4. 朴素贝叶斯与情感分类

分类是人类和机器智能的核心。决定将什么字母、单词或图像呈现给我们的感官，识别面部或声音；分类邮件，分配家庭作业等级， 这些都是将类别分配给输入的示例。寓言家 Jorge Luis Borges(1964)强调了这项任务的潜在挑战，他设想将动物分为以下几种:

(a) 属于皇帝的人，(b) 不朽的人，(c) 受过训练的人，(d) 乳猪，(e) 美人鱼，(f) 神话般的人，(g) 流浪狗，(h) 包括在此类别中的那些，(i) 像疯了似的发抖的人，(j) 数不胜数的人，(k) 用非常细的骆驼毛刷画的人，(l) 其他的，(m) 那些刚刚把花瓶弄坏了的东西，(n) 像远处的苍蝇。

许多语言处理任务都涉及到分类，不过幸运的是，与 Borges 的类相比，我们的类更容易定义。本章将介绍朴素贝叶斯算法并将其应用于**文本分类(text categorization)**，即为整个文本或文档分配**标签（label）**或类别的任务。

我们聚焦于一个常见的文本分类任务**情感分析(sentiment analysis)**、情感提取、作者对某一对象表达的积极或消极倾向。网上对电影、书籍或产品的评论表达了作者对该产品的情感，而社论或政治文本则表达了作者对候选人或政治行动的情感。因此，从市场到政治领域，提取消费者或公众情感都是相关的。

情感分析最简单的版本是一种二进制分类任务，而评论中的单词提供了极好的线索。例如，考虑下面来自对电影和餐馆的正面和负面评论中提取出来的短语。有些词像伟大的、丰富的、令人敬畏的、可悲的、可怕的和荒谬的，是非常有益的线索：

+ ...滑稽的人物和丰富的讽刺作品，以及一些曲折的情节
- 太可悲了。最糟糕的部分是拳击场面。
+ ...很棒的焦糖酱和甜甜的烤杏仁。我喜欢这个地方！
- ...糟糕的披萨和高得离谱的价格...

垃圾邮件检测(spam detection)是另一个重要的商业应用程序，它是二进制分类任务，用于将电子邮件分配给垃圾邮件或非垃圾邮件这两个类中的一个。可以使用许多词汇和其他特征来进行这种分类。例如，你可能会相当合理地怀疑一封包含“在线药品”或“不需要任何费用”或“亲爱的赢家”等短语的电子邮件。

我们可能想知道的另一件事是文本的编写语言。例如，社交媒体上的文本可以是多种语言，我们需要进行不同的处理。因此，**语言 ID(language id)**的任务是大多数语言处理流水线中的第一步。相关的文本分类任务，例如**作者身份归属(authorship attribution)**(确定文本的作者)，也与数字人文科学、社会科学和法医语言学有关。

最后，文本分类中最古老的任务之一是为文本分配图书馆主题类别或主题标签。确定研究论文涉及流行病学还是胚胎学，是信息检索的重要组成部分。存在各种类别的主题集，例如 MeSH(医学主题词)同义词典。实际上，正如我们将要看到的那样，主题类别分类是朴素贝叶斯算法于 1961 年发明的任务。

分类对于文档级别以下的任务也很重要。我们已经看到了句点消歧(确定句点是句子的结尾还是单词的一部分)和单词符记(确定字符是否应该是单词边界)。甚至语言建模也可以看作是一种分类:每个单词都可以看作一个类：每个下一个单词都被认为是一个类，因此，预测下一个单词就是，将到目前为止的上下文(context-so-far)，分类到每个后续单词(each next word)的类别中。词类**标记器(tagger)**(第八章)就是将一个词在句子中的每一个出现归类为名词或动词。

分类的目的是进行单个观察，提取一些有用的特征，从而将这个观察分类到一组离散类集合中的一个。分类文本的一种方法是使用手写规则。在语言处理的许多领域中，基于规则的手写分类器构成了最先进(SOTA)的系统，或者至少是其中的一部分。

然而，规则可能是脆弱的，因为情况或数据会随着时间的推移而变化，而且对于某些任务，人类不一定擅长提出规则。语言处理中的大多数分类都是通过**监督机器学习(supervised machine learning)**完成的，这将是本章剩下部分的主题。在监督学习中，我们有一组输入观察数据，每一个都与一些正确的输出(“监督信号”)相关联。该算法的目标是学习如何从一个新的观察映射到一个正确的输出。

形式上，监督分类的任务是获取一个输入 x 和一组固定的输出类别 Y=(y_1，y_2，...，y_M)并返回预测的类别 y∈Y。对于文本分类，我们有时会谈论 c(用于“类”)代替 y 作为我们的输出变量，而 d(用于“文档”)代替 x 作为我们的输入变量。在有监督的情况下，我们有 N 个文档的训练集，每个文档都手工标记了一个类：(d_1，c_1)，....，(d_N，c_N)。我们的目标是学习一个能够将新文档 d 映射到其正确类别 c∈C 的分类器。概率分类器还会告诉我们观察结果在类中的概率。在类上的全部分配可以为下游决策提供有用的信息。合并系统时，避免早期做出离散决策会很有用。

许多机器学习算法被用来建立分类器。本章介绍了朴素贝叶斯;下面介绍逻辑回归。这些例子说明了两种分类的方法。像朴素贝叶斯这样的**生成式分类器(Generative classifiers)**构建了一个类如何生成输入数据的模型。给定一个观察结果，它们返回最有可能生成该观察结果的类。而像逻辑回归这样的**区分式分类器(Discriminative classifiers)**则会从输入中了解哪些特征对区分不同可能的类最有用。而区分系统通常更准确，因此更常用，生成式分类器仍然起作用。

4.1.朴素贝叶斯分类器

在本节中，我们介绍多项式朴素贝叶斯分类器，之所以这么称呼是因为它是贝叶斯分类器，它对要素之间的相互作用进行了简化(朴素)假设。

分类器的直觉如图 4.1 所示。我们将文本文档表示为一个词袋，即一个无序单词的集合，其位置被忽略，仅在文档中保持其出现频率。在该图的示例中，我们没有在“I love this movie” 和“I would recommend it”之类的所有短语中表示单词顺序，而是简单地注意到在整个摘录中，单词 I 出现了 5 次，it 出现 6 次，love、recommend 和 movie 各出现 1 次，依此类推。

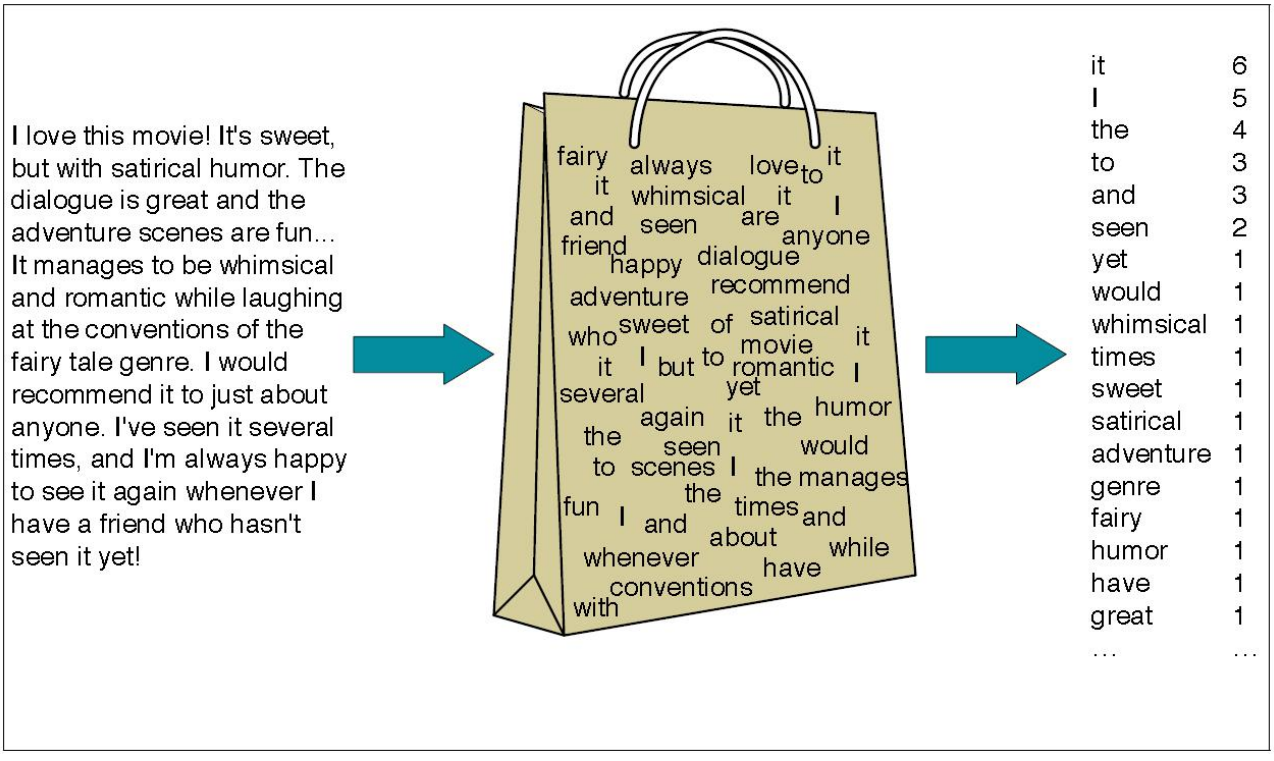

图 4-1：多项式朴素贝叶斯分类器的直观感受(词袋假设)

