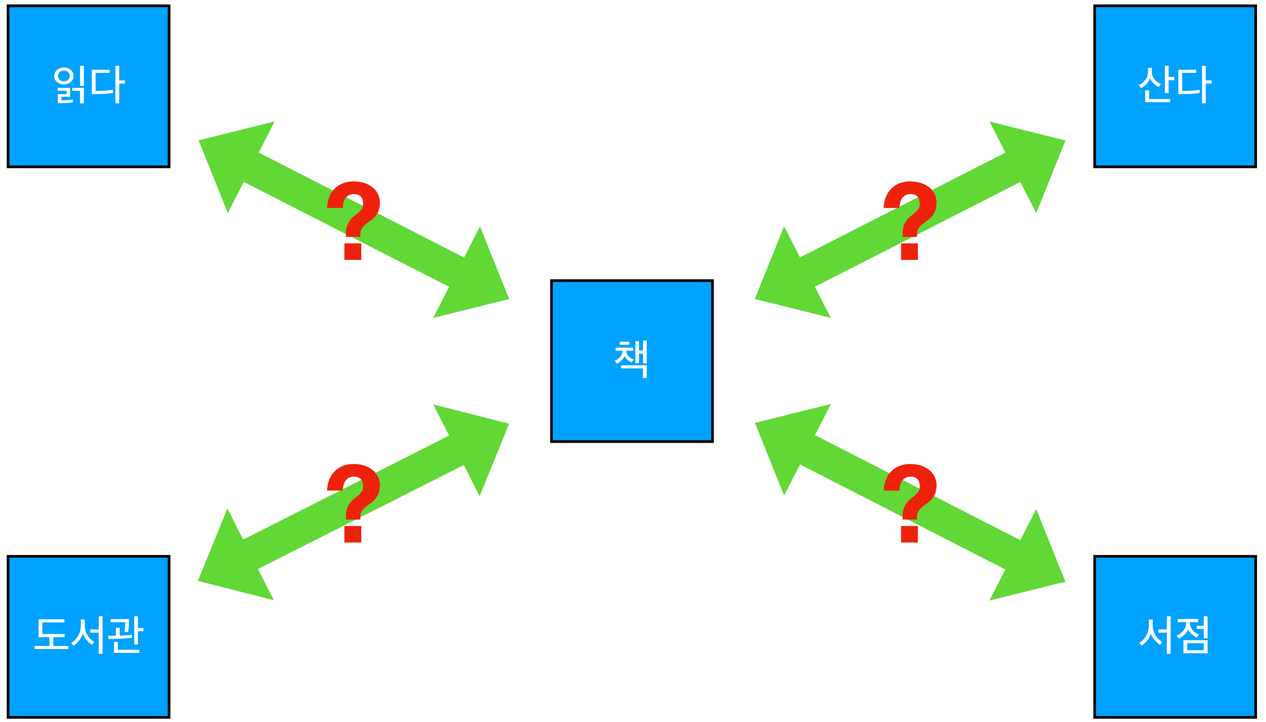
ICT이노베이션스퀘어 AI복합교육 고급 언어과정

자연어처리를 위한 Word Embedding

현청천

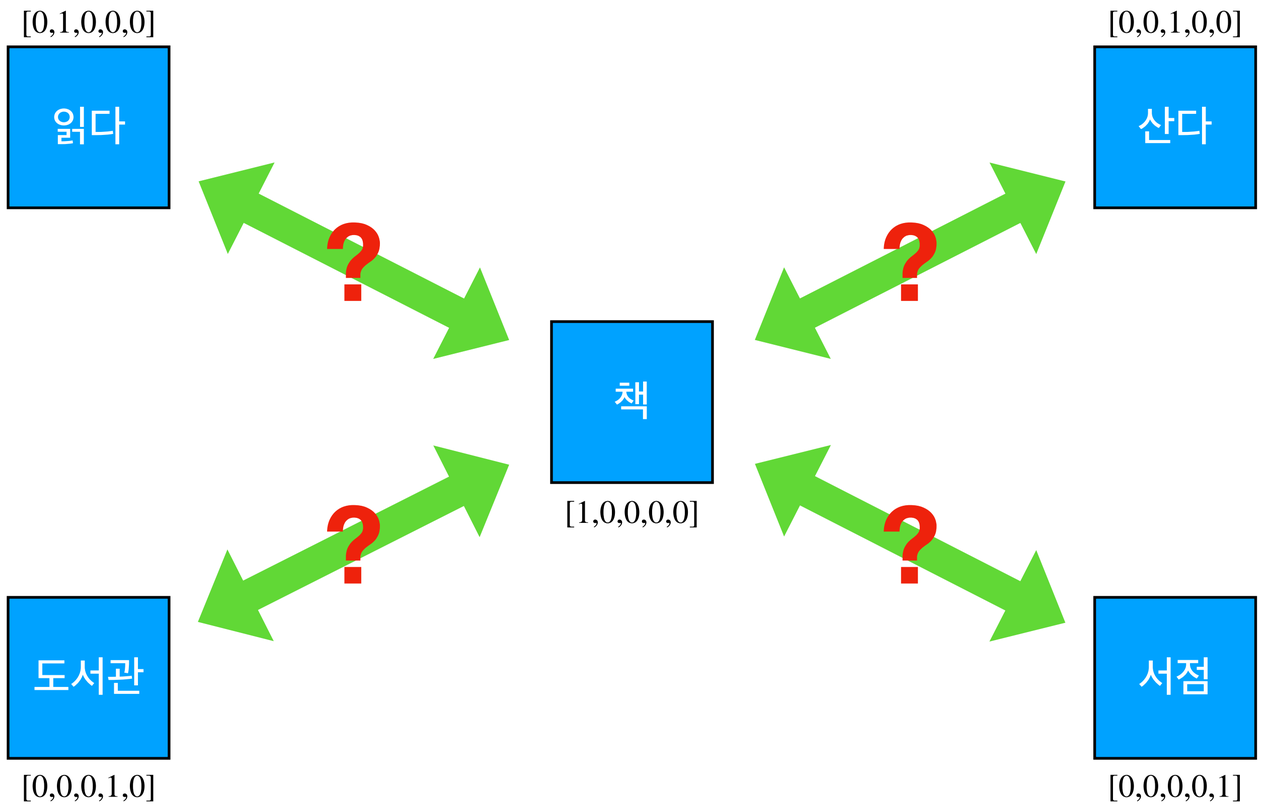
2021.04.19

Word Representation



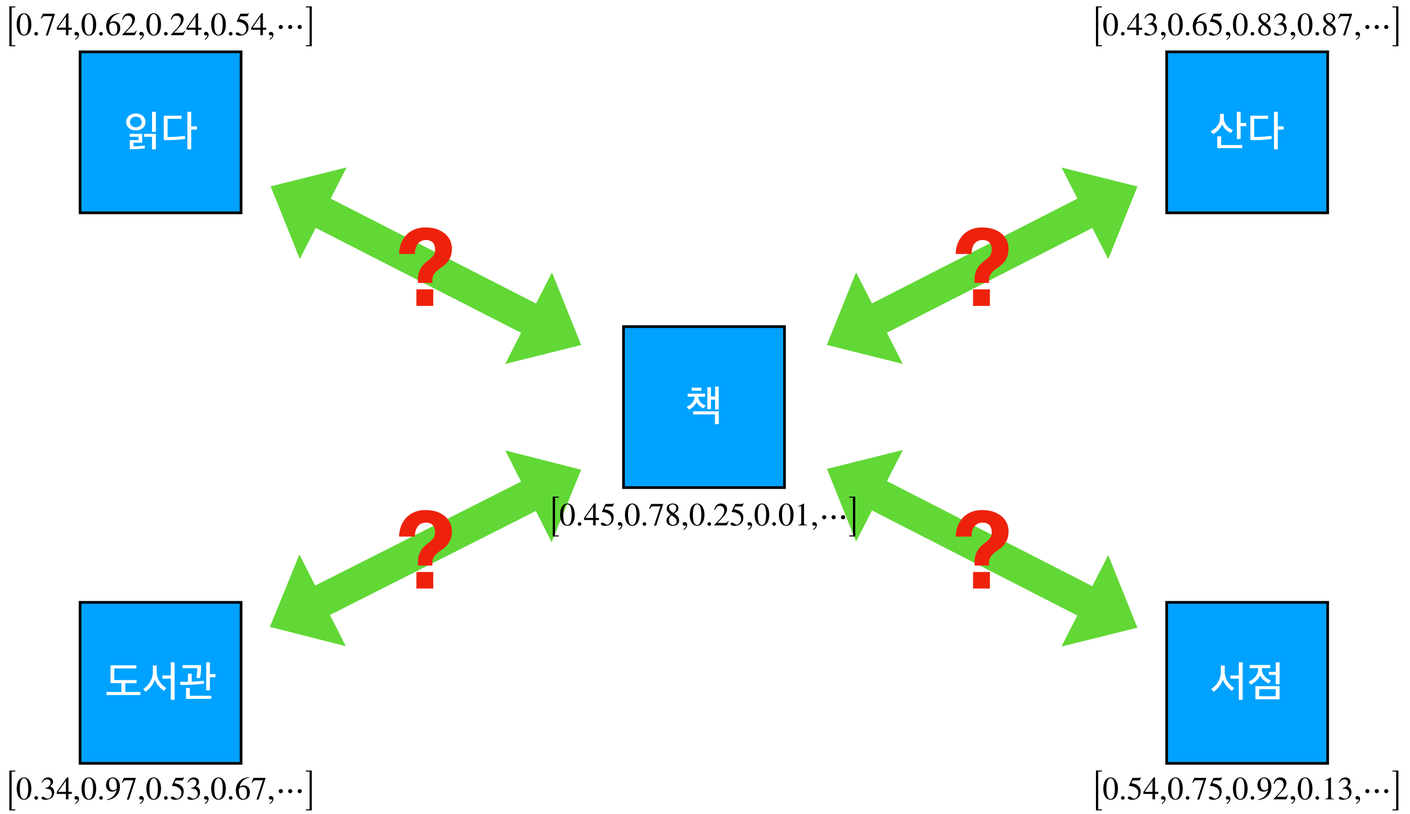
각 단어 벡터들이 의미를 잘 표현하게 하는 방법

Word Representation (One-Hot)



모든 벡터들이 직교하며 단어 간에 어떠한 의미도 표현하지 못함

Word Representation (Dens Vector)



학습을 통해 단어벡터들의 관계로 단어의 의미를 표현

Word Representation (Distributional semantics)

