Word2Vec

CBOW, Skip-gram

김유리

Word2Vec
Why?

Word Embedding?

: 컴퓨터에게 문장을 학습 시키기 위해 단어를 수치화 시키자

Word2Vec
Why?

기존 방법 → **one-hot encoding**

I watch the movie yesterday.

0 : I	[1, 0, 0, 0, 0] = I
1 : watch	[0, 1, 0, 0, 0] = watch
2 : the	[0, 0, 1, 0, 0] = the
3 : movie	[0, 0, 0, 1, 0] = movie
4 : yesterday	[0, 0, 0, 0, 1] = yesterday

Word2Vec
Why?

기존 방법 → **one-hot encoding**

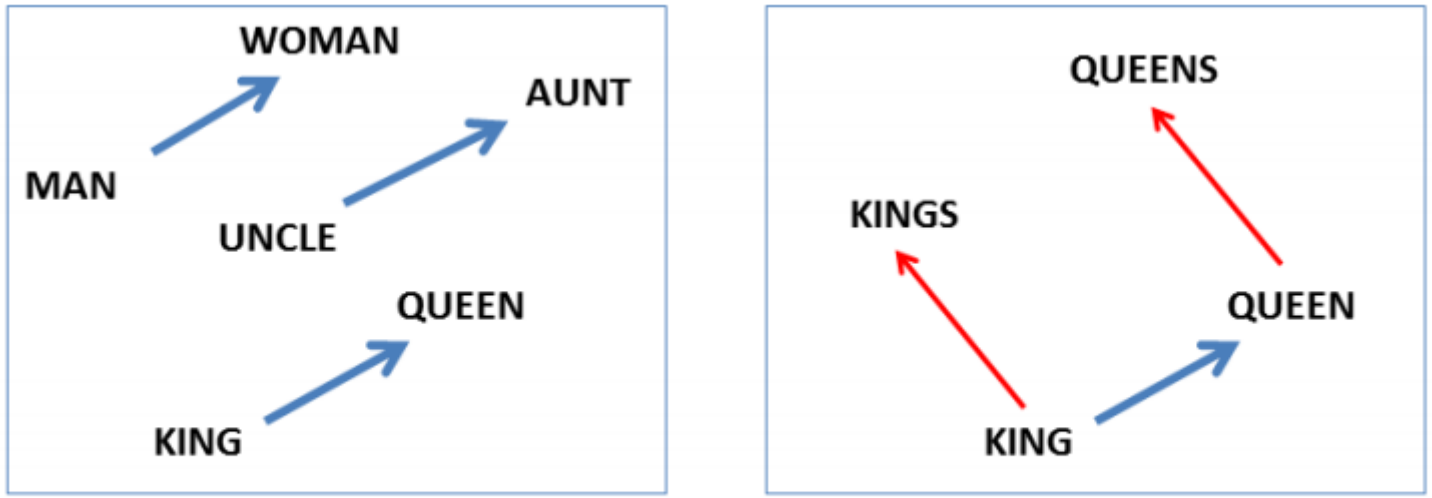
단어를 수치화 하는 데 성공 했지만 문제점 존재

- 단어간 유사도 표현 불가능 (두 벡터 내적 하면 0)
- 문장 데이터가 많아질 수록 사전벡터의 크기가 커짐
- 벡터가 너무 크고 sparse함

Word2Vec
Why?

→ 단어를 유의미한 수치를 가지는 벡터로 만들기 위한 방법 (word embedding) 고안

∴ 단어는 각 벡터로 표현되고, 이 벡터는 단어들간 유사도 또한 나타냄



(Mikolov et al., NAACL HLT, 2013)

(그림4) 단어간 유사도 나타내는 벡터 예시

Word2Vec
Idea

Word2Vec

언어학의 Distributional Hypothesis

→ 비슷한 분포를 가진 단어들은 비슷한 의미를 가진다

→ 같이 등장하는 횟수가 많을 수록 두 단어는 비슷한 의미를 가진다

(CBOW, skip-Gram 두가지 모델 존재)

