인공지능특론
AAI

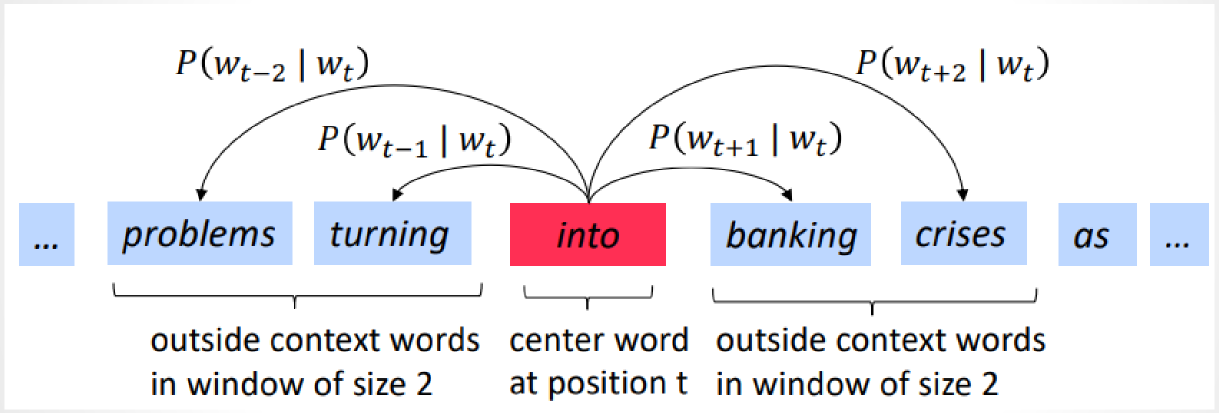
Word-to-Vector

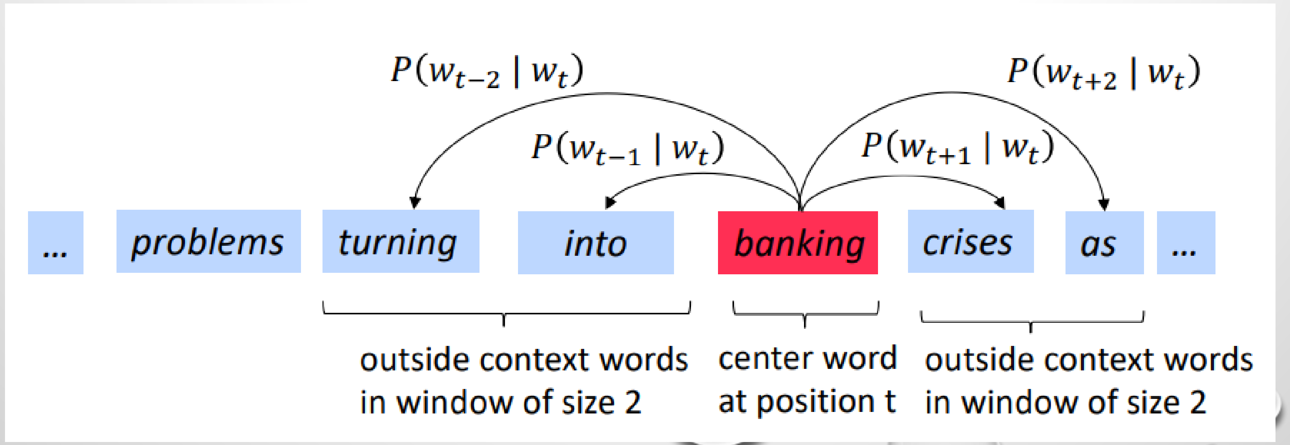
Word2Vec

- Concept
  - We have a large corpus of text
  - Every word in a fixed vocabulary is represented by a vector
  - Go through each position t in the text, which has a center word c and context words o
  - Use the similarity of the word vectors for c and o to calculate the probability of o given c
  - Keep adjusting the word vectors to maximize this probability

Word2Vec

- Using LM probability (target word prediction)





Word2Vec

- Objective function
  - For each position $t = 1, \cdots, T$, predict context words within a window of fixed size m, given center word w_j

$$\underset{\uparrow}{L(\theta)} = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} P\left(w_{t+j} \mid w_t; \theta\right)$$

  - The objective function $J(\theta)$ is the (average) negative log likelihood

$$\underset{\downarrow}{J(\theta)} = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} \underset{\uparrow}{\log P\left(w_{t+j} \mid w_t; \theta\right)}$$

