文本自动分类系统

学号：201618013727061
姓名：毛廷运
研究所：电子学研究所
手机号码：15611566381
电子邮箱：1753206676@qq.com

摘要

文本自动分类是自然语言处理领域的重要应用，可以在较大程度上解决信息杂乱现象的问题,方便用户准确地定位所需的信息。本项目以“搜狗实验室”的新闻数据作为语料库，其中包括了军事、财经、IT、汽车等 10 类新闻文档。在完成对文档的字符格式转换之后，用结巴分词工具对语料库进行批量分词，并划分训练语料和测试语料，接着对训练语料使用了卡方统计(CHI)和信息增益两种方法分别选择文档特征以用于实验对比，再计算特征项对应的 TF-IDF 作为其权值，最后分别训练了朴素贝叶斯、SVM 和随机森林三种分类器，用测试语料进行测试，对比发现随机森林能取得更好的分类效果。

关键词：文本自动分类；卡方统计；信息增益；TF-IDF；朴素贝叶斯；SVM；随机森林

一、绪论

1、国内外研究现状

上世纪 50 年代，H.P.Luhn 的一篇文章[1]在文本分类领域取得了巨大的成功，该文章在文本分类研究中引入了词频统计的思想。60 年代在 Journal of ACM 上，由 Maron 发表的自动文本分类的论文[2]对后续的关于搜索引擎和文本分类的研究产生了深刻的影响。Salton 等人于 1973 年首次提出了 VSM[3]，利用向量来表示文本的特征项，是一种经典的文本表示模型。一直到 80 年代末 90 年代初，文本分类依旧属于知识工程。在进入了 20 世纪末期，由于互联网的发展，数据以指数形式增加，于是以机器学习为基础自动分类成为了主流。随着研究的逐步深入，许多种机器学习领域的算法都被应用于文本分类领域。随着数据量的进一步扩大，文本的特征空间维度也随之增加，以前的一些分类算法在大型的数据集上效果比较差。因此对于这个问题的解决方案主要有两种思路，一种是改进现有的分类器以适应高维的庞大的数据集，另一种是对文本进行降维。

第一种思路，改进分类器。卜凡军利用投影寻踪理论和 iDistance 索引结构改进了 k 近邻(KNN)算法，利用一维投影距离得到一个较小的训练集的子集，通过计算样本和子集中数据的相似度就能得到相似度最大的 K 个样本，不需要对整个训练集计算相似度，所以在提高速度的同时也提高了精度[4]。李东晖等人通过壳向量降低支持向量机中二次规划的复杂度，提高了分类的准确度同时减少了训练的时间[5]。邸鹏及段利国在朴素贝叶斯算法中提高后验概率的权值，达到了增加分类精度的效果，同时分类时不对先验概率进行计算提高了计算速度，并利用实验验证了该方法的有效性[6]。

第二种思路，文本降维。对特征降维可以从特征选择和特征提取两方面进行研究。在特征选择方面，徐振强等人提出了一种改进的 TFIDF 算法，该算法利用特征项在文档中出现的位置及分布的离散度对 TFIDF 算法进行改进，并且在三个不同语料上进行了对比实验，结果表明该方法是可行和有效的[7]。王行恒等人依据词语在类内和类间的分布利用循环迭代算法优化特征词的选取，并在使用支持向量机(SVM)在大量数据集上进行分类测试取得了良好的效果[8]。特征提取方面主要是利用主题模型来提取文档特征：谢静先对文本利用词频和文档频率进行过滤，然后分别对每个部分利用潜在狄利克雷分布(LDA)建模，最终用 SVM 分类器进行分类，实验验证该方法比文本向量模型的方法效果好[9]。张志飞等对短文本分类中的上下文依赖问题进行了分析，提出了一种利用 LDA 模型生成文档．主题矩阵解决该问题的方法，并且结合 KNN 对新闻标题进行分类，取得了比较好的效梨[10]。

2、项目目标

“自然语言处理”课上讲的都是理论知识，如果没有实验经验就无法真正的掌握自然语言处理的相关技术。

本项目通过实现一个文本的自动分类系统，加深对理论知识的认识，提高个人编程能力，学会对语料库进行预处理，掌握文本特征选择的相关方法，熟练掌握常用分类器的编码实现、训练和调参，提高对实验现象的分析能力。

二、语料库的预处理

我们获得的原始语料是一系列纯文本格式的文档，在进行特征选择之前，需要进行一些预处理，包括文档字符编码格式的转换、文档分词和去停用词。

1、语料获取

本项目以“搜狗实验室”的新闻数据作为语料库，一共有 10 个新闻类别，分别是汽车、财经、IT、健康、体育、旅游、教育、招聘、文化和军事，每个类别都包含 300 个文档。对于语料库的存储，每个类别对应一个文件夹，文件夹的名称是类别对应的编码，每个文件夹下存放相应类别的文档，具体地如图 2.1 所示。

图 2.1 语料库的存储目录

将原始语料库的构成以表格形式呈现，如表 2.1 所示。

表 2.1 本项目使用的原始语料库构成

类别	类别编号	包含的文档数量
汽车	C000007	300
财经	C000008	300
IT	C000010	300
健康	C000013	300
体育	C000014	300
旅游	C000016	300
教育	C000020	300
招聘	C000022	300
文化	C000023	300
军事	C000024	300

每个类别下的文档数量都是一样的，接下来每个类别都以相同的比例划分训练语料和测试语料，这样就不会出现数据偏斜。

2、语料库字符编码的格式转换

本项目的所有实验将在 ubuntu 操作系统下完成，而 ubuntu 操作系统下，字符编码格式是 UTF-8，如果格式不正确就无法进行分词等操作。本项目下载的语料库是 GBK 格式的，必

须转换成 UTF-8 格式，可以使用 enca 转码工具对语料库进行批量的格式转换。具体地操作是，在每个类别的目录下打开终端，输入命令： enca -x utf-8 *。

3、对语料库进行批量分词

3.1 结巴分词工具的介绍和测试

中文文本不像英文那样每个单词之间有分隔符，中文文本在进行应用时需要分词处理，分词可以使用网上开源的分词工具，比如结巴、ICTCLAS、 Stanford 汉语分词工具等，本项目使用 python 语言编写程序，所以选择用 python 编写的结巴分词工具，具有支持用户加入自定义的词典，扩展性好，简单易用等特点。

结巴分词默认使用隐马尔科夫模型(HMM)，现测试其分词效果，所用的句子来源于语料库中的新闻语句。

（1）测试 1：

原句：随着 7 月 1 日大限的临近，强制三者险的相关准备工作也在保险业内紧张地进行着。不过，对于普通车主来说，强制三者险应该是福音与“噩耗”并存的：权益有保障的同时钱包受考量！

分词：随着/7/月/1/日/大限/的/临近/，/强制/三者/险/的/相关/准备/工作/也/在/保险业/内/紧张/地/进行/着/。/不过/，/对于/普通/车主/来说/，/强制/三者/险/应该/是/福音/与/“/噩耗/”/并存/的/：/权益/有/保障/的/同时/钱包/受/考量/！

（2）测试 2：

原句：中国四大国有银行之一的中国银行(Bank of China)准备进行 6 年来全球最大的上市行动，试图筹资至多 99 亿美元。

分词：中国/四大/国有银行/之一/的/中国银行/(/Bank/ /of/ /China/)/准备/进行/6/年来/全球/最大/的/上市/行动/，/试图/筹资/至多/99/亿美元/。

