Trường Công nghệ Thông tin và Truyền thông
Khoa Công nghệ thông tin

XỬ LÝ NGÔN NGỮ TỰ NHIÊN

GVHD: Lâm Nhựt Khang
lnkhang@ctu.edu.vn

CHƯƠNG 2

MÔ HÌNH NGÔN NGỮ

Nội dung Chương

1. Giới thiệu mô hình ngôn ngữ
2. Một số xác suất cơ bản
3. Ước lượng xác suất N-gram
4. Đánh giá mô hình ngôn ngữ
5. Mô hình ngôn ngữ và OOV
6. Làm mịn và chiết khấu

Nội dung Chương

1. **Giới thiệu mô hình ngôn ngữ**
2. Một số xác suất cơ bản
3. Ước lượng xác suất N-gram
4. Đánh giá mô hình ngôn ngữ
5. Mô hình ngôn ngữ và OOV
6. Làm mịn và chiết khấu

Dự đoán từ

Trump chưa đồng ý gỡ thuế cho _____.

- A. Trung Quốc
- B. hạt điều
- C. Putin
- D. dệt may

Mô hình ngôn ngữ

- Mô hình ngôn ngữ (**Language Models - LM**) hay mô hình ngôn ngữ xác suất là mô hình thống kê ước lượng xác suất xuất hiện của một câu, một chuỗi các từ hoặc một từ.
- 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 được từ sắp xuất hiện là từ nào, từ có đúng chính tả hay câu đó có đúng ngữ pháp hay không.
- Con người thường dự đoán từ dựa vào:
  - Miền kiến thức: như khi nói đến "mùa đông" thì chúng ta có kiến thức "mùa đông" thông thường sẽ "lạnh".
  - Kiến thức về cú pháp: trong tiếng Việt danh từ đứng trước tính từ, còn trong tiếng Anh thì tính từ đứng trước danh từ ("ngôi nhà hạnh phúc" – "happy house").
  - Kiến thức về từ vựng: chúng ta có kiến thức từ vựng "trà xanh" (green tea), "trà sữa" (milk tea) và "trà đá" (ice tea).

Mô hình ngôn ngữ

- Mô hình ngôn ngữ N-gram hay gọi tắt là mô hình N-gram (**N-gram model**) ước lượng xác suất của từ trong một ngữ cảnh cho trước.
- Ví dụ: P(tin|công nghệ thông) là xác suất xuất hiện của từ "tin" trong ngữ cảnh cho trước "công nghệ thông".
- Nếu chỉ đơn giản thực hiện đếm số lần xuất hiện của cụm từ "công nghệ thông tin" và chia cho số lần xuất hiện của ngữ cảnh cho trước "công nghệ thông", ta được:

$$P(tin|công\ nghệ\ thông) = \frac{count(công\ nghệ\ thông\ tin)}{count(công\ nghệ\ thông)}$$

Mô hình ngôn ngữ

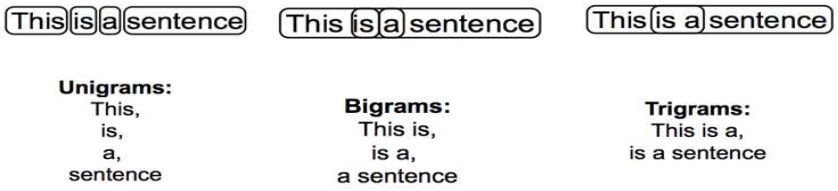
- Mô hình N-gram sử dụng N-1 từ trong ngữ cảnh cho trước để dự đoán từ thứ N:
  - Nếu N=1, ta có mô hình 1-gram hay unigram,
  - Nếu N=2, ta có mô hình 2-gram hay bigram,
  - Nếu N=3, ta có mô hình 3-gram hay trigram.
- Một cách tổng quát ta có thể mở rộng mô hình lên 4-gram, 5-gram…. hay N-gram

N-gram	**Chuỗi từ "ngôi nhà hạnh phúc của tôi"**
N=1 (unigram)	{ngôi, nhà, hạnh, phúc, của, tôi}
N=2 (bigram)	{ngôi nhà, nhà hạnh, hạnh phúc, phúc của, của tôi}
N=3 (trigram)	{ngôi nhà hạnh, nhà hạnh phúc, hạnh phúc của, phúc của tôi}
N=4	{ngôi nhà hạnh phúc, nhà hạnh phúc của, hạnh phúc của tôi}

Mô hình ngôn ngữ N-gram

Kích thước corpus không đủ lớn để ước lượng giá
- *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*.