- 단어의 의미는 주변에 자주 나타나는 단어를 통해 알 수 있음
- 한 단어가 나타났을 때 주변의 일정한 범위의 단어에 의미가 포함되어 있음
- 한 단어가 나타나는 많은 문장을 이용해 단어의 표현을 학습

- 나는 책을 서점에서 샀다.
- 우리는 도서관에서 책을 읽었다.
- 나는 영어책을 집에 놓고 학교에 갔다.
- 이번에 새로 나온 수학책이 좋은 것 같다.
- 요즘 재미있는 소설책은 어떤 것이 있나요?

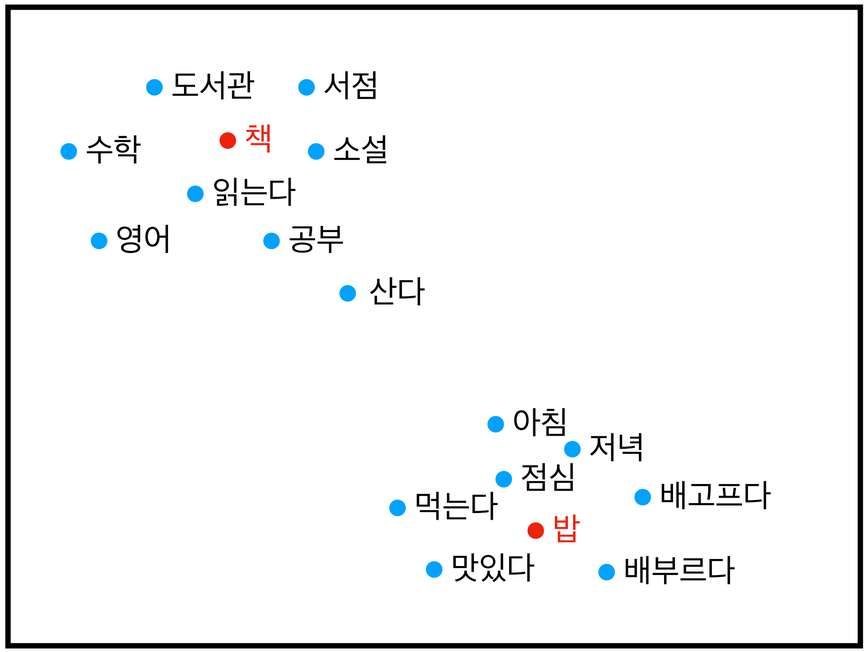
- 오늘은 아침밥을 안 먹어서 배가 고프다.
- 점심밥을 너무 많이 먹어서 배가 부르다.
- 어제 저녁에 먹은 식당밥이 너무 맛있었다.
- 다이어트 해야 하는데 밥맛이 너무 좋다.
- 나는 밥 보다는 빵이 좋다.

