哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

自然语言处理
第四章 -- 统计语言模型

徐永东

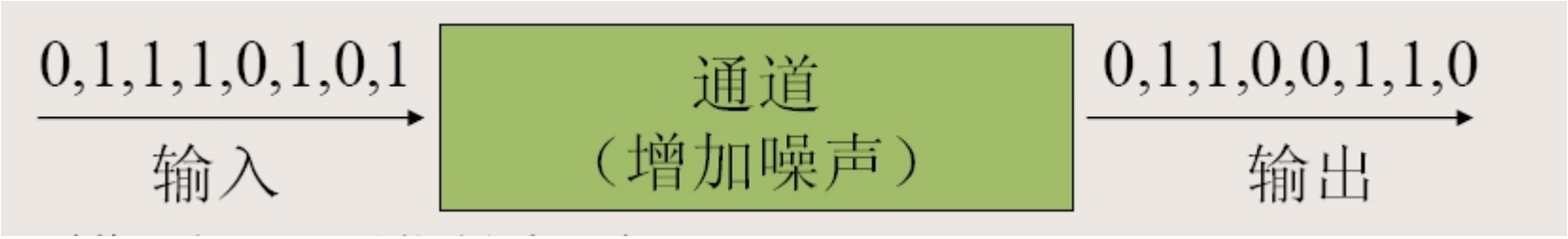
ydxu@hit.edu.cn

目录

概述

理论基础——信源－信道模型

信源信道模型（噪声信道模型）



模型：出错的概率

举例：p(0|1)=0.3, p(1|1)=0.7,
p(1|0)=0.4,p(0|0)=0.6

任务是：

已知带有噪声的输出

想知道输入是什么（也称为：Decoding）

信源—信道模型

信源模型

以概率$P(I)$生成输入信号。

信道模型

信道以概率分布$P(O/I)$将输入信号转换成输出信号。

信源—信道模型

已知输出，求解最可能的输入。

该任务的数学描述是：

$$I = \arg\max_I P(I|O) = \arg\max_I \frac{P(O|I)P(I)}{P(O)} = \arg\max_I P(O|I)P(I)$$

信源－信道模型的应用

信源－信道模型

是一种常用模型，具有广泛应用。

可根据实际问题，定义信源－信道模型的I/O 。例如：

语音识别：输入：文本 → 输出：语音。

文字识别：输入：文本 → 输出：图像。

机器翻译：输入：目标语言句子→ 输出：源语言句子。

音字转换：输入：文本 → 输出：拼音。

例子：微软拼音输入法：

任务：将用户输入的拼音流转换成文本句子。

信源－信道模型的I/O定义：输入：文本→ 输出：拼音。

微软拼音输入法的音字转换程序：

$$\hat{文本} = \arg\max_{文本} P(文本 \mid 拼音) = \arg\max_{文本} \frac{P(拼音 \mid 文本)P(文本)}{P(拼音)} = \arg\max_{文本} P(拼音 \mid 文本)P(文本)$$

语言模型：计算文本句子的概率 $P(文本)$ 。

统计语言模型

Statistical Language Model

统计语言模型试图捕获自然语言的统计规律以改善各种自然语言应用系统的性能

广泛地应用于语音识别、手写体文字识别、机器翻译、键盘输入、信息检索等领域

统计语言建模(Statistical Language Modeling)相当于对各种语言单位如字、词、句子或整篇文章进行概率分布的估计

语言模型

什么是语言模型(Language Model)

一个概率模型，用来估计语言句子出现的概率。

对于词序列 $S = w_1 w_2 \cdots w_m$

如何计算 $P(S)$？

根据链式规则：

$$P(S) = P(S = w_1 w_2 \cdots w_m) = P(w_1) \cdot P(w_2 \mid w_1) \cdot \cdots P(w_m \mid w_1 w_2 \cdots w_{m-1})$$

即使对于很小的m，上面的理想公式也很难计算，因为参数太多。

N-gram语言模型

考虑有限的记忆能力

不考虑太“旧”的历史，只记住前n-1个词，

n-1阶Markov链近似

$$P(S) = P(S = w_1 w_2 \ldots w_m)$$
$$= P(w_1) * P(w_2|w_1) * P(w_3|w_1 w_2) * \cdots * P(w_i|w_{i-n+1}^{i-1})$$