Nội dung Chương

1. Giới thiệu mô hình ngôn ngữ
2. **Một số xác suất cơ bản**
3. Ước lượng xác suất N-gram
4. Đánh giá mô hình ngôn ngữ
5. Mô hình ngôn ngữ và OOV
6. Làm mịn và chiết khấu

Một số xác suất cơ bản

Xác suất	Ý nghĩa	Cách tính và/hoặc ví dụ
P(X)	xác suất X xảy ra	$P(sv\ nam) = 0{,}5$ (50% sinh viên là nam) $P(sv\ họ\ Nguyễn) = 0{,}3$ (30 trong 100 sinh viên có họ Nguyễn)
P(X,Y)	xác suất cả X và Y đều xảy ra	$P(họ\ Nguyễn, sv\ nam) = \frac{số\ lượng\ sv\ nam\ họ\ Nguyễn}{tổng\ số\ sv}$
P(X|Y)	xác suất X xảy ra khi Y khi đã xảy ra	$P(X\|Y) = \frac{P(X,Y)}{P(Y)}$ $P(sv\ họ\ Nguyễn\|\ sv\ nam) = \frac{P(sv\ họ\ Nguyễn, sv\ nam)}{P(sv\ nam)}$
Định lý Bayes		$P(X\|Y) = \frac{P(Y\|X)P(X)}{P(Y)}$ $P(sv\ họ\ Nguyễn\|\ sv\ nam) = \frac{P(sv\ nam\|sv\ họ\ Nguyễn)P(sv\ họ\ Nguyễn)}{P(sv\ nam)}$

Nội dung Chương

1. Giới thiệu mô hình ngôn ngữ
2. Một số xác suất cơ bản
3. **Ước lượng xác suất N-gram**
4. Đánh giá mô hình ngôn ngữ
5. Mô hình ngôn ngữ và OOV
6. Làm mịn và chiết khấu

Mô hình ngôn ngữ N-gram

- 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 $w_1 w_2 \dots w_n$

$$w_1^n = w_1 \dots w_n$$

- Áp dụng quy tắc chuỗi xác suất (***chain rule of probability***) đối với từ

→ xác suất xuất hiện chuỗi w_1^n là $P(w_1^n)$

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

Mô hình ngôn ngữ 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.

$$P(tin|công\ nghệ\ thông\ ) \approx P(tin|thông)$$

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 :

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

Mô hình ngôn ngữ N-gram

➢ Để ước lượng xác suất bigram hay Ngram, *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*).

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

Một số câu xây dựng được từ unigram model

- 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

Một số câu xây dựng được từ bigram model

- 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

Mô hình ngôn ngữ N-gram

- Có thể mở rộng lên 3-grams, 4-grams, 5-grams
- Không hiệu quả vì ngôn ngữ "phụ thuộc xa" (long-distance dependencies)
  - Syntactic dependencies
    - "The ***man*** next to the large oak tree near the grocery store on the corner **is** tall."
    - "The ***men*** next to the large oak tree near the grocery store on the corner **are** tall."
  - Semantic dependencies
    - "The ***bird*** next to the large oak tree near the grocery store on the corner **flies** rapidly."
    - "The ***man*** next to the large oak tree near the grocery store on the corner **talks**

Ví dụ

- 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*:

a) Hãy dự đoán từ tiếp theo sau các chuỗi từ dưới đây
(1) <s>Sam …
(2)<s>Sam I do . ..
(3)<s>Sam I am Sam ...
(4)<s>do I like ..

b) Câu nào sau đây là tốt nhất? Vì sao?
(5)<s>Sam I do I like</s>
(6)<s>I do Sam</s>
(7)<s>I do like Sam I am</s>

Ví dụ (tl.)