주변 단어들이 중심 단어(밥, 책)를 표현

Word Representation (Distributional semantics)

$$\text{책} = \begin{bmatrix} 0.323 \\ -0.654 \\ 0.632 \\ 0.213 \\ 0.203 \\ 0.342 \\ 0.875 \end{bmatrix}$$

$$\text{밥} = \begin{bmatrix} 0.245 \\ -0.283 \\ 0.435 \\ 0.927 \\ 0.121 \\ 0.654 \\ 0.987 \end{bmatrix}$$



주변 단어들이 중심 단어(밥, 책)를 표현

Word2Vec

- Mikolov et al. 2013 google
- 단어 벡터를 학습하기 위한 framework

- Idea
  - 많은 text data (Wiki, book corpus, news, ...)가 존재함
  - 모든 단어는 vector로 표현됨
  - 문장 내에서 위치 t에 대한 중심 단어 c와 주변 단어 o 가 있음
  - c에 대해 o가 나타날 확률을 계산하는데 c와 o의 벡터의 유사성을 이용함
  - 단어의 벡터를 변경하여 c에 대해 o가 나타날 확률을 최대화 함

참조: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/slides/cs224n-2019-lecture01-wordvecs1.pdf

Word2Vec

Skip-gram

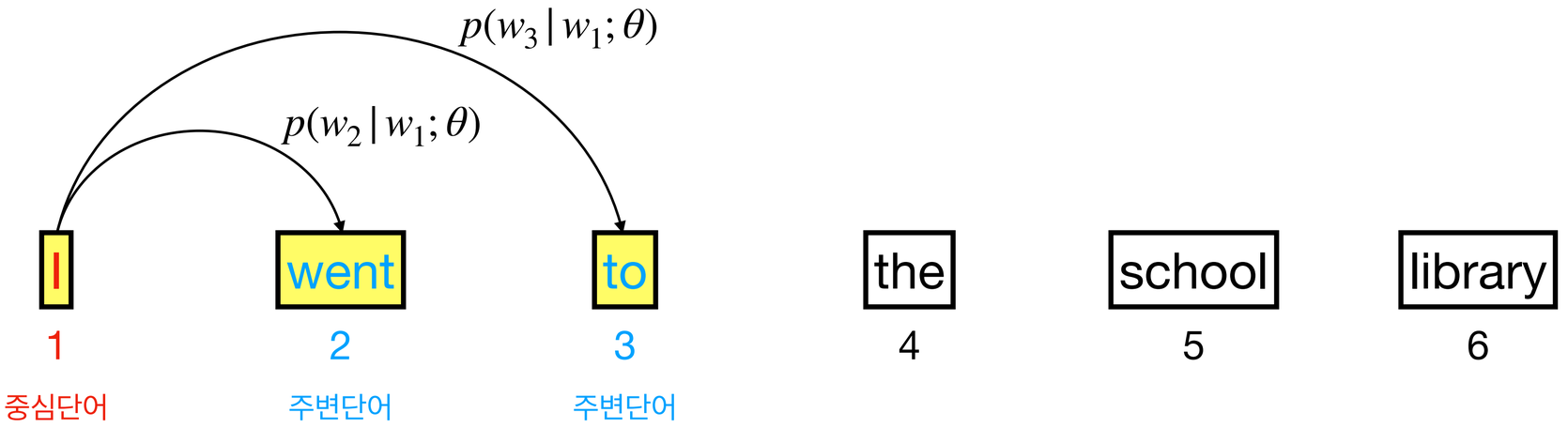
I went to the school library 하나의 중심 단어를 통해 주변 단어를 예측

CBOW (Continuous Bag of Words) 여러 주변 단어를 통해 중심 단어를 예측

I went to the school library

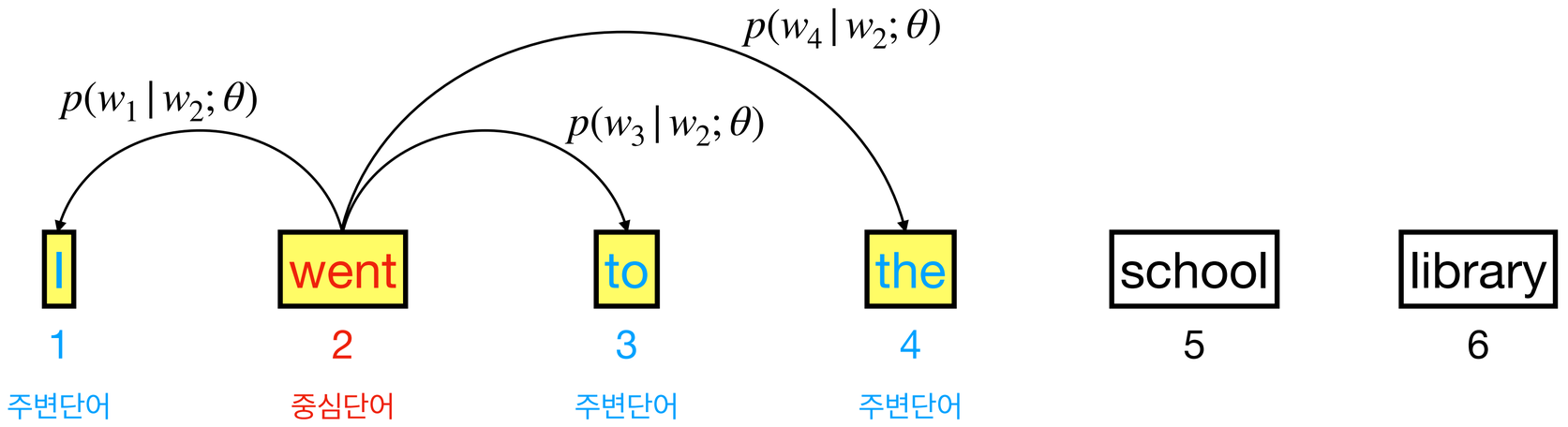
Word2Vec

Skip-gram (window size 2)



