

Word Embedding

오지은

WHY?

- 딥러닝에서 모델의 입력은 벡터(숫자) → 텍스트는 들어갈 수 없음
- 그렇다면 텍스트를 숫자로 표현해야 함
- 가장 쉬운 방법: one-hot encoding
- ❖ “I like apple” → [1 0 0] [0 1 0] [0 0 1]
- vocabulary의 크기가 50만이라면?
 - 너무 큰 차원
 - 너무 sparse함
 - 단어들 사이에서 아무런 관계도 찾을 수 없음 (모든 벡터 사이의 거리가 0) → 아무 정보 없음

WHY?

- 단어들을 One-hot encoding 말고 더 낮은 차원에 **dense**한 표현으로 나타내보자
- ❖ “I like apple” → [1, 2] [1, 3] [2,2]
- 차원이 줄었음
- 단어 벡터들 간에 similarity 판별 가능
- word embedding: 단어를 dense vector로 표현하는 것
- embedding vector: 워드 임베딩의 결과로 나온 벡터

Word2Vec

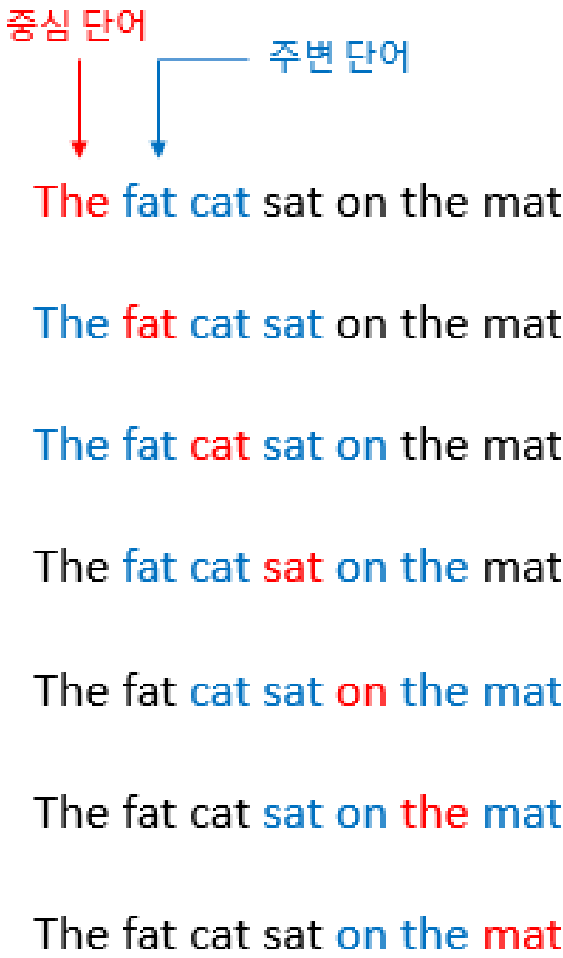
- 가정: 같은 문맥에 등장하는 단어들은 비슷한 의미를 가질 것이다
- ❖ “내 친구 집 강아지는 귀엽다.” “옆집 강아지는 귀엽다.” “귀여운 강아지가 보인다.”
→ ‘강아지’ ‘귀엽다’ 사이에는 관계가 있을 가능성이 높음
- 방법 1: 주변 단어에서 중심 단어 예측 (CBOW)
- 방법 2: 중심 단어에서 주변 단어 예측 (Skip-gram)

❖ The fat cat __ on the mat

- 중심 단어: __ (답: sat)
- 주변 단어: the, fat, cat, on, the, mat → 이것들로부터 __ 을 예측
- Window size: __ 좌우로 몇 개의 단어를 볼 것인가
 - Window size가 2라면: 주변 단어는 fat, cat, on, the

