Ngram语言模型

孙栩
信息科学技术学院
xusun@pku.edu.cn
http://klcl.pku.edu.cn/member/sunxu/index.htm

语言建模(Language Modeling)

- **从统计角度看，自然语言中的一个句子s可以由任何词串构成。不过P(s)有大有小。如：**
  - a= 我准备去散步。
  - b= 我去散步准备。
  - P(a) > P(b)

- **对于给定的句子s而言，通常P(s)是未知的。**

- **对于一个句子空间A，其概率分布D表示任意可能句子的概率分布。估计句子空间A的概率分布D的过程被称作语言建模**

语言建模(Language Modeling)

- 根据语言样本估计出的概率分布D就称为语言(空间)A的语言模型

$$\sum_{a \in A} P(a) = 1$$

$$\forall a \in A, P(a) = ?$$

- 语言建模技术首先在语音识别研究中提出，后来陆续用到OCR、手写体识别、机器翻译、信息检索等领域

- 在语音识别中，如果识别结果有多个，则可以根据语言模型计算每个识别结果的可能性，然后挑选一个可能性较大的识别结果

语言建模(Language Modeling)

- 对于给定的句子a = w1 w2 … wm，如何计算P(a)?
- 链式规则(chain rule)

$$
\begin{aligned}
P(a) &= P(w_1 w_2 \dots w_m) \\
&= P(w_1)P(w_2|w_1)P(w_3|w_1 w_2)\dots P(w_m|w_1 w_2 \dots w_{m-1}) \\
&= \prod_{k=1}^{m} P(w_k|w_1 \dots w_{k-1})
\end{aligned}
$$

- 举例

$$
\begin{aligned}
&P(\text{Do you still remember}) \\
&= P(\text{Do}) \times P(\text{you}|\text{Do}) \times P(\text{still}|\text{Do you}) \times P(\text{remember}|\text{Do you still})
\end{aligned}
$$

“Shannon Game”

- “John read a _____”
- 给定一个句子中前面n-1个词，预测下面的词是哪个词
- 由于语言的规律性，句子中前面出现的词对后面可能出现的词有很强的预示作用。

N-gram

$$
\begin{aligned}
P(a) &= P(w_1 w_2 \ldots w_m) \\
&= P(w_1)P(w_2|w_1)P(w_3|w_1 w_2)\ldots P(w_m|w_1 w_2 \ldots w_{m-1}) \\
&= \prod_{k=1}^{m} P(w_k|w_1 \ldots w_{k-1})
\end{aligned}
$$

历史信息

- 为了便于计算，通常考虑的历史不能太长，一般只考虑前面n-1个词构成的历史：

$$
P(a) = \prod_{k=1}^{m} P(w_k|w_{k-n+1} \ldots w_{k-1})
$$

历史窗口为 n → n-gram

N-gram

- **“the large green _____ .”**
  - “broccoli”？“tree”？
- **“Sue swallowed the large green _____ .”**
  - “broccoli”？“tree”？

- **如果知道“Sue swallowed ”会缩小可选择的下一个词的范围**

- **如何选择n?**

N-gram

- **n 较大时**
  - 提供了更多的语境信息，语境更具区别性
  - 但是，参数个数多、计算代价大、训练语料需要多、参数估计不可靠

- **n 较小时**
  - 语境信息少，不具区别性
  - 但是，参数个数少、计算代价小、训练语料无需太多、参数估计可靠

N-gram

- **unigram (n=1)**
  - $p(w_i)$ 若语言中有20000个词，则需要估计20000个参数
- **bigram (n=2)**
  - $p(w_i|w_{i-1})$ 若语言中有20000个词，则需要估计20000^2个参数
- **trigram (n=3)**
  - $p(w_i|w_{i-2}\ w_{i-1})$ 若语言中有20000个词，则需要估计20000^3个参数

