

COLE.VN
connecting knowledge

Chủ đề:
Mô hình ngôn ngữ
Trích chọn đặc trưng trong NLP

Mục đích buổi học

- Học viên nắm được các kỹ thuật trích chọn đặc trưng trong xử lý ngôn ngữ tự nhiên

Nội dung chính

Mô hình ngôn ngữ

Trích chọn đặc trưng

Mô hình ngôn ngữ

Mô hình hoá ngôn ngữ

- Mô hình ngôn ngữ là mô hình dự đoán xác xuất của một chuỗi các từ.
 - $P(W) = P(w_1, w_2, \dots, w_n)$
 - Ví dụ:
 - $S_1 = \text{"con mèo nhảy qua con chó"}$, $P(S_1) \sim 1$
 - $S_2 = \text{"qua con mèo con chó nhảy"}$, $P(S_2) \sim 0$

Ứng dụng

- Dịch máy
 - $P(\text{high winds tonight}) > P(\text{large winds tonight})$
- Sửa lỗi văn bản
 - The office is about fifteen **minuets** from my house
 - $P(\text{"about fifteen minutes from"}) > P(\text{about fifteen minuets from})$
- Nhận dạng giọng nói
 - $P(\text{I saw a van}) > P(\text{eyes awe of an})$
- Nhận dạng chữ viết
 - $P(\text{Act naturally}) > P(\text{Abt naturally})$
- Tóm tắt, hỏi – đáp, ..

Xác suất có điều kiện

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

$$P(A, B) = P(A) \cdot P(B|A)$$

$$P(A, B, C, D) = P(A) \cdot P(B|A) \cdot P(C|A, B) \cdot P(D|A, B, C)$$

Xác suất có điều kiện

$$P(S) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1, w_2) \cdots P(w_n|w_1, w_2, \dots, w_{n-1})$$

$$P(S) = \prod_i^n P(w_i|w_1, w_2, \dots, w_{i-1})$$

$$\begin{aligned} P(\text{Computer, can, recognize, speech}) &= P(\text{Computer}) \cdot \\ &\quad P(\text{can}|\text{Computer}) \cdot \\ &\quad P(\text{recognize}|\text{Computer can}) \cdot \\ &\quad P(\text{speech}|\text{Computer can recognize}) \end{aligned}$$

Mô hình ngôn ngữ

- Mô Hình Ngôn Ngữ = mô hình dự đoán từ
- Quy tắc dây chuyền

$$P(x_1, x_2, x_3, \dots, x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)\dots P(x_n|x_1, \dots, x_{n-1})$$

- Ví dụ
 - $P(\text{"its water is so transparent"}) = P(\text{its}) \times P(\text{water}|\text{its}) \times P(\text{is}|\text{its water}) \times P(\text{so}|\text{its water is}) \times P(\text{transparent}|\text{its water is so})$
- Mô hình ngôn ngữ là mô hình dự đoán từ dựa trên các từ phía trước

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 w_2 \dots w_{i-1})$$

Xác suất của từ

- Tính xác suất của từ trong ngữ cảnh dựa trên N-gram

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 w_2 \dots w_{i-1})$$

- $P(\text{speech}|\text{Computer can recognize}) = ?$

$$P(\text{speech}|\text{Computer can recognize}) = \frac{\#(\text{Computer can recognize speech})}{\#(\text{Computer can recognize})}$$

Có vấn đề gì với cách tính trên?

Giả thuyết Markov

$$P(S) = \prod_{i=1}^n P(w_i | w_1, w_2, \dots, w_{i-1})$$



Andrei Markov

$$P(S) = \prod_{i=1}^n P(w_i | w_{i-1})$$

Giả thuyết Markov

$$P(\text{Computer, can, recognize, speech}) = P(\text{Computer}) \cdot P(\text{can}|\text{Computer}) \cdot P(\text{recognize}|\text{Computer can}) \cdot P(\text{speech}|\text{Computer can recognize})$$



$$P(\text{Computer, can, recognize, speech}) = P(\text{Computer}) \cdot P(\text{can}|\text{Computer}) \cdot P(\text{recognize}|can) \cdot P(\text{speech}|recognize)$$

$$P(\text{speech}|\text{recognize}) = \frac{\#(\text{recognize speech})}{\#(\text{recognize})}$$

Mô hình n-gram

Unigram (1-gram): xác suất của một từ không phụ thuộc vào từ phía trước

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i)$$

Bigram (2-gram): xác suất một từ xuất hiện sau một từ cho trước

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i | w_{i-1})$$

Trigram (3-gram): xác suất một từ phụ thuộc vào 2 từ phía trước

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i | w_{i-1}, w_{i-2})$$

N-gram: xác suất 1 từ phụ thuộc vào N từ phía trước

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i | w_{i-1}, w_{i-2}, \dots, w_{i-N})$$

Ước lượng xác suất

Sử dụng Maximum Likelihood Estimate

$$P(w_i | w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

$$P(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Ví dụ

$$P(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>

<s> Sam I am </s>

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

$$P(\text{I} | \text{<s>}) = \frac{2}{3} = .67$$

$$P(\text{Sam} | \text{<s>}) = \frac{1}{3} = .33$$

$$P(\text{am} | \text{I}) = \frac{2}{3} = .67$$

$$P(\text{</s>} | \text{Sam}) = \frac{1}{2} = 0.5$$

$$P(\text{Sam} | \text{am}) = \frac{1}{2} = .5$$

$$P(\text{do} | \text{I}) = \frac{1}{3} = .33$$

Thống kê trên dữ liệu

- Berkeley Restaurant Project (BeRP)
- BeRP chứa các câu tự vấn trong lĩnh vực nhà hàng thành phố Berkeley, California
- Chứa 9222 câu. Ví dụ:
 - 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
 - ...

Đếm đồng xuất hiện

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Xác xuất Bigram

Chuẩn hóa bằng Unigrams:

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

Kết quả

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Xác xuất Bigram

Tính xác suất câu dựa trên các bigram

$$P(<\text{s}> \text{ I want english food } </\text{s}>) =$$

$$P(\text{I}|<\text{s}>)$$

$$\times P(\text{want}|\text{I})$$

$$\times P(\text{english}|\text{want})$$

$$\times P(\text{food}|\text{english})$$

$$\times P(</\text{s}>|\text{food})$$

$$= .000031$$

Tính xác suất câu dựa trên các bigram

Các xác suất đã tính được

- $P(\text{english}|\text{want}) = .0011$
- $P(\text{chinese}|\text{want}) = .0065$
- $P(\text{to}|\text{want}) = .66$
- $P(\text{eat} | \text{to}) = .28$
- $P(\text{food} | \text{to}) = 0$
- $P(\text{want} | \text{spend}) = 0$
- $P (\text{i} | \langle s \rangle) = .25$

Các mô hình ngôn ngữ có sẵn

Google Book N-grams

- <http://ngrams.googlecode.com/>

KenLM

- <https://kheafield.com/code/kenlm/>

Extrinsic evaluation - Đánh giá ngoài

- Mô hình ngôn ngữ A và B được sử dụng trong một bài toán X khác:
 - Bài toán speech recognition, spelling, machine translation
- So sánh mô hình A với mô hình B tương ứng với so sánh kết quả ứng dụng A với B trong bài toán X.

Intrinsic evaluation - Đánh giá trong

- Sử dụng dữ liệu Test là các câu trong ngôn ngữ
- Sử dụng độ đo Perplexity (độ phức tạp)
 - Đánh giá xấp xỉ không tốt
 - Chỉ khi dữ liệu test giống dữ liệu train (về bộ từ vựng)
 - Tốt cho thí nghiệm nhưng không tốt cho thực tế

Ý tưởng của Perplexity

Shannon Game:

- Ta có thể tiên đoán từ tiếp theo không?

