Word2Vec

NLP Study

2018.04.25

Q. Word2Vec이란?

단어를 벡터로 표현하는 방법...

책상	(1,0,0,0,0)
바다	(0,1,0,0,0)
피자	(0,0,1,0,0)
호수	(0,0,0,1,0)
들판	(0,0,0,0,1)

Q. Word2Vec이란?

단어를 문맥적 의미를 내포하는 벡터로 표현하는 방법

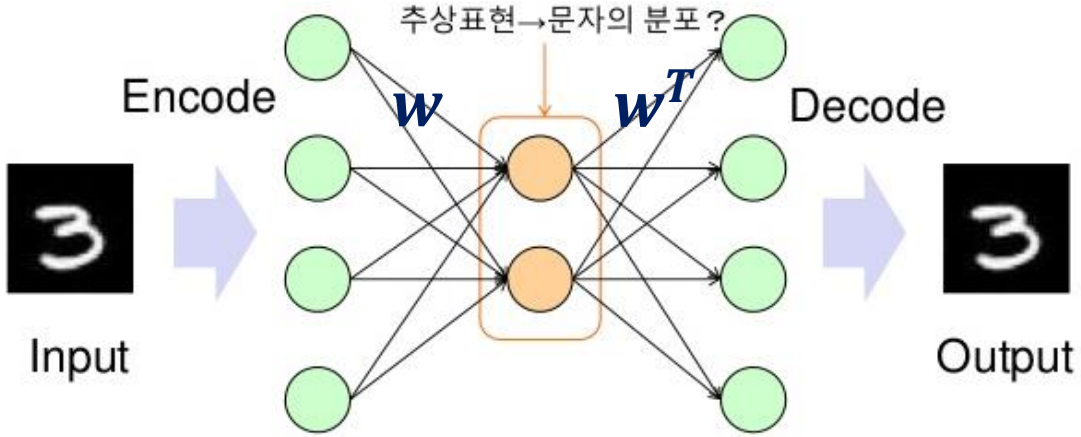
책상	(1,0,0,0,0)	(0.12, 0.23, 0.56)
바다	(0,1,0,0,0)	(0,38, 0.77, 0.93)
피자	(0,0,1,0,0)	(0.88, 0.17, 0.28)
호수	(0,0,0,1,0)	(0.24, 0.72, 0.67)
들판	(0,0,0,0,1)	(0.31, 0.26, 0.89)

Q. 어떻게 단어를 문맥적 의미를 가진 벡터로 표현할 수 있는가?

- 간단한 NN 모형을 이용
- Auto-encoder와 유사함

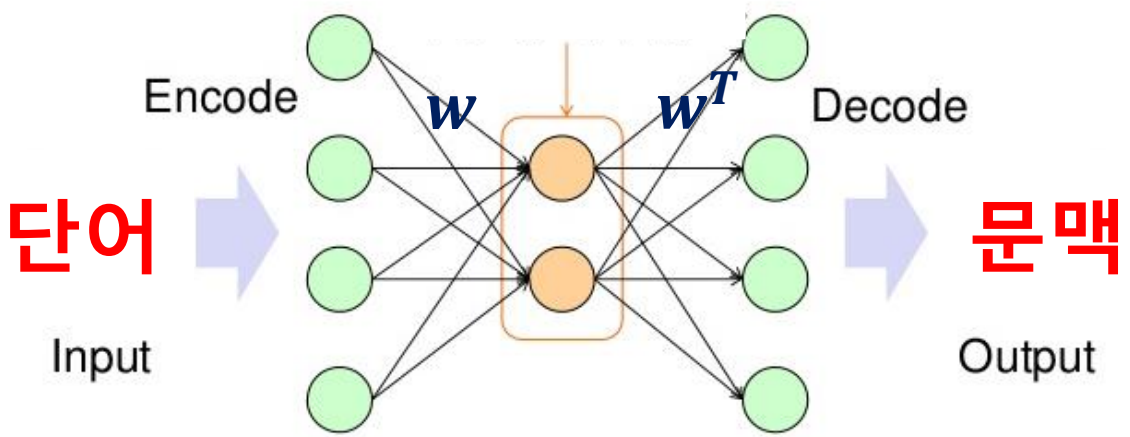
Q. 어떻게 단어를 문맥적 의미를 가진 벡터로 표현할 수 있는가?

