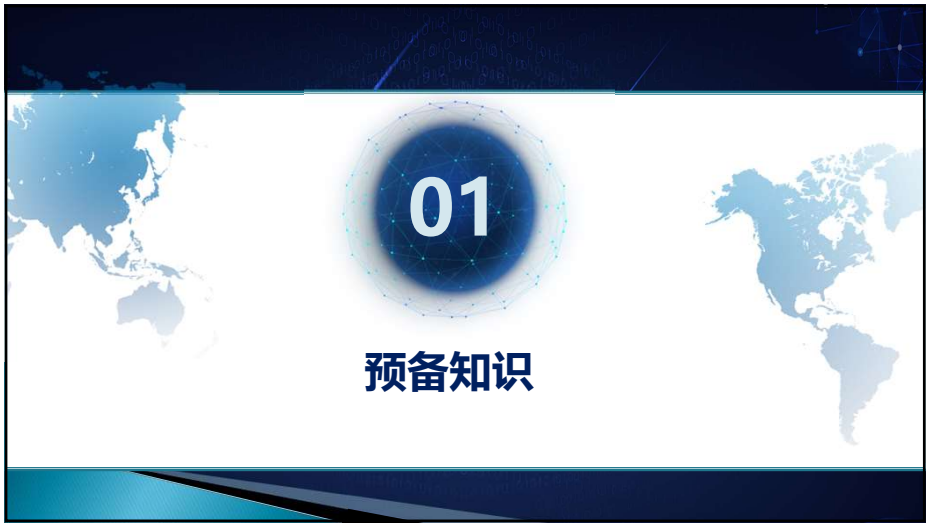
统计语言模型



目录

什么是词

- 语法学定义:能够独立运用的最小的音 义结合体
  - 分词规范: 结合紧密，使用稳定
- 词表(词典)定义:枚举""“词型”(type)
- 语料库定义:枚举"词例”(token)

什么是词

例：语料库： “A rose is a rose”
词表：{a rose is}， 词型数:3; 词例数：5

例：语料库：a=“Father read Holy Bible”
b=“Mother read a text book”
c=“He read ”a book by grandpa”
词表：？词型数:？；词例数：？

符号约定

词表定义:
"词型" (type)

语料库定义:
"词例" (token)

N	训练实例的数量
B	训练实例的类别数
W_{1n}	训练文本中的一个 n 元组 $W_1 \cdots W_n$
$C(W_1 \cdots W_n)$	训练文本中的一个 n 元组 $W_1 \cdots W_n$ 出现的频率
r	一个 n 元组的频率
$f(\cdot)$	一个模型的频率估计
N_r	含有 r 个训练实例的类别数目
T_r	在更多数据中频率为 r 的 n 元组的总数
h	先前词的历史

什么是词

例：语料库：a=“Father read Holy Bible”
b=“Mother read a text book”
c=“He read a book by grandpa”
词表：？词型数:？；词例数：？

词表： “father read Holy Bible mother a text book he by grandpa”
词型数: 11；词例数：15

Zipf 定律

- 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)*

  在自然语言的语料库里，一个单词出现的次数与它在频率表里的排名成反比。

- 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)

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

Word	Freq. (f)	Rank (r)	$f \cdot r$	Word	Freq. (f)	Rank (r)	$f \cdot r$
the	3332	1	3332	turned	51	200	10200
and	2972	2	5944	you'll	30	300	9000
a	1775	3	5235	name	21	400	8400
he	877	10	8770	comes	16	500	8000
but	410	20	8400	group	13	600	7800
be	294	30	8820	lead	11	700	7700
there	222	40	8880	friends	10	800	8000
one	172	50	8600	begin	9	900	8100
about	158	60	9480	family	8	1000	8000
more	138	70	9660	brushed	4	2000	8000
never	124	80	9920	sins	2	3000	6000
Oh	116	90	10440	Could	2	4000	8000
two	104	100	10400	Applausive	1	8000	8000

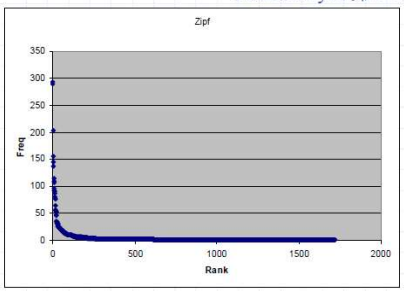
- $k \approx$ 8000-9000
- 有例外
  - 前3个最常用的词
  - $r = 100$ 时

