

# 贝叶斯分类器

## 设计思路

由于我们需要得到的是给定一篇文章来判定他属于哪个分类的问题，即  $P(\text{category}|\text{doc})$  但是目前的根据已知的数据集来看更好知道  $P(\text{doc}|\text{category})$  即在一个分类下，一篇文章的概率。由于每个文章由多个词语构成，每个事件相互独立，我们又可以知道  $P(\text{doc}|\text{category}) = P(\text{word1}|\text{category}) * P(\text{word2}|\text{category}) * \dots * P(\text{wordn}|\text{category})$

由贝叶斯公式得知  $P(\text{category}|\text{doc}) = P(\text{category}) * P(\text{doc}|\text{category}) / P(\text{doc})$

$P(\text{category})$  分类的概率，由于分类中的数据个数相同，随机一个文章属于分类的概率相同  $1/20$

$P(\text{doc})$  随机抽取一篇文章的概率也相同，所以问题转换为求  $P(\text{doc}|\text{category})$  概率然后每个分类比大小，最大的即是预测的结果

## 拆分训练数据和测试数据

取20\_newsgroups 中每个分类里前 700 个作为训练数据，后300个作为测试数据

## 文件分词

分析：每种不同的文件中的文本内容粗略的可以按照空格分开，另外可以过滤或者分开特殊字符类似 ! {} ? , 等语气助词和分隔符。这一点在后面测试中发现非常重要，词分的不够好，非常影响测试效果。

通过通用算法 `def get_words(doc_path):` 读取文件，并按照如下拆分规则进行分解 `words = re.split(r'[\~ / , ; \{ ? ! \ " \ $ \% \ ^ \ & \ * \ ( \ ) \ < \ > \ n \ - \ + \ t \ : \ _]', letters)`

过滤：

对单词做处理，首先归一化为小写，防止漏判，另外把空字符或者小于1的词语(一般是无意义的 a l 类似的)过滤。另外设置一组 stopwords，把助词语气词都过滤掉

## 训练数据

遍历所有文件把词语和对应 category 中的数据统计出来：

使用字典 `word_category_data` 来记录，结构类似 `{“love”:{“talk.politics.mideast”: 5, “rec.sport.baseball”: “10”}, “him”:{“talk.politics.mideast”: 1, “rec.autos”:2}}`

使用 `category_count_data` 记录每个分类和分类中包含的所有词的个数

求  $P(\text{doc}|\text{category})$ ，由前面的设计思路知道  $P(\text{doc}|\text{category}) = P(\text{word1}|\text{category}) * P(\text{word2}|\text{category}) * \dots * P(\text{wordn}|\text{category})$

转而求给定一个分类的中每个次的概率

比如 love 这个词在 talk.politics.mideast 的概率，  $5 / \text{talk.politics.mideast}$  中次的总数。需要注意一点的是由于有些次在特定的分类中没有出现，所以我们要对原始概率进行加权，不然会出现0，导致最后结果为0，所以我们给  $5 + 1 / \text{talk.politics.mideast} + 1$  分子分母都加1，由于分母足够大，对结果没有影响。或者给定一个极小值。

得出给定一个文章中每个次在的概率  $p$  然后相乘。

由于数据量大，导致一个词在给定的category 中概率太小，乘机会使得数据更小导致后期float 超出判断为0，我们取  $\log$  来修正。由于每种结果都是同样的  $\log$  操作，所以对比对大小没有影响

$$\log(p_1 * p_2 * p_3 * p_4 * \dots * p_n) = \log(p_1) + \log(p_2) + \log(p_3) + \dots + \log(p_n)$$

从而得到每个文章的相对概率

## 测试数据

拿剩下的30% 数据进行测试。算出特定分类下，判断正确的个数 / 总个数，打出每个分类判断的概率。

## 代码实现

Online Code:

<https://github.com/jackrex/AllLesson/tree/master/L2>

## 测试结果

### 20\_newgroups

comp.sys.mac.hardware prob is : 0.906666666667

talk.politics.misc prob is : 0.77

soc.religion.christian prob is : 0.996632996633

rec.motorcycles prob is : 0.97

sci.med prob is : 0.936666666667

comp.graphics prob is : 0.886666666667

comp.windows.x prob is : 0.91

comp.sys.ibm.pc.hardware prob is : 0.846666666667

talk.politics.guns prob is : 0.9

alt.atheism prob is : 0.796666666667

comp.os.ms-windows.misc prob is : 0.673333333333

sci.space prob is : 0.943333333333  
talk.religion.misc prob is : 0.546666666667  
misc.forsale prob is : 0.836666666667  
rec.sport.hockey prob is : 0.97  
rec.sport.baseball prob is : 0.963333333333  
talk.politics.mideast prob is : 0.943333333333  
rec.autos prob is : 0.913333333333  
sci.electronics prob is : 0.9

