# 介绍

游戏内聊天系统中会出现广告、色情等垃圾信息，给玩家带来一些负面影响，因此需要搭建一个识别这些垃圾信息的过滤系统。本文中，我们使用机器学习算法来实现这一系统。

垃圾信息过滤在机器学习领域中属于“文本分类”问题，下面我们简要地介绍一下本文中解决文本分类问题的主要步骤。

## 1.1 文本分类问题的主要步骤

### （1）数据集的预处理

* 考虑中文数据的编码问题：写入文件时以utf-8编码保存，在程序中处理时则使用Unicode格式。
* 删除多余字符，包括：
  + 空格
  + 标点符号
  + 特殊符号
  + 其他语言字符

使文本中只留下中文、英文以及数字。

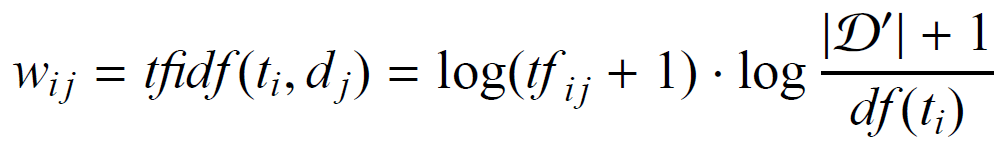
* 对文本进行**分词**，包括：
  + 使用现有的分词程序进行分词（JIEBA分词）——不遗漏单字特征和多字特征，但可能会发生划分错误
  + 使用bigram方法，将文本中每一对连续的字符作为一个“词”——得到所有2字词串，不遗漏所有的2字词特征
  + 将以上两种分词方法得到的词集合并，作为混合词集
* 去除停用词：类似“啊”，“吧”，“的”，“了”的对分类无意义的词

### （2）文本向量化

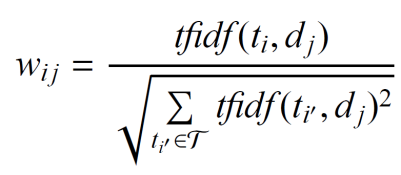
* **特征提取**，包括两种特征：
  + **由筛选过的词集构成的特征词集S1（用于构建向量空间模型）；**

筛选词集的方法有很多，本文采取以下三种：

* 卡方检验（CHI-square）方法
* 信息增益（Information Gain, IG）方法
* 互信息（Mutual Information, MI）方法
  + **根据文本特点定义的其他特征属性集S2；**
* **特征的权重向量表示**（针对S1），本文中采用三种表示方式：
  + 词集模型：即用0或1表示该特征词的出现与否
  + 词袋模型：以特征词在文本样例中出现的次数作为它的权重，注意需要进行规格化处理。
  + TF-IDF模型：以特征词的（词频\*反文档频率）的值作为它的权重



在进行特征向量的权重表示之后，还要进行**权重向量的规格化**，以消除文本长度对特征权重取值的影响。以TF-IDF模型为例：



**代码如下：**

|  |
| --- |
| def word2vec\_tfidf(words,dictionary): # 以TFIDF表示特征权重  voc = dict(zip(dictionary,dictionary))  doc\_dict\_mat = np.zeros((len(words),len(dictionary)), np.float64)  term\_df\_dict = stats\_term\_df(words,voc) #原文本中所有词的文档频率的字典  lenD = len(words)  for i in range(len(words)):  for word in words[i].split():  if word in voc:  tf = words[i].split().count(word)  w = np.log(tf+1) \* np.log((lenD+1) / term\_df\_dict[word])  doc\_dict\_mat[i][dictionary.index(word)] = w  '''权重向量的规格化'''  for i in range(len(words)): #对每一篇文档  regu = 1  for j in range(len(dictionary)): #对该文档中的每一个特征词  regu += doc\_dict\_mat[i][j] \* doc\_dict\_mat[i][j]  regu = math.sqrt(regu)  doc\_dict\_mat[i] = doc\_dict\_mat[i] / regu  del voc,term\_df\_dict  gc.collect()  return doc\_dict\_mat |

### （3）训练分类器

综合考虑模型效果、训练时间等因素，本文在众多机器学习算法中，选择了如下几种算法进行编码实现：

* **贝叶斯算法**
  + Naïve Bayes
  + Semi-supervised Bayes with Expectation Maximization
  + Naïve Bayes with AdaBoost
* **支持向量机**
  + 线性核SVM
  + 高斯核SVM
* \*神经网络

### （4）测试与评价

本文使用“留出验证法”和“5折交叉验证法”对上述分类器进行了多次测试和比较，完整的测试结果见下文。

## 1.2 数据集

本文使用的数据集是在互联网上找到的用于垃圾短消息分类的数据集，共有100万条，其中80万条已标注，20万条未标注。

部分样例如下：

|  |
| --- |
| 1 0 商业秘密的秘密性那是维系其商业价值和垄断地位的前提条件之一  2 1 南口阿玛施新春第一批限量春装到店啦   春暖花开淑女裙、冰蓝色公主衫 气质粉小西装、冰丝女王长半裙、 皇  3 0 带给我们大常州一场壮观的视觉盛宴  4 0 有原因不明的泌尿系统结石等  5 0 23年从盐城拉回来的麻麻的嫁妆  6 0 感到自减肥、跳减肥健美操、  7 1 感谢致电杭州萧山全金釜韩国烧烤店，本店位于金城路xxx号。韩式烧烤等，价格实惠、欢迎惠顾【全金釜韩国烧烤店】  8 0 这款UVe智能杀菌机器人是扫地机的最佳伴侣  9 1 一次价值xxx元王牌项目；可充值xxx元店内项目卡一张；可以参与V动好生活百分百抽奖机会一次！预约电话：xxxxxxxxxxx  10 0 此类皮肤特别容易招惹粉刺、黑头等  11 0 乌兰察布丰镇市法院成立爱心救助基金  12 1 (长期诚信在本市作各类资格职称（以及印 /章、牌、 ……等。祥：x x x x x x x x x x x 李伟%  13 1 《依林美容》三．八．女人节倾情大放送活动开始啦！！！！超值套餐等你拿，活动时间x月x日一x月xx日， 详情进店咨询。美丽热线x  14 0 品牌墙/文化墙设计参考  15 0 苏州和无锡两地警方成功破获了一起劫持女车主的案件  16 0 自然之友苏州小组今日下午按原计划举办小组读书活动暨“我为城市量体温”启动仪式  17 0 共查扣违法三轮车304辆、残疾证23本  18 0 扬州公司工作人员正在扬州瓜洲镇境内  19 0 而我不愿意将就~今天调研团的小伙伴们来到了丁莲芳生产基地  20 0 任何职业的entrylevel都是最简单枯燥的重复  21 1 红都百货x楼婷美专柜x.x节活动火热进行中。一年仅一次的最大活动力度！充值送：充xxx送xxxxxxx送xxxxxxx送xxxxxxx送xxxxxxxx送xxxx时间：x.xx-x.x日。欢迎各位美女们前来选购！ |

