基于朴素贝叶斯的中文文本分类

【摘要】 计算机和通讯技术的发展带来了海量数据，对这些文本信息进行分类的意义也变得越来越重大。基于概率模型的朴素贝叶斯算法是解决文本分类较为有效的方法。本文的主要任务是讨论朴素贝叶斯中文本分类器的理论基础以及描述用 Java 语言实现该分类器的过程。该分类器的实现可以归为特征提取和对特征进行计算两部分。特征提取包括了中文分词及去除停用词。对特征进行计算包括了计算先验概率、似然函数值、最大后验估计。此外，本文采用了搜狗实验室文本分类语料库作为训练集和测试用例集对本文研究的方法进行了简单的测试，测试结果显示，该分类器的微平均准确率在 39%-56%之间，其性能有待提高。最后，本文还对该分类器提出了改良方案，展望了应用前景。

【关键词】 朴素贝叶斯　文本分类　中文分词　停用词

Chinese Text Classification Based on Naive Bayes

[Abstract] The development of computer and communications technology has resulted in huge amount of data. The automatic text classification technique has become very significant. Naive Bayes algorithm is based on probabilistic model. It is an effective way to deal with automatic text classification. The main task of this paper is to discuss the theoretical basis of Naive Bayes text classifier and describe the process of using Java language to accomplish the classifier. We can divide the classifier into two parts: the feature extraction and the calculation according to the feature. In the feature extraction part, I use the Chinese word segmentation method and the stop words filtering. In the classification part, I calculate the prior probability, the likelihood function value and the maximum a posterior estimation. During the simple test, the author uses the Sogou laboratory's text classification corpus as the training set and the test set. During the test, the accuracy is between 39% to 56 %. The results show that there is still room for improvement. The paper also includes the discussion of its improvement methods and wider application.

[Keywords] Naive Bayes, text classification, Chinese word segmentation, stop words

目录

引言

随着以微博、微信为代表的社交网络的蓬勃发展，以及云计算、物联网等技术的突飞猛进，数据正以惊人的速度在增长和累积。[1]

面对海量的以文本为主的数据信息，我们希望能够有效地利用它们。文本分类能对错综复杂的文本数据信息进行初步的组织和管理。因此，文本自动分类的重要性也越来越明显。而文本分类系统作为一种能对文本数据进行自动分类的组织工具，不论是面对海量数据还是少量数据，它都有助于信息资源的发现、过滤，有助于信息资源更大程度地发挥其价值。

目前，统计学习法是文本分类的主流方法。统计学习法背后有着坚实的理论基础，它有着评价标准明确和实际表现良好的优点。本文介绍的朴素贝叶斯算法属于统计学习法，它也是被广泛运用的文本分类方法。

在本文中，我将和大家分享我对贝叶斯公式的新理解以及我对构建朴素贝叶斯中文文本分类器的探究过程。

第一章　绪论

1.1 朴素贝叶斯算法简介

被证明是简单而强大的推理框架的贝叶斯算法源自数学家托马斯•贝叶斯为解决“逆概”问题而写的一篇文章。后来贝叶斯算法席卷了概率论，并且在许多问题领域得到了运用。[2]

贝叶斯算法的核心——贝叶斯公式是公式 1-1。

$$p(\mathrm{A}\,|\,\mathrm{B})=\frac{p(\mathrm{B}\,|\,\mathrm{A})p(\mathrm{A})}{p(\mathrm{B})} \tag{1-1}$$

在公式 1-1 中，

$p(\mathrm{A}\,|\,\mathrm{B})$ 表示事件 B 发生的前提下，事件 A 发生的概率。此项又被称为“逆向概率”或者“后验概率”。

$p(\mathrm{B}\,|\,\mathrm{A})$ 表示事件 A 发生的前提下，事件 B 发生的概率。此项又被称为 A 关于 B 的“似然函数”。

$p(\mathrm{A})$ 表示事件 A 发生的概率。此项又被称为“正向概率”或者“先验概率”。

$p(\mathrm{B})$ 表示事件 B 发生的概率。此项又被称为“证据因子”。

由公式 1-1 可知，先验概率和似然函数值的乘积与后验概率是正相关的。所以各组先验概率和似然函数值的乘积可以反映出各组后验概率的大小关系。

此外，$\frac{p(\mathrm{B}\,|\,\mathrm{A})}{p(\mathrm{B})}$ 被称为“调整因子”，它使得预估概率更接近真实概率。[3]

因此，可以将贝叶斯公式理解成公式 1-2。

后验概率　＝　先验概率 × 调整因子;　　（1-2）

此外，贝叶斯定理如下：

设 $\mathrm{A}_1,\mathrm{A}_2,\mathrm{A},...,\mathrm{A}_n,\mathrm{B}$ 为一些事件，$p(\mathrm{B})>0$，$\mathrm{A}_1,\mathrm{A}_2,\mathrm{A},...,\mathrm{A}_n$ 互不相交，$p(\mathrm{A}_i)>0$，$i=1,2,...,\mathrm{n}$，且 $\sum_i p(\mathrm{A}_i)=1$，则对于 $k=1,2,3,...,n$，公式 1-3 成立。

$$p(\mathrm{A}_k\,|\,\mathrm{B})=\frac{p(\mathrm{B}\,|\,\mathrm{A}_k)p(\mathrm{A}_k)}{\sum_i p(\mathrm{A}_i)p(\mathrm{B}\,|\,\mathrm{A}_i)} \tag{1-3}$$

