揚州大學
YANGZHOU UNIVERSITY

自 然 语 言 处 理

L2: 语言模型

第2章

CONTENTS

01 N元语法

02 评估语言模型

03 泛化与零

04 平滑

Part.01

N元语法

揚州大學 YANGZHOU UNIVERSITY

概率语言模型

学习目标：分配一个概率给一个句子

为什么？

- **机器翻译：**
  - P(**high** winds tonite) > P(**large** winds tonite)
- **拼写更正：**
  - The office is about fifteen **minuets** from my house
  - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
- **语音识别：**
  - P(I saw a van) >> P(eyes awe of an)
- **摘要提取, 问答, 等等。**

概率语言建模

目标：计算一个句子或者一个单词序列的概率

- $P(W) = P(w_1,w_2,w_3,w_4,w_5...w_n)$

相关的任务：计算下一个单词的概率

- $P(w_5|w_1,w_2,w_3,w_4)$

一个用于计算下列两种概率的模型：

- $P(W)$ or $P(w_n|w_1,w_2...w_{n-1})$ 则被称为语言模型

如何计算 P(W)？

如何计算下面的联合概率？

- P(its, water, is, so, transparent, that)

直觉：可以使用概率的链式法则

什么是链式法则？

回顾一下条件概率的定义：

$p(B|A) = P(A,B)/P(A)$　可改写为：$P(A,B) = P(A)P(B|A)$

当有更多变量时：

$P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$

链式法则：当变量个数为n

$P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$

将链式法则用于计算包含多个单词的句子的联合概率

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

P(“its water is so transparent”) =

P(its) × P(water|its) × P(is|its water)

× P(so|its water is) × P(transparent|its water is so)

如何去计算这些概率？

是否可以仅仅去统计，并做除法，然后去计算？

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

假设词汇表的大小为V，生成长度为n的序列有多少种可能？

A) $n * V$

B) n^V

C) V^n

D) V/n

- 词汇表的大小为V
  长度为n的序列数目：V^n

- 正常英语的词汇表约等于40000
  即使长度小于11的句子可能性为：$4*10^{50}$

因此，有太多可能的句子！

我们永远无法获取到足够多的相关数据（句子）去计算这些概率。

马尔科夫假设（Markov Assumption）

Andrei Markov

简化假设：

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

或者：

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

马尔科夫假设（Markov Assumption）

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i \mid w_{i-k} \dots w_{i-1})$$

我们对每个元素在集合中的概率做了近似。

$$P(w_i \mid w_1 w_2 \dots w_{i-1}) \approx P(w_i \mid w_{i-k} \dots w_{i-1})$$

一元模型(Unigram model)

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i)$$

从一元模型中自动生成的一些句子：

```
fifth, an, of, futures, the, an, incorporated, a,
a, the, inflation, most, dollars, quarter, in, is,
mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the
```

二元模型(Bigram model)

基于前一个单词的条件概率

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

从二元模型中自动生成的一些句子：

texaco, rose, one, in, this, issue, is, pursuing, growth, in, a, boiler, house, said, mr., gurria, mexico, 's, motion, control, proposal, without, permission, from, five, hundred, fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached

this, would, be, a, record, november

N元模型(N-gram model)

基于一元模型和二元模型，我们可以扩展到三元模型，四元模型，五元模型

一般来说，这是这些有限元都是一种不充分的语言模型，为什么？

因为语言本身具有**长距离依赖性**：

The computer which I had just put into
the machine room on the fifth floor crashed.”

我们可以摆脱使用N元模型

Part.01

计算N元概率

估计二元概率

最大似然估计：

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

或写为：

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

例子

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

$$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67 \qquad P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33 \qquad P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$$

$$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5 \qquad P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5 \qquad P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$$

更多例子

Berkeley Restaurant Project sentences

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i'm looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- i'm looking for a good place to eat breakfast
- when is caffe venezia open during the day

原始二元计数

- 一共 9222 条句子

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

原始二元概率计算

- 使用二元语法标准化:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

- 结果:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

句子概率的二元估计

P(<s> I want english food </s>) =

P(I|<s>)

× P(want|I)

× P(english|want)

× P(food|english)

× P(</s>|food)

= .000031

可以从上述例子中获取那些知识？

- P(english|want) = .0011
- P(chinese|want) = .0065
- P(to|want) = .66
- P(eat | to) = .28
- P(food | to) = 0
- P(want | spend) = 0
- P (i | <s>) = .25

实际的一些问题

- 我们在对数空间(log space)中完成一切时，可以：
  - 防止下溢出(underflow)
  - 而且加法的效率要高于乘法

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

语言建模的工具包

- SRILM
  - http://www.speech.sri.com/projects/srilm/
- KenLM
  - https://kheafield.com/code/kenlm/

揚州大學 YANGZHOU UNIVERSITY

谷歌于2006年发布的N元语法模型

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.