- 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>

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

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

→ 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

Ví dụ (tl.)

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

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 = 0,20
P(Sam|<s>) = 3/5= 0,60
P(am|I) = 2/5= 0,40
P(do|I) = 1/5= 0,20
P(</s>|Sam) = 2/5= 0,40
P(Sam|am) = 0/2 = 0,00

P bigram	</s>	<s>	I	Sam	am	do	like
</s>	0	0	0	0	0	0	0
<s>	0	0	0,20	0,60	0	0,20	0,00
I	0	0	0	0	0,40	0,20	0,40
Sam	0,40	0	0,60	0	0	0	0
am	0,50	0	0	0,50	0	0	0
do	0	0	0,50	0	0	0	0,50
like	0,67	0	0	0,33	0	0	0

LNK 23

Ví dụ (tl.)

P bigram	</s>	<s>	I	Sam	am	do	like
</s>	0	0	0	0	0	0	0
<s>	0	0	0,20	0,60	0	0,20	0,00
I	0	0	0	0	0,40	0,20	0,40
Sam	0,40	0	0,60	0	0	0	0
am	0,50	0	0	0,50	0	0	0
do	0	0	0,50	0	0	0	0,50
like	0,67	0	0	0,33	0	0	0

Câu nào sau đây là tốt nhất? Vì sao?

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

$$P(< s > Sam\ I\ do\ I\ like < /s >)$$
$$= P(Sam| < s >) * P(I|Sam) * P(do|I) * P(I|do) * P(like|do) * P(</s\ > |like)$$
$$= 0{,}60 * 0{,}60 * 0{,}20 * 0{,}50 * 0{,}20 * 0{,}67 = 0{,}004824$$

(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

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

LNK 24

Ví dụ: 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

Bài tập

Corpus được sử dụng là tác phẩm Truyện Kiều của Nguyễn Du và bản dịch của Truyện Kiều sang tiếng Anh do tác giả Huỳnh Sanh Thông thực hiện (Du, 1987)

Giả sử corpus là Truyện Kiều

Tiếng Việt	**Tiếng Anh**
Thúy Kiều là chị em là Thúy Vân	Thuy Kieu was oldest, younger was Thuy Van
Vân xem trang trọng khác vời	In quiet grace Van was be yond compare
Kiều càng sắc sảo mặn mà	Yet Kieu possessed a keener, deeper charm

→ "Kiều trang trọng sắc sảo"

Bài tập

Giả sử tập dữ liệu có 5 câu:

"bạn ăn cơm chưa", "cơm bạn ăn chưa", "ăn cơm chưa",
"bạn chưa ăn cơm", "chưa ăn".

Sử dụng mô hình ngôn ngữ bigram,

a. Hãy dự đoán từ kế tiếp theo sau chuỗi từ "<s> cơm chưa" và "<s> chưa bạn"

b. Câu nào sau đây là tốt nhất? "<s> chưa bạn </s>" và "<s> cơm chưa ăn </s>"

Nội dung Chương

1. Giới thiệu mô hình ngôn ngữ
2. Một số xác suất cơ bản
3. Ước lượng xác suất N-gram
4. **Đánh giá mô hình ngôn ngữ**
5. Mô hình ngôn ngữ và OOV
6. Làm mịn và chiết khấu

LM như thế nào là tốt ?

- Gán xác suất cao hơn cho các câu "thật" (*real or frequently observed*)
- Các câu "không thật" (*ungrammatical or rarely observed*) được gán xác suất thấp hơn

Đánh giá LM

Cách tốt nhất để đánh giá một mô hình ngôn ngữ là "nhúng" mô hình vào một ứng dụng cụ thể và đo lường mức độ cải tiến của ứng dụng → Đánh giá điểm cuối (*end-to-end*) hay đánh giá bên ngoài (***extrinsic evaluation***).