由贝叶斯定理可知，贝叶斯定理可以为衡量多个假设的置信度提供定量的方法。[4]

贝叶斯算法对机器学习十分重要。这背后的深刻原因在于，现实世界

具有不确定性，事物与事物具有关联性，人类的观察能力具有局限性。适于做不精确推理的贝叶斯公式能在一定的程度上反映以上几个特性。

“朴素贝叶斯算法”之所以被冠以“朴素”二字，是因为它把要被组合的各个假设看成是相互独立的。[5](即认为A_a的发生与A_b的发生互不影响，其中$a,b\in\{1,2,...,n\}$且$a\neq b$)。

1.2 朴素贝叶斯分类器简介

分类器的工作就是把概念赋给实例。[6]而赋予“朴素贝叶斯算法”某些实际背景后，可以得到“朴素贝叶斯分类器”。

假设某个体有 n 项特征（Feature），分别为$F_1,F_2,...,F_n$。现有 m 个类别(Category)，分别为$C_1,C_2,...,C_m$。贝叶斯分类器所做的事是求公式 1-4 的最大值。

$$p(C_i|F_1,F_2,...,F_n)=\frac{p(F_1,F_2,...,F_n|C_i)p(C_i)}{p(F_1,F_2,...,F_n)}(i=1,2,...,m) \qquad (1\text{-}4)$$

由于$p(F_1,F_2,...,F_n)$是独立于类别的量，因此可以略去此项，不对它进行计算。因此要求$p(C_i|F_1,F_2,...,F_n)$的最大值可以转化成求$p(C_i|F_1,F_2,...,F_n)p(C_i)$的最大值。

朴素贝叶斯分类器是更进一步地假设所有特征都彼此独立，这一假设可用公式 1-5 表示。

$$p(F_1,F_2,...,F_n|C_i)p(C_i)=p(F_1|C_i)p(F_2|C_i)...p(F_n|C_i)p(C_i) \qquad (1\text{-}5)$$

而，个体最可能属于的类别是最大后验估计，它可用公式 1-6 表示。

$$C_{MAP}=\arg\max_{C_i\subset C} P(F_1|C_i)P(F_2|C_i)...P(F_n|C_i)P(C_i) \qquad (1\text{-}6)$$

公式 1-6 等号右端的每一项都可以从统计资料中得到。由此就可以计算出每个类别对应的概率，从而找出概率最大的那个类别。[7]

朴素贝叶斯分类器的分类正确率与独立性假设的满足程度有关。当个体的特征满足独立性假设时，分类的正确度较高，否则可能较低。 虽然这种特征间不存在依赖关系的独立性假设在现实中不大可能成立，但这种做法可以大大简化计算。并且朴素贝叶斯分类器在许多实际应用中还是有着较好的分类正确率。

朴素贝叶斯分类器的分类正确率与先验概率有关。因为先验概率是根据历史资料或主观判断所确定的概率，并未经过试验检验。为了尽量提高分类正确率，我们需要较准确地对这些先验概率进行掌握和估计。

朴素贝叶斯分类器的分类正确率与处理数据的容量大小有关。一般来说，处理数据的容量越大，分类的正确率会越高。

第二章　搭建朴素贝叶斯文本分类器的整体思路

2.1 朴素贝叶斯分类器与朴素贝叶斯文本分类器的对应关系

为“朴素贝叶斯分类器”中的概念赋予一定的实际背景后，可以对文本进行分类。朴素贝叶斯分类器与朴素贝叶斯文本分类器在概念上的对应关系如表 2-1 所示。

表 2-1 朴素贝叶斯分类器与朴素贝叶斯文本分类器在概念上的对应关系

	朴素贝叶斯分类器	朴素贝叶斯文本分类器
概念	类别 $C_1, C_2, ..., C_m$	各文本类别
	特征 $F_1, F_2, ..., F_n$	从待分类文本中提取的各有效词
	个体 （贝叶斯公式里没有直接体现这项，它可以理解为 $F_1, F_2, ..., F_n$ 的原始数据）	待分类文本
	分类结果 C_{MAP}	朴素贝叶斯文本分类器计算出的待分类文本所属的文本类别
	某类别的个体数占全部个体数的百分比 $P(C_i)$	某文本类别的文本数占全部文本数的百分比
	在属于某类别的前提下，某特征出现的概率 $p(F_1 \mid C_i)$	在属于某文本类别的前提下，某有效词出现的概率
备注	$C_1, C_2, ..., C_m$ 、 $F_1, F_2, ..., F_n$ 、 C_{MAP} 、 $P(C_i)$ 、 $p(F_1 \mid C_i)$ 均来自公式 1-6	

