Mô hình ngôn ngữ
(Language Models)

Nguyễn Tấn Phú
ntanphu@ctuet.edu.vn

Bộ môn HTTT
Khoa CNTT - Đai học KT - CN Cần Thơ

Nội dung

1. Giới thiệu mô hình ngôn ngữ
2. Ước lượng xác suất N-gram
3. Đánh giá và mức độ hỗn loạn thông tin (evaluation & perplexity)
4. Làm mịn và chiết khấu (smoothing & discounting)

Giới thiệu mô hình ngôn ngữ (Language Models)

- Mô hình ngôn ngữ xác suất (Probabilistic Language Models) hay mô hình ngôn ngữ (Language Models - LM) là **mô hình thống kê ước lượng xác suất** của một câu (sentence), một chuỗi các từ (word sequence) hoặc một từ (word).

- Việc ước lượng xác suất xuất hiện của một từ, chuỗi các từ hay của một câu giúp cho chúng ta có thể dự đoán (predict) được từ sắp xuất hiện là từ, từ có đúng chính tả, hay câu đó có đúng chính tả hay không

Giới thiệu mô hình ngôn ngữ (Language Models)

- **Miền kiến thức nào đó (domain knowledge):**
  - Khi nói đến "máu" thì chúng ta có kiến thức "máu" sẽ có màu "đỏ" - "máu đỏ"
- **Kiến thức về cú pháp (syntactic knowledge):**
  - Trong tiếng Việt danh từ sẽ đứng trước tính từ còn trong tiếng Anh thì tính từ sẽ đứng trước danh từ.
- **Kiến thức về từ vựng (lexical knowledge):**
  - Chúng ta có kiến thức từ vựng "trà xanh" (green tea), "trà sữa" (milk tea) và "trà đá" (ice tea)

Mô hình ngôn ngữ N-gram

- Mô hình ngôn ngữ N-gram (N-gram language model) hay gọi tắt là mô hình N-gram (N-gram model) ước lượng xác suất của từ (word) trong một ngữ cảnh cho trước (prior context).

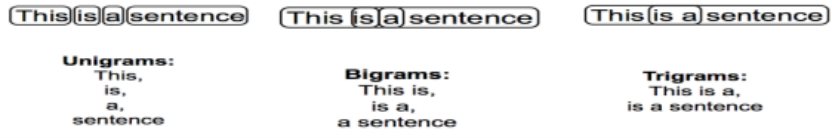
- **P(the|its water is so transparent that:)**
  - Là xác suất xuất hiện của từ **"the"** trong ngữ cảnh cho trước **"its water is so transparent that"**.
  - Nếu chỉ đơn giản đếm (count) và chia (divide).

$$\mathbf{P}(\textit{the}|\textit{its water is so transparent that}) = \frac{\mathbf{C}(\textit{its water is so transparent that the})}{\mathbf{C}(\textit{its water is so transparent that})}$$

Mô hình ngôn ngữ N-gram

- Kích thước corpus không đủ lớn để ước lượng ⇒ count có khả năng bằng 0 ⇒ chưa đạt hiệu quả

- Mô hình N-gram sử dụng N-1 từ trong ngữ cảnh cho trước để dự đoán từ thứ N



Ước lượng xác suất N-gram

- **Mô hình ngôn ngữ:** Từ (word) ký hiệu là w, chuỗi các từ (word sequences) ký hiệu w_1^n là một chuỗi gồm n từ theo thứ tự w_1,w_2,...w_n

$$w_1^n = w_1 ... w_n$$

- Áp dụng quy tắc chuỗi xác suất (chain rule of probability) đối với từ
$\Rightarrow$ xác suất xuất hiện chuỗi w_1^n là $P(w_1^n)$

$$P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)...P(w_n|w_1^{n-1}) = \prod_{k=1}^{n} P(w_k|w_1^{k-1})$$

Ước lượng xác suất N-gram

- Mô hình Ngram sử dụng N-1 từ trong ngữ cảnh cho trước để dự đoán từ thứ N.

