1. Mô hình ngôn ngữ

Mô hình ngôn ngữ là một phân bố xác suất trên các tập văn bản. Nói đơn giản, mô hình ngôn ngữ có thể cho biết xác suất một câu (hoặc cụm từ) thuộc một ngôn ngữ là bao nhiêu.

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

P[*"hôm qua là thứ năm"*] = 0.001

P[*"năm thứ hôm là qua"*] = 0

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

2. Mô hình ngôn ngữ N-gram

Nhiệm vụ của mô hình ngôn ngữ là cho biết xác suất của một câu $w_1w_2...w_m$ là bao nhiêu. Theo công thức Bayes**: P(AB) = P(B|A) * P(A),** thì:

$$P(w_1w_2\ldots w_m) = P(w_1) * P(w_2|w_1) * P(w_3|w_1w_2) *\ldots* P(w_m|w_1w_2\ldots w_{m-1})$$

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). 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:

$$P(w_m|w_1,w_2,\ldots, w_{m-1}) = P(w_m|w_{m-n},w_{n-m+1}, \ldots,w_{m-1})$$

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 ($w_{m-n}w_{m-n+1}\ldots w_{m-1}$) chứ không phải phụ

thuộc vào toàn bộ dãy từ đứng trước ($w_1w_2\ldots w_{m-1}$). Như vậy, công thức tính xác suất văn bản được tính lại theo công thức:

$$P(w_1w_2\ldots w_m) = P(w_1) * P(w_2|w_1) * P(w_3|w_1w_2) *\ldots* P(w_{m-1}|w_{m-n-1}w_{m-n}\ldots w_{m-2})* P(w_m|w_{m-n}w_{m-n+1}\ldots w_{m-1})$$

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.

Một cụm N-gram là 1 dãy con gồm n phần tử liên tiếp nhau của 1 dãy các phần tử cho trước.

2.1 Công thức tính "xác suất thô":

Gọi $C(w_{i-n+1}...w_{i-1}w_i)$ là tần số xuất hiện của cụm $w_{i-n+1}...w_{i-1}w_i$ trong tập văn bản huấn luyện.

Gọi $P(w_i|w_{i-n+1}...w_{i-1})$ là xác suất w_i đi sau cụm $w_{i-n+1}..w_{i-2}w_{i-1}$.

Ta có công thức tính xác suất như sau:

$$P(w_i|w_{i-n+1}...w_{i-1}) = \frac{C(w_{i-n+1}...w_{i-1}w_i)}{\sum_w C(w_{i-n+1}...w_{i-1}w)}$$

Dễ thấy, $\sum_w C(w_{i-n+1}..w_{i-1}w)$ chính là tần số xuất hiện của cụm $w_{i-n+1}...w_{i-1}$ trong văn bản huấn luyện. Do đó công thức trên viết lại thành:

$$P(w_i|w_{i-n+1}...w_{i-1}) = \frac{C(w_{i-n+1}...w_{i-1}w_i)}{C(w_{i-n+1}...w_{i-1})}$$

Tỉ lệ ở vế phải còn gọi là tỉ lệ tần số. Cách tính xác suất dựa vào tỉ lệ tần số còn gọi là ước lượng xác suất cực đại. Cũng có thể gọi đây là công thức tính "*xác suất thô*" để phân biệt với các cách tính xác suất theo các thuật toán hiệu quả hơn.

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

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. Đề khắc phục tình trạng này, người ta phải sử dụng một số phương pháp "làm mịn".

Kích thước bộ nhớ của mô hình ngôn ngữ

Khi kích thước tập văn bản huấn luyện lớn, số lượng các cụm Ngram và kích thước của mô hình ngôn ngữ cũng rất lớn. Nó không những gây khó khăn trong việc lưu trữ mà còn làm tốc độ xử lý của mô hình ngôn ngữ giảm xuống do bộ nhớ của máy tính là hạn chế. Để xây dựng mô hình ngôn ngữ hiệu quả, chúng ta phải giảm kích thước của mô hình ngôn ngữ mà vẫn đảm bảo độ chính xác.

