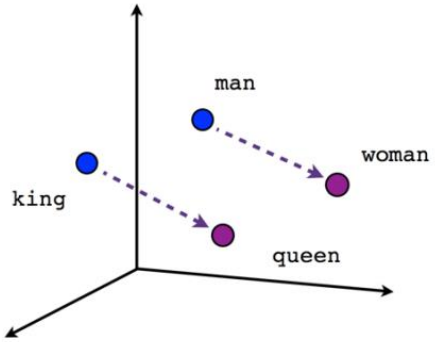
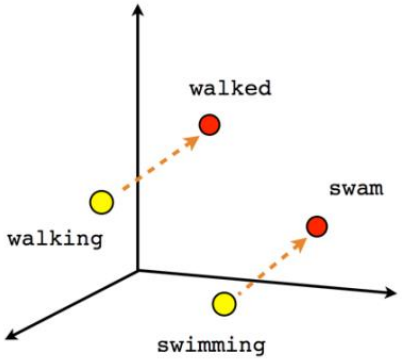
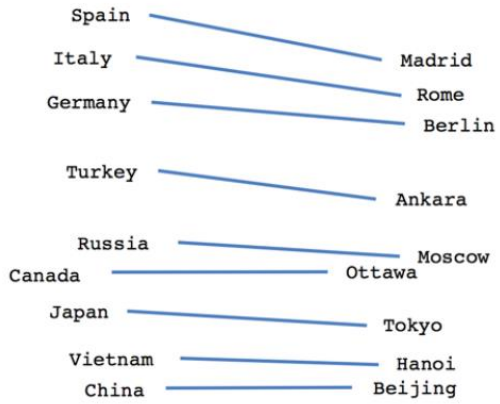


Male-Female



Verb tense



Country-Capital

Lecture 5: Text Representation II Distributed Representations

Pilsung Kang

School of Industrial Management Engineering

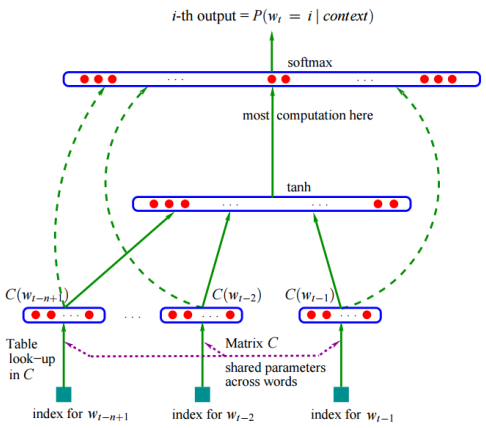
Korea University

AGENDA

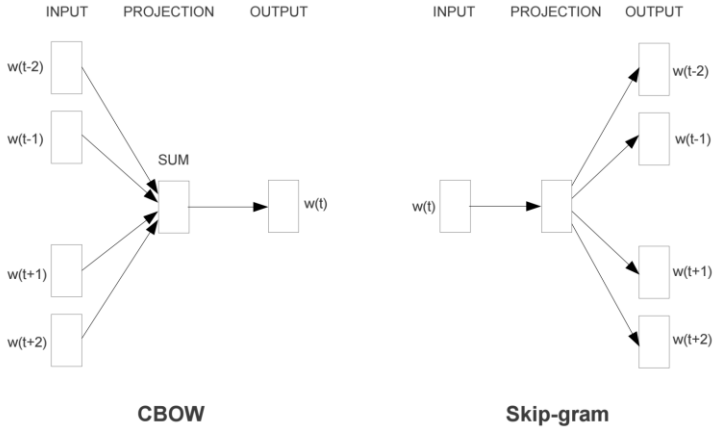
Distributed Representation: Word Embedding

- Word Embedding
  - ✓ The purpose of word embedding is to map the words in a language into a vector space so that semantically similar words are located close to each other.
  - ✓ Hypothetically, the number of token in English is estimated about 13M, there exist a d (< 13M) dimensional optimal space that can embed the meaning of all words.

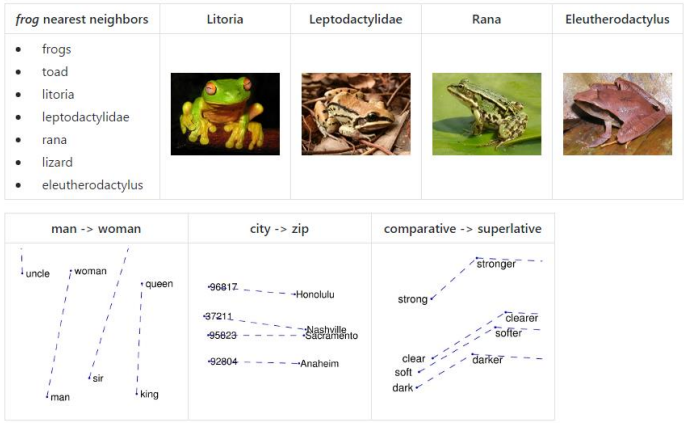
NNLM



Word2Vec



GloVe



Distributed Representation: Word Embedding

- Word vectors: one-hot vector
  - ✓ The most simple & intuitive representation

$$w^{aardvark} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{a} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{at} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \ldots w^{zebra} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

  - ✓ Can make a vector representation, but similarities between words cannot be preserved.

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

$$(w^{hotel})^\top w^{motel} = (w^{hotel})^\top w^{cat} = 0$$

Distributed Representation: Word Embedding

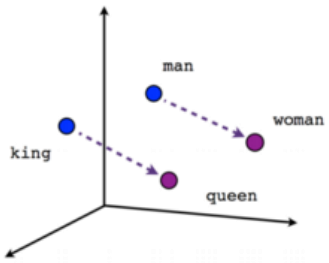
- Word vectors: distributed representation
  - ✓ A parameterized function mapping words in some language to a certain dimensional vectors

$$W : \text{words} \rightarrow \mathbb{R}^n$$

$$W(\text{“cat”}) = (0.2,\ \text{-}0.4,\ 0.7,\ \ldots)$$

$$W(\text{“mat”}) = (0.0,\ 0.6,\ \text{-}0.1,\ \ldots)$$

- Interesting feature of word embedding
  - ✓ Semantic relationship between words can be preserved



Male-Female

Verb tense

Country-Capital

https://www.tensorflow.org/versions/ro.7/tutorials/word2vec/index.html

Neural Network Language Model (NNLM)

Bengio et al. (2003)

- Purpose
  - ✓ Fighting the curse of dimensionality with distributed representations

> ✓ Associate with each word in the vocabulary a **distributed word feature vector** (a real valued vector in R^m),
> ✓ Express the joint **probability function** of word sequences in terms of the feature vectors of these words in the sequence,
> ✓ Learn simultaneously the **word feature vectors** and the parameters of that **probability function**

Neural Network Language Model (NNLM)

- Why it works?
  - ✓ If we knew that `dog` and `cat` played similar roles (semantically and synthetically), and similarity for (`the`, `a`), (`bedroom`, `room`), (`is`, `was`), (`running`, `walking`), we could naturally generalize from

```
The cat is walking in the bedroom
```

to

```
A dog was running in a room
The cat is running is a room
A dog is walking in a bedroom
The dog was waling in the room
...
```

Neural Network Language Model (NNLM)

Kim et al. (2016)

- Comparison with Count-based Language Models
  - ✓ Count-based Language Models

By the chain rule, any distribution can be factorized as

$$p(w_1, \ldots, w_T) = \prod_{t=1}^{T} p(w_t | w_1, \ldots, w_{t-1})$$

Count-based n-gram language models make a Markov assumption:

$$p(w_t | w_1, \ldots, w_t) \approx p(w_t | w_{t-n}, \ldots, w_{t-1})$$

Need smoothing to deal with rare n-grams.

Neural Network Language Model (NNLM)

- Language Model Example

Neural Network Language Model (NNLM)

Kim et al. (2016)

- Comparison with Count-based Language Models
  - ✓ NNLM
    - Represent words as dense vectors in $\mathbb{R}^n$ (word embeddings).

      $\mathbf{w}_t \in \mathbb{R}^{|\mathcal{V}|}$: One-hot representation of word $\in \mathcal{V}$ at time t
      $\Rightarrow \mathbf{x}_t = \mathbf{X}\mathbf{w}_t$: Word embedding ($\mathbf{X} \in \mathbb{R}^{n \times |\mathcal{V}|}, n < |\mathcal{V}|$)

    - Train a neural net that composes history to predict next word.

$$p(w_t = j | w_1, \ldots, w_{t-1}) = \frac{\exp(\mathbf{p}^j \cdot g(\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}) + q^j)}{\sum_{j' \in \mathcal{V}} \exp(\mathbf{p}^{j'} \cdot g(\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}) + q^{j'})}$$
$$= \text{softmax}(\mathbf{P} g(\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}) + \mathbf{q})$$

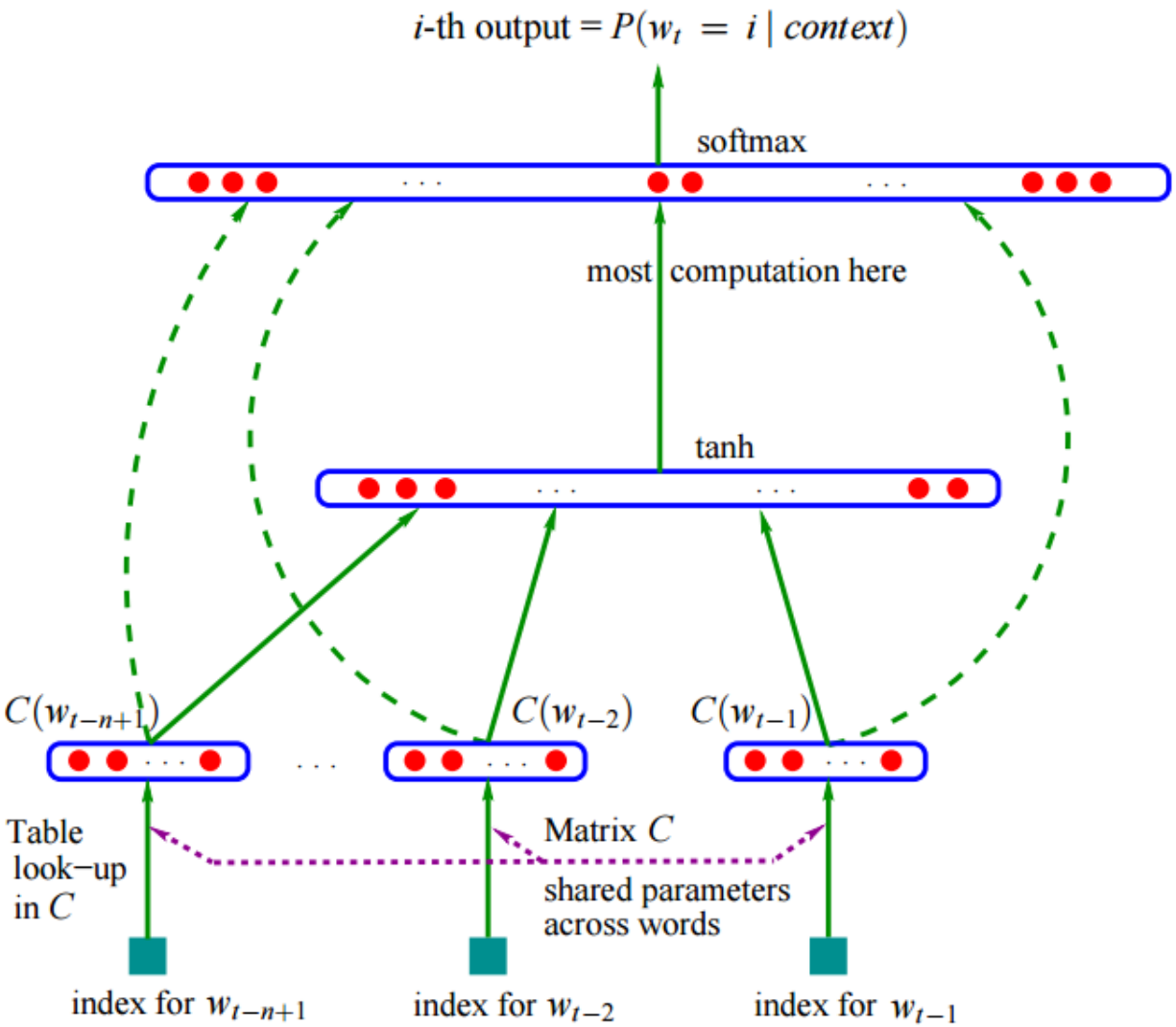
$\mathbf{p}^j \in \mathbb{R}^m, q^j \in \mathbb{R}$: Output word embedding/bias for word $j \in \mathcal{V}$

g : Composition function

Neural Network Language Model (NNLM)

Bengio et al. (2003)

- Learning NNLM



Neural Network Language Model (NNLM)

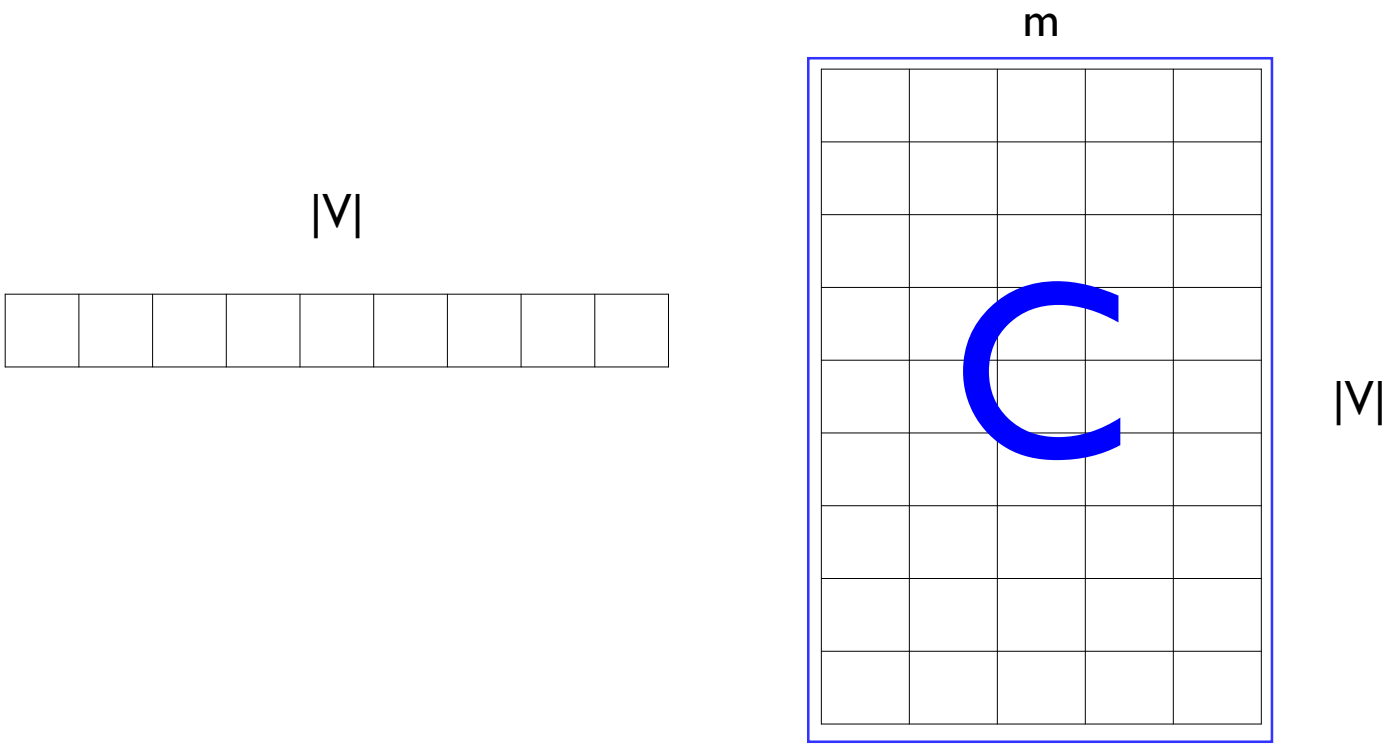
Bengio et al. (2003)