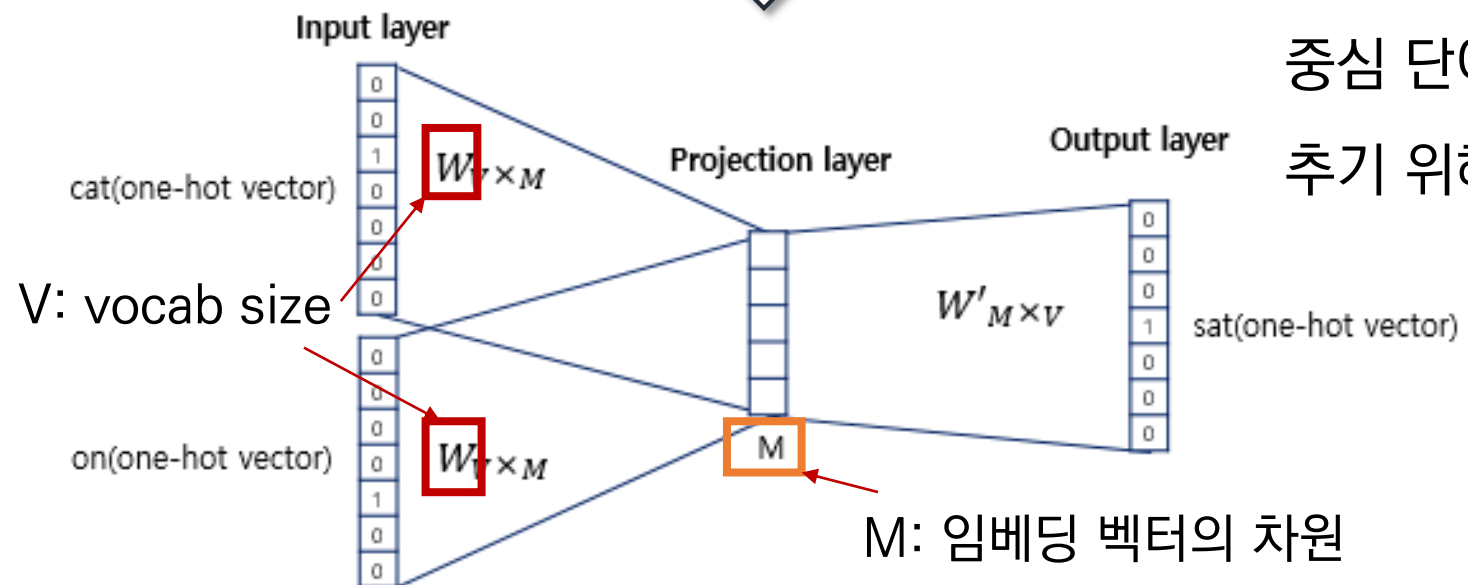
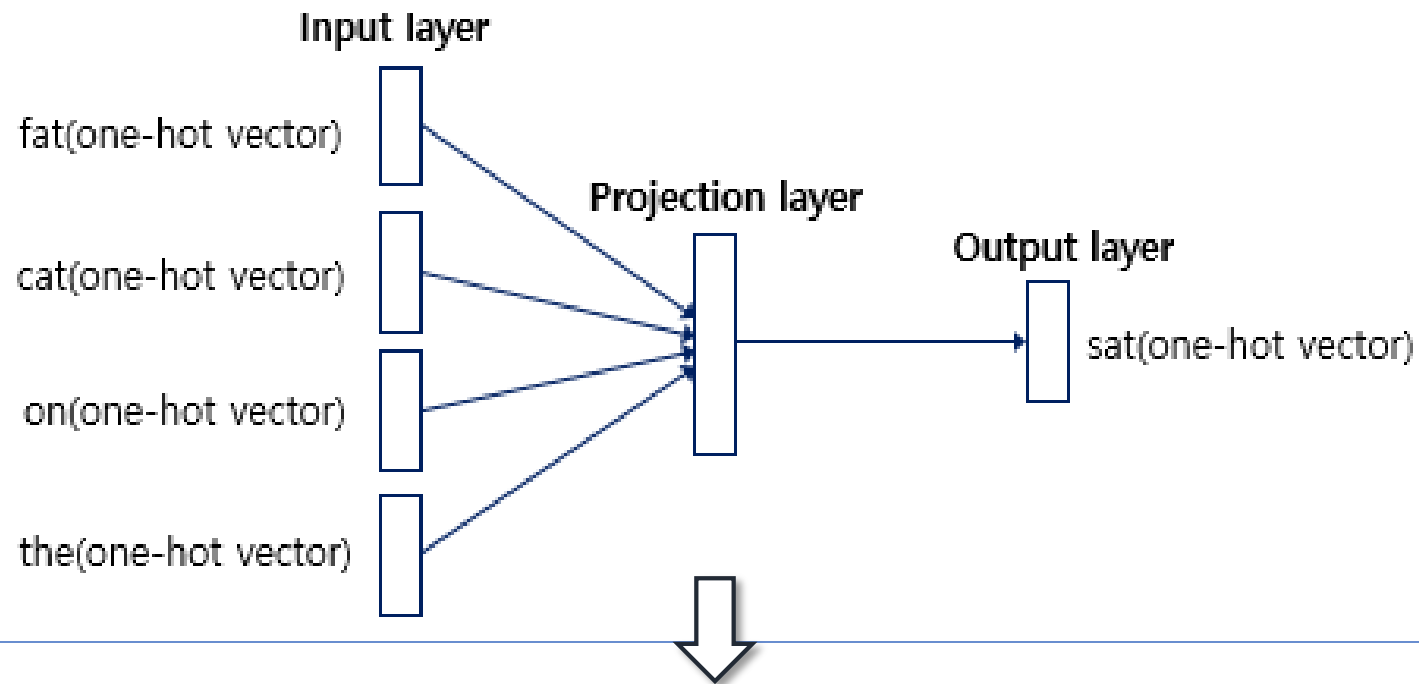
W2V – CBOW

- Sliding window: window를 계속 이동하면서 중심/주변 단어 선택을 바꾸는 것
- 각 입력은 모두 one-hot encoding

