TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÀI TẬP LỚN

MÔN: Xử lý ngôn ngữ tự nhiên

Đề tài: Các mô hình ngôn ngữ N-gram và Ứng dụng

Nhóm sinh viên thực hiện:

Kim Đình Sơn 20102089

Đặng Ngọc Thuyên 20102277

Phùng Văn Chiến 20101163

Ngô Thành Đạt 20102624

Giảng viên hướng dẫn: TS. Hoàng Anh Việt

Hà Nội, tháng 12 năm 2013

Mục lục

1 Tổng quan về ngôn ngữ 3
2 Mô hình ngôn ngữ N-gram 3
2.1 Một số khái niệm 3
2.2 Mô hình N-gram 4
2.2.2 Chuỗi Markov 5
2.2.3 Ước lượng xác suất cho mô hình N-gram 5
2.3 Khó khăn khi xây dựng mô hình ngôn ngữ N-gram 5
2.3.1 Phân bố không đều 5
2.3.2 Kích thước bộ nhớ của mô hình ngôn ngữ 6
3 Các phương pháp làm mịn 6
3.1 Phương pháp chiết khấu (discounting) 6
3.1.1 Phương pháp làm mịn Add-one (Laplace) 7
3.1.2 Phương pháp làm mịn Good-Turing 8
3.1.3 Phương pháp làm mịn Witten-Bell 11
3.2 Phương pháp truy hồi (Back-Off) 12
3.3 Phương pháp nội suy (Recursive Interpolation) 13
3.3 Phương pháp làm mịn Kneser – Ney 13
3.3.1 Phương pháp Kneser-Ney với mô hình truy hồi 13
3.3.1 Phương pháp Kneser-Ney cải tiến 14
4 Kỹ thuật làm giảm kích thước dữ liệu, tối ưu hóa cho mô hình ngôn ngữ 15
4.1 Sử dụng Entropy và Độ hỗn loạn thông tin để đánh giá mô hình Ngram 16
4.1.1 Entropy 16
4.1.2 Độ hỗn loạn thông tin (perplexity) 17
4.2 Các kỹ thuật làm giảm kích thước dữ liệu của mô hình ngôn ngữ 18
4.2.1 Count cutoffs 19
4.2.2 Weighted Difference pruning 20
4.2.3 Stolcke pruning 21
5.Ứng dụng 24
Tài liệu tham khảo 26

1 Tổng quan về ngôn ngữ

Ngôn ngữ tự nhiên là những ngôn ngữ được con người sử dụng trong các giao tiếp hàng ngày: nghe, nói đọc, viết. Mặc dù con người có thể dễ dàng hiểu được và học các ngôn ngữ tự nhiên nhưng việc làm cho máy hiểu được ngôn ngữ tự nhiên không phải là chuyện dễ dàng. Sở dĩ có khó khăn là do ngôn ngữ tự nhiên có các bộ luật, cấu trúc ngữ pháp phong phú hơn nhiều các ngôn ngữ máy tính, hơn nữa để hiểu đúng nội dung các giao tiếp, văn bản trong ngôn ngữ tự nhiên cần phải nắm được ngữ cảnh của nội dung đó. Do vậy, để có thể xây dựng được một bộ ngữ pháp, từ vựng hoàn chỉnh, chính xác để máy có thể hiểu ngôn ngữ tự nhiên là một việc rất tốn công sức và đòi hỏi người thực hiện phải có hiểu biết về ngôn ngữ học.

Các phương pháp xử lý ngôn ngữ tự nhiên dựa trên thống kê không nhắm tới việc con người tự xây dựng mô hình ngữ pháp mà lập chương trình cho máy tính có thể "học" nhờ vào việc thống kê các từ và cụm từ có trong văn bản. Cốt lõi nhất của phương pháp xử lý ngôn ngữ tự nhiên dựa trên thống kê chính là việc xây dựng mô hình ngôn ngữ.

Mô hình ngôn ngữ là một phân bố xác suất trên các tập văn bản. Cụ thể thì mô hình ngôn ngữ cho biết xác suất một câu (một cụm từ hoặc một từ) trong bộ dữ liệu mẫu là bao nhiêu.

Ví dụ : Khi áp dụng mô hình ngôn ngữ cho tiếng Việt :

P["*ngày tết thật là vui*"] = 0,001.

P["*vui là thật tết ngày*"] = 0.

Mô hình ngôn ngữ được áp dụng trong rất nhiều lĩnh vực của xử lý ngôn ngữ tự nhiên như: kiểm tra lỗi chính tả, dịch máy hay phân đoạn từ... Chính vì vậy, nghiên cứu mô hình ngôn ngữ chính là tiền đề nghiên cứu các lĩnh vực tiếp theo.

Mô hình ngôn ngữ có nhiều hướng tiếp cận, nhưng chủ yếu được xây dựng theo mô hình N-gram mà ta sẽ đề cập dưới đây.

2 Mô hình ngôn ngữ N-gram

2.1 Một số khái niệm

Ngữ liệu:

Ngữ liệu (Corpus) là 1 dữ liệu tập hợp các văn bản, ngôn ngữ đã được số hoá. Cách dịch thông thường ở Việt Nam là "kho ngữ liệu" hoặc tập huấn luyện trong một số bài báo khoa học. Ví dụ về corpus như "tuyển tập các tác phẩm của Nam Cao", hay "tuyển tập ca từ của Trịnh Công Sơn"...

N-gram:

Là tần suất xuất hiện của n kí tự (hoặc từ) liên tiếp nhau có trong dữ liệu của corpus.

- Với n = 1, *unigram*, và tính trên kí tự, ta có thông tin về tần suất xuất hiện nhiều nhất của các chữ cái. Điều này được ứng dụng để làm keyboard: các phím hay xuất hiện nhất sẽ ở những vị trí dễ sử dụng nhất (e,a,…).
- Với n = 2, ta có khái niệm *bigram*. Ví dụ với các chữ cái tiếng Anh, '*th*','*he*','*in*','*an*','*er*' là các cặp kí tự hay xuất hiện nhất. Ngoài ra, ta có thể biết thêm rằng sau kí tự 'q' thì phần lớn đều là kí tự 'u'.
- Với n = 3, ta có *trigram*. Nhưng vì n càng lớn thì số trường hợp càng lớn nên thường người ta chỉ sử dụng với n = 1,2 hoặc đôi lúc là 3. Ví dụ với các kí tự tiếng Việt, tiếng Việt sử dụng 29 kí tự, vậy với n = 1 thì số trường hợp là 29, n = 2 thì số trường hợp là 29^2 = 841 trường hợp, n = 3 có 24389 trường hợp.

Bigram được sử dụng nhiều trong việc phân tích hình thái (từ, cụm từ, từ loại) cho các ngôn ngữ khó phân tích như tiếng Việt, tiếng Nhật, tiếng Trung, … Dựa vào tần suất xuất hiện cạnh nhau của các từ, người ta sẽ tính cách chia 1 câu thành các từ sao cho tổng bigram là cao nhất có thể. Với thuật giải phân tích hình thái dựa vào trọng số nhỏ nhất, người ta sử dụng n = 1 để xác định tuần suất xuất hiện của các từ và tính trọng số.

Để đảm bảo tính thống kê chính xác đòi hỏi các corpus phải lớn và có tính đại diện cao.

History (về ngôn ngữ)**:** ta hiểu là tiền ngữ, chẳng hạn "ngôn ngữ tự nhiên" và "ngôn ngữ tự chế" có chung tiền ngữ là "ngôn ngữ tự". Cụ thể, với n-gram $w = w_1 \dots w_{n-1}w_n$, thì history của w_n trong w chính là $w_1, w_2 \dots, w_{n-1}$.

N-gram không nhìn thấy (Unseen N-Grams):

Giả sử ta nhìn thấy "xử lý ngôn ngữ" trong tập ngữ liệu, nhưng ta hoàn toàn không tìm thấy "xử lý ngôn ngữ tự", khi đó,

$$p(\text{tự}|\text{xử lý ngôn ngữ}) = 0$$

Khi đó ta nói cụm "xử lý ngôn ngữ tự" là không nhìn thấy, có xác suất là 0.

