自然语言处理技术基础

Natural Language Processing，NLP

网络空间安全与计算机学院

第3章 n元语法模型

3.1 n元语法的基本概念

3.2 数据平滑技术

3.3 开发和测试模型的数据集

3.4 基于词类的n-gram模型

语言模型（language model, LM）

通俗解释：判断该语言序列是否是正常语句

标准定义：计算该语言序列出现的概率

- 100个句子中有一个OK：P(OK) = 0.01
- P(An Apple ate the chicken) = 0
- P(我爱学习) > P(学习爱我)

语言模型

语言模型主要用途：

① 已知若干词，预测下一个词

② 决定哪个词序列的可能性更大

广泛应用于：机器翻译、信息检索、键盘输入、词性标注等领域。

目前主要语言模型：

- n元语法模型（n-gram model）

3.1 n元语法（n-gram）的基本概念

句子（语言序列）：$s = w_1 w_2 \cdots w_l$

共包含l个单词(各单词具有先后顺序，不要求单词之间互不相同。)

概率计算公式为：

$$p(s) = p(w_1)p(w_2|w_1)p(w_3|w_1 w_2) \cdots p(w_l|w_1 w_2 \cdots w_{l-1})$$

例如：$s = Father\ read\ a\ book$

$p(s) = p(Father|BOS)p(read|BOS\ Father)p(a|BOS\ \text{father}\ read)p(book|BOS\ \text{father}\ read\ a)p(EOS|BOS\ \text{father}\ read\ a\ book)$

3.1 n元语法（n-gram）的基本概念

存在问题： 计算代价过大

解决方案： 马尔科夫假设

$$p(s) = p(w_1)p(w_2|w_1)p(w_3|w_1w_2)\cdots p(w_l|w_1w_2\cdots w_{l-1})$$

马尔科夫假设（Markov Assumption）

一个词的出现概率仅与它之前的(n-1)个词有关：

$$p(s) = p(w_1)p(w_2|w_1)p(w_3|w_1w_2)\cdots p(w_l|w_1w_2\cdots w_{l-1})$$

$$p(s) = p(w_1)p(w_2|w_1)p(w_3|w_1w_2)\cdots p(w_l|\boldsymbol{w_{l-(n-1)}}\cdots w_{l-1})$$

$$p(\boldsymbol{s}) = \prod_{i=1}^{l} p(w_i \mid w_{i-n+1}\cdots w_{i-1})$$

$$p(s) = \prod_{i=1}^{l} p(w_i \mid w_{i-n+1} \cdots w_{i-1})$$

- 当$n = 1$时的n元语法称为一元语法（unigram）
- 当$n = 2$时的n元语法称为二元语法（bigram）
- 当$n = 3$时的n元语法称为三元语法（trigram）
- 当$n >= 4$时数据稀疏和计算代价又变得显著起来，实际工程中几乎不使用。

二元语法（bigram）一个词的出现仅依赖于它前面出现的一个词

$$p(s) = \prod_{i=1}^{l} p(w_i \mid w_{i-1})$$

{I, love}, {love, deep}, {love, deep}, {deep, learning}

二元语法模型

“Father read a book”的概率:

$$p(Father\ read\ a\ book) = p(Father| < BOS >)p(read|Father)p(a|read)p(book|a)p(< EOS > |book)$$

训练语料库S:

Father read Holy Bible.

Mother read a text book.

He read a book by grandpa.

最大似然估计（MLE）:

$$p(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})}$$

$$p(Father| < BOS >) = \frac{1}{3}$$

$$p(read|Father) = \frac{1}{1}$$

$$p(a|read) = \frac{2}{3}$$

$$p(book|a) = \frac{1}{2}$$

$$p(< EOS > |book) = \frac{1}{2}$$

$$p(Father\ read\ a\ book) = 0.06$$

数据稀疏问题

“Grandpa read a book”的概率:

$$p(Grandpa\ read\ a\ book) = p(Grandpa|<BOS>)p(read|Grandpa)p(a|read)p(book|a)p(<EOS>|book)$$