- **以下相对代价较大，使用相对少**
  - four-gram (n=4)
  - five-gram (n=5)
  - …

建立n-gram

- **数据准备:**
  - 确定训练语料
  - 对语料进行tokenization 或切分
  - 句子边界，增加两个特殊的词[B] 和 [E]
  - I am walking .
    - **[B] I am walking . [E]**
  - I am eating .
    - **[B] I am eatting . [E]**

- **n-gram概率估计**
  - 利用训练语料，估计n-gram的概率

建立n-gram

- **N-gram的最大似然估计(MLE)**
  - 选择一组参数，使得训练样本的概率最大
  - 选择能使训练样本取得最大概率值得分布作为总体分布
  - 令c(w_1,..,w_n) 表示n-gram w_1,..,w_n 在训练语料中出现的次数。则

$$P_{MLE}(w_n|w_1 \ldots w_{n-1}) = \frac{c(w_1 \ldots w_n)}{c(w_1 \ldots w_{n-1})}$$

建立n-gram

- **N-gram的最大似然估计(MLE)**
  - 假定训练语料如下
  - [B] The cat is walking [E]
  - [B] The dog is swimming [E]
  - [B] The dog is running [E]

$$P(\text{The}|[\text{B}]) = \frac{c([\text{B}]\ \text{The})}{c([\text{B}])} = \frac{3}{3} = 1$$

$$P(\text{dog}|\text{The}) = \frac{c(\text{The dog})}{c(\text{The})} = \frac{2}{3}$$

$$P(\text{is}|\text{dog}) = \frac{c(\text{dog is})}{c(\text{dog})} = \frac{2}{2} = 1$$

$$P(\text{walking}|\text{is}) = \frac{c(\text{is walking})}{c(\text{is})} = \frac{1}{3}$$

建立n-gram

- **N-gram的最大似然估计(MLE)**
  - 计算句子The dog is walking的概率
  - 首先，转换为标准形式：
  - [B] The dog is walking [E]
  - 用 2-gram语言模型计算：

$$
\begin{aligned}
&P(\text{[B] The dog is walking [E]}) \\
&= P(\text{[B]}) \times P(\text{The}|\text{[B]}) \times P(\text{dog}|\text{The}) \\
&\quad \times P(\text{is}|\text{dog}) \times P(\text{walking}|\text{is}) \times P(\text{[E]}|\text{walking}) \\
&= 1 \times 1 \times \frac{2}{3} \times 1 \times \frac{1}{3} \times 1 \\
&= \frac{2}{9}
\end{aligned}
$$

建立n-gram

- **N-gram的最大似然估计(MLE)**
  - 计算句子The dog is walking的概率
  - 句子的概率表现为若干bigram参数的乘积，若句子太长，计算时，会引起下溢(underflow)，可以采用取对数并相加的方式

$$
\begin{aligned}
&\log_{10} P(\text{[B] The dog is walking [E]}) \\
&= \log_{10} P(\text{[B]}) + \log_{10} P(\text{The}|\text{[B]}) + \log_{10} P(\text{dog}|\text{The}) \\
&\quad + \log_{10} P(\text{is}|\text{dog}) + \log_{10} P(\text{walking}|\text{is}) + \log_{10} P(\text{[E]}|\text{walking}) \\
&= \log_{10} 1 + \log_{10} 1 + \log_{10} \frac{2}{3} + \log_{10} 1 + \log_{10} \frac{1}{3} + \log_{10} 1 \\
&= -0.6532
\end{aligned}
$$

建立n-gram

- **N-gram的最大似然估计(MLE)**
  - 计算句子The dog is walking的概率
  - 怎么实现纯对数层的计算(避免下溢)
    - 比如有时候需要计算概率之和，比如多个词属于一个词类
    - 求 log(e^x + e^y)，但是如何避免直接计算 e^x 和 e^y，以防止下溢？
  - 可以使用如下算法

$$\log(e^x + e^y) = x + \log(1 + e^{y-x})$$

  - 可见，e^{y-x} 更不容易下溢，避免直接计算e^x等
  - 证明

$$\begin{aligned} e^{x+\log(1+e^{y-x})} &= e^x e^{\log(1+e^{y-x})} \\ &= e^x(1 + e^{y-x}) \\ &= e^x + e^y \end{aligned}$$

