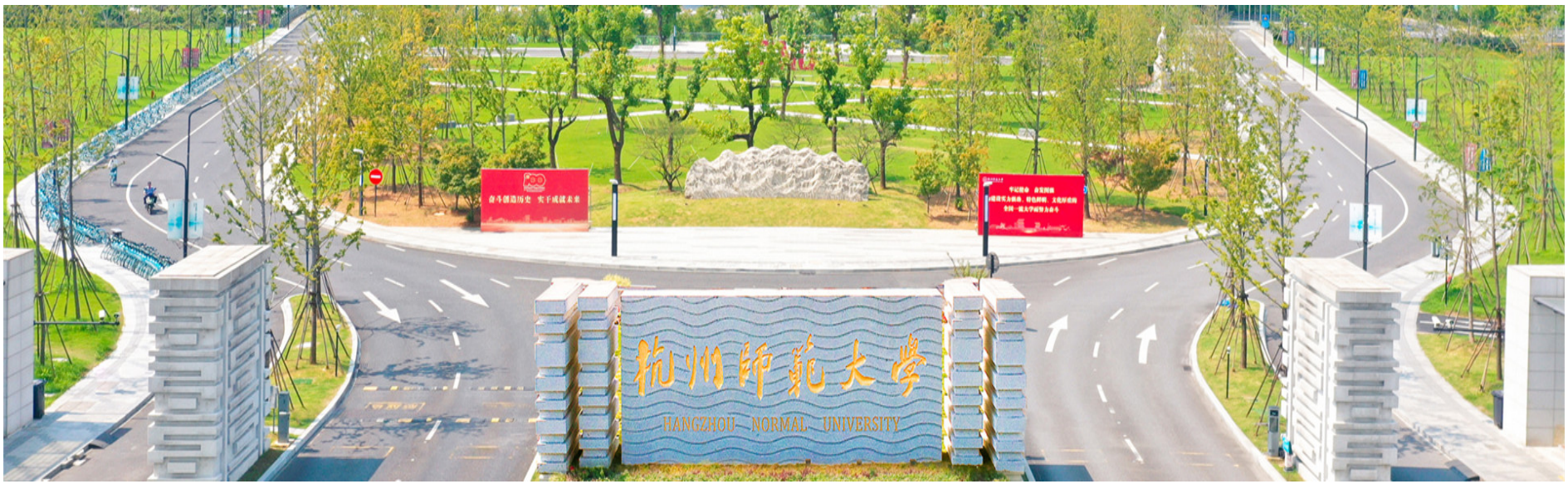
第四讲 - NLP基础算法：朴素贝叶斯

张建章

阿里巴巴商学院
杭州师范大学

2025-02-01



目录

文本分类是NLP乃至整个人工智能领域中的基础核心任务之一。理论上，分类问题也可能非常复杂，实际中，文本分类问题面对的往往是**定义相对明确的类别**。文本分类被广泛应用于情感分析、垃圾邮件检测、语言识别、以及图书主题分类等任务中。

1. 文本分类的基本方法

基于规则的方法：早期的文本分类方法依赖于专家手写的规则，这种方法虽然在某些领域内可见成效，但随着数据和应用场景的变化，其鲁棒性和适应性往往不足。

监督机器学习方法：当前最常用的文本分类方法是基于监督学习的统计方法。通过构建包含大量标注实例（即（文本，类别）对）的训练数据集，模型可以自动学习输入文本与输出类别之间的映射关系。该方法不仅能够给出分类决策，还能在一定程度上计算出每个类别的后验概率，为进一步决策提供支持。

2. 文本分类问题形式化

给定一个输入文本 d 以及预定义的类别集合 $Y = \{y_1, y_2, \ldots, y_M\}$，目标是寻找**一种映射**，使得能够对新的文本 d 正确预测其所属类别 $y \in Y$。这种映射可以用概率模型来描述：

