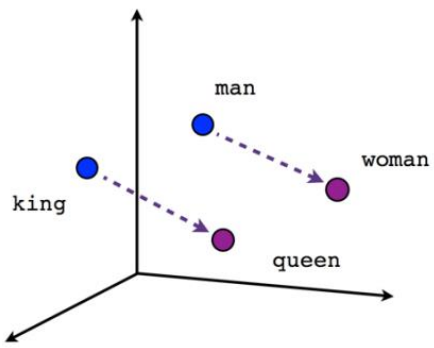
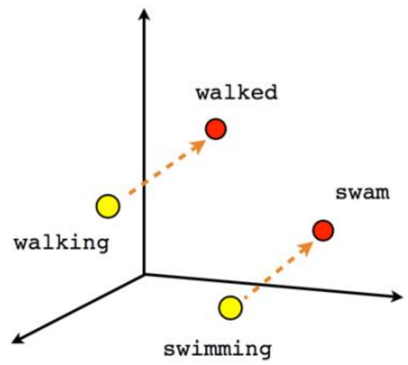
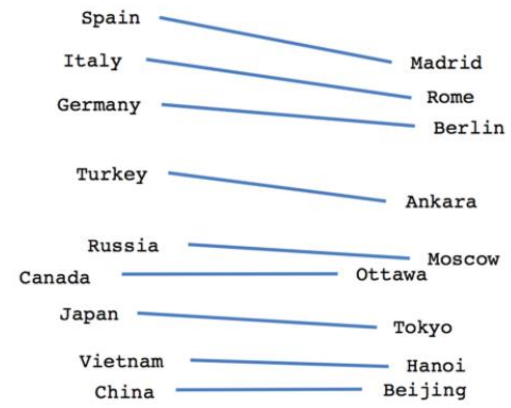


Male-Female



Verb tense



Country-Capital

Lecture 5: Text Representation II
Distributed Representations

Pilsung Kang

School of Industrial Management Engineering

Korea University

AGENDA

01 Word-level: NNLM

02 Word-level: Word2Vec

03 Word-level: GloVe

04 Word-level: Fasttext

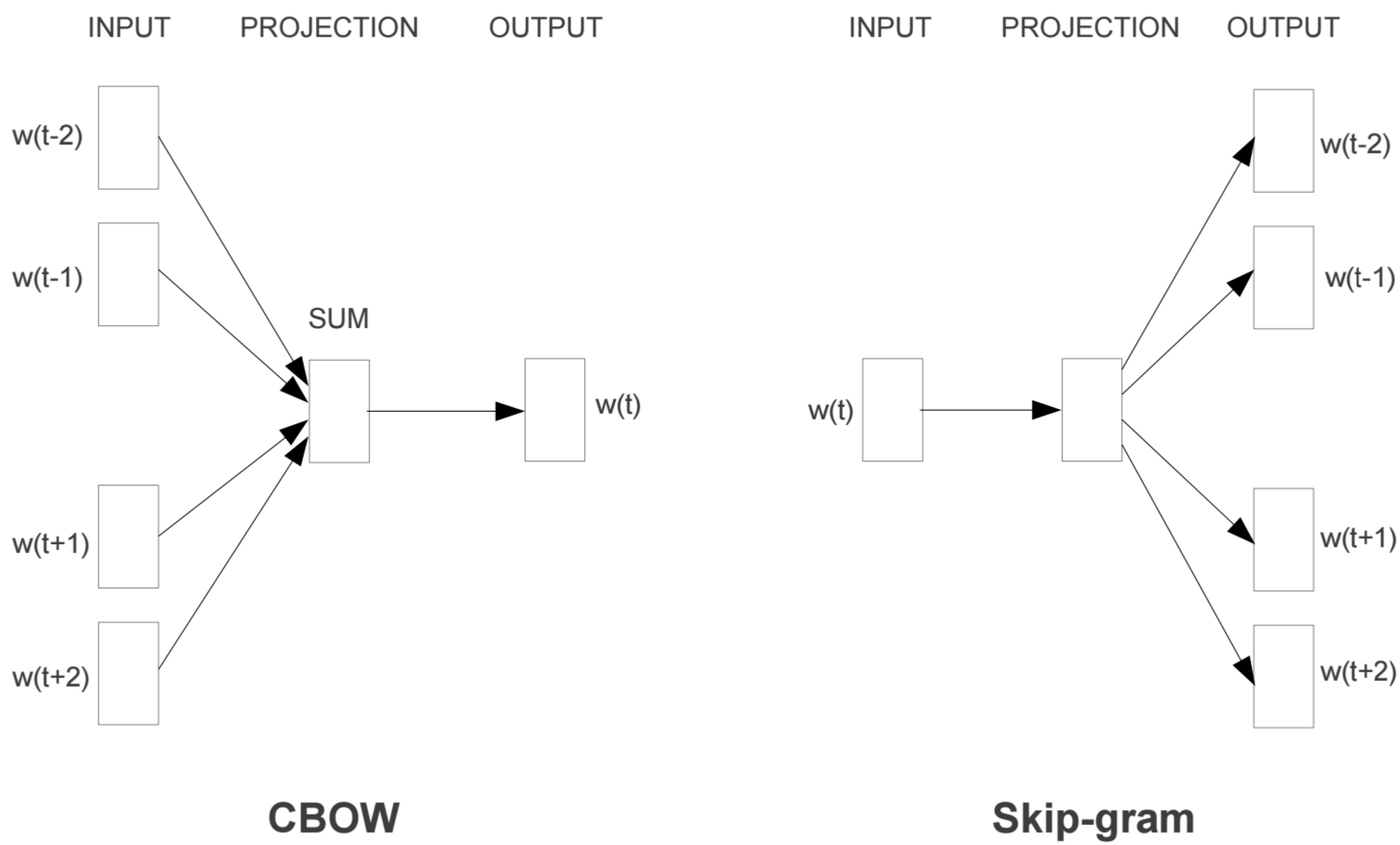
05 Sentence/Paragraph/Document-level

06 More Things to Embed?

Word2Vec

Mikolov et al. (2013)

