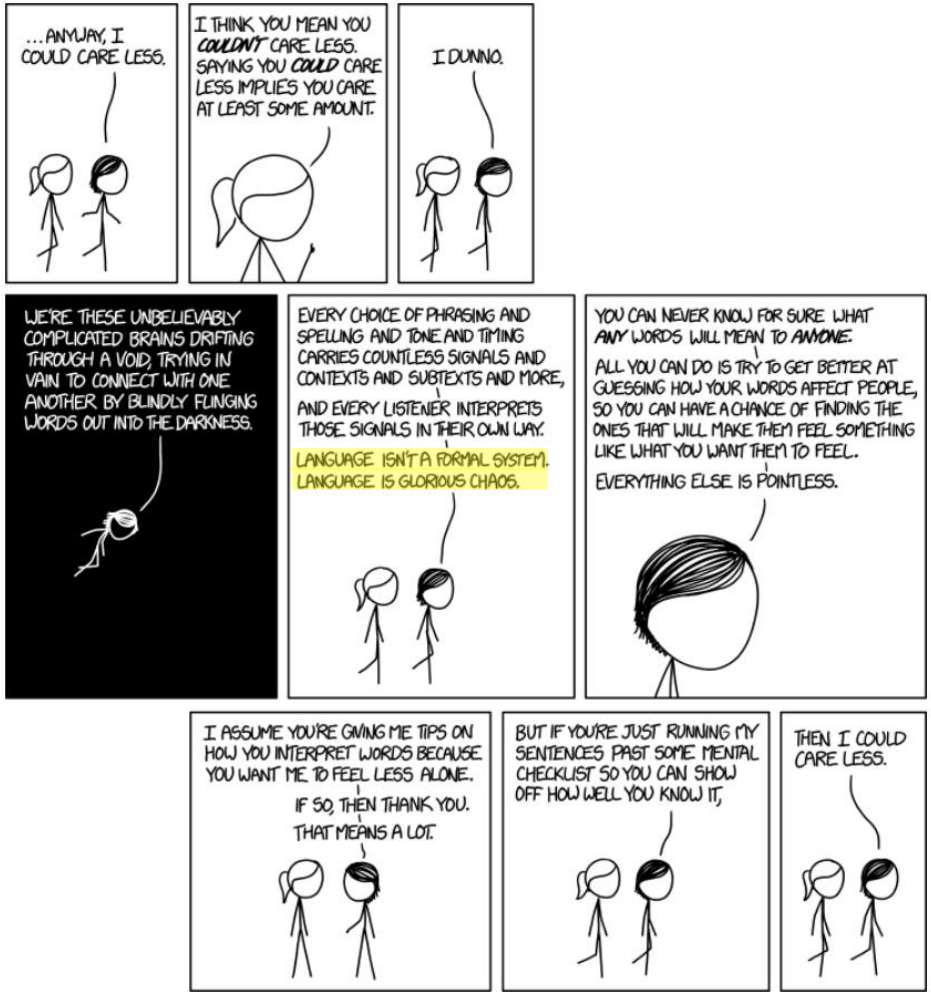
Stanford CS224N NLP with Deep Learning | Winter 2021 | Lecture 1 - Introduction and Word Vectors

2022.03.17

발표자 : 김병찬

GOAL

- NLP 적용을 위한 효과적인 딥러닝 방법론 수립(ex) RNN, Attention, transformer)
- 인간언어의 이해 및 이를 이해하고 생산하는 것들에 대한, 언어를 처리하는 감각 함양
- NLP Task를 해결함에 있어서 PyTorch를 활용하여 시스템을 구축하고 이해



출처 : https://xkcd.com/1576/

언어는 심오하다.

언어는 이해하기 어렵다.

언어는 다른 사람에게 각기 다른 영향과 해석을 줄 수 있다.

어떻게 단어의 의미를 표현할까??

•의미 :
- 말이나 글의 뜻.
- 행위나 현상이 지닌 뜻.
- 사물이나 현상의 가치.