（3）测试 3：

原句：去年 11 月，迪志公司发现易趣网未经其许可，允许并配合其用户在网上公开拍卖该电子出版物，且这 17 张光盘均属盗版。

分词：去年/11/月/，/迪志/公司/发现/易趣网/未经/其/许可/，/允许/并/配合/其/用户/在/网上/公开/拍卖/该/电子/出版物/，/且/这/17/张/光盘/均/属/盗版/。

从整体上看，结巴分词对于语句中的的常用词具有较好的识别能力，但是也存在歧义划分问题，比如将“三者险”，划分成“三者/险”，歧义消解是所有分词器面临的问题，可以通过加入用户自定义词典的方式来提高分词器的歧义消解能力，本项目的重点在于掌握文

本自动分类系统设计的整个过程，对于分词器的选择，只要能满足基本要求即可，所以本项目直接使用结巴分词工具的默认词典，没有加入自定义词典。

3.2 使用结巴分词工具对语料库进行批量分词

语料库包含 10 个新闻类别，每个类别下包含 300 个文档，每个文档都是一篇相应类别的新闻，需要对整个语料库进行批量分词。在实际分词的过程中发现有 9 个类别下的若干文档字符编码格式无法识别而导致程序执行中断，处理的方法是，通过使用 try/finally 语句来过滤掉那些无法识别的文档。分词批处理代码贴图如图 2.2 所示。

```
print(".............Segmentation begins.....................")
for eachFolder in FolderList:
    file_count=0
    for i in range(FolderTextCount):
        # file_count
        readfilename = ReadFilePathPrefix+eachFolder+"/"+str(i)+".txt"  # 输入文档的路径
        writefilename = WriteFilePathPrefix+eachFolder+"/"+str(file_count)+".seg" #分词处理后的文档路径
        try:
            testFile = io.open(readfilename, mode='r')
            testFileContent = testFile.readlines()
            resultfile = io.open(writefilename, mode='w')  # 避免同名文件的存在
            resultfile.close()
            resultfile = io.open(writefilename, mode='a')
            for eachLine in testFileContent:
                eachLine = eachLine.strip('\n')  # 去掉换行符"\n"
                if len(eachLine) > 0:
                    #  cutResult = jieba.cut("中国的首都是北京，我在中国科学院大学。", cut_all=False)
                    cutResult = jieba.cut(eachLine, cut_all=False)  # 结巴分词选择"精确模式"
                    for eachword in cutResult:
                        resultfile.write(eachword + " ")
                    resultfile.write(u"\n")
                    print(cutResult)
        except Exception as err:
            print(err)
            file_count -= 1
        finally: # finally相当于continue，跳过本次循环
            file_count +=1
            resultfile.close()
print(".....................OK.....................")
```

图 2.2 分词批处理代码贴图

经过分词批处理后得到分词后的语料库，分词效果如图 2.3 所示。

消息 传来 ： 强制 缴纳 1800 元 / 年 的 保费 ， 最高 却 只 赔 5.2 万 ！ 随着 7 月 1 日 大限 的 临近 ， 强制 三者 险 的 相关 准备 工作 也 在 保险业 内 紧张 地 进行 着 。 不过 ， 对于 普通 车主 来说 ， 强制 三者 险 应该 是 福音 与 “ 噩耗 ” 并存 的 ： 权益 有 保障 的 同时 钱包 受 考量 ！ – – 虽然 有 “ 先行 垫付 ” 、 “ 赔偿金 10 日须 到位 ” 和 “ 七种 行为 可能 吊销 执照 ” 等 条款 保护 投保人 权益 ， 但 强制保险 的 最高 限额 仅为 5.2 万 ： “ 驾驶员 有 过错 死亡 伤残 赔偿 限额 4 万 、 医疗 费用 赔偿 限额 1 万 、 财产 损失赔偿 限额 2000 元 。 ” 与此同时 ， 传出 的 消息 是 车主 为此 必须 付出 1800 元 的 “ 天价 ” 强制 保险费 ！ 1800 元 的 天价 保费 是不是 在 抢银行 ？ 这个 消息 是 中国消费者协会 副 秘书长 武高汉 日前 透露 的 。 但 保监会 相关 负责人 当即 表示 对此 不 做 置评 并 在 几天 后 予以 “ 辟谣 ” ， 更进一步 宣称 要待 听证 后 决定 ； 而 各 财险 公司 负责人 也 纷纷 以 不知情 为 由 婉拒 了 各位 记者 的 进一步 采访 。 事 不止 此 ， 请 注意 一个 细节 是 某 保险公司 人士 不 留神 说道 ， “ 比 之前 2600 元 的 传闻 价 已经 低 了 。 ” – – 也就是说 ， 不管 是不是 1800 元 ， 笔者 猜测 总在 1500 元 以上 ， 何况 主持 听证会 的 有 不 “ 黑 ” 的 吗 ？ 最终 还 不是 在 抢银行 ！ 不是 吗 ？ 据 国家 安全 生产 管理局 的 数字 显示 ， 2005 年 全国 共 发生 道路 交通事故 约 45 万起 ， 死亡 近 10 万人 ， 受伤 超过 46 万人 ， 直接 财产损失 18.8 亿 。 实行 强制保险 后 ， 绝大多数 机动车 必须 投保 ， 2006 年 机动车 保有量 约 达 1.5 亿辆 ， 每车 1800 元 的 强制 保费 其 总额 就 达 2700 亿 ！ 这里 问题 就 来 了 ， 如果 2006 年 全国 交通事故 死伤 人数 与 去年 持平 ， 即便 按照 “ 驾驶员 有 过错 死亡 伤残 赔偿 限额 4 万 、 医疗 费用 赔偿 限额 1 万 、 财产 损失赔偿 限额 2000 元 ” 的 最高 标准 ， 强制保险 基金 也 只 需 赔付 死亡 伤残 赔偿金 40 亿 、 医疗 费用 46 亿 以及 财产损失 11 亿 共计 97 亿 。 – – 赔付 仅仅 约 为 收取 保费 2700 亿 的 1 / 30 多一点 ！ 以 1 元搏 30 元 赚 29 元 ， 利润 大大的 ！ 剩下 的 两千多个 亿 哪 去 了 ？ 修 三峡大坝 吗 ？ 这 还 不是 抢银行 ？ ！ 强制 三者 险 属于 非盈利 险种 ， 如果 根据 有关 方面 “ 强制保险 是 不 赢 不 赚 的 ” 的 说法 ， 按照 上面 的 估算 所 应该 收取 的 保费 只 应该 是 1800 / 30 = 60 元 ！ 而 这时 ， 强制保险 已经 是 做 了 最大 赔付 的 ！ 否则 哪怕 只要 多收 一分 强制保险 都 应该 是 有 赚 的 。 当然 ， 如果 只多缴 区区 60 元 ， 我们 广大 车主 也 不会 计较 太 多 ， 都 会 踊跃 投保 的 ； 可是 ， 有关 方面 会 有 这么 “ 好心 ” 吗 ？ 真 不 赚 吗 ？ “ 我 买 的 商业 三者 险 保额 是 10 万 ， 价钱 才 791 元 。 ” “ 原来 上 5 项 险种 才 是 3000 元 ， 而 现在 1 项 险种 就 抵 原来 的 一半 。 ” 目前 商业 三者 险 投保 限额 最低 为 5 万 ， 根据 保险公司 的 不同 ， 缴纳 的 保费 为 700 ～ 1200 元 （ 无 优惠 时 ） ， 而且 这个 数 可以 最多 打 7 折 ！ 如果 优惠 下来 ， 最低 还 不到 500 元 ！ 而 就是 仅收 500 元 的 商业 三者 险 ， 就 已 让 整体 亏