2.3 Các phương pháp làm mịn

Để khắc phục tình trạng các cụm N-gram phân bố thưa như đã đề cập, người ta đã đưa ra 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.

2.3.1 Các thuật toán chiết khấu (discounting):

Nguyên lý của các thuật toán chiết khấu là giảm xác suất của các cụm Ngram có xác suất lớn hơn 0 đề bù cho các cụm Ngram chưa từng xuất hiện trong tập huấn luyện [10]. Các thuật toán này sẽ trực tiếp làm thay đổi tần số xuất hiện của tất cả các cụm Ngram. Ở đây đề cập đến 3 thuật toán chiết khấu phổ biến:

- Thuật toán Add-one
- Thuật toán Witten-Bell
- Thuật toán Good-Turing

2.3.2Phương pháp làm mịn Add-one:

Thuật toán làm mịn Add-one cộng thêm 1 vào tần số xuất hiện của tất cả các cụm N-gram rồi nhân với phân số chuẩn hóa (để bảo toàn tổng xác suất).

Với unigram, khi cộng thêm 1 vào tần số của mỗi cụm unigram, thì tổng số cụm unigram đã xuất hiện bằng:

M' = M + V với M là tổng số cụm unigram đã xuất hiện

V là kích thước bộ từ vựng

Để bảo toàn tổng số cụm unigram vẫn bằng M, thì tần số mới của các cụm unigram được tính lại theo công thức:

$$C_i^* = (C_i+1)\frac{M}{M'}$$ với C_i là tần số của cụm unigram trước khi làm mịn

Như vậy, xác suất của các cụm unigram cũng được tính lại:

$$P_i^* = \frac{C_i^*}{M} = \frac{(C_i+1)}{M+V}$$

Xét các cụm N-gram với N>1, thay M bằng $C(w_{i-n+1}...w_{i-1})$ thì xác suất của cụm $w_{i-n+1}...w_{i-1}w_i$ được tính theo công thức sau:

$$P(w_i|w_{i-n+1}...w_{i-1}) = \frac{C(w_{i-n+1}...w_{i-1}w_i)+1}{C(w_{i-n+1}...w_{i-1})+V}$$

Chúng ta có thể thấy thuật toán này sẽ làm thay đổi đáng kể xác suất của các cụm Ngram đã xuất hiện trong tập huấn luyện nếu kích thước bộ từ điển V là rất lớn. Trong thực nghiệm, một vài cụm Ngram có xác suất giảm đi gần 10 lần, do kích thước bộ từ điển là lớn trong khi tần số xuất hiện của cụm Ngram đó không cao. Để thuật toán thêm hiệu quả, người ta sử dụng công thức sau:

$$\mathbf{P(w_1w_2...w_n) = \frac{C(w_1w_2...w_n)+\lambda}{C(w_1w_2...w_{n-1})+M\lambda}}$$

Công thức trên là một phiên bản cải tiến thông dụng của thuật toán add-one. Để bảo toàn tổng xác suất của tất cả các cụm Ngram, thì λ được chọn trong khoảng [0, 1], với một số giá trị thông dụng sau:

- λ = 0: không làm mịn
- λ = 1: thuật toán add-one

- $\lambda = \frac{1}{2}$: được gọi là thuật toán Jeffreys - Perks

2.3.3 Phương pháp làm mịn Witten - Bell:

Thuật toán Witten-Bell hoạt động dựa trên nguyên tắc:

Khi gặp những cụm N-gram có tần số 0, ta coi đây là lần đầu tiên cụm từ này xuất hiện. Như vậy, xác suất của cụm N-gram có tần số bằng 0 có thể tính dựa vào xác suất gặp một cụm N-gram lần đầu tiên.