Word2Vec
CBOW

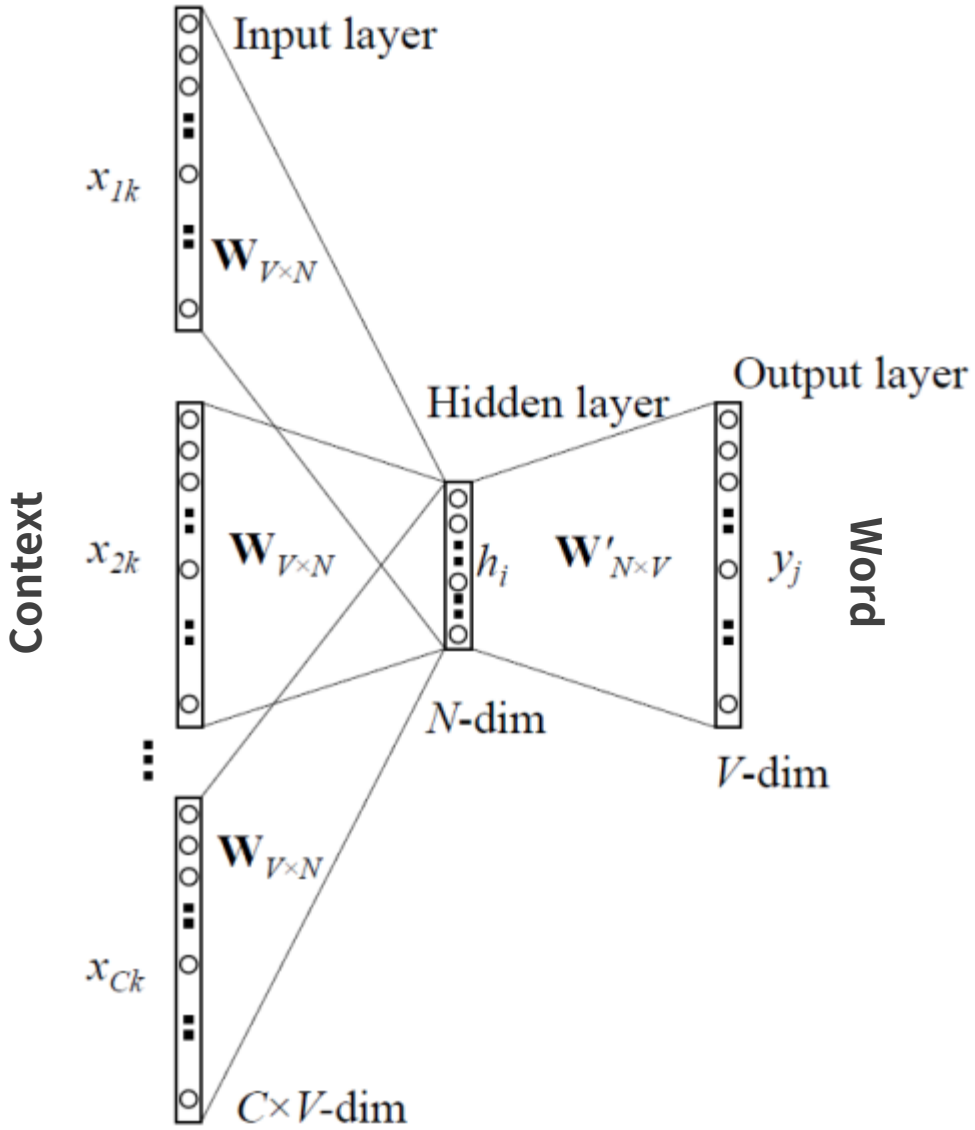
CBOW (Continuous Bag of Words Model)

Idea

주변단어를 통해 주어진 단어가 무엇인지 찾는 것

앞뒤로 c/2개의 단어를 (총c개) 통해 주어진 단어를 예측 함

ex. “아침을 안 먹었더니 __가 너무 고프다”



(그림5) CBOW Architecture

Word2Vec
CBOW

CBOW (Continuous Bag of Words Model)

"I watch the movie yesterday."

Mechanism

① 학습시킬 문장의 모든 단어들을 one-hot encoding

$$x_k = [0, \dots, 0, 1, 0, \dots, 0]$$

① 0 : I — [1, 0, 0, 0, 0] = I
1 : watch — [0, 1, 0, 0, 0] = watch
2 : the — [0, 0, 1, 0, 0] = the
3 : movie — [0, 0, 0, 1, 0] = movie
4 : yesterday — [0, 0, 0, 0, 1] = yesterday

② 하나의 중심단어에 대해 2m개의 단어 벡터를 input 값으로 가짐 (m : window size)

$$(x^{c-m}, x^{c-m+1}, \dots, x^{c-1}, x^{c+1}, \dots, x^{c+m-1}, x^{c+m}) \in \mathbb{R}^{|V|}$$

②

m = 2	중심 단어	주변 단어
I watch the movie yesterday	[1, 0, 0, 0, 0]	[0, 1, 0, 0, 0], [0, 0, 1, 0, 0]
I watch the movie yesterday	[0, 1, 0, 0, 0]	[1, 0, 0, 0, 0], [0, 0, 1, 0, 0], [0, 0, 0, 1, 0]
I watch the movie yesterday	[0, 0, 1, 0, 0]	[1, 0, 0, 0, 0], [0, 1, 0, 0, 0], [0, 0, 0, 1, 0], [0, 0, 0, 0, 1]
I watch the movie yesterday	[0, 0, 0, 1, 0]	[0, 1, 0, 0, 0], [0, 0, 1, 0, 0], [0, 0, 0, 0, 1]
I watch the movie yesterday	[0, 0, 0, 0, 1]	[0, 0, 1, 0, 0], [0, 0, 0, 1, 0]