2.2 Mô hình N-gram

Nhiệm vụ của một mô hình ngôn ngữ là cho biết xác suất của một từ hoặc cụm từ $W = w_1w_2 \dots w_m$ thì $p(W)$ bao nhiêu.

Theo công thức Bayes**:** $\boldsymbol{P(AB) = P(B|A) * P(A)}$ thì:

$$P(w_1w_2 \dots w_n) = P(w_1) * P(w_2|w_1) * P(w_3|w_1w_2) * \dots * P(w_n|w_1w_2 \dots w_{n-1}) \quad (2.1)$$

Theo công thức này, mô hình ngôn ngữ cần phải có một lượng bộ nhớ vô cùng lớn để có thể lưu hết xác suất của tất cả các chuỗi độ dài nhỏ hơn m. Rõ ràng, điều này là không thể khi m là độ dài của các văn bản ngôn ngữ tự nhiên (m có thể tiến tới vô cùng).

2.2.2 Chuỗi Markov

Giả thiết rằng, xác suất tính cho 1 sự kiện:

- Chỉ phụ thuộc vào các history trước.
- Giới hạn bộ nhớ: chỉ có k từ được đưa vào trong history (các từ "cũ hơn" có khả năng ít liên quan), chẳng hạn chuỗi $w = w'w_1 \dots w_k$, ta sẽ coi như $w \sim w_1 \dots w_k$. Ta gọi là **mô hình Markov bậc** k.

2.2.3 Ước lượng xác suất cho mô hình N-gram

Để có thể tính được xác suất của văn bản với lượng bộ nhớ chấp nhận được, ta sử dụng xấp xỉ Markov bậc n, thay vì tính theo (2.1) ta sử dụng công thức (2.2) dưới đây:

$$P(w_i|w_1, w_2, \dots, w_{i-1}) = P(w_i w_{i-n}, w_{i-n+1}, \dots, w_{i-1}) \quad (2.2)$$

Nếu áp dụng xấp xỉ Markov, xác suất xuất hiện của một từ (w_m) được coi như chỉ phụ thuộc vào *n* từ đứng liền trước nó ($w_{i-n} w_{i-n+1} \dots w_{i-1}$) chứ không phải phụ thuộc vào toàn bộ dãy từ đứng trước ($w_1 w_2 \dots w_{i-1}$). Như vậy, công thức tính xác suất văn bản được tính lại theo công thức :

$$\begin{aligned} &P(w_1 w_2 \dots w_i) \\ &\quad = P(w_1) * P(w_2|w_1) * P(w_3|w_1 w_2) * \dots * P(w_{i-1}|w_{i-n-1} w_{i-n} w_{i-2}) \\ &\quad * P(w_i|w_{i-n} w_{i-n+1} w_{i-1}) \end{aligned}$$

Với công thức này, ta có thể xây dựng mô hình ngôn ngữ dựa trên việc thống kê các cụm có ít hơn $n + 1$ từ. Mô hình ngôn ngữ này gọi là mô hình ngôn ngữ N-gram.

2.3 Khó khăn khi xây dựng mô hình ngôn ngữ N-gram

2.3.1 Phân bố không đều

Khi sử dụng mô hình N-gram theo công thức "xác suất thô", sự phân bố không đều trong tập văn bản huấn luyện có thể dẫn đến các ước lượng không chính xác. Khi các N-gram phân bố thưa, nhiều cụm n-gram không xuất hiện hoặc chỉ có số lần xuất hiện nhỏ, việc ước lượng các câu có chứa các cụm n-gram này sẽ có kết quả tồi. Với V là kích thước bộ từ vựng, ta sẽ có V^n cụm N-gram có thể sinh từ bộ từ vựng. Tuy nhiên, thực tế thì số

cụm N-gram có nghĩa và thường gặp chỉ chiếm rất ít.

Ví dụ: tiếng Việt có khoảng hơn 5000 âm tiết khác nhau, ta có tổng số cụm 3-gram có thể có là: 5.000^3 = 125.000.000.000 Tuy nhiên, số cụm 3-gram thống kê được chỉ xấp xỉ 1.500.000. Như vậy sẽ có rất nhiều cụm 3-gram không xuất hiện hoặc chỉ xuất hiện rất ít.

Khi tính toán xác suất của một câu, có rất nhiều trường hợp sẽ gặp cụm Ngram chưa xuất hiện trong dữ liệu huấn luyện bao giờ. Điều này làm xác suất của cả câu bằng 0, trong khi câu đó có thể là một câu hoàn toàn đúng về mặt ngữ pháp và ngữ nghĩa. Để khắc phục tình trạng này, người ta phải sử dụng một số phương pháp "làm mịn" kết quả thống kê.

2.3.2 Kích thước bộ nhớ của mô hình ngôn ngữ

Để khắc phục tình trạng các cụm N-gram phân bố thưa, người ta áp dụng các phương pháp "*làm mịn*" kết quả thống kê nhằm đánh giá chính xác hơn (mịn hơn) xác suất của các cụm N-gram. Các phương pháp "làm mịn" đánh giá lại xác suất của các cụm N-gram bằng cách:

Gán cho các cụm N-gram có xác suất 0 (không xuất hiện) một giá trị khác 0.

Thay đổi lại giá trị xác suất của các cụm N-gram có xác suất khác 0 (có xuất hiện khi thống kê) thành một giá trị phù hợp (tổng xác suất không đổi).

Các phương pháp làm mịn có thể được chia ra thành loại như sau :

- **Chiết khấu (Discounting)**: giảm (lượng nhỏ) xác suất của các cụm Ngram có xác suất lớn hơn 0 để bù cho các cụm Ngram không xuất hiện trong tập huấn luyện.
- **Truy hồi (Back-off) :** tính toán xác suất các cụm Ngram không xuất hiện trong tập huấn luyện dựa vào các cụm Ngram ngắn hơn có xác suất lớn hơn 0.
- **Nội suy (Interpolation):** tính toán xác suất của tất cả các cụm Ngram dựa vào xác suất của các cụm Ngram ngắn hơn.

3 Các phương pháp làm mịn

3.1 Phương pháp chiết khấu (discounting)

Có 3 phương pháp cơ bản và thông dụng hiện nay là Add-one, Good-Turing và Witten-Bell.

Ký hiệu $w \coloneqq w_1 w_2 \dots w_i$ là một cụm n-gram có độ dài i trong tập ngữ liệu mẫu, chẳng hạn "*xử lý ngôn ngữ tự nhiên*" là một cụm n-gram có độ dài 6, đồng thời bộ (xử, lý, ngôn, ngữ, tự, nhiên) là các cụm n-gram có độ dài 1 (unigram), bộ (xử lý, lý ngôn, ngôn

ngữ, ngữ tự, tự nhiên) là các cụm n-gram có độ dài 2 (bigram), … Trong ngữ cảnh khác, ta có thể ký hiệu chính xác hơn: $w_{i-n+1}^{i-1} \coloneqq w_{i-n+1} \dots w_{i-1}$ là cụm n-gram có độ dài n. Ở đây, w_i là các từ tố hay unigram.

3.1.1 Phương pháp làm mịn Add-one (Laplace)

Phương pháp này sẽ cộng thêm vào số lần xuất hiện của mỗi cụm n-gram lên 1, khi đó xác suất của cụm n-gram sẽ được tính lại là:

$$p = \frac{c+1}{n+v} \quad (3.1)$$

Trong đó, c là của số lần xuất hiện cụm n-gram trong tập ngữ liệu mẫu, n là số cụm n-gram, v là kích thước của toàn bộ từ vựng.

(Ở đây, $\sum c = n$ vì thế sau khi thêm 1 vào tần suất xuất hiện mỗi cụm n-gram, tổng này trở thành $\sum(c+1) = n+v$, do đó ta cập nhật lại công thức tính xác suất của cụm n-gram như trên.)

Với Unigram, ta có thể viết lại (3.1) như sau,

$$p^{(1)} = \frac{c^{(1)}+1}{n^{(1)}+v}$$

Ta có, $f^{(1)} = \left(c^{(1)}+1\right) \cdot \frac{n^{(1)}}{n^{(1)}+v}$ là tần suất của unigram, $c^{(1)}$ là số lần xuất hiện của Unigram trước khi làm mịn bằng phương pháp Add-one.