由上述样例可知，此数据集虽然数量级很大，且为很难得的中文二分类标注数据集，但其内容稂莠不齐，很多被标记为“ham”的短消息并不能构成一个完整的、语义通顺的句子，而仅仅是由一些短语拼凑而成。因而，此数据集的质量并不是很高。

不过，本文也只是使用它们进行算法正确性的验证，若某一算法能在这样的数据集上面取得较好的分类效果，说明该算法的代码实现没有问题，且很适合于垃圾信息的过滤，接下来便可以用于真正的游戏聊天数据了。因此，能有这样的标注数据集来训练模型，已经很好了。

# 特征选择

在本实验中，每一个短消息文本被表示为(,y)，其中，是由n个特征的权重值构成的特征权重向量，而y{spam，ham} 则是该文本的标签。

进一步地，特征权重向量 由两个分量 和 组成。 表示由文本中的特征词集S1以一定的权重计算方式计算得到的权重值构成的向量，这种向量构成方式又称为“**向量空间模型**”；而 则是由根据文本特点定义的其他特征属性集S2经过规格化之后构成的向量。

最终，由每个文本的特征向量构成的特征矩阵如下所示：

特征词集S1的权重向量

特征属性S2的权重向量

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 特征1-1 | 特征1-2 | …… | 特征2-1 | 特征2-2 | …… |
| 文本1 | 0.0 | 0.065 | …… | 0.0 | 0.113 | …… |
| 文本2 | …… | …… | …… | …… | …… | …… |
| 文本3 | …… | …… | …… | …… | …… | …… |
| …… | …… | …… | …… | …… | …… | …… |
| 文本n | …… | …… | …… | …… | …… | …… |

下面，我们分别对 和 进行详细的介绍。

## 2.1 特征集S1及权重向量（向量空间模型）

要构建文本的向量空间模型，分为以下几步：（1）将文本进行分词（用分词程序或使用bigram方法），分词后的结果作为原始特征集；（2）使用特征提取算法从原始特征集中提取出k个最有价值的特征，作为最终的特征词集S1，也就是俗称的“字典”;（3）使用不同的方法将字典里每个特征在每个文本中对应的权重值计算出来，构成以权重表示的特征向量（所有文本的特征向量一起构成了特征矩阵）；（4）将以权重表示的特征向量进行规格化处理。

### 文本分词

作者在本实验中使用两种方法对文本进行分词：

1. 使用现有的、较为可靠的分词程序进行分词（JIEBA分词）。这样做的好处是不遗漏在语义上独立的单字特征和多字特征，但缺点是，由于分词程序的局限性，可能会发生划分错误，使本该连在一起的汉字分开，或使应该分开的汉字连在了一起；
2. 使用bigram方法，将文本中每一对连续的汉字作为一个“词”。这种分词方法可以得到所有的2字词串，因此不会遗漏所有可能对分类有价值的2字词特征；
3. 为了获得较为完整和全面的初始特征集，作者又将以上两种分词方法得到的词集合并，作为混合初始特征集。

在实验中，我们分别使用这三种原始特征集进行测试，以确定最优的文本分词方式。

### 特征提取

本实验使用了三种特征提取方法从原始特征集中筛选出前k个最具有价值的特征，这些特征组成特征集S1，也即我们所说的“字典”。

这三种方法为：（1）卡方检验法；（2）信息增益法；（3）互信息法。下面进行详细介绍：

* **卡方检验（Chi-square）法**

卡方检验最基本的思想就是**通过观察实际值与理论值的偏差来确定原假设（某特征与类别无关）的正确与否。**

具体做的时候常常先假设两个变量是**独立的**（“原假设”），然后观察实际值（观察值）与理论值（这个理论值是指“如果两者确实独立”的情况下应该有的值）的偏差程度，如果偏差足够小，我们就认为误差是很自然的样本误差，是测量手段不够精确导致或者偶然发生的，两者确确实实是独立的，此时就接受原假设；如果偏差大到一定程度，使得这样的误差不太可能是偶然产生或者测量不精确所致，我们就认为两者实际上是相关的，即否定原假设，而接受备择假设。

现在我们设理论值为E，实际值为x，那么偏差程度的计算公式为：

这个式子就是卡方检验使用的差值衡量公式。当提供了数个样本的观察值x1，x2，……xi，……xn之后，代入到式中就可以求得偏差程度的值，用这个值与事先设定的阈值比较，如果大于阈值（即偏差很大），就认为原假设不成立，反之则认为原假设成立。

在文本分类的特征选择阶段，一般使用“词t与类别c不相关（即它们独立）”来做原假设，计算出的偏差程度值越大，说明距原假设的偏离越大，我们越倾向于认为原假设的反面情况是正确的，即词t与类别c相关。因此，**卡方检验法中，特征选择的过程即是为每个词计算它与类别c的卡方值，从大到小排个序（此时值越大越相关），取前k个作为最优特征。**

例如，N篇文档，分类有体育和非体育，考察特征词“篮球”与类别“体育”的相关性。首先，我们假设体育类与特征词“篮球”无关。

|  |  |  |  |
| --- | --- | --- | --- |
| 特征选择 | 属于“体育” | 不属于“体育” | 总计 |
| 包含“篮球” | A | B | A+B |
| 不包含“篮球” | C | D | C+D |
| 总数 | A+C | B+D | N |

**注意，这里的ABCD指的是特征词出现的文档数目，而不是出现的总次数。经测试，使用总次数进行计算，精确率和召回率都仅有85%左右，远低于使用文档数目的正确率，故此处使用文档数目表示。**

根据原假设（体育类与“篮球”无关），“体育”类别中包含“篮球”的文档比例应与所有文档中包含“篮球”的文档比例相同。故A（体育类中包含的“篮球”的文档数）的理论值应为：

http://img.my.csdn.net/uploads/201211/07/1352272953_7306.gif

差值：

http://img.my.csdn.net/uploads/201211/07/1352273080_7298.gif

同样计算剩下三种情况的差值D12，D21，D22。最后计算“篮球”与“体育”类文章的卡方值：

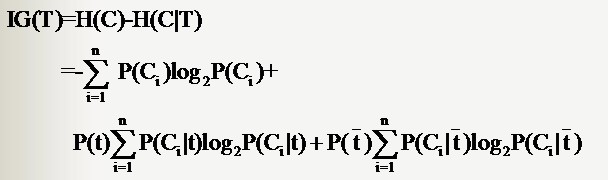
http://img.my.csdn.net/uploads/201211/07/1352273293_1882.gif

