单位代码	10445
学　　号	2016313004
分 类 号	O0213
学习方式	全日制

山东师范大学

硕士学位论文

（专业学位）

论文题目　基于朴素贝叶斯的中文文本分类及Python实现

专业学位名称　应用统计硕士

方向领域名称　应用统计

申请人姓名　张航

指导教师　房莹　副教授

论文提交时间　2018　年 03 月 19 日

独 创 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得_____（注：如没有其他需要特别声明的，本栏可空）或其他教育机构的学位或证书使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：

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，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权__学校__可以将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、汇编学位论文。（保密的学位论文在解密后适用本授权书）

学位论文作者签名：　　导师签字：

签字日期：2018年5月15日　　签字日期：2018年5月15日

单 位 代 码	10445
学　　号	2016313004
分 类 号	00213
研究生类别	全日制

山东师范大学

硕士学位论文

（专业学位）

论文题目　基于朴素贝叶斯的中文文本分类及 Python 实现

学科专业名称　　应用统计

申请人姓名　　张航

指 导 教 师　　房莹　副教授

论文提交时间　　2018 年 03 月 19 日

摘　要

当前，随着计算机不断普及以及互联网的快速发展，在这个新科技知识不断涌现和拥有空前规模信息量的“信息爆炸时代”；信息来源渠道极广，传播速度极快，浩如烟海的信息鱼龙混杂，在短时间内能从中获取有用的信息成为人们迫切的需求。为了满足人们的需要，应运而生了文本数据挖掘中的中文文本分类方法，它是将统计方法与机器学习方法结合应用于文本分类中。中文文本分类是根据文本内容的主题词等属性特征将其划分到用户根据需求定义的相应类别中，一般是通过输入文本的特征向量，得到输出结果文本分类类别。

本文首先介绍文本分类的研究背景、国内外研究现状以及这种方法实际应用的价值，然后介绍了中文文本分类的理论分析流程以及朴素贝叶斯分类器和逻辑回归分类器的理论思想。实验阶段选取“搜狗语料库”下 5 个类别的新闻数据按照理论流程用 *Python* 的集成环境 *anaconda* 进行编程操作。先对数据集进行分词和删除停用词处理，然后将 *TF-IDF* 与 *N-Gram* 结合进行特征降维处理，先后构造朴素贝斯分类器和逻辑回归分类器进行中文文本分类，为使得分类器性能指标中的精确率、召回率等能够更加精确一点，使用了交叉验证方法，最后还对分类器的最优参数进行了寻找。经过对比发现朴素贝叶斯分类器的分类效果更好一些。

关键词：中文文本分类，*TF-IDF*，朴素贝叶斯分类器，*Python*

Abstract

At present, with the continuous popularization of computers and the rapid development of the Internet, there is a constant emergence of new scientific and technological knowledge in this era of information explosion with an unprecedented amount of information. The sources of information are extremely broad, and the speed of dissemination is extremely fast. It has become an urgent need for people to obtain valuable information from vast information in a short period of time. In order to meet people's needs, a Chinese text classification method in text data mining has been developed. It is a combination of statistical methods and machine learning methods applied to text classification. Chinese text classification is based on attribute features such as topic words of text content, and the corresponding text is divided into categories defined by the user according to requirements. Generally, the text classification category of the output result is obtained by inputting the feature vector of the text.

This paper first introduces the research background of text classification, research status at home and abroad, and the practical application value of this method. Then it introduces the theoretical analysis process of Chinese text classification and the theoretical thinking of naive Bayes classifier and logistic regression classifier. In the experiment stage, the news data of the five categories under the “Sogou Corpus” was selected, and then the corpus was programmed according to the theoretical process of Chinese text classification using Python's integrated environment anaconda. Firstly, word segmentation and deletion word processing were performed on the data set. Then TF-IDF was combined with N-Gram to perform dimensionality reduction processing. The Naïve Bayes classifier and logistic regression classifier were constructed to classify Chinese texts. In order to make the precision and recall rate of the classifier performance indicators more accurate, a cross-validation method was used. Finally, the classifier's optimal parameters were searched. After comparison, it was found that the naive Bayes classifier has better classification effect.

Key Words:Chinese text classification,TF-IDF,Naive Bayesian classfier, Python

目 录

第一章 绪 论

1.1 研究背景与意义

随着计算机的发展以及宽带提速降费，越来越多的人都融入到互联网这个大家庭。由于参与人数众多外加强有力的技术支持、信息收集交换传播速度极快的互联网将用户带入到信息网络时代。网络时代的特点是信息来源渠道广泛、巨大的信息容量、查询速度以及传播速度、更新速度都达到了空前的规模。尽管它使得用户获得信息变得更加便利以及可以看到来自世界各地各色各样的新闻娱乐科技等消息。但是随之而来也带来一些负面效应：如对于铺天盖地各种来源渠道的信息，用户要从中精准的获得真正对自己有价值的信息变得越来越难。这对于信息检索技术的要求变得越来越高，由此过去对信息的手动分类已经不能满足当前的用户需求，基于此，自动文本分类技术应运而生。

虽然网络上的信息呈现形态和模式是多样化的，但绝大多数都是以文本形式存在的，从中甄别和提取关键的文本信息变得至关重要。因而如何对原始的文本信息库进行有组织的管理成为应对信息爆炸的关键。文本数据挖掘中的文本分类技术是一项实用价值较高、应用领域广泛的技术之一[1]。

文本分类技术就是将体现文本内容的属性特征根据一定的规则将其划分到用户预先定义的不同类别中，使每个类别对应不同的主题，进而帮助用户更快捷、更高效地检索真正需要的信息。文本分类技术在英文和中文分类上有一定的区别，本文主要是讨论中文文本分类相关技术。根据市场的需求，对文本分类技术要求也在不断深入研究，伴随理论技术的发展，它在各领域的应用也变得越来越广泛，如：搜索引擎、论坛舆情分析、数字图书馆、垃圾邮件过滤、新闻分发等[2]。

1.2 文本分类研究现状

1.2.1 国外研究现状

在 20 世纪 50 年代末到 80 年代间是文本分类理论研究阶段：*H.P. Luhn* 第一次提出词频统计思想[9]，并将文本内容中的词和对应文本建立索引机制进行匹配，开启了词频进行分类处理的先河[3]。随后，*Maron* 和 *Kuhn* 发表的《*On Relevance，Probabilistic*

Indexing and Information Retrial》首次提出文本自动分类，开始了文本自动分类作为独立研究课题良好的开端[4]。上世纪 70 年代，*Salton* 在关于信息检索方面的论文中提出了向量空间模型（*VSM* ）[5]。此阶段的研究在信息检索领域应用较多。

20 世纪 80 年代到 90 年代期间，文本分类主导方法是以知识工程为准则，即利用各领域权威专家建立的规则进行手工分类，虽然准确率高，但是较为耗时耗力，并且不同领域之间指定的规则不能平行的被移植，因此人们对于效果更好效率更高的技术需求变得更加迫切[14]。

在 20 世纪 90 年代开始之后，随着信息爆炸式的不断增加，计算机网络以及机器学习等技术的不断成熟，以前传统的文本分类技术以及手工分类已经不能满足用户日益增长的需求。顺应时代发展与需求，与机器学习方法结合的文本分类技术快速的崛起。出现了基于贝叶斯算法的关键词提取，以及以文本中同义词词典作为基础的贝叶斯网络模型[17] [4]。2010 年，*Sriram* 等人提出了针对微博等短文本分类的特定领域特征提取方法[4]。*Malliaros* 和 *Skianis* 等提出了通过将文本中不同特征之间的关系进行编码，以图的形式呈现，最后再基于图的特征进行加权以此用于文本分类中[28]。2015 年，*Javed* 等人提出了先通过计算信息增益对特征排序，再利用马尔可夫链过滤器对排序的特征进行降维筛选的二阶段特征选择法[4]以及后续有各学者专家对人工蜂群算法的文本分类研究。在转入到以机器学习为主的文本分类研究阶段，效率和准确度都得到了大幅度的提高，并且在不同领域之间相应算法可以相互被移植应用，算法在不同领域之间的限制变得越来越小。

1.2.2 国内研究现状

我国在文本分类的相关研究中尽管起步比较晚，但是基于国外对英文文本分类的技术成果进而探索创新应用在中文文本分类中发展还是非常迅速的，产生了许多优秀的研究成果。在 1981 年，侯汉清教授对国外的文本分类相关工作进行了概要总结，并对在图书馆文献分类方面应用的自动分类技术进行了介绍[4]。1998 年，新闻语料汉语文本自动分类模型产生[1]。还有很多研究者利用各种参数的结合改进朴素贝叶斯算法进行垃圾邮件过滤等文本分类处理。唐华和曾碧卿提出了先用信息熵来生成初始种群，再采用优化的遗传算法提取分类规则的文本分类算法[5]。王超学等人利用 *K-Means* 算法对训练样

本集进行聚类，得到了改进的加权 *KNN* 文本分类算法[5]。2009 年，黄秀丽等人强调了低频特征在中文文本分类中的作用[1]。2013 年张玉芳等人提出了综合特征的文档频率、平均词频等四项因素在内的新的特征选择方法[1]。还有其他学者等通过提取主体特征进而对文本分类标记、对卷积神经网络以及多层潜在狄利克雷分配模型在文本分类应用方面不断的深入探索也取得了丰硕的研究成果。

发展至今，结合机器学习算法的中文文本分类方法已经不断成熟，形成了以朴素贝叶斯概率分类、以模糊聚类为基础、基于神经网络分类、基于仿生模式识别分类的比较完整、完善的中文文本分类体系[1]。

1.3 论文的组织安排

本文的核心内容是通过 *TF-IDF* 结合文本特征 *N-Gram* 对特征词进行降维处理，再根据入选的对分类有较大影响的特征词项利用朴素贝叶斯分类器和逻辑回归分类器进行中文语料库文本分类。实验过程中还使用了交叉验证法，对通过交叉验证构造的多个模型的评价指标值取平均作为最终该分类器的评价指标值。最后还对贝叶斯分类器的最优参数进行了查找。整个实验过程利用 *Python* 的集成环境 *anaconda* 完成编码操作。

第一章绪论：此部分主要介绍本论文主题，描述了文本分类的研究背景与实际意义，概要的总结了文本分类在国内和国外的发展进程，然后对本论文的整体布局安排加以描述。

第二章中文文本分类理论综述：此部分先介绍了中文文本分类理论上的一般流程，然后对中文语料库理论流程中预处理和特征处理环节中的所要用到的理论知识和方法进行了详细的描述。

