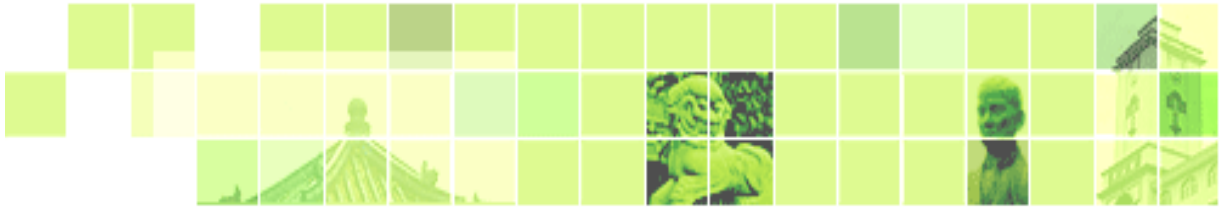
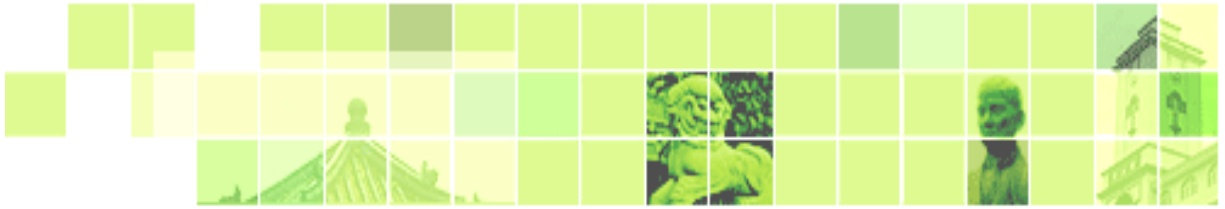



# 第4节 语言模型

# 知识回顾

■条件概率的定义：

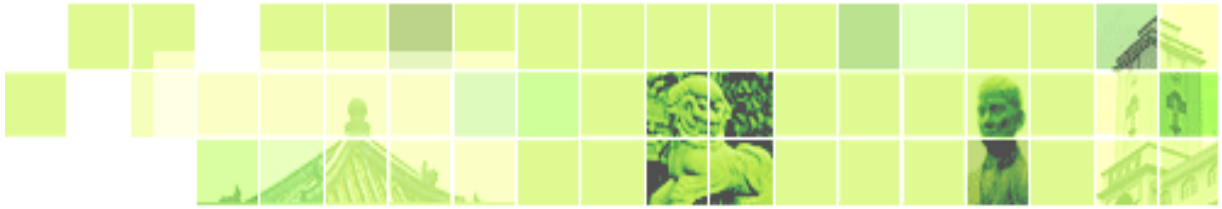
P(B|A) = P(A, B)/P(A)，即P(A, B) = P(A)P(B|A)

■如果有多个值，表示为：

P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)

■链式规则的通用表示：

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$



# 内容概览

- n元文法
- 数据平滑方法
- 困惑度

在实际应用中，我们需要解决一类问题，即一个句子出现的概率（一个句子是否合理可以通过它出现的可能性大小来判定），例如：

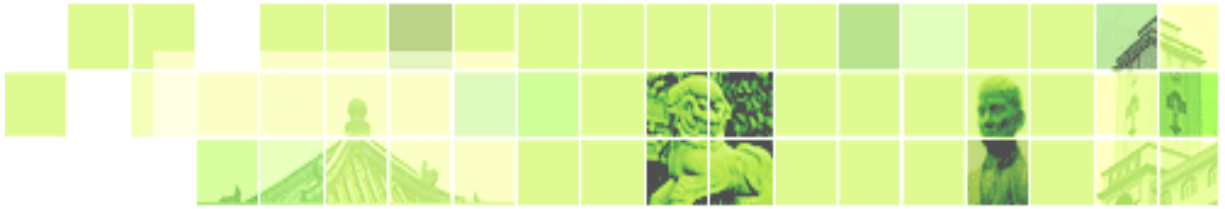
■机器翻译： P(hign winds tonite) > P(large winds tonite)