进一步化简，注意如果给定了一个文档集合（例如我们的训练集）和一个类别，则N， A+C和B+D对同一类别文档中的所有词来说都是一样的，故实际计算的时候我们可使用如下化简的式子：

http://img.my.csdn.net/uploads/201211/07/1352273407_8936.gif

即：

* **信息增益（Information Gain, IG）法**



**信息增益就是考虑所有特征时（包括特征t）系统的信息熵与给定特征t（即固定t，也就是不再考虑它）时系统的信息熵的差值。 又即，从不包括特征t到包括特征t，对于我们预知C提供了多少信息。（从信息论的角度）**

**若H（C）= H（C|T），则说明类别C与特征t无关。**

信息增益在机器学习算法中是一种用来选择特征的指标，信息增益越大，则这个特征对于分类的贡献就越大。

对一个特征而言，分类系统有它和没它时信息熵将发生变化，而前后信息熵的差值就是这个特征给系统带来的信息增益。它的计算公式如下：

**信息增益 = 信息熵 – 条件熵**

也即：

http://images0.cnblogs.com/blog2015/605905/201506/162013009355725.png

下面详细介绍这个公式是如何得到的。

1. 信息量

我们知道信息量有如下性质：

* 越小概率的事件发生了，产生的信息量越大。因此一个具体事件的信息量应该是随着其发生概率而递减的，且不能为负；
* 如果我们有俩个不相关的事件x和y，那么我们观察到的俩个事件同时发生时获得的信息应该等于观察到的事件各自发生时获得的信息之和，即：h(x,y) = h(x) + h(y)， 由于x，y是两个不相关的事件，那么满足p(x,y) = p(x)\*p(y). 根据上面推导，我们很容易看出h(x)一定与p(x)的对数有关

因此我们有信息量公式如下：

1. 信息熵

信息量度量的是一个具体事件发生了所带来的信息，而信息熵则是在结果出来之前对可能产生的信息量的期望——考虑该随机变量的所有可能取值，即所有可能发生事件所带来的信息量的期望。即：

1. 条件熵

条件熵是用来解释信息增益而引入的概念。假设对分类系统来说，类别C是随机变量，它可能的取值是C1，C2，……，Cn，而每一个类别出现的概率是P(C1)，P(C2)，……，P(Cn)，那么此时分类系统的信息熵就是：

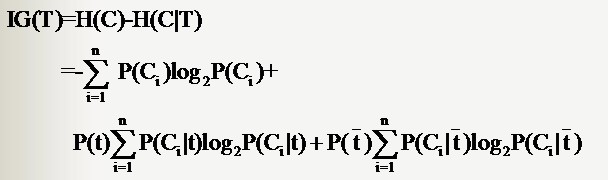
那么，当分类系统不包括某个特征t时，系统的条件熵是多少呢？答案是，将特征t固定住，使t不能变化，计算此时的系统条件熵。（可以这么想：一个班有很多学生，每个学生每天有两种可能情况，出勤或缺勤，因而整个班的出勤情况是非常多的；但是现在有一个学生，他被固定在座位上了/固定在家里，即他的情况一定是出勤/缺勤，那么，我们已经可以确定这个同学的出勤与否了，这个同学的出勤情况就不再影响这个班的出勤情况了。也就是说，固定这个学生的状态，就是在不包括这个学生的情况下，得到班级的信息熵）

而t可以被固定为两个值：出现，或不出现；若我们用T表示该特征出不出现的变量，t表示出现，那么固定特征t的条件熵就为：

该条件熵可以理解为：当分类系统不包括该特征t时，系统的信息熵。

1. 信息增益

由上可知：特征t给系统带来的的信息增益 = 整个系统的信息熵（包括特征t）- 不包括特征t的系统信息熵（固定t后的条件熵）。



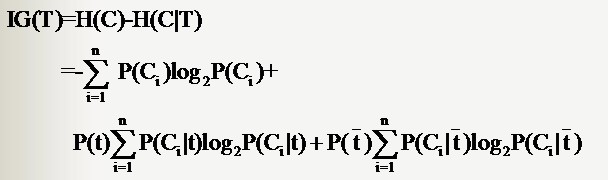
由于对每个特征来说，系统的信息熵都是相同的，因此，在实际计算中，可以将其忽略不计，只对其后的条件熵进行比较。借用Chi-square中关于“A,B,C,D”的定义，可得：

**注意，这里的ABCD指的是特征词出现的总次数，而不是出现的文档数目。经测试，使用总次数进行计算，精确率和召回率都高于使用文档数目的正确率，故此处使用总次数表示。**

信息增益最大的问题在于它只能考察特征对整个系统的贡献，而不能具体到某个类别上，这就使得它只适合用来做所谓“全局”的特征选择（指所有的类都使用相同的特征集合），而无法做“本地”的特征选择（每个类别有自己的特征集合，因为有的词，对这个类别很有区分度，对另一个类别则无足轻重）。 不过对于本项目无影响，因为垃圾邮件的过滤是二分类。

* **高倾向性的互信息法（Mutual Information, MI）法**

在上文中，我们介绍信息增益时，由于对于每个特征来说，系统的信息熵都是相同的，因此在实际计算中，我们将其忽略不计，只对其后的条件熵进行比较。即，信息增益的实际计算式为：



做一下变形：

可以看出，化简后的IG计算式是对所有类别的综合的考虑，而我们接下来定义的互信息法则仅仅针对一个类别：Spam类（公式中用表示）。

可以看出，信息增益即是互信息的加权和。

用上面的A、B、C、D表示的话，即为：

由于仅仅针对一个类别，也就是spam类，因此用这种方式选出来的特征，往往带有非常强的spam类的倾向性，这也导致了使用MI算法的结果准确率非常高，而召回率很低。

### 特征的权重向量表示 & 特征向量的规格化

此部分与章节一中“文本向量化”的第二小节一致，不做赘述。

特征向量的规格化完成后，每一个信息文本便可以表示为一个长度为len(dictionary)的数值向量（即），这些向量便可以用于训练分类模型了。

值得一提的是，若使用**贝叶斯算法**构建模型，到目前为止计算出的向量就足以达到很好的效果了。实验表明，使用向量(+)训练出的贝叶斯模型的泛化性能反而不如使用向量训练出的贝叶斯模型。

## 2.2 特征集S2及权重向量（其他特征属性）

除了利用划分特征词得到的向量空间模型外，我们还可以通过观察目标数据集中不同类别文本的特点来找到其他一些可以很好地代表文本类别特性的特征属性。

在本实验中，作者提出了7条这样的特征属性，期望可以通过它们更好地对spam和ham进行区分。

通过观察数据集，我们可以很容易地发现，垃圾消息中很大一部分都包含有广告、商业宣传信息或游戏代充信息，因而这些消息中往往会含有发布人的联系方式（QQ号、电话号码等）或商品的价格信息等——这就带来了大量的数字串。

同时，发布人为了防止消息被过滤，经常会在数字中间夹杂很多非数字符号，比如字母或标点符号、特殊符号等。对于夹杂的字母，我们将其视为数字串的一部分，对其进行相同的处理；而对于夹杂的其他符号，我们则在预处理时将其删除，使得符号前后的数字串再次连接起来。为了充分利用这些字母/数字串蕴含的信息，我们提出下面两条特征属性：

**（1）每个样本中字母/数字串的个数；**

1. 长度大于等于2的字母数字串的个数
2. 长度大于等于3的字母数字串的个数

**（2）每个样本中字母/数字串的最大长度；**

**注意：**在游戏聊天数据集中，本条特征有较好的区分度，因为在聊天过程中，正常消息很少会包含长串的数字、字母；然而，在测试数据集（80w标注数据集）中，由于其中的ham消息都是从网上随机爬取的文字片段，里面经常有很多英文单词，因而本条特征的效果就不会很好了。