$$P(w_n|w_1^{n-1} \approx P(w_n|w_{n-N+1}^{n-1})$$

- Giả định Markov (Markov assumption) giả định xác suất xuất hiện của một từ chỉ phụ thuộc vào một số từ trước đó thay vì phải phụ thuộc vào tất cả các từ trước đó:

$$\mathbf{P(the|its\ water\ is\ so\ transparent\ that) \approx P(the|that)}$$

Mô hình ngôn ngữ N-gram

- Sử dụng mô hình bigram và giả định Markov, xác suất xuất hiện chuỗi w_1^n:

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

- Sử dụng mô hình Ngram và giả định Markov, xác suất xuất hiện chuỗi w_1^n:

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k|w_{k-N+1}^{k-1})$$

Mô hình ngôn ngữ N-gram

- Để ước lượng xác suất bigram hay Ngram, "ước lượng xác suất cực đại" (Maximum Likelihood Estimate - MLE) được sử dụng bằng cách đếm (count) số lần xuất hiện của từ/chuỗi từ trong corpus và chuẩn hóa (normalizing) giá trị **count** để giá trị **P** nằm trong phạm vi **0** đến **1**:

$$P(w_n|w_{n-1}) = \frac{count(w_{n-1},w_n)}{count(w_{n-1})} = \frac{c(w_{n-1},w_n)}{c(w_{n-1})}$$

- **Bigram:**

$$P(w_n|w_{n-1}) = \frac{c(w_{n-1}w_n)}{c(w_{n-1})}$$

- **Ngram:**

$$P(w_n|w_{n-N+1}^{n-1}) = \frac{c(w_{n-N+1}^{n-1}w_n)}{c(w_{n-N+1}^{n-1})}$$

- **Chú ý: Cần thêm vào các token (<s>) và (</s>) vào đầu và cuối mỗi câu trong văn bản và xem chúng như các "từ" (additional words).**

Mô hình ngôn ngữ N-gram

N-GRAM MODEL FORMULAS	
Word sequences	$w_1^n = w_1 \ldots w_n$
Chain rule of probability	$P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)\ldots P(w_n \mid w_1^{n-1}) = \prod_{k=1}^{n} P(w_k \mid w_1^{k-1})$
Bi-gram approximation	$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-1})$
N-gram approximation	$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-N+1}^{k-1})$
N-gram approximation and Markov assumption	$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-N+1}^{n-1})$
The probability of a complete word sequence	$P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)\ldots P(w_n \mid w_1^{n-1}) = \prod_{k=1}^{n} P(w_k \mid w_1^{k-1})$ $P(w_1^n) \approx \prod_{k=1}^{n} P(w_k \mid w_{k-1})$

Một số ví dụ đơn giản

- Sử dụng ví dụ từ (Jurafsky, D. and Martin, J.H, 2019). Giả sử corpus gồm 5 câu:**"I am Sam", "Sam I am", "Sam I like", "Sam I do like", "do I like Sam"** . Sử dụng mô hình ngôn ngữ bigram:
- Hãy dự đoán từ tiếp theo sau các chuỗi từ dưới đây:
  - <s>Sam ...
  - <s>Sam I do ...
  - <s>Sam I am Sam ...
  - <s>do I like ...
- Câu nào sau đây là tốt nhất? Vì sao?
  - <s>Sam I do I like</s>
  - <s>I do Sam</s>
  - <s>I do like Sam I am</s>

Một số ví dụ đơn giản

- **Ví dụ:**

Trước khi thực hiện cần thêm vào token <s> và token </s> vào đầu và cuối mỗi câu .
Vậy ta có tập dữ liệu như sau:
<s> I am Sam </s>
<s> Sam I am </s>
<s> Sam I like </s>
<s> Sam I do like </s>
<s> do I like Sam </s>

→ Tập dữ liệu gồm 5 câu, số lượng token là 27, tập từ vựng V gồm 7 từ.

Ví dụ (tl.)