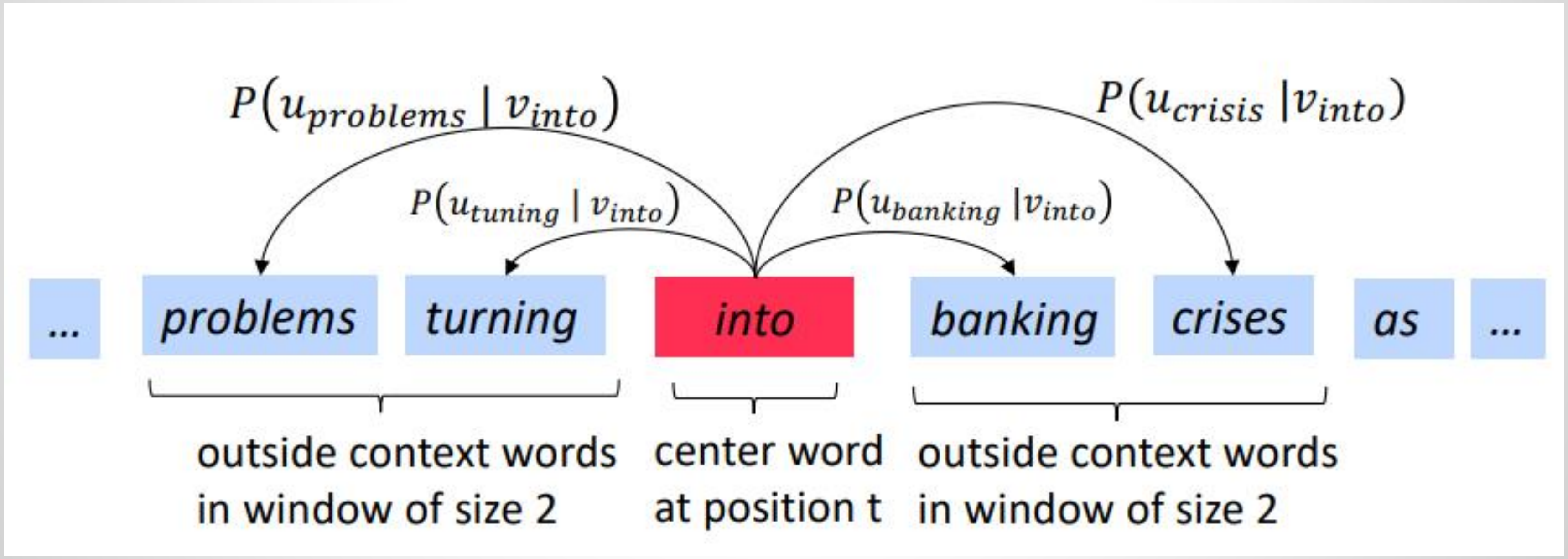
Word2Vec

- Objective function
  - Minimization of $J(\theta)$
  - Problems
    - How to calculate $P(w_{t+j}|w_t;\theta)$?
  - Hint
    - Word similarity
    - We will use two vectors per word w:
      - v_w when w is a center word
      - u_w when w is a context word
  - Then for a center word c and a context word o

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Word2Vec

- Center/context word vector based probability model



$P(u_{problems} \mid v_{into})$
$P(u_{turning} \mid v_{into})$
$P(u_{banking} \mid v_{into})$
$P(u_{crisis} \mid v_{into})$
...
problems
turning
into
banking
crises
as
...
outside context words in window of size 2
center word at position t
outside context words in window of size 2

Word2Vec

- Prediction function

Exponentiation makes anything positive

$$P(o|c) = \frac{\exp(\boxed{u_o^T v_c})}{\boxed{\sum_{w \in V} \exp(u_w^T v_c)}}$$

Dot product compares similarity of *o* and *c*.
$u^T v = u.v = \sum_{i=1}^{n} u_i v_i$
Larger dot product = larger probability

Normalize over entire vocabulary
to give probability distribution

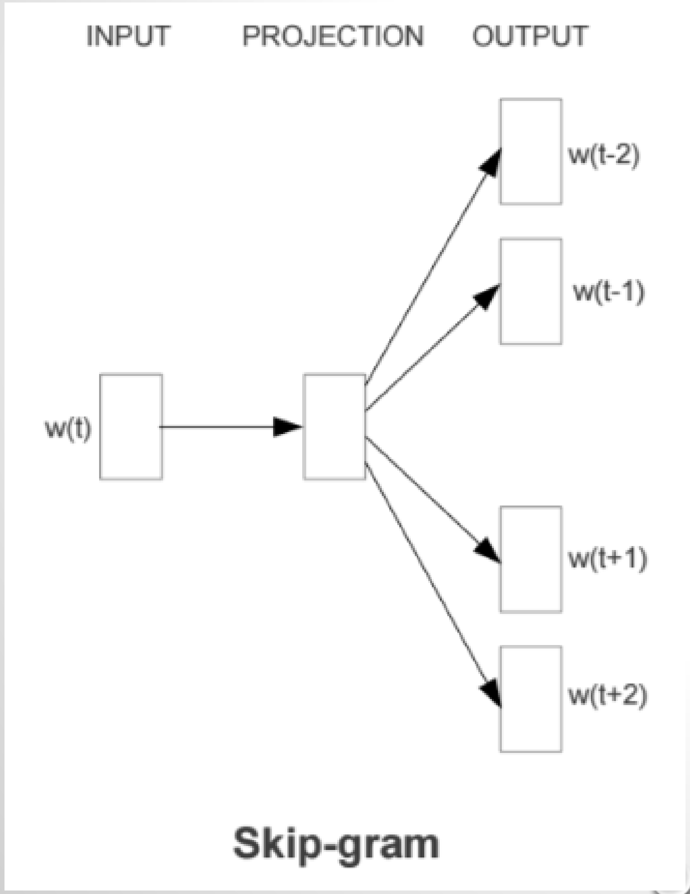
  - This is an example of the softmax function

