Chapter4 Naive Bayes and Sentiment Classification

1、Text Classfication

```
Sentiment analysis
Spam detection
Authorship identification
Language Identification
Assigning subject categories, topics, or genres
```

2、The Naive Bayes（NB） Classifier

1.

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

2.

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(c \mid d)$$

MAP is "maximum a posteriori" = most likely class

$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(d \mid c)P(c)}{P(d)}$$

Bayes Rule

$$= \underset{c \in C}{\operatorname{argmax}} P(d \mid c)P(c)$$

Dropping the denominator

"Likelihood" "Prior"

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} \underbrace{P(d \mid c)}_{\text{"Likelihood"}} \underbrace{P(c)}_{\text{"Prior"}}$$

$$= \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c)P(c)$$

Document d represented as features x1..xn

3.

"Likelihood" "Prior"

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(d \mid c) P(c)$$

$$= \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

Document d represented as features x1..xn

4.独立性假设

$$P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet P(x_3 \mid c) \bullet \ldots \bullet P(x_n \mid c)$$

5.

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c_j) \prod_{x \in X} P(x \mid c)$$

NB Classifiers for Text Classification

1.

positions $\leftarrow$ all word positions in test document

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

2. Problem with this formula

Multiplying lots of probabilities can result in floating-point underflow!

.0006 * .0007 * .0009 * .01 * .5 * .000008….

Idea: Use logs, because $\log(ab) = \log(a) + \log(b)$

We'll sum logs of probabilities instead of multiplying probabilities!

3. Solution

Instead of this: $c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$

This: $c_{\text{NB}} = \underset{c_j \in C}{\operatorname{argmax}} \left[\log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i \mid c_j) \right]$

NB: Learning

1.

$$\hat{P}(c_j) = \frac{N_{c_j}}{N_{total}}$$

$$\hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

2. Problem

$$\hat{P}(\text{"fantastic"} \mid \text{positive}) = \frac{count(\text{"fantastic"}, \text{positive})}{\sum_{w \in V} count(w, \text{positive})} = 0$$

3. Laplace (add-1) smoothing for Naïve Bayes

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

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

4. Training Processing

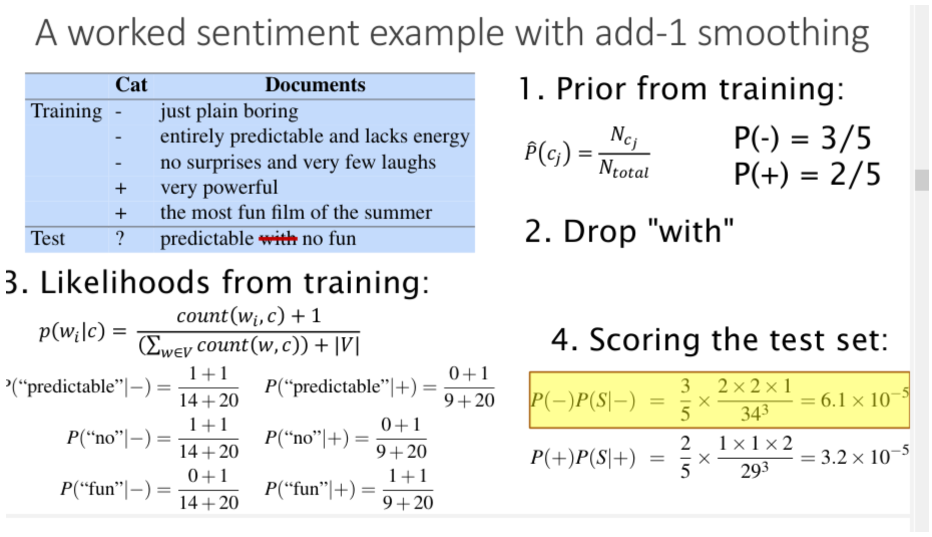
```
Calculate $P(c_j)$ terms
    ◦ For each $c_j$ in $C$ do
        $docs_j \leftarrow$ all docs with class $= c_j$
        $P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$
```

```
• Calculate $P(w_k \mid c_j)$ terms
    • $Text_j \leftarrow$ single doc containing all $docs_j$
    • For each word $w_k$ in $Vocabulary$
        $n_k \leftarrow$ # of occurrences of $w_k$ in $Text_j$
        $P(w_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$
```