N-gram语言模型

相当于n-1阶Markov链。

“n-gram” = n个词构成的序列，

- Unigram n = 1; $P(S) = P(w_1) * P(w_2) * \cdots * P(w_m)$
- bigram n = 2; $P(S) = P(w_1) * P(w_2|w_1) * \cdots * P(w_m|w_{m-1})$
- trigram n = 3; $P(S) = P(w_1) * P(w_2|w_1) * \cdots * P(w_m|w_{m-2}, w_{m-1})$

例子（Bigram,Trigram）

p(我是一个学生)
= p(我,是,一,个,学生)
= p(我)·
p(是 |我)·
p(一 |是)·
p(个 |一)·
p(学生 |个)

p(我是一个学生)
= p(我,是,一,个,学生)
= p(我)·
p(是 |我)·
p(一 |我,是)·
p(个 |是,一)·
p(学生 |一,个)

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}

N的选择：可靠性 vs. 辨别力

“我　正在　_____”

讲课?图书馆?听课?学习?借书?……

“我　正在　图书馆　_____”

学习? 借书?……

更大的 n: 对下一个词出现的约束性信息更多，更大的辨别力

更小的n: 在训练语料库中出现的次数更多，更可靠的统计结果，更高的可靠性

可靠性和可区别性成反比，需要折中。

N-gram模型应用-音字转换

给定拼音串：ta shi yan jiu sheng wu de

可能的汉字串

踏实研究生物的

他实验救生物的

他使烟酒生物的

他是研究生物的

… …

音字转换计算公式：

$$\hat{文本} = \arg\max_{文本} P(文本|拼音)$$

$$= \arg\max_{文本} \frac{P(拼音|文本)P(文本)}{P(拼音)}$$

$$= \arg\max_{文本} P(拼音|文本)P(文本)$$

$$= \arg\max_{文本} P(文本)$$

$$P(S) = P(S = w_1 w_2 \ldots w_m)$$
$$= P(w_1) * P(w_2|w_1) * P(w_3|w_1 w_2) * \cdots * P(w_i|w_{i-n+1}^{i-1})$$

计算句子的频率——n-gram语言模型

N-gram模型应用-音字转换

可能的转换结果，分词结果

踏实研究生物的：踏实/研究/生物/的

他实验救生物的：他/实验/救生/物/的

他使烟酒生物的：他/使/烟酒/生物/的

他是研究生物的：他/是/研究/生物/的

……

如果使用Bigram计算：

P(踏实研究生物的) =P(踏实)×P(研究|踏实)×P(生物|研究)×P(的|生物)

P(他实验救生物的) =P(他)×P(实验|他)×P(救生|实验)×P(物|救生))×P(的|物)

P(他是研究生物的) =P(他)×P(是|他)×P(研究|是)×P(生物|研究)×P(的|生物)

选择概率最大的句子，作为转换结果

N-gram模型应用-中文分词

给定汉字串：他是研究生物的。

可能的分词结果：

1) 他|是|研究生|物|的

2) 他|是|研究|生物|的

采用Bigram计算

$$\widehat{Seg} = \arg\max_{Seg} p(Seg \mid Text)$$

$$= \arg\max_{Seg} \frac{p(Text \mid Seg) \times p(Seg)}{p(Text)}$$

$$= \arg\max_{Seg} p(Text \mid Seg) \times p(Seg)$$

$$= \arg\max_{Seg} p(Seg)$$

P(他/是/研究生/物/的)
=P(他)×P(是|他)×P(研究生|是)×P(物|研究生)×P(的|物)×P(的)

P(他/是/研究/生物/的)
= P(他)×P(是|他)×P(研究|是)×P(生物|研究)×P(的|生物)×P(的)

