Text Analytics

Ch5 : Text Representation

서수원

Business Intelligence Lab.

산업경영공학과, 명지대학교

01

NNLM

Word Embedding

- Word Embedding
  - 단어를 특정한 공간의 벡터 스페이스로 맵핑한다.
  - 의미적으로 유사한 단어들이 벡터 공간상 각각에 근처에 위치하게 된다.
    - ✓ 이론적으로 영어의 토큰은 1300만개가 있는데, 이는 1300만 차원이 필요하다는 의미이다.
      - ❖ 이는 비효율이기 때문에, 분산표상을 이용한다.

Word Embedding

- Word vectors : one-hot vector
  - 가장 쉽고 이해하기 쉬운 표현 방법이다.

$$w^{aardvark} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{a} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{at} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \ldots w^{zebra} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

motel [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0] = 0

$$(w^{hotel})^{\top} w^{motel} = (w^{hotel})^{\top} w^{cat} = 0$$

Word Embedding

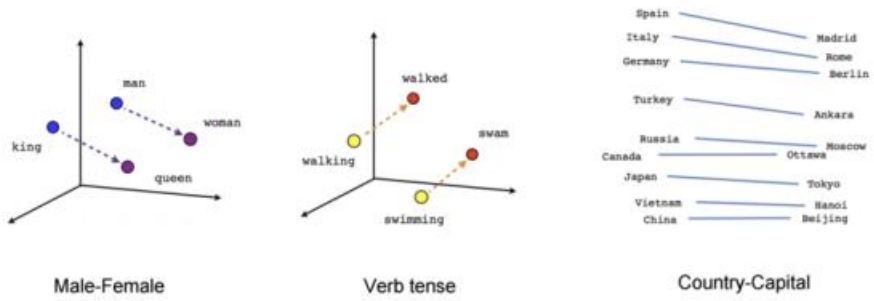
- Word vectors : distributed representation
  - 분산표상, 특정 언어를 특정 차원을 갖는 벡터로 변환하는 매개변수 함수를 의미한다.
    - ✓ N <<lVl (V : voca.)

$$W : \text{words} \rightarrow \mathbb{R}^n$$

$$W(\text{“cat”}) = (0.2,\ \text{-}0.4,\ 0.7,\ \ldots)$$

$$W(\text{“mat”}) = (0.0,\ 0.6,\ \text{-}0.1,\ \ldots)$$

- Word embedding의 특징
  - 단어의 의미적 유사성이 벡터 공간에 보존된다.



Male-Female　　Verb tense　　Country-Capital

NNLM

- NNLM의 목적
  - 분산표상을 처음으로 사용한 방법론이다.
  - One-Hot vector 의 단점을 분산표상으로 해결함이 목적이다.
    - ✓ One-Hot vector는 희소성의 문제, 저장공간의 문제 등이 있다.
  - 각 단어들을 분산표상에 대한 워드벡터로 표현이 가능하다.
  - 단어의 연속성에 의한 결합확률 분포를 통해 단어가 나올 확률을 알 수 있다.
  - 어떤 백터가 좋은지에 대한 것과, 단어의 연관성이 높은지에 대한 확률을 동시에 계산한다.

NNLM

- NNLM
  - 분산 표상을 통해, (강아지,고양이),(the,a),(bedroom,room)의 비슷한 의미를 이해 할 수 있다.
    - ✓ 이는 아래와 같은 새로운 문장을 생성할 수 있음을 의미한다.

The cat is walking in the bedroom

to

A dog was running in a room

The cat is running is a room

A dog is walking in a bedroom

The dog was waling in the room

...

NNLM

- Comparison with Count–based Language Models
  - Count-based Language Models
    - ✓ 체인룰에 의해 모든 분포는 밑과 같이 표현될 수 있다.

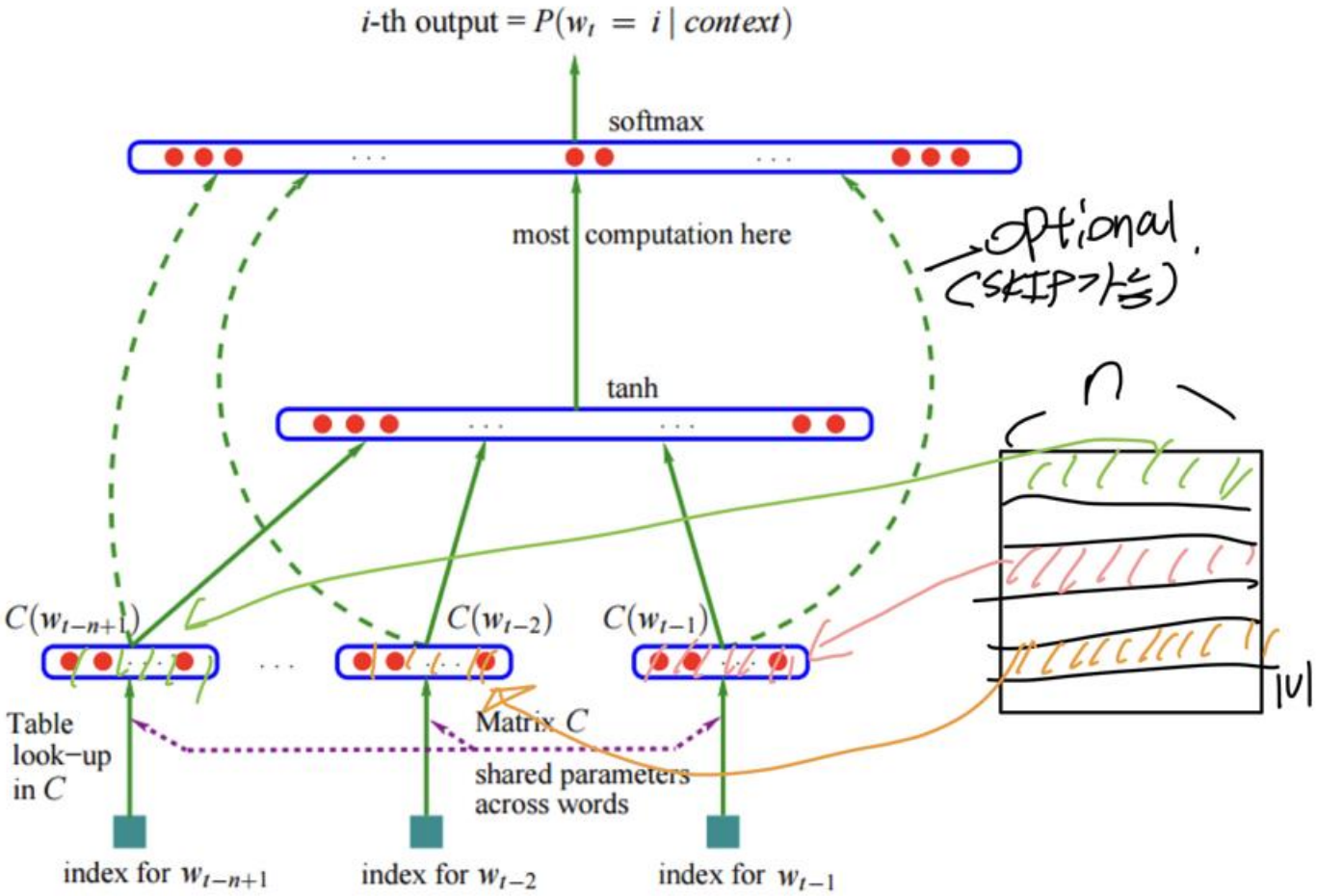
$$p(w_1, \ldots, w_T) = \prod_{t=1}^{T} p(w_t | w_1, \ldots, w_{t-1})$$