5. Unknown Word， Stop Words

We ignore and remove them

Worked example



A worked sentiment example with add-1 smoothing

	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. Prior from training:

$$\hat{P}(c_j) = \frac{N_{c_j}}{N_{total}}$$

P(-) = 3/5
P(+) = 2/5

2. Drop "with"

3. Likelihoods from training:

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

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

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

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

4. Scoring the test set:

$$P(-)P(S|-) = \frac{3}{5} \times \frac{2\times 2\times 1}{34^3} = 6.1\times 10^{-5}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1\times 1\times 2}{29^3} = 3.2\times 10^{-5}$$

Optimizing for Sentiment Analysis

Binary Multinomial Naive Bayes
on a test document *d*

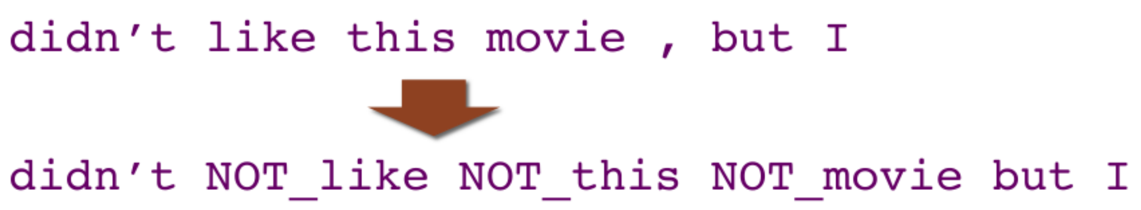
First remove all duplicate words from *d*

1. Then compute NB using the same equation:

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(w_i | c_j)$$

2.dealing with negation

```
Simple baseline method:
Add NOT_ to every word between negation and following punctuation:
```



didn't like this movie , but I

didn't NOT_like NOT_this NOT_movie but I

3. Lexicon

NB for other text classification tasks

1.Spam Filtering

```
提及数百万（美元）（（美元）NN，NNN，NNN.NN）
◦ 发件人：以许多数字开头
◦ 主题均为大写
◦ HTML的文本与图像区域的比率很低
◦ “百分之百保证”
◦ 您可以从列表中删除的索赔
```

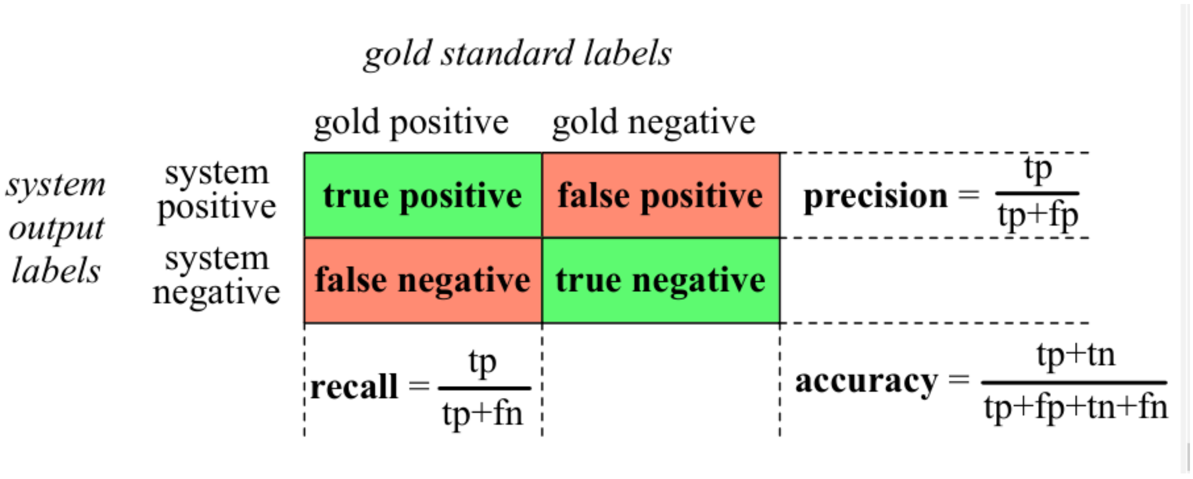
2.NB in Language ID

确定文本是用什么语言编写的。基于字符n-grams的功能非常好。重要的是，要训练每种语言的多种多样性（例如，美国英语变体，如非洲裔美国英语，或世界各地的英语变体，例如印度英语）