Với cụm n-gram $w \coloneqq w_1 w_2 \dots w_k$, $k > 1$, ta có ,

$$\sum_w C(w_1 w_2 \dots w_{k-1} w) = C(w_1 w_2 \dots w_{k-1})$$

Do đó,

$$P(w_k | w_1 w_2 \dots w_{k-1}) = \frac{C(w_1 w_2 \dots w_k)+1}{C(w_1 w_2 \dots w_{k-1})+V} \quad (3.2)$$

Để ý rằng, có rất nhiều cụm n-gram không nhìn thấy (*bậc thấp*) so với những n-gram nhìn thấy (*bậc cao*). Trong khi đó, có những cụm n-gram có nghĩa (cần thiết) bị giảm đi còn những cụm n-gram tối nghĩa lại có xác suất tăng lên. Để hạn chế điều này, người ta đưa thêm hệ số α thay vì cộng 1 nhằm cân đối lại xác suất (Phương pháp làm mịn Add-α).

$$p = \frac{c + \alpha}{n + \alpha v} \quad (3.3)$$

Trong đó, $\alpha < 1$. Đặc biệt, khi $= \frac{1}{2}$, được gọi là phương pháp Jeffreys – Perks.

Đặt $M = \alpha V$, khi đó (3.2) có thể viết lại thành,

$$P(w_k | w_1 w_2 \dots w_{k-1}) = \frac{C(w_1 w_2 \dots w_k) + M \cdot \left(\frac{1}{V}\right)}{C(w_1 w_2 \dots w_{k-1}) + M}$$

Dễ thấy với một Unigram, tỷ số $\frac{1}{V}$ chính là xác suất xảy ra của mỗi unigram, hay

$$P(w_k | w_{k-1}) = \frac{C(w_{k-1} w_k) + M \cdot P(w_k)}{C(w_{k-1}) + M}$$

3.1.2 Phương pháp làm mịn Good-Turing

Để khắc phục tình trạng một cụm n-gram có nghĩa thực sự có số lần xuất hiện quá nhỏ để thấy được (hay không bao giờ thấy—*unseen*), Alan Turing và cộng sự của mình là I.J. Good sử dụng thuật toán làm mịn dựa trên ước lượng cùng tần suất, nghĩa là nhóm các cụm n-gram có số lần xuất hiện tương đương nhau.

Ký hiệu N_c là số cụm n-gram xuất hiện c lần. Ví dụ,

Tôi học môn học môn sinh học

học 3

Tôi 1

môn 2

sinh 1

Như vậy ta có, $N_1 = 2, N_2 = 1$ và $N_3 = 1$. Vấn đề đặt ra là ta chỉ đếm chính xác được những cụm từ thấy được trong ngữ liệu mẫu, còn với những cụm từ mà ta không thấy được sẽ cần phải ước lượng N_0.

Ví dụ, khi đếm liệt kê các loại cá, từ dữ liệu gốc, ta thu được kết quả:

10 cá chép, 3 cá rô, 2 cá lạt(trắng), 1 cá hồi biển, 1 cá hồi sông, 1 lươn = 18 loại

Như vậy, ước lượng tần suất của loại "cá hồi sông" là $\frac{1}{18}$. Tuy nhiên, với một loại cá mới khác mà ta không có liệt kê ở trên (chẳng hạn cá da trơn) thì ước lượng là bao nhiêu? Để

đơn giản, ta coi đó là $\frac{N_1}{N} = \frac{3}{18}$, vì thế nếu tần suất của "cá hồi sông" phải bé hơn $\frac{1}{18}$ (để cân bằng lại xác suất). Cách tính của thuật toán Good-Turing như sau:

Ký hiệu P^*_{GT} là xác suất của cụm từ tính theo thuật toán Good-Turing, khi đó,

$$P^*_{GT}(c(w) = 0) = \frac{N_1}{N} \quad (3.4)$$

Và số lần đếm được c được thay bằng,

$$c^* = (c+1) \cdot \frac{N_{c+1}}{N_c} \quad (3.5)$$

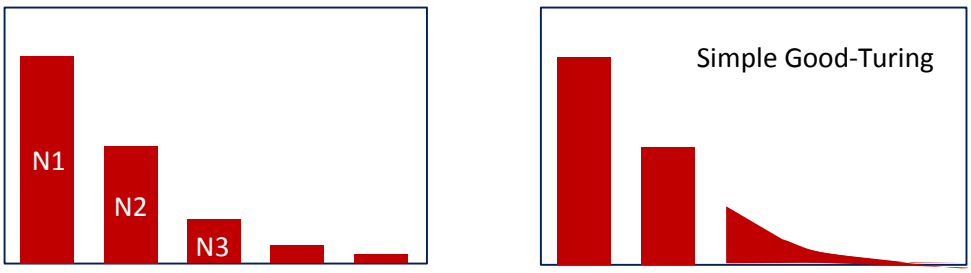
Nói cách khác,

$$P^*_{GT}(unseen) = \frac{N_1}{N} \text{ và } P^*_{GT}(seen) = \frac{c^*}{N} = \frac{(c+1)N_{c+1}}{NN_c} \quad (3.6)$$

Ở ví dụ trên, $P^*_{GT}(unseen) = \frac{N_1}{N} = \frac{3}{18}$,

$$\mathrm{c}^*(\text{cá hồi sông}) = 2 \cdot \frac{\mathrm{N}_2}{\mathrm{N}_1} = 2 \cdot \frac{1}{3} = \frac{2}{3} \Longrightarrow \mathrm{P}^*_{\mathrm{GT}}(\text{cá hồi sông}) = \frac{\frac{2}{3}}{18} = \frac{1}{27}$$

Chú ý rằng, $N_k > N_{k+1}, \forall\, k \geq 1$ (số cụm n-gram xuất hiện càng nhiều sẽ ít dần vì nó càng chiếm kích thước của ngữ liệu huấn luyện). Khi k càng lớn, N_k sẽ đột ngột nhảy về gần 0. Khi đó, cần một thay thế cho N_k bằng một giải thuật đơn giản khác. Gale và Sampson mô tả bằng một giải thuật gọi là *Simple Good-Turing*, thay thế N_k bằng một luật đếm phù hợp nhất so với việc đếm không đảm bảo tin cậy trên.



Hình 3.1

Chứng minh công thức (3.5).

Ta biết rằng, mỗi cụm n-gram riêng biệt α trong tập ngữ liệu đều có xác suất xảy ra là p. Giả thiết tất cả các cụm n-gram α là độc lập với nhau. Khi đó, số lần α xuất hiện trong tập ngữ liệu được xác định:

$$p(c(\alpha) = r) = \binom{N}{r} p^r (1-p)^{N-r}$$

Mục đích chính của phương pháp Good-Turing là nhằm xác định kỳ vọng đếm c^* thay cho số lần xuất hiện c của mỗi n-gram. Kỳ vọng đếm này được xác định như sau:

$$E\big(c^*(\alpha)\big) = \sum_{r=0}^{N} r \cdot p(c(\alpha) = r) = \sum_{r=0}^{N} r \cdot p^r (1-p)^{N-r}$$

Ở đây, giá trị xác suất p là chưa biết, vì thế ta chưa thể xác định ngay được kỳ vọng này.

Tiếp theo ta xác định số kỳ vọng của các cụm n-gram xuất hiện r lần: $E_N(N_r)$. Gọi s là số các cụm n-gram phân biệt từ tập ngữ liệu, giả sử là $\alpha_1, \alpha_2, \dots, \alpha_s$, tương ứng với xác suất $p_1, p_2, \dots, p_s$. Ta có,

$$E_N(N_r) = \sum_{i=1}^{s} p_i(c(\alpha_i) = r) = \sum_{i=1}^{s} \binom{N}{r} p_i^r (1-p_i)^{N-r} \approx N_r \ (r \text{ nhỏ})$$

Ở đây, p_i là chưa biết. Ta đã biết trước c, vì vậy, ta có thể xác định kỳ vọng $E(c^*(\alpha)|c(\alpha) = r)$.

