

# 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](mailto: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<sup>[1]</sup>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<sup>[2]</sup>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<sup>[3]</sup>, 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<sup>[4]</sup>。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<sup>[5]</sup>。邱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<sup>[6]</sup>。

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<sup>[7]</sup>。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<sup>[8]</sup>。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<sup>[9]</sup>。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. 主题矩阵解决该方法, 并且结合 KNN 对新闻标题进行分类, 取得了比较好的效果<sup>[10]</sup>。

### 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/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？据国家安全生产监督管理局的数字显示，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。

?	>	啊	和	依
、	<	阿	何	依
。		哎	何	照
“	nbsp1	哎呀	处	矣
”	nbsp2	哎哟	何	以
《	nbsp3	唉	况	以
》	nbsp4	俺	何	使
!	nbsp5	俺们	时	及
,	nbsp6	按照	嘿	以
:	nbsp7	吧	哼	免
;	nbsp8	吧	哼	以
?	nbsp9	吧	哼	至
@	nbsp10	把	乎	于
#	3.6	罢了	还	以
\$	about	被	有	致
%	there	本	换	抑
^	see	本	句	或
&	can	着	说	因
*	U	比	之	此
(	L	比	或	因
)	in	方	是	而
-	a	如	或	为
+	0144	鄙	者	用
=	our	人	了	由
\	1	彼	及	此
	2	此	极	可
}	3	边	及	见
[	4	别	其	由
}	5	说	至	于
{	6	别	即	有
,	7	并	或	的
/	8	且	令	有
	9	比	即	些
t	0	成	若	又
		单	使	于
		不	几	是
		但	时	乎
		独	已	与
			既	此
			然	同
			显	时
				否
				与
				其
				越
				是
				云

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。

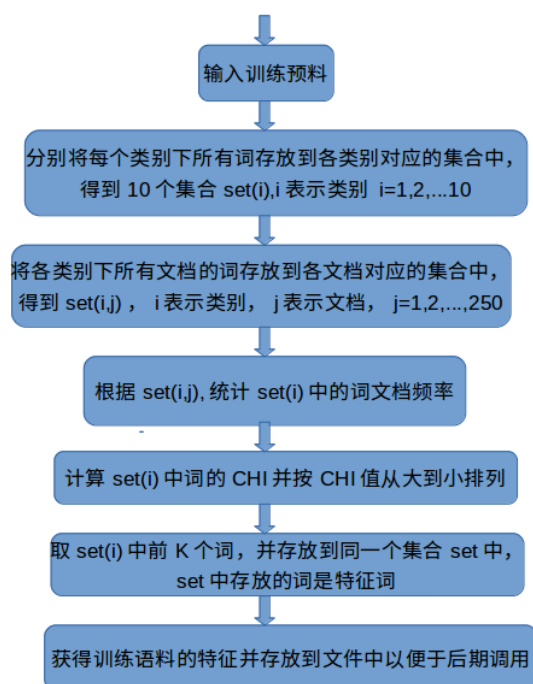


图 4.1 基于 CHI 的特征选择流程图

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:{w1: chi, w2: chi2, ....., wk:chik}}
  
```

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。

1	Negroponte	97	卡洛斯	148	招待	2586	悦达起亚	3500	坐落
2	雄浑	98	首轮	149	人机界面	2587	不良反应	3501	乐园
3	□	99	国防科技	150	市场份额	2588	造型	3502	理发店
4	九寨沟	100	自驾游	151	正股	2589	1945	3503	高俊闻
5	0.12	101	主战	152	开奖	2590	补救	3504	上班
6	直升机	102	沙尔克	153	一等奖	2591	有没有	3505	择业
7	专电	103	晚年	154	队伍	2592	激素	3506	开市价
8	配装	104	同声传	155	利率	2593	相关	3507	得手
9	歌	105	成本	156	陆军	2594	见习生	3508	篮球
10	律师	106	首席	157	民俗风情	2595	装配	3509	人才流动
11	燃油	107	壮美	158	the	2596	华晨	3510	长假
12	孕妇	108	提交	159	贾秀全	2597	转让	3511	期同
13	机载	109	with	160	指引	2598	基础知识	3512	微错
14	休闲	110	职责	161	公司股票	2599	尊贵	3513	例
15	董事长	111	订票	162	中国男篮	2600	金融市场	3514	导游
16	晚清	112	总决赛	163	中新网	2601	创作	3515	编辑
17	场	113	毛主席	164	org	2602	认真	3516	客观
18	下一代	114	KL178	165	著名作家	2603	血糖	3517	武汉队
19	安全气象	115	企业	166	易用	2604	卓越	3518	粘膜
20	死	116	丰田公司	167	38500	2605	安全隐患	3519	例
21	汽车站	117	探测	168	然	2606	山水	3520	迟刻
22	二队	118	中旅总社	169	to	2607	右脚	3521	王毓国
23	大连	119	进气道	170	人类	2608	队徽	3522	财报