I always order pizza with cheese and _____

The 33rd President of the US was _____

I saw a _____

- Có thể dùng unigram không?

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

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

Độ phức tạp (Perplexity)

Độ phức tạp tương đương số trường hợp rẽ nhánh

- Giả thiết 1 câu gồm các chữ số ngẫu nhiên. Khi đó độ phức tạp của câu dựa trên 1 mô hình sẽ gán $P=1/10$ đ/v mỗi chữ số.

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

Độ phức tạp (Perplexity)

Độ phức tạp tương đương số trường hợp rẽ nhánh

- Giả thiết 1 câu gồm các chữ số ngẫu nhiên. Khi đó độ phức tạp của câu dựa trên 1 mô hình sẽ gán $P=1/10$ đ/v mỗi chữ số.

Cực tiểu perplexity tương đương với việc cực đại xác suất

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

Độ phức tạp (Perplexity)

Perplexity thấp = mô hình tốt

- Bộ dữ liệu WSJ
- Tập huấn luyện 38 triệu từ, kiểm tra 1.5 triệu từ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Giá trị n

Giá trị phù hợp của n là bao nhiêu?

- Về mặt lý thuyết, rất khó xác định
- Tuy nhiên: nhiều nhất có thể (\rightarrow tiệm cận với mô hình “hoàn hảo”)
- Về mặt thực nghiệm, phổ biến **n = 3**
 - Ước lượng tham số? (độ tin cậy, dữ liệu, lưu trữ, không gian, ...)
 - 4 là quá lớn: $|V|=60k \rightarrow 1.296 \times 10^{19}$ tham số
 - nhưng: 6-7 có thể nếu có đủ dữ liệu: trong thực tế, chúng ta có thể khôi phục bản gốc từ 7-grams!

Hiện tượng quá khớp dữ liệu (overfitting)

N-grams chỉ tiên đoán từ tốt nếu tập test giống tập train.

Ta cần tạo ra mô hình có tính tổng quát, nghĩa là có thể xử lý các trường hợp xác suất = 0 (những TH không có trong tập train nhưng có trong tập test)

- Tập train:

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

- Tập test

- ... denied the offer
- ... denied the loan

$$P(\text{"offer"} \mid \text{denied the}) = 0$$

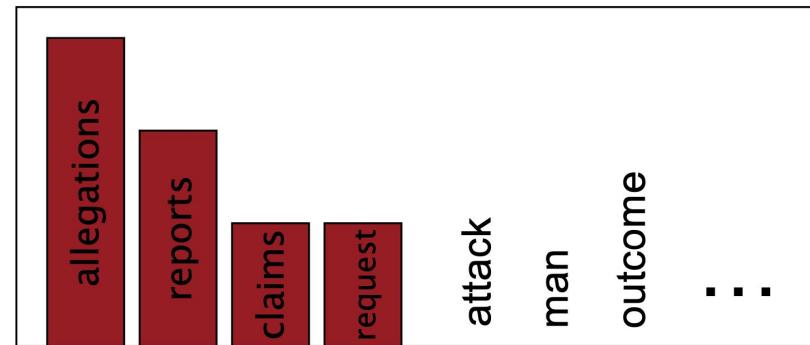
→ xác suất của 1 câu hoặc một cụm từ về 0

Vấn đề làm mịn

Khi chúng ta có thống kê thừa:

$P(w | \text{denied the})$

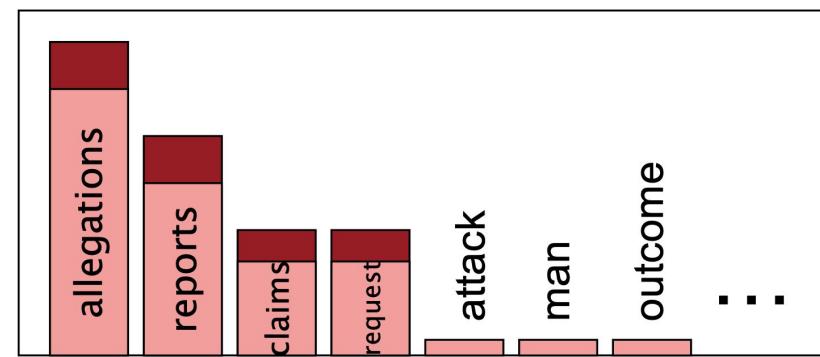
- 3 allegations
- 2 reports
- 1 claims
- 1 request



Xác suất trên tập train:

Giảm xác suất các n-gram có xác suất lớn hơn 0 để bù cho các n-gram có xác suất bằng 0.

- $P(w | \text{denied the})$
- 2.5 allegations
- 1.5 reports
- 0.5 claims
- 0.5 request
- 2 other



Trích chọn đặc trưng

Phân loại văn bản

Đầu vào:

- Một văn bản d
- Một tập các lớp $C = \{c_1, c_2, \dots, c_J\}$

Đầu ra:

Dự đoán lớp $c \in C$ của d

Luật thủ công

- Chọn đặc trưng
- Luật (Rules) dựa vào kết hợp các đặc trưng của từ hoặc các đặc trưng khác
 - spam: black-list-address OR (“dollars” AND “have been selected”)
- Độ chính xác có thể cao
 - Nếu luật được thiết kế cẩn thận bởi chuyên gia
- Xây dựng và duy trì những bộ luật này là tốn kém