图注：应用于电影评论的多项式朴素贝叶斯分类器的直观感受，单词的位置被忽略(词袋假设)，我们利用每个单词的频率。

朴素贝叶斯是一个概率分类器，这意味着对于文档 d，在所有类别 c∈C 中，分类器返回具有给定文档最大后验概率的类别 $\hat{c}$。在公式 4.1 中，我们使用帽子符号ˆ来表示“我们对正确类别的估计”。

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) \tag{4.1}$$

贝叶斯推理(Bayesian inference)的这一思想在贝叶斯(1763)的著作中就已经为人所知，并由

Mosteller 和 Wallace(1964)首次应用于文本分类。贝叶斯分类的直觉是使用贝叶斯规则将公式 4.1 转换成其他具有一些有用性质的概率。贝叶斯规则如公式 4.2 所示;它为我们提供了一种将任何条件概率 P(x|y)分解为其他三种概率的方法:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} \tag{4.2}$$

我们可以将公式 4.2 代入公式 4.1 得到公式 4.3:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} \frac{P(d|c)P(c)}{P(d)} \tag{4.3}$$

我们可以通过去掉分母 P(d)来简化公式 4.3。这是可能的，因为我们将为每个可能的类计算 P(d|c)P(c)/P(d)。但是 P(d)不会因为每个类而改变;

我们总是关心同一文档 d 的最有可能的类，这个类必须具有相同的概率 P(d)。

因此，我们可以选择使这个简单公式最大化的类:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} P(d|c)P(c) \tag{4.4}$$

我们将朴素贝叶斯称为生成模型，因为我们可以将公式 4.4 解读为一种关于文档如何生成的隐含假设:首先从 P(c)中采样一个类，然后从 P(d|c)中采样生成单词。(事实上，我们可以想象通过这个过程生成人工文档，或者至少生成它们的单词计数)。我们将在第 5 章更多地讨论生成模型的本质。

回到分类话题：给定某些文档 d，我们通过选择一个类来计算最可能的类$\hat{c}$，即$\hat{c}$具有以下两个概率的最高乘积：文档的**似然度(likelihood)**P(d|c)和类的**先验概率(prior probability)**P(c)：

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(d|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}} \tag{4.5}$$

不失一般性，我们可以将文档 d 表示为一组特征 f_1，f_2，...，f_n：

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(f_1, f_2, \ldots, f_n|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}} \tag{4.6}$$

不幸的是， 公式 4.6 仍然很难直接计算：如果不做一些简化的假设，则估计特征的每种可能组合(例如，每个可能的单词和位置的集合)的概率将需要大量的参数和不可能的庞大训练集。因此，朴素贝叶斯分类器制订了(make)两个简化的假设。

第一个假设是上文直观讨论的词袋假设:我们假设位置无关紧要，并且对于单词“love”，无论出现在文档中的第 1 个、第 20 个还是最后一个单词，都对分类有相同的影响。因此我们假设特征 f_1，f_2，...，f_n 只编码词的身份，不编码词的位置。

第二个假设通常被称为**朴素贝叶斯假设(naive Bayes assumption)**:这是一个条件独立假设，即概率 $P(f_i|c)$在给定类 c 的情况下是独立的，因此可以朴素地作如下乘法:

$$P(f_1, f_2, \ldots, f_n|c) = P(f_1|c) \cdot P(f_2|c) \cdot \ldots \cdot P(f_n|c) \tag{4.7}$$

因此，朴素的贝叶斯分类器为该类选择的最终方程为：

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{f \in F} P(f|c) \tag{4.8}$$

要将朴素的贝叶斯分类器应用于文本，我们需要通过简单地遍历文档中每个单词位置的索引来考虑单词位置：

$$\begin{aligned} \text{positions} &\leftarrow \text{all word positions in test document} \\ c_{NB} &= \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in positions} P(w_i|c) \end{aligned} \tag{4.9}$$

朴素贝叶斯的计算，就像语言建模的计算，是在对数空间中完成的，以避免下溢和提高速度。因此公式 4.9 通常被表示为：

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} \log P(c) + \sum_{i \in positions} \log P(w_i|c) \tag{4.10}$$

通过考虑对数空间中的特征，公式 4.10 把预测类的计算当作输入特征的线性函数。使用输入的线性组合来做出分类决策的分类器(如朴素贝叶斯和逻辑回归)被称为**线性分类器(linear classifiers)**。

4.2. 朴素贝叶斯分类器的训练

我们如何学到概率 P(c)和 $P(f_i|c)$?让我们首先考虑最大似然估计。我们将简单地使用数据中的频率。对于 P(c)之前的类，我们要求(ask)训练集中文档在每个类 c 中的百分比。设 N_c 为训练数据中文档的数量，N_{doc} 为文档的总数。然后:

$$\hat{P}(c) = \frac{N_c}{N_{doc}} \tag{4.11}$$

要了解概率 $P(f_i|c)$，我们假设，一个特征只是文档词袋中一个单词的存在，因此我们需要 $P(w_i|c)$，将其计算为次数的分数，这个次数是指，在主题 c 的所有文档的所有单词中出现单词 w_i 的次数。我们首先将所有具有类别 c 的文档拼接为一个大的“类别 c”文本。然后，我们使用此拼接文档中 w_i 的频率来给出概率的最大似然估计：

$$\hat{P}(w_i|c) = \frac{count(w_i, c)}{\sum_{w \in V} count(w, c)} \tag{4.12}$$

这里的词汇表 V 由所有类中的所有单词类型的联合组成，而不是只有一个 c 类的单词。

然而，最大似然训练存在一个问题。想象一下，我们试图估计给定类别为阳性的单词“fantastic”的似然度，但假设没有既包含单词“fantastic”又被分类为阳性的训练文档。也许“fantastic”这个词碰巧(讽刺地?)出现在阴性类别中。在这种情况下，该特征的概率将为零:

$$\hat{P}(\text{“fantastic”}|\text{positive}) = \frac{count(\text{“fantastic”}, \text{positive})}{\sum_{w \in V} count(w, \text{positive})} = 0 \tag{4.13}$$

但是，由于朴素贝叶斯天真地将所有特征的可能性相乘在一起，任何类的可能性术语中的零概率将导致该类的概率为零，不管其他证据是什么!