整体准确度大概能在90%以上

## 总结

比较影响结果的两个因素：

1. 分词，分词要分的足够细，把次的特征过滤出来

如果 `re.split(r'[ ,;?!@#$%^&*()<>|t\n+|-~\{\}\[\]\|:\\"/_=\\']` 中把前几个, ; 等去掉会导致结果质量直线下降

2. 把不在该分类的单词概率设置的尽量避免干扰正常次概率

3. 设定一定的stopwords 有助于提升概率

4. 使用log 加和的结果是负数，判断大小稍微注意下。

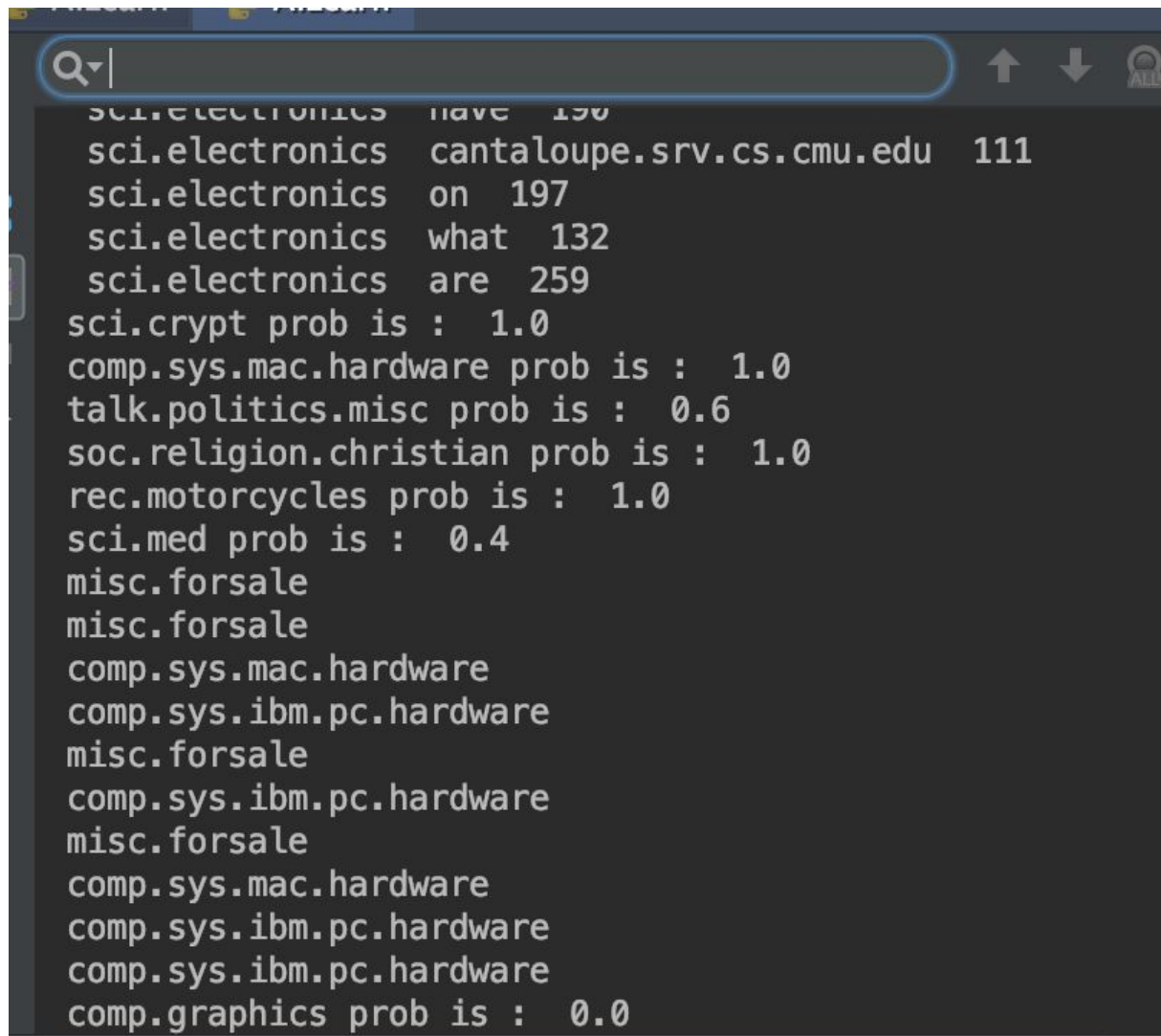
## 记录和优化

第一次测试成绩 Test mini 数据

sci.crypt prob is : 1.0  
comp.sys.mac.hardware prob is : 1.0  
talk.politics.misc prob is : 0.5  
soc.religion.christian prob is : 1.0  
rec.motorcycles prob is : 1.0  
sci.med prob is : 0.4  
**comp.graphics prob is : 0.0**  
comp.windows.x prob is : 0.3  
comp.sys.ibm.pc.hardware prob is : 0.7  
talk.politics.guns prob is : 0.9  
alt.atheism prob is : 0.6  
comp.os.ms-windows.misc prob is : 0.1  
sci.space prob is : 0.6  
talk.religion.misc prob is : 0.3  
misc.forsale prob is : 1.0  
rec.sport.hockey prob is : 0.8

rec.sport.baseball prob is : 1.0  
talk.politics.mideast prob is : 0.9  
rec.autos prob is : 0.7  
sci.electronics prob is : 0.2

看到 comp.graphics prob 概率全部被误判



```
Q|
sci.electronics have 190
sci.electronics cantaloupe.srv.cs.cmu.edu 111
sci.electronics on 197
sci.electronics what 132
sci.electronics are 259
sci.crypt prob is : 1.0
comp.sys.mac.hardware prob is : 1.0
talk.politics.misc prob is : 0.6
soc.religion.christian prob is : 1.0
rec.motorcycles prob is : 1.0
sci.med prob is : 0.4
misc.forsale
misc.forsale
comp.sys.mac.hardware
comp.sys.ibm.pc.hardware
misc.forsale
comp.sys.ibm.pc.hardware
misc.forsale
comp.sys.mac.hardware
comp.sys.ibm.pc.hardware
comp.sys.ibm.pc.hardware
comp.graphics prob is : 0.0
```

打出 **comp.graphics** 这个测试组中任意一个doc 的word 统计, 打出被误判的结果, 以及打出每个分组中前20% 的top 次

comp.graphics has 77  
comp.graphics date 75  
comp.graphics data 126  
comp.graphics image 208  