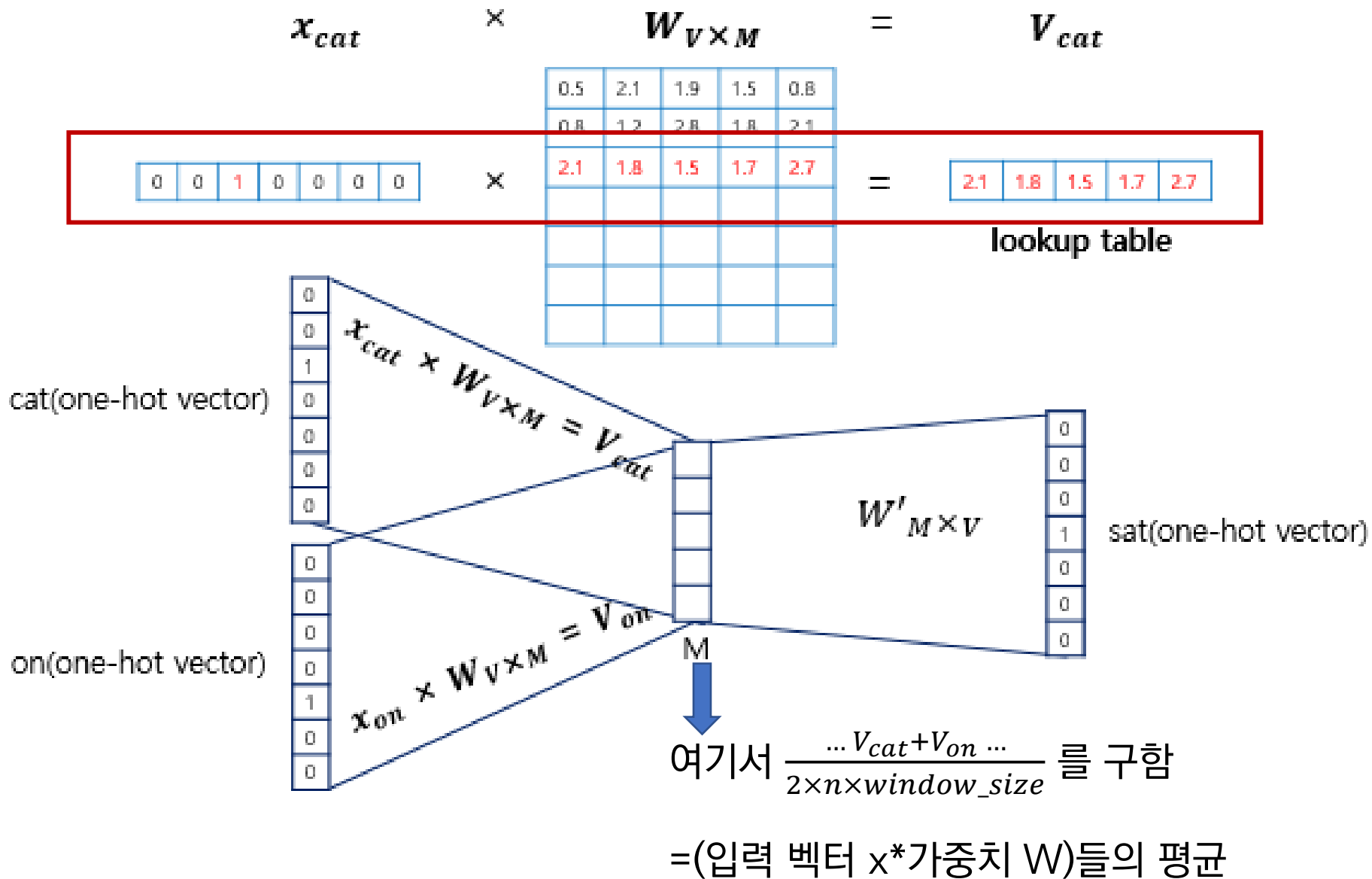


중심 단어	주변 단어
[1, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0]
[0, 1, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0]
[0, 0, 1, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0]
[0, 0, 0, 1, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0]
[0, 0, 0, 0, 1, 0, 0]	[0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 0, 1, 0], [0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 1, 0]	[0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 0, 1]	[0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0]