24	赛迪	120	重量	171	高涨	2609	短信	3523	精度
25	毛泽东	121	教学大纲	172	后防	2610	临床实验	3524	遗产
26	任天堂	122	登录	173	笔法	2611	扭矩	3525	脓肿
27	束河	123	飞度	174	精神	2612	诊所	3526	防守反击
28	冲锋枪	124	黑博士	175	台式机	2613	总分	3527	有点
29	训练营	125	凯迪拉克	176	冷战	2614	接入	3528	试飞员
30	分析师	126	雷达	177	大师赛	2615	再次	3529	消息
31	画廊	127	好身材	178	血液循环	2616	中国网通	3530	频
32	范文	128	追债	179	赶走	2617	剂量	3531	整形外科
33	CT	129	哈尔滨	180	公众股	2618	张舵	3532	修订稿
34	注意事项	130	跑车	181	就业人数	2619	弘扬	3533	有限责任
35	现代化	131	轿车	182	巴士	2620	字	3534	一生
36	内部	132	支持者	183	交通事故	2621	不予	3535	难点
37	国浩	133	顾萌	184	官兵们	2622	专业	3536	油画
38	学者	134	军费	185	药房	2623	天第	3537	亿至
39	拌	135	洗头妹	186	腹胀	2624	晨报讯	3538	提示
40	效应	136	登记手续	187	重大	2625	驾驶	3539	非流通股
41	年薪	137	系列	188	护士	2626	恢复性	3540	还原
42	待遇	138	解决方案	189	参考价格	2627	东风	3541	红色旅游
43	表决权	139	语文	190	受不了	2628	书写	3542	艺术家
44	通便	140	肇俊哲	191	职业联赛	2629	SS	3543	职业咨询
45	城市	141	驾驶者	192	以晴	2630	实际	3544	思念
46	豪华型	142	VS	193	志愿	2631	讯	3545	今天下午
47	门诊	143	陈水扁	194	糖分	2632	沙尘	3546	合股
48	结算	144	两走	195	出来	2633	行使	3547	文件
49	升空	145	迪志	196	怀孕	2634	亿美元	3548	小型车
50	周于	146	柱	197	罚球	2635	旧址	3549	体外
51	出席会议	147	地理信息系统	198	重庆力帆	2636	造化	3550	发美
52	亩田	148	接待	199	陈逸飞	2637	事权责任	3551	不做

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：

$$IG(ti) = - \sum_{j=1}^c P(C_j) * \log P(C_j) + P(ti) * \left( \sum_{j=1}^c P(C_j/ti) * \log P(C_j/ti) \right) + P(ti') * \left( \sum_{j=1}^c P(C_j/ti') * \log P(C_j/ti') \right)$$

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：

$$P(C_j) = \frac{A_{ij} + C_{ij}}{N} \quad P(ti) = \frac{A_{ij} + B_{ij}}{N}$$

$$P(ti') = \frac{C_{ij} + D_{ij}}{N}$$

$$P(C_j/t_i) = \frac{A_{ij}+1}{A_{ij}+B_{ij}+c}$$

$$P(C_j/t_i') = \frac{C_{ij}+1}{C_{ij}+D_{ij}+c}$$

以上公式中：

$A_{ij}$ ：j 类中，含特征项  $t_i$  的文档数量

$B_{ij}$ ：在非 j 类中，含特征项  $t_i$  的文档数量

$C_{ij}$ ：j 类中，不含特征项  $t_i$  的文档数量

$D_{ij}$ ：在非 j 类中，不含特征项  $t_i$  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。

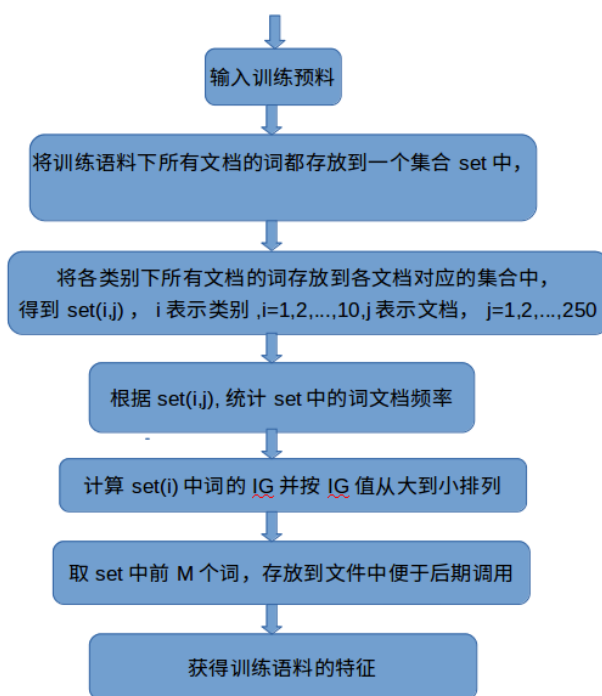


图 4.4 基于 IG 的特征选择流程图

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。

1 恩迪	763 回去	865 1600	1051 内心	2385 小时
2 []	764 战绩	866 初	1052 21.4	2386 3328
3 雅光	765 来临	867 万余辆	1053 令人满意	2387 车坛
4 换车	766 亚之杰	868 工时费	1054 马自达	2388 拆下来
5 数个	767 控股	869 很长	1055 入市	2389 一支
6 税收	768 19.28	870 美丽	1056 客流	2390 途胜
7 学者	769 六年	871 较大	1057 购车者	2391 轻型
8 直升机	770 上涨	872 配套	1058 中国银行	2392 车均
9 专电	771 民航	873 尖端科技	1059 改革方案	2393 网页
10 差速器	772 深刻	874 关注度	1060 连锁反应	2394 CAMRY
11 闪光灯	773 责任险	875 放大	1061 保修	2395 公司
12 律师	774 悬崖	876 布局	1062 观光	2396 流行
13 燃油	775 进一步	877 800	1063 16000	2397 壮大
14 C200K	776 1.8	878 胎	1064 高级轿车	2398 试飞
15 手动	777 豪华车	879 排薄	1065 NBA	2399 无可争议
16 第三者	778 网络	880 亮点	1066 续聘	2400 手术
17 克莱斯勒	779 CD	881 战略部署	1067 山同	2401 飞
18 场	780 SRS	882 挺	1068 预算	2402 大
