

# 自然語言處理實作 (2003-)

## Ngram 語言模型

張俊盛 [jason@nlplab.cc](mailto:jason@nlplab.cc)


助教：郭俊豪 [isaac@nlplab.cc](mailto:isaac@nlplab.cc)

許瑋芩 [winnie@nlplab.cc](mailto:winnie@nlplab.cc)

課程網站：<https://elearn.nthu.edu.tw/enrol/index.php?id=8875>

2020 1008 每星期四 15:30 到作業做出來 資電館 323

# 機率語言模型 Probabilistic Language Models

- 目的：自然語言系統輸出句子的機率值 
  - 機器翻譯 Machine Translation:
    - $P(\text{strong winds tonight}) > P(\text{high winds tonite})$
  - 拼字改錯 Spell Correction: *The office is about fifteen **minuets** from my house*
    - $P(\text{about fifteen minutes from}) > P(\text{about fifteen minuets from})$
  - 語音辨識 Speech Recognition
    - $P(\text{I saw a van}) \gg P(\text{eyes awe of an})$
  - 摘要 Summarization
  - 自動問題 Question-answering 等等

strong/high winds/wind



?

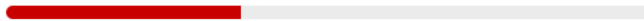
N-gram

Percent

Count

Example

high winds

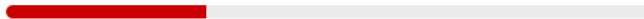


36.5 %

236,920

Show

strong winds

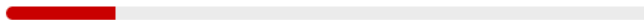


31.1 %

201,933

Show

strong wind



17 %

110,591

Show

high wind



15.4 %

99,678

Show

# 語言模型的數學推導

- 目的：計算句子、詞、詞串的機率

$$P(W) = P(w_1, w_2, w_3, w_4, w_5, \dots, w_n) \quad n \text{ 詞句子的機率}$$

- 中間步驟：計算下個詞的機率（以前面的詞為條件）

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

- 計算全句或下一詞的方式，都叫做「語言模型」 language model

$$P(W) \quad \text{or} \quad P(w_n | w_1, w_2 \dots w_{n-1})$$

- 甚至我們可以說是語言模型就是「文法」 cover grammar (用機率涵蓋合法句)

## 如何計算整句或 $n$ -連詞 $w$ 的機率 $P(w)$

- 如何計算任意長度“its water is so transparent that” 的機率？
  - $P(\text{its, water, is, so, transparent, that})$
- 直覺
  - 不容易取到太長的一串詞的樣本（6個詞很難重覆）
  - 我們需要用 Chain Rule 來簡化長串詞的機率計算
  - 固定長度，不太長（容易取樣） 不太短（保留相依關係）

## 還記得 Chain Rule

- 先看看條件機率的定義（用聯合機率來定義）

$$P(B|A) = P(A,B) / P(A) \quad \text{改寫: } P(A,B) = P(A) \times P(B|A)$$

- 從 A, B，推廣到 A, B, C, D

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

- 廣義的 Chain Rule（推廣到  $x_1, x_2, x_3, \dots, x_n$ ）

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

## 用 Chain Rule 把聯合機率化減為單詞機率的乘積

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

所以  $P(\text{its water is so transparent}) =$

$P(\text{its}) \times P(\text{water} \mid \text{its})$

$\times P(\text{is} \mid \text{its water}) \times P(\text{so} \mid \text{its water is})$

$\times P(\text{transparent} \mid \text{its water is so})$

# 如何用樣本估算這些單詞條件機率

- 我們就計算樣本（訓練資料）中的詞彙、n-連詞的次數，然後相除

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

$\frac{\text{Count}(\text{its water is so transparent that the})}{\text{total \# of all words}}$

$\frac{\text{Count}(\text{its water is so transparent that})}{\text{total \# of all words}}$