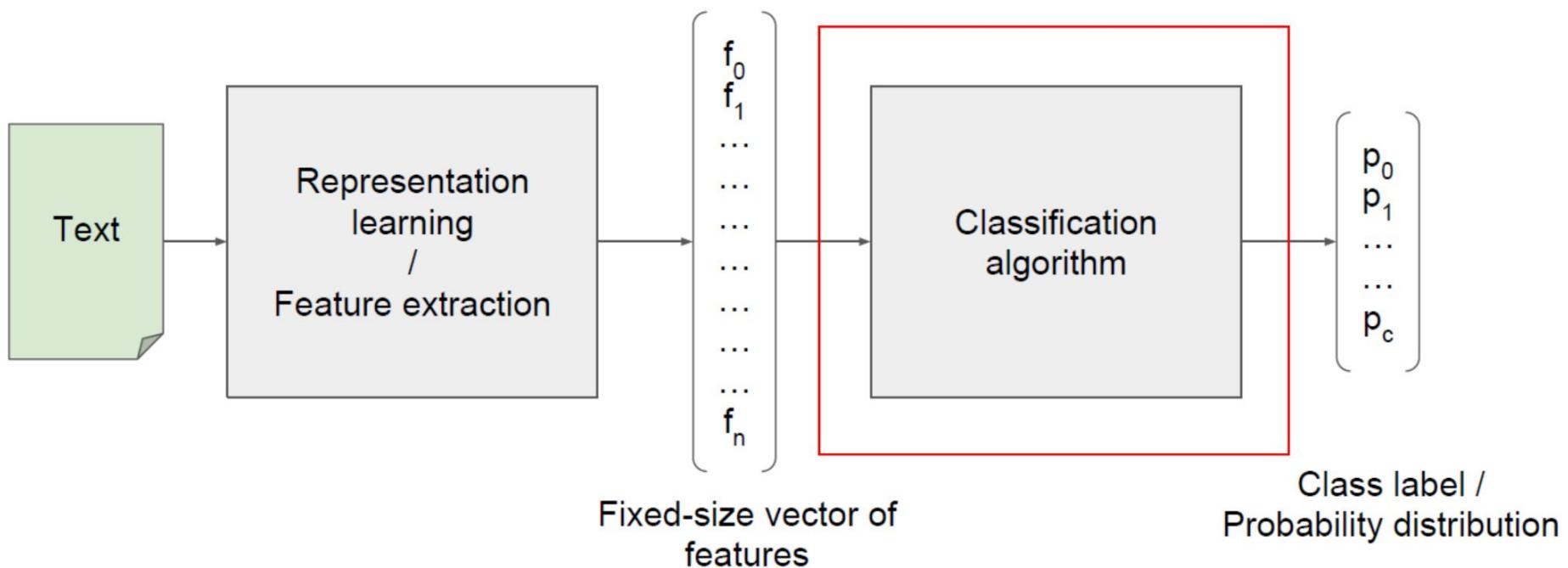
Học máy có giám sát

- Đầu vào:
 - Một văn bản (email) d
 - Một tập các lớp $C = \{c_1, c_2, \dots, c_J\}$
 - Một tập dữ liệu huấn luyện gồm m mẫu đã được gán nhãn
 $(d_1, c_1), \dots, (d_m, c_m)$
- Đầu ra:
 - Một bộ phân lớp $\gamma: d \rightarrow c$

Một số phương pháp

- Sử dụng bất kỳ các loại phân lớp
 - Naïve Bayes
 - Logistic regression
 - Support-vector machines
 - Neural Network
 - ...

Hệ thống phân loại văn bản

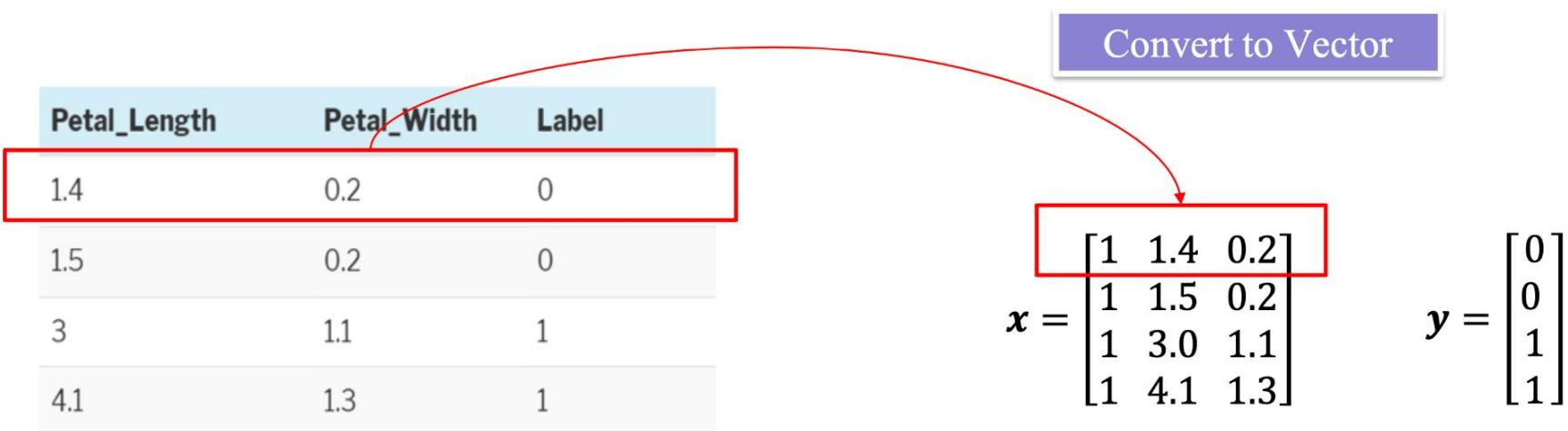


Đặc trưng (Features)

- Một biến có thể đo lường, khác biệt của một thứ mà chúng ta muốn lập mô hình.
- Chúng ta thường chọn đặc trưng mà hữu ích để xác định thứ gì đó, ví dụ như để phân lớp (loại).
 - **Ví dụ:** Cô gái đó **rất đẹp** trong bữa tiệc hôm đó.
- Chúng ta thường cần một số đặc trưng để lập mô hình đầy đủ.
 - *nhưng không quá nhiều!*

Đặc trưng (Features)

- Numeric Representation

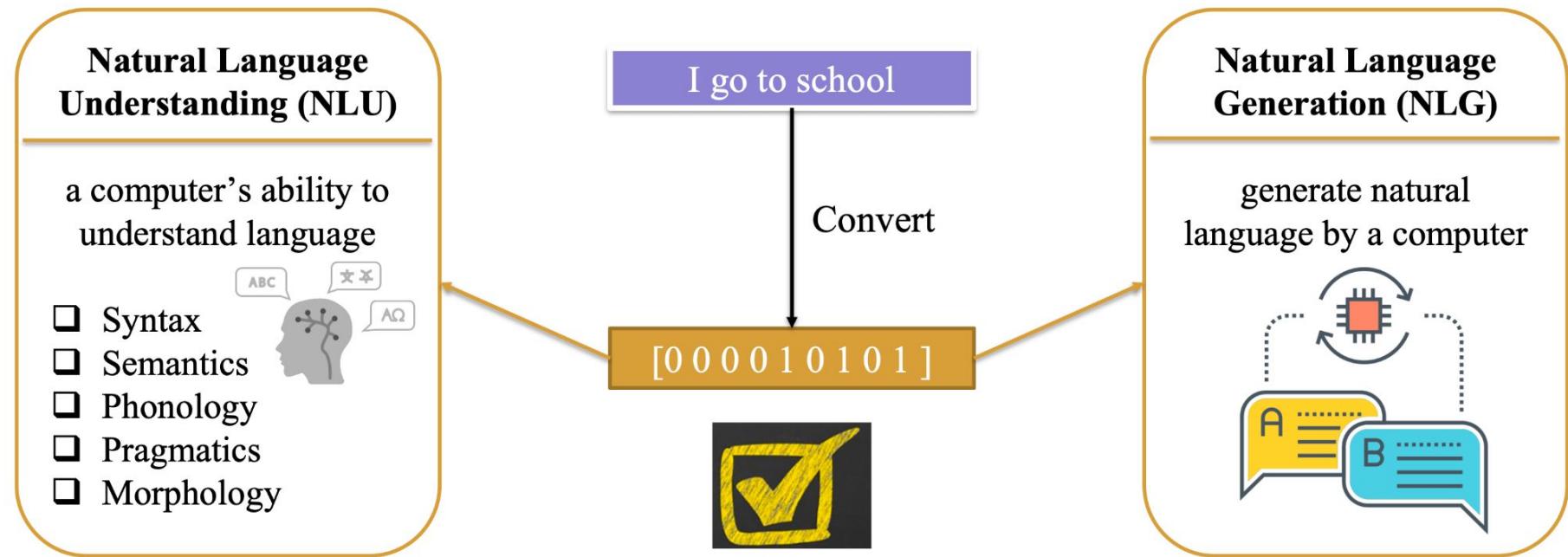


Đặc trưng (Features)

- Numeric Representation



Từ văn bản sang đặc trưng

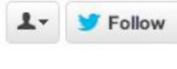


Có thể việc sinh ra các đặc trưng hữu ích bằng thủ công (feature engineering)

- Đặc trưng thống kê: độ dài, vị trí, ...
- Sử dụng điểm từ các từ điển:
 - Sentiment dictionaries: SentiWordNet, SentiWords, ...
 - Subjectivity/objectivity dictionaries: MPQA
- Các đặc trưng cú pháp:
 - POS tags
- Đặc trưng cho bài toán: e.g. number of emojis (😜 , 😍)

Vector đặc trưng

Giá trị của một vài đặc trưng của một quan sát có thể đưa vào một vector

 Damien Fahey 

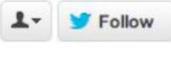
Rush Limbaugh looks like if someone put a normal human being in landscape mode.