    - ✓ 카운트 기반 n-gram language models는 마르코프 가정을 이용한다.

$$p(w_t | w_1, \ldots, w_t) \approx p(w_t | w_{t-n}, \ldots, w_{t-1})$$

NNLM

- NNLM

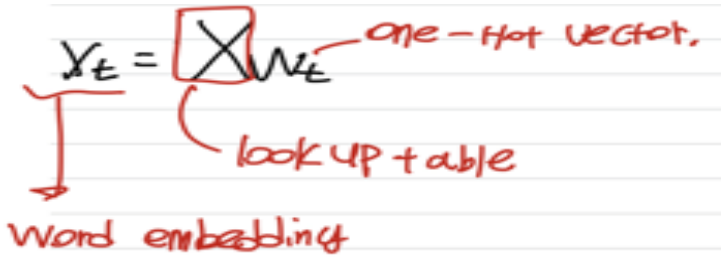

i-th output = P(w_t = i | context)
softmax
most computation here
optional (skip가능)
tanh
C(w_{t-n+1})
C(w_{t-2})
C(w_{t-1})
|V|
Table look-up in C
Matrix C
shared parameters across words
index for w_{t-n+1}
index for w_{t-2}
index for w_{t-1}

NNLM

- NNLM

$$W(\text{“cat”}) = (0.2,\ -0.4,\ 0.7,\ \ldots)$$

  - 단어를 R^n(lnl<lvl)공간의 차원에 dense vectors로 표현한다..(실수로 되어있는 벡터)
  - W_t : One-hot 벡터로 표현된 것을 의미한다.
  - 이때 $x_t = XW_t$: Word embedding



$$P(W_t = j \mid W_1, \ldots, W_{t-1}) = \frac{\exp(p^j \cdot g(x_1, \ldots, x_{t-1}) + q^j)}{\sum_{j' \in V} \exp(p^{j'} \cdot g(x_1, \ldots, x_{t-1}) + q^{j'})} = \text{softmax}(Pg(x_1, \ldots, x_{t-1}) + q)$$