N-gram模型应用-中文分词

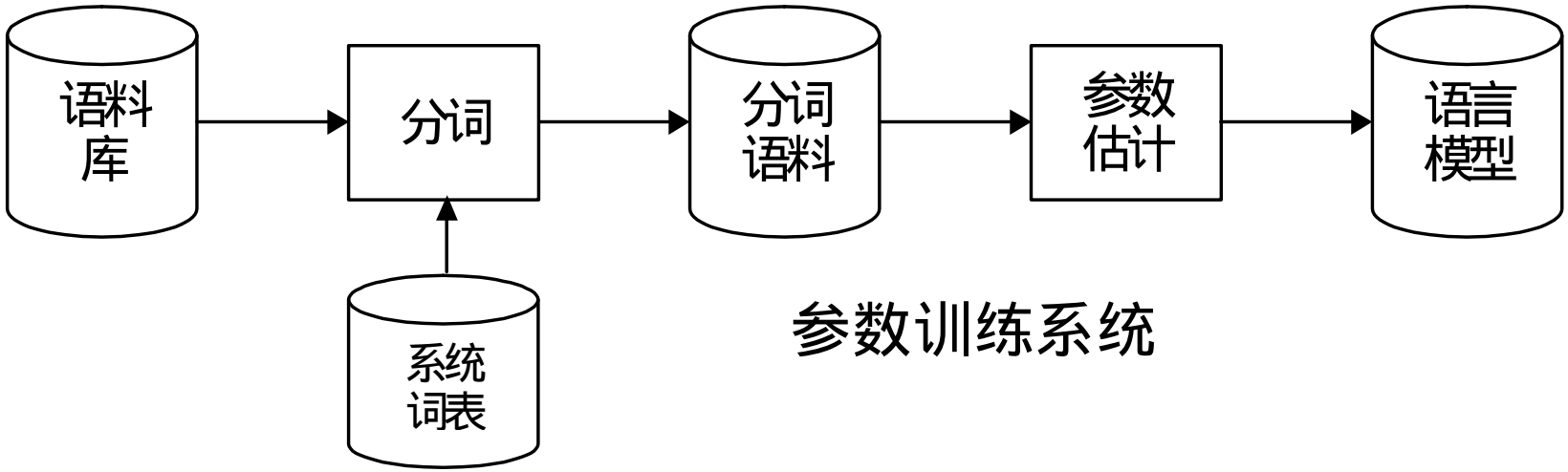
问题：如何获得二元语法模型？

模型参数估计——模型训练

两个概念

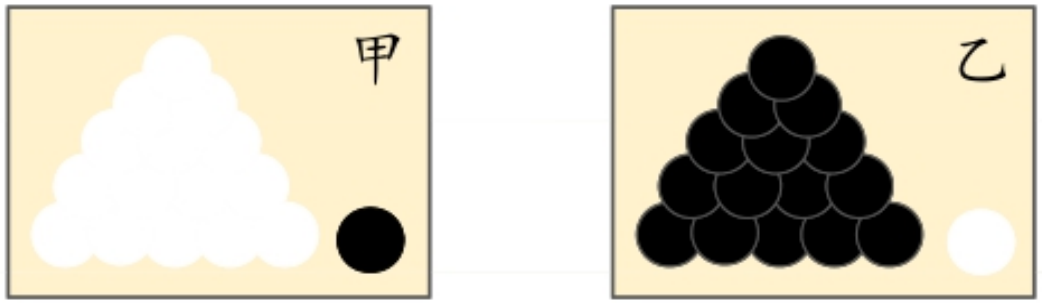
训练语料：用于建立模型的给定语料。

最大似然估计（MLE）：用相对频率计算概率的方法。



最大似然估计MLE

最大似然原理



- 例：有两个外形完全相同的箱子，甲箱中有99只白球，1只黑球；乙箱中有99只黑球，1只白球。一次试验取出一球，结果取出的是黑球。
- 问：黑球从哪个箱子中取出？

- 人们的第一印象就是：“此黑球最像是从乙箱中取出的”，这个推断符合人们的经验事实。“最像”就是“最大似然”之意，这种想法常称为“最大似然原理”（maximum-likelihood）。

最大似然估计：利用已知的样本结果，反推最有可能（最大概率）导致这样结果的参数值。

最大似然估计MLE

极大似然估计提供了一种给定观察数据来评估模型参数的方法，即“模型已定，参数未知”。通过若干次试验，观察其结果，利用试验结果得到某个参数值能够使样本出现的概率为最大，则称为极大似然估计。

样本集合 $D=\{x_1,x_2,\cdots,x_N\}$

似然函数 $l(\theta)=p(D\,|\,\theta)=p(x_1,x_2,\cdots,x_N\,|\,\theta)=\prod_{i=1}^{N}p(x_i\,|\,\theta)$