 Reply  Retweet  Favorite  More

 Faux John Madden 

BREAKING: Apple Maps projecting Barack Obama to win Brazil.

 Reply  Retweet  Favorite  More

 Jim Gaffigan 

If there was an award for most pessimistic, I probably wouldn't even be nominated.

 Reply  Retweet  Favorite  More

# proper nouns	# 1st person pronouns	# commas
2	0	0
5	0	0
0	1	1

Vector đặc trưng

Các đặc trưng sẽ hữu ích trong việc phân biệt giữa các thể loại

Table 3: Features to be computed for each text

- Counts:
 - First person pronouns
 - Second person pronouns
 - Third person pronouns
 - Coordinating conjunctions
 - Past-tense verbs
 - Future-tense verbs
 - Commas
 - Colons and semi-colons
 - Dashes
 - Parentheses
 - Ellipses
 - Common nouns
 - Proper nouns
 - Adverbs
 - *wh*-words
 - Modern slang acronyms
 - Words all in upper case (at least 2 letters long)
 - Average length of sentences (in tokens)
 - Average length of tokens, excluding punctuation tokens (in characters)
 - Number of sentences
-
- The diagram consists of three colored arrows pointing from specific sections of the feature list to interpretation boxes. A blue arrow points from the first section of counts to a blue box. A green arrow points from the second section of counts to a green box. An orange arrow points from the third section of counts to an orange box.
- Higher values → this person is referring to themselves (to their opinion, too?)**
- Higher values → looking forward to (or dreading) some future event?**
- Lower values → this tweet is more formal. Perhaps not overly sentimental?**

Acti

Biểu diễn văn bản

Basic Representation

- One-hot Encoding
- Bag of Words (BoW)
- Bag of N-grams
- TF-IDF

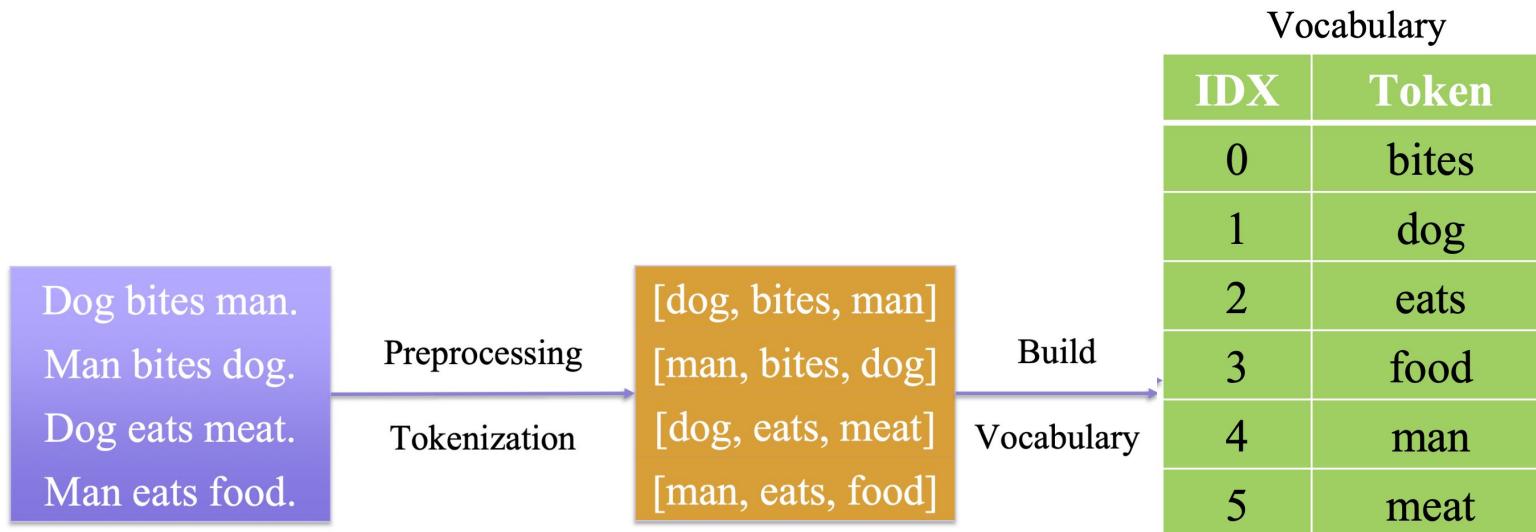
Rome = [1, 0, 0, 0, 0, 0, ..., 0]
Paris = [0, 1, 0, 0, 0, 0, ..., 0]
Italy = [0, 0, 1, 0, 0, 0, ..., 0]
France = [0, 0, 0, 1, 0, 0, ..., 0]

word V → 0

The diagram illustrates the basic representation of words as vectors. It shows four words: Rome, Paris, Italy, and France, each mapped to a specific vector in a high-dimensional space. The vectors are represented as lists of binary values (0s and 1s). The word 'Rome' is mapped to [1, 0, 0, 0, 0, 0, ..., 0]. The word 'Paris' is mapped to [0, 1, 0, 0, 0, 0, ..., 0]. The word 'Italy' is mapped to [0, 0, 1, 0, 0, 0, ..., 0]. The word 'France' is mapped to [0, 0, 0, 1, 0, 0, ..., 0]. An arrow labeled 'word V' points to the final zero in the vector for France, indicating that the dimensionality of the vector is very large.

