CSE: **Faculty of Computer Science and Engineering**
Thuyloi University

BIỂU DIỄN TỪ
(WORD REPRESENTATION)

TS. Nguyễn Thị Kim Ngân
Email: ngannguyen@tlu.edu.vn

Có tham khảo bài giảng của PGS.TS Nguyễn Thanh Hương, trường đại học Bách khoa Hà Nội

Biểu diễn từ

- Tách từ (Tokenizers)
  - Tách từ tiếng Anh
  - Tách từ tiếng Việt
- Biểu diễn từ (Word Embedding)

Biểu diễn từ

- Tách từ
  - **Tách từ tiếng Anh**
  - Tách từ tiếng Việt
- Biểu diễn từ

Ví dụ

- Cats -> CAT + N(oun) + PL(ural)
- disadvantages = dis + advantage + s

Hình thái từ

- **Hình thái từ** nghiên cứu cách từ được tạo ra từ các đơn vị nhỏ hơn gọi là hình vị (morpheme)
  - disadvantages = dis + advantage + s

- 2 lớp:
  - Hình thái học biến thể (inflectional morphology)
  - Hình thái học dẫn xuất (derivational morphology)

Hình thái học biến thể

- Gốc từ + hình vị ngữ pháp → từ:
  - Cùng lớp với từ gốc
  - Liên quan đến ngữ pháp của câu

  Example: sự phù hợp giữa chủ thể - động từ
  - He hit-s the ball
  - We hit the ball
- Số nhiều và sở hữu
  - Cats, cat’s

Hình thái học dẫn xuất

- Gốc từ + hình vị ngữ pháp → từ:
  - Khác lớp, vd, transmit->transmission (Verb thành Noun)
- Đổi nghĩa

Hậu tố	Động từ/tính từ gốc	Dẫn xuất từ
-ation	computerize(V)	computerization(N)
-ee	appoint(V)	appointee(N)
-er	love(V)	lover(N)
-ness	fuzzy(Adj)	fuzziness
-less	clue(N)	clueles

Vấn đề

- Xây dựng bộ phận phân tích hình thái từ

Đầu vào	**Đầu ra**
Cats	Cat + N + PL
Cat	Cat + N + SG
Cities	City + N + PL
Goose	Goose + N + SG
Geese	Goose + N + PL
Gooses	Goose + V + 3SG
Merging	Merge + V + PRES-PART
caught	(catch + V + PAST-PART) or (catch + V + PAST-PART)

Giải pháp: Phân tích từng hình vị từ

- mis + interpret + ation + s

  MIS + INTERPRET + noun form + plural

=> Không thực tế: không thể thấy tất cả các phần trong từ điển

- Ex: cities ≠ citie + s; cities ≠ citi + es

Định nghĩa bài toán

Các tri thức cần

- Các hậu tố nào theo sau từ gốc, và theo trật tự nào
  - Cat/cats (biến thể)
  - Dog/dogged (dẫn xuất)
- Mỗi đuôi chỉ đi với vài từ
  - Do+er ok; nhưng không có be + er
- Luật thay đổi vần điều chỉnh dạng biểu diễn bên ngoài so với dạng từ vựng:
  - Get+er→ double the t → getter
  - Fox+s → insert e → foxes
  - Fly+s → insert e → flyes → Y to I → flies

Thuật ngữ cơ bản và động cơ

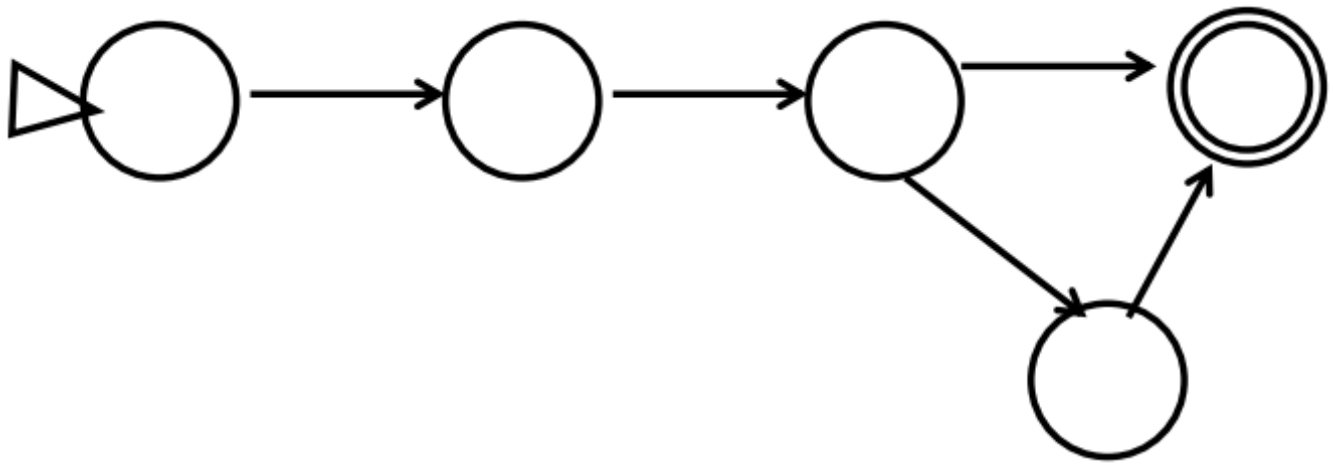
- Gốc từ (stem): đơn vị mang nghĩa gốc của từ (morpheme)
- Phụ tố (affixes): các đơn vị kết hợp với gốc từ để biến đổi ý nghĩa và chức năng ngữ pháp
  - Tiền tố (prefix): un- , anti-, ...
  - Hậu tố (suffix): -ity, -ation, ...
  - Trung tố (infix):

    Tagalog: um+hinigi → humingi (borrow)

Cách thực hiện

- Muốn mô hình hóa sự kết hợp các hình vị
- Cần nhớ một số hình vị chỉ kết nối được với một số hình vị khác
- Sử dụng automat hữu hạn trạng thái (finite-state automata – fsa)