- **N-gram的最大似然估计(MLE)**
  - 例-1（注：按计算机运算过程）

$$\begin{aligned}\log(e^{-200}+e^{-220}) &= -200+\log(1+e^{-220+200}) \\ &= -200+\log(1+e^{-20}) \\ &= -200\end{aligned}$$

  - 计算机直接计算的话，产生下溢：

$$\log(e^{-200}+e^{-220}) = \log(0+0) = -\infty$$

- N-gram的最大似然估计(MLE)
  - 例-2（注：按计算机运算过程）

$$\begin{aligned}\log(e^{-200}+e^{-202}) &= -200+\log(1+e^{-202+200}) \\ &= -200+\log(1+e^{-2}) \\ &= -199.8731\end{aligned}$$

  - 计算机直接计算的话，产生下溢：

$$\log(e^{-200}+e^{-202}) = \log(0+0) = -\infty$$

数据稀疏问题(Data Sparsness)

- **考虑计算句子The dog is watching的概率**
  - c(is watching) =0
  - → P(watching|is) =0
  - → P(The dog is watching) =0

- MLE给训练样本中未观察到的事件赋以0概率

- 若某n-gram在训练语料中没有出现,则该n-gram的概率必定是0

数据稀疏问题(Data Sparsness)

- 解决的办法是扩大训练语料的规模。但是无论怎样扩大训练语料，都不可能保证所有的词在训练语料中均出现
- 由于训练样本不足而导致所估计的分布不可靠的问题，称为数据稀疏问题
- 在NLP领域中，数据稀疏问题永远存在，不太可能有一个足够大的训练语料，因为语言中的大部分词都属于低频词

Zipf 定律

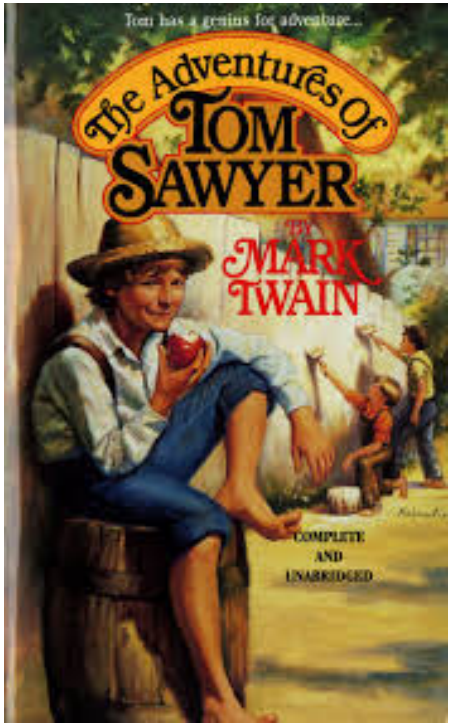
- **Zipf 定律描述了词频以及词在词频表中的位置之间的关系**
  - 针对某个语料库，若某个词w的词频是f，并且该词在词频表中的序号为r (即w是所统计的语料中第r常用词)，则
  - $f \times r = k$ (k是一个常数)
  - 若w_i在词频表中排名50，w_j在词频表中排名150，则w_i的出现频率大约是w_j的频率的3倍。

- **例：马克吐温的小说Tom Sawyer**
  - 共71,370 词(word tokens)
  - 出现了8,018 个不同的词(word types)

Zipf 定律

- 《Tom Sawyer》

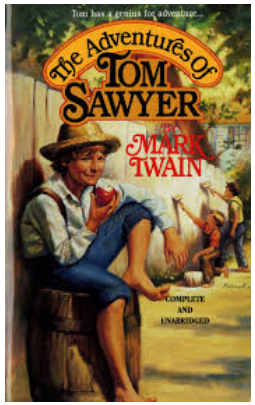
Word	Freq.	Use
the	3332	determiner (article)
and	2972	conjunction
a	1775	determiner
to	1725	preposition, verbal infinitive marker
of	1440	preposition
was	1161	auxiliary verb
it	1027	(personal/expletive) pronoun
in	906	preposition
that	877	complementizer, demonstrative
he	877	(personal) pronoun
I	783	(personal) pronoun
his	772	(possessive) pronoun
you	686	(personal) pronoun
Tom	679	proper noun
with	642	preposition