- Two Architectures
  - ✓ Continuous bag-of-words (CBOW) vs. Skip-gram



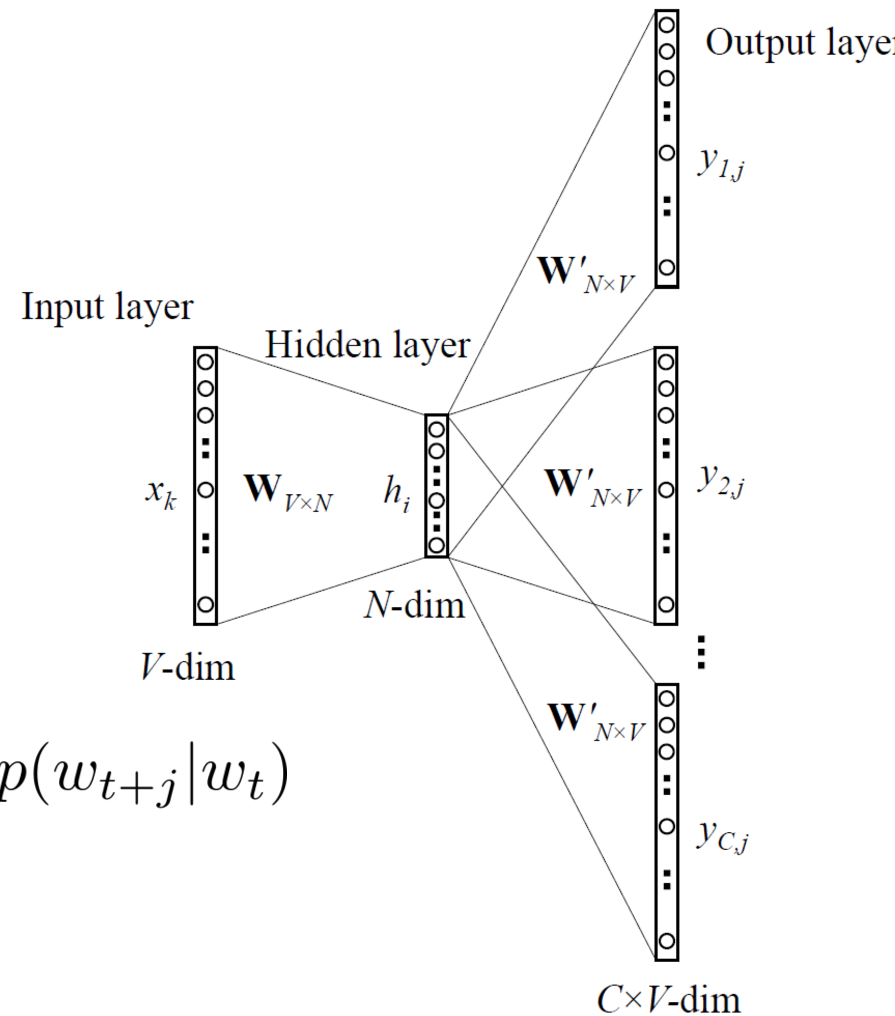
Word2Vec

Rong (2014)

- Learning representations: Skip-gram approach
  - ✓ Predict surrounding words in a window of length m of every word
- Objective function
  - ✓ Maximize the log probability of any context word given the current center word

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

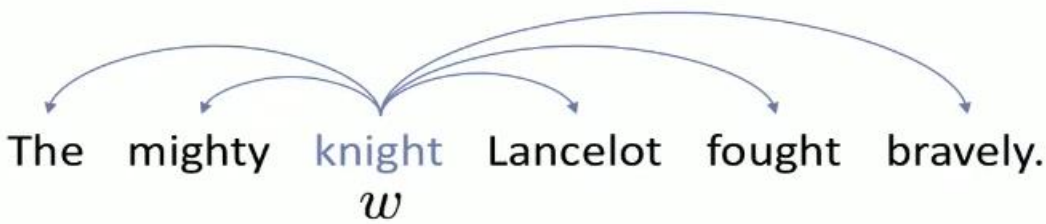
  - ✓ where θ represents all variables we optimize



Word2Vec

Bojanowski (2016)