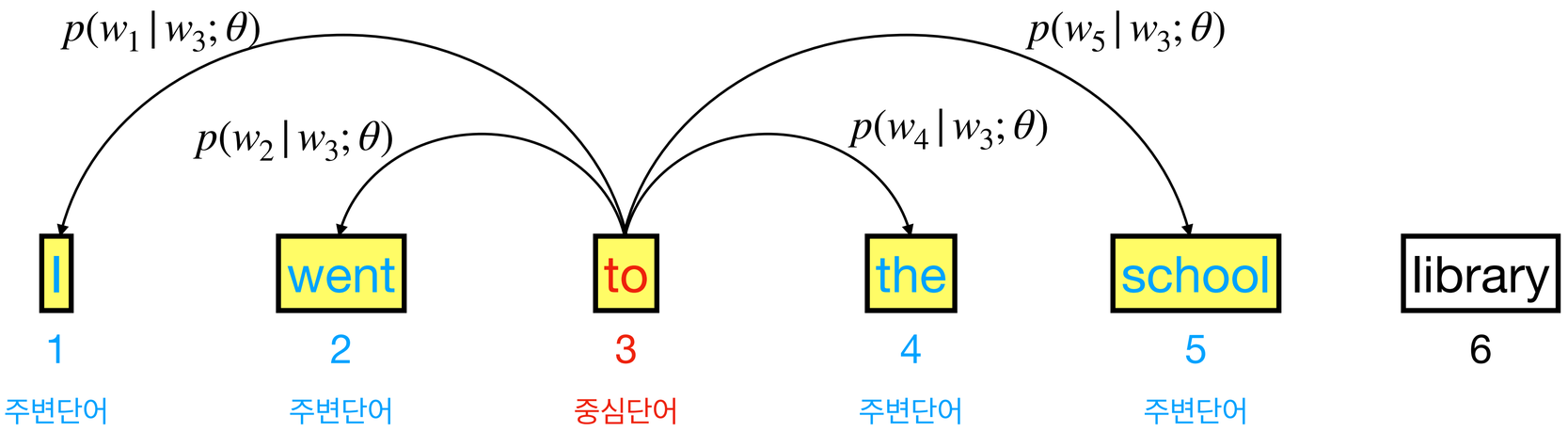
Word2Vec

Skip-gram (window size 2)



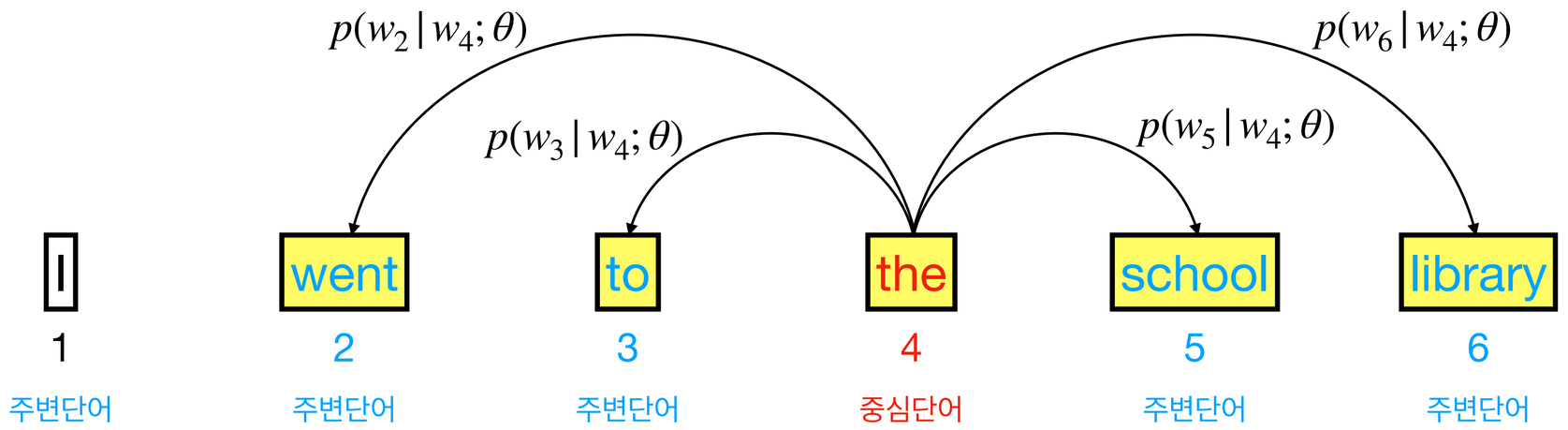
Word2Vec

Skip-gram (window size 2)



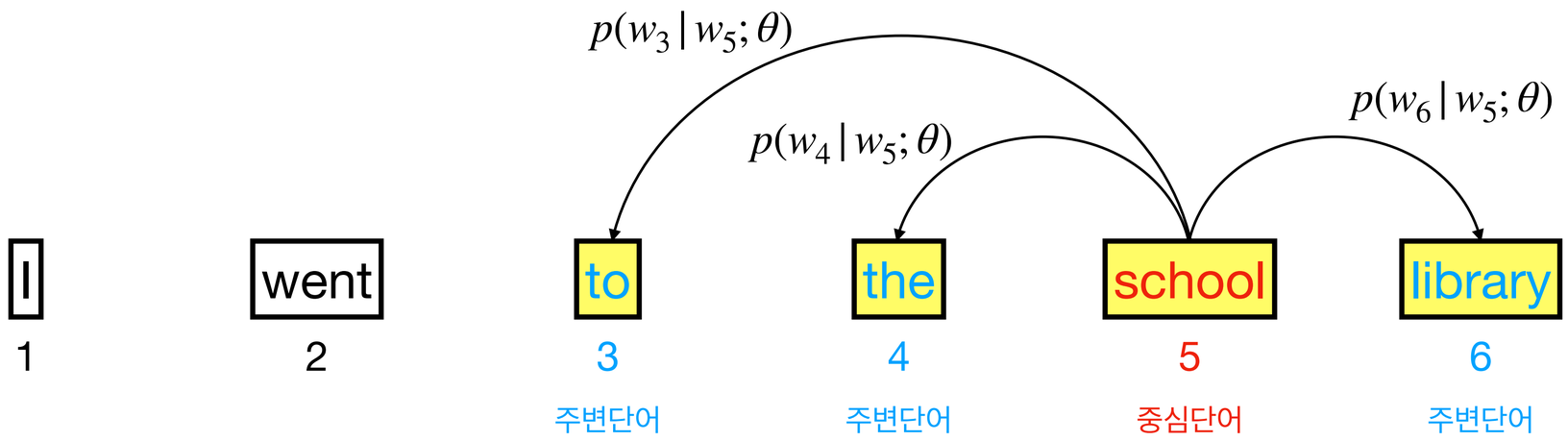
Word2Vec

Skip-gram (window size 2)