■拼写纠错 ：P(about fifteen **minutes** from) > P(about fifteen **minuets** from)

■语音识别 ：P(I saw a van) >> P(eye awe of an)

■音字转换：P(你现在干什么 | nixianzaiganshenme) > P(你西安在干什么 | nixianzaiganshenme)

■自动文摘、问答系统等

# 语言模型

- 计算一个句子或者一组词的概率：

$$P(W) = P(w_1, w_2, w_3, w_4, w_5 \ldots w_n)$$

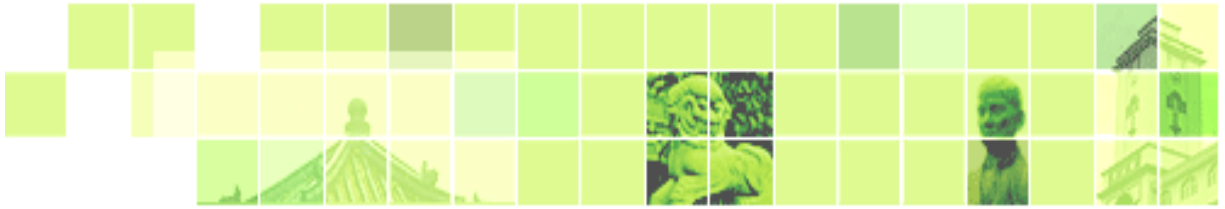
- 计算一个即将出现的词的概率：

$$P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w_2)\ldots P(w_n \mid w_1, w_2 \ldots w_{n-1})$$

- 计算以上任意一种概率的模型：

$$P(W) \quad \text{或} \quad P(w_n \mid w_1, w_2, \ldots, w_{n-1})$$

称为**语言模型（Language model or LM ）**



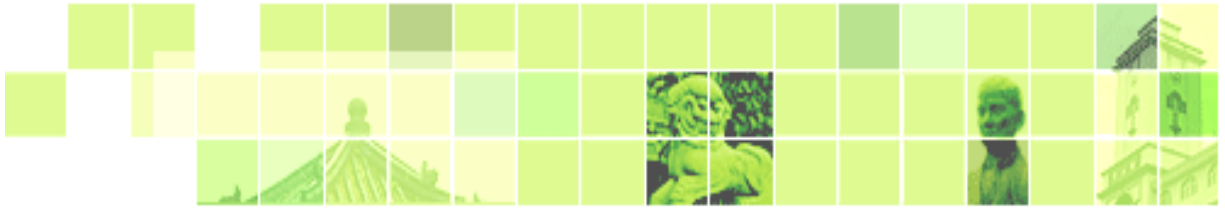
# 语言模型的定义

一个语言模型通常构建为字符串s的概率分布p(s)，这里p(s)试图反映的是字符串s作为一个句子出现的频率，对于一个由n个基元构成的句子 $s = w_1, w_2, \ldots, w_n$ ，其概率计算公式可以表示为：

$$\mathrm{P}(s) = P(\mathrm{w}_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w_2)\ldots P(w_n \mid w_1, w_2 \ldots w_{n-1})$$

$$= \prod_{i=1}^{n} P(w_i \mid w_1 \ldots w_{i-1})$$

- 当i=1时，$\mathrm{P}(\mathrm{w}_1 \mid w_0) = P(w_1)$

说明：

- $w_i$ 可以是字、词、短语或者词类等，称为统计基元

- $w_i$ $(1 \le i \le n$ )的概率由已产生的i-1个词 $w_1,\ldots,w_{i-1}$ 决定；一般得，我们把前i-1个 $w_1,\ldots,w_{i-1}$ 词构成的一个序列，称为$w_i$ 的历史(history)

计算 $P(w_i \mid w_1, w_2 \ldots w_{i-1})$ 最简单的方法是直接计数做除法

$$P(w_i \mid w_1, w_2 \ldots w_{i-1}) = \frac{count(w_1 \ldots w_{i-1}, w_i)}{count(w_1 \ldots w_{i-1})}$$