第三章分类器构造：介绍了朴素贝叶斯分类器和逻辑回归分类器的理论内容以及常用的分类器性能评判指标等，并对交叉验证方法进行了相关的描述。

第四章实验与结果分析：根据第二三章中文文本分类的理论流程结合实际情况运用 *python* 代码对样本数据集“搜狗语料库”进行实验分析。

第五章总结与展望：此部分是对本论文的总结与思考，对本论文从背景介绍到理论分析以及实验分析部分进行了全方位总结。并针对本文进行文本分类过程中采用的各种

方法在实验结束后发现的可以选择更优的方法进行了相关的概述，明确了在以后学习和工作中需要努力的方向。

第二章 中文文本分类理论综述

文本分类是文本数据挖掘的方法之一。文本数据挖掘一般是指从大量的非结构化文本信息中通过一定的统计方法与机器学习方法结合获取用户需要的、感兴趣的但事先未知的、可理解的，最终可用的知识以及信息的过程。它将原本为非结构化的数据通过转化为计算机可识别的结构化数据，然后按照事先制定的规则程序进行自动分析进而得到处理结果，最终将处理结果与实际背景知识结合进行组织分析和应用。

通常数学上的定义为：在给定文档集$\mathrm{D}=\{d_1,d_2,\cdots,d_n\}$和类别集$\mathrm{C}=\{c_1,c_2,\cdots,c_m\}$上，文本分类就是找到从文档集到分类集之间的映射函数f，f: $d_i \rightarrow c_j, i=1,2,\cdots,n; j=1,2,\cdots m$[23][24]．此映射函数即就是从训练集上构造的分类器，若d_i经过分类器 f 判定属于c_j类，当符合实际情况时，则d_i成为类别c_j的正例，否则为负例，通过最终分类结果中正例和负例的个数可以计算精确率、召回率等评价指标。

2.1 中文文本分类流程

文本分类，顾名思义就是将原始待分类的文本信息按照事先指定的规则要求，划分到根据用户需求设定的类别中去的过程[25]。实验过程中，一般将数据集分为两类：一类是训练文档集，它是用来训练一个分类器，训练集不仅要确保该分类器通过交叉验证方法得到的各超参数有合适取值并且保证该分类器具有一定的泛化能力；另一类是测试文档集，它是用来测试训练集上学习到的分类器的泛化性能优劣程度，一般通过准确率、召回率、*F1* 值等评判指标衡量。

一般通过网络等渠道直接获取到的原始文本信息集会以多种多样的方式呈现：其中可能会有乱码排列，网址视频链接以及其他与文本内容无关的符号信息等，这种对反映文本主题信息无关的干扰特征项种类繁多，并且被分类的原始文本对象是计算机所不能识别的非结构化的数据，因此对包含训练文档集和测试文档集上的全部文档数据均需进行同等的预处理操作。对这两个数据集上的各处理如图 2.1 所示：

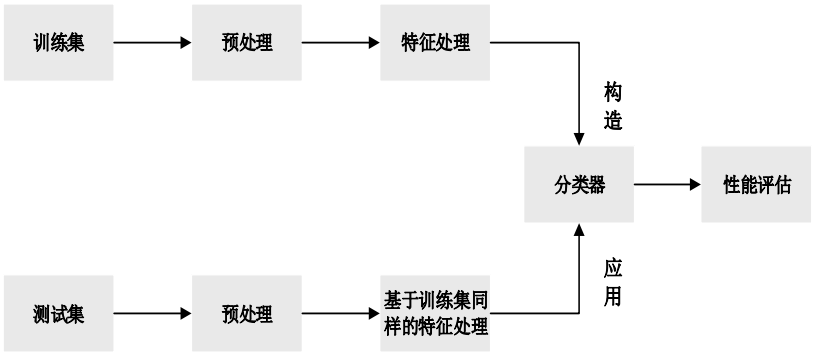

图 2.1 文本分类流程

从图2.1中可知，对全部文本数据集进行预处理后，先要在训练集上进行特征选择，目的是对文本特征词项进行降维处理；然后进行特征加权，对经过特征选择后保留的特征词项赋予不同的权重，权重大小表示该特征词项能使最终分类达到更好效果所占的比重大小，目的是使不同的特征词项因为重要程度而被区别对待，提高分类性能。到此可以初步构造分类器。为确保学习到的分类器不仅在训练集上有良好的分类效果，而且有较好的泛化能力，此时在训练集上会进行交叉验证，修正初始分类器各参数取值使其达到合适标准，这一步可以进一步提高分类性能。然后将构建好的分类器应用到测试集上，对测试集上的文档集进行分类，输出所属类别，并根据正例和负例个数等计算精确率、召回率，*F1* 值等评判指标，进而对分类器分类性能的优劣进行评判。

2.2 中文文本预处理阶段

文本预处理是文本分类任务的始发点，是使得后续步骤进行顺利的前提和基础，此过程对最终分类的效果和效率会产生重要的影响，所以是必不可少的阶段。对于全部数据集的文本信息，它的具体内容中不仅包含一些规范的纯文本信息，还存在着一些对分类没有任何价值的噪声数据如：视频网址链接、图片信息、标点符号、数字、特殊字符等。为了确保分类器的分类速度较快并且保证分类的准确率等指标较高，故必须过滤掉这些噪声数据[33]。

英文文本一般是通过空格将词与词进行分开，而中文文本是由字与字、字与词语等构造的没有空格的句子和段落，因此要将其按照一定的分词算法划分成以词语为最基本语言单位，以空格作为分隔符的文本内容，并且分词软件处理中文文本时还需专门设置

编码问题。因此中文文本和英文文本在分类任务中的预处理方式是有一些区别的。中文的预处理一般包括删除文本标记、对文本内容进行分词处理以及删除停用词等。

2.2.1 文本标记处理

文本标记是指出现在文本内容中的某些无关标签。现在获取到的文本数据一般都来源于网络，为了尊重版权作者等信息，文本内信息往往会在文章开头或文章结尾处标注新闻等来源链接地址、作者标签等信息，正文部分往往为了配合相应的文字说明还会附加上音乐、采访转载视频、图片等信息。然而这些标签对判定该文本内容最终属于哪一类别是不起作用的，即他们属于噪声数据，因此要删除这些标签，即删除文本标记。

删除文本标记不仅可以减少文本数据中的噪声，而且可以降低后续构造特征空间的维度，使得文本数据集中包含的规范纯文本信息越多，为更好的进行文本分类后续工作打下坚实的基础[3]。

2.2.2 文本分词处理

对于训练和测试文本集都需要进行同样的文本分词预处理。中文文本内容不同于英文文本内容中每个单词会以空格进行区分，一般获取到的中文文本集中文本内容都是以逗号句号等标点符号进行分隔的句子段落的形式呈现。中文文本分词的目的就是通过相应的分词方法把以连续的句子段落表示的文本内容划分为以空格作为分隔符号，并且以词语作为最基本的语言单位来表述的文本内容。目前有 3 种主流的中文文本分词方法：机械分词法，基于概率统计的，基于语义理解的分词算法。

（1）机械分词法

该方法也称为基于字符串匹配的分词算法。操作过程通常是：先根据语料库构建一个尽可能足够大的分词词典，然后将待分词文本内容句子中的各字符串与该分词词典中的词语进行匹配；若在该词典中能找到相应的字符串，则说明匹配成功，此时就将该字符串作为一个词从待分词的句子中切分出，否则就将单个字作为一个分词处理[29]。

机械分词方法的思想理解起来比较简单，实现过程也较为容易。但是因为它是将待分词的内容与事先构造的词典匹配，即对词典的依赖性过大，导致分词过程比较机械，并且在分词的过程中逐一匹配字符串的速度较慢，缺乏自主学习的能力[3]。

（2）基于概率统计分词法

它的基本思想是根据文本内容，认为相邻的字之间如果同时出现的频率很高，那么就很可能构成一个词语，随即就将其从待分词的文本内容中切分开。即认为相邻字出现频率的大小能较好的反映这两个字可组成一个词语的可信度高低。本文介绍语言模型和*N-Gram* 模型。

语言模型基本思想：若一句话 S 由 *n* 个词$d_1, d_2, \ldots\ldots d_n$组成，那么第 *n* 个词的出现只与这个词前面出现的 *n*-1 个词相关，而与其他任何词以及句子均无关。因此整个句子出现的概率就等于组成该句子的各个词出现的概率乘积，如下式（2.1）所示：

$$P(S) = P(d_1)P(d_2|d_1)\cdots P(d_n|d_1, d_2, \ldots, d_{n-1}) \quad (2.1)$$

该“语言模型”会产生两个问题：一是数据组合最后出现频数过于稀疏；二是对应数据有它的参数，那么过于稀疏的数据对应的参数空间会过大。

为了防止该“语言模型”的缺陷引入 *N-Gram* 模型，该模型是假设下一个词的出现只依赖它前面出现的有限个词即一个词或者几个词，而不是前面出现的所有词[39]。

如 $N = 2$，$P(S) = P(d_1)P(d_2|d_1)P(d_3|d_1, d_2)\cdots P(d_n|d_{n-2}, d_{n-1})$ （2.2）

实际分类中一般 *N*=1，2 或 3，效果还比较理想，当 N 取值高于 3 时会导致模型参数量级增大很多，但精度改善却不明显甚至可能降低。

该方法在使用过程中不需要根据语料库建立分词词典，而是直接根据上下文内容进行分词处理，实际使用价值较高。但是有一个缺点：因为它不考虑词的实际意义，造成两个相邻的字组成的词出现频率虽然高，但是不属于可以理解词语搭配范畴，即这样的高频组合词实际上是无意义的[40]。

（3）基于语义理解的分词法

该方法就是通过让计算机模仿人，基于人们平时阅读文章时如何停顿，如何根据上下文语义理解词语进而进行分词处理。由于中文语言组成博大精深，尽管这种方法比较智能，但是也不能保证分词就能达到很高的准确度，因此该方法仍旧处于不断深入探索阶段。

2.2.3 删除停用词处理

删除停用词：在文本分词的处理结果基础上，删除与文本主题内容无关的停用词。

“停用词”：一般包括助词（如：“啊”，“吧”），形容词，副词，大量重复出现的人称代词（如：“你”，“我们”，“它”，“他们”等），各种特殊符号（回车，空行等）以及标点符号（“，”，“．”，“￥”，“！”，“？”等）。这些词通常都是对文本主题以及分类基本不起作用的词，删除它们不仅可以节省存储空间，提高代码运行速度，还可以降低特征选择的维度，提高分类效率和准确度[27]。根据大量的实践，将各种可能出现的“停用词”组合在一张表中就构成了“停用词表”。

实际实验的过程中，在根据某一停用词表删除文本中的停用词之后，可以先统计一下文章的词频，结合实验内容，将可能的高频词但又对分类无用的词补充添加到停用词表中，然后利用新的停用词表再对文本数据集进行一次删除停用词操作。这一操作可以节省存储空间，降低特征空间维数，提高分类性能。