Với unigram, gọi T là số cụm unigram khác nhau đã xuất hiện, còn M là tổng số các cụm unigram đã thống kê, khi đó tổng số sự kiện sẽ là (T+M), và xác suất để gặp cụm unigram lần đầu tiên (hay tổng xác suất của các cụm unigram chưa xuất hiện lần nào) được tính bằng: $\frac{T}{T+M}$

Gọi V là kích thước bộ từ vựng, còn Z là số cụm unigram chưa xuất hiện lần nào: Z = V - T

Xác suất xuất hiện của một cụm unigram chưa xuất hiện lần nào (có tần số bằng 0) được tính bằng:

$$P^* = \frac{T}{Z(T+M)}$$

Và xác suất xuất hiện của các cụm unigram có tần số khác 0 được tính lại theo công thức:

$$P(w) = \frac{c(w)}{T+M}$$ với c(w) là số lần xuất hiện của cụm w

Cũng giống thuật toán add-one, khi xét các cụm N-gram với N>1, thay M bằng $C(w_{i-n+1}...w_{i-1})$ thì xác suất của cụm $w_{i-n+1}...w_{i-1}w_i$ với $C(w_{i-n+1}...w_{i-1}w_i) = 0$ được tính theo công thức sau:

$$P(w_i|w_{i-n+1}...w_{i-1}) = \frac{T(w_{i-n+1}...w_{i-1})}{Z(w_{i-n+1}...w_{i-1})(C(w_{i-n+1}...w_{i-1}) + T(w_{i-n+1}...w_{i-1}))}$$

Với $C(w_{i-n+1}...w_{i-1}w_i) > 0$, thì xác suất cụm $w_{i-n+1}...w_{i-1}w_i$ tính bằng công thức:

$$P(w_i|w_{i-n+1}...w_{i-1}) = \frac{C(w_{i-n+1}...w_{i-1}w_i)}{C(w_{i-n+1}...w_{i-1}) + T(w_{i-n+1}...w_{i-1})}$$

2.3.4 Phương pháp làm mịn Good - Turing:

Thuật toán Good-Turing dựa trên việc tính toán Nc, với Nc là số cụm N-gram xuất hiện c lần. Như vậy:

N_0 là số cụm n-gram có tần số 0 (số cụm N-gram không xuất hiện lần nào)

N_1 là số cụm n-gram có tần số 1 (số cụm N-gram xuất hiện 1 lần)

…

N_c có thể hiểu đơn giản là: $N_c = \sum_{w:count(w)=c} 1$

Khi đó, thuật toán Good-Turing sẽ thay thế tần số c bằng một tần số mới c* theo công thức:

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

Xác suất của một cụm N-gram với tần số là c được tính lại theo công thức:

$$\mathbf{P(w) = \frac{c^*}{N}} \text{ với } \mathbf{N = \sum_{c=0}^{c=\infty} N_c c = \sum_{c=0}^{c=\infty} N_c c^* = \sum_{c=0}^{c=\infty} N_{c+1}(c+1)}$$

Trên thực tế, người ta không tính toán và thay thế mọi tần số c bởi một tần số mới c*. Người ta chọn một ngưỡng k nhất định, và chỉ thay thế tần số c bởi tần số mới c* khi c nhỏ hơn hoặc bằng k, còn nếu c lớn hơn k thì giữ nguyên tần số. Để đơn giản, người ta chọn k đủ lớn dựa vào kết quả huấn luyện (ví dụ giá trị lớn nhất)

2.3.5 Phương pháp truy hồi:

Trong các phương pháp chiết khấu như Add-One hay Witten-Bell, nếu cụm $w_{i-n+1}...w_{i-1}w_i$ không xuất hiện trong tập huấn luyện, và cụm $w_{i-n+1}...w_{i-1}$ cũng không xuất hiện, thì xác suất của cụm $w_{i-n+1}...w_{i-1}w_i$ sau khi làm mịn vẫn bằng 0. Phương pháp truy hồi tránh rắc rối trên bằng cách ước lượng xác suất các cụm Ngram chưa

xuất hiện lần nào dựa vào xác suất của các cụm Ngram ngắn hơn có xác suất khác 0 [10][4].