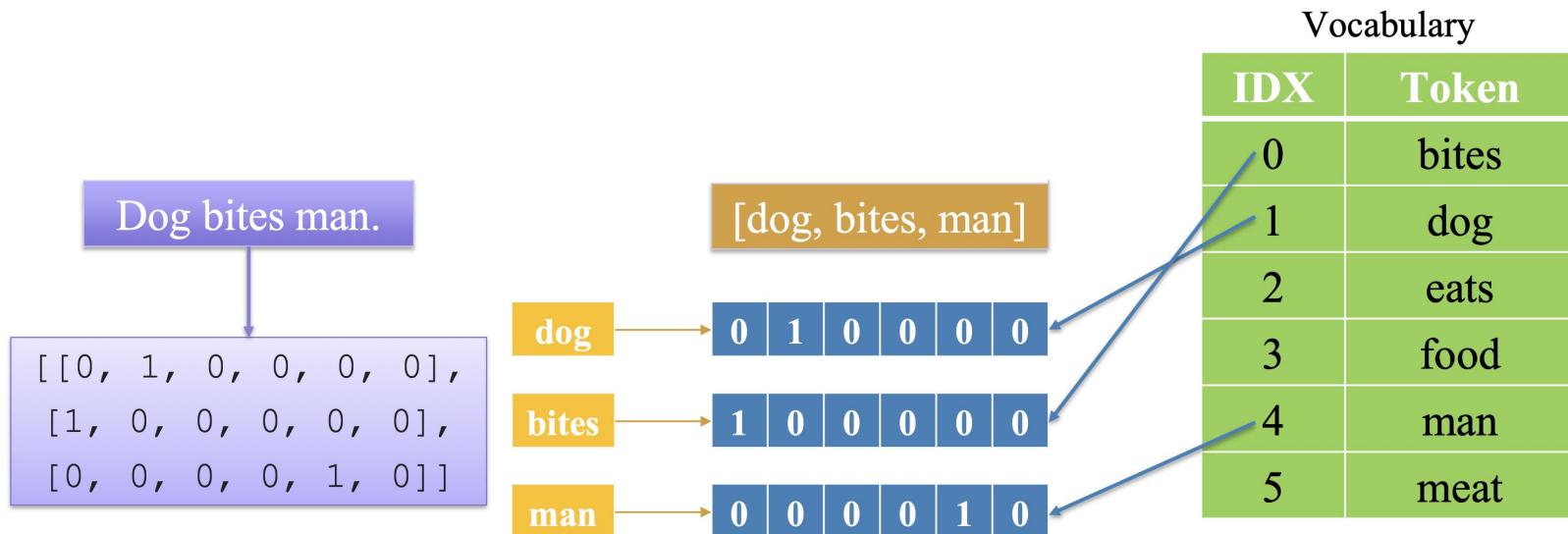
One-hot encoding

- Token-Level
- Được biểu diễn bằng vectơ nhị phân



One-hot encoding

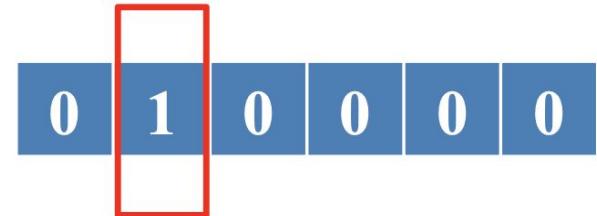
- Token-Level
- Được biểu diễn bằng vectơ nhị phân



One-hot encoding

Nhược điểm

- Ma trận thưa (hầu hết là số 0)
 - Số chiều của Vector tỷ lệ thuận với kích thước của số lượng từ vựng
- Không nắm bắt được ý nghĩa của từ
 - Chỉ bao gồm các số 0 và 1
- Out of vocabulary (OOV)



and ← How to represent

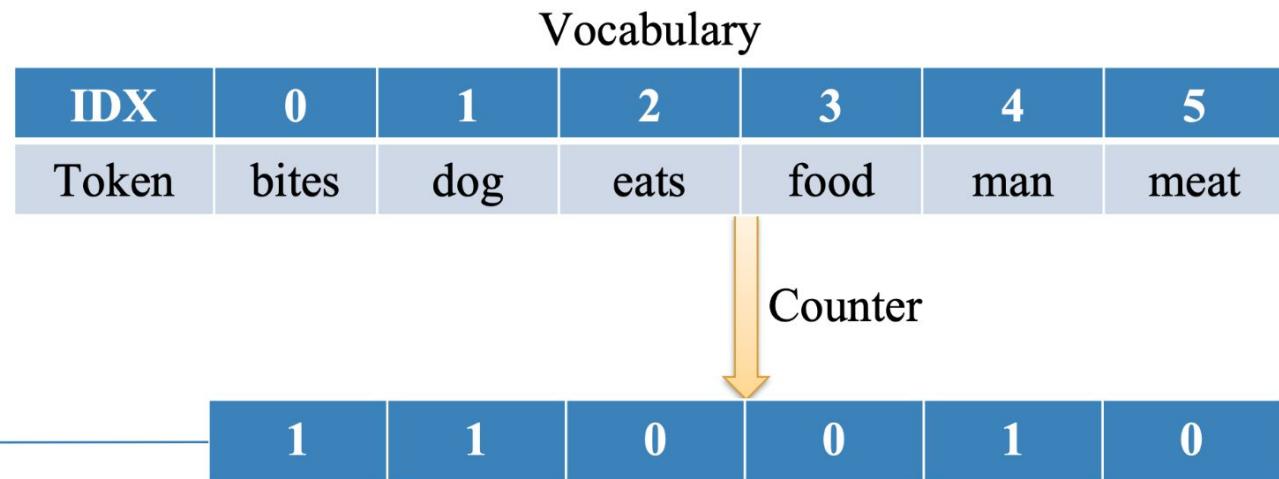
Vocabulary	
IDX	Token
0	bites
1	dog
2	eats
3	food
4	man
5	meat

Bag of Words (BoW)

Document-Level: Coi văn bản như một túi (bộ sưu tập) các từ

Được biểu diễn bằng vectơ n-chiều

- số lần xuất hiện của từ trong tài liệu

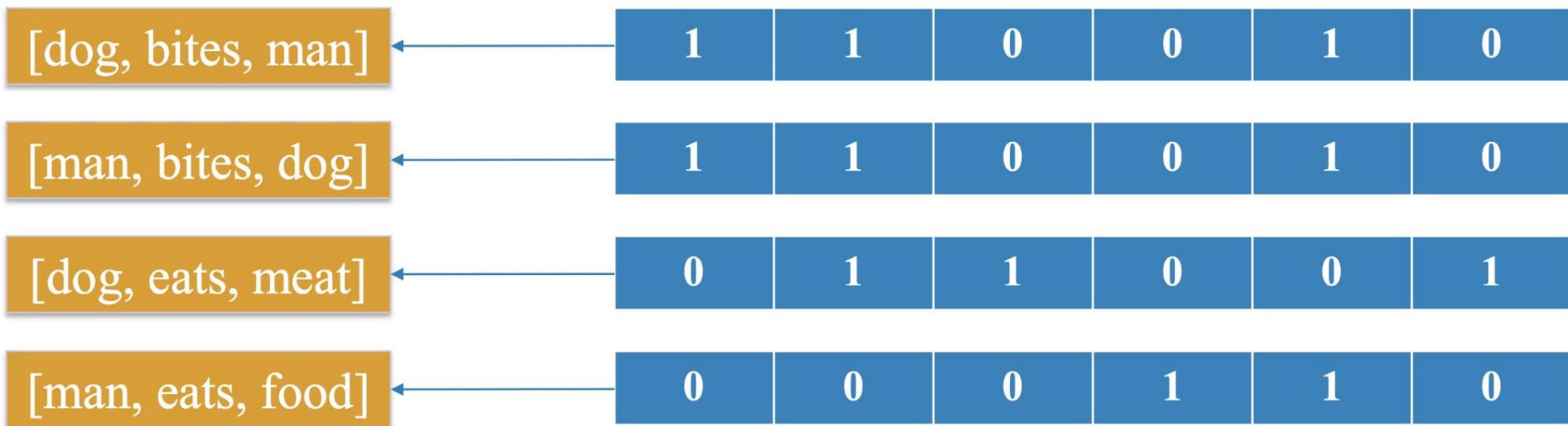


Bag of Words (BoW)