2.3 文本表示阶段

要将上述非结构化文本转换为计算机可理解的结构化二进制形式，须进行文本表示。本文介绍两种文本表示模型：布尔（*Boolean*）模型，向量空间模型（*VSM*）。

（1）*Boolean* 模型

布尔模型是将文本中每一个特征项对应一个只取 0 或 1 的二值变量，将原先用特征项表示的文本转换为用各种特征变量表示；如果该特征项在文本内容中出现，那么它所对应的变量取值为 1，否则为 0。在检索过程中采用精确匹配：即如果要检索 *A* 和 *B* 同时出现，若其中只出现一个，则匹配不成功。虽然表述简单，但是由于精确匹配过于严格，对于文本内容理解上就会导致缺乏灵活性，局限性较大。

(2) *VSM* 模型

VSM 模型：给定一个由 *n* 个文档组成的文本集合 $D=\{d_1,d_2,...,d_n\}$，再给定经过特征选择得到的一个 *m* 维特征项集 $T=\{t_1,t_2,...,t_{\mathrm{m}}\}$，对应 *m* 维的特征项集给定一个 *m* 维的权重向量集 $W=\{\omega_1,\omega_2,...,\omega_{\mathrm{m}}\}$，将原始文本集 *D* 转换为 $D'=\{W_1,W_2,...,W_n\}$，其中 W_i 表

示第i(i =1,2,..,n)个文档特征项对应的权重向量，每个W_i为 m 维的权重向量。如下表 2.1 所示：

表 2.1 VSM 转换原理

特征项 / 文档	t_1	t_2	…	…	t_m
d_1	w_{11}	w_{12}	…	…	w_{1m}
d_2	w_{21}	w_{22}	…	…	w_{2m}
…	…	…	…	…	…
…	…	…	…	…	…
d_n	w_{n1}	w_{n2}	…	…	w_{nm}

在向量空间模型中，去掉换行符的各文本内容实际上就是一行词袋（*bag of word*）：即在词向量空间中构建的词典不考虑语序以及文本的语法，仅将其看成单词的独立组合集合。将文档按照上述表格转换为向量时，对于文本之间的相似度可以通过数学方法进行计算。

余弦相似度计算：

$$\mathrm{Sim}(d_i, d_j) = \frac{\sum_{k=1}^{m} \omega_{ik}\omega_{jk}}{\sqrt{\sum_{k=1}^{m} \omega_{ik}}\sqrt{\sum_{k=1}^{m} \omega_{jk}}} \qquad (2.3)$$

式（2.3）计算结果表示文档d_i与文档d_j的相似度，计算结果值越大，表示相似度越大，将文本归类于与其相似度最大的已知的类别中。

2.4 特征处理阶段

经过上述两步骤处理过的新的文本集，尽管因为去除停用词的原因特征空间随之缩小，但是为了提高最终分类的效率和准确率，还须进一步降低特征空间的维数，所以要进行特征选择和特征加权处理。特征选择就是先通过一定的方法对特征空间中包含的特征词项重要程度进行量化，然后进行特征词项的选择。不同的特征选择方法有各自的侧

重点，卡方统计量是通过衡量特征词项与类别之间的关联关系，卡方值越大表明关联性越强，该特征应该被保留。信息增益是通过“条件熵”衡量，即计算特征词项能够为分类传递多少信息量，传递的信息量越大，表明该特征词项越重要，进而越应该保留。

特征加权是对已经保留的特征词项根据一定方法衡量它对分类的贡献程度，然后根据不同的贡献程度赋予不同的权重大小。

2.4.1 特征选择

特征选择的准则要求是：在不改变原始向量空间主题性质的条件下，去除掉多余噪声特征，即只保留从原始向量空间中选择的一部分对分类结果会产生重要影响且具有代表性的特征组成一个新的低维度特征空间；最后，这个新的低维特征空间能够确保文本分类的效率和准确率都能有所提高。

要进行特征选择，一般步骤为：

第一步：先将预处理分词后已转换为向量表示的文本内容作为最原始的特征集 T_1；

第二步：根据特征加权的方法计算 T_1 中每个特征对应的权重，并对权重大小按降序排列，将排序好的特征项作为新的特征集 T_2；

第三步：从 T_2 中选择权重排列靠前的 k 个特征项作为最终的特征集 T；

常用的特征提取方法包括：频率统计，*CHI* 统计量，信息增益、词袋模型等。

（1）频率统计：

频率统计包含两部分：一部分是词频词频（*TF*）统计，一部分是文档频率统计（*DF*）。

词频（*TF*）是指某一文本中某一个词语在该文本中出现的频率，公式表示即：

$$\mathrm{TF} = \frac{\text{某个词在该文章中出现的个数}}{\text{该文章的总词数个数}} \qquad (2.4)$$

事先设定一个阈值，将计算出的词频与之比较，如果 *TF* > 阈值，则该词可入选特征项集，否则就删除。此方法在进行特征选择时易理解、操作性强，效率高，但是它单纯的认为低于设定阈值的低频词对文本分类没有价值，过度重视高频词汇其实是比较片面的。实际上，有些低频词虽然出现次数少但它很具有代表性以至于它们刚好可以反映文章的主旨内容。因此，该方法一般会结合其他方法一块综合使用进行特征选择[1]。

文档频率统计（*DF*）是最为简单的一种特征选择算法，它指的是在整个数据集中有多少个文本包含这个单词。在训练文本集中对每个特征一一计算它的文档频次，并且根据预先设定的阈值去除那些文档频次特别低和特别高的特征[5]。

在训练文档集中对每个特征项计算它的文档频数，若某特征项的 *DF* 值大于阈值下限小于阈值上限，则将其保留，不在此范围的删除。删除的特征项分别代表了“没有代表性”和“没有区分度”2 种极端的情况。*DF* 的优点在于简单易理解，不仅计算量小，执行速度快，而且计算复杂度低，在大规模数据集的特征选择上有很好的效果。缺点是在整个语料库中可能是“稀有词”但它刚好又是某一类别文档中的代表词，即对某一类别有重要的判别价值，直接删除可能会影响分类器的精度[38]。

（2）*CHI* 统计量：

作为数理统计中常见的检验统计量 *CHI* 即卡方检验(*chi-squre test*，X^2)，它一般是用来检验两个事件的独立性。将其应用到本文中的中文文本分类中：即就是判断特征项与分类类别之间的是否具有关联关系。当式（2.5）计算结果卡方值越大，就表明要拒绝原假设，即认为特征项和类别之间有关联关系，进而表明该特征项对分类类别的判别价值越大，应将其保留并加入到最终特征属性集合当中。具体过程如下：

H_0：词t_i与类别c_j独立

$$X^2(t_i, c_j) = \frac{N \times (AD - BC)^2}{(A + C)(B + D)(A + B)(C + D)} \qquad (2.5)$$

在上述公式中，*N* 代表训练文本集的总文档数。*A* 为训练文档集中类别c_j包含词语t_i的文档数，*B* 为训练文档集中不是类别c_j但包含词语t_i的文档数，*C* 为训练文档集中类别c_j但不包含词语的t_i文档数，*D* 为训练文档集中不是类别c_j并不包含词语t_i的文档数。如下表 2.2 所示：

表 2.2 卡方统计量公式各字母含义

词类别	c_j	$\overline{c_j}$	总数
t_i	*A*	*B*	*A+B*
$\overline{t_i}$	*C*	*D*	*C+D*
总数	*A+C*	*B+D*	*N*

χ^2统计量比较了每个词项对一个类别的贡献与对其他类别的贡献，以及不同词项对分类的影响[5]。由公式可知，若$AD-BC>0$则表明该词项与类别呈现正相关，即该词出现那么某个分类类别也有可能出现；若$AD-BC<0$表明词项与类别呈现负相关，即该词出现那么某个分类类别很有可能不出现，当$AD-BC=0$时，表明特征与类别独立不存在相关关系[5]，此时χ^2=0。从卡方值计算公式可看出：若计算得到的卡方值较大，则意味着拒绝彼此独立的原假设，认为特征词项与类别具有相关关系。先设定一个阈值，将特征项中的所有词与各类别之间全部都计算卡方值，χ^2与阈值比较，若大于该阈值则保留该特征词，否则删除。保留的特征词构成最终的特征属项集。

尽管卡方检验在通过选择较大卡方值判断时进行了特征降维处理，但是卡方检验在实际应用过程统计消耗大，并且它对每个具体文本内容中的低频词有所偏袒，只考虑了某个词在所有文档集合中出现的文档个数，而没有考虑这些词在每个文本中出现的频数，使得单纯看每个文本中的低频词时，不管他在每个文本中出现的次数有多少，但是只要它出现，它的卡方值就和每个文本内容中同样出现的高频词得到一样的卡方值，即这种方法对低频词有所偏袒。所以，在实际应用中一般会和其他方法结合使用进行特征选择。

（3）信息增益

在文本分类中，信息增益实质上是针对每个特征词项的。先统计特征空间中有某个特征词项时和没有这个特征词项时该特征空间的“信息量”各是多少，然后计算两种情况的差值就是这个特征词项给这个特征空间带来的信息量，即信息增益。

假若类别 C 共有 $c_1, c_2, \ldots\ldots, c_n$ n 个不同的类别取值，此时整个分类系统的信息量计算如下式（2.6）：

$$H(C) = -\sum_{i=1}^{n} p(c_i)\log p(c_i) \qquad (2.6)$$

当固定某个特征词项 T 时即没有这个特征词项时的条件熵是：

$$H(C|T) = -p(t)\sum_{i=1}^{n} p(c_i|t)\log p(c_i|t) - p(\bar{t})\sum_{i=1}^{n} p(c_i|\bar{t})\log p(c_i|\bar{t}) \qquad (2.7)$$

那么该特征词项 T 给分类系统带来的信息增益IG(T)为：

$$\mathrm{IG(T)} = \mathrm{H(C)} - \mathrm{H(C|T)} \qquad (2.8)$$

式（2.7）中的$\mathrm{p}(c_i)$表示类别c_i出现的概率，实际计算时只要用 1 除以类别总数就可以得到（这样计算表示忽略每个类别的大小即平等的看待每个类别）；$\mathrm{p(t)}$是特征项 T 出现的概率，计算时用包含 T 的文档数除以总文档数；$\mathrm{p}(c_i|t)$表示包含特征项 T 的时候，类别c_i出现的概率，计算时用包含 T 并且属于类别c_i的文档数除以包含 T 的文档数。