朴素贝叶斯分类器假设所有特征都彼此独立。而朴素贝叶斯文本分类器则假设任意一个词语不会影响另外一个词语在同一篇文档中的出现。

2.2 朴素贝叶斯文本分类器的总体程序流程图及其描述

朴素贝叶斯文本分类器的总体程序流程图如图 2-1 所示。

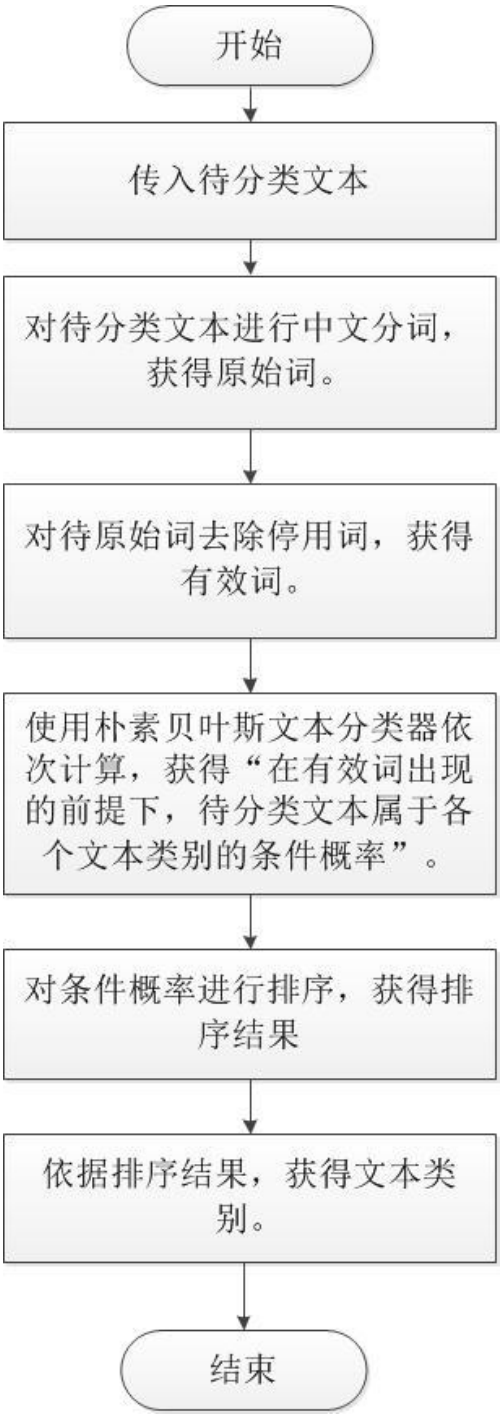

图 2-1 朴素贝叶斯文本分类器的总体程序流程图

对待分类文本进行中文分词、去除停用词、使用朴素贝叶斯文本分类器计算条件概率、对条件概率排序等系列操作后，可以对待分类文本进行分类。

第三章　朴素贝叶斯中文文本分类器具体实现

3.1 开发环境

本项目使用的开发环境如表 3-1 所示。

表 3-1 “贝叶斯文本分类器”开发环境

操作系统	Windows 7
开发语言 开发工具 开发工具包	Java Eclipse(Version:Kepler Service Release) je-analysis-1.5.1.jar lucene-core-2.2.0.jar

3.2 本项目的代码模块划分

本项目包含了待分类文本预处理、训练集管理器、朴素贝叶斯分类器主体、简单 Java 对象等核心模块，以及由用户图形界面、测试用例管理类、应用程序入口组成的算法测试模块。在本项目中，包与各个模块是具有对应关系的。本项目的各模块信息如表 3-2 所示。

表 3-2 “贝叶斯中文文本分类器”各模块信息

包名	备注
pretreatment	pretreatment 是预处理包。该包由： ChineseSpliter.java（用于中文分词）、 DictionaryManager.java（用于管理极易中文分词组件词库）、 StopWordsHandler.java（用于去除停用词） 等 Java 文件组成。
trainer	trainer 是训练集管理器包。该包由 TrainingDataManager.java（用于管理语料库，获取语料库各项数据）这个 Java 文件组成。
trunkOfNaiveBayesClassifier	trunkOfNaiveBayesClassifier 是朴素贝叶斯分类器主体包。其中， LikelihoodCalculator.java（用于计算似然函数）、 PriorProbabilityCalculator.java（用于计算先验概率）、 PosteriorProbabilityCalculator.java（用于计算后验概率）这三个 Java 文件体现了贝叶斯分类器的核心公式。而 CoreOfNaiveBayesClassifier.java（朴素贝叶斯文本分类器核心）与 ClassifyResultComparator（分类结果对象比较器）共同实现了图 2-1 表示的流程。

pojo	pojo 是简单 Java 对象包。该包由 ClassifyResult.java（分类结果对象）这个 Java 文件组成。分类结果对象具有类别名和条件概率这两个属性。
test	test 是算法测试包，该包由 Tester.java（应用程序的入口） MainWindow.java（测试用用户图形窗体） TestCaseManager.java（用于管理测试用例）等 Java 文件组成。