Auto-Encoder
Input = output



- AutoEncoder에 의한 이미지의 압축·재구성
- 중간층에서 이미지의 추상표현이 획득

Word2Vec
Input = 단어 / output = 문맥



- Word2Vec에 의한 단어의 압축·재구성
- 중간층에서 단어의 추상표현(벡터)이 획득

Q. [문맥→단어](CBOW) VS [단어→문맥](SG)

1) Continuous bag of words (CBOW)

- 주변단어(문맥)으로 중심단어를 예측

2) Skip-Gram (SG)

- 중심단어로 주변단어(문맥)을 예측
- 성능이 더 좋다
- 일반적으로 사용한다

Q. Skip-Gram이 성능이 더 좋은 이유는?

The boy went to the bank

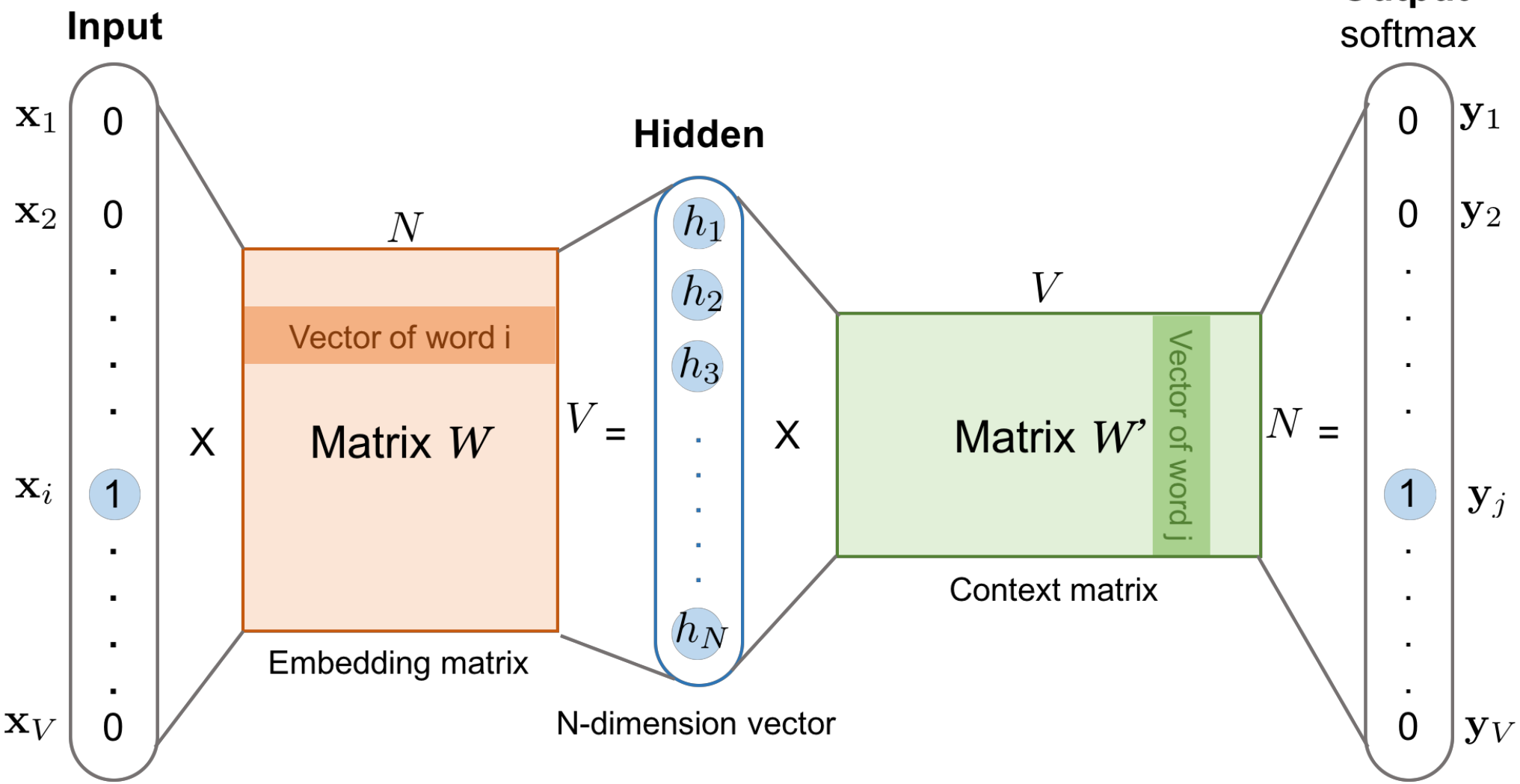
Window size	CBOW	Skip-Gram
1	{(boy, to), went}	{went, boy}, {went, to}
2	{(the, boy, to, the), went}	{went, the}, {went, boy}, {went, to}, {went, the}