从以上讨论中可以看出，信息增益也是考虑了特征项出现和不出现两种情况，是比较全面的，因此作为特征处理效果还不错。但信息增益缺点在于它只能考察特征项对整个系统的贡献，而不能具体到某个类别上，即每个类别都有自己的特征集合，而有一些词，对某个类别很有区分度，但对其他类别则没有太大价值[5]。

（4）词袋模型（*Bag-of-Words*）：

文本分类的的原始数据是非结构化的数据，在计算机上无法直接利用算法进行分析。词袋模型就相当于将整个数据集中含有的词放在一个袋子里，然后统计每个词在整个袋子中出现的频数。即整个原始数据集可以看做一个用词表示的文档矩阵，矩阵的每一行表示原始数据集某一类别的一个小文档，每一列是某个词在该文档中出现的频数。通过词袋模型，就将原始的文档数据集转化为以数值表示的矩阵，即该模型用词出现的频数代表文档内容，这一数值化过程称为向量化。

因为原始数据集每一类别下的文档中包含的词仅是整个数据集所有词的一部分子集，因此通过词袋模型表示的矩阵是稀疏的，矩阵的绝大多数位置上都是 0 值。

2.4.2 特征加权

经过特征选择后保留的特征词项，因为它们对文本内容的代表程度是各不相同的，此时须进一步根据每个特征词项对文本内容主旨贡献的重要程度赋予权重，特征词项的权重大小反映了该词项对文本内容主旨的贡献大小，而文本的主旨内容就是分类的依据。即不同特征词项的权重大小表示了它们对文本所属类别的区分能力[30]。

（1）布尔权重法：

将特征词项在文本内容中出现次数大于 0 的权值均设定为 1，出现次数为 0 的权值均被设定为 0。它没有考虑特征项是否对分类类别有影响，因此分类效果一般不理想，适用于特征项较少的情况[3]。

（2）*TF-IDF* 算法：

逆文档频率（*IDF*）：如果某些特征词比较少见，并且它几乎只在某个类别下的文档中出现，那么就说明这些特征词的类别区分能力很强[6]。这些特征词正是我们所需要的关键词，因为它是和整个文本集类别划分息息相关的，一般认为 *IDF* 越大越好。

$$\text{逆文档频率(IDF)} = \log\left(\frac{\text{语料库中的文档总数}}{\text{包含该特征词的文档数}+1}\right) \qquad (2.9)$$

$\text{TF-IDF} = \text{TF} * \text{IDF}$，从 *TF-IDF* 的定义式可知，它考虑了两方面：一是通过 *TF* 考虑了每个特征词在具体文本内容中出现的频率；二是通过 *IDF* 考虑了特征词在整个文本集下出现的文档个数。即特征词在某个文本中的权重与它在当前文本中出现的频率成正比，与整个文本集中包含该特征词的文档数量成反比[6]。*TF-IDF* 越大，表明包含该特征词的文本数量比较少并且该特征词在某个文本内容中属于高频词。

尽管 *TF-IDF* 权重可以对特征词划分类别的重要程度进行简单有效快速的衡量，但是，它在进行特征选择降维处理时会对高频词有所偏袒，即可能会过滤掉对文本分类有很大贡献的低频词或比较稀有的特征词，简单粗暴的认为它们对分类没有太大效用。即这种方法没有考虑到下述两种情况：

a.未考虑特征词项在不同类别之间的分布：如果一个词在各个类别之间出现比较均匀，这样的特征词对分类基本上起不到任何作用，即“无代表性”。因为按照公式（2.9）计算这些特征词结果没有差异，这样可能会造成维数选择过多的可能；

b.另外一种是某些特征词在某类文本集中可能出现频率很高，但在其他类中可能几乎不出现，这种被称为“稀有特征词”的词项实际就能代表这个类别的特征，恰好是分别类别的重要特征词；

2.5 本章小结

本章首先是对整个中文文本分类流程做了简要的综述。然后对分类流程中预处理和特征处理部分所涉及到的相关方法进行详细陈述。此章节为下一章节的分类器构造打下了坚实的理论基础。

对文本预处理阶段包含的三个过程：删除文本标记、中文分词的三种分词算法以及删除停用词操作进行了详细的论述，包括每一步处理对最终分词产生的效果和各分词算法优缺点。接着介绍了两种文本表示的方法：*Boolean* 类型表示和 *VSM* 模型表示；然后将特征处理阶段分为两部分：一是先进行特征选择，在此介绍了 4 种特征选择方法，分别是频率统计、信息增益、卡方统计量，词袋模型，并对每一种方法的优缺点以及改进之处进行了详细介绍；二是进行特征加权，对经过特征选择之后保留的特征词项进行加权处理，此处介绍了 2 种方法，分别是布尔权重、*TF-IDF*。

第三章 分类器构造

文本分类的核心问题是分类模型的构造。常用在文本分类中的方法有：逻辑回归（*LR*）、朴素贝叶斯，支持向量机(*SVM*)，卷积神经网络（RNN）等。本文采用监督学习中的逻辑回归和朴素贝叶斯算法，解决如何根据训练文档集学习构建一个分类器也称为分类函数，然后将其应用在测试文档集中，通过判断其分类的准确率、召回率等指标进行相关参数的调整。

3.1 文本分类算法之朴素贝叶斯算法

朴素贝叶斯分类（*NBC*）是以贝叶斯定理为基础并且假设特征条件之间相互独立的方法，先通过已给定的训练集，以特征词之间独立作为前提假设，学习从输入到输出的联合概率分布，再基于学习到的模型，输入 X 求出使得后验概率最大的输出 Y。

通过下式（3.1），可以求得基于互相独立的特征项t_i如果出现在某文档x中，它属于 c 类别的条件概率。

$$P(c\,|\,x)=\frac{P(c)\prod_{i=1}^{n}P(t_i\,|\,c)}{\prod_{i=1}^{n}P(t_i)} \tag{3.1}$$

其中：类别 c 的先验概率 $P(c)$是根据现有的文档集所得，$\mathrm{P}(t_i|c)$是t_i出现在 c 类别文本的条件概率；n是文档x所包含的词项数目。文本分类的目的是找出某文本内容最有可能属于的哪个类别，此处“最有可能”就是如何求得后验概率最大。此时使用最大似然估计（*MLE*）求$\mathrm{P}(c)$以及$\mathrm{P}(t_i|c)$，在最大似然估计下，类别 c 的先验概率$\mathrm{P}(c)=\frac{N_c}{N}$，其中$N_c$是训练集中 c 类别所含的文档总数量，$N$ 为训练集中全部类别下的文档总数；条件概率

$$P(t_i|\mathrm{c})=\frac{T_{\mathrm{ct}}}{\sum T_{ct}} \tag{3.2}$$

上式（3.2）中T_{ct}表示在 c 类别下包含特征项 t 的文档个数，$\sum T_{ct}$是 c 类别下所含文档总数目。

3.2 文本分类算法之逻辑回归

逻辑回归算法不是回归，它的本质是进行分类。

3.2.1 二分类逻辑回归

假设有 2 个类别，m 个影响分类的因素，$\omega_i(i = 1,2,\cdots,m)$分别代表每个影响因素的权重，则 m 个因素共同作用产生的影响可表示为：

$$z = \theta_0 + w_1\theta_1 + \omega_2\theta_2 + \cdots + w_m\theta_m \quad (3.3)$$

此时计算出的 z 值是任意实数，不能直接进行分类操作。然后利用 *sigmoid* 函数 $g(z) = \frac{1}{1+e^{-z}}$将 z 值映射到$[0,1]$的区间上，恰好是概率的取值的范围。

假设y代表类别，记其中一个正例类别为 1，负例类别为 0，一般是希望得到正例的概率越大越好。记预测为类别 1 的概率为：

$$p(y = 1|x;\theta) = h_\theta(x) = g(\theta^T X) = \frac{1}{1+e^{-\theta^T X}} \quad (3.4)$$

那么预测为类别 0 的概率为$p(y = 0|x:\theta) = 1 - p(y = 1|x:\theta)$，将其整合成类似二项分布的表达式即为：

$$p(y|x;\theta) = [h_\theta(x)]^y[1 - h_\theta(x)]^{1-y} \quad (3.5)$$

为了使估计的参数值可以接近真实值，一般会采用最大似然函数进行参数$\hat{\theta}$的估计，再将$\hat{\theta}$带入式（3.4）和式（3.5）得到属于类别 1 的概率值，若该值大于设定的阈值，则认为它属于类别 1，否则为类别 0。

3.2.2 多分类逻辑回归

在二分类的基础上有两种方式的改进可以实现多分类逻辑回归。

（1）建立多个分类器：

假设有 k 个类别，n 个样本和 m 个特征组成了$n \times m$矩阵，取其中的一个类别 c 标记为 1，其余的标记为 0，这样可以训练一个二分类逻辑回归分类器；对其他的类别重复这样的操作，最终会得到 k 个二分类逻辑回归分类器。当在测试集上应用时，每一个

测试样本都会通过 k 个分类器得到 k 个 $h_\theta(x)$值，从中选取最大值对应的类别作为该测试样本的类别归属如下式（3.6）：

$$\arg\max_c h_c(x)\ \ c = 1,2,\cdots,k \tag{3.6}$$

（2）*softmax* 回归：

多类型回归模型的输出结果为该样本分别属于 k 个类别的概率值，分类函数如下式（3.7），从式（3.7）的计算结果中选取最大值对应的类别作为测试样本的预测类别。

$$h_\theta(x^i) = \begin{bmatrix} p(y^i = 1|x^{(i)};\theta) \\ p(y^i = 2|x^{(i)};\theta) \\ \vdots \\ p(y^i = k|x^{(i)};\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix} \tag{3.7}$$

先利用式（3.8）最大似然函数对参数$\hat{\theta}$进行拟合，求解过程可以使用牛顿法或者梯度下降法，再将$\hat{\theta}$值带入式（3.7）中得出预测为各类别的概率值。

$$l(\theta) = \sum_{i=1}^{n} \log \prod_{l=1}^{k} \left(\frac{e^{\theta_l^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}}\right)^{1\{y^i = l\}} \tag{3.8}$$

