



1. Introduction and Word Vectors

1. Human language and word meaning

- 인간의 언어는 아주 큰 컴퓨터 네트워크와 비슷함
 - 언어를 구성하는 문장/단어들은 각각의 의미를 가지고 있음
 - 이러한 단어들의 “의미” 파악을 통해 상대방과 원활한 의사소통을 할 수 있음

단어의 의미?

signifier (symbol) \Leftrightarrow signified (idea or thing)

= denotational semantics

tree \Leftrightarrow {, , , ...}

2. 단어의 의미를 표현하는 방법

2-1. WordNet

- 이전에는 주로 ‘단어 사전’을 생성하는 방식을 활용
 - ex> WordNet 등

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')]
```

- WordNet의 한계
 - WordNet은 동의어, 상하관계 언어의 집합임
 - 주관적인 판단 기준, 뉘앙스를 파악하기 어려움
 - 신조어 생성, 관리에 지속적 인력 투입이 필요
 - 단어 의미 간의 유사도와 관계를 얻기 어려움

2-2. One-hot Vector

- 기존의 NLP에서는 단어를 이산형(discrete) 객체로 취급함
 - **One-Hot Vector**로 표현 → 지역적(localist) 정보

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

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

- 단어의 개수 = Vector의 차원
 - Vector의 차원이 기하급수적으로 늘어나는 문제
- 벡터 차원에서 두 단어 벡터는 “orthogonal”
 - 유사도를 표현하기에 어려움

2-3. Distributional Semantics

- 단어의 문맥을 고려한 방법

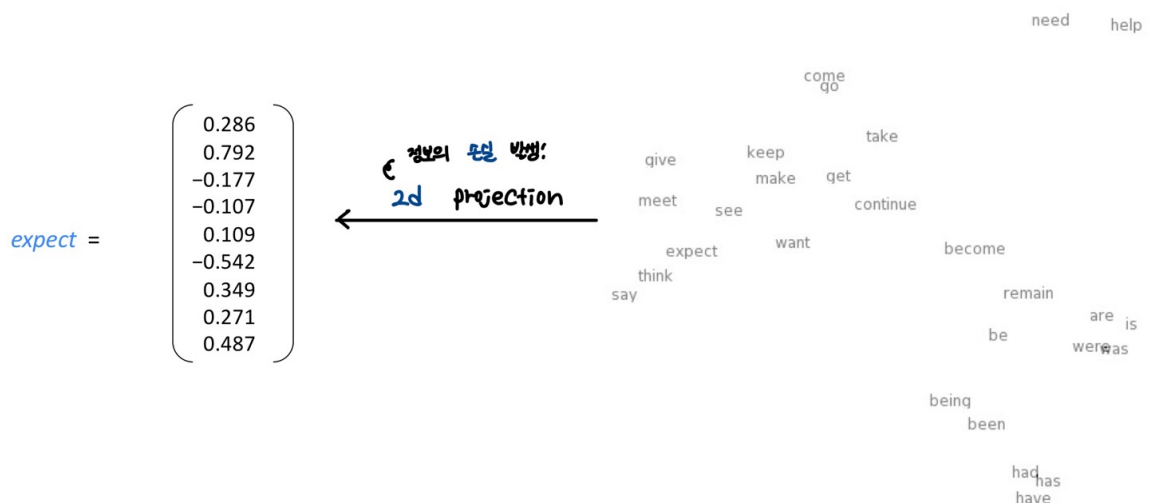
⇒ “가까이 있는 단어일수록 비슷한 의미일 가능성이 높다.”

- fixed size window를 통해 단어를 표현할 때 주위(context)를 살펴, 비슷한 문맥에서 나타나는 비슷한 단어들끼리 유사한 벡터를 가짐

...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...

These context words will represent **banking**

- Word Embeddings, Word Representations 라고도 함
- 벡터 공간(Vector Space)
 - 각 vector들의 배치를 2D 공간에 투영한 것
 - 정확히 투영되지는 않지만(→ 정보의 손실은 발생함) 유사한 단어가 유사한 위치에 있음을 확인할 수 있음



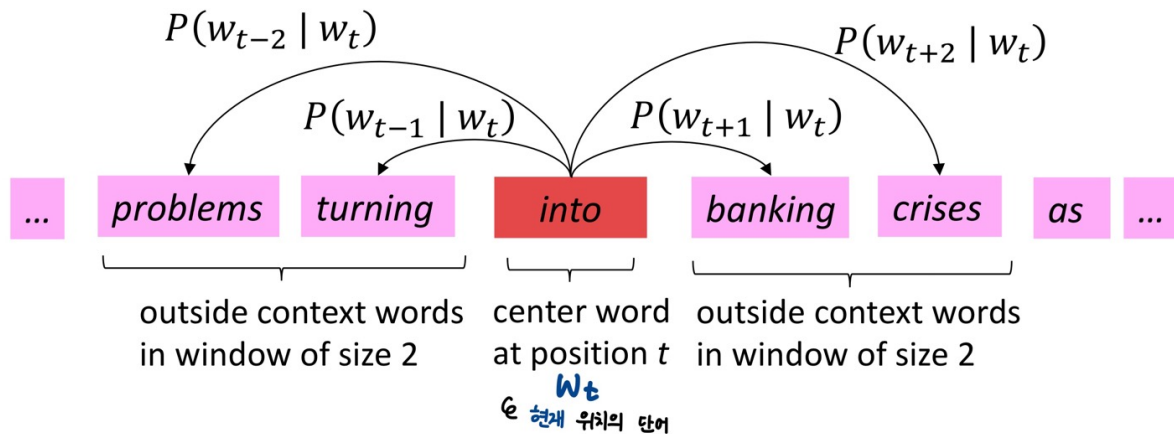
3. Word2Vec

3-1. 개념

- Word Vector을 학습에 쓰이는 프레임워크(= 알고리즘)
- Idea
 - 충분한 양의 corpus를 바탕으로, Random Vector에서부터 시작하여 각 단어를 잘 표현하는 Vector 값을 찾음

- 단어 벡터간의 유사도를 이용해 맥락에서 특정 단어가 나타날 확률을 계산

3-2. 과정



1. 현재 위치 t 에 있는 단어를 W_t , 주변에 있는 단어를 W_{t+n} , W_{t-n} 이라고 할 때, $P(W_{t+n}|W_t)$, $P(W_{t-n}|W_t)$ 를 구함
2. $P(W_{t+n}|W_t)$, $P(W_{t-n}|W_t)$ 를 최대화하는 vector 찾기
3. corpus 안의 모든 단어에 대해 1~2를 반복

3-3. 계산법

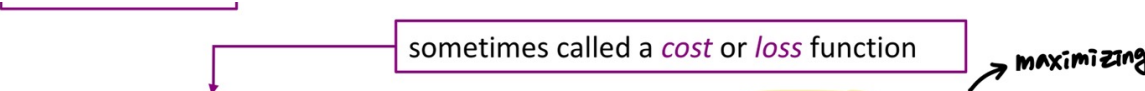