19 不想	781 当时	883 持有	1069 金额	2403 未休
20 安全气囊	782 往年	884 个体户	1070 心中	2404 攀升
21 死	783 飞行	885 商务部	1071 混行	2405 警督
22 被保险人	784 一辆车	886 父母	1072 搭载	2406 手机
23 合理	785 CR	887 高价位	1073 事务所	2407 最低
24 潜艇	786 SRX	888 连腾	1074 痛苦	2408 建立
25 现行	787 销售收入	889 超长	1075 解放军报	2409 咨询
26 Honda	788 持股数	890 成	1076 公众股	2410 思想
27 江铃	789 打造	891 指	1077 突破	2411 停牌
28 广丰	790 色彩	892 5A	1078 十辆车	2412 受让
29 分析师	791 技术支持	893 息差	1079 稳步	2413 NCAP
30 道奇	792 BAS	894 曲轴	1080 费用	2414 效率
31 出色	793 对方	895 不俗	1081 收益率	2415 氢
32 上限	794 中共党员	896 总资产	1082 内部空间	2416 热线电话
33 主流产品	795 万台	897 天籁	1083 商用车	2417 检查
34 内部	796 商务	898 涨幅	1084 敞篷	2418 异响
35 国浩	797 欧宝	899 官方网站	1085 调校	2419 拉活
36 柴油发动机	798 刚性	900 参阅	1086 上场	2420 空间
37 惊诧	799 坝上	901 开启	1087 国内汽车	2421 战车
38 排气	800 首长	902 配有	1088 看看	2422 概括
39 中院	801 正时	903 本品	1089 苏联	2423 CTS3
40 表决权	802 感受	904 合并	1090 损失	2424 Tec2
41 四驱	803 里	905 死者家属	1091 这是	2425 内置
42 事	804 高起点	906 进口车	1092 清洁	2426 规则
43 高档轿车	805 A520	907 特	1093 保监会	2427 权衡
44 豪华型	806 认购	908 停产	1094 功能	2428 划伤
45 二	807 承诺	909 每笔	1095 化油器	2429 影片
46 浮座	808 :	910 地方	1096 车	2430 误导性
47 结算	809 CVT	911 发改委	1097 一路	2431 9.6
48 自动控制	810 成交	912 驱动车	1098 150	2432 复利
49 五	811 志愿	913 利害	1099 影响	2433 证券报
50 用于	812 第二名	914 学员	1100 撞倒	2434 财经
51 出席会议	813 离合器	915 尾气	1101 授权	2435 据说
52 衣服	814 童	916 独立族群		

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。



4 20:0.00999549690897 73:0.00799888944303 303:0.0137548665855 306:0.00909117881925 342:0.0533993964965 488:0.200595880695  
510:0.0139579390942 511:0.0277117861316 524:0.0244325033092 630:0.105502567963 635:0.0168061488217 641:0.0603362782048 646:0  
.0427468910387 750:0.0122262217621 755:0.0146281660622 838:0.0141098817028 1044:0.0144788816807 1155:0.0141572997128 1194:0  
.0170591827807 1247:0.0167775026873 1307:0.0184649841687 1416:0.0161188820369 1599:0.0160043085872 1636:0.0193317927431  
1692:0.017550579217 1698:0.0308347320173 1991:0.0197291927443 2134:0.0170143517746 2171:0.0176465422236 2196:0.0134086088325  
2244:0.023250173832 2258:0.0783809298924 2285:0.0144618580945 2575:0.0287379074716 2593:0.014529027958 2615:0.0164234066736  
2680:0.0147229766802 2754:0.0380964819179 2807:0.0381830678645 2921:0.0142242926335 3156:0.0163280170655 3177:0.34069350601  
3247:0.0389837998329 3272:0.0156693699303 3330:0.0178213162674 3357:0.0138648092634 3558:0.0455779064463 3613:0.0530903492804  
3614:0.0174871537267 3615:0.0174161297209 3616:0.0177235729327 3619:0.0180191306479 3635:0.0143066457553 3636:0.0143264075159  
3748:0.0165071516244 3768:0.0176798315885 3793:0.0382909434195 3819:0.0158054980386 3842:0.0209557118387 4008:0.68871424552  