最简单的解决方案是第 3 章中介绍的加一(Laplace)平滑。尽管在语言建模中通常用更复杂的平滑算法来代替 Laplace 平滑，但它在朴素贝叶斯文本分类中常被使用：

$$\hat{P}(w_i|c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} (count(w, c) + 1)} = \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c)\right) + |V|} \tag{4.14}$$

再次注意，至关重要的是，词汇表 V 由所有类中的所有单词类型组成，而不仅仅是一个类 c 中的单词(试着说服自己为什么这必须是真的;见本章末尾的练习)。

对于出现在测试数据中但根本不在词汇表中的单词，因为它们在任何类的任何训练文档中都没有出现，我们该怎么办？ 解决此类**未知词(unkown word)**的方法是忽略它们-从测试文档中删除它们，根本不包含任何概率。

最后，某些系统选择完全忽略另一类单词：**停用词(stop word)**，非常常见的词诸如 the 和 a。可以通过按训练集中的频率对词汇进行排序，然后将前 10-100 个词汇条目定义为停用词，或者使用许多可用的在线预定义停用词列表之一来完成。最后将这些停用词的每个实例都从训练和测试文档中删除，就好像它们从未发生过一样。但是，在大多数文本分类应用程序中，使用停用词列表并不能提高性能，因此更常见的是利用整个词汇表而不使用停用词列表。

图 4.2 显示了最终的算法。

function TRAIN NAIVE BAYES(D, C) **returns** log $P(c)$ and log $P(w|c)$

for each class $c \in C$ # Calculate $P(c)$ terms
  N_{doc} = number of documents in D
  N_c = number of documents from D in class c
  $logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$
  $V \leftarrow$ vocabulary of D
  $bigdoc[c] \leftarrow$ **append**(d) **for** d $\in$ D **with** class c
  for each word w in V # Calculate $P(w|c)$ terms
    $count(w,c) \leftarrow$ # of occurrences of w in $bigdoc[c]$
    $loglikelihood[w,c] \leftarrow \log \frac{count(w,c) + 1}{\sum_{w' \text{ in } V} (count(w',c) + 1)}$
return $logprior$, $loglikelihood$, V

function TEST NAIVE BAYES($testdoc$, $logprior$, $loglikelihood$, C, V) **returns** best c

for each class $c \in C$
  $sum[c] \leftarrow logprior[c]$
  for each position i in $testdoc$
    $word \leftarrow testdoc[i]$
    if $word \in V$
      $sum[c] \leftarrow sum[c] + loglikelihood[word,c]$
return $\text{argmax}_c\ sum[c]$

图 4-2：使用加一平滑处理的朴素贝叶斯算法

图注：若要用加 α 平滑来代替，则请把+1 改为+α，以进行训练中的对数似然计数。

4.3. 实例

让我们来看一个使用加一平滑来训练和测试朴素贝叶斯的示例。我们将情感分析域分为正面(+)和负面(-)两个类，并从实际电影评论中简化以下微型训练和测试文档。

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

这两个类的先验 P(c)通过公式 4.11 作为 N_c/N_{doc} 计算:

$$P(-)=\frac{3}{5} \qquad P(+)=\frac{2}{5}$$

单词 with 没有出现在训练集中，所以我们完全放弃它(如上所述，我们不使用未知词的朴素贝叶斯模型)。训练集之中剩下的三个单词“predictable”、“no”和“fun”的似然度，根据公式 4.14(计算训练集中剩余单词的概率留给读者作为练习)，可以得到如下结果：

$$P(\text{“predictable”}|-)=\frac{1+1}{14+20} \qquad P(\text{“predictable”}|+)=\frac{0+1}{9+20}$$

$$P(\text{“no”}|-)=\frac{1+1}{14+20} \qquad P(\text{“no”}|+)=\frac{0+1}{9+20}$$

$$P(\text{“fun”}|-)=\frac{0+1}{14+20} \qquad P(\text{“fun”}|+)=\frac{1+1}{9+20}$$

对于测试句 S = "predicted with no fun"，去掉“with”后，通过公式 4.9，所选类计算如下:

$$P(-)P(S|-) = \frac{3}{5}\times\frac{2\times2\times1}{34^3}=6.1\times10^{-5}$$

$$P(+)P(S|+) = \frac{2}{5}\times\frac{1\times1\times2}{29^3}=3.2\times10^{-5}$$

因此，对于测试句，该模型预测的类是负面的(negative)。

4.4. 情感分析的优化

尽管标准的朴素贝叶斯文本分类可以很好地用于情感分析，但通常会进行一些小的更改以提高性能。首先，对于情感分类和许多其他文本分类任务，一个单词是否出现似乎比其出现频率更重要。因此，通常可以将每个文档中的单词计算限制为 1 来提高性能(有关这些结果，请参见本章结尾)。此变量称为二进制多项式朴素贝叶斯或二进制 NB。该变体使用了相同的公式 4.10，除了对于每个文档，我们在将它们连接到单个大文档之前，我们删除了所有重复的单词。图 4.3 显示了一个示例，其中将四个文档的集合(在此示例中是缩短文档和文本规范化的文档)重新映射为二进制，其修改后的计数如右表所示。该示例在没有加一平滑的情况下工作，以使差异更加清晰。请注意，结果计数不必为 1； 甚至对于二进制 NB，单词 great 的计数也为 2，因为它出现在多个文档中。

对情感进行文本分类时，通常要进行的第二个重要补充是处理否定(negation)。考虑一下“I really like this movie ”(正)和“I didn't like this movie”(负)之间的区别。用"didn't"表示的否定完全改变了我们从谓词"like"得出的推论。同样地，否定可以修饰一个消极的词，产生一个积极的评价(“don't dismiss this film, doesn't let us get bored”)。

在情感分析中通常使用一个非常简单的基线来处理否定语：在文本规范化过程中，在逻辑否定符记(n't, not, no, never)之后的每个单词前加前缀 NOT_，直到下一个标点符号。因此这句话：

didn’t like this movie , but I

变成：

didn’t NOT_like NOT_this NOT_movie , but I

Four original documents:

- − it was pathetic the worst part was the boxing scenes
- − no plot twists or great scenes
- + and satire and great plot twists
- + great scenes great film

After per-document binarization:

- − it was pathetic the worst part boxing scenes
- − no plot twists or great scenes
- + and satire great plot twists
- + great scenes film

	NB Counts		Binary Counts	
	+	−	+	−
and	2	0	1	0
boxing	0	1	0	1
film	1	0	1	0
great	3	1	2	1
it	0	1	0	1
no	0	1	0	1
or	0	1	0	1
part	0	1	0	1
pathetic	0	1	0	1
plot	1	1	1	1
satire	1	0	1	0
scenes	1	2	1	2
the	0	2	0	1
twists	1	1	1	1
was	0	2	0	1
worst	0	1	0	1

图 4-3：二进制朴素贝叶斯算法的二值化示例

这样，新出现的诸如 NOT_like、NOT_recommend 的单词将更多地出现在负面文档中，并成为负面情绪的线索，而诸如“NOT_bored、NOT_dismiss 的单词将获得正面联想。我们将回到第 16 章去使用解析来更精确地处理这些否定词与它们所修饰的谓词之间的范围关系的方法，但是这种简单的基线在实践中效果很好。

最后，在某些情况下，如果使用训练集中的所有单词来估计积极和消极情绪，那么，我们可能没有足够的带有标签的训练数据来训练准确的朴素贝叶斯分类器。

在这种情况下，我们可以取而代之的是从**情感词汇(sentiment lexicons)**中提取正词和负词特征，

这些词表会预先添加正词或负词。在这种情况下，我们可以取而代之的是从情感词汇中提取正词和负词特征，这些词表会预先添加正词或负词。四种流行的词汇是 General Inquirer(Stone 等人，1966)、LIWC(Pennebaker 等人，2007)、Hu 和 Liu 的意见词汇 LIWC(2004a)和 MPQA 的主观词典(Wilson 等人，2005 年)。

例如，MPQA 主观性词汇有 6885 个词，2718 个正词和 4912 个负词，每个词都被标记为是强偏还是弱偏。MPQA 词汇中的一些正和负词汇包括:

+ : *admirable, beautiful, confident, dazzling, ecstatic, favor, glee, great*

- : *awful, bad, bias, catastrophe, cheat, deny, envious, foul, harsh, hate*

在朴素的贝叶斯分类器中使用词汇的一种常见方法是添加一个特征，只要出现该词汇中的单词，该特征就会被计数。因此，我们可以添加一个名为“此词出现在正词汇中”的特征，并将词汇中所有单词的实例都视为该特征的计数，而不是分别计算每个单词。同样，我们可以在负词汇中添加“此词出现在负词汇中”作为第二个特征。如果我们有很多训练数据，并且测试数据与训练数据匹配，那么仅使用两个特征就不能像使用所有单词一样有效。但是，当训练数据稀疏或不能代表测试集时，使用密集词汇特征而不是稀疏的单个单词特征可能会更好地推广。