  - Pj,qj= Output word embedding, g = NN

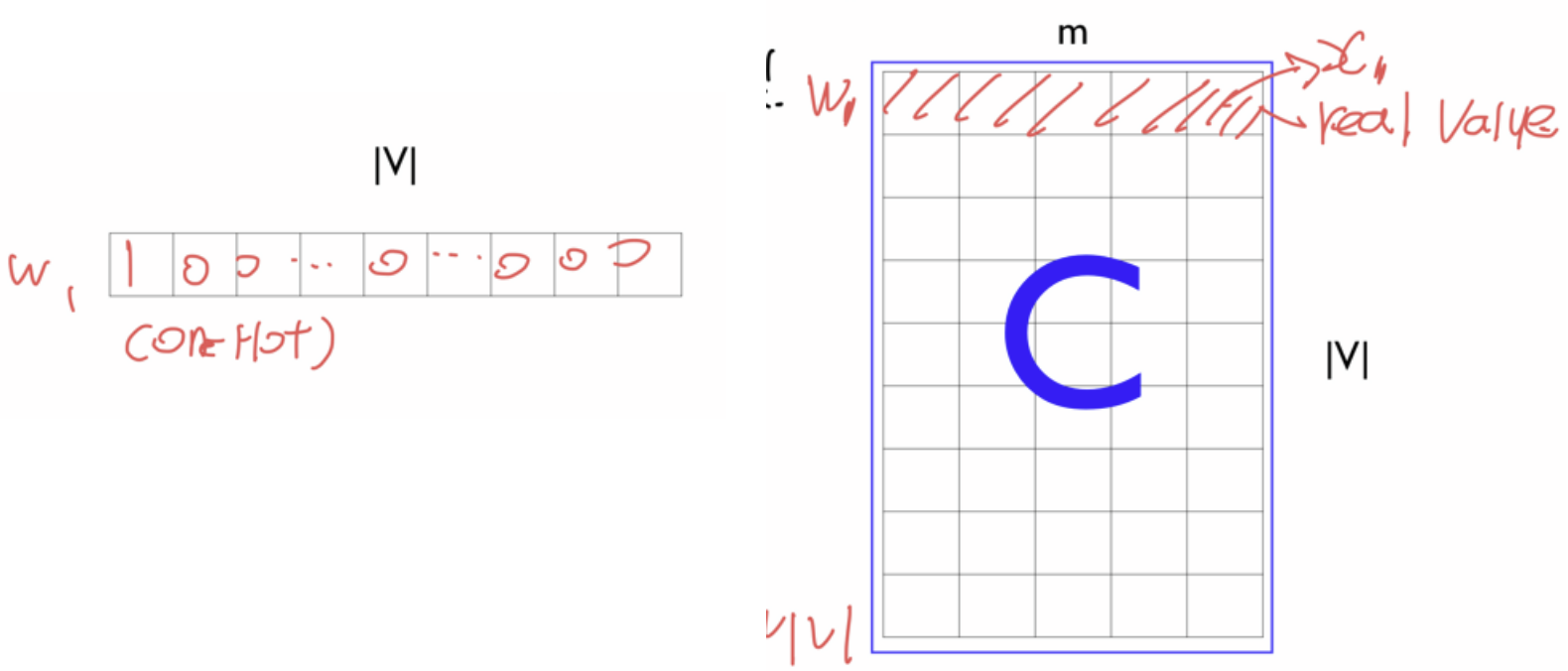
NNLM

- NNLM
  - NNLM의 목적은 높은 우도값을 가지는 $f(w_t, \cdots w_{t-n+1}) = \hat{P}(w_t|w_1^{t-1}),$ 모델을 만드는 것이다.
    - ✓ W는 windows를 의미한다.
  - 두가지 제약 조건이 존재한다.
    - ✓ $w_1^{t-1}, \sum_{i=1}^{|V|} f(i, w_{t-1} \cdots w_{t-n+1}) = 1$ 어떤 조건에서도 이후 단어들이 생성될 확률의 총합은 1이다.
    - ✓ f >= 0(단어가 생성될 확률은 0보다 크거나 같다.)

NNLM

- NNLM

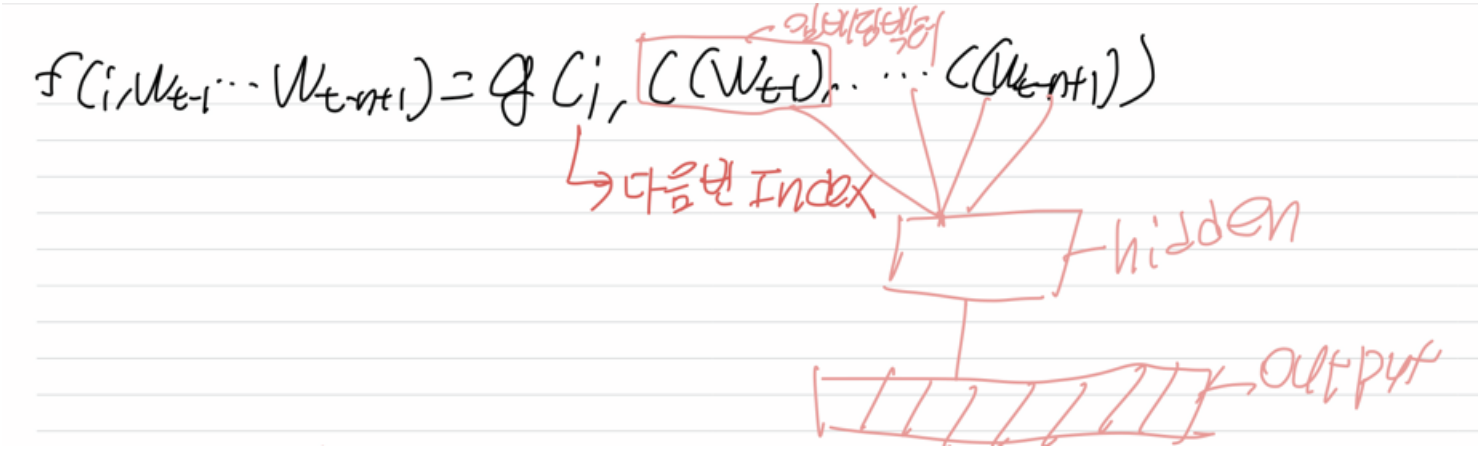
$$f(w_t, \cdots w_{t-n+1}) = \hat{P}(w_t | w_1^{t-1}),$$



NNLM

- NNLM
  - C는 lookup table에서 꺼내쓰는 벡터이다.
  - F(w_t…)에서 w_t는 단어 이다.
  - G는 NN부분이다.
    - ✓ I값을 최대로 하는 것이 목표이다.

$$f(w_t, \cdots w_{t-n+1}) = \hat{P}(w_t | w_1^{t-1}),$$



NNLM

- NNLM

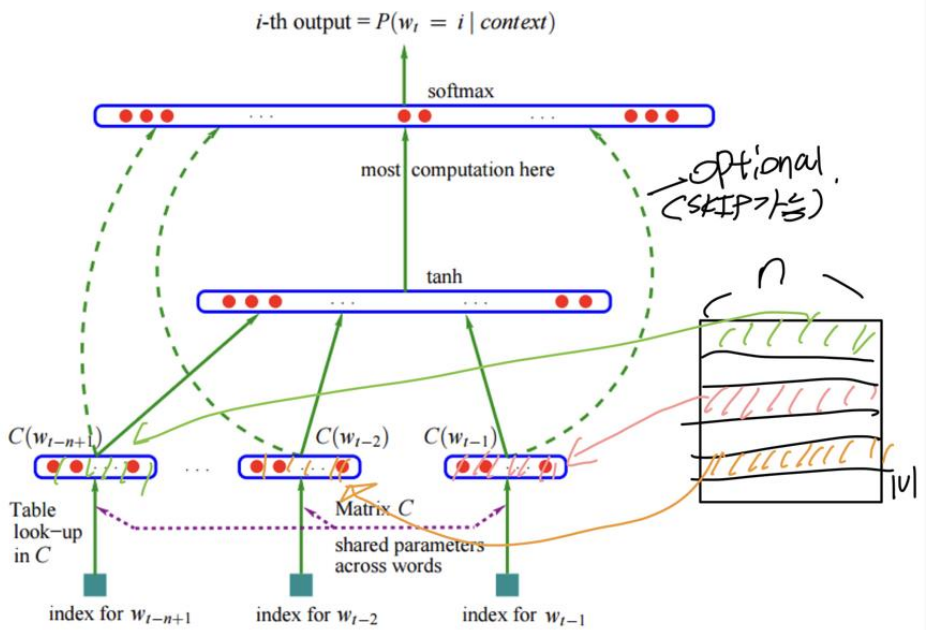
g(준다|너에게, 나의 입술을, 처음으로) = ?

g(지운다|너에게, 나의 입술을, 처음으로) = ?