目标：求解使得出现该组样本的概率最大的θ值

$$\hat{\theta}=\arg\max_{\theta} l(\theta)=\arg\max_{\theta}\prod_{i=1}^{N}p(x_i\,|\,\theta)$$

途径： $\dfrac{dl(\theta)}{d\theta}=0$ 或者等价于 $\dfrac{dH(\theta)}{d\theta}=\dfrac{d\ln l(\theta)}{d\theta}=0$

模型参数估计——模型训练

对于 n-gram，参数 $P(w_i \mid w_{i-n+1}^{i-1})$ 可由最大似然估计求得:

$$P(w_i \mid w_{i-n+1}^{i-1}) = f(w_i \mid w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i})}{\sum_{w_i} c(w_{i-n+1}^{i})} \quad \ldots(5.5)$$

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

$f(w_i \mid w_{i-n+1}^{i-1})$ 是在给定 w_{i-n+1}^{i-1} 的条件下 w_i 出现的相对频度。

总结：N-gram模型结构

模型结构

模型：由一组模型参数组成。 $< w_1 w_2 .. w_n >$

每个N-gram模型参数：n-gram及其频度信息，形式为：

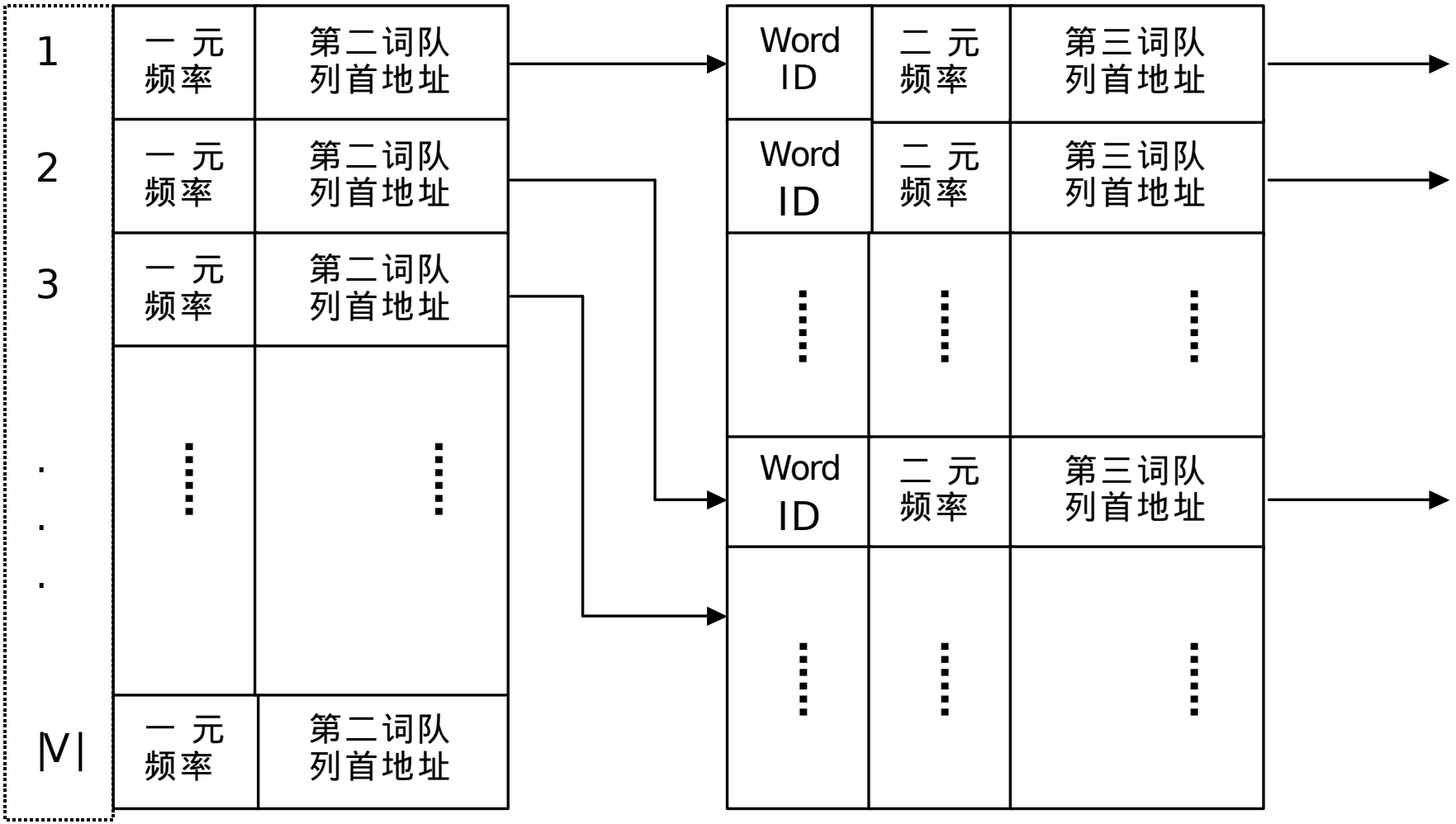
$< n-gram, f(n-gram) >$ 或 $< (w_1 w_2 .. w_n), c(w_1 w_2 .. w_n) >$