我们将在第 20 章回到词汇的使用，展示如何自动学习这些词汇，以及如何将它们应用于情感分类之外的许多其他任务。

4.5.朴素贝叶斯用于其他文本分类任务

在上一节中，我们指出朴素贝叶斯并不要求我们的分类器使用训练数据中的所有单词作为特征。事实

上，朴素贝叶斯的特征可以表示我们想要的输入文本的任何性质。

考虑一下垃圾邮件检测任务，即确定某封特定的邮件是否为垃圾邮件(不请自来的批量电子邮件)，以及朴素贝叶斯在文本分类中的第一个应用(Sahami 等人，1998)。

这里的一种常见解决方案是，不要将所有的单词作为单独的特征使用，而是将可能的单词或短语集预先定义为特征，并与非纯语言的特征相结合。例如，开源的 SpamAssassin 工具[1]预先定义了类似短语“百分之百保证(one hundred percent guaranteed)”或“提及数百万美元(mentions millions of dollars)”这样的特性，这是一个正则表达式，与可疑的大笔资金相匹配。但它也包括像“HTML 的文本与图像区域的低比率(HTML has a low ratio of text to image area)”这样的特征，这不是纯粹的语言，可能需要一些复杂的计算，或完全非语言的特征，例如，电子邮件到达的路径。更多 SpamAssassin 特征示例:

•电子邮件主题行均为大写字母
•包含紧急短语，例如“紧急回复”
•电子邮件主题行包含“在线药品”
•HTML 的“ head”标签不平衡
•声明你可以从列表中被删除

对于其他任务，例如语言 ID(确定给定一段文字使用哪种语言)，最有效的朴素贝叶斯特征根本不是单词，而是字符 n-gram，2-gram('zw')，3-gram('nya'，'Vo')或 4-gram('ięz'，'thei')，甚至是更简单的字节 n-gram，在这里，我们不使用称为代码点(codepoints)的多字节 Unicode 字符表示，而是假设所有内容都是原始字节字符串。因为空格可以算作一个字节，所以字节 n-gram 可以对单词的开头或结尾的统计信息建模。一种广泛使用的朴素贝叶斯系统叫 langid.py (Lui 和 Baldwin, 2012)，它是从长度为 1-4 的所有可能的 n-gram 开始，使用特征选择来筛选出信息最丰富的 7000 个最终特征。

语言 ID 系统在多语言文本上进行训练，如维基百科(使用了 68 种不同语言的维基百科文本(Lui 和 Baldwin, 2011))或新闻通讯社。为了确保此多语言文本正确反映不同的地区、方言和社会经济类别，系统还添加了在许多地区进行地理标记的多种语言的 Twitter 文本(对于从英语人口众多的国家--例如尼日利亚或印度--获取世界英语方言非常重要)、圣经(Bible)和古兰经(Quran)的翻译、俚语网站(例如 Urban Dictionary)、非裔美国人白话语料库(Blodgett 等人，2016)、还有其它(Jurgens 等人，2017)。

4.6. 朴素贝叶斯的语言模型

正如我们在前一节中看到的，朴素贝叶斯分类器可以使用任何类型的特征：词典、URL、电子邮件地址、网络特性、短语等等。但是，如前一节所述，如果我们只使用单个单词的特征，并且我们使用文本中的所有单词(不是一个子集)，那么朴素贝叶斯与语言建模有重要的相似之处。具体地说，一个朴素贝叶斯模型可以被看作是一组特定于类的 unigram 语言模型，其中每个类的模型实例化了一个 unigram 语言模型。由于朴素贝叶斯模型的似然特征为每个单词 P(word | c)分配了概率，因此该模型也为每个句子分配了概率：

$$P(s|c) = \prod_{i \in positions} P(w_i|c) \tag{4.15}$$

因此考虑一个朴素贝叶斯模型，其类为正(+)和负(-)，模型参数如下:

w	P(w|+)	P(w|-)
I	0.1	0.2
love	0.1	0.001
this	0.01	0.01
fun	0.05	0.005
film	0.1	0.1
...	...	...

[1] https://spamassassin.apache.org

以上两列都实例化了一个语言模型，该模型可以为“I love this fun film”这句话分配概率:

P(“I love this fun film” | +) = 0.1×0.100×0.01×0.050×0.1 = 0.0000005000

P(“I love this fun film” | −) = 0.2×0.001×0.01×0.005×0.1 = 0.0000000010

碰巧的是，正模型赋予这句话更高的概率:P(s|pos) > P(s|neg)。请注意，这只是朴素贝叶斯模型的似然部分；一旦我们在先验中相乘，完全朴素贝叶斯模型可能会做出不同的分类决定。

4.7.评估：精度，召回率，F 量度

为了介绍评估文本分类的方法，让我们首先考虑一些简单的二进制检测任务。例如，在垃圾邮件检测中，我们的目标是将每个文本标记为属于垃圾邮件类别(“正”)或不属于垃圾邮件类别(“负”)。因此，对于每个项目(电子邮件文档)，我们需要知道我们的系统是否认为它是垃圾邮件。我们还需要知道电子邮件是否真的是垃圾邮件，即我们试图匹配的每个文档的人为定义的标签。我们将把这些人类标签称为**黄金标签(gold labals)**。

假设您是 Delicious Pie 公司的首席执行官，并且您需要了解人们在社交媒体上对您的馅饼(Pie)的评价，因此您可以构建一个系统来检测有关 Delicious Pie 的推文。在这里，正类是关于美味派的推文，负类是所有其他推文。

在这两种情况下，我们都需要一个度量标准来了解我们的垃圾邮件检测器(或 pie-tweet-detector)的工作情况。为了评估任何探测事物的系统，我们首先建立如图 4.4 所示的**混淆矩阵(confusion matrix)**。混淆矩阵是一个可视化表格，该表格展示了算法相对于人类黄金标签是如何执行的。表格使用了二个维度(系统输出和黄金标签)，并且每个单元标记了一组可能的结果。例如，在垃圾邮件检测情况下，真阳性(true positives)是指确实是垃圾邮件的文档(由人工创建的黄金标签指示)，而我们的系统正确地将其称为垃圾邮件。假阴性(False negatives)是指确实是垃圾邮件，但我们的系统错误地标记为非垃圾邮件的文档。

表格的右下角是准确度公式，它要求(ask)我们的系统正确标记的所有观察结果(对于垃圾邮件或 Pie 示例，意味着所有电子邮件或推特)的百分比。虽然准确度似乎是一个自然的度量标准，但我们通常不会将它用于文本分类任务。这是因为当类不平衡时，准确度就不能很好地发挥作用(就像垃圾邮件和推特一样，垃圾邮件是电子邮件的主要内容，而推特主要不是关于 Pie 的内容)。

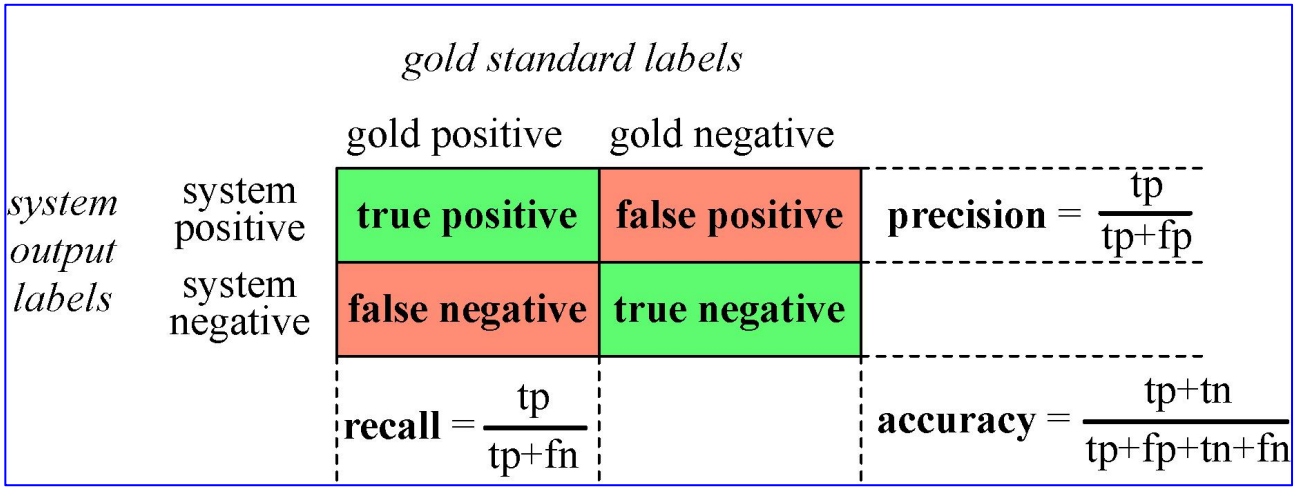

图 4-4：混淆矩阵在黄金标签下的表现

图注：一个混淆矩阵，用于可视化二进制分类系统在黄金标签下的表现。

更明确地说，想象我们看了 100 万条推文，假设其中只有 100 条在讨论他们对我们的 Pie 的爱(或恨)，而其他 999,900 条推文都是关于一些完全无关的事情。想象一下，一个简单的分类器傻傻地把每条推文都分类为“not about pie”。这个分类器将有 999,900 个真否定，只有 100 个假否定，准确度为 999,900/1,000,000 或 99.99%！多么惊人的准确度水平！我们真该对这个分类器感到高兴吗?然而，这个绝妙的“no pie”分类器将是完全无用的，因为它不会找到一个单独的我们正在寻找的客户意见。换句话说，如果目标是发现一些罕见的东西，或者至少在频率上不完全平衡(这是世界上很常见的情况)，那么准确度