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

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。

什么是统计语言模型

- 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。不过概率*P(s)有大有小。如：*
  - *s1= 我刚吃过晚饭*
  - *s2= 刚我过晚饭吃*
  - ***P(s1) > P(s2)***
- 对于给定的自然语言而言，通常*P(s)是未知的。*
- 对于一个服从某个未知概率分布*P的语言L，根*据给定的语言样本估计*P的过程被称作语言建*模。

02

统计语言模型

什么是统计语言模型

- 语言模型是用来计算一个句子的概率的概率模型
  - 例如：P(w1 ,w 2 ,…,w n)
- 语言模型的用途
  - 决定哪一个词序列的可能性更大
  - 已知若干个词，预测下一个词

$$\sum_{s \in L} P(s) = 1$$

- 应用
  - 语音识别
  - 机器翻译
    Claude E. Shannon. “Prediction and Entropy of Printed English”, *Bell System Technical Journal* 30:50-64. 1951.
  - 上下文敏感的拼写检查

应用于语音识别

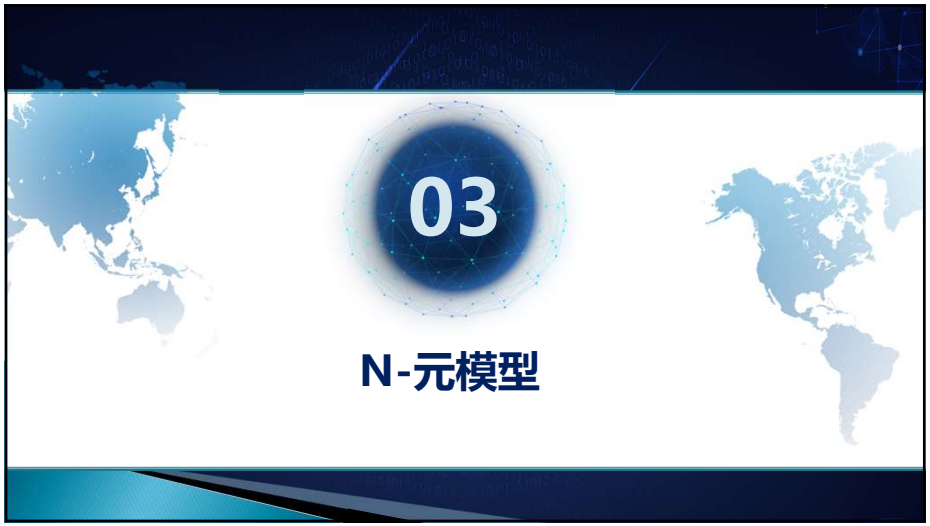
- 有的词序列听起来很像，但并不都是正确的句子
  - 例子1：
    - I went to a party. √
    - Eye went two a bar tea.
  - 例子2：
    - 你现在在干什么？ √
    - 你西安载感什么？

应用于拼写检查

- 举例
  - 汉语
    - 我自己知道 √
    - 我自已知道
  - 英语
    - Wang Gang appeared on TV. √
    - Wang Gang appeared of TV.

应用于机器翻译

- 给定一个汉语句子
  - 例如：王刚出现在电视上。
  - 英文译文：
    - Wang Gang appeared in TV.
    - In Wang Gang appeared TV.
    - Wang Gang appeared on TV. √



语言模型

- P(T) :语言模型，如何计算P(T)?
- 根据链规则

$$P(T)=P(S)=P(w_1w_2...w_n)=p(w_1)p(w_2|w_1)p(w_3|w_1w_2)...p(w_n|w_1w_2...w_{n-1})$$

- 问题:

1、参数空间过大，无法实用！

2、数据稀疏问题

例:
She swallowed the large green _____.
1) pill 2) frog 3)tree 4)mountain
???