Document-Level: Coi văn bản như một túi (bộ sưu tập) các từ

Được biểu diễn bằng vectơ n-chiều

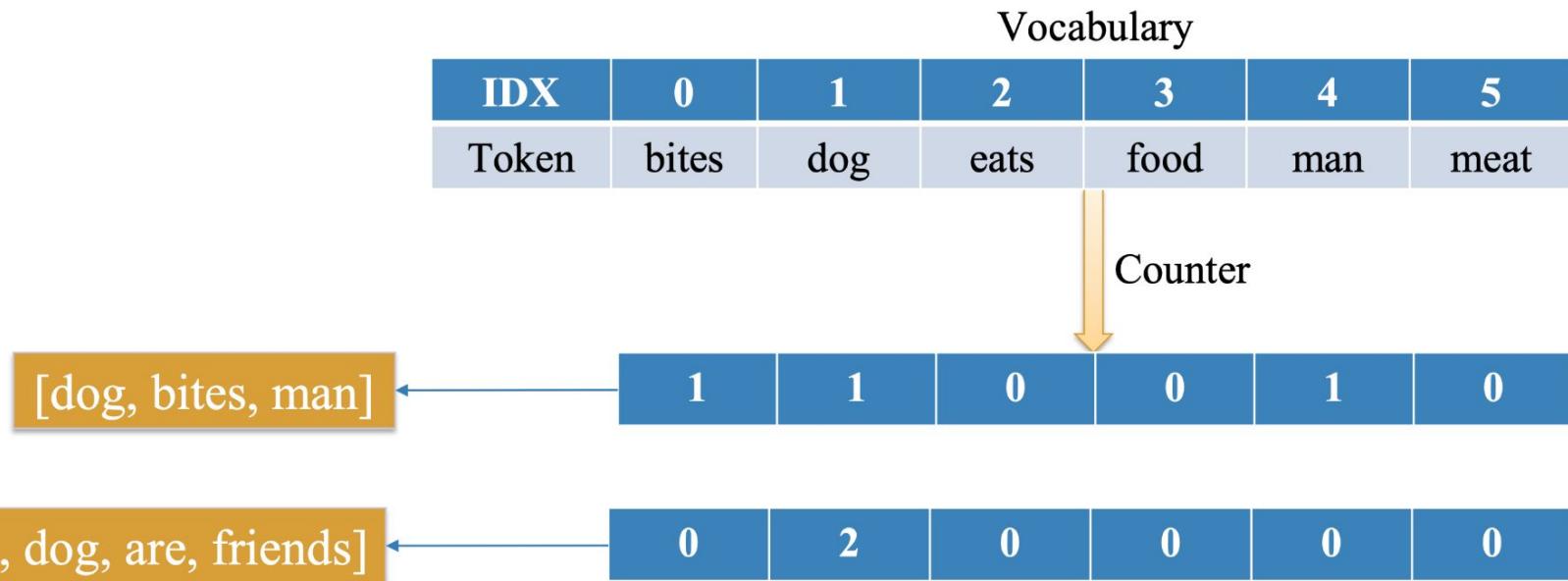
- số lần xuất hiện của từ trong tài liệu



Bag of Words (BoW)

Nhược điểm

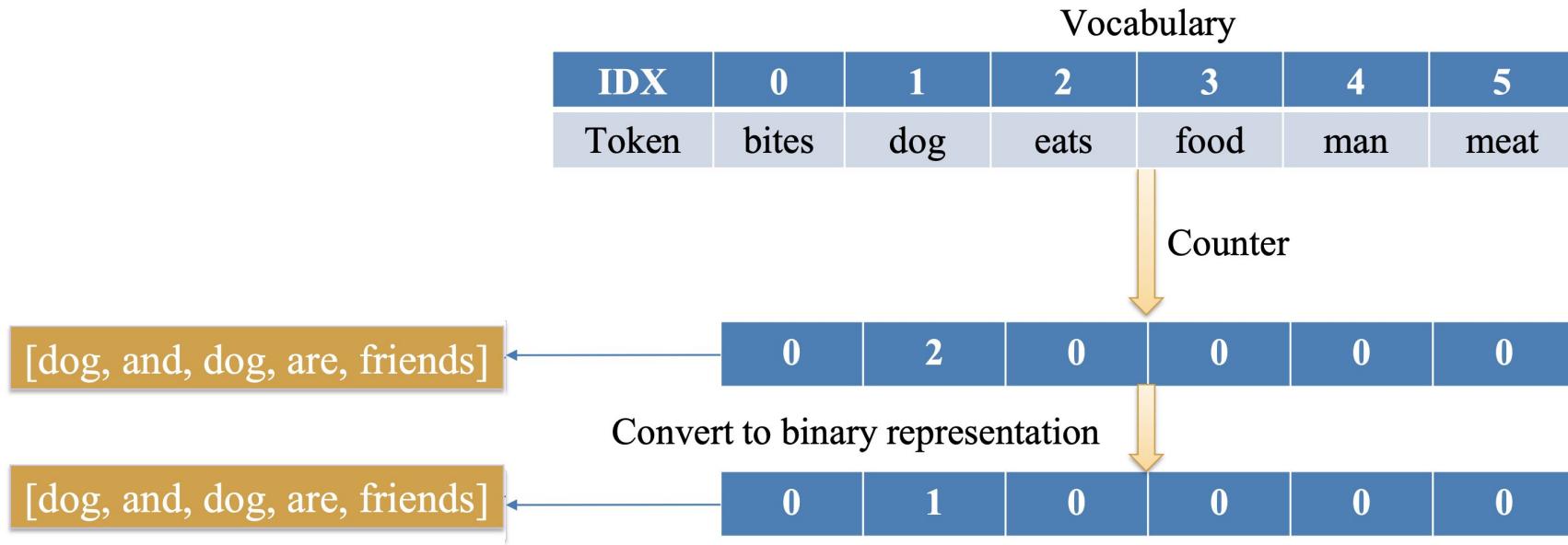
- Out of vocabulary (OOV): từ không nằm trong từ điển



Bag of Words (BoW)

Nhược điểm

Vector biểu diễn không tính đến tần suất xuất hiện của từ



Bag of Words (BoW)

Ưu điểm

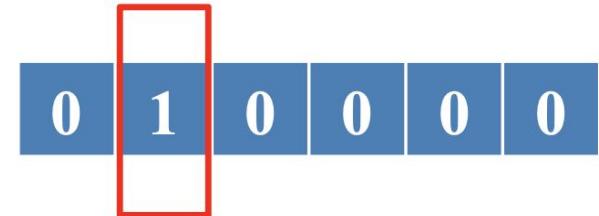
- Giữ được sự tương đồng về ngữ nghĩa của tài liệu



Bag of Words (BoW)

Nhược điểm

- Ma trận thưa (hầu hết là số 0)
 - Số chiều của Vector tỷ lệ thuận với kích thước của số lượng từ vựng
- Không nắm bắt được ý nghĩa của từ
 - Chỉ bao gồm các số 0 và 1
- Out of vocabulary (OOV)



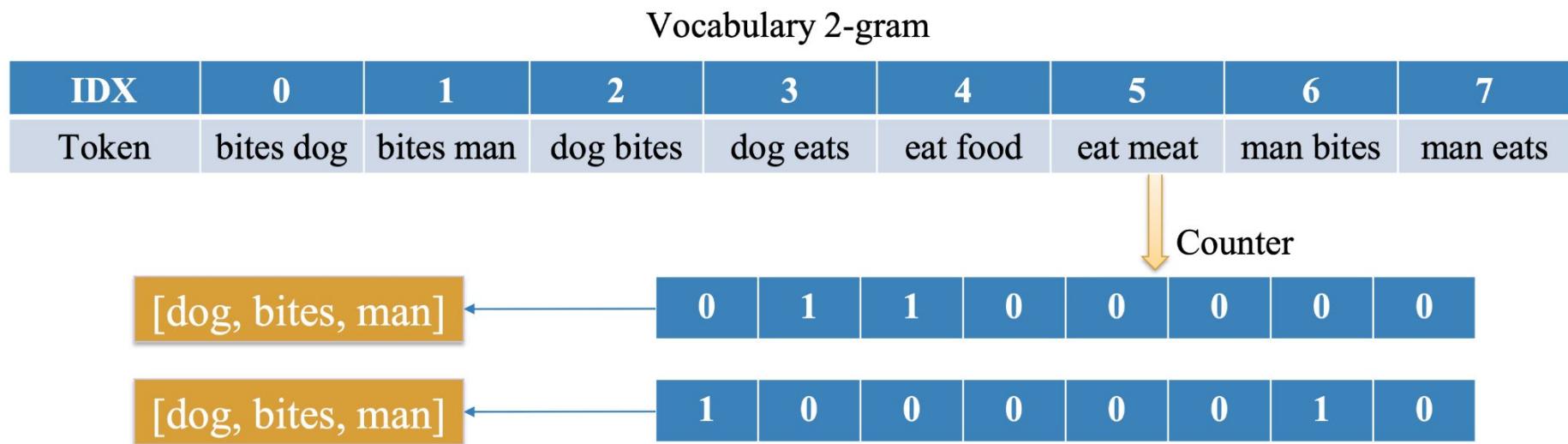
[dog, and, dog, are, friends]