Cụ thể, xác suất của cụm $w_{i-n+1}...w_{i-1}w_i$ được tính lại theo công thức sau:

$$\mathbf{P_B(w_i|w_{i-n+1}...w_{i-1}) = \begin{cases} P(w_i|w_{i-n+1}...w_{i-1}) & \text{nếu } C(w_{i-n+1}...w_{i-1}w_i) > 0 \\ \alpha * P_B(w_i|w_{i-n+2}...w_{i-1}) & \text{nếu } C(w_{i-n+1}...w_{i-1}w_i) = 0 \end{cases}}$$

Áp dụng cho bigram, ta có:

$$P_B(w_i|w_{i-1}) = \begin{cases} P(w_i|w_{i-1}) \text{ nếu } C(w_{i-1}w_i) > 0 \\ \alpha * P(w_i) \text{ nếu } C(w_{i-1}w_i) = 0 \end{cases}$$

Công thức trên có thể viết lại thành:

$$P_B(w_i|w_{i-1}) = P(w_i|w_{i-1}) + \mu(w_{i-1}w_i) * \alpha * P(w_i) \text{ với } u(x) = \begin{cases} 1 \text{ nếu } C(x) = 0 \\ 0 \text{ nếu } C(x) > 0 \end{cases}$$

Tương tự, khi áp dụng cho trigram ta có:

$$P_B(w_i|w_{i-2}w_{i-1}) = \begin{cases} P(w_i|w_{i-2}w_{i-1}) & \text{nếu } C(w_{i-2}w_{i-1}w_i) > 0 \\ \alpha_1 * P(w_i|w_{i-1}) & \text{nếu } C(w_{i-2}w_{i-1}w_i) = 0 \text{ và } C(w_{i-1}w_i) > 0 \\ \alpha_2 * P(w_i) & \text{nếu } C(w_{i-2}w_{i-1}w_i) = 0 \text{ và } C(w_{i-1}w_i) = 0 \end{cases}$$

Công thức trên cũng có thể viết lại thành:

$$P_B(w_i|w_{i-2}w_{i-1}) = P(w_i|w_{i-2}w_{i-1}) + \mu(w_{i-2}w_{i-1}w_i) * \alpha_1 * P(w_i|w_{i-1}) + \mu(w_{i-1}w_i) * \alpha_2 * P(w_i)$$

Sự chính xác của mô hình truy hồi phụ thuộc vào các tham số α_1 và α_2. Có vài kỹ thuật giúp lựa chọn được những tham số này, tùy theo tập huấn luyện và mô hình ngôn ngữ.

Một cách đơn giản, có thể chọn α_1 và α_2 là các hằng số. Tuy nhiên rất khó có thể chọn được hai hằng số để tổng xác suất của tất cả các cụm Ngram không thay đổi. Việc chọn hằng số không chính xác, sẽ làm ảnh hưởng lớn đến độ chính xác của cả mô hình ngôn ngữ. Do đó, ta có thể chọn tham số α như một hàm của Ngram:

$\alpha_1 = \alpha_1(w_{i-1}w_i)$ và $\alpha_2 = \alpha_2(w_{i-1}w_i)$

Tuy nhiên, trong phương pháp truy hồi, tổng xác suất của tất cả các cụm Ngram sẽ luôn lớn hơn 1, do xác suất của các cụm Ngram đã xuất hiện thì không thay đổi, trong khi xác suất của các cụm Ngram chưa xuất hiện thì được tăng lên. Do đó, để thuật toán chính xác hơn, thì ta cần kết hợp nó với một thuật toán chiết khấu như Witten-Bell hay Good-Turing để làm giảm xác suất của các cụm Ngram đã xuất hiện. Do đó, trong thực tế, chúng ta có công thức sau:

$$P(w_i|w_{i-2}w_{i-1}) = \begin{cases} P'(w_i|w_{i-2}w_{i-1}) & \text{nếu } C(w_{i-2}w_{i-1}w_i) > 0 \\ \alpha_1 * P'(w_i|w_{i-1}) & \text{nếu } C(w_{i-2}w_{i-1}w_i) = 0 \text{ và } C(w_{i-1}w_i) > 0 \\ \alpha_2 * P'(w_i) & \text{nếu } C(w_{i-2}w_{i-1}w_i) = 0 \text{ và } C(w_{i-1}w_i) = 0 \end{cases}$$

Trong đó P' chính là xác suất của cụm Ngram khi áp dụng thuật toán làm mịn chiết khấu.

