文本的类别；2)对每个类别训练得到一个预测实数积分的决策函数。分类时把积分大于阈值的类别都判为待测文本的类别。然而，这两种做法都不能很好地考虑类别之间的相关性。多标签分类比单标签分类具有更广泛的实际应用，因此下一步的工作可以对多标签分类问题进行相关研究。

参考文献

[1] Peter Lyman, Hal R. Varian, et al. How Much Information? http:// www2.sims. berkeley.edu/research/projects/how-much-info-2003/

[2] 第 25 次中国互联网络发展状况统计报告. http://tech.qq.com/zt/ 2010/cnnic25/

[3] Jiang S, Pang G, Wu M, et al. An improved K-nearest-neighbor algorithm for text categorization. Expert Systems with Applications, 2012, 39(1): 1503-1509

[4] Luo Q, Chen E, Xiong H. A semantic term weighting scheme for text categorization. Expert Systems with Applications, 2011, 38(10): 12708-12716

[5] Luhn H P. The automatic creation of literature abstracts. IBM Journal of research and development, 1958, 2(2): 159-165

[6] Maron M E, Kuhns J L. On relevance, probabilistic indexing and information retrieval. Journal of the ACM (JACM), 1960, 7(3): 216-244

[7] Rocchio J J. Relevance feedback in information retrieval. In the SMART Retrieval System: Experiments in Automatic Document Processing, 1971,313-323

[8] Salton G. Experiments in automatic document processing. In the SMART retrieval system.1971

[9] Hayes P J, Weinstein S P. CONSTRUE/TIS: A System for Content-Based Indexing of a Database of News Stories. In: Proc of IAAI. 1990, 90: 49-64

[10] Bhakkad A, Dharmadhikari S C, Emmanuel M, et al. E-VSM: Novel text representation model to capture contex-based closeness between two text documents. In: Proc of IEEE International Conference on Intelligent Systems and Control (ISCO), 2013: 345-348

[11] Chen J, Huang H, Tian S, et al. Feature selection for text classification with Naïve Bayes. Expert Systems with Applications, 2009, 36(3): 5432-5435

[12] Zhang W, Yoshida T, Tang X. A comparative study of TF* IDF, LSI and multi-words for text classification. Expert Systems with Applications, 2011, 38(3): 2758-2765

[13] Taşcı Ş, Güngör T. Comparison of text feature selection policies and using an adaptive framework. Expert Systems with Applications, 2013, 40(12): 4871-4886

[14] Jiang J Y, Liou R J, Lee S J. A fuzzy self-constructing feature clustering algorithm for text classification. IEEE Transactions on Knowledge and Data Engineering, 2011, 23(3): 335-349

[15] Hong T P, Lin C W, Yang K T, et al. Using TF-IDF to hide sensitive itemsets. Applied intelligence, 2013, 38(4): 502-510

[16] Behl D, Handa S, Arora A. A bug mining tool to identify and analyze security bugs using Naive Bayes and TF-IDF. In: Proc of IEEE International Conference on Optimization, Reliabilty, and Information Technology (ICROIT), 2014: 294-299

[17] Sriram B, Fuhry D, Demir E, et al. Short text classification in twitter to improve information filtering. In: Proc of the 33rd international ACM SIGIR conference on Research and development in information retrieval. ACM, 2010: 841-842

[18] Ordovás-Pascual I, Almeida J S. A fast version of the k-means classification algorithm for astronomical applications. arXiv preprint arXiv:1404.3097, 2014

[19] Dattola R T. A fast algorithm for automatic classification. Information Technology and Libraries, 2013, 2(1): 31-48

[20] 侯汉清， 黄刚. 电子计算机与文献分类. 计算机与图书馆，1982，1：001

[21] 闫晨. KNN 文本分类研究：[燕山大学学位论文] . 秦皇岛：燕山大学信息科学与工程学院，2010

[22] 黄萱菁. 独立于语种的文本分类方法. 中文信息学报, 2000，14(6)：1-7

[23] 李荣陆，王建会，陈晓云，等. 使用最大熵模型进行中文文本分类. 计算机研究与发展，2005，42(1)：94-101

[24] Yun-tao Z, Ling G, Yong-cheng W. An improved TF-IDF approach for text classification. Journal of Zhejiang University Science A, 2005, 6(1): 49-55

[25] 董小国，甘立国. 基于句子重要度的特征项权重计算方法. 计算机与数字工程，2006，34(8)：34-37

[26] 田冬阳. 一种基于改进支持向量机的文本倾向性分类算法. 微型电脑应用，2011，27 (3)：34-38

[27] Shi K, Li L, Liu H, et al. An improved KNN text classification algorithm based on density. In: Proc of IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS), 2011: 113-117

[28] Yu C, Feng W, Zhou L, et al. Spam messages classification algorithm based on BP and isomap. In: Proc of IEEE International Conference on Cross Strait

Quad-Regional Radio Science and Wireless Technology Conference (CSQRWC), 2013: 369-372