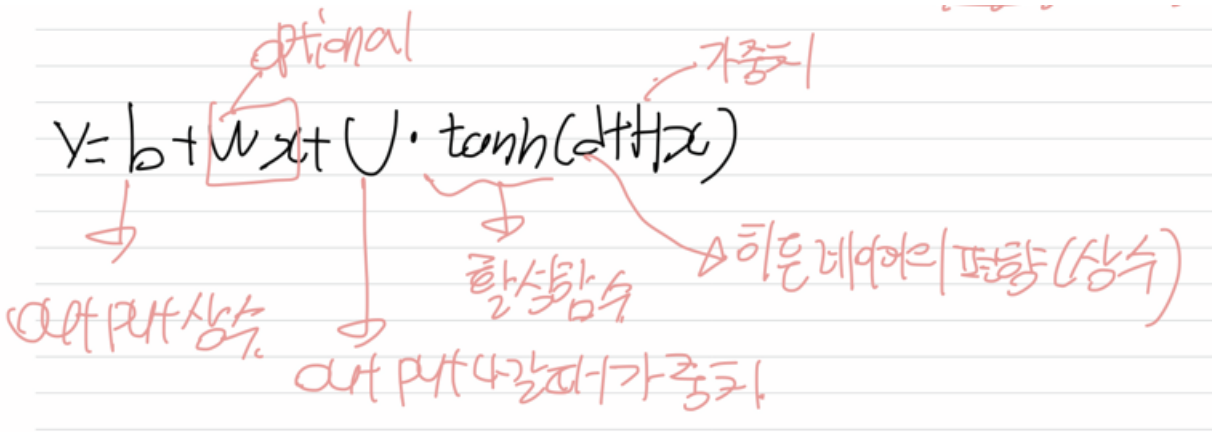
g(맡긴다|너에게, 나의 입술을, 처음으로) = ?

NNLM

- NNLM
  - Hidden layer을 통할 수도, 그냥 지나칠수도 있다.
    - ✓ Wx 가 optional 이라는 것에서 확인 할 수 있다.
  - 바로 밑 식으로 soft-max이고, 합이 1이 되게 한다.
    - ✓ Output lvl의 확률값이 1이 된다.
      - ❖ 제약 조건을 확인하면 된다.



$$\hat{P}(w_t|w_{t-1},\cdots,w_{t-n+1}) = \frac{\exp(y_{w_t})}{\sum_i \exp(y_i)}$$

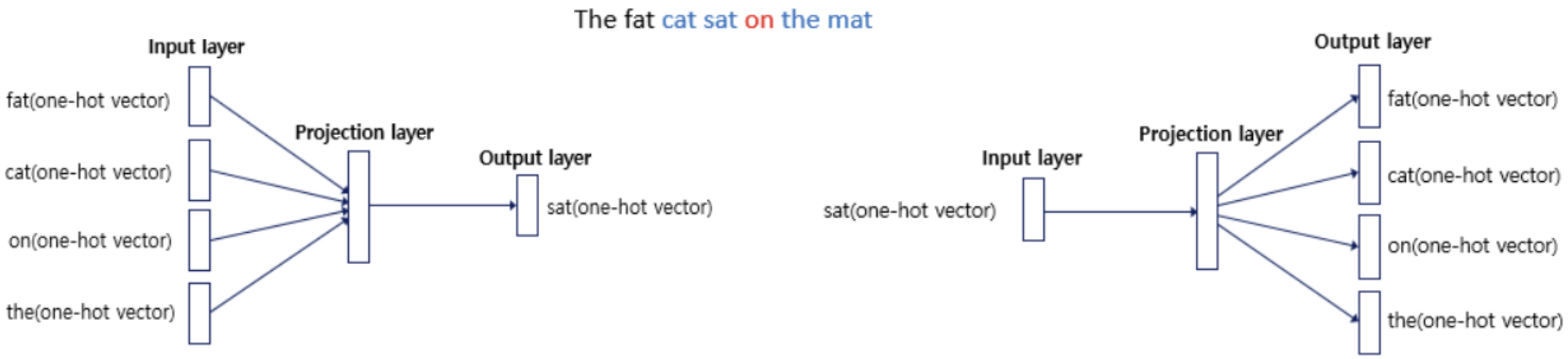


02

Word2Vec

Word2Vec

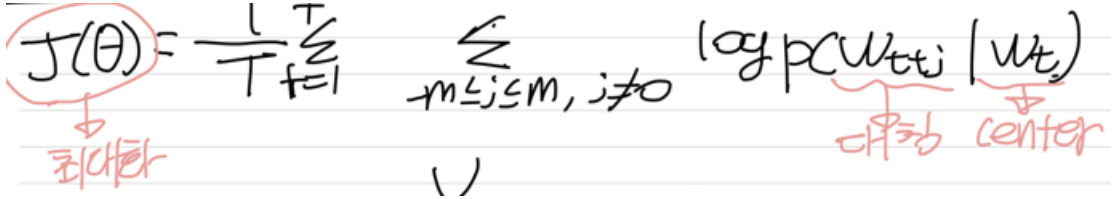
- Word2Vec : Two Architectures
  - CBOW vs Skip-gram
    - ✓ CBOW는 주변에 있는 단어들을 입력으로 중간에 있는 단어를 예측하는 방법이다.
      - ❖ 밑 예는 윈도우 크기가 2일때 예시이다.
    - ✓ Skip-gram은 중간에 있는 단어들을 입력으로 주면 주변 단어를 예측하는 방법이다.