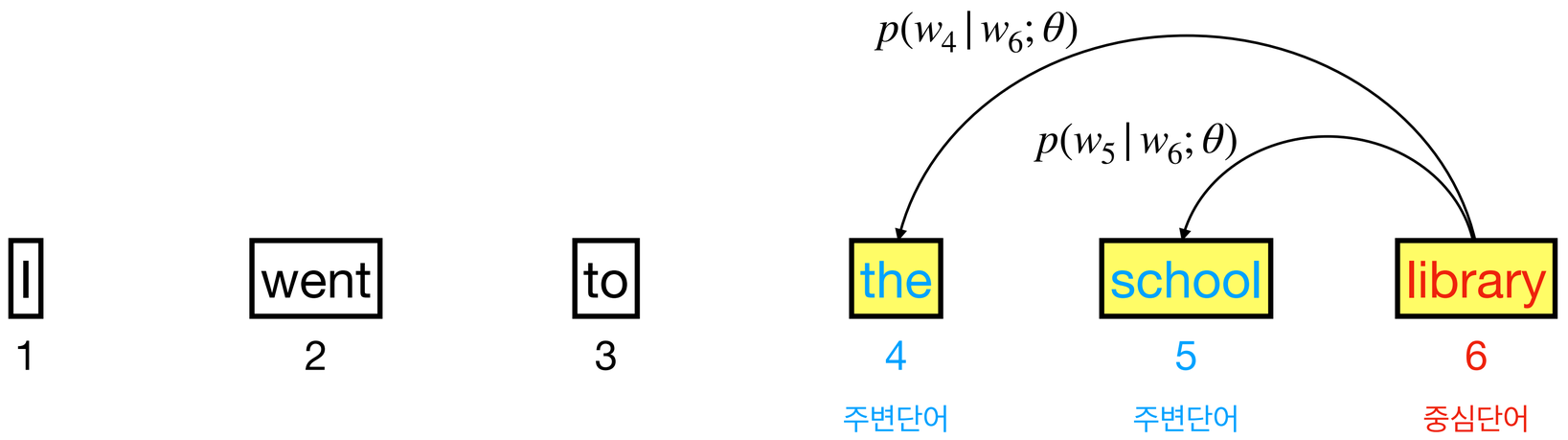
Word2Vec

Skip-gram (window size 2)



Word2Vec

Skip-gram (window size 2)



Word2Vec

Skip-gram

Likelihood
(maximize)

$$L(\theta) = \prod_{t=1}^{T} \prod_{-m \le j \le m}^{j \neq 0} p(w_{t+j} \mid w_t; \theta)$$

- t: 단어의 위치 [1..T]
- m: 윈도우 사이즈

Objective Function
(minimize)

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

참조: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/slides/cs224n-2019-lecture01-wordvecs1.pdf

Word2Vec

Skip-gram

$$p(w_{t+j} \mid w_t; \theta)$$

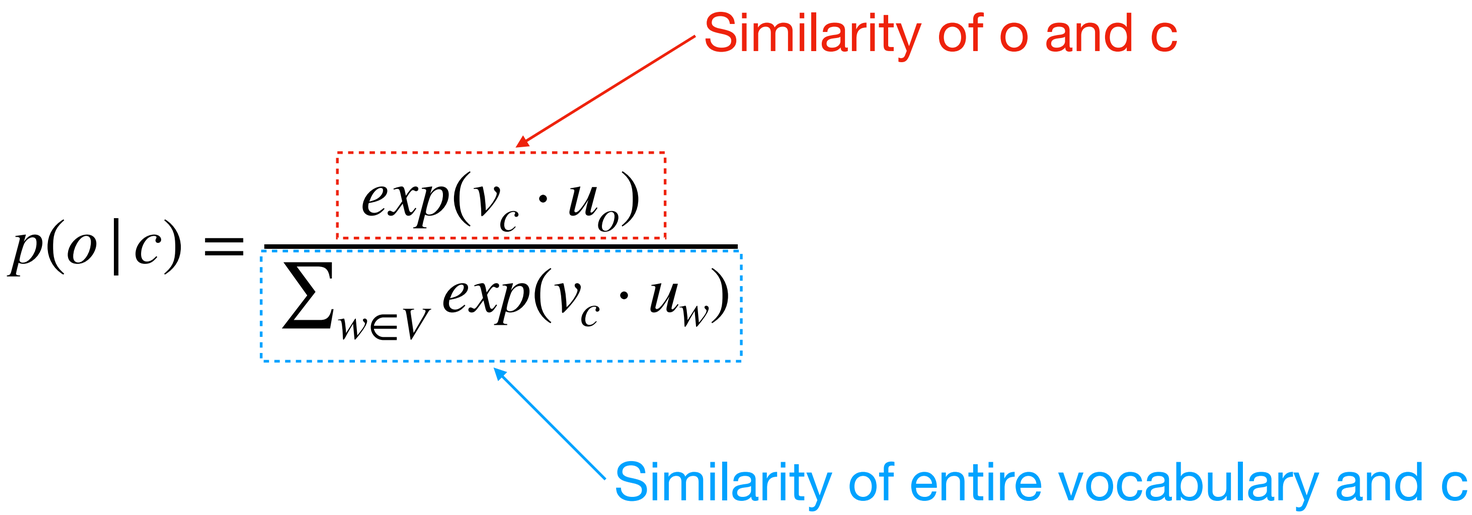
- v_w: center word 벡터
- u_w: outer word 벡터
- V: entire vocabulary

$$p(o \mid c) = \frac{exp(v_c \cdot u_o)}{\sum_{w \in V} exp(v_c \cdot u_w)}$$

참조: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/slides/cs224n-2019-lecture01-wordvecs1.pdf