比如，对于句子NLP is so interesting有：

P(“NLP is so interesting”)

= P(NLP)✖P(is | NLP)✖P(so | NLP is) ✖P(interesting | NLP is so)

那么，这些概率的计算方法如下：

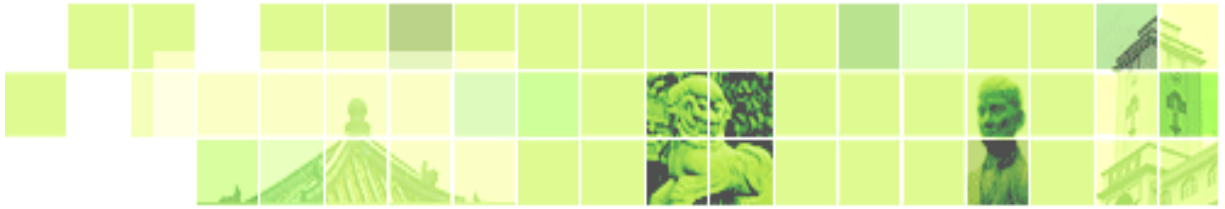
$$P(\text{interesting} \mid NLP \quad \text{is} \quad so) = \frac{count(NLP \quad \text{is} \quad so \quad \text{int}eresting)}{count(NLP \quad is \quad so)}$$

这里，我们面临两个重要问题：

●数据稀疏(Sparse Data)严重：如果按这样计算，某些的句子中一系列词同时出现的次数是很少的，组合阶数高时尤其明显，这可能导致过多的条件概率趋于0

●参数空间过大：随着历史基元数量的增加，不同的“历史”按指数级增长。假设词汇表有L个不同的基元，那么i基元就有 $L^{i-1}$ 种不同的历史情况，按照公式1，我们必须考虑在所有 $L^{i-1}$ 种不同历史情况下产生第i个基元的概率。那么模型中有 $L^{i}$ 个自由参数 $P(w_i \mid w_1, w_2 \ldots w_{i-1})$

- 假如L=5000，i=3，那么自由参数的数目就是1250亿个！这使我们几乎不可能从训练数据中正确地估计出这些参数

## 问题的解决：马尔可夫假设（Markov Assumption）

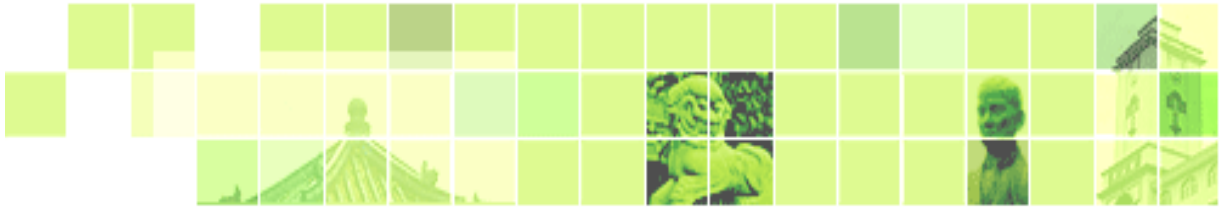
为了解决第2个问题，基于马尔可夫假设提出：下一个词的出现仅依赖于它前面的一个或者几个词