Cần đánh giá 2 mô hình ngôn ngữ A và B, đánh giá bên ngoài sẽ thực hiện các bước như sau:

- Sử dụng 2 mô hình ngôn ngữ này để giải quyết cùng một nhiệm vụ (*task*). Ví dụ như sửa lỗi chính tả hay hệ thống máy dịch.
- Thực hiện tác vụ và đo lường độ chính xác của 2 mô hình. Ví dụ: bao nhiêu từ sai chính tả được sửa chữa đúng, bao nhiêu từ được dịch đúng.
- So sánh độ chính xác (*accuracy*) của mô hình A và mô hình B

Đánh giá LM

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

Các bước thực hiện của phương pháp đánh giá nội tại như sau:

- "Xử lý" các tham số (*parameters*) của mô hình trên một tập huấn luyện (*training set* hay *training corpus*).
- Kiểm tra tính hiệu quả của mô hình trên tập dữ liệu chưa từng "thấy" (*unseen*) hay gọi là tập dữ liệu kiểm tra (*test set* hay *test corpus*). Một tập dữ liệu kiểm tra là một tập dữ liệu khác với tập dữ liệu huấn luyện hay nói cách khác tập dữ liệu kiểm tra chứa các dữ liệu đã không sử dụng trong tập dữ liệu huấn luyện.
- Sử dụng độ đo đánh giá (*evaluation metric*) để đánh giá tính hiệu quả của mô hình trên tập dữ liệu kiểm tra

Độ hỗn loạn thông tin – Perplexity (PP)

Giả sử sử dụng mô hình unigram với xác suất như sau:

(xuân; 0,002)	(cát; 0,0005)	(khăn; 0,0012)
(đá; 0,0008)	(mưa; 0,0001)	(muối; 0,00002)

Hãy dự đoán từ có trong các câu sau:

"Tôi đang nấu ăn nhưng bị thiếu ……."

"Cuộc sống thì ……"

"Họ đang xây cất ……".

A better model of a text is one which assigns a higher probability to the word that actually occurs

Minimize Perplexity → Maximize probability of the sentence
Low perplexity = Better Model

Perplexity (PP)

PP của một mô hình ngôn ngữ trên tập test là xác suất nghịch đảo (*inverse probability*) của tập test đó, chuẩn hóa (*normalized*) bởi số từ (*the number of words*)

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

Áp dụng quy tắc chuỗi xác suất: $PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \dots w_{i-1})}}$

Vậy nếu sử dụng mô hình bigram thì PP(W): $PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$

Cực tiểu PP cũng đồng nghĩa với cực đại hóa xác suất.

Thông thường PP bằng với kích thước của bộ từ vựng.

Ví dụ

W là tập hợp các chữ số {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, sử dụng mô hình unigram:

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

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}** $*$

P(a) = P(b) = P(c) = P(d) = 1/4 (không phụ thuộc vào ngữ cảnh).

Với mỗi $w \in L$ trong mô hình này thì PP của mỗi *w* được tính

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

Bài tập

Cho **L = {a, b,c, d}** $*$.

Giả sử xét chuỗi W chứa *a* nhiều gấp 3 lần *b*, *c* và *d*.

Tìm PP(W)

Bài tập

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".

Bài tập (tl.)

P bigram	</s>	<s>	I	Sam	am	do	like
</s>	0	0	0	0	0	0	0
<s>	0	0	0,20	0,60	0	0,20	0,00
I	0	0	0	0	0,40	0,20	0,40
Sam	0,40	0	0,60	0	0	0	0
am	0,50	0	0	0,50	0	0	0
do	0	0	0,50	0	0	0	0,50
like	0,67	0	0	0,33	0	0	0

Tập dữ liệu huấn luyện gồm 5 câu:

"<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>"

Xác suất xuất hiện của chuỗi "<s>I do like Sam" là:

$$P(< s > I\ do\ like\ Sam) = \frac{1}{5} * \frac{1}{5} * \frac{1}{2} * \frac{1}{3} = \frac{1}{150}$$

Vậy PP của chuỗi:

$$PP(< s > I\ do\ like\ Sam) = \sqrt[4]{150} = 3.5$$

Bài tập

Giả sử tập dữ liệu có 5 câu:

"bạn ăn cơm chưa", "cơm bạn ăn chưa", "ăn cơm chưa",
"bạn chưa ăn cơm", "chưa ăn".