[표3] window size=2일 때 CBOW를 위한 전체 데이터 셋

Word2Vec
CBOW

CBOW (Continuous Bag of Words Model)

Mechanism

③ one-hot encoding word vector들을 embedded word vector로 변환 (parameter : $W_{V\times N}$)

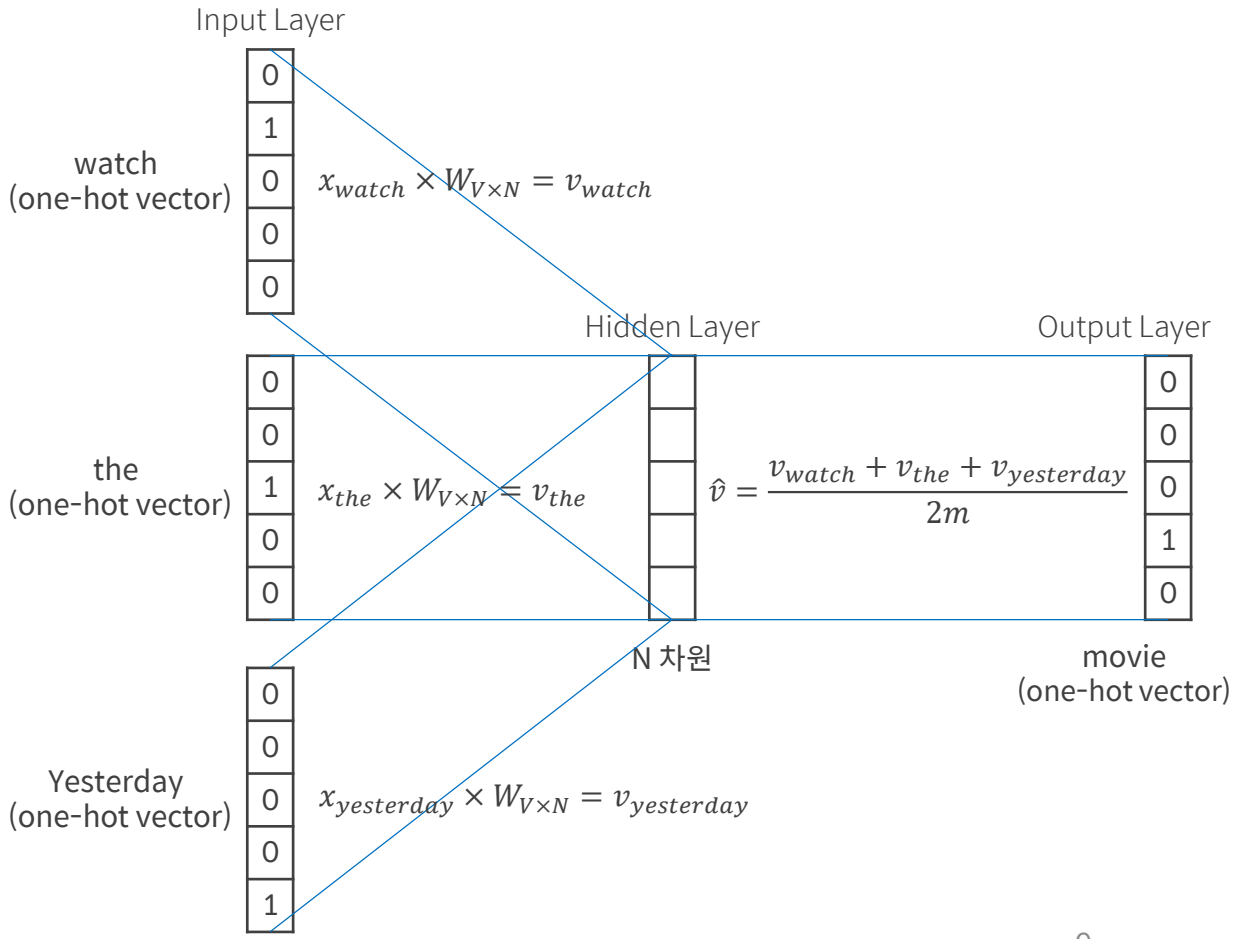
$$(v_{c-m} = Wx^{c-m}, \dots, v_{c+m} = Wx^{c+m}) \in \mathbb{R}^n$$

④ $2m$개 embedded word vector의 평균을 구함 (= Hidden Layer의 값)

$$\hat{v} = \frac{v_{c-m} + v_{c-m+1} + \cdots + v_{c+m}}{2m} \in \mathbb{R}^n$$

"I watch the movie yesterday."

m = 2	중심 단어	주변 단어
I watch the movie yesterday	[0, 0, 0, 1, 0]	[0, 1, 0, 0, 0], [0, 0, 1, 0, 0], [0, 0, 0, 0, 1]



Word2Vec
CBOW

CBOW (Continuous Bag of Words Model)

Mechanism

⑤ output layer로 전달할 score 계산 (가까운 위치의 단어들이 높은 값을 갖도록)