那么，假设下一个词依赖于前k个词，那么我们的 $P(W)$，计算简化如下：

$$P(w_1, w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

也就是说：

$$P(w_i \mid w_1 \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$



这种情况下的语言模型称为 **n 元文法(n-gram)**

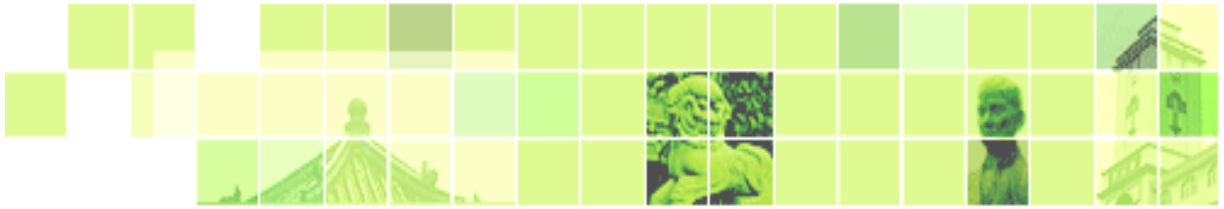
■当 k = 0时，即基元 wi 不与任何词相关，每个词之间相互独立；此时对应的模型叫做一元模型(Unigram model)， n-gram 被称为一阶马尔可夫链(uni-gram 或monogram) ,$P(W)$ 计算如下：

$$P(w_1, w_2 \ldots w_n) \approx \prod_i P(w_i)$$

■当 k = 1时, 即基元 wi 仅依赖与它前面的一个词，此时对应的模型称为二元模型(Bigram model)或者2阶马尔可夫链(bi-gram)

$$P(w_1, w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-1})$$

■当 k = 2时，即基元 wi 仅依赖与它前面的两个词，n-gram 被称为3阶马尔可夫链(tri-gram)



同样地，我们能拓展到trigram或者4-gram，5-gram...

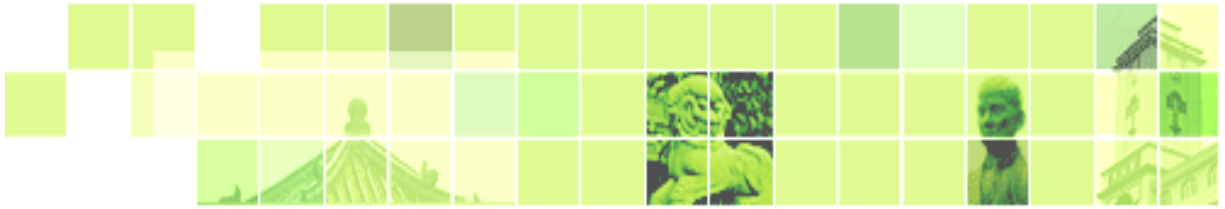
- 当k = n-1时，对应的模型为n元模型，即N-gram

总的来说，N-gram仍是一个不足的语言模型，因为有些句子存在长依赖关系(long-distance dependencies)

通常来说，对于依赖词的个数

- n值愈大，下一个词出现的依赖条件更多，辨别力更大
- n值愈小，统计数据在训练语料库中出现的次数更多，统计信息更可靠

在实际使用经验中，我们常选取bi-gram或者tri-gram模型



# N-gram概率的计算

## ——极大似然估计（The Maximum Likelihood Estimate，MLE）

对于n-gram，每一项条件概率 $P(w_i \mid w_1, w_2 \ldots w_{i-1})$ 可由最大似然估计求得：

$$P(w_i \mid w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i})}{\sum_{w_i} c(w_{i-n+1}^{i})}$$

其中，$c(w_{i-n+1}^{i})$ 是历史串 $w_{i-n+1}^{i-1}$ 在给定语料中出现的次数

为了保证条件概率在 i=1时有意义，同时为了保证句子内所有字符串的概率和为1，即 $\sum_s p(s) = 1$ ，可以在句子首尾两端增加两个标志: $< s > w_1, w_2 \ldots w_n < / s >$

具体地，以bi-gram为例，我们有一个由三句话组成的语料库EString如下：

<s>I am Sam</s>
<s>Sam am I</s>
<s>I do not like green eggs and ham</s>

P(I | <s>) = 2/3=.67　　P(Sam|<s>)=1/3=.33　　P(am|I)=2/3=.67

P(</s>|Sam) = 1/2=.5　　P(Sam|am)=1/2=.5　　P(do|I)=1/3=.33

…

则，P(EString1)=P(I|<s>)✖P(am|I)✖P(Sam|am)✖P(<s>|Sam)

