Natural Language Processing

- Nguyễn Trung Hiếu –

Đại học Sư phạm Kỹ thuật Thành phố Hồ Chí Minh

Khoa CNTT

Email: hieuanthonydisward@gmail.com

1. Mô hình ngôn ngữ

- Tính xác suất của câu, từ hoặc chuỗi từ trong ngôn ngữ.

- Một mô hình tốt sẽ có các câu đúng ngữ pháp, trôi chảy hơn là các câu có các từ được xếp ngẫu nhiên

Ví dụ: P("hôm nay trời đẹp") > P("trời đẹp nay hôm")

2. Mô hình ngôn ngữ N-gram

Là mô hình ngôn ngữ xác suất được sử dụng để dự đoán hoặc ước lượng xác suất của một chuỗi từ trong một ngôn ngữ.

Mục tiêu: Tính xác suất của một câu hoặc một chuỗi từ

$$P(W) = P(w_1, w_2, w_3, w_4, w_5, \dots, w_m)$$

Trong đó W là chuỗi các từ, w_1, w_2, w_3,.. là các từ trong câu

Mục đích chính của mô hình là đánh giá hoặc tạo ra các câu có khả năng xuất hiện trong ngôn ngữ tự nhiên, dựa trên các từ và cấu trúc.

Công thức xác suất có điều kiện

- **Xác suất có điều kiện**

$$P(B|A) = \frac{P(A,B)}{P(A)}$$

Tức là xác suất B xảy ra khi biết A đã xảy ra

Xác xuất chung: $P(A,B) = P(A).P(B|A)$

Công thức xác suất có điều kiện với nhiều biến

$$P(A,B,C,D) = P(A).P(B|A).P(C|A,B).P(D|A,B,C)$$

Tức là, xác suất của một chuỗi các sự kiện phụ thuộc vào xác suất của từng sự kiện trước đó.

Chuỗi xác suất(The Chain Rule)

Để tính xác suất một câu hoặc chuỗi từ:
$P(w_1, w_2, w_3, \dots, w_m) =$
$P(w_1).P(w_2|w_1).P(w_3|w_1, w_2) \dots P(w_m|w_1, w_2, \dots, w_{m-1})$

Công thức này được gọi là Quy tắc chuỗi (Chain Rule) trong xác suất, và nó giúp chia nhỏ một xác suất phức tạp thành nhiều xác suất điều kiện đơn giản hơn.

Ví dụ: Cho câu "hôm nay trời đẹp" chuỗi W = {hôm, nay, trời, đẹp}

Áp dụng quy tắc chuỗi:

$P(\text{"hômnaytrờiđẹp"})$
$= P(hôm).P(nay|hôm).P(trời|hôm, nay).P(\text{đẹp}|hôm, nay, trời)$

Vấn đề là, nếu số lượng từ tăng lên thì tính toán sẽ rất khó:

Giả thiết Markov

Để đơn giản hóa, giả thiết Markov cho rằng:

- Một từ chỉ phụ thuộc vào $n-1$ từ ngay trước nó, thay vì toàn bộ câu.
- Điều này dẫn đến mô hình **n-gram**, trong đó mối quan hệ giữa các từ được giới hạn trong $n-1$ từ gần nhất.

Công thức:

$$P(w_m|w_1, w_2, \dots, w_{m-1}) \approx P(w_m|w_{m-n+1}, w_{m-n+2}, \dots., w_{m-1})$$

Ước lượng xác suất n-gram

Định nghĩa mô hình n-gram:

Unigram(n = 1): Mỗi từ độc lập với từ khác:

$$P(w_m) = P(w_m)$$

Bigram(n = 2): Mỗi từ phụ thuộc vào từ ngay trước đó:

$$P(w_m|w_{m-1})$$

Trigram(n = 3): Mỗi từ phụ thuộc vào 2 từ ngay trước đó:

$$P(w_m|w_{m-2}, w_{m-1})$$

- Unigram model: $P(w_1 w_2 \dots wn) \sim \prod_i P(w_i)$
- Bigram model: $P(w_1 w_2 \dots wn) \sim \prod_i P(w_i|w_{i\_1})$
- Trigram model: $P(w_1 w_2 \dots wn) \sim \prod_i P(w_i|w_{i\_1} w_{i\_2})$