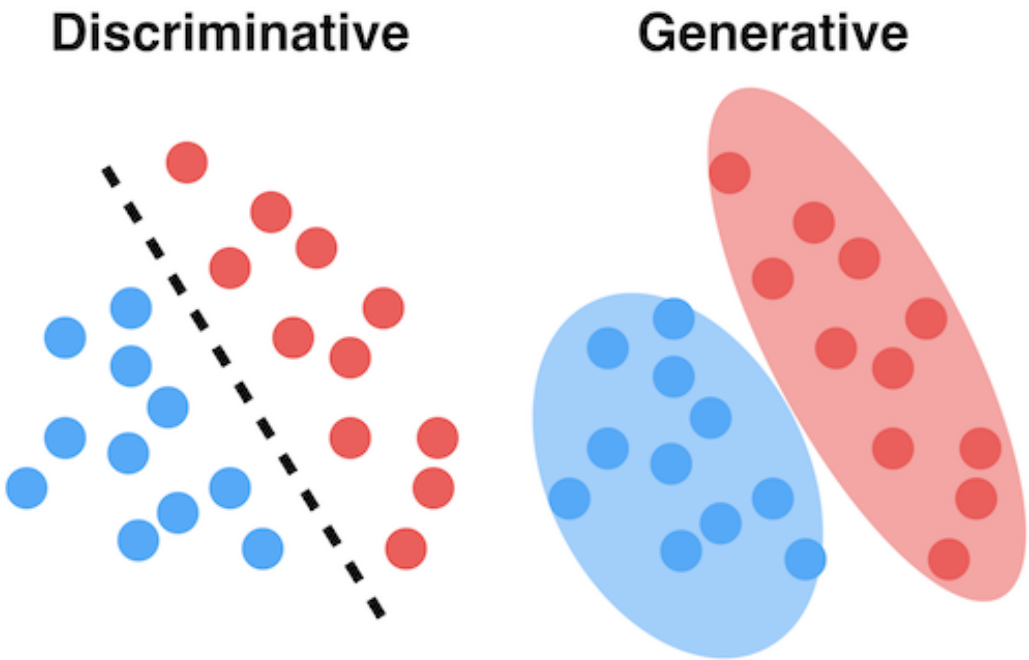
$$\hat{c} = \arg\max_{c \in C} P(c|d)$$

其中 $P(c|d)$ 表示在给定文本 d 的条件下，文本属于类别 c 的后验概率。

3. 监督机器学习方法中的生成模型与判别模型

生成模型：（如，朴素贝叶斯分类器）通过建立每个类别生成文本的概率模型来进行分类，即**假设文本是由某一类别生成的**，然后选择使得生成概率最大的类别。

判别模型：（如，逻辑回归）直接学习输入特征与类别之间的**判别边界**，从而实现分类决策。虽然**判别模型通常在准确率上更为优越**，但生成模型在某些情形下（如数据稀疏或需要生成新数据）仍然具有不可替代的优势。



目录

分类问题的形式化与贝叶斯推断

文本分类问题被形式化为：给定文档 d 以及预先定义的类别集合 C，目标是选择使后验概率 $P(c|d)$ 最大的类别，即

$$\hat{c} = \arg\max_{c \in C} P(c|d)$$

利用贝叶斯定理，后验概率可以分解为

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

由于 $P(d)$ 对所有类别均相同，故分类决策可以简化为

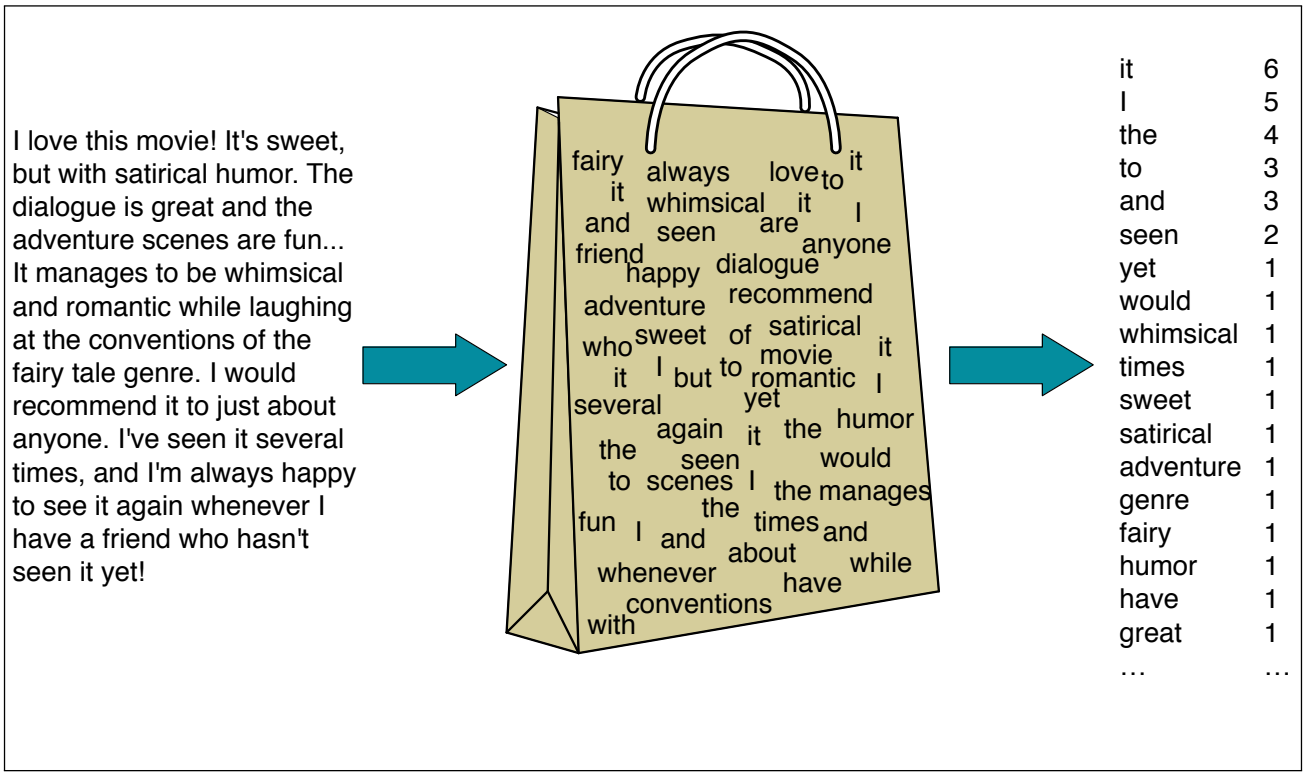
$$\hat{c} = \arg\max_{c \in C} P(d|c)P(c) \tag{1}$$