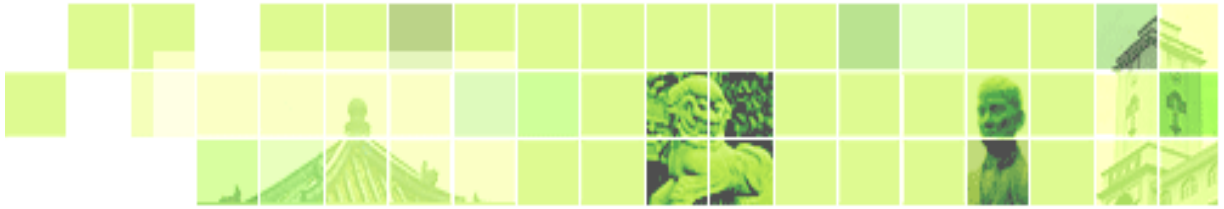
P(EString2)=P(Sam|<s>)✖P(am|Sam)✖P(I|am)✖P(<s>|I)

…

在实际情况中，我们通常在在对数空间中计算概率，原因有两个：

- 考虑在一个计算的长句子，最后得到的概率会很小甚至溢出
- 在对数空间中，加法可以代替乘法，计算更快

$$\log(p1 \times p2) = \log(p1) + \log(p2)$$



## 语言模型性能评价

最好的语言模型是一个最能预测一个看不见的测试集的模型

- 定义困惑度（Perplexity）：$PP(W) = P(w_1 w_2 ... w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 ... w_N)}}$
- 根据链式规则：$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i \mid w_1 ... w_{i-1})}}$
- 对于bi-gram：$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i \mid w_{i-1})}}$

**Lower perplexity = better model**

例子

- What is the perplexity of this sentence according to a model that assign P=1/10 to each digit?

$$\begin{aligned} \mathrm{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \left(\frac{1}{10}^{N}\right)^{-\frac{1}{N}} \\ &= \frac{1}{10}^{-1} \\ &= 10 \end{aligned}$$

- Training 38 million words, test 1.5 million words, WSJ

| N-gram Order | Unigram | Bigram | Trigram |
|---|---|---|---|
| Perplexity | 962 | 170 | 109 |

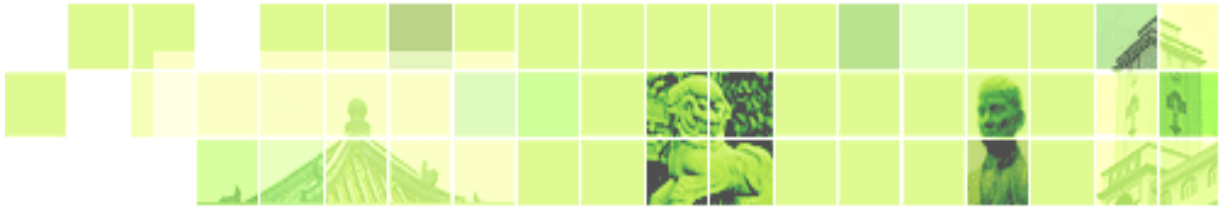
考虑一个例子，给定训练语料库如下：

<s>John read Moby Dick</s>
<s>Mary read a different book</s>
<s>She read a book by Cher</s>

根据二元文法求句子概率有：

P(John read a book)= P(John|<s>) × P(read|John) × P(a|read) ×
P(book|a) × P(</s>|book) = $\frac{1}{3}\times 1\times\frac{2}{3}\times\frac{1}{2}\times\frac{1}{2}\approx 0.06$

P(Cher read a book)= P(Cher|<s>) × P(read|Cher) × P(a|read) × P(book|a)
× P(</s>|book) = $\frac{0}{3}\times\frac{0}{1}\times\frac{2}{3}\times\frac{1}{2}\times\frac{1}{2}=0$



