上海海事大学
SHANGHAI MARITIME UNIVERSITY

自然语言处理

2024–2025 学年第 2 学期

信息工程学院　谢雨波

语言模型

概率语言模型

- 什么是概率语言模型（Probabilistic Language Model）？
- **为句子赋予概率值**
  - 机器翻译：
    - 将 “今晚有大风” 翻译为英文
    - $P(\text{high winds tonight}) > P(\text{large winds tonight})$
  - 拼写检查：$P(\text{about 5 minutes ago}) > P(\text{about 5 minuets ago})$
  - 语音识别：$P(\text{去超市买点东西}) \gg P(\text{去超时买点东西})$
  - 文本摘要，自动问答，等等

概率语言模型

- 目标：计算一个句子或是一个词序列的概率

$$P(w) = P(w_1, w_2, w_3, w_4, \cdots, w_n)$$

- 或者：下一个单词的概率

$$P(w_5 \mid w_1, w_2, w_3, w_4)$$

- 计算以下任一概率的模型：

$$P(w) \text{ 或 } P(w_n \mid w_1, w_2, \cdots, w_{n-1})$$

  叫做**语言模型（Language Model，LM）**

- 或者叫做 Grammar，但 Language Model 或 LM 更常用

如何计算 $P(w)$？

- 给定一个句子 “its water is so transparent that…”，如何计算

$$P(\text{its, water, is, so, transparent, that})$$

链式法则

- 条件概率的定义：

$$P(B\,|\,A) = \frac{P(A,B)}{P(A)} \quad \Rightarrow \quad P(A,B) = P(A)\,P(B\,|\,A)$$

- 包含更多变量时：

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

- 更一般地：

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

使用链式法则来计算句子的概率

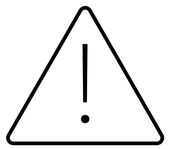
$$P(w_1 w_2 \cdots w_n) = \prod_{i=1}^{n} P(w_i \mid w_1 w_2 \cdots w_{i-1})$$

$$\begin{aligned} P(\text{“its water is so transparent”}) = \\ P(\text{its}) \times P(\text{water} \mid \text{its}) \times P(\text{is} \mid \text{its water}) \\ \times P(\text{so} \mid \text{its water is}) \times P(\text{transparent} \mid \text{its water is so}) \end{aligned}$$

如何计算每一项

- 是否可以直接计数然后相除?

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



不可行！有太多可能的句子组合！

永远不会有足够的训练数据让我们计算出 Count() 的值。

马尔可夫假设

- 简单假设：

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

- 或者：

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

马尔可夫假设

$$P(w_1 w_2 \cdots w_n) = \prod_{i=1}^{n} P(w_i \mid w_{i-k} w_{i-k+1} \cdots w_{i-1})$$

也就是说，式子中的每一项都近似为：

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

N 元语法模型（N-gram Models）

- 最简单的情形：一元语法模型（Unigram model）

$$P(w_1 w_2 \cdots w_n) = \prod_i P(w_i)$$

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

一个一元语法模型自动生成的句子:

```
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
```

N 元语法模型（N-gram Models）

- 二元语法模型（Bigram model）

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

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

一个二元语法模型自动生成的句子:

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 元语法模型（N-gram Models）

- 类似地，我们有三元语法模型、四元、五元……
- N 元语法模型是语言的一个不完全模型：
  - 自然语言通常有**长距离依赖**的性质
  - “The computer which I had just put into the machine room on the fifth floor crashed.”
- 但是，N 元语法模型对于某些场景已经够用

估算二元语法（Bigram）概率

- 最大似然估计（Maximum Likelihood Estimation）

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

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

w_{i-1}, w_i 出现的次数

w_{i-1} 出现的次数

估算二元语法（Bigram）概率

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

<s> I am Sam </s>

<s> Sam I am </s>

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

$$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = 0.67 \qquad P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = 0.33 \qquad P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = 0.67$$

$$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5 \qquad P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = 0.5 \qquad P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = 0.33$$

