ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Bài 10:
Một số ứng dụng học sâu trong xử lý ngôn ngữ tự nhiên
(Phần 1)

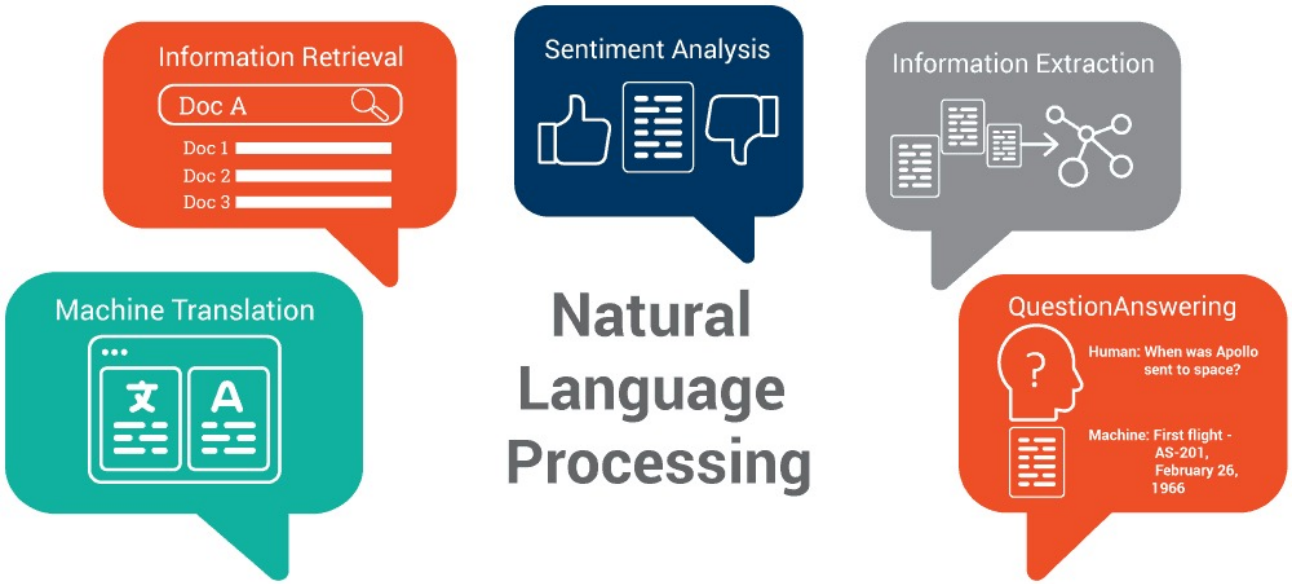
Nội dung

- Tổng quan về xử lý ngôn ngữ tự nhiên
- Biểu diễn từ và văn bản
- Thư viện Xử lý NNTN và một số mô hình huấn luyện sẵn

Tổng quan về xử lý ngôn ngữ tự nhiên

Thế nào là Xử lý NNTN?

- Xử lý NNTN là một nhánh của trí tuệ nhân tạo liên quan đến sự tương tác giữa máy tính và ngôn ngữ của con người.
- Mục đích của xử lý NNTN là giúp máy tính có khả năng đọc, hiểu và rút ra ý nghĩa từ ngôn ngữ của con người.



Các mức phân tích

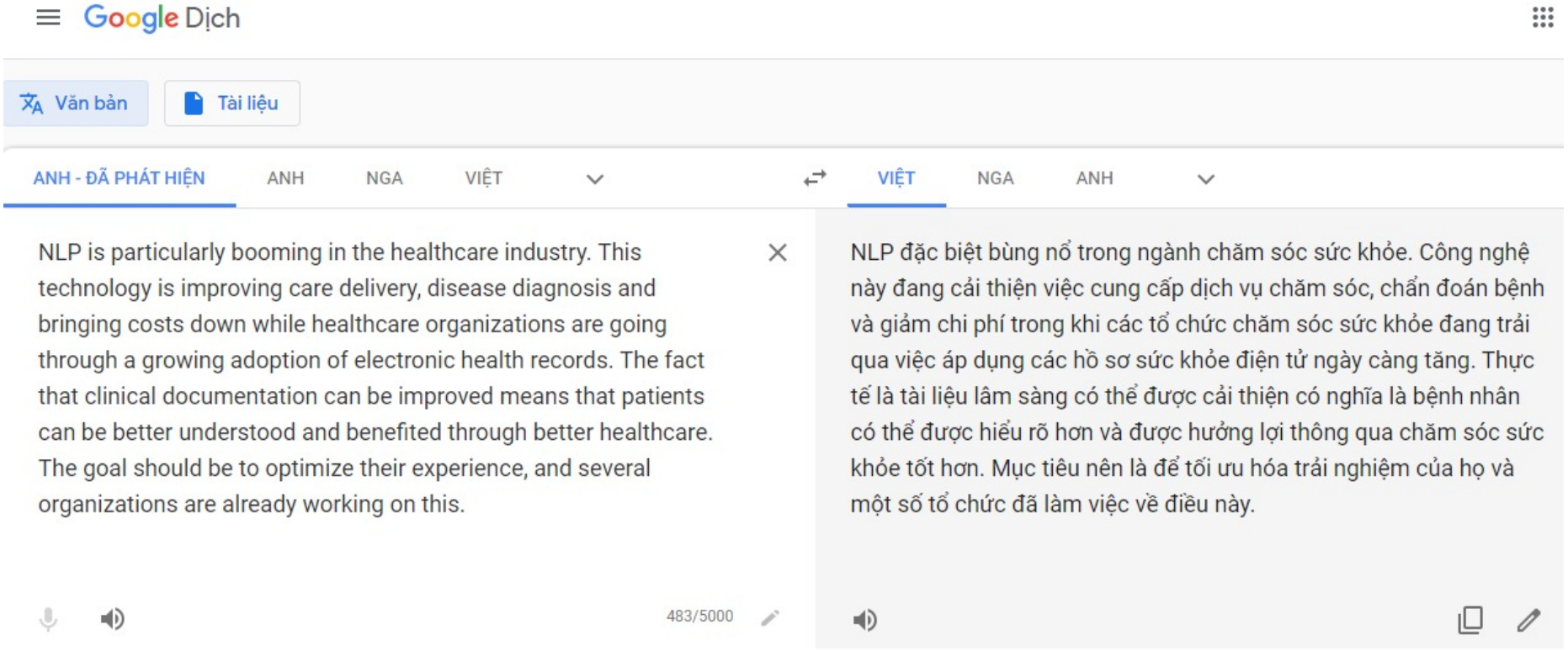
- **Morphology** (hình thái học): cách từ được xây dựng, các tiền tố và hậu tố của từ
- **Syntax** (cú pháp): mối liên hệ về cấu trúc ngữ pháp giữa các từ và ngữ
- **Semantics** (ngữ nghĩa): nghĩa của từ, cụm từ, và cách diễn đạt
- **Discourse** (diễn ngôn): quan hệ giữa các ý hoặc các câu
- **Pragmatic** (thực chứng): mục đích phát ngôn, cách sử dụng ngôn ngữ trong giao tiếp
- **World Knowledge** (tri thức thế giới): các tri thức về thế giới, các tri thức ngầm