$$p(Grandpa|<BOS>) = 0 \Rightarrow p(Grandpa\ read\ a\ book) = 0$$

“Grandpa read a book.”这句话在实际生活中，是有可能出现的，概率不应为零。

出现这种情况的原因是：训练文本存在局限性和片面性

这就是**数据稀疏问题**

3.2 数据平滑技术

“数据稀疏”问题：

n元语法模型中，统计结果中出现了“零概率事件”，即：这种现象本来就不该出现；

但更多的时候，是由于模型的训练文本的规模及其分布存在着一定的局限性和片面性。

“数据平滑技术”：

为了产生更准确的概率来调整最大似然估计的技术。

基本思想就是：提高低概率（如零概率），降低高概率，尽量使概率分布趋于平均。

平滑算法，就是**劫富济贫**。

3.2 数据平滑技术

3.2.1 Laplace法则
3.2.2 Good-Turing估计
3.2.3 绝对折扣和线性折扣
3.2.4 Witten-Bell平滑算法
3.2.5 扣留估计
3.2.6 交叉校验
3.2.7 删除插值法 | 线性插值平滑
3.2.8 Katz回退算法

3.2.1 Laplace法则

拉普拉斯（Pierre-Simon Laplace，1749－1827）：法国分析学家、概率论学家和物理学家，法国科学院院士。

拉普拉斯平滑（Laplace Smoothing）：

又称 **加1平滑**。解决零概率问题。

零概率问题：

若一个词语没有在训练样本中出现，则该词语出现概率为0。

Add-one (Laplace) Smoothing

最简单、最直观、最古老的一种平滑算法：

1. 每一种情况出现的次数加1
2. 规定任何一个n-gram在训练语料至少出现一次
3. 没有出现过的n-gram的概率不再是0

$$P_{Lap}(w_1 \cdots w_n) = \frac{C(w_1 \cdots w_n) + 1}{N + T}$$

扩展：Add-k Smoothing（Lidstone's law）

应用Laplace平滑的二元文法的条件概率

序号	w_{i-1}	w_i	$C(w_{i-1}w_i)$	$P_{MLE}(w_i\|w_{i-1})$	$P_{Lap}(w_i\|w_{i-1})$
1	Tom	write	50	50 / 100=0.5	(50+1) / (100+4) ≈ 0.49
2	Tom	look	40	40 / 100=0.4	(40+1) / (100+4) ≈ 0.39
3	Tom	eat	10	10 / 100=0.1	(10+1) / (100+4) ≈ 0.11
4	Tom	read	0	0 / 100=0	(0+1) / (100+4) ≈ 0.01
总计		V=4	N=100	1	1

$$P_{MLE}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})} \qquad P_{Lap}(w_i|w_{i-1}) = \frac{C(w_iw_{i-1}) + 1}{C(w_{i-1}) + |V|}$$

$$P_{MLE}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})} \Rightarrow P_{Lap}(w_i|w_{i-1}) = \frac{C(w_iw_{i-1}) + 1}{C(w_{i-1}) + |V|}$$

$$p(Father\ read\ a\ book) = p(Father|<BOS>)p(read|Father)p(a|read)p(book|a)p(<EOS>|book)$$

$$p(Grandpa\ read\ a\ book) = p(Grandpa|<BOS>)p(read|Grandpa)p(a|read)p(book|a)p(<EOS>|book)$$

词表V = { Father, read, Holy, Bible, Mother, a, text, book, He, by, grandpa, <BOS>,<EOS> }

|V| = 13

$$p(Father\ read\ a\ book) = \frac{2}{16} \times \frac{2}{14} \times \frac{3}{16} \times \frac{2}{15} \times \frac{2}{15} = 0.00006$$

$$p(Grandpa\ read\ a\ book) = \frac{1}{16} \times \frac{1}{14} \times \frac{3}{16} \times \frac{2}{15} \times \frac{2}{15} = 0.000015$$

训练语料库S:

Father read Holy Bible.

Mother read a text book.