- 不可行—事件太多，而樣本太少 (語料庫有限，組合是天文數字)
- 事件常不在訓練資料中 (次數為 0) 我們無法可靠的估算次數、機率值

很危險！會把整個機率連乘式子變成0



# 馬可夫假設 Markov Assumption



- 可以假設只需前面 1 詞做條件

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

- 或許假設只需前面 2 詞做條件

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

# 馬可夫假設

- 假設詞  $w_i$  只和前  $k$  詞  $w_{i-k} \dots w_{i-1}$  有關 (不必從頭開始)

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

- 換言之，乘積的每一項都簡化

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

## N-連詞模型 N-gram models

- 我們可以擴充到 trigrams, 4-grams, 5-grams (愈來愈合文法)
- 但嚴格來說，這些都不是「語言」的充分有效模型
  - 因為，語言長距離相依性 **long-distance dependencies**  
(例如，下句中的 is 或 are 依賴前面的 computer 單複數)
  - The computer(s) which I had just put into the machine room on the fifth floor is (are) crashing.
- 但是，大部分的時候用 N-連詞模型就夠了

## 基本的雙連詞機率估算– Maximum Likelihood Estimate, MLE

$$P(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})} \quad \text{或簡寫} \quad P(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

# MLE Bigram 的計算實例

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

假設的樣本（語料庫）

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

代表句子開頭的符號

代表句子結束的符號

$$P(I | <s>) = \frac{2}{3} = .67$$

$$P(\text{Sam} | <s>) = \frac{1}{3} = .33$$

$$P(\text{am} | I) = \frac{2}{3} = .67$$

$$P(</s> | \text{Sam}) = \frac{1}{2} = 0.5$$

$$P(\text{Sam} | \text{am}) = \frac{1}{2} = .5$$

$$P(\text{do} | I) = \frac{1}{3} = .33$$

## 訓練資料：Berkeley Restaurant Project (9,222 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

# 雙連詞的原始次數 (Berkeley Restaurant Project 9,222 句)

$c(\text{want}) = 927$ ,  $c(\text{want to}) = 686$ ,  $P(\text{to} | \text{want}) = ?$

	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

# 雙連詞一次數除以單詞次數轉成機率 (MLE)

$c(\text{want}) = 927$ ,  $c(\text{want to}) = 686$ ,  $P(\text{to} | \text{want}) = 0.66$

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

	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



## 有了雙連詞機率就可以估算句子的機率

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

$$P(I | <s>)$$

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

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

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

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

$$= .000031$$

## 用到的知識—表中的條件機率

- $P(\text{english} | \text{want}) = .0011$
- $P(\text{chinese} | \text{want}) = .0065$
- $P(\text{to} | \text{want}) = .66$
- $P(\text{eat} | \text{to}) = .28$
- $P(\text{food} | \text{to}) = 0$
- $P(\text{want} | \text{spend}) = 0$
- $P(i | \langle s \rangle) = .25$

## 實際的考慮

- 把機率轉對數值  $\log$  來計算（機率相乘 = 對數相加）

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

- 好處
  - 加法計算比較快
  - 避免機率愈乘愈低，低於電腦預設的有效位數 underflow

# 可以用 LM 工具來做比 MLE 更好的機率估算

- SRILM

- <http://www.speech.sri.com/projects/srilm/>

- KenLM

- <https://kheafield.com/code/kenlm/>

# 語言模型

如何評估語言模型？

# 那資料呢？有現成的嗎？Google Web 1T N-grams

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.