基本思想

- “马尔科夫假设” －下一个词的出现仅仅依赖于它前面的一 个词或者几个词.
  - 假设下一个词的出现依赖于它前面的一个词

    bigram

$$P(I)=P(S)=P(w_1w_2..w_n)=p(w_1)p(w_2|w_1)p(w_3|w_1w_2)..p(w_n|w_1w_2..w_{n-1})$$
$$\approx p(w_1)p(w_2|w_1)p(w_3|w_2)..p(w_n|w_{n-1})$$

  - 假设下一下一个词的出现依赖于它前面的两个词

    trigram $\approx p(w_1)p(w_2|w_1)p(w_3|w_1w_2)..p(w_n|w_{n-2}w_{n-1})$

    ……

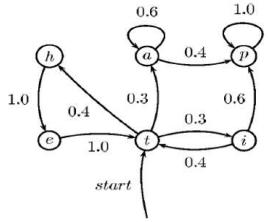
N元语法模型

- N元语法模型(N-gram Model)

$$P(w)=p(w_1)p(w_2|w_1)...p(w_n|w_1w_2...w_{n-1})$$
$$=\prod_{i=1}^{n}p(w_i|w_1w_2...w_{i-1})$$
$$\approx\prod_{i=1}^{n}p(w_i|w_{i-N+1}w_{i-N+2}...w_{i-1})$$

- 假设:单词w;出现的概率只与其前面的N-1个单词有关

二元语法模型-图示



P(t-i-p) = p(X_1 = t)p(X_2 = i|X_1 = t)p(X_3 = p|X_2 = i)

= 1.0×0.3×0.6= 0.18

N-gram语言模型

- N元文法对下一个单词的条件概率逼近的通用等式是：

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-N+1}^{n-1})$$

N的选择

- 词表中词的个数 |V| = 20,000 词

n	所有可能的n-gram的个数
2 (bigrams)	400,000,000
3 (trigrams)	8,000,000,000,000
4 (4-grams)	1.6×10^{17}

可靠性 vs. 辨别力

- *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。

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<BOS>和<EOS>
    - I eat . □ <BOS> I eat . <EOS>
    - I sleep . □ <BOS> I sleep . <EOS>
- 参数估计
  - 利用训练语料，估计模型参数

N元语法模型的参数估计

- 最大似然估计:
  - 选择参数，使得训练语料出现的概率最大

$$p(w_n|w_1 w_2 \ldots w_{n-1}) = \frac{f(w_1 \ldots w_n)}{f(w_1 \ldots w_{n-1})}$$

—

- 用实际样本中事件出现的频率来估计该事件的概率

构造（训练）N-gram语言模型

举例

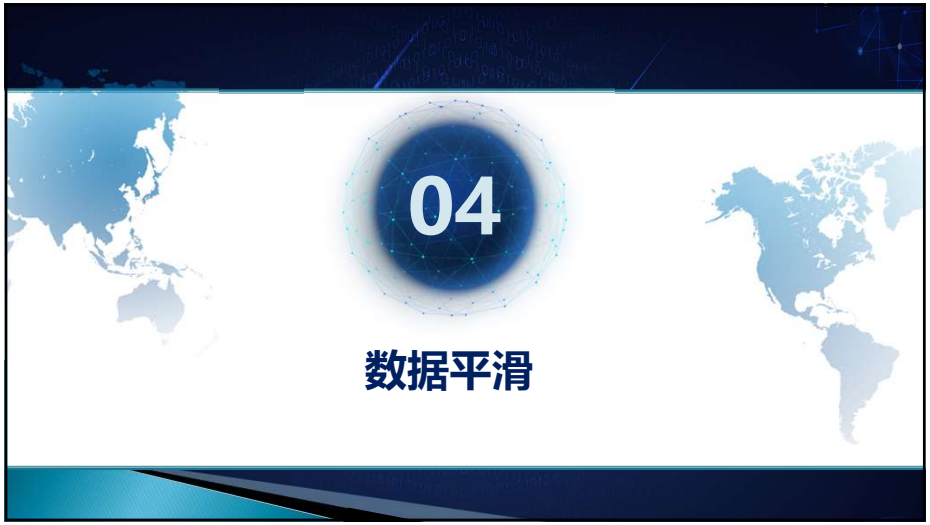
- 语料库S={a,b,c}:
  - a="Father read Holy Bible"
  - b="Mather read a text book"
  - c="He read "a book by grandpa"

P(Grandpa/<BOS>)=0
P(read|father)=1/1
P(a|read)=2/3
P(book/a)=1/2
P(<EOS>|book)=1/2
P(Grandpa read a book)=?

举例

- 语料库S={a,b,c}:
  - a="Father read Holy Bible"
  - b="Mather read a text book"
  - c="He read a book by grandpa"

P(father/<BOS>)=1/3
P(read|father)=1/1
P(a|read)=2/3
P(book/a)=1/2
P(<EOS>|book)=1/2
P(Father read a book)=?