一个特征的TF-IDF值可以很好地代表该特征对于数据集的其中一份文件的重要程度。TF的意思是词频（Term Frequency），IDF的意思是反文档频率（Inverse Document Frequency）。一个特征的重要性随着它在某一文本中出现的的次数成正比增加，但同时会随着它在整个数据集的文本中出现的频率成反比下降，因此，TF-IDF值高的特征往往具有很好的类别区分能力，适合用来分类。

在上文中，我们已经使用TF-IDF作为特征向量的一种表示方式了，但是，为了不浪费TF-IDF的良好特性，在这里，我们再次启用它，将它作为一个单独的特征来为分类系统服务。

区别于前面使用的每个特征的TF-IDF值，为了表达出某一个文档对于分类系统的整体重要性，我们将一个文档中的所有特征词（出现在字典中的）的TF-IDF值相加，作为该文档的对应特征值。即：

**（3）每个样本的TF-IDF值**

在观察数据集时，我们发现，垃圾信息往往会比正常信息的长度更大；这是因为，不同于正常信息有时只需几个字几个词就能结束一句话，垃圾信息需要表达的内容更多（代充、商业宣传），且通常含有一些无意义的字母和符号。

同时，我们还发现，即使普通信息的长度也很长，但在垃圾信息中，生僻字的数量也会比正常信息多很多；因为发送者为了使敏感词过滤系统失效，有时会使用一些非常生僻而又字形相近的汉字或符号来替代原本的内容，这就使得标记为spam的消息其Unicode编码的累加和通常会大于正常信息。于是我们提出了下面两条特征：

**（4）每个样本的文本长度** Document length = #words

**（5）文本中所有字符（除空格）的 unicode 16 进制编码的累加和（或平均值，经检验二者效果近似）**

注：增加第四、五条特征使得分类器的**精确率得到了提高**，但是，由于（5）和（4）综合起来使得分类器趋向于认定较长的文本为垃圾文本，一些较短的垃圾文本往往被放过，所以，反而**降低了召回率**；因此，第四、五条特征可根据需要增减。

不使用（5）时，分类器的精确率和召回率较为均衡，均在90%~95%之间。

在进行文本预处理的时候，我们会将所有非汉字、数字和字母的字符过滤掉，因此，在划分特征词的时候，不会含有特殊符号、其他语言字符、全角数字字符等。但是，这些字符的使用者往往也正是垃圾信息的发送者（发送正常信息的用户一般来说不会专门打出奇怪的语言或符号），因此，我们需要利用好这一部分被删除掉的信息。

除了第五条中累加Unicode编码外，我们还可以通过计算被删掉的字符数量来获取隐含的信息，即：

**（6）被过滤掉的字符长度（一个汉字或全角字符长度为3，半角字符为1）**

例如：

“要橙装橙将宝物进阶石加扣１３６：５０８：３２２。先给uLiNK”

“要橙装橙将宝物进阶石加扣先给uLiNK”

Ans = 36

“五十送50万元宝+VIP+橙装+神将QQ20-183，138-19”

“五十送50万元宝VIP橙装神将QQ2018313819”

Ans = 8

以上7条特征组成了特征集S2，接着，对它们的值进行规格化后形成权重向量。

# 分类器算法

在完成了上述“数据预处理 – 特征提取 – 特征向量向量化”的步骤之后，便可以使用训练数据集的特征向量构成的特征矩阵训练分类器了。

本文选择了5种分类器进行实验，并分别进行了模型的实现与测试。这5种分类器为：（1）朴素贝叶斯；（2）使用AdaBoost算法的朴素贝叶斯；（3）使用EM算法的朴素贝叶斯；（4）线性核SVM；（5）高斯核SVM。

后期，若时间允许，还将进行神经网络分类器的研究。

## 贝叶斯算法

在使用贝叶斯算法构建分类器的过程中，我们仅使用了向量空间模型作为备选特征，即S1

### 朴素贝叶斯

假设消息文本中包含的特征词汇为Wi，垃圾消息为spam(S)，正常消息为ham(H)。 当给定一个消息文本，其中包含的特征词为Wi，要判断该消息是否是垃圾消息，即计算P（S|Wi）这个条件概率。根据Bayes' theorem：

[Bayes' theorem](https://camo.githubusercontent.com/680fa43d97cc667266ec5ec7b3309f96abe4d559/68747470733a2f2f75706c6f61642e77696b696d656469612e6f72672f6d6174682f612f362f652f61366537663863353231646366303138623634383061383936373737336163332e706e67)

其中：

* Pr(S|Wi)： 出现特征词Wi的消息是垃圾消息的条件概率（即后验概率）；
* Pr(S)：训练阶段数据集中垃圾消息的概率，或实际调查的垃圾消息的概率（即先验概率）；
* Pr(Wi|S) 垃圾消息中词汇Wi出现的概率；
* Pr(H) 训练阶段数据集中正常消息的概率，或实际调查的正常消息的概率；
* Pr(Wi|H) 正常消息中特征词Wi出现的概率；

**注：**程序中，通过计算并比较Pr(S|W)和Pr(H|W)的大小，判断给定文本是垃圾邮件还是正常邮件。我们发现Pr(S|W)和Pr(H|W)计算的分母相同，所以我们只需要比较分子即可。

**但存在两个问题：**

1. 当词汇不存在时，Pr(S|Wi) = 0，会造成P=0，无法比较
2. 当Pr(S|Wi)较小时，连乘操作会造成下溢出问题

**解决方案：**

* 计算P(Wi|S)和P(Wi|H)时，将所有词汇初始化出现的次数为1，并将分母初始化为2（或根据样本/实际调查结果调整分母的值）。

# 统计语料库中词汇在S和H中出现的次数

wordsInSpamNum = np.ones(numWords)

wordsInHealthNum = np.ones(numWords)

spamWordsNum = 2.0

healthWordsNum = 2.0

* 计算P(Wi|S)和P(Wi|H)时，对概率取对数

pWordsSpamicity = np.log(wordsInSpamNum / spamWordsNum)

pWordsHealthy = np.log(wordsInHealthNum / healthWordsNum)

所以最终比较的是，P(W1|S)P(W2|S)....P(Wn|S)P(S)和P(W1|H)P(W2|H)... P(Wn|H) P(H)的大小。

ps = sum(testWordsMarkedArray \* pWordsSpamicity) + np.log(pSpam)

ph = sum(testWordsMarkedArray \* pWordsHealthy) + np.log(1 - pSpam)

**代码为：**

|  |
| --- |
| def trainNB(trainMatrix,trainCategory):  numTrainDocs = len(trainMatrix) #the number of training examples  numWords = len(trainMatrix[0]) #the number of words in one email  p\_spam = sum(trainCategory)/float(numTrainDocs)  p0num = ones(numWords); p1num = ones(numWords)  p0total = 2.0; p1total = 2.0  for i in range(numTrainDocs):  if trainCategory[i] == 1:  p1num += trainMatrix[i]  p1total += sum(trainMatrix[i])  else:  p0num += trainMatrix[i]  p0total += sum(trainMatrix[i])  p1vec = np.log(p1num / p1total)  p0vec = np.log(p0num / p0total)  return p0vec,p1vec,p\_spam  def classify(vec2classify, p\_words\_ham, p\_words\_spam, p\_spam):  ps = sum(vec2classify \* p\_words\_spam) + np.log(p\_spam)  ph = sum(vec2classify \* p\_words\_ham) + np.log(1 - p\_spam)  if ps > ph:  return 1,ps,ph  else:  return 0,ps,ph |