He read a book by grandpa.

$$p(Father|<BOS>) = \frac{1}{3}$$
$$p(read|Father) = \frac{1}{1}$$
$$p(a|read) = \frac{2}{3}$$
$$p(book|a) = \frac{1}{2}$$
$$p(<EOS>|book) = \frac{1}{2}$$

$p(Father\ read\ a\ book) = 0.06$

$p(Grandpa|<BOS>) = 0 \Rightarrow p(Grandpa\ read\ a\ book) = 0$

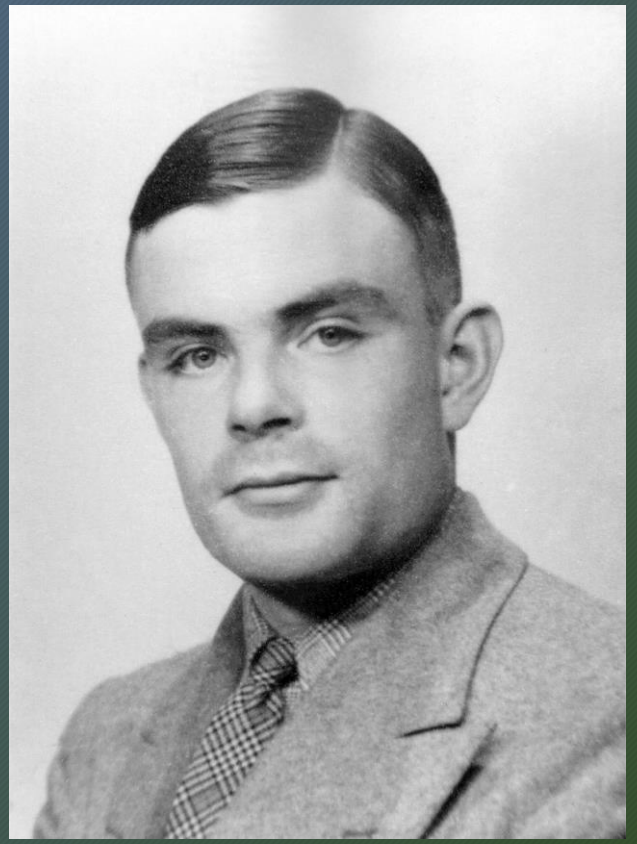
3.2.2 Good-Turing估计

- 古德-图灵估计 是 很多平滑技术的核心
- 1953年，古德（I. J. Good）引用 图灵（Turing）的方法提出

基本思想：

对于任何发生r次的n元语法，都假设它发生了r^* 次

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



应用Good-Turing估计的二元文法的条件概率

r（次数）	n_r（词数）	r^*
0	60	(0+1) × 50 / 60 = 0.83
1	50	(1+1) × 40 / 50 = 1.60
2	40	(2+1) × 30 / 40 = 2.25
3	30	(3+1) × 20 / 30 = 2.67
4	20	(4+1) × 10 / 20 = 2.50
5	10	-
总计	V=210	

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

3.2.3 绝对折扣 和 线性折扣

- 绝对减值法（Absolute discounting）
- 线性减值法（Linear discounting）

Absolute discounting

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。

r（次数）	n_r（词数）	$P_{MLE}(w)$	$P_{AD}(w)$ (b=0.1)
0	16	0	(40 - 16) × 0.1 / (50 × 16) = 0.003
1	10	0.02	(1 – 0.1) / 50 = 0.018
2	6	0.04	(2 – 0.1) / 50 = 0.038
3	5	0.06	(3 – 0.1) / 50 = 0.058
4	2	0.08	(4 – 0.1) / 50 = 0.078
5	1	0.10	(5 – 0.1) / 50 = 0.098
总计	R=40, N=50		

$$p_r = \begin{cases} \dfrac{r-b}{N} & r > 0 \\ \dfrac{b(R-n_0)}{Nn_0} & r = 0 \end{cases}$$

n_0为样本中未出现的事件的数目：16

R 为所有可能事件的数目：16+10+6+5+2+1 = 40

N 为样本中出现了r 次的事件总次数：
$n_r \times r$ = 1*10+ 2*6 + 3*5 +4*2 +5*1= 10+12+15+8+5 = 50