例子：Berkeley Restaurant Project 语料库

- 语料库中的一些例句：
  - 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

二元语法（Bigram）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

二元语法（Bigram）概率

- 通过一元语法（Unigram）进行归一化：

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

- 得到：

	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

估算语句的概率

$P(\texttt{<s>}\ \text{I want english food}\ \texttt{</s>}) =$

$P(\text{I} \mid \texttt{<s>})$

$\times P(\text{want} \mid \text{I})$

$\times P(\text{english} \mid \text{want})$

$\times P(\text{food} \mid \text{english})$

$\times P(\texttt{</s>} \mid \text{food})$

$= .000031$

实际情况中，所有计算都在 log 空间内完成：

(1) 避免浮点数下溢

(2) 同时，加法也比乘法更快

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

$$= \log p_1 + \log p_2 + \log p_3 + \log p_4$$

语言模型的评估

外在评估（Extrinsic Evaluation）

为了比较模型 A 和 B：

1. 将模型 A 和 B 放入一个真实任务中

- 机器翻译、文本分类，等等

2. 运行任务，模型 A 和 B 各得到一个分数

- 正确翻译的单词数
- 正确分类的样本数

3. 比较模型 A 和 B 的准确率

内在评估（Intrinsic Evaluation）

- 外在评估有时不可行
  - 昂贵且耗时
  - 并不总能泛化至其他应用领域
- **内在评估（Intrinsic Evaluation）：困惑度（Perplexity）**
  - 直接衡量语言模型预测单词的性能
  - 并不总是与实际应用的性能一致
  - 为语言模型提供一种单一的通用指标
  - 对于 N 元语法模型和大语言模型（LLM）都适用

训练集和测试集

- 模型参数在**训练集（Training Set）**上进行训练
- 模型性能在未见过的数据上进行测试
  - **测试集（Test Set）**是和训练集不同的、未见过的数据集
  - 衡量模型对于新数据的**泛化能力**
  - 评估标准（Evaluation Metric，例如困惑度 Perplexity）告诉我们模型在测试集上的表现如何

训练集和测试集的选择

- 如果模型针对某一特定任务：
  - 测试集应该反映此特定任务的特点
- 如果模型是通用的：
  - 训练数据应该是多种多样的
  - 我们不希望训练集或测试集只属于一个领域、作者或语言

在测试集上进行训练

- 测试数据**不应该**进入训练集中
  - 否则，当我们在测试集中看到这个句子时，语言模型就会人为地赋予它很高的概率
  - 从而给整个测试集分配一个虚假的高概率
  - 让语言模型看起来比实际情况更好
- 这种现象称为 **“数据泄露”（Data Leakage），应该避免！**

验证集

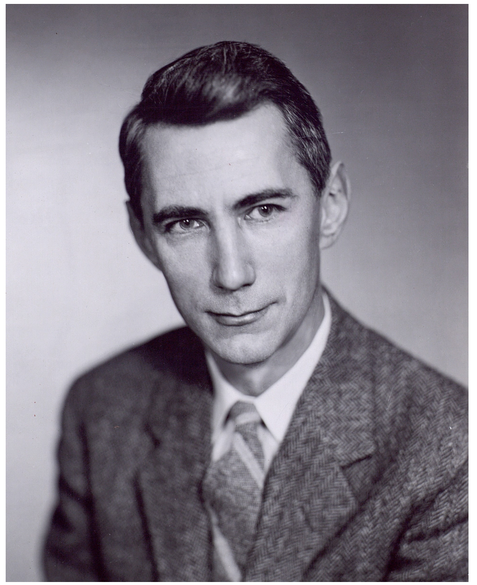
- 如果在测试集上反复测试并调整模型，那么模型最终便会学习到测试集的特征
  - 相当于在测试集上训练模型
- 因此，一般只在测试集上运行一次或几次
- 这意味着我们需要第三个数据集：
  - **验证集 / 开发集（Validation Set / Development Set）**
  - 在最终部署之前都在验证集上进行测试
  - 最终，在测试集上进行一次测试

如何评估语言模型?

- 直觉：**一个好的语言模型更偏向 “真实的” 句子**
  - 为 “真实的” 或 “经常见到的” 句子赋予更高的概率
  - 为 “虚假的” 或 “很少见到的” 句子赋予更低的概率

如何评估语言模型?

- 香农游戏：预测序列的下一个单词
  - Once upon a ____
  - That is a picture of a ____
  - For breakfast I ate my usual ____
- 一元语法模型（Unigrams）的效果如何?
- **一个好的语言模型会为实际出现的下一个词赋予更高的概率**



克劳德 · 香农

(Claude Shannon)

如何评估语言模型?