W2V - CBOW



W2V – CBOW

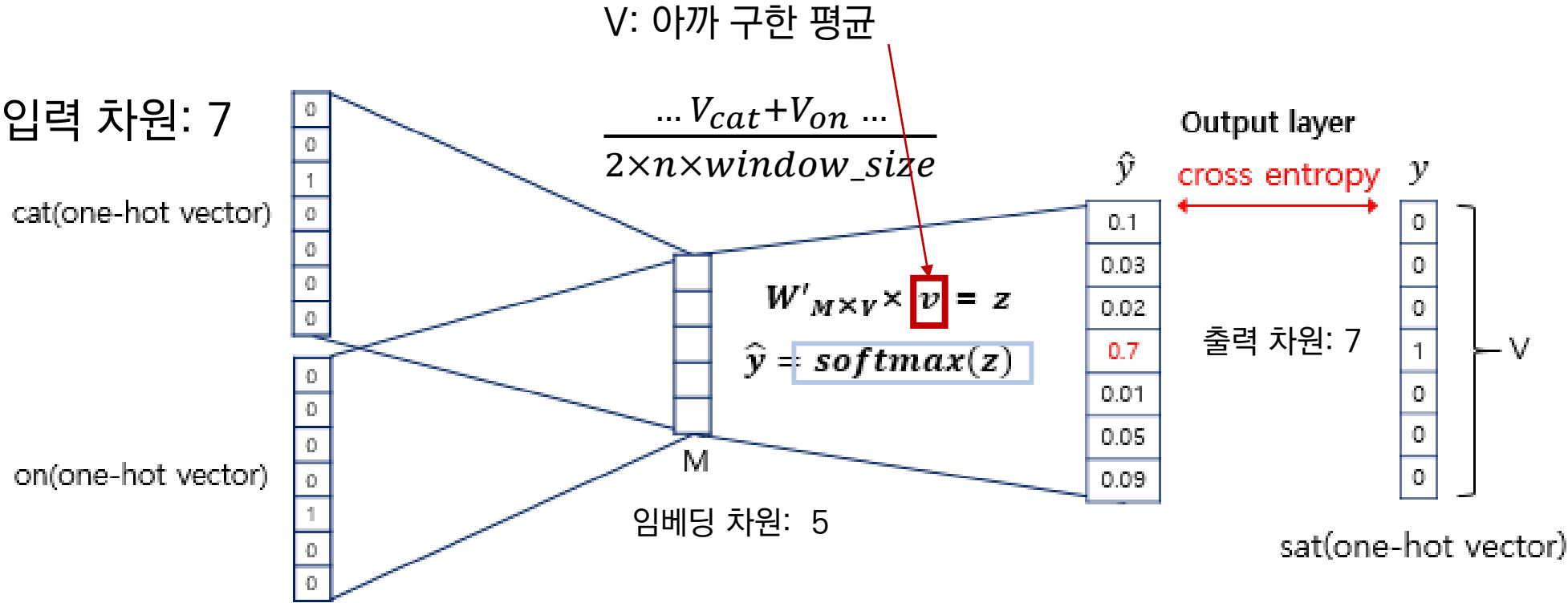


입력 one-hot * W
= W의 그 행과 같음
= lookup

즉 W의 각 행이
임베딩 벡터가 됨

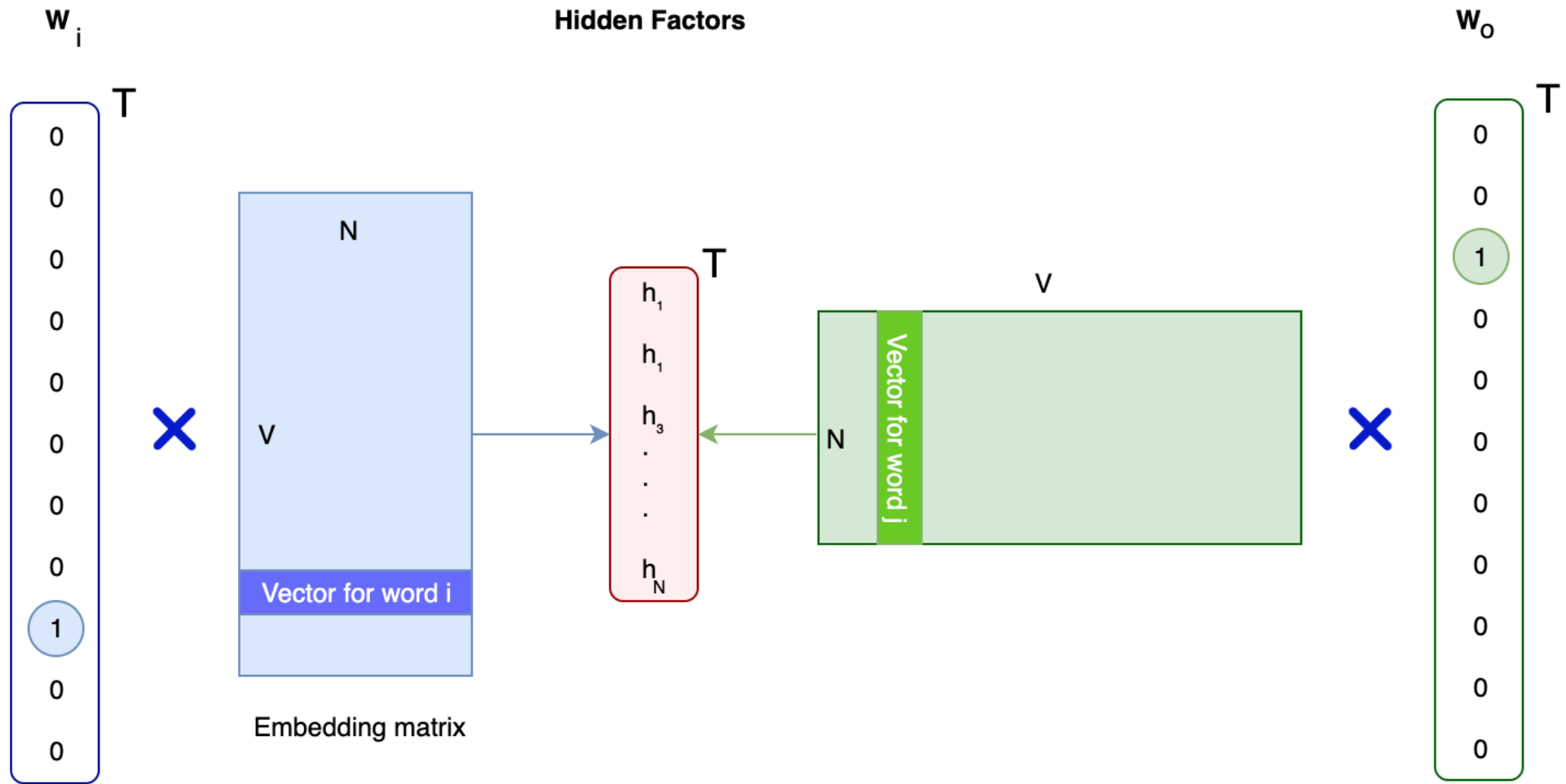
→ W의 학습이
목표인 이유