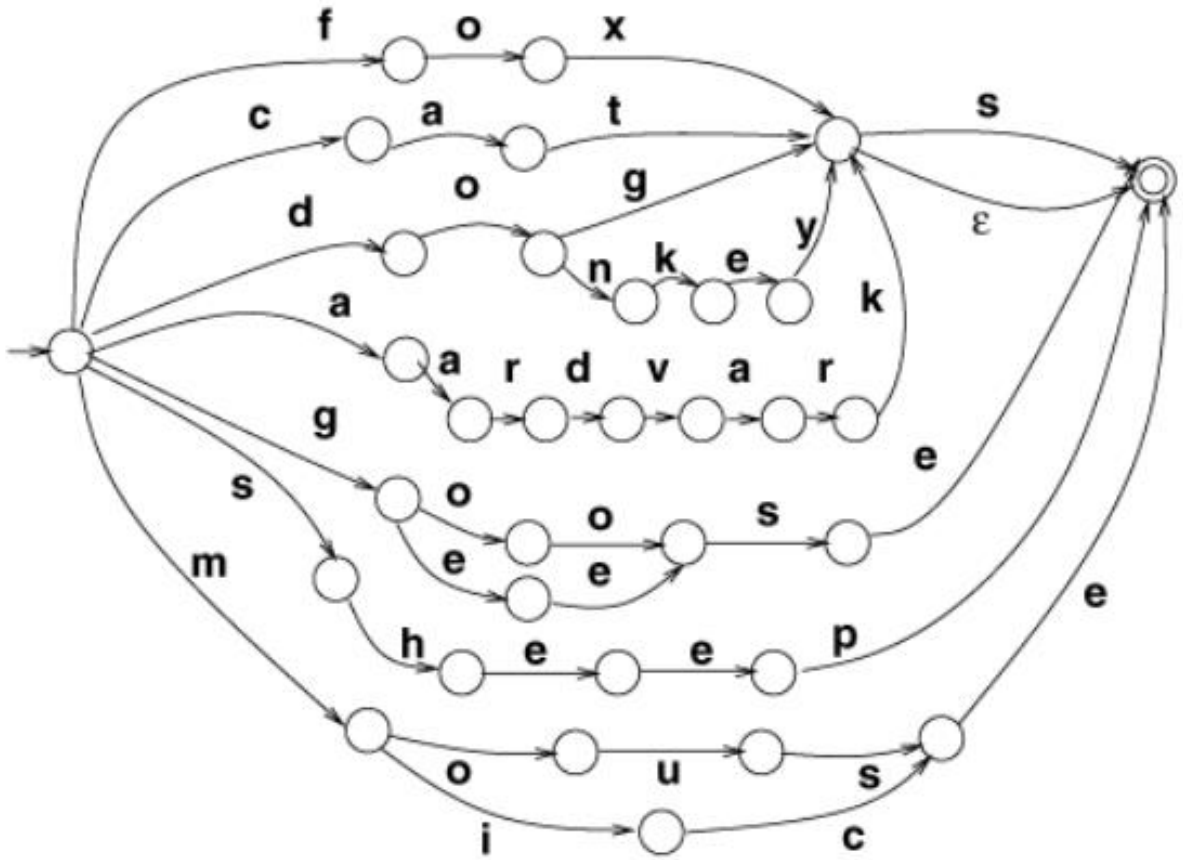
Mô hình automat hữu hạn trạng thái



Định nghĩa automat hữu hạn trạng thái

- automat hữu hạn trạng thái (đơn định) là một bộ 5 $(Q,\Sigma, \delta, q_0, F)$ với
- Q : tập hữu hạn các trạng thái
- Σ : tập hữu hạn các ký hiệu kết thúc (bảng chữ
- cái)
- $q_0 \in Q$: trạng thái đầu
- $F \subseteq Q$: tập các trạng thái kết thúc
- δ : hàm ánh xạ từ Q x Σ sang Q, là hàm chuyển

Ví dụ



Aardvarks, foxs, …

FSA vs. FST

- FSA xác định 1 ngôn ngữ hình thức (1 tập các xâu)
- FST (finite state transducer): định nghĩa tập các quan hệ giữa các xâu đó

Lấy gốc từ (Stemmer & Lemmatizer)