<s> I am Sam </s>
<s> Sam I am </s>
<s> Sam I like </s>
<s> Sam I do like </s>
<s> do I like Sam </s>

Unigram counts	</s>	<s>	I	Sam	am	do	like	Tổng
count	5	5	5	5	2	2	3	27

Bigram counts	</s>	<s>	I	Sam	am	do	like
</s>	0	0	0	0	0	0	0
<s>	0	0	1	3	0	1	0
I	0	0	0	0	2	1	2
Sam	2	0	3	0	0	0	0
am	1	0	0	1	0	0	0
do	0	0	1	0	0	0	1
like	2	0	0	1	0	0	0

Một số ví dụ đơn giản

Unigram counts	</s>	<s>	I	Sam	am	do	like	Tổng
count	5	5	5	5	2	2	3	27

Ví dụ (tl.)

Bigram counts	</s>	<s>	I	Sam	am	do	like
</s>	0	0	0	0	0	0	0
<s>	0	0	1	3	0	1	0
I	0	0	0	0	2	1	2
Sam	2	0	3	0	0	0	0
am	1	0	0	1	0	0	0
do	0	0	1	0	0	0	1
like	2	0	0	1	0	0	0

Xác suất Bigram thô (thực hiện bằng cách normalize bigram với unigram)

Xác suất của một số bigram trong corpus:

P(I|<s>) = 1/5 = 0,2000
P(Sam|<s>) = 3/5= 0,6000
P(am|I) = 2/5= 0,4000
P(do|I) = 1/5= 0,5000
P(</s>|Sam) = 2/5= 0,400
P(Sam|am) = 1/2 = 0,5000

P bigram	</s>	<s>	I	Sam	am	do	like
</s>	0	0	0	0	0	0	0
<s>	0	0	0,2000	0,6000	0	0,2000	0,0000
I	0	0	0	0	0,4000	0,2000	0,4000
Sam	0,4000	0	0,6000	0	0	0	0
am	0,5000	0	0	0,5000	0	0	0
do	0	0	0,5000	0	0	0	0,5000
like	0,6667	0	0	0,3333	0	0	0

Một số ví dụ đơn giản

Ví dụ (tl.)

P bigram	</s>	<s>	I	Sam	am	do	like
</s>	0	0	0	0	0	0	0
<s>	0	0	0,2000	0,6000	0	0,2000	0,0000
I	0	0	0	0	0,4000	0,2000	0,4000
Sam	0,4000	0	0,6000	0	0	0	0
am	0,5000	0	0	0,5000	0	0	0
do	0	0	0,5000	0	0	0	0,5000
like	0,6667	0	0	0,3333	0	0	0

Câu nào sau đây là tốt nhất? Vì sao?

(5)<s>Sam I do I like</s>

$$P(< s > Sam\ I\ do\ like < /s >)$$
$$= P(Sam| < s >) * P(I|Sam) * P(do|I) * P(I|do) * P(like|do) * P(< /s > |like)$$
$$= 0{,}60 * 0{,}60 * 0{,}20 * 0{,}50 * 0{,}50 * 0{,}67 = 0{,}01206$$

(6)<s>I do Sam</s>

$$P(< s > I\ do\ Sam < /s >) = P(I| < s >) * P(do|I) * P(Sam|do) * P(< /s > |Sam) = 0{,}20 * 0{,}2 * 0 * 0{,}40 = 0$$

(7)<s>I do like Sam I am</s>

$$P(< s > I\ do\ like\ Sam\ I\ am < /s >)$$
$$= P(I| < s >) * P(do|I) * P(like|do) * P(Sam|like) * P(I|Sam) * P(am|I) * P(< /s > |am)$$
$$= 0{,}20 * 0{,}20 * 0{,}50 * 0{,}33 * 0{,}60 * 0{,}40 * 0{,}50 = 0{,}000792$$

→Trong thực tế, thực hiện trong log space để advoid underflow

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

Bài tập

- **Tính xác suất bigram:**
  <s> I am Sam </s>
  <s> Sam I am </s>
  <s> I do not like green eggs and ham </s>

☞ **Tính:**

- P(I|<s>) = ?
- P(Sam|<s>) = ?
- P(am|I) = ?
- P(</s>|Sam) = ?
- P(Sam|am) = ?
- P(do|I) = ?