**尚未解决的问题：**

1、数据匮乏(稀疏) (Sparse Data) 引起零概率问题。

2、如何解决数据匮乏问题？

问题的解决：数据平滑

## 数据平滑(Data Smoothing)的基本思想：

调整最大似然估计的概率值,使零概率增值，使非零概率下调，“劫富济贫”，消除零概率，改进模型的整体正确率

- 基本约束 $\sum_{w_i} P(w_i \mid w_1, w_2 \ldots w_{i-1}) = 1$

- 基本目标：测试样本语言模型的困惑度越小越好

## ■ 加一法Add-one (Laplace) smoothing

基本思想：每种情况出现的次数加一，即保证每个 n-gram 在训练语料中至少出现 1次

对于2-gram有：

$$P(w_i \mid w_{i-1}) = \frac{1 + c(w_{i-1}w_i)}{\sum_{w_i}[1 + c(w_{i-1}w_i)]} = \frac{1 + c(w_{i-1}w_i)}{|V| + \sum_{w_i} c(w_{i-1}w_i)}$$

其中，V为被考虑语料的词汇量，即全部可能的基元数

考虑前面提到的例子

词汇量：|V|=11

加1平滑以后：

<s>John read Moby Dick</s>
<s>Mary read a different book</s>
<s>She read a book by Cher</s>

P(Cher|<s>) =(0+1)/(11+3)=1/14

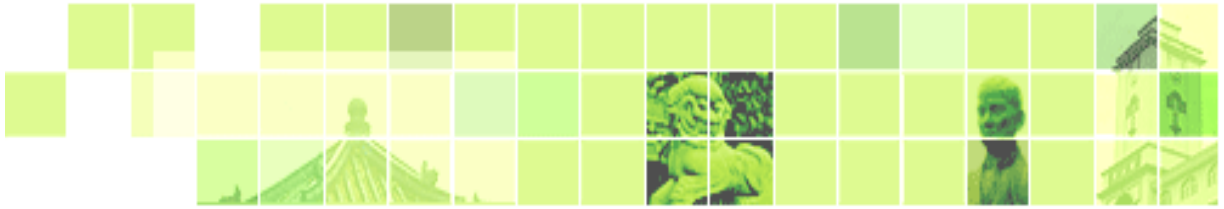
P(read|Cher) = (0+1)/(11+1)=1/12

P(a|read) = (1+2)/(11+3)=3/14

P(book|a) = (1+1)/(11+2)=2/13

P(</s>|book) = (1+1)/(11+2)=2/13

P(Cher read a book)= $\frac{1}{14}\times\frac{1}{12}\times\frac{3}{14}\times\frac{2}{13}\times\frac{2}{13}\approx 0.00003$

## ■ 减值法 (Discounting)

基本思想：修改训练样本中的事件的实际计数，使样本中不同事件的概率之和小于1，剩余的概率量分配给未见概率

- **古德-图灵（Good-Turing）估计法**

Good-Turing估计法是很多平滑技术的核心。I. J. Good 1953年引用 Turing 的方法来估计概率分布。

基本思路是：对于任何一个发生r次的n元语法，都假设它发生 $r^*$ (<r)次，其中

$$r^* = (r+1)\frac{n_{r+1}}{n_r}$$

假设N 是样本数据的大小， $n_r$是训练语料中恰好发生r次的n元语法的数目，要把这个统计数转化为概率，只需要归一化处理：对于统计数为r的n元语法，其概率为：

$$P_r = \frac{r^*}{N}$$

其中， $N=\sum_{r=0}^{\infty} n_r r^* = \sum_{r=0}^{\infty}(r+1)n_{r+1} = \sum_{r=1}^{\infty} n_r r$

这样，样本中所有事件的概率之和为：

$$\sum_{r>0} n_r \times P_r = 1 - \frac{n_1}{N} < 1$$

因此，有 $\frac{n_1}{N}$ 的剩余的概率量可以均分给所有的未见事件（r=0）

Good-Turing 估计适用于大词汇集产生的符合多项式分布的大量的观察数据

- *论文：A. Nadas. On Turing’s Formula for Word Probabilities. In IEEE Trans. On ASSP-33, Dec. 1985. Pages 1414-1416.*