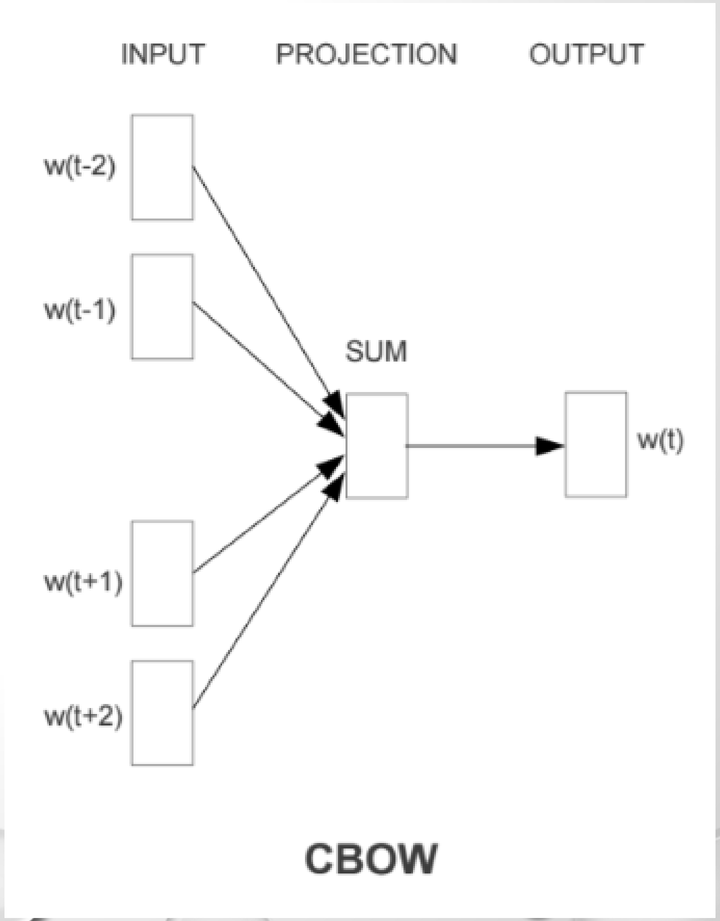
$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

    - “max” because amplifies probability of largest
    - “soft” because still assigns some probability to smaller

Word2Vec

- Two model variants





Word2Vec

- Objective function

CBOW	$P(c\|o) = log \frac{\exp(u_c^T \hat{v}_o)}{\sum_{w=1}^{\|V\|} \exp(u_w^T \hat{v}_o)}$
Skip-gram	$P(o\|c) = log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{\|V\|} \exp(u_w^T v_c)}$

- $P(c|o)$ maximization : predicts context words using center word
- $P(o|c)$ maximization : predicts center word using context words

Continuous BOW

- Training

$$J = -logP(w_c|w_{c-m},\cdots,w_{c-1},w_{c+1},\cdots,w_{c+m})$$

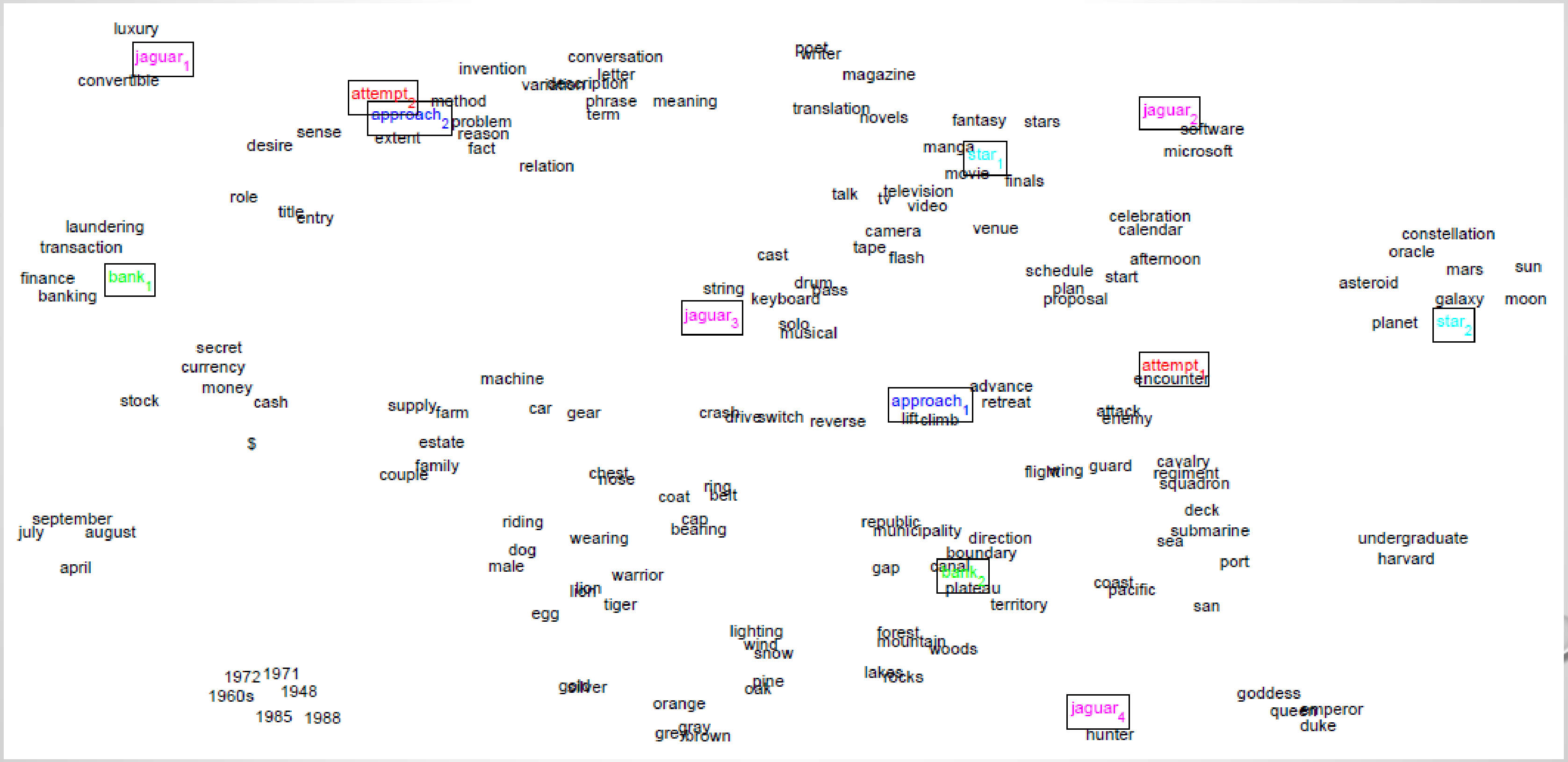
$$= -logP(u_c|\hat{v}_o) = -log\frac{\exp(u_c^T\hat{v}_o)}{\sum_{w=1}^{|V|}\exp(u_w^T\hat{v}_o)}$$