comp.graphics ftp 69  
comp.graphics also 75  
comp.graphics by 86  
comp.graphics graphics 73

comp.graphics we 52  
comp.graphics can 163  
comp.graphics not 80  
comp.graphics message 75  
comp.graphics your 57  
comp.graphics organization 70  
comp.graphics processing 53  
comp.graphics software 61  
comp.graphics state.edu 60  
comp.graphics subject 72  
comp.graphics id 73  
comp.graphics images 63  
comp.graphics lines 95  
comp.graphics gmt 59  
comp.graphics about 61  
comp.graphics path 71  
comp.graphics would 56  
comp.graphics any 68  
comp.graphics there 70  
comp.graphics all 61  
comp.graphics program 55  
comp.graphics newsgroups 73  
comp.graphics files 53  
comp.graphics comp.graphics 103  
comp.graphics apr 70  
comp.graphics use 57  
comp.graphics which 61  
comp.graphics available 66  
comp.graphics other 65

misc.forsale new 53  
misc.forsale path 71  
misc.forsale misc.forsale 96  
misc.forsale id 71  
misc.forsale date 71  
misc.forsale message 76  
misc.forsale sale 55  
misc.forsale newsgroups 71  
misc.forsale state.edu 75  
misc.forsale organization 69  
misc.forsale apr 74  
misc.forsale lines 71  
misc.forsale subject 73  
misc.forsale gmt 66

smith --2  
sunvax.sun.ac.za --3  
uncc.edu --2  
use --2  
besmith --4  
ve --2  
chessboard --2  
brian --2  
unccsun.uncc.edu --3  
misc.forsale ----  
/Users/jackrex/Desktop/AILesson/L2/mini\_newsgroups/comp.graphics/39048

可以被误判的分类中语气词的部分占大多数，猜想1，语气词被作为了某一分类特征，导致误判，去掉后有一点提升，但是不大。

去掉语气词后发现给定一个文本中，在forsale 占的top 次 和 graphics 都很少，但还是判断错误，检查代码发现之前求一个单词没有在这个分类中使用的是  $\text{basic\_prob} = \text{float}(\text{numerator} + 1) / (\text{denominator} + 1)$ 。由于 denominator 比较大，不在该分类次的概率和出现一次词的概率相似，这是不正确的，于是设定为  $0.000001/\text{denominator}$