Skip-Gram이 학습셋이 더 많다

Q. Word2Vec NN의 구조는?

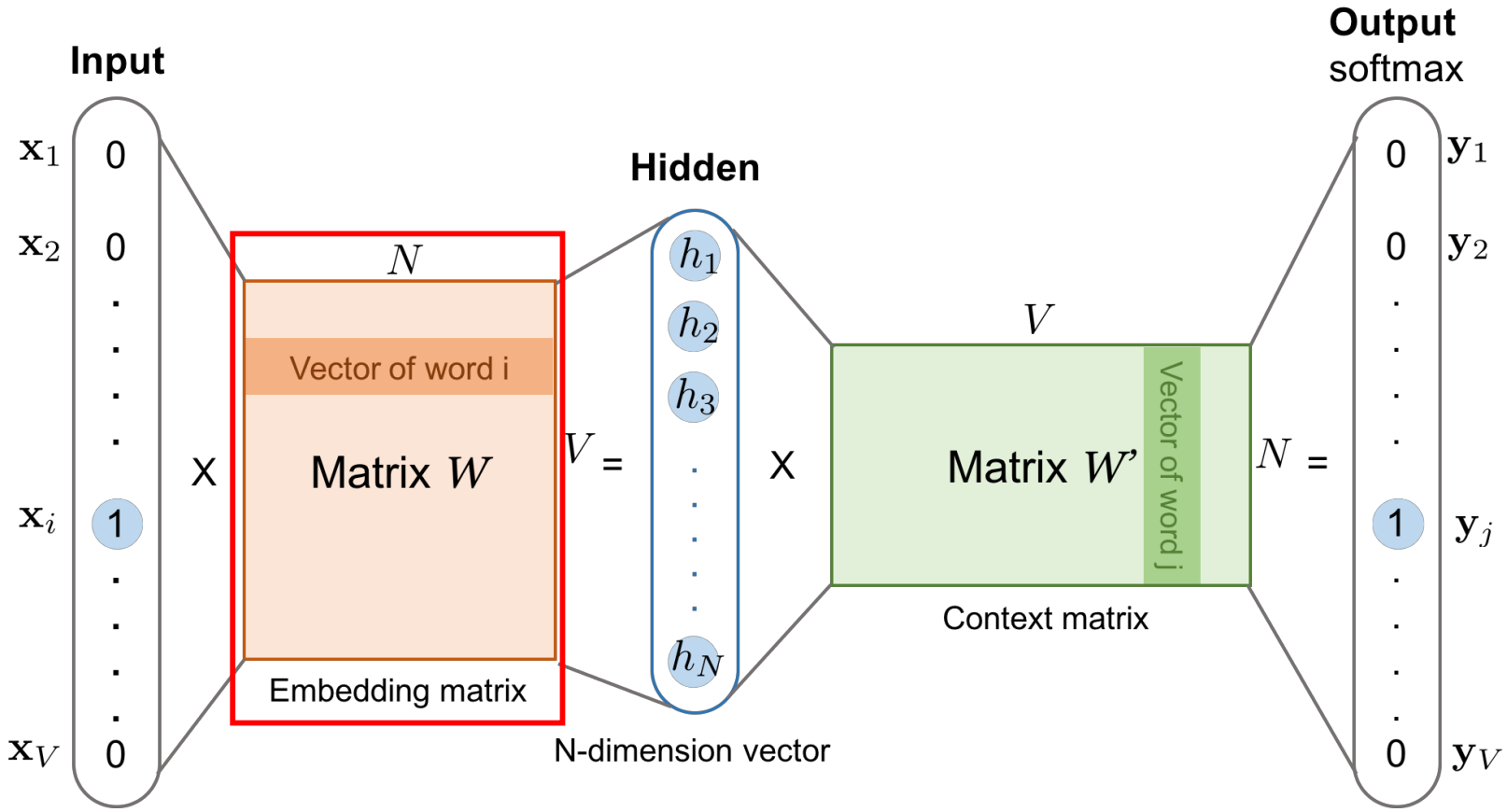
went

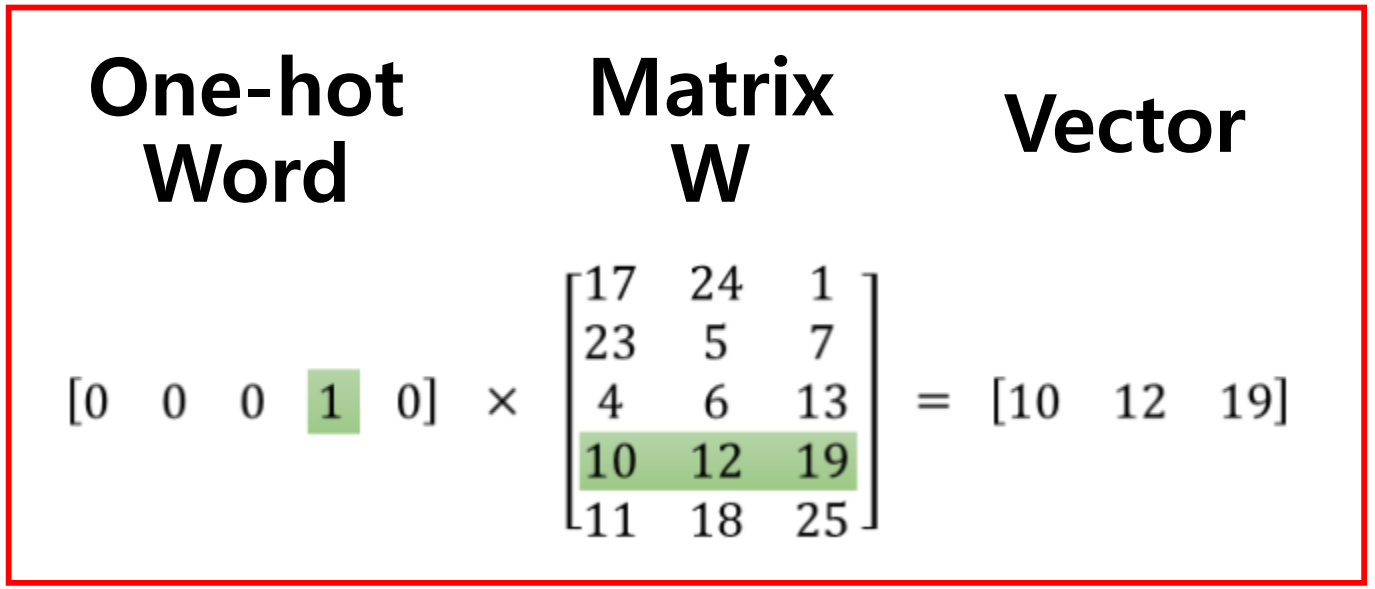
to