Evaluation & Perplexity

- **Đánh giá các mô hình ngôn ngữ:**
  - Gán xác suất cao cho các câu thực hoặc các câu có tần suất xuất hiện lớn (Không gán các câu sai ngữ pháp hoặc các câu ít xuất hiện)
  - Huấn luyện mô hình trên một tập huấn luyện (training set)
  - Đánh giá trên một tập dữ liệu mới (test set)
  - Sử dụng ma trận độ đo để đánh giá mức độ tốt của mô hình trên tập test

Đánh giá mô hình

- **So sánh 2 mô hình A và B:**
  - Sử dụng mỗi mô hình cho một nhiệm vụ cụ thể (sửa lỗi chính tả, nhận dạng tiếng nói, dịch máy, ...)
  - Thử nghiệm (chạy) nhiệm vụ đó, tính độ chính xác khi sử dụng mô hình A và B (Bao nhiêu từ sai được sửa đúng, Bao nhiêu từ được dịch đúng)
  - So sánh độ chính xác khi sử dụng A và B

Đánh giá mô hình

- Đánh giá bên ngoài khá tốn thời gian $\Rightarrow$ Đánh giá bên trong (intrinsic evaluation) hay đánh giá nội tại thường được sử dụng hơn do không phụ thuộc vào ứng dụng.

- Đánh giá trong sử dụng độ đo perplexity (độ phức tạp)

☞ Đánh giá xác suất không tốt:

  - Chỉ khi dữ liệu test giống dữ liệu train (về bộ từ vựng)

  - Tốt cho thí nghiệm nhưng không tốt cho thực tế

Ý tưởng của Perplexity

- Shannon Game:
  - Ta có thể tiên đoán từ tiếp theo không?
    - I always order pizza with cheese and ____
    - The 33rd President of the US was ____
    - I saw a ____
  - Có thể dùng unigram không?

mushrooms 0.1
pepperoni 0.1
anchovies 0.01
....
fried rice 0.0001
....
and 1e-100

- Mô hình tốt sẽ gán xác suất cao cho từ thường xuyên xuất hiện ở vị trí dự đoán

Độ phức tạp (Perplexity)

- Mô hình tốt nhất là mô hình dự đoán từ chưa nhìn thấy tốt nhất:
  - Cho xác suất câu cao nhất P(sentence)

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

  - Perplexity là nghịch đảo xác suất trên tập test, chuẩn hóa theo số từ
  - Luật chuỗi: $PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 ... w_{i-1})}}$
  - Với bigrams: $PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$

Ví dụ

- W là tập hợp các chữ số 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, sử dụng mô hình unigram thì PP(W)

$$PP(W) = P(w_1 w_2 ... w_n)^{-\frac{1}{N}} = \left(\left(\frac{1}{10}\right)^{10}\right)^{-\frac{1}{10}} = \left(\frac{1}{10}\right)^{-1} = 10$$

- Tương tự, nếu sử dụng mô hình unigram với dãy kí tự L gồm a, b, ..., z thì PP(L) là 26. Perplexity của bảng mã ASCII sử dụng mô hình unigram là 256.

Ví dụ

- Cho L = $\{a, b, c, d\}$ $*$, với P(a) = P(b) = P(c) = P(d) = $\frac{1}{4}$ (không phụ thuộc vào ngữ cảnh). Với mỗi w $\in$ L trong mô hình này thì PP của mỗi w được tính

$$PP(w) = P(w_1 w_2 ... w_n)^{-\frac{1}{N}} = \left(\frac{1}{4} * \frac{1}{4} * \frac{1}{4} * \frac{1}{4}\right)^{-\frac{1}{4}} = 4$$

Hiện tượng Overfitting

- N-grams chỉ tiên đoán từ tốt nếu tập test giống tập train.
  - Cần tạo ra mô hình có tính tổng quát, nghĩa là có thể xử lý các trường hợp xác suất = 0 (những TH không có trong tập train nhưng có trong tập test)

Zeros and Zero probability bigrams

- TH xác suất = 0
  - **Tập train:**
    ... denied the allegations
    ... denied the reports
    ... denied the claims
    ... denied the request
  - **Tập test:**
    ... denied the offer
    ... denied the loan
  - P("offer" | denied the) = 0
  - xác suất của 1 câu hoặc một cụm từ về 0
- **Sử dụng các phương pháp làm mịn**