Linear discounting

从每个计数r中减去与该计数成正比的量，剩余概率量被未见事件均分。

r（次数）	n_r（词数）	$P_{MLE}(w)$	$P_{AD}(w)$ （b=0.1）	$P_{LD}(w)$ （α=0.1）
0	16	0	(40 - 16) × 0.1 / (50 × 16) = 0.003	0.1 / 16 = 0.00625
1	10	0.02	(1 – 0.1) / 50 = 0.018	(1-0.1) × 1 / 50 = 0.018
2	6	0.04	(2 – 0.1) / 50 = 0.038	(1-0.1) × 2 / 50 = 0.036
3	5	0.06	(3 – 0.1) / 50 = 0.058	(1-0.1) × 3 / 50 = 0.054
4	2	0.08	(4 – 0.1) / 50 = 0.078	(1-0.1) × 4 / 50 = 0.072
5	1	0.10	(5 – 0.1) / 50 = 0.098	(1-0.1) × 5 / 50 = 0.090
总计	V=40, N=50			

$$p_r = \begin{cases} \dfrac{(1-\alpha)r}{N} & r > 0 \\ \dfrac{\alpha}{n_0} & r = 0 \end{cases}$$

绝对减值法产生的n-gram 通常优于线性减值法。

3.2.4 Witten-Bell平滑算法

T. C. Bell，J. G. Cleary，I. H. Witten 1991年提出的平滑算法

是Jelinek-Mercer算法的一种特例

基本思想：

如果测试过程中一个实例在训练语料库中未出现过，它是第一次出现。

那么

可以用在训练语料库中第一次出现的实例的概率来代替未出现实例的概率。

“出现次数为零的词”分到的概率

r（次数）	n_r（词数）
0	60
1	50
2	40
3	30
4	20
5	10
总计	V=210

总数量：N = 1×50 + 2×40 + 3×30 + 4×20 + 5×10 = 350

种类数：T = 10+20+30+40+50 = 150

训练语料库看到新实例概率：T / (N+T) = 150 / (350 +150) = 0.3

0.3 / 60 = 0.005

应用Witten-Bell的二元文法的条件概率

序号	w_{i-1}	w_i	$C(w_{i-1}w_i)$	$P_{MLE}(w_i\|w_{i-1})$	$P_{WB}(w_i\|w_{i-1})$
1	father	write	50	50 ÷ 97=0.52	50 ÷ (3+97) = 0.5
2	father	look	37	37 ÷ 97=0.38	37 ÷ (3+97) = 0.37
3	father	eat	10	10 ÷ 97=0.10	10 ÷ (3+97) = 0.10
4	father	read	0	0 ÷ 97=0	3 ÷ (3+97) ÷ 2= 0.015
5	father	father	0	0 ÷ 97=0	3 ÷ (3+97) ÷ 2= 0.015
总计		V=5	N=97	1	1

3.2.5 扣留估计 (Held-out Estimation)

给语料分块，利用语料块间的差异来平滑参数空间。

算法思想:

把训练数据分成两部分,

一部分：建立最初的模型,

一部分：精炼这个模型。

扣留估计算法 举例

序号	w_i	语料a的$C(w_i)$	语料b的$C(w_i)$	r^*
1	an	20	15	(15+30) / 2 = 22.5
2	apple	20	30	(15+30) / 2 = 22.5
3	book	30	35	(35+25+25) / 3 = 28.3
4	chicken	30	25	(35+25+25) / 3 = 28.3
5	eat	30	25	(35+25+25) / 3 = 28.3
6	good	40	40	(40+35+45) / 3 = 40
7	hello	40	35	(40+35+45) / 3 = 40
8	the	40	45	(40+35+45) / 3 = 40
总计		250	250	

语料a中相同的分为一组，用该组在语料b中的“元组之和” 除以“元组个数”

3.2.6 交叉校验 （Cross Validation）

扣留估计的缺点：最初训练数据少，概率估计不可靠

改进方案：

训练数据的每一部分 即作为最初的训练数据，也作为留存数据；

对两部分数据分别进行训练和平滑。

统计学上称之为：**交叉检验**

双向交叉检验：删除估计（Deleted Estimation）

交叉校验 举例

序号	w_i	语料a的$C(w_i)$	语料b的$C(w_i)$	r^*
1	an	40	80	(280+160) / (3+1) = 110
2	apple	40	80	(280+160) / (3+1) = 110
3	book	40	120	(280+160) / (3+1) = 110
4	chicken	80	120	(240+360) / (2+4) = 100
5	eat	80	120	(240+360) / (2+4) = 100
6	good	120	80	(240+360) / (2+4) = 100
7	hello	120	160	(240+360) / (2+4) = 100
8	the	160	80	(240+120) / (3+1) = 90
9	yes	160	120	(240+120) / (3+1) = 90
10	what	160	40	(240+120) / (3+1) = 90
总计		1000	1000	