- Learning NNLM
  - ✓ The objective is to learn a good model $f(w_t, \cdots w_{t-n+1}) = \hat{P}(w_t|w_1^{t-1})$, in the sense that it gives high out-of-sample likelihood
  - ✓ Two constraints
    - For any choice of $w_1^{t-1}, \sum_{i=1}^{|V|} f(i, w_{t-1} \cdots w_{t-n+1}) = 1$ (어떤 조건에서도 이후 단어들이 생성될 확률의 총 합은 1)
    - f ≥ 0 (각 단어가 생성될 확률은 0보다 크거나 같아야 함)

Neural Network Language Model (NNLM)

Bengio et al. (2003)

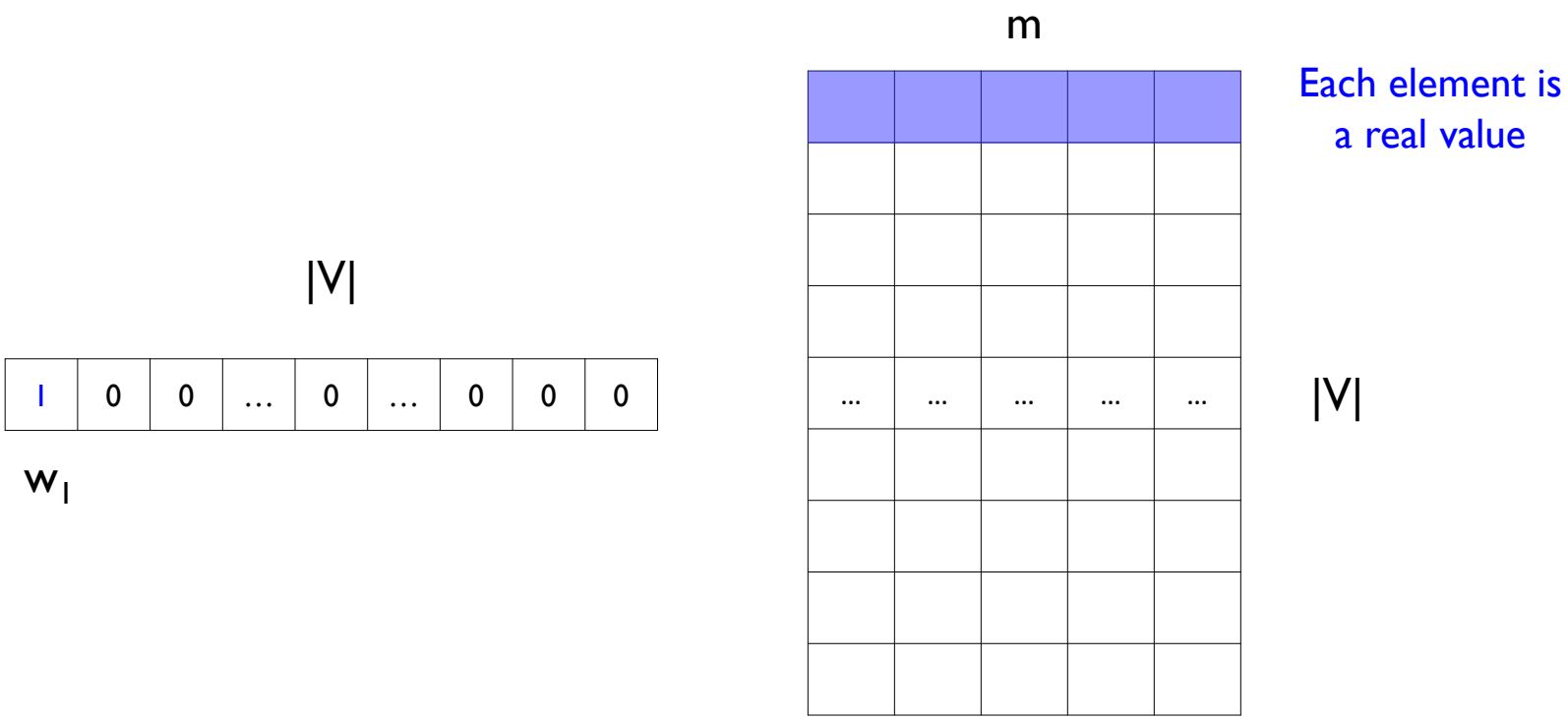
- Learning NNLM
  - ✓ Decompose the function $f(w_t, \cdots w_{t-n+1}) = \hat{P}(w_t|w_1^{t-1})$ in two parts:
    - A mapping *C*, a.k.a the lookup table, from any element *i* of *V* to a real vector $C(i) \in R^m$, it represents the distributed feature vectors associated with each word in the vocabulary. *C* is represented by a $|V| \times m$ matrix of free parameters



Neural Network Language Model (NNLM)

Bengio et al. (2003)

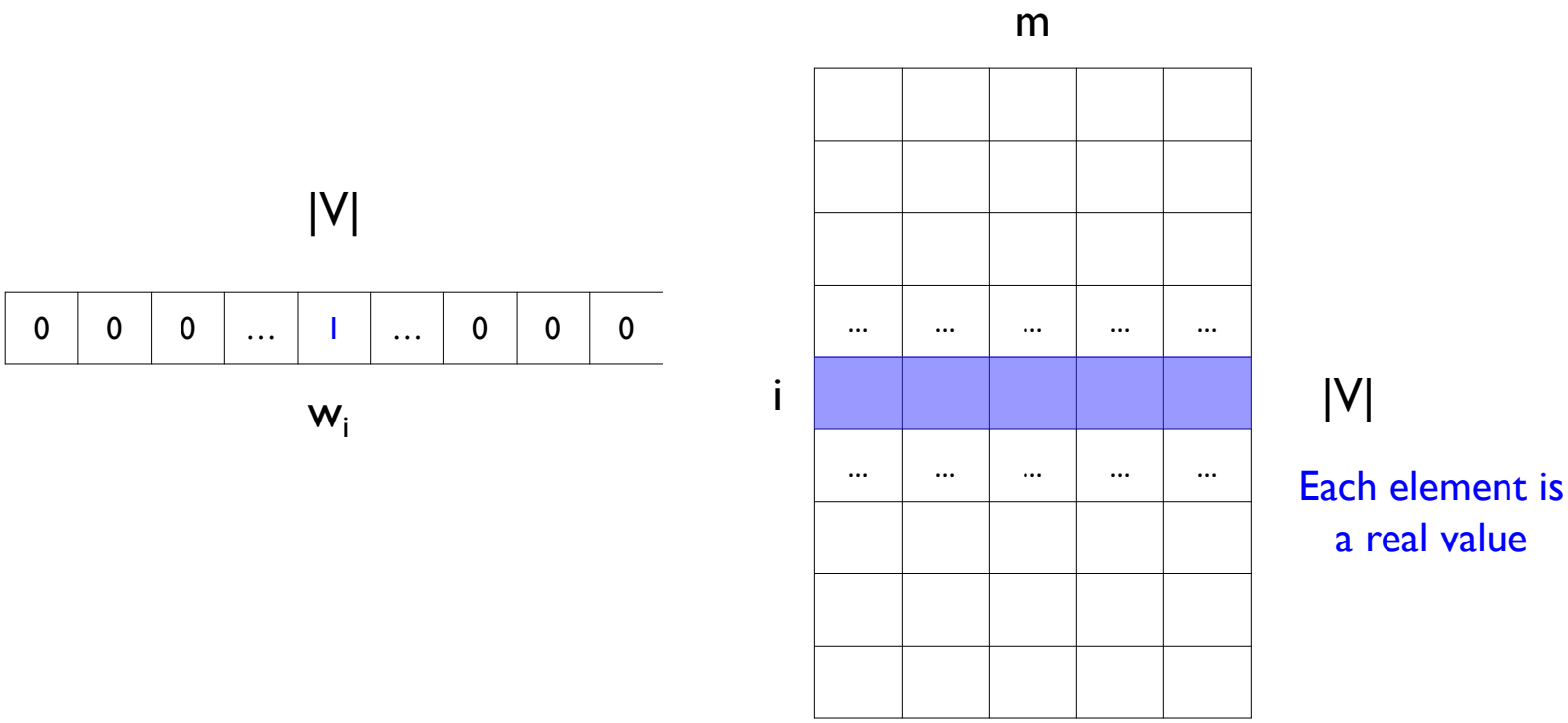
- Learning NNLM
  - ✓ Decompose the function $f(w_t, \cdots w_{t-n+1}) = \hat{P}(w_t|w_1^{t-1})$ in two parts:
    - A mapping *C*, a.k.a the lookup table, from any element *i* of *V* to a real vector $C(i) \in R^m$, it represents the distributed feature vectors associated with each word in the vocabulary. *C* is represented by a $|V| \times m$ matrix of free parameters



Neural Network Language Model (NNLM)

Bengio et al. (2003)

- Learning NNLM
  - ✓ Decompose the function $f(w_t, \cdots w_{t-n+1}) = \hat{P}(w_t|w_1^{t-1})$ in two parts:
    - A mapping *C*, a.k.a the lookup table, from any element *i* of *V* to a real vector $C(i) \in R^m$, it represents the distributed feature vectors associated with each word in the vocabulary. *C* is represented by a $|V| \times m$ matrix of free parameters



Neural Network Language Model (NNLM)

Bengio et al. (2003)

- Learning NNLM
  - ✓ Decompose the function $f(w_t, \cdots w_{t-n+1}) = \hat{P}(w_t|w_1^{t-1})$ in two parts:
    - The probability function over words, expressed with *C*: a function g maps an input sequence of feature vectors for words in context, $(C(w_{t-n+1}), \cdots C(w_{t-1}))$, to a conditional probability distribution over words in V for the next word w_t. The output of g is a vector whose i^{th} element estimates the probability $\hat{P}(w_t = i|w_1^{t-1})$

g(준다|너에게, 나의 입술을, 처음으로) = ?

g(지운다|너에게, 나의 입술을, 처음으로) = ?

g(맡긴다|너에게, 나의 입술을, 처음으로) = ?

Neural Network Language Model (NNLM)

Bengio et al. (2003)

- Learning NNLM

$$f(i, w_{t-1} \cdots w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$$

✓ The function f is a composition of these two mappings (C and g), with C being shared across all the words in the context.

- The parameters of the mapping C are simply the feature vectors themselves, represented by a $|V| \times m$ matrix C whose row i is the feature vector C(i) for word i
- The function g may be implemented by a feed-forward or recurrent neural network or another parameterized function, with parameters ω.

✓ Training is done by maximizing the penalized log-likelihood of the training corpus

$$L = \frac{1}{T} \sum_t \log f(i, w_{t-1} \cdots w_{t-n+1}; \theta) + R(\theta)$$

Neural Network Language Model (NNLM)

Bengio et al. (2003)

- Learning NNLM
  - ✓ The neural network has one hidden layer beyond the word features mapping, and optionally, direct connections from the word features to the output.
  - ✓ Computation of the output layer

$$\hat{P}(w_t|w_{t-1},\cdots,w_{t-n+1}) = \frac{\exp(y_{w_t})}{\sum_i \exp(y_i)}$$

$$y = b + Wx + U \cdot tanh(d + Hx)$$

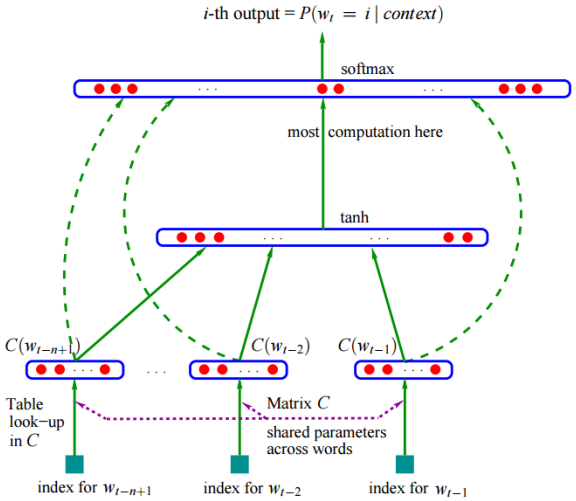
  - W is optionally zero (no direct connections from the input to the output)
  - x is the word features layer activation vector $x = (C(w_{t-1}),\cdots,C(w_{t-n+1}))$
  - h is the number of hidden units

Neural Network Language Model (NNLM)

Bengio et al. (2003)

- Learning NNLM
  - ✓ The free parameters of the model

$$y = b + Wx + U \cdot tanh(d + Hx)$$



  - the output bias b (|V| elements)
  - the hidden layer biases d (with h elements)
  - the hidden-to-output weights U (a |V| by h matrix)
  - the word features to output weights W (a |V| by (n-1)m matrix)
  - the hidden layer weight H (a h by (n-1)m matrix)

  - ✓ Stochastic gradient ascent

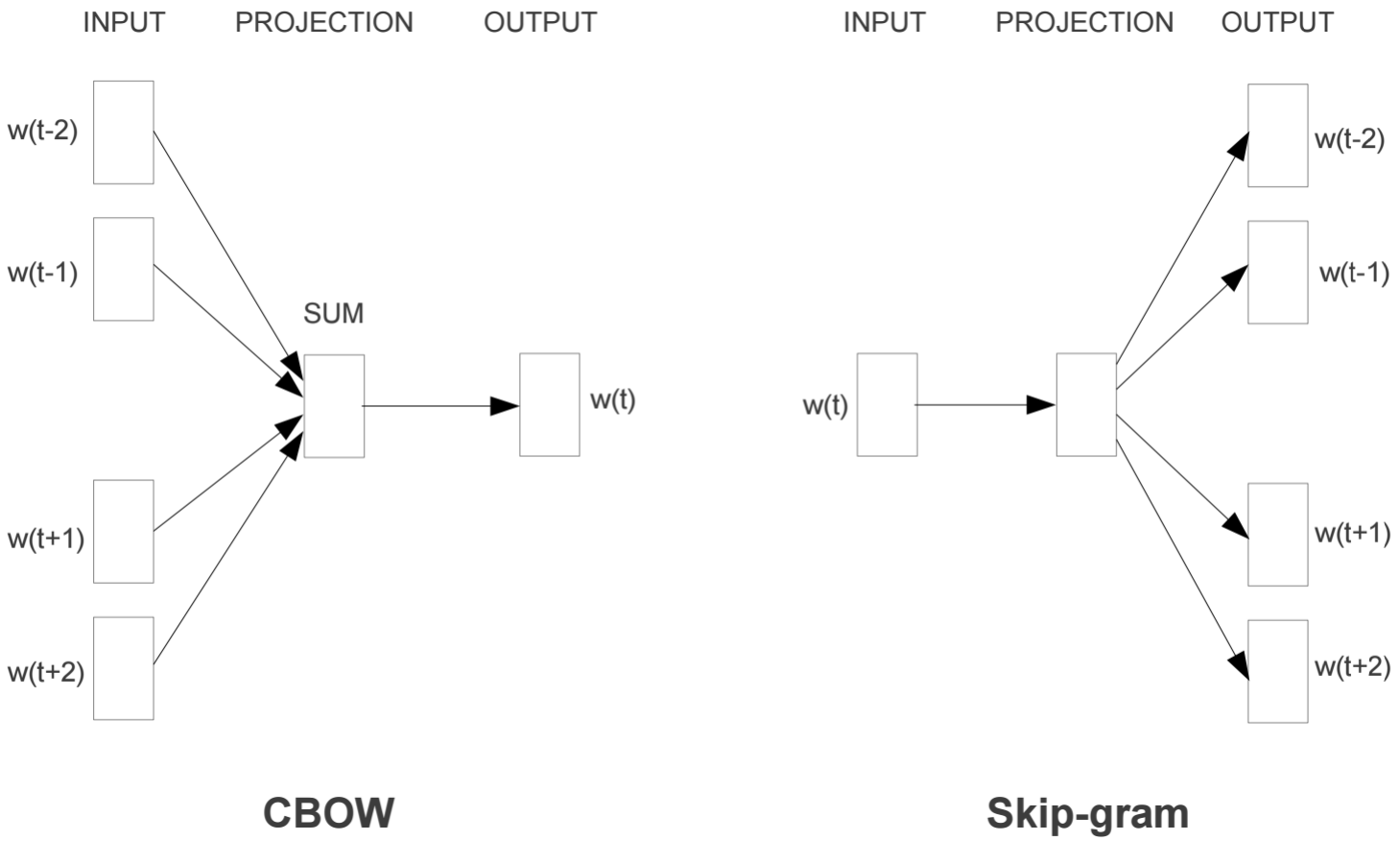
$$\theta \leftarrow \theta + \varepsilon \frac{\partial \log \hat{P}(w_t | w_{t-1}, \cdots, w_{t-n+1})}{\partial \theta}$$