### 使用AdaBoost算法的朴素贝叶斯

我们在计算上面提到的ps和ph时，可引入一个调整因子DS，其作用是调整词汇表中某一词汇的“垃圾程度”。

ps = sum(testWordsMarkedArray \* pWordsSpamicity \* DS) + np.log(pSpam)

ph = sum(testWordsMarkedArray \* pWordsHealthy) + np.log(1 - pSpam)

其中DS通过Adaboost算法迭代获取最佳值。原理如下：

设定adaboost循环的次数count

交叉验证随机选择1000个样本

DS初始化为和词汇列表大小相等的全一向量

迭代循环count次：

设定最小分类错误率为inf

对于每一个样本：

在当前DS下对样本分类

如果分类出错：

计算出错的程度，即比较ps和ph的相差alpha

如果样本原本是spam，错分成ham：

DS[样本包含的词汇] = np.abs(DS[样本包含的词汇] + np.exp(alpha) / DS[样本包含的词汇])

如果样本原本是ham，错分成spam：

DS[样本包含的词汇] = DS[样本包含的词汇] - np.exp(alpha) / DS[样本包含的词汇]

计算错误率

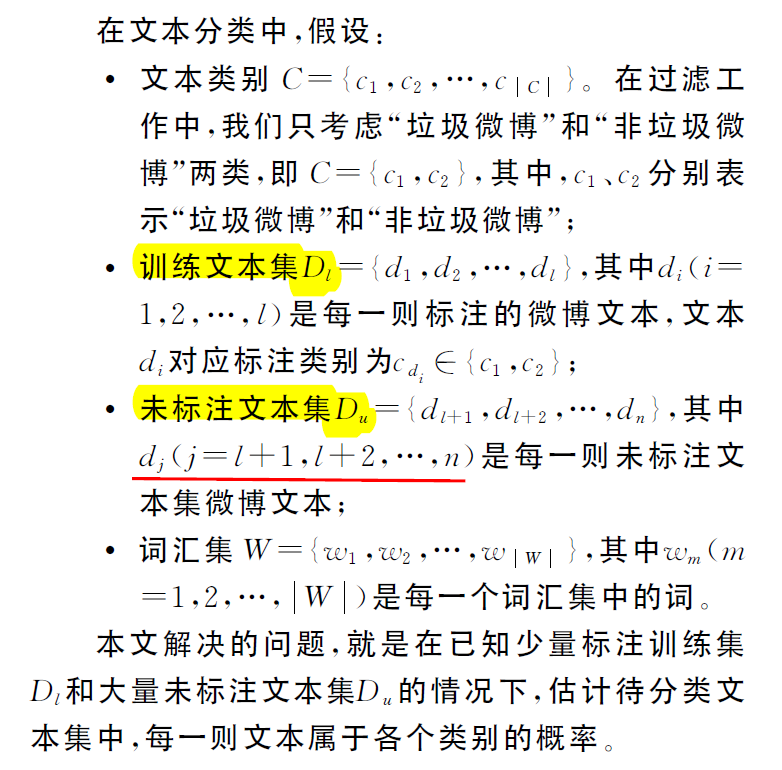
保存最小的错误率和此时的词汇列表、P(Wi|S)和P(Wi|H)、DS等信息，即保存训练好的最佳模型的信息

经检验，该算法的效果并没有达到预期。详细结果见下一节。

### 使用EM算法的半监督贝叶斯

我们真正需要利用的数据集是大量的、未标注的聊天信息数据集，因此，采用传统的有监督垃圾信息过滤方法，会浪费非常多的成本在标注数据上，于是，我们根据调研的论文，实现了一种使用EM算法的半监督贝叶斯分类模型。其原理如下所示：

* 符号定义：



* 算法总体流程：

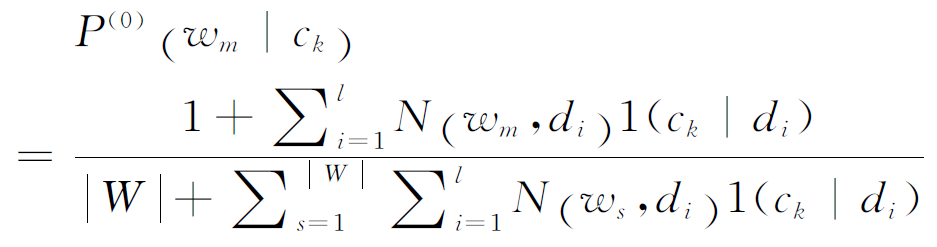


* 算法步骤：

本算法总体上分为3步：（1）训练初始分类器；（2）EM算法中的E-step；（3）EM算法中的M-step。

* + 训练初始分类器

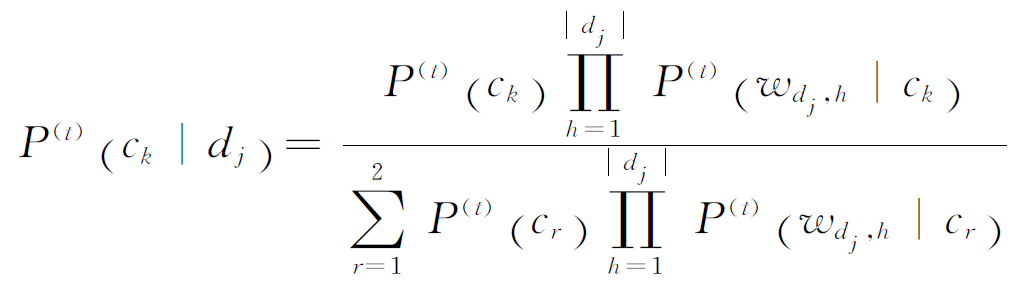
首先，在标注样本下，训练初始的朴素贝叶斯分类器，计算不同类别下出现词的概率参数，记为。



其中，表示文本中出现词的次数。

* + E-step

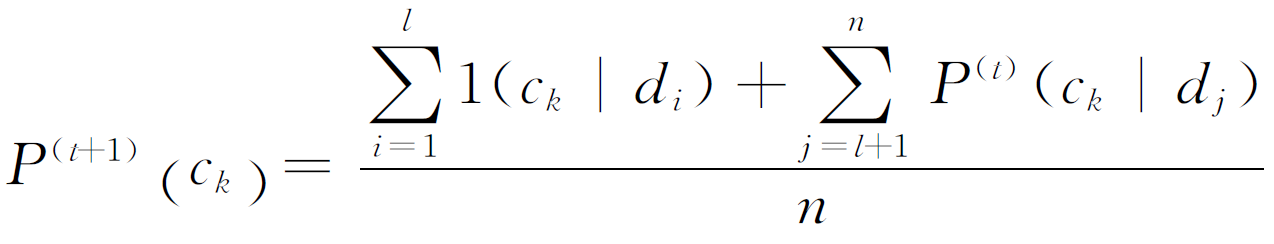
根据此时的，重新计算未标注文本集中每一则文本分别属于spam和ham的概率：和。注意，表示标注文本集，而表示未标注文本集。