模型作用：计算词串概率

$$P(w_n \mid w_1 w_2 .. w_{n-1}) = \frac{c(w_1 w_2 .. w_n)}{c(w_1 w_2 .. w_{n-1})}$$

模型训练：在训练语料库中统计获得n-gram的频度信息

N-gram语言模型的一种存储结构



实战：基于NLTK的N-gram模型

调用jieba分词包，对给定文本进行词语切分，并对结果进行初步词频统计

```
import jieba
txt=open("《三重门》.txt","r",encoding="utf-8").read()
words=jieba.lcut(txt)
counts={}
for word in words:
    if len(word)==1:
        continue
    else:
        counts[word]=counts.get(word,0)+1
items=list(counts.items())
items.sort(key=lambda x:x[1],reverse=True)
for i in range(10):
    word,count=items[i]
    print("{0:<10}{1:>5}".format(word,count))
```

实战：基于NLTK的N-gram模型

利用nltk获取bigrams串和对应频率

```
from nltk.tokenize import word_tokenize
from nltk import bigrams, FreqDist
from math import log
w2gram = {}  # 可能存在的以w为开头的2-gram的种类数量
bigramsDist = FreqDist()
for sentence in words:
        sWordFreq = FreqDist(bigrams(word_tokenize(sentence)))
        for j in sWordFreq:
                if j in bigramsDist:
                        bigramsDist[j] += sWordFreq[j]
                else:
                        bigramsDist[j] = sWordFreq[j]
                        if j[0] in w2gram:
                                w2gram[j[0]] += 1
                        else: w2gram[j[0]] = 1
```

零概率问题

大量的低频词，无论训练数据的规模如何扩大，其出现频度仍旧很低甚至根本不出现。如果采用MLE估算它们的概率分布，将出现大量的 $p(w_i \mid w_{i-1}) = 0$ 从而导致 $p(s) = 0$ 的情况，这种情况大大削弱了该模型的描述能力。

假设我们使用Trigram模型

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

如果某个词串$w_{i-2}w_{i-1}w_i$在训练语料库中的频率为0，则

$$p(w_i \mid w_{i-2}w_{i-1}) = \frac{c(w_{i-2}w_{i-1}w_i)}{c(w_{i-2}w_{i-1})} = 0$$

那么P(S)=0，这就是**数据稀疏问题(零概率问题)**

必须保证 $c \neq 0$ 从而使 $p \neq 0$

Zipf统计定律

对于大量的低频词(N元对)来说，无论训练语料库的规模如何扩大，其出现频度依然很低或根本不出现，无法获得其足够的统计特性，用于可靠的概率估计。

直接根据频度对N-gram概率进行极大似然性估计是不可取的

N-gram模型参数的统计结果(53.7MB语料,词典59461个词)

出现次数 r	各级 N-gram 模型的出现次数=r 的 N 元对的个数 n_r						
	1-gram	2-gram	3-gram	4-gram	5-gram	6-gram	7-gram
1	22490	1342337	3474704	4343780	4411786	4158005	3789897
2	21633	585321	811778	629939	449441	324290	50817
3	20785	367848	377958	229827	131690	79959	25215
4	19971	272266	237503	129336	69213	40707	15460
5	19177	215650	168119	85473	44068	25408	10869
6	18441	179525	129037	62726	31841	18097	8255
7	17762	153201	103248	48466	24230	13790	6605
8	17171	133921	85647	39180	19583	11153	5364
9	16620	118833	72637	32571	16109	9179	4433
10	16147	106975	62718	27638	13664	7732	1230
20	12802	52520	24236	9775	4800	2456	594
30	10957	34059	13840	5487	2771	1269	215
50	8723	19421	6683	2532	1148	496	36
100	6226	8601	2392	796	298	93	1
500	2312	1056	223	65	21	5	0
1000	1314	368	62	18	6	0	0
5000	242	21	2	0	0	0	0
10000	101	4	0	0	0	0	0
20000	42	2	0	0	0	0	0
50000	9	0	0	0	0	0	0
100000	4	0	0	0	0	0	0
200000	3	0	0	0	0	0	0