Sử dụng mô hình ngôn ngữ bigram, hãy cho biết giá trị độ hỗn loạn thông tin của chuỗi "<s>cơm chưa ăn </s>" là bao nhiêu?

Nội dung Chương

1. Giới thiệu mô hình ngôn ngữ
2. Một số xác suất cơ bản
3. Ước lượng xác suất N-gram
4. Đánh giá mô hình ngôn ngữ
5. **Mô hình ngôn ngữ và OOV**
6. Làm mịn và chiết khấu

Generate new text

Choose a random bigram

(<s>, w) according to its probability

Now choose a random bigram (w, x) according to its probability

…..

And so on until we choose </s>

Then, string the words together

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

I want to eat Chinese food

Vấn đề "overfitting"

- N-grams chỉ dự đoán từ tốt nếu *test corpus* giống/gần giống với *training corpus*
  - Trong thực tế: rất khó để có thể xảy ra
  - Chúng ta cần *train* mô hình có khả năng khái quát (*generalize*)
  - One kind of generalization: Zeros!
    - Things that don't ever occur in the training set
      - But occur in the test set

Unknown word

Test set có thể chứa:
- Unknown words; hoặc
- Unseen N-grams

Nếu tập kiểm thử xuất hiện "từ" *w*, nhưng *w* này lại chưa từng xuất hiện trong tập huấn luyện

- Từ *w* sẽ được ước lượng giá trị xác suất P bằng 0
- Toàn bộ xác suất của tập test cũng bằng 0
- PP không tính được do chúng ta không thể chia cho 0

unknown word *"out of vocabulary" (OOV)*

Cách xử lý OOV

Cách 1:

- Xây dựng một danh sách các từ vựng trước,
- Chuyển đổi các từ OOV trong tập huấn luyện thành <UNK>,
- Xem <UNK> như một từ trong danh sách từ vựng và thực hiện ước lượng xác suất của từ <UNK>

Cách 2:

- Trong trường hợp không thể xây dựng được danh sách các từ vựng trước, nếu từ trong tập huấn luyện có số lần xuất hiện nhỏ hơn ngưỡng, chúng ta sẽ thay thế các từ đó bằng từ <UNK>
- Thực hiện ước lượng xác suất xuất hiện của từ <UNK> như bình thường.

Nội dung Chương

1. Giới thiệu mô hình ngôn ngữ
2. Một số xác suất cơ bản
3. Ước lượng xác suất N-gram
4. Đánh giá mô hình ngôn ngữ
5. Mô hình ngôn ngữ và OOV
6. **Làm mịn và chiết khấu**
   1. **Làm mịn Lidstone**
   2. **Chiết khấu và backoff**
   3. **Good Turing**
   4. **Nội suy**
   5. **Kneser-Ney**

Làm mịn và chiết khấu

- Khi các N-gram phân bố thưa, nhiều N-gram không xuất hiện hoặc số lần xuất hiện nhỏ → ước lượng các câu có chứa các N-gram này sẽ có kết quả không tốt.
- Trong thực tế khi tính xác suất của một câu, có rất nhiều trường hợp sẽ gặp N-gram chưa từng xuất hiện trong dữ liệu huấn luyện bao giờ → 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 được làm mịn để tính lại xác suất cho các Ngram chưa từng xuất hiện

Maximum Likelihood Estimates - MLE

MLE: Gets N-gram counts from a corpus and normalizes so that the values lie between 0 and 1

- Ví dụ:
  - 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

Làm mịn

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

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

➢ Phương pháp **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 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 bộ từ vựng:

$$P_{Laplace}(w_i) = \frac{c(w_i) + 1}{\sum_w c(w_i) + V} = \frac{c(w_i) + 1}{N + V}$$

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 Bigram

- Nếu MLE được sử dụng để tính xác suất bigram:

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

- Áp dụng phương pháp làm mịn Laplace vào mô hình bigram

$$P_{Laplace}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}$$

- Điều chỉnh giá trị *count*

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

Ví dụ

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

Giả sử tập dữ liệu có 5 câu:

"bạn ăn cơm chưa", "cơm bạn ăn chưa", "ăn cơm chưa",
"bạn chưa ăn cơm", "chưa ăn".