나무

signifier (symbol) ⇔ signified (idea or thing)

= denotational semantics

표시론적 의미론

컴퓨터에서는 이 의미를 어떻게 사용할 수 있을까

Ex) Wordnet
- 유의어 집단으로 분류하여 간략하고 일반적 정의한 어휘목록
- 유의어 사이의 관계를 그래프로 정의
- 유사한 단어를 파악 가능

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')]
```

정답

나무

기둥이 있다

가지가 달려있다

그 가지에 잎사귀가 있다.

유의어로 사용!

나무라는 특징을 갖고 있는가 ?
YES!

WORDNET의 문제점

1 . 뉘앙스를 다루기 힘들다.
Ex) proficient (능숙한) == good(좋은, 훌륭한)

2. 새로운 의미의 단어를 적용하기 어려움.

3. 주관적이다.

4. 인간의 노동력이 많이 소모됨.

전통적인 NLP 문제점

- > 단순한 심볼로 여겼다. (명목형 변수)

WHY ?
통계적 추론 방법 중 하나인 로지스틱 회귀분석을 사용하기 위한 FEATURE로 여겼기 때문

In traditional NLP, we regard words as discrete symbols:
hotel, conference, motel – a localist representation

Means one 1, the rest 0s

Such symbols for words can be represented by one-hot vectors:

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]

전통적인 NLP 문제점

In traditional NLP, we regard words as discrete symbols:
hotel, conference, motel – a localist representation

Means one 1, the rest 0s

Such symbols for words can be represented by one-hot vectors:

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]

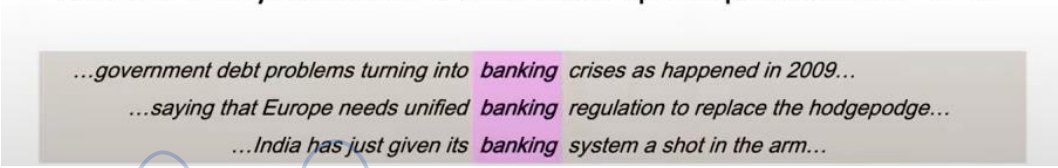
문제점
1. 데이터 용량이 너무 많이 사용됨
2. 단어 들 간 유사성을 알 수 없음.

해결방안
- 과거에는 wordnet
- ★문맥으로부터 단어 표현

문맥으로부터 단어 표현

분포의미론 :
단어의 의미는 빈번하게 등장하는, 근접한 단어에 의해서 정해진다.

- When a word *w* appears in a text, its **context** is the set of words that appear nearby (within a fixed-size window).
- Use the many contexts of *w* to build up a representation of *w*



Q : 나는 오전 8시에 ( )을 먹고 출근을 한다.

1. 축구 2.영화관 3.아침밥 4.어제

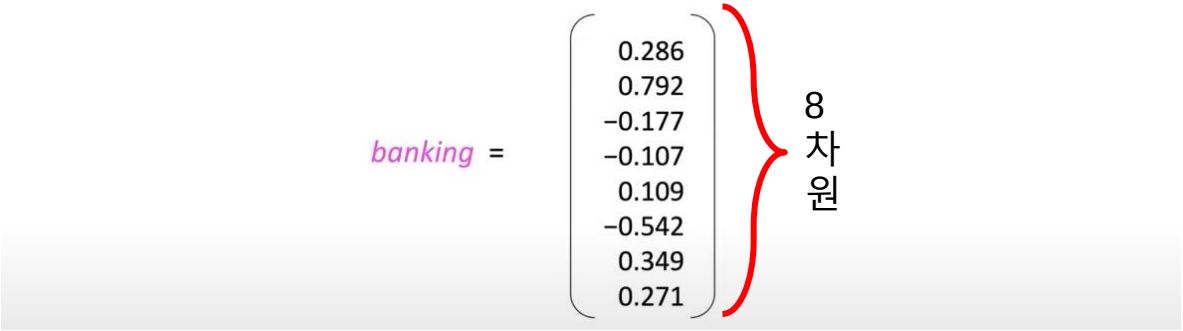
학교 가기전에 ( ) 꼭 먹고 가라

( )을 먹는 것은 하루를 시작하는 데에 중요한 역할을 합니다.

Word Vector

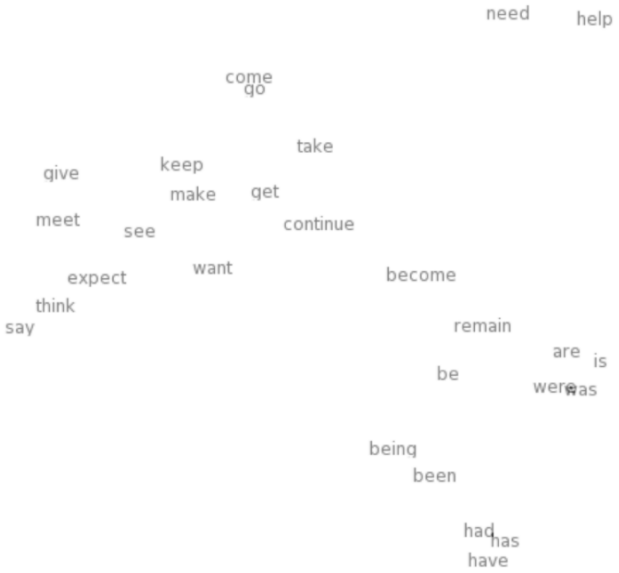
그렇다면 분포의미론에 의거해서 단어들을 어떤 식으로 표현할 수 있을까 ?
A : Word Vector
-> 각 단어에 대한 real-value 값으로 문맥에서 나오는 값과 유사성을 계산 가능.

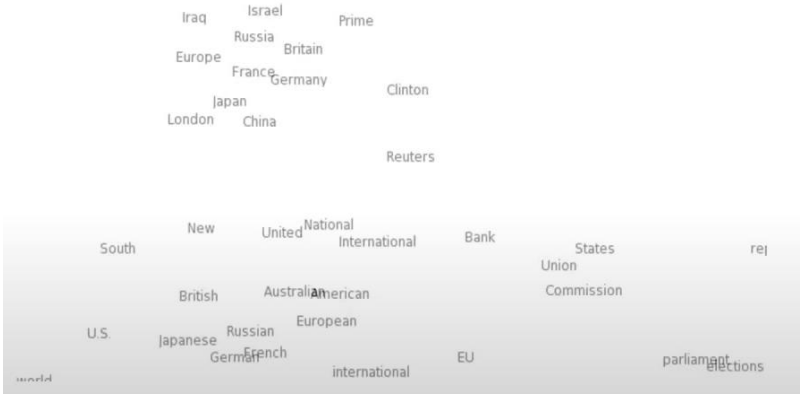
We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts



‘banking’ 이라는 단어는 8개의 차원으로 의미가 분산됨.