# Web 1T N-grams 的 4-連詞範例

- 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 indispensable 40
- serve as the individual 234

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

# Google 的各種 N-grams

- Web 1T N-grams 是 2006 那一年的英文網頁資料
- 有沒有其他語言？
  - 有（中文、日文、歐洲10語言）
- 有沒有印刷術發明 500 年來，人類出版的書籍的資料
  - 有各種語言的書籍
  - 叫做 Google Book N-grams <http://ngrams.googlelabs.com/>



# 語言模型

如何評估語言模型？

# 評估：這個模型多有效？

- 模型是否可以偏好「好的句子」而排斥「不好的句子」
  - 真實發生，常見的句子應該得到高機率
  - 不合文法，少見的句子應該得到低機率
- 我們用訓練資料學習模型中的參數 parameters（準備）
- 重要：必須用另外的資料來評估模型的表現（考試）
  - 測試資料必須完全和訓練資料無關
  - 量化評估指標測試
    - 用機率函數本身來評估（內在評估）
    - 用某一任務來評估（外部評估）

# 用訓練資料來測試？

- 不能用訓練資料來測試
- 如果那麼做，模型會給測試資料不正常的高機率
- 這樣做「不科學」也「不誠實」（偷看考卷）

# N-連詞機率的外在評估

- 評估模型 A 和模型 B 的最好方式
  - 把兩種模型用在某一任務上
    - 拼字改錯、語音辨識、機器翻譯
  - 執行任務，看看兩種模型的表現
    - 比較改正有多少錯字
    - 比較正確地翻譯了多少字（或句子）
  - 比較上述正確率

# 外部、實務的評估上的困難

- 外部評估太曠日費時
- 替代方案-內在評估（任務無關）
  - 我們常常用自身、內在的評估方式：複雜度 perplexity
  - 不是非常好的方法
    - 除非測試資料和訓練資料非常接近
    - 所以，一般而言只用來作初步實驗、初步評估
  - 不論如何，考慮複雜度是有用的

## 內在評估：複雜度的簡單解釋

- Shannon 實驗（用一組真實例句）給系統猜

- 猜下一個詞時，很準確嗎？

I always order pizza with cheese  
and \_\_\_\_\_

The 33<sup>rd</sup> President of the US was

- 用單詞模型表現不佳（沒有利用上下文關連）
- 要怎樣才算好表現（預測得比較準確）
  - 指定高機率給真正出現的下一個詞

mushrooms  
0.1

pepperoni 0.1

anchovies 0.01

....

fried rice  
0.0001

....

and 1e-100

## 複雜度 Perplexity

- 複雜度 = 句子機率倒數，再用詞數正規化
- 好模型能預測未見過測試句，給予高機率
- 降低複雜度 = 提高機率值

Chain rule: For bigrams:

$$\begin{aligned} PP(W) &= P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}} \end{aligned}$$

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

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

複雜度也可以看成接下來（平均）可能出現幾個不同的詞

- 假設句子由 10 字組成
- 如果雙詞機率是平均分布，每個下一詞的機率  $P = 1/10$
- 那麼複雜度就是 10 因為下一詞有 10 個可能

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



複雜度愈低 = 模型愈精確

- 用華爾街日報 3 千 8 百萬詞訓練，1 百 50 萬詞測試
- 愈高階，複雜度愈低，模型愈精確
- 第 3 階相當合理，再高階或許有資料不足的問題

各階N-連詞模型	Unigram	Bigram
複雜度	962	170

## 如何看見 (visualize) 語言模型LM—觀察用 N-連詞產生隨機句子 (Shannon)

- 依機率，隨機選擇雙連詞 (<s>, w) ~~w~~**s** > I
  - 接著，隨機選擇另一雙連詞 (w, x) I want
  - 直到，我們隨機產生 </s> want to
  - 最後，把所有詞串起來成為句子 to eat
- eat Chinese
- Chinese
- food
- food </s>
- I want to eat Chinese food

## 語言模型

反向操作：用 LM 產生文字

# Approximating Shakespeare

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.

