# 语言模型



# 基本概念

1. 基于大规模语料库和统计方法，可以：
   - 发现语言使用的普遍规律；
   - 进行机器学习，自动获取语言知识；
   - 对未知现象进行推测。
2. 语言模型：以一段文字为单位统计相对频率，再根据句子构成单位的概率计算联合概率。

> 计算语句$s = w_1 w_2 \cdots w_n$的先验概率：
>
> $$\begin{aligned} p(s) &= p(w_1) \times p(w_2|w_1) \times p(w_3|w_2 w_1) \times \cdots \times p(w_m|w_1 \cdots w_{m-1}) \\ &= \prod_{i=1}^{m} p(w_i|w_1 \cdots w_{i-1}) \quad (i = 1, p(w_1|w_0) = p(w_1)) \end{aligned}$$
>
> 其中：
>
> - $w_i$被称为**统计基元**或"**词**"，可以是字、词、短语或词类等。
> - $w_i$的概率由$w_1, \cdots, w_{i-1}$决定，由特定的一组$w_1, \cdots, w_{i-1}$构成的一个序列，称为$w_i$的**历史**。

3. 问题：**不同的"历史"**随统计基元数量的**指数级增长**。

> 如果有$L$个不同的基元，则对于第$i(i > 1)$个统计基元，其具有$i-1$个历史基元，则有$L^{i-1}$种不同的历史。则模型中共有$L^m$个自由参数。
>
> $$p(w_m|w_1 \cdots w_{m-1})$$

4. 解决方法：划分等价类。

> **$n$元文法（$n$-gram）**：将两个历史映射到同一个等价类，当且仅当这两个历史中最近的$n-1$个基元相同，即：
>
> $$S(w_1,w_2,\cdots,w_i)=S(v_1,v_2,\cdots,v_k)\iff H_1:(w_{i-n+1},\cdots,w_i)=H_2:(v_{k-n+1},\cdots,v_k)$$
>
> - 当$n=1$时，则第$i$位的基元$w_i$独立于历史。一元文法也称uni-gram或monogram；
> - 当$n=2/3$时，分别称2-gram（bi-gram）/ 3-gram（tri-gram) 为1/2阶**马尔科夫链**。
>
> 首位标志符：为使条件概率在$i=1$时有意义，为保证且句子内所有字符串的概率和为1，可以在句子首尾添加标志符：$\langle BOS\rangle,\langle EOS\rangle$。不失一般性地，对$n>2$的$n$-gram，$p(s)$可以分解为：
>
> $$p(s)=\prod_{i=1}^{m+1}p(w_i|w_{i-n+1}^{i-1})$$
>
> 其中，$w_i^j$表示词序列$w_i\cdots w_j$，$w_{j-n+1}$从$w_0$开始，

5. 应用：（1）音字转换问题；（2）汉语分词问题。
    - 音字转换问题：

      > $\widehat{CString}=\argmax_{CString}p(CString).$

    - 汉语分词问题：

      > $\widehat{Seg}=\argmax_{Seg}p(Seg).$

# 参数估计

6. 重要概念：（1）**训练语料**：用于建立模型，确定模型参数的已知预料；（2）**最大似然估计**：用相对频率计算概率的方法。

> 对于$n$-gram，参数$p(w_1|w_{i-n+1}^{i-1})$可由最大似然估计求得：
>
> $$p(w_i|w_{i-n+1}^{i-1})=f(w_i|w_{i-n+1}^{i-1})=\frac{c(w_{i-n+1}^i)}{\sum_{w_i}c(w_{i-n+1}^i)}$$

7. **数据平滑**：数据匮乏(Sparse Data）或稀疏引起的**零概率问题**。

# 数据平滑

8. 基本思想：调整最大似然估计的概率值，使零概率增值、非零概率下调。从而消除零概率，改进模型的整体正确率。
    - 基本目标：测试样本的语言模型困惑度越小越好。
    - 基本约束：$\sum_{w_i}p(w_i|w_1,w_2,\cdots,w_{i-1})=1$。
9. 基本方法：（1）**加1法**；（2）**减值法/折扣法**；（3）**删除插值法**。