기존의 localist representation과 비교했을 때 존재했을 때만 의미를 가지고 나머지는 필요없는 정보가 되는 것에 비해 정보의 손실이 적으며 다른 단어와 상호작용할 요소가 많아짐

Word Vector


need
help
come
go
take
give
keep
make
get
meet
see
continue
expect
want
become
think
say
remain
are
is
be
were
was
being
been
had
has
have


Iraq
Israel
Prime
Russia
Britain
Europe
France
Germany
Clinton
Japan
London
China
Reuters
New
United
National
International
Bank
South
States
Union
Commission
British
Australian
American
European
U.S.
Japanese
Russian
French
German
EU
parliament
elections
international

Word2Vec

: Word vector를 학습시키기 위한 framework

원리

- 다량의 말뭉치를 가지고 있음(data 多)
- 각 단어들은 벡터로 표현됨.(자르더라도 본질은 없어지지 않게)
  -> 이 말뭉치로부터 얻은 워드 벡터들은 문맥에서 어떤 단어가 나올지 예측이 가능하다.
- T개의 텍스트 중 중심단어 c와 바깥단어o를 가지게 됨.
- c와 o의 word vector 간 유사도를 사용하여 확률을 계산 (o | c)
- 이 확률을 최대화하며 조정

Word2Vec

O(바깥단어)

C(중심단어)

WINDOW_SIZE : 2

T : 5

나는 | 8시에 | 아침밥을 | 꼭 | 먹는다.

P(나는 | 아침밥을) P(8시에 | 아침밥을) P(꼭 | 아침밥을) P(먹는다 | 아침밥을)

For each position $t = 1, \dots, T$, predict context words within a window of fixed size m, given center word w_j. Data likelihood:

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

θ is all variables to be optimized

우도 : 주어진 파라미터가 가정한 모델에 얼마나 적합한지에 대한 조건부 확률 즉 클수록 모델이 관측된 데이터를 잘표현함을 의미

sometimes called a *cost* or *loss* function

The objective function $J(\theta)$ is the (average) negative log likelihood:

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

Negative log-likelihood 사용 이유
: 학습이 잘될 수록 target과 output 간 격차가 줄어들어야 함. 즉 학습이 잘될수록 작은 값이 도출되게끔 해야함.

Minimizing objective function ⟺ Maximizing predictive accuracy

Word2Vec

- We want to minimize the objective function:

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

- **Question:** How to calculate $P(w_{t+j} \mid w_t; \theta)$?
- **Answer:** We will *use two* vectors per word *w*:
  - v_w when *w* is a center word
  - u_w when *w* is a context word
- Then for a center word *c* and a context word *o*:

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