# 莎士比亞全集語料庫（百萬詞、三萬詞彙表）

- $N=884,647$  tokens,  $V=29,066$
- 看看雙連詞
  - 可能的組合  $V^2= 844$  百萬（差不多九億）
  - 莎士比亞用了 300,000 不同的雙連詞 (0.04%)
  - 不用其中的 99.96% (表中的次數 0)
- 三連詞、四連詞 0 次數的比例更高
- 用雙連詞產生隨機句子—很像莎士比亞寫的
  - 因為那都是莎士比亞寫的少數單詞、雙連詞

## 華爾街日報就很不像 Shakespeare

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

## 猜猜這些隨機三連詞句子的作者是誰？

- 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.

## 語言模型

處理沒看過的  $n$ -連詞

沒看過就失敗——流於死背



# 適者生存，但是過度適應 overfitting 也很危險

- N-連詞模型預測詞很有效，如何測試和訓練很接近
  - 實際上，測試和訓練不容易接近
  - 我們需要比較強韌的模型，讓模型可以推廣到見過的情況
  - 其中的一種狀況：0 次
    - 未出現在訓練資料（所以 MLE 會失敗）
    - 但是出現在測試資料（所以需要處理）

# 零和非零顯示的過度適應

- 訓練資料集

- ... denied the allegations
  - ... denied the reports
  - ... denied the claims
  - ... denied the request

- 測試資料集

- ... denied the offer
  - ... denied the loan

如果用 MLE

$P(\text{"offer"} \mid \text{denied the})$   
 $= 0$

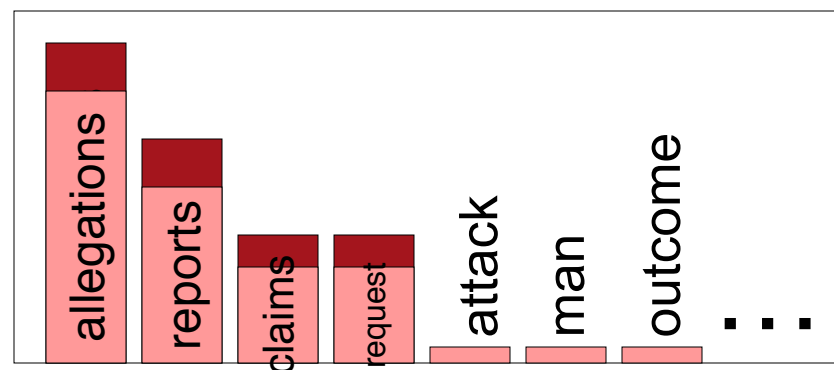
# 機率為 0 的雙連詞

- 0 次的雙連詞，其機率為 0 (根據 MLE)
  - 不浪費機率值給未出現在訓練資料集—訓練資料集的MLE機率最佳
  - 那麼測試
    - 資料集的其中一筆，機率為 0
    - 整個資料集的機率為 0
- 因此，計算機率、複雜度時就會出問題 (不能除以 0)

## 平滑化的簡單概念 (非 0 的減一點，0 次的加一點點)

- 我們有資料稀疏的問題 (很多 0 次)
- 偷一點機率總值，讓模型比較有普遍性

$P(w \mid \text{denied the})$        $P(w \mid \text{denied the})$



3 allegations	2.5
2 reports	allegations
1 claims	1.5 reports
1 request	0.5 claims
	0.5 request
7 total	2 other

## 語言模型

似乎有理的簡單作法：加 1

# 全部加 1 估算 Add-one estimation

- 又稱 Laplace smoothing
- 假裝每個事件次數都少一次，我們因此加 1

- MLE 估算: 
$$P_{MLE}(w_i | w_{i-1}) = \frac{\alpha(w_{i-1}, w_i)}{\alpha(w_{i-1})}$$

- Add-1 估算: 
$$P_{Add-1}(w_i | w_{i-1}) = \frac{\alpha(w_{i-1}, w_i) + 1}{\alpha(w_{i-1}) + V}$$