4027:0.0206589990792 4128:0.044236680101 4235:0.0205808598191 4277:0.0600781518928 4301:0.0422283818909 4332:0.013941742441  
4351:0.0425215443046 4353:0.0138648092634 4412:0.0601041168656 4421:0.0167849198559 4522:0.0165472976432 4568:0.0195050380445  
4573:0.0193055075117 4580:0.0194078925973 4691:0.200595880695 4696:0.0514240355849 4767:0.015051322132 4802:0.0144846648266  
4 17:0.0151115899418 36:0.0140742618359 102:0.0725145264508 144:0.023089505257 488:0.17899324739 491:0.0267028922638 594:0  
.0289491045711 630:0.0448289299771 647:0.0252462853206 676:0.0195867200042 694:0.0203206147627 742:0.0741591178849 750:0  
.0363651724207 755:0.0217547085028 853:0.025256727133 917:0.058032302591 967:0.0286721427355 972:0.0259204729064 1037:0  
.027193257942 1195:0.0291656228495 1276:0.027924939863 1436:0.02055687818 1445:0.0310213313337 1473:0.0812768871738 1550:0  
.059330772679 1705:0.020187201772 1795:0.0811414082173 1928:0.0224290794711 2060:0.095543690993 2134:0.0532037129705 2206:0  
.0282625120864 2442:0.04230808779976 2463:0.0204228356821 2572:0.0303902745142 2673:0.0592972981512 2815:0.0311628631204  
2850:0.0261175842089 2898:0.031972915133 2901:0.0274394058374 2921:0.0211540762241 2983:0.061471671169 3005:0.0268405378655  
3023:0.0148126250818 3211:0.0322119863946 3272:0.0232139347682 3300:0.0255061005515 3323:0.0208750301575 3332:0.0252014219924  
3355:0.023658183981 3403:0.0268114484594 3409:0.0275042825166 3412:0.0219036266335 3477:0.382528201115 3538:0.027129729741  
3585:0.0236946744602 3615:0.0250012083147 3618:0.0262168719169 3619:0.0207076933738 3635:0.021339011636 3636:0.021305393826  
3669:0.0663867644102 3762:0.056174467855 3831:0.0246001544002 3833:0.0220096285384 4003:0.0292198751315 4128:0.021235277315  
4332:0.0414669490338 4352:0.0219454354711 4399:0.0227023026477 4522:0.0246088016232 4542:0.06020802994095 4679:0.0281356482106  
4691:0.17899324739 4791:0.0994342580843 4826:0.0275198358665  
4 17:0.0151115899418 36:0.0140742618359 102:0.0725145264508 144:0.023089505257 488:0.17899324739 491:0.0267028922638 594:0  
.0289491045711 630:0.0448289299771 647:0.0252462853206 676:0.0195867200042 694:0.0203206147627 742:0.0741591178849 750:0  
.0363651724207 755:0.0217547085028 853:0.025256727133 917:0.058032302591 967:0.0286721427355 972:0.0259204729064 1037:0  
.027193257942 1195:0.0291656228495 1276:0.027924939863 1436:0.02055687818 1445:0.0310213313337 1473:0.0812768871738 1550:0  
.059330772679 1705:0.020187201772 1795:0.0811414082173 1928:0.0224290794711 2060:0.095543690993 2134:0.0532037129705 2206:0  
.0282625120864 2442:0.04230808779976 2463:0.0204228356821 2572:0.0303902745142 2673:0.0592972981512 2815:0.0311628631204  