$P(c)$ 表示类别的先验概率，$P(d|c)$ 则反映了在类别 c 下生成文档 d 的可能性。

生成模型与词袋假设

朴素贝叶斯是一种生成模型，其**假设**文档生成过程为：首先依据先验概率 $P(c)$ 选择一个类别，然后按照类别条件分布 $P(d|c)$ 生成文档。

为了简化问题，采用**词袋模型**（bag-of-words）的表示文档，即忽略词语的顺序，只关注各词在文档中出现的频率。这种表示方式将文档 d 转化为一组特征（通常为各个词的出现次数）。



条件独立性假设

在词袋模型的基础上，朴素贝叶斯进一步引入了**条件独立性假设**：在给定类别 c 的条件下，文档d中各个词（特征）是相互独立的：

$$P(w_1, w_2, \ldots, w_n|c) = \prod_{i=1}^{n} P(w_i|c)$$

这一定理化假设虽然在实际中不完全成立，但极大地简化了参数估计和计算过程，模型在许多文本分类任务中依然能取得较好的效果，将上式带入 (1)式，得到**朴素贝叶斯文本分类决策**：

$$\hat{c} = \arg\max_{c \in C} P(c) \prod_{i=1}^{n} P(w_i|c) \tag{2}$$

对数空间计算与线性分类器的实现

为避免直接计算乘积时可能出现的**数值下溢**问题，同时加快计算速度，常在对数空间内进行计算。此时，朴素贝叶斯分类决策公式可写为：

$$\hat{c} = \arg\max_{c \in C} \left(\log P(c) + \sum_i \log P(w_i|c) \right)$$

这种对数转换使得最终的分类决策转化为输入特征的线性组合，从而使得朴素贝叶斯也被归类为一种**线性分类器**。

朴素贝叶斯分类器的变体

朴素贝叶斯文本分类器做了2个简化假设：词袋模型和条件独立假设。此外，朴素贝叶斯文本分类器对文档生成过程的假设还可以细分为：

- **多项式朴素贝叶斯**：对于选定的类别c，抛掷一个多面体骰子n次，文档中包含n个词，每次生成文档中的一个词，多面体的每个面对应词典中的一个词；

- **多变量伯努利朴素贝叶斯**：对于选定的类别c，抛掷$|V|$次硬币（$|V|$个硬币每个抛掷一次），正面朝上表示该硬币对应的词语出现在文档中。其中，$|V|$表示词典中词语的个数，对应$|V|$个不均匀硬币；

目录

朴素贝叶斯分类器参数估计

利用训练数据来**估计模型中的两个核心概率**：类别先验概率 $P(c)$ 以及条件概率 $P(w|c)$（这里 w 表示文档中出现的词语）。

1. 最大似然估计（Maximum Likelihood Estimation）

类别先验概率的估计：对于每个类别 c，先验概率 $P(c)$ 被定义为该类别在训练数据中出现的比例。设 N_c 表示类别 c 的文档数量，N_{doc} 表示总文档数，则有

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

条件概率的估计：针对每个词 w_i 在类别 c 下的条件概率 $P(w_i|c)$，采用的方法是将所有属于类别 c 的文档“拼接”成一个大文档，然后计算词 w_i 出现的频次在该大文档中所占的比例，即