图 2.3 分词后的文档

由于过滤掉了一些无法识别的文档，经分词处理后的语料库中，各类别下的文档数目有所变化，具体地，以表格形式统计，如表 2.2 所示。

表 2.2 分词后的语料库构成

类别	类别编号	包含的文档数量
汽车	C000007	293
财经	C000008	295
IT	C000010	296
健康	C000013	296
体育	C000014	297
旅游	C000016	298
教育	C000020	297
招聘	C000022	300
文化	C000023	297
军事	C000024	297

4、去除语料库中的停用词

停用词指的是那些对区分文档类别没有作用的词，这些词通常在每个文档中大量出现，比如标点符、语气助人称代词等。在进行特征选择之前需要去除停用词，一方面可以降低计算量，更重的是可以选出更好的特征词。

4.1 定义停用词表

本项目定义的停用词有：标点符号、特殊符号、数字、语料库中每个文档频繁出现的词、语气助词、副词、介词、连接词、人称代词等，一共定义了 1063 个。停用词表部分截图如图 2.4 所示。

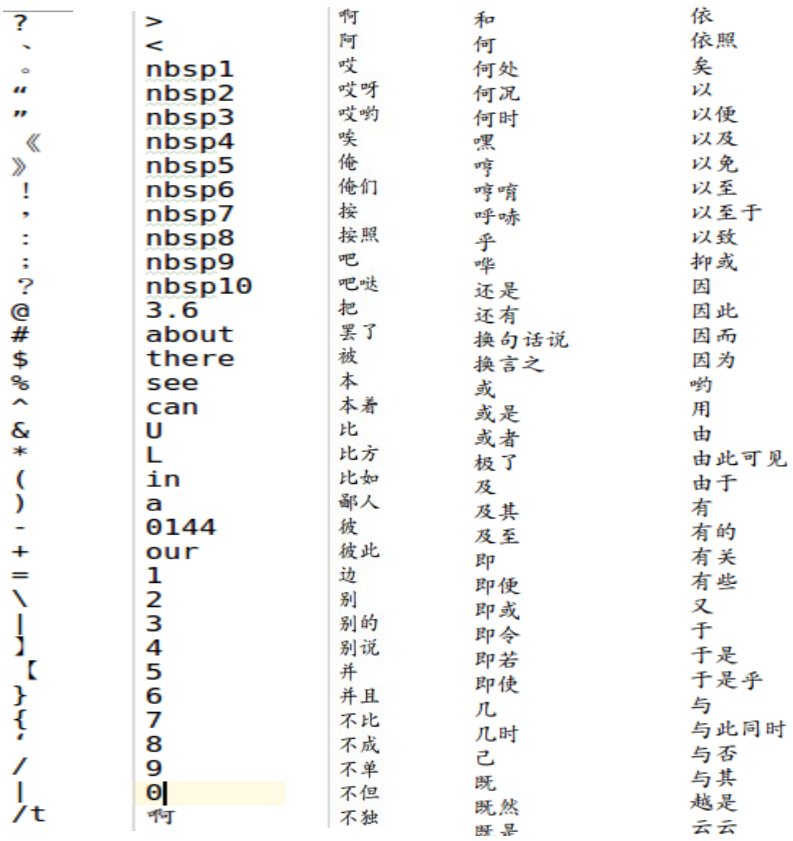

图 2.4 停用词表的部分截图

停用词影响后面文本特征词的选择，从而影响分类器的性能。得到文本的特征词后，可以查看特征集中是否含有一些无价值的特殊符号，如果有则将其定义成停用词，然后重新进行特征选择，这样可以得到更好的特征，从而提高分类准确率。

4.2 去除停用词

定义的停用词存放在“stopword.txt”文件中，下一步要去除已经分词的语料库中的停用词，具体的实现过程：如果一个词是停用词则丢弃，否则存放到词典中构成新的数据集。判断是否为停用词的代码如图 2.5 所示。

```
# 判断是否为停用词
def stop_word(s):
    with open("stopword.txt", 'r') as f:
        all_stopword = f.readlines()
        # print(all_stopword)
        for word in all_stopword:
            word = word.strip("\n")
            if s == word:
                return True
        return False
```

图 2.5 停用词判断代码贴图

三、划分训练语料和测试语料

语料库已经过分词、去停用词处理，现在划分出训练语料与测试语料以用接下来的特征选择、分类器训练的测试。语料库在分词处理时已经过滤了若干文档，为避免训练数据倾斜，每个类别下的前 250 个文档作为训练数据文档，剩下的作为测试文档。训练语料和测试语料的构成以表格形式统计，如表 3.1 所示。

表 3.1 训练语料和测试语料构成

类别	类别编码	训练文档数量	测试文档数量
汽车	C000007	250	43
财经	C000008	250	45
IT	C000010	250	46
健康	C000013	250	46
体育	C000014	250	47
旅游	C000016	250	48
教育	C000020	250	47
招聘	C000022	250	50
文化	C000023	250	47
军事	C000024	250	47

每个类别下的测试文档数量相差很小，对训练好的分类器进行测试，得到的平均准确率能较好评价分类器的对各类别的预测能力。

四、特征选择

特征指某一物质自身所具备的特殊性质，是区别于其他物质的基本标志。对于文本而言，特征就是能表征一类文本，同时又能区别于其他类别文本的一个词的集合，特征选择的目的是降低特征空间维数,通过选取类别区分度较大的词作为特征项,过滤作用小的词,使分类的准确率得到提升。

对于一类文档，可以用这个类别下所有文档的词构成一个集合，这个集合可以看成是这类文档的属性，如果以这个集合作为特征，不仅维度很高，而且类别区分度较差，所以必须选择出其中最具有区分能力的词来构成特征项，特征选择的方法很多，比如文档频率、TF-IDF、互信息、信息增益、卡方统计(CHI)等。本项目分别使用了卡方统计和信息增益这两种方法，各自对训练语料进行特征选择，得到两种特征，这么做主要是想对这两种方法进行对比。

1、基于卡方统计的特征选择

1.1 卡方统计

卡方统计量(CHI)用于检验两个事件之间的独立性, 它度量了期望计数和观察计数之间相互之间关系。

卡方统计公式如下：

$$CHI(t,c)=\frac{N(AD-BC)*(AD-BC)}{(A+C)(A+B)(C+D)(B+D)}$$

公式中变量的含义如下：

t:表示某个特征项，即一个词

c:表示某个类别

N：训练语料中所有文档的数量

A：在类别 c 下，包含词 t 的文档数量

B：不在类别 c 下，包含词 t 的文档数量

C：在类别 c 下，不含词 t 的文档数量

D：不在类别 c 下，不含词 t 的文档数量

N=A+B+C+D

在具体实现中，先对每个类别下的词进行 CHI 计算，然后按 CHI 值从大到小排序，取前 K 个词作为各类别的特征项，K 是一个超参数，是由自己确定的。CHI 的计算公式中(A+C)、(B+D)和 N 对排序没有影响，可以对计算公式进行简化以减少计算量，简化的公式如下：

$$CHI(t,c)=\frac{(AD-BC)*(AD-BC)}{(A+B)(C+D)}$$

1.2 使用卡方统计选择特征

训练语料包含 10 个类别，每个类别下有 250 个文档，一共 2500 个文档，特征选择过程如下：

（1）分别对每个类别下所有文档的词进行汇总，存放到类别对应的集合中，集合中的词都是非重复的，一个得到 10 个集合。

（2）计算每个集合中词的 CHI，然后按 CHI 值从大到小排列。

（3）每个集合中取前 K 个词作为所在类别的特征项。

（4）对每个类别的特征项进行汇总，存放到一个集合中，这个集合就是我们要用于分类的特征集。

值得一提的是 Python 中提供了集合类 set，将文档中的词存放到 set 类中，得到一个词集，这个词集中没有重复的词，所以在编程实现时，使用 set 类可以起到去重的作用。