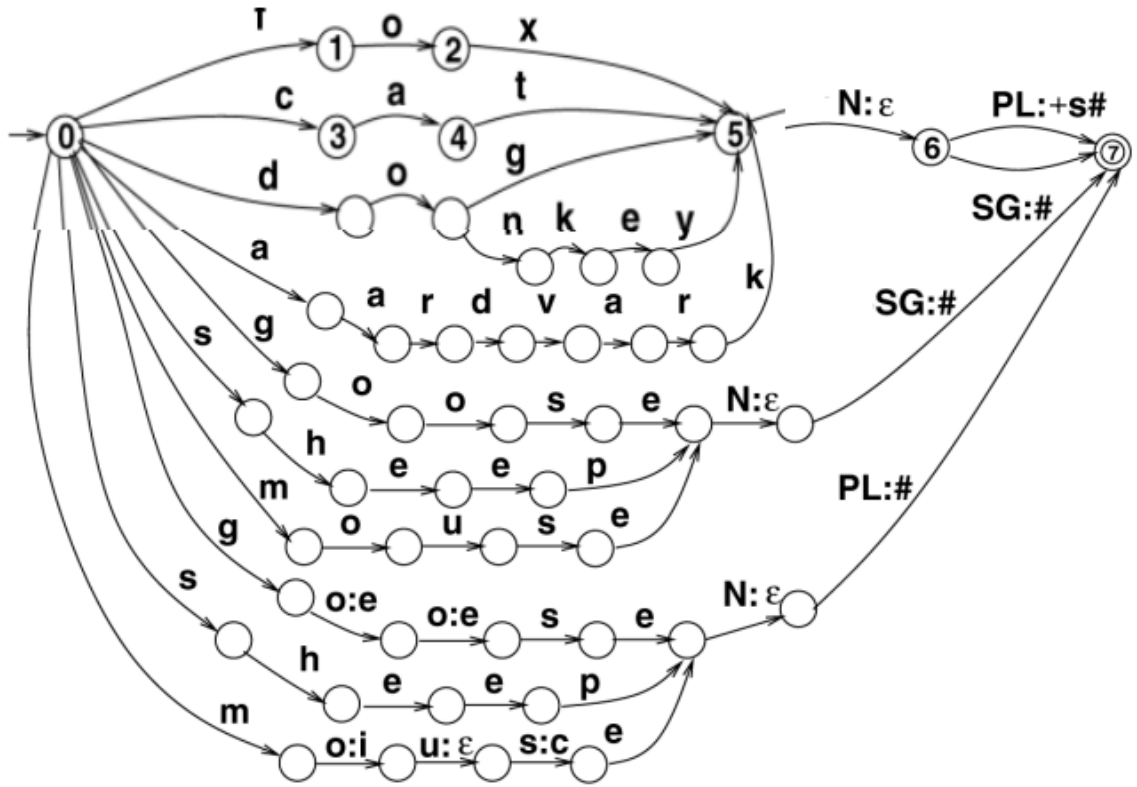
- Stemmer

Rule			Example		
SSES	→	SS	caresses	→	caress
IES	→	I	ponies	→	poni
SS	→	SS	caress	→	caress
S	→		cats	→	cat

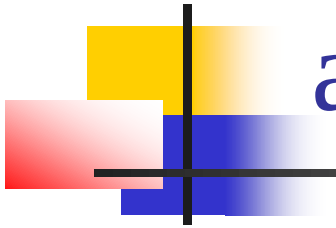
- Các luật sử dụng phép đo 1 từ, kiểm tra số lượng âm tiết có đủ dài không sau khi cắt
  - Vd: ( ) EMENT chỉ áp dụng cho replacement → replac, không cement → c.
- Stemmer sử dụng các luật ngôn ngữ ít hơn
- Lemmatizer cần bộ từ vựng đầy đủ và phân tích hình thái từ để lấy gốc từ

Kết hợp chuyển đổi các hậu tố và gốc từ

- T_{num} : Bộ chuyển đổi các hậu tố số ít/nhiều
- T_{stem}: Automat cho các gốc từ
- $T_{lex}=T_{num}\ o\ T_{stems}$



Two-level morphology parsing (analysis) algorithm



1. Initialize set of paths to P = {}
2. Read input symbols, one at a time.
3. At each symbol, generate all lexical symbols possibly corresponding to the 0 (empty) symbol
4. Prolong all paths in P by all such possible (x:0) pairs
5. Check each new path extension against the phonological FST and lexical FSA (lexical symbols only); delete impossible paths prefixes
6. Repeat 4-5 until max. # of consecutive 0s reached

Two-level morphology parsing (analysis) algorithm

7. Generate all possible lexical symbols (get from all FSTs) for the current input symbol, form pairs
8. Extend all paths from P using all such pairs
9. Check all paths from P (next step in FST/FSA)

   Delete all outright impossible paths
10. Repeat from 3 until end of input
11. Collect lexical “glosses” from all surviving paths

Generation algorithm

- Do not use the lexicon (well you have to put the “right” lexical strings together somehow!)
- Start with a lexical string L.
- Generate all possible pairs l:s for every symbol in L
- Find all (hopefully only 1!) traversals through the FST which end in a final state
- From all such traversals, print out the
- sequence of surface letters

Biểu diễn từ

- Tách từ (Tokenizers)
  - Tách từ tiếng Anh
  - Tách từ tiếng Việt
- Biểu diễn từ (Word Embedding)

Từ vựng

- Tiếng Việt là ngôn ngữ không biến hình
- Từ điển từ tiếng Việt (Vietlex): >40.000 từ,

Trong đó:

- 81.55% âm tiết là từ : từ đơn
- 15.69% các từ trong từ điển là từ đơn
- 70.72% từ ghép có 2 âm tiết
- 13.59% từ ghép ≥ 3 âm tiết
- 1.04% từ ghép ≥ 4 âm tiết

Độ dài	# từ	%
1	6,303	15.69
2	28,416	70.72
3	2,259	5.62
4	2,784	6.93
5	419	1.04
Tổng	40,181	100

Qui tắc cấu tạo từ tiếng Việt

- Từ đơn: dùng một âm tiết làm một từ.
  - tôi, bác, người, cây, hoa, đi, chạy, vì, đã, à, nhỉ, nhé...
- Từ ghép: tổ hợp (ghép) các âm tiết lại, giữa các âm tiết đó có quan hệ về nghĩa với nhau.
  - Từ ghép đẳng lập: Các thành tố cấu tạo có quan hệ bình đẳng với nhau về nghĩa
    - chợ búa, bếp núc
  - Từ ghép chính phụ: Các thành tố cấu tạo này phụ thuộc vào thành tố cấu tạo kia. Thành tố phụ có vai trò phân loại, chuyên biệt hoá và sắc thái hoá cho thành tố chính.
    - tàu hoả, đường sắt, xấu bụng, tốt mã, ngay đơ, thẳng tắp, sưng vù...

Qui tắc cấu tạo từ tiếng Việt

- Từ láy: các yếu tố cấu tạo có thành phần ngữ âm được lặp lại; nhưng vừa lặp vừa biến đổi. Một từ được lặp lại cũng cho ta từ láy
- Biến thể của từ: được coi là dạng lâm thời biến động hoặc dạng "lời nói" của từ
  - Rút gọn một từ dài thành từ ngắn hơn
    - ki-lô-gam → ki lô/ kí lô
  - Lâm thời phá vỡ cấu trúc của từ, phân bố lại yếu tố tạo từ với những yếu tố khác ngoài từ chen vào. Ví dụ:
    - khổ sở → lo khổ lo sở
    - ngặt nghẽo → cười ngặt cười nghẽo
    - danh lợi + ham chuộng → ham danh chuộng lợi

Qui tắc cấu tạo từ tiếng Việt

- Các diễn tả gồm nhiều từ (vd, “bởi vì”) cũng được coi là 1 từ
- Tên riêng: tên người và vị trí được coi là 1 đơn vị từ vựng
- Các mẫu thường xuyên: số, thời gian

Các hướng tiếp cận

- Tiếp cận dựa trên từ điển
- Tiếp cận dựa trên học máy
- Kết hợp hai phương pháp trên