How to represent

Vocabulary	
IDX	Token
0	bites
1	dog
2	eats
3	food
4	man
5	meat

Bag of N-grams

- Document Level: Xem văn bản như một túi (bộ sưu tập) các từ
- Vocabulary: tập hợp tất cả các n-gram duy nhất từ kho ngữ liệu



- **TF (Term Frequency)**: dùng để ước lượng tần suất xuất hiện của từ trong văn bản. Tuy nhiên với mỗi văn bản thì có độ dài khác nhau, vì thế số lần xuất hiện của từ có thể nhiều hơn . Vì vậy số lần xuất hiện của từ sẽ được chia độ dài của văn bản (tổng số từ trong văn bản đó)

$$tf_{t,d} = \frac{count(t, d)}{len(d)}$$

- **TF (Term Frequency)**

$$tf_{t,d} = \frac{count(t, d)}{len(d)}$$

Example

[dog, bites, man]
[man, bites, dog]
[dog, eats, meat]
[man, eats, food]

	bites	dog	eats	food	man	meat
D1						
D2						
D3						
D4						

- **TF (Term Frequency)**

$$tf_{t,d} = \frac{count(t, d)}{len(d)}$$

Example

[dog, bites, man]

[man, bites, dog]

[dog, eats, meat]

[man, eats, food]

	bites	dog	eats	food	man	meat
D1	1/3	1/3	0	0	1/3	0
D2	1/3	1/3	0	0	1/3	0
D3	0	1/3	1/3	0	0	1/3
D4	0	0	1/3	1/3	1/3	0

IDF (Inverse Document Frequency)

- Đo lường tầm quan trọng của từ trên một kho ngữ liệu

$$idf_t = \frac{N}{df_t}$$

N: Tổng số tài liệu trong kho văn bản

df_t : Số lượng tài liệu có thuật ngữ t trong đó

IDF (Inverse Document Frequency)

Khi tính tần số xuất hiện tf thì các từ đều được coi là quan trọng như nhau. Tuy nhiên có một số từ thường được sử dụng nhiều nhưng không quan trọng để thể hiện ý nghĩa của đoạn văn , ví dụ :

- Từ nối: và, nhưng, tuy nhiên, vì thế, vì vậy, ...
- Giới từ: ở, trong, trên, ...
- Từ chỉ định: ấy, đó, nhỉ, ...

Vì vậy ta cần giảm đi mức độ quan trọng của những từ đó bằng cách sử dụng IDF

$$idf_t = \ln \frac{N + 1}{df_t + 1} + 1$$

IDF (Inverse Document Frequency)

$$idf_t = \ln \frac{N + 1}{df_t + 1} + 1$$

Example

[dog, bites, man]

[man, bites, dog]

[dog, eats, meat]

[man, eats, food]

bites	dog	eats	food	man	meat
1.511	1.223	1.511	1.916	1.223	1.916

TF-IDF

- là trọng số của một từ trong văn bản thu được qua thống kê thể hiện mức độ quan trọng của từ này trong một văn bản, mà bản thân văn bản đang xét nằm trong một tập hợp các văn bản.

$$w_{t,d} = tf_{t,d} \times idf_t$$

- Khi đó, biểu diễn vectơ TF-IDF cho một tài liệu chỉ đơn giản là điểm TF-IDF cho từng thuật ngữ trong tài liệu đó.

TF-IDF

Example

[dog, bites, man]

[man, bites, dog]

[dog, eats, meat]

[man, eats, food]

bites	1.511
dog	1.223
Eats	1.511
Food	1.916
Man	1.223
meat	1.916

	bites	dog	eats	food	man	meat
D1	1/3	1/3	0	0	1/3	0
D2	1/3	1/3	0	0	1/3	0
D3	0	1/3	1/3	0	0	1/3
D4	0	0	1/3	1/3	1/3	0

	bites	dog	eats	food	man	meat
D1						
D2						
D3						
D4						

TF-IDF

Example

[dog, bites, man]
[man, bites, dog]
[dog, eats, meat]
[man, eats, food]

bites	1.511
dog	1.223
Eats	1.511
Food	1.916
Man	1.223
meat	1.916

	bites	dog	eats	food	man	meat
D1	1/3	1/3	0	0	1/3	0
D2	1/3	1/3	0	0	1/3	0
D3	0	1/3	1/3	0	0	1/3
D4	0	0	1/3	1/3	1/3	0

	bites	dog	eats	food	man	meat
D1	0.504	0.408	0	0	0.400	0
D2	0.504	0.408	0	0	0.408	0
D3	0	0.408	0.504	0	0	0.639
D4	0	0	0.504	0.639	0.408	0

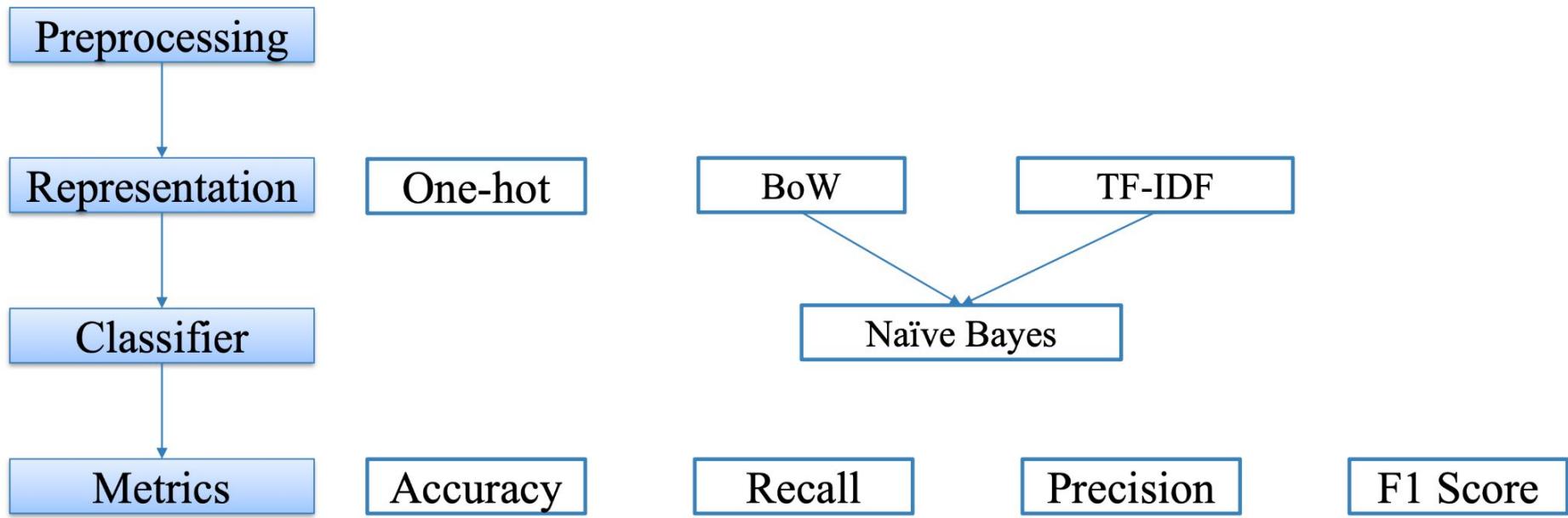
TF-IDF

Ưu điểm

- Nắm bắt một số điểm tương đồng về ngữ nghĩa
- Hữu ích cho việc truy xuất thông tin và phân loại văn bản

	bites	dog	eats	food	man	meat
[dog, bites, man]	D1	0.504	0.408	0	0	0.400
[man, bites, dog]	D2	0.504	0.408	0	0	0.408
[dog, eats, meat]	D3	0	0.408	0.504	0	0
[man, eats, food]	D4	0	0	0.504	0.639	0.408