谷歌于2006年发布的N元语法模型

在语料库中使用四元语法模型统计的一些例子：

- serve as the incoming 92
- serve as the incubator 99
- serve as the independent 794
- serve as the index 223
- serve as the indication 72
- serve as the indicator 120
- serve as the indicators 45
- serve as the indispensable 111
- serve as the indispensible 40
- serve as the individual 234

http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

Part.02

评估语言模型

如何评估一个语言模型是好的？

- 该模型偏好于好的句子还是坏的句子？
  - 赋予’真实‘或者’经常被观测到的‘句子以更高的概率
  - 赋予‘不合语法’或者‘少有被观测到的’句子以更低的概率
- 使用训练数据集(**training set**)来训练模型的参数
- 使用测试集(**test set**)来测试训练好的模型
  - 用于测试的数据是从未用于训练模型的
  - 使用评估标准(**evaluation metric**)来估计训练好的模型在测试集上的效果。

对N元模型进行对外评估

通过对比不同模型(A和B)的性能

- 将这些模型用于执行同一个任务，比如
  - 拼写修正，语音识别，机器翻译等
- 运行任务后，计算模型A和B输出的准确率，比如
  - 有多少拼写的错误被正确的修改
  - 有多少单词被准确翻译
- 最后对比模型A和B的准确率

对N元模型进行对外(对内)评估的难点

对外评估

- 耗时，通常需要几天或者几周来实现他人的工作

因此

- 有时候需要使用对内评估：**困惑度（perplexity）**
- 对内评估时需注意
  - 测试集不能类似于训练集
  - 对内评估仅仅可用于试点实验(pilot experiment)

什么是困惑度？

- The **Shannon Game**:
  - 如何去预测一个好的单词?

    I always order pizza with cheese and ____

    The 33rd President of the US was ____

    I saw a ____

  - 一元模型是不适合该游戏的. (为什么?)

mushrooms 0.1

pepperoni 0.1

anchovies 0.01

….

fried rice 0.0001

….

and 1e-100

- 能够更好预测‘好的单词’的模型
  - 为已经出现的词(上下文)分配一个概率更高的词。

什么是困惑度？

一个好的语言模型是可以最优地预测未曾‘**见过**’的测试集

使得概率P(sentence)最大化

困惑度是测试集的逆向概率，由单词总数进行归一化处理

$$PP(W) = P(w_1 w_2 ... w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 ... w_N)}}$$

链式法则：

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}$$

二元链式法则：

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-1})}}$$

困惑度最小化即是概率最大化

Shannon Game的困惑度

困惑度是等效加权分支因子

- 识别数字序列’ 0,1,2,3,4,5,6,7,8,9’ 的困惑度
  - 困惑度为 10

- 在 Microsoft 中识别30,000个名字有多难
  - 困惑度为 30,000

- 一个电话系统收到120,000个电话并且需要识别
  - ” 操作员 “ (每4个电话中出现一次)
  - ” 销售" (每 4个电话中出现一次)
  - ” 技术支持" (每 4个电话中出现一次)
  - 30,000 不同名字(在120K个电话中每个名字仅出现一次)
  - 困惑度是多少？ 下一页

Shannon Game的困惑度

我们首先通过乘以120K个概率(其中90K个1/4，30K个1/120K)来计算该长度为120K序列的困惑度，然后取它的-1/120K次方。我们可以得到：

Perp = (¼ * ¼ * ¼* ¼ * ¼ * …. * 1/120K * 1/120K * ….)^(-1/120K)

算术层面上可以简化为N = 4：操作员(1/4)，销售(1/4)，技术支持(1/4)，且一共30K个名字(1/120K)

Perplexity= ((¼ * ¼ * ¼ * 1/120K)^(-1/4) = 52.6

困惑度作为分支因子

假设一个句子有随机的多个数字组成

那么如果模型分配给每一个数字的概率P= 1/10时，当前句子的困惑度是多少？

$$
\begin{aligned}
\mathrm{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\
&= \left(\frac{1}{10}^{N}\right)^{-\frac{1}{N}} \\
&= \frac{1}{10}^{-1} \\
&= 10
\end{aligned}
$$

困惑度越低 == 模型越好

训练了3800万单词，并测试了1500万单词的WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Part.03

泛化与零

Shannon 可视化方法

根据概率选择一个随机二元组

(<s>, w)

根据概率选择一个随机二元组

(w, x)

……

直到字符</s>

最后将所有单词连接起来

```
<s> I
    I want
      want to
           to eat
              eat Chinese
                  Chinese food
                          food  </s>
I want to eat Chinese food
```

近似莎士比亚的作品

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, ’tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv’d in; –It cannot be but so.