3.3 交叉验证

交叉验证算法从总体上来讲是一个求稳的操作。因为一般情况下只将原始数据集划分为训练集和测试集时，仅仅用一个训练集训练分类器可能会导致模型的过拟合问题出现，在测试集上预测效果不好，即模型的“泛化能力”不理想。为了防止这种情况的发生，它将数据集均分，组合多种训练集学习模型，然后取多次训练模型指标值的平均值作为最终该分类器模型的指标值。二类别 3 折交叉验证如下图 3.1 所示：

k-折交叉验证一般步骤为：

（1） 将原始数据集先按照每一类别分别均分为 k 份，从每一类别下各取一份组成测试集，其余全部作为训练集；

（2） 将步骤 1 的操作执行 k 次；

（3） 每一次的抽取训练都会学习到一个分类器模型，一共会学到 k 个训练器模型，取 k 次相应评价指标的平均值作为最终该模型的指标评判值。

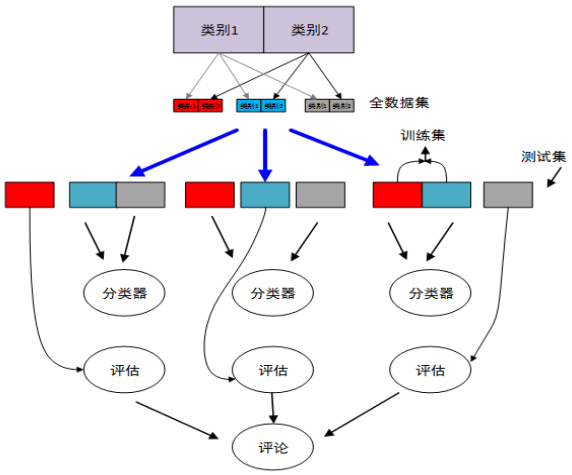

图 3.1 二类别 3 折交叉验证图

3.4 分类器性能评价

当通过训练文本集确定好最终模型分类器的参数后，将其应用在测试文本集上进行分类预测，将测试文本集上的预测类别值与真实类别值进行比较，计算相应的分类性能评估指标，进而对该分类器做出评价。常用的分类器性能评估指标有：精确率、召回率和 F_1 值。

精确率：对于给定的已知类别的测试数据集，分类器对其进行正确分类的样本数与总样本数之比。

召回率：对于给定的已知类别的测试数据集，指分类器正确判断该类的样本个数与属于该类的实际样本总数之比。

F_1 值：精确率和召回率的调和平均。

如对类别空间$\{c_1,c_2,...,c_k\}$中的一个类别对应精确率 P 和召回率 R 的公式以及它们的调和平均定义如下式（3.9）所示：

$$P=\frac{TP}{TP+FP},\quad R=\frac{TP}{TP+FN},\quad F_1=\frac{2TP}{2TP+FP+FN} \tag{3.9}$$

其中，TP 值表示正确分配到类别 c_i 的文本数，值 FP 表示不正确分配到类别 c_i 的文本数，值 FN 表示属于类别 c_i 但未分到该类别中的文本数。

另外有一种可将分类器分类性能形象化展示的工具即混淆矩阵，如下表 3.1 所示：

表 3.1　混淆矩阵

真实类别	预测类别	
	正例	负例
正例	TP	FN
负例	FP	TN

3.5 本章小结

本章首先把在中文文本分类中应用较广的朴素贝叶斯分类器和逻辑回归分类器的相关理论进行了相关的陈述；然后介绍了交叉验证法，它能够依据一定的规则将数据集分为多个训练集和测试集，得到多个分类器模型，进而取多个分类器模型评价指标的均值作为最终的该分类器模型的评价指标值，通过该方法可以提高模型的泛化能力；最后对精准率，召回率，*F1* 值 3 个评价分类器性能优劣的指标进行了相关的描述。

第四章 实验与结果分析

4.1 语料库

本文实验数据集来自搜狗实验室公开免费的搜狗新闻数据（*SougouCS*），数据集来源地址为：*http://www.sogou.com/labs/resource/cs.php*。本文下载的数据集是包含一个月新闻内容的精简版数据集（共计 347*MB*）。该数据共包含 18 个类别，本文从其中 “健康”、“教育”、“军事”、“旅游”，“娱乐”5 个类别下分别提取 2900 篇文章进行实验分析。

本文实验环境配置：*Windows32*，*Pycharm3.6*，*Anaconda3*。

样本数据集：从原始数据集 5 个类别中各取 2900 个*.txt* 文本，即总共 14500 篇新闻文档作为样本数据集，其中一篇“健康”类别下的新闻文档如下图 4.1 所示：

三餐补充蛋白质和纤维素，　相对碳水化合物和脂肪，蛋白质需要更多的时间去消化　相对碳水化合物和脂肪，蛋白质需要更多的时间去消化，所以能让人保持长时间的饱肚感。纤维素能帮助吸收水分，所以在胃中膨胀让人觉得很饱而没有空间塞下其他食物了。　非看不可，四分之三以上的读者看过这篇文章后都觉得很有用。　下面介绍的这几招，只要改变日常，就可以让你不再三餐大块哚饴、毫无节制。　给自己少点食物　选择在家里尽量少放些零食，而且可供选择的零食品种越少越好。研究发现，面对更多的选择，人的胃口就更好。　非看不可，四分之三以上的读者看过这篇文章后都觉得很有用。　下面介绍的这几招，只要改变日常，就可以让你不再三餐大块哚饴、毫无节制。　不要在光线不好的地方就餐　如果你出去餐馆就餐，选择餐厅中光线比较亮的地方就坐　如果你出去餐馆就餐，选择餐厅中光线比较亮的地方就坐。在这些地方，你摄入的卡路里更少，因为你潜意识

图 4.1　健康类别下 111.txt 文档内容

从图 4.1 可看出，原始数据集除了句子和段落之间有明显的标点符号分隔之外，其余内容中词与词之间基本是没有任何分隔符的。因此要对文本进行预处理，将词与词之间用空格进行分隔。

4.2 文本预处理

首先是通过 *python* 程序遍历 5 个类别下的所有新闻文档内容，对其利用 *python* 第三方库“*jieba*”进行中文分词处理，“*jieba*”分词默认采用“精确模式”处理，实质上就是根据文档中上下文内容以及语义进行分词处理。*Python* 实现代码如下图 4.2 所示：

```
def load_fenci(in_path, out_path, per_class_max_docs):
    corpus = []
    if not os.path.isdir(in_path):
        print('path error')
    with codecs.open(out_path, 'a', encoding='utf-8') as f:
        for files in os.listdir(in_path):
            cur_path = os.path.join(in_path, files)
            print(cur_path)
            if os.path.isdir(cur_path):
                count = 0
                docs = []
                for file in os.listdir(cur_path):
                    count += 1
                    if count > per_class_max_docs:
                        break
                    file_path = os.path.join(cur_path, file)

                    with codecs.open(file_path, 'r', encoding='utf-8') as fd:
                        docs.append('%s' % (files) + ' ' + ' '.join(jieba.cut(re.sub('[ _\n\r\t]+', '', fd.read()))))
                        f.write('%s' % (files) + ' ' + ' '.join(jieba.cut(re.sub('[ _\n\r\t]+', '', fd.read()))))
            corpus.append(docs)
    with codecs.open(out_path, 'a', encoding='utf-8') as f:
        for docs in corpus:
            for doc in docs:
                f.write(doc + '\n')
    return corpus
```

图 4.2 *Python* 分词代码

对经过分词操作后的新闻内容再进行删除停用词操作：通过观察实际的新闻内容，发现有很多如◆，% ，⊙，ê，à，/ 这样的无用内容，将其添加到停用词表 *stopwords*.txt 中，利用停用词表对分词后的新闻内容进行处理，下图 4.3 展示了经过分词和删除停用词两步预处理操作之后的新闻内容：

健康 夏季 炎热 出汗 体内 丢失 水分 脾胃 消化 功
差 进 稀食 夏季 饮食 养生 重 方法 夏季 炎热
出汗 体内 丢失 水分 脾胃 消化 功 差 进 稀食 夏季
饮食 养生 重 方法 早 晚 进餐 时食 粥 午餐 时 喝汤
样 生津止渴 清凉 解暑 补养 身体 煮粥 时加些
荷叶 称 荷叶 粥 味道 清香 粥 中 略 苦味 醒 脾 开胃
消解 暑热 养胃 清肠 生津止渴 作用 煮粥 时加些
绿豆 单用 绿豆 煮汤 消暑 止渴 清热解毒 生津 利尿
作用 夏季 营养 消耗 大 天气炎热 影响 人 食欲
注意 饮食 清洁 清淡 主 外 还 注意 补充 营养物质
补充 充足 维生素 吃些 西红柿 青椒 冬瓜 西瓜 杨梅
甜瓜 桃 李 新鲜 果蔬 补充 水 无机盐 特 注意
钾 补充 豆类 豆制品 香菇 水果 蔬菜 都 钾 很 好
吃些 清热 利湿 食物 西瓜 苦瓜 桃 乌梅 草莓 西

图 4.3 健康类别下 111.txt 预处理结果

通过对比图 4.1 与图 4.3 可知，经过分词和删除停用词两步预处理操作后的各类别文档内容没有了标点符号、语气助词等无用词汇，词与词之间是通过空格进行分隔。占用内存大小也随之缩小，这说明文本预处理在中文文本分类中是必不可少的一步。对预处理后的新闻内容中包含的词进行“词云”展示，如下图 4.4 所示：

图 4.4 词云

从图 4.4 可知：最明显的“中国”两字表示本文采用的数据集中绝大多少都是介绍关于国内发生的新闻事件。“游客”，“旅游”，“黄金周”等词很明显的表明它们是旅游类别下高频词汇，“火炮”，“导弹”，“台军”等词是军事类别下可能会出现的

高频词，“考生”，“志愿”，“学校”等都是关于教育类别下的高频词。综上，从绘制的词云图可发现，代表不同类别的“词汇”区分度还是比较明显的。

经过预处理后的文本内容是按照类别顺序依次排列的，即目前的样本排列中前 2900 篇文档都是关于“健康”的，第 2901-5800 篇文档内容都属于“旅游”类别，以此类推，每 2900 篇文档属于一个类别。而利用这样的数据集直接划分训练集和测试集很可能造成数据分类不均衡，因此须先对数据进行随机打乱排列，使得前 2900 篇文档可以包含各类别的新闻文档。

4.3 特征处理及分类器构建

4.3.1 基于 *bag-of-words*

词袋模型它是忽略文本内容的语法和语序，先将所有文档的分词结果构成词典，相同词不重复计入词典，并且词典中的词出现的顺序与该词在文档内容句子中出现的顺序是没有关联的，然后计算每个文档内容中的词出现在全词库的频数，构成一个词向量，词向量的长度是词典的大小，本文共构建 14500 个词向量。基于该特征处理构建朴素贝叶斯分类器，*python* 实现代码如下图 4.5 所示，实现结果如图 4.6 所示：

```
corpus = shuffle_data('../data/fencihou')
corpu
input_x, y = corpus
input_x = feature_extractor(input_x, 'bagofwords')
train_x, test_x, train_y, test_y = split_train_test([input_x, y])
print(input_x)
t0 = time()
print('\t\t使用 bag-of-words 进行特征选择的朴素贝叶斯文本分类\t\t')
fit_and_predicted(train_x, train_y, test_x, test_y)
print('time uesed: %0.4fs' %(time() - t0))
```