Tách từ dựa trên từ điển

- Thuật toán so khớp từ dài nhất
- Yêu cầu:
  - Từ điển
  - Chuỗi đầu vào đã tách các dấu câu và âm tiết
- Tư tưởng: thuật toán tham lam
  - Đi từ trái sang phải hoặc từ phải sang trái, lấy các từ dài nhất có thể, dừng lại khi duyệt hết
  - Độ phức tạp tính toán: O(n . V)
    - n: Số âm tiết trong chuỗi
    - V: Số từ trong từ điể

Thuật toán so khớp từ dài nhất

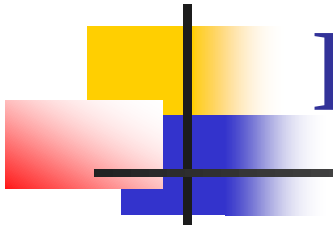
- **BẮT ĐẦU** — *khởi tạo*
- (1) Cho chuỗi đầu vào [*w_0 w_1 … w_n-1*]
- (2) *words* ← *[]*
- (3) *s* ← *0*

- (4) *e* ← *n* — *lặp*
- (5) Khi [*w_s … w_e*] **chưa là một từ**: *e* ← *e* – *1*
- (6) *words* ← *words* + *[w_s … w_e]*
- (7) *s* ← *e* + *1*
- (8) Nếu *e* < *n*: Quay lại bước (4)

- (9) Lấy ra chuỗi đã tách từ *words* — *kết thúc*
- **KẾT THÚC**

Thuật toán so khớp từ dài nhất

- Ưu điểm:
  - Cài đặt đơn giản
  - Độ phức tạp tính toán hợp lý
  - Không yêu cầu dữ liệu huấn luyện
- Nhược điểm:
  - Phụ thuộc vào từ điển
  - Chưa giải quyết được vấn đề nhập nhằng



Bài tập

- Cài đặt thuật toán so khớp từ dài nhất trên Python
- Từ điển:
- Một số mẫu thử:
  - Thời khóa biểu đang được cập nhật
  - Môn học xử lý ngôn ngữ tự nhiên
  - Ông già đi nhanh quá
  - Con ngựa đá con ngựa đá
  - Học sinh học sinh họ

Cách tách từ đơn giản

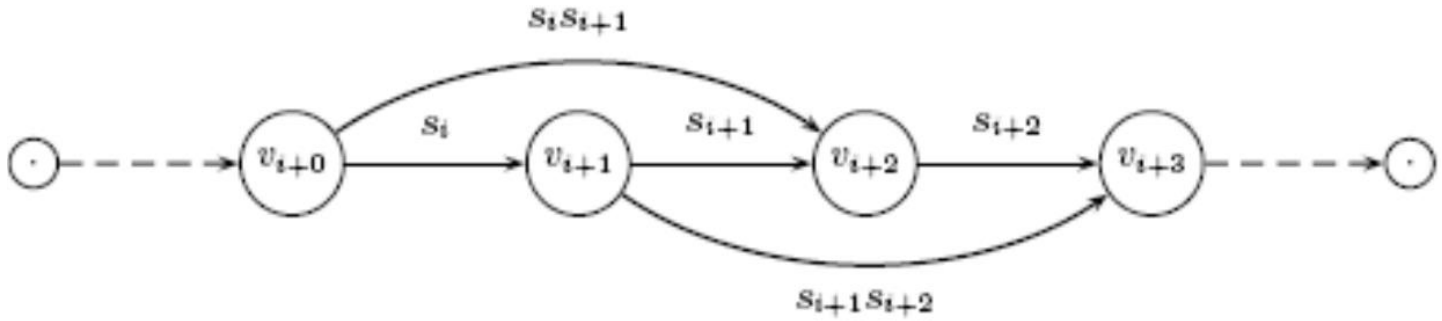
- Phát hiện các mẫu thông thường như tên riêng, chữ viết tắt, số, ngày tháng, địa chỉ email, URL,… sử dụng biểu thức chính qui
- Chọn chuỗi âm tiết dài nhất từ vị trí hiện tại và có trong từ điển, chọn cách tách có ít từ nhất
- Hạn chế: có thể đưa ra cách phân tích không đúng
- Giải quyết: liệt kê tất, có 1 chiến lược để chọn cách tách tốt nhất

Tách từ sử dụng biểu thức chính quy

- Là một khuôn mẫu được so sánh với một chuỗi
- Các ký tự đặc biệt:
  - * - bất cứ chuỗi ký tự nào, kể cả không có gì
  - x – ít nhất 1 ký tự
  - + - chuỗi trong ngoặc xuất hiện ít nhất 1 lần
- Ví dụ:
  - Email: x@x(.x)+
  - dir *.txt
  - ‘*John’ -> ‘John’, ‘Ajohn’, ‘Decker John’
- Biểu thức chính quy được sử dụng đặc biệt nhiều trong:
  - Phân tích cú pháp
  - Xác nhận tính hợp lệ của dữ liệu
  - Xử lý chuỗi
  - Trích rút thông tin

Lựa chọn cách tách từ

- Biểu diễn đoạn bằng chuỗi các âm tiết $s_1\ s_2\ \ldots s_n$
- Trường hợp nhập nhằng thường xuyên nhất là 3 từ liền nhau $s_1s_2s_3$ trong đó s_1s_2 và s_2s_3 đều là từ



- Biểu diễn 1 đoạn bằng đồ thị có hướng tuyến tính G = (V,E), V = $\{v_0 , v_1 , \ldots , v_n , v_{n+1}\}$
- Nếu các âm tiết $s_{i+1}, s_{i+2} , \ldots , s_j$ tạo thành 1 từ -> trong G có cạnh (v_i , v_j)
- Các cách tách từ = các đường đi ngắn nhất từ v_0 đến v_{n+1}

Thuật toán

Thuật toán 1. Xây dựng đồ thị cho chuỗi $s_1 s_2 \ldots s_n$

```
1: V ← ∅;
2: for i = 0 to n + 1 do
3:     V ← V ∪ {v_i};
4: end for
5: for i = 0 to n do
6:     for j = i to n do
7:         if (accept(A_W, s_i · · · s_j)) then
8:             E ← E ∪ {(v_i, v_{j+1})};
9:         end if
10:    end for
11: end for
12: return G = (V,E);
```