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

其中，词汇表 V 为所有类别中出现过的词的集合，而非单一类别中的词。

零概率问题与平滑处理

1. 零概率问题

由于训练数据中可能存在某个词在某一类别下从未出现，直接采用最大似然估计会导致对应的 $P(w|c)$ 为零。而朴素贝叶斯分类器在计算文档整体似然时需要将各个词的概率相乘，若其中任一项为零，将导致整个文档的概率为零，从而影响分类结果。

2. 加一平滑（Laplace Smoothing）

为解决上述问题，采用加一平滑方法，即在每个词的计数上加1，得到调整后的估计公式：

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

这种平滑方法能够有效避免零概率现象，确保即使某个词在某个类别中未出现，其概率也不为零。

未知词与停用词的处理

1. 未知词

对于测试数据中出现但在训练数据中完全未出现的词汇，解决办法是将其忽略，不纳入概率计算中。这是因为**这部分词汇无法提供任何关于类别的信息**。

2. 停用词的处理

一些应用可能会选择忽略非常频繁的词（如“the”、“a”等停用词），方法是从训练数据中按照词频选取前10-100个高频词并予以删除。然而，在大多数文本分类应用中，不使用停用词列表（即保留整个词汇表）往往能获得更好的性能。

朴素贝叶斯文本分类器算法流程

function TRAIN NAIVE BAYES(D, C) **returns** V, log $P(c)$, 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 $\text{argmax}_c\ sum[c]$

Figure 4.2 The naive Bayes algorithm, using add-1 smoothing. To use add-α smoothing instead, change the $+1$ to $+\alpha$ for loglikelihood counts in training.

目录

以情感分析为背景，利用一个极简化的训练集和测试集说明**模型参数估计**和**通过概率计算进行分类决策**的过程。

	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

1. 计算类别先验概率

根据训练数据，利用公式

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

计算出负面类别的先验概率 $P(-) = \frac{3}{5}$ 与正面类别的先验概率 $P(+) = \frac{2}{5}$

2. 条件概率的估计与加一平滑

- 为估计每个词在各类别下的条件概率 $P(w_i|c)$，将属于同一类别的所有文档合并成一个“大文档”，计算词频。

- 考虑到某些词在训练数据中可能未出现（例如在正面类中“predictable”和“no”均未出现），直接使用最大似然估计会导致概率为零，从而影响整体文档概率的计算。

- 因此，引入加一平滑方法，修正条件概率的估计公式为

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

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

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

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

3. 测试阶段及分类决策

测试句子为“predictable with no fun”。首先，未知词“with”因未在训练数据中出现被忽略；然后，仅对“predictable”、“no”和“fun”进行概率计算。利用朴素贝叶斯分类公式

$$\hat{c} = \arg\max_{c \in C} P(c) \prod_{w_i \in d} P(w_i|c)$$

分别计算两个类别的联合概率：

$$\begin{aligned} P(-)P(S|-) &= \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5} \\ P(+)P(S|+) &= \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5} \end{aligned}$$