Ta có, xác suất để α là α_i là,

$$p(\alpha = \alpha_i | c(\alpha) = r) = \frac{p(c(\alpha_i) = r)}{\sum_{j=1}^{s} p\big(c\big(\alpha_j\big) = r\big)}$$

Khi đó,

$$E(c^*(\alpha)|c(\alpha) = r) = \sum_{i=1}^{s} N\, p_i\, p(\alpha = \alpha_i | c(\alpha) = r) = \sum_{i=1}^{s} N\, p_i\, \frac{p(c(\alpha_i) = r)}{\sum_{j=1}^{s} p\big(c\big(\alpha_j\big) = r\big)}$$
$$= \frac{\sum_{i=1}^{s} N\, p_i\, p(c(\alpha_i) = r)}{\sum_{j=1}^{s} p\big(c\big(\alpha_j\big) = r\big)}$$

Mặt khác,

$$\sum_{i=1}^{s} N\ p_i\ p(c(\alpha_i) = r) = \sum_{i=1}^{s} N\ p_i\ \binom{N}{r} p_i^r (1-p_i)^{N-r}$$

$$= N\ \frac{N!}{(N-r)!\,r!}\ p_i^{r+1}(1-p_i)^{N-r}$$

$$= N\ \frac{(r+1)}{N+1}\ \frac{(N+1)!}{(N-r)!\,(r+1)!}\ p_i^{r+1}(1-p_i)^{N-r}$$

$$= (r+1)\frac{N}{N+1}\ E_{N+1}(N_{r+1})$$

$$\approx (r+1)\ E_{N+1}(N_{r+1}) \text{ (khi } N \text{ đủ lớn)}$$

Mặt khác, $E_N(N_r) \approx N_r$, suy ra,

$$r^* = E(c^*(\alpha)|c(\alpha) = r) = \frac{(r+1)\ E_{N+1}(N_{r+1})}{E_N(N_r)} = \frac{(r+1)\ N_{r+1}}{N_r}$$

3.1.3 Phương pháp làm mịn Witten-Bell

Cũng như thuật toán Good-Turing, thuật toán Witten-Bell được đưa ra nhằm giải quyết vấn đề về việc đếm các cụm n-gram không nhìn thấy (có số lần đếm là 0). Vớii những n-gram như vậy, ta coi như chúng lần đầu tiên xuất hiện, và việc tính toán có thể dựa vào xác suất gặp cụm n-gram đầu tiên.

Trường hợp, unigram, ta gọi T là số cụm unigram phân biệt đã xuất hiện trong tập ngữ liệu, N là số các cụm unigram đã thống kê, khi đó kích thước dữ liệu sẽ là $T + N$. Do đó, xác suất để gặp cụm unigram lần đầu tiên chưa xuất hiện chính là $\frac{T}{T+N}$.

Gọi V là kích thước của bộ dữ liệu từ vựng, còn Z là số cụm unigram không nhìn thấy (chưa xuất hiện lần nào), ta có: $Z = V - T$. Xác suất để một cụm unigram không nhìn thấy để xuất hiện là:

$$P^* = \frac{T}{Z(T+N)} \quad (3.6)$$

Khi đó, xác suất xuất hiện các cụm unigram w nhìn thấy được tính lại theo công thức:

$$P(w) = \frac{c(w)}{T+N}$$

Đối với một n-gram độ dài $k > 1$, $w = w_1 w_2 \ldots w_k$, Khi đó,

$$P_{C(w)=0}(w_i|w_1 w_2 \dots w_{i-1}) = \frac{T(w_1 w_2 \dots w_{i-1})}{Z(w_1 w_2 \dots w_{i-1})\big(C(w_1 w_2 \dots w_{i-1}) + T(w_1 w_2 \dots w_{i-1})\big)} \quad (3.7)$$

và,

$$P_{C(w)>0}(w_i|w_1 w_2 \dots w_{i-1}) = \frac{C(w_1 w_2 \dots w_{i-1})}{C(w_1 w_2 \dots w_{i-1}) + T(w_1 w_2 \dots w_{i-1})} \quad (3.8)$$

3.2 Phương pháp truy hồi (Back-Off)

Để ý rằng, trong các phương pháp như Add-one (hoặc Add $-\alpha$) nếu như cụm $w_1 \dots w_{i-1}$ không nhìn thấy được thì cụm $w_1 \dots w_{i-1} w_i$ vẫn có xác suất là 0 sau khi được làm mịn. Kết hợp với phương pháp truy hồi dưới đây sẽ khắc phục điều này.

Ta ước lượng xác suất các cụm n-gram không nhìn thấy dựa vào xác suất các cụm n-gram ngắn hơn mà xác suất khác 0.

Cụ thể ta có công thức truy hồi sau đây,

$$P_{BO}^{i}(w_i|w_1 \dots w_{i-1}) = \begin{cases} \alpha_i(w_i|w_1 \dots w_{i-1}) & nếu\ c(w_1 \dots w_i) > 0 \\ d_i(w_1 w_2 \dots w_{i-1}) * P_{BO}^{i-1}(w_i|w_2 \dots w_{i-1}) & nếu\ c(w_1 \dots w_i) = 0 \end{cases} \quad (3.9)$$

Trong đó, $\alpha_i(w_i|w_1 \dots w_{i-1})$ là xác suất (đã điều chỉnh) theo mô hình đã tiên đoán và hàm chiết khấu $d_i(w_1, w_2, \dots, w_{i-1})$.

Ta có thể kết hợp phương pháp truy hồi với các phương pháp chiết khấu để thu được kết quả tốt hơn so với ban đầu. Ví dụ, kết hợp truy hồi với phương pháp làm min Good-Turing,

Xác suất (tương đối) tính cho một unigram,

$$p(w_2|w_1) = \frac{c(w_1, w_2)}{c(w_1)}$$

Thuật toán Good-Turing điều chỉnh count c về kì vọng c^*, với $c^*(w_1, w_2) \leq c(w_1, w_2)$. Khi đó,

$$\alpha(w_2|w_1) = \frac{c^*(w_1, w_2)}{c(w_1)}$$

Điều này dẫn đến công thức cho hàm chiết khấu là

$$d_2(w_1) = 1 - \sum_{w_2} \alpha(w_2|w_1)$$

3.3 Phương pháp nội suy (Recursive Interpolation)

Cũng giống với phương pháp truy hồi, sử dụng các cụm n-gram ngắn hơn để tính xác suất của cụm n-gram dài hơn. Công thức tính xác suất theo phương pháp nội suy như sau:

$$p_I^i(w_i|w_1 w_2 \dots w_{i-1}) = \lambda_{w_1 \dots w_{i-1}}\, p_i(w_i|w_1 \dots w_{i-1}) + \left(1 - \lambda_{w_1 \dots w_{i-1}}\right) p_I^{i-1}(w_i|w_2 \dots w_{i-1}) \quad (3.10)$$

Trong đó, điều kiện $\lambda_{w_1 \dots w_{i-1}}$ là trọng số nội suy. Ví dụ, với bigram và trigram, ta có:

$$p_I^2(w_2|w_1) = \lambda\, p(w_2|w_1) + (1 - \lambda)\, p(w_2)$$

và

$$p_I^3(w_3|w_1 w_2) = \lambda\, p(w_3|w_1 w_2) + (1 - \lambda)\, p(w_1 w_2)$$
$$= \lambda_1\, p(w_3|w_1 w_2) + \lambda_2\, p(w_3|w_2) + \lambda_3(w_3)$$

Lưu ý, trong công thức thứ 2 đối với với trigram, $\lambda = \lambda_1, 1 - \lambda = \lambda_2 + \lambda_3$, do đó $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

3.3 Phương pháp làm mịn Kneser – Ney

Một phần mở rộng cho phương pháp chiết khấu, phương pháp Kneser-Ney (KN) xây dựng theo hai mô hình trùy hồi và nội suy.

3.3.1 Phương pháp Kneser-Ney với mô hình truy hồi

Ta xét một ví dụ, trong một bigram "Bắc Kạn", từ "Kạn" chỉ đi sau từ "Bắc", trong khi đó, từ "Bắc" có số lần đếm nhiều hơn rất nhiều so với từ "Kạn" (có thể không nhìn thấy), xác suất của bigram "Bắc Kạn" bằng 0. Vì thế, ý tưởng của phương pháp này là giảm đi số lần đếm được của những n-gram nhìn thấy (tất nhiển không giảm về 0), bù cho những n-gram không nhìn thấy với hệ số chiết khấu cố định δ (chiết khấu tuyệt đối).