Một số ứng dụng chính của NLP

- Nhận dạng giọng nói (speech recognition)
- Khai phá văn bản
  - Phân cụm văn bản
  - Phân lớp văn bản
  - Tóm tắt văn bản
  - Mô hình hóa chủ đề (topic modelling)
  - Hỏi đáp (question answering)
- Gia sư ngôn ngữ (Language tutoring)
  - Chỉnh sửa ngữ pháp/đánh vần
- Dịch máy (machine translation)

Dịch máy

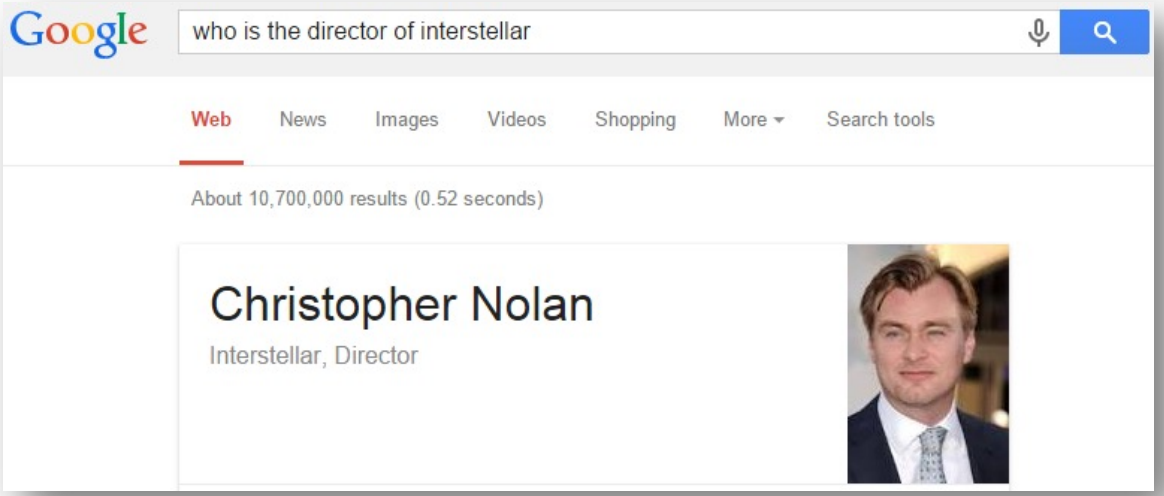
- Google translate



Các hệ thống hội thoại

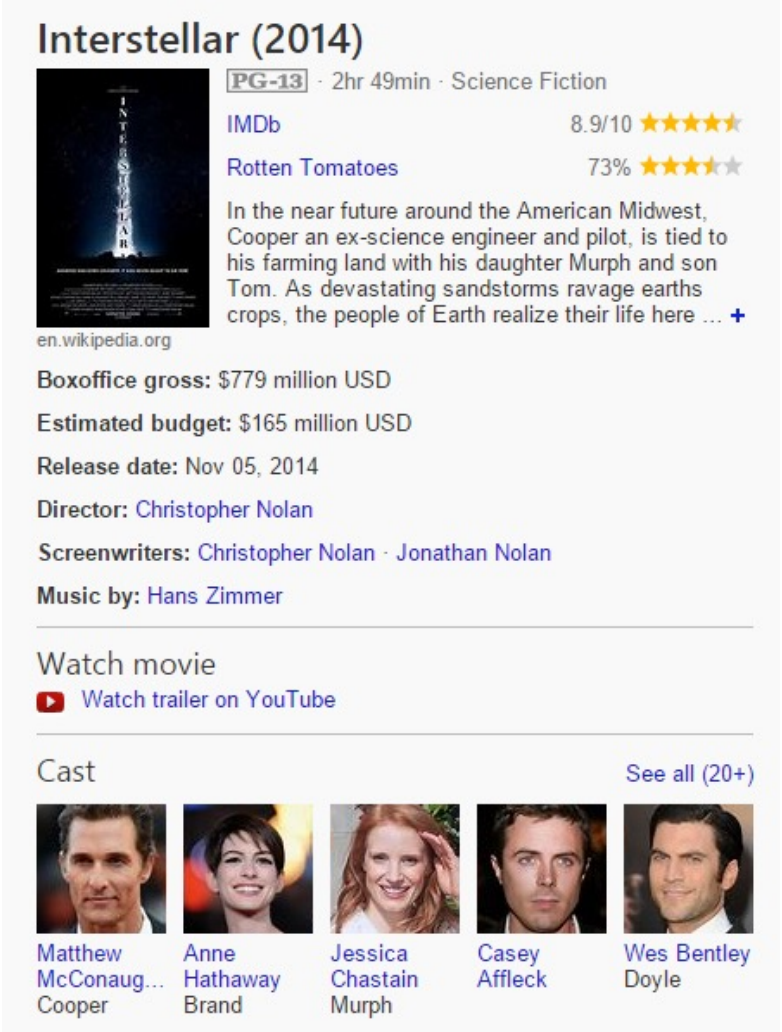
- Chatbot, trợ lý ảo, hỏi đáp tự động

Apple’s siri system



Google search

Trích rút thông tin (Information extraction)



Google Knowledge Graph

Wiki Info Box

Token hóa (Tokenization)

- Chia văn bản thành các từ và các câu

There was an earthquake near D.C. I've even felt it in Philadelphia, New York, etc.

There + was + an + earthquake + near + D.C.

I + ve + even + felt + it + in + Philadelphia, + New + York, + etc.

Part-of-Speech tagging

- Xác định từ loại của từng từ trong văn bản

A + dog + is + chasing + a + boy + on + the + playground

A	dog	is	chasing	a	boy	on	the	playground
Det	Noun	Aux	Verb	Det	Noun	Prep	Det	Noun

Nhận dạng thực thể định danh (Named entity recognition)

- Tìm kiếm và phân loại các thành phần trong văn bản vào những loại xác định trước như là tên người, tổ chức, địa điểm, thời gian, số lượng, giá trị tiền tệ…