- **Back-off（后退）方法**

基本思想：当某一事件在样本中出现的频率大于K (通常取 K 为0 或1)时，运用最大似然估计减值来估计其概率，否则，使用低阶模型代替高阶模型，即 (n-1)gram 的概率替代 n-gram 概率。

$$P(\mathrm{w}_n \mid w_1 \ldots w_{n-1}) = \begin{cases} (1 - a(f(w_1 \ldots w_n)) \dfrac{f(w_1 \ldots w_n)}{f(w_1 \ldots w_{n-1})} & \text{当} \mathrm{f}(w_1 \ldots w_n) > K \\ a(f(w_1 \ldots w_n)) P(\mathrm{w}_n \mid w_n \ldots w_{n-1}) & \text{当} \mathrm{f}(w_1 \ldots w_n) \le K \end{cases}$$

- α 归一化因子，为f的函数，保证概率和为1
- f(w)是指w的频率， $P(\mathrm{w}_n \mid w_2 \ldots w_{n-1})$ 指(n-1)gram概率
- $\dfrac{f(w_1 \ldots w_n)}{f(w_1 \ldots w_{n-1})}$ 是用最大似然估计方法求概率

- **绝对减值法**

基本思想：从每个计数 r 中减去同样的量，剩余的概率量由未见事件均分

设K为所有可能事件的数目（当事件为n-gram时，如果统计基元为词，且词汇集的大小为L，则 $K = L^{n}$ ）；那么，样本中出现了r次的事件的概率可以由如下公式估计：

$$P_{r} = \begin{cases} \dfrac{r-b}{N} & \text{当} r > 0 \\ \dfrac{b(K-n_0)}{Nn_0} & \text{当} r = 0 \end{cases}$$

- $n_0$为样本中未出现的事件的数目。b 为减去的常量
- $\dfrac{b(K-n_0)}{N}$ 是由于减值而产生的剩余概率量
- b值上限 $b \le \dfrac{n_1}{n_1 + 2n_2} < 1$

## ■ 插值法（Interpolation）

基本思想：将高阶模型和低阶模型作线性组合，利用低元 n-gram 模型对高元 n-gram 模型进行线性插值

- 简单插值：

$$\hat{P}(w_n \mid w_{n-2}w_{n-1}) = \lambda_1 P(w_n \mid w_{n-2}w_{n-1}) + \lambda_2 P(w_n \mid w_{n-1}) + \lambda_3 P(w_n)$$

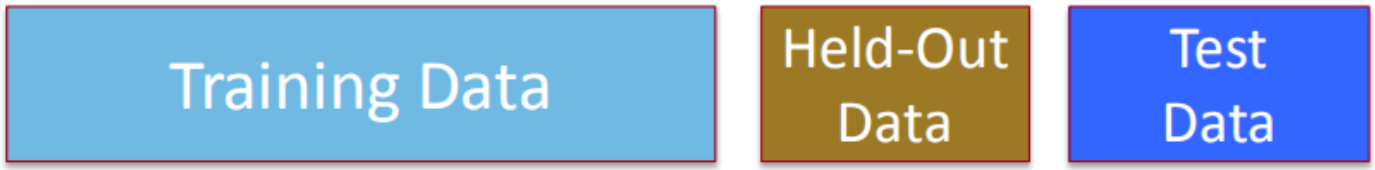
$0 \le \lambda i \le 1$，$\sum_i \lambda_i = 1$

- 上下文相关插值：

$$\hat{P}(w_n \mid w_{n-2}w_{n-1}) = \lambda_1(w_{n-2}^{n-1}) P(w_n \mid w_{n-2}w_{n-1}) + \lambda_2(w_{n-2}^{n-1}) P(w_n \mid w_{n-1}) + \lambda_3(w_{n-2}^{n-1}) P(w_n)$$

## lambdas值的估计

- 使用留存语料库(held-out corpus)：首先将数据分为训练数据集、留存数据集和测试数据集；其中训练数据集用以训练n-gram模型、然后在留存数据集里检验模型计算出的概率，再反过来修正训练数据集中的lambdas值，用以找到能使留存数据集中的概率最大的lambdas值