Word2Vec

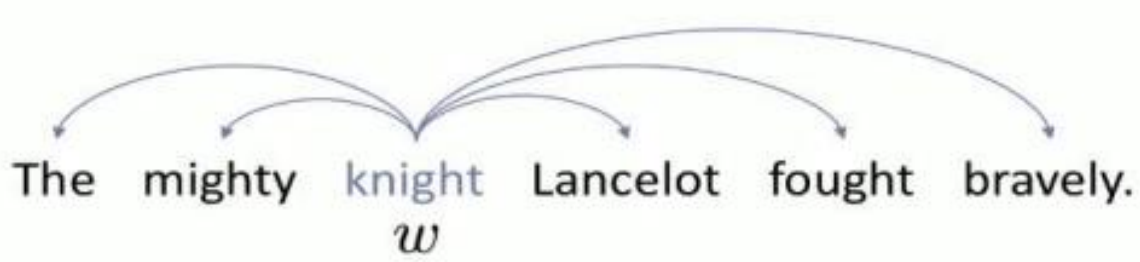
- Word2Vec : Skip-gram
  - 윈도우의 크기 만큼의 주변 단어들을 예측한다.
  - 목적함수는 현재 중앙값으로부터, 주변 단어의 log확률을 최대화 하는 것이다.

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} \sum_{-m \le j \le m, j \neq 0} \log p(w_{t+j}|w_t)$$



Word2Vec

- Word2Vec : Skip-gram
  - 한번에 모든 주변 단어를 다 구해도 되지만, 하나씩 구하더라도 gradient의 값은 동일하다.



knight → The
knight → mighty
knight → Lancelot
knight → fought
knight → bravely.

  - 주어진 단어에 대한 주변 단어의 모델 확률은 밑과 같다.
    - ✓ 분자는 C번째 단어의 내적을 연산한 것과 분모는 모든 단어에 대한 내적을 의미한다.(softmax)

feature for word w: x_w
classifier for word c: v_c

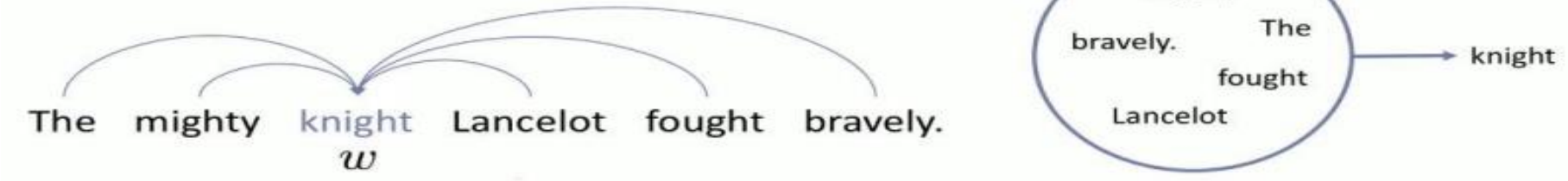
$$p(c|w) = \frac{e^{x_w^\top v_c}}{\sum_{k=1}^{K} e^{x_w^\top v_k}}$$

  - 워드 벡터의 크기는 다음과 같다.

$$x_w \in \mathbb{R}^d$$

Word2Vec

- Word2Vec : CBOW model
  - 주변 단어들을 가지고 단어를 구한다.



  - 주어진 단어에 대한 주변 단어의 모델 확률은 밑과 같다.
    - ✓ 대문자C는 단어의 조합을 의미한다.