(parameter : $W'_{N\times V}$)

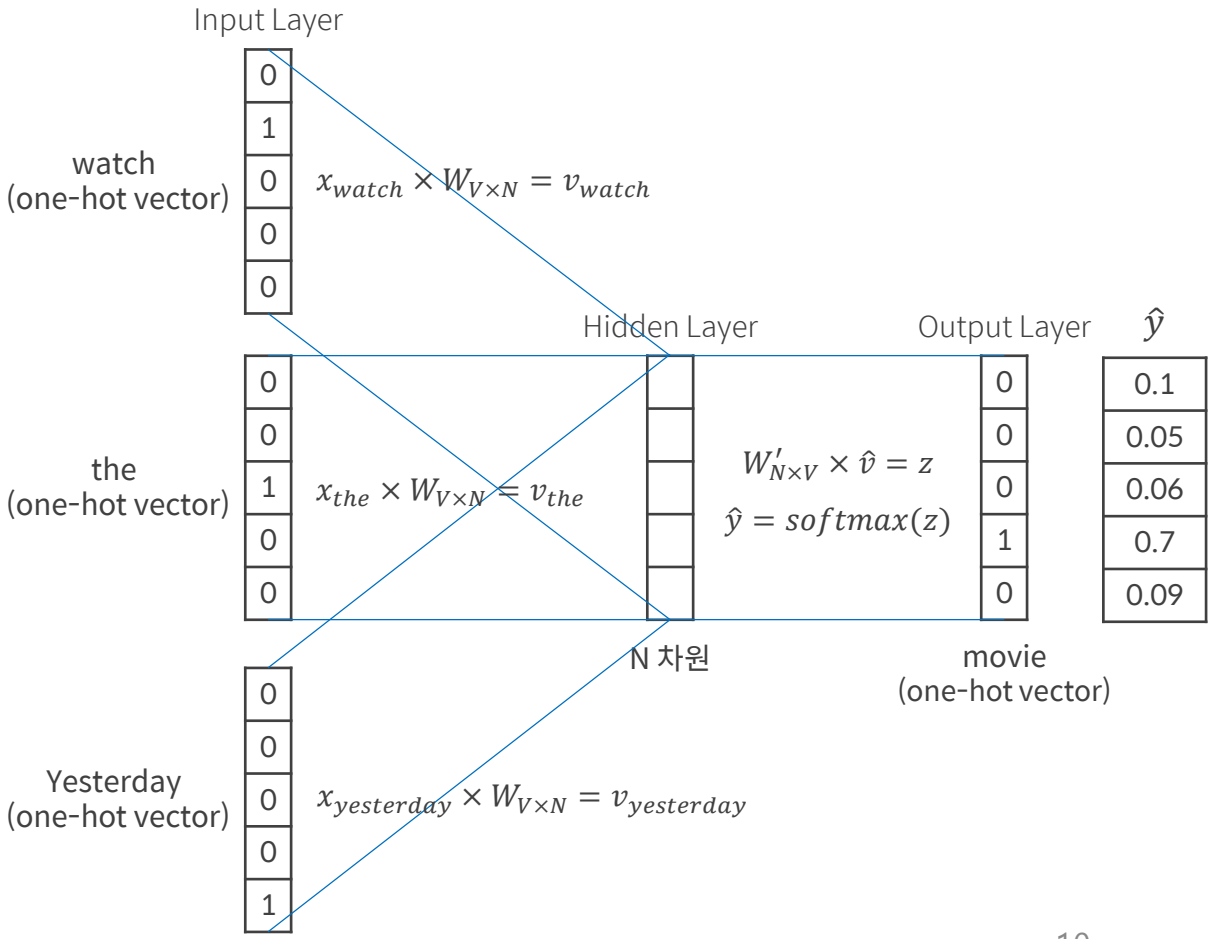
$$z = W'_{N\times V} \times \hat{v} \in \mathbb{R}^{|V|}$$

⑥ score를 확률 값으로 계산

$$\hat{y} = softmax(z) \in \mathbb{R}^{|V|}$$

"I watch the movie yesterday."

m = 2	중심 단어	주변 단어
I watch the movie yesterday	[0, 0, 0, 1, 0]	[0, 1, 0, 0, 0], [0, 0, 1, 0, 0], [0, 0, 0, 0, 1]



Word2Vec
CBOW

CBOW (Continuous Bag of Words Model)