AGENDA

Word2Vec

Mikolov et al. (2013)

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



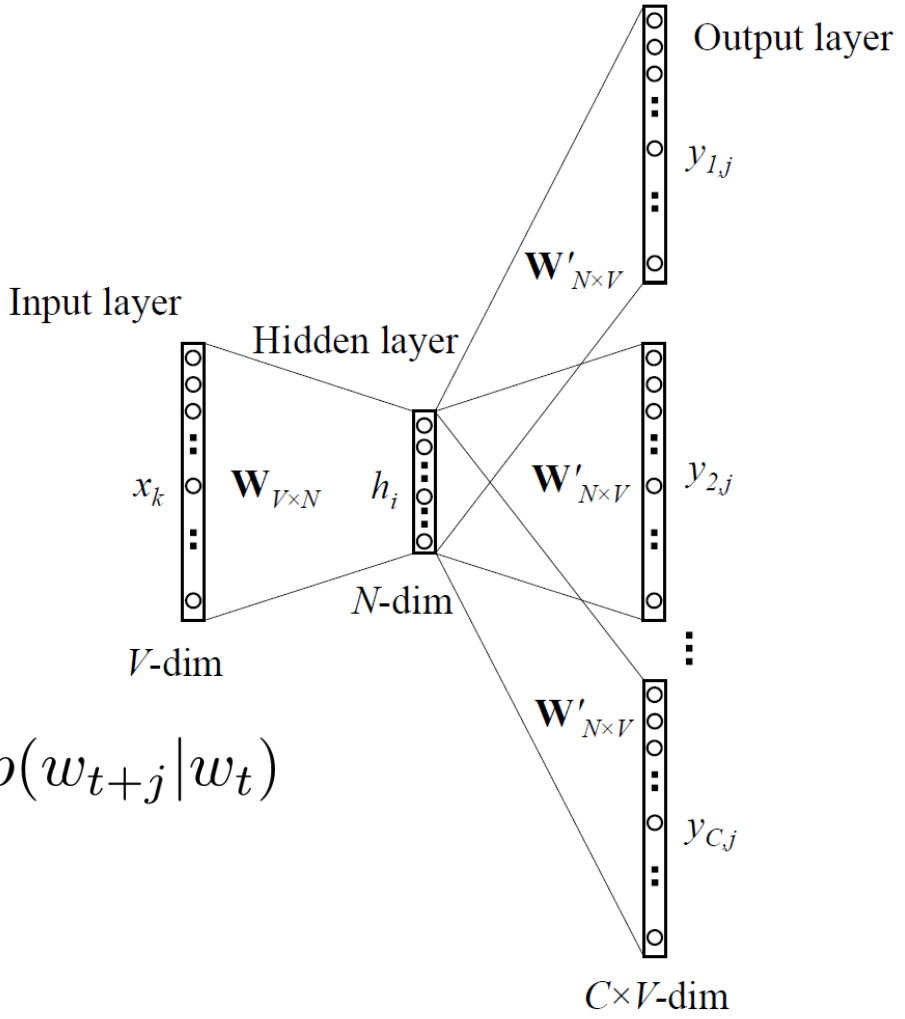
Word2Vec

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

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

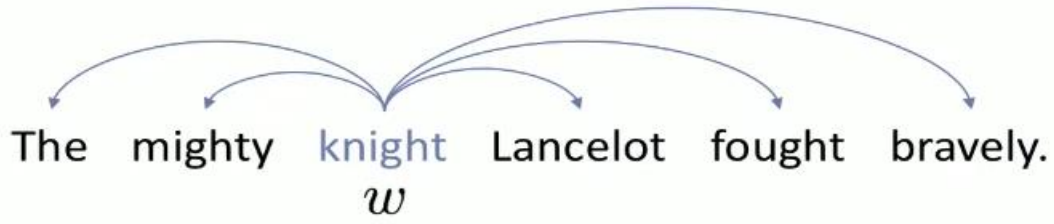
  - ✓ where θ represents all variables we optimize



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

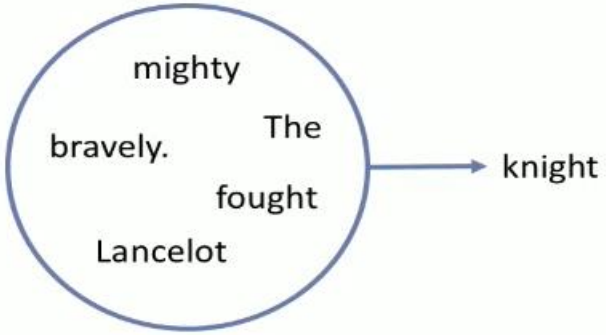
$$p(c|w) = \frac{e^{x_w^\top v_c}}{\sum_{k=1}^{K} e^{x_w^\top v_k}}$$

- 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

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

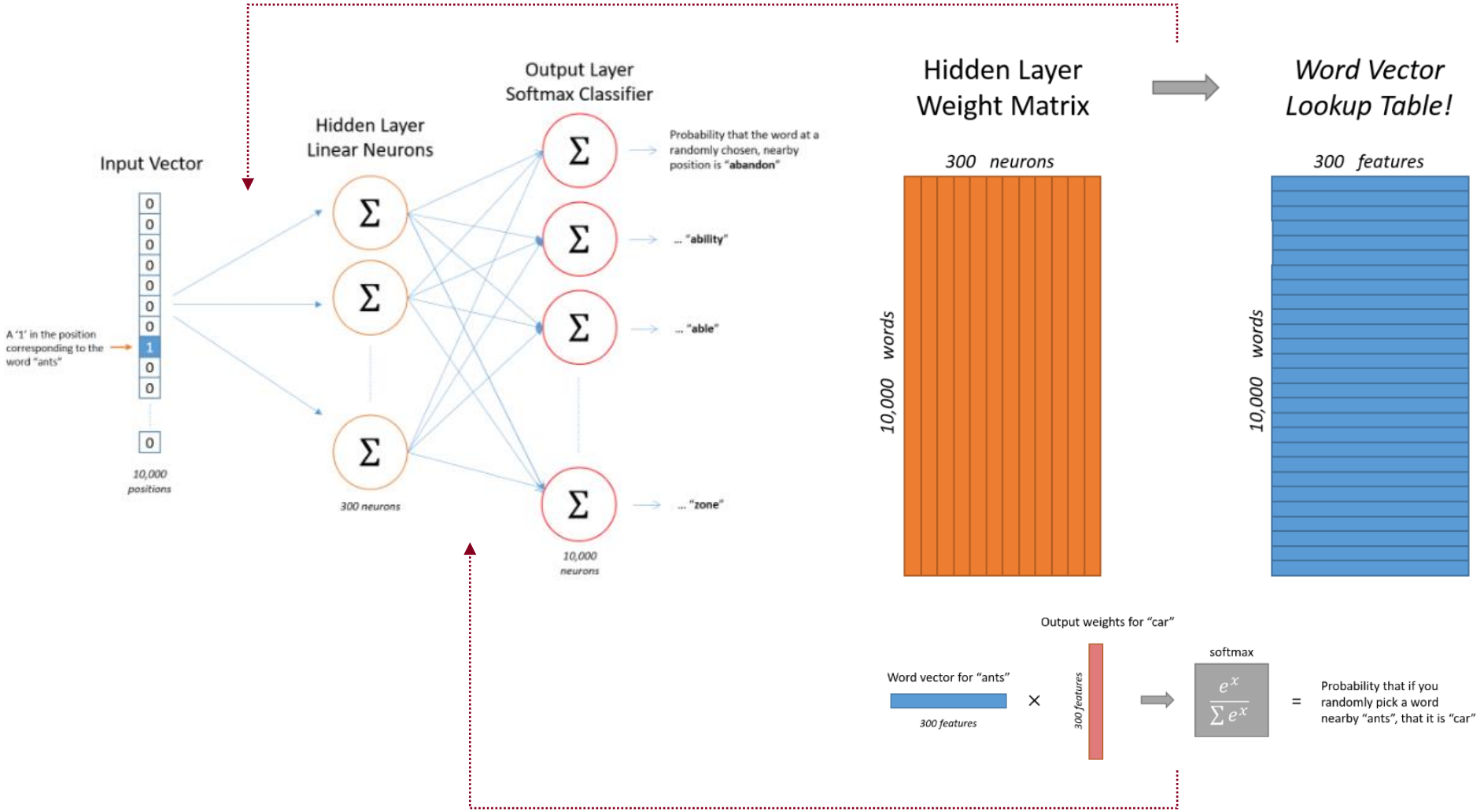
- Continuous Bag Of Words

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

Word2Vec

McCormick

- Another architecture explanation

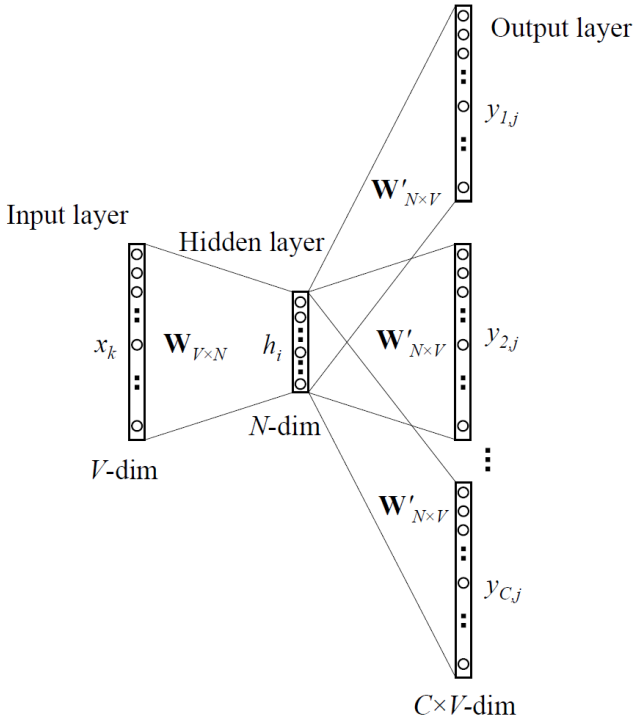


Word2Vec

- 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

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

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

A 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

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

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

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

Word2Vec

- Learning parameters with Gradient Ascent

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

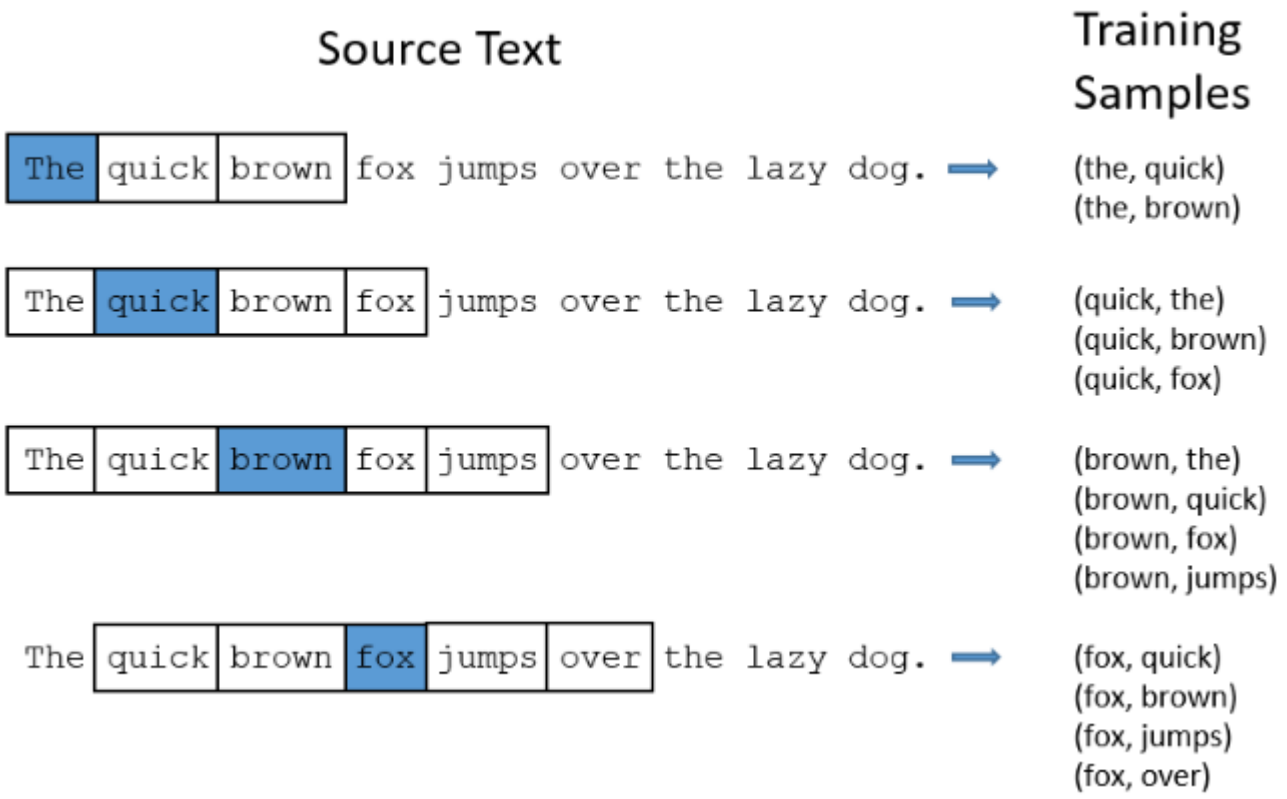
- Update the weight vector

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

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

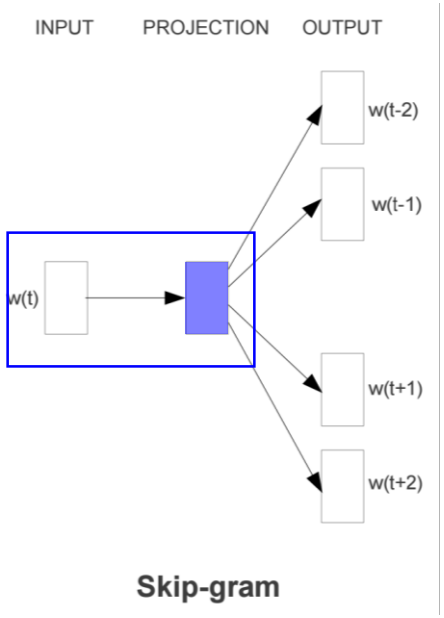
$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log p(w_{t+j}|w_t) = \frac{1}{T}\sum_{t=1}^{T} J_t(\theta)$$