将莎士比亚的作品集作为语料库

N = 884647个tokens，其中 V = 29066

从V^2= 8.44亿个可能的二元类型中一共生成了300K个二元类型，

- 因此大约有99.96%的二元类型没有出现在语料库中

四元模型的应用更糟：生成的句子看起来像莎士比亚因为它就是莎士比亚。

华尔街日报不是莎士比亚

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

猜一下训练集的作者？

该模型使用了三元语法生成下列句子

- They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and gram Brazil on market conditions
- This shall forbid it should be branded, if renown made it empty.
- “You are uniformly charming!” cried he, with a smile of associating and now and then I bowed and they perceived a chaise and four to wish for.

过拟合的问题

如果用于测试的语料库类似于训练集，那么N-grams仅适用于单词预测

- 在实际的模型训练中，通常不会这么做
- 需要训练一个具有泛化能力且更加稳定的模型。
- 其中一种泛化为：零
  - 也就是从未出现在训练集中
  - 单出现在测试集中

零的例子

- 训练集:
  - … denied the allegations
  - … denied the reports
  - … denied the claims
  - … denied the request

- 测试集:
  - … denied the offer
  - … denied the loan

P(“offer” | denied the) = 0

二元模型中的零概率

二元模型中出现概率为0时

- 意味着需分配P=0给测试集

因此无法计算困惑度，因为无法除以0

Part.04

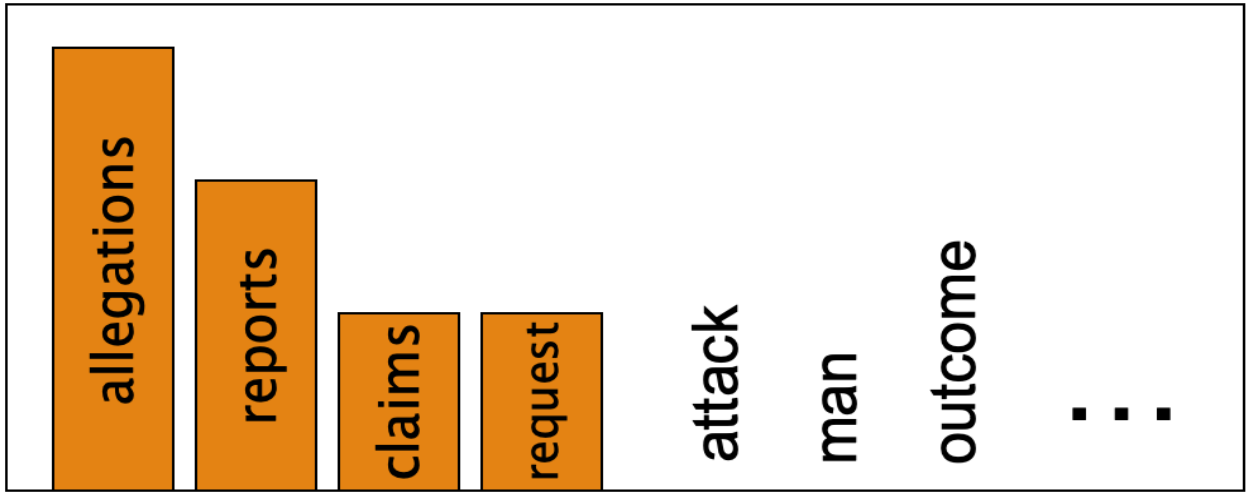
平滑

什么是平滑

当有以下稀疏的统计

P(w | denied the)
3 allegations
2 reports
1 claims
1 request

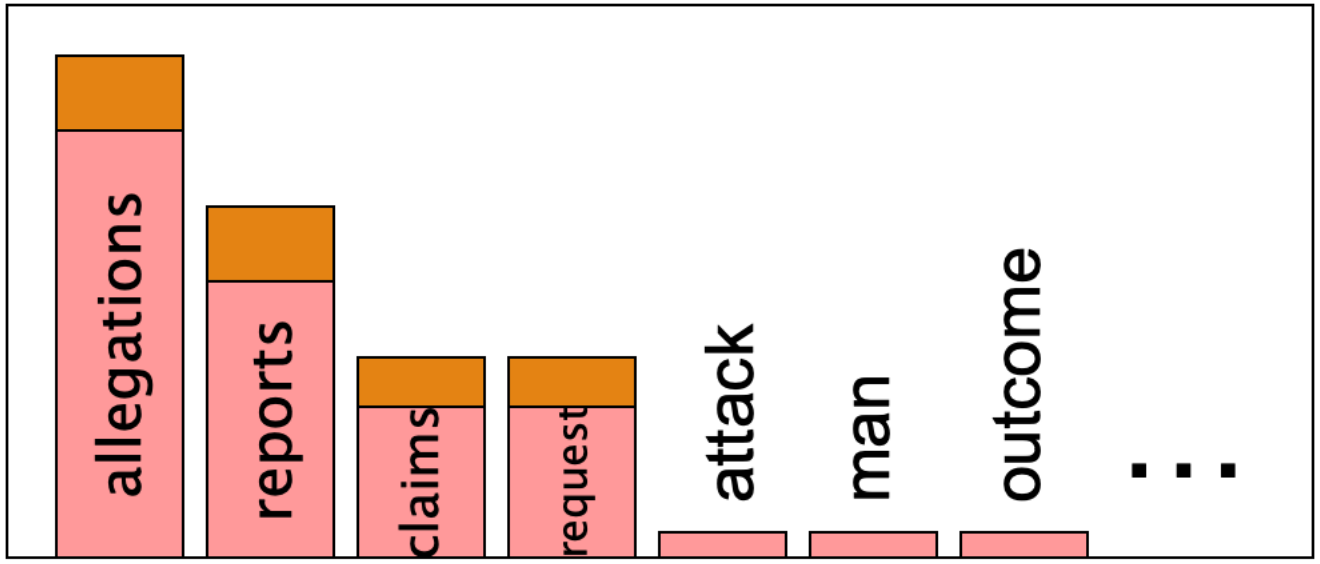
7 total



去掉部分概率质量可以更好地进行泛化

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other

7 total



加一估计

也称为拉普拉斯平滑(Laplace smoothing)

假设所有的单词次数都多一次：

- 即加一计数

语言模型估计(MLE):

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

加一估计((Add-1):