$$= -u_c^T\hat{v}_o + log\sum_{w=1}^{|V|}\exp(u_w^T\hat{v}_o)$$

$$\frac{\partial J}{\partial \hat{v}_o} = -u_c^T + \sum_{w=1}^{|V|}\frac{\exp(u_w^T\hat{v}_o)\,u_w}{\sum_{w=1}^{|V|}\exp(u_w^T\hat{v}_o)}$$

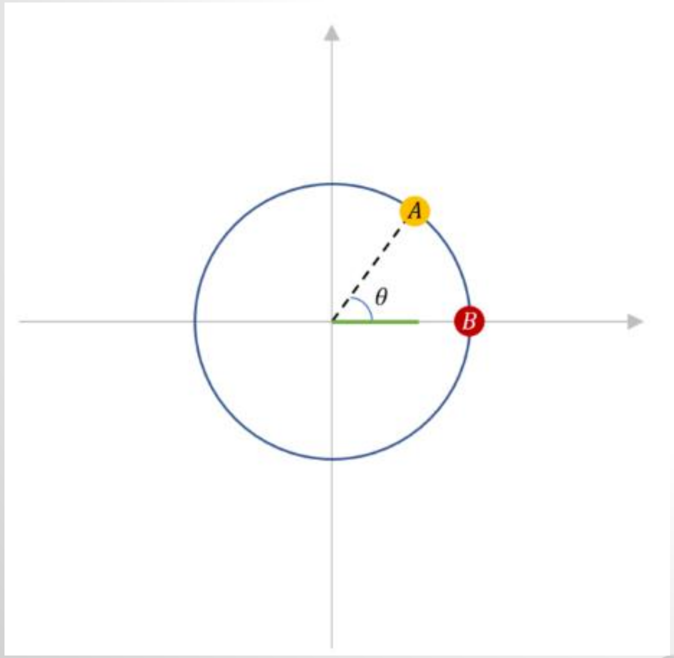
$$v_o^{t+1} = v_o^t - \alpha(-u_c^T + \sum_{w=1}^{|V|}P(w_w|\widehat{w}_o)u_w)$$

Continuous BOW



Continuous BOW

- Objective function
  - Training the relation among center word and context words
- Cosine similarity



- $\theta = 0$
  - A & B vectors are same direction
  - Inner dot product = 1
- $\theta = 90$
  - Inner dot product = 0
- $\theta = 180$
  - A 및 B vectors are opposite direction
  - Inner dot product = -1

Skip-gram

- Training

$$J = -logP(w_{c-m}, \cdots, w_{c-1}, w_{c+1}, \cdots, w_{c+m} | w_c)$$

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

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

$$= - \sum_{j=0, j\neq m}^{2m} u_{c-m+j}^T v_c + 2mlog \sum_{w=1}^{|V|} \exp(u_w^T v_c)$$

$$\frac{\partial J}{\partial v_c} = - \sum_{j=0, j\neq m}^{2m} u_{c-m+j}^T + 2m \sum_{w=1}^{|V|} P(w_{c-m+j} | w_c) u_w$$

Skip-gram

- Training

$$\frac{\partial J}{\partial v_c} = -\sum_{j=0, j\neq m}^{2m} u_{c-m+j}^T + 2m\sum_{w=1}^{|V|} P(w_{c-m+j}|w_c)u_w$$

For one context

$$\frac{\partial J}{\partial v_c} = -u_{c-i}^T + 2m\sum_{w=1}^{|V|} P(w_{c-i}|w_c)u_w$$

$$v_c^{t+1} = v_c^t - \alpha(-u_{c-i}^T + \sum_{w=1}^{|V|} P(w_w|w_c)u_w)$$

Skip-gram

- Objective function
  - Training the relation among center word and context words
  - Maximize :

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

- Inner product $u_o^T v_c$ maximization → locate nearby in vector space
  - Center word와 context words are a similar location
- Inner product $u_w^T v_c$ minimization → locate away in vector space
  - The words that appear regardless of the center word are far away

Skip-gram

- Center word & context words
  - Fixed context window
- Given a window size,
  - Decide context by moving by context window size
  - Words within the context window get closer in vector space, and Words not in the context window are learned away from the vector space
  - ***Recall count-based embedding***
    - Word2Vec preserves information of frequently appearing words
    - Methodology for learning count-based embedding