- 一个好的语言模型会为实际出现的下一个词赋予更高的概率
- 扩展到所有单词:
  - **最好的语言模型会为整个测试集赋予最高的概率**
- 比较两个语言模型 A 和 B
  - 计算 P_A(测试集) 和 P_B(测试集)
  - 更好的那个语言模型将会为测试集赋予更高的概率

如何评估语言模型?

- 概率值依赖于测试集的大小
  - 文本越长，概率值越小
  - 更好的方法：以单词为单位，按长度归一化
- **困惑度（Perplexity）**是测试集概率的**倒数**，并且按长度**归一化**

$$PPL(W) = P(w_1 w_2 \cdots w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 \cdots w_N)}}$$

困惑度（Perplexity）

- **困惑度（Perplexity）**是测试集概率的**倒数**，并且按长度**归一化**

$$PPL(W) = P(w_1 w_2 \cdots w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 \cdots w_N)}}$$

- 概率取值范围：[0, 1]；困惑度取值范围：[1, ∞]
- **最小化困惑度相当于最大化概率值**

困惑度（Perplexity）

$$PPL(W) = P(w_1 w_2 \cdots w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 \cdots w_N)}}$$

$$PPL(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i \mid w_1 w_2 \cdots w_{i-1})}} \qquad \text{（链式法则）}$$

$$PPL(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i \mid w_{i-1})}} \qquad \text{（Bigram）}$$

困惑度（Perplexity）

- 一个语言的**加权平均分支因子（Weighted Average Branching Factor）**
- **分支因子（Branching Factor）**：
  - 任意词汇后面跟着的所有可能词汇的数目
- 例如：确定性语言 L = {red, blue, green}
  - 分支因子 = 3（任意词汇后面都可以跟 red, blue, green）

困惑度（Perplexity）

- 假设语言模型 A 中，每一个词汇后面跟着任意一个其他词汇的概率均为 1/3
- 测试数据集 T = “red red red red blue”

$$PPL_A(T) = P_A(\text{red red red red blue})^{-1/5} = \left[(1/3)^5\right]^{-1/5} = 3$$

- 假设 “red” 在训练数据集中频率很高，即语言模型 B 有：

$$P(\text{red}) = 0.8, \quad P(\text{green}) = 0.1, \quad P(\text{blue}) = 0.1$$

- 那么模型 B 在 T 上的概率会更高（即困惑度更低）：

$$\begin{aligned} PPL_B(T) &= P_B(\text{red red red red blue})^{-1/5} \\ &= (0.8 \times 0.8 \times 0.8 \times 0.8 \times 0.1)^{-1/5} \\ &= 1.89 \end{aligned}$$

困惑度（Perplexity）

- 更低的困惑度 = 更好的语言模型
- 训练数据：WSJ 语料库，3.8 千万词，词典大小：19,979
- 测试数据：WSJ 语料库，1.5 百万词

	Unigram	Bigram	Trigram
困惑度	962	170	109

语言模型的采样

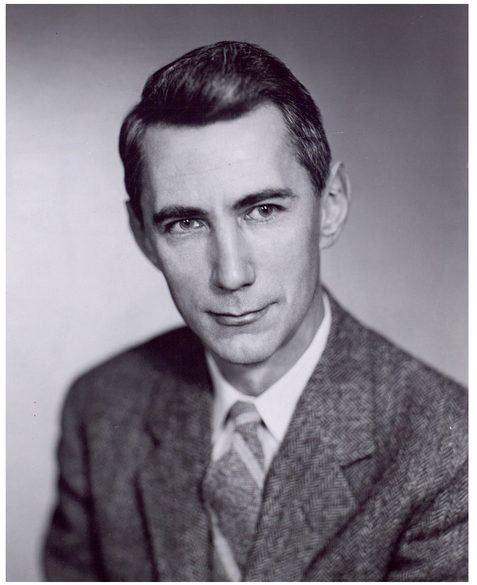
香农的语言模型可视化方法（1948）

- 从语言模型中进行词汇**采样**
- Unigram:

  ```
  Representing And Speedily Is An Good Apt Or
  Come Can Different Natural Here He The A In
  Came The To Of To Expert Gray Come To
  Furnishes The Line Message Had Be These.
  ```

- Bigram:

  ```
  The Head And In Frontal Attack On An English
  Writer That The Character Of This Point Is
  Therefore Another Method For The Letters That
  The Time Of Who Ever Told The Problem For An
  Unexpected.
  ```