feature for context $\mathcal{C}$: $h_{\mathcal{C}}$

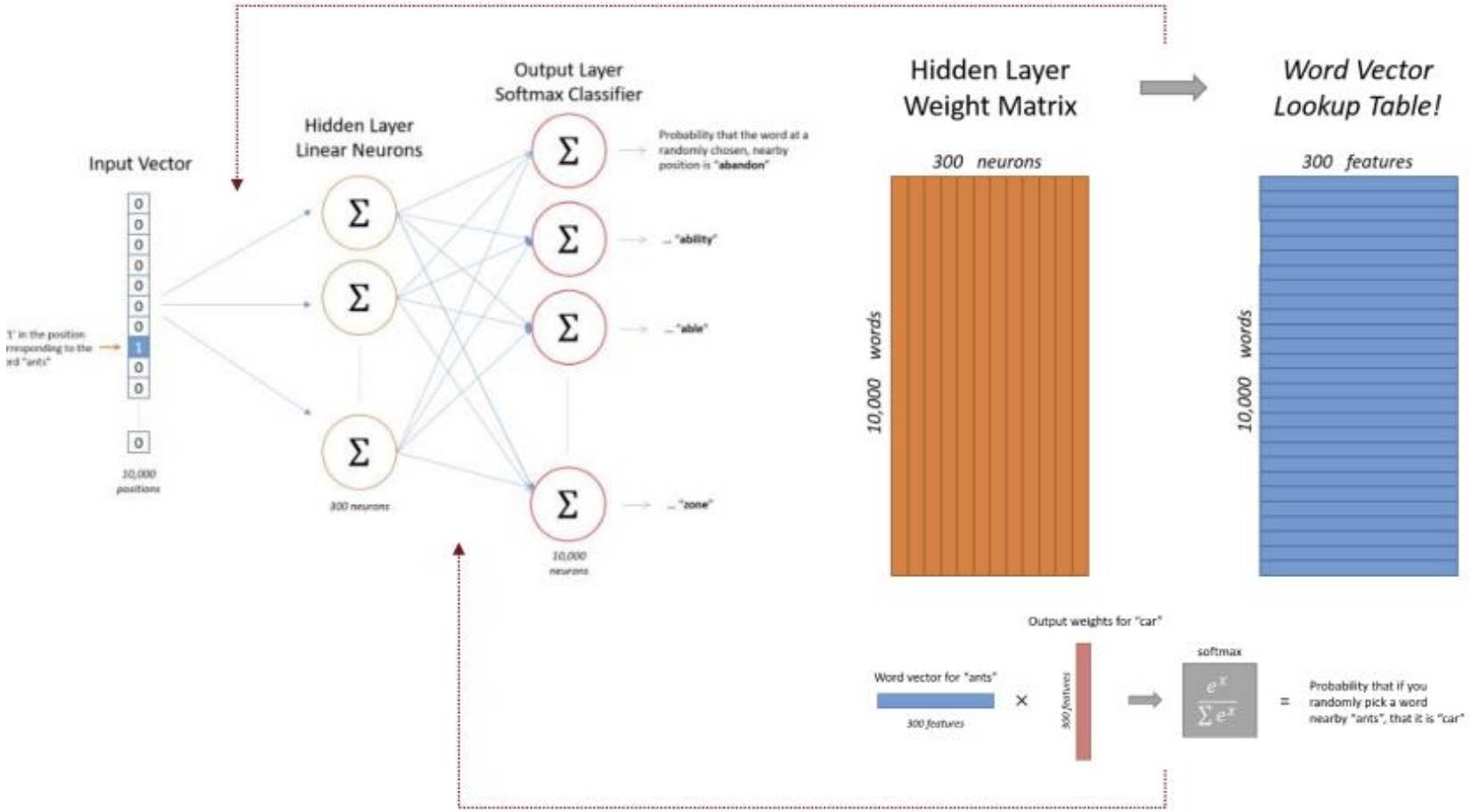
classifier for word w: v_w

$$p(w|\mathcal{C}) = \frac{e^{h_{\mathcal{C}}^{\top} v_w}}{\sum_{k=1}^{K} e^{h_{\mathcal{C}}^{\top} v_k}}$$

$$h_{\mathcal{C}} = \sum_{c \in \mathcal{C}} x_c$$

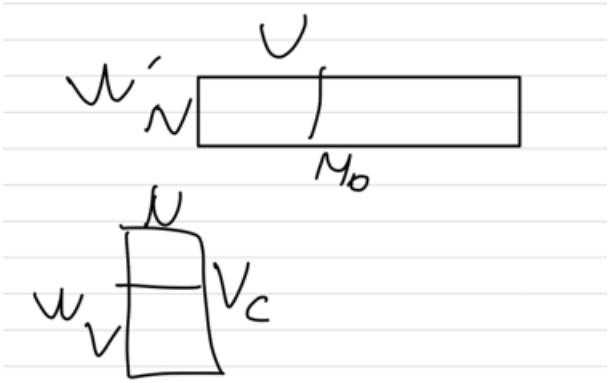
Word2Vec

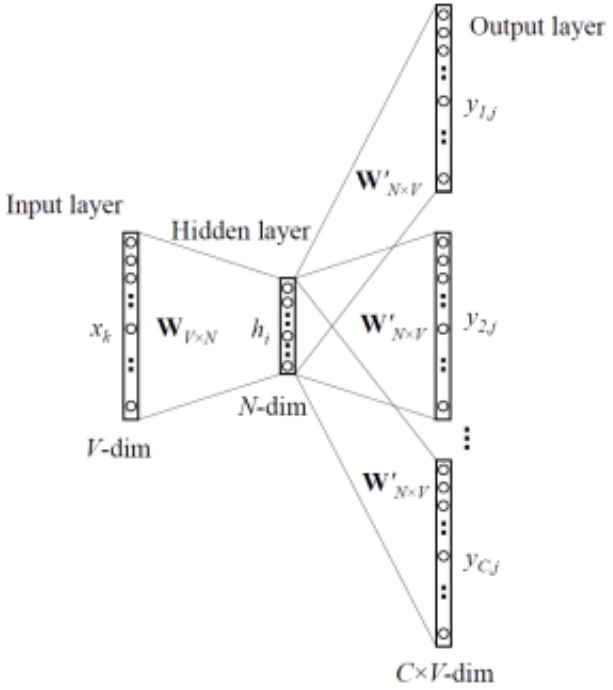
- Word2Vec
  - 밑은 다른 방식으로 표현한 것을 나타낸 것이다.



Word2Vec

- Word2Vec
  - 간단히 하기위해 $p(w_{t+j}|w_t)$ 대신 $p(o|c) = \frac{exp(u_o^T v_c)}{\sum_{w=1}^{W} exp(u_w^T v_c)}$ 을 넣어본다.
    - ✓ O = output, c = center,
    - ✓ W_{t+j} = context , W_t = center
  - 모든 단어는 두개의 벡터를 가진다.
    - ✓ V는 W의 행, U는 W프라임의 열이다.
      - ❖ W프라임과 W^t는 다르지만 편의상 같다고 가정한다.





Word2Vec

- Word2Vec
  - 경사상승법으로 파라미터를 학습한다.

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

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

$$\frac{\partial}{\partial v_c}\log p(o|c) = \frac{\partial}{\partial v_c}\log\frac{exp(u_o^T v_c)}{\sum_{w=1}^{W} exp(u_w^T v_c)}$$

$$= \underbrace{\frac{\partial}{\partial v_c} u_o^T v_c}_{A} \underbrace{- \frac{\partial}{\partial v_c}\log\sum_{w=1}^{W} exp(u_w^T v_c)}_{B}$$

Word2Vec

- Word2Vec
  - 경사상승법으로 파라미터를 학습한다.

✓ For chunk A

$$\frac{\partial}{\partial v_c} u_o^T v_c = u_o$$

✓ For chunk B

$$-\frac{\partial}{\partial v_c} \log \sum_{w=1}^{W} exp(u_w^T v_c)$$

$$= -\frac{1}{\sum_{w=1}^{W} exp(u_w^T v_c)} \cdot \left(\sum_{w=1}^{W} exp(u_w^T v_c) \cdot u_w \right)$$

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

Word2Vec

- Word2Vec
  - 경사상승법으로 파라미터를 학습한다.