2850:0.0261175842089 2898:0.031972915133 2901:0.0274394058374 2921:0.0211540762241 2983:0.061471671169 3005:0.0268405378655  
3023:0.0148126250818 3211:0.0322119863946 3272:0.0232139347682 3300:0.0255061005515 3323:0.0208750301575 3332:0.0252014219924  
3355:0.023658183981 3403:0.0268114484594 3409:0.0275042825166 3412:0.0219036266335 3477:0.382528201115 3538:0.027129729741  
3585:0.0236946744602 3615:0.0250012083147 3618:0.0262168719169 3619:0.0207076933738 3635:0.021339011636 3636:0.021305393826  
3669:0.0663867644102 3762:0.056174467855 3831:0.0246001544002 3833:0.0220096285384 4003:0.0292198751315 4128:0.021235277315  
4332:0.0414669490338 4352:0.0219454354711 4399:0.0227023026477 4522:0.0246088016232 4542:0.06020802994095 4679:0.0281356482106  
4691:0.17899324739 4791:0.0994342580843 4826:0.0275198358665  
4 17:0.0151115899418 36:0.0140742618359 102:0.0725145264508 144:0.023089505257 488:0.17899324739 491:0.0267028922638 594:0  
.0289491045711 630:0.0448289299771 647:0.0252462853206 676:0.0195867200042 694:0.0203206147627 742:0.0741591178849 750:0  
.0363651724207 755:0.0217547085028 853:0.025256727133 917:0.058032302591 967:0.0286721427355 972:0.0259204729064 1037:0  
.027193257942 1195:0.0291656228495 1276:0.027924939863 1436:0.02055687818 1445:0.0310213313337 1473:0.0812768871738 1550:0  
.059330772679 1705:0.020187201772 1795:0.0811414082173 1928:0.0224290794711 2060:0.095543690993 2134:0.0532037129705 2206:0  
.0282625120864 2442:0.04230808779976 2463:0.0204228356821 2572:0.0303902745142 2673:0.0592972981512 2815:0.0311628631204  
2850:0.0261175842089 2898:0.031972915133 2901:0.0274394058374 2921:0.0211540762241 2983:0.061471671169 3005:0.0268405378655  
3023:0.0148126250818 3211:0.0322119863946 3272:0.0232139347682 3300:0.0255061005515 3323:0.0208750301575 3332:0.0252014219924  
3355:0.023658183981 3403:0.0268114484594 3409:0.0275042825166 3412:0.0219036266335 3477:0.382528201115 3538:0.027129729741  
3585:0.0236946744602 3615:0.0250012083147 3618:0.0262168719169 3619:0.0207076933738 3635:0.021339011636 3636:0.021305393826  
3669:0.0663867644102 3762:0.056174467855 3831:0.0246001544002 3833:0.0220096285384 4003:0.0292198751315 4128:0.021235277315  
4332:0.0414669490338 4352:0.0219454354711 4399:0.0227023026477 4522:0.0246088016232 4542:0.06020802994095 4679:0.0281356482106  
4691:0.17899324739 4791:0.0994342580843 4826:0.0275198358665  
4 17:0.0151115899418 36:0.0140742618359 102:0.0725145264508 144:0.023089505257 488:0.17899324739 491:0.0267028922638 594:0  
.0289491045711 630:0.0448289299771 647:0.0252462853206 676:0.0195867200042 694:0.0203206147627 742:0.0741591178849 750:0  