Q. Word2Vec의 목표 output은?

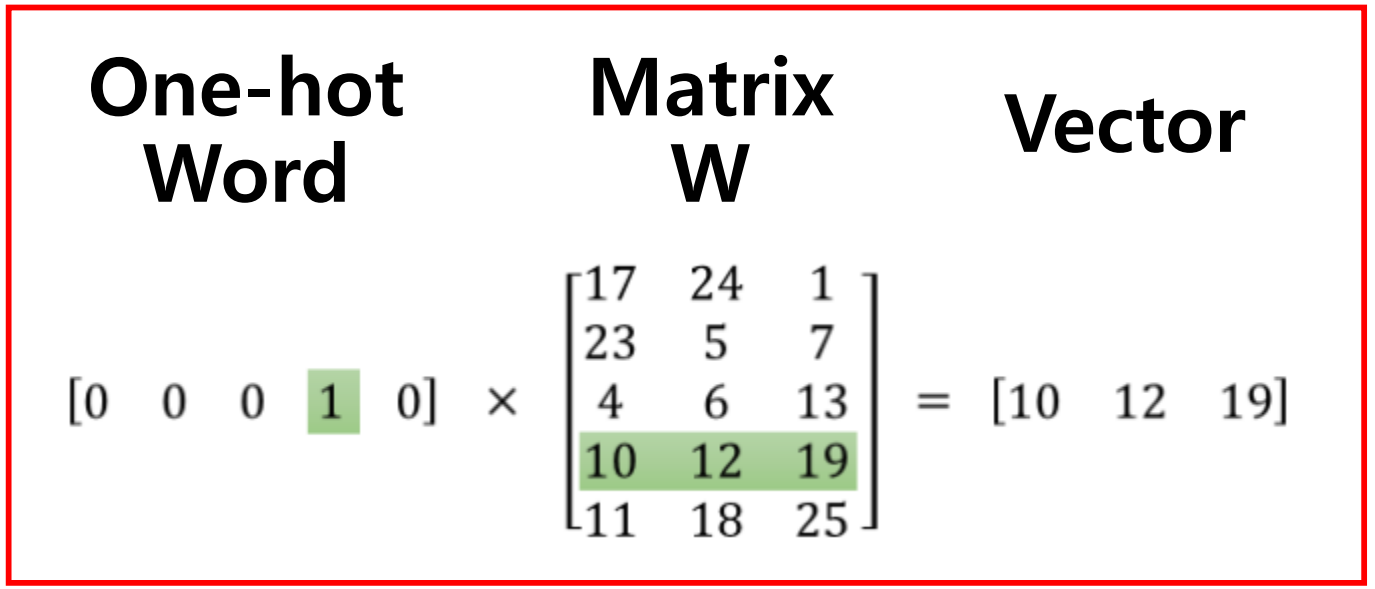
- Matrix W(V x N)의 파라미터 학습
- Input이 one-hot이므로 W가 **임베딩 단어벡터** 모음