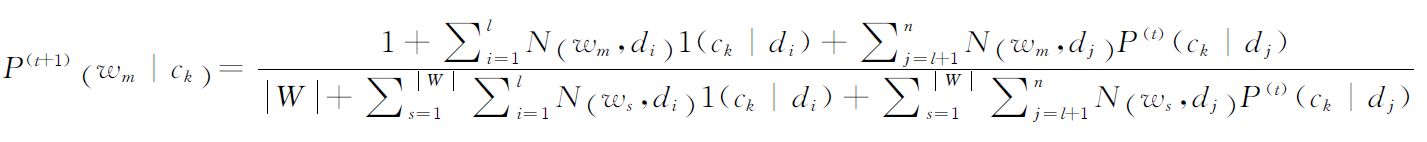


其中，表示文本中第h个位置处的词，|表示文本的词数。

* + M-step

重新估计分类器的参数。

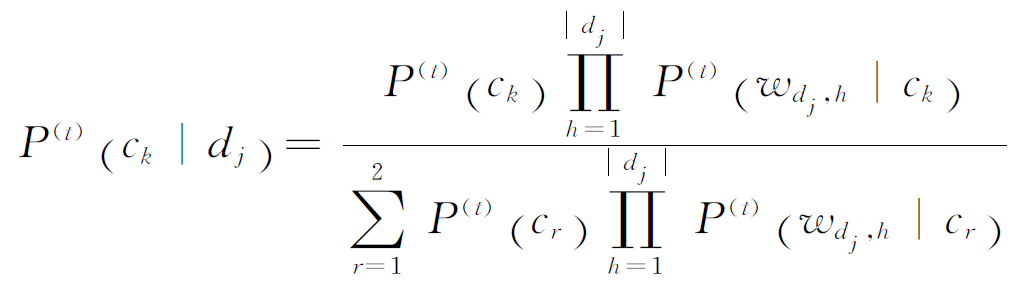




接着，迭代进行E-step和M-step，直至分类器收敛。

* 遇到的问题：

在计算时，我们有计算公式——



然而，若文本中第h个位置处的词（即）从未出现过，不在字典中，那么，该项就会为零，使得分子为零，整个式子也为零；再者，若某文本中的每一个词都不在字典中，那么式子的分子和分母都将为零，整个式子无意义，Python计算出的结果就会为NaN（not a number）。为了解决这个问题，我们提出了三种解决方法：

1. 设min\_pr为一个很小的概率，当遇到某个为零时，就用min\_pr代替0进行计算。但是，若0项很多，多个替代的min\_pr连乘，也会被Python近似约为0，因此，效果不是很好。
2. 使用拉普勒斯平滑算法，分子加一，分母加上常数W，W有三种选择
   * 1. W = 2
     2. W = 数据集大小
     3. W = 字典长度
3. 简单地略过0项

经检验，2.2及2.3的效果最好。具体实验结果见下一节。

## 支持向量机

由于手动实现SVM模型较为复杂，且当前已有很多开源的可靠的SVM库可以调用，故在这里我们直接选用已有的SVM库实现所需的分类器模型。

作者共尝试了两种当前较流行的开源SVM库：sklearn.svm，libsvm。其中，使用libsvm测试的效果非常差，且关于该库的接口使用说明较少，很难对其进行优化；而sklearn.svm库具有详细的官方说明文档，很适合根据实际情况对模型进行优化和修改，故我们选用sklearn.svm进行分类器的实现。

综合考虑分类性能和训练时间，我们选择了两种SVM核作为测试对象：linear核以及高斯核。

最初的测试中，我们仅仅使用默认的参数和默认的特征向量降维算法进行模型的训练，发现效果并不好。于是，作者将linear核SVM和高斯核SVM分别封装成一个类，在类的成员函数中添加了learn\_best\_param的函数，迭代寻找最优参数。同时，将上文中计算得到的特征集S1和S2同时启用，以这些特征代替默认的降维算法找到的特征。经验证，SVM的分类性能得到了很大的提升。

然而，由于训练SVM所需的时间非常长（几个小时，甚至几天），为了快速检测模型的优劣，我们不得不将训练集缩小到5000条/2万条/10万条文本这样的大小。过小的训练集使得分类器的性能一直处于上下波动的状态，既有精确率、召回率几乎100%的情况，也出现了精确率和召回率都只有85%左右的情况，而大多数的测试结果则表现出较高的精确率（95%左右）和稍低的召回率（90%左右）。

在唯一一次以10万数据集进行训练的实验中（花费时间一天多），精确率达到了98%，然而召回率仅有73%。目前，大数据集的测试还需要进一步进行。

为了更好地衡量S2特征集中每个特征对于分类系统的重要程度，我们需要对每个特征进行打分，找到最合适的特征，删去最无用的特征，为此，我们使用以下算法来评价特征们：递归特征消除法。

### 递归特征消除法

递归特征消除法（Recursive Feature Elimination, RFE）的主要思想是反复地构建模型（如SVM或回归模型），然后根据每个特征的系数选出最好的（或者最差的）特征；把选出来的特征放到一边，然后在剩余的特征上重复这个过程，直到所有的特征都遍历了。这个过程中特征被消除的次序就是特征的重要程度排序。

这个方法直接使用了SVM模型中每个特征的权重（也就是系数），非常直观。

### 稳定性选择

稳定性选择的主要思想是：在不同的数据子集和特征子集上运行特征选择算法，不断地重复，最终汇总特征选择结果。比如可以统计某个特征被认为是重要特征的频率（被选为重要特征的次数除以它所在的子集被测试的次数）。理想情况下，重要特征的得分会接近100%，稍微弱一点的特征得分会是非零的数，而最无用的特征得分将会接近于0.