Ví dụ với bigram(n=2):
Tính P("hômnaytrờiđẹp")

$$P(\text{"hômnaytrờiđẹp"}) \approx P(hôm).P(nay|hôm).P(trời|\ nay)$$

Thay vì tính hết thì chỉ cần tính xác suất các cặp từ liên tiếp

Ước lượng xác suất bằng tần suất

Xác suất trong mô hình n-gram được ước lượng dựa trên tần suất xuất hiện của các từ trong một tập dữ liệu văn bản lớn (corpus):

Biagram(n = 2): Công thức MLE:

$$P(w_m|w_{m-1}) = \frac{Số\ lần\ xuất\ hiện\ của\ cặp(w_{m-1}, w_m)}{Số\ lần\ xuất\ hiện\ của\ w_{m-1}}$$

Ví dụ:

Tập dữ liệu: "hôm nay trời đẹp. hôm nay mưa."

Count(hôm,nay) = 2

Count(hôm) = 2

P(nay|hôm) = $\frac{2}{2} = 1$

Ví dụ:

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

$$P(I| < s >) = \frac{2}{3} = 0.67 \quad P(Sam| < s >) = \frac{1}{3} = 0.33 \quad P(am|I) = \frac{2}{3}$$

$$P(</s > |Sam) = \frac{1}{2} = 0.5 \quad P(Sam|am) = \frac{1}{2} = 0.5 \quad P(do|I) = \frac{1}{3}$$

Bài tập 1:

Cho các thông tin 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>

Dự đoán các từ tiếp theo:

<s> Sam … </s>

<s> Sam I do … </s>

<s> Sam I am Sam … </s>

<s> do I like … </s>

$$P(Sam|<s>) = \frac{3}{5} = 0.6 \quad P(</s>|Sam) = \frac{2}{5} = 0.4$$

$$P(am|I) = \frac{2}{5} = 0.4 \quad P(Sam|am) = \frac{1}{2} = 0.5$$

$$P(I|Sam) = \frac{3}{5} = 0.6 \quad P(</s>|am) = \frac{1}{5} = 0.2$$

$$P(like|I) = \frac{2}{5} = 0.4 \quad P(I|<s>) = \frac{1}{5} = 0.2$$

$$P(</s>|like) = \frac{2}{3} = 0.67 \quad P(do|I) = \frac{1}{5} = 0.2$$

$$P(like|do) = \frac{1}{2} = 0.5 \quad P(do|<s>) = \frac{1}{5} = 0.2$$

$$P(I|do) = \frac{1}{2} = 0.5 \quad P(Sam|like) = \frac{1}{3} = 0.33$$

<s> Sam I </s>

<s> Sam I do like </s>

<s> Sam I am Sam I </s>

<s> do I like Sam </s>

Xác suất xuất hiện của một câu:

Công thức tổng quát:

$$P(w_1, w_2, .., w_n)$$
$$= P(w1| < s >) \times P(w2|w1) \times P(w3|w2) \times \dots$$
$$\times P(w_n|w_{n-1}) \times P(</s > |w_n)$$

Ví dụ: Cho câu "<s> I want english food </s>"

Với các xác suất:

• P(english|want) = 0.0011

• P(chinese|want) = 0.0065

• P(to|want) = 0.66

• P(eat | to) = 0.28

• P(food | to) = 0

• P(want | spend) = 0

• P (i |) = 0.25

$$P(< s > I\ want\ english\ food| </s >) =$$

$$P(I| < s >) \times P(want|I) \times P(english|want)$$
$$\times P(food|english) \times P(</s > |food)$$

Các vấn đề thực tiễn:

Vấn đề Underflow:

Trong tính toán, underflow xảy ra khi một số quá nhỏ để được biểu diễn bằng kiểu dữ liệu số dấu phẩy động (floating-point). Trong ngữ cảnh của xác suất, chúng ta thường nhân rất nhiều xác suất nhỏ lại với nhau (như trong việc tính xác suất của một câu bằng mô hình bigram).