就不是一个好的度量标准。

这就是为什么我们通常会转向图 4.4 中的精度和召回率这两个指标而不是准确度。**精度(precision)**衡量的是系统检测到的项目(即标记为阳性的系统)实际上是阳性(即根据人类的金标为阳性)的项目所占的百分比。精度定义为：

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

召回率(Recall)衡量输入中实际存在的，被系统正确识别的项目所占的百分比。召回率定义为：

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

精度和召回率将有助于解决无用的“nothing is pie”分类器的问题。尽管这个分类器的准确率高达 99.99%，但它有一个可怕的召回率为 0(因为没有真阳性结果，并且有 100 个假阴性结果，所以召回率为 0/100)。你应该说服自己，找到相关推文的精度也是有问题的。因此，与准确性不同，精度和召回率强调真阳性：找到我们应该寻找的东西。

有许多方法可以定义包含精度和召回率两个方面的单个度量标准。这些组合中最简单的是 **F-度量(F-measure)**(van Rijsbergen 1975)，定义为:

$$F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

β参数可能会根据应用程序的需求，以不同的方式加权召回率和精度的重要性。β> 1 的值有助于召回率，而β<1 的值有助于精度。当β= 1 时，精度和召回率是相等的； 这是最常用的指标，称为 F_β= 1 或仅称为 F_1：

$$\mathrm{F}_1 = \frac{2PR}{P+R} \tag{4.16}$$

F 度量来自精度和召回率的加权调和均值。一组数的调和均值是倒数算术平均数的倒数:

$$\text{HarmonicMean}(a_1, a_2, a_3, a_4, \ldots, a_n) = \frac{n}{\frac{1}{a_1}+\frac{1}{a_2}+\frac{1}{a_3}+\ldots+\frac{1}{a_n}} \tag{4.17}$$

因此 F 度量是：

$$F = \frac{1}{\alpha\frac{1}{P}+(1-\alpha)\frac{1}{R}} \quad \text{or}\left(\text{with } \beta^2 = \frac{1-\alpha}{\alpha}\right) \quad F = \frac{(\beta^2+1)PR}{\beta^2 P + R} \tag{4.18}$$

使用调和均值是因为它是一个保守度量；两个值的调和均值比算术均值更接近这两个值的最小值。因此，这两个数字中较低的更重。

4.7.1. 用两个以上的类进行评估

到目前为止，我们只描述了两个类的文本分类任务。但语言处理中的许多分类任务都有两个以上的类。对于情感分析，我们通常分为 3 类(积极的、消极的、中性的)，甚至在词类标注、词义消歧、语义角色标签、情绪检测等任务中还会有更多的类。幸运的是，朴素贝叶斯算法已经是一个多类分类算法。

但是，我们需要稍微修改一下精度和召回率的定义。考虑图 4.5 所示的假设 3 种单向电子邮件分类决策(紧急，正常，垃圾邮件)的样本混淆矩阵。矩阵显示，例如，系统错误地将一个垃圾邮件文档标记为紧急文档，并且我们展示了如何为每个类别计算不同的精度和召回率。为了得出一个告诉我们系统运行状况的度量标准，可通过两种方式组合这些值。在**宏平均(macroaveraging)**中，我们计算每个类别的性能，然

后对各个类别进行平均。在**微平均(microaveraging)**中，我们将所有类的决策收集到一个混淆矩阵中，然后从该表中计算精度和召回率。图 4.6 分别显示了每个类的混淆矩阵，及其微平均和宏观平均精度的计算。