1) $L(\theta)$, Likelihood

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

θ is all variables to be optimized

- word vector θ (parameter)가 주어졌을 때, window 내의 context word가 해당 위치에 나타날 확률의 곱

2) $J(\theta)$, 목적 함수

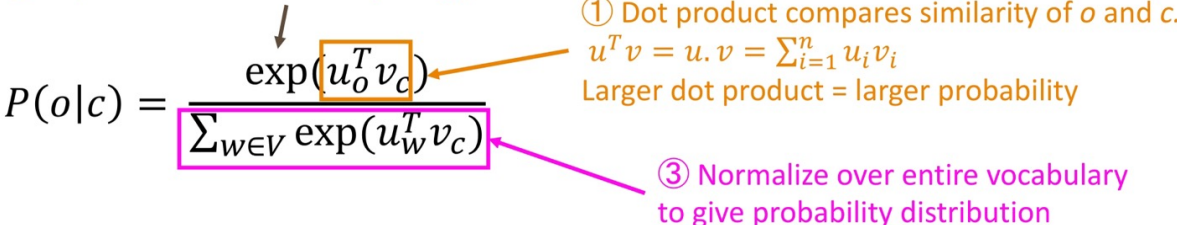

 sometimes called a *cost* or *loss* function → maximizing
 The **objective function** $J(\theta)$ is the (average) **negative log likelihood**:

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

- negative log likelihood 를 거쳐 objective function을 만든 후, 이를 최소화 하는 θ (parameter)를 구함
- objective function을 최소화 \Leftrightarrow predictive accuracy를 최대화

3) P(o|c)

② Exponentiation makes anything positive



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

- 단어마다 두 개의 vector를 학습
 - V_x : x 가 center word임
 - u_x : x 가 context word임
- 내적(dot product)을 통해 유사도 측정
 - 이후 softmax 과정을 거쳐 최종 계산

4) 최적화

- 목적함수를 **최소화** 하는 파라미터 θ , 즉 u 와 v (\Rightarrow word를 나타내는 두 vector) 찾기
 - 손실 최소화 = 우리가 실제로 중심 단어의 맥락에서 본 단어의 확률을 극대화
- V 개의 단어가 존재하고 θ 가 d -dimension vector일 때, word vector는 u, v 를 포함하므로 $2dV$ 차원을 가짐

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

- 모든 벡터의 기울기 계산 → 미분

3-4. 종류

- Skip Gram: center word로 context word를 예측
- CBOW: context word로 center word를 예측

4. Optimization basics

- 편미분을 신나게(?) 해보자
 - Chain Rule(→ 합성함수 미분법) 활용

* Objective Function

목표 > maximize $J'(\theta) = \prod_{t=1}^T \prod_{(w_{t+1}, c_t) \in \text{dataset}} P(w_{t+1} | w_t; \theta)$

i.e. minimize $J(\theta) = - \frac{1}{T} \sum_{t=1}^T \sum_{(w_{t+1}, c_t) \in \text{dataset}} \log P(w_{t+1} | w_t)$

↑ neg
- log likelihood

where $P(o|c) = \frac{\exp(u_o^T \cdot v_c)}{\sum_{w=1} \exp(u_w^T \cdot v_c)}$

↑ softmax

* 모든 단어 type은 (vocab entry)
2개의 word vector를 가지고 o
⇒ $\begin{bmatrix} \text{center word} \\ \text{context word} \end{bmatrix}$

→ 최소화? 미분하자!

6 편미분