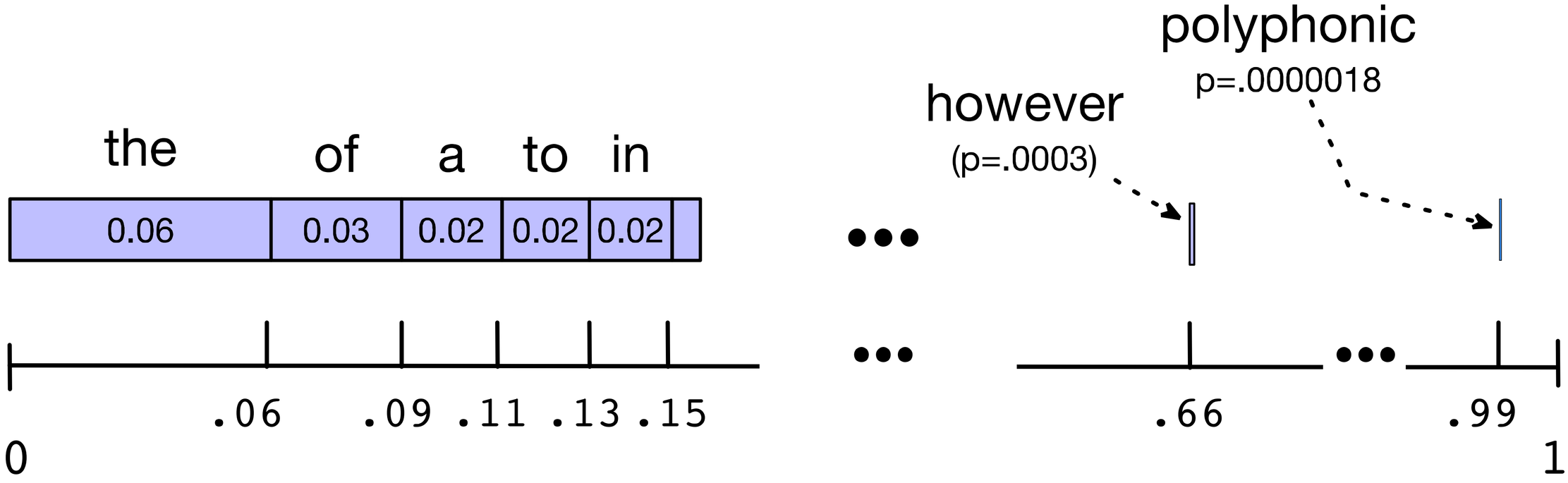


克劳德 · 香农

(Claude Shannon)

从概率分布中采样词汇

一元语法（Unigram）采样：



二元语法（Bigram）采样

- 根据 Bigram 概率 $P(w \mid \text{<s>})$ 随机采样一个 Bigram (<s>, w)
- 根据概率 $P(x \mid w)$ 随机采样一个 Bigram (w, x)
- 直至采样到 </s>
- 将所有采样的词汇串在一起

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

其他采样方法

- 针对基于神经网络的语言模型
- 可以避免从分布的尾部（概率十分小）生成词
- 之后会讲到：
  - Temperature 采样
  - Top-k 采样
  - Top-p 采样

训练数据的选择

- 如果针对特定任务，则使用和任务相似类型的训练语料库
  - 例如：法律文本、医学文本等
- 如果是通用语言模型，确保涵盖不同种类的语料库、作者等

零样本

- 只有在测试语料库和训练语料库相似时，N-gram 效果才好
  - 但是，即使是精心挑选训练集，测试集依旧会有例外情况
  - 需要训练泛化能力强的健壮模型！
- 泛化的一种：**零样本（Zeros）**
  - 在训练集中从未出现
  - 但在测试集中出现

零样本

- 训练集：
  - … ate lunch
  - … ate dinner
  - … ate a
  - … ate the
- $P(\text{breakfast} \mid \text{ate}) = 0$

- 测试集：
  - … ate lunch
  - … ate breakfast

概率为 0 的 Bigram

- 概率值为 0 的 Bigram
  - 当这些单词出现在测试集中时
  - 整个测试集的概率将是 0
- 困惑度便会无法计算!