## 加1法

10. **加1法**：**每一种情况出现的次数加1**。
    - 对于2-gram有：

> $$
> \begin{aligned}
> p(w_i|w_{i-1}) &= \frac{1+c(w_{i-1}w_i)}{\sum_{w_i}[1+c(w_{i-1}w_i)]} \\
> &= \frac{1+c(w_{i-1}w_i)}{|V|+\sum_{w_i}c(w_{i-1}w_i)}
> \end{aligned}
> $$
>
> 其中$V$为被考虑语料的词汇量。

## 减值法/折扣法

11. 减值法/折扣法：修改训练样本中事件的实际计数，使样本中（实际出现的）不同事件概率之和小于1。**剩余的概率分配给未见事件。**
    - 主要方法：（1）Good-Turing 估计；（2）Back-off方法（/Katz平滑方法）；（3）绝对减值法；（4）线性减值法。

### Good-Turing 估计

12. **Good-Turing**估计：

> 设$N$是原来训练样本数据的大小，$n_r$是正好出现$r$次的事件的数目，则：
>
> $$N=\sum_{r=1}^{\infty}n_r r$$
>
> 由于：$N=\sum\limits_{r=0}^{\infty}n_r r^*=\sum\limits_{r=0}^{\infty}(r+1)n_{r+1}=\sum\limits_{r=1}^{\infty}n_r r$，所以 Good-Turing 平滑计数为：
>
> $$r^*=(r+1)\frac{n_{r+1}}{n_r}$$
>
> Good-Turing 估计在样本中出现$r$次的事件的概率为：
>
> $$p_r=\frac{r^*}{N}=\frac{(r+1)n_{r+1}}{N\cdot n_r}$$
>
> 实际应用中直接用$n_{r+1}$代替$E(n_{r+1})$，用$n_r$代替$E(n_r)$。因此原样本中所有事件概率和为：
>
> $$\sum_{r>0}n_r\times p_r=1-\frac{n_1}{N}<1$$
>
> **故有$\frac{n_1}{N}$的剩余的概率量可以均分给所有未见事件。**

### Back-off方法/Katz平滑方法

13. **Back-off方法/Katz平滑方法**：对每个计数$r>0$的**n**-gram的出现次数减值，把因减值而节省下来的剩余概率根据低阶的$(n-1)$-gram分配给未见事件。

> $$
> p(w_i|w_{i-n+1}^{i-1})=\begin{cases}
> \left(1-\alpha\left(c(w_{i-n+1}^{i})\right)\right)\dfrac{c(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})}, & \text{if } c(w_{i-n+1}^{i})>K \\
> \alpha\left(c(w_{i-n+1}^{i-1})\right)p(w_i|w_{i-n+2}^{i-1}), & \text{if } c(w_{i-n+1}^{i})\le K
> \end{cases}
> $$
>
> 其中$K$可取0。

### 绝对减值法

14. **绝对减值法**：从每个计数$r$中减去相同的量，剩余的概率量由未见事件均分。

> 样本中出现了$r$次事件的概率可由如下公式估计：
>
> $$p_r = \begin{cases} \frac{r-b}{N}, & \text{when } r > 0 \\ \frac{b(R-n_0)}{N \cdot n_0}, & \text{when } r = 0 \end{cases}$$
>
> 其中$n_0$为样本中未出现的事件的数目。$b$为减去的常量，$b \leq 1$。$\frac{b(R-n_0)}{N}$是由于减值而产生的剩余概率量。
> 利用留一法（leave one out）可求得$b$的上限为：
>
> $$b \leq \frac{n_1}{n_1 + 2n_2} < 1$$
>
> 实际运用中，常用该上限代替优化的$b$。