W2V - CBOW



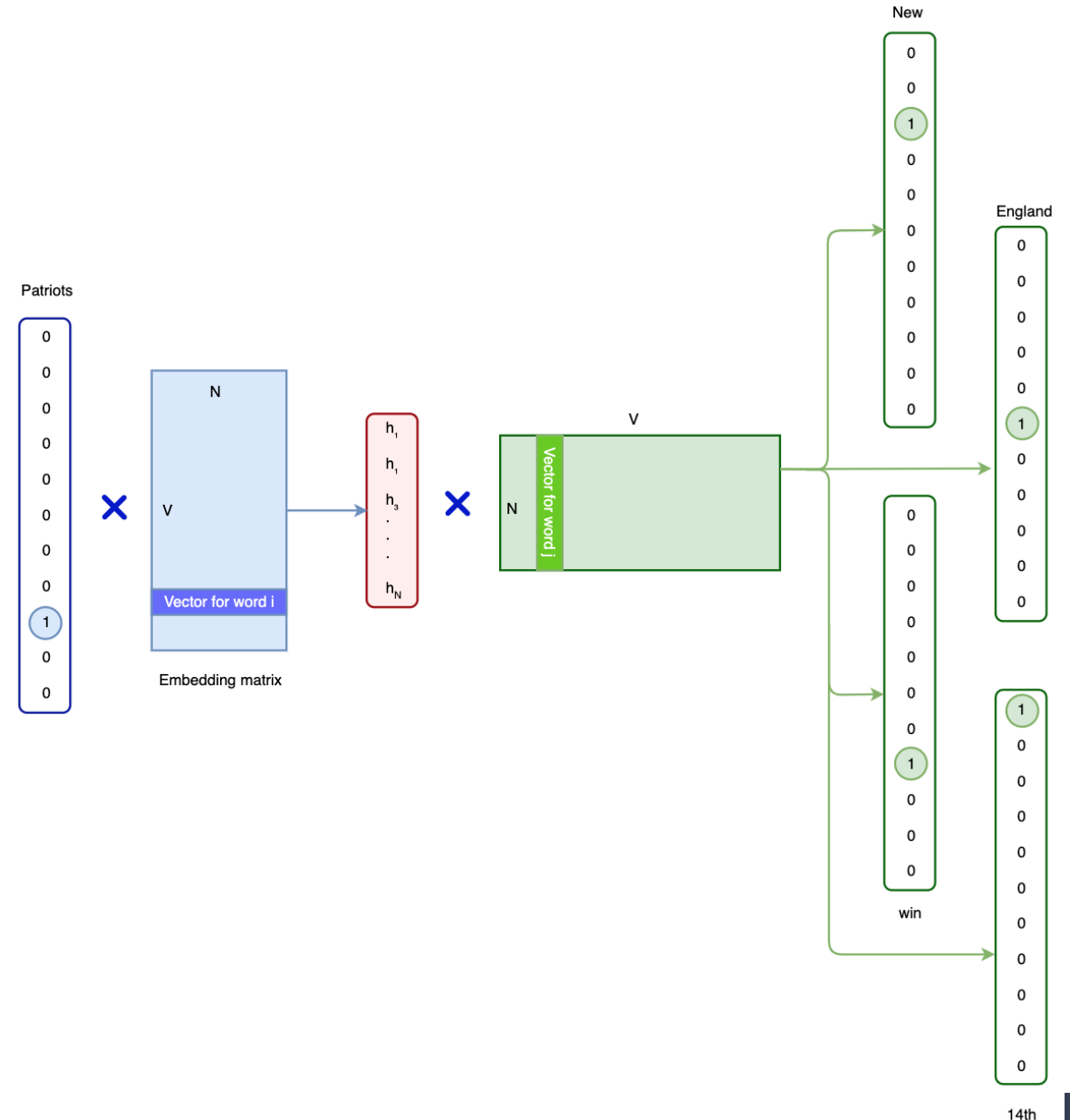
평균 $V \times$ 임베딩 행렬 $W' =$ 입력 one-hot 벡터와 같은 차원의 벡터
그 결과에 softmax \rightarrow 실제 one-hot 벡터 sat과 가까워지도록 학습
즉, 다른 부분은 0에 가깝고 정답 부분은 1에 가까워져야 함