.0363651724207 755:0.0217547085028 853:0.025256727133 917:0.058032302591 967:0.0286721427355 972:0.0259204729064 1037:0  
.027193257942 1195:0.0291656228495 1276:0.027924939863 1436:0.02055687818 1445:0.0310213313337 1473:0.0812768871738 1550:0  
.059330772679 1705:0.020187201772 1795:0.0811414082173 1928:0.0224290794711 2060:0.095543690993 2134:0.0532037129705 2206:0  
.0282625120864 2442:0.04230808779976 2463:0.0204228356821 2572:0.0303902745142 2673:0.0592972981512 2815:0.0311628631204  
2850:0.0261175842089 2898:0.031972915133 2901:0.0274394058374 2921:0.0211540762241 2983:0.061471671169 3005:0.0268405378655  
3023:0.0148126250818 3211:0.0322119863946 3272:0.0232139347682 3300:0.0255061005515 3323:0.0208750301575 3332:0.0252014219924  
3355:0.023658183981 3403:0.0268114484594 3409:0.0275042825166 3412:0.0219036266335 3477:0.382528201115 3538:0.027129729741  
3585:0.0236946744602 3615:0.0250012083147 3618:0.0262168719169 3619:0.0207076933738 3635:0.021339011636 3636:0.021305393826  
3669:0.0663867644102 3762:0.056174467855 3831:0.0246001544002 3833:0.0220096285384 4003:0.0292198751315 4128:0.021235277315  
4332:0.0414669490338 4352:0.0219454354711 4399:0.0227023026477 4522:0.0246088016232 4542:0.06020802994095 4679:0.0281356482106  
4691:0.17899324739 4791:0.0994342580843 4826:0.0275198358665  
4 17:0.0151115899418 36:0.0140742618359 102:0.0725145264508 144:0.023089505257 488:0.17899324739 491:0.0267028922638 594:0  
.0289491045711 630:0.0448289299771 647:0.0252462853206 676:0.0195867200042 694:0.0203206147627 742:0.0741591178849 750:0  
.0363651724207 755:0.0217547085028 853:0.025256727133 917:0.058032302591 967:0.0286721427355 972:0.0259204729064 1037:0  
.027193257942 1195:0.0291656228495 1276:0.027924939863 1436:0.02055687818 1445:0.0310213313337 1473:0.0812768871738 1550:0  
.059330772679 1705:0.020187201772 1795:0.0811414082173 1928:0.0224290794711 2060:0.095543690993 2134:0.0532037129705 2206:0  
.0282625120864 2442:0.04230808779976 2463:0.0204228356821 2572:0.0303902745142 2673:0.0592972981512 2815:0.0311628631204  
2850:0.0261175842089 2898:0.031972915133 2901:0.0274394058374 2921:0.0211540762241 2983:0.061471671169 3005:0.0268405378655  
3023:0.0148126250818 3211:0.0322119863946 3272:0.0232139347682 3300:0.0255061005515 3323:0.0208750301575 3332:0.0252014219924  
3355:0.023658183981 3403:0.0268114484594 3409:0.0275042825166 3412:0.0219036266335 3477:0.382528201115 3538:0.027129729741  
3585:0.0236946744602 3615:0.0250012083147 3618:0.0262168719169 3619:0.0207076933738 3635:0.021339011636 3636:0.021305393826  
3669:0.0663867644102 3762:0.056174467855 3831:0.0246001544002 3833:0.0220096285384 4003:0.0292198751315 4128:0.021235277315  