在熟悉以上过程后，编写基于卡方统计的特征选择 Python 脚本，程序流程图如图 4.1 所示。

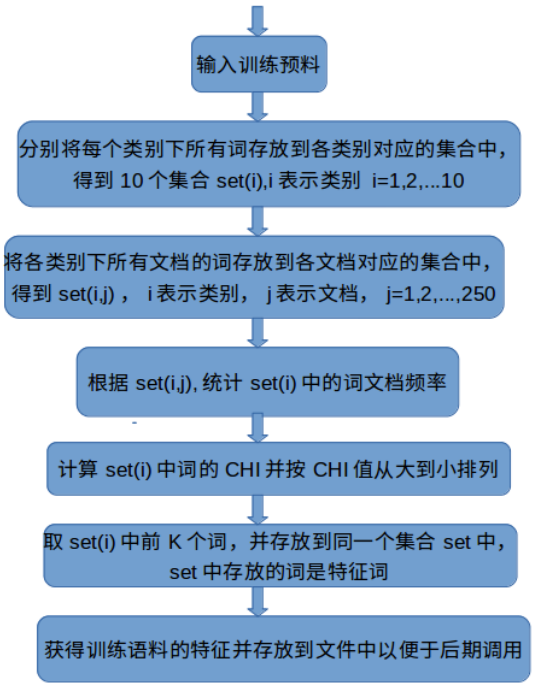

图 4.1 基于 CHI 的特征选择流程图

算法实现的部分代码贴图如图 4.2 所示。

```
termCountDic = dict()
for key in termDic:
    classWordSets = termDic[key]
    classTermCountDic = dict()
    for eachword in classWordSets:  # 对某个类别下的每一个单词的 a b c d 进行计算
        a = 0
        b = 0
        c = 0
        d = 0
        for eachclass in termClassDic:
            if eachclass == key:  # 在这个类别下进行处理
                for eachdocset in termClassDic[eachclass]:
                    if eachword in eachdocset:
                        a = a + 1
                    else:
                        c = c + 1
            else:  # 不在这个类别下进行处理
                for eachdocset in termClassDic[eachclass]:
                    if eachword in eachdocset:
                        b = b + 1
                    else:
                        d = d + 1
        # print("a+c:"+str(a+c)+"b+d"+str(b+d))
        eachwordcount = ChiCalc(a, b, c, d)
        # print(eachwordcount)
        classTermCountDic[eachword] = eachwordcount
    # 对生成的计数进行排序选择前K个
    # 这个排序后返回的是元组的列表
    sortedClassTermCountDic = sorted(classTermCountDic.items(), key=lambda d: d[1], reverse=True)
    # count = 0
    subDic = dict()
    for i in range(K):
        subDic[sortedClassTermCountDic[i][0]] = sortedClassTermCountDic[i][1]
    termCountDic[key] = subDic
return termCountDic  # termCountDic={类别1：{w1: chi, w2: chi2, ....., wk:chik}, ....., 类别10：{
```

图 4.2 基于 CHI 的特征选择核心代码部分贴图

在实验中 K=500，K 的取值是自己定的，K 的取值影响最后分类器的分类准确率。选取的特征存放到文件中，一共有 4840 个词。特征词的部分截图如图 4.3 所示。

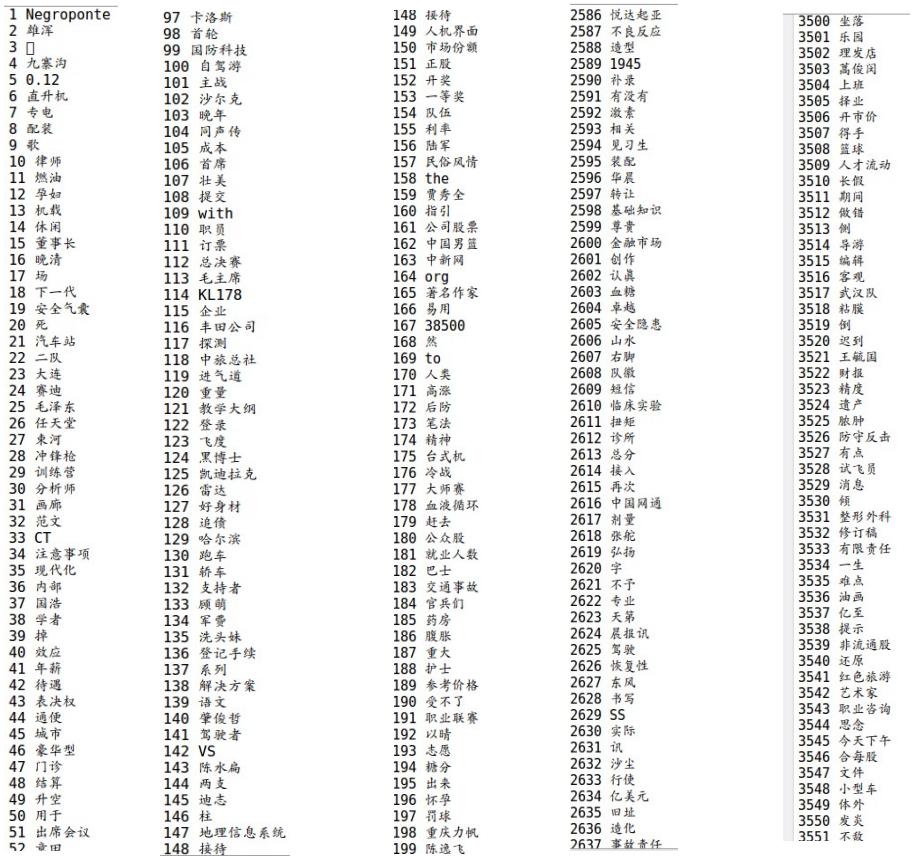

图 4.3 基于卡方统计选取的特征部分截图

2、基于信息增益的特征选择

2.1 信息增益

信息增益(IG)衡量特征能够为分类系统带来多少信息，其计算公式如下所示：

$$IG(ti)=-\sum_{j=1}^{c}P(Cj)*logP(Cj)+P(ti)*(\sum_{j=1}^{c}P(Cj/ti)*logP(Cj/ti))+P(ti')*(\sum_{j=1}^{c}P(Cj/ti')*logP(Cj/ti'))$$