2.3.6 Phương pháp nội suy:

Phương pháp này có chung nguyên lý với phương pháp truy hồi: "sử dụng các cụm Ngram ngắn hơn để tính xác suất của cụm Ngram dài hơn"[1][2]. Tuy nhiên, phương pháp này khác phương pháp truy hồi ở điểm: phương pháp này không phụ thuộc vào sự xuất hiện của các cụm Ngram.

Công thức tính xác suất theo phương pháp nội suy như sau:

$$\mathbf{P_I(w_i|w_{i-n+1}...w_{i-1}) = \lambda P(w_i|w_{i-n+1}...w_{i-1}) + (1-\lambda)P_I(w_i|w_{i-n+2}...w_{i-1})}$$

Áp dụng cho bigram và trigram ta có:

$$\mathbf{P_I(w_i|w_{i-1}) = \lambda P(w_i|w_{i-1}) + (1-\lambda)P(w_i)}$$

$$\mathbf{P_I(w_i|w_{i-n+1}...w_{i-1}) = \lambda_1 P(w_i|w_{i-2}w_{i-1}) + \lambda_2 P(w_i|w_{i-1}) + \lambda_3 P(w_i) \text{ với } \sum_i \lambda_i = 1}$$

Ở công thức trên, do tổng của tất cả các tham số λ bằng 1 nên để đơn giản ta có thể chọn tất cả λ bằng nhau và bằng $\frac{1}{3}$.

Tuy nhiên, cũng có thể chọn các tham số λ như là một hàm của Ngram:

$\lambda_1 = \lambda_1(w_{i-2}w_{i-1}w_i)$, $\lambda_2 = \lambda_2(w_{i-1}w_i)$ và $\lambda_3 = \lambda_3(w_i)$

2.3.7 Phương pháp làm mịn Kneser - Ney:

Thuật toán Kneser-Ney xây dựng theo hai mô hình: truy hồi và nội suy, tuy nhiên trong thuật toán này không cần phải áp dụng các thuật toán chiết khấu trước khi áp dụng công thức truy hồi.

- Mô hình truy hồi:

$$P_{\text{-----}}(w_i|w_{i-n+1}..w_{i-1}) = \begin{cases} \dfrac{C(w_{i-n+1}...w_i) - D}{C(w_{i-n+1}...w_{i-1})} & \text{nếu } C(w_{i-n+1}...w_i) > 0 \\ \alpha(w_{i-n+1}...w_{i-1})P_{BKN}(w_i|w_{i-n+2}...w_{i-1}) & \text{nếu } C(w_{i-n+1}...w_i) = 0 \end{cases}$$

Trong đó:

- $P_{BKN}(w_i) = \dfrac{N(vw_i) - D}{\sum_w N(vw)}$ với N(vw) là số lượng từ v khác nhau xuất hiện trước w trong tập huấn

- $\alpha(w_{i-n+1}..w_{i-1}) = \dfrac{1 - \dfrac{\sum_{w:C(w_{i-n+1}..w_{i-1}w)>0} C(w_{i-n+1}..w_{i-1}w) - D}{C(w_{i-n+1}..w_{i-1})}}{1 - \sum_{w:C(w_{i-n+1}..w_{i-1}w>0)} P_{BKN}(w|w_{i-n+2}..w_{i-1})}$

Như vậy:

$$P_{BKN}(w_i|w_{i-2}w_{i-1}) = \begin{cases} \dfrac{C(w_{i-2}w_{i-1}w_i) - D}{C(w_{i-2}w_{i-1})} & \text{nếu } C(w_{i-2}w_{i-1}w_i) > 0 \\ \alpha(w_{i-2}w_{i-1})P_{BKN}(w_i|w_{i-1}) & \text{nếu } C(w_{i-2}w_{i-1}w_i) = 0 \end{cases}$$