[29] 王进，金理雄，孙开伟. 基于演化超网络的中文文本分类方法. 江苏大学学报 (自然科学版)，2013，34(2)：196-201

[30] Wei J, Shi H B, Ji S Q. Distributed Naive Bayes Text Classification Using Hadoop. Jisuanji Xitong Yingyong- Computer Systems and Applications, 2012, 21(2): 210-213

[31] 张玉芳，陈小莉，熊忠阳. 基于信息增益的特征词权重调整算法研究. 计算机工程与应用，2007，43(35)：159-161

[32] 李学明，李海瑞，薛亮，等. 基于信息增益与信息熵的 TFIDF 算法. 计算机工程，2012，38(08)：37-40

[33] Wang N, Wang P, Zhang B. An improved TF-IDF weights function based on information theory. In: Proc of IEEE International Conference on Computer and Communication Technologies in Agriculture Engineering (CCTAE), 2010, (3): 439-441

[34] Huang X, Wu Q. Micro-blog commercial word extraction based on improved TF-IDF algorithm. In: Proc of IEEE International Conference on TENCON 2013-2013 Region 10, 2013: 1-5

[35] Jie G, Li-chao C. Research of improved IF-IDF Weighting algorithm. In: Proc of IEEE International Conference on Information Science and Engineering (ICISE), 2010: 2304-2307

[36] 李媛媛，马永强. 基于潜在语义索引的文本特征词权重计算方法. 计算机应用，2008，28(6)：1460-1462

[37] Zhu D, Xiao J. R-tfidf, a Variety of tf-idf Term Weighting Strategy in Document Categorization. In: Proc of IEEE International Conference on Semantics Knowledge and Grid (SKG), 2011: 83-90

[38] Forman G. BNS feature scaling: an improved representation over tf-idf for svm text classification. In: Proc of the 17th ACM conference on Information and knowledge management. ACM, 2008: 263-270

[39] Li B, Guoyong Y. Improvement of TF-IDF Algorithm Based on Hadoop Framework. 2012

[40] Jing Y, Gou H, Zhu Y. An Improved Density-Based Method for Reducing Training Data in KNN. In: Proc of IEEE International Conference on Computational and Information Sciences (ICCIS), 2013: 972-975

[41] 张孝飞，黄河燕．一种采用聚类技术改进的 KNN 文本分类方法．模式识别与人工智能，2009， 22(6)：936-940

[42] YU WANG1, ZHENG-OU WANG. A FAST KNN ALGORITHM FOR TEXT CATEGORIZATION.Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong, 2007:19-22

[43] Yang C, Wei J. A Fast KNN Categorization Algorithm Based on Feature Space Indexing.Recent Advances in Computer Science and Information Engineering. Springer Berlin Heidelberg, 2012: 171-176

[44] 李静．基于云计算平台的分布式 KNN 分类算法的设计与实施．科技通报，2013，29(6)：92-94

[45] Shi K, Li L, Liu H, et al. An improved KNN text classification algorithm based on density. In: Proc of IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS), 2011: 113-117

[46] 茅剑，刘晋明，曹勇．一种基于密度的改进 KNN 文本分类算法．漳州师范学院学报，2012，25(2)：45-48

[47] Liu X, Ren F, Yuan C. Use relative weight to improve the kNN for unbalanced text category. In: Proc of IEEE International Conference on Natural Language Processing and Knowledge Engineering (NLP-KE), 2010: 1-5

[48] 王小青．基于并行遗传算法的 KNN 分类方法．西南师范大学学报：自然科学版，2010，35(2)：103-106

[49] Gong A, Liu Y. Improved KNN Classification Algorithm by Dynamic Obtaining K. Advanced Research on Electronic Commerce, Web Application, and Communication. Springer Berlin Heidelberg, 2011: 320-324

[50] 周红鹃，祖永亮．基于后验概率制导的 B-KNN 文本分类方法．计算机工程，2011，37(21)：114-116

[51] Cao J F, Chen J J. An improved web text classification algorithm based on SVM-KNN. Applied Mechanics and Materials, 2013, 278: 1305-1308

[52] Li Z, Xiong Z, Zhang Y, et al. Fast text categorization using concise semantic analysis. Pattern Recognition Letters, 2011, 32(3): 441-448

[53] 顾益军，樊孝忠，王建华，等．中文停用词表的自动选取．北京理工大学学报，2005，25(4)：337-340

[54] Salton G, Wong A, Yang C S. A vector space model for automatic indexing. Communications of the ACM, 1975, 18(11): 613-620

[55] Gheyas I A, Smith L S. Feature subset selection in large dimensionality domains. Pattern recognition, 2010, 43(1): 5-13

[56] 杨营辉. 基于密度的样本裁剪算法的改进及在 KNN 中的应用研究：[重庆大学学位论文]. 重庆 ：重庆大学计算机学院，2010，11-15

[57] Emmanuel M, Khatri S M, Babu D R. A Novel Scheme for Term Weighting in Text Categorization: Positive Impact Factor. In: Proc of IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2013: 2292-2297