3.2.7 删除插值法 （ Deleted Interpolation ）

也称：线性插值法（ Linear Interpolation ）； Jelinek-mercer平滑

前面几种平滑算法存在的问题：

对于未出现或很少出现的n-gram都给予相同的概率估计，不太合理。

假设：

C(send the) = 0； C(send thou) = 0

采用前面的几种平滑方法，都会得到结论：

P(the | send) = P(thou | send)

但是直觉上，我们认为应该有：

P(the | send) > P(thou | send)

解决方案

改进方案：

如果(n-1)-gram本身**很少**出现，给n-gram一个**较低**的估计值；

如果(n-1)-gram是个**中等**频率，给n-gram一个**较高**的估计值。

组合估计法

- 删除插值法
- Katz回退算法

Bigram模型中的删除插值法

线性插值法（Linear Interpolation）； Jelinek-mercer平滑

第n阶平滑模型可以递归的定义为:
n阶最大似然估计模型和n-1阶平滑模型之间的差值

$$P_{Interp}(w_i|w_{i-1}) = \lambda P_{ML}(w_i|w_{i-1}) + (1-\lambda)P_{ML}(w_i) \quad , 0 \le \lambda \le 1$$

因为: $P_{ML}(the|send) = P_{ML}(thou|send) = 0$

$$P_{Interp}(the|send) = \lambda P_{ML}(the|send) + (1-\lambda)\, P_{ML}(the) = (1-\lambda)\, P_{ML}(the)$$

$$P_{Interp}(thou|send) = \lambda P_{ML}(thou|send) + (1-\lambda)\, P_{ML}(thou) = (1-\lambda)\, P_{ML}(thou)$$

因为: $P_{ML}(the) \gg P_{ML}(thou)$

$$P_{Interp}(the|send) > P_{Interp}(thou|send)$$

3.2.8 Katz回退算法

- Back-off (Katz) smoothing
- Katz平滑方法，1987

根据低一阶的分布，将从**非零计数中减去的计数量**分配给**计数量为零**的高元语法。

是Good-Turing估计方法的扩展，加入了高阶模型与低阶模型的结合。

Katz回退算法 举例

序号	w_{i-1}	w_i	$C(w_{i-1}w_i)$	$C(w_i)$	$P_{\mathrm{WB}}(w_i \mid w_{i-1})$
1	father	write	50	600	50 × 0.9 ÷ 100= 0.45
2	father	look	40	500	40 × 0.9 ÷ 100= 0.36
3	father	eat	10	300	10 × 0.9 ÷ 100= 0.09
4	father	**read**	0	150	0.1 × 150 ÷ (150+50) = 0.075
5	father	**father**	0	50	0.1 × 50 ÷ (150+50) = 0.025
总计		V=5	N=100		1

按照低一阶的分布 **C(read)**和 **C(father)**来分配

3.3 开发和测试模型的数据集

- 训练集（训练数据）
- 测试集（测试数据）

3.4 基于词类的n-gram模型

一般来说，可以将“词类”和“词性”视为相同的意思。

- **词类 Word Classes**
- **词性 Part of speech**

如果仔细区分这两个词语的意思的话，二者的意思略有区别：

- “词类”的范围更大一些
- “词性”的单位略小一些

3.4 基于词类的n-gram模型

Trigram 3-gram模型

$$P(w_3|w_1,w_2) = P(C_3|C_1,C_2)P(w_3|C_3)$$

1. 类模型提出的意义
   - ① 降低模型参数的规模
   - ② 稀疏矩阵的一种解决方式

2. 词类的构造方法
   - ① 基于词性的n-gram模型
   - ② 基于词的自动聚类的n-gram模型

Two red pencils.
Three green pencils.
Four blue pencils.
Five beautiful little birds.

$P(pencils|two, red)$、
$P(pencils|\text{three}, green)$、
$P(pencils|four, blue)$

$P(n|num, adj)$

"Four yellow pencils"
基于词:
$P(pencils|four, yellow) = 0$
基于词类:
$P(pencils|four, yellow) = P(n|num, adj)P(pencils|n)$
= (3/4) × (3/4) = 0.5625

THE END