此外，项目中所调用的文件或文件夹信息如表 3-3 所示。

表 3-3 “贝叶斯中文文本分类器”所调用的文件或文件夹信息

文件或文件夹名	说明
TrainingSet	放在工程的根目录下，里面放置了搜狗实验室文本分类语料库 mini 版。该语料库里有 IT、财经、健康、教育、军事、旅游、汽车、体育、文化、招聘这 10 种文本类别的目录。每个目录下各有 10 篇该类别的 txt 格式文本。TrainingDataManager.java 文件能对这个目录进行管理。
TestCaseSet	放在工程的根目录下，里面放置了测试语料库。该测试语料库是从搜狗实验室文本分类语料库精简版中抽取出来的。TestCaseManager.java 文件能对这个目录进行管理。
stopWords.txt	放在工程的根目录下，里面放置了去除停用词模块所用到的停用词表。StopWordsHandler.java 文件能对这个文件进行管理。
wannaAdd.txt	放在工程的根目录下，里面放置了想在极易中文分词组件词库里出现的新词。（即使本来的词库已经包含了这个词，也可以将该词放在这里。） DictionaryManager.java 文件能对这个文件进行管理。
wannaDelete.txt	放在工程的根目录下，里面放置了不想在极易中文分词组件词库里出现的词。（即使本来的词库不包含这个词，也可以将该词放在这里。） DictionaryManager.java 文件能对这个文件进行管理。

3.3 中文分词模块实现

中文分词(Chinese Word Segmentation)指的是将一个汉字序列切分成单独的词。中文分词常常是中文自然语言处理的基础。中文分词有如下的显著特点：

1、与英文等语言不同，中文的词与词之间没有空格这样的明显分界符。

2、在中文里，“词”比“单字”的表义能力更强。

中文分词的方法体系众多，比如基于字符串匹配的体系、基于理解的体系、基于统计的体系等。[8]本项目使用的极易中文分词组件属于基于字符串匹配的体系里的最大匹配算法（Maximum Matching，简称 MM 算法）下的正向最大匹配算法。极易中文分词组件的优缺点如表 3-4 所示。

表 3-4 “极易中文分词组件”的优缺点

优点	缺点
支持英文、数字、简体中文混合分词 。	不是开源的。可拓展性比较弱。
支持常用的数量和人名的匹配 。	功能比较少。
词库超过 22 万。	分词速度比较慢。
使用方法简易。	不能依据语境来改变分词策略。
分词效果良好。	

极易中文分词组件所运用的分词算法是基于中文词典的匹配算法。并且在当代社会，流行语、新名词频出。因此，词库对该分词组件的影响是重大的，为此，本项目特地实现了对极易中文分词组件的词库增删词语的功能。

在这里举个例子，为中文分词模块传入“我们是厦门大学嘉庚学院的学生。”这段文本，由于极易中文分词组件的默认词库里面并没有“厦门大学嘉庚学院”这一个特征明显的专有名词，分词结果如下：

“我们/厦门/大学/嘉/庚/学院/学生/”。

显然，这样的分词效果会割裂词的含义，我们并不希望看到这种现象。此时我们可以为极易中文分词组件增加“厦门大学嘉庚学院”这个词汇。此时的分词结果如下：

“我们/厦门大学嘉庚学院/学生/”。

“增加词汇”的功能有助于提高分词效果。“增加词汇”与“删除词汇”的功能互相配合，有助于个性化词库的组建。

中文分词模块的方法列表如表 3-5 所示。

表 3-5 “中文分词模块”的方法列表

类名	类中主要方法名	方法备注
ChineseSpliter （中文分词器）	split	功能： 调用极易中文分词组件里的MMAnalyzer类对象的segment方法对待分词的文本进行中文分词。 传入方法的参数： String text（待分词文本） String splitToken（分割标记） 返回值： 分词后的文本（String）
DictionaryManager （词库管理器）	addMyWordsToDictionary	功能： 依据文件里的词语列表增加词语到极易中文分词组件的词库，以组建个性化词库 传入方法的参数： String filePath（文件路径） 返回值： void
	removeMyWordsFromDictionary	功能： 依据文件里的词语列表删除当前在极易中文分词组件里的词语，以组建个性化词库 String filePath（文件路径） 返回值： void

3.4 去除停用词模块实现

为了通过减少词的总数来提高程序的处理速度以及降低停用词对有效词造成的噪音干扰，本项目特地实现了“去除停用词”的功能。

应该注意的是,待分类文本预处理后获得的有效词必须具备足够的普遍性，即使常出现，也不能普遍到每一篇文档里都能找到。停用词不能设置得过多。否则，可能会破坏到有效词，进而影响到实验结果。并没有一