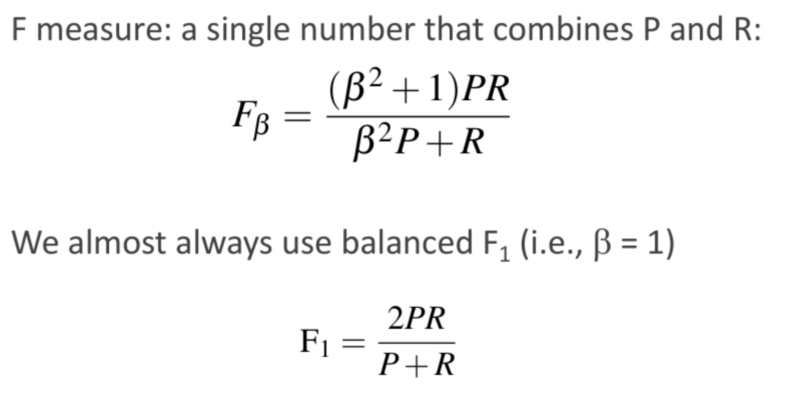
Summary

```
1.Very Fast, low storage requirements
2.Work well with very small amounts of training data
3.Robust to Irrelevant Features
4.Very good in domains with many equally important features
5.Optimal if the independence assumptions hold
6.A good dependable baseline for text classification
```

3.Evaluation : Pre, Rec, F1



F-measure:


F measure: a single number that combines P and R:

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

We almost always use balanced F_1 (i.e., $\beta = 1$)