Làm mịn và chiết khấu (Smoothing & Discounting)

❖ **Làm mịn và chiết khấu:**

- Khi các N-gram phân bố thưa, nhiều N-gram không xuất hiện hoặc số lần xuất hiện nhỏ dẫn đến ước lượng các câu có chứa các N-gram này sẽ có kết quả không tốt.
- Khi Trong thực tế khi tính xác suất của một câu, có rất nhiều trường hợp sẽ gặp N-gram chưa xuất hiện trong dữ liệu huấn luyện bao giờ $\Rightarrow$ xác suất của cả câu bằng 0 mặc dù câu đó có thể là một câu hoàn toàn đúng về mặt ngữ pháp và ngữ nghĩa.

☞ Phương pháp **"làm mịn" (smoothing)** hay **"chiết khấu" (discounting)** được sử dụng, các tham số (parameters) được làm mịn (smoothed) để tính lại xác suất cho các Ngram chưa từng xuất hiện

Làm mịn (Smoothing)

❖ **Maximum Likelihood Estimates - MLE:**

- Maximum likelihood estimation (MLE): Get n-gram counts from a corpus and normalize so that the values lie between 0 and 1.
- Ước lượng xác suất cực đại MLE dùng để: ước lượng tham số của mô hình M từ tập dữ liệu huấn luyện T
- VD:
  - Giả sử từ "bagel" xuất hiện 400 lần trong corpus gồm 1 triệu từ.
  - Xác suất để một từ ngẫu nhiên trong một văn bản khác là từ "bagel" là bao nhiêu?
  - Ước lượng MLE là 400/1,000,000 = 0.0004

☞ Điều này có thể không tốt trong một vài corpus:

- But it is the estimate that makes it most likely that "bagel" will occur 400 times in a million word corpus.

Làm mịn (Smoothing)

- Ý tưởng của phương pháp làm mịn là:**Giả sử mỗi từ xuất hiện nhiều hơn một lần so với thực tế** hay nói cách khác là thực hiện cộng vào mỗi giá trị số lần xuất hiện (count-c) của từ.
- Giả sử gọi V là kích thước của bộ từ vựng, áp dụng phương pháp làm mịn vào mô hình bigram ta được:

$$P(w_n|w_{n-1}) = \frac{c(w_{n-1}w_n)+k}{\sum_{w^*} c(w_{n-1}w^*) + Vk}$$

- Phương pháp làm mịn này được gọi là làm mịn Lidstone. Tùy trường hợp đặc biệt sẽ có những tên gọi khác nhau:
  - Add-1 smoothing hay Laplace smoothing nếu k = 1
  - Jeffrey-Perks law nếu k =0.5

Làm mịn Laplace và mô hình Unigram

Nếu chỉ sử dụng MLE để tính xác suất của từ w_i thì xác suất unigram của w_i được tính là số lần xuất hiện của w_i là $c(w_i)$ chia cho tổng số token N với $N = \sum_{w \in V} c(w)$

Vậy $P_{MLE}(w_i)$ được tính theo công thức: $P(w_i) = \frac{c(w_i)}{\sum_{w \in V} c(w)} = \frac{c(w_i)}{N}$

Áp dụng phương pháp làm mịn Laplace với $k=1$ vào mô hình unigram, với V là kích thước của bộ từ vựng:

$$P(w_n|w_{n-1}) = \frac{c(w_{n-1}w_n) + k}{\sum_{w'} c(w_{n-1}w') + Vk} \quad \Rightarrow \quad P_{Laplace}(w_i) = \frac{c(w_i) + 1}{\sum_w c(w_i) + V} = \frac{c(w_i) + 1}{N + V}$$

Để dễ so sánh với phương pháp MLE, giá trị $c(w_i)$ có thể được điều chỉnh thành:

$$c^*(w_i) = [c(w_i) + 1] \frac{N}{N + V}$$

Vậy giá trị chiết khấu tương đối (relative discount) d_c được tính : $d_c = \frac{c^*}{c}$

Làm mịn Laplace và mô hình Bigram

Nếu MLE được sử dụng để tính xác suất bigram: $P(w_i|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$

Áp dụng phương pháp làm mịn Laplace vào mô hình bigram

$$P(w_n|w_{n-1}) = \frac{c(w_{n-1}w_n)+k}{\sum_{w'} c(w_{n-1}w')+Vk} \quad \Rightarrow \quad P_{Laplace}(w_i|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

Điều chỉnh giá trị *count*

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

Berkeley Restaurant Corpus: Laplace smoothed bigram counts

- Add-one smoothed bigram counts of 9332 sentences and V=1446

	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

Compute the Add-one smoothed bigram probabilities ?