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)+1}{c(w_{i-1})+V}$$

使用拉普拉斯平滑对berkeley餐馆语料库进行单词计数

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

基于拉普拉斯平滑的二元模型

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

计数重构

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n)+1]\times C(w_{n-1})}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

对比重构前的二元计数

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

加一平滑的适用性

加一平滑适用于一些NLP模型，例如：

- 文本分类
- 或者当某些任务中零的数量不是很多时

并不适用于N元模型，

- 有更好的方法

揚州大學 YANGZHOU UNIVERSITY

Interpolation插值法

- 混合使用多种N-grams
- 例如：混合使用unigram, bigram, trigram

$$P_{int}(w_i|w_{i-1},w_{i-2}) = \lambda_3 P_{ML}(w_i|w_{i-1},w_{i-2}) + \lambda_2 P_{ML}(w_i|w_{i-1}) + \lambda_1 P_{ML}(w_i)$$

$$\lambda_3 + \lambda_2 + \lambda_1 = 1$$

Backoff回退法

- 当有足够计数时，使用N元语法，
- 否则使用N-1，N-2，…，bigrams, unigrams

$$P_{bo}(w_i|w_{i-n+1}\cdots w_{i-1}) = \begin{cases} d_{w_{i-n+1}\cdots w_i}\dfrac{C(w_{i-n+1}\cdots w_i)}{C(w_{i-n+1}\cdots w_{i-1})} & if\ \ C(w_{i-n+1}\cdots w_i) > k \\ \alpha_{w_{i-n+1}\cdots w_{i-1}} P_{bo}(w_i|w_{i-n+2}\cdots w_{i-1}) & otherwise \end{cases}$$

其中d，a和k分别为参数。k一般选择为0，但是也可以选其它的值。

回退法和插值法使用场景和对比

两种方法适用于有更少上下文的训练集

- 即以较少的上下文为条件

插值法有更好的性能

Linear Interpolation线性插值法

简易插值法

$$\hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1 P(w_n|w_{n-2}w_{n-1}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n)$$