Accept(A,s): automat A nhận xâu vào s

Phân giải nhập nhằng

- Xác suất xâu s:

$$P(s) = \prod_{i=1}^{m} P(w_i|w_1^{i-1}) \approx \prod_{i=1}^{m} P(w_i|w_{i-n+1}^{i-1})$$

- $P(w_i | w_1^{i-1})$: xác suất w_i khi có i-1 từ trước đó
- n = 2: bigram; n = 3: trigram

Phân giải nhập nhằng

- Khi n = 2, tính giá trị $P(w_i | w_{i-1})$ lớn nhất maximum likelihood (ML)

$$P_{ML}(w_i|w_{i-1}) = \frac{P(w_{i-1}w_i)}{P(w_{i-1})} = \frac{c(w_{i-1}w_i)/N}{c(w_{i-1})/N} = \frac{c(w_{i-1}w_i)}{c(w_{i-1})}$$

- c(s): số lần xâu s xuất hiện; N: tổng số từ trong tập luyện
- Khi dữ liệu luyện nhỏ hơn kích cỡ toàn bộ tập dữ liệu, thì P ~ 0
  - Sử dụng kỹ thuật làm trơn

Kỹ thuật làm trơn

$$\widehat{P}(w_i|w_{i-1}) = \lambda_1 P_{ML}(w_i|w_{i-1}) + \lambda_2 P_{ML}(w_i)$$

với $\lambda_1 + \lambda_2 = 1$ và $\lambda_1, \lambda_2 \geq 0$

$P_{ML}(w_i) = c(w_i)/N$

Với tập thử nghiệm T = {s1 ,s2 ,….,sn }, xác suất P(T) của tập thử:

$$P(T) = \prod_{i=1}^{n} P(s_i)$$

Xác định giá trị λ_1 , λ_2

Từ tập dữ liệu mẫu, định nghĩa $C(w_{i-1}, w_i)$ là số lần (w_{i-1}, w_i) xuất hiện trong tập mẫu. Ta cần chọn λ_1 , λ_2 để làm cực đại giá trị

$$L(\lambda_1, \lambda_2) = \sum_{w_{i-1}, w_i} C(w_{i-1}, w_i) \log_2 \hat{P}(w_i | w_{i-1})$$

với $\lambda_1 + \lambda_2 = 1$ và $\lambda_1 , \lambda_2 \geq 0$

Thuật toán

$\lambda_1 \leftarrow 0.5, \lambda_2 \leftarrow 0.5;$

$\epsilon \leftarrow 0.01;$

repeat

$\widehat{\lambda}_1 \leftarrow \lambda_1, \quad \widehat{\lambda}_2 \leftarrow \lambda_2;$

$c_1 \leftarrow \sum_{w_{i-1},w_i} \frac{C(w_{i-1},w_i)\lambda_1 P_{ML}(w_i|w_{i-1})}{\lambda_1 P_{ML}(w_i|w_{i-1})+\lambda_2 P_{ML}(w_i)};$

$c_2 \leftarrow \sum_{w_{i-1},w_i} \frac{C(w_{i-1},w_i)\lambda_2 P_{ML}(w_i)}{\lambda_1 P_{ML}(w_i|w_{i-1})+\lambda_2 P_{ML}(w_i)};$

$\lambda_1 \leftarrow \frac{c_1}{c_1+c_2}, \quad \lambda_2 \leftarrow 1-\widehat{\lambda}_1;$

$\widehat{\epsilon} \leftarrow \sqrt{(\widehat{\lambda}_1-\lambda_1)^2+(\widehat{\lambda}_2-\lambda_2)^2};$

until $(\widehat{\epsilon} \leq \epsilon);$

return $\lambda_1, \lambda_2;$

Cách tiếp cận lai

<Phuong Le-Hong et al., A hybrid approach to word segmentation of Vietnamese texts, Proceedings of the 2nd International Conference on Language and Automat Theory and Applications, LATA 2008, Tarragona, Spain, 2008.>

- Kết hợp phân tích automat hữu hạn + biểu thức chính quy + so khớp từ dài nhất + thống kê (để giải quyết nhập nhằng)

Kết quả

- Sử dụng tập dữ liệu gồm 1264 bài trong báo Tuổi trẻ, có 507,358 từ
- Lấy ε = 0.03, các giá trị λ hội tụ sau 4 vòng lặp

Step	λ_1	λ_2	ϵ
0	0.500	0.500	1.000
1	0.853	0.147	0.499
2	0.952	0.048	0.139
3	0.981	0.019	0.041
4	0.991	0.009	0.015

- Độ chính xác = số từ hệ thống xác định đúng/tổng số từ hệ thống xác định = 95%

Một số công cụ tách từ

- JvnSegmenter (Nguyễn Cẩm Tú): CRF

  https://jvnsegmenter.sourceforge.net/
- Pyvi (Trần Việt Trung)

  https://github.com/trungtv/pyvi

Cài đặt và chạy chương trình tách từ Pyvi

```
1  from pyvi import ViTokenizer
2
3  ex1 = "thời khóa biểu đang được cập nhật"
4  ex2 = "môn học xử lý ngôn ngữ tự nhiên"
5  ex3 = "con ngựa đá con ngựa đá"
6  ex4 = "học sinh học sinh học"
7  ex5 = "Tách từ là bài toán nhận diện từ trong văn bản tiếng Việt"
8
9  if __name__ == "__main__":
10     print (ViTokenizer.tokenize(ex5))
```

Biểu diễn từ

- Tách từ
  - Tách từ tiếng Anh
  - Tách từ tiếng Việt
- **Biểu diễn từ**

Biểu diễn từ

- Mô hình xử lý ngôn ngữ tự nhiên có đầu vào chỉ là các chữ cái kết hợp với dấu câu
- Làm sao chúng ta có thể lượng hoá được những từ ngữ để làm đầu vào cho mạng nơ ron?