个明确的停用词表能够适用于所有的工具。在不同的应用背景下，停用词的含义是有差异的。例如，在基于词的检索系统中，停用词是指出现频率太高、没有太大检索意义的词。在自动问答系统中，停用词因其问题的不同而动态变化。在机器翻译、知识抽取中几乎没有真正的停用词，只是把频率太高的虚词作为临时的停用词特殊处理，切分完后仍要进行标记。[9]

经过考虑，本项目收集的停用词包括以下几部分：

（1）部分标点符号、数学符号及其它特殊符号

（2）部分频率特高的单汉字

（4）部分叹词

（5）部分拟声词

（6）部分方位词

（7）数字

在这里举个例子，仍旧为中文分词模块传入“我们是厦门大学嘉庚学院的学生。”。分词结果如下：

“我们/厦门大学嘉庚学院/学生/”。

我么知道，“我们”是一个特征很不明显的词，它的出现是不利于文本分类的，此时如果在项目中把“我们”这个词汇设置为停用词。去除停用词的结果如下：

“厦门大学嘉庚学院/学生/”。

我们可以发现，此时的中文文本分类器已经把“我们”过滤掉了。这将对文本分类产生积极影响。

去除停用词模块的方法列表如表 3-6 所示。

表 3-6 “去除停用词模块”的方法列表

类名	类中主要方法名	方法备注
StopWordsHandler （停用词处理类）	isStopWord	功能： 判别一个词语是不是停用词 传入方法的参数： String word（词语） 返回值： 词语是不是停用词 （boolean）

	dropStopWords	功能： 停用词过滤方法 传入方法的参数： String[] oldWords（未去除停用词的词语数组） 返回值： 已去除停用词的词语数组(String[])
	getStopWordsListFromFile	功能： 通过读取文件保存的停用词来初始化停用词表。 传入方法的参数： String filePath（文件路径） 返回值： 存放停用词的数组(String[])

3.5 训练集管理器模块实现

训练集管理器模块是朴素贝叶斯文本分类器不可或缺的一部分,因为它能为后者提供许多计算用数据。这些计算用数据来自于训练集管理器管辖的训练集。训练集是一组已经分配概念的实例。为了知道应该将哪个类别赋予待分类文本,朴素贝叶斯文本分类器需要读取 TrainingSet 这一组已经分配概念的实例。本项目使用的训练集是搜狗实验室文本分类语料库 mini 版。此外，我们也可以根据需求，个性化地定义训练集语料库。

训练集管理器模块的方法列表如表 3-7 所示。

表 3-7 “训练集管理器”的方法列表

类名	类中主要方法名	方法备注
TrainingDataManager （训练集管理器）	getTrainingFilesAmountOfTrainingSet	功能： 返回训练集中的文本总数。 String filePath（文件路径） 返回值： 训练集中的文本总数

		（int）
	getTrainingCategories	功能： 返回由训练集的所有类别名组成的字符串数组。 返回值： 由训练集中所有类别名组成的字符串数组（String[]）
	getTrainingFileAmountOfTheCategory	功能： 返回训练集中在给定文本类别名下的训练文本数目。 传入方法的参数： String nameOfTheCategory（类别名） 返回值： 训练集中在给定文本类别下的训练文本数目（int）
	getAmountOfFilesWhichContainKeyInTheCategory	功能： 返回给定文本类别中包含有效词的训练文本的数目。 传入方法的参数： String nameOfTheCategory（类别名） String key（给定的有效词） 返回值： 给定文本类别中包含有效词的训练文本的数目（int）

	getFilesPath	功能： 根据训练文本类别返回这个文本类别下的所有训练文本的路径 传入方法的参数： String nameOfTheCategory（类别名） 返回值： 给定分类下所有文件的路径（String[]）
	getText	功能： 返回指定路径的文本文件内容 传入方法的参数： String filePath（给定的文本文件路径） 返回值： 文本内容（String）

3.6 朴素贝叶斯中文文本分类器主干模块实现

3.6.1 文本分类流程简述

朴素贝叶斯分类器主干模块可以调用中文分词以及去除停用词的类方法来获得一组可代表待分类文档的有效词组，接着依据这组有效词进行分类决策。朴素贝叶斯文本分类器处理有效词数组的大致程序流程图如图3-1所示。