图 4.5 基于 *bag-of-words* 特征处理

```
                使用 bag-of-words 进行特征选择的朴素贝叶斯文本分类
             precision    recall  f1-score   support

         健康       0.97      0.96      0.97       606
         军事       0.94      0.94      0.94        48
         娱乐       0.94      1.00      0.97       566
         教育       0.93      0.96      0.95       145
         旅游       0.99      0.94      0.96       585

avg / total       0.96      0.96      0.96      1950

accuracy_score: 0.96359s
time uesed: 0.0970s
```

图 4.6 基于 *bag-of-words* 特征处理的分类器实现结果

通过选取样本数据集的 80%作为训练数据集，20%的数据作为测试数据集。全词库词向量的 *size* 是 2900×10154，表示共 100154 个词构成了无顺序的词典，共 2900 个文档进行该词向量的构建。在训练集上训练分类器，在测试集上预测对比，图 4-6 显示测试集上 5 个类别的平均精确率约为 0.96，召回率约为 0.96，*F1* 值约为 0.96，整个测试集上的准确率为 0.96359，效果还是不错的。

4. 3. 2 基于 *TF-IDF* 特征处理

```
input_x, y = corpus
input_x = feature_extractor(input_x, 'tfidf')
train_x, test_x, train_y, test_y = split_train_test([input_x, y])
t0 = time()
print('\t\t使用 TF-IDF 进行特征选择的朴素贝叶斯文本分类\t\t')
fit_and_predicted(train_x, train_y, test_x, test_y)
print('time uesed: %0.4fs' %(time() - t0))
```

图 4.7 基于 *TF-IDF* 特征处理

使用 TF-IDF 进行特征选择的朴素贝叶斯文本分类

	precision	recall	f1-score	support
健康	0.88	0.98	0.93	606
军事	0.00	0.00	0.00	48
娱乐	0.94	1.00	0.97	566
教育	0.95	0.66	0.78	145
旅游	0.95	0.94	0.94	585
avg / total	0.90	0.92	0.91	1950

accuracy_score: 0.92308s
time uesed: 0.1670s

图 4.8 基于 *TF-IDF* 特征处理实现结果

根据 *TF-IDF* 计算结果选取其中的前 20000 个特征词进行分类器的训练，在测试集上预测对比。图 4.8 显示测试集上 5 个类别的平均精确率约为 0.90，平均召回率约为 0.92，平均 *F1* 值约为 0.91，整个测试集上的准确率为 0.92308。该结果没有 *bag-of-words* 特征处理效果好，可能是因为此次只选择了其中的 2 万个特征词进行实验分析，仅仅是全词库特征词量的 1/5。基于此，在 *TF-IDF* 的基础上结合 *N-Gram* 对特征进行处理，看是否可以提高分类性能。

4.3.3 *TF-IDF* 结合 *N-Gram*

在 *TF-IDF* 特征加权基础上增加文本的 *N-Gram* 特征进行特征处理，基于 *N-Gram* 中分别抽取 *bigram*(N=2)，*trigram*(N=3)，*python* 实现结果如下图 4.9 和图 4.10、图 4.11：

使用 n_gram（unigram,bigram） 进行特征选择的朴素贝叶斯文本分类

	precision	recall	f1-score	support
健康	0.88	0.98	0.93	606
军事	1.00	0.02	0.04	48
娱乐	0.97	1.00	0.98	566
教育	0.95	0.66	0.78	145
旅游	0.95	0.95	0.95	585
avg / total	0.93	0.93	0.92	1950

accuracy_score: 0.93077s
time uesed: 0.4830s

4.9　文本特征 *N-Gram* = (1，2)

使用 n-gram（unigram、bigram和trigram） 进行特征选择的朴素贝叶斯文本分类

	precision	recall	f1-score	support
健康	0.88	0.98	0.93	606
军事	1.00	0.02	0.04	48
娱乐	0.97	0.99	0.98	566
教育	0.95	0.67	0.79	145
旅游	0.95	0.96	0.96	585
avg / total	0.94	0.93	0.92	1950

accuracy_score: 0.93231s
time uesed: 0.8260s

图 4.10 文本特征 N-Gram = (1，3）

仅仅使用 bigram 进行特征选择的朴素贝叶斯文本分类

	precision	recall	f1-score	support
健康	0.96	0.96	0.96	606
军事	1.00	0.15	0.25	48
娱乐	0.88	0.99	0.93	566
教育	0.96	0.69	0.80	145
旅游	0.93	0.95	0.94	585
avg / total	0.93	0.92	0.92	1950

accuracy_score: 0.92410s
time uesed: 0.4080s

图 4.11　文本特征 *N-Gram* =(2，2)

从上述 3 图结果对比可知：在 *TF-IDF* 基础上增加文本特征 *N-Gram* 后，分类性能指标值的确有所提升。当 *N-Gram* =（1, 3）时比 *N-Gram* =（1, 2）时效果有提高一点，但当 *N-Gram*=（2, 2）效果反而下降了。因此最终选择 *TF-IDF* 结合 *N-Gram* =（1, 3）进行特征处理。*N-Gram* =（1, 3）表示搜索既包含一个字组成的词、两个字组成的词还有三个字组成的词语。

4.4 交叉验证

在上面一系列分析过程中，均是在随机状态参数等于 10 的情况下进行训练集和测试集的取样，但是这样划分有时存在偶然结构，选择不同的状态参数会导致数据质量差异变化大，样本数据有时不能真实的反映实际的数据分布。因此为了防止这样的偶然性结构发生，取一个“求稳”的操作即进行交叉验证，这样使得构造的分类器不仅在训练

集上效果不错，而且模型的泛化能力也会不错。5 折交叉验证代码如下图 4-12 所示，运行结果如下图 4-13 所示：当 *CV*=5 时，即将数据集均分 5 份时其中 1 份作为测试集，其余 4 份作为训练集，共进行 5 次试验操作。

```
from sklearn.model_selection import cross_validate
from sklearn.metrics import recall_score
def  cross_cv(corpus, cv=5, alpha=0.1, fit_prior=True):
    input_x, y = corpus
    scoring = ['precision_macro', 'recall_macro', 'f1_macro']
    clf = MultinomialNB(alpha=alpha, fit_prior=fit_prior)
    scores = cross_validate(clf, input_x, y, scoring=scoring,
                            cv=cv, return_train_score=True)
    sorted(scores.keys())
    return scores
input_x, y = corpus
input_x = feature_extractor(input_x, 'tfidf', n_gram=(1,3))
scores = train_and_test_with_CV([input_x, y])
print(scores)
```

图 4.12 交叉验证代码

```
{'fit_time': array([0.08300447, 0.0880053 , 0.08600497, 0.08100462, 0.08300495]),
 'score_time': array([0.04100251, 0.04400253, 0.04000235, 0.04000235, 0.04000211]),
 'test_f1_macro': array([0.91362373, 0.93084115, 0.92737281, 0.92453394, 0.93615021]),
 'test_precision_macro': array([0.95023596, 0.96467747, 0.96027357, 0.95634149, 0.95616368]),
 'test_recall_macro': array([0.88795675, 0.90545338, 0.90330727, 0.90259317, 0.91937942]),
 'train_f1_macro': array([0.97241276, 0.97298279, 0.97277164, 0.97363177, 0.97299456]),
 'train_precision_macro': array([0.97935126, 0.97812214, 0.98028825, 0.98088723, 0.9798916 ]),
 'train_recall_macro': array([0.96601377, 0.96817445, 0.96586986, 0.96705175, 0.96672819])}
```

图 4.13 *CV* = 5 模型结果

从上图 4.13 中可知当进行 5 折交叉验证后，整个测试集上的平均准确率约为 0.9575，平均召回率约为 0.9037，平均 *F1* 值约为 0.9265。该结果比未进行交叉验证的平均准确率和平均 *F1* 值均有所提高，说明进行交叉验证是有明显效果的。

当 *CV* = 10 时，即将数据集均分 10 份，其中 1 份作为测试集，其余 9 份作为训练集，共进行 10 次试验操作，运行结果如下图 4.14 所示：

```
{'fit_time': array([0.09700561, 0.0870049 , 0.08700514, 0.08600473, 0.09800553,
        0.10100579, 0.08400488, 0.09100509, 0.08400464, 0.08700466]),
 'score_time': array([0.02800155, 0.02800155, 0.02900171, 0.02900195, 0.03400207,
        0.02700162, 0.02800155, 0.02800155, 0.02900171, 0.02800179]),
 'test_f1_macro': array([0.91934633, 0.9389927 , 0.95325593, 0.93016256, 0.92100308,
        0.94580736, 0.92614811, 0.93191665, 0.96642198, 0.92477503]),
 'test_precision_macro': array([0.93946089, 0.97276272, 0.96920776, 0.96574852, 0.9535614 ,
        0.96817678, 0.95141269, 0.96379718, 0.97732426, 0.94339166]),
 'test_recall_macro': array([0.90246743, 0.91552788, 0.93938484, 0.90352916, 0.89811835,
        0.92720307, 0.90794168, 0.91011494, 0.9566092 , 0.90902299]),
 'train_f1_macro': array([0.97479887, 0.97353745, 0.97524603, 0.97507034, 0.97396116,
        0.974706  , 0.97576919, 0.97357977, 0.97474811, 0.97391668]),
 'train_precision_macro': array([0.98049165, 0.97896157, 0.97966817, 0.9790719 , 0.98036429,
        0.97949679, 0.98098377, 0.97990623, 0.97979949, 0.98006995]),
 'train_recall_macro': array([0.96954793, 0.96845623, 0.97112198, 0.97133942, 0.96806695,
        0.97020434, 0.97093557, 0.96780862, 0.97004311, 0.96821951])}
```

图 4.14 *CV* = 10 模型结果

从上图 4.14 计算结果可知：当进行 10 折交叉验证后，整个测试集上的平均准确率约为 0.9605，平均召回率约为 0.9170，平均 *F1* 值约为 0.9357。通过对 5 折和 10 折交叉验证结果对比，选择结果更好的 10 折交叉验证效果更好。

4.5 最佳参数选择

朴素贝叶斯的参数比较少，通过阅读 *sklearn* 的官方文档发现它的参数主要是平滑项参数 *alpha*，通过下图 4.15 的 *python* 代码进行最佳参数的选择：

```
def best_params(corpus, cv, param_grid):
    input_x, y = corpus

    scoring = ['precision_macro', 'recall_macro', 'f1_macro']
    clf = MultinomialNB()
    grid = GridSearchCV(clf, param_grid, cv=cv, scoring='accuracy')

    scpres = grid.fit(input_x, y)

    print('parameters:')
    best_parameters = grid.best_estimator_.get_params()
    for param_name in sorted(best_parameters):
        print('\t%s: %r' %(param_name, best_parameters[param_name]))
    return scores
k_alpha = [0, 0.1,0.2,0.3,0.4,1,2,3]
fit_prior= [True, False]
param_grid = dict(alpha=k_alpha, fit_prior=fit_prior)
print(param_grid)
scores = best_params([input_x, y], cv=10, alpha=0.1)
print(scores)
```