Objective function

(score vector $= \hat{y}, one - hot\ vector = y$)

$$H(\hat{y}, y) = -\sum_{j=1}^{|V|} y_j \log(\hat{y}_j)$$

$$H(\hat{y}, y) = -y_i \log(\hat{y}_i)$$

$$\left(\because \hat{y}_j = one - hot\ vector\right)$$

단어를 정확하게 예측 했다면 $H = 0$

embedding vector : N차원 행렬 W의행이나 W'의 열

확률분포에 대한 식으로 Objective function 표현

$$\mathrm{m}inimize J = -logP(w_c|w_{c-m}, \dots, w_{c+m})$$

$$= -logP(u_c|v)$$

$$= -log\frac{\exp(u_c^T \hat{v})}{\sum_{j=1}^{|V|} \exp(u_j^T \hat{v})}$$

$$= -u_c^{intercal} \hat{v} + log\sum_{j=1}^{|V|} \exp(u_j^T \hat{v})$$

Loss function : cross-entropy

Optimize : SGD

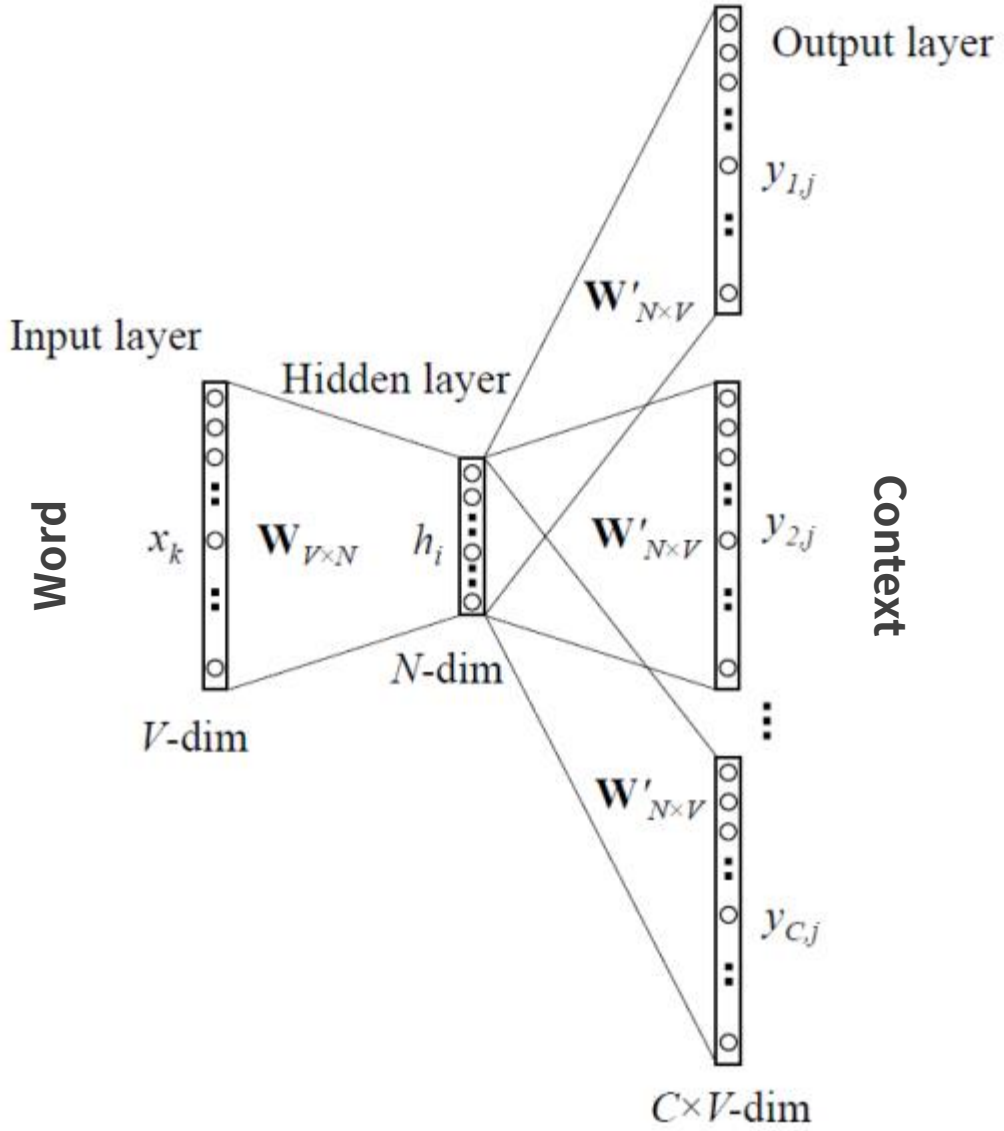
Word2Vec
Skip-Gram

Skip-Gram (Continuous Skip-gram)

Idea

중심단어를 통해 주변 단어가 무엇인지 찾는 것

ex. “_____ 배가_____ ”



(그림5) Skip-Gram Architecture

Word2Vec
Skip-Gram

Skip-Gram (Continuous Skip-gram)