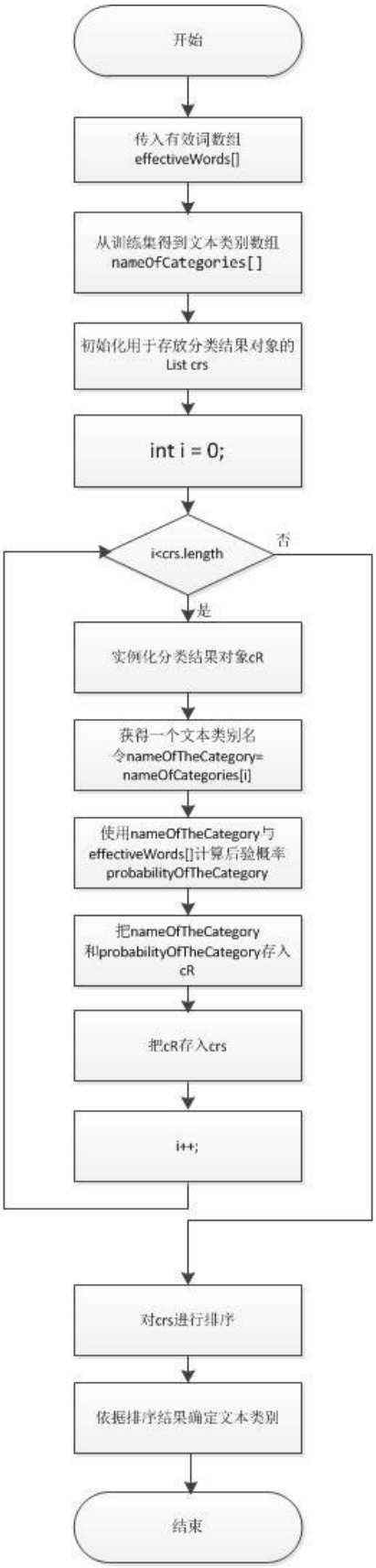

图 3-1 朴素贝叶斯文本分类器处理有效词数组的大致程序流程图

计算后验概率是朴素贝叶斯文本分类器的最关键步骤。而要计算后验概率需要先计算先验概率以及似然函数值。

3.6.2 计算先验概率

先验概率是根据历史资料或主观判断所确定的概率[10]，未经试验检验。有时不容易被确定。本项目使用的先验概率计算公式是公式 3-1。

$$p=\frac{\text{训练集中某文本类别下的文本总数}}{\text{训练集的文本总数}} \tag{3-1}$$

计算先验概率的方法列表如表 3-8 所示。

表 3-8 “计算先验概率”的方法列表

类名	类中主要方法名	方法备注
PriorProbabilityCalculator （先验概率计算器）	calculatePriorProbability	功能： 计算先验概率 传入方法的参数： String nameOfTheCategory （类别名） 返回值： 从训练集中随机抽出一篇文章，该文章属于类别名的概率值（double）

3.6.3 计算似然函数值

计算似然函数值可以说是计算后验概率的最关键步骤。对有效词数组里的每个有效词的似然函数值进行累乘可以得到整个有效词数组的似然函数值，它也是用来代表整段待分类文本的似然函数值。

计算单个词语的似然函数值是这部分计算的难点和重点。我认为，在编写似然函数值的计算公式时，应该就待分类文本的特点做适当的调整。本项目使用的计算单个词语的似然函数值计算公式是公式 3-2。

$$p=\frac{(\text{训练集给定分类下的包含当前词语的文本个数}+1)\times\text{调整因子}}{\text{训练集给定分类下的文本个数}+\text{训练集里的文本总数}} \tag{3-2}$$

在这里，分子项加 1 的原因是为了防止出现训练集在给定文本类别下包含当前有效词的训练文本个数为 0 的情况会使得整段待分类文本的似然函数值为 0。乘以调整因子并不会影响程序的排序结果，并且它有助于观察实验数据和防止运算结果溢出。

计算似然函数值的方法列表如表 3-9 所示。

表 3-9 “计算似然函数值”的方法列表

类名	类中主要方法名	方法备注
LikelihoodCalculator（似然函数值计算器）	calculateLikelihoodOfTheWord	功能： 计算给定的词似然函数值 传入方法的参数： String word（给定的词） String nameOfTheCategory（给定的文本类别名） 返回值： 似然函数值（double） 其他备注： 在本贝叶斯文本分类器中，该项表示的是，在给定的文本类别中，给定的词出现的概率。
	calculateLikelihood	功能： 计算有效词数组的似然函数值 传入方法的参数： String[] words（给定的有效词数组） String nameOfTheCategory（给定的文本类别名） 返回值： 有效词数组的似然函数值（double） 其他备注： 该方法通过 for 循环调用 calculateLikelihoodOfTheWord 方法来实现累乘功能。

3.6.4 计算后验概率

将前面所得的计算似然函数值结果与计算先验概率结果相乘即为我们所求的后验概率。

计算后验概率的方法列表如表 3-10 所示。

表 3-10 “计算后验概率”的方法列表

类名	类中主要方法名	方法备注
PosteriorProbabilityCalculator（后验概率计算器）	calculatePosteriorProbability	功能： 计算后验概率 传入方法的参数： String[] words（由待分类文本中提取的有效词组成的数组） String nameOfTheCategory（给定的文本类别名） 返回值： 后验概率（double） 其他备注： 该方法会调用 LikelihoodCalculator 里的 calculateLikelyhood 方法以及 PriorProbabilityCalculator 里的 calculatePriorProbability 方法来计算后验概率。