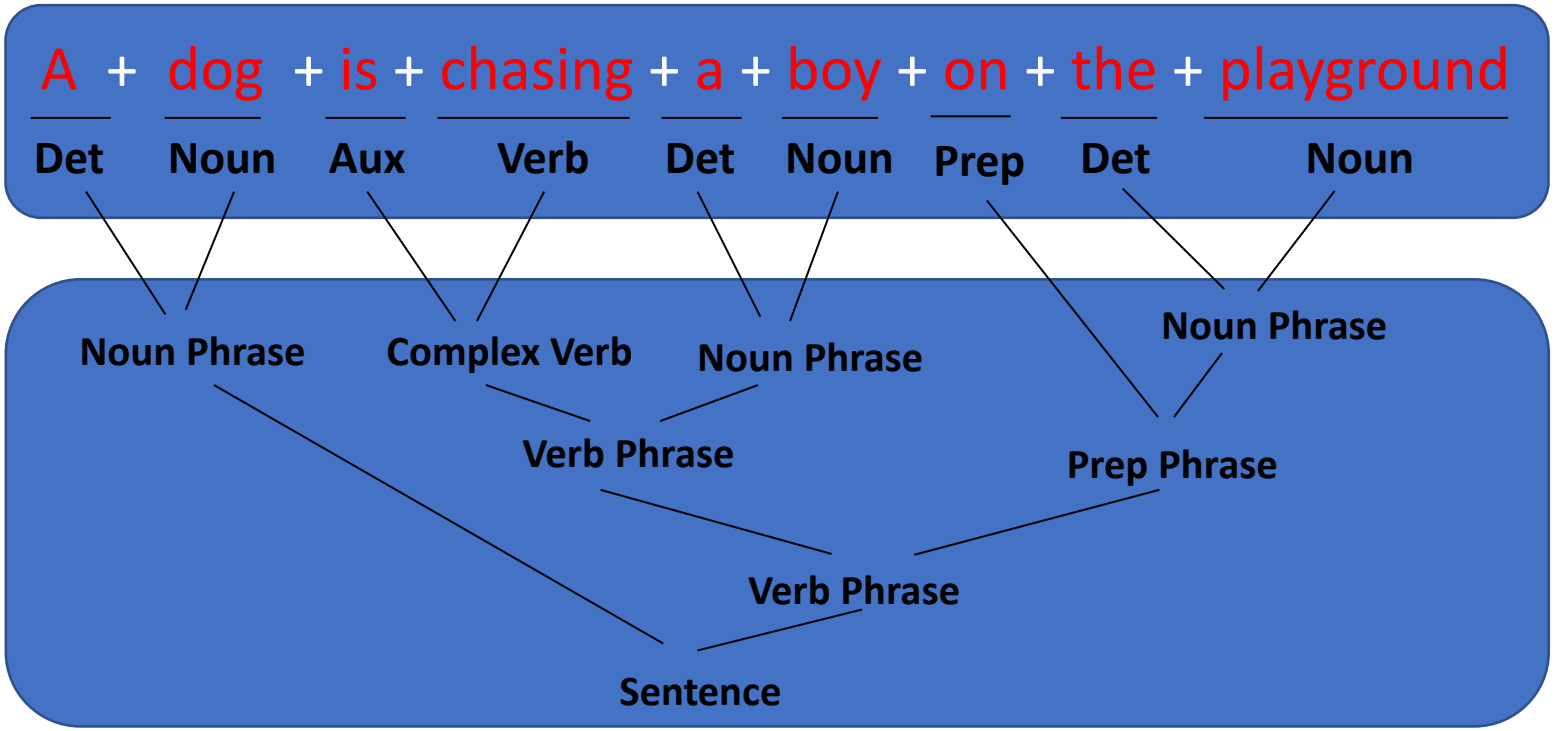
Its initial Board of Visitors included U.S. Presidents Thomas Jefferson, James Madison, and James Monroe.

Its initial Board of Visitors included U.S. Presidents Thomas Jefferson, James Madison, and James Monroe.

Organization, **Location**, **Person**

Syntactic parsing

- Phân tích ngữ pháp của một câu cho trước theo các quy tắc ngữ pháp



Trích rút quan hệ (Relation extraction)

- Xác định quan hệ giữa các thực thể
  - Phân tích ngữ nghĩa ở mức nông

Its initial Board of Visitors included U.S. Presidents Thomas Jefferson, James Madison, and James Monroe.

1. Thomas Jefferson Is_Member_Of Board of Visitors
2. Thomas Jefferson Is_President_Of U.S.

Suy diễn logic

- Phân tích ngữ nghĩa mức sâu

Its initial Board of Visitors included U.S. Presidents Thomas Jefferson, James Madison, and James Monroe.

$\exists x$ (Is_Person(x) & Is_President_Of(x,'U.S.') & Is_Member_Of(x,'Board of Visitors'))

Biểu diễn từ và văn bản

Biểu diễn từ như thế nào?

- WordNet: một từ điển chứa danh sách các từ đồng nghĩa (synonym sets) và bao hàm nghĩa (hypernyms)

e.g. synonym sets containing "good":

```
from nltk.corpus import wordnet as wn
poses = { 'n':'noun', 'v':'verb', 's':'adj (s)', 'a':'adj', 'r':'adv'}
for synset in wn.synsets("good"):
    print("{}: {}".format(poses[synset.pos()],
            ", ".join([l.name() for l in synset.lemmas()])))
```

```
noun: good
noun: good, goodness
noun: good, goodness
noun: commodity, trade_good, good
adj: good
adj (sat): full, good
adj: good
adj (sat): estimable, good, honorable, respectable
adj (sat): beneficial, good
adj (sat): good
adj (sat): good, just, upright
…
adverb: well, good
adverb: thoroughly, soundly, good
```