- Skip-gram model



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

- Model probability of a context word given a word

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

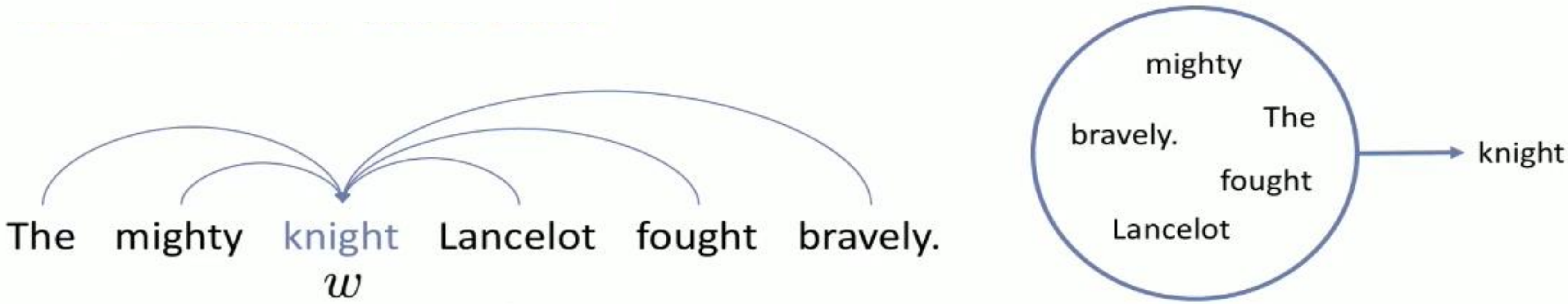
$$p(c|w) = \frac{e^{x_w^\top v_c}}{\sum_{k=1}^{K} e^{x_w^\top v_k}}$$

- Word vectors $x_w \in \mathbb{R}^d$

Word2Vec

Bojanowski (2016)