은닉층을 계산하는 것

= 가중치에서 단어에 해당하는 **행벡터를 참조(lookup)**하는 것

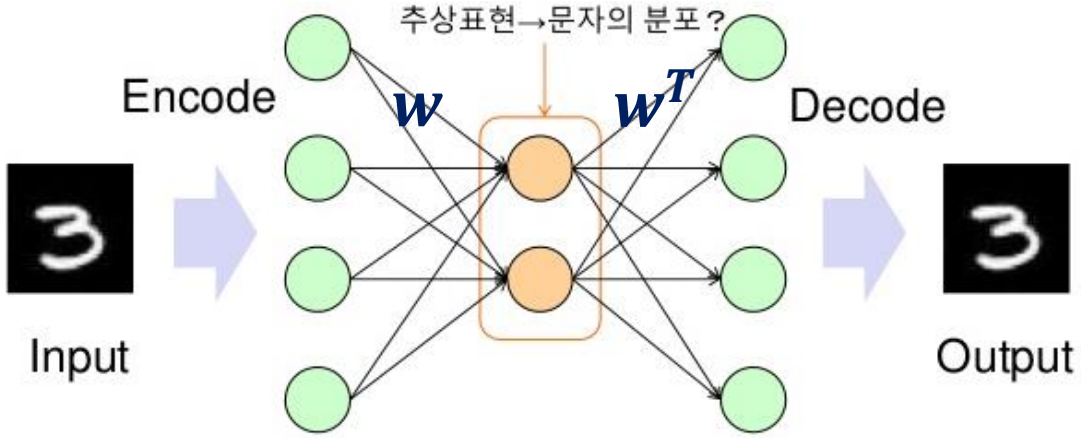


은닉층을 계산하는 것
= 가중치에서 단어에 해당하는 **행벡터를 참조(lookup)**하는 것

일반적인 NN의 목표 (parameter 학습) =
단어를 Word vector로 변환

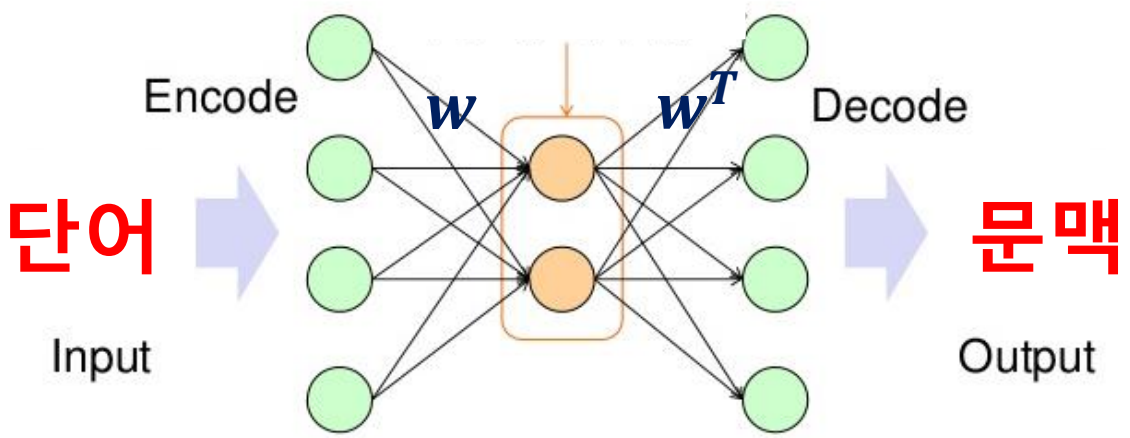
중간층 = W에서 해당하는 단어벡터를 lookup한 것

Auto-Encoder
Input = output



- AutoEncoder에 의한 이미지의 압축·재구성
- 중간층에서 이미지의 추상표현이 획득

Word2Vec
Input = 단어 / output = 문맥



- Word2Vec에 의한 단어의 압축·재구성
- 중간층에서 단어의 추상표현(벡터)이 획득

Q. Parameter를 학습하는 방법은?

- 적정한 Loss function을 설정하는 것
- 단어를 입력하여 문맥이 출력될 확률을 높이면 좋음

$$J'(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} p\left(w_{t+j} \middle| w_t ; \theta\right)$$

중심단어가 주어졌을 때 주변단어(문맥)이 등장할 확률

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

내적 : 코사인유사도

Q. Parameter를 학습하는 방법은?

- 적정한 Loss function을 설정하는 것
- 단어를 입력하여 문맥이 출력될 확률을 높이면 좋음

$$J'(\theta) = \prod_{t=1}^{T} \prod_{-m \le j \le m} p\left(w_{t+j} \middle| w_t ; \theta\right)$$