$$P_{BKN}(w_i|w_{i-1}) = \begin{cases} \dfrac{C(w_{i-1}w_i) - D}{C(w_{i-1})} & \text{nếu } C(w_{i-1}w_i) > 0 \\ \alpha(w_{i-1})P_{BKN}(w_i) & \text{nếu } C(w_{i-1}w_i) = 0 \end{cases}$$

$$P_{BKN}(w_i) = \frac{N(vw_i) - D}{\sum_w N(vw)}$$

- Mô hình nội suy:

$$\mathbf{P_{IKN}(w_i|w_{i-n+1}..w_{i-1}) = \frac{C(w_{i-n+1}..w_i) - D}{C(w_{i-n+1}..w_{i-1})} + \lambda(w_{i-n+1}..w_{i-1})P_{IKN}(w_i|w_{i-n+2}..w_{i-1})}$$

Trong đó: □□ □□

- $\lambda(w_{i-n+1}..w_{i-1}) = \frac{D\ N(w_{i-n+1}..w_{i-1}v)}{C(w_{i-n+1}..w_{i-1})}$ với $N(w_{i-n+1}..w_{i-1}v)$ là số lượng từ v khác nhau xuất hiện liền sau cụm $w_{i-n+1}..w_i$ trong tập huấn luyện
- $P_{IKN}(w_i) = \frac{N(vw_i) - D}{\sum_w N(vw)} + \lambda \frac{1}{V}$ với N(vw) là số lượng từ v khác nhau xuất hiện liền trước từ w trong tập huấn luyện.
- $\lambda = \frac{D\ N(v)}{\sum_w N(vw)}$

Như vậy:

$$P_{IKN}(w_i|w_{i-2}w_{i-1}) = \frac{C(w_{i-2}w_{i-1}w_i) - D}{C(w_{i-2}w_{i-1})} + \lambda(w_{i-2}w_{i-1})P_{IKN}(w_i|w_{i-1})$$

$$P_{IKN}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i) - D}{C(w_{i-1})} + \lambda(w_{i-1})P_{IKN}(w_i)$$

$$P_{IKN}(w_i) = \frac{N(vw_i) - D}{\sum_w N(vw)} + \lambda \frac{1}{V}$$

Trong cả 2 mô hình nội suy và truy hồi, D được chọn: $D = \frac{N_1}{N1 + 2N_2}$

2.3.8 Phương pháp làm mịn Kneser - Ney cải tiến bởi Chen - GoodMan:

Công thức tính toán của thuật toán Kneser-Ney cải tiến bởi Chen và GoodMan giống công thức của thuật toán Kneser-Ney, tuy nhiên hằng số D bị thay đổi.

Chen và GoodMan chọn D như sau:

$$D = \begin{cases} 0 \text{ nếu } c(w_{i-n+1}..w_i) = 0 \\ D_1 \text{ nếu } c(w_{i-n+1}.. w_i) = 1 \\ D_2 \text{ nếu } c(w_{i-n+1}.. w_i) = 2 \\ D_3 \text{ nếu } c(w_{i-n+1}.. w_i) >= 3 \end{cases}$$

Với $Y = \frac{N_1}{(N_1 + 2N_2)}$

$D_1 = 1 - 2Y \frac{N_2}{N_1}$

$D_2 = 1 - 3Y \frac{N_3}{N_2}$

$D_3 = 1 - 4Y \frac{N_4}{N_3}$

Trong đó: N_i là số lượng cụm N-gram có số lần xuất hiện bằng i

Chú ý rằng: với mỗi bậc của N-gram ta lại có một bộ 3 hằng số trên. Điều đó có nghĩa là: unigram, bigram, ... có các hằng số trên là khác nhau.