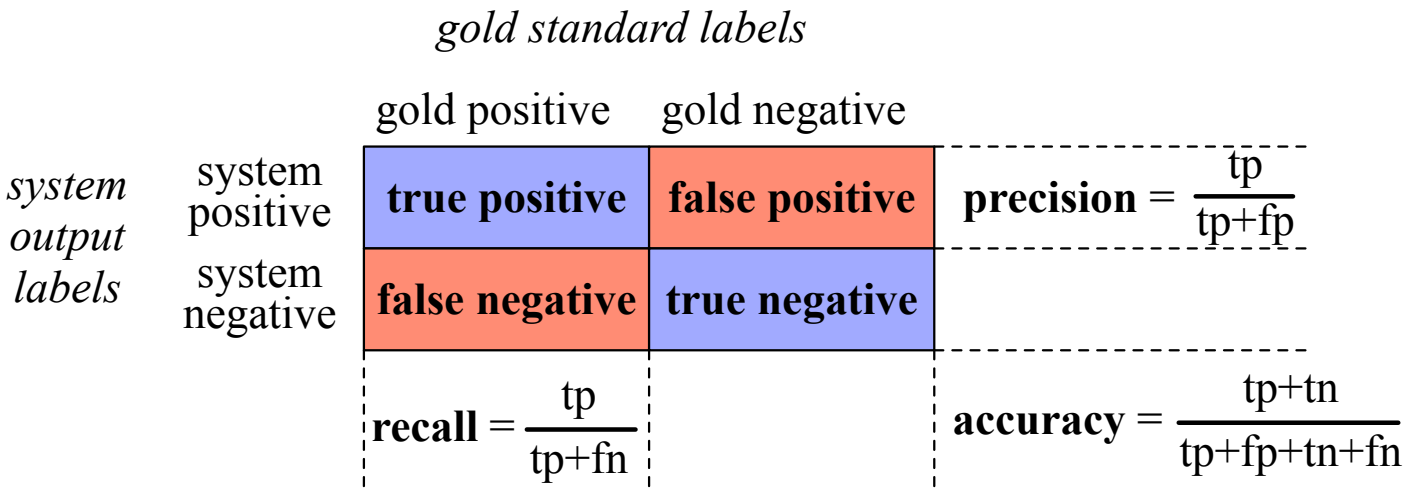
由于负面类别的概率较高，因此模型最终将测试句子判定为负面情感。

目录

混淆矩阵

评估文本分类系统时，通常利用混淆矩阵将系统输出与人类标注的“金标准”标签进行比较。混淆矩阵的四个基本单元包括：

- 真正例（True Positives, TP）：系统正确预测为正例的样本；
- 假正例（False Positives, FP）：系统错误预测为正例的样本；
- 真负例（True Negatives, TN）：系统正确预测为负例的样本；
- 假负例（False Negatives, FN）：系统错误预测为负例的样本；



准确率、精确率与召回率

准确率（Accuracy）：表示所有预测正确的样本比例：

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

精确率（Precision）：表示系统预测为正例的样本中，实际为正例的比例：

$$\text{Precision} = \frac{TP}{TP + FP}$$

召回率（Recall）：表示实际正例中被系统正确检测出来的比例：

$$\text{Recall} = \frac{TP}{TP + FN}$$

注意：在类别分布不均衡的情况下，准确率可能会产生误导性结果，因此更常用精确率和召回率来衡量分类系统**对目标类别的识别效果**。

F值（F-measure）：精确率与召回率的调和平均

为了同时考虑精确率和召回率，引入F值，其基本形式为：

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

其中，参数 β 用于平衡召回率与精确率的重要性；当 $\beta = 1$ 时，即精确率和召回率同等重要，得到常用的F1分数：

$$F_1 = \frac{2PR}{P + R}$$

采用调和平均的原因在于：其更倾向于 (接近)较小的那一项，因此能够有效反映系统在较弱指标上的不足，从而给出一个**较为保守的综合性能评估**。当 $\beta > 1$ 时，更强调召回率，反之，更强调精确率。