Skip-gram

- Subsampling frequent words
  - Word2vec's parameter : v vectors, u vectors
  - Matrix $V \times N$, $N \times V$ (ex, V=100,000 words & N=100-dim embedding)
  - Computation increases exponentially as words increase
  - Solution:
    - Reduce the amount of learning for frequently occurring words
    - $P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$
      - $f(w_i)$: Frequency of occurrence of the word/number of total words, t=0.00001

Skip-gram

- Negative sampling
  - Output layer is softmax
  - In case of 100,000 words, the calculation of softmax is huge
  - Softmax does not target all words but computes through sampling only a few words
  - Extracts negative examples

Skip-gram

- Negative sampling
  - (w, c) : word w & context pair
  - $P(D = 1|w, c)$: probability that pair (w, c) is in corpus
  - $P(D = 0|w, c)$: probability that pair (w, c) is not in corpus
- Probability density
  - $P(D = 1|w, c, \theta) = \frac{\exp(u_w^T v_c)}{1+\exp(u_w^T v_c)} = \sigma(u_w^T v_c)$
  - If Pair (w, c) is in corpus, $P(D = 1|w, c, \theta)$ is maximized
  - Otherwise, parameter θ is learned for maximization of $P(D = 0|w, c, \theta)$

Skip-gram

- Negative sampling
  - Likelihood

$$
\begin{aligned}
\theta &= \arg\max_{\theta} \prod_{(w,c)\in D} P(D=1|w,c,\theta) \prod_{(w,c)\in \tilde{D}} P(D=0|w,c,\theta) \\
&= \arg\max_{\theta} \prod_{(w,c)\in D} P(D=1|w,c,\theta) \prod_{(w,c)\in \tilde{D}} (1-P(D=1|w,c,\theta)) \\
&= \arg\max_{\theta} \sum_{(w,c)\in D} log P(D=1|w,c,\theta) + \sum_{(w,c)\in \tilde{D}} log(1-P(D=1|w,c,\theta)) \\
&= \arg\max_{\theta} \sum_{(w,c)\in D} log \frac{1}{1+exp(-u_w^T v_C)} + \sum_{(w,c)\in \tilde{D}} log(1-\frac{1}{1+exp(-u_w^T v_c)}) \\
&= \arg\max_{\theta} \sum_{(w,c)\in D} log \frac{1}{1+exp(-u_w^T v_C)} + \sum_{(w,c)\in \tilde{D}} log(\frac{1}{1+exp(u_w^T v_c)})
\end{aligned}
$$

Skip-gram

- Negative sampling
  - Loss function

$$J = -\sum_{(w,c)\in D} log \frac{1}{1+exp(-u_w^T v_c)} - \sum_{(w,c)\in \tilde{D}} log(\frac{1}{1+exp(u_w^T v_c)})$$

  - For Center word w_c, objective function with context word w_{c-m+j}
    - $-\log \sigma\left(u_{c-m+j}^T v_c\right) - \sum_{k=1}^{K} \log \sigma(-u_k^T w_c)$
  - Sampling tip for decision of k words

is: $0.9^{3/4} = 0.92$

Constitution: $0.09^{3/4} = 0.16$

bombastic: $0.01^{3/4} = 0.032$

Skip-gram

- Negative sampling
  - Sampling tip for decision of k words

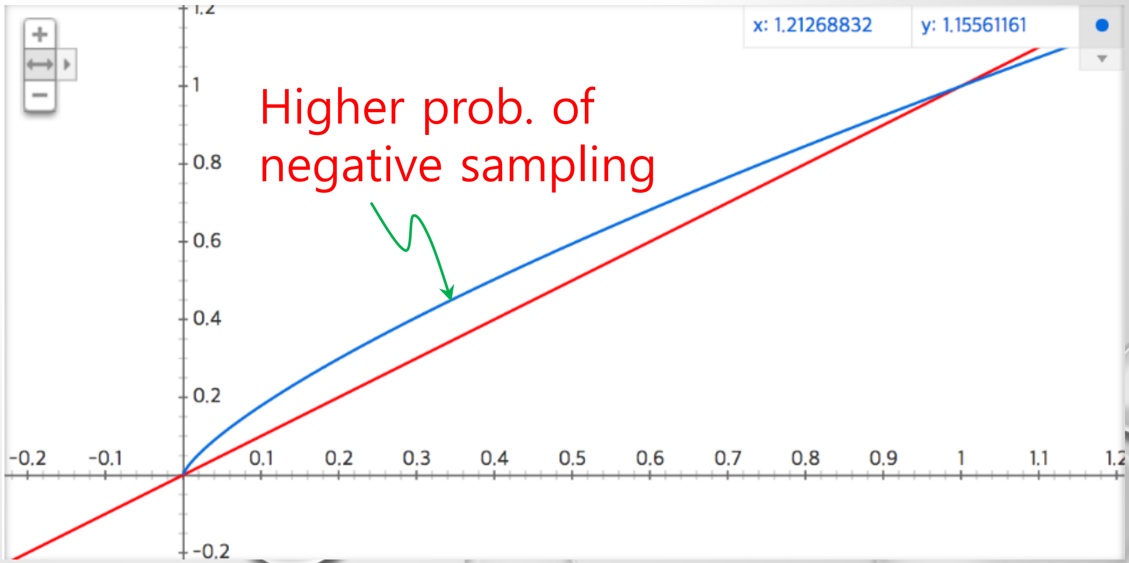
is: $0.9^{3/4} = 0.92$

Constitution: $0.09^{3/4} = 0.16$

bombastic: $0.01^{3/4} = 0.032$

  - Weighting based on the frequency of words
  - The more words that appear, the higher the probability of negative sampling

$$p(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=1}^{10000} f(w_j)^{3/4}}$$