数据平滑算法

❑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：$\sum_{w_i} P(w_i \mid w_1, w_2, \cdots, w_{i-1}) = 1$

❑基本目标：

测试样本语言模型的困惑度越小越好。

数据平滑(Smoothing)

平滑：对根据极大似然估计原则得到的概率分布进一步调整，确保统计语言模型中的每个概率参数均不为零，同时使概率分布更加趋向合理、均匀

算法基本思想：劫富济贫

调整字符串的频率值，使得零频率增值，非零频率下调

基本约束：

$$\sum_{w_i} P(w_i \mid w_1, w_2, \cdots, w_{i-1}) = 1$$

主要的数据平滑技术

- 加法平滑
- Good-Turing估计
- 回退平滑
- 线性插值

加1平滑

Unigram

$$p_{add}(w_i) = \frac{c(w_i)+1}{\sum_{w_i}[c(w_i)+1]} = \frac{c(w_i)+1}{|V|+\sum_{w_i} c(w_i)}$$

Bigram

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

2、Good-Turing估计

对于任何发生r次数的n元语法，都假设它发生了r^*次。

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

其中，n_r表示N-gram的训练集中实际出现r次的N元对的个数。n_r不能为零，本身需要平滑。

Good-Turing估计公式中缺乏利用低元模型对高元模型进行插值的思想，它通常不单独使用，而作为其他平滑算法中的一个计算工具。

3、线性插值平滑 (Linear Interpolation)

利用低元 N-gram模型对高元 N-gram模型进行线性插值。平滑公式：

$$p_{interp}(w_i \mid w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} \cdot p_{ML}(w_i \mid w_{i-n+1}^{i-1}) + (1-\lambda_{w_{i-n+1}^{i-1}}) \cdot p_{interp}(w_i \mid w_{i-n+2}^{i-1})$$

N-gram模型可以递归地定义为由最大似然估计原则得到的 N-gram模型和(N-1)-gram 模型的线性插值。为了结束以上的递归定义：可以令 Unigram 模型为最大似然估计模型，或者令 0-gram 模型为一个均匀分布模型 $p_{o阶}(w_i) = \frac{1}{|V|}$。

插值系数 $\lambda_{w_{i-n+1}^{i-1}}$，可以采用 Baum-Welch 算法估计出来，也可根据经验人工给出。

例子-Bigram的线性插值

$$p_{interp}(w_i \mid w_{i-1}) = \lambda \cdot p_{ML}(w_i \mid w_{i-1}) + (1-\lambda) \cdot p_{interp}(w_i)$$

4、回退式数据平滑(Backing-off)

当一个 N 元对 w_{i-n+1}^{i} 的出现次数 $c(w_{i-n+1}^{i})$ 足够大时，$p_{ML}(w_i \mid w_{i-n+1}^{i-1})$ 是 w_{i-n+1}^{i} 可靠的概率估计。而当 $c(w_{i-n+1}^{i})$ 不是足够大时，采用 Good-Turing 估计对其平滑，将其部分概率折扣给未出现的 N 元对。当 $c(w_{i-n+1}^{i})=0$ 时，模型回退到低元模型，按着 $p_{katz}(w_i \mid w_{i-n+2}^{i-1})$ 比例来分配被折扣给未出现的 N 元对的概率：

$$p_{katz}(w_i \mid w_{i-n+1}^{i-1}) = \begin{cases} p_{ML}(w_i \mid w_{i-n+1}^{i-1}) & if\,(c(w_{i-n+1}^{i}) \geq k) \\ \alpha \cdot p_{GT}(w_i \mid w_{i-n+1}^{i-1}) & if\,(1 \leq c(w_{i-n+1}^{i}) < k) \\ \beta \cdot p_{katz}(w_i \mid w_{i-n+2}^{i-1}) & if\,(c(w_{i-n+1}^{i}) = 0) \end{cases}$$