Mechanism

① 중심단어를 one-hot vector(x)로 변환 $x \in \mathbb{R}^{|V|}$

② embedded vector(v_c) 구하기 $v_c = W_x \in \mathbb{R}^n$

③ score vector(z) 구하기 $z = W'v_c$

④ score vector를 확률 값($\hat{y}$)으로 변환 $\hat{y} = softmax(z)$

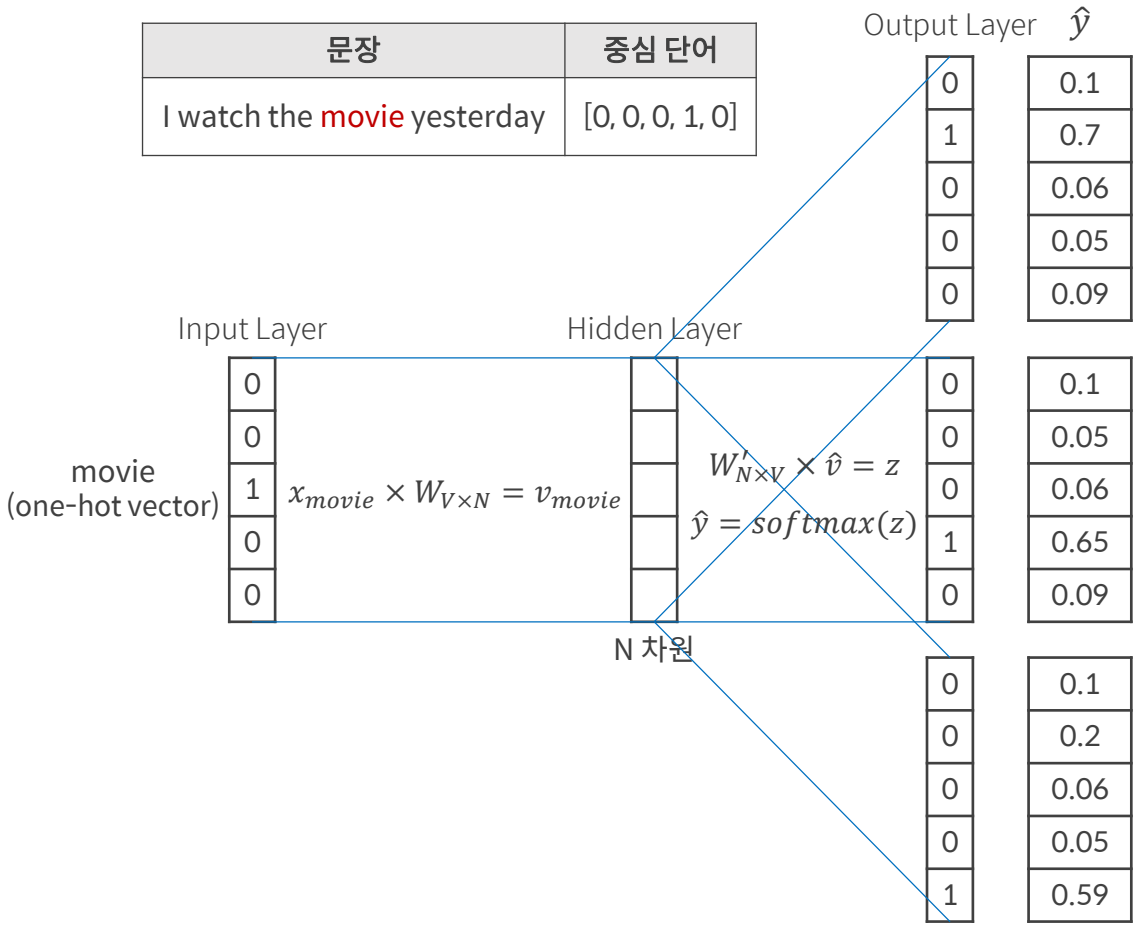
⑤ 구한 확률 값 $2m$개에 대해 각 위치의 정답과 비교

$$\hat{y}_{c-m}, \dots, \hat{y}_{c-1}, \hat{y}_{c+1}, \dots, \hat{y}_{c+m}$$

$$y^{(c-m)}, \dots, y^{(c-1)}, y^{(c+1)}, \dots, y^{(c+m)}$$

"I watch the movie yesterday."

문장	중심 단어
I watch the movie yesterday	[0, 0, 0, 1, 0]



Word2Vec
Skip-Gram

Skip-Gram (Continuous Skip-gram)

Objective function

$$minimize\ J = -\log P(w_{c-m}, \dots, w_{c-1}, w_{c+1}, \dots, w_{c+m}|w_c)$$

$$= -\log \prod_{j=0, j\neq m}^{2m} P(w_{c-m+j}|w_c)$$

$$= -\log \prod_{j=0, j\neq m}^{2m} \frac{\exp\left(u_{k-m+j}^T v_c\right)}{\sum_{k=1}^{|V|} \exp(u_k^T v_c)}$$

$$= -\sum_{j=0, j\neq m}^{2m} u_{k-m+j}^T v_c + 2m log \sum_{k=1}^{|V|} \exp(u_k^T v_c)$$