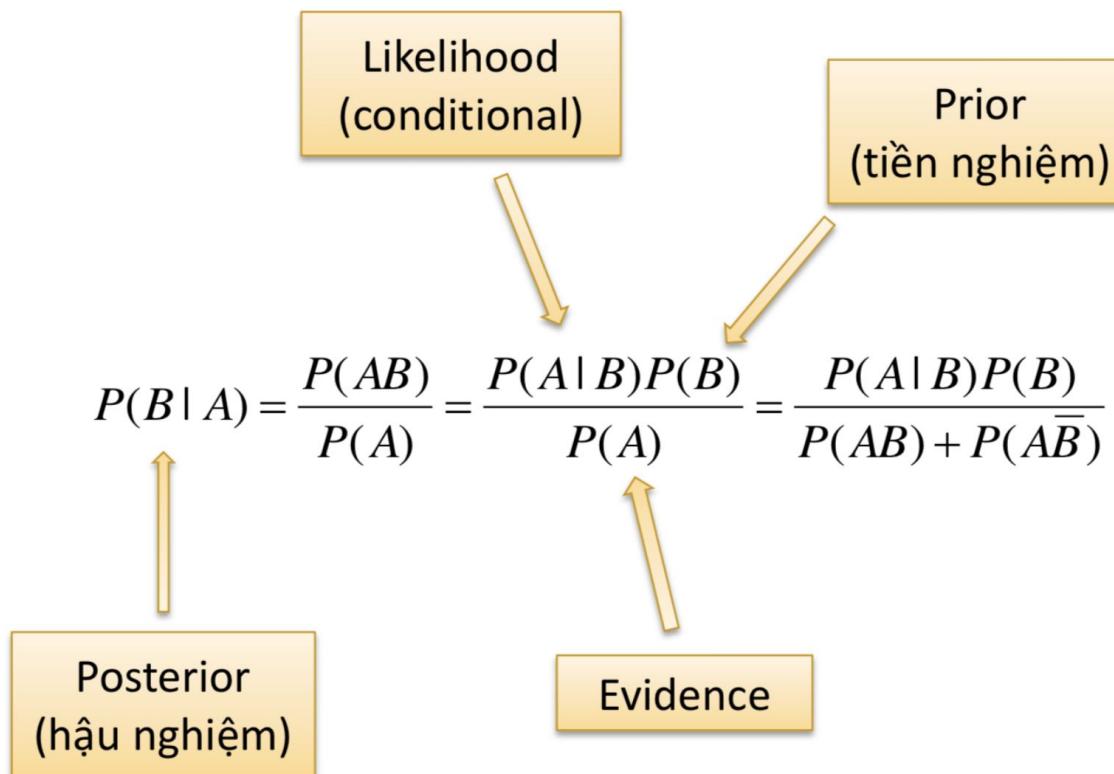
Text Classification



Phương pháp Naïve Bayes

- **Giả định Naïve Bayes:** Các thuộc tính mà mô tả dữ liệu là độc lập điều kiện theo bởi giả thuyết phân lớp
- Cực đại giả thuyết hậu nghiệm
- Một trong những phương pháp học máy thực tế nhất
- Các ứng dụng thành công:
 - Chẩn đoán sức khỏe
 - Phân lớp văn bản, lọc nội dung thư rác (spam mail)

Phương pháp Naive Bayes



Posterior = Likelihood * Prior / Evidence

Phân loại văn bản

- **Mục tiêu:** gán một văn bản mới vào một lớp có xác suất hậu nghiệm cao nhất $P(\text{class} | \text{document})$
- Ví dụ: Phân loại Spam mail
 - Một email là spam nếu $P(\text{spam} | \text{message}) > P(\neg\text{spam} | \text{message})$

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virtue of its nature as being utterly confidential and top secret. ...



TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

99 MILLION EMAIL ADDRESSES FOR ONLY \$99



Ok, I know this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

Phân loại văn bản

- **Mục tiêu:** gán một văn bản mới vào một lớp có xác suất hậu nghiệm cao nhất $P(\text{class} | \text{document})$
- Chúng ta có $P(\text{class} | \text{document}) \propto P(\text{document} | \text{class})P(\text{class})$
- Mô hình cần ước lượng **likelihoods** $P(\text{document} | \text{class})$ cho tất cả các lớp **priors** $P(\text{class})$

Thuật toán Naïve Bayes

- Mục tiêu: ước lượng xác suất hậu nghiệm $P(\text{document} | \text{class})$ và xác suất tiền nghiệm $P(\text{class})$
- **Likelihood**: giả sử dùng ***bag of words***
 - Một Văn Bản Là Một Chuỗi Các Từ (w_1, \dots, w_n)
 - Thứ tự các từ không quan trọng
 - Mỗi từ phụ thuộc điều kiện với lớp của văn bản như sau

$$P(\text{document} | \text{class}) = P(w_1, \dots, w_n | \text{class}) = \prod_{i=1}^n P(w_i | \text{class})$$

- Như vậy, vấn đề được đơn giản hóa bằng quá trình ước lượng xác suất của từng từ $P(w_i | \text{class})$

Ước lượng tham số

- Tham số của mô hình: là các xác suất likelihoods $P(\text{word} | \text{class})$ và xác suất tiền nghiệm $P(\text{class})$
 - Làm thế nào để xác định giá trị của các tham số?
 - Cần tập dữ liệu huấn luyện

$$P(\text{word} | \text{class}) = \frac{\text{\# of occurrences of this word in docs from this class}}{\text{total \# of words in docs from this class}}$$

- Quá trình ước lượng maximum likelihood (ML) từ tập dữ liệu huấn luyện như sau:

$$\prod_{d=1}^D \prod_{i=1}^{n_d} P(w_{d,i} | \text{class}_{d,i})$$

d: index of training document, i: index of a word

Ước lượng tham số

- Ước lượng tham số:

$$P(\text{word} \mid \text{class}) = \frac{\# \text{ of occurrences of this word in docs from this class}}{\text{total } \# \text{ of words in docs from this class}}$$

- Làm mịn tham số (parameter smoothing): xử lý trường hợp các từ không xuất hiện hoặc xuất hiện quá nhiều

$$P(\text{word} \mid \text{class}) = \frac{\# \text{ of occurrences of this word in docs from this class} + 1}{\text{total } \# \text{ of words in docs from this class} + V}$$

(V: total number of unique words)

Tổng kết buổi học



THANK YOU !

COLE.VN
Connecting knowledge



www.cole.vn