- Normalize by unigrams:

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

Laplace-smoothed bigrams

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

	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

Reconstructed counts

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

	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

Compare with raw bigram counts

	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

Ví dụ

- Giả sử corpus gồm 5 câu: “I am Sam”, “Sam I am”, “Sam I like”, “Sam I do like”, “do I like Sam”. Sử dụng mô hình bigram, hãy cho biết giá trị độ hỗn loạn thông tin của chuỗi “<s>I do like Sam </s>”. Sử dụng mô hình bigram và làm mịn Laplace, hãy cho biết câu nào dưới đây tốt hơn ?
  - 1) <s> do Sam I like </s>
  - 2) <s> Sam do I like </s>

Ví dụ

Giả sử corpus gồm 5 câu: “I am Sam”, “Sam I am”, “Sam I like”, “Sam I do like”, “do I like Sam”. Sử dụng mô hình bigram, hãy cho biết giá trị độ hỗn loạn thông tin của chuỗi “<s>I do like Sam”.

- Xác suất bigram

$$P_{Laplace}(do| < s >) = \frac{c(<s>do)+1}{c(<s>)+V} = \frac{1+1}{5+7} = \frac{2}{12} \qquad P_{Laplace}(Sam| < s >) = \frac{3+1}{5+7} = \frac{2}{12}$$

$$P_{Laplace}(Sam|do) = \frac{c(do\ Sam)+1}{c(do)+V} = \frac{0+1}{2+7} = \frac{1}{9} \qquad P_{Laplace}(I|do) = \frac{1+1}{2+7} = \frac{2}{9}$$

$$P_{Laplace}(I|Sam) = \frac{c(Sam\ I)+1}{c(Sam)+V} = \frac{3+1}{5+7} = \frac{4}{12} \qquad P_{Laplace}(do|Sam) = \frac{0+1}{5+7} = \frac{1}{12}$$

$$P_{Laplace}(like|I) = \frac{c(I\ like)+1}{c(like)+V} = \frac{2+1}{3+7} = \frac{3}{12} \qquad P_{Laplace}(</s > |like) = \frac{0+1}{5+7} = \frac{1}{12}$$

- Vậy xác suất các câu

$$P(< s > do\ Sam\ I\ like\ </s >) = \frac{2}{12} * \frac{1}{9} * \frac{4}{12} * \frac{3}{12} * \frac{1}{12} = 1{,}2860$$

$$P(< s > Sam\ do\ I\ like\ </s >) = \frac{2}{12} * \frac{1}{12} * \frac{2}{9} * \frac{3}{12} * \frac{1}{12} = 1{,}2860$$

→ Do cả 2 câu có xác suất giống nhau nên cả 2 đều tốt ngang nhau

Làm mịn Add-k

- Việc thêm 1 đã làm thay đổi đáng kể xác suất của các N-gram. Phiên bản cải tiến của thuật toán **Add-one smoothing** là **Add-k smoothing**
- Add-k smoothing đòi hỏi phải chọn lựa giá trị k $\Rightarrow$ có thể thực hiện bằng cách tối ưu hóa (optimizing) trên tập devset

$$P^*_{Add-k}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+k}{C(w_{n-1})+kV}$$

- **Chú ý:** Add-one smoothing và Add-k smoothing không hiệu quả trong mô hình ngôn ngữ nên nó thường được sử dụng cho các mô hình NLP khác như trong các lĩnh vực có số lượng số 0 không phải là quá lớn hoặc phân loại văn bản.