3.6.5 排序

如果将各组先验概率和似然函数值的乘积标准化，我们可以得到后验概率分布，这有助于分析数据。但进行标准化的操作对贝叶斯文本分类器来说是不必要的。贝叶斯文本分类的最后一步是对前面求得的条件概率进行排序。

排序的方法列表如表 3-11 所示。

表 3-11 “排序”的方法列表

类名	类中主要方法名	方法备注
ClassifyResultComparator（分类结果比较器）	compare	功能： 比较两个分类结果对象的概率属性的大小。 传入方法的参数： Object arg0（类对象）

		Object arg1（类对象） 返回值： 1、-1、0
java.util.Collections	sort	根据指定比较器产生的顺序对指定列表进行排序。

第四章　测试朴素贝叶斯中文文本分类器

在文本分类器设计完成之后，我们还对它进行了测试。经过测试后，我们可以依据实验数据来评估文本分类器的性能。我们也可以依据实验数据来对文本分类器进行补缺补漏。在分类器投入到正式使用之前，对训练集的调整以及对文本分类器的测试可能要重复多次。

测试朴素贝叶斯分类器的方法列表如表 4-1 所示。

表 4-1 “测试朴素贝叶斯分类器”的方法列表

类名	类中主要方法名	方法备注
Tester （测试类）	main	应用程序的入口
MainWindow （测试用 GUI 界面）	actionPerformed	功能： 响应在测试用 GUI 界面发生的事件
TestCaseManager （测试用例管理类）	getTheCorrectRateOfClassify	功能： 计算由若干个文本组成的测试用例集经贝叶斯文本分类器分类后的分类正确率。（该方法默认测试用例集中的文档属于同一文本类别，且这些文本的文本类别与传入方法的字符串都是它们的上一层文件夹名） 传入方法的参数： String nameOfTheCategory （测试用例集里的文章所属的正确的文本类别名） 返回值： Void 其他说明： 测试用例管理类与训练集管理类里的方法非常相似，因为它们所管理的文件目录的结构是一致的。

为了测验文本分类器的分类正确率，我设计了以下两个实验。

实验 1：测试长文本的微平均准确率

实验 2：测试短文本的微平均准确率

微平均准确率的计算公式是公式 4-1。

$$micro-p=\frac{\text{所有分类正确的文本数}}{\text{总文本数}}\ [11] \tag{4-1}$$

由于在实际应用的时候，文本分类器面对的数据可能与训练集语料库有区别。因此本实验选用的训练集语料库与测试用例的来源是略有不同的。训练集语料库是搜狗实验室文本分类语料库 mini 版，测试用例来自搜狗实验室语料库精简版。后者的文本份数会多于前者。实验用的测试文本类别包括 IT、体育、教育、军事、文化这 5 个类别。每个类别各有 25 篇文章。实验 1 的测试文本是直接从搜狗实验室文本分类语料库精简版中随机抽取的全文。实验 2 测试用的文本与实验 1 的文本来源相同，只不过我只保留了每篇文章的第一个句子。

实验 1 的分类效果实验记录如表 4-2 所示。

表 4-2 “实验 1”实验记录表

	IT	体育	教育	军事	文化
测试文本数	25	25	25	25	25
分类正确文本数	11	16	18	15	1
分类错误文本数	14	9	7	10	24
分类正确率	0.44	0.64	0.72	0.6	0.04

各去掉一个分类正确率的最高值和最低值，在实验 1 中，贝叶斯文本分类器的微平均准确率为 0.56。

实验 2 的分类效果实验记录如表 4-3 所示。

表 4-3 “实验 2”实验记录表

	IT	体育	教育	军事	文化
测试文本数	25	25	25	25	25
分类正确文本数	1	12	16	22	1
分类错误文本数	24	13	9	3	24
分类正确率	0.04	0.48	0.64	0.88	0.04

各去掉一个分类正确率的最高值和最低值，在实验 2 中，贝叶斯文本分类器的微平均准确率为 0.39。

我对实验现象的分析如表 4-4 所示。

表 4-4 实验现象分析表

现象	推测原因
“文化”类别的文章分类正确率极低。	搜狗语料库里的文化类文章里有不少是比较生活化的小说。这些文字的特征很不明显。
“IT”类文章的文章分类正确率要明显低于“体育”、“教育”、“军事”类文章。	搜狗语料库里的 IT 类文章里有不少是科技新闻。这些文字的题材较为宽泛，因此特征不大明显。
实验 1 的分类正确率一般会高于实验 2 的分类正确率。	实验 2 的待分类文本长度短于实验 1 的待分类文本长度，文本特征仍暴露得比较不明显。增大待分类文本的长度可能有助于提高本分类器的分类正确率。
无论是实验 1 还是实验 2，微平均准确率都比较低。	本分类器不能很好地对文本的特征进行提取。似然函数值的计算方法也有待改进。本分类器的训练集语料库规模还比较小。