Giải pháp được đưa ra: Sử dụng logarit

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

Ví dụ:

$$p1 = 0.0001\ p_2 = 0.002\ p_3 = 0.01\ p_4 = 0.1$$

Sử dụng thử logarit cơ số 10(log10):

$$\log_{10}(p_1) = -4\ \log_{10}(p_2) = -2.7(Gần\ đúng)$$

$$\log_{10}(p_3) = -2 \qquad \log_{10}(p_4) = -1$$

$$\log(p_1 \times p_2 \times p_3 \times p_4) = -4 - 2.7 - 2 - 1 = -9.7$$

Các công cụ mô hình ngôn ngữ:

•Google Book N-grams: http://ngrams.googlelabs.com/

•KenLM : https://kheafield.com/code/kenlm/

Tiêu chí đánh giá hiệu quả của mô hình:

Một mô hình ngôn ngữ tốt cần đáp ứng các tiêu chí sau:

- **Ưu tiên câu tốt hơn câu xấu:** Mô hình nên gán xác suất cao hơn cho các câu có nghĩa, ngữ pháp đúng và thường được sử dụng trong thực tế, so với các câu vô nghĩa, sai ngữ pháp hoặc ít khi được sử dụng.
- **Gán xác suất cao hơn cho câu "thực" hoặc "thường xuyên quan sát được":** Tương tự như trên, mô hình nên phân biệt được giữa các câu tự nhiên, thường gặp trong ngôn ngữ, với các câu ít gặp hoặc không tự nhiên.

Quy trình đánh giá:

Để đánh giá mô hình, chúng ta tuân theo quy trình sau:

1. **Tập huấn luyện (Training set):** Chúng ta sử dụng một tập dữ liệu lớn (tập huấn luyện) để huấn luyện các tham số của mô hình. Trong quá trình huấn luyện, mô hình "học" các quy tắc và đặc trưng của ngôn ngữ từ dữ liệu này.

2. **Tập kiểm tra (Test set):** Để đánh giá khách quan hiệu suất của mô hình, chúng ta sử dụng một tập dữ liệu *khác* với tập huấn luyện, được gọi là tập kiểm tra. Tập kiểm tra này chứa các câu mà mô hình *chưa từng thấy* trong quá trình huấn luyện. Điều này rất quan trọng để đảm bảo đánh giá không bị thiên vị (overfitting).
3. **Độ đo đánh giá (Evaluation metric):** Chúng ta sử dụng một độ đo cụ thể để định lượng hiệu suất của mô hình trên tập kiểm tra. Độ đo này cho chúng ta biết mô hình hoạt động tốt như thế nào trên dữ liệu mới.

Tầm quan trọng của tập kiểm tra:

Việc sử dụng một tập kiểm tra riêng biệt là rất quan trọng vì:

- **Tránh overfitting:** Nếu chúng ta đánh giá mô hình trên chính tập huấn luyện, mô hình có thể "nhớ" các câu trong tập huấn luyện và đạt kết quả rất tốt, nhưng lại hoạt động kém trên dữ liệu mới. Điều này gọi là overfitting.
- **Đánh giá khách quan:** Tập kiểm tra cho phép chúng ta đánh giá khả năng tổng quát hóa của mô hình, tức là khả năng áp dụng kiến thức đã học vào dữ liệu mới.

Đánh giá ngoại sinh:

Đánh giá ngoại sinh (extrinsic evaluation), còn được gọi là đánh giá dựa trên nhiệm vụ (task-based evaluation), đo lường hiệu suất của một mô hình bằng cách nhúng nó vào một ứng dụng hoặc nhiệm vụ thực tế và xem nó ảnh hưởng đến hiệu suất của nhiệm vụ đó như thế nào.

Quy trình đánh giá ngoại sinh:

Để so sánh hai mô hình N-gram A và B bằng phương pháp đánh giá ngoại sinh, chúng ta thực hiện các bước sau:

1. **Chọn một nhiệm vụ:** Chọn một nhiệm vụ cụ thể mà mô hình ngôn ngữ sẽ được sử dụng. Ví dụ:
    - **Spelling corrector (Công cụ sửa lỗi chính tả):** Mô hình ngôn ngữ giúp xác định từ nào là đúng chính tả trong các từ có thể bị sai.
    - **Speech recognizer (Hệ thống nhận dạng giọng nói):** Mô hình ngôn ngữ giúp xác định chuỗi từ nào phù hợp nhất với tín hiệu âm thanh đầu vào.
    - **Machine translation (Hệ thống dịch máy):** Mô hình ngôn ngữ giúp tạo ra bản dịch tự nhiên và ngữ pháp đúng.
2. **Nhúng mô hình vào nhiệm vụ:** Thay thế mô hình ngôn ngữ hiện tại trong ứng dụng bằng mô hình A, sau đó bằng mô hình B.
3. **Chạy nhiệm vụ và đo lường độ chính xác:** Chạy ứng dụng với từng mô hình và đo lường hiệu suất của nó. Các độ đo có thể là:
    - **Số từ được sửa lỗi chính tả đúng:** Trong công cụ sửa lỗi chính tả.
    - **Số từ được dịch đúng:** Trong hệ thống dịch máy (thường dùng các độ đo như BLEU, METEOR).
    - **Word Error Rate (WER - Tỷ lệ lỗi từ):** Trong hệ thống nhận dạng giọng nói.
4. **So sánh độ chính xác:** So sánh kết quả của mô hình A và mô hình B. Mô hình nào đạt độ chính xác cao hơn được coi là tốt hơn cho nhiệm vụ đó.

Đánh giá nội tại (Intrinsic Evaluation):

Đánh giá nội tại tập trung vào việc đánh giá trực tiếp chất lượng của mô hình ngôn ngữ, độc lập với bất kỳ ứng dụng cụ thể nào. Nó đo lường khả năng của mô hình trong việc dự đoán xác suất của dữ liệu ngôn ngữ.

Perplexity (Độ phức tạp):

Perplexity là một độ đo phổ biến được sử dụng trong đánh giá nội tại của mô hình ngôn ngữ. Nó đo lường mức độ "bối rối" của mô hình khi gặp một chuỗi từ. Perplexity càng thấp, mô hình dự đoán chuỗi từ càng tốt.

Công thức perplexity:

Cho một chuỗi từ $w_1, w_2, \dots, w_N$,perplexity được tính như sau:

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

- $P(w_1, w_2, \dots., w_n)$ *là xác suất của chuỗi từ*
- N là số lượng từ trong chuỗi(Là số lượng tất cả cặp từ, tính cả <s> và </s>)

Branching Factor(hệ số phân nhánh)

Branching Factor là số lượng lựa chọn trung bình mà mô hình phải xem xét cho mỗi vị trí trong chuỗi. Ví dụ, nếu chúng ta đang dự đoán chữ cái tiếp theo trong một từ tiếng Anh, và trung bình có khoảng 26 lựa chọn (26 chữ cái), thì hệ số phân nhánh là 26.

Ví dụ: Có 10 chữ số, và mô hình gán xác suất bằng nhau là 1/10 cho mỗi chữ số. Điều này có nghĩa là mô hình "không biết" chữ số nào sẽ xuất hiện tiếp theo và coi tất cả 10 chữ số đều có khả năng như nhau.

Vì mỗi chữ số được gán xác suất 1/10 và chúng ta giả định các chữ số độc lập với nhau, perplexity của một chuỗi N chữ số là:

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

Nguy cơ của overfitting

Overfitting xảy ra khi một mô hình học quá "sát" dữ liệu huấn luyện, đến mức nó "nhớ" các chi tiết cụ thể của dữ liệu huấn luyện thay vì học các đặc trưng tổng quát của dữ liệu. Kết quả là, mô hình hoạt động rất tốt trên dữ liệu huấn luyện, nhưng lại hoạt động kém trên dữ liệu mới (dữ liệu kiểm tra).

Ảnh hưởng của overfitting

Mô hình N-gram chỉ hoạt động tốt cho việc dự đoán từ nếu tập dữ liệu kiểm tra giống với tập dữ liệu huấn luyện): Đây là hệ quả trực tiếp của overfitting. Nếu mô hình N-gram chỉ "nhớ" các n-gram xuất hiện trong dữ liệu huấn luyện, nó sẽ gặp khó khăn khi gặp các n-gram mới trong dữ liệu kiểm tra.

- Cách giải quyết: Để khắc phục vấn đề overfitting, chúng ta cần huấn luyện các mô hình có khả năng "generalize" (khái quát hóa), tức là khả năng áp dụng kiến thức đã học vào dữ liệu mới.