e.g. hypernyms of "panda":

```
from nltk.corpus import wordnet as wn
panda = wn.synset("panda.n.01")
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

Nhược điểm WordNet

- Thiếu sắc thái
  - Ví dụ “hy sinh” đồng nghĩa với “chết”
- Thiếu nghĩa các từ mới
  - Các từ mới về công nghệ, ngôn ngữ teen…
- Phụ thuộc suy nghĩ chủ quan của người làm
- Cần sức lao động lớn để tạo ra và chỉnh sửa
- Không thể tính độ tương đồng giữa hai từ

Biểu diễn one-hot

- Biểu diễn từ như các ký hiệu rời rạc
- Độ dài vector bằng số từ trong từ điển

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

Vấn đề của biểu diễn one-hot

- Người dùng tìm kiếm “Hanoi hotel”, ta cũng sẽ muốn hiển thị các kết quả của “Hanoi motel”
- Nhưng hai từ này biểu diễn trực giao, độ tương đồng bằng 0!

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0]

hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0]

- **Giải pháp:**
  - Dựa vào WordNet? nhưng WordNet không hoàn thiện và nhiều nhược điểm…
  - **Học để mã hóa sự tương đồng trong các biểu diễn véctơ**

Vấn đề của biểu diễn one-hot

- Biểu diễn quá dài
- Với ngôn ngữ hàng ngày khoảng 20K từ, dịch máy 50K từ, khoa học vật liệu 500K từ, google web crawl 13M từ

motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

hotel = [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]

Biểu diễn từ bằng ngữ cảnh của nó

- Ngữ nghĩa phân tán: Ý nghĩa một từ được quyết định bởi các từ thường xuất hiện gần nó

  "You shall know a word by the company it keeps"