CBOW 모델 같이 cross-entropy($H()$) 함수로 확률 값 정의

(차이점 : 각 단어에 대해 독립적이라고 가정)

$$J = -\sum_{j=0, j\neq m}^{2m} logP(u_{c-m+j}|v_c)$$

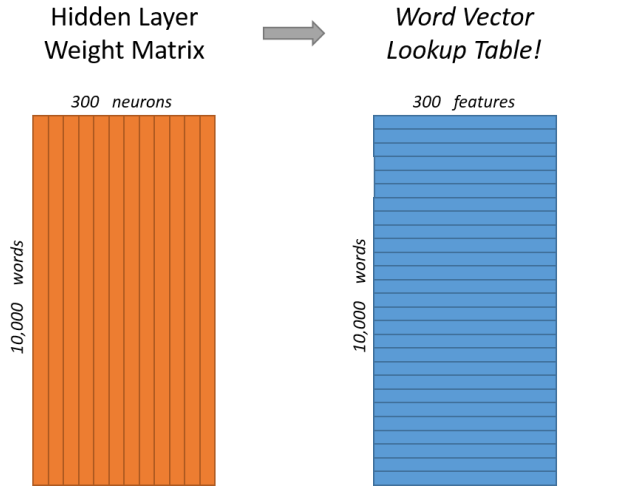
$$= \sum_{j=0, j\neq m}^{2m} H(\hat{y}, y_{c-m+j})$$