多类别情形下的评估方法

扩展到多类别分类：虽然前述讨论主要针对二分类任务，但在实际中，文本分类常涉及多类别问题（如邮件分类中的紧急、正常与垃圾）。

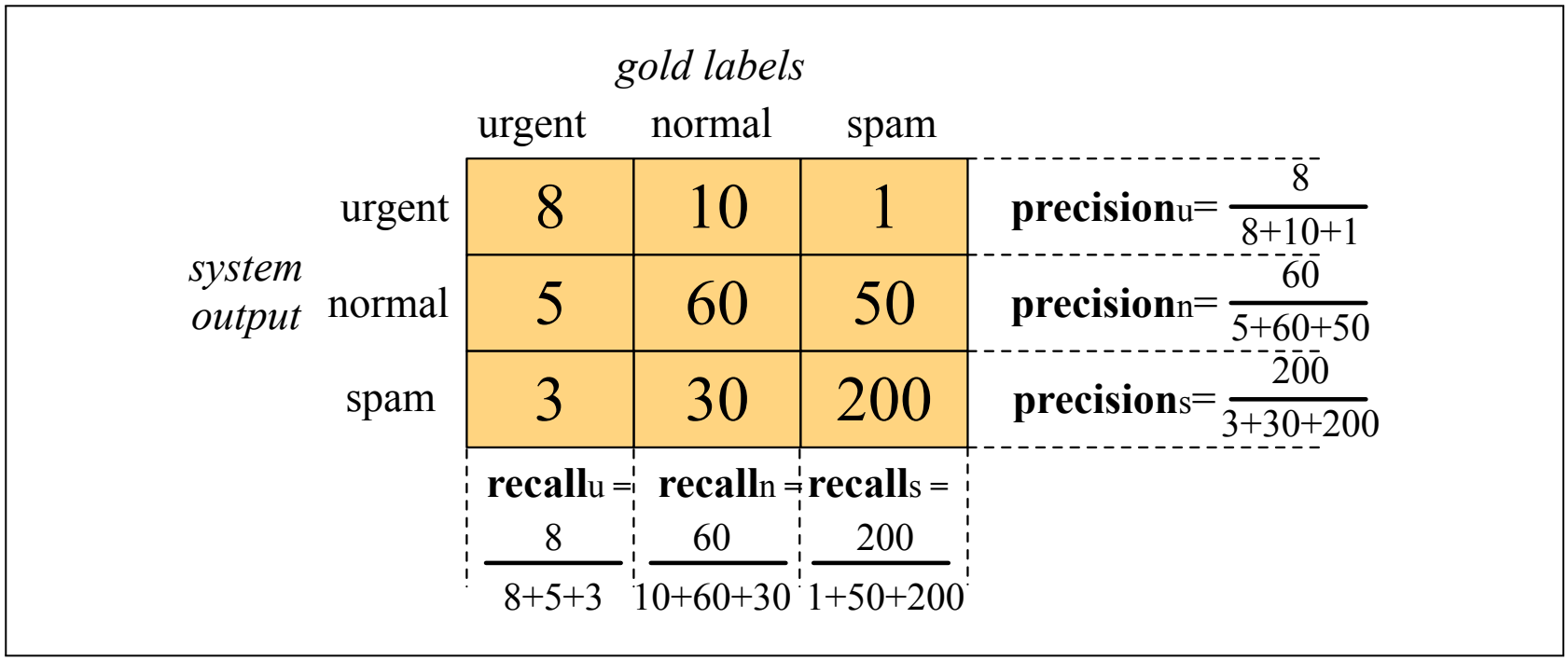


Figure 4.5 Confusion matrix for a three-class categorization task, showing for each pair of classes (c_1, c_2), how many documents from c_1 were (in)correctly assigned to c_2.

宏平均与微平均：反映系统整体性能的单一指标

宏平均（Macroaveraging）：分别计算每个类别的精确率和召回率，再取平均。这种方法对各类别一视同仁，能较好地反映少数类别的表现。对不同类别施加不同权重 (如，按照测试集中各类别的样本数进行加权)可计算得到加权宏平均。

微平均（Microaveraging）：将所有类别的混淆矩阵合并后，基于总体的TP、FP、FN计算精确率和召回率，更易受到频数较多类别的影响。

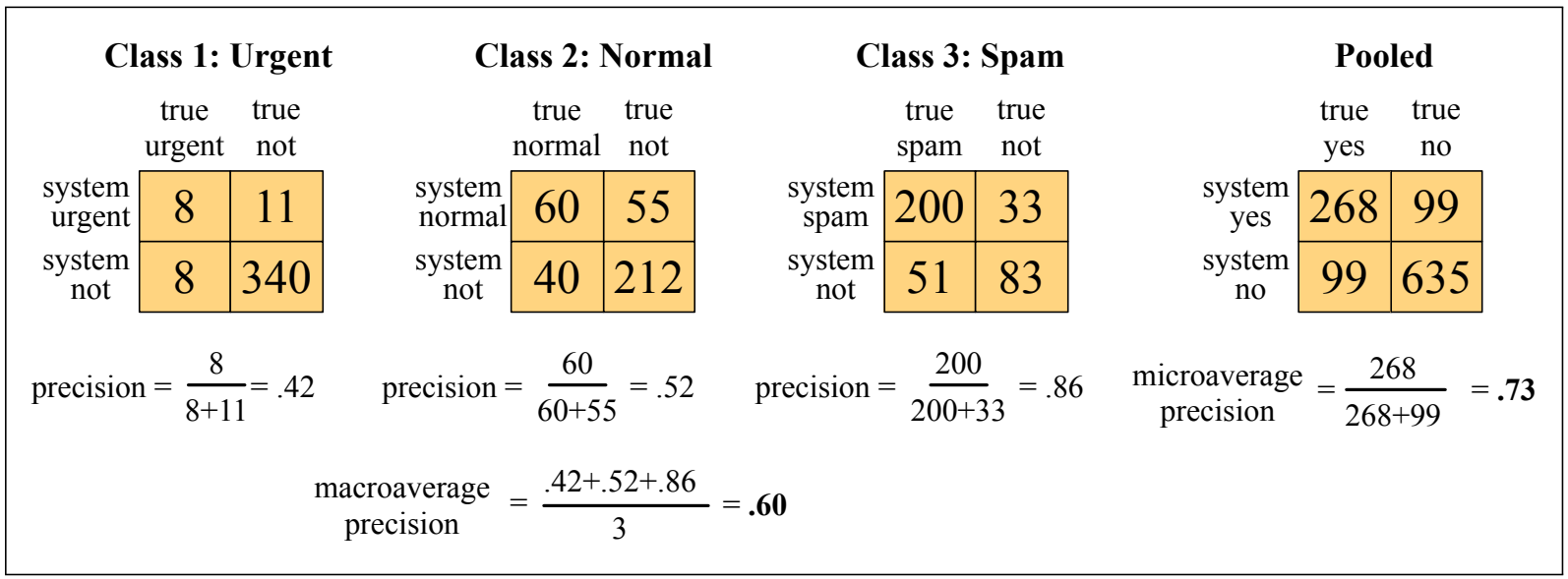


Figure 4.6 Separate confusion matrices for the 3 classes from the previous figure, showing the pooled confusion matrix and the microaveraged and macroaveraged precision.