  (J. R. Firth 1957: 11)

- Khi một từ xuất hiện trong văn bản, ngữ cảnh của nó là tập hợp các từ xuất hiện bên cạnh (trong một cửa sổ có kích thước cố định)
- Dùng nhiều ngữ cảnh khác nhau của một từ để xây dựng ý nghĩa của nó

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

Word vector

- Mỗi từ được biểu diễn bởi một véc-tơ dày (dense) sao cho véc-tơ này tương tự với các véc-tơ biểu diễn các từ khác mà thường xuất hiện trong các ngữ cảnh tương tự
- Word vectors còn được gọi là word embeddings hay word representations

$$banking = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

Word vector

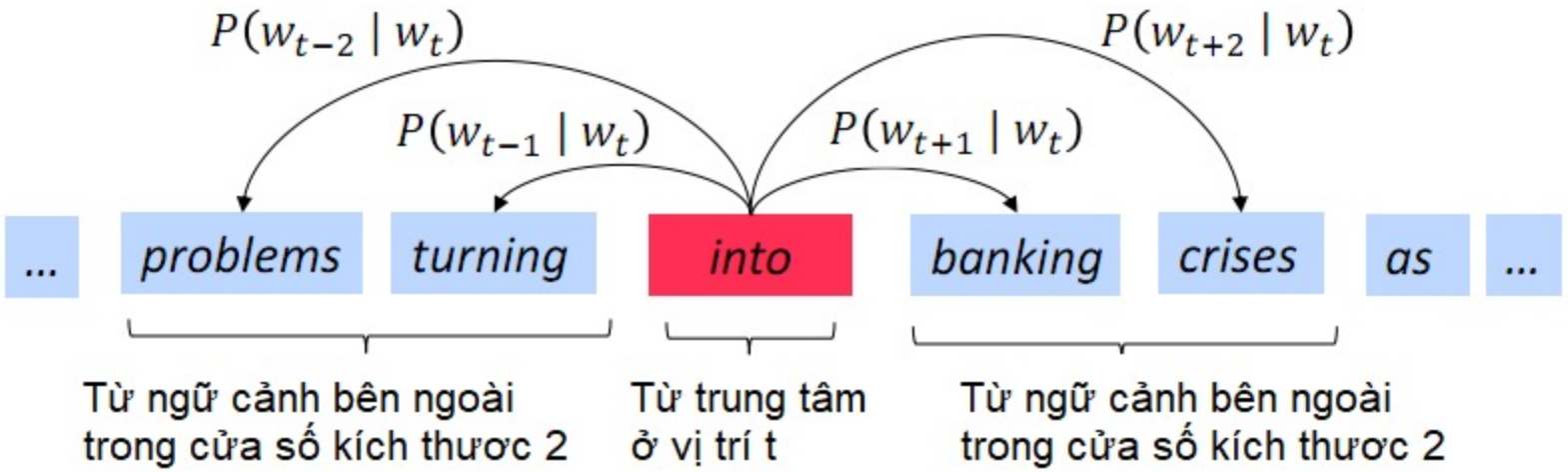
$$expect = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}$$

Word2vec

- Word2vec (Mikolov et al. 2013) là phương pháp để học biểu diễn từ
- **Ý tưởng:**
  - Sử dụng một tập lớn nhiều văn bản (corpus)
  - Mỗi từ trong tập từ vựng cố định được biểu diễn bằng một véctơ
  - Duyệt từng vị trí t trong văn bản, mỗi vị trí chứa từ trung tâm c và các từ ngữ cảnh bên ngoài o
  - Sử dụng độ tương đồng của các véc-tơ biểu diễn c và o để tính xác suất xuất hiện o khi có c (hoặc ngược lại)
  - Tinh chỉnh word véctơ để cực đại hóa xác suất này

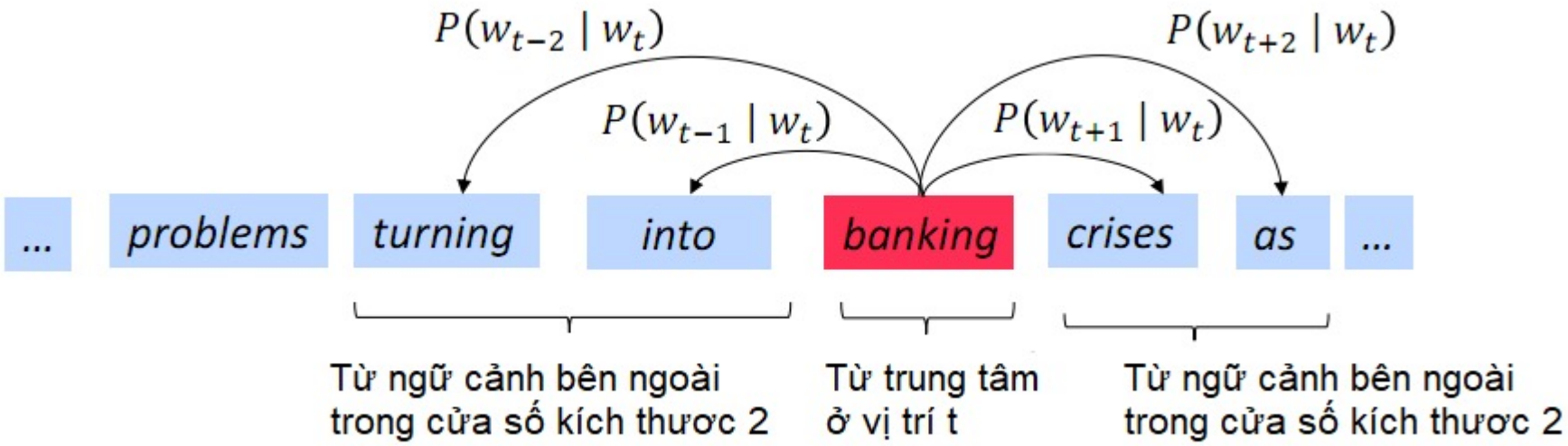
Word2vec

- Ví dụ tính $P(w_{t+j} \mid w_t)$ trong cửa sổ kích thước 2



Word2vec

- Ví dụ tính $P(w_{t+j} \mid w_t)$ trong cửa sổ kích thước 2



Word2vec: Hàm mục tiêu

- Likelihood:

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$$