Zipf 定律

Word Frequency	Frequency of Frequency
1	3993
2	1292
3	664
4	410
5	243
6	199
7	172
8	131
9	82
10	91
11–50	540
51–100	99
> 100	102

- 大部分词是低频词，3993 (50%)词(word types)仅仅出现了一次
- 常用词极为常用，前100个高频词占了整个文本的51% (word tokens)

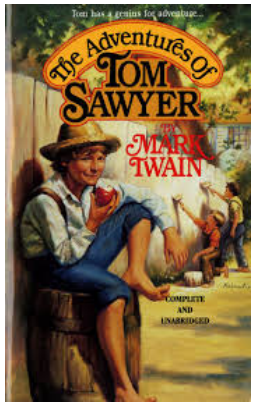


Zipf 定律

Word	Freq. (f)	Rank (r)	$f \cdot r$
the	3332	1	3332
and	2972	2	5944
a	1775	3	5235
he	877	10	8770
but	410	20	8400
be	294	30	8820
there	222	40	8880
one	172	50	8600
about	158	60	9480
more	138	70	9660
never	124	80	9920
Oh	116	90	10440
two	104	100	10400

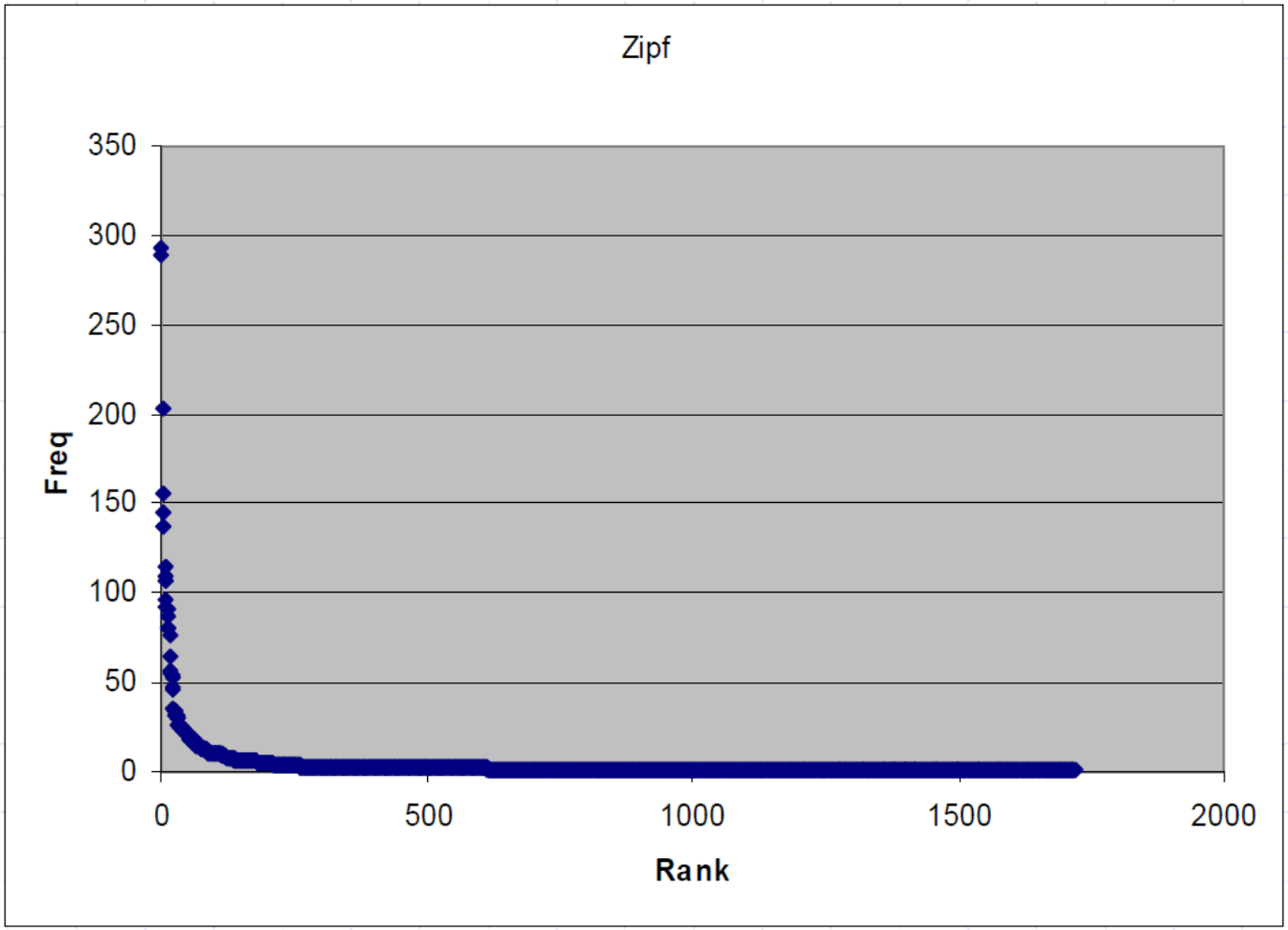
Word	Freq. (f)	Rank (r)	$f \cdot r$
turned	51	200	10200
you'll	30	300	9000
name	21	400	8400
comes	16	500	8000
group	13	600	7800
lead	11	700	7700
friends	10	800	8000
begin	9	900	8100
family	8	1000	8000
brushed	4	2000	8000
sins	2	3000	6000
Could	2	4000	8000
Applausive	1	8000	8000