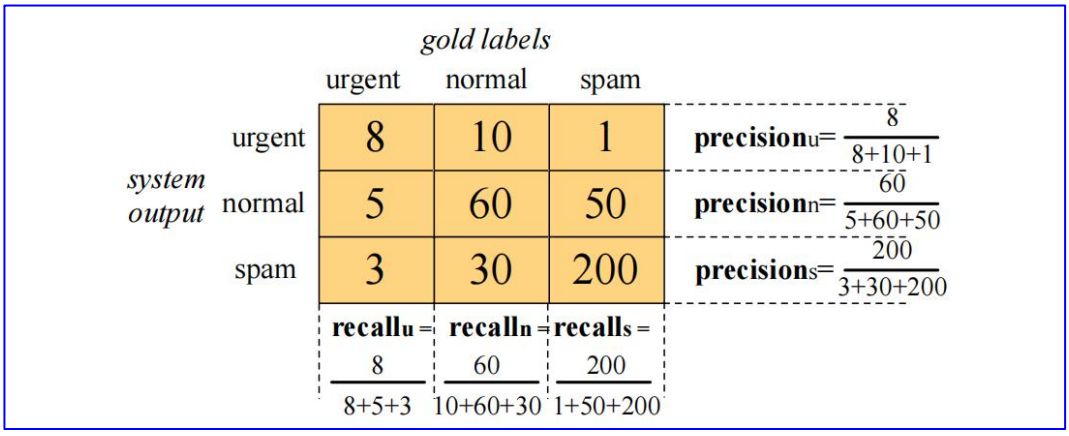

图 4-5：三类分类任务的混淆矩阵

图注：图中显示了对于每一对类(c_1，c_2)，c_1 中有多少文档被正确分配给 c_2。

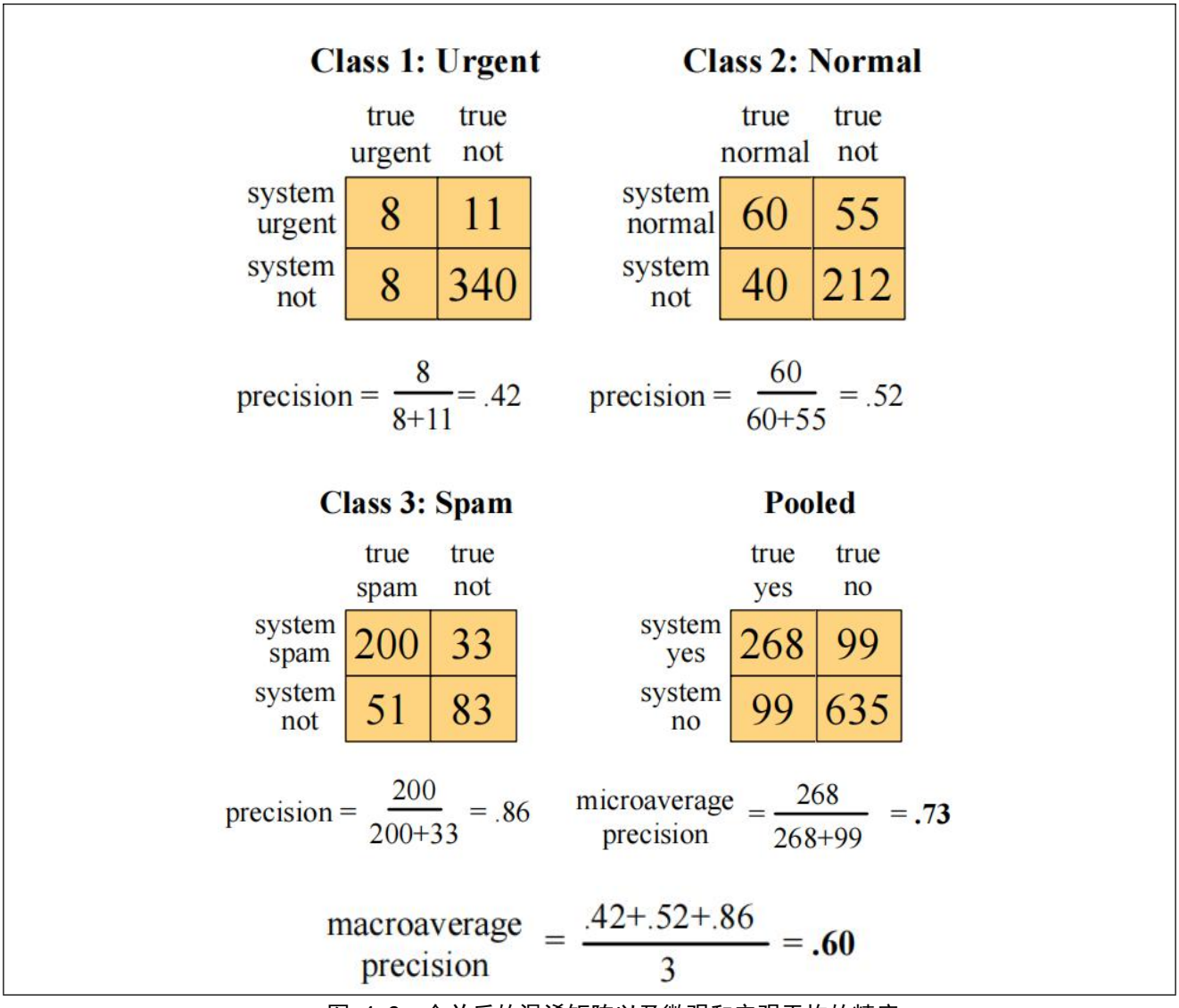

图 4-6：合并后的混淆矩阵以及微观和宏观平均的精度

图注：将 3 类混淆矩阵与上一张图分开，显示了合并后的混淆矩阵以及微观平均和宏观平均的精度。

如图所示，由于计数是合并的，因此微平均值由更频繁的类(在这种情况下为垃圾邮件)所主导。宏平均值更好地反映了较小类的统计信息，因此，当所有类的性能同等重要时，它更合适。

4.8. 测试集和交叉验证

文本分类的训练和测试过程遵循我们在语言建模中看到的内容(第 3.2 节)：我们使用训练集训练模型，然后使用**开发测试集(devset)**来调整一些参数，最后从总体上决定最好的模型是什么。一旦提出了我们认为最好的模型，就可以在(迄今未见的)测试集上运行该模型以报告其性能。

尽管使用开发集可以避免过度拟合测试集，但是具有固定的训练集、开发集和测试集会产生另一个问题：为了保存大量数据进行训练，测试集(或开发集)可能不够大，不具有代表性。如果我们能够以某种方式将所有数据用于训练，而仍将所有数据用于测试，会不会更好呢？我们可以通过**交叉验证(cross-validation)**：我们随机选择一种训练集和测试集的数据切分来训练分类器，然后计算测试集上的错误率，最后重复(随机选择另一种不同的训练集和测试集)。我们做这个采样过程 10 次，并对这 10 次运行进行平均，以得到一个平均错误率。这被称为**十倍交叉验证(10-fold cross-validation)**。

交叉验证的唯一问题是，由于所有数据都用于测试，因此我们要求整个语料库是盲目的。我们无法检查任何数据来暗示可能的特征，并且通常可以看到正在发生的事情，因为我们会偷看测试集，而这种作弊行为会使我们高估系统的性能。但是，查看语料库以了解发生了什么对设计 NLP 系统很重要！该怎么办？因此，通常先创建一个固定的训练集和测试集，然后在训练集内进行十倍交叉验证，但是计算错误率是按照测试集的正常方式计算的，如图 4.7 所示：

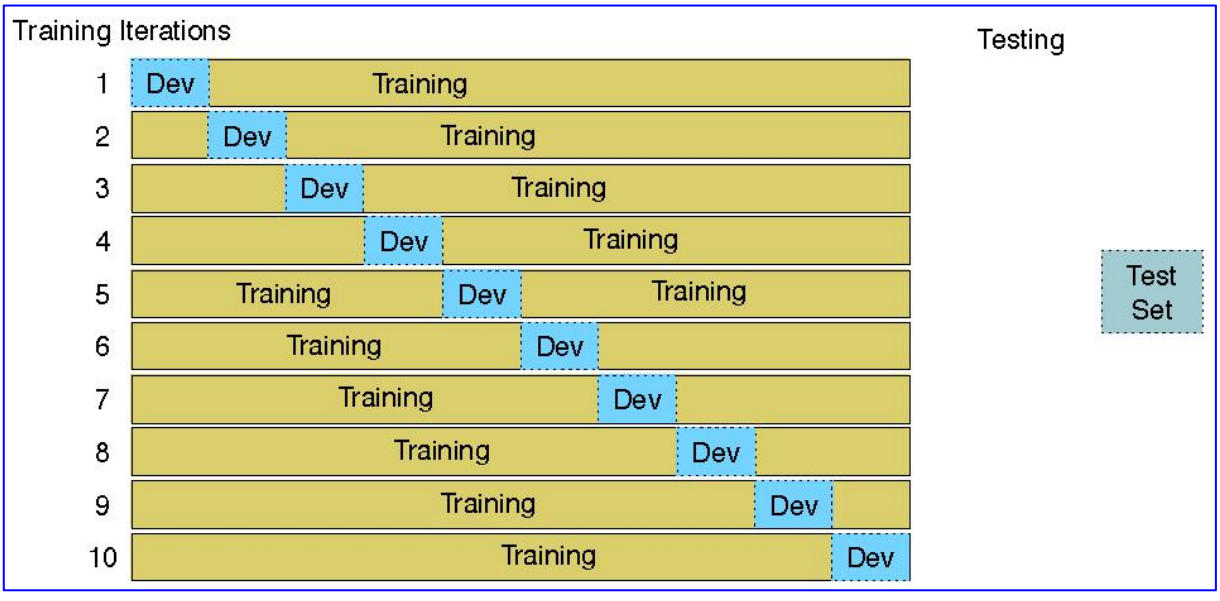

图 4-7：十倍交叉验证

4.9. 统计显著性检验

在构建系统时，我们经常需要比较两个系统的性能。我们如何才能知道我们刚刚建立的新系统是否比我们的旧系统更好?或者比文献中描述的其他系统更好? 这是统计假设检验的领域，在本节中，我们将介绍 NLP 分类器的统计显著性检验，特别是借鉴 Dror 等人(2020)和 Berg-Kirkpatrick 等人(2012)的工作。

假设我们正在比较分类器 A 和 B 在度量 M 上的性能，比如 F_1，或者准确度。也许我们想知道，在特定测试集 x 上，逻辑回归情感分类器 A(第 5 章)是否比朴素贝叶斯情感分类器 B 获得更高的 F_1 分数。我们称 M(A，x)为系统 A 在测试集 x 上获得的得分，而 δ (x)为 A 和 B 在 x 上的性能差异：

$$\delta(x) = M(A,x) - M(B,x) \tag{4.19}$$

我们想知道是否 δ (x)> 0，这意味着逻辑回归分类器在 X 上的 F_1 高于朴素贝叶斯分类器。δ (x)称为**效应量(effect size)**，较大的 δ 表示 A 似乎比 B 好得多；较小的 δ 意味着 A 似乎只稍好一点。

我们为什么不只检查 δ (x)是否为正？假设我们这样做了，我们发现 A 的 F_1 得分比 B 高 0.04。我们可以确定 A 更好吗？我们不可以！这是因为在这个特定的 x 上，A 可能刚好胜过 B。我们还需要更多东西：

我们想知道，如果我们检查另一个测试集 x'或在其他情况下，那么，A 优于 B 的优势是否还会继续存在。

在统计假设检验的范式中，我们通过形式化两个假设来检验这一点。

$$H_0 : \delta(x) \leq 0$$
$$H_1 : \delta(x) > 0 \tag{4.20}$$

假设 H_0，称为**零假设(null hypothesis)**，假设 δ (x)实际上为负或零，这意味着 A 并不比 B 好。我们想知道是否可以自信地排除该假设，而是支持 H_1，即 A 更好。为此，我们创建了一个覆盖所有测试集的随机变量 X。现在我们问，如果零假设 H_0 是正确的，那么在这些测试集中我们会遇到我们发现的 δ (x)值的可能性有多大。我们将这种似然度形式化为 p 值(p-value)：假设零假设 H_0 为真，则可以看到已知的 δ (x)或更大的概率：

$$P(\delta(X) \geq \delta(x) | H_0 \text{ is true}) \tag{4.21}$$

因此，在我们的示例中，该 p 值是，在假设 A 不比 B 好的情况下，我们看到 δ (x)的概率。如果 δ (x)很大(假设在 x 上，A 的 F_1 是非常可观的 0.9，而 B 的 F_1 只是可怕的 0.2)，则我们可能会感到惊讶，因为如果 H_0 实际上为真，则极不可能发生，所以 p 值会很低(如果 A 实际上不比 B 好，则不可能有这么大的 δ)。但是，如果 δ (x)非常小，即使 H_0 为真，而且 A 的确不比 B 好，这对我们来说不足为奇，因此 p 值会更高。