- Hàm mục tiêu:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

Word2vec

- Làm sao để tính $P(w_{t+j} \mid w_t; \theta)$?

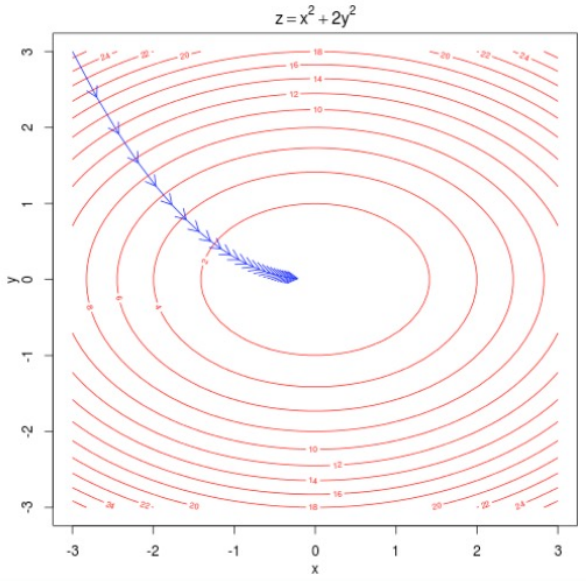
- Ta sẽ dùng hai véctơ cho mỗi từ w:
  - v_w khi w là từ trung tâm
  - u_w khi w là từ ngữ cảnh ngoài
- Khi đó với từ trung tâm c và từ ngữ cảnh ngoài o ta có:

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Word2vec

- Tham số mô hình:

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$

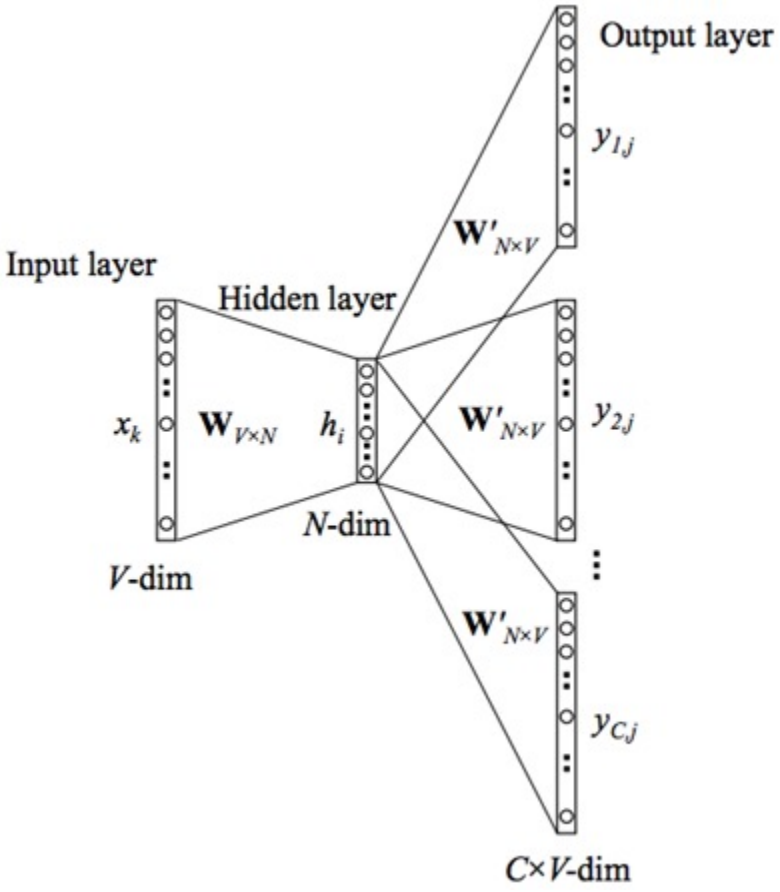


- Huấn luyện bằng SGD:

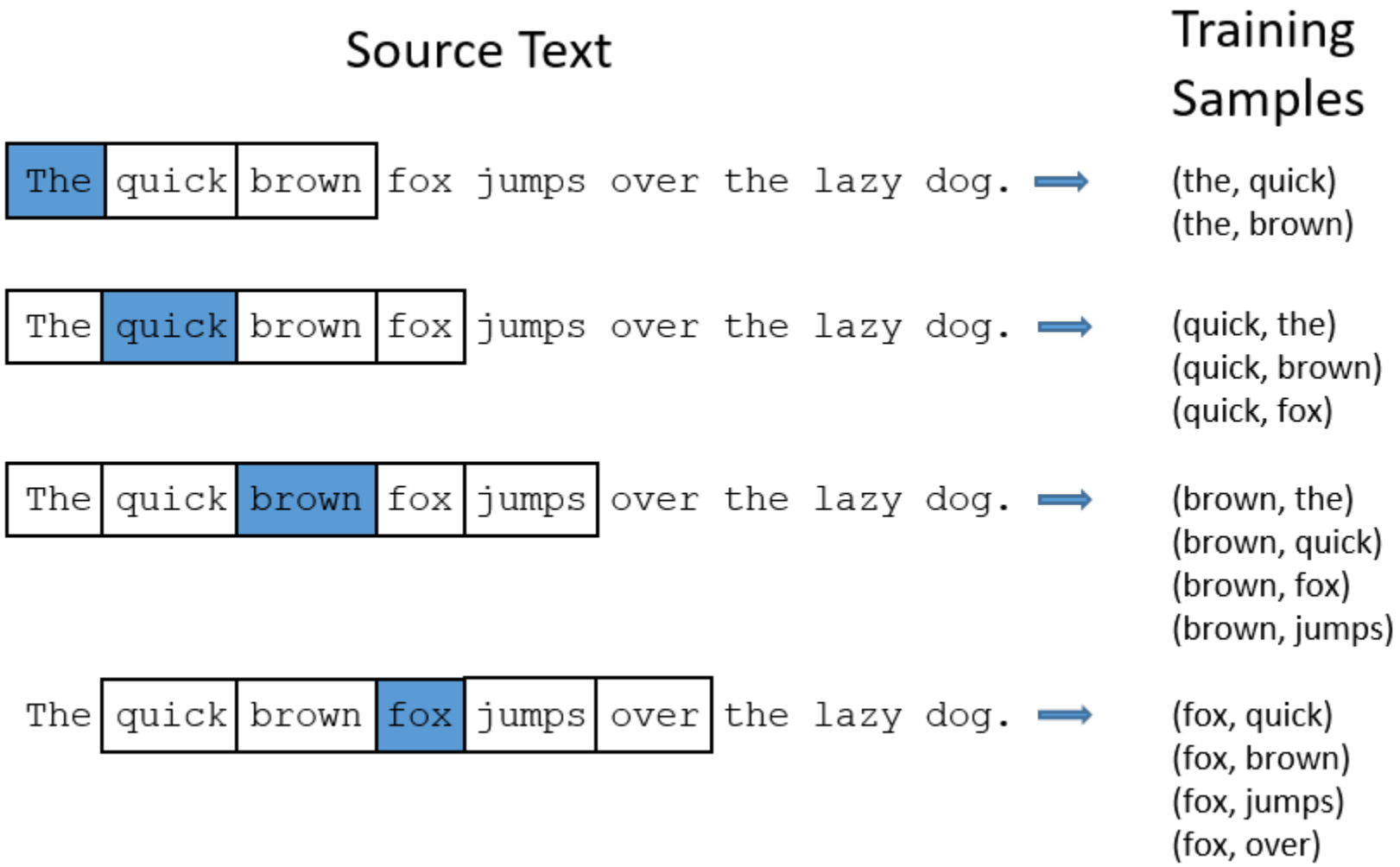
$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$$

Word2vec: The skip-gram model

- Kích thước từ điển: V
- Lớp input: mã hóa one-hot của từ trung tâm.
- Hàng thứ k của ma trận W_{VxN} là véctơ trung tâm biểu diễn từ thứ k.
- Cột thứ k của ma trận W'_{NxV} là véctơ ngữ cảnh của từ thứ k trong V. Chú ý mỗi từ được biểu diễn bởi 2 véctơ, cả hai đều khởi tạo ngẫu nhiên.



Word2vec: The skip-gram model



Word2vec: The skip-gram model

- Vấn đề: Mẫu số tính toán rất lâu!

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

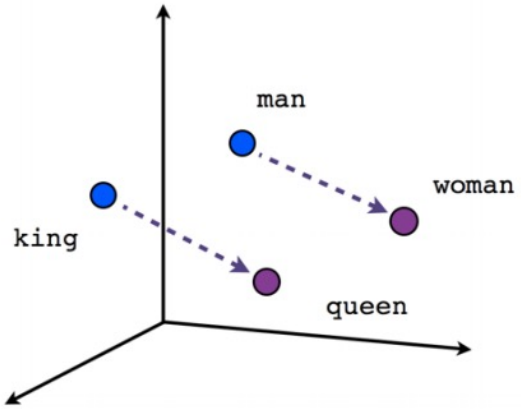
- Sử dụng negative sampling:

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} J_t(\theta)$$

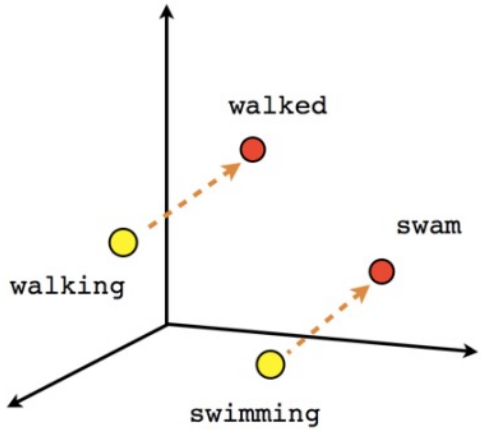
$$J_t(\theta) = \log \sigma\left(u_o^T v_c\right) + \sum_{i=1}^{\kappa} \mathbb{E}_{j \sim P(w)} \left[\log \sigma\left(-u_j^T v_c\right)\right]$$