# 测试与评估

下面，我们将针对每一种算法，以及使用的每一种分词法、特征选择方式、权重表示方式分别进行测试与评估。

在测试之前，我们首先提出一些前提：

1. **分类器的评估标准：**

TP：True Positive，即实际为spam且判断为spam的样本的数量；

FN：False Negative，即实际为spam但判断为ham的样本的数量；

FP：False Positive，即实际为ham但判断为spam的样本的数量；

TN：True Negative，即实际为ham且判断为ham的样本的数量。

精确率Precise = TP / (TP + FP)

召回率Recall = TP / (TP + FN)

F1-score = (2\*TP) / (2\*TP + FP + FN)

F1-score越高，说明该分类模型越稳健。

1. 由于实际的数据集尚未标记，因此，我们需要找到一种可减少人工标注成本的半监督算法。有两种途径：
   1. 算法具有非常高的召回率，使得在仅仅标注了少量样本时，便可通过算法找到几乎全部的spam。精确率不高也没关系，因为之后可以再在标注为spam的样本中人工分辨出非spam的样本，这样标注成本也不会很高。
   2. 算法具有非常高的精确率，使得在仅仅标注了少量样本时，便可以以极高的正确率找到部分spam；接着可利用这部分确定的spam以及之前已标记的样本，再次运行算法，找到新的确定的spam……这样迭代多轮，到最后需要人工分辨的模糊样本也不会很多了。
2. 精确率和召回率，哪一个更重要？

①真实的垃圾消息过滤器应有这样的原则：宁可放过，不可错杀。意思是，可以在稍微牺牲召回率的基础上，最大限度地保证精确率。因为在实际生活中，用户偶尔收到几条漏网的垃圾信息并不会有很大损失，但是如果发送的正常信息被系统过滤掉了，便有可能引起用户的不满，甚至损失。

②真实的垃圾消息过滤器应有这样的原则：宁可错杀，不可放过。意思是，可以在稍微牺牲精确率的基础上，最大限度地保证召回率。因为在实际生活中，用户在聊天时可以实时得到消息是否发送成功的反馈，若消息被判为垃圾消息从而被过滤（这里，系统是否应该提示用户？），用户便可得知发送失败，从而重新发送或放弃发送；但如果垃圾消息被误判为正常消息发送到了聊天大厅，便有可能扰乱游戏秩序，影响玩家心情。

## 4.1贝叶斯算法

在对贝叶斯算法的测试中，可改变的参数有如下几种：分词方法、特征选择方法、特征数量、特征向量的权重表示方式、训练/测试数据集大小。

### 4.1.1朴素贝叶斯

**（1）使用100万条数据的数据集**

由于数据集量级较大，训练模型需要较长时间，故仅使用了原始特征集较小的JIEBA分词法，旨在检验本算法在大数据集上的分类性能。测试结果如下，可见，**未进行任何优化且特征数量很少（仅1000条特征）**的贝叶斯分类器在大数据集的情况下便可以达到较好的效果。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 分词  方法 | 特征选  择方法 | 特征  数量 | 特征权重  表示方式 | 精确率 | 召回率 | F1 |
| 结巴 | IG | 1000 | 词袋 | 0.800 | 0.960 | **0.873** |
| 结巴 | IG | 1000 | TFIDF | 0.926 | 0.898 | **0.912** |
| 结巴 | CHI | 1000 | 词集 | 0.879 | 0.931 | **0.904** |

**（2）使用10万条数据的数据集**

下面我们将以不同特征权重表示方式为大类进行分别的测试。

* **词集方式 Set-of-words**

实验数据列表如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 分词  方法 | 特征选  择方法 | 特征  数量 | 精确率 | 召回率 | F1 |
| bigram | CHI | 3000 | 0.877 | 0.928 | **0.902** |
| 混合 | CHI | 3000 | 0.832 | 0.960 | **0.892** |
| 混合 | CHI | 3000 | 0.834 | 0.977 | **0.900** |
| 结巴 | CHI | 3000 | 0.816 | 0.977 | **0.889** |
| 结巴 | IG | 3000 | 0.827 | 0.971 | **0.893** |
| 结巴 | MI | 3000 | 0.998 | 0.491 | **0.658** |
| bigram | CHI | 5000 | 0.851 | 0.962 | **0.903** |
| 结巴 | MI | 10000 | 0.996 | 0.541 | **0.701** |

在此次测试中，CHI和IG方法的性能近似，因此此处仅列出了CHI和MI的结果比较图。

可见，使用词集方式表示特征权重且使用CHI（或IG）法选择特征时，不论使用哪一种分词法，精确率都不是很高，但可达到**较高的召回率**；且在精确率方面，bigram法优于混合法和结巴分词法；在召回率方面，混合法和结巴分词法则优于bigram法。

在使用词集方式表示特征权重且使用MI法选择特征时，可以达到接近100%的精确率，但是召回率却非常低，不到50%。

由上图，5000条特征相对于3000条特征，具有更高的召回率，但精确率略有降低。

而当使用10000条特征、MI特征选择法时，不论使用哪一种分词法，都可以达到非常高的精确率和相当低的召回率。

同时，我们发现，当特征数较少（如3000条）时，MI法仅在使用结巴分词法时表现较好（近1的精确率，近0.5的召回率），而在使用混合和bigram法时表现较差（召回率过低）；而当特征数较多（如10000条）时，这个差别便不那么明显了。

**综上，词集方式的优点是可达到较高的召回率和一般的精确率（IG或CHI时）；**

**而一旦使用MI法，则会得到很高的精确率和很低的召回率。**

* **词袋方式**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 分词  方法 | 特征选  择方法 | 特征数量 | 特征权重  表示方式 | 精确率 | 召回率 | F1 |
| 混合 | CHI | 3000 | 词袋  (未规格化） | 0.718 | 0.973 | **0.826** |
| 结巴分词 | CHI | 3000 | 词袋  (未规格化） | 0.773 | 0.974 | **0.862** |
| bigram | CHI | 3000 | 词袋  (未规格化） | 0.814 | 0.951 | **0.877** |
| 混合 | IG | 3000 | 词袋  (未规格化） | 0.703 | 0.992 | **0.823** |
| 结巴分词 | IG | 3000 | 词袋  (未规格化） | 0.785 | 0.975 | **0.870** |
| bigram | IG | 3000 | 词袋  (未规格化） | 0.749 | 0.984 | **0.850** |
| 混合 | IG | 5000 | 词袋 | 0.923 | 0.914 | **0.918** |
| 混合 | IG | 4000 | 词袋 | 0.922 | 0.924 | **0.923** |
| 混合 | IG | 3500 | 词袋 | 0.934 | 0.925 | **0.929** |
| 混合 | IG | 3000 | 词袋 | 0.927 | 0.891 | **0.909** |
| 混合 | IG | 3500 | 词袋 | 0.933 | 0.903 | **0.918** |
| 混合 | IG | 3500 | 词袋 | 0.925 | 0.902 | **0.913** |
| 结巴分词 | IG | 3500 | 词袋 | 0.974 | 0.854 | **0.910** |
| 结巴分词 | IG | 3500 | 词袋 | 0.960 | 0.857 | **0.905** |
| bigram | IG | 3500 | 词袋 | 0.915 | 0.910 | **0.912** |
| bigram | IG | 3500 | 词袋 | 0.905 | 0.890 | **0.897** |

由上图所示，在使用未规格化的词袋方式作为特征权重表示方式时，CHI和IG的分类效果是非常相近的，但IG的召回率要更高一些，因而效果更好。因此，在后面的测试中，我们以IG作为主要方式。并且，可以得出，**CHI或IG方法在使用未规格化的词袋方式时，可以达到很高的召回率和较低的精确率。**

同时，在这种情况下，三种分词方法的效果也非常接近，很难说明谁更好一些。不过，为了简化操作，我们以混合分词法为代表进行下面的测试。

由上图可见，在以上条件下（IG法和词袋方式）测试时，精确率、召回率和F1值都比较高，且很稳定，说明**IG和词袋搭配可以起到较好且均衡的分类效果。**同时也可以看出，特征数为3500时，分类效果最好。