4332:0.0414669490338 4352:0.0219454354711 4399:0.0227023026477 4522:0.0246088016232 4542:0.06020802994095 4679:0.0281356482106  
4691:0.17899324739 4791:0.0994342580843 4826:0.0275198358665  
4 17:0.0151115899418 36:0.0140742618359 102:0.0725145264508 144:0.023089505257 488:0.17899324739 491:0.0267028922638 594:0  
.0289491045711 630:0.0448289299771 647:0.0252462853206 676:0.0195867200042 694:0.0203206147627 742:0.0741591178849 750:0  
.0363651724207 755:0.0217547085028 853:0.025256727133 917:0.058032302591 967:0.0286721427355 972:0.0259204729064 1037:0  
.027193257942 1195:0.0291656228495 1276:0.027924939863 1436:0.02055687818 1445:0.0310213313337 1473:0.0812768871738 1550:0  
.059330772679 1705:0.020187201772 1795:0.0811414082173 1928:0.0224290794711 2060:0.095543690993 2134:0.0532037129705 2206:0  
.0282625120864 2442:0.04230808779976 2463:0.0204228356821 2572:0.0303902745142 2673:0.0592972981512 2815:0.0311628631204  
2850:0.0261175842089 2898:0.031972915133 2901:0.0274394058374 2921:0.0211540762241 2983:0.061471671169 3005:0.0268405378655  
3023:0.0148126250818 3211:0.0322119863946 3272:0.0232139347682 3300:0.0255061005515 3323:0.0208750301575 3332:0.0252014219924  
3355:0.023658183981 3403:0.0268114484594 3409:0.0275042825166 3412:0.0219036266335 3477:0.382528201115 3538:0.027129729741  
3585:0.0236946744602 3615:0.0250012083147 3618:0.0262168719169 3619:0.0207076933738 3635:0.021339011636 3636:0.021305393826  
3669:0.0663867644102 3762:0.056174467855 3831:0.0246001544002 3833:0.0220096285384 4003:0.0292198751315 4128:0.021235277315  
4332:0.0414669490338 4352:0.0219454354711 4399:0.0227023026477 4522:0.0246088016232 4542:0.06020802994095 4679:0.0281356482106  
4691:0.17899324739 4791:0.0994342580843 4826:0.0275198358665  
4 17:0.0151115899418 36:0.0140742618359 102:0.0725145264508 144:0.023089505257 488:0.17899324739 491:0.0267028922638 594:0  
.0289491045711 630:0.0448289299771 647:0.0252462853206 676:0.0195867200042 694:0.0203206147627 742:0.0741591178849 750:0  
.0363651724207 755:0.0217547085028 853:0.025256727133 917:0.058032302591 967:0.0286721427355 972:0.0259204729064 1037:0  
.027193257942 1195:0.0291656228495 1276:0.027924939863 1436:0.02055687818 1445:0.0310213313337 1473:0.0812768871738 1550:0  
.059330772679 1705:0.020187201772 1795:0.0811414082173 1928:0.0224290794711 2060:0.095543690993 2134:0.0532037129705 2206:0  
.0282625120864 2442:0.04230808779976 2463:0.0204228356821 2572:0.0303902745142 2673:0.0592972981512 2815:0.0311628631204  
2850:0.0261175842089 2898:0.031972915133 2901:0.0274394058374 2921:0.0211540762241 2983:0.061471671169 3005:0.0268405378655  