Một số khái niệm cơ bản

- Document (Văn bản): Là tập hợp các câu trong cùng một đoạn văn có mối liên hệ với nhau. Văn bản có thể được coi như một bài báo, bài văn,….
- Corpus (Bộ văn bản): Là một tợp hợp gồm nhiều văn bản thuộc các đề tài khác nhau, tạo thành một nguồn tài nguyên dạng văn bản. Một số bộ văn bản trong tiếng việt có thể được download từ nguồn Wikipedia, VNCoreNLP

Một số khái niệm cơ bản

- Character (kí tự): Là tợp hợp gồm các chữ cái (nguyên âm và phụ âm) và dấu câu. Mỗi một ngôn ngữ sẽ có một bộ các kí tự khác nhau
- Word (từ vựng): Là các kết hợp của các kí tự tạo thành những từ biểu thị một nội dung, định nghĩa xác định, chẳng hạn con người có thể coi là một từ vựng. Từ vựng có thể bao gồm từ đơn có 1 âm tiết và từ ghép nhiều hơn 1 âm tiết
- Dictionary (từ điển): Là tợp hợp các từ vựng xuất hiện trong văn bản
- Volcabulary (từ vựng): Tợp hợp các từ được trích xuất trong văn bản. Tương tự như từ điển

Biểu diễn one-hot véc tơ

- Giả sử, từ điển = {anh, em, gia đình, bạn bè,...}
- Mỗi từ sẽ được đại diện bởi một giá trị chính là index của nó. Từ **anh** có index = 0, **gia đình** có index = 2, …
- One-hot véc tơ của từ vựng thứ i, i≤(n-1) sẽ là véc tơ e_i=[0, …, 0, 1, 0, …, 0] ∈ R^n sao cho các phần tử e_{ij} của véc tơ thỏa mãn:

$$\begin{cases} e_{ij} = 0, & \text{if } i \neq j \\ e_{ii} = 1 \end{cases}$$

∀i, j ∈ N; 0≤i, j ≤n-1

Biểu diễn one-hot véc tơ

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
words = ['anh', 'em', 'gia đình', 'bạn bè', 'anh', 'em']
le.fit(words)

print('Class of words: ', le.classes_)
# Biến đổi sang dạng số
x = le.transform(words)
print('Convert to number: ', x)
# Biến đổi lại sang class
print('Invert into classes: ', le.inverse_transform(x))
```

```
Class of words:  ['anh' 'bạn bè' 'em' 'gia đình']
Convert to number:  [0 2 3 1 0 2]
Invert into classes:  ['anh' 'em' 'gia đình' 'bạn bè' 'anh' 'em']
```

Biểu diễn one-hot véc tơ

```
from sklearn.preprocessing import OneHotEncoder
import numpy as np

oh = OneHotEncoder()
classes_indices = list(zip(le.classes_,np.arange(len(le.classes_))))
print('Classes_indices: ', classes_indices)

oh.fit(classes_indices)
print('One-hot categories and indices:', oh.categories_)

# Biến đổi list words sang dạng one-hot
words_indices = list(zip(words, x))
print('Words and corresponding indices: ', words_indices)
one_hot = oh.transform(words_indices).toarray()
print('Transform words into one-hot matrices: \n', one_hot)
print('Inverse transform to categories from one-hot matrices: \n',
oh.inverse_transform(one_hot))
```

```
Classes_indices:  [('anh', 0), ('bạn bè', 1), ('em', 2), ('gia đình', 3)]
One-hot categories and indices: [array(['anh', 'bạn bè', 'em', 'gia đình'],
      dtype=object), array([0, 1, 2, 3], dtype=object)]
Words and corresponding indices:  [('anh', 0), ('em', 2), ('gia đình', 3),
      ('bạn bè', 1), ('anh', 0), ('em', 2)]
Transform words into one-hot matrices:
 [[1. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 1. 0.]
 [0. 0. 0. 1. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0. 1. 0. 0.]
 [1. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 1. 0.]]
Inverse transform to categories from one-hot matrices:
 [['anh' 0]
 ['em' 2]
 ['gia đình' 3]
 ['bạn bè' 1]
 ['anh' 0]
 ['em' 2]]
```

Biểu diễn One-hot véc tơ

- One-hot véc tơ chưa phản ảnh được mối quan hệ về mặt ngữ nghĩa của các từ
  - Mối quan hệ tương quan giữa các cặp từ bất kì luôn là không tương quan (tức bằng 0). Do đó không có tác dụng trong việc tìm mối liên hệ về nghĩa
  - Kích thước của véc tơ sẽ phụ thuộc vào số lượng từ vựng có trong bộ văn bản dẫn đến chi phí tính toán rất lớn khi tập dữ liệu lớn
  - Khi bổ sung thêm các từ vựng mới số chiều của véc tơ có thể thay đổi theo dẫn đến sự không ổn định trong shape

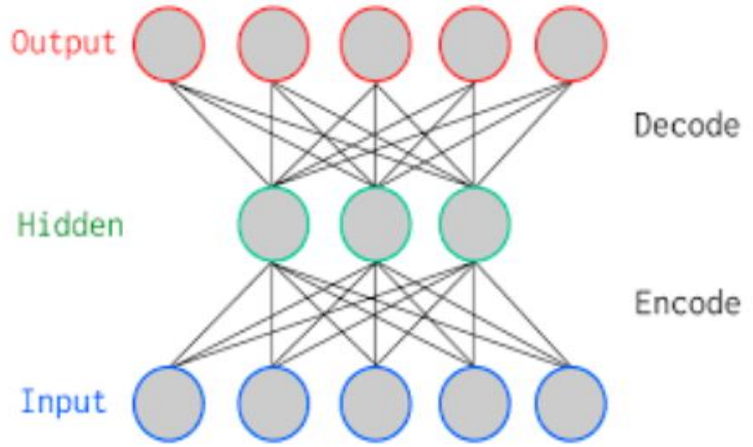
Word Embedding

Thuật toán nhúng từ được tạo ra nhằm mục đích tìm ra các véc tơ đại diện cho mỗi từ sao cho:

- Một từ được biểu diễn bởi một véc tơ có số chiều xác định trước
- Các từ thuộc cùng 1 nhóm thì có khoảng cách gần nhau trong không gian
- Có nhiều phương pháp nhúng từ khác nhau có thể kể đến. Trong đó có 3 nhóm chính:
  - Sử dụng thống kê tần xuất: tfidf
  - Các thuật toán giảm chiều dữ liệu: SVD, PCA, auto encoder, word2vec
  - Phương pháp sử dụng mạng nơ ron: word2vec, ELMo, BERT

Phương pháp auto encoder

- Auto encoder được xây dựng trên một mạng nơ ron có 3 layer: input, hidden layer và output
  - Số units ở input và output là bằng nhau
  - Số units ở hidden layer sẽ qui định số chiều của véc tơ biểu diễn từ và thông thường sẽ nhỏ hơn số units ở đầu vào



Phương pháp auto encoder