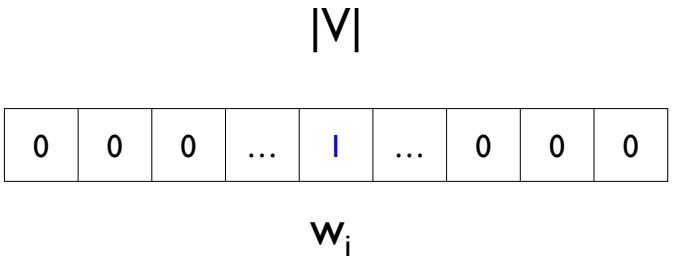
$$J_t(\theta) = \log\sigma(u_o^T v_c) + \sum_{i=1}^{k} E_{i\sim P(w)}\left[\log\sigma(-u_i^T v_c)\right]$$

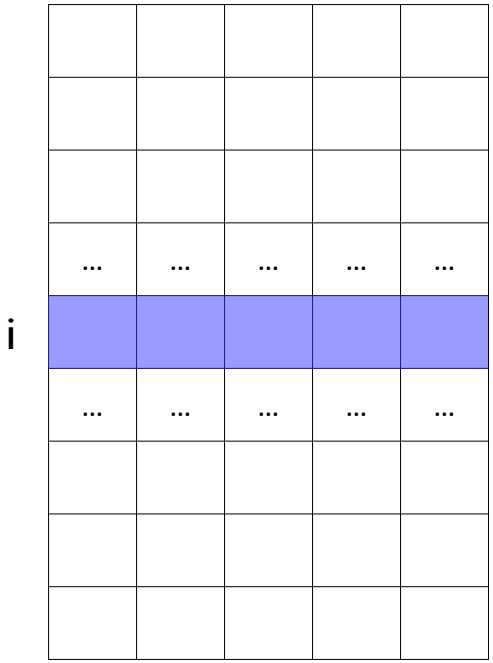
$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{n}\left(f(w_i)^{3/4}\right)}$$

Word2Vec

- Negative Sampling Example



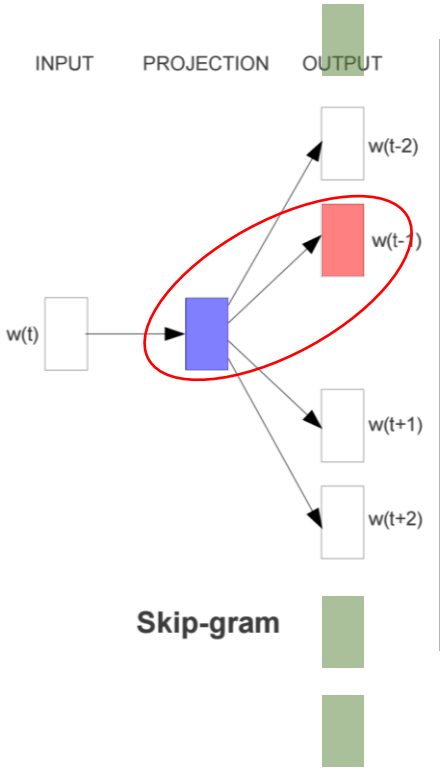


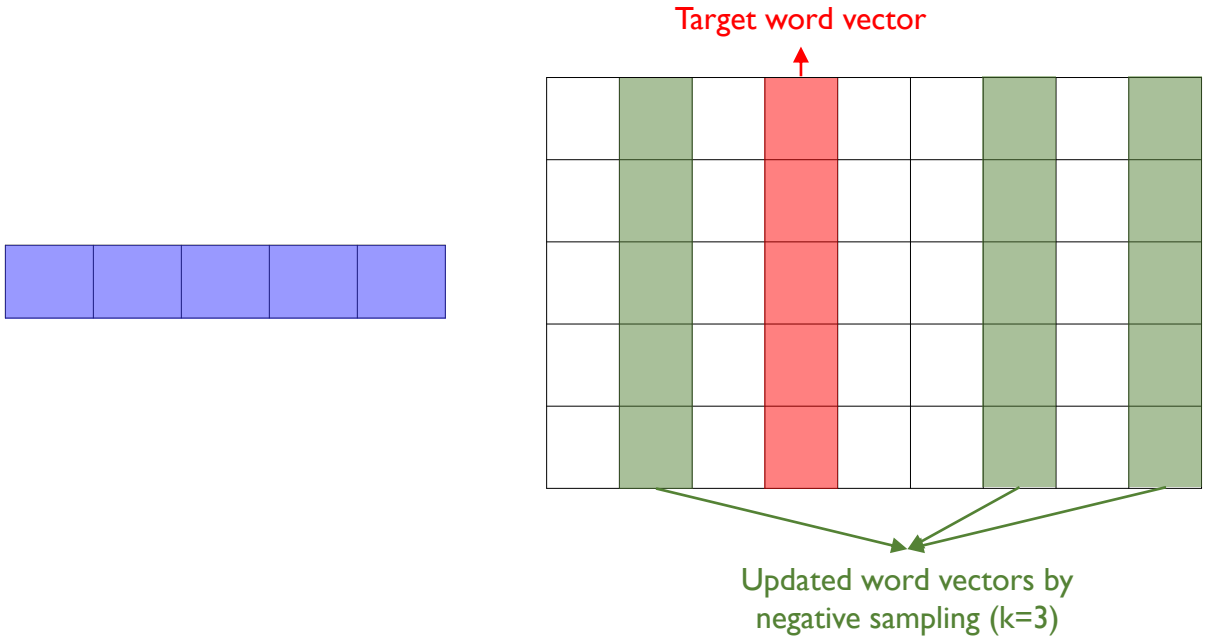


Context word vector

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):

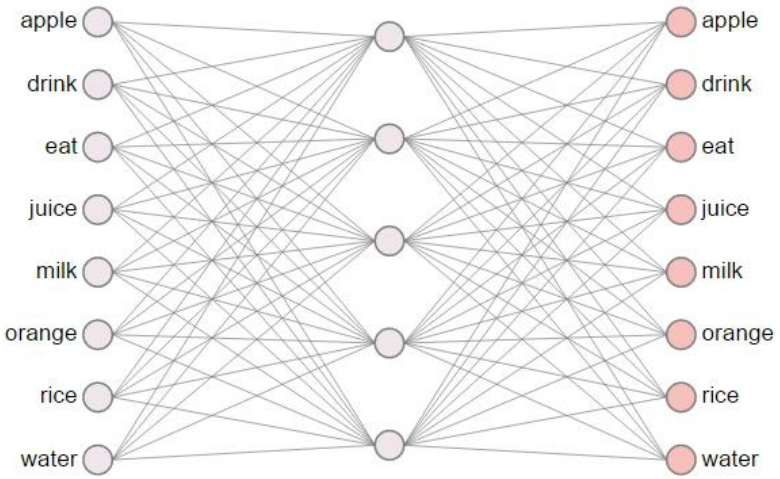
```
eat|apple,eat|orange,eat|rice,drink|juice,drink|milk,drink|water,orange|juice,apple|juice,rice|milk,milk|drink,water|drink,juice|drink
```

Presets: Fruit and juice

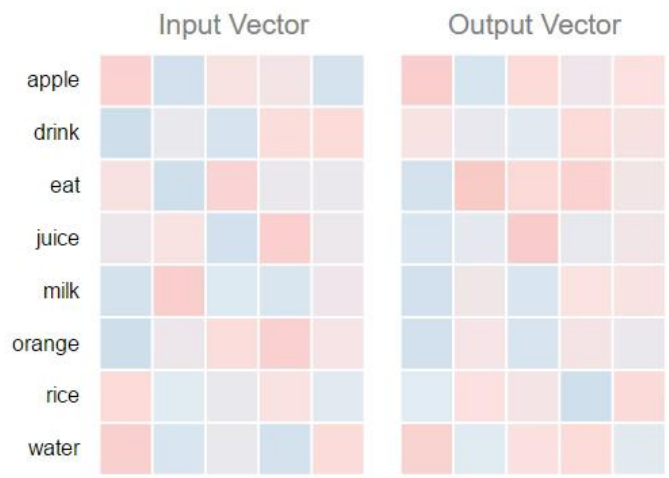
Update and Restart | Update Learning Rate