Word2Vec

Skip-gram

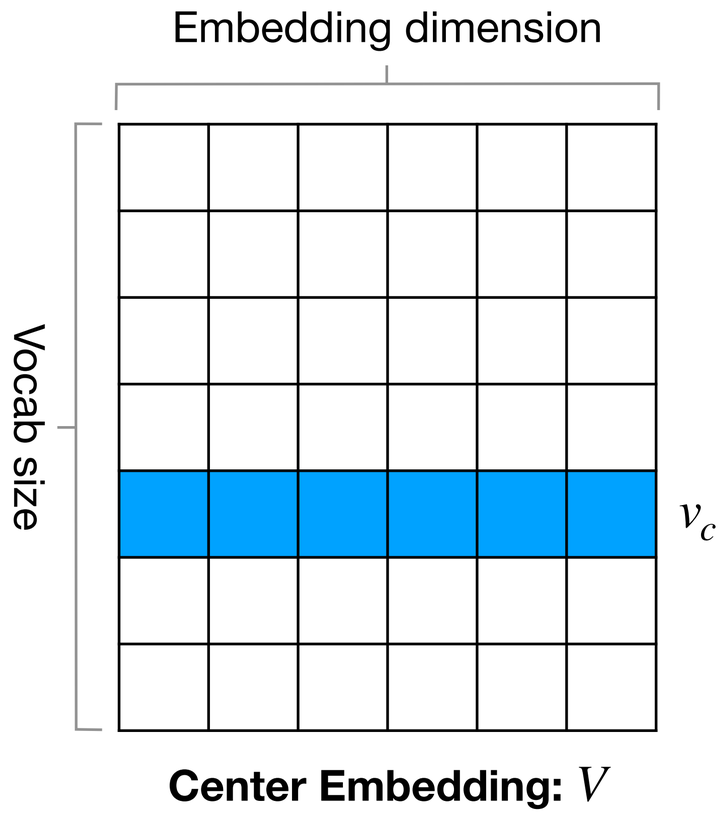


$$softmax(x_i) = \frac{exp(x_i)}{\sum_{j=1}^{n} exp(x_j)} = p(x_i)$$

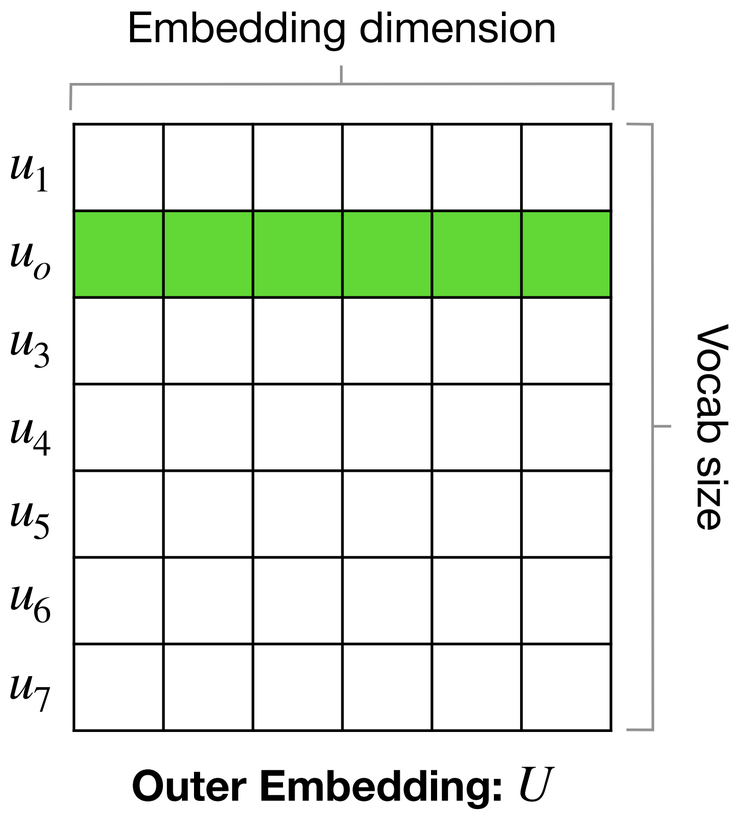
참조: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/slides/cs224n-2019-lecture01-wordvecs1.pdf