Chiết khấu (discounting)

- Chiết khấu (discounting) sẽ "mượn" xác suất của N-gram quan sát được (observed) và phân phối lại (redistribute) các xác suất đã mượn đó
  - Trong làm mịn Lidstone, việc vay mượn khối lượng xác suất được thực hiện bằng cách tăng mẫu số của các ước lượng tần số tương đối (relative frequency estimates); khối lượng xác suất đã vay mượn sau đó được phân phối lại bằng cách tăng tử số cho tất cả N-gram
  - Một cách tiếp cận khác là mượn cùng một khối lượng xác suất từ tất cả các N-gram quan sát được và chỉ phân phối lại cho các N-gram không quan sát được (unobserved). Quá trình này được gọi là chiết khấu tuyệt đối (absolute discounting)
  - Chiết khấu vay mượn một lượng xác suất từ N-gram quan sát được và không cần phân phối lại lượng xác suất đã mượn này ngang nhau. Thay vào đó, chúng ta có thể sử dụng mô hình ngôn ngữ bậc thấp hơn (lower-order model), hay còn gọi là back-off.

Back off

- Back-off là mô hình ngôn ngữ bậc thấp hơn (lower-order model):
  - Nếu N-gram có giá trị 0 $\Rightarrow$ chúng ta back-off về (N-1)-gram.
  - Nếu (N-1)-gram cũng không có giá trị khác 0 $\Rightarrow$ tiếp tục back-off về (N-2)-gram.
- Quá trình back-off được thực hiện cho đến khi đạt được giá trị khác 0.
- Ví dụ
  - Nếu có 4-gram $\Rightarrow$ sử dụng 4-gram;
  - Nếu không có 4-gram $\Rightarrow$ sử dụng 3-gram;
  - Nếu không có 3-gram $\Rightarrow$ sử dụng 2-gram;
  - Nếu không có 2-gram $\Rightarrow$ sử dụng 1-gram
- Cách thực hiện này còn gọ là Katz back-off

Back off

Nếu sử dụng MLE để tính xác suất thì tổng các xác xuất MLE sẽ bằng 1,00 → Nếu sử dụng các xác suất MLE và thực hiện *back-off* về mô hình ngôn ngữ bậc thấp hơn (khi xác suất MLE bằng không) sẽ dẫn đến tổng xác suất có khả năng lớn hơn 1,00.

Do đó, P_{Katz} được tính như sau:

$$P_{Katz}(w_n|w_{n-N+1}^{n-1}) = \begin{cases} P^*(w_n|w_{n-N+1}^{n-1}), & if\ C(w_{n-N+1}^n) > 0 \\ \alpha(w_{n-N+1}^{n-1}) P_{Katz}(w_n|w_{n-N+2}^{n-1}), & if\ C(w_{n-N+1}^n) = 0 \end{cases}$$

Trong đó, $\alpha(w_{n-N+1}^{n-1})$ là giá trị chiết khấu trong ngữ cảnh w_{n-N+1}^n

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

Back off

Áp dụng vào trường hợp trigram

$$P_{Katz}(w_n|w_{n-2}w_{n-1}) = \begin{cases} P^*(w_n|w_{n-2}w_{n-1}), & if\ C(w_{n-2}w_{n-1}w_n) > 0 \\ \alpha(w_{n-2}, w_{n-1})P_{Katz}(w_n|w_{n-1}), & else\ if\ C(w_{n-2}w_{n-1}) > 0 \\ P^*(w_n), & otherwise \end{cases}$$

Tương tự đối với trường hợp bigram

$$P_{Katz}(w_n|w_{n-1}) = \begin{cases} P^*(w_n|w_{n-1}), if\ C(w_{n-1}, w_n) > 0 \\ \alpha(w_{n-1})P^*(w_{n-1}), otherwise \end{cases}$$

Katz back-off thường được kết hợp với phương pháp Good-Turing để ước lượng xác suất P^* và giá trị α.