一个非常小的 p 值意味着我们观察到的差异在零假设下是非常不可能的，因此我们可以拒绝零假设。什么才算很小呢?通常使用.05 或.01 这样的值作为阈值。值 0.01 表示，如果 p 值(假设 H_0 为真时我们观察到的 δ 的概率)小于 0.01，那么我们将拒绝原假设，并假设 A 的确优于 B。我们说，如果我们看到的 δ 有低于阈值的概率，那么这个结果(例如，“A 比 B 好”)就是**统计显著(statistically significant)**，因此我们拒绝这个零假设。

我们如何计算 p 值所需的概率？ 在 NLP 中，我们通常不会使用您可能熟悉的简单参数测试(例如 t 检验或 ANOVA)。参数测试对测试统计量的分布(例如正态性)做出假设，而这些假设通常在我们的案例中不成立。因此，在 NLP 中，基于采样，我们通常使用非参数测试：我们人为地创建了许多版本的实验设置。例如，如果我们有很多不同的测试集 x'，则我们就可以测量所有 x'的所有 δ (x')。这给了我们一个概率分布。现在我们设置一个阈值(例如.01)，如果我们在此分布中看到 99%或更多的这些增量小于我们观察到的增量，即 p-value(x)小于.01(p 值是我们现在看见的 δ (x)的概率，它和我们过去看见的一样大)，那么我们可以拒绝原假设，并同意 δ (x)是一个足够令人惊讶差异，而且 A 确实比 B 更好。

有两种常见的非参数检验应用于 NLP：**近似随机(approximate randomization)**和**引导(bootstrap)**检验。我们将在下面描述引导，显示测试的配对版本，这也是 NLP 中最常见的版本。**配对(paired)**测试是我们比较两组对齐的观察值的方法：一组观察值可以与另一组观察值配对。当我们在同一测试集上比较两个系统的性能时，自然会发生这种情况。我们可以将系统 A 在单个观测 x_i 上的性能与系统 B 在同一 x_i 上的性能配对。

4.9.1. 配对引导测试

引导测试(bootstrap test)(Efron 和 Tibshirani，1993)可以适用于任何度量标准， 从精度、召回率或 F_1 到机器翻译中使用的 BLEU 指标。引导一词指的是从原始的较大样本中重复抽取大量较小的样本，并进行替换(称为引导样本)。引导测试的直觉是我们可以通过从观察到的测试集中重复采样来创建许多虚拟测试集。该方法仅假设样本可以代表总体。

考虑一个包含 10 个文档的测试集 x 的小型文本分类示例。图 4.8 的第一行显示了两个分类器(A 和 B)在该测试集中的结果，每个文档被标注为四种可能中的一种:(A 和 B 都对，A 和 B 都错，A 对 B 错，A 错 B 对);字母中的斜杠(/)意味着分类器得到了错误的答案。在第一个文档中，A 和 B 都得到了正确的类(AB)，而在第二个文档中，A 得到了正确的类，但 B 得到了错误的类。为了简单起见，我们假设我们的度量标准是准确度，A 的准确度是 0.70,B 的准确度是 0.50，所以 δ (x)是 0.20。

现在我们创建 b(可能是大数 10^5)个虚拟测试集 $x^{(i)}$，每个大小为 n = 10。图 4.8 给出了几个例子。为

了创建每个虚拟测试集 $x^{(i)}$，我们重复(n = 10 次)从行 x 中选择一个要替换的单元格。例如，要创建第一个虚拟测试集 $x^{(1)}$的第一个单元格，如果我们碰巧随机选择了 x 行的第二个单元格;我们将其值(A 和斜杠 B)复制到我们的新单元格中，并继续创建 $x^{(1)}$的第二个单元格，每次从原始的 x 中采样(随机选择)并进行替换。

	1	2	3	4	5	6	7	8	9	10	A%	B%	$\delta()$
x	AB	A/B	AB	A/B	A/B	A/B	A/B	AB	A/B	A/B	.70	.50	.20
$x^{(1)}$	A/B	AB	A/B	A/B	A/B	A/B	A/B	AB	A/B	AB	.60	.60	.00
$x^{(2)}$	A/B	AB	A/B	A/B	A/B	AB	A/B	A/B	AB	AB	.60	.70	-.10
...													
$x^{(b)}$													

图 4-8：配对的引导测试

图注：从初始真实测试集 x 创建 b 个伪测试集 $x^{(i)}$的示例。通过替换 n = 10 次采样来创建每个伪测试集； 因此，单个样本就是一个单元格，一个带有黄金标签的文档，以及分类器 A 和 B 的正确或不正确的性能。当然，实际的测试集不会只有 10 个示例，并且 b 也必须很大。

现在我们有了 b 个测试集，提供了抽样分布，我们可以对 A 具有偶然优势的频率进行统计。有多种方法可以计算这种优势。在这里，我们遵循 Berg-Kirkpatrick 等人(2012)提出的版本。假设 H_0(A 并不比 B 好)，我们可以预期 $\delta(X)$为零(根据许多测试集估算出的)；更高的值会令人惊讶，因为 H_0 专门假设 A 不比 B 好。为了准确地测量我们观察到的 $\delta(x)$有多令人惊讶，我们将在其他情况下,在 b 个测试集上对出现 $\delta(x^{(i)})-\delta(x)\geq 0$ 的情况进行计数，据此来计算 p-value：

$$\text{p-value}(x)=\sum_{i=1}^{b}\mathbb{1}\left(\delta(x^{(i)})-\delta(x)\geq 0\right)$$

但是，尽管在许多测试集上$\delta(X)$的期望值=0 通常是正确的(再次假设 A 并不优于 B)，但对于我们创建的引导测试集却并非如此。那是因为我们没有从均值为 0 的分布中抽取这些样本；我们碰巧是从原始测试集 x 创建它们的，而恰好是 A 的偏差值(0.20)。因此，为了准确地测量我们观察到的 $\delta(x)$有多令人惊讶，我们在 b 个测试集上对出现 $\delta(x^{(i)})-\delta(x)\geq\delta(x)$的情况进行计数，据此来计算 p-value：

$$\begin{aligned}\text{p-value}(x) &= \sum_{i=1}^{b}\mathbb{1}\left(\delta(x^{(i)})-\delta(x)\geq\delta(x)\right)\\ &= \sum_{i=1}^{b}\mathbb{1}\left(\delta(x^{(i)})\geq 2\delta(x)\right)\end{aligned}\tag{4.22}$$

因此，例如，如果我们有 10,000 个测试集 $x^{(i)}$和阈值 0.01，并且仅在 47 个测试集中，我们发现 $\delta(x^{(i)})\geq 2\delta(x)$，则得到的 p 值=0.0047 小于 0.01，表明 $\delta(x)$确实足够令人惊讶，并且我们可以拒绝零假设，并得出结论 A 优于 B。

完整的引导算法如图4.9所示。它给出了一个测试集x，样本数b，并计算了在b个引导测试集上 $\delta(x^{(i)})>2\delta(x)$的百分比。这个百分比作为一个单边的经验 p 值。

function BOOTSTRAP(test set x, num of samples b) **returns** p-$value(x)$

Calculate $\delta(x)$ # how much better does algorithm A do than B on x
$s = 0$
for i = 1 **to** b **do**
  for j = 1 **to** n **do** # Draw a bootstrap sample $x^{(i)}$ of size n
    Select a member of x at random and add it to $x^{(i)}$
  Calculate $\delta(x^{(i)})$ # how much better does algorithm A do than B on $x^{(i)}$
  $s \leftarrow s + 1$ **if** $\delta(x^{(i)}) > 2\delta(x)$
p-value$(x) \approx \frac{s}{b}$ # on what % of the b samples did algorithm A beat expectations?
return p-value(x) # if very few did, our observed δ is probably not accidental