Next | 20 | 100 | 500 | PCA

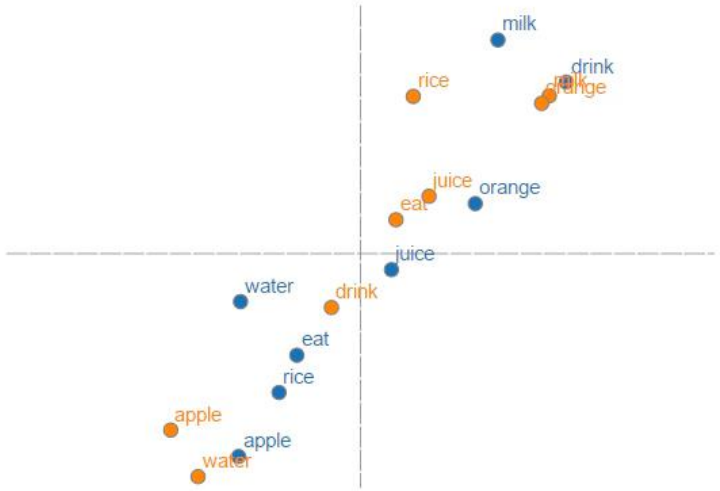
Neurons



Weight Matrices

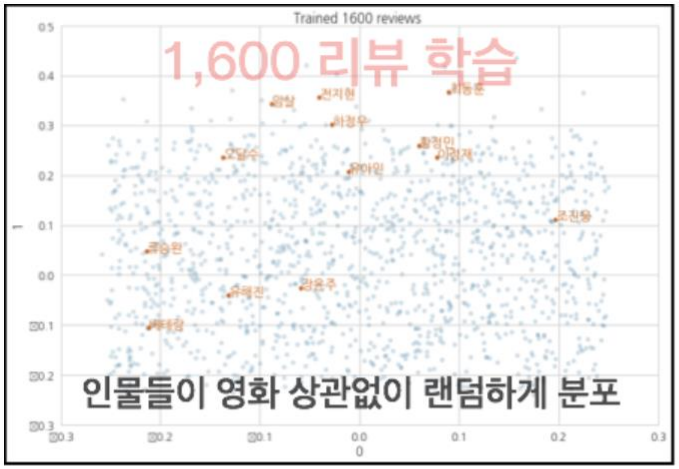


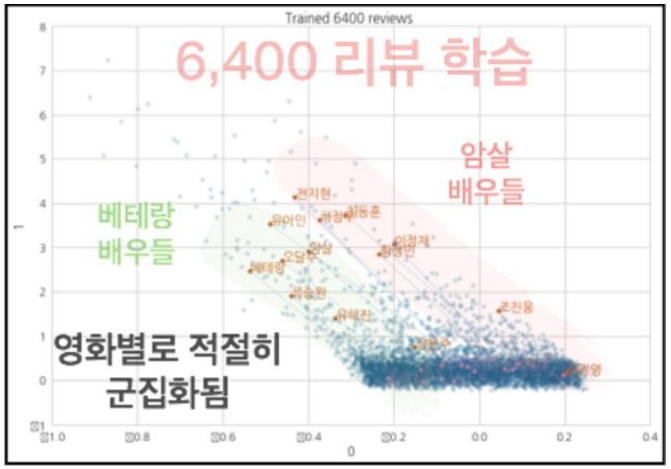
Vectors

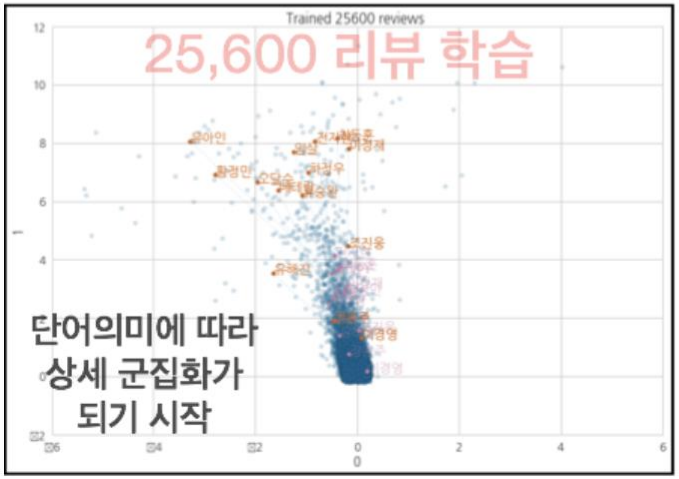


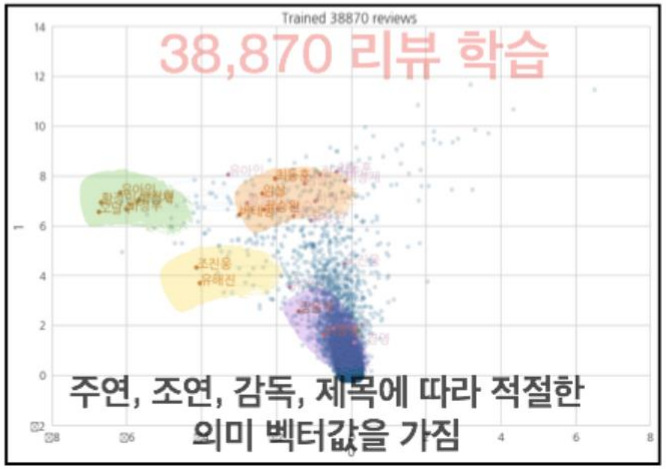
Word2Vec: Example

- Trained with movie reviews





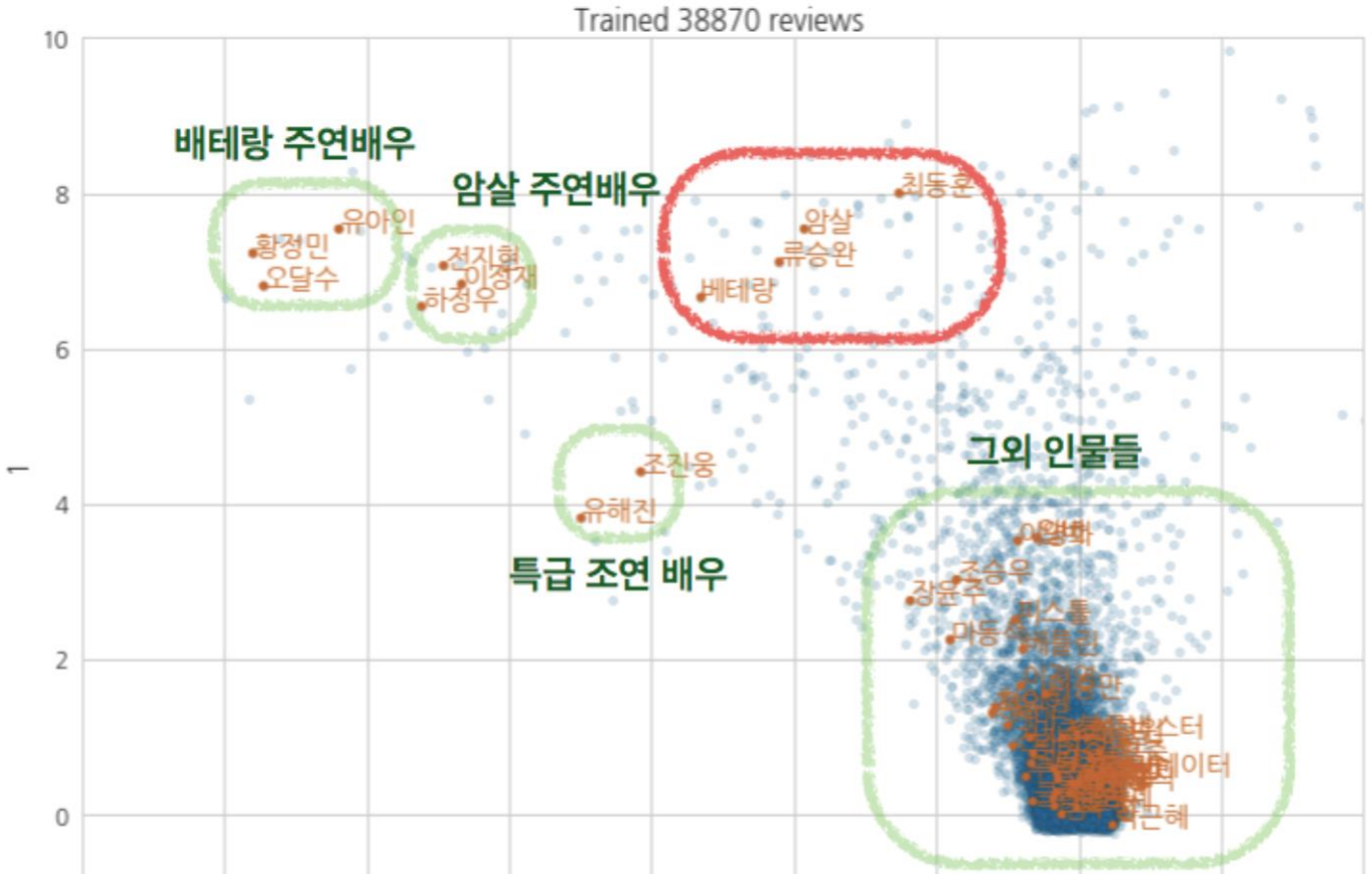




https://brunch.co.kr/@goodvc78/16

Word2Vec: Example

- Trained with movie reviews



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

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

AGENDA

GloVe

Pennington et al. (2014)

- Limitations of Word2Vec
  - ✓ The network spends so much time to train some overwhelmingly used words
    - Ex: to learn a distribution for P(w|the)

Theatre or theater is a collaborative form of fine art that uses live performers to present the experience of a real or imagined event before a live audience in a specific place. The performers may communicate this experience to the audience through combinations of gesture, speech, song, music, and dance. Elements of art and stagecraft are used to enhance the physicality, presence and immediacy of the experience. The specific place of the performance is also named by the word "theatre" as derived from the Ancient Greek (thatron, "a place for viewing"), itself from (theomai, "to see", "to watch", "to observe"). Modern Western theatre comes from large measure from ancient Greek drama, from which it borrows technical terminology, classification into genres, and many of its themes, stock characters, and plot elements. Theatre artist Patrice Pavis denes theatricality, theatrical language, stage writing, and the specificity of theatre as synonymous expressions that differentiate theatre from the other performing arts, literature, and the arts in general. Theatre today, broadly denfied, includes performances of plays and musicals, ballets, operas and various other forms.

GloVe

Pennington et al. (2014)

- GloVe
  - ✓ Based on matrix factorization method
  - ✓ http://nlp.stanford.edu/projects/glove/
  - ✓ Notations
    - $X \in \mathbb{R}^{V \times V}$ word co-occurrence matrix
    - X_{ij} frequency of word i co-occurring with word j
    - $X_i = \sum_k^V X_{ik}$ total number of occurrences of word i in corpus
    - $P_{ij} = P(j|i) = \frac{X_{ij}}{X_i}$ a.k.a. probability of word j occurring within the context of word i
    - $w \in \mathbb{R}^d$ a word embedding of dimension d
    - $\tilde{w} \in \mathbb{R}^d$ a context word embedding of dimension d

GloVe

Pennington et al. (2014)

- Motivation

Prob. 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}
$\frac{P(k\|ice)}{P(k\|steam)}$	8.9	8.5×10^{-2}	1.36	0.96

✓ For words *k* related ice but not steam (solid), the ratio P_{ik}/P_{jk} is large

✓ For words *k* related steam but not ice (gas) the ratio P_{ik}/P_{jk} is small

✓ For words *k* that are either related to both ice and steam, or to neither, the ratio should be close to 1

GloVe

Pennington et al. (2014)

- Formulation
  - ✓ Express the relationship among three words using a function F

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

  - ✓ Relationship between w_i and w_j is expressed by subtraction

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

  - ✓ Inner product is used to link $\tilde{w}_k$ with w_i and w_j

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

GloVe

Pennington et al. (2014)

- Homomorphism

Prob. 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}
$\frac{P(k\|ice)}{P(k\|steam)}$	8.9	8.5×10^{-2}	1.36	0.96

✓ Want to preserve $\dfrac{P(k|ice)}{P(k|steam)}$ using $F\left((w_i - w_j)^T \tilde{w}_k\right) = \dfrac{P_{ik}}{P_{jk}}$

$$\frac{P(solid|ice)}{P(solid|steam)} = F\left((ice - steam)^T solid\right)$$

$$\frac{P(solid|steam)}{P(solid|ice)} = F\left((steam - ice)^T solid\right)$$

GloVe

Pennington et al. (2014)

- Homomorphism

$$F\Big((ice - steam)^T solid\Big) = \frac{P(solid|ice)}{P(solid|steam)} = \frac{1}{F\Big((steam - ice)^T solid\Big)}$$

$$(ice - steam)^T solid = -(steam - ice)^T solid$$

inverse element of addition

$$F\Big((ice - steam)^T solid\Big) = \frac{1}{F\Big((steam - ice)^T solid\Big)}$$

inverse element of multiplication

✓ Homomorphism preserves an operation, which in turn preserves the inverse element

✓ Need a homomorphism from $(\mathbb{R}, +)$ to $(\mathbb{R}_{>0}, \times)$

GloVe

Pennington et al. (2014)

- Homomorphism
  - ✓ Function F: homomorphism that maps $(\mathbb{R}, +)$ to $(\mathbb{R}_{>0}, \times)$

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

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

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

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

  - ✓ Finally, we can drive that $F(x) = \exp(x)$

GloVe

Pennington et al. (2014)

- Solution
  - ✓ We know that $F(x) = \exp(x)$ and $F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)} = \frac{P_{ik}}{P_{jk}}$

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

$$\downarrow$$

$$w_i^T \tilde{w}_k = \log P_{ik} = \log X_{ik} - \log X_i$$

$$\downarrow$$

$$w_i^T \tilde{w}_k = \log P_{ik} = \log X_{ik} - \log X_i$$

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

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

GloVe

Pennington et al. (2014)

- Objective Function
  - ✓ A least squared objective function

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

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

where f has the following desiderata:

1. $f(0) = 0$
2. $f(x)$ should be non-decreasing so that rare co-occurrences are not overweighted.
3. $f(x)$ should be relatively small for large values of x, so that frequent co-occurrences are not overweighted.

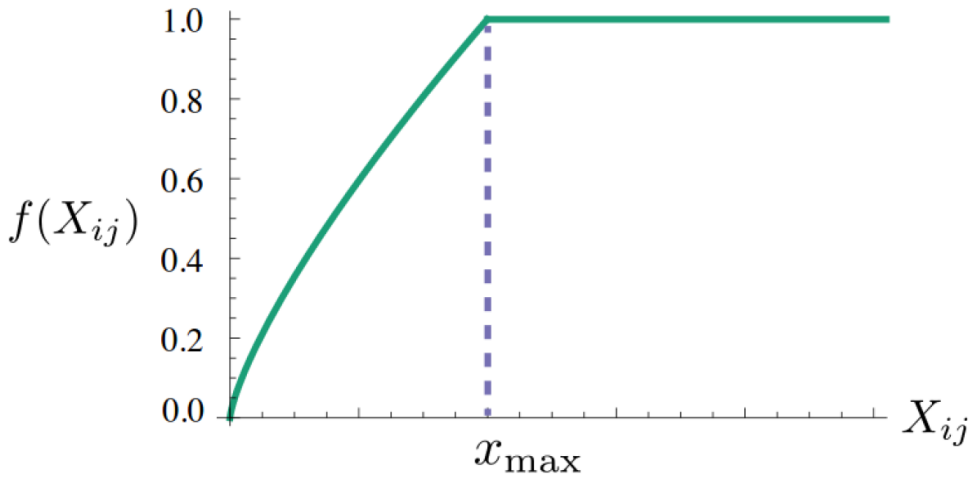
GloVe

Pennington et al. (2014)

- Objective Function
  - ✓ A least squared objective function

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

where $f(x) = \begin{cases} \left(\frac{x}{x_{\max}}\right)^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$



GloVe

Pennington et al. (2014)

- Results

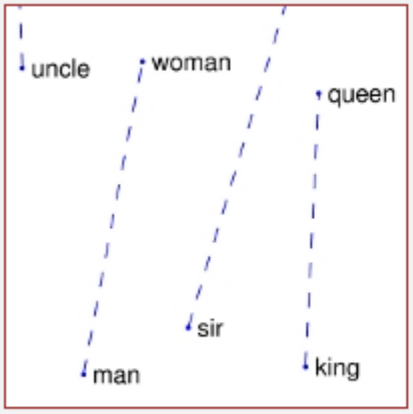
0. *frog*
1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus


3. litoria

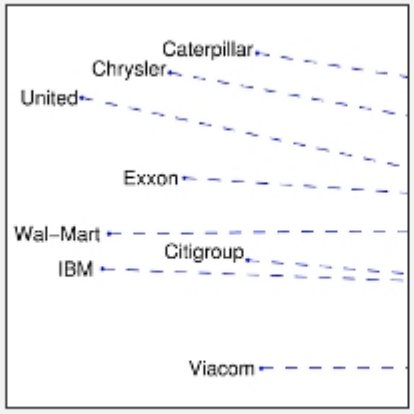

4. leptodactylidae


5. rana

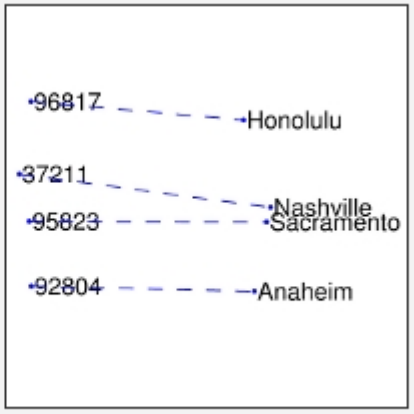

7. eleutherodactylus



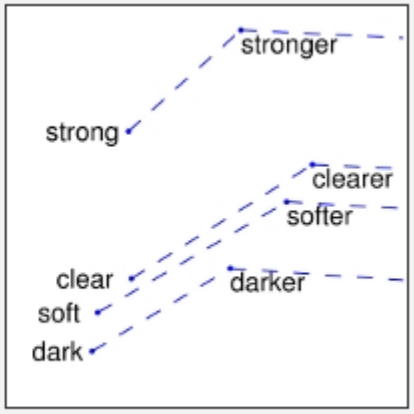
man - woman



company - ceo



city - zip code



comparative - superlative

http://nlp.stanford.edu/projects/glove/

AGENDA

FastText

Bojanowski et al. (2016)

- Limitations of NNLM, Word2Vec, and GloVe
  - ✓ Ignores the morphology or words by assigning a distinct vector to each word
  - ✓ Difficult to apply to morphologically rich languages with large vocabularies and many rare words (Turkish or Finnish)
- Goal
  - ✓ Learn representations for character n-grams
  - ✓ Represent words as the sum of n-gram vectors

FastText

Bojanowski et al. (2016)

- Revisit Negative Sampling in Word2Vec

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

  - ✓ Score is just a dot product between the two embeddings

- Subword model

  - ✓ Define the set of n-grams appearing in w: $\mathcal{G}_w \subset \{1, ..., G\}$

$$score(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^T \mathbf{v}_c$$

  - ✓ Represent a word by the sum of the vector representations of its n-grams

FastText

Bojanowski et al. (2016)

- Subword model
  - ✓ n-gram representation
    - Include the word w in the set of its n-grams
    - Keep all the n-grams of size 3, 4, 5, and 6
    - Different vectors are assigned to a word and a n-gram sharing the same sequence of characters

Word2Vec

parameter	...	...	...	...	...

FastText

p	...	...	...	...	...
...	...	...	...	...	...
r	...	...	...	...	...
pa	...	...	...	...	...
...	...	...	...	...	...
er	...	...	...	...	...
par	...	...	...	...	...
...	...	...	...	...	...
ameter	...	...	...	...	...
Avg.	...	...	...	...	...

Word Embedding Examples

- Word Embedding examples: English
  - ✓ Word lists that are close to a given word after embedding

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

Collobert et al. (2011)

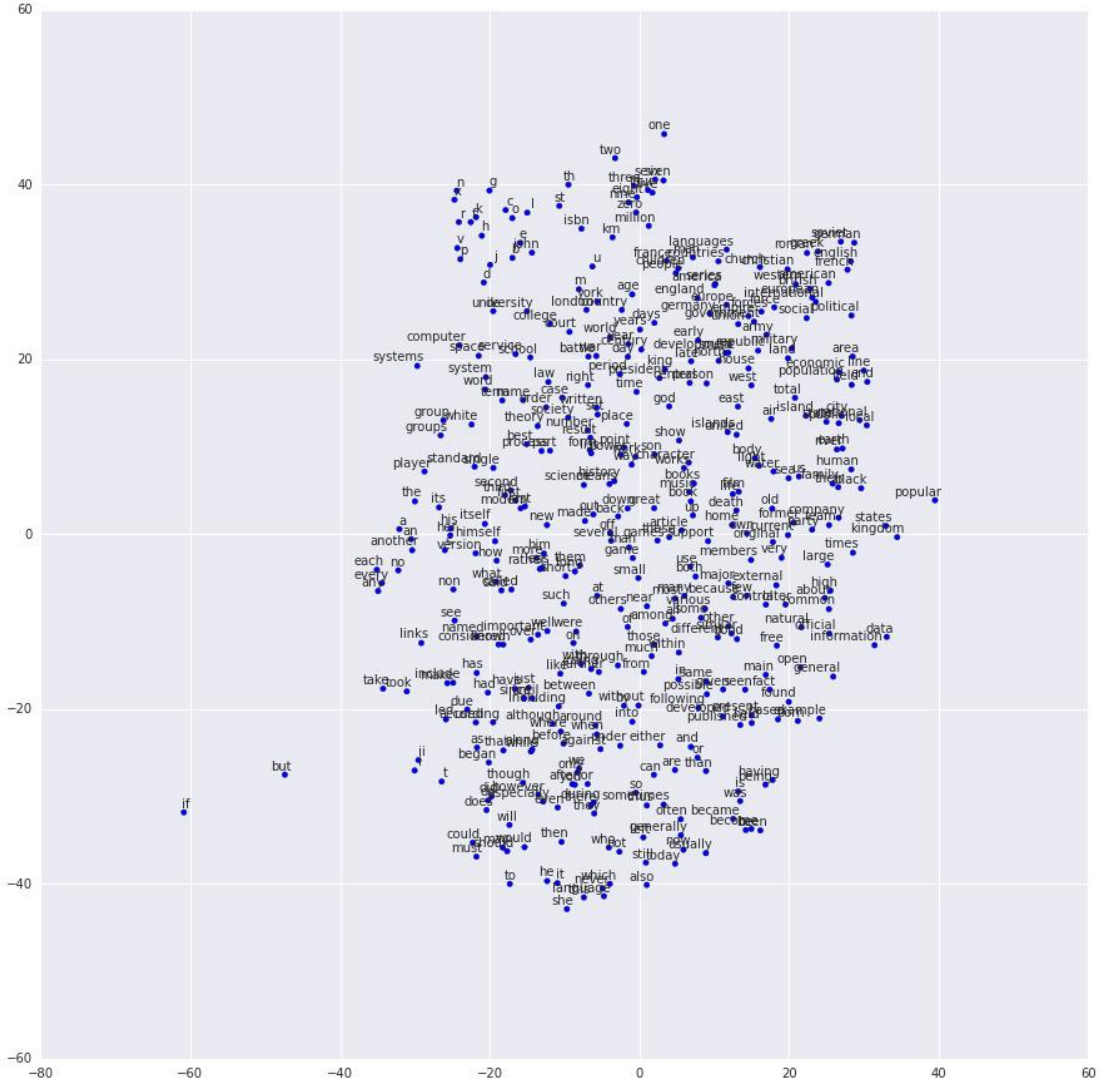
  - ✓ Relationship pairs in a word embedding