- CBOW model



- Model probability of a word given a context

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

classifier for word w: v_w

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

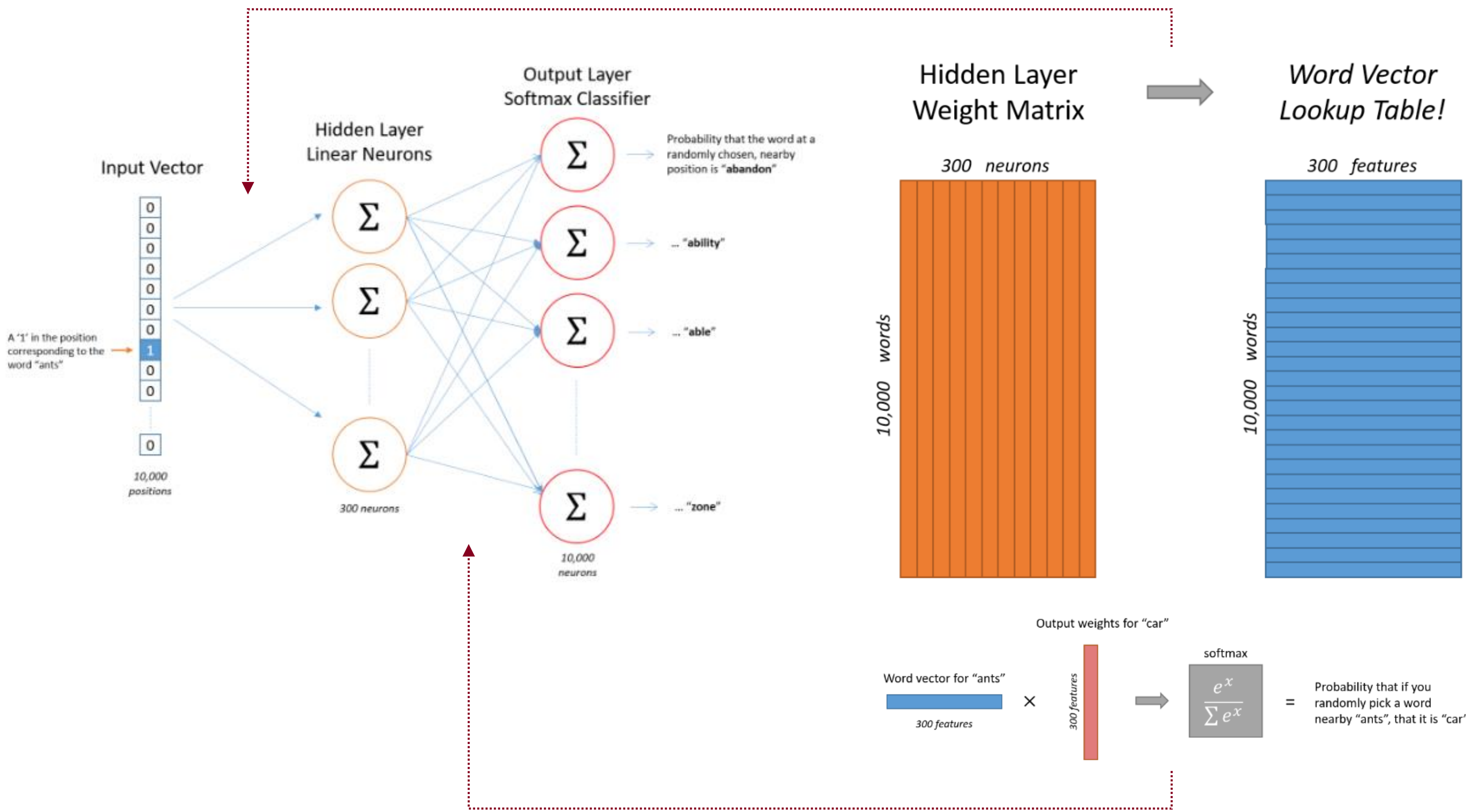
- Continuous Bag Of Words

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

Word2Vec

McCormick

- Another architecture explanation



Word2Vec

- For simplicity, we use the following notation instead of $p(w_{t+j}|w_t)$

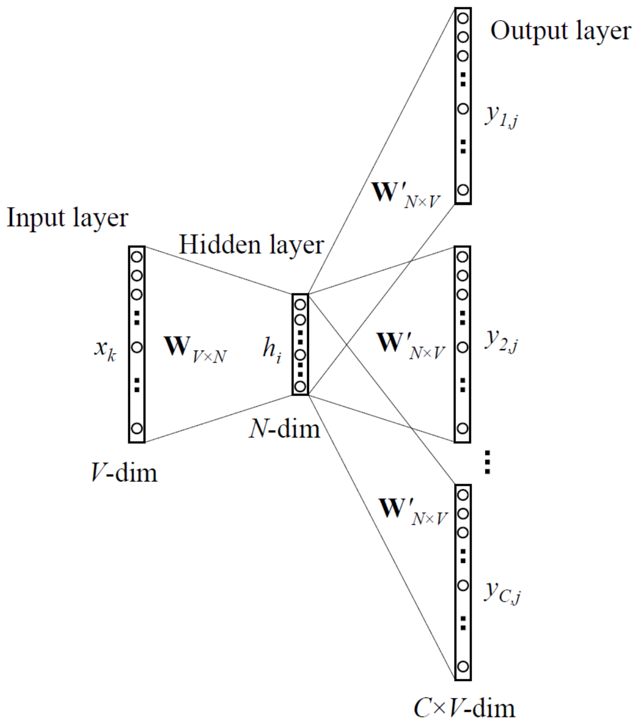
$$p(o|c) = \frac{exp(u_o^T v_c)}{\sum_{w=1}^{W} exp(u_w^T v_c)}$$

✓ where o is the outside (output) word id, c is the center word id, u and v are "outside" and "center" vectors of o and c

✓ Every word has two vectors!

- v is a row of matrix W
- u is a column of matrix W'

✓ Use W' = W^T in practice for efficient computation



Word2Vec

- Learning parameters with Gradient Ascent

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

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

✓ Compute the gradient

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

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

Word2Vec

- Learning parameters with Gradient Ascent
  - ✓ For chunk A

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

  - ✓ For chunk B

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

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

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

Word2Vec

- Learning parameters with Gradient Ascent

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

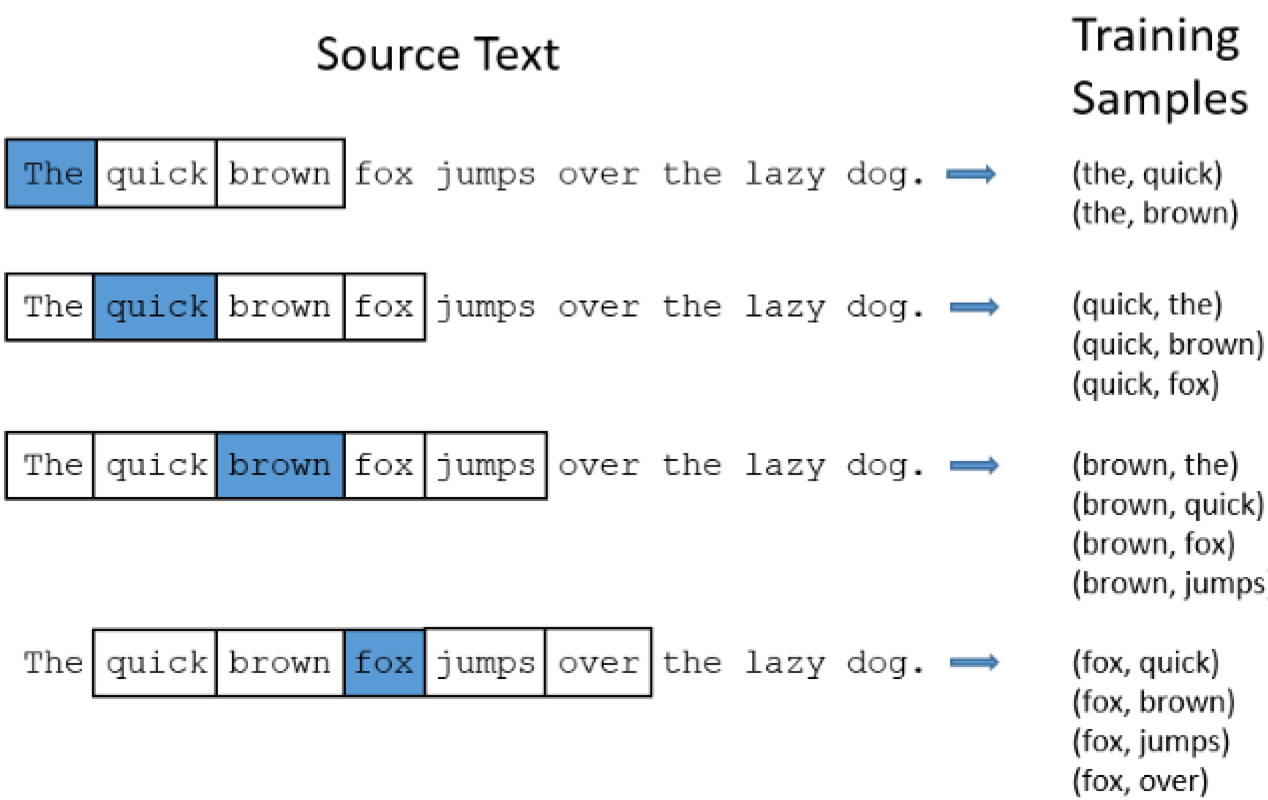
- Update the weight vector

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

Word2Vec

McCormick

- Learning strategy
  - ✓ Do not use all nearby words, but one per each training



Word2Vec

- The number of weights to be trained: 2 x V x N (Huge network!)
  - ✓ Word pairs and phrases
    - Treating common word pairs or phrases as single "word"
  - ✓ Subsampling frequent words
    - To decrease the number of training examples
    - The probability of word w_i being removed