2.4 Kỹ thuật làm giảm kích thước dữ liệu:

Các kỹ thuật này làm giảm kích thước của mô hình ngôn ngữ. Mặc dù đều có chung một mục tiêu, nhưng mỗi kỹ thuật lại có hiệu quả khác nhau. Có ba kỹ thuật chính, bao gồm:

- **Pruning (loại bỏ):** làm giảm số lượng các cụm Ngram trong mô hình ngôn ngữ bằng cách loại bỏ các cụm Ngram không quan trọng
- **Quantization (lượng tử hóa):** thay đổi cấu trúc thông tin của mỗi cụm Ngram trong mô hình ngôn ngữ.
- **Compression (nén)**: nén cấu trúc dữ liệu sử dụng trong việc lưu trữ các cụm Ngram trong mô hình ngôn ngữ

2.4.1 Loại bỏ (pruning):

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, ... [10] 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 2 phương pháp chính:

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

2.4.1.1 Cắt bỏ (cut-off):

Phương pháp này là phương pháp thông dụng, thường được sử dụng để làm giảm kích thước mô hình ngôn ngữ. 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 cut-off 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 trihf 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 cut-off 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 cut-off 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 cut-off. 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

2.4.1.2 Sự khác biệt trọng số (Weighted difference):

Phương pháp cut-off chỉ quan tâm đến việc loại bỏ các cụm Ngram có tần số thấp, trong khi phương pháp weighted difference(sự khác biệt trọng số) 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, ... [10] 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. Sự khác biệt trọng số của một cụm Ngram được định nghĩa bằng:

w.d.factor = K * log((xác suất ban đầu) - log(xác suất truy hồi))

K chính là tham số sử dụng trong phương pháp làm mịn Good Turing. Dựa vào nhân tố 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 cut-off 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 cut-off thì chỉ hoạt động trong mô hình ngôn ngữ lưu trữ dữ liệu dưới dạng tần số.

2.4.3 Đồng hóa (Quantization):

Thuật toán quantization (đồng hóa) làm giảm số lượng bit dùng để lưu trữ các biến trong mô hình ngôn ngữ. Thuật toán này gồm hai bước chính

Bước thứ nhất, liệt kê và lưu trữ tất cả các tần số của các cụm Ngram vào một bảng. Sau đó, thay thế tần số của các cụm Ngram trong mô hình ngôn ngữ bằng chỉ số của tần số trong bảng. Như vậy, thay vì sử dụng b = $\log_2$(tần số lớn nhất) bit để lưu trữ tần số của một cụm Ngram, thì chúng ta chỉ cần sử dụng b’ = $\log_2$(kích thước của bảng) bit cho mỗi cụm Ngram. Do kích thước của bảng nhỏ hơn nhiều so với giá trị tần số lớn nhất của các cụm Ngram nên b’ < b, tức là kích thước mô hình ngôn ngữ đã giảm so với cách lưu trữ ban đầu.

Tuy nhiên, để tăng tính hiệu quả, ở bước thứ hai, thuật toán này đồng hóa một số giá trị trong bảng tần số. Điều đó có nghĩa là, một số giá trị trong bảng có giá trị gần với nhau sẽ được thay thế bằng một con số chung. Sau bước này, chúng ta sẽ thu được một bảng tần số với ít giá trị hơn, cũng tức là đã làm giảm kích thước của mô hình ngôn ngữ đi một lần nữa.

2.4.4 Nén (Compression):

Mô hình ngôn ngữ nào cũng có một cấu trúc dữ liệu. Do đó nếu cấu trúc dữ liệu đó được nén lại bằng các thuật toán nén, thì kích thước của mô hình ngôn ngữ tất

nhiên là giảm. Tuy nhiên, khi một mô hình ngôn ngữ bị nén, thì độ chính xác và tốc độ của mô hình ngôn ngữ đều giảm (do phải giải nén, hoặc bị mất dữ liệu do thuật toán nén chưa tốt) [10] Do không hiệu quả nên kỹ thuật này hiện nay không còn phổ biến như hai kỹ thuật trên, tuy vẫn được sử dụng bởi Microsoft (trong modul kiểm lỗi chính tả của Microsoft Office 2007)

2.5 Độ đo:

Để xây dựng được một hình ngôn ngữ hiệu quả, chúng ta phải có cách để đánh giá chúng. Dưới đây là một số phương pháp phổ biến để đánh giá một mô hình ngôn ngữ:

- **Entropy** - Độ đo thông tin
- **Perplexity** - Độ hỗn loạn thông tin
- **Error rate** - Tỉ lệ lỗi