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

未知词

- 如何处理测试集中从未见过的单词?
- **闭合词典（Closed Vocabulary）**
  - 预先知道所有可能出现的词
  - 不允许测试集中出现未知词
  - BPE 分词方法（基于神经网络的语言模型）

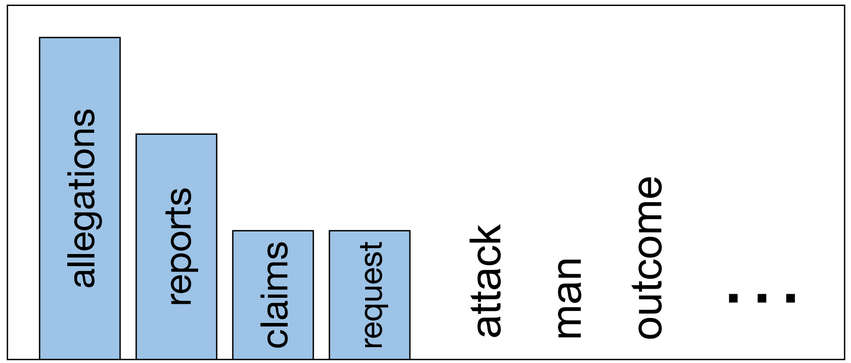
未知词

- 用特殊字 <UNK> 表示未知词（Unknown Word, Out-Of-Vocabulary Word, OOV）
- **开放词典（Open Vocabulary）**
  - 选择一个词典（词列表）并且固定
  - 将训练语料库中所有的未知词（不在词典中的词）转换为 <UNK>
  - 用同样的方式估计 <UNK> 的概率值
  - 测试时，同样把未知词转换为 <UNK>

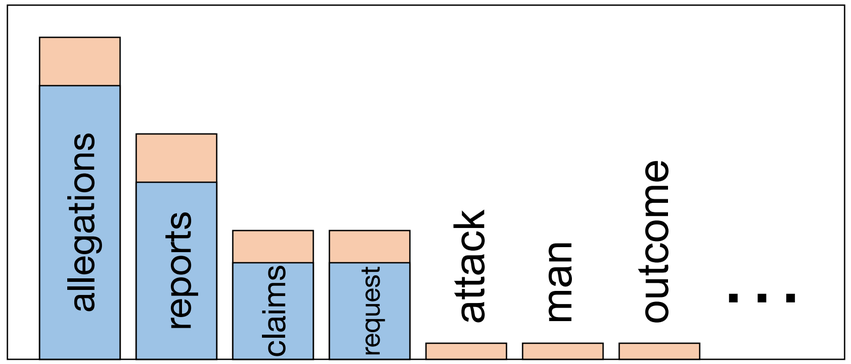
语言模型的平滑

平滑

- 稀疏的统计数据：



- “转移”一些概率质量，以增强泛化能力



加 1 平滑

- **加 1 平滑（Add-One Smoothing）也叫拉普拉斯平滑（Laplace smoothing）**
- 假装每个单词都多看到了一次，在所有的计数上加 1
- 最大似然估计：

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

- 拉普拉斯平滑：

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i) + 1}{C(w_{i-1}) + |V|} \qquad \left(= \frac{C(w_{i-1}, w_i) + 1}{\sum_{w_i \in V} [C(w_{i-1}, w_i) + 1]} \right)$$

例子：Berkeley Restaurant Project 语料库

- 语料库中的一些例句：
  - 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

二元语法（Bigram）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

拉普拉斯平滑之后

	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

拉普拉斯平滑之后

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

	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

计数重组

$$C^*(w_{i-1}, w_i) = \frac{[C(w_{i-1}, w_i) + 1] \times C(w_{i-1})}{C(w_{i-1}) + |V|}$$

	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

平滑前后的计数比较

	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

折扣（Discount）：

$$d_C = \frac{C^*}{C} = \frac{8.2}{82} = 0.1$$

加 k 平滑

- “转移” **稍少一些**的概率质量至未见过的事件
- **加 k 平滑（Add-k Smoothing）**：

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

加 k 平滑

- 加 k 平滑一般不适用于 N-gram 模型
  - 可能会导致零样本似然的过度估计，从而影响语言模型的准确度
- 但是加 k 平滑可以用于平滑其他 NLP 模型
  - 文本分类
  - 以及其他零样本现象不严重的任务