Ký hiệu $N_{1+}(\bullet\, w) = |\{w_i : c(w_i, w) > 0\}|$ là số lượng các n-gram khác nhau kết thúc bởi w và $N_{1+}(\bullet\, \bullet) = \sum_{w_i} N_{1+}(w_i w)$. Khi đó, xác suất đối với trường hợp unigram ($i = 1$) là

$$P_{KN}(w) = \frac{N_{1+}(\bullet\, w)}{N_{1+}(\bullet\, \bullet)}$$

Xác định,

$$\max\{c(w_1 \dots w_i) - \delta, 0\} = \begin{cases} c(w_1 \dots w_i) - \delta & nếu\ c(w_1 \dots w_i) > 0 \\ 0 & nếu\ c(w_1 \dots w_i) = 0 \end{cases}$$

và với các n-gram nhìn thấy:

$$\alpha_i(w_i|w_1 \dots w_{i-1}) = \frac{c(w_1 \dots w_i) - \delta}{\sum_{\substack{w \\ c(w_1 \dots w_i)>0}} c(w_1 \dots w_{i-1}w)}$$

Công thức tính xác suất thu gọn từ (3.9):

$$P_{KN}^{i}(w_i|w_1 \dots w_{i-1}) = \frac{\max\{c(w_1 \dots w_i) - \delta, 0\}}{\sum_{w_i} c(w_1 \dots w_i)} + \frac{\delta}{\sum_{w_i} c(w_1 \dots w_i)} N_{1+}(w_1 \dots w_{i-1} \bullet) P_{KN}^{i-1}(w_i|w_2 \dots w_{i-1}) \quad (3.11)$$

Thông thường, $\delta = \frac{N_1}{N_1+2N_2}$, ở đây, N_c là số đếm các n-gram xuất hiện đúng c lần (tương tự phương pháp Good-Turing, phần 3.1.2).

3.3.1 Phương pháp Kneser-Ney cải tiến

Phương pháp Kneser-Ney được cải tiến bởi Chen-GoodMan, kết hợp cả phương pháp truy hồi và nội suy, cụ thể, công thức tính xác suất vẫn theo phương pháp truy hồi:

$$P_{BO}^{i}(w_i|w_1 \dots w_{i-1}) = \begin{cases} \alpha_i(w_i|w_1 \dots w_{i-1}) & nếu\ c(w_1 \dots w_i) > 0 \\ d_i(w_1w_2 \dots w_{i-1}) * P_{BO}^{i}(w_i|w_2 \dots w_{i-1}) & nếu\ c(w_1 \dots w_i) = 0 \end{cases}$$

Tương tự, ta xác định, xác suất điều chỉnh cho các cụm n-gram nhìn thấy (bậc cao):

$$\alpha_i(w_i|w_1 \dots w_{i-1}) = \frac{c(w_1 \dots w_i) - \delta}{\sum_{\substack{w \\ c(w_1 \dots w_i)>0}} c(w_1 \dots w_{i-1}w)}$$

Nhưng ta xác định giá trị δ theo các trường hợp,

$$\delta(c) = \begin{cases} \delta_1 & nếu\ c = 1 \\ \delta_2 & nếu\ c = 2 \\ \delta_{3+} & nếu\ c \geq 3 \end{cases}$$

Trong đó,

$$\gamma = \frac{N_1}{N_1 + 2N_2}$$

$$\delta_1 = 1 - 2\gamma\frac{\mathrm{N}_2}{N_1}$$

$$\delta_2 = 2 - 3\gamma\frac{\mathrm{N}_3}{N_2}$$

$$\delta_{3+} = 3 - 4\gamma\frac{\mathrm{N}_4}{N_3}$$

Ký hiệu, N_c là số đếm các n-gram xuất hiện đúng c lần.

Với các cụm n-gram không nhìn thấy (bậc thấp):

$$\alpha(w_i|w_1 \dots w_{i-1}) = \frac{N_{1+}(\bullet\, w_1 \dots w_{i-1}w_i) - \delta}{\sum_w N_{1+}(\bullet\, w_1 \dots w_{i-1}w)}$$

trong đó, $\delta = \delta(c)$ được xác định như phần trên.

Hàm chiết khấu tính (chung cho cả các n-gram nhìn thấy và không nhìn thấy)

$$d_i(w_1 w_2 \dots w_{i-1}) = \frac{\sum_{i\in\{1,2,3+\}}\delta_i N_i(w_1 \dots w_{i-1}\,\bullet)}{\sum_{w_i} c(w_1 \dots w_{i-1}w_i)}$$

Lưu ý rằng, nếu chỉ dùng mô hình hay phương pháp Truy hồi đối với các n-gram bậc cao. Nếu như tập ngữ liệu thưa, mô hình này sẽ không đáng tin cậy. Khi có 2 n-gram khác nhau nhưng cùng chung một "history" sẽ có cùng một xác suất, lúc đó n-gram thực sự lại không được chọn trong khi output là một n-gram không chính xác khác trong tập huấn luyện.

Vì vậy, ta luôn xét mô hình truy hồi đối với các n-gram bậc thấp. Hàm điều chỉnh α chuyển thành hàm nội suy α_I bằng cách thêm vào hàm chiết khấu truy hồi.

$$\alpha_I(w_i|w_1 \dots w_{i-1}) = \alpha(w_i|w_1 \dots w_{i-1}) + d(w_1 \dots w_{i-1})\, p_I^{i-1}(w_i|w_2 \dots w_{i-1})$$

Do đó, hàm d cần điều chỉnh thực sự hợp lý để có được mô hình hiệu quả.

4 Kỹ thuật làm giảm kích thước dữ liệu, tối ưu hóa cho mô hình ngôn ngữ

Trước khi đi vào phần chính chúng ta tìm hiểu một số khái niệm liên quan

4.1 Sử dụng Entropy và Độ hỗn loạn thông tin để đánh giá mô hình Ngram

4.1.1 Entropy

Entropy là thước đo thông tin, có giá trị rất lớn trong xử lý ngôn ngữ. Nó thể hiện mức độ thông tin trong ngữ pháp, thể hiện sự phù hợp của một câu với một ngôn ngữ, và dự đoán được từ tiếp theo trong cụm Ngram.

Để tính entropy chúng ta cần tạo 1 biến ngẫu nhiên X liên quan tới thông tin chúng ta cần tính, và thông tin đó phải có 1 hàm xác suất đặc trưng, gọi là $p(x)$. Entropy của biến ngẫu nhiên X này được tính theo công thức:

$$H(X) = -\sum_{X} p(x)\log_2 p(x)$$

Phần log trong công thức về cơ bản có thể tính ở cơ số bất kỳ; nếu sử dụng cơ số 2 thì kết quả của entropy được tính là bit.

Với công thức trên, entropy về cơ bản có thể hiểu là cận dưới số bit cần để mã hóa 1 đoạn thông tin X.

Với 1 mô hình ngôn ngữ, chúng ta cần xem biến ngẫu nhiên X là chuỗi các từ $W = \{\dots w_0, w_1, w_2, \dots, w_n\}$. Như vậy chúng ta có thể tính entropy của 1 biến ngẫu nhiên là các cụm từ có độ dài n trong 1 ngôn ngữ L là:

$$H(w_1, w_2, \dots, w_n) = -\sum_{L} p(W_i^n)\log p(W_i^n)$$

Vậy entropy của mỗi từ sẽ là: $\frac{1}{n}H(W_i^n) = -\frac{1}{n}\sum_L p(W_i^n)\log p(W_i^n)$

Nhưng để tính entropy của 1 ngôn ngữ, chúng ta phải coi các cụm từ có độ dài vô hạn, nghĩa là:

$$H(L) = \lim_{n\to\infty}\frac{1}{n}H(w_1, w_2, \dots, w_n) = \lim_{n\to\infty} -\frac{1}{n}\sum_{L} p(W_i^n)\log p(W_i^n)$$

Định lý Shannon-McMillan-Breiman phát biểu rằng nếu ngôn ngữ sử dụng là "ổn định" thì công thức trên có thể viết lại như sau:

$$H(L) = \lim_{n\to\infty} -\frac{1}{n}\log p\,(w_1, w_2, \dots, w_n)$$

Như vậy, chúng ta có thể lấy 1 cụm từ đủ dài thay vì tính tổng toàn bộ các cụm có thể có. Điểm dễ thấy của định lý trên là 1 cụm đủ dài các từ sẽ chứa trong nó các cụm ngắn hơn, và các cụm ngắn hơn này sẽ lại xuất hiện trong các cụm dài hơn nữa dựa vào xác suất của chúng.