由上图可见，综合来说，混合分词法的效果最稳健，但结巴分词法的精确率较高一点。

* **TF-IDF方式**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 分词  方法 | 特征选  择方法 | 特征数量 | 特征权重  表示方式 | 精确率 | 召回率 | F1 |
| 混合 | CHI | 3000 | TFIDF | 0.907 | 0.915 | **0.911** |
| bigram | CHI | 3000 | TFIDF | 0.933 | 0.877 | **0.904** |
| 结巴分词 | CHI | 3000 | TFIDF | 0.934 | 0.904 | **0.919** |
| 结巴分词 | MI | 3000 | TFIDF | 0.993 | 0.149 | **0.259** |
| 混合 | IG | 3000 | TFIDF | 0.915 | 0.970 | **0.942** |
| 结巴分词 | IG | 3000 | TFIDF | 0.924 | 0.932 | **0.928** |
| bigram | IG | 3000 | TFIDF | 0.902 | 0.933 | **0.917** |

由上图，可见在同等情况下，IG方式确实比CHI方式得到的泛化性能要好；同时，三种分类方法不相上下，但为了稳定起见，还是选择混合分词法最适合。我们还可以得出结论，即使用TF-IDF方式作为特征权重是在三种权重表示方式中最稳定也是最好的方法，当然，规格化之后的词袋方式也能达到很好的效果。

另外，根据表中数据可知，在使用TF-IDF和MI特征选择方法时，可以达到非常高的精确率和低到无法容忍的召回率。在此，我们提出假设：**TF-IDF法（经过检验，词袋方式也一样）不适合与MI法一起使用（会导致非常非常低的召回率）。**后面的测试结果也验证了这一点。

### 4.1.2使用AdaBoost算法的朴素贝叶斯

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 算法 | 特征选  择方法 | 数据集 | 精确率 | 召回率 | F1 |
| AB\_NB  （40轮迭代） | IG | 10w | 0.766 | 0.724 | **0.745** |
| AB\_NB  （100轮迭代） | IG | 10w | 0.678 | 0.512 | **0.583** |

使用AdaBoost算法的朴素贝叶斯模型的部分测试结果如上所示，可见，此算法的实际效果并不是很好。分析原因如下：

（1）每次分类错误，改变其样本垃圾程度是针对样本中每个词来进行的，这样实际上将无关词的垃圾程度也进行了修改，增大了分类的贝叶斯概率的不准确性;

（2）由于迭代多轮，即使每个样本的垃圾程度收敛到较为准确的地步，也可能出现过拟合的情况，使得测试模型时，其泛化误差非常大（训练集错误率0.06,测试集错误率0.45）

### 4.1.3使用EM算法的半监督贝叶斯

使用10万条数据作为训练数据集。这里的备注1，2，3指的是在解决EM算法中遇到问题时的三种方案。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 分词  方法 | 特征选  择方法 | 特征权重  表示方式 | 备注 | 精确率 | 召回率 | F1 |
| 混合 | MI | 词袋 | 1 | 0.917 | 0.725 | **0.810** |
| 混合 | MI | 词袋 | 2.2 | 1.000 | 0.385 | **0.556** |
| 混合 | MI | 词袋 | 2.3 | 1.000 | 0.385 | **0.556** |
| 混合 | MI | 词袋 | 3 | 0.926 | 0.692 | **0.792** |
| 混合 | MI | 词集 | 2.3 | 0.994 | 0.509 | **0.673** |
| 混合 | MI | 词集 | 2.2 | 0.994 | 0.523 | **0.685** |
| 混合 | MI | 词集 | 2.1 | 0.995 | 0.079 | **0.146** |
| 混合 | MI | 词集 | 1 | 0.942 | 0.492 | **0.647** |
| 混合 | MI | 词集 | 3 | 0.942 | 0.492 | **0.647** |
| 混合 | MI | TFIDF | 2.3 | 1.000 | 0.021 | **0.041** |
| 混合 | MI | TFIDF | 2.2 | 0.979 | 0.029 | **0.056** |
| 混合 | IG | TFIDF | 2.2 | 0.980 | 0.832 | **0.856** |
| 混合 | IG | 词集 | 2.2 | 0.855 | 0.970 | **0.836** |
| 混合 | IG | 词袋 | 2.2 | 0.780 | 0.980 | **0.868** |

由上图，使用MI特征提取法和词袋权重表示方式时，能得到非常高的精确率和很低的召回率。同时，可以发现1方案和3方案的效果较差，而2方案中的2.2和2.3效果较好。（2.1方案效果很差，已经从结果数据中删去了。）

由于1，3，2.1的效果不好，我们以2.2和2.3为主。可见，使用词集模型与使用词袋模型的分类效果类似，但词集模型的召回率要优于词袋模型10个百分点左右。

正如上面的测试结果分析中所提到的，MI法和TF-IDF法的结合会带来非常差的召回效果，在日后的实验中，会避免这种搭配。

在上面的测试中，我们使用的特征选择方法都是MI，这里我们改为IG方法。可见，在使用IG法筛选特征时，可在利用词集和词袋方式表示特征权重时取得**非常高的召回率和稍低的精确率**；但是，当使用TF-IDF方式表示权重时，**结果却是相反的**（多次测试结果反映）。

### 4.1.4总结

综合所有贝叶斯算法的测试结果，我们得到以下结论：

* 三种分词方法的效果近似，混合法略胜一筹，且更稳定；
* 三种特征提取方法中，MI方法始终能够得到近乎100%的精确率和非常低的召回率（但不建议和TF-IDF一起使用）；而IG和CHI效果近似，IG要更好一些。
* （针对CHI和IG）在朴素贝叶斯模型中：
  + 词集方法能够取得很高的召回率和较低的精确率；
  + 未规格化的词袋方法类似词集方法，可以达到很高的召回率和较低的精确率；
  + 规格化的词袋方法以及TF-IDF法得到的结果比较平衡也比较好，一般精确率和召回率都在90%以上，且TF-IDF法要更好一些。
* 使用AdaBoost算法的朴素贝叶斯模型则效果很差。
* 在使用EM算法的半监督贝叶斯模型中：
  + 使用MI特征提取算法时，可以在少量标识数据集的条件下取得近乎100%的精确率和一般在50%左右的召回率（词集方式优于词袋方式，不建议使用TF-IDF方式）；
  + 使用IG特征提取算法时：
    - TF-IDF方式可取得非常高的精确率和较低的召回率；
    - 词袋和词集方式可取得非常高的召回率和较低的精确率。

## 4.2支持向量机

以下的测试结果是在包括所有S2特征的情况下得到的（即以精确率为第一目标）。但是经过测试，若仅用S1特征集进行测试，会使得召回率非常的低，且精确率也不是很高。

1. 5000条数据的训练集
2. 2万条数据的训练集

在此训练集下，精确率的平均值约为0.96，召回率的平均值约为0.90。

1. 10万条数据的训练集

在唯一一次以10万数据集进行训练的实验中（花费时间一天多），精确率达到了98%，然而召回率仅有73%。但根据在较小数据集上测试得到的结果，多次测试后，召回率的平均值应该也在90%附件。