2.5.1 Entropy – Độ đo thông tin:

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[1][10]. Entropy của một biến ngẫu nhiên X được tính theo công thức**:**

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

Xét các câu gồm hữu hạn m từ $\mathbf{W = (w_1, w_2, ..., w_m)}$ trong ngôn ngữ L. Ta có công thức tính entropy như sau:

$$H(w_1, w_2, ..., w_m) = - \sum_{W \in L} p(w_1, w_2, ..., w_m)\log_2 p(w_1, w_2, ..., w_m)$$

Từ công thức trên, ta có thể đưa ra công thức tính tỉ lệ entropy trên các từ như sau:

$$\frac{1}{m} H(w_1, w_2, ..., w_m) = - \frac{1}{m} \sum_{W \in L} p(w_1, w_2, ..., w_m) \log_2 p(w_1, w_2, ..., w_m)$$

Thực tế thì tỉ lệ entropy trên các từ thường được sử dụng vì giá trị của nó không phụ thuộc vào độ dài các câu [9]. Tuy nhiên, để tính được entropy của một ngôn ngữ L theo công thức trên thì ta phải xét tới các câu dài vô hạn (tất cả các câu có thể có trong ngôn ngữ L), đó là điều không thể. Do đó, ta có thể tính xấp xỉ tỉ lệ entropy trên các từ theo công thức sau:

$$H(L) = - \lim_{m \to \infty} \frac{1}{m} H(w_1, w_2, ..., w_m)$$

$$= - \lim_{m \to \infty} \frac{1}{m} \sum_{W \in L} p(w_1, w_2, ..., w_m) \log_2 p(w_1, w_2, ..., w_m)$$

Định lý **Shannon-McMillan-Breiman** đã chỉ ra rằng nếu ngôn ngữ ổn định (chứa các câu gồm các từ với cấu trúc thông dụng) thì công thức trên có thể biến đổi thành:

$$H(L) = - \lim_{m \to \infty} \frac{1}{m} \log p(w_1, w_2, ..., w_m)$$

Với công thức trên, ta có thể sử dụng công thức Bayes và xác suất của các n-gram để tính $p(w_1, w_2, ..., w_n)$:

$$H(L) = - \lim_{m \to \infty} \frac{1}{m} \log [p(w_n|w_1 w_2 .. w_{n-1}) * p(w_{n+1}|w_2 w_3 .. w_n) * ... * p(w_m|w_{m-n+1} ... w_{m-1})]$$

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.

2.5.2 Perplexity – Độ hỗn loạn thông tin:

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 có thể tính theo công thức:

$$P(L) = 2^{H(L)}$$

Ví duL dãy kí tự a, b, …, z có perplexity là 26 còn bảng mã ASCII có perplexity là 256.

2.5.3 Error rate – Tỉ lệ lỗi:

Người ta thường sử dụng độ đo entropy và perplexity để so sánh độ chính xác của các mô hình ngôn ngữ khi xây dựng một mô hình ngôn ngữ tổng quát. Trong các bài toán cụ thể, người ta sử dụng tỉ lệ lỗi để so sánh độ chính xác của các mô hình ngôn ngữ [10].

Soát lỗi chính tả: xét tỉ lệ giữa số lỗi phát hiện sai hoặc không phát hiện được trên tổng số lỗi có trong văn bản.

Phân đoạn từ: xét tỉ lệ giữa từ phân đoạn sai trên tổng số từ có trong văn bản

Bỏ dấu tự động: xét tỉ lệ giữa số từ bị bỏ dấu nhầm trên tổng số từ có trong văn bản

Tỉ lệ lỗi thấp chứng tỏ mô hình ngôn ngữ hiệu quả. Việc sử dụng tỉ lệ lỗi để đánh giá đưa lại kết quả chính xác nhất khi muốn chọn lựa mô hình ngôn ngữ phù hợp để giải quyết bài toán cụ thể. Tỉ lệ lỗi thường tỉ lệ thuận với giá trị entropy nhưng đôi khi mức độ tăng/giảm của tỉ lệ lỗi và entropy không đều.