数据稀疏问题

- 假设我们使用trigram模型

$$P(S)=p(w_1)p(w_2\mid w_1)p(w_3\mid w_1w_2)..p(w_n\mid w_{n-2}w_{n-1})$$

- 如果某个 $p(w_i\mid w_{i-2}w_{i-1})=\frac{C(w_{i-2}w_{i-1}w_i)}{C(w_{i-2}w_{i-1})}=0$
- 那么P(S)=0
- 数据稀疏问题 语料库中的Zipf定律
- 必须保证 $C\neq 0$从而使 $P\neq 0$

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
  - Good-Turing平滑
  - 回退
  - 删除插值
  - 交叉验证
- ……

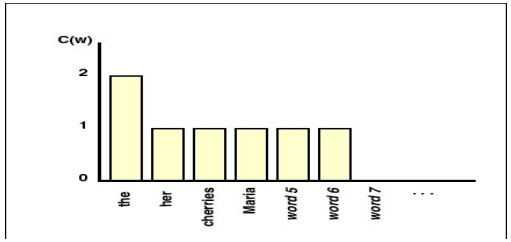
数据平滑基本思想

- 数据稀疏问题
  - 如果 $f(w_1\ldots w_n)=0$，那么出现零概率，导致整个文本的出现概率为零
- 解决办法：劫富济贫
- 约束：概率的归一性

$$\sum_{w_n} p(w_n\mid w_1w_2\ldots w_{n-1})=1$$

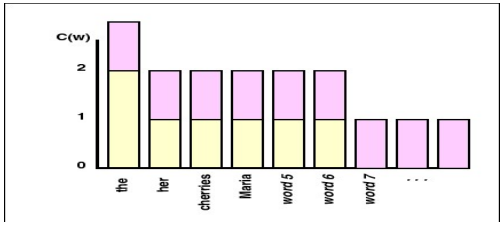
拉普拉斯定律LaPlace's Law

(加一平滑法adding one)



$$P_{Lap}(w_1w_2..w_n)=\frac{C(w_1w_2..w_n)+1}{N+B},(B=|V|^n)$$

拉普拉斯定律*(adding one)*



see the abacus	1	1/3	2	2/20003
see the abbot	0	0/3	1	1/20003
see the abduct	0	0/3	1	1/20003
see the above	2	2/3	3	3/20003
see the Abram	0	0/3	1	1/20003
...				
see the zygote	0	0/3	1	1/20003
Total	3	3/3	20003	20003/20003

可以看到对于大的词典，我们分配给未知事物的概率太大了

Add-One Smoothing

xya	1	1/3	2	2/29
xyb	0	0/3	1	1/29
xyc	0	0/3	1	1/29
xyd	2	2/3	3	3/29
xye	0	0/3	1	1/29
...				
xyz	0	0/3	1	1/29
Total xy	3	3/3	29	29/29

语料库S={a,b,c}:

a="Father read Holy Bible"

b="Mather read a text book"

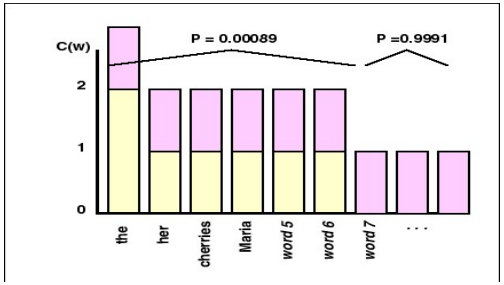
c="He read a book by grandpa"

P(Father read a book)=?

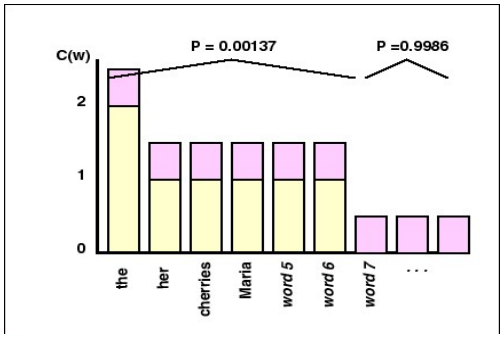
P(Grandpa read a book)=?

$$p_{Lap}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i)+1}{\sum_w [C(w_{i-1}, w)+1]} = \frac{C(w_{i-1}, w_i)+1}{\sum_w C(w_{i-1}, w)+|V|} = \frac{C(w_{i-1}, w_i)+1}{C(w_{i-1})}$$