# 最大似然估計量 Maximum Likelihood Estimates (MLEs)

- 最大似然估計法
  - 最佳化訓練資料的機率（根據模型）
  - 所有模型中，這個模型給訓練資料最高的機率值
- 假設 bagel 在百萬詞語料庫中，這個詞出現 400 次
- 那麼，在其他資料中，隨機一詞剛好是 bagel 的機率是多少？
- 根據 MLE，機率為  $400/1,000,000 = .0004$
- 或許，這個估算對其他語料庫不好
- 但是，對於訓練資料的樣本（bagel 出現百萬詞語料庫中 400 次）最好，機率最高

## 再看看 Laplace 平滑化 (Berkeley Restaurant Corpus)

	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



## 用 Laplace平滑化後之雙連詞機率

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

	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

重算次數  $C^*(w_{n-1} w_n) = P(w_n | w_{n-1}) \times C(w_{n-1})$

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

	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

608 -> 238 調降太多！

# 比較原始與重建次數 (正規化後，減太多)

	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-1 估算太粗糙

- 所以，不能用「加 1 估算法」
- 看看有沒有更好的方法
- 但是「加 1 估算法」用於其他 NLP 模型
  - 文件分類 text classification
  - 其他 0 次事件不多的情況

# 語言模型的應用

文件分類、易讀性分級

# 文件分類

- 可以用簡易貝氏分類器 (Naive Bayse Classifier)

$$\begin{aligned} p(C_k | x_1, \dots, x_n) \\ p(C_k | \mathbf{x}) &= \frac{p(C_k) p(\mathbf{x} | C_k)}{p(\mathbf{x})} \\ &= p(C_k) \prod_{i=1}^n p(x_i | C_k) \end{aligned}$$

- 其實基本上，和某一分類 + Unigram LM 很類似

- $C_k$ 
$$P(w_1, \dots, w_m) = \prod_{i=1}^m P(w_i | w_1, \dots, w_{i-1}) \approx \prod_{i=1}^m P(w_i | w_{i-(n-1)}, \dots, w_{i-1})$$

## 文件分類資料集：Reuters-21578

- <http://www.daviddlewis.com/resources/testcollections/reuters21578/>

- 例子

<TOPICS><D>cocoa</D></TOPICS>

<TITLE>BAHIA COCOA REVIEW</TITLE>

the Bahia cocoa zone, alleviating the drought since early January and improving prospects for the coming temporaao, although normal humidity levels have not been restored, Comissaria Smith said in its weekly review.

# 易讀性分類資料集：English Voc/Grammar Profile

- **English Vocabulary Profile: 從單字的角度：** [www.englishprofile.org/wordlists](http://www.englishprofile.org/wordlists)
  - earn v. A2 She **earns** about \$50,000 a year.
  - earn v. A2 We can wash cars to **earn** some money for the necessary materials.
  - earn v. B2 You can't expect to **earn** a living from your painting.
  - earn v. B2 We work to **earn** a living.
  - earn v. C2 As a teacher you have to **earn** the respect of your students.
  - earn v. C2 In Japan they always had to keep a certain image to **earn** the respect they deserved but in that house they could set free their emotions.
- **English Grammar Profile: 從文法的角度** [www.englishprofile.org/english-grammar-profile](http://www.englishprofile.org/english-grammar-profile)
  - A and A C2 A timid, shy, self-conscious, over-sensitive and vulnerable person can yearn to make friends with someone who is very **self-assured, confident, decisive, even bossy.**



# 語言模型

退後與內插

# 退後與內插 Backoff and Interpolation

- 有時候，可以用比較少的前文 context
  - 如果沒有太多資料，不如減小前文 (透過「退後」與「內插」)
- 退後 Backoff:
  - 如果有證據（資料）那就使用三連詞機率 trigram  $i$
  - 否則，退到雙連詞，如果還沒有資料，退到單詞機率
- 內插 Interpolation:
  - 用單詞、雙連詞、三連詞的機率的加總
- 內插優於退後
-