Một kiểu truy xuất ngẫu nhiên được coi là "ổn định" nếu xác suất nó gán cho 1 cụm từ là bất biến khi dịch 1 khoảng thời gian. Nói cách khác, xác suất phân bố cho các từ ở thời điểm t phải bằng xác suất ở thời điểm $t + 1$. Ngram là 1 mô hình ổn định thế này. Tuy nhiên ngôn ngữ tự nhiên thì không ổn định, vì vậy các mô hình thống kê như Ngram chỉ cho các xác suất và entropy xấp xỉ của ngôn ngữ tự nhiên.

Công thức trên đã được biến đổi qua nhiều bước với các xấp xỉ gần đúng, do vậy để tăng tính chính xác khi sử dụng độ đo entropy thì câu kiểm tra cần phải đủ dài và tổng quát (phân tán rộng) để tránh tập trung vào các xác suất lớn (chỉ chứa các cụm thông dụng).

Các bước biến đổi gần đúng công thức trên khiến giá trị $H(L)$ tính theo công thức cuối cùng sẽ lớn hơn giá trị $H(L)$ gốc. Do vậy, khi tính $H(L)$ của các mô hình ngôn ngữ khác nhau trên ngôn ngữ L, mô hình nào cho $H(L)$ nhỏ hơn thì mô hình ngôn ngữ đó thể hiện chính xác ngôn ngữ L hơn.

4.1.2 Độ hỗn loạn thông tin (perplexity)

Độ hỗn loạn thông tin (perplexity) cũng được dùng làm thước đo để đánh giá độ chính xác của một mô hình ngôn ngữ. Trong mô hình ngôn ngữ, độ hỗn loạn thông tin của một văn bản với từ "cái" thể hiện số từ có thể đi sau từ "cái". Độ hỗn loạn thông tin của một mô hình ngôn ngữ nói chung, có thể hiểu đơn giản là số lựa chọn từ trung bình mà mô hình ngôn ngữ phải đưa ra quyết định. Như vậy, độ hỗn loạn thông tin càng thấp, thì độ chính xác của mô hình ngôn ngữ càng cao.

Độ hỗn loạn thông tin của 1 mô hình p trên 1 cụm từ W có thể được tính theo công thức:

$$\text{Perpelexity}(W) = 2^{H(W)} = p(w_1, w_2, \dots, w_n)$$

$$= \sqrt[n]{\frac{1}{p(w_1, w_2, \dots, w_n)}}$$

$$= \sqrt[n]{\prod_{i=1}^{n} \frac{1}{p(w_i | w_1, \dots, w_{i-1})}}$$

Dựa vào công thức ta thấy rằng xác suất điều kiện của cụm từ càng cao thì độ hỗn loạn thông tin càng thấp. Vì vậy giảm thiểu độ hỗn loạn thông tin đồng nghĩa với việc tăng cực đại xác suất của tập thực nghiệm tương ứng với mô hình ngôn ngữ.

Dưới đây là 1 ví dụ về việc sử dụng độ hỗn loạn thông tin để so sánh các mô hình Ngram. Chúng ta huấn luyện unigram, bigram và trigram dưới dạng truy hồi và làm mịn Good-Turing trên tập 38 triệu từ thuộc bộ dữ liệu huấn luyện của Tờ báo phố Wall. Sử dụng 1 từ điển 19,979 từ , chúng ta tính độ hỗn loạn thông tin của mỗi mô hình trên 1 tập thực nghiệm là 1.5 triệu từ bằng công thức trên. Bảng dưới cho kết quả tương ứng:

Ngram	Unigram	Bigram	Trigram
Perpelexity	962	170	109

Dựa vào bảng ta có thể thấy rằng, càng nhiều thông tin mà Ngram cung cấp về các cụm từ thì độ hỗn loạn thông tin càng nhỏ.

4.2 Các kỹ thuật làm giảm kích thước dữ liệu của mô hình ngôn ngữ

Các mô hình ngôn ngữ cho các ứng dụng lớn như nhận diện giọng nói cần phải huấn luyện hàng trăm triệu, thậm chí hàng tỷ từ. 1 mô hình chưa được nén sẽ có kích cớ ngang với kích cỡ dữ liệu được huấn luyện. Có 1 số kỹ thuật đã được dùng để giảm kích thước mô hình ngôn ngữ, và cho đến nay được dùng nhiều nhất là phương pháp pruning (loại bỏ). Phương pháp loại bỏ gồm có các phương pháp cắt bỏ (count cutoffs), loại bỏ dựa vào trọng số khác nhau (Weighted Difference pruning), và loại bỏ Stolcke.

Số lượng các cụm Ngram xuất hiện vài lần trong tập huấn luyện thường là lớn so với tổng số các cụm Ngram. Các cụm Ngram đó thường là lỗi ngữ pháp trong tập huấn luyện, hoặc là một số dạng đặc biệt như: tên riêng, từ viết tắt, ... Những cụm Ngram này thường rất ít sử dụng trong thực tế, do đó việc tồn tại của chúng có thể làm ảnh hưởng đến độ chính xác của mô hình ngôn ngữ. Chính vì lý do đó, kỹ thuật pruning tập trung vào việc loại bỏ các cụm Ngram như vậy. Có 3 phương pháp chính:

- *Count cutoffs* (cắt bỏ): phương pháp này tập trung vào việc loại bỏ các cụm Ngram có tần số thấp trong tập huấn luyện.

- *Weighted Difference pruning* (loại bỏ dựa vào sự khác biệt trọng số): phương pháp này tập trung vào việc đánh giá và loại bỏ các cụm Ngram không hiệu quả dựa vào xác suất của các cụm Ngram trước và sau khi làm mịn theo phương pháp truy hồi.

- *Stolcke pruning*: phương pháp này sẽ loại bỏ các cụm Ngram mà thay đổi độ hỗn loạn thông tin dưới 1 mức cho trước dựa trên entropy quan hệ giữa mô hình gốc và

mô hình đã được giảm kích thước, entropy này có thể được tính toán chính xác trên các mô hình đã được làm mịn theo phương pháp truy hồi.

Sau đây chúng ta sẽ đi sâu vào từng phương pháp.

4.2.1 Count cutoffs

Phương pháp này là phương pháp thông dụng vì tính đơn giản của nó, thường được sử dụng để làm giảm kích thước mô hình ngôn ngữ lưu trữ dưới dạng tần số của các từ. Trong thực tế, trong tập văn bản huấn luyện, có rất nhiều cụm bigram và trigram chỉ xuất hiện một hoặc hai lần trong đoạn văn bản chứa trên một triệu từ. Khi loại bỏ các cụm Ngram này ra khỏi mô hình ngôn ngữ, thông tin về chúng (bao gồm tần số và xác suất) của chúng vẫn có thể nhận lại được thông qua việc sử dụng mô hình truy hồi hay nội suy.

Phương pháp count cutoffs hoạt động như sau: Nếu cụm Ngram xuất hiện ít hơn k lần trong tập văn bản huấn luyện thì cụm Ngram đó sẽ bị loại bỏ ra khỏi mô hình ngôn ngữ. Khi tính toán, nếu gặp lại các cụm Ngram này, thì tần số và xác suất của chúng sẽ được tính toán thông qua các phương pháp làm mịn đã trình bày ở trên.