중심단어가 주어졌을 때 주변단어(문맥)이 등장할 확률

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

$j \neq 0$

내적 : 코사인유사도

윈도우 크기보다 가까운 단어와 문맥의 유사도는 높이면서,
윈도의 크기보다 먼 단어와 문맥의 유사도는 낮춤

Q. Parameter를 업데이트하는 방법은?

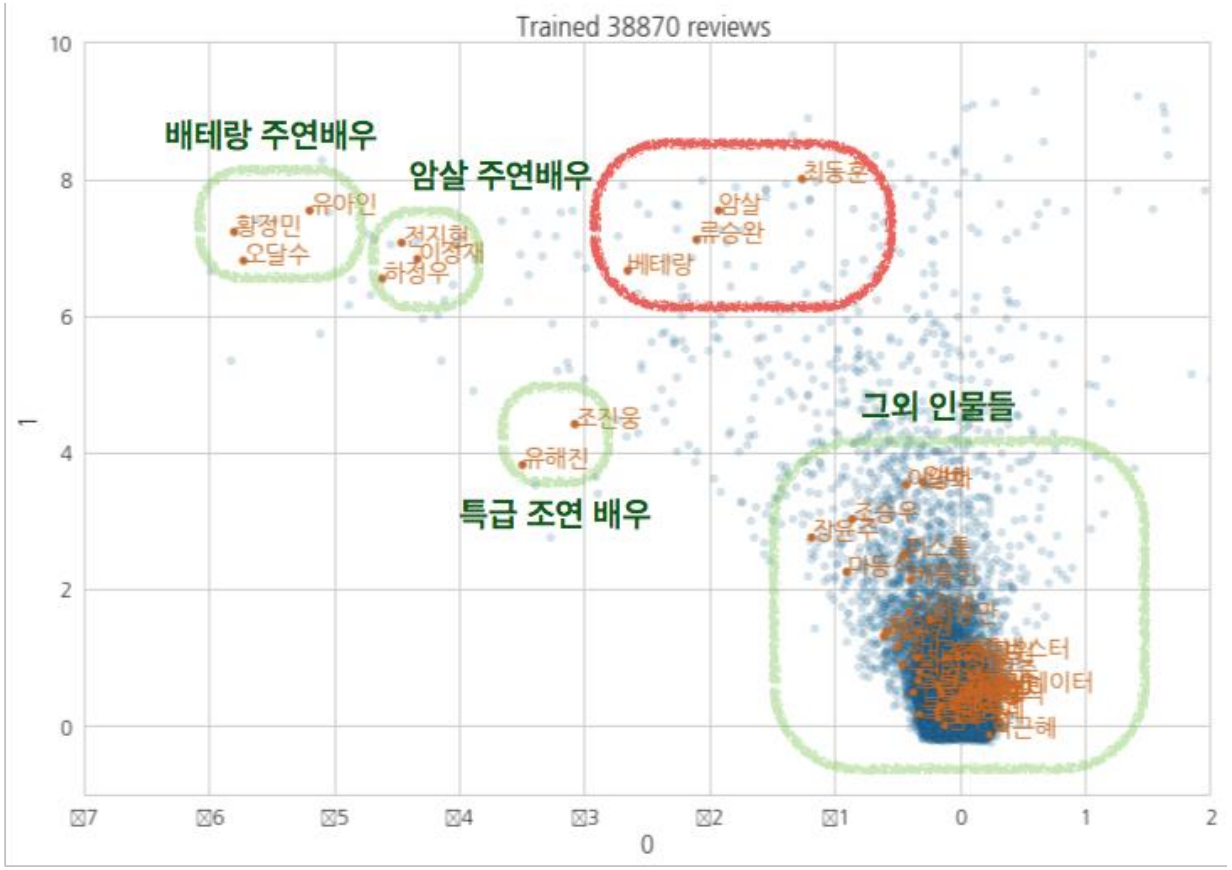
$$\boldsymbol{maximize}\ J'(\theta) \Leftrightarrow \boldsymbol{maximize}\ \frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m\\ j=0}} \log p(w_{t+j}|w_t)$$

$$v_c^{t+1} = v_c^t + \alpha\left(u_o - \sum_{x=1}^{v} p(x|c)u_x\right)$$

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

Softmax 형태

Q. 문맥적 의미를 학습한 단어벡터란?



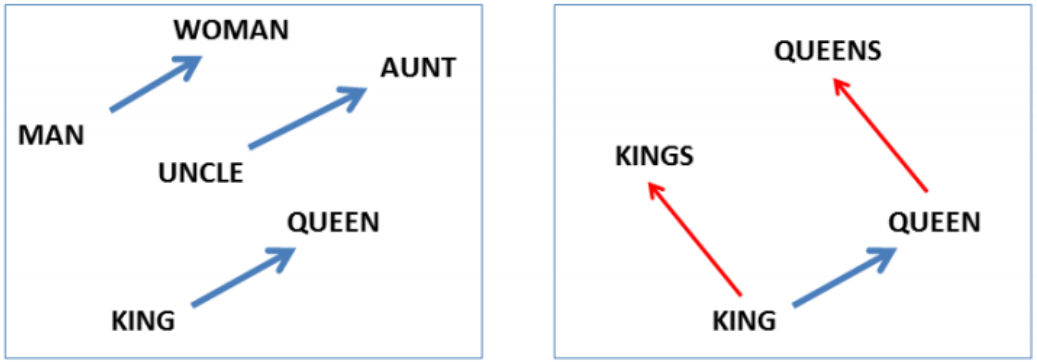
38,870의 리뷰를 학습한 결과
주연배우들의 이름 간 유사도가 높음
(문맥 내의 유사한 의미)