Good Turing

Ý tưởng của phương pháp làm mịn Good Turing là sử dụng số lần xuất hiện *count* của những sự kiện đã xuất hiện một lần (*seen one*) để ước lượng giá trị *count* cho những sự kiện chưa được biết (*have never seen*)

- Gọi *Nc* là tần số của tần số *c* (*frequency of frequency*) hay là số lần xuất hiện (*count*) của tần số *c*

$$P^*_{GT}\ (\text{những sự kiện có tần số xuất hiện là } 0) = \frac{N_1}{N}$$

- Và như vậy xác suất của những sự kiện có tần số xuất hiện c lần sẽ được ước lượng lại :

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

$$P^*_{GT}(\text{sự kiện có tần số xuất hiện } c \text{ lần}) = \frac{c^*}{N} = (c+1)\frac{N_{c+1}}{N * N_c}$$

Nội suy - Interpolation

Back-off là một trong những cách kết hợp các mô hình N-gram khác nhau. Một hướng tiếp cận khác để kết hợp các mô hình N-gram là phương pháp *interpolation:* các N-gram vẫn được kết hợp với nhau nhưng được nhân với một trọng số (*weight* - λ)

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

λ được tính dựa trên ngữ cảnh. Với giả định giá trị *count* của tri-gram dựa trên giá trị count của bi-gram thì λ của tri-gram sẽ có giá trị lớn hơn λ của bi-gram:

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

Giá trị λ được học từ *held-out* corpus.

Held-out corpus là một training corpus được sử dụng để tìm các tham số như λ, bằng cách lựa chọn λ sao cho tối đa hóa giá trị *likelihood* của held-out corpus. Thuật toán EM có thể được sử dụng để tối ưu hóa cục bộ (locally optimal) các λ

Inpterpolation hiệu quả hơn Back off

Nội suy - Interpolation

- Phương pháp tính **λ**
- Sử dụng held-out corpus

- Chọn λs để cực đại hóa xác suất của held-out data:
  - Cố định xác suất N-gram trong training data
  - Kế đến, tìm λs sao cho xác suất lớn nhất trong held-out corpus:

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

Kneser-Ney

- Một trong những phương pháp làm mịn N-gram hiệu quả và được sử dụng nhiều nhất là Kneser-Ney, được thực hiện dựa trên phương pháp chiết khấu tuyệt đối (*absolute discounting*).
- Ý tưởng của phương pháp là làm giảm (*discount*) số lượng (*count*) của N-gram bằng số lượng chiết khấu trung bình trong held-out corpus. Xác suất chiết khấu tuyệt đối sử dụng trong mô hình bigram được tính như sau:

$$P_{AbsoulteDiscounting}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) - d}{\sum_v C(w_{n-1}v)} + \lambda w_{n-1} P(w_n)$$

Trong đó:

- $\frac{C(w_{n-1}w_n)-d}{\sum_v C(w_{n-1}v)}$ là chiết khấu bigram
- λw_{n-1} là trọng lượng nội suy (*interpolation weight*)
- $P(w_n)$ là unigram

Kneser-Ney

Giả thuyết đặt ra là các từ xuất hiện trong ngữ cảnh ở quá khứ sẽ có khả năng xuất hiện ở những ngữ cảnh mới. Với *v* là ngữ cảnh đã nhìn thấy trước đó, *C(vw)* là số lượng *count* của từ *w* trong ngữ cảnh *v*, xác suất từ *w* xuất hiện trong ngữ cảnh mới được gọi là $P_{CONTINUATION}$ được tính:

$$P_{CONTINUATION}(w) = \frac{|\{v: C(vw) > 0\}|}{\sum_{w' \in V} |\{v: C(vw') > 0\}|}$$

Làm mịn nội suy Kneser-Ney được áp dụng trong mô hình bigram

$$P_{KN}(w_n|w_{n-1}) = \frac{max(C(w_{n-1}w_n) - d, 0)}{C(w_{n-1})} + \lambda w_{n-1} P_{CONTINUATION}(w_n)$$

Trong đó trọng lượng nội suy (interpolation weight) λw_{n-1} được tính : $\lambda w_{n-1} = \frac{d}{C(w_{n-1})} |\{w|C(w_{n-1}w) > 0\}|$

Với

- $\frac{d}{C(w_{n-1})}$ là chiết khấu (*normal discounting*)
- $|\{w|C(w_{n-1}w) > 0\}|$ là số loại từ theo sau w_{n-1} , còn được tính là số từ loại (word types) mà chúng ta đã chiết khấu hoặc số lần chúng ta áp dụng chiết khấu