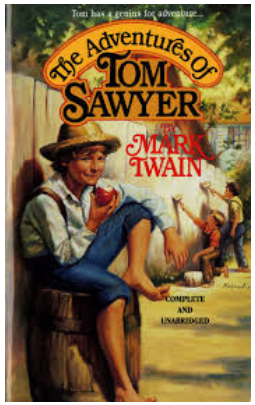
- **k ≈ 8000-9000**
- **有例外**
  - 前3个最常用的词
  - r = 100 时



Zipf 定律

- Tom Sawyer 第1-3章





Zipf 定律

- **Zipf定律告诉我们**
  - 语言中只有很少的常用词
  - 语言中大部分词都是低频词(不常用的词)
- **Zipf的解释是Principle of Least effort**
  - 讲话的人和听话的人都想省力的平衡
  - 说话人只想使用少量的常用词进行交流
  - 听话人只想使用没有歧义的词(量大低频)进行交流
- **Zipf 定律告诉我们**
  - 对于语言中的大多数词，它们在语料中的出现是稀疏的
  - 只有少量词，语料库可以提供它们规律的可靠样本

数据稀疏问题

- **Balh 等人的工作**
  - 用150 万词的训练语料训练trigram模型
  - 测试语料（同样来源）中23%的trigram没有在训练语料中出现过。
  - 对语言而言，由于数据稀疏的存在，MLE 不是一种很好的参数估计办法。

- **解决办法: 平滑技术**
  - 把在训练样本中出现过的事件的概率适当减小
  - 把减小得到的概率密度分配给训练语料中没有出现过的事件
  - 这个过程有时也称为discounting(减值)

平滑技术

- **目前已经提出了很多数据平滑技术，如：**
  - Add-one 平滑
  - Add-delta 平滑
  - Witten-Bell平滑
  - Good-Turing平滑
  - Church-Gale平滑
  - Jelinek-Mercer平滑
  - Katz平滑
  - ……