### 线性减值法

15. **线性减值法**：从每个计数$r$中减去与该计数成正比的量（减值函数为线性的），剩余概率量$\alpha$被$n_0$个未见事件均分。

> $$p_r = \begin{cases} \frac{(1-\alpha)r}{N}, & \text{when } r > 0 \\ \frac{\alpha}{n_0}, & \text{when } r = 0 \end{cases}$$
>
> 自由参数$\alpha$的优化值为$\frac{n_1}{N}$。
> **绝对减值法产生的$n$-gram通常优于线性减值法。

## 删除插值法

16. 删除插值法：用低阶语法估计高阶语法，即$n$-gram的值不能从训练数据中准确估计时，用$n-1$-gram来替代。

> 插值公式：
>
> $$p(w_3|w_1w_2) = \lambda_3 p'(w_3|w_1w_2) + \lambda_2 p'(w_3|w_2) + \lambda_1 p'(w_3)$$
>
> 其中$\lambda_1 + \lambda_2 + \lambda_3 = 1$。

# 语言模型的自适应

## 基于缓存的语言模型

17. **基于缓存的语言模型**：
    - 针对问题：文中刚刚出现过的一些词在后面句子中再次出现的可能性往往较大，比标准的$n$-gram模型预测的概率要大。
    - 基本思路：**语言模型通过$n$-gram的线性插值求得**：

> $$\hat{p}(w_i|w_1^{i-1}) = \lambda \hat{p}_{\text{Cache}}(w_i|w_1^{i-1}) + (1-\lambda)\hat{p}_{n-\text{gram}}(w_i|w_{i-n+1}^{i-1})$$

> 处理方法：在缓存中保留前面的$K$个单词，每个词的概率（缓存概率）用其在缓存中出现的相对频率计算得出：
>
> $$\hat{p}_{\text{Cache}}(w_i|w_1^{i-1}) = \frac{1}{K}\sum_{j=i-K}^{i-1} I_{w_j=w_i}$$
>
> 其中，$I_\epsilon$为指示器函数。若$\epsilon$表示的情况出现，则$\epsilon=1$，否则为0。

## 基于混合方法的语言模型

18. **基于混合方法的语言模型**：
    - 针对问题：大规模训练语料本身是异源的，来自不同领域的语料无论在主题还是风格方面都具有一定差异。为获得最佳性能，语言模型必须适应各种不同类型的语料对其性能的影响。
    - 处理方法：将语言模型划分为$n$个子模型$M_1, M_2, \cdots, M_n$，则整个语言模型的概率通过下面的线性插值公式计算得到：

$$\hat{p}(w_i|w_1^{i-1}) = \sum_{j=1}^{n} \lambda_j \hat{p}_{M_j}(w_i|w_1^{i-1}),\ 0 \le \lambda \le 1, \sum_{j=1}^{n}\lambda_j = 1$$

## 基于最大熵的语言模

19. **基于最大熵的语言模型**：通过结合不同信息源的信息构建**一个语言模型**。每个信息员提供一组关于模型参数的约束条件，在所有满足约束的模型中，选择熵最大的模型。

> 例如：考虑两个模型$M_1, M_2$。假设$M_1$为标准的2-元模型，表示为$f$函数：
>
> $$\hat{p}_{m_1}(w_i|w_1^{i-1}) = f(w_i, w_{i-1})$$
>
> $M_2$是距离为2的2-元模型，表示为$g$函数：
>
> $$\hat{p}_{M_2}(w_i|w_1^{i-1}) = f(w_i, w_{i-2})$$
>
> 通过线性插值法取两者的平均值，并用后备（back-off）平滑计数来解决这个问题。最大熵原则奖**所有的信息源组合成一个模型**，对于该模型的约束并不是让上述两个公式对所有可能的历史均成立，而是**平均成立即可**，因此：
>
> $$E(\hat{p}_{M_1}(w_i|w_1^{i-1})|_{w_{i-1}=a}) = f(w_i, a)$$
> $$E(\hat{p}_{M_2}(w_i|w_1^{i-1})|_{w_{i-2}=b}) = g(w_i, b)$$
>
> 若约束条件是一致的，则总有模型满足这些条件。利用**通用迭代计算法**选择使熵最大的模型即可。

# 语言模型应用举例

# 神经网络语言模型