Word2vec: prediction function

② Exponentiation makes anything positive

① Dot product compares similarity of *o* and *c*.
$u^T v = u.v = \sum_{i=1}^{n} u_i v_i$
Larger dot product = larger probability

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

③ Normalize over entire vocabulary to give probability distribution

- This is an example of the **softmax function** $\mathbb{R}^n \to (0,1)^n$ (Open region)

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n}\exp(x_j)} = p_i$$

- The softmax function maps arbitrary values x_i to a probability distribution p_i
  - "max" because amplifies probability of largest x_i
  - "soft" because still assigns some probability to smaller x_i
  - Frequently used in Deep Learning

But sort of a weird name because it returns a distribution!

Word2Vec

O(바깥단어)

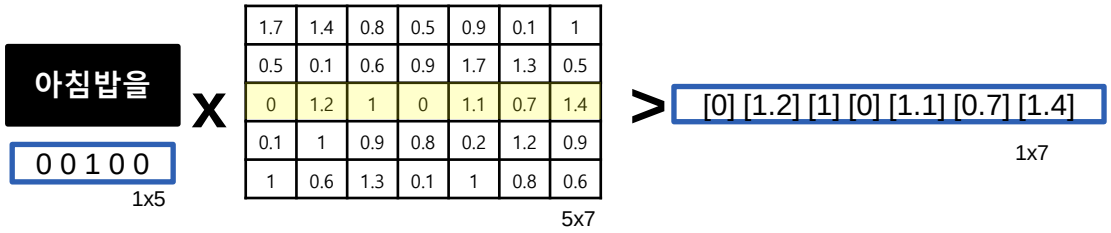
C(중심단어)

WINDOW_SIZE : 2

T : 5

나는 | 8시에 | 아침밥을 | 꼭 | 먹는다.

P(나는 | 아침밥을) P(8시에 | 아침밥을) P(꼭 | 아침밥을) P(먹는다 | 아침밥을)



Word2Vec

O(바깥단어)

C(중심단어)

WINDOW_SIZE : 2

T : 5

나는 | 8시에 | 아침밥을 | 꼭 | 먹는다.

P(나는 | 아침밥을) P(8시에 | 아침밥을) P(꼭 | 아침밥을) P(먹는다 | 아침밥을)

[0] [1.2] [1] [0] [1.1] [0.7] [1.4]

1x7

X

0.3	1.1	0.2	0	1.4
0.9	0.9	0.2	0.1	1.6
0.6	0.1	0	0.1	0.3
1.4	0.2	0.5	1.4	0.7
0.3	0.2	1.4	0.1	1
1.3	0.8	0.7	0	0.3
1.4	1.7	0.3	1.4	0.3

7x5

>

[0.46205623,
0.26926246,
0.05171183,
0.03466348,
0.182306])

1x5

나는 [1,0,0,0,0]

8시에 [0,1,0,0,0]

꼭 [0,0,0,1,0]

먹는다 [0,0,0,0,1]

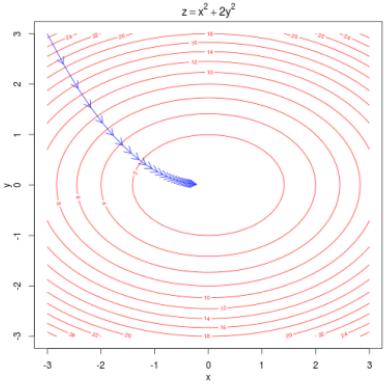
Word2Vec

To train the model: Optimize value of parameters to minimize loss

To train a model, we gradually adjust parameters to minimize a loss

- Recall: θ represents **all** the model parameters, in one long vector
- In our case, with *d*-dimensional vectors and *V*-many words, we have:
- Remember: every word has two vectors

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



- We optimize these parameters by walking down the gradient (see right figure)
- We compute all vector gradients!