平滑技术

- **Add-one 平滑(Laplace’s law)**
  - 规定任何一个n-gram在训练语料至少出现一次（即规定没有出现过的n-gram在训练语料中出现了一次）
  - 则: new_count(n-gram) = old_count(n-gram) + 1
  - 没有出现过的n-gram的概率不再是0

平滑技术

- **Add-one 平滑(Laplace’s law)**
  - 则: new_count(n-gram) = old_count(n-gram) + 1

$$P_{add1}(w_1 w_2 \ldots w_n) = \frac{c(w_1 w_2 \ldots w_n) + 1}{N + V}$$

  - N: 训练语料中所有的n-gram的数量(token)
  - V: 所有的可能的不同的n-gram的数量(type)

平滑技术

- **Add-one 平滑**
  - 假定训练语料如下
  - [B] The cat is walking [E]
  - [B] The dog is swimming [E]
  - [B] The dog is running [E]
  - 考虑计算句子The dog is watching的概率
  - 原始方法：
  - c(is watching) =0
  - → P(watching|is) =0
  - → P(The dog is watching) =0

- **Add-one 平滑**
  - 考虑计算句子The dog is watching的概率
  - **Add-one平滑方法：**

$$
\begin{aligned}
&P(\text{[B] The dog is watching [E]}) \\
&= P(\text{[B]}) \times P(\text{The}|\text{[B]}) \times P(\text{dog}|\text{The}) \\
&\quad \times P(\text{is}|\text{dog}) \times P(\text{watching}|\text{is}) \times P(\text{[E]}|\text{watching}) \\
&= \frac{3+1}{3+1} \times \frac{3+1}{3+1} \times \frac{2+1}{3+1} \times \frac{2+1}{2+1} \times \frac{0+1}{3+1} \times \frac{0+1}{0+1} \\
&= 1 \times 1 \times \frac{3}{4} \times 1 \times \frac{1}{4} \times 1 \\
&= \frac{3}{16}
\end{aligned}
$$

平滑技术

- **Add-one 平滑**
  - 训练语料中未出现的n-gram的概率不再为0，而是一个大于0的较小的概率值
  - 但由于训练语料中未出现n-gram数量太多，平滑后，所有未出现的n-gram占据了整个概率分布中的一个很大的比例
  - 因此，在NLP中，Add-one给训练语料中没有出现过的n-gram分配了太多的概率空间。
  - 认为所有未出现的n-gram概率相等，这是否合理？
  - 出现在训练语料中的那些n-gram，都增加同样的频度值，这是否公平？(低频、高频)

平滑技术

- **Add-one 平滑**

- **AP 语料数据(Church and Gale, 1991)**
  - 语料共含有22,000,000 个bigrams (token)
  - 语料中共出现了273,266 个词，因此共有74,674,306,760 个可能的bigrams
  - 74,671,100,000 bigrams 在语料中没有出现
  - 根据Add-one平滑，每个未出现过的bigram 有一定的概率分布

- **所有未出现过的bigram的概率分布：**
  - ~99.96%

平滑技术

- **Add-delta 平滑 (Lidstone’s law)**
  - 不是加1, 而是加一个小于1的正数δ

$$P_{add\delta}(w_1 w_2 \dots w_n) = \frac{c(w_1 w_2 \dots w_n) + \delta}{N + \delta V}$$

  - 通常λ = 0.5，此时又称为Jeffreys-Perks Law或ELE

$$P_{ELE}(w_1 w_2 \dots w_n) = \frac{c(w_1 w_2 \dots w_n) + 0.5}{N + 0.5V}$$

  - 效果比Add-one好，但是仍然不理想

平滑技术

- **留存估计(Held-out Estimation)**
  - 留存数据(Held-out data)

- **把训练语料分作两个部分:**
  - 训练语料(training set): 用于初始的频率估计。
  - 留存语料(held out data): 用于改善最初的频率估计