```
import scipy.linalg as ln
import numpy as np
from underthesea import word_tokenize


sentence = 'Khoa học dữ liệu là một lĩnh vực đòi hỏi kiến thức về toán và lập trình. Tôi rất yêu thích Khoa học dữ liệu.'
token = word_tokenize(sentence)
# Tokenize câu search
print('tokenization of sentences: ', token)
```

Phương pháp auto encoder

```
from scipy.sparse import coo_matrix
# Tạo ma trận coherence dưới dạng sparse thông qua khai báo vị trí khác 0 của trục x và y
row = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13]
col = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14]
data =      [2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

X = coo_matrix((data, (row, col)), shape=(15, 15)).toarray()
X
```

Phương pháp auto encoder

```
array([[0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
```

Phương pháp auto encoder

```
1   from keras.layers import Dense, Input
2   from keras.models import Model, Sequential
3   from keras.optimizers import RMSprop, Adam
4
5   def autoencoder(input_unit, hidden_unit):
6       model = Sequential()
7       model.add(Dense(input_unit, input_shape = (15,), activation = 'relu'))
8       model.add(Dense(hidden_unit, activation = 'relu'))
9       model.add(Dense(input_unit, activation = 'softmax'))
10      model.compile(loss = 'categorical_crossentropy', optimizer = Adam(),
11                    metrics = ['accuracy'])
12      model.summary()
13      return model
14
15  model_auto = autoencoder(input_unit = 15, hidden_unit = 6)
16
17  model_auto.fit(X, X, epochs = 5, batch_size = 3)
```

Phương pháp auto encoder

```
1  embedding_matrix = model_auto.layers[2].get_weights()[0]
2  bias = model_auto.layers[2].get_weights()[1]
3
4  print('Shape of embedding_matrix: ', embedding_matrix.shape)
5  print('Embedding_matrix: \n', embedding_matrix)
```

```
Shape of embedding_matrix:  (6, 15)
Embedding_matrix:
 [[ 0.38889918  0.5211232  -0.35681784 -0.29142842  0.25496536  0.47015667
   0.12295379  0.34093136  0.36910903  0.09683032 -0.41072607 -0.07050186
   0.28118226  0.14136976 -0.398313  ]
 [ 0.18342797 -0.14228119 -0.29116338  0.40031028  0.47284338  0.5166124
  -0.47880676  0.49956253  0.36308518  0.07943692  0.46039233 -0.04482159
   0.14367305  0.46219113 -0.37292722]
 [ 0.4906134  -0.00613014 -0.09216617  0.3174584   0.08535323  0.03718374
  -0.0576647   0.13673814 -0.0192671   0.16489299 -0.3544627  -0.4466407
  -0.46152878  0.35548216  0.19229826]
 [-0.04221632 -0.2623642  -0.2671243  -0.14902063 -0.08061455  0.08999895
   0.22966935 -0.54198337 -0.2509707   0.46091208 -0.06831685 -0.5284586
  -0.21089761 -0.13299096  0.36479107]
 [ 0.09093584  0.38861293  0.24202171  0.20458116 -0.25571942  0.05853903
  -0.267772   -0.12935235  0.27599117 -0.25800633  0.2633568  -0.25931272
  -0.03536293 -0.29268453 -0.4267695 ]
 [ 0.26897088  0.24455284 -0.27629155  0.4157534  -0.27802745  0.12034645
   0.47979772  0.5275412   0.00355813  0.26329502 -0.18948056  0.00509128
   0.4196368   0.4636546   0.08472057]]
```

Phương pháp auto encoder

```
from sklearn.metrics.pairwise import cosine_similarity
from numpy.linalg import norm