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Mikolov et al. (2013)

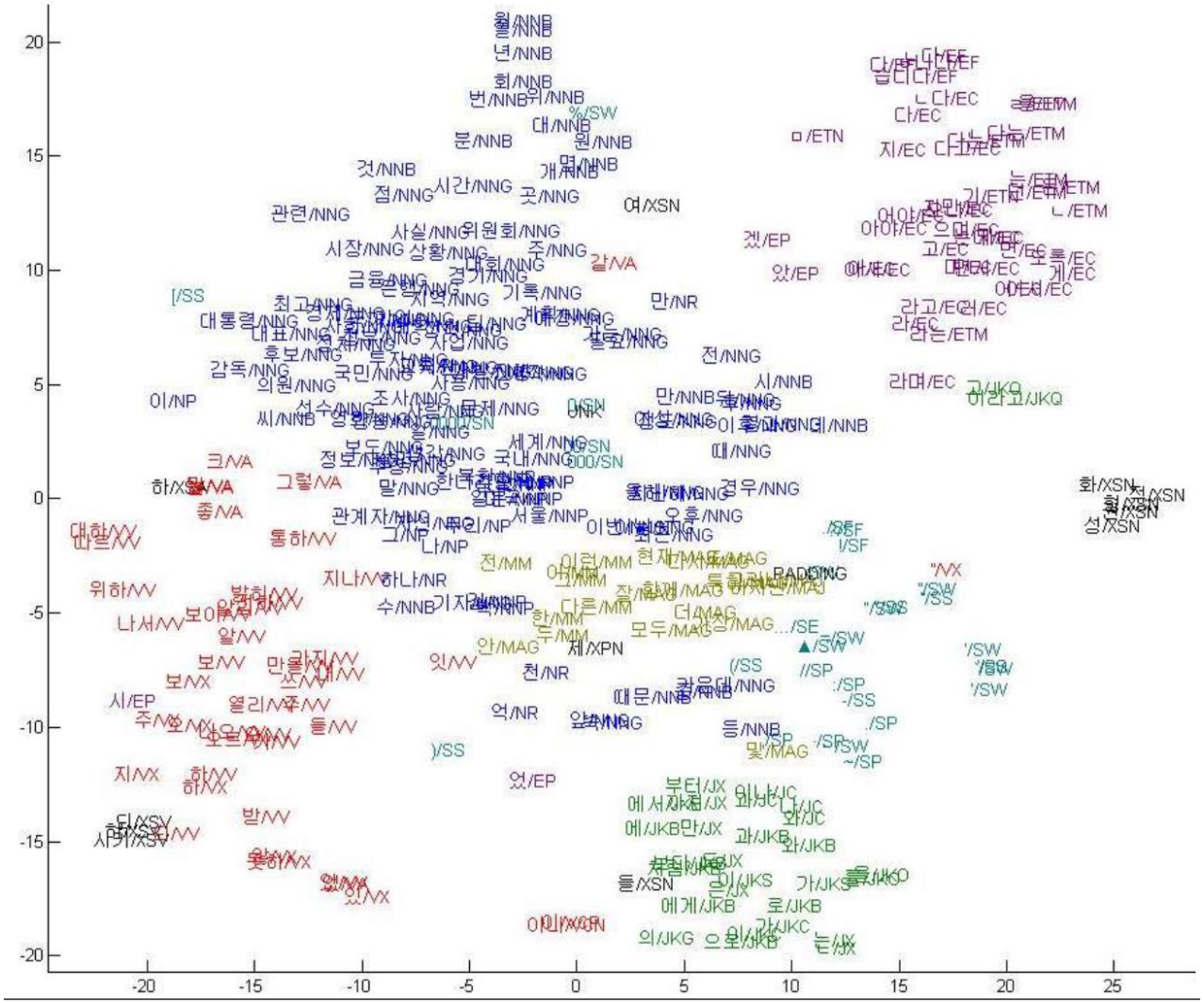
Word Embedding Examples

- Word Embedding examples: English



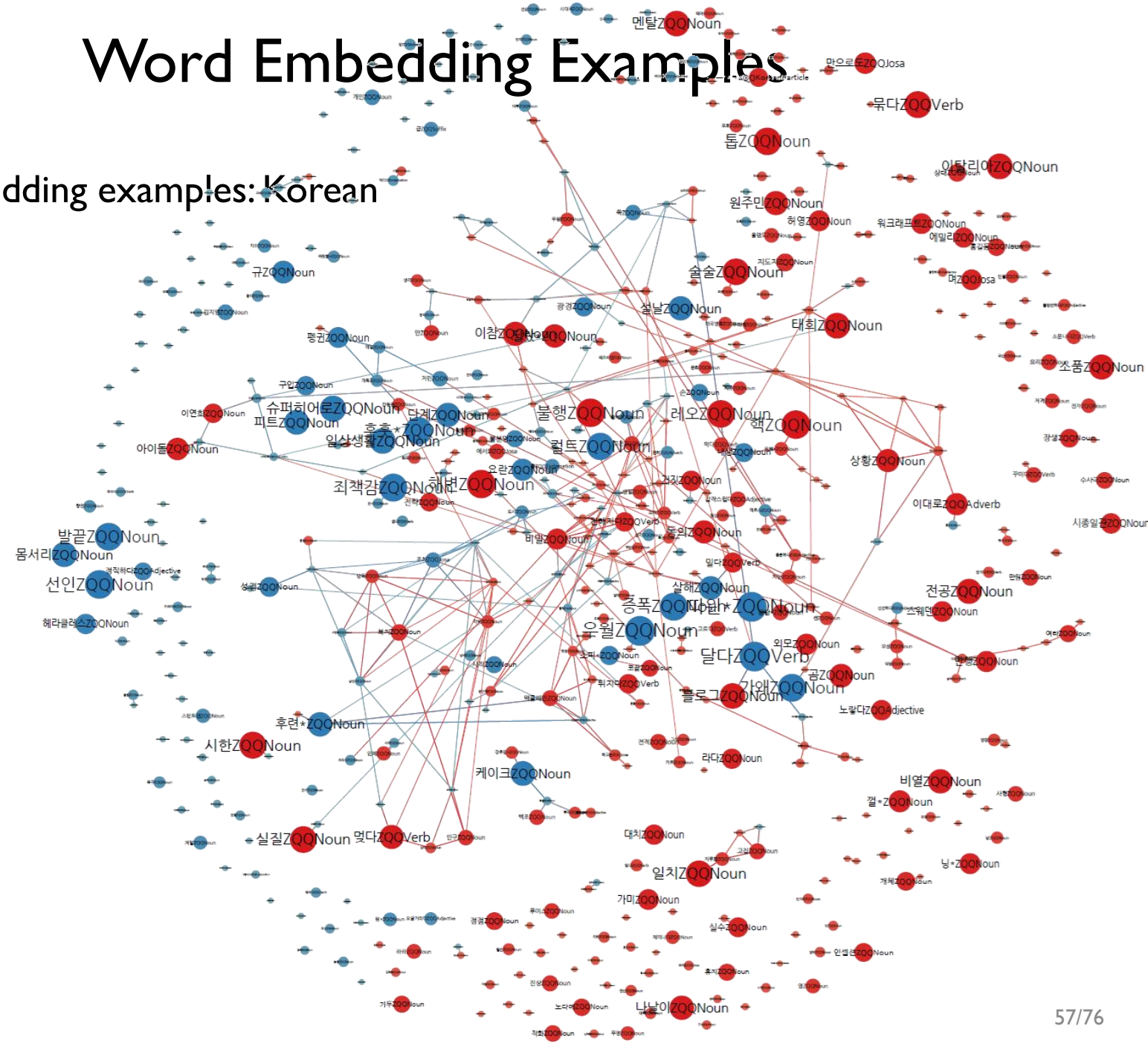
Word Embedding Examples

- Word Embedding examples: Korean



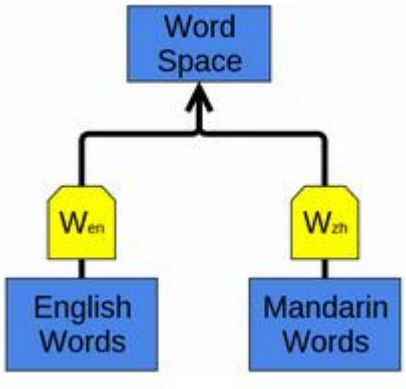
Word Embedding Examples

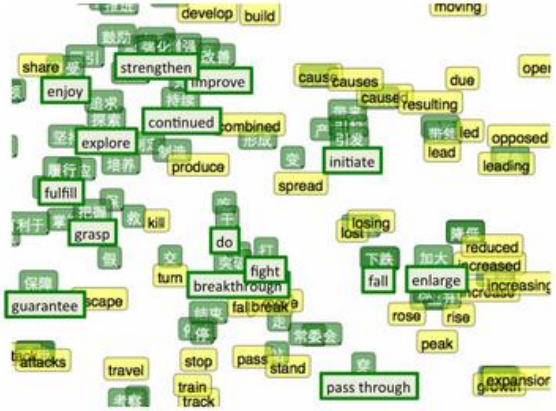
- Word Embedding examples: Korean



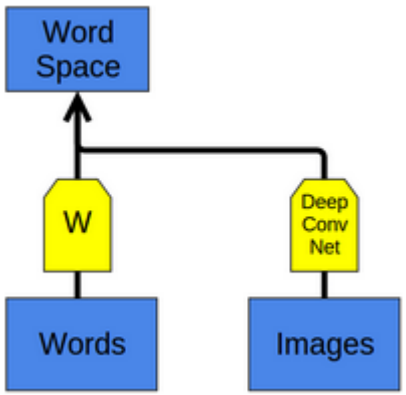
Word Embedding Examples

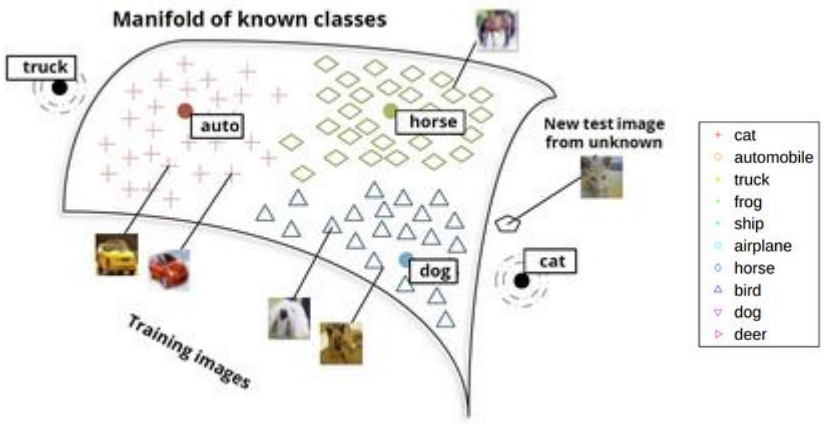
- Word Embedding with two different languages

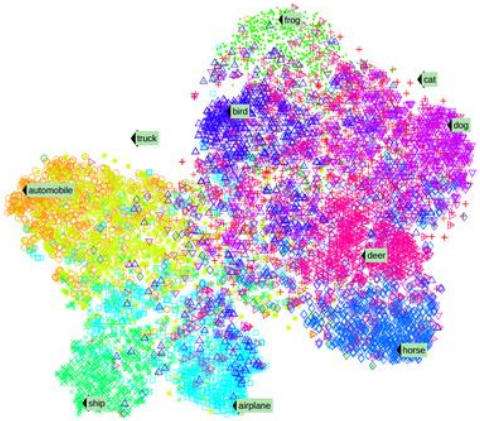




- Word Embedding with Images







AGENDA

01 Word-level: NNLM

02 Word-level: Word2Vec

03 Word-level: GloVe

04 Word-level: Fasttext

05 Sentence/Paragraph/Document-level

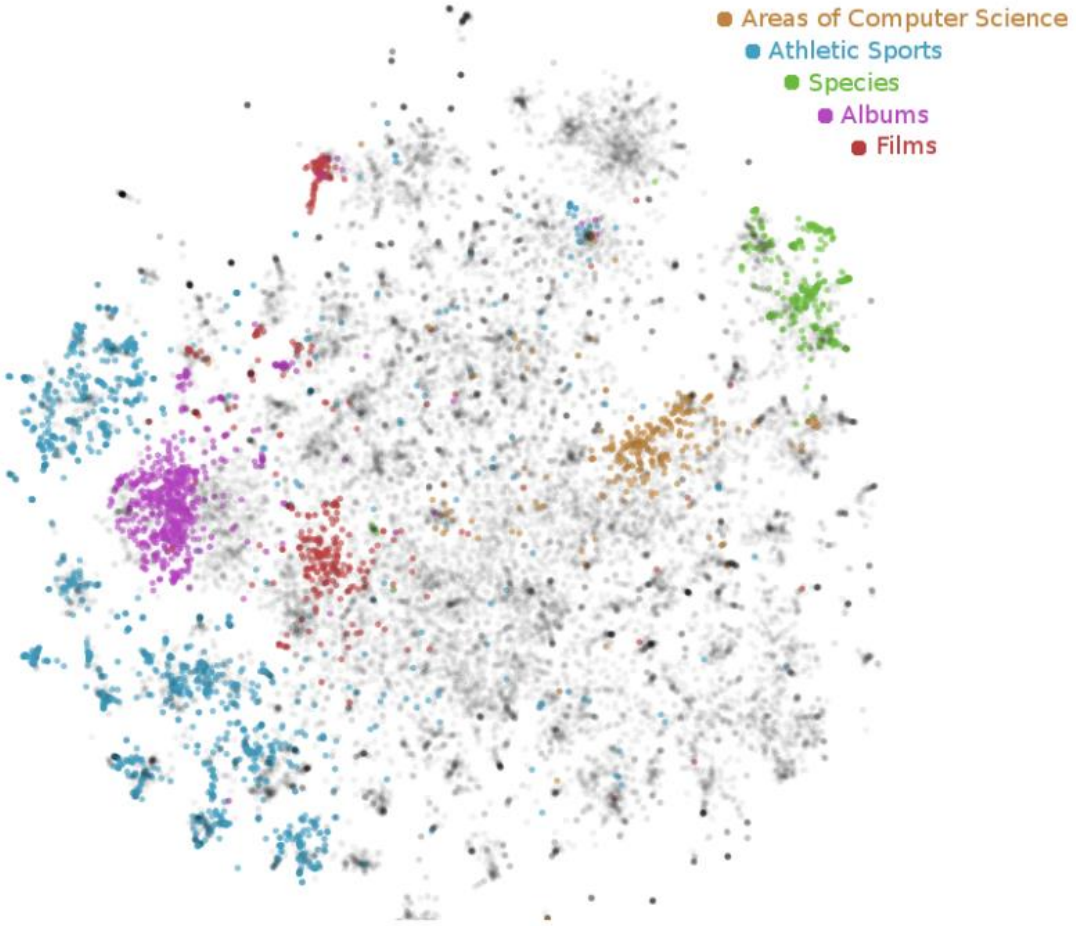
06 More Things to Embed?