图 4.15 最佳参数选择代码

当平滑参数值 *Alpha* = 0.1 时，朴素贝叶斯分类器运行效果最佳，整个测试集上的平均准确率约为 0.9751，平均召回率约为 0.9275，平均 *F1* 值约为 0.9401。综上：对整个搜狗语料库经过分词和去除停用词的预处理后，选择 *TF-IDF* 结合 *N-Gram*=（1,3）进行特征处理，再结合 10 折交叉验证，选取平滑参数值 *Alpha* = 0.1 时的朴素贝叶斯分类器进行中文文本分类效果最好。

4.6 逻辑回归实验分析

利用“逻辑回归”算法对数据集进行实验操作：先是对数据进行同等的预处理；然后通过不同的方法进行特征处理，进而构造逻辑回归分类器进行训练预测。在 *TF-IDF* 基础上结合文本特征 *N-Gram* 的不同取值，最终确定当 *N-Gram* =（1,2）时并且 *max_df*，*min_df* = (0.8,0.2)时效果最好，如下图 4.16 所示：

使用 max_df,min_df=(0.8,0.2) 进行特征选择的逻辑回归文本分类

	precision	recall	f1-score	support
健康	0.92	0.94	0.93	579
军事	0.90	0.56	0.69	48
娱乐	0.92	0.94	0.93	580
教育	0.88	0.83	0.86	175
旅游	0.91	0.91	0.91	568
avg / total	0.91	0.91	0.91	1950

accuracy_score: 0.91282s
time uesed: 3.2132s

图 4.16 *N-Gram* =（1,2）

从上图 4-16 结果可知：在 *TF-IDF* 基础上结合文本特征 *N-Gram*=（1,2）后，整个测试集上的平均准确率约为 0.91，平均召回率约为 0.91，平均 *F1* 值约为 0.91。此时的分类器效果没有“朴素贝叶斯分类器”的效果好。

为了提高模型在测试集上的分类性能，对数据集进行交叉验证，当 *CV* = 5 和 *CV* = 10 时分类效果分别如下图 4.17 和图 4.18：

```
scores  #5折交叉验证
```

```
{'fit_time': array([39.62026596, 39.62526655, 38.99523044, 39.95128489, 39.76527429]),
 'score_time': array([0.31301785, 0.1960113 , 0.23001313, 0.20001149, 0.19701147]),
 'test_f1_macro': array([0.91815741, 0.93083531, 0.92277633, 0.9236875 , 0.93581121]),
 'test_precision_macro': array([0.97037806, 0.95610766, 0.95872032, 0.95078858, 0.95838131]),
 'test_recall_macro': array([0.885744  , 0.91068114, 0.89720872, 0.90271275, 0.91832048]),
 'train_f1_macro': array([0.95694705, 0.95550557, 0.95736041, 0.95794164, 0.95304556]),
 'train_precision_macro': array([0.97702269, 0.97623876, 0.98020974, 0.9767495 , 0.97865527]),
 'train_recall_macro': array([0.94058073, 0.93864925, 0.93937072, 0.94236362, 0.93294255])}
```

图 4.17 *CV* = 5 模型结果

```
scores  #10折交叉验证
```

```
{'fit_time': array([40.82033491, 41.50837421, 41.52737522, 48.01474643, 58.24933171,
        51.88196754, 49.50683165, 47.38971043, 48.28176117, 49.8598516 ]),
 'score_time': array([0.21101213, 0.15100837, 0.14300823, 0.19601083, 0.14300799,
        0.14600849, 0.17200971, 0.14400816, 0.15900922, 0.14700842]),
 'test_f1_macro': array([0.93168481, 0.92677658, 0.92222975, 0.94294579, 0.92595198,
        0.93019896, 0.91356041, 0.92903265, 0.96605662, 0.91109023]),
 'test_precision_macro': array([0.97715709, 0.96786789, 0.93805828, 0.9784515 , 0.97010034,
        0.94407815, 0.95495521, 0.93725748, 0.97010856, 0.95039101]),
 'test_recall_macro': array([0.90027416, 0.89949766, 0.90915709, 0.91605364, 0.89926139,
        0.91865262, 0.8855364 , 0.92166667, 0.96235632, 0.88557471]),
 'train_f1_macro': array([0.95793285, 0.9583689 , 0.96018548, 0.9579873 , 0.95882176,
        0.95851936, 0.95996174, 0.95911033, 0.95719122, 0.95749237]),
 'train_precision_macro': array([0.97793801, 0.97478107, 0.97807731, 0.97853672, 0.97937766,
        0.97875822, 0.97774707, 0.97810311, 0.97905665, 0.9791479 ]),
 'train_recall_macro': array([0.94157162, 0.94459233, 0.94517835, 0.94141508, 0.94224153,
        0.94207582, 0.94519291, 0.94335246, 0.9395637 , 0.93990374])}
```

图 4.18 *CV* = 10 模型结果

通过进行 5 折和 10 折交叉验证对比，发现当 CV = 5 时效果最好，此时整个测试集上的平均准确率约为 0.9589，平均召回率约为 0.9029，平均 *F1* 值约为 0.9262。将两个分类在数据集上分类的效果做如下对比：

表 4.1 分类器结果对比

	Precision	Recall	F1
朴素贝叶斯	0.9751	0.9275	0.9401
逻辑回归	0.9589	0.9029	0.9262

从表 4.1 中可看出，在本文选取的数据集基础上，利用朴素贝叶斯方法进行分类器的训练和预测效果比逻辑回归方法更好一些。

4.7 本章小结

本章是运用 *python* 代码对样本数据集进行中文文本分类建模。首先通过对比词袋模型和 *TF-IDF* 模型进行特征处理的结果，选择更优的 *TF-IDF* 进行特征处理；然后在 *TF-IDF* 基础上结合文本特征 *N-Gram* 进行特征处理；再对模型利用交叉验证法构造多个训练集和测试集，取构造的多个朴素贝叶斯分类器指标值的均值作为最终模型的性能评判指标值；最后还将逻辑回归方法应用在该数据集上进行中文文本分类，以此对比两种不同分类器对于同一数据集分类性能的优劣。最终的结果对比显示朴素贝叶斯分类器在该数据集上效果更优一些。

第五章 总结与展望

5.1 总结

随着互联网快速发展以及各种相关技术的不断进步，爆炸式增长的网络信息席卷而来，在给人们实时获取来自世界各地消息的同时，也带来了很多困扰。杂乱的大量信息使得人们很难直接和高效的获取到自己需要的有价值的信息。中文文本分类技术不仅是一种自然语言处理技术也是进行文本挖掘的一项重要技术，它能针对无从下手的海量消息根据已有的分类类别进行有针对的鉴别并将其划分到真正的所属类别中。硬件技术发展的同时机器学习的各种算法也在不断的成熟。自动文本分类方法中对于各种技术方法的综合运用使得高效准确的检索对用户有价值的信息变得越来越方便。

本文先是从引入文本分类概念开始介绍，核心是介绍中文文本分类相关技术。包括文本分类方法的研究背景、研究价值到这种技术在国内国外的发展进程和以后的发展走向。选取主流的中文文本分类语料库“搜狗新闻语料库”作为实验样本；用 *Python* 对本论文中介绍的中文文本分类理论流程进行程序实现。本文采用网上经过大量研究者实践总结的“停用词表”进行预处理阶段的停用词删除操作；在特征处理阶段将 *TF-IDF* 统计量与文本特征 *N-Gram* 结合进行特征选择；为防止样本数据划分可能偏离真实的数据分布，选用交叉验证进行多个分类器的构造，取其评价指标值的平均作为最终模型的评价指标；最后对实验数据集采用朴素贝叶斯算法的和逻辑回归算法进行了对比分析，结果显示朴素贝叶斯分类器在测试集上的分类效果更优一些。

5.2 展望

文本分类的方法现阶段主要就是利用数据挖掘、机器学习、深度学习方面的理论知识和算法。现阶段直接获取到的未处理的原始信息格式五花八门，在文本分类方面的处理技术还可以通过各种算法的进一步融合得到提高。在书写论文的过程中，会不断的遇到难点或遇到一个新的方法切入点，分类的每一步都可以被拿来无限的研究。文本分类技术的每一步都是息息相关的，前一步的运行结果是下一步运行的基础。每一步都是从众多备选方法中根据实际情况选择一个或者多个进行组合应用，其中每一步所要用的理

论知识以及在真正程序实现环节各参数调整改进都可以作为一个大类单独进行深入的分析研究。

本文由于个人时间和知识储备的限制，对 *Python* 的编程能力还有所欠缺，需要进一步实践练习和提高编程能力。本文在最后分类器选择时仅将逻辑回归分类器和朴素贝叶斯分类器进行了对比，没有做其他相关算法分类器的研究。因此在未来的学习和工作中，会继续对中文文本分类的其他算法进行相关理论研究并实际动手利用 *python* 进行编程实现。

致 谢

两年的硕士生活即将告一个段落，在我以后的生活中，这两年在山东师范大学的学习与生活对我的影响是非常重要的。对于即将步入社会，开始新人生旅途的我来说，十分感恩这两年在学校的日子。

首先，非常感谢我的导师房莹副教授。两年前十分有幸能成为她的学生，感谢老师在这两年期间对我的帮助与照顾。在我攻读硕士期间，老师为我们提供了优越的自主式学习讨论环境。无论是在论文的选题，还是论文修改或完成论文的时间安排上，老师都为我进行了悉心的指导。老师在专业方面的造诣有目共睹，为我们提供的建议和指导有的放矢，避免我们走一些弯路。在生活中，老师为人谦和，也会教给我一些做人的道理，对我以后的工作和生活都会产生很大的作用。

另外要感谢我的同门以及师兄师姐们，李颖、亓小臻等人，他们为我的学术研究也提供了很多指导与帮助，他们的经验对我有很重要的指导意义。在讨论班学习以及上课的日子里，每一天都过得非常充实。我们一起讨论学术问题，一起聊实习工作问题，一起聊人生，我们共同奋斗努力实现自己的个人价值。从他们每个人的身上，我都能学到很多，感谢他们的陪伴与帮助，使我这两年的生活变得更加难忘。

感谢研究生期间所有的代课老师，为我打下坚实的专业基础知识。

感谢我的同学、舍友以及朋友，他们对我学习与生活的陪伴和理解，使我的两年研究生时光充实而美好。

特别感谢我的父母，感谢他们一直为我默默付出，他们的支持就是我最大的动力，他们对我无私的爱才使我更加的努力向上。

最后，衷心的感谢各位专家教授、老师同学在百忙之中抽出时间来评审我的论文以及参与我的毕业论文答辩，并对参与论文答辩评审的各位老师表示衷心的感谢。

参 考 文 献

[1] 董露露.基于特征选择及 LDA 模型的中文文本分类研究与实现[D].合肥：安徽大学,2014.