W2V – CBOW



W2V - Skip-gram

- 중심 단어에서 주변 단어 예측
- The ___ sat ___ mat
- 단, projection layer에 들어가는 입력이 하나이기 때문에 평균을 구하는 과정은 없음
- CBOW가 the, fat, on, the 한 번 학습 할 동안 skip-gram은 sat, the / sat, fat / sat, on / sat the 4번 학습
- 일반적으로 skip-gram이 더 좋음



GloVe

- W2V의 문제점: 지정한 window 내에서만 학습 → 말뭉치 전체의 global co-occurrence가 반영되기 어렵다 (통계 정보 반영 x)
- 말뭉치 전체의 통계 정보를 반영하기 위해, **동시 등장 확률**을 이용해보자
“ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning”

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k ice)/P(k steam)$	8.9	8.5×10^{-2}	1.36	0.96

GloVe - cooccurrence

- 동시 등장 행렬: 단어 i 근처에 단어 k 가 등장한 횟수를 (i, k) 에 기재한 행렬
- 근처 = 지정된 크기의 window 안

I like deep learning

I like NLP

I enjoy flying

에서의 동시등장행렬

카운트	I	like	enjoy	deep	learning	NLP	flying
I	0	2	1	0	0	0	0
like	2	0	0	1	0	1	0
enjoy	1	0	0	0	0	0	1
deep	0	1	0	0	1	0	0
learning	0	0	0	1	0	0	0
NLP	0	1	0	0	0	0	0
flying	0	0	1	0	0	0	0

- 동시 등장 확률 $P(k|i)$: 동시 등장 행렬에서 중심 단어 i 행의 모든 값을 더한 것을 분모로, i 행 k 열의 값을 분자로 한 값

$P(\text{deep} | \text{learning})$

카운트	I	like	enjoy	deep	learning	NLP	flying
I	0	2	1	0	0	0	0
like	2	0	0	1	0	1	0
enjoy	1	0	0	0	0	0	1
deep	0	1	0	0	1	0	0
learning	0	0	0	1	0	0	0
NLP	0	1	0	0	0	0	0
flying	0	0	1	0	0	0	0

GloVe - cooccurrence

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k ice)/P(k steam)$	8.9	8.5×10^{-2}	1.36	0.96

Very small or large:

solid is related to ice but not steam, or
gas is related to steam but not ice

close to 1:

water is highly related to ice and steam, or
fashion is not related to ice or steam.

Ice와 solid가 함께 나올 확률: 높음

Steam과 solid가 함께 나올 확률: 낮음

Solid와 ice가 함께 나올 확률은 solid와 steam이 함께 나올 확률의 8.9배

Ice와 fashion, steam과 fashion이 함께 나올 확률의 비: 서로 고만고만함 (≈ 1)

GloVe의 목표: 중심 단어와 주변 단어의 내적이 전체 코퍼스에서의 동시 등장 확률이 되도록 만드는 것

$$\rightarrow w_i \times w_k = P(k|i) = P_{ik}$$

더 정확히는 $\log P(k|i)$

- 말뭉치 전체에 대해 여러 단어 간 상호 비율 정보를 반영하고 싶다!

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

중심 단어 i가 있을 때
원도우 안에 k가 나올 확률

중심 단어 j가 있을 때
원도우 안에 k가 나올 확률

Objective function: 이 식을 만족하는 함수 F

i가 ice, j가 steam, k가 solid라고 할 때 $F(\text{ice}, \text{steam}, \text{solid}) = 8.9$

k가 주어졌을 때 두 단어의 내적이 둘의 동시등장확률 비가 되도록 함

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)}$$

$$F(w_i^T \tilde{w}_k - w_j^T \tilde{w}_k) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)}$$

나눗셈의 로그는 로그의 뺄셈과 같다는 성질을 이용하여 동시등장확률비의 로그를 단어 벡터 간의 차이와 연관시키기로 함

→ F의 입력을 두 단어의 차이와 단어 k로 정함

우변은 스칼라인데 좌변은 벡터이므로

좌변을 내적하여 스칼라로 만듦

Homomorphism을 만족하기 위한 변형

GloVe – objective

- 중심 단어와 주변 단어 i, j, k 는 모두 임의로 선택되는 것 → 서로 언제든지 바뀔 수 있음
- i, j, k 가 서로 교환되어도 식이 성립되도록 하기 위해 homomorphism을 만족
- Homomorphism(준동형) : $F(a+b) = F(a)F(b)$ 인 것
- 그런데 a, b 가 벡터라면 F 의 결과로 스칼라가 나올 수 없음 → a, b 는 사실 두 벡터의 내적
- GloVe 식에서의 입력은 단어 i 와 단어 j 벡터의 차이 → 덧셈은 뺄셈으로, 곱셈은 나눗셈으로
변하면 준동형 성립

$$F(v_1^T v_2 - v_3^T v_4) = \frac{F(v_1^T v_2)}{F(v_3^T v_4)}, \forall v_1, v_2, v_3, v_4 \in V$$