Document Embedding

Dai et al. (2015)

- If we can embed words, why not sentences, phrases, or documents?

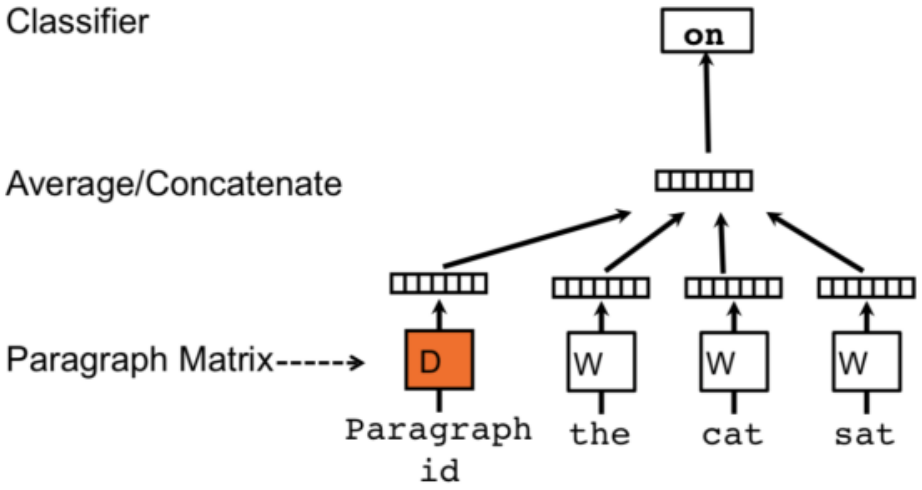
Visualization of Wikipedia paragraph vectors using t-SNE



Document Embedding

Le and Mikolov (2015)

- Paragraph Vector model: Distributed Memory (PV-DM) model

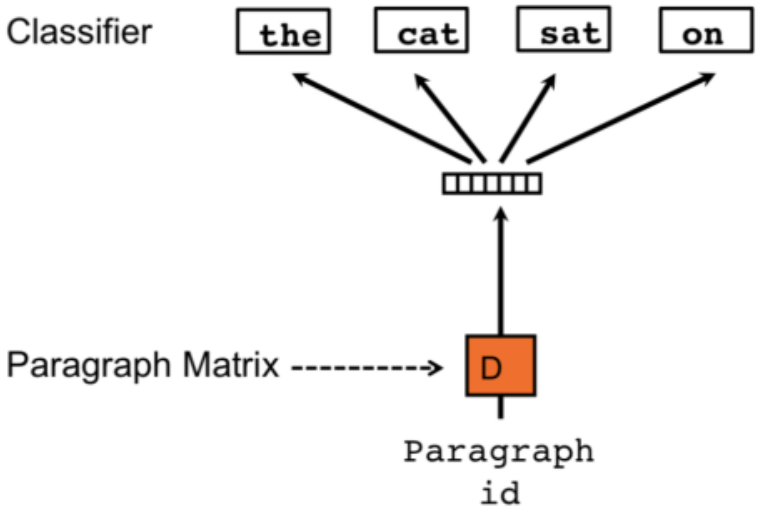


- ✓ The paragraph vectors are also asked to contribute to the prediction task of the next word given many contexts sampled from the paragraph
- ✓ Paragraph vectors are shared for all windows generated from the same paragraph, but not across paragraphs
- ✓ Word vectors are shared across all paragraphs

Document Embedding

Le and Mikolov (2015)

- Paragraph Vector model: Distributed Bag of Words (PV-DBOW)



✓ Ignore the context words in the input, and force the model to predict words randomly sampled from the paragraph in the output

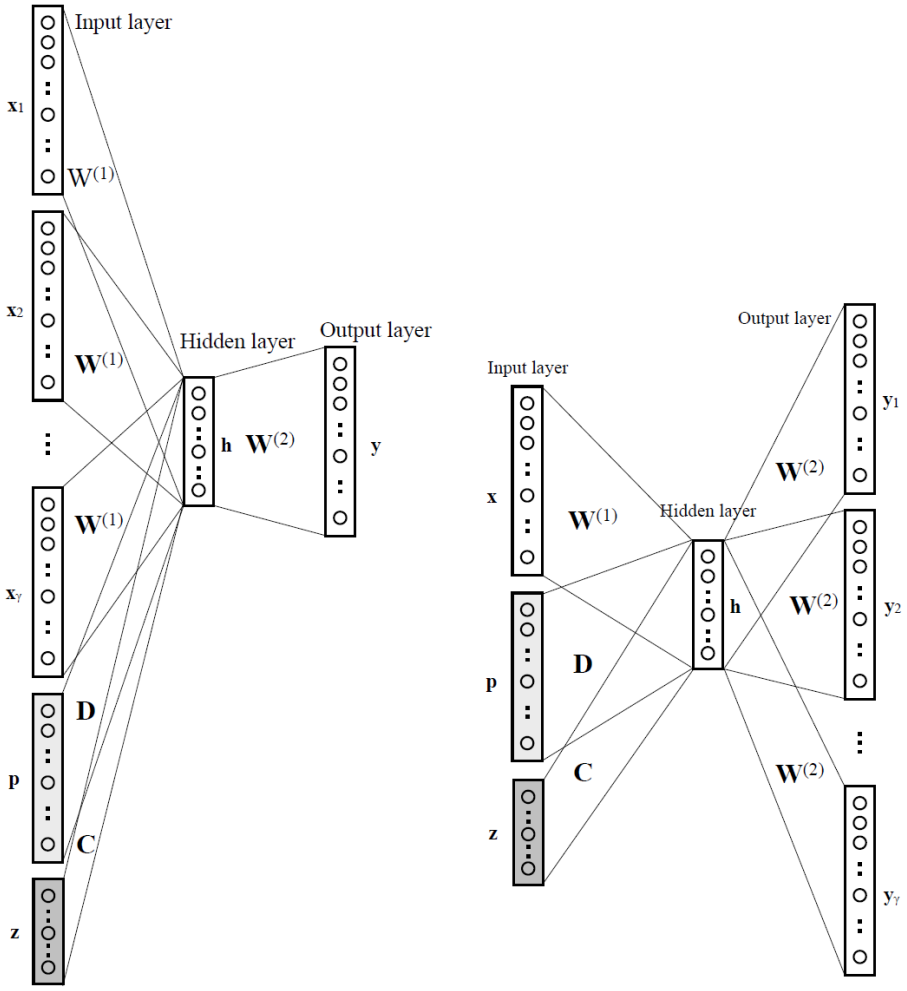
✓ Does not need word vectors

✓ PV-DM alone usually works well for most tasks, but the combination of PV-DM and PV-DBOW are recommended

Let's Embed Everything!

Park et al. (2016+)

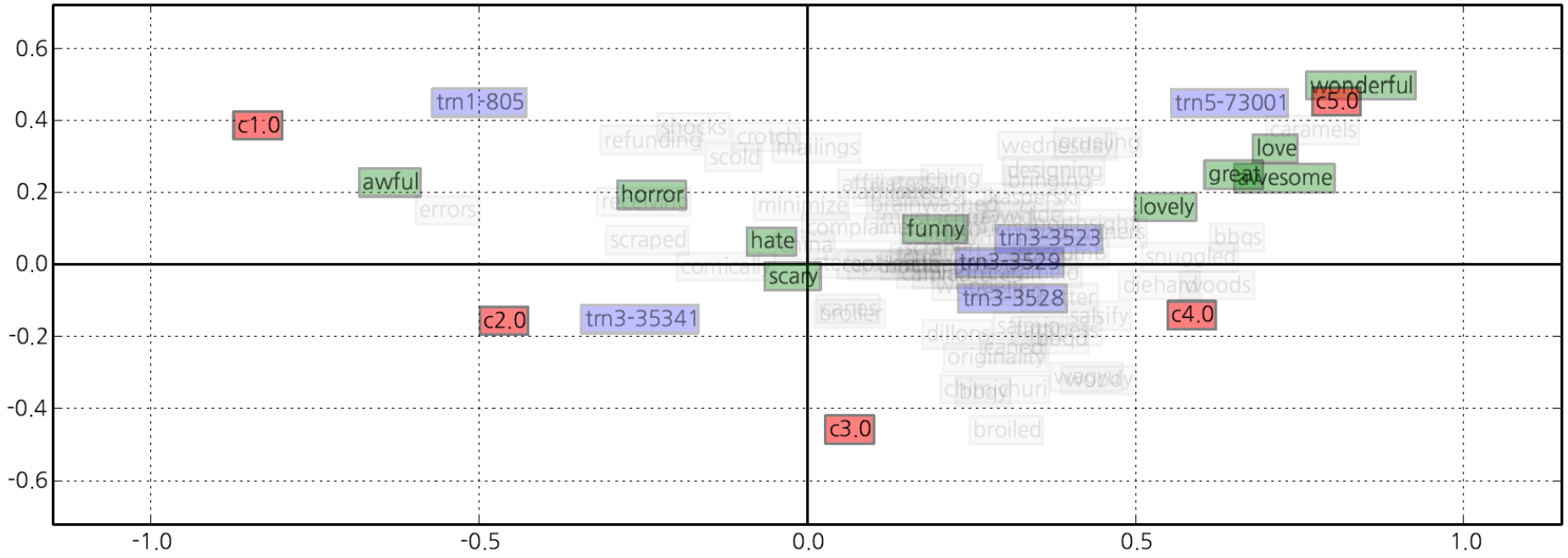
- Supervised Paragraph Vector (SPV) for Class Embedding



Let's Embed Everything!

Park et al. (2016+)

- Supervised Paragraph Vector (SPV) for Class Embedding



Let's Embed Everything!

Park et al. (2016+)

- Supervised Paragraph Vector (SPV) for Class Embedding

	imdb							
n(epochs)	1	5	10	30	50	70	100	t
BOW-TF	**85.30**	-	-	-	-	-	-	-
BOW-TFIDF	**85.55**	-	-	-	-	-	-	-
PV-DM	77.06	80.78	80.84	79.68	81.22	81.49	**82.16**	123.14
PV-DBOW	85.89	88.19	**88.47**	88.27	88.22	88.12	88.04	115.22
SPV-DM	82.57	81.68	82.05	**82.66**	82.42	82.53	82.61	121.51
SPV-DBOW	87.58	***88.87**	88.69	88.53	88.51	88.56	88.49	117.33

	yelp							
n(epochs)	1	5	10	30	50	70	100	t
BOW-TF	**58.42**	-	-	-	-	-	-	-
BOW-TFIDF	**58.93**	-	-	-	-	-	-	-
PV-DM	50.59	51.70	52.67	**52.97**	51.73	51.81	52.70	546.10
PV-DBOW	58.53	**59.37**	58.91	59.13	59.10	59.06	59.21	534.03
SPV-DM	51.48	51.42	52.40	53.14	53.77	53.75	**53.84**	552.71
SPV-DBOW	60.13	***60.21**	60.04	59.93	59.85	59.77	59.93	538.95

	amazon							
n(epochs)	1	5	10	30	50	70	100	t
BOW-TF	**85.91**	-	-	-	-	-	-	-
BOW-TFIDF	**85.97**	-	-	-	-	-	-	-
PV-DM	76.47	77.38	78.86	**79.26**	75.83	77.00	78.57	361.52
PV-DBOW	86.87	87.96	88.30	88.34	88.31	**88.42**	88.18	339.14
SPV-DM	79.05	78.35	78.66	80.26	80.36	80.62	**80.70**	371.95
SPV-DBOW	88.58	89.21	89.16	***89.34**	89.08	89.06	89.29	342.40

	20news							
n(epochs)	1	5	10	30	50	70	100	t
BOW-TF	**58.58**	-	-	-	-	-	-	-
BOW-TFIDF	**63.43**	-	-	-	-	-	-	-
PV-DM	24.45	41.17	45.36	49.04	52.34	53.85	**54.27**	71.64
PV-DBOW	53.51	69.16	72.51	74.76	75.22	75.16	**75.40**	70.37
SPV-DM	44.76	64.18	67.01	67.84	68.32	**68.34**	67.70	71.88
SPV-DBOW	69.41	76.97	77.95	78.93	78.93	79.47	***79.59**	72.18

AGENDA

01 Word-level: NNLM

02 Word-level: Word2Vec

03 Word-level: GloVe

04 Word-level: Fasttext

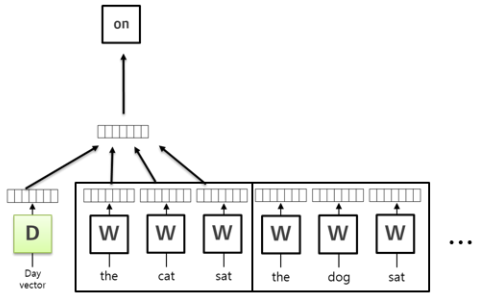
05 Sentence/Paragraph/Document-level

06 More Things to Embed?

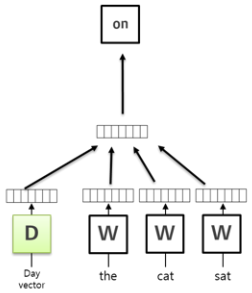
Let's Embed Everything!

- Day Embedding in News corpus

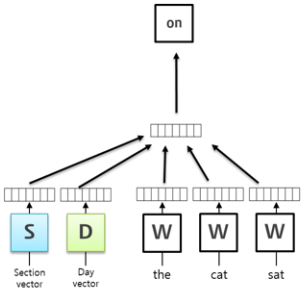
Approach 01: 하루 동안의 기사제목들을 병합 후 Day 태그



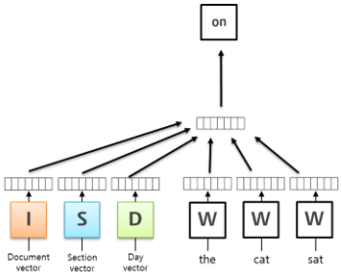
Approach 02: 각 뉴스기사 제목에 Day 태그



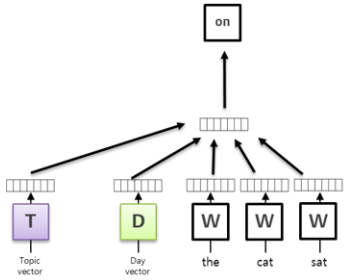
Approach 03: 각 뉴스기사제목에 Day, Section 태그



Approach 04: 각 뉴스기사제목에 Day, Section, IDX 태그



Approach 05: 각 뉴스기사제목에 Day, Topic 태그



Let's Embed Everything!