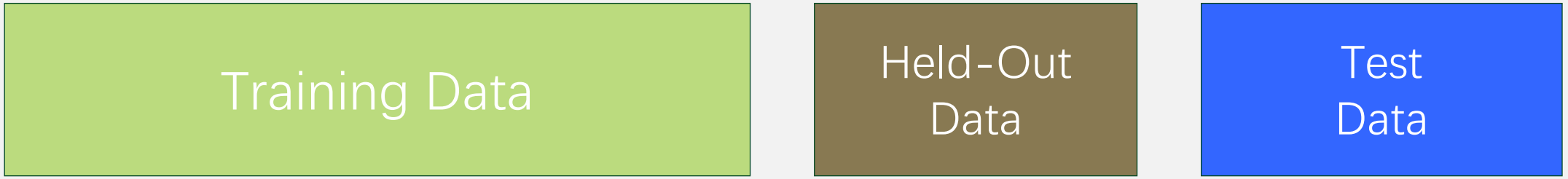
$$\sum_i \lambda_i = 1$$

基于条件上下文的lambdas

$$\hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1(w_{n-2}^{n-1})P(w_n|w_{n-2}w_{n-1}) + \lambda_2(w_{n-2}^{n-1})P(w_n|w_{n-1}) + \lambda_3(w_{n-2}^{n-1})P(w_n)$$

如何设置lambdas？

使用Held-out 语料库



选择一个 λs 来最大化held-out数据的概率

- 在训练集上修复N元概率
- 然后搜索一个可以最大化held-out数据的λs

$$\log P(w_1 \ldots w_n \mid M(\lambda_1 \ldots \lambda_k)) = \sum_i \log P_{M(\lambda_1 \ldots \lambda_k)}(w_i \mid w_{i-1})$$

计数的绝对减少：对每一个计数减去一点点

假设想从4中减去一些以保存零的概率质量，

应该减去多少？

Church 和 Gale的idea是：
分割2200万字大小的美联社数据集为

- Training set 和held-out set
- 对于训练集中的每个二元组，
- 其对应的在held-out数据集的实际计数
- 近似于c* = (c - 0.75)

Bigram count in training	Bigram count in heldout set
0	.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

绝对减少插值法(Absolute discounting)

直接减去0.75或者其他d值

discounted bigram　Interpolation weight

$$P_{\text{AbsoluteDiscounting}}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w)$$

unigram

- 或许可以为计数为1和2的N-grams保留一些额外的d值

但是否应该仅仅使用常规一元模型P(W)？

未知单词：开放式和封闭式的词汇表任务

假设提前知道词汇表上的所有单词

- 词汇表V是固定的
- 封闭式词汇表任务

通常不会收集到所有单词，即

- 词汇表以外的单词 为OOV单词(out of vocabulary)
- 开放式词汇表任务

创建一个未知的token <UNK>

- 对<UNK>的概率进行训练
  - 创建一个大小为V的固定词典L
  - 在文本归一化处理阶段，任何未在L中出现的单词都标记为<UNK>
  - 然后按照普通单词一样来训练它的概率

在解码阶段

- 若文本输入时，对于未曾出现在训练阶段的未知单词都使用<UNK>的概率进行替换。

大型互联网规模的N元模型

如何处理？比如谷歌N元语料库

剪枝(Pruning)

- 给定一个阈值(threshold)，若计数大于该阈值，则统计相应的N-grams
  - 移除低于阈值的高阶N-grams
- 基于信息熵的剪枝技术

效率(Efficiency)

- 高效的数据结构，例如使用
- 布隆过滤器：近似语言模型
- 将单词存储为索引，而不是字符串
  - 使用霍夫曼编码将大量单词匹配为两个字节
- 量化概率（4-8 位而不是 8 字节浮点数）

对互联网规模的N元模型进行平滑处理

- “Stupid backoff” 平滑处理方法 (Brants *et al*. 2007)
- 不是减少计数, 而是使用相对频率

$$S(w_i \mid w_{i-k+1}^{i-1}) = \begin{cases} \dfrac{\text{count}(w_{i-k+1}^{i})}{\text{count}(w_{i-k+1}^{i-1})} & \text{if} \quad \text{count}(w_{i-k+1}^{i}) > 0 \\ 0.4S(w_i \mid w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

$$S(w_i) = \frac{\text{count}(w_i)}{N}$$

N元平滑总结

加一平滑:

- 适用于文本分类,不适合语言建模

最长使用的平滑处理方法

- Kneser-Ney扩展插值法

对于基于大型语料库的N元模型

- Stupid backoff