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

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

- The number of weights to be trained: 2 x V x N (Huge network!)
  - ✓ Negative sampling
    - Instead of updating the weights associated with all output words, update the weight of a few (5-20) words

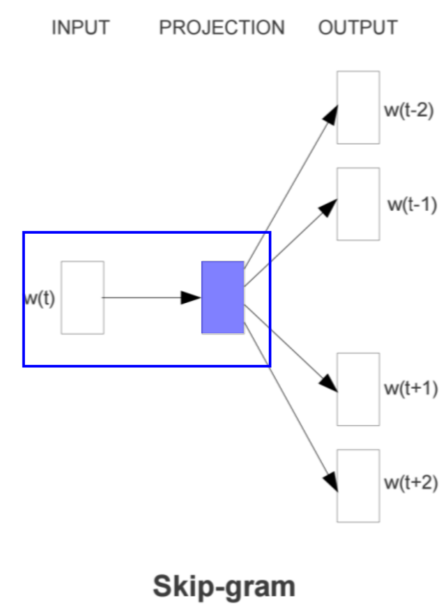
$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log p(w_{t+j}|w_t) = \frac{1}{T}\sum_{t=1}^{T} J_t(\theta)$$

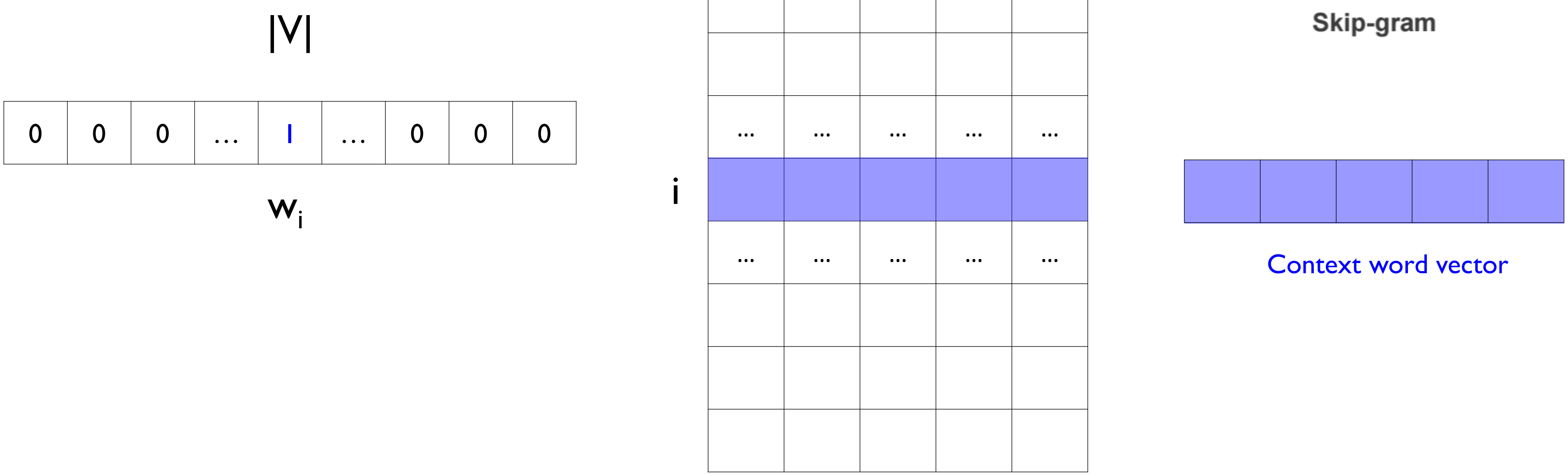
$$J_t(\theta) = \log\sigma(u_o^T v_c) + \boxed{\sum_{i=1}^{k} E_{i\sim P(w)}\left[\log\sigma(-u_i^T v_c)\right]}$$