图 4-9：配对引导算法

图注：Berg-Kirkpatrick 等人(2012)之后的配对引导算法的一个版本。

4.10. 避免分类中的危害

重要的是要避免分类器可能造成的损害，这些损害既存在于朴素的贝叶斯分类器中，也存在于我们将在后面的章节中介绍的其他分类算法中。一类伤害是**代表性伤害(representational harms)**(Crawford 2017, Blodgett 等人 2020)，该类由贬低社会群体的系统造成的伤害，例如通过对他们的负面刻板印象的永恒化。例如，Kiritchenko)和 Mohammad(2018)在相同的句子对上检查了 200 个情感分析系统的性能，除了包含一个常见的非裔美国人的名字(如 Shaniqua)或一个常见的欧裔美国人的名字(如 Stephanie)之外，这些名字都是相同的，这些名字取自第 6 章讨论的 Caliskan 等人(2017)的研究。他们发现，大多数系统会对带有非裔美国人名字的句子赋予较低的情感和更多的负面情绪，这反映并延续了将非裔美国人与负面情绪联系在一起的刻板印象(Popp 等人，2003)。

在其他任务中，分类器可能同时导致代表性损害和其他损害，如审查制度。例如，**毒性检测(toxicity detection)**的重要文本分类任务是检测仇恨言论、虐待、骚扰或其他种类的毒性语言的任务。尽管此类分类器的目标是帮助减少社会危害，但毒性分类器本身可能会造成危害。例如，研究人员表明，一些广泛使用的毒性分类器错误地将其标记为无毒的有毒句子，但仅提及少数群体身份，例如女性(Park 等人，2018)、盲人(Hutchinson 等人，2020)或同性恋者(Dixon 等人，2018)，或者干脆使用非裔美国人白话英语等变种的语言特征(Sap 等人 2019，Davidson 等人 2019)。如果这些假阳性错误被毒性检测系统在没有人为监督的情况下使用，则可能导致这些人群或有关人群对话语的审查。

这些模型问题可能是由训练数据中的偏差或其他问题引起的； 通常，机器学习系统会复制甚至放大其训练数据中的偏差。但是，这些问题也可能是由下列因素导致的：标签(例如，由人类标签的偏差)、所使用的资源(例如词汇，或模型组件诸如预训练的嵌入)、甚至是模型体系结构(例如模型被训练成优化的)。尽管减轻这些偏差(例如，通过仔细考虑训练数据源)是一个重要的研究领域，但我们目前尚无一般解决方案。因此，在引入任何 NLP 模型时，重要的是要研究这些因素并弄清楚它们。一种实现方法是为模型的每个版本发布**模型卡(model card)**(Mitchell 等人，2019)，该卡记录了机器学习模型，其中包含以下信息：

- 训练算法和参数
- 训练数据源、动机和预处理
- 评估数据源、动机和预处理
- 预期的用途和用户
- 跨不同人员背景或其他群体和环境状况的模型性能

4.11. 总结

本章介绍了**朴素贝叶斯**（Naive Bayes）分类模型，并将其应用于**情感分析**的**文本分类**任务。

•许多语言处理任务可以看作是**分类**任务。

•文本分类，将整个文本从一个有限集合中分配一个类，包括**情感分析**、**垃圾邮件检测**、语言识别和作者归属等任务。

•情感分析将文本分类为反映作者对某些对象表达的积极或消极倾向(**情感**)。

•朴素贝叶斯是一种**生成**模型，它制订了(make)单词假设（位置无关）和条件独立假设（给定类别的单词在条件上彼此独立）。

•具有二值化特征的朴素贝叶斯似乎对许多文本分类任务更有效。

•基于**精度**和**召回率**对分类器进行评估。

•使用不同的训练集、开发集和测试集来训练分类器，包括在训练集中使用**交叉验证**。

•应该使用统计显著性检验来确定我们是否可以确信一个分类器版本优于另一个版本。

•分类器的设计者应仔细考虑模型可能造成的危害(包括训练数据和其他组件)，并在**模型卡**中报告模型特色。

4.12. 文献和历史说明

多项式朴素贝叶斯文本分类是由 RAND 公司的 Maron(1961)提出的，用于为期刊摘要分配主题类别。他的模型引入了本文提出的现代形式的大部分特征，将分类任务近似为其中的一个分类任务，并实现了 add-δ 平滑和基于信息的特征选择。

朴素贝叶斯的条件独立假设和贝叶斯文本分析的思想似乎已经多次出现。在 Maron 的论文发表的同一年，Minsky(1961)提出了一个用于视觉和其他人工智能问题的朴素贝叶斯分类器，Mosteller 和 Wallace(1963)也将贝叶斯技术应用于作者归属的文本分类任务中。人们早就知道，Alexander Hamilton, John Jay, 和 James Madison 在 1787-1788 年撰写了匿名发表的联邦党人论文，说服纽约批准美国宪法。然而，尽管这 85 篇论文中有一些显然是出自某个作者之手，但有 12 篇的作者在 Hamilton 和 Madison 之间存在争议。Mosteller 和 Wallace(1963)训练了 Hamilton 写作的贝叶斯概率模型和 Madison 写作的另一个模型，然后计算了每一篇有争议的文章的最大可能作者。Heckerman 等人(1998)首次将朴素贝叶斯应用于垃圾邮件检测。

Metsis 等人(2006)，Pang 等人(2002)，以及 Wang 和 Manning(2012)表明，使用多项式朴素贝叶斯的布尔属性比使用完整计数效果更好。二元多项式朴素贝叶斯有时会与另一种也使用二进制表示的朴素贝叶斯相混淆:多元伯努利朴素贝叶斯。而伯努利变量则估计 P(w|c)为包含术语的文档的部分，并包括一个术语是否不在文档中的概率。McCallum 和 Nigam(1998)以及 Wang 和 Manning(2012)表明，在情感或其他文本任务中，朴素贝叶斯的多元伯努利变异并不像多项算法那样有效。

有各种各样的来源涵盖了多种文本分类任务。情感分析见 Pang 和 Lee (2008)，Liu 和 Zhang(2012)。Stamatatos(2009)调查了作者归属算法。关于语言识别见 Jauhiainen 等人(2018);Jaech 等人(2016)是一个重要的早期神经系统。新闻通讯的索引任务经常被用作文本分类算法的测试用例，基于路透社-21578 新闻通讯的文章集合。参见 Manning 等人(2008)和 Aggarwal 和 Zhai(2012)关于文本分类的论述;一般来说，机器学习教科书中涵盖了分类(Hastie 等人 2001, Witten 和 Frank 2005, Bishop 2006, Murphy 2012)。

计算统计显著性的非参数方法在 MUC 竞赛中首次在 NLP 中使用(Chinchor 等人，1993)，甚至更早在语音识别中使用(Gillick 和 Cox 1989, Bisani 和 Ney 2004)。我们对自举的描述借鉴了 Berg-Kirkpatrick 等人(2012)的描述。最近的工作主要集中在多个测试集和多个度量(Søgaard 等人 2014，Dror 等人 2017)。

特征选择是一种去除不太可能很好概括的特征的方法。特征通常是根据它们对分类决策的信息量来排序的。一个非常常见的度量标准，信息增益，告诉我们这个词的出现给了我们多少位信息来猜测这个类。其他特征选择指标包括 χ^2、逐点互信息和基尼(GINI)指数;参见 Yang 和 Pedersen(1997)的比较，以及 Guyon 和 Elisseeff(2003)对特征选择的介绍。

4.13. 练习

4.1 Assume the following likelihoods for each word being part of a positive or negative movie review, and equal prior probabilities for each class.

	pos	neg
I	0.09	0.16
always	0.07	0.06
like	0.29	0.06
foreign	0.04	0.15
films	0.08	0.11

What class will Naive bayes assign to the sentence “I always like foreign films.”?

4.2 Given the following short movie reviews, each labeled with a genre, either comedy or action:

1. fun, couple, love, love **comedy**
2. fast, furious, shoot **action**
3. couple, fly, fast, fun, fun **comedy**
4. furious, shoot, shoot, fun **action**
5. fly, fast, shoot, love **action**

and a new document D:

fast, couple, shoot, fly

compute the most likely class for D. Assume a naive Bayes classifier and use add-1 smoothing for the likelihoods.

4.3 Train two models, multinomial naive Bayes and binarized naive Bayes, both with add-1 smoothing, on the following document counts for key sentiment words, with positive or negative class assigned as noted.

doc	“good”	“poor”	“great”	(class)
d1.	3	0	3	pos
d2.	0	1	2	pos
d3.	1	3	0	neg
d4.	1	5	2	neg
d5.	0	2	0	neg

Use both naive Bayes models to assign a class (pos or neg) to this sentence:

A good, good plot and great characters, but poor acting.

Recall from page 60 that with naive Bayes text classification, we simply ignore (throw out) any word that never occurred in the training document. (We don’t throw out words that appear in some classes but not others; that’s what add-one smoothing is for.) Do the two models agree or disagree?