公式中各变量的含义：

ti:第 i 个特征项，即一个词

ti’：非 ti，即特征项中除了 ti 以外的词

Cj: 类别，j=1,2,…,c, c 是类别数量

公式中各概率的计算公式：

$$P(Cj)=\frac{Aij+Cij}{N} \qquad P(ti)=\frac{Aij+Bij}{N}$$

$$P(ti')=\frac{Cij+Dij}{N}$$

$$P(Cj/ti)=\frac{Aij+1}{Aij+Bij+c} \qquad P(Cj/ti')=\frac{Cij+1}{Cij+Dij+c}$$

以上公式中：

Aij：j 类中，含特征项 ti 的文档数量

Bij：在非 j 类中，含特征项 ti 的文档数量

Cij：j 类中，不含特征项 ti 的文档数量

Dij：在非 j 类中，不含特征项 ti 的文档数量

2.2 使用信息增益选择特征

训练语料包含 10 个类别，每个类别下有 250 个文档，一共 2500 个文档，特征选择过程如下：

（1）对训练语料中所有文档的词进行汇总，存放到同一个集合中，集合中的词都是非重复的。

（2）计算每个集合中词的 IG，然后按 IG 值从大到小排列。

（3）取前 M 个词作为选取的特征。

在熟悉以上过程后，编写基于信息增益的特征选择 Python 脚本，程序流程图如图 4.4 所示。

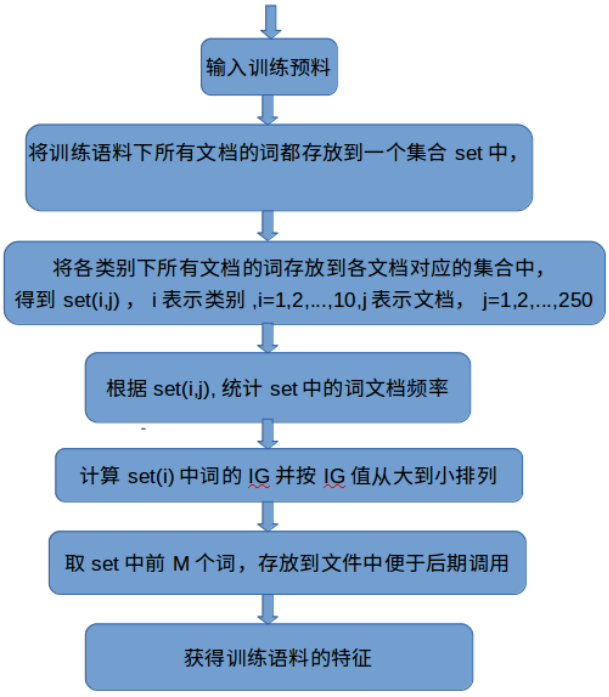

图 4.4 基于 IG 的特征选择流程图

算法实现的部分代码贴图如图 4.5 所示。

```
for eachword in all_class_set:  # 对某个类别下的每一个单词的 a b c d 进行计算
    H_C_t = 0
    H_C_not_t = 0
    a = 0
    b = 0
    c = 0
    d = 0
    for key in ClassCode:
        for eachclass in termClassDic:
            # key = ClassCode[class_local]
            # class_local +=1
            if eachclass == key:  # 在这个类别下进行处理
                for eachdocset in termClassDic[eachclass]:
                    if eachword in eachdocset:
                        a = a + 1
                    else:
                        c = c + 1
            else:  # 不在这个类别下进行处理
                for eachdocset in termClassDic[eachclass]:
                    if eachword in eachdocset:
                        b = b + 1
                    else:
                        d = d + 1
        P_C_t = float(a+1)/float(a+b+C)
        P_C_not_t = float(c+1)/float(c+d+C)
        H_C_t += P_C_t*math.log(P_C_t)
        H_C_not_t += P_C_not_t * math.log(P_C_not_t)
    P_t = float(a + b)/Nall
    P_not_t = float(c + d)/Nall
    eachwordcount_IG = math.log(float(C)) + (P_t * H_C_t + P_not_t * H_C_not_t)
    classTermCountDic[eachword] = eachwordcount_IG
# 这个排序后返回的是元组的列表
sortedClassTermCountDic = sorted(classTermCountDic.items(), key=lambda d: d[1], reverse=True)
# count = 0
termCountDic = dict()
for i in range(K):
    termCountDic[sortedClassTermCountDic[i][0]] = sortedClassTermCountDic[i][1]

return termCountDic  # termCountDic={{w1: IG1, w2: IG2, ......, wk:IGk}}
```

图 4.5 基于 IG 的特征选择核心代码部分贴图

在实验中 M=4840，M 的取值是自己定的，M 的取值影响最后分类器的分类准确率。选取的特征存放到文件中，一共有 4840 个词。特征词的部分截图如图 4.6 所示。

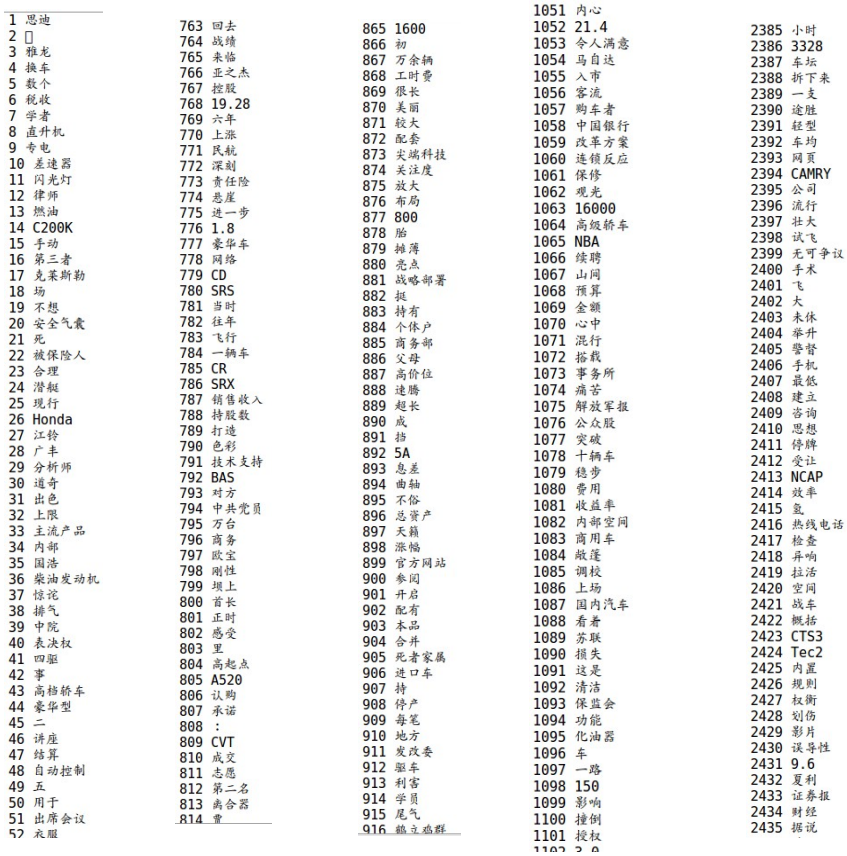

图 4.3 基于信息增益选取的特征部分截图

五、特征权重值

在之前的工作中，已经得到了训练语料的特征，要想对文档进行分类就需要将文档转换成向量的形式，向量的属性是特征中相应的词，向量的值即特征权重，实际上有多种选择，比如布尔变量、词频、逆文档频率、TF-IDF 等。本项目以 TF-IDF 作为特征权重，构建向量以适应分类器的输入。