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

Word2Vec

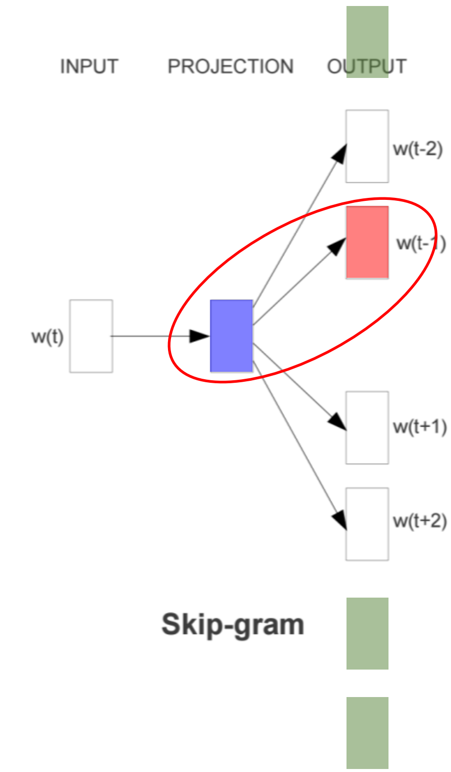
- Negative Sampling Example

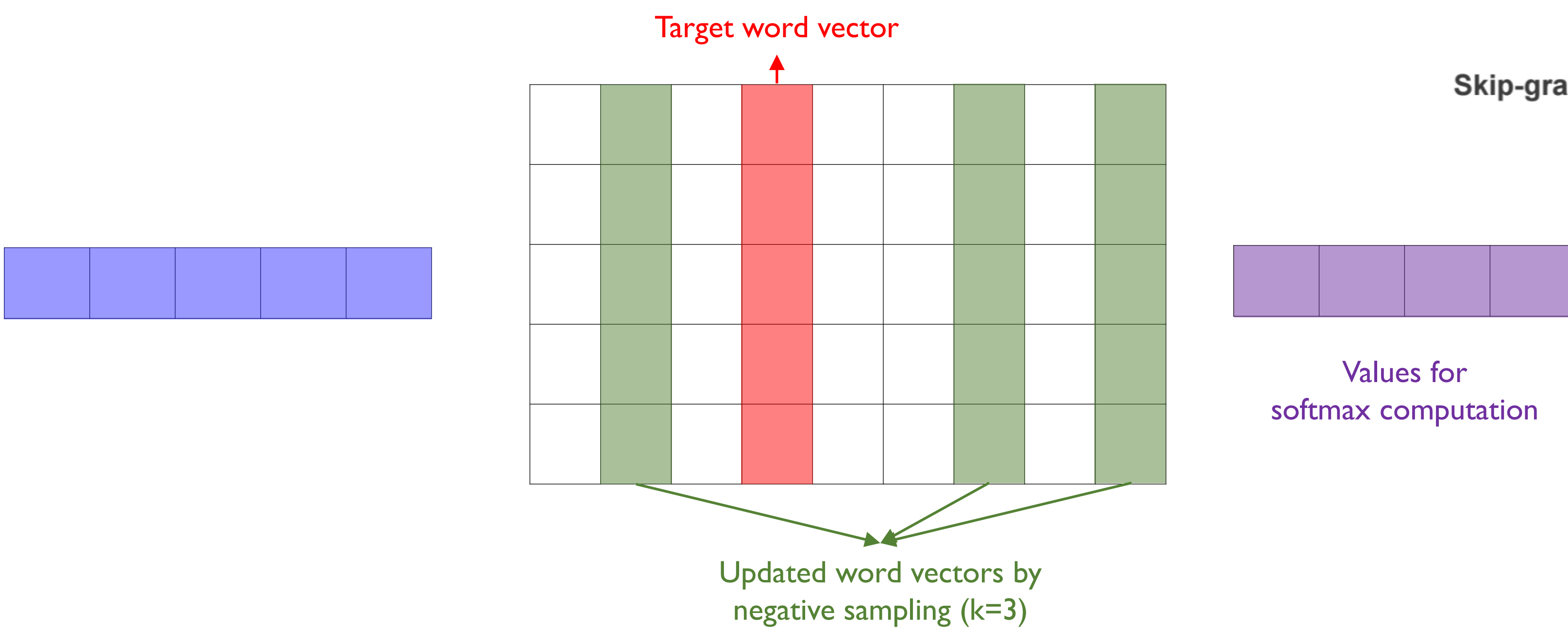


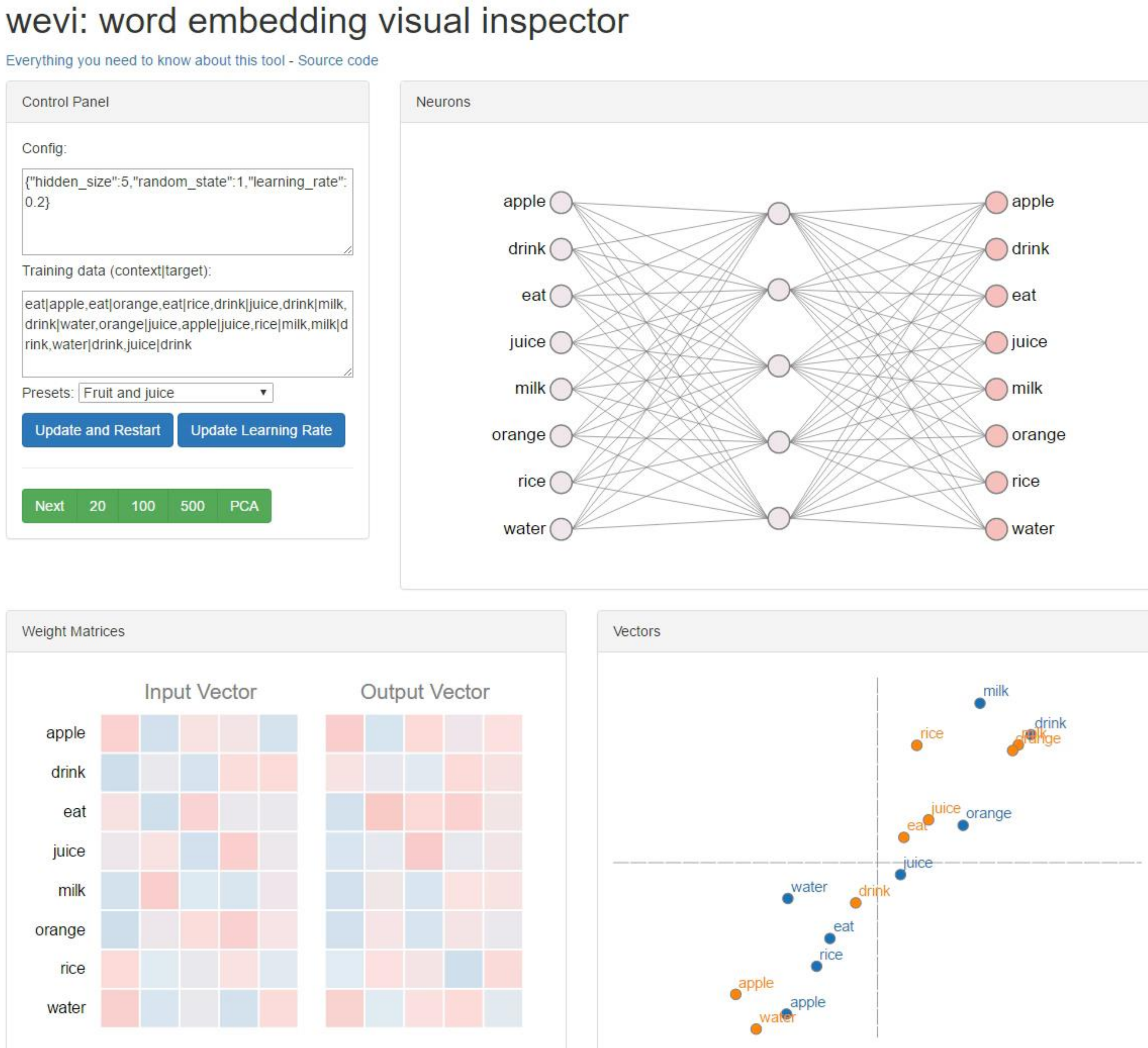


Word2Vec

- Negative Sampling Example
  - ✓ No. of 0 values for softmax computation is reduced from |V| to (k+1)

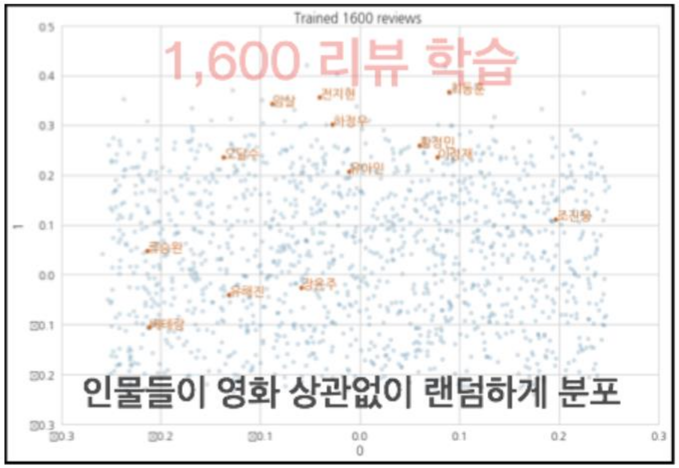


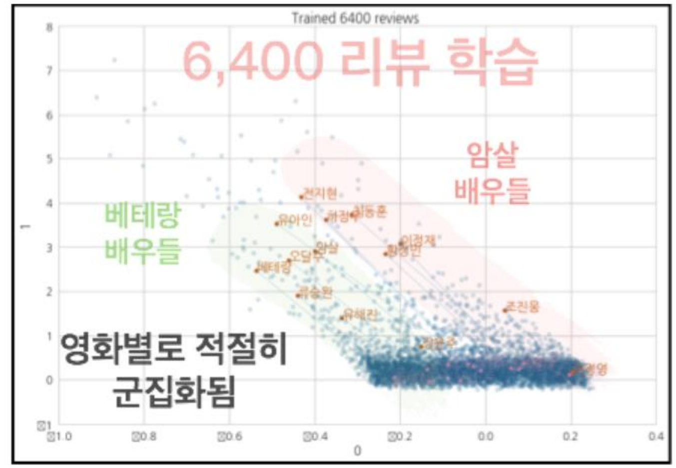


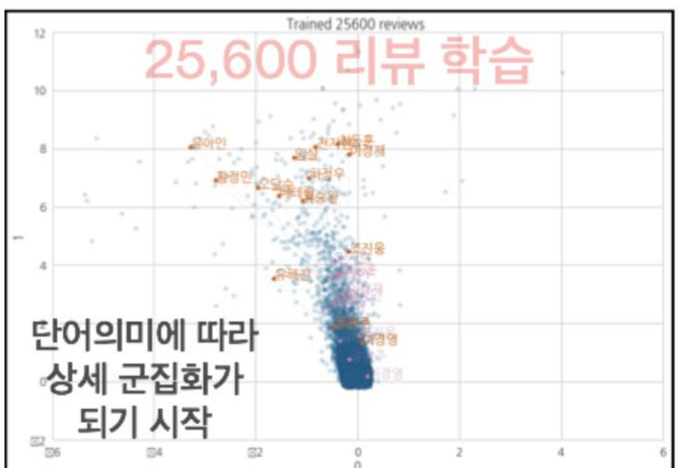

wevi: word embedding visual inspector
Everything you need to know about this tool - Source code
Control Panel
Config:
{"hidden_size":5,"random_state":1,"learning_rate":0.2}
Training data (context|target):
eat|apple,eat|orange,eat|rice,drink|juice,drink|milk,drink|water,orange|juice,apple|juice,rice|milk,milk|drink,water|drink,juice|drink
Presets: Fruit and juice
Update and Restart
Update Learning Rate
Next
20
100
500
PCA
Neurons
apple
drink
eat
juice
milk
orange
rice
water
Weight Matrices
Input Vector
Output Vector
Vectors
milk
drink
rice
juice
orange
eat
water
apple