- 公式：

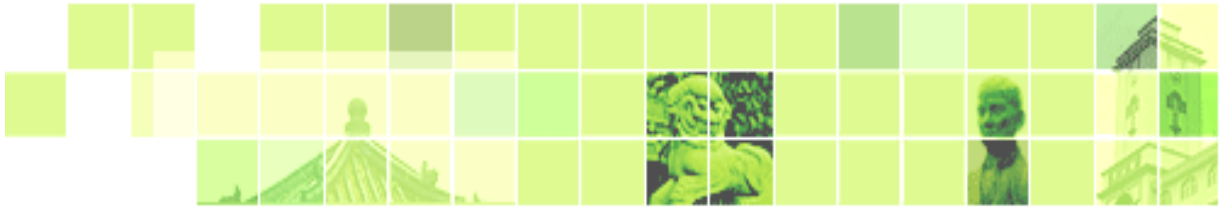
$$\log P(w_1 \ldots w_n \mid M(\lambda_1 \ldots \lambda_k)) = \sum_i \log P_{M(\lambda_1 \ldots \lambda_k)}(w_i \mid w_{i-1})$$

## ■ Stupid backoff

处理一些庞大的语料库，比如Google N-gram语料库，不能直接使用；需要剪枝(缩小规模，仅使用出现概率大于阈值的n-gram，丢弃一些高阶n-grams)及考虑存储效率。对于这类大规模语料库建立的语言模型，使用Stupid Backoff平滑方法，利用相对频率。公式：

$$S(w_i \mid w_{i-k+1}^{i-1}) = \begin{cases} \dfrac{\text{count}(w_{i-k+1}^{i})}{\text{count}(w_{i-k+1}^{i-1})} & \text{if} \quad \text{count}(w_{i-k+1}^{i}) > 0 \\ 0.4S(w_i \mid w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

$$S(w_i) = \frac{\text{count}(w_i)}{N}$$

**N-gram Smoothing Summary**

- Add-1 smoothing:

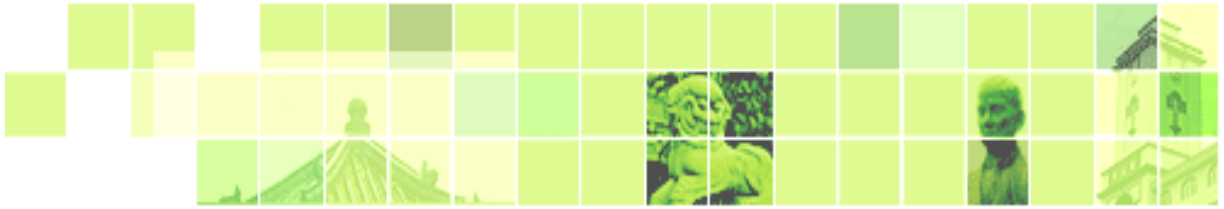
  OK for text categorization, not for language modeling

- The most commonly used method:

  Extended Interpolated Kneser-Ney

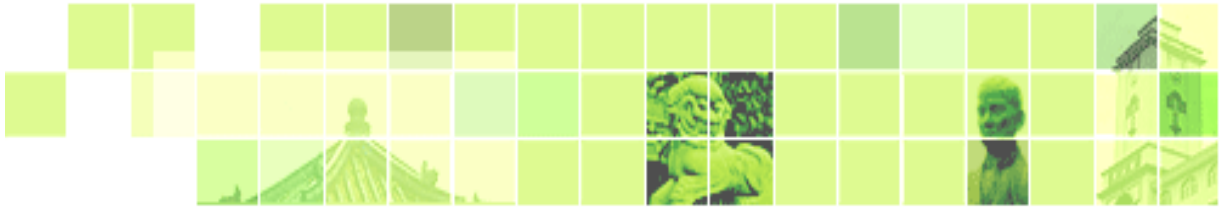
- For very large N-grams like the Web:

  Stupid backoff

## 课外资料

- SRILM: http://www.speech.sri.com/projects/srilm/
- KenLM：https://kheafield.com/code/kenlm/

Thank you！