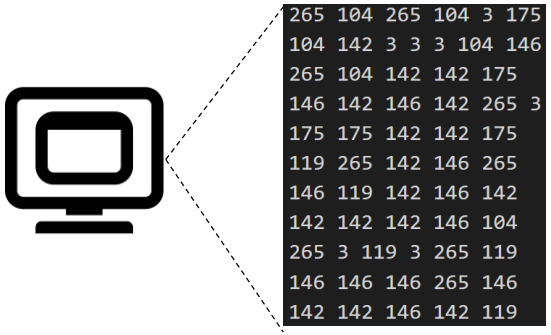
- System Call Trace Embedding for System Anomaly Detection

Data Preparation | Vectorization | Anomaly Detection

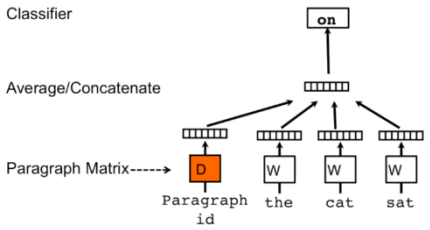
Host-based

ADFA-LD Dataset

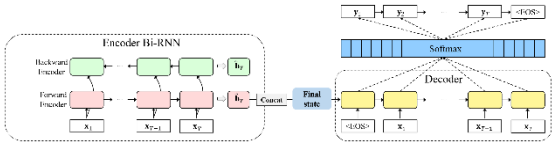
Syscall Trace



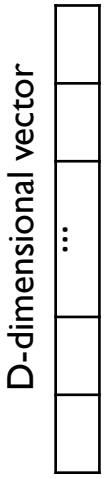
Doc2vec

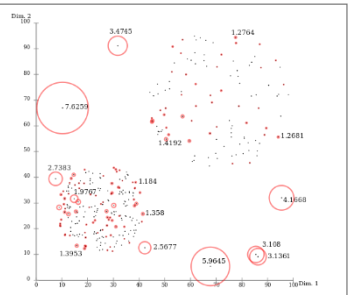


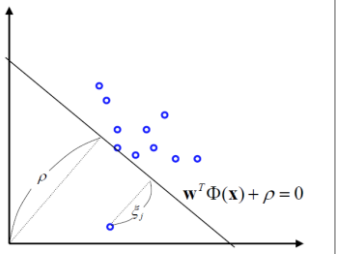
RNN-AutoEncoder



D-dimensional vector



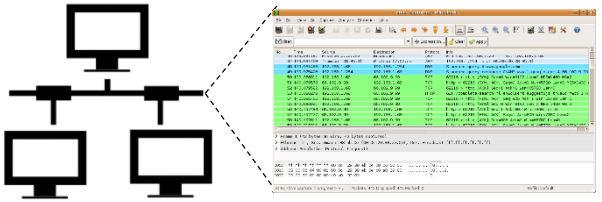




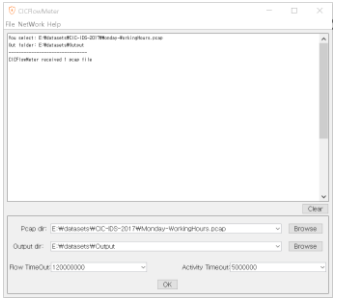
Network-based

CICIDS2017 Dataset

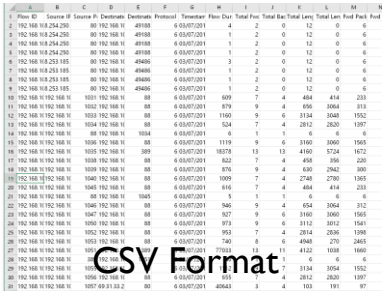
Packet Capture



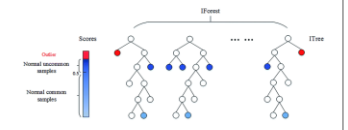
CIC-FlowMeter

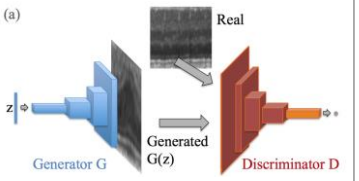


Features



CSV Format

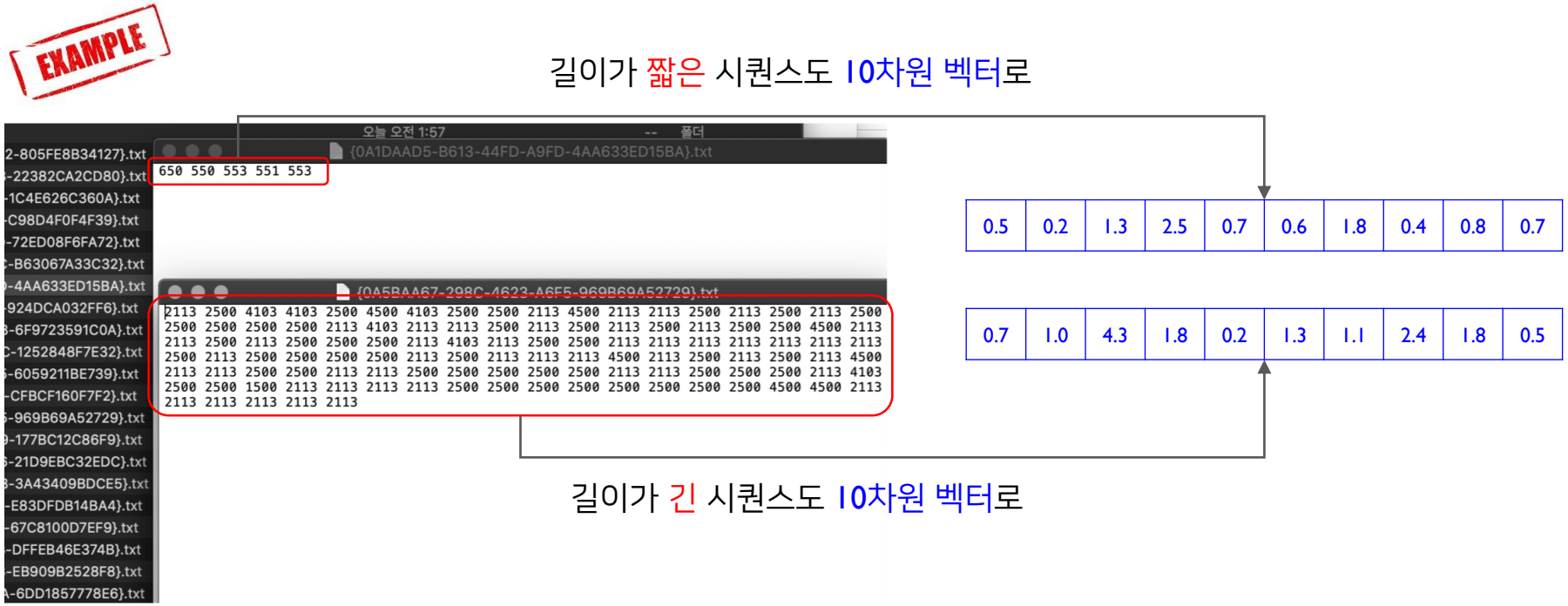




Let's Embed Everything!

- Question

  ✓어떻게 하면 가변 길이의 Syscall Trace를 고정 길이의 벡터로 변환할 수 있을까?



Let's Embed Everything!

- Sequence Embedding based on Doc2Vec
  - ✓ Syscall2Vec: 하나의 System Call Trace를 Document로 취급하고, 개별 syscall을 word로 취급하여 임베딩 수행

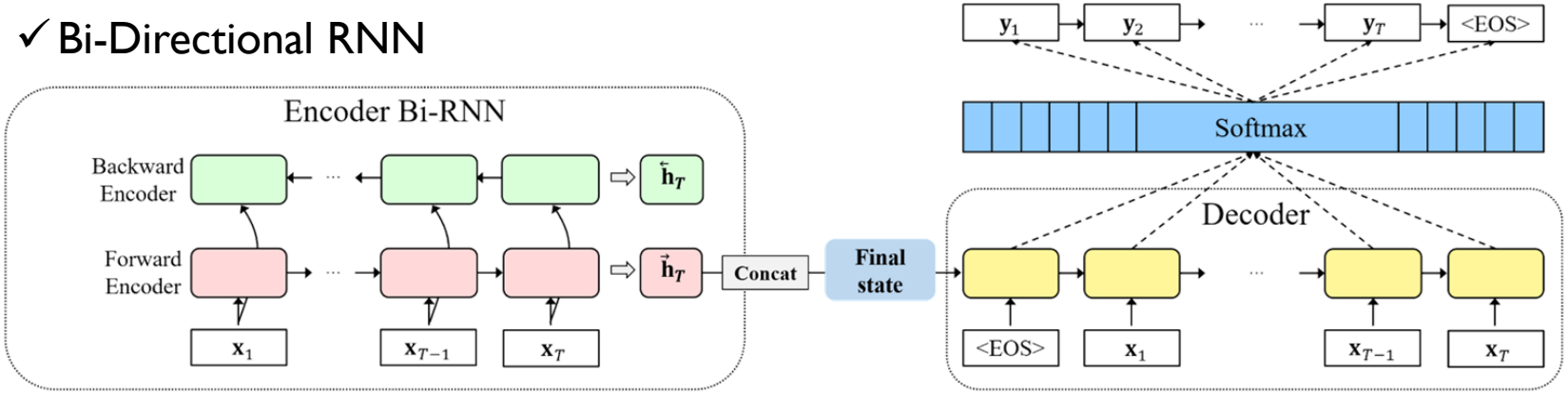
Document

168 3 3 265 168 3 43 168 3 168 168 43 265 168 3 168 43 168 43 168 265 43 265 265 168 265 265 168 168 168 3 168 3 265 168 3 168 168 168 168 3 168 168 168 3 3 168 168 265 168 3 168 265 168 168 3 168 265 43 168 265 43 3 265 43 43 3 ...

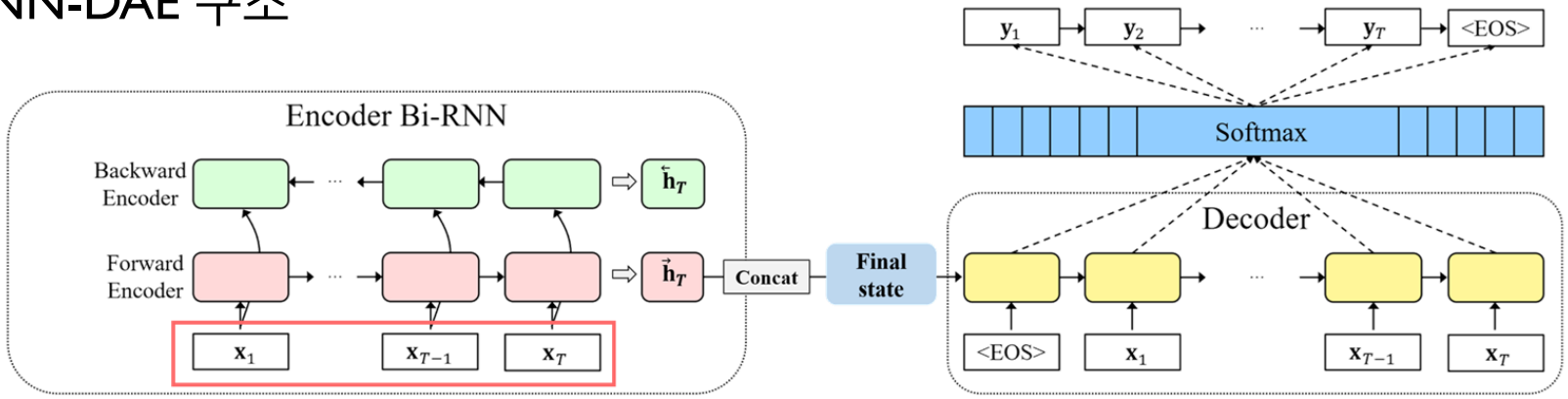
Word 1 Word 2 Word 3 Word 4

Let's Embed Everything!

- RNN-AE 구조
  - ✓ Bi-Directional RNN



- RNN-DAE 구조



Corruption Model

- 임의의 syscall을 p의 확률로 drop
- 임의의 syscall sequence를 permutation

Let's Embed Everything!

- Live2Vec in afreecaTV

[그림10] 문장과 Live 방송의 추론의 예시

Let's Embed Everything!

- Live2Vec in afreecaTV

[그림11] 유사도 측정 비교(Nearest neighbor vs Live2Vec)


ANY
questions?

References

Research Papers

- Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A Neural Probabilistic Language Model. Journal of Machine Learning Research, 3, 1137-1155.
- Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606.
- Collobert et al. (2011). Natural Language Processing (Almost) from Scratch. Journal of Machine Learning Research 12: 2493-2537.
- Dai, A. M., Olah, C., & Le, Q. V. (2015). Document embedding with paragraph vectors. arXiv preprint arXiv:1507.07998.
- Furnkranz, J. (1998). A Study using N-gram Features for Text Categorization. Austrian Research Institute for Artificial Intelligence Technical Report OEFAI-TR-98-30 Schottengasse.
- Le, Q., & Mikolov, T. (2014, January). Distributed representations of sentences and documents. In International Conference on Machine Learning (pp. 1188-1196).
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).
- Paltoglou, G. and Thelwall, M. (2010). A Study of Information Retrieval Weighting Schemes for Sentiment Analysis. In Proceddings of the 48th Annual Meeting of the Association for Computational Linguistics: 1386-1395.
- Park, E., Kang, P., & Cho, S. (2016). Supervised paragraph vector: document classification by jointly embedding words, documents, and class labels.
- Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global Vectors for Word Representation. In EMNLP (Vol. 14, pp. 1532-1543).
- Roelleke, T. (2013). Information Retrieval Models: Foundations and Relationships. Morgan & Claypool Publishers.
- Rong, X. (2014). word2vec parameter learning explained. arXiv preprint arXiv:1411.2738.

References

Other materials

- Figure in the front page: http://slideplayer.com/slide/7073400/
- Bojanowski, P. (2016). fastest: A library for efficient text classification and word representation. https://www.youtube.com/watch?v=CHcExDsDeHU
- CS224d-L2: Lecture note on "Simple Word Vector representations: word2vec, GloVe", http://cs224d.stanford.edu/lectures/CS224d-Lecture2.pdf
- CS224d-L3: Lecture note on "Advanced word vector representations: language models, softmax, single layer networks", http://cs224d.stanford.edu/lectures/CS224d-Lecture3.pdf
- Images in the title page: http://nlp.stanford.edu/projects/glove/
- Kauchak, D. (2009). TF-IDF. http://www.cs.pomona.edu/~dkauchak/classes/f09/cs160-f09/lectures/lecture5-tfidf.pdf
- Kim, Y., Hernite, Y., Sontag, D., and Rush, A. (2016). Character-Aware Neural Language Models, AAAI 2016 presentation slides, http://www.people.fas.harvard.edu/~yoonkim/data/aaai_2016_slides.pdf
- McCormick, C. Word2Vec Tutorial Part I & 2, http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/
- Nayak, P. and Raghavan, P. (2014). Lecture 6: Scoring, Term Weighting and the Vector Space Model. http://web.stanford.edu/class/cs276/handouts/lecture6-tfidf-handout-1-per.pdf
- Word2Vec online example: http://bit.ly/wevi-online
- 박은정, Supervised Feature Representationsfor Document Classification, PhD Thesis, Seoul National University, 2016.
- 이재천, 김수경, 홍성연. (2015). Data Mining을 이용한 고려대 강의평가 분석: Klue 사이트 강의평가를 기준으로. 2015 캡스톤디자인 II Term Project.