[2] 王海鹃,韩立新,甄志龙.基于索引项权重的文本特征选择方法[J].计算机工程与设计,2010,31(5):1149-1151.

[3] 杨孟英.基于支持向量机的中文文本分类研究[D].北京:华北电力大学,2017.

[4] 贾隆嘉.文本分类中特征加权算法和文本表示策略研究[D].长春:东北师范大学,2016.

[5] 黄娟娟.基于 KNN 的文本分类特征选择与分类算法的研究与改进[D].厦门:厦门大学,2014.

[6] 石俊涛.中文文本分类中卡方特征提取和对 TF_IDF 权重改进[D].成都:西华大学,2017.

[7] 李航.统计学习方法[M].北京:清华大学出版社,2012.

[8] 侯汉清.分类法的发展趋势简论[J].情报科学,1981,13(1):58-63.

[9] 肖明,沈英.自动分类研究进展[J].现代图书情报技术,2000,16(5):25-28.

[10]万斌候.文本分类中的特征降维方法研究[D].重庆:重庆大学,2012.

[11]肖婷.基于 χ^2 统计的中文文本分类特征选择方法研究[D].西南大学,2009.

[12]马治涛.文本分类停用词处理和特征选择技术研究[D].西安:西安电子科技大学,2014.

[13]姚海英.中文文本分类中卡方统计特征选择方法和 TFIDF 权重计算方法的研究[D].吉林:吉林大学,2016.

[14]李荣陆.文本分类及相关技术研究[D].上海:复旦大学,2005.

[15]裴英博,刘晓霞.文本分类中改进型 CHI 特征选择方法的研究[J].计算机工程与应用,2011,4(4)：128-130.

[16]李原.中文文本分类中分词和特征选择方法研究[D].吉林:吉林大学,2011.

[17]刘依璐.基于机器学习的中文文本分类方法研究[D].西安：西安电子科技大学,2015.

[18]宋惟然.中文文本分类中的特征选择和权重计算方法研究[D].北京:北京工业大学,2013.

[19]代六玲,黄河燕,陈肇雄.中文文本分类中特征抽取方法的比较研究[J].中文信息报,2004,18(1)26-32.

[20]尚文倩.文本分类及其相关技术研究[D].北京：北京交通大学,2007.

[21]姜远,周志华.基于词频分类器集成的文本分类方法[J].计算机研究与发展,2006,43(10):1681-1687.

[22]陆旭.文本挖掘中若干关键问题研究[M].合肥：中国科学技术大学出版社,2008.

[23]王博.文本分类中特征选择技术的研究[D].长沙：国防科学技术大学，2009.

[24]刘赫.文本分类中若干问题研究[D].长春：吉林大学，2009.

[25]杜芳华.基于半监督学习的文本分类算法研究[D].北京：北京工业大学,2014.

[26]辛竹.文本分类中的特征提取算法研究与改[D].北京：北京邮电大学,2014.

[27]杨海.SVM 核参数优化研究与应用[D].浙江大学, 2014.

[28]马雯雯，邓一贵.新的短文本特征权重计算方法[J].计算机应用.2013，33(8):2280-2282.

[29]张岩.基于 SVM 算法的文本分类器的实现[D].电子科技大学，2011.

[30]詹增荣，曾青松.基于径向基函数插值与 SVM 的协同过滤算法[J].计算机与现代化,2015,(08):98-103.

[31]Quan X, Wenyin L, Qiu B. Term Weighting Schemes for Question Categorization[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2011, 33(5):1009-1021.

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

[33]Dilrukshi I, Zoysa K D. Twitter news classification: Theoretical and practical comparison of SVM against Naive Bayes algorithms [C]. International Conference on Advances in ICT for Emerging Regions, Colombo Sri Lanka, 2014:278-278.

[34]Luhn H P. An experiment in auto-abstracting: Auto-abstracts of Area 5 Conference Papers [C]. International Conference on Scientific Information, Washington DC, 1958:16-21.

[35]De Campos LM, Romero A E.Bayesian network models for hierarchical text classification from a thesaurus[J]. International Journal of Approximate Reasoning, 2009, 50(7): 932-944.

[36]Saeys Y, Abeel T, van de Peer Y. Robust Feature Selection Using Ensemble Feature Selection Techniques [C]. Proc of European Conference on Machine Learning and Knowledge Discovery in Databases, ECML/PKDD 2008, Antwerp Belgium, 2008:313-325.

[37]Hansen L K, Salamon P. Neural network ensembles [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1990, 12(10): 993-1001.

[38]刘玲玲.文本分类中的特征选择研究[D].中国石油大学,2011

[39]于津凯，王映雪，陈怀楚.一种基于 N_Gram 改进的文本特征提取算法[J].图书情报工作，2004，48（8）：48-49

[40]毛伟，徐蔚然，郭军.基于 n_gram 语言模型和链状朴素贝叶斯分类器的中文文本分类系统[J].中文信息学报,2005,20（3）：30-31

附 录

本文 *python* 编程主要代码如下：

```
# coding : utf-8
import os
import codecs
import jieba
import re
```

1. 分词处理

```
category = ['健康', '教育', '军事', '旅游', '娱乐']
def load_fenci(in_path, out_path, per_class_max_docs):
    corpus = []
    if not os.path.isdir(in_path):
        print('path error')
    with codecs.open(out_path, 'a', encoding='utf-8') as f:
        for files in os.listdir(in_path):
            cur_path = os.path.join(in_path, files)
            print(cur_path)
            if os.path.isdir(cur_path):
                count = 0
                docs = []
                for file in os.listdir(cur_path):
                    count += 1
                    if count > per_class_max_docs:
                        break
                    file_path = os.path.join(cur_path, file)

                    with codecs.open(file_path, 'r', encoding='utf-8') as fd:
                        docs.append('%s' % (files) + ' ' + ' '.join(jieba.cut(re.sub('[    \n\r\t]+', '',
  fd.read()))))
                        f.write('%s' % (files) + ' ' + ' '.join(jieba.cut(re.sub('[    \n\r\t]+', '',
   fd.read()))))
            corpus.append(docs)
    with codecs.open(out_path, 'a', encoding='utf-8') as f:
        for docs in corpus:
            for doc in docs:
                f.write(doc + '\n')
    return corpus
if __name__ == "__main__":
```

```
    in_path = "../毕业论文/data_before"
    out_path = "../毕业论文/fencihou.txt"
    corpus = load_fenci(in_path, out_path, 2000)
print('corpus size(%d,%d)' %(len(corpus), len(corpus[0])))

# print(corpus[0][1])
```

2. 过滤停用词

```
stopwords = []
delst_alltxt = []

st = codecs.open('E:/pythontest/lunwendata/stopwords-hgd.txt', 'rb',encoding='utf-8')
delst_result = codecs.open('E:/pythontest/lunwendata/delst_result.txt', 'a', encoding='utf-8')

for line in st:
    line = line.strip()
    stopwords.append(line)
print('*******开始删除停用词*******')
for docs in corpus:
    delst_singletxt = []
    for word in docs:
        word = word.strip()
        if word not in stopwords:
            if word >= u'\u4e00' and word <= u'\u9fa5':
                delst_singletxt.append(word)
    delst_alltxt.append(delst_singletxt)

for delst_singletxt in delst_alltxt:
    for everyword in delst_singletxt:
        delst_result.write(everyword + '\t')
        delst_result.write('\n')
delst_result.close()
print('停用词删除结束！删除停用词后的分词数据是 after_delst_result.txt' + '\n')
```

3. TF-IDF

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cross_validation import train_test_split
from sklearn.metrics.scorer import make_scorer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
```

```
from time import time
def feature_extractor(input_x, case='tfidf', n_gram=(1,1)):
    if n_gram == (1,1):
        if case.lower() == 'tfidf':
            return TfidfVectorizer().fit_transform(input_x)
        elif case.lower() == 'bagofwords':
            return CountVectorizer().fit_transform(input_x)
    else:
        if case.lower() == 'tfidf':
            return TfidfVectorizer(ngram_range=n_gram).fit_transform(input_x)
        elif case.lower() == 'bagofwords':
            return CountVectorizer(ngram_range=n_gram).fit_transform(input_x)
```

4. 切分数据

```
def split_train_ test(corpus, indices=0.2, random_state=10, shuffle=True):
    input_x, y = corpus
    x_train, x_test, y_train, y_test = train_test_split(input_x, y, test_size=indices, random_state=10)
    print("Vocabulary Size: {:d}".format(input_x.shape[1]))
    print("Train/Test split: {:d}/{:d}".format(len(y_train), len(y_test)))
    return x_train, x_test, y_train, y_test
```

5. 训练和预测数据

```
def fit_to_predicted(train_x, train_y, test_x, test_y):
    clf = MultinomialNB().fit(train_x, train_y)
    predicted = clf.predict(test_x)
    print(metrics.classification_report(test_y, predicted))
    print('accuracy_score: %0.5fs' % (metrics.accuracy_score(test_y, predicted)))
```

6. 交叉验证

```
from sklearn.model_selection import cross_validate
from sklearn.metrics import recall_score
def train_and_test_with_CV(corpus, cv=5, alpha=1, fit_prior=True):
    input_x, y = corpus
    scoring = ['precision_macro', 'recall_macro', 'f1_macro']
    clf = MultinomialNB(alpha=alpha, fit_prior=fit_prior)
    scores = cross_validate(clf, input_x, y, scoring=scoring,cv=cv, return_train_score=True)
    sorted(scores.keys())
    return scores
```

7. 寻找构造器的最优参数

```
from sklearn.grid_search import GridSearchCV
def train_and_predicted_with_graid(corpus, cv, param_grid):
    input_x, y = corpus
```

```
scoring = ['precision_macro', 'recall_macro', 'f1_macro']
clf = MultinomialNB()
grid = GridSearchCV(clf, param_grid, cv=cv, scoring='accuracy')
scpres = grid.fit(input_x, y)
print('parameters:')
best_parameters = grid.best_estimator_.get_params()
for param_name in sorted(best_parameters):
    print('\t%s: %r' %(param_name, best_parameters[param_name]))
return scores
```