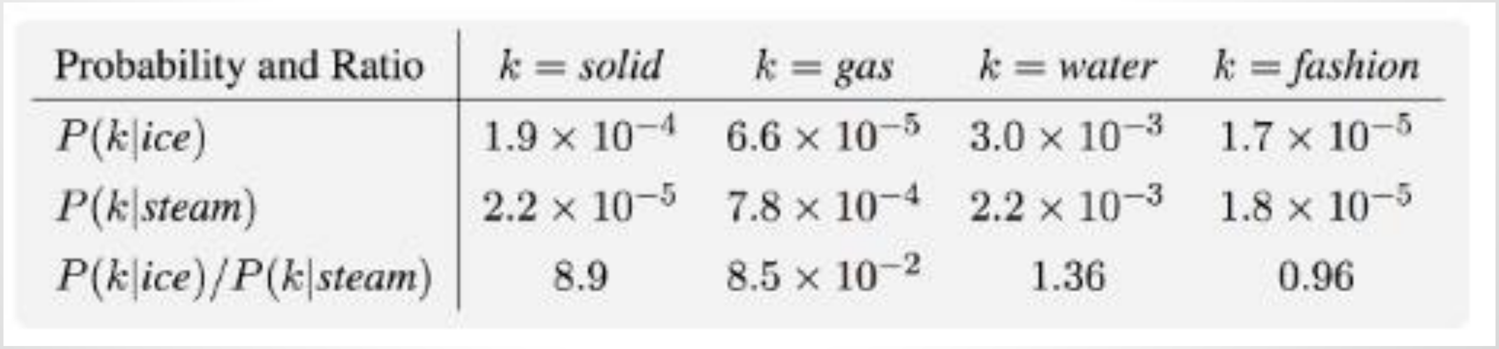


GloVe

- Word2Vec weakness
  - While Word2Vec (Skip-Gram) has good performance in measuring similarity between word vectors embedded in low-dimensional vector space, due to learning / analysis is performed only within the window size specified by the user, it is difficult to reflect co-occurrence information of the whole corpus

GloVe

- Word2Vec
  - Inner dot product is used for similarity
- Glove
  - The inner dot product of two embedded word vectors is defined as the probability of simultaneous appearance in the whole corpus

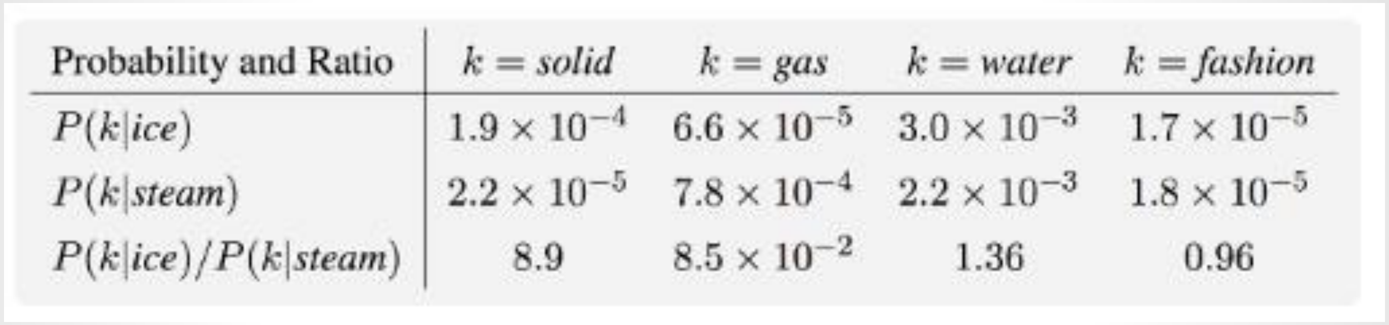


Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

  - When embedding the word 'solid' into a vector space, it means to embed it closer to the 'ice' side of 'ice' or 'steam'

GloVe

- Co-occurrence probability of the whole corpus

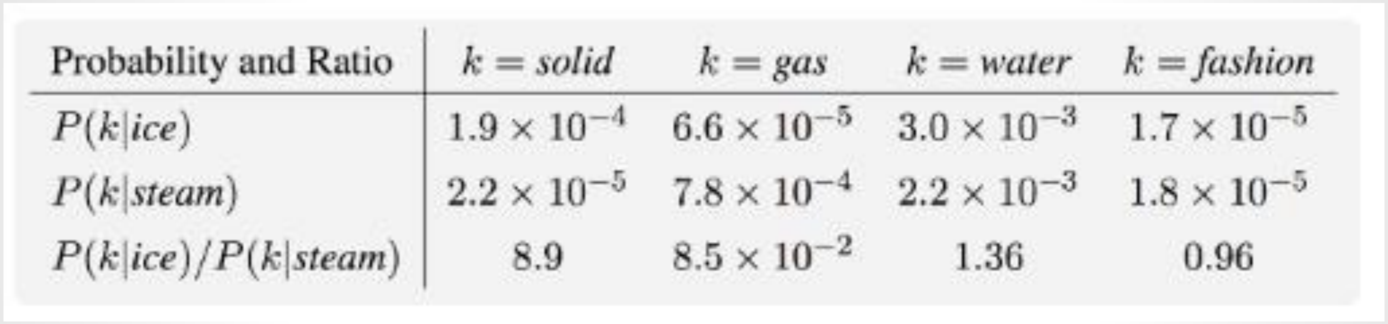


Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

  - The probability of the word 'solid' will appear is higher when 'ice' is given than when 'stream' is given $\frac{P(soild|ice)}{P(soild|stream)} = 8.9$
  - For 'gas', the probability ratio of all words is low
  - For the words 'ice' and 'stream', The probability ratio between 'water' that is highly correlated and 'fashion' that is less correlated is close to 1