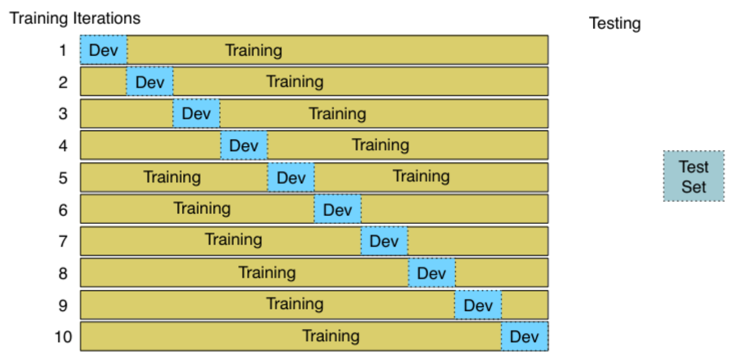
$$F_1 = \frac{2PR}{P+R}$$

Development Test Sets ("Devsets") and Cross-validation

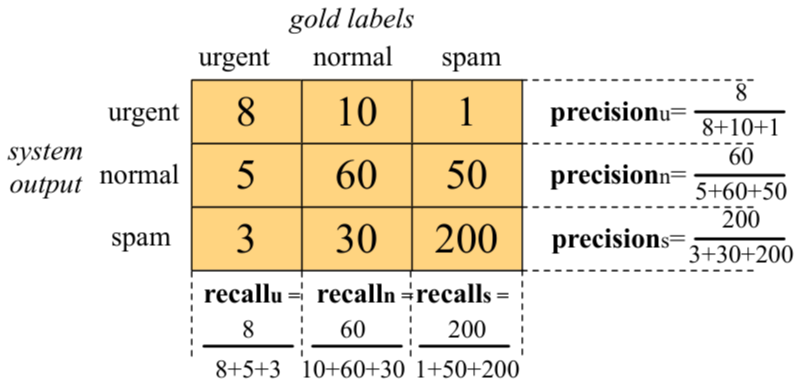
在训练集上训练，在验证集上调优，在测试集上报告

◦ 这避免了过度拟合（“调整到测试集”）
◦ 对性能的更保守估计
◦ 但矛盾的是：需要尽可能多的数据用于训练，也需要尽可能的数据用于测试；如何拆分

Cross-validation: multiple splits

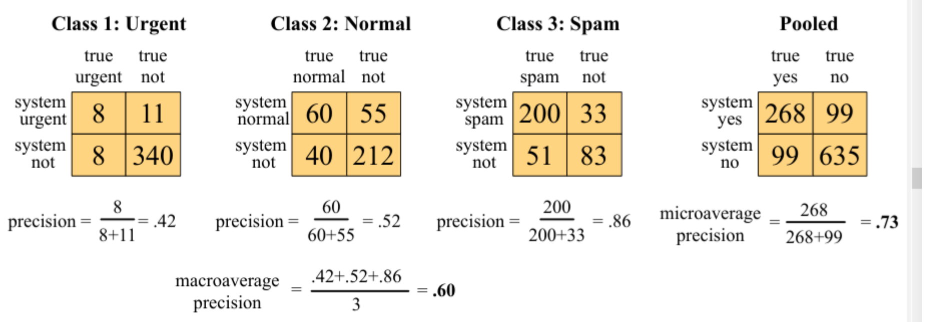


Evaluation with more than two classes



gold labels

system output	urgent	normal	spam	
urgent	8	10	1	$\textbf{precision}_u = \frac{8}{8+10+1}$
normal	5	60	50	$\textbf{precision}_n = \frac{60}{5+60+50}$
spam	3	30	200	$\textbf{precision}_s = \frac{200}{3+30+200}$
	$\textbf{recall}_u = \frac{8}{8+5+3}$	$\textbf{recall}_n = \frac{60}{10+60+30}$	$\textbf{recall}_s = \frac{200}{1+50+200}$	



Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$\text{precision} = \frac{8}{8+11} = .42$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$\text{precision} = \frac{60}{60+55} = .52$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$\text{precision} = \frac{200}{200+33} = .86$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

$\text{microaverage precision} = \frac{268}{268+99} = \mathbf{.73}$

$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = \mathbf{.60}$

4.Statistical Significance Testing

统计显著性检验

> 统计显著性检验是一种通过检验观察数据与假设之间差异的统计方法。它的目的是确定一个样本数据是否有足够的证据支持假设，从而判断这个假设在整个总体中是否成立。

问题引入

```
鉴于：
◦ 分类器A和B
◦ 度量M：M（A，x）是A在测试集x上的性能
◦ δ（x） ：x上A、B之间的性能差异：
◦ δ（x） =M（A，x）-M（B，x）
◦ 我们想知道δ（x） >0，表示A优于B
◦ δ（x） 称为效果大小
◦ 假设我们看到了δ（x） 为阳性。我们完成了吗？
◦ 不,这可能只是一个测试集的意外，或者是实验的环境。
```

P-value

```
考虑两个假设
A不比B好    Ho: δ(x)≤0
A比B好     H1：δ(x)>0
想排除Ho, 我们在测试集上创建一个随机变量X
如果Ho是真的，那么在这些测试集中，我们看到δ(x)的可能性有多大？

形式化为 p-value：
P（ δ(X)≥δ(x)| Ho为真）



P值是指观察数据出现在零假设（null hypothesis）下的概率。
具体地说：在零假设成立的情况下，得到观测结果或者比这个结果更极端的结果的概率。

如果P值非常小（通常设置一个置信水平a(0.05,0.01等)，如果P<a我们拒绝原假设
```

如何计算P-value

```
分为参数性检测和非参数检测，在NLP中，我们不倾向于使用参数检验（如t检验），我们使用基于抽样的非参数测试
```

```
统计假设检验两种常见方法：
◦ 近似随机化
◦ 引导测试



Paired tests:
◦ 比较两组观测值，其中一组观测值中的每个观测值可以与另一组观测结果配对。
◦ 例如，当在同一测试集上查看系统A和B时，我们可以在每个相同的观察xi上比较系统A和系统B的性能
```

The Paired Bootstrap Test

可以应用于任何度量（准确度、精度、召回率、F1等）。引导意味着从原始较大样本中重复抽取大量较小样本并替换（称为引导样本）

```
**function** BOOTSTRAP(test set $x$, num of samples $b$) **returns** $p\text{-}value(x)$

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 $\delta$ is probably not accidental
```

Bootstrap example

Now we create, many, say, *b*=10,000 virtual test sets *x*(*i*), each of size *n* = 10.

To make each *x*(*i*), we randomly select a cell from row *x*, with replacement, 10 times*:*

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