Một cách để khái quát hóa: Zeros

Vấn đề "Zeros" (Số không):

Trong mô hình N-gram, nếu một n-gram chưa bao giờ xuất hiện trong dữ liệu huấn luyện, xác suất của nó sẽ được tính là 0 theo Maximum Likelihood Estimation (MLE). Điều này gây ra vấn đề khi tính xác suất của một câu chứa n-gram đó, vì kết quả sẽ là 0 (do phép nhân với 0).

Các kỹ thuật xử lý "Zeros" (Smoothing - Làm mịn):

Để giải quyết vấn đề "Zeros" và cải thiện khả năng khái quát hóa của mô hình, chúng ta sử dụng các kỹ thuật "smoothing" (làm mịn). Các kỹ thuật này phân bổ một phần xác suất cho các n-gram chưa được nhìn thấy, tránh việc gán xác suất 0. Một số kỹ thuật smoothing phổ biến bao gồm:

- **Add-one smoothing (Laplace smoothing):** Cộng 1 vào số đếm của tất cả các n-gram.
- **Add-k smoothing:** Cộng một giá trị k (nhỏ hơn 1) vào số đếm của tất cả các n-gram.
- **Good-Turing smoothing:** Ước tính xác suất của các n-gram chưa được nhìn thấy dựa trên tần suất của các n-gram đã được nhìn thấy một lần.
- **Kneser-Ney smoothing:** Một phương pháp smoothing phức tạp và hiệu quả hơn.

Vấn đề Dữ liệu Thưa thớt (Sparse Statistics):

Trong mô hình ngôn ngữ N-gram, chúng ta tính xác suất của một từ dựa trên N từ đứng trước nó. Tuy nhiên, trong thực tế, không phải tất cả các N-gram có thể xuất hiện trong dữ liệu huấn luyện. Điều này dẫn đến vấn đề "dữ liệu thưa thớt", tức là có nhiều N-gram có số đếm bằng 0.

Một giải pháp có thể sử dụng(mượn một phần của các xác suất để chia cho các xác suất khác), ví dụ:

Thay vì:

- P(allegations | denied the) = 3/7
- P(reports | denied the) = 2/7
- P(claims | denied the) = 1/7
- P(request | denied the) = 1/7

Chúng ta "đánh cắp" một phần xác suất và phân bổ cho các từ khác:

- P(allegations | denied the) = 2.5/7
- P(reports | denied the) = 1.5/7
- P(claims | denied the) = 0.5/7

- P(request | denied the) = 0.5/7
- P(other | denied the) = 2/7 (chúng ta tạo ra một nhóm "other" để phân bổ xác suất còn lại)

Laplace Smothing(Ước tính cộng một)

Maximum Likelihood Estimation (MLE) được sử dụng để ước tính xác suất của một bigram (hoặc N-gram) dựa trên tần suất xuất hiện của nó trong dữ liệu huấn luyện. Công thức MLE cho bigram là:

$$P_{MLE}(w_i|w_{i-1}) = \frac{c(w_{i-1}|w_i)}{c(w_{i-1})}$$

$P_{MLE}(w_i|w_{i-1})$: Xác suất của từ w_i xuất hiện sau từ w_{i-1} theo MLE

$c(w_{i-1}|w_i)$: $Số\ lần\ bigram\ (w_{i-1}|w_i)$ xuất hiện trong train data(cặp từ trước sau-hiện tại)

$c(w_{i-1})$: Số lần từ w_{i-1} xuất hiện trong dataset

Giải pháp: Add-one estimation (Làm mịn Laplace):

Add-one estimation là một kỹ thuật làm mịn đơn giản nhất để giải quyết vấn đề "zeros". Ý tưởng cơ bản là "giả vờ" rằng chúng ta đã nhìn thấy mỗi bigram *thêm một lần* so với thực tế. Điều này được thực hiện bằng cách cộng 1 vào số đếm của tất cả các bigram.

Công thức Add-one estimation cho bigram là:

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

V: Tổng số từ khác nhau trong tập train

$P_{Add-1}(w_i|w_{i-1})$: Xác suất của từ w_i xuất hiện sau từ w_{i-1} theo ước tính Add- One

Tham khảo:

Tham khảo từ slide môn xử lý ngôn ngữ tự nhiên của HUST

Nguồn Slide: https://github.com/hoanglong1712/Dai-Hoc-Bach-Khoa-Ha-Noi/tree/main/Natural%20Language%20Processing