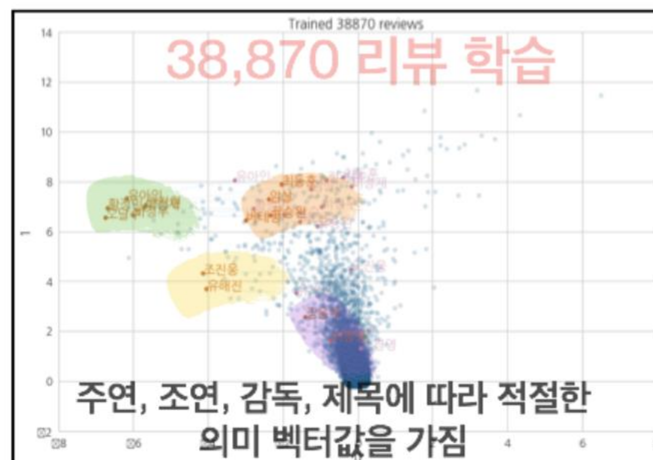
Word2Vec: Example

- Trained with movie reviews





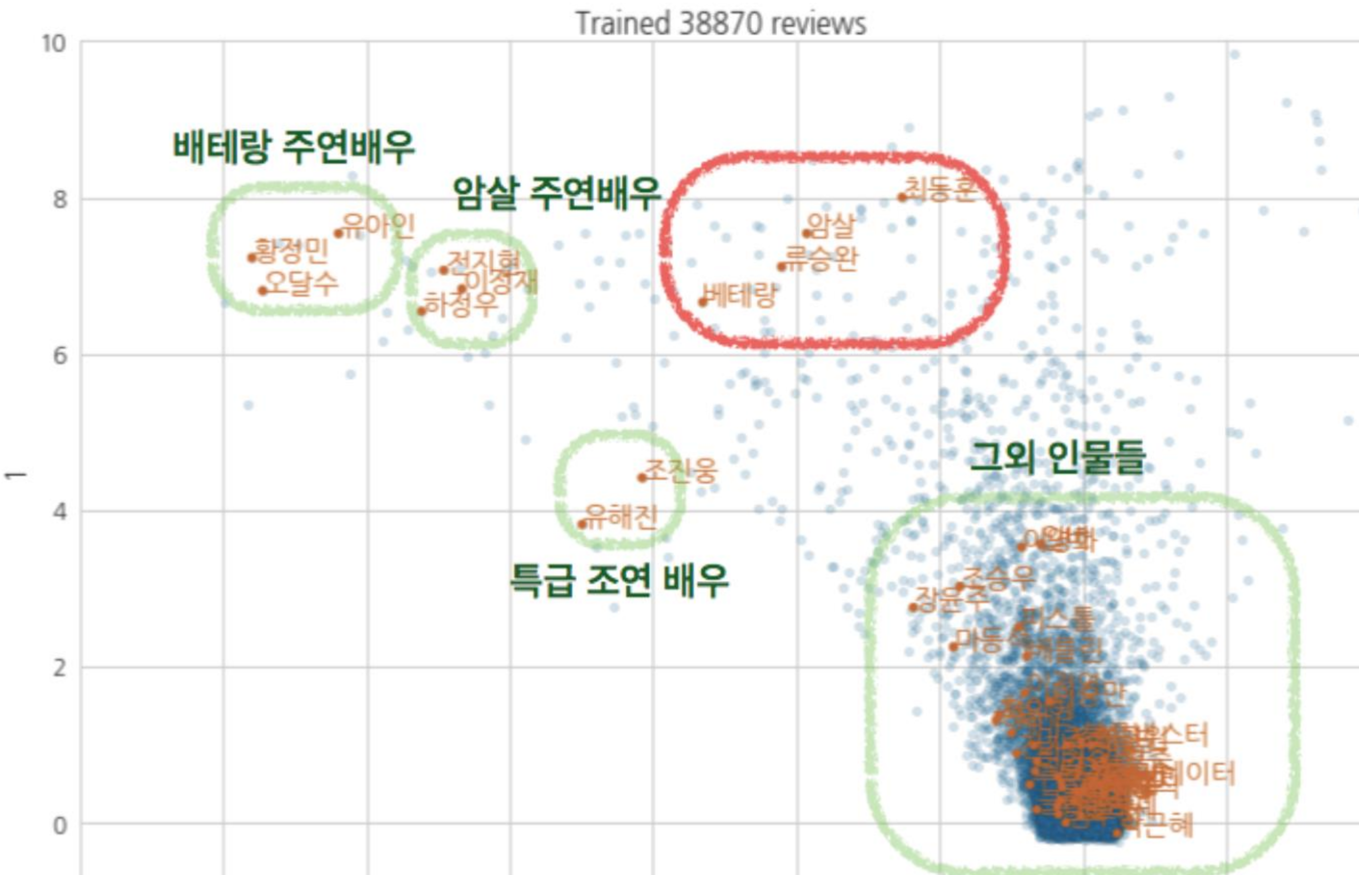




https://brunch.co.kr/@goodvc78/16

Word2Vec: Example

- Trained with movie reviews



Vec[류승완] - Vec[베테랑] ≒ Vec[최동훈] - Vec[암살]

https://brunch.co.kr/@goodvc78/16


ANY
questions?