- **对于每一个n-gram $w_1 \ldots w_n$ 计算:**

$c_{tr}(w_1 \ldots w_n)$ 该ngram在训练语料中出现的频率

$c_{ho}(w_1 \ldots w_n)$ 该ngram在留存数据中出现的频率

平滑技术

- **留存估计(Held-out Estimation)**
  - r = 某个n-gram 在训练语料中出现的频率
  - Nr = 在训练语料中出现了r次的不同的n-gram 的个数。
  - Tr = 所有在训练语料中出现了r次的n-gram 在留存语料中出现的频率之和
  - T = 留存语料中所有的n-gram数(token)
  - 有：

$$P_{ho}(w_1 \ldots w_n) = \frac{T_r}{T} \times \frac{1}{N_r}$$

$$\text{其中} \quad r = c_{tr}(w_1 \ldots w_n)$$

平滑技术

- **留存估计(Held-out Estimation)**
  - r = 某个n-gram 在训练语料中出现的频率
  - Nr = 在训练语料中出现了r次的不同的n-gram 的type数。
  - Tr = 所有在训练语料中出现了r次的n-gram 在留存语料中出现的频率之和
  - T = 留存语料中所有的n-gram数(token)
  - 有：

$$P_{ho}(w_1 \dots w_n) = \frac{T_r}{T} \times \frac{1}{N_r}$$

所有在训练语料中出现r次的n-gram 在留存语料中的概率

由于在训练语料中共有Nr不同的出现了r次的n-gram，让它们等概率分布

平滑技术

- **留存估计(Held-out Estimation)**
- **例如：假定**
  - r=5 并且有10个不同的n-gram (types) 在训练语料中出现了5次
    - 即: N5 = 10
  - 这10个不同的n-gram在留存语料中共出现了20次
    - 即: T5 = 20
  - 留存语料中共包含2000 个n-gram (token)
    - 即: T=2000

$$P(\text{an ngram with r=5}) = \frac{20}{2000} \times \frac{1}{10} = 0.001$$

平滑技术

- **删除估计(Deleted Estimation)**
  - 如果有很多训练语料的话，可以使用留存估计
  - 如果训练语料不多的话，可以
    - 把训练语料分成两个部分part0 和part1
    - 把part0 作为训练语料，把part1 作为留存语料进行建模
    - 再用part1 作为训练语料，把part0 作为留存语料进行建模
    - 对所得到的两个模型加权平均，求得最后的模型
  - 这样的方法通常称作删除估计或双向交叉验证(two-fold cross validation)

平滑技术

- **Good-Turing 平滑**
- **主要思想:**
  - 利用高频率n-gram的频率调整低频的n-gram的频率

$$r^* = (r+1)\frac{N_{r+1}}{N_r}$$

出现了r+1 次的n-gram的type数

出现了r 次的n-gram的type数

  - 分配给所有未出现的n-gram的概率空间

$$N_0 \frac{r_0^*}{N} = N_0 \frac{(0+1)\frac{N_{0+1}}{N_0}}{N} = N_0 \frac{\frac{N_1}{N_0}}{N} = \frac{N_1}{N}$$

平滑技术

- 回退模型(back-off model)
- 基本思想
  - 在高阶模型可靠时，尽可能的使用高阶模型
  - 必要时，使用低阶模型
  - 回退模型的一般形式

$$P_{back}(w_n|w_1^{n-1}) = \begin{cases} \bar{P}(w_n|w_1^{n-1}) & \text{if } c(w_1^n) > 0 \\ \alpha(w_1^{n-1}) P_{back}(w_n|w_2^{n-1}) & \text{if } c(w_1^n) = 0 \end{cases}$$

  - 参数 $\alpha(w_1^{n-1})$ 是归一因子，以保证

$$\sum_{w_n} P_{back}(w_n|w_1^{n-1}) = 1$$

平滑技术

- **Katz回退模型**[1]
  - 回退公式

$$P_{back}(w_n|w_1^{n-1}) = \begin{cases} d_{c(w_1^n)} P(w_n|w_1^{n-1}) & \text{if } c(w_1^n) > 0 \\ \alpha(w_1^{n-1}) P_{back}(w_n|w_2^{n-1}) & \text{if } c(w_1^n) = 0 \end{cases}$$

  - $d_{c(w_1^n)}$ 为0到1之间的一个实数值，称为折扣率(discount ratio)
  - 不同的r具有不同的折扣率

[1] Slava M. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. IEEE Transactions on Acoustics, Speech and Signal Processing, ASSP-35(3):400-401, March.