[58] Ko Y. A study of term weighting schemes using class information for text classification. In: Proc of the 35th international ACM SIGIR conference on Research and development in information retrieval. ACM, 2012: 1029-1030

[59] 赵小化. KNN 文本分类中特征词权重算法的研究：［太原理工大学硕士学位论文］. 太原 ：太原理工大学计算机与软件学院，2010，19-21

[60] 魏建. 基于语义中心的 KNN 文本分类算法研究：［南京理工大学硕士学位论文］. 南京 ：南京理工大学计算机科学与技术学院，2007，10-12

[61] Dean J, Ghemawat S. MapReduce: a flexible data processing tool. Communications of the ACM, 2010, 53(1): 72-77

[62] 李成华，张新访，金海，等. MapReduce：新型的分布式并行计算编程模型. 计算机工程与科学，2011，33(003)：129-135

[63] 李建江，崔健，王聃，等. MapReduce 并行编程模型研究综述. 电子学报，2012，39(11)：2635-2642

[64] He B, Fang W, Luo Q, et al. Mars: a MapReduce framework on graphics processors. In: Proc of the 17th international conference on Parallel architectures and compilation techniques. ACM, 2008: 260-269

[65] Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. Information processing & management, 1988, 24(5): 513-523

[66] 刘强. 文本的特征提取及 KNN 分类优化问题研究：[华南理工大学学位论文]. 广州：华南理工大学软件学院，2009，7-9

[67] 刘群，李素建. 基于《知网》的词汇语义相似度计算. 中文计算语言学，2002，7(2)：59-76

附录 A 攻读学位期间所发表的学术论文目录

[1] KeHua Yang，Dan Ye. A Novel Improved TFIDF Algorithm.The 2014 International Conference on Computer Science and Service System,(已录用)

附录 B 攻读硕士学位期间所参与的科研活动

[1] 国家工信部核高基项目:实时嵌入式操作系统及开发环境,项目编号:2009ZX01038-001.

[2] 中央高校基本科研业务费项目:信息物理融合系统中数据管理的关键技术研究,编号:531107040273.

致 谢

当论文的写作接近尾声时，连日阴雨的长沙总算迎来期待已久的晴空。看基地实验室窗外，阳光明媚，云卷云舒，树影斑驳，鸟语花香，岳麓山下的校园氤氲在温润的光泽中。整个校园显得如此的宁静与祥和！而此刻我的心情却异常的激动，回想三年研究生的生活，点点滴滴，历历在目。在这三年的求学生涯中，我得到了许多的关心和帮助，借此机会，向所有曾经关心、帮助过我的老师、家人和同学朋友表示衷心地感谢！

首先，我要由衷的感谢我的导师杨科华老师。杨老师严谨求实的科研态度，精益求精的科研精神使我受益颇多，更为我以后的学习、工作树立了很好的榜样，将是我学习的楷模。三年来，杨老师多次组织学术交流活动，为我们营造了良好的学术讨论氛围，这也极大地调动了我的研究热情。与此同时，杨老师还为我们提供了良好的实践平台，让我们在项目中实践，在实践中成长。在本课题的研究和学习过程中，杨老师时刻关心我的研究进展，并给我提了许多宝贵意见，本文的顺利完成和杨老师的支持和帮助是分不开的。再次感谢杨老师对我这三年来的关心、栽培和鞭策！师恩难忘，铭记于心。

其次，我要感谢我的同门高贺庆、陈琳、袁琼、杨翔、易革军、冯玉萍。虽然你们其中的三位已经离校，走向工作岗位，但我永远不会忘记一起开会讨论时的“争锋相对”，不会忘记一起爬岳麓山时的“谈笑风生”，一起唱K时的“风风火火”。三年来，大家互相交流、互相帮助、共同成长、共同进步。感谢你们，让我度过了三年美好而难忘的生活。特别感谢我的师弟易革军，感谢你在我论文的撰写过程中，细心的帮助我修改论文中的语法、格式、问题，没有你的帮助，我的小论文不会那么顺利得以录用和发表。感谢李显玉、钱勤、盛莹莹三位室友，感谢你们在生活中对我的关心和谦让。同时感谢计科2班所有的同学，感谢研究生生涯中遇到的所有朋友，你们的出现为我的生活增添了色彩，留下了许多美好的回忆。

最后，我要感谢我最亲爱的父母，是你们为我创造了一个温馨的家庭，让我快乐的生活与成长，是你们为我撑起一片蓝天，让我自由的翱翔，没有你们默默的支持和付出，就没有我的今天。

谨以此文献给我的导师、同学、朋友和亲人，感谢你们对我的教育和帮助，以及在整个硕士阶段，对我自始至终的鼓励与关怀。

叶 丹

2014年5月

KNN文本分类及特征加权算法研究

作者：叶丹
学位授予单位：湖南大学