1、TF-IDF

TF-IDF 方法实际上是计算 TF 值乘以逆向文档频率(Inverse Document Frequency,IDF)值,其中 TF 指的是特征项在文档中出现的频率,对于给定词语 t 的词频,其计算公式是:

$$TF(t)=\frac{n}{N}$$

其中，N 是一个文档中词的数量，n 是该文档中词 t 的数量。

IDF 也称为逆文档频率,是对一个词的普遍重要性的度量,表示包含某个特征项的文档数量越少,这样的文档越能和其他类别的文档区分开来。对于给定特征项 t 的 IDF 值的计算公式是:

$$IDF(t)=\log(\frac{Nall}{n+1})$$

其中，Nall 是所有文档的数量，n 是包含特征 t 的文档数量，n+1 是为了防止 n=0。

$$TFIDF(t)=TF*IDF$$

2、将所有文档转换成向量形式

现在已经获得了特征，并确定以 TFIDF 作为特征的权重值，接下来要将文档转换成向量形式。向量的属性是特征项，属性值是 TFIDF 值。

本项目使用了 libsvm、随机森林做分类器。所以要将文档分别转换成 libsvm 和随机森林接受的输入格式。本项目得到的特征的维度是 4840。

libsvm 的输入数据格式：（类别编号 ，特征项 1 的索引：TFIDF1，特征项 2 的索引：TFIDF2，…… ,特征项 4840 的索引：TFIDF4840），如果 TFIDFi=0（i=1,2,3,…,4840）则对应的特征项无需记入。举个例子来说明，假如特征维度是 3，向量：（2, 1:0.33, 3：0.44）那么 2 是类别编号， 1:0.33 中 1 表示第 1 个特征项的索引, 0.33 表示该特征项对应的 TFIDF 值。该向量中没有第 2 个特征项，说明该特征项对应的 TFIDF 值为 0。

随机森林的输入格式：（类别编号，TFIDF1，TFIDF2，TFIDF3，……,TFIDF4840）

将文档转换成以上两种形式的向量，并将其存放到文件中，得到 libsvm 的输入文件如图 5.1 所示。

图 5.1 libsvm 的输入文件部分截图

随机森林的输入文件如图 5.2 所示。

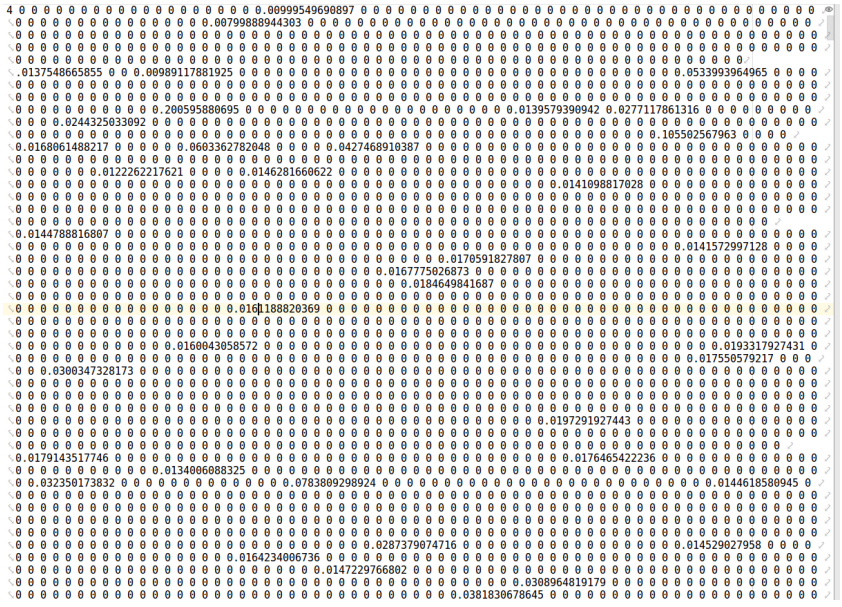

图 5.2 随机森林的输入文件部分截图

从图可以看出，向量的维度很高，总共 4840 维，但是数据很稀疏。

六、分类器训练与结果分析

本项目训练了三种分类器：朴素贝叶斯、SVM 和随机森林, 并对分类器的性能作对比，同时比较卡方统计和信息增益选择的特征的优劣。

1、分类器

1.1 朴素贝叶斯

贝叶斯分类器基于贝叶斯公式，贝叶斯公式如下：

$$P(B \mid A) = \frac{P(A,B)}{P(A)} = \frac{P(B)P(A \mid B)}{P(A)}$$

贝叶斯决策理论：

$$P(c_j \mid \mathbf{x}) = \frac{P(c_j, \mathbf{x})}{P(\mathbf{x})} = \frac{P(c_j)P(\mathbf{x} \mid c_j)}{P(\mathbf{x})}$$

其中，x 表示特征， 表C_j示类别 j。

选择后验概率最大值对应的类别，即为分类器预测的类别：

$$c^* = \arg\max_{j=1,\ldots,C} P(c_j \mid \mathbf{x}) = \arg\max_{j=1,\ldots,C} P(c_j)P(\mathbf{x} \mid c_j)$$

如果对特征 x 作类条件独立假设，即

$$P(\mathbf{x} \mid c_j) \approx P([w_1,\ldots,w_N] \mid c_j) \approx \prod_{k=1}^{N} P(w_k \mid c_j) = \prod_{i=1}^{M} P(w_i \mid c_j)^{N(w_i)}$$

得到朴素贝叶斯分类器：

$$c^* = \arg\max_{j=1,\ldots,C} P(c_j) \prod_{i=1}^{M} P(w_i \mid c_j)^{N(w_i)}$$

1.2 libsvm

实验使用台湾大学林智仁教授提供的 LIBSVM 软件包。该软件包能解决多种分类及回归问题。其中包括 svm-train、svm-predict、svm-scale 和 grid.py 几个主要文件。svm-train 用于训练分类器,svm-predict 对测试集进行分类得到分类结果,svm-scale 用来缩放数据范围,grid.py 会通过网格遍历自动选择最优的参数。

LIBSVM 具体使用流程如下:

Step1:准备相应格式的数据,具体格式为<label><indexl>:<valuel><index2>:<value2>...,其中<label>表示类别信息,<index>为属性的编号,<value>为属性的值。

Step 2:使用 svm-scale 对数据集进行缩放,通常的范围选择为[0,1]。

Step 3:通过调整参数可以选择径向基核函数,利用 svm-train 训练分类器,可以得到 model 文件。

Step4:利用 model 文件和 svm-predict 对测试集进行分类,最终得到分类结果。

1.3 随机森林

随机森林（random forest），顾名思义,随机就是随机抽取，森林就是说这里不止一棵树，而由一群决策树组成的一片森林，连起来就是用随机抽取的方法训练出一群决策树来完成分类任务。

随机森林用了两次随机抽取,一次是对训练样本的随机抽取；另一次是对变量(特征)的随机抽取。这主要是为了解决样本数量有限的问题。

随机森林的核心是由弱变强思想的运用。每棵决策树由于只用了部分变量、部分样本训练而成,可能单个的分类准确率并不是很高。但是当一群这样的决策树组合起来分别对输入数据作出判断时,可以带来较高的准确率。有点类似于俗语 三个臭皮匠顶个诸葛亮。随机森林的分类过程如图 6.1 所示。