Sử dụng mô hình bigram và làm mịn Laplace,

Trong hai câu "<s> chưa bạn </s>" và "<s> cơm ăn </s>", câu nào tốt hơn?

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* → có thể thực hiện bằng cách tối ưu hóa (*optimizing*) trên tập devset

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

Gale và Church [53] cho biết phương pháp làm mịn Add-one và Add-k chỉ dựa trên giá trị count của từ và thất bại khi ước lượng giá trị cho các N-gram chưa từng "thấy". → Do đó, các phương pháp làm mịn này chỉ hiệu quả với bài toán phân loại văn bản, không hiệu quả trong các mô hình ngôn ngữ.

Nội dung Chương

1. Giới thiệu mô hình ngôn ngữ
2. Một số xác suất cơ bản
3. Ước lượng xác suất N-gram
4. Đánh giá mô hình ngôn ngữ
5. Mô hình ngôn ngữ và OOV
6. **Làm mịn và chiết khấu**
   1. Làm mịn Lidstone
   2. **Chiết khấu và backoff**
   3. Good Turing
   4. Nội suy
   5. Kneser-Ney

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.
- Chiết khấu tuyệt đối (***absolute discounting***): 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*).
- 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

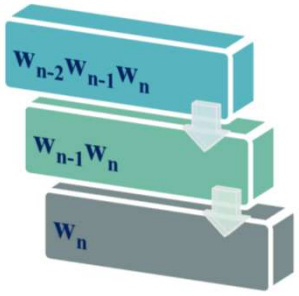
Back-off là mô hình ngôn ngữ bậc thấp hơn (*lower-order model*):

- Nếu N-gram có giá trị 0 → chúng ta back-off về (N-1)-gram;
- Nếu (N-1)-gram cũng không có giá trị khác 0 → 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 → sử dụng 4-gram;
- Nếu không có 4-gram → sử dụng 3-gram;
- Nếu không có 3-gram → sử dụng 2-gram;
- Nếu không có 2-gram → 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*.

Nội dung Chương

1. Giới thiệu mô hình ngôn ngữ
2. Một số xác suất cơ bản
3. Ước lượng xác suất N-gram
4. Đánh giá mô hình ngôn ngữ
5. Mô hình ngôn ngữ và OOV
6. **Làm mịn và chiết khấu**
   1. Làm mịn Lidstone
   2. Chiết khấu và backoff
   3. **Good Turing**
   4. Nội suy
   5. Kneser-Ney

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*

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

Vậy xác suất của những sự kiện có tần số xuất hiện *c* lần sẽ được ước lượng lại :

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

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

Ví dụ

Một nhà kinh doanh laptop từng mua bán các loại máy sau: 10 Dell Inspiron, 3 HP Elitebook, 2 Asus Vivo, 1 Dell XPS, 1 Lenovo ThinkPad và 1 Asus ZenBook. Trên thị trường vẫn còn nhiều dòng laptop khác mà người này chưa từng mua bán (gọi chung là *unseen*) như Dell Alienware, Apple Macbook, Acer Aspire… Lần lượt sử dụng các phương pháp:

a. MLE

b. Good Turing

c. Good Turing và giả sử số lượng dòng máy tính nhà kinh doanh chưa từng mua bán unseen là 20.

Hãy cho biết xác suất để người này mua bán một laptop trong nhóm *unseen* ở lần kế tiếp là bao nhiêu? Xác suất để mua bán một laptop Lenovo ThinkPad ở lần kế tiếp là bao nhiêu?