3023:0.0148126250818 3211:0.0322119863946 3272:0.0232139347682 3300:0.0255061005515 3323:0.0208750301575 3332:0.0252014219924  
3355:0.023658183981 3403:0.0268114484594 3409:0.0275042825166 3412:0.0219036266335 3477:0.382528201115 3538:0.027129729741  
3585:0.0236946744602 3615:0.0250012083147 3618:0.0262168719169 3619:0.0207076933738 3635:0.021339011636 3636:0.021305393826  
3669:0.0663867644102 3762:0.056174467855 3831:0.0246001544002 3833:0.0220096285384 4003:0.0292198751315 4128:0.021235277315  
4332:0.0414669490338 4352:0.0219454354711 4399:0.0227023026477 4522:0.024608801

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

贝叶斯决策理论：

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

其中， $\mathbf{x}$  表示特征， $c_j$  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：

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

如果对特征  $\mathbf{x}$  作类条件独立假设，即

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

得到朴素贝叶斯分类器：

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

## 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><index1>:<value1><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。

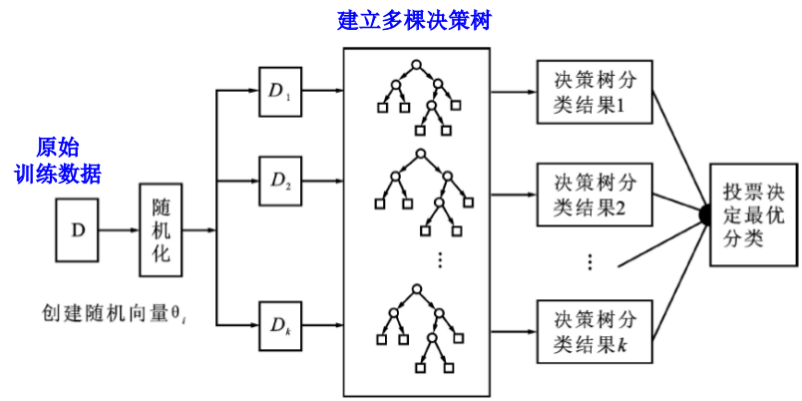


图 6.1 随机森林的分类过程

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 F1 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	$C_j$	$C_j'$
YES	$a_j$	$b_j$
NO	$c_j$	$d_j$

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

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

$$P_j = \frac{a_j}{a_j + b_j}$$

召回率是正确判断为类别  $C_j$  的文本数与类别  $C_j$  中所有样本总数之比:

$$R_j = \frac{a_j}{a_j + c_j}$$

## 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:

$$F1_j = \frac{2 * P_j * R_j}{P_j + R_j}$$

## 2.3 宏平均

上文所述的正确率、召回率及 F1 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:

$$P_{macro} = \frac{\sum_{i=1}^c P_i}{c}$$

$$R_{macro} = \frac{\sum_{i=1}^c R_i}{c}$$

$$F1_{macro} = \frac{2 * P_{macro} * R_{macro}}{P_{macro} + R_{macro}}$$

## 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]//Proceedings Research and development in retrieval.ACM,201 3:343-352.
- space[J].arXiv preprint vector on abstracts[J].IBM Journal of research and development,1958,2(2):1 59 • 1 65.
- [2] Maron ME,Kuhns J L.On relevance,probabilistic indexing and retrieval[J].Journal
- [3] Salton Journal information of the ACM(JACM),1960,7(3):216-244. G,Yang C.On the specification of term values in automatic indexing[J].of documentation,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,(01):71 • 75.
- [7] 徐振强,李保利.结合词语分布信息 I 的 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 TFIDF 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