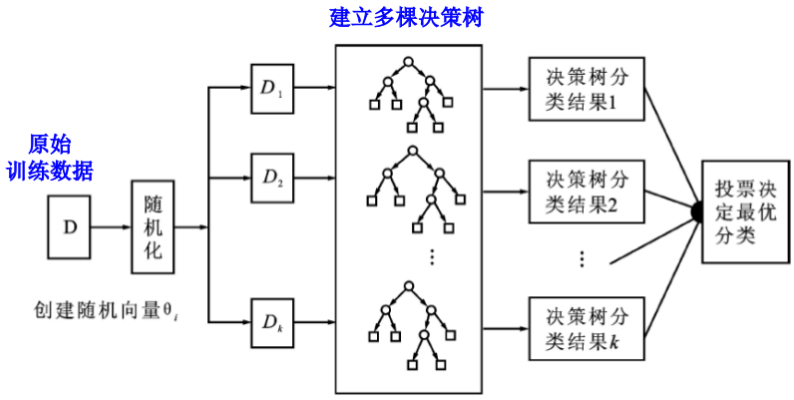

图 6.1 随机森林的分类过程

随机森林有两个重要参数: 随机森林中树的个数和树节点预选的变量个数。这两个参数是训练过程中要设置的，测试准确率主要由这两个参数决定。

随机森林有许多优点：结构简单易懂，能应用于小数量集，分类效果好，训练时间短（相对神经网络），鲁棒性好（因为采用的是投票方式）等。

2、分类评价标准

对于不同的实验目的,分类器的评价标准有很多种,常用的是从信息检索中引入的包括正确率、召回率、F 测度等。以上指标都是针对单个类别而言,对于整体的分类效果主要有两种平均的方法,宏平均及微平均。由于宏 F1 综合考虑了宏正确率及召回率,比单独观察正确率和召回率更能体现分类效果的优劣,因此本项目使用宏 Fl 值作为主要评价指标。

2.1 正确率与召回率

分类器对于某个文本的输出结果会有四种情况,如表 6.1 所示。

表 6.1 分类结果表

分类器判断	Cj	Cj’
YES	aj	bj
NO	cj	dj

表中，对于某个类别 Cj，aj 表示文档实际属于类别 Cj 同时分类器判断文档属于类别 Cj 的文档数, bj 表示文档不属于类别 Cj 但分类器判断其属于类别 Cj 的数量, cj 表示文本属于类别 Cj 而分类器将其分到其他类别的文档的数量, dj 表示文档不属于类别 Cj 分类器也判断其不属于类别 Cj 的数量。

正确率是正确判断为类别 Cj 的文本数与所有判断为类别 Cj 的文本数的比,其公式如下:

$$Pj=\frac{aj}{aj+bj}$$

召回率是正确判断为类别 Cj 的文本数与类别 Cj 中所有样本总数之比:

$$Rj=\frac{aj}{aj+cj}$$

2.2 F1 测度

由于正确率和召回率通常呈反比的关系,在提高正确率时通常召回率都会降低,所以用正确和召回率来评价分类器往往存在缺陷,因此需要引入其他的评价标，通常使用 F1 测度，其公式如下：

$$F1\,j=\frac{2*Pj*Rj}{Pj+Rj}$$

2.3 宏平均

上文所述的正确率、召回率及 Fl 值都是针对单个类别的评价标准,为了判断文档集的整体分类效果需要对它们取平均值。平均的方式有宏平均和微平均两种。宏平均就是将每一类的指标相加,然后除以类别的数量。令 c 表示类别的数量,那么宏平均的计算公式如下:

$$Pmacro=\frac{\sum_{i=1}^{c}Pi}{c}$$

$$Rmacro=\frac{\sum_{i=1}^{c}Ri}{c}$$

$$F1\,macro=\frac{2*Pmacro*Rmacro}{Pmacro+Rmacro}$$

3、实验结果及分析

本项目使用了两种特征选择方法：卡方统计和信息增益。这两种方法选择的特征维度都设成 4840 维。接着，以 TFIDF 作为特征权重值，并将文档转换成分类器接受的向量格式。最后训练了 3 种分类器：朴素贝叶斯、SVM 和随机森林。

分类器的超参数设置如下：

（1）对于 libsvm，使用默认的交叉验参数，得到最优的超参数： c:32, g:0.0078125，其中惩罚因子，g 是径向基函数指数因子。

（2）对于随机森林，设置的超参数：树的个数：1300, 树节点预选的变量个数：2

3.1 基于卡方统计选择特征的分类器测试结果

训练了 3 种基于卡方统计选择特征的分类器： 朴素贝叶斯、SVM 和随机森林。训练好后用测试语料进行测试，使用宏 F1 对分类性能进行评估。三种分类器测试结果分别如表 6.2、6.3 和 6.4 所示。

表 6.2 朴素贝叶斯分类器测试结果

	正确率(%)	召回率(%)	F1 测度(%)
宏平均	84.06	80.95	82.47
汽车	82	95.35	88.17
财经	85.37	77.78	81.4
IT	88.57	67.39	76.54
健康	77.78	76.09	76.92
体育	100	87.23	93.18
旅游	100	58.33	73.68
教育	75.51	78.72	77.08
招聘	76.67	92	83.64
文化	54.67	87.23	67.21
军事	100	89.36	94.38

表 6.3 libsvm 测试结果

	正确率(%)	召回率(%)	F1 测度(%)
宏平均	86.95	84.79	85.86
汽车	95.45	97.67	96.55
财经	97.22	77.78	86.41
IT	81.25	84.78	82.97
健康	72.92	76.09	74.47
体育	97.44	80.85	88.37
旅游	94.59	72.97	82.35
教育	86.66	82.98	84.78
招聘	87.04	94	90.38
文化	59.15	89.36	71.19
军事	97.72	91.49	94.51

表 6.4 随机森林测试结果

	正确率(%)	召回率(%)	F1 测度(%)
宏平均	87.07	86.31	86.69
汽车	95.45	97.67	96.55

财经	97.36	82.22	89.16
IT	78	84.78	81.25
健康	75.51	80.43	77.89
体育	97.83	95.74	96.77
旅游	92.68	79.17	85.39
教育	86.36	80.85	83.52
招聘	82.14	92	86.79
文化	69.81	78.72	74
军事	95.56	91.48	93.48

总结以上实验结果，统计 3 种分类器的宏 F1 测度和各类下的 F1 测度，如表 6.5 所示。

表 6.5　基于卡方统计特征选择下 3 种分类器的宏 F1 测度和各类下的 F1 测度

	朴素贝叶斯	libsvm	随机森林
宏 F1	82.47	85.86	86.69
汽车的 F1	88.17	96.55	96.55
财经的 F1	81.4	86.41	89.16
IT 的 F1	76.54	82.97	81.25
健康的 F1	76.92	74.47	77.89
体育的 F1	93.18	88.37	96.77
旅游的 F1	73.68	82.35	85.39
教育的 F1	77.08	84.78	83.52
招聘的 F1	83.64	90.38	86.79
文化的 F1	67.21	71.19	74
军事的 F1	94.38	94.51	93.48

3.2 基于信息增益选择特征的分类器测试结果

本项目还训练了 3 种基于信息增益选择特征的分类器： 朴素贝叶斯、SVM 和随机森林。训练好后用测试语料进行测试，使用宏 F1 对分类性能进行评估。三种分类器测试结果分别如表 6.6、6.7 和 6.8 所示。