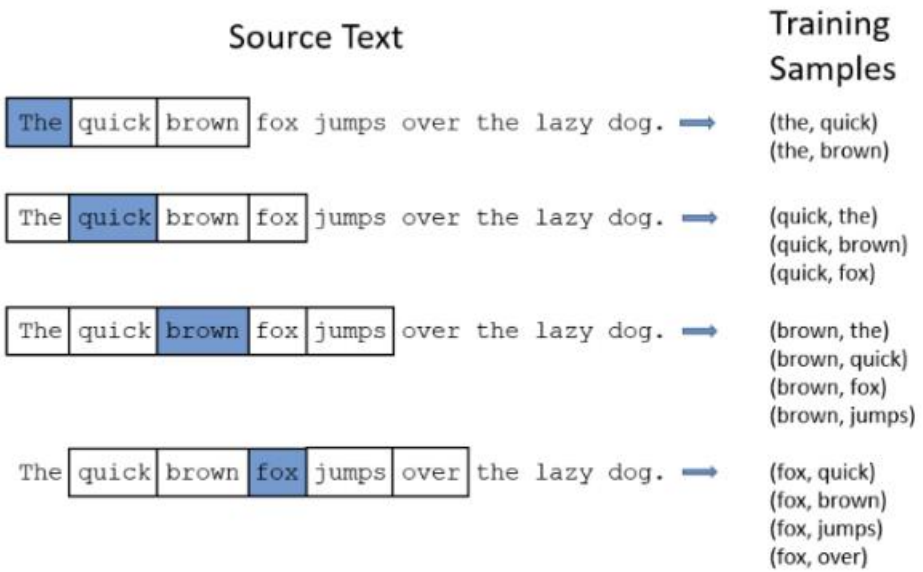
$$\frac{\partial}{\partial v_c}\log p(o|c) = u_o - \sum_{w=1}^{W} P(w|c) \cdot u_w$$

  - 가중치 벡터를 업데이트 하는 식이다.

$$v_c(t+1) = v_c(t) + \alpha\left(u_o - \sum_{w=1}^{W} P(w|c) \cdot u_w\right)$$

Word2Vec

- Word2Vec
  - 꼭 한번에 여러 단어를 구할 필요 없이 하나씩 구해도 된다.
    - ✓ Gradient는 각각 계산하나, 하나씩 하나 같기 때문이다.



Word2Vec

- Word2Vec
  - 가중치가 크다.(2XVXN)
    - ✓ 자주 사용되는 두개의 단어나 구는 하나의 단어로 취급한다.
      - ❖ Ex) machine learning..
    - ✓ 빈번 출현 단어에 대해 Subsampling을 한다.
      - ❖ 코퍼스 등장 확률이 높을수록 지울 확률이 증가 하는 것을 볼 수 있다.

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

← Threshold (10^{-5})

← Term frequency in the corpus

$$if\ f(w_i) = 10^{-4},\ P(w_i) = 1 - \sqrt{\frac{1}{10}} = 0.6838$$

$$if\ f(w_i) = 10^{-2},\ P(w_i) = 1 - \sqrt{\frac{1}{1000}} = 0.9684$$

Word2Vec

- Word2Vec
  - 가중치가 크다.(2XVXN)
    - ✓ Negative sampling을 활용한다.
      - ❖ 가중치 업데이트에 모든 단어를 사용하지 않고 몇 개의 예만 사용한다.

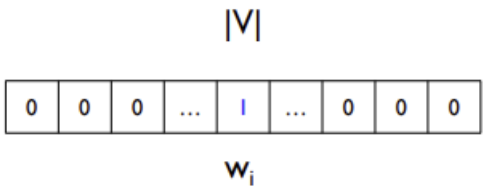
$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} \sum_{-m \le j \le m, j \neq 0} \log p(w_{t+j}|w_t) = \frac{1}{T}\sum_{t=1}^{T} J_t(\theta)$$

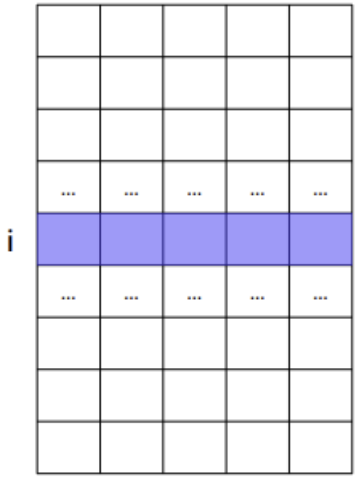
$$J_t(\theta) = \log\sigma(u_o^T v_c) + \sum_{i=1}^{k} E_{i \sim P(w)}\left[\log\sigma(-u_i^T v_c)\right]$$

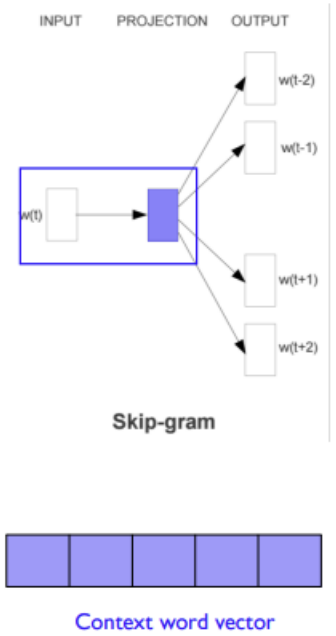
$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{n}\left(f(w_i)^{3/4}\right)}$$

Word2Vec

- Word2Vec
  - 가중치가 크다.(2XVXN)
    - ✓ Negative sampling을 활용한다.
      - ❖ 가중치 업데이트에 모든 단어를 사용하지 않고 몇 개의 예만 사용한다.