(1) 단어 유사도 측정 가능

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

Q. 문맥적 의미를 학습한 단어벡터란?

(2) 단어 벡터 자체가 단어 간 의미론(Semantic)적인 관계를 포함



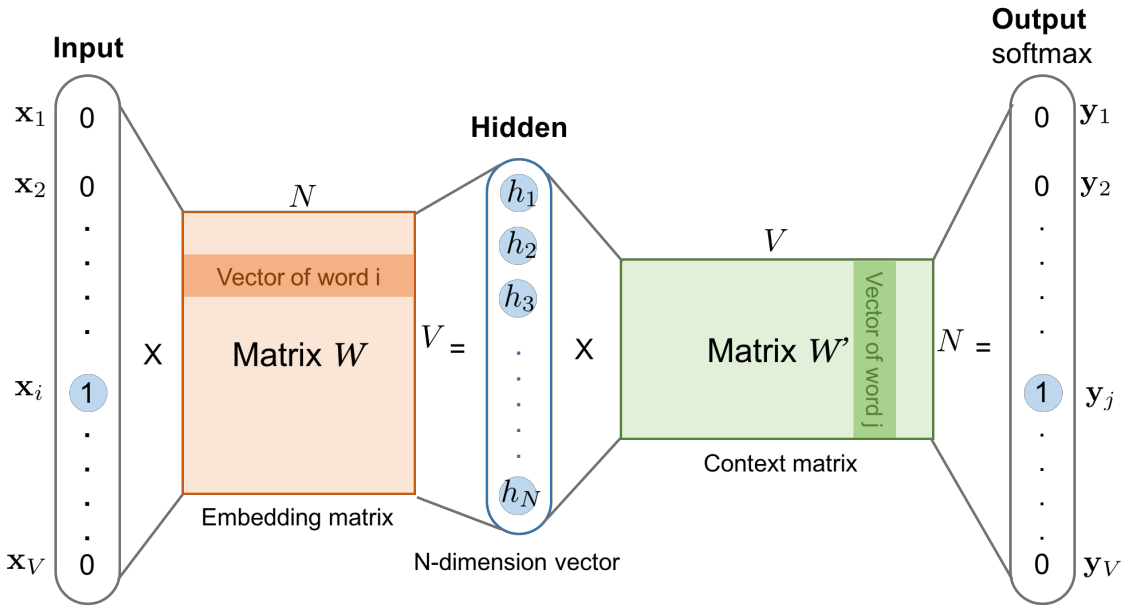
(Mikolov et al., NAACL HLT, 2013)

King – man + woman ≒ queen

Madrid – Spain + France ≒ Paris

Biggest – Big + Small ≒ Smallest

Q. Word2Vec 학습 문제점은?



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

V개 단어를 N차원의 벡터로 변환
: 단어가 늘어날수록 추정해야 할
파라미터가 급증함

Loss function 내의 softmax 계산
: 단어가 늘어날수록 계산량이 폭증함

Q. Word2Vec 학습 문제점의 해결책은?

- 1) 공통적인 단어나 문맥을 하나의 단어로 사용
- 2) Subsampling
- 3) Negative sampling
- 4) Noise-constrative estimation(NCE)
- 5) Hierarchical Softmax

Q. Word2Vec 학습 문제점의 해결책은?

- 1) 공통적인 단어나 문맥을 하나의 단어로 사용
- 2) Subsampling
- 3) Negative sampling
- 4) Noise-constrative estimation(NCE)
- 5) Hierarchical Softmax

"Distributional Representations of Words and Phrases and their Compositionality"

2) Subsampling : 학습셋을 Subsetting

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

$p(w_i) : i^{th}$ 단어를 학습에서 제외시킬 확률
$w_i : i^{th}$ 단어
$t : hyper - parameter\ (0.00001)$
$f(w_i) = w_i$ 등장 비율

특정 단어가 많이 등장하면 단어의 학습횟수를 확률적으로 줄인다
반대로 특정 단어가 거의 등장하지 않으면 반드시 학습한다

학습셋 자체를 줄여 계산량을 감소시킴
빈번하게 등장하지 않는 단어들에 대한 표현 정확도를 높임

3) Negative sampling : softmax 단순하게

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

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

[Positive Sample] : $window$ 내에 등장하는 단어
[Negative Sample] : $window$ 내에 등장하지 않는 단어

$window$ 내에 등장하지 않는 단어는 조금만 뽑는다
어느 정도로 조금만? (small datasets : 5~20 / Large datasets : 2~5)

$p(w_i)$: i^{th} 단어를 Negative Sample로 뽑을 확률 (pre-determined)

3) Negative sampling : softmax 단순하게

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

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

전체 단어(약 10만개)에 대해 분모를 계산하는 것이 아니라
window size + 그 외 일부에 대해서만 계산

학습셋을 줄이는 것이 아니라, 동일한 학습셋에 대해서 계산량을 줄임
빈번하게 등장하는 단어에 대한 표현 정확도를 높임

감사합니다 ☺