目录

1. 固定数据集划分及其局限性

在传统的训练与测试流程中，通常将数据划分为训练集、开发集（dev set）和测试集。训练集用于模型参数的学习，开发集用于调参和模型选择，而测试集则用于评估最终模型的性能。然而，为了节省训练数据，**固定的测试集或开发集往往较小，可能不足以代表整个数据分布，从而使得评估结果存在偏差**。

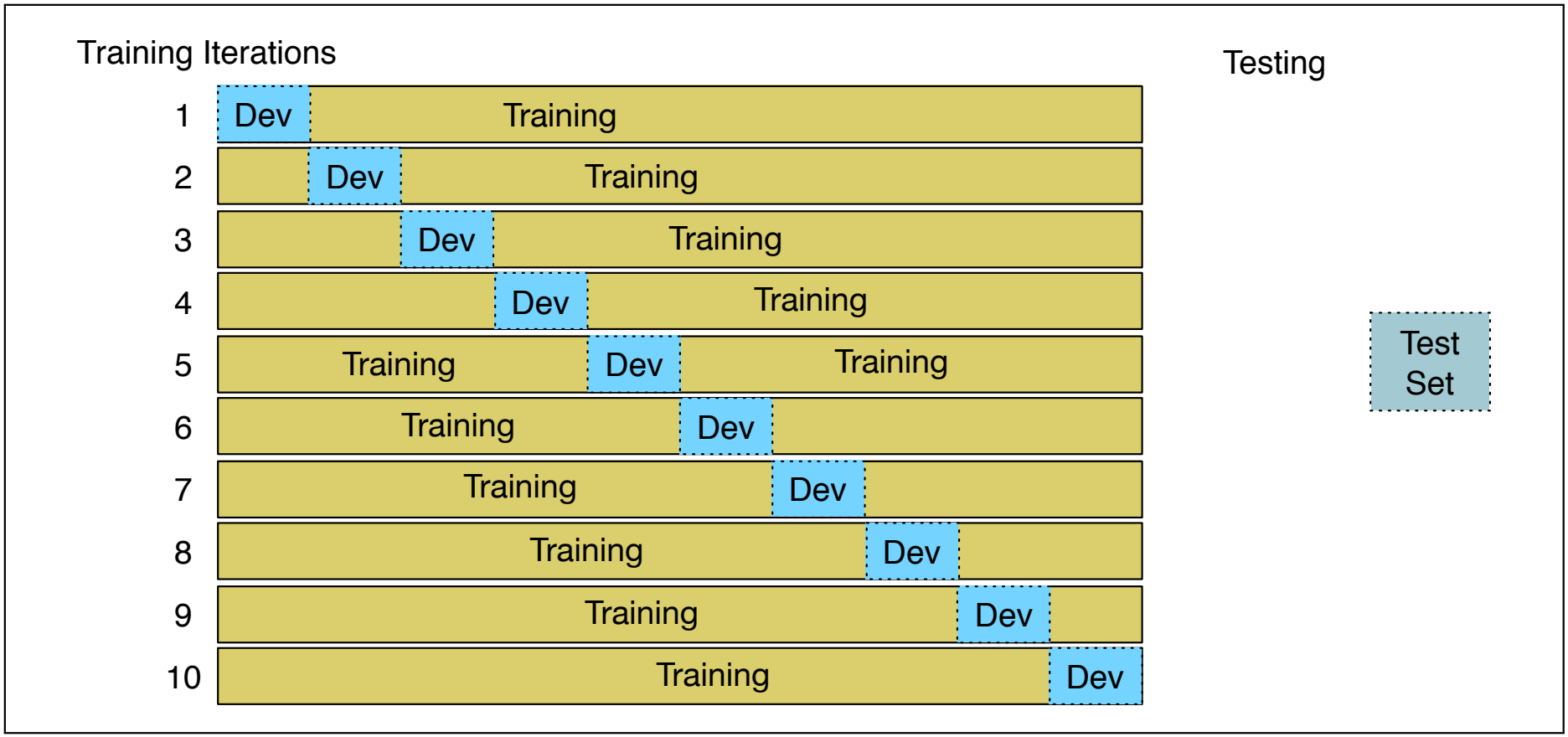
2. 交叉验证的原理与流程

为解决测试集代表性不足的问题，引入交叉验证（cross-validation）方法。其基本思想是将全部数据划分为 k 个互不重叠的子集（folds），然后依次将每一个子集作为测试集，其余 k−1 个子集用来训练模型。经过 k 次这种“轮换”过程后，可以**计算 k 个测试集上的错误率，并对其进行平均**，以获得一个更稳定、可靠的性能估计。