Word2Vec

Skip-gram



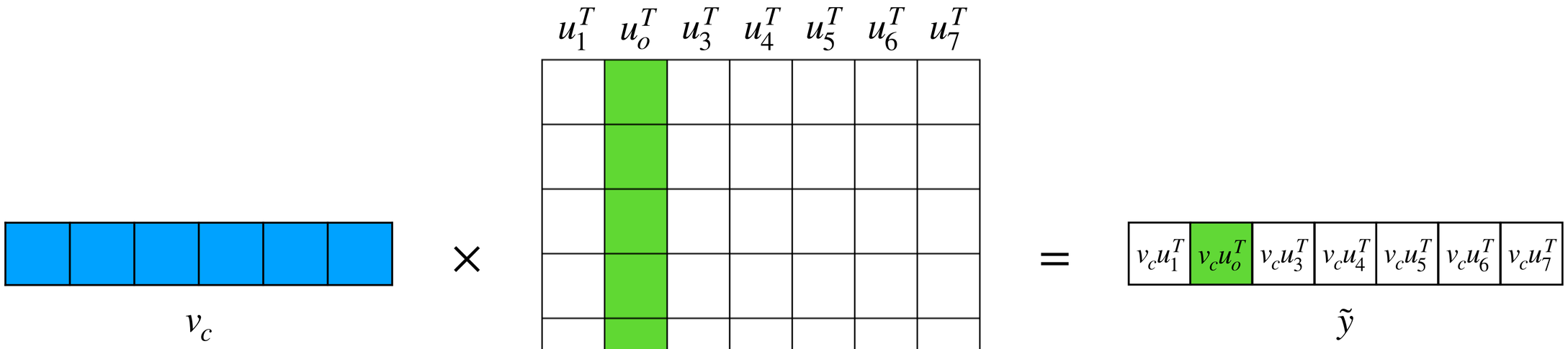
Center Embedding: V



Outer Embedding: U

Word2Vec

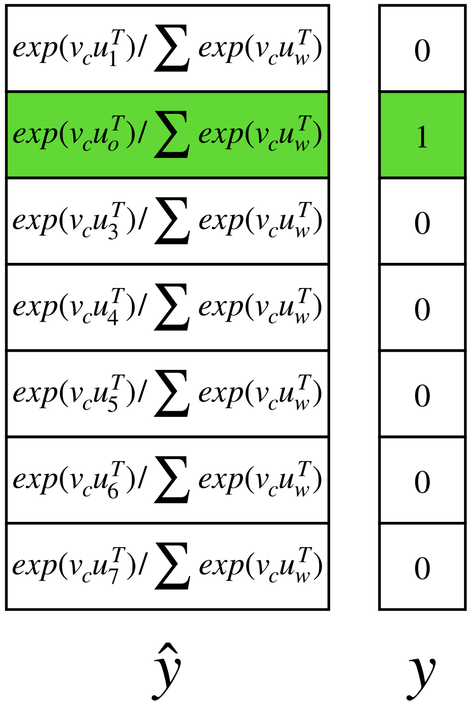
Skip-gram



Outer Embedding: U^T

Word2Vec

Skip-gram



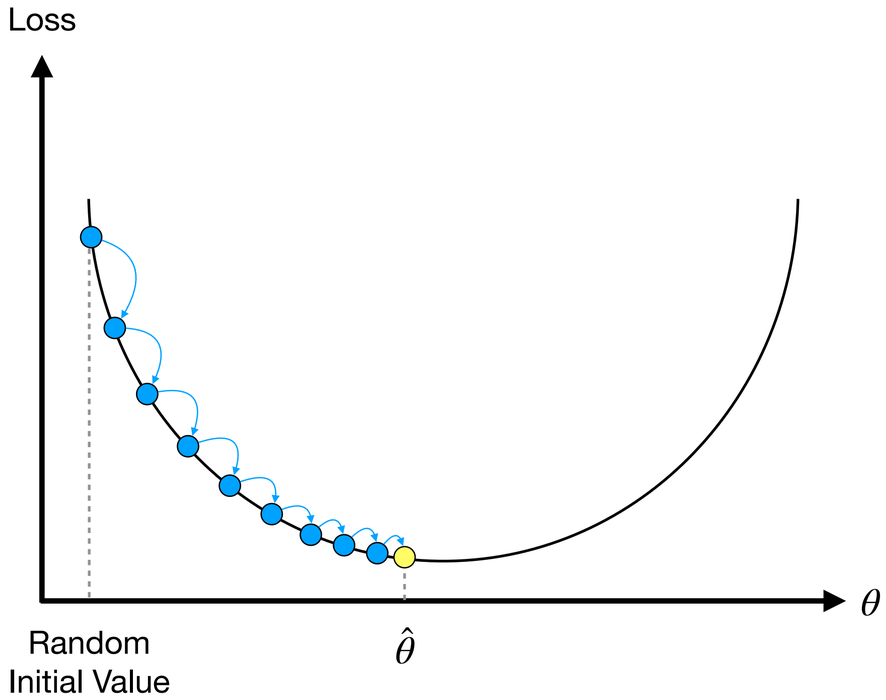
$$CE = -\sum_{w \in V} y_w \log \hat{y}_w$$

$$= -\log \hat{y}_o$$

$$= -\log \frac{exp(v_c u_o^T)}{\sum exp(v_c u_w^T)}$$

Word2Vec

Skip-gram



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

$$\theta_{new} = \theta_{old} - \alpha\nabla_{\theta}J(\theta)$$

모든 center word와 outer word의
loss 계산이 어려움

Stochastic gradient descent (SGD)

참조: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/slides/cs224n-2019-lecture01-wordvecs1.pdf

Word2Vec

Efficacy in training

계산량이 vocab size에 비례함
(vocab size: 10k ~ 1,000k)

$$\hat{y} = softmax(\tilde{y}) = \frac{exp(v_c \cdot u_o)}{\sum_{w \in V} exp(v_c \cdot u_w)}$$

- Solutions
  - Negative Sampling

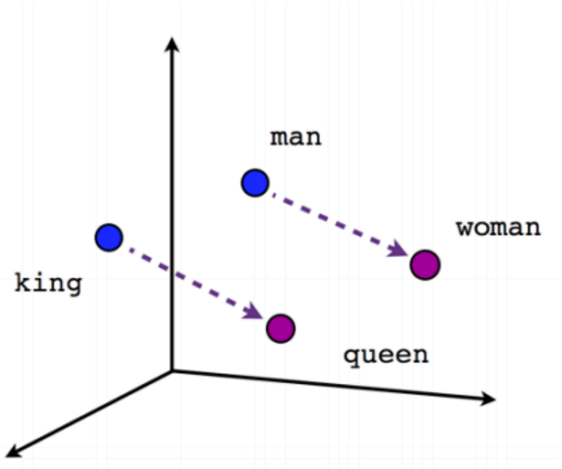
$$\sum_{w \in V} exp(v_c \cdot u_w) \quad \Rightarrow \quad \sum_{w \in \{o\} \cup S} exp(v_c \cdot u_w)$$

S: Negative Sample (small subset)

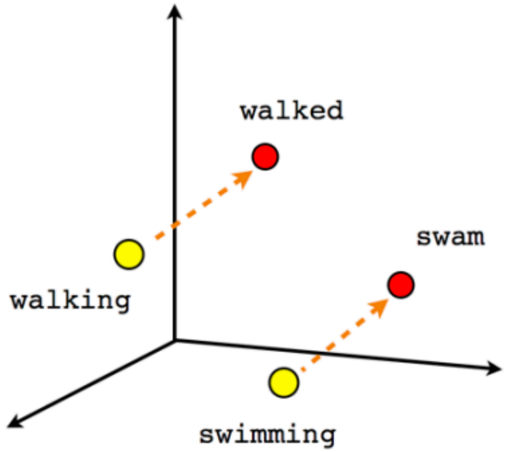
참조: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/slides/cs224n-2019-lecture01-wordvecs1.pdf

Word2Vec

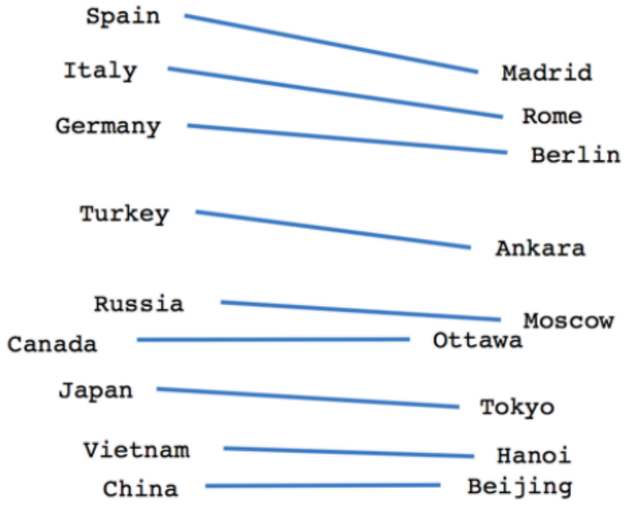
Vector relation



Male-Female



Verb tense



Country-Capital

$$\overrightarrow{v}_{king} - \overrightarrow{v}_{man} + \overrightarrow{v}_{woman} = \overrightarrow{v}_{queen}$$