Trong một mô hình ngôn ngữ, chúng ta có thể sử dụng các tham số k khác nhau với các cụm Ngram có độ dài khác nhau. Ví dụ: với unigram thì sử dụng $k = 10$, với bigram thì $k = 1$, và trigram thì $k = 5$

Như vậy, việc chọn tham số k cho phương pháp count cutoffs chính là vấn đề chính của kỹ thuật này. Nếu k quá lớn, chúng ta sẽ bỏ sót thông tin về một số cụm Ngram, hiệu suất của ứng dụng cũng bị giảm. Nhưng ngược lại, nếu k quá nhỏ, thì kích thước của mô hình ngôn ngữ cũng giảm không đáng kể. Có 2 cách để chọn k: chọn k theo phương pháp chạy thử nhiều lần hoặc chọn k theo tỉ lệ phần trăm số lượng các cụm Ngram.

Chọn k theo phương pháp chạy thử nhiều lần nghĩa là ta dùng phương pháp count cutoffs cho mô hình ngôn ngữ với nhiều giá trị k khác nhau rồi đánh giá độ hỗn loạn thông tin (perplexity) của tập văn bản đầu vào sau khi sử dụng phương pháp count cutoffs. Sau khi có kết quả, ta sẽ chọn tham số k sao cho mô hình ngôn ngữ là hiệu quả nhất (độ hỗn loạn thông tin của tập văn bản huấn luyện và kích thước mô hình ngôn ngữ đều thấp). Kỹ thuật này giúp chúng ta chọn được k phù hợp, tuy nhiên rất mất thời gian do phải chạy thử với rất nhiều giá trị của k. Tuy nhiên, để đạt được một mô hình ngôn ngữ hiệu quả thì đây là một phương pháp tốt.

Phương pháp thứ hai, chọn k dựa theo tỷ lệ phần trăm của số lượng các cụm Ngram phải bảo đảm rằng số cụm Ngram xuất hiện không quá k lần chiếm h% so với tổng số các cụm Ngram. Ví dụ: nếu h=50, thì chọn k sao cho số lượng các cụm Ngram xuất hiện không quá

k lần (sẽ bị loại bỏ) chiếm 50% tổng số các cụm Ngram đã thống kê. Phương pháp này tuy nhanh hơn nhưng độ chính xác không cao bằng phương pháp thứ nhất đã đề cập ở trên.

Vậy nếu cắt bỏ các cụm bigram và trigram theo cách này thì sẽ tiết kiệm bộ nhớ được bao nhiêu? Trong bộ nhận diện giọng nói Sphinx II của đại học Carnegie Mellon, mỗi cụm trigram chiếm 4byte bộ nhớ (2byte cho từ và 2byte cho xác suất) và mỗi cụm bigram chiếm 8byte (2byte cho từ, 2byte cho xác suất, 2byte cho trọng số truy hồi và 2byte cho 1 con trỏ tới các cụm trigram). Bộ nhớ yêu cầu để lưu trữ xác suất các unigram và các hằng có thể được xem là không đổi và sẽ không được tính ở đây. Sử dụng 1 từ điển 58000 từ và 45 triệu từ thuộc dữ liệu huấn luyện của Tờ báo phố Wall(1992 – 1994), các yêu cầu về bộ nhớ của các mô hình được tạo với các hệ số k khác nhau có thể tính được. 1 số mẫu được cho ở bảng dưới, với hệ số cutoff có dạng (cắt bigram – cắt trigram).

Hệ số cutoff	Số các bigram	Số các trigram	Dung lượng (MB)
(0-0)	4,627,551	16,838,937	104
(0-1)	4,627,551	3,581,187	51
(1-1)	1,787,935	3,581,187	29
(0-10)	4,627,551	367,928	38
(10-10)	347,647	367,928	4

Với dữ liệu này, 78,5% trigram và 61% bigram chỉ xuất hiện 1 lần, vì vậy chúng ta có thể thấy dung lượng sẽ tiêt kiệm đáng kể chỉ bằng cách loại bỏ các cụm bigram và trigram xuất hiện ít lần.

4.2.2 Weighted Difference pruning

Phương pháp count cutoffs chỉ quan tâm đến việc loại bỏ các cụm Ngram có tần số thấp, trong khi phương pháp weighted difference thì quan tâm đến nhiều thông tin trong mô hình ngôn ngữ hơn như mối quan hệ giữa các cụm Ngram, xác suất của từng cụm Ngram, ... Như đã trình bày ở các phần trên, nếu một cụm Ngram không xuất hiện trong tập huấn luyện, thì xác suất của nó được ước lượng thông qua xác suất của các cụm Ngram ngắn hơn (phương pháp làm mịn kiểu truy hồi). Do đó, nếu xác suất thực tế của một cụm Ngram xấp xỉ với xác suất có được theo công thức truy hồi, thì chúng ta chẳng cần lưu trữ cụm Ngram đó làm gì nữa. Đó chính là ý tưởng của phương pháp weighted difference. Trọng số khác biệt (weighted difference factor) của một cụm Ngram được định nghĩa bằng:

$$\text{w. d. factor} = \text{K} * \log\Big(\big(\text{xác suất ban đầu}\big) - \log\big(\text{xác suất truy hồi}\big)\Big)$$

Trong đó, K chính là tham số sử dụng trong phương pháp làm mịn Good Turing. Dựa vào trọng số w.d.factor ở trên, chúng ta sẽ biết nên giữ lại hay loại bỏ một cụm Ngram. Nếu

w.d.factor nhỏ hơn một ngưỡng nhất định, thì cụm Ngram đó sẽ bị loại bỏ khỏi mô hình ngôn ngữ. Và ngưỡng nhất định đó chúng ta có thể bằng cách tìm theo phương pháp thử sai hoặc đặt nó bằng một giá trị hằng số.

Trong thực tế, phương pháp này mất nhiều thời gian hơn phương pháp count cutoffs do phải tính toán hệ số w.d.factor cho tất cả các cụm Ngram trong mô hình ngôn ngữ. Và sự khác biệt lớn nhất giữa 2 phương pháp loại bỏ này chính là phương pháp weighted different chỉ hoạt động trong mô hình ngôn ngữ kiểu truy hồi, còn phương pháp count cutoffs thì chỉ hoạt động trong mô hình ngôn ngữ lưu trữ dữ liệu dưới dạng tần số.

4.2.3 Stolcke pruning

Tương tự phương pháp trọng số khác biệt nêu trên, phương pháp nãy cũng chỉ hoạt động với mô hình ngôn ngữ kiểu truy hồi. Nhác lại về mô hình truy hồi, 1 mô hình ngôn ngữ N-gram biểu diễn 1 phân phối xác suất cho 1 cụm từ w, đi kèm bộ (N-1) từ đi trước, gọi là lược sử h. Chỉ 1 tập giới hạn các N-gram (w, h) có xác suất điều kiện tường minh trong mô hình. Các N-gram còn lại được gán 1 xác suất bởi quy luật truy hồi:

$$p(w, h) = \alpha(h) p(w|h')$$

trong đó h' là phần lược sử h khi đã bỏ đi từ đầu tiên và $\alpha(h)$ là 1 hệ số truy hồi tương ứng với lược sử h để chắc chắn rằng $\sum_w p(w|h) = 1$

Mục đích của phương pháp pruning là loại bỏ các xác suất $p(w|h)$ khỏi mô hình, từ đó giảm số lượng các tham số nhưng đồng thời cũng phải giảm thiểu mất mát hiệu năng. Chú ý rằng sau khi loại bỏ, xác suất của các cụm N-gram được giữ lại là không đổi, nhưng trọng số truy hồi cần phải tính lại, từ đó thay đổi giá trị các ước lượng xác suất đã truy hồi. Vì vậy, phương pháp tiếp cận này sẽ độc lập với phương pháp ước lượng được dùng để xác định các xác suất của các cụm N-gram.

Vì 1 trong những mục tiêu là loại bỏ mà không cần truy cập đến bất kỳ thống kê nào không chứa trong mô hình, 1 chuẩn tự nhiên là giảm đi "khoảng cách" của sự phân bố xác suất trong mô hình gốc và mô hình đã được loại bỏ. Một đại lượng chuẩn để tính toán sự khác nhau giữa 2 phân bố đó là entropy quan hệ.

Với $p(\bullet \mid \bullet)$ là xác suất của các cụm Ngram trong mô hình gốc, $p'(\bullet \mid \bullet)$ là xác suất trong mô hình đã loại bỏ thì entropy quan hệ giữa 2 mô hình là:

$$D(p||p') = -\sum_{w_i} p(w_i, h)[\log p'(w_i|h) - \log p(w_i|h)]$$

trong đó tổng này sẽ áp dụng với tất cả các cụm từ w_i và các lược sử h_j.