这种方法充分利用了所有数据，使得每个样本都有机会被用作测试数据，从而**减少了因测试集样本不足而带来的评估偏差**。

在交叉验证过程中，因为所有数据都被用作了测试，容易出现“窥视”测试集的风险，从而可能高估模型性能。

常见的做法是：首先固定划分出训练集和测试集，在训练集内部采用如10折交叉验证的方法进行参数调优和模型选择，最后再利用固定的测试集来评估模型的最终性能。



目录

利用统计显著性检验来判断两个分类系统（例如朴素贝叶斯与其他分类器）的性能差异是否足够大，从而排除偶然因素的影响。

1. 效果差异的量化与假设检验框架：为比较两个系统（例如系统A和系统B）在某一性能指标（如F1分数或准确率）上的表现，首先定义效果差异为：

$$\delta(x) = M(A, x) - M(B, x)$$

其中，$M(A, x)$和$M(B, x)$分别表示系统A和B在测试集x上的评分。基于这一差异，构造统计假设：

- 原假设H_0：$\delta(x) \leq 0$，即系统A不优于系统B；
- 备择假设H_1：$\delta(x) > 0$，即系统A确实优于系统B。

2. p值的定义与意义：p值表示在原假设成立的前提下，在测试集X上观察到当前或更大效果差异（提升）的概率。即：

$$p\text{-value}(x) = P(\delta(X) \geq \delta(x) \mid H_0\text{成立})$$

X表示一个随机变量，它的取值范围是所有可能的测试集。如果p值非常小（通常阈值设定为0.05或0.01），则可以认为**观察到的差异极不可能仅由随机抽样引起**，从而拒绝原假设，支持系统A优于系统B的结论。

p值的含义

p值是基于原假设成立时，观察到当前或更极端数据的概率。当p值非常小（如小于0.05或0.01）时，认为在原假设为真的情况下，出现这种结果的概率极低，从而有理由怀疑原假设，进而接受备择假设。

原假设背景：在进行假设检验时，首先设定一个“原假设”（H_0），比如说“系统A不优于系统B”。这意味着，假设没有真实差异，所有的观察差异都是由于随机波动产生的。

p值的定义：p值表示的是在原假设成立的前提下，会观察到像目前这样（或更极端）的数据的概率。

如何解读p值：

- 如果计算p值得到 0.03，这意味着如“系统A不优于系统B”，观测到当前（甚至更好）差异的概率只有3%，非常低，因此更相信系统A确实优于系统B，拒绝原假设。

- 如果计算p值为0.20，意味着在“系统A不优于系统B”的前提下，观察到这种差异的概率高达20%，这不算小，因此没有足够证据拒绝原假设，不能确定A一定更强。

3. 非参数检验方法在NLP中的应用：由于在NLP任务中，**很难满足诸如正态分布等参数检验的前提条件**，因此建议使用基于抽样的非参数检验方法。

常用的非参数检验方法包括近似随机化和引导法（bootstrap test），其中，**引导法的配对版本**较为常用，因为它利用了在同一测试集上两个系统的配对比较信息。

4. 引导法配对版本的具体实现

① 抽样过程：假设测试集包含10个文档，并给出系统A与系统B在每个文档上的分类结果。为了构造大量的“虚拟”测试集$x^{(i)}$，采用有放回抽样的方法，从原测试集中随机抽取n个样本，重复进行b次，从而得到b个虚拟测试集；

② 效果差异的再计算：对每一个虚拟测试集$x^{(i)}$，计算效果差异$\delta(x^{(i)})$。；

③ p值的计算：为了衡量观察到的$\delta(x)$有多“惊人”，计算满足

$$\delta(x^{(i)}) \geq 2\delta(x)$$

的虚拟测试集的比例，该比例即作为一侧的**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** $\delta(x^{(i)}) \geq 2\delta(x)$
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

Figure 4.9 A version of the paired bootstrap algorithm after Berg-Kirkpatrick et al. (2012).

THE END