## 線性內插 linear interpolation

- 簡單內插（權重為常數）

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

- 函數型內插（視文脈決定權重）

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

# 如何設定權重函數?

- 使用保留資料 **held-out data**

Training Data

Held-Out

Test

- 調整  $\lambda$  函數，使得保留資料的機率最大化
  - 固定N-連詞機率 (在訓練資料中)
  - 找一組  $\lambda$  使得保留資料的機率值最大化 (以雙連詞機率為例)

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

# 未知詞 Unknown 的處理：開放與封閉詞彙集

- 如果我們事先知道所有涉及的詞彙—封閉詞彙集任務
  - 詞彙表  $V$  是固定的
- 通常，我們不知道所有涉及的詞彙—開放詞彙集任務
  - 那就會有未知詞 OOV 詞 **Out Of Vocabulary**
- 與其處理一個個未知詞，我們可以規定一個代表未知詞符號 <UNK>
- 用訓練資料，估算 <UNK> 的機率
  - 先做一個大小為  $V$  的固定詞彙表  $L$  在文字正規化階段，把不在  $L$  的詞彙，改為 <UNK>
  - 用正常的方式，估算機率函數
- 在執行階段
  - 如果在處理資料時，碰到非  $L$  詞彙，以 <UNK> 來計算機率值

# 超大規模網路 n-連詞

- 如何運用如 Google Web 1T 5-grams (百萬乘百萬個樣本，一兆， $10^{12}$ 次方)
- 刪減 pruning
  - 只儲存次數大於等於門檻 (40次)
  - 高階 n-連詞：使用 <UNK> 較低的門檻
  - 利用熵值的刪減 ( ? )
- 效率
  - 為了快速檢索，常用 tries 資料結構，用 Bloom filters 取 LM 近似值
  - 用編號取代詞的字串 (甚至用 Huffman 編碼，用較少平均字元數)
  - 機率量子化 quantize probabilities—犧牲精確度
    - 用 4-8 bits 整數，不用 8-byte 的浮點實數)

# 網路 N-連詞機率的平滑化

- 資料那麼大，用簡單的平滑化 stupid backoff 就可以了 (Brants *et al.* 2007)
- 不需要減次數，直接用相對頻率 + 後退法

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

$$\mathcal{S}(w_i) = \frac{\text{count}(w_i)}{N} \quad (\text{退到最後})$$

# 平滑化小結

- 加-1 平滑化
  - 不能用在語言模型，或許文件分類還可以
- 最常用的方法
  - Kneser-Ney 的內插法的延伸
- 對於超大型網路n-連詞
  - Stupid backoff



語言模型

進階語言模型

# 進階語言模型

- 判別模型 Discriminative models:
  - 調整機率值 (或權重) 來改善某一任務，而非適應訓練資料集
- 剖析為本的語言模型 (?)
- 快取型的語言模型
  - 配合資料的區域性（最近出現的詞，比較容易再出現）

$$P_{CACHE}(w | history) = \lambda P(w_i | w_{i-2} w_{i-1}) + (1 - \lambda) \frac{\alpha(w \in history)}{|history|}$$

- 但是在語音辨識中，成效不佳 (why?)

# 減固定次數 absolute discounting (非0者小量減少)

- 假設我們想減雙連詞的次數分給 0 次者
- 我們應該減多少?
- Church and Gale (1991) 有個巧妙想法
- 把 2 千 2 百萬的美聯社新聞分成
  - 訓練集、保留集
  - 對訓練集的某一次數 ( $c = 1, 8$ ) 的雙連詞
  - 統計在保留集次數的平均值  $c^*$
- 看起來很有規律  $c^* = c - 0.75$  ( $c > 1$ )
- 或者  $c^* = c - 0.51$  ( $c = 1$ )

訓練集原次數	保留集平均次數
0	0.0000270
1	0.49
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21

## 固定減次內插法

- 簡單就一致減 0.75 (或某一固定  $d$  值)