拉普拉斯定律



Jeffreys-Perks Law



Lidstone 定律(Lidstone's Law)

$$P_{Lid}(w_1 \cdots w_n) = \frac{C(w_1 \cdots w_n) + \lambda}{N + B\lambda}$$

P = n-gram $w_1w_2…w_n$的概率

C = n-gram $w_1w_2…w_n$在训练语料库中的个数

N = 训练语料库中的 n-grams 总数

B = 所有可能的n-gram个数

λ =一个小的整数

M.L.E最大相似度估计: λ = 0
LaPlace's Law拉普拉斯定律: λ = 1
Jeffreys-Perks 定律: λ = ½

Lidstone's Law存在的问题

- λ的确定.
- 对所有未出现的n-gram都给与相同的概率
- 与最大相似度估计成线性关系

Good-Turing估计

I.J.Good于1953年引用Turing的方法来估计概率分布。

Good-Turing基本思想是:用观察计数较高的N元语法数重新估计概率量的大小，并把它指派给那些具有零计数或者较低计数的N元语法。

- N是原来训练样本数据的大小
- n_r 是在样本中正好出现 r 次的事件的数目(此处事件为 n-gram), 即出现 1 次的n-gram有 有 n_1 个，出现 2 次的 n-gram 有n_2 个, ……，出现 r 次的有 n_r 个

出现次数为r的 N-gram 词组的个数(type)

Good-Turing估计示例

- 建立频度-n-gram(本例为bigram)个数表(词表中词数14585，语料库中出现的各不相同的bigram总数199252个，bigram总数为617091个)

1	138741
2	25413
3	10531
4	5997
5	3565
6	2486
7	1754
8	1342
9	1106
10	896

Good-Turing估计

$$N=\sum_{r=1}^{\infty} n_r r$$

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

那么，**Good-Turing** 估计在样本中出现 ***r*** 次的事件的概率为：

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

stefanie

Good-Turing估计示例

r	n_r	r^*
0	14585*14585-199252	0.00065
1	138741	0. 18317
2	25413	0.82879
3	10531	1.59443
4	5997	…..
5	3565	…..
6	2486	…..
7	1754	…..
8	1342	…..
9	1106	…..
10	896	不变

- 对于未出现的bigram

N_0= 14585*14585-199252

$r^*=(r+1)\,N_1/N_0=0.0065$

$P_0(w_1,..,w_n) \approx r^*/N$
$=0.0065/617091$
$=1.058*10E-9$

$N_{r+1}=0$

Good-Turing估计示例

- 计算语料库中， 出现了1次的某bigram 概率，
  P= (r+1)N(r+1)/(N(r)N)
  =2*25413/(138741*617091)
  =5.94E-7

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。

有关(5-8)式的推导，请参阅：A. Nadas. on Turing's Formula for Word Probabilities. In *IEEE Trans. on ASSP*-33, Dec. 1985. Pages 1414-1416.

r（次数）	n_r	r*	Pr
0	60	?	?
1	50	?	?
2	40	?	?
	|V|=150		

r	n_r	r*
0	14585*14585-199252	0.00065
1	138741	0. 18317
2	25413	0.82879
3	10531	1.59443
4	5997	.....
5	3565	.....
6	2486	.....
7	1754	.....
8	1342	.....
9	1106	.....
10	896	不变

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

注意：$\sum_{r=0}^{7} p_r \neq 1$

因此，需要归一化处理：

$$\hat{p}_r = \frac{p_r}{\sum_r p_r}$$

stefanie

平滑算法—Good-Turing法

- 问题：对于最高频事件（最大的r），*由于$N_{r+1}=0$，会导致$P_r=0$，这显然是不合理的。*
- 解决办法：
  - 只对低频事件进行平滑（比如说$r<10$），*对高频时间不做平滑*。为保持概率的归一性，需要进行再次归一化操作。
  - 对最高频事件（$r=R$）*不做平滑，对其他事件都进行平滑：*

$$P_r = \begin{cases} P_R = \frac{R}{N}, & \text{当} r=R \\ (1-n_R P_R)\frac{r+1}{N}\frac{n_{r+1}}{n_r}, & \text{当} 0 \leqslant r \leqslant R-1 \end{cases}$$

  - 用一个函数S(r)对N_r*进行拟合*

r*=（r+1）S(r+1)/S(r)
S(r+1)=n_{r+1}

Good-Turing估计示例

- 简单Good-Turing [Gale & Sampson,1995]:
  - 对于比较大的r.$N_r=ar^b$ (b <-1)，用线性回归的方法估算a和 b：

    log N= log a +b log r,
  - 对于比较小的r,直接使用N_r .