GloVe

- Co-occurrence probability of the whole corpus



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

- Idea of GloVe
  - When learning the whole corpus, ice or steam may correspond to symbol, and if the ratio information between words is reflected in the corpus at the same time, more accurate word embedding can be achieved

GloVe

- Objective function

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

  - $P_{ik} = p(w_k|w_i)$: occurrence probability of kth word for ith word
  - $P_{ik} = \frac{C_{ik}}{\sum_k c_{ik}} = \frac{c_{ik}}{c_i}$: C_{ik} is co-occurrence frequency of kth word and ith word

$$F(w_{ice}, w_{steam}, w_{solid}) = \frac{P_{ice,solid}}{P_{steam,solid}} = \frac{P(solid|ice)}{P(solid|steam)} = \frac{1.9 \times 10^{-4}}{2.2 \times 10^{-5}} = 8.9$$

GloVe

- Objective function

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

  - P_{ik} is a count-based method that preserves information that appears together as a probability of simultaneous appearance → When they appear together, they reflect the increase in the inner dot product
    - $P_{ik} = F(w_i^T \widetilde{w}_k)$
  - For relationship among three words $w_i, w_j, \widetilde{w}_k$, if $F(w_i, w_j, \widetilde{w}_k)$ and $F(w_i, \widetilde{w}_k)$ are close after subtracting w_j from w_i, it is considered that the relationship between $\widetilde{w}_k$ and w_i is high
    - $F(w_i, w_j, \widetilde{w}_k) = \mathrm{F}((w_i - w_j)^T \widetilde{w}_k)$

GloVe

- Objective function

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

$$\mathrm{F}\left(\left(w_i - w_j\right)^T \widetilde{w}_k\right) = F(w_i^T \widetilde{w}_k - w_j^T \widetilde{w}_k) = \frac{F(w_i^T \widetilde{w}_k)}{F(w_j^T \widetilde{w}_k)}$$

  - Satisfying the above conditions is an exponential function

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

$$w_i^T \widetilde{w}_k = \log P_{ik} = \log C_{ik} - \log C_i$$

  - Symmetric satisfaction : In the whole corpus, the words w_i and w_k can be the center and surrounding words

GloVe

- Objective function

$$w_i^T \widetilde{w}_k = \log P_{ik} = \log C_{ik} - \log C_i$$

$$w_k^T \widetilde{w}_i = \log P_{ki} = \log C_{ki} - \log C_k$$

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

- Loss function

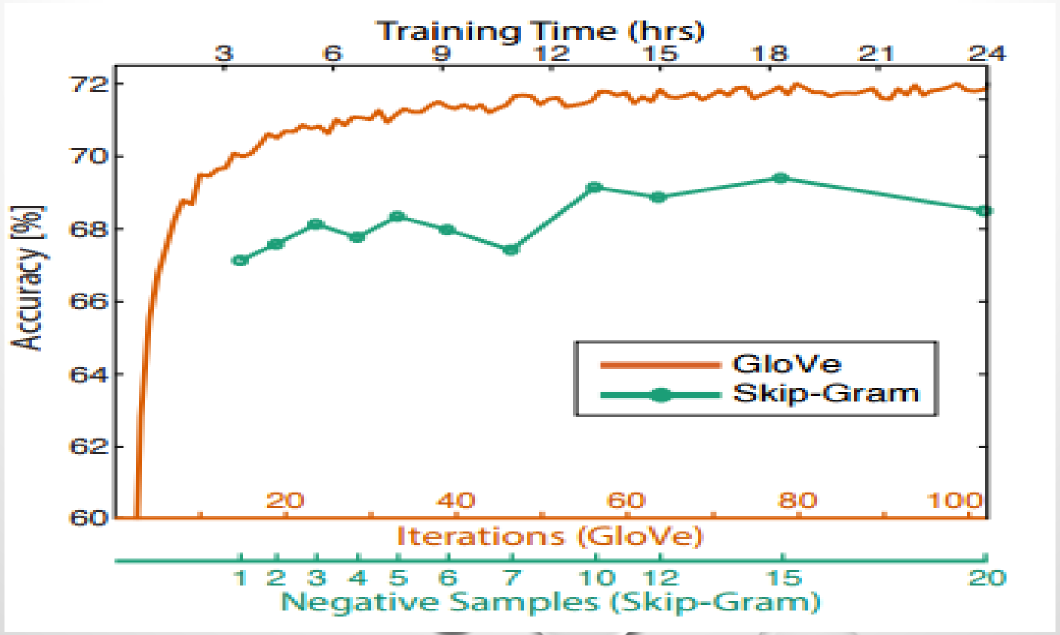
$$J = \sum_{i,j=1}^{|V|} (w_i^T \widetilde{w}_k + b_i + \tilde{b}_k - \log C_{ik})^2$$

$$J = \sum_{i,j=1}^{|V|} f(C_{ik})(w_i^T \widetilde{w}_k + b_i + \tilde{b}_k - \log C_{ik})^2$$