Bài tập

Giả sử có corpus “b a b a a c b c a c a c”, sử dụng mô hình bigram và làm mịn Good Turing đến *c* là 2, hãy cho biết xác suất xuất hiện của chuỗi “abcab” là bao nhiêu?

Nội dung Chương

1. Giới thiệu mô hình ngôn ngữ
2. Một số xác suất cơ bản
3. Ước lượng xác suất N-gram
4. Đánh giá mô hình ngôn ngữ
5. Mô hình ngôn ngữ và OOV
6. **Làm mịn và chiết khấu**
   1. Làm mịn Lidstone
   2. Chiết khấu và backoff
   3. Good Turing
   4. **Nội suy**
   5. Kneser-Ney

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.

Inpterpolation hiệu quả hơn Back off

Nội suy Interpolation

Thuật toán 2-1: **EM cho mô hình ngôn ngữ sử dụng làm mịn nội suy**

for $z \in \{1, 2, \ldots, n_{max}\}$ **do** — Khởi tạo

  $\lambda_z \leftarrow \frac{1}{n_{max}}$

repeat

  for $m \in \{1, 2, \ldots, M\}$ **do** — E-step

    for $z \in \{1, 2, \ldots, n_{max}\}$ **do**

      $q_m(z) \leftarrow p_z^*(w_m|w_{1:m-}) \times \lambda_z$

    $q_m(z) \leftarrow Normalize\ (q_m)$

  for $z \in \{1, 2, \ldots, n_{max}\}$ **do** — M-step

    $$\lambda_z \leftarrow \frac{1}{M}\sum_{m=1}^{M} q_m(z)$$

until tired

return λ

Nội dung Chương

1. Giới thiệu mô hình ngôn ngữ
2. Một số xác suất cơ bản
3. Ước lượng xác suất N-gram
4. Đánh giá mô hình ngôn ngữ
5. Mô hình ngôn ngữ và OOV
6. **Làm mịn và chiết khấu**
   1. Làm mịn Lidstone
   2. Chiết khấu và backoff
   3. Good Turing
   4. Nội suy
   5. **Kneser-Ney**

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

Assignment#1

Bài 1.

Download tập dữ liệu VNTC 27 topics
https://github.com/duyvuleo/VNTC/tree/master/Data/27Topics/Ver1.1

Hãy sử dụng *công cụ tách từ tiếng Việt* phù hợp. Mỗi câu hỏi sau đây hãy trả lời kết quả nhóm theo từng domain /train/test

1. Hãy viết chương trình đếm số lượng sub-directories trong corpus và đếm số lượng document trong corpus. Trình bày pseudo-code và cung cấp số liệu thống kê về số lượng document.
2. Hãy viết chương trình đếm số lượng câu và số lượng từ trong corpus. Trình bày pseudo-code và cung cấp số liệu thống kê về số lượng câu và số lượng từ trong corpus? Hãy cho biết câu dài nhất và ngắn nhất có bao nhiêu từ? Độ dài trung bình của câu là bao nhiêu?

Bài 2.

Download tập dữ liệu VNTC 27 topics
https://github.com/duyvuleo/VNTC/tree/master/Data/27Topics/Ver1.1

Tách từ theo khoảng trắng . Mỗi câu hỏi sau đây hãy trả lời kết quả nhóm theo từng domain /train/test

1. Hãy viết chương trình đếm số lượng câu và số lượng từ trong corpus. Trình bày pseudocode và cung cấp số liệu thống kê về số lượng câu và số lượng từ trong corpus? Hãy cho biết câu dài nhất và ngắn nhất có bao nhiêu từ? Độ dài trung bình của câu là bao nhiêu?
2. Hãy viết chương trình đếm số lượng unigram, bigram frequencies trong corpus. Không sử dụng bất kỳ phương pháp làm mịn nào, hãy cho biết 10 unigram và 10 bigram có số lần xuất hiện nhiều nhất. Trình bày pseudocode.
3. Sử dụng phương pháp làm mịn Good-Turning, Hãy cho biết 10 unigram và 10 bigram có số lần xuất hiện nhiều nhất. Trình bày pseudocode.