참조: https://www.tensorflow.org/images/linear-relationships.png

Word2Vec

Evaluate Word Vectors

- Intrinsic
  - Evaluate on a specific/Intermediate sub task
  - Fast to compute
  - Help to understand that subsystem
  - Not clear if really helpful to real task

- Extrinsic
  - Evaluate on a real task
  - Can take a long time to compute accuracy
  - Unclear if the subsystem is the problem or its interaction or other subsystems
  - If replacing exactly one subsystem with another improves accuracy → winning

참조: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/slides/cs224n-2019-lecture02-wordvecs2.pdf

FastText

- Bojanowski et al. 2017 Facebook
- Mikolov et al.
- 단어 벡터를 학습하기 위한 framework

- Idea
  - Sub-word를 이용해서 Word2Vec을 개선
  - Sub-word 단위로 학습
  - 오타에 의한 OOV(Out of Vocabulary) 문제를 개선

참조: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/slides/cs224n-2019-lecture01-wordvecs1.pdf

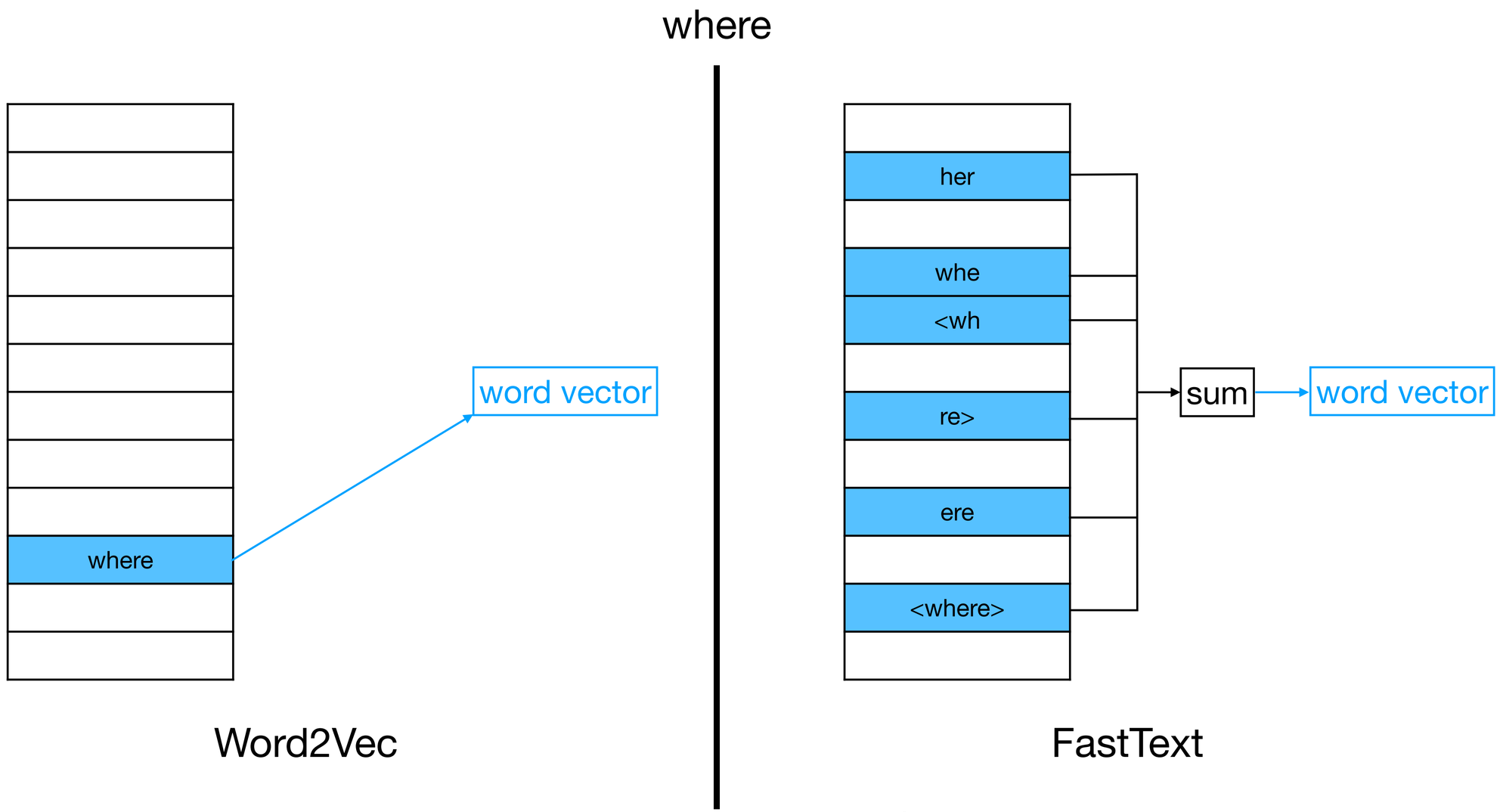
FastText

- 단어를 char n-gram으로 표현
  - where: <wh, whe, her, ere, re> <where>
- 각각의 n-gram은 벡터로 표현 됨
- 단어의 벡터는 n-gram 벡터의 합으로 표현 됨

Sum of vectors for char n-gram

참조: https://github.com/yandexdataschool/nlp_course/blob/2019/resources/slides/nlp19_01_word_embeddings.pdf

Word2Vec vs FastText



참조: https://github.com/yandexdataschool/nlp_course/blob/2019/resources/slides/nlp19_01_word_embeddings.pdf

FastText

book → <bo, boo, ook, ok>, <book>

shelf → <sh, she, hel, elf, lf>, <shelf>

bookshelf → <bo, boo, ook, oks, ksh, she, hel, elf, lf>, <bookshelf>

Vocabulary에 없는 단어라도 공통으로 존재하는 부분의 벡터를 이용하면 비슷한 단어 벡터를 얻을 수 있음
오타 또한 같은 원리로 비슷한 벡터를 얻을 수 있음

감사합니다.