避退与插值

避退

- 有时考虑更少的上下文效果更好
- **避退（Backoff）**：
  - 如果 Trigram 有训练样本，则使用 Trigram 来估计
  - 否则使用 Bigram 来估计；否则使用 Unigram

$$P(w_i \mid w_{i-2}w_{i-1}) \longleftarrow P(w_i \mid w_{i-1}) \longleftarrow P(w_i)$$

插值

- **插值（Interpolation）**：
  - 混合 Unigram, Bigram, Trigram
  - 效果比避退好

$$\hat{P}(w_i \mid w_{i-2}w_{i-1}) = \lambda_1 P(w_i \mid w_{i-2}w_{i-1}) + \lambda_2 P(w_i \mid w_{i-1}) + \lambda_3 P(w_i)$$

简单插值

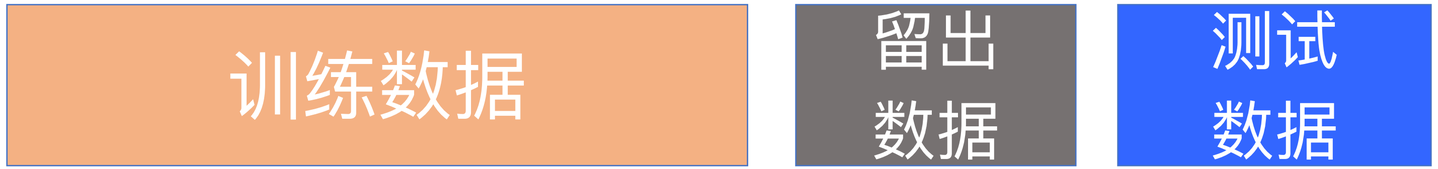
$$\sum_i \lambda_i = 1$$

$$\hat{P}(w_i \mid w_{i-2}w_{i-1}) = \lambda_1(w_{i-2}^{i-1})\, P(w_i \mid w_{i-2}w_{i-1}) + \lambda_2(w_{i-2}^{i-1})\, P(w_i \mid w_{i-1}) + \lambda_3(w_{i-2}^{i-1})\, P(w_i)$$

条件插值

λ 如何设定?

- 使用一个留出数据集（Held-Out Data）



- 选择使得留出数据概率最大的 λ
  - 在训练数据上，确定 N-gram 的概率值
  - 搜索 λ，使得留出数据的概率值最大：

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

避退 + 平滑

- 如果只是简单地进行避退，那么高元语法的概率相加将大于 1！
- 需要对高元语法进行平滑
- **Katz 避退（Katz Backoff）**：

$$P_{BO}(w_i \mid w_{i-n+1}^{i-1}) = \begin{cases} \alpha(w_{i-n+1}^{i-1}) \cdot P_{BO}(w_i \mid w_{i-n+2}^{i-1}), & \text{如果 } C(w_{i-n+1}^{i-1} w_i) = 0 \\ \hat{P}(w_i \mid w_{i-n+1}^{i-1}), & \text{否则} \end{cases}$$

其中，$\hat{P}$ 是原始估计概率；α 是避退权重，确保概率和为 1

大规模语言模型

- 在互联网或其他大量文本数据上，构建大规模语言模型：
  - Web 1 Trillion 5-gram (Google)
    - 英语网页，1,024,908,267,229 个单词
  - Google Books Ngrams
    - 8 种语言的书籍，8 千亿个 Token
  - COCA (Corpus of Contemporary American English)
    - 美式英语，10 亿单词

4-gram	Count
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

Google Web 4-gram

效率优化

- **存储优化**：
  - 内存中单词用 64 位哈希值表示，实际的单词存储在硬盘中
  - 概率值使用 4-8 位量化（而不是 64 位浮点数）
  - 使用 Trie 结构存储 N-gram
- **模型剪枝（Pruning）**：
  - 只存储计数大于阈值的 N-gram
  - 基于熵值的剪枝

Stupid Backoff

- 适用于大规模语言模型的一种简单避退算法：

$$S(w_i \mid w_{i-n+1}^{i-1}) = \begin{cases} \dfrac{C(w_{i-n+1}^{i-1} w_i)}{C(w_{i-n+1}^{i-1})} & \text{如果 } C(w_{i-n+1}^{i-1} w_i) > 0 \\ 0.4 S(w_i \mid w_{i-n+2}^{i-1}) & \text{否则} \end{cases}$$

（非概率分布！）

$$S(w_i) = \frac{C(w_i)}{N}$$