- **Katz回退模型**
  - 回退公式

$$P_{back}(w_n|w_1^{n-1}) = \begin{cases} d_{c(w_1^n)} P(w_n|w_1^{n-1}) & \text{if } c(w_1^n) > 0 \\ \alpha(w_1^{n-1}) P_{back}(w_n|w_2^{n-1}) & \text{if } c(w_1^n) = 0 \end{cases}$$

  - 给定 w_1^{n-1} , 从 r>0 的n-grams中**扣除的概率**为：

$$S(w_1^{n-1}) = 1 - \sum_{w_n \in M(w_1^{n-1})} P_{back}(w_n|w_1^{n-1})$$

$$\text{其中} \quad M(w_1^{n-1}) = \{w_n | c(w_1^{n-1} w_n) > 0\}$$

平滑技术

- **Katz回退模型**
  - 对于给定的 w_1^{n-1}，令

$$Q(w_1^{n-1}) = \{w_n | c(w_1^{n-1} w_n) = 0\}$$

  - 如何把 $M(w_1^{n-1})$ 分配给 $Q(w_1^{n-1})$ 集合中的那些元素？
    - 回退到(n-1)-gram，因为后者更不容易数据稀疏
  - 以2-gram为例，对于Q集合中的一个w_n，如果其原始概率$P(w_n)$比较大，则应该分配更多的概率给它
  - 所以，如果r=0，则

$$P_{back}(w_2|w_1) = S(w_1) \times \frac{P(w_2)}{\sum_{w \in Q} P(w)}$$

- **Katz回退模型**
  - 回退公式

$$P_{back}(w_n|w_1^{n-1}) = \begin{cases} d_{c(w_1^n)} P(w_n|w_1^{n-1}) & \text{if } c(w_1^n) > 0 \\ \alpha(w_1^{n-1}) P_{back}(w_n|w_2^{n-1}) & \text{if } c(w_1^n) = 0 \end{cases}$$

以2-gram为例，根据前面的定义，有：

$$\alpha(w_1) = \frac{S(w_1)}{\sum_{w \in Q_{w_1}} P(w)} = \frac{1 - \sum_{w \in M_{w_1}} P_{back}(w|w_1)}{\sum_{w \in Q_{w_1}} P(w)}$$

- **Katz回退模型**
  - 回退公式

$$P_{back}(w_n|w_1^{n-1}) = \begin{cases} d_{c(w_1^n)} P(w_n|w_1^{n-1}) & \text{if } c(w_1^n) > 0 \\ \alpha(w_1^{n-1}) P_{back}(w_n|w_2^{n-1}) & \text{if } c(w_1^n) = 0 \end{cases}$$

  - 如何计算d_r?
  - 若r > k, 不折扣，即$d_r = 1$ (Katz提出k = 5)
  - 若0 < r ≤ k, 按照和Good-Turing估计类似的方式折扣，即按照 r*/r 进行折扣，根据Good-Turing中 r* 的定义，有：

$$d_r \approx \frac{r^*}{r} = \frac{(r+1)\frac{N_{r+1}}{N_r}}{r} = \frac{(r+1)N_{r+1}}{rN_r}$$

课后进一步学习指导：参考书

- **第一讲：总体介绍**
  - 《统计自然语言处理》第一章：page 1-16
  - 其它各章的框架部分
- **第二讲：数学基础**
  - 《统计自然语言处理》第二章：page 18-31
- **第三讲：ngram语言模型**
  - 《统计自然语言处理》第五章：page 83-102

谢谢！

QUESTION？