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

**discounted bigram**   **Interpolation weight**  
**unigram**

- (或許可以視  $C(w_{i-1}, w_i) = 1, 2$  減不同的  $d$  值)
- 然後加上原本的  $P(w_i)$

# 更複雜的減次內插法 Kneser-Ney Smoothing I

- 應該有比低階單詞機率更好的內插值
  - Shannon game: *I can't see without my reading*\_\_\_\_\_?
  - “Francisco” 的單詞機率比 “glasses” 高
  - 但是 “Francisco” 常常在 “San” 後出現，所以比較不可能
- 不該用  $P(w)$  — 因為  $P(w)$  有利於 seen 而不利於 unseen
- 所以問題不是「 $w$  是否比較可能」，而是  $w$  是否易出現在前一詞的 unseen ngram
- $P_{CONTINUATION}(w) = w$  是否容易造成不常見的下一詞 novel continuation
  - 對每一詞  $w$  計算  $w$  出現的雙連詞  $(x, w)$  種數 (types)
  - 每個雙連詞第一次出現都是 novel continuation

Francisco  
glasses

$$P_{CONTINUATION}(w) \propto |\{w_{i-1} : \alpha(w_{i-1}, w) > 0\}|$$

# Kneser-Ney Smoothing (續)

- 詞  $w$  以新的下一詞出現

$$P_{CONTINUATION}(w) \propto |\{w_{i-1} : \alpha(w_{i-1}, w) > 0\}|$$

- 用雙連詞種數  $|\{(w_{j-1}, w_j) : \alpha(w_{j-1}, w_j) > 0\}|$  來除 (正規化)

$$P_{CONTINUATION}(w) = \frac{|\{w_{i-1} : \alpha(w_{i-1}, w) > 0\}|}{|\{(w_{j-1}, w_j) : \alpha(w_{j-1}, w_j) > 0\}|}$$

# Kneser-Ney Smoothing (續)

- 另外說法： $w$  的前一詞  $w_{i-1}$  的種數  $|\{w_{i-1} : \alpha(w_{i-1}, w) > 0\}|$

- 用前一詞的個數來做平滑化 
$$P_{CONTINUATION}(w) = \frac{|\{w_{i-1} : \alpha(w_{i-1}, w) > 0\}|}{\sum_{w'} |\{w'_{i-1} : \alpha(w'_{i-1}, w') > 0\}|}$$

- 高頻詞 (如 Francisco) 前一詞，都是 San 所以 Francisco 的連續機率低

# Kneser-Ney Smoothing (續)

$$P_{KN}(w_i | w_{i-1}) = \frac{\max(\alpha(w_{i-1}, w_i) - d, 0)}{\alpha(w_{i-1})} + \lambda(w_{i-1})P_{CONTINUATION}(w_i)$$

$\lambda$  is a normalizing constant; the probability mass we've discounted

$$\lambda(w_{i-1}) = \frac{d}{\alpha(w_{i-1})} |\{w : \alpha(w_{i-1}, w) > 0\}|$$

the normalized discount

The number of word types that can follow  $w_{i-1}$   
= # of word types we discounted  
= # of times we applied normalized discount



# Kneser-Ney Smoothing (遞迴式)

$$P_{KN}(w_i | w_{i-n+1}^{j-1}) = \frac{\max(c_{KN}(w_{i-n+1}^j) - d, 0)}{c_{KN}(w_{i-n+1}^{j-1})} + \lambda(w_{i-n+1}^{j-1}) P_{KN}(w_i | w_{i-n+2}^{j-1})$$

$$c_{KN}(\bullet) = \begin{cases} \text{count}(\bullet) & \text{高階 ghest order} \\ \text{continuationcount}(\bullet) & \text{for lower order} \end{cases}$$

*continuation count* = •的不同前一詞個數

# 本次作業

- 等助教發佈
- 作業說明
- 資料
- 種子程式
- 驗收方式