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

GloVe

- Implementation
  - For whole corpus, co-occurrence matrix is needed to compute
  - 10,000 words case: 10,000x10,000 matrix elements should be computed → computation load



FastText

- Word2Vec extention
  - Dividing Word into sub-word units to learn vector space
  - OOV (Out-of-vocabulary) problem can be solved
- Each word is represented as a character n-gram
  - For ‘apple’, 3-gram character division
    - “<ap”, “app”, “ppl”, “ple”, “le>” : five embeddings
  - If word ‘birthplace’ is OOV
    - Create word vectors from sub-words of words ‘birth’ and ‘place’, and “thp” and “hpl” vectors

FastText

- Low frequency word problem is solved
  - The word2vec has a disadvantage that it does not learn much about low-frequency words
  - But, Embedding accuracy is somewhat poor
- Sub-word embedding is available
  - Advantages of increased learning frequency regardless of word frequency
  - Improved embedding accuracy of low frequency words

FastText

- FastText for Korean
  - For '자연어처리'
    - Character 3-gram-level embedding

<자연 , 자연어 , 연어처 , 어처리 , 처리>

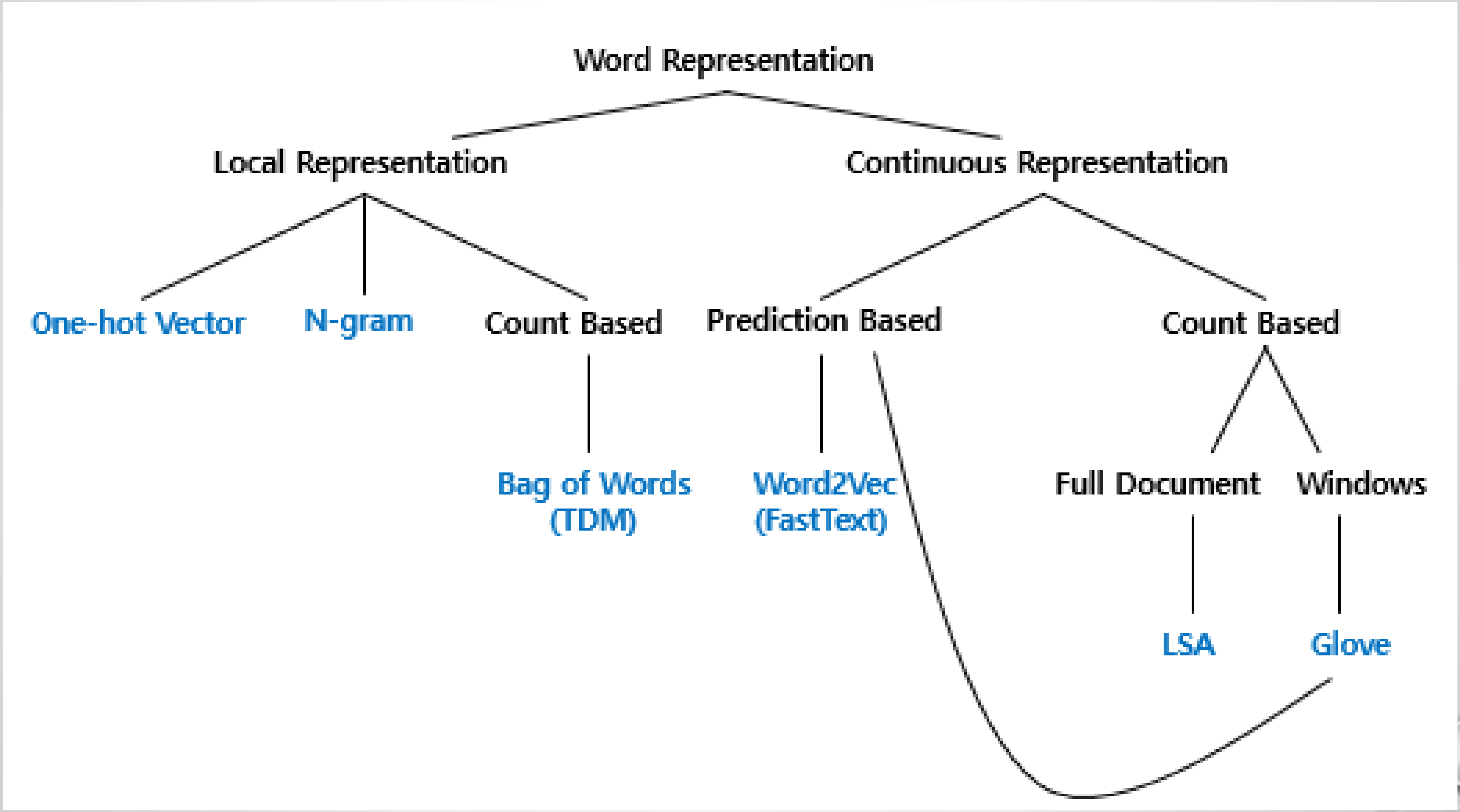
    - 초성, 중성, 종성 3-gram-level embedding

ㅈ ㅏ _ ㅇ ㅕ ㄴ ㅇ ㅓ _ ㅊ ㅓ _ ㄹ ㅣ _

↓

<ㅈㅏ ㅈㅏ_ ㅏ_ㅇ ...... ㅣ_>

Word Representation



Latent Semantic Analysis (LSA)