- 원래 식 $F((w_i - w_j)^T \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$ 에 준동형 적용

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

좌변을 풀어주면

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

우변에 준동형 형태 적용

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)}$$

즉 $F(w_i^T \tilde{w}_k) = P_{ik}$

$$F(w_i^T \tilde{w}_k - w_j^T \tilde{w}_k) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)}$$

GloVe - objective


- 그래서 그걸 다 만족하는 F는 무엇인가: exponential 함수

$$\exp(w_i^T \tilde{w}_k - w_j^T \tilde{w}_k) = \frac{\exp(w_i^T \tilde{w}_k)}{\exp(w_j^T \tilde{w}_k)}$$

$$\exp(w_i^T \tilde{w}_k) = P_{ik} = \frac{X_{ik}}{X_i}$$

단어 i 근처에 단어 k가 나오는 횟수
단어 i가 나오는 횟수

아까 목표라고 했던 것


$$\boxed{w_i^T \tilde{w}_k = \log P_{ik}} = \log \left(\frac{X_{ik}}{X_i} \right) = \log X_{ik} - \log X_i$$

문제: i와 k는 언제든지 바뀔 수 있어야 하는데, $\log P_{ik}$ 와 $\log P_{ki}$ 는 같지 않다

→ $\log X_i$ 부분을 상수 b로 대체

$$w_i^T \tilde{w}_k = \log X_{ik} - b_i - \tilde{b}_k$$

$$\boxed{w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log X_{ik}}$$

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log X_{ik}$$

- 좌변: 학습하는 변수들
- 우변: 정답 (동시등장행렬 $X[i,k]$)
- 우변과의 차이를 줄이는 방향으로 좌변의 변수들 학습



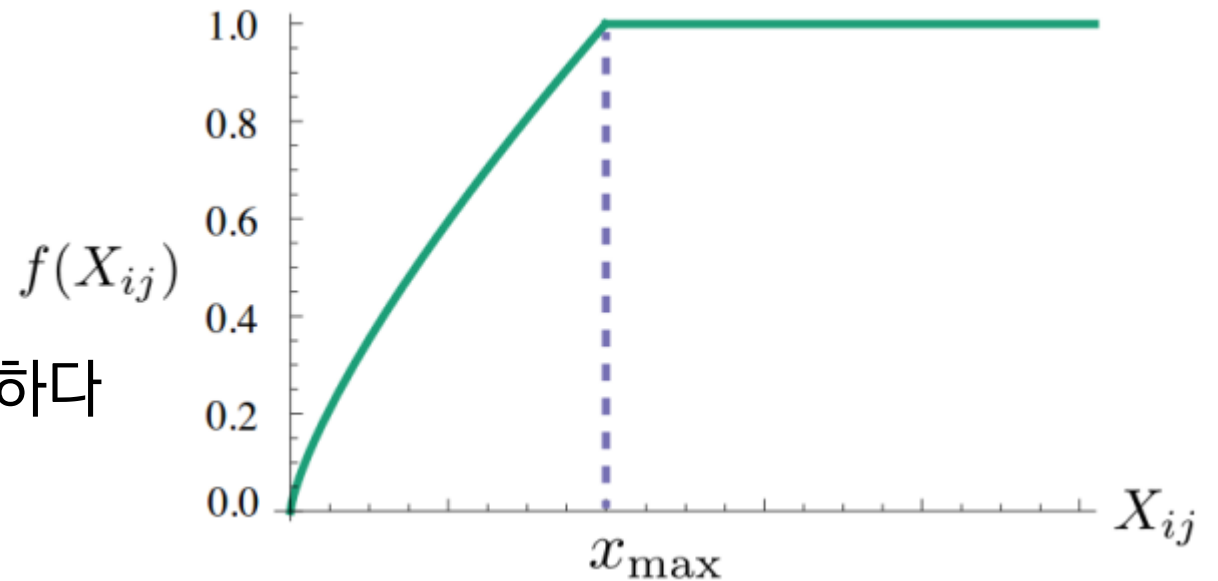
$$J = \sum_{i,j=1}^V (w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