Word2Vec
Lookup Table

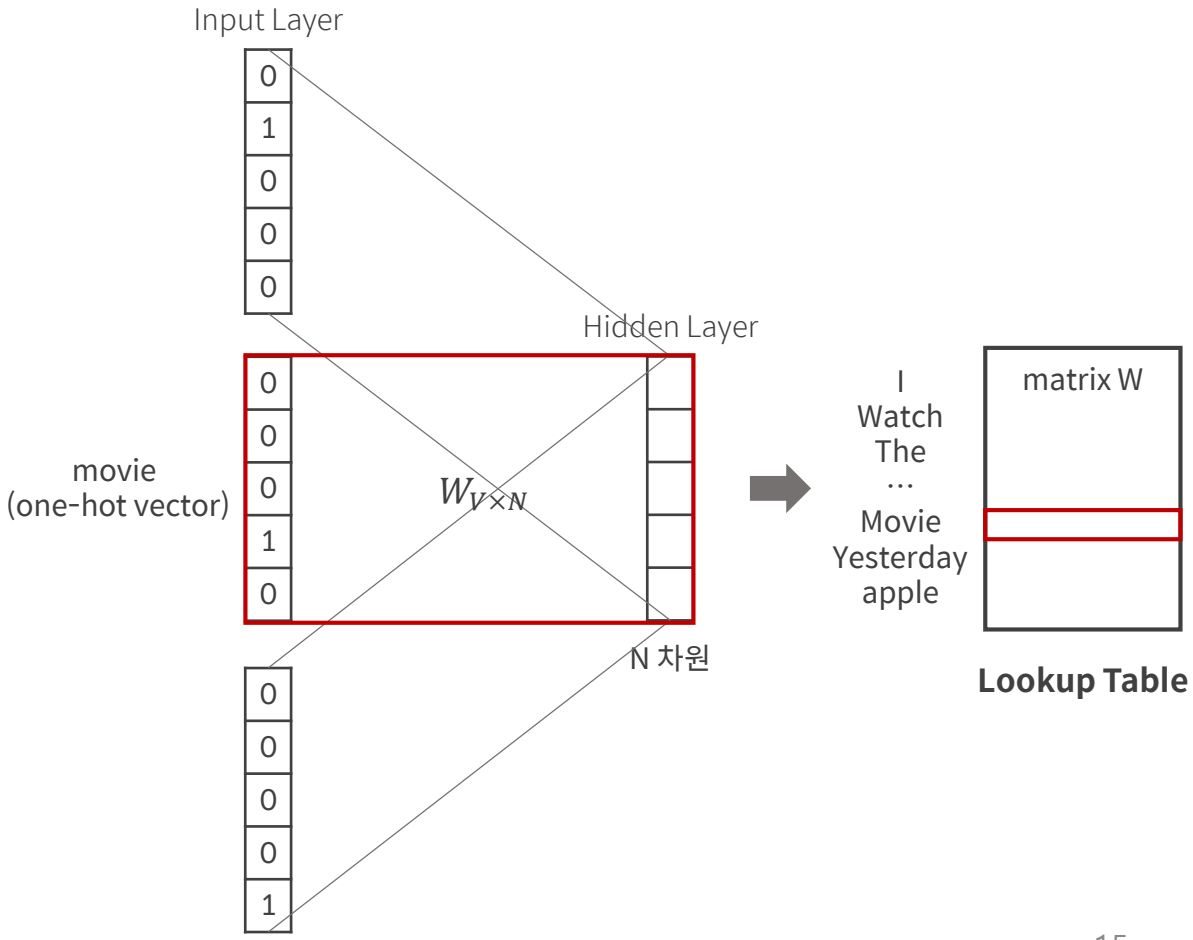
Lookup Table

Word2Vec의 hidden layer를 계산 하는 작업

: 가중치 행렬 W에서 단어에 해당하는 행벡터를 참조(lookup)하는 방식과 같음



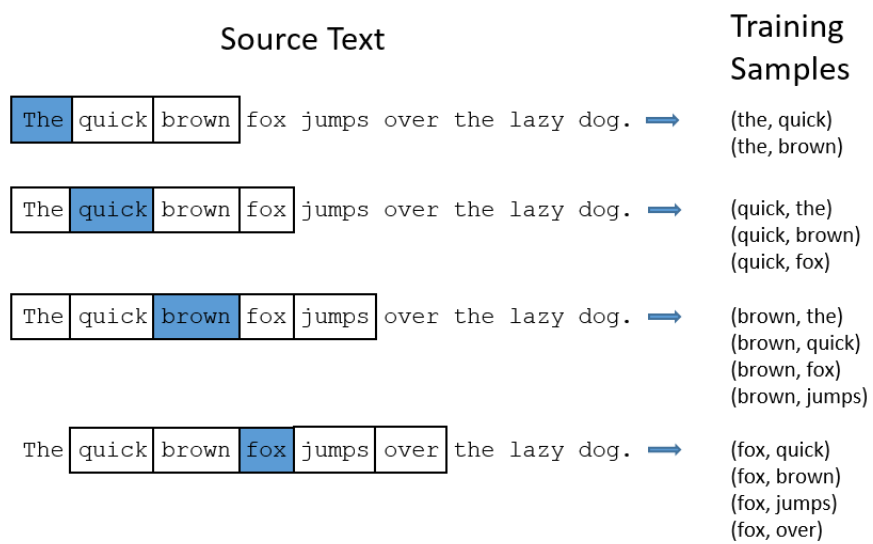
$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$



Word2Vec
CBOW vs. Skip-Gram

Skip-gram이 CBOW보다 성능이 좋다!

∵ Skip-gram이 CBOW보다 중심단어의 업데이트 기회가 더 많기 때문



(그림6) Skip-Gram 학습 진행 과정

Word2Vec
Subsampling frequent words

Subsampling frequent words

Why

Word2Vec의 문제점 : 속도가 느리다 (∵출력 층)

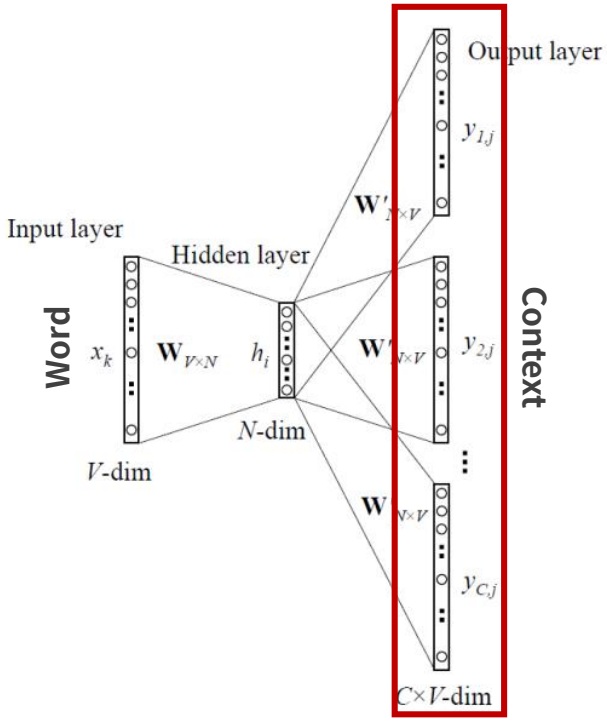
Idea

말뭉치에서 자주 등장하는 단어의 학습량을 줄이자!

i번째 단어 w_i를 학습에서 제외 시키기 위한 확률 $P(w_i)$

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

$P(w_i)$가 큰 '은/는'같은 단어는 학습에서 제외



(그림5) Skip-Gram Architecture

softmax함수
↓
오차 산출
↓
임베딩 조정

: 모든 단어 집합에 대해 수행함

$f(w_i)$: 해당 단어가 말뭉치에 등장한 비율(해당 단어 빈도/전체 단어 수)

t : 사용자가 지정해 주는 값 (0.0001권장)

$P(w_i)$ 가 0에 가까우면 해당 단어가 나올 때 마다 제외하지 않고 학습

Word2Vec
Negative Sampling

Negative Sampling

Idea

Softmax 확률을 구할 때 전체 단어를 대상으로 하지 않고, 일부 단어만 뽑아서 계산

Window내에 등장하지 않은 어떤 단어 w_i가 negative sample로 뽑힐 확률

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

negative sample : window size 내에 등장하지 않는 단어

① negative sample 추출

② negative sample을 정답 단어와 합쳐 softmax확률 계산

ex. window size = 5

→ 최대 25개 단어를 대상으로만 softmax계산