其中，$p_{ML}(w_i \mid w_{i-n+1}^{i-1})$ 为最大似然估计模型。$p_{GT}(w_i \mid w_{i-n+1}^{i-1})$ 为 Good-Turing 概率估计。阈值 k 为一个常量。参数 α 和 β 保证模型参数概率的归一化约束条件，即 $\sum_{w_i} p_{katz}(w_i \mid w_{i-n+1}^{i-1}) = 1$。

练习：加一平滑程序

```
import nltk
corpus="hello how are you doing? Hope you find the book interesting. ".split()
sentence = "what are you doing".split()
cfd = nltk.ConditionalFreqDist(nltk.bigrams(corpus))
cprob=[]
for (a,b) in nltk.bigrams(sentence):
        cprob.append ( 1.0*(1+cfd[a][b])/(len(corpus)+cfd[a].N()) )
print cprob
```

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

现有的主要语言模型

N元文法统计模型

$$P(w_i = w \mid C) = P(w_i = w \mid w_{i-n+1}^{i-1})$$

自从几十年前在大词表语言识别系统中首次使用Trigram以来,直到现在,Trigram模型仍旧是在实际应用中表现最佳的语言模型,并且成为许多其他的语言模型的重要组成部分.

现有的主要语言模型

N-pos模型（基于词类的N-Gram模型）

$$P(w_i = w|C) = P(w_i = w|g(w_{i-n+1})g(w_{i-n+2})..g(w_{i-1}))$$

或者

$$P(w_i = w|C) = P(g(w_i)|g(w_{i-n+1})g(w_{i-n+2})..g(w_{i-1})) \cdot P(w_i = w|g(w_i))$$

$g(w)$ 表示词w的词类

参数空间较小 $|G|^{N-1}|V|$,不如n-gram语言模型精确

意义

降低模型参数的规模

数据稀疏问题的一种解决方式

N-POS模型

N-POS模型构造方法

按词性（Part-of-Speech）进行词类划分，借助于词性标注技术，构造基于词性的N-POS模型

采用词的自动聚类技术，自动构造基于词的自动聚类的类N-gram模型

例子：　设C_i为词w_i所属的词性，多种基于词性的模型结构可被使用。典型地，一个Trigram可选择如下计算方法：

$$p(w_3 \mid w_1, w_2) = p(w_3 \mid C_3) \cdot p(C_3 \mid C_1, C_2)$$

N-gram与N-POS比较

基于词的N-gram模型对近邻的语言约束关系的描述能力最强，应用程度最为广泛。一般N<=3，难以描述长距离的语言约束关系

N-POS模型的参数空间最小，一般不存在数据稀疏问题，可以构造高元模型，用于描述长距离的语言约束关系。但由于词性数目过少，过于泛化，因此又限制了语言模型的描述能力

自动聚类生成的词类数量介于词和词性的数量之间，由此建立的类N-gram模型，既不存在严重的数据稀疏问题，又不存在过于泛化问题

动态、自适应、基于缓存的语言模型

在自然语言中，经常出现某些在文本中通常很少出现的词，在某一局部文本中突然大量出现的情况。

能够根据词在局部文本中出现情况动态地调整语言模型中的概率分布数据的语言模型称为动态、自适应或者基于缓存的语言模型。

动态、自适应、基于缓存的语言模型

方法

将N个最近出现过的词$w_{i-N}w_{i-N+1}..w_{i-1}$存于一个缓存中,作为独立的训练数据.

通过这些数据,计算动态频度分布数据$P_{dynamic}(w_i \mid w_{i-2}w_{i-1})$

将动态频度分布数据与静态分布数据(由大规模性语料训练得到)通过线性插值的方法相结合:

$$P_{combination}(w_i \mid w_{i-2}w_{i-1}) = \lambda P_{dynamic}(w_i \mid w_{i-2}w_{i-1}) + (1-\lambda)P_{static}(w_i \mid w_{i-2}w_{i-1})$$

$$0 < \lambda < 1$$

其他语言模型

各种变长、远距离N-gram模型

决策树模型

链文法模型

最大熵模型

BERT模型、ERNIE模型、ELMO模型、GPT模型……

谢谢！