Mục tiêu của chúng ta sẽ là chọn những cụm N-gram mà có $D(p||p')$ là nhỏ nhất. Tuy nhiên, sẽ khó có thể chọn nhiều nhất có thể các cụm N-gram. Thay vào đó, chúng ta thừa nhận rằng các cụm N-gram ảnh hưởng đến entropy quan hệ 1 cách độc lập, và tính toán $D(p||p')$ dựa vào mỗi cụm N-gram riêng biệt. Sau đó chúng ta có thể xếp hạng các cụm dựa vào sự tác động lên entropy của mô hình, và loại bỏ đi các cụm làm tăng entropy quan hệ lên ít nhất.

Để chọn ngưỡng loại bỏ, chúng ta sẽ xem 1 cách trực quan $D(p||p')$ trong quan hệ với độ hỗn loạn thông tin. Độ hỗn loạn thông tin của mô hình gốc được tính dựa trên các cụm N-gram đã phân bố xác suất là:

$$PP = e^{-\sum_{h,w} p(h,w)\log p(w|h)}$$

Và ta cũng có độ hỗn loạn thông tin của mô hình đã loại bỏ (dựa trên phân bố xác suất của mẫu gốc) là:

$$PP' = e^{-\sum_{h,w} p(h,w)\log p'(w|h)}$$

Như vậy sự thay đổi quan hệ trong độ hỗn loạn thông tin của mô hình bây giờ có thể được tính toán dựa trên entropy quan hệ là:

$$\frac{PP' - PP}{PP} = e^{D(p||p')} - 1$$

Từ đây chúng ta đề xuất 1 thuật toán tạo ngưỡng loại bỏ đơn giản sau:

1. Chọn 1 ngưỡng θ

2. Tính toán độ tăng của độ hỗn loạn thông tin quan hệ dựa vào loại bỏ mỗi cụm N-gram riêng biệt.

3. Loại bỏ tất cả các cụm N-gram làm tăng độ hỗn loạn thông tin nhỏ hơn θ, và tính toán lại trọng số truy hồi.

Tính toán entropy quan hệ

Bây giờ chúng ta sẽ xem cách để tính entropy quan hệ $D(p||p')$ dựa vào việc loại bỏ 1 cụm N-gram 1 cách hiệu quả và chính xác. Chú ý đến việc loại bỏ 1 cụm N-gram gồm từ w và lược sử h là điều này sẽ dẫn đến 2 thay đổi tới ước lượng xác suất

- Trọng số truy hồi $a(h)$ liên quan tới lịch sử h thay đổi, và kéo theo tất cả các xác suất đã truy hồi liên quan đến lược sử h. Chúng ta sử dụng ký hiệu $BO(w_i, h)$để biểu diễn điều này, tức là mô hình gốc sẽ không chứa 1 xác suất cho (w_i, h). Đặt $\alpha(h)$ là trọng số truy hồi cũ, và $\alpha'(h)$ là trọng số truy hồi mới.

- Xác suất $p(w|h)$ được thay bởi 1 xác suất truy hồi:

$$p'(w|h) = \alpha'(h)p(w|h')$$

với h' là lược sử thu được bằng cách bỏ từ đầu tiên trong h

Tất cả các xác suất không liên quan đến lược sử h giữ nguyên không đổi, khi đó gán tất cả các giá trị $BO(w_i, h)$ là false

Thế vào (1) ta có:

$$D(p||p') = -\sum_{w_i} p(w_i, h)[\log p'(w_i|h) - \log p(w_i|h)]$$

$$= -p(w, h)[\log p'(w|h) - \log p(w|h)] - \sum_{w_i: BO(w_i,h)} p(w_i, h)[\log p'(w_i|h) - \log p(w_i|h)]$$

$$= -p(h)\left\{p(w|h)[\log p'(w|h) - \log p(w|h)] + \sum_{w_i: BO(w_i,h)} p(w_i, h)[\log p'(w_i|h) - \log p(w_i|h)]\right\}$$

$$= -p(h)\left\{p(w|h)[\log p(w|h') + \log\alpha'(h) - \log p(w|h)] + \sum_{w_i: BO(w_i,h)} p(w_i, h)[\log\alpha(h') - \log\alpha(h)]\right\}$$

$$= -p(h)\left\{p(w|h)[\log p(w|h') + \log\alpha'(h) - \log p(w|h)] + [\log\alpha(h') - \log\alpha(h)] \sum_{w_i:\, BO(w_i,h)} p(w_i,h)\right\}$$

Tổng ở dòng cuối cùng là cụm toàn bộ xác suất đã được cho để truy hồi, được tính chỉ 1 lần với 1 h, và có thể tính một cách hiệu quả dựa vào tổng tất cả các ước lượng chưa truy hồi:

$$\sum_{w_i:\, BO(w_i,h)} p(w_i,h) = 1 - \sum_{w_i:\, \neg BO(w_i,h)} p(w_i,h)$$

Xác suất $p(h)$ được tính bằng cách nhân các xác suất điều kiện $p(h_1)p(h_2|h_1)$ ...

Cuối cùng, chúng ta cần tính toán trọng số truy hồi hiệu chỉnh $\alpha'(h)$ 1 cách hiệu quả, nghĩa là trong thời gian hằng số với mỗi N-gram. Với công thức tính $\alpha(h)$:

$$\alpha(h) = \frac{1 - \sum_{w_i:\, \neg BO(w_i,h)} p(w_i,h)}{1 - \sum_{w_i:\, \neg BO(w_i,h)} p(w_i,h')}$$

$\alpha'(h)$ có thể tính được bằng cách bỏ các số hạng chứa các N-gram (w,h) đã loại bỏ từ 2 tổng ở cả tử và mẫu. Vì vậy, chúng ta tính tử và mẫu gốc với mỗi lược sử h, sau đó cộng tương ứng $p(w|h)$ và $p(w|h')$ vào tử và mẫu để tính được $\alpha'(h)$ với mỗi cụm từ w đã loại bỏ.

5 Ứng dụng

Mô hình n-gram hiện nay được ứng dụng rộng rãi trong xác xuất, các thuyết về truyền đạt ngôn ngữ, học máy, nén dữ liệu. Một số ứng dụng thực tế gần gũi:

- Đoán trước từ để hoàn thành câu

  Ví dụ:Làm ơn tắt…

  Chương trình của bạn không…

- Dựa đoán các đầu vào văn bản

  Hệ thống dự đoán đầu vào có thể đoán được những gì bạn đang gõ và đưa ra lựa chọn để hoàn thành nó

- Ứng dụng chỉnh đúng khi gõ sai hoặc đánh máy nhầm
- Kết hợp xử lý tiếng nói, tách lời trong một bài hát, một đoạn hội thoại,…

Tài liệu tham khảo

[1] *Xây dựng mô hình ngôn ngữ N-gram cho Tiếng Việt*, Cao Văn Kiệt, Khóa luận tốt nghiệp, K51, Khoa học máy tính, Trường Đại học Công nghệ

[2] *Good-Turing Smoothing Without Tears*, William A. Gale, AT&T Bell Laboratories

[3] *Power Law Discouting for N-gram Language Models*, Songfang Huang, Steve Renals, The Centre for Speech Technology Research, University of Edinburgh, United Kingdom

[4] Book: *Statistical Machine Translation*, slides Chapter 7, *Language models*, Philipp Koehn, http://www.statmt.org/book/

[5] *Class-Based n-gram Models of Natural Language*, Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, Jenifer C. Lai, IBM T. J. Watson Research Center

[6] *N-gram*, Daniel Jurafsky &James H. Martin, 2006

[7] *Entropy-based Pruning of Backoff Language Models*, Andreas Stolcke, Speech Technology And Research Laboratory SRI International, link: here

[8] *Scalable Trigram Backoff Language Models*, Kristie Seymore, Ronald Rosenfeld, 1996, School of Computer Science Carnegie Mellon University Pittsburgh

[9] Lectures online: 4 - 7 - Good-Turing Smoothing - Stanford NLP - Professor Dan Jurafsky & Chris Manning (http://youtu.be/XdjCCkFUBKU)