此外，我认为提高训练集语料库的质量也能提高分类器的性能。我认为搜狗实验室的语料库尚有改进空间。比如，不应该让“科学家在百慕大海域新发现一种栉水母……”这样的文章在“IT”类别里出现。比如，可以细化训练集语料库的类别。例如，为“体育”类别下设“篮球”、“足球”、“羽毛球”等子类别。但训练集语料库不是影响分类器好坏的最关键因素。

第五章 展望

概率论是一门与现实生活紧密相连的学科，许多事件的发生都是有一定的随机性、规律性。因此在给予适当的建模后，朴素贝叶斯文本分类器将会有很广的应用领域。

朴素贝叶斯中文文本分类器可以应用到搜索引擎上。例如，在进行相关性排序时，可以根据网页类型做出不同的排序动作。在做页面信息抽取时，可以根据页面分类的结果采用不同的抽取策略[12]。在做检索意图识别时，可以根据用户的历史搜索记录改进搜索结果。贝叶斯文本分类器既能降低服务器压力又能优化用户体验。

朴素贝叶斯中文文本分类器也可以应用到推荐系统上。例如在 Web 2.0 应用中分析某用户的历史发言记录，分类器可以猜测出该用户的性格特征、兴趣爱好等信息。接着，Web2.0 应用可以利用这些信息对用户进行推荐操作。

我们生活在一个资讯爆炸的年代，各种数据围绕着我们的生活。我们有理由相信，贝叶斯文本分类器能在有更多更大的平台大显身手。

结论

在人工智能领域，人们感兴趣的是如何给随机变量分配一个概率值。[13] 为随机变量分配一个概率值是贝叶斯算法的强项。贝叶斯算法的基本思想并不算很复杂。它成功的一大原因来自计算机强大的计算能力。

经过探究，我觉得本项目的主要优点是：

1.实现了中文分词时的自定义个性化词库的功能。

2.实现了去除停用词时的自定义停用词表功能。

3.在贝叶斯文本分类器主干模块中，Java 类的组织方式与朴素贝叶斯文本分类器的核心公式比较一致，代码可读性较高。

我发现自己实现的文本分类器分类速度很低，分类正确率也有待提高。这个问题会直接影响系统能否投入实际应用。本系统中存在大量的可改进之处。经过考虑后，我对本项目实现的朴素贝叶斯中文文本分类器提出以下几点改良方案：

1.增设主题词增加权重方案，例如某段待分类文本中有“软件工程”、“Java”、“罗技”等词汇，那么这段待分类文本被分类为“IT”的概率应该得到适当的放大。

2.使用自动文本摘要技术来高效地提取待分类文本中的特征（有效词）的个数，以期大大降低程序的时间复杂度及空间复杂度并提高判断正确率。

3.对训练集也进行一定的加工处理，比如进行中文分词、去除停用词等操作。

4.充分挖掘训练集里的信息并将这些信息持久化，以期减少 I/O 开销和计算浪费。

5.对计算似然函数值的方法进行调整，例如改“不考虑单词在文档中的出现频次，仅考虑单词在文档中是否出现”的方法为“考虑单词在文档中的出现频次”。

6.继续完善去除停用词模块，构建更合理的停用词表。

7.尝试使用更高效的数据结构与算法。

参考文献

[1]林子雨.大数据技术基础[EB/OL].[2013-9].
http://dblab.xmu.edu.cn/sites/default/files/files/linziyu-BigData-Book-All.pdf.
[2]刘未鹏.数学之美番外篇:平凡而又神奇的贝叶斯方法[EB/OL].[2008-9-21].
http://www.mindhacks.cn/2008/09/21/the-magical-bayesian-method/.
[3]阮一峰.贝叶斯推断及其互联网应用（一）[EB/OL].[2011-8-25].
http://www.ruanyifeng.com/blog/2011/08/bayesian_inference_part_one.html.
[4](美)Mitchell,T.M.机器学习[M].曾华军等译.北京:机械工业出版社,2003.1.
[5](美)Segaran.T.集体智慧编程[M].莫映,王开福译.北京:电子工业出版社，2009.1.
[6](美)Marmanis.H,Babenko.D.智能 Web 算法[M].阿稳,陈钢译.北京:北京工业出版社,2011.7.
[7]阮一峰.朴素贝叶斯分类器的应用[EB/OL].[2013-12-16].
http://www.ruanyifeng.com/blog/2013/12/naive_bayes_classifier.html.
[8]吴军.数学之美[M].北京:人民邮电出版社.2012.6.
[9]化柏林.知识抽取中的停用词处理技术[j].现代图书情报技术,2007:48-51.
[10]肖筱南.现代信息决策方法[M].北京:北京大学出版社.2006.10.
[11]康恺,林坤辉,周昌乐.基于主题词频数特征的文本主题划分[J].计算机应用,2006,第 26 卷第 8 期.
[12]腾讯大讲堂.搜索引擎中的网页分类技术[EB/OL].[2013-9].
http://djt.qq.com/article/view/18.
[13]马少平,朱小燕.人工智能[M].北京:清华大学出版社,2004.8.