GloVe – objective

- 그런데 동시 등장 행렬 X 자체가 sparse하다
- 많은 값이 0이거나 아주 작은 수치

→ 도움이 되는 정보가 아님

- 동시등장행렬 X 의 값에 영향을 받는 함수 $f(x)$ 도입
 - $X[i,k]$ 의 값이 크면 가중치가 커지지만, 무한정 커지지는 않음



$$\text{Loss function} = \sum_{m,n=1}^V f(X_{mn}) (w_m^T \tilde{w}_n + b_m + \tilde{b}_n - \log X_{mn})^2$$

fastText

- Word2Vec의 문제

- 단어의 morphological한 특징을 반영하지 않음
- 출현횟수가 적은 단어는 임베딩이 잘 되지 않음

→ 단어들을 쪼개서 임베딩하자는 아이디어

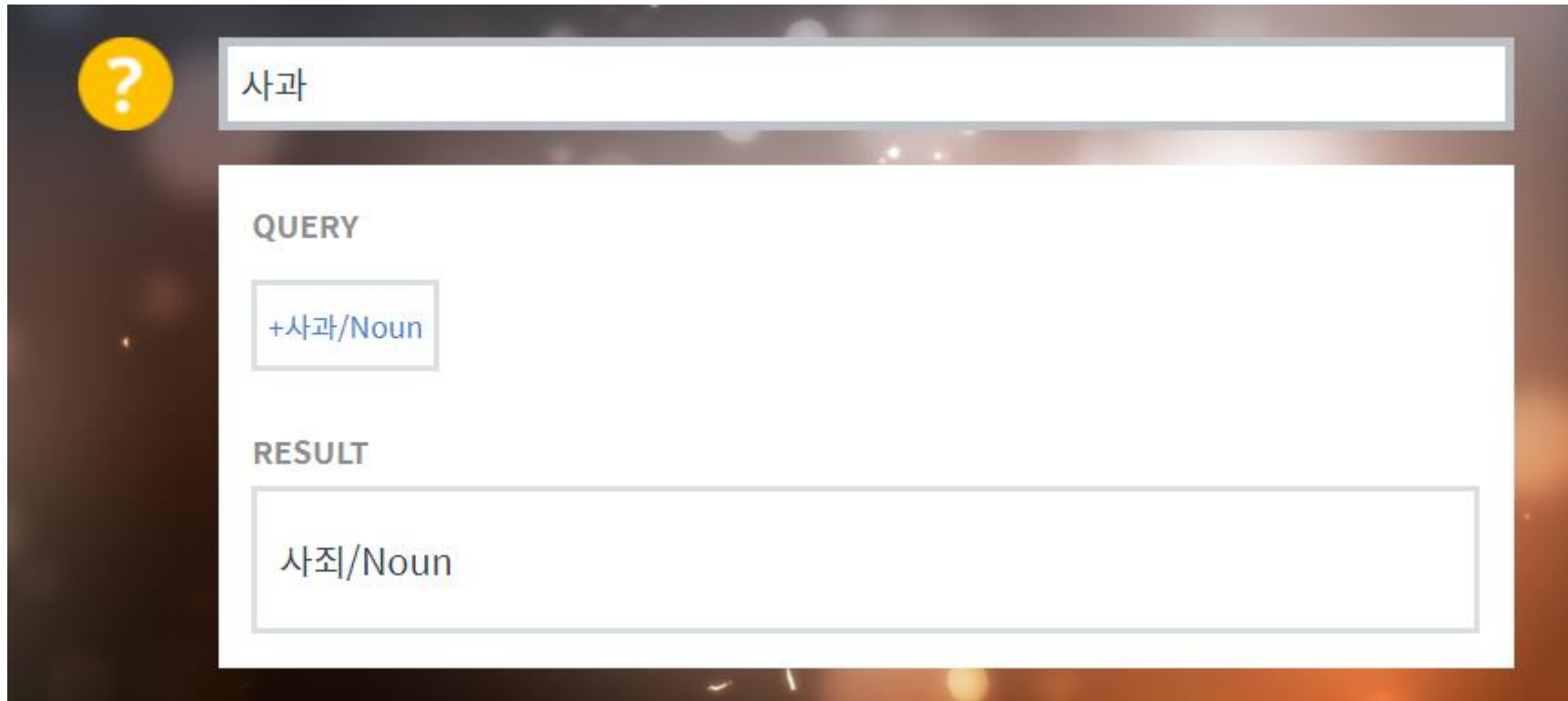
- 단어를 문자들의 ngram으로 취급: 한 단어에 대한 벡터가 ngram들의 합

❖ where → 〈wh, whe, her, ere, re〉 (n=3, n은 하이퍼파라미터)

- OOV 문제 해결 가능 : 학습할 때 존재하지 않았던 단어도 표현

Problem

Problem of static embedding



A screenshot of a search interface. At the top left is a yellow circle with a white question mark. To its right is a search bar containing the Korean word '사과'. Below the search bar, the word '사과' is also displayed. Underneath, the word is shown with its part of speech: '+사과/Noun'. Below that, the word is shown with its meaning: '사죄/Noun'.

- 사과(apple)와 사과(apology)의 의미가 전혀 다름에도 불구하고 같은 벡터로 임베딩
- 문맥 정보에 대한 고려가 없고 동음이의어, 다의어를 처리할 수 없음

Contextualized Word Embedding



references

- https://medium.com/@jonathan_hui/nlp-word-embedding-glove-5e7f523999f6
- <https://ratsgo.github.io/>
- <https://wikidocs.net/22644>
- <https://nlp.stanford.edu/projects/glove/>
- <https://research.fb.com/blog/2016/08/fasttext/>
- <https://word2vec.kr/search/>

End of Document