平滑算法－回退（Back-off）法

- 当某一事件的频率小于K时，用n-1元语法来代替n元语法

$$P(x_n|x_1\dots x_{n-1})=\begin{cases}(1-\alpha(f(x_1\dots x_n)))\dfrac{f(x_1\dots x_n)}{f(x_1\dots x_{n-1})}, & \text{当 } f(x_1\dots x_n)>K\\ \alpha(f(x_1\dots x_{n-1}))P(x_n|x_2\dots x_{n-1}), & \text{当 } f(x_1\dots x_n)\le K\end{cases}$$

α 是归一化因子

平滑算法－Good-Turing 法

- 优点：
  - 有很好的理论基础：理论上，可以采用留一方法（交叉检验的一种），通过最大似然估计推导出这种方法
  - 其它平滑技术的基础
- 缺点：
  - 无法保证概率估计的“有序性”，即出现次数多的事件的概率大于出现次数少的事件的概率。
  - p_r 与 r/N 不能很好地近似，好的估计应当保证 $p_r<=r/N$ 。
- 适用范围：对 0<*r*<6 的小计数事件进行估计。

平滑算法－删除插值法

- 将高阶语法与低阶语法混合使用

$$P(w_3|w_1w_2)=\lambda_3P'(w_3|w_1w_2)+\lambda_2P'(w_3|w_2)+\lambda_1P'(w_3)$$

其中， $\lambda_1+\lambda_2+\lambda_3=1$

- 理论上与回退法等价

平滑的效果

- 数据平滑的效果与训练语料库的规模有关
  - 数据平滑技术是构造高鲁棒性语言模型的重要手段
  - 训练语料库规模越小,数据平滑的效果越显著,
  - 训练语料库规模越大,数据平滑的效果越不显著,甚至可以忽略不计

留存估计(Held-out Estimation)

令:

- r = 某个 n-gram 在训练语料中出现的频率
- N_r = 在训练语料中出现了r次的不同的 n-gram 的个数。
- T_r = 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。

$$T_r = \sum_{\{w_1 \cdots w_n \mid C_{tr}(w_1 \cdots w_n) = r\}} C_{ho}(w_1 \cdots w_n)$$

- T = 留存语料中所有的n-gram数(token)

$$P_{\text{ho}}(w_1 \cdots w_n) = \frac{T_r}{N_r T} \qquad \text{其中 } C_1(w_1 \cdots w_n) = r$$

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:
  - $C_{tr}(w_1 \ldots w_n)$ $w_1 \ldots w_n$ *在训练语料中出现的频率。*
  - $C_{ho}(w_1 \ldots w_n)$ $w_1 \ldots w_n$ *在留存数据中出现的频率*

留存估计(Held-out Estimation)

$$P_{ho}(w_1 \cdots w_n) = \frac{T_r}{T} \times \frac{1}{N_r}, \quad r = C_{tr}(w_1 \cdots w_n)$$

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 N_r 不同的出现了r次的 n-gram，让它们等概率分布

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: $N_5 = 10$
- 这10个不同的n-gram在留存语料中共出现了20次，即:$T_5 = 20$
- 留存语料中共包含2000 个 n-gram (token)

$$P_{ho}(\text{an } n\text{-gram with } r = 5) = \frac{20}{2000} \times \frac{1}{10} = 0.001$$

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part 0 和part 1
  - 把part 0 作为训练语料，把part 1 作为留存语料进行建模
  - 再用part 1 作为训练语料，把part 0 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

$$P_{\text{del}}(w_1 \cdots w_n) = \frac{T_r^{01} + T_r^{10}}{N(N_r^0 + N_r^1)} \qquad \text{其中} \quad C(w_1 \cdots w_n) = r$$

删除估计或双向交叉验证
(Deleted Estimation or two-way cross validation)

Google N-Gram Release, August 2006

AUG 3

All Our N-gram are Belong to You

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects,

…

That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

N元语法模型工具

- 开源工具：
- -SRI Language Model

http://www.speech.sri.com/projects/srilm/

- - IRST Language Model (in Moses)

Google Book N-grams

- http://ngrams.googlelabs.com/
- http://open.163.com/movie/2014/5/S/D/M9PEG3P9U_M9PJF4ESD.html

Thank You !