表 6.6 朴素贝叶斯分类器测试结果

	正确率(%)	召回率(%)	F1 测度(%)
宏平均	80.26	72.43	76.15
汽车	56.16	95.34	70.68
财经	80.43	82.22	81.32
IT	82.60	41.30	55.07
健康	88.57	67.39	76.54
体育	100	63.83	77.92
旅游	100	41.66	58.82
教育	72.91	74.46	73.68
招聘	77.58	90	83.33
文化	44.33	91.48	59.72
军事	100	57.44	69.87

表 6.7 libsvm 测试结果

	正确率(%)	召回率(%)	F1 测度(%)
宏平均	86.91	85.03	85.96
汽车	91.30	97.67	94.38
财经	97.36	82.22	89.16
IT	85.11	86.96	86.02
健康	76.59	78.26	77.41
体育	97.56	85.11	90.91
旅游	94.29	68.75	79.52
教育	88.89	85.11	86.96
招聘	82.65	96	88.89
文化	57.58	80.85	67.25
军事	97.67	89.36	93.33

表 6.8 随机森林测试结果

	正确率(%)	召回率(%)	F1 测度(%)
宏平均	84.92	83.63	84.27
汽车	97.22	81.39	88.61
财经	97.36	82.22	89.15

IT	76.47	84.78	80.41
健康	76.08	76.08	76.08
体育	92	97.87	94.85
旅游	86.84	68.75	76.74
教育	86.04	78.72	82.22
招聘	80.70	92	85.98
文化	65	82.98	72.89
军事	91.48	91.48	91.48

总结以上实验结果，统计 3 种分类器的宏 F1 测度和各类下的 F1 测度，如表 6.9 所示。

表 6.9 基于信息增益特征选择下，3 种分类器的宏 F1 测度和各类下的 F1 测度

	朴素贝叶斯	libsvm	随机森林
宏 F1	76.15	85.96	84.27
汽车的 F1	70.68	94.38	88.61
财经的 F1	81.32	89.16	89.15
IT 的 F1	55.07	86.02	80.41
健康的 F1	76.54	77.41	76.08
体育的 F1	77.92	90.91	94.85
旅游的 F1	58.82	79.52	76.74
教育的 F1	73.68	86.96	82.22
招聘的 F1	83.33	88.89	85.98
文化的 F1	59.72	67.25	72.89
军事的 F1	69.87	93.33	91.48

3.3 实验分析

3.3.1 对基于卡方统计选择特征的分类器测试结果进行分析

（1）以宏 F1 作为 3 种分类器整体性能的评估指标，对比表 6.5 知，分类器的性能：贝叶斯分类器<libsvm<随机森林。

（2）对于各类别，随机森林下的 F1 测度多数比其他两个分类器的要高，也正是因此随机森林的整体性能比其他的两个分类器要高。

（3）各分类器对“文化”的 F1 值都是最低的，因为文化的范畴很广，实验中对“文化”类文档选取的特征的覆盖面小，如果增大“文化”类的训练语料，可以提高分类器对该类的预测性能，从而提高分类器的整体预测性能。

（4）虽然朴素贝叶斯分类器的整体性能最低，但是对“健康”和“体育”这两类的 F1 值比 libsvm 的要高一些，对这两类的分类适合使用朴素贝叶斯。

3.3.2 对基于信息增益选择特征的分类器测试结果进行分析

（1）以宏 F1 作为 3 种分类器整体性能的评估指标，对比表 6.9 知，分类器的性能：贝叶斯分类器<随机森林 <libsvm。

（2）对于各类别，libsvm 下的 F1 测度多数比其他两个分类器的要高，也正是因此 libsvm 的整体性能比其他的两个分类器要高。

（3）各分类器对“文化”的 F1 值仍是最低的，因为文化的范畴很广，实验中对“文化”类文档选取的特征的覆盖面小，如果增大“文化”类的训练语料，可以提高分类器对该类的预测性能，从而提高分类器的整体预测性能。

（4）朴素贝叶斯分类器的整体性能最低，其对各类别的 F1 值比其他两个分类器的要低很多。

3.3.3 对基于卡方统计和信息增益选择特征的分类器测试结果进行对比分析

对比表 6.5 和表 6.9：

（1）对于朴素贝叶斯分类器，使用卡方统计选取的特征比信息增益选取的，在性能上有很大的提升。

（2）对于随机森林，使用卡方统计选取的特征比信息增益选取的，在性能上有所提升。

（3）卡方统计选择的特征整体上比信息增益选择的要好些。

总结对分类器性能的影响因素有：分词质量、停用词表、特征选择方法、分类器的选择、分类器超参数的设置等。要想提高分类性能的重点在于选择出非常好的特征，可以使用多种方法融合使用的方式来提高选出到的特征的质量。

七、总结

在做本项目前，我对 NLP 的理解只停留在课程 PPT 上，对实际中如何作文本处理没有一点概念，在选择项目题目时，我起初想选有关分词的，因为分词在中文处理中占有及其重要的地位，再者，因为我上过“随机过程”这门课程，想借此更加深入理解 HMM，但又想到我的专业和 NLP 的应用有关，而时间又有限，就选择了只做文本分类。起初通过网上的博客了解了分类的大致过程，然后在 github 上查找了有关的代码，找到了使用 TFIDF 来选择特征，使用 libsvm 做分类的代码，通过看代码知道了算法的具体实现，然后在此基础上换成其他方法，比如我使用了卡方统计和信息增益这两种方法来选择特征，最后增加了随机森林和朴素贝叶斯。通过本项目，提高了我的编程能力，让我掌握了如何使用卡方统计和信息增益来提取特征，以及如何计算 TFIDF 作为特征权值，最后掌握了朴素贝叶斯的编程实现、随机森林的超参数的调试，以及学会了使用 libsvm 这个简单好用的 svm 工具。

最后感谢老师们精彩的讲解，是老师们把我带到了神奇的 NLP 世界。

参考文献

[1] Luhn ranking[C]//ProceedingsResearch and development in retrieval.ACM,201 3:343-352.
space[J].arXiv preprint vector on abstracts[J].IBM Journal of research and development,1
958,2(2):1 59・1 65.
[2] Maron ME,Kuhns J L.On relevance,probabilistic indexing and retrieval[J].Journal
[3] Salton Journal information ofthe ACM(JACM),1960,7(3):216-244. G,Yang C.On the
specification of term values in automatic indexing[J].ofdocumentation,1973,29(4):351・372.
[4]卜凡军.KNN 算法的改进及其在文本分类中的应用【D】.江南大学,2009.
[5] 李东晖,杜树新,吴铁军.基于壳向量的线性支持向量机快速增量学习算法[J】.浙江大学学报(工学版),2006,(02):202.206.
[6]邸鹏,段利国.一种新型朴素贝叶斯文本分类算法[J】.数据采集与处理,2014,(0I):71・75.
[7]徐振强,李保利.结合词语分布信息 I 钓 TFIDF 关键词抽取方法研究【J】.中原工学院学报,2014,(06):59.63.
[8]王行恒,曹军,邓学,等.基于循环迭代算法改进的矸 IDF 方法及应用【J】.计算机应用与软件,2012,(11):305.308.
[9]谢静.基-于 LDA 与 SVM 的文本分类研究【D】.河北大学,2012.
[10] 张志飞,苗夺谦,高灿.基于 LDA 主题模型的短文本分类方法【J】.计算机应用, 201 3,(06):1 587.1 590.
[11] 数据挖掘 文本分类（一） 综述 - yangshaoby 的专栏 - 博客频道 – CSDN.NET
[12] 文本分类中的特征选择_german_新浪博客