def cosine(x, y):
    cos_sim = np.dot(x, y)/(norm(x)*norm(y))
    return cos_sim
# Véc tơ biểu diễn từ khoa học
e0 = list(embedding_matrix[:, 0])
# Véc tơ biểu diễn từ dữ liệu
e1 = list(embedding_matrix[:, 1])
# Quan hệ tương quan ngữ nghĩa giữa từ khoa học và dữ liệu
cosine(e0, e1)
```

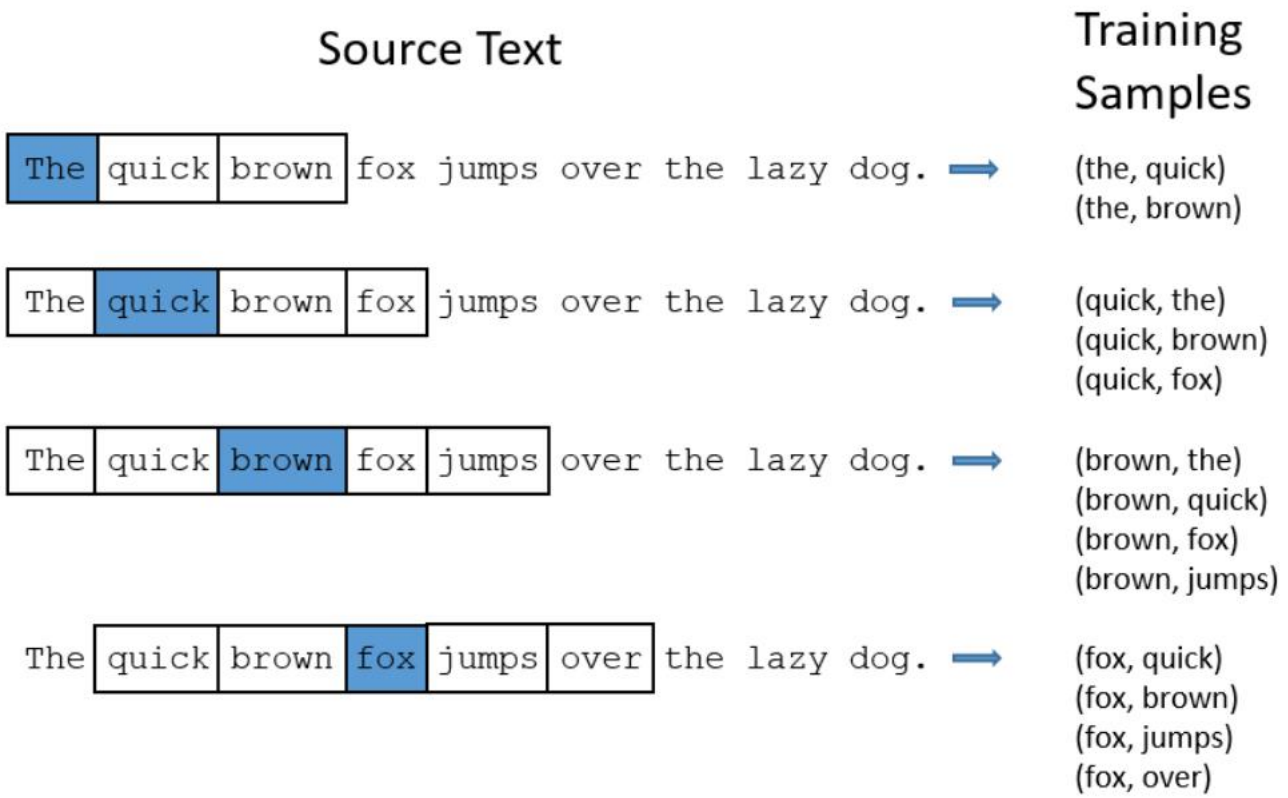
```
0.5303333
```

Mô hình word2vec

- Target word (từ đích) là 1 từ trong câu
- Context words (từ ngữ cảnh): những từ xung quanh target word được gọi là context words. Với mỗi từ đích trong một câu, các từ ngữ cảnh được định nghĩa là các từ trong cùng câu có vị trí cách từ đích một khoảng không quá C/2 (C $\in N^*$)
- Với mỗi từ đích, ta sẽ có một bộ không quá C từ ngữ cảnh
- Mô hình word2vec có 2 phương pháp chính là skip-grams và CBOW

Ví dụ

"The quick brown fox jump over the lazy dog" với C=4

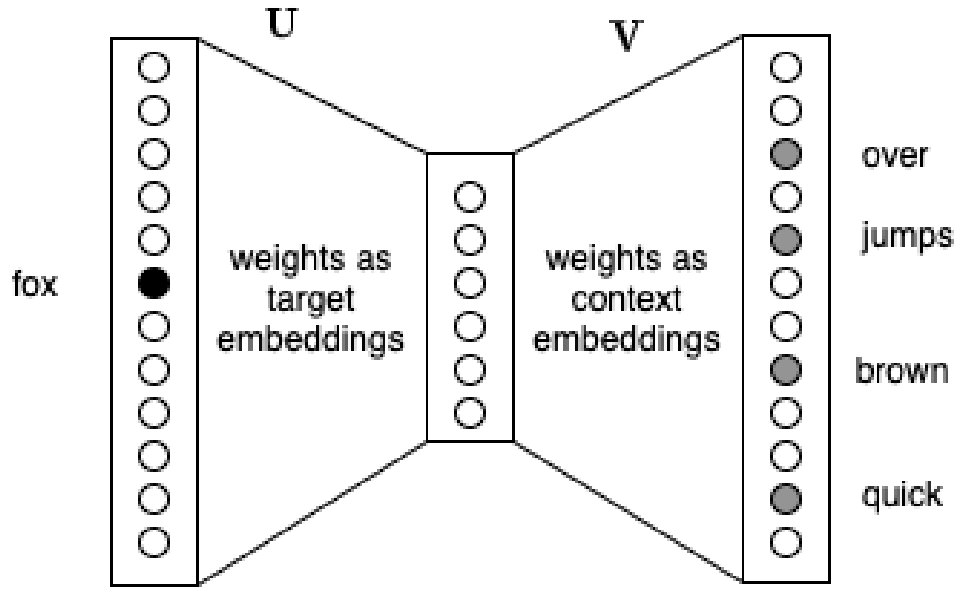


Skip-gram

- Mục tiêu: Dự đoán những từ ngữ cảnh nếu biết trước từ đích

Skip-gram

- Mô hình sẽ biểu diễn một từ bối cảnh dưới dạng one-hot véc tơ $\mathbf{o}_c$
- $\mathbf{o}_c$ là đầu vào cho một mạng nơ ron có tầng ẩn gồm 300 units
- Output layer là một hàm softmax tính xác xuất để các từ mục tiêu phân bố vào những từ trong vocabulary (10000 từ)
- Dựa trên quá trình feed forward và back propagation mô hình sẽ tìm ra tham số tối ưu để kết quả dự báo từ mục tiêu là chuẩn xác nhất
- Khi đó quay trở lại tầng hidden layer ta sẽ thu được đầu ra tại tầng này là ma trận nhúng (E)



Continous Bag of Words (CBOW)

- Mục tiêu: Dựa vào những từ ngữ cảnh để dự đoán từ đích

=> Tìm xác suất xảy ra từ đích khi biết các từ ngữ cảnh xung quanh

Continous Bag of Words (CBOW)

Kiến trúc mạng nơ ron của CBOW sẽ gồm 3 layers:

- Input layers: Là các từ bối cảnh xung quanh từ mục tiêu
- Projection layer: Lấy trung bình véc tơ biểu diễn của toàn bộ các từ input để tạo ra một véc tơ đặc trưng
- Output layer: Là một dense layers áp dụng hàm softmax để dự báo xác xuất của từ mục tiêu
- Trích xuất ma trận nhúng (E) tại đầu ra của hidden layer