- $p(w)=U(w)^{3/4}/Z$, trong đố U(w) là phân bố 1-gram.

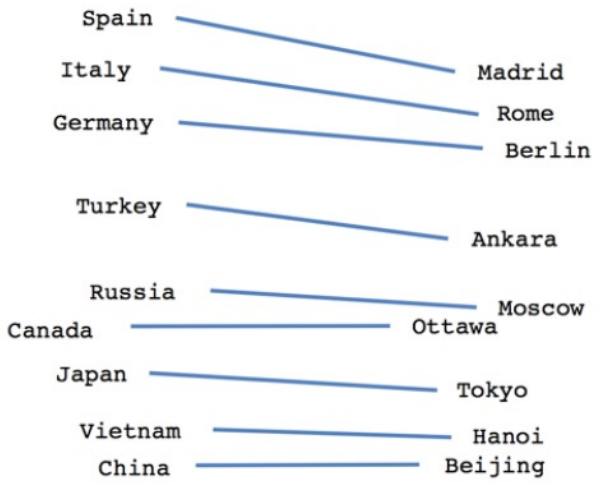
Một số kết quả word2vec



Male-Female

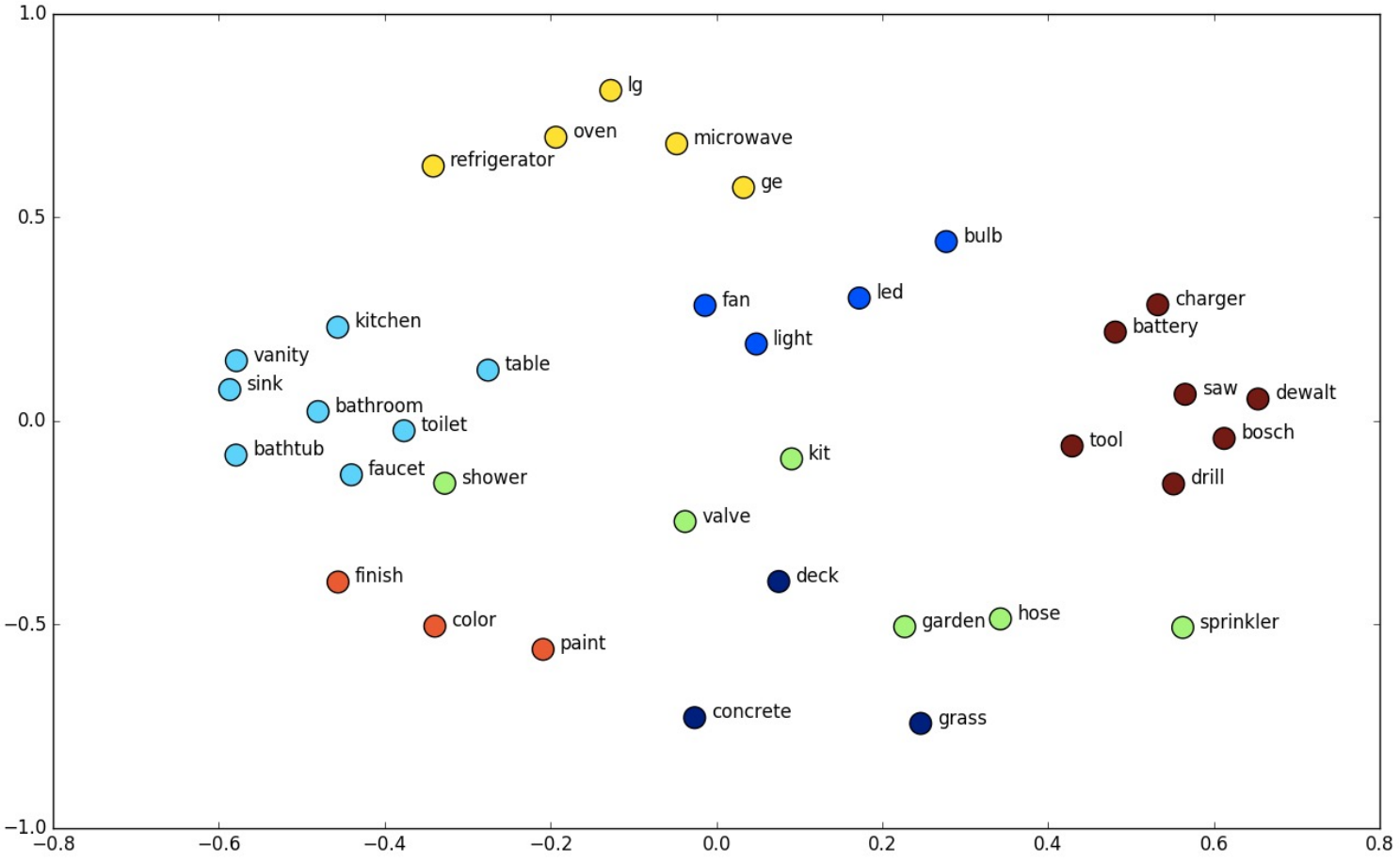


Verb tense

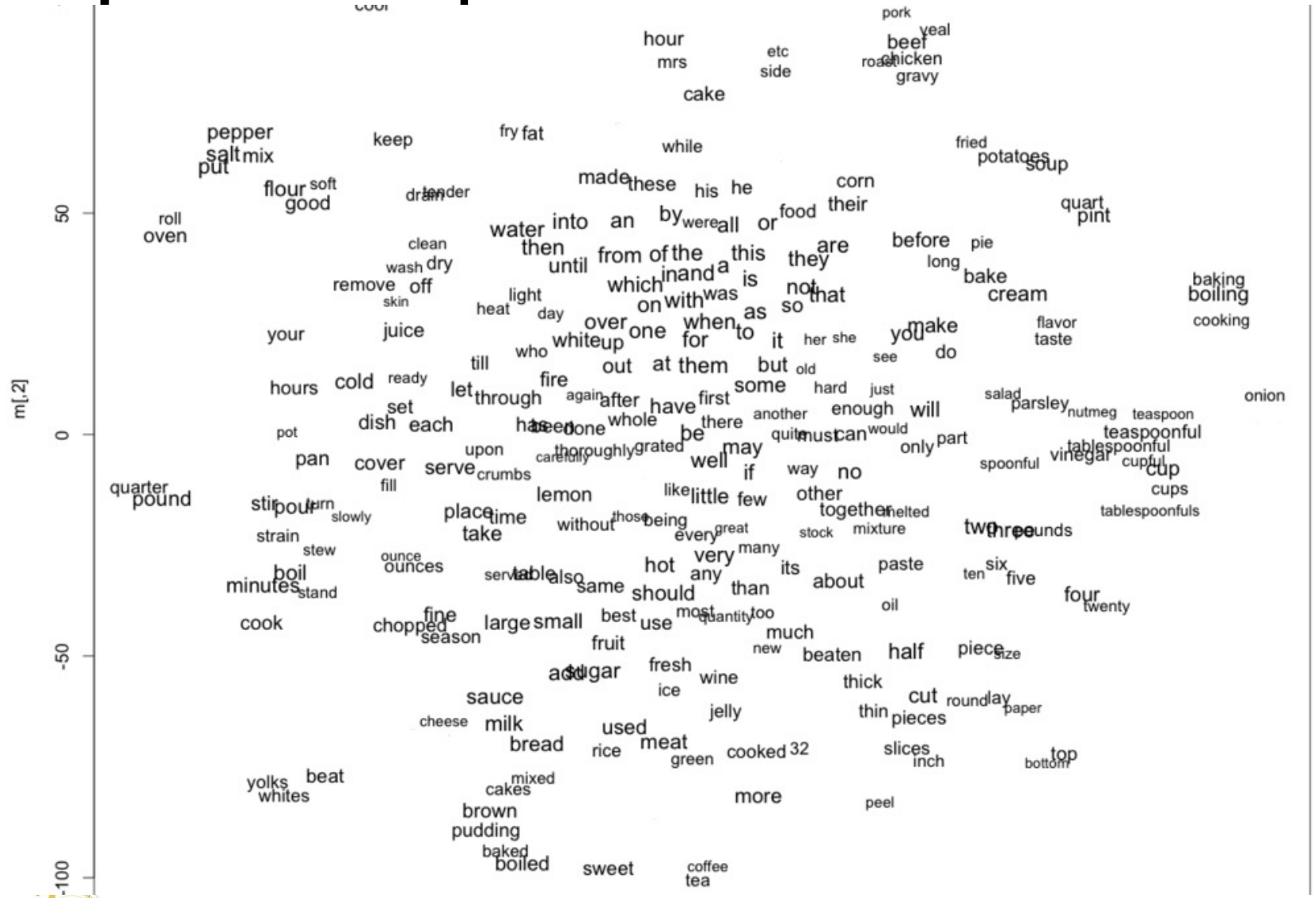


Country-Capital

Một số kết quả word2vec

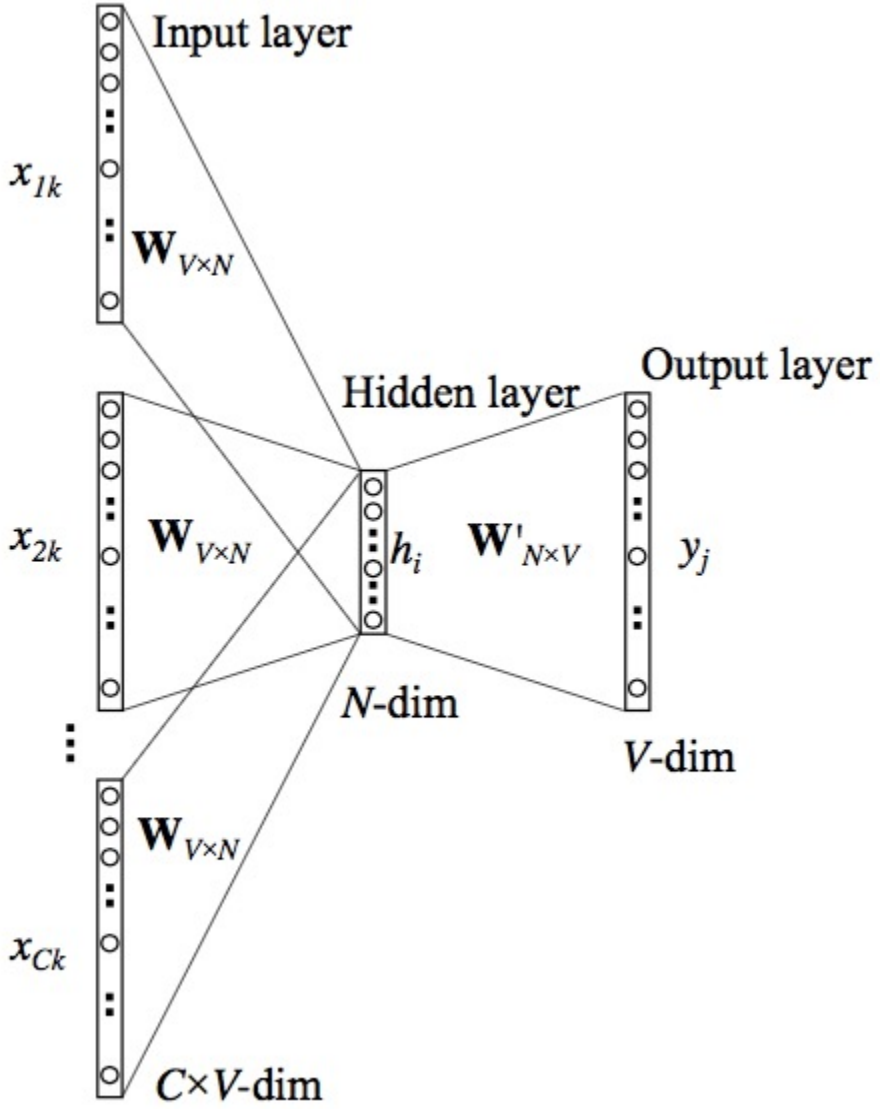


Một số kết quả word2vec



Word2vec: Continuous BOW

- Dùng các từ ngữ cảnh để đoán từ trung tâm



Window based co-occurrence matrix

- Kích thước cửa sổ 1 (thường 5-10)
- Đối xứng (không phân biệt trái phải)
- Ví dụ corpus:
  - I like deep learning.
  - I like NLP.
  - I enjoy flying.

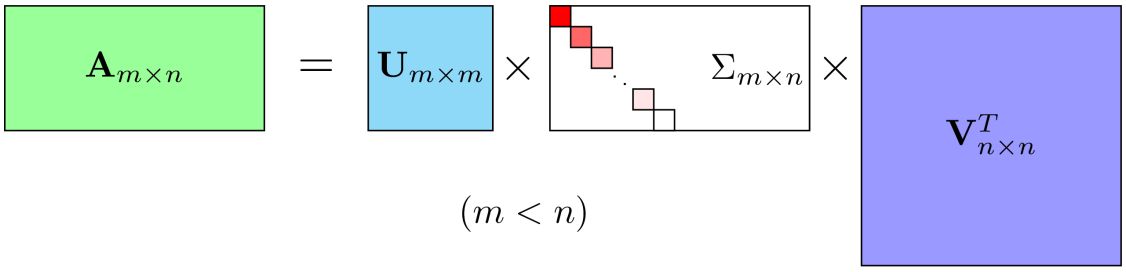
Ma trận đồng xuất hiện dựa trên cửa sổ (co-occurrence matrix)

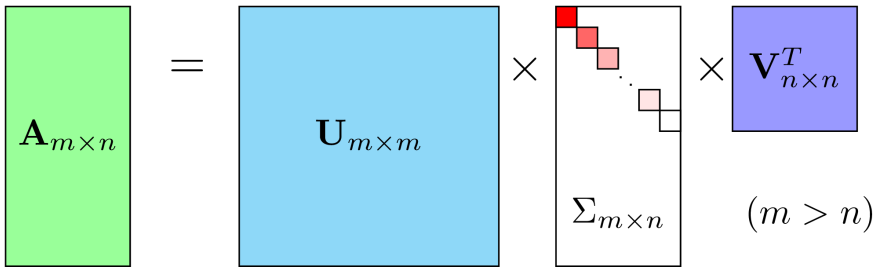
- Kích thước cửa sổ 1 (thường 5-10)
- Đối xứng (không phân biệt trái phải)
- Ví dụ corpus:
  - I like deep learning.
  - I like NLP.
  - I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Vấn đề ma trận đồng xuất hiện

- Kích thước tăng khi số từ tăng
- Số chiều cao, đòi hỏi nhiều bộ nhớ lưu trữ
- **Giải pháp:**
  - Giảm chiều
  - Thường 25-1000 chiều (tương đương word2vec)



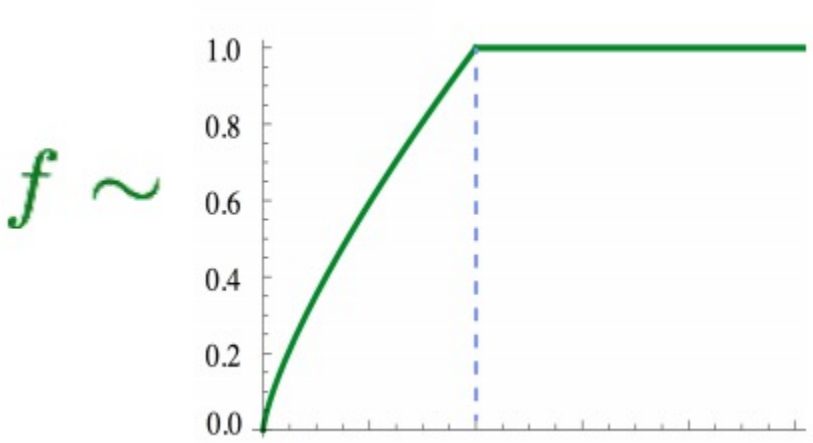


GloVe

- Kết hợp word2vec và ma trận đồng xuất hiện:

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

- Huấn luyện nhanh
- Có thể mở rộng cho corpus lớn
- Hiệu năng tốt ngay cả với corpus nhỏ và véctơ bé



Thư viện Xử lý NNTN và một số mô hình huấn luyện sẵn

Gensim

- Cài đặt: pip install gensim