$$\frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T \cdot v_c)}{\sum_{\pi} \exp(u_{\pi}^T \cdot v_c)} = \underbrace{\frac{\partial}{\partial v_c} \log \exp(u_o^T \cdot v_c)}_{\textcircled{1}} - \underbrace{\frac{\partial}{\partial v_c} \log \sum_{\pi} \exp(u_{\pi}^T \cdot v_c)}_{\textcircled{2}}$$

$$\textcircled{1} \frac{\partial}{\partial v_c} \log \exp(u_o^T \cdot v_c) = \frac{\partial}{\partial v_c} u_o^T \cdot v_c = u_o$$

$\uparrow v_j$ $\frac{\partial}{\partial (v_c)_j} u_o^T \cdot v_c = \frac{\partial}{\partial (v_c)_j} \sum_{\pi} (u_o)_\pi (v_c)_\pi$
 $= (u_o)_j$
 ($\pi \neq j$ 이면 모두 0으로 소거된다.)

$$\textcircled{2} \frac{\partial}{\partial v_c} \log \sum_{\pi} \exp(u_{\pi}^T \cdot v_c) = \frac{1}{\sum_{\pi} \exp(u_{\pi}^T \cdot v_c)} \cdot \frac{\partial}{\partial v_c} \sum_{\pi} \exp(u_{\pi}^T \cdot v_c)$$

$\uparrow f(x) = \log x \quad \uparrow g(v_c) = \sum_{\pi} \exp(u_{\pi}^T \cdot v_c)$
 (Chain Rule)

$$= \frac{1}{\sum_{\pi} \exp(u_{\pi}^T \cdot v_c)} \cdot \sum_{\pi} \frac{\partial}{\partial v_c} \exp(u_{\pi}^T \cdot v_c)$$

$$= \frac{1}{\sum_{\pi} \exp(u_{\pi}^T \cdot v_c)} \cdot \sum_{\pi} \exp(u_{\pi}^T \cdot v_c) \cdot \frac{\partial}{\partial v_c} (u_{\pi}^T \cdot v_c)$$

$$= \sum_{\pi} \exp(u_{\pi}^T \cdot v_c) \cdot u_{\pi}$$

$$\Rightarrow \frac{\partial}{\partial v_c} \log(p(o|c)) = u_o - \frac{1}{\sum_{\pi} \exp(u_{\pi}^T \cdot v_c)} \times \sum_{\pi} \exp(u_{\pi}^T \cdot v_c) \cdot u_{\pi}$$

$$= u_o - \frac{\sum_{\pi} \exp(u_{\pi}^T \cdot v_c) \cdot u_{\pi}}{\sum_{\pi} \exp(u_{\pi}^T \cdot v_c)}$$

$$= u_o - \frac{\sum_{\pi} p(x|c) \cdot u_{\pi}}{\sum_{\pi} p(x|c)} \rightarrow \text{observed - expected}$$

\uparrow 기댓값!
 \uparrow 학습한 가중된 context vector의 평균

- 목적함수를 편미분한 것은 실제 단어와 예측한 단어와의 차이와 같음
 \Rightarrow (Observed, 실제) - (Expected, 예측)
 \Rightarrow **gradient descent**를 통해 실제에 더 가깝게 예측할 수 있음