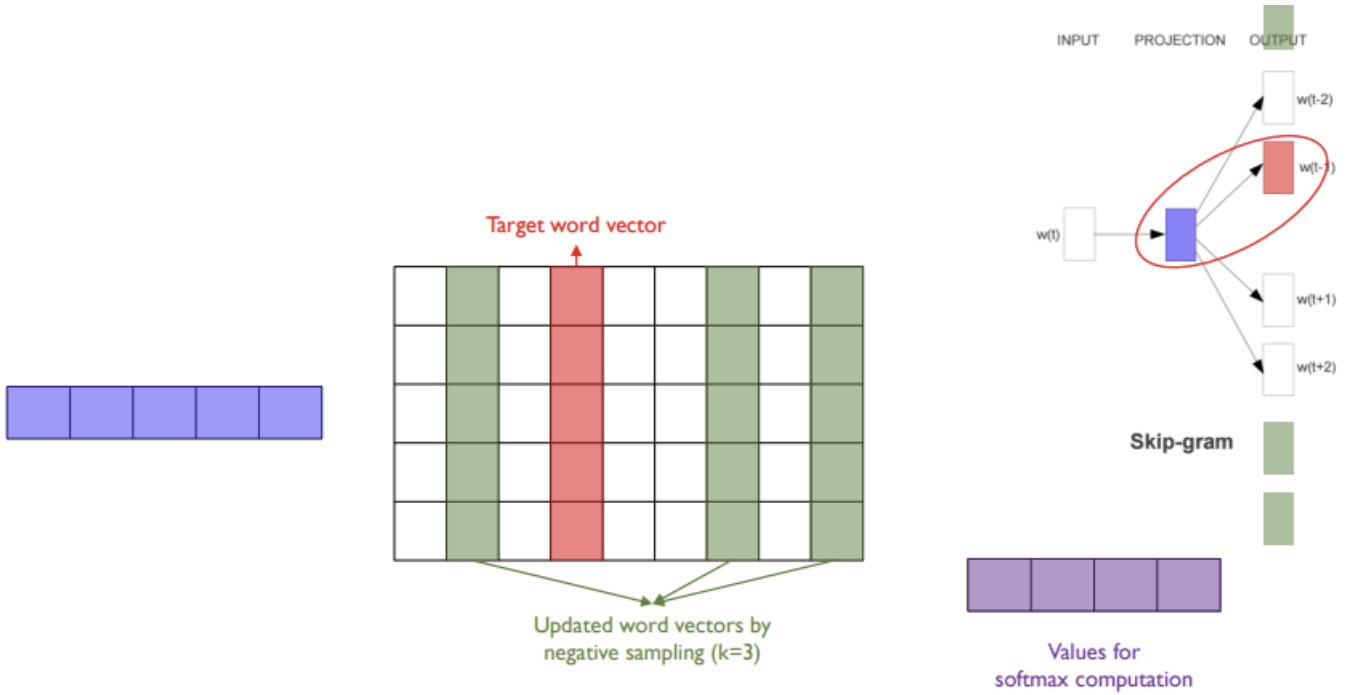






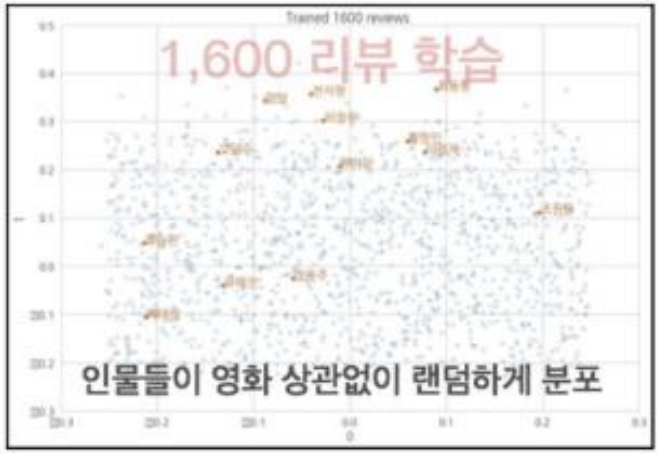
Word2Vec

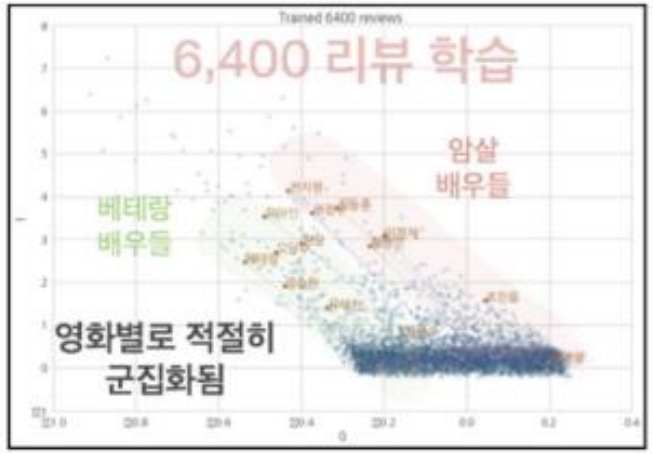
- Word2Vec
  - 가중치가 크다.(2XVXN)
    - ✓ Negative sampling을 활용한다.
      - ❖ 가중치 업데이트에 모든 단어를 사용하지 않고 몇 개의 예만 사용한다.

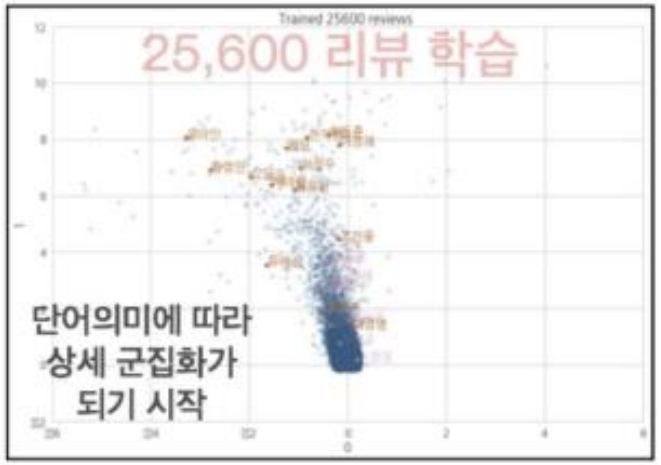


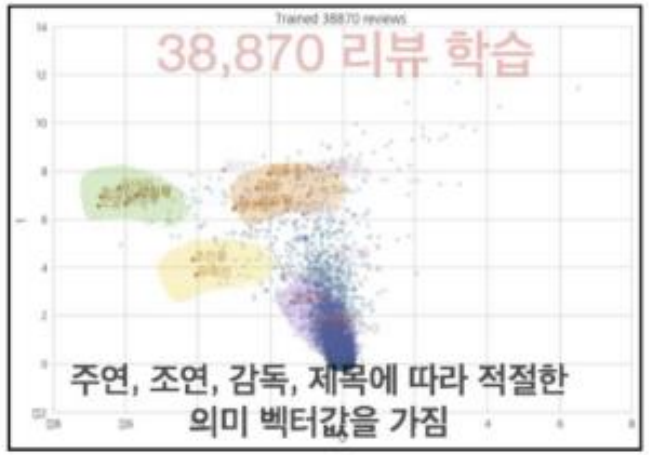
Word2Vec

- Word2Vec









.