```
from gensim.models.word2vec import Word2Vec
from multiprocessing import cpu_count
import gensim.downloader as api

# Download dataset
dataset = api.load("text8")
data = [d for d in dataset]

# Split the data into 2 parts. Part 2 will be used later to update the model
data_part1 = data[:1000]
data_part2 = data[1000:]

# Train Word2Vec model. Defaults result vector size = 100
model = Word2Vec(data_part1, min_count = 0, workers=cpu_count())

# Get the word vector for given word
model['topic']
#> array([ 0.0512,  0.2555,  0.9393, ... ,-0.5669,  0.6737], dtype=float32)
```

Gensim

```
model.most_similar('topic')
#> [('discussion', 0.7590423822402954),
#>  ('consensus', 0.7253159284591675),
#>  ('discussions', 0.7252693176269531),
#>  ('interpretation', 0.7196053266525269),
#>  ('viewpoint', 0.7053568959236145),
#>  ('speculation', 0.7021505832672119),
#>  ('discourse', 0.7001898884773254),
#>  ('opinions', 0.6993060111999512),
#>  ('focus', 0.6959210634231567),
#>  ('scholarly', 0.6884037256240845)]

# Save and Load Model
model.save('newmodel')
model = Word2Vec.load('newmodel')

# Update the model with new data.
model.build_vocab(data_part2, update=True)
model.train(data_part2, total_examples=model.corpus_count, epochs=model.iter)
model['topic']
# array([-0.6482, -0.5468,  1.0688,  0.82  , ... , -0.8411,  0.3974], dtype=float32)
```

Gensim

- Sử dụng pretrained từ Gensim

```
import gensim.downloader as api

# Download the models
fasttext_model300 = api.load('fasttext-wiki-news-subwords-300')
word2vec_model300 = api.load('word2vec-google-news-300')
glove_model300 = api.load('glove-wiki-gigaword-300')

# Get word embeddings
word2vec_model300.most_similar('support')
# [('supporting', 0.6251285076141357),
#  ...
#  ('backing', 0.6007589101791382),
#  ('supports', 0.5269277691841125),
#  ('assistance', 0.520713746547699),
#  ('supportive', 0.5110025405883789)]
```

Một số pretrained

BERT:

- Github: https://github.com/google-research/bert
- Bài báo: Bidirectional Encoder Representations from Transformers

XLNet:

- Github: https://github.com/zihangdai/xlnet
- Bài báo: XLNet: Generalized Autoregressive Pretraining for Language Understanding

Tài liệu tham khảo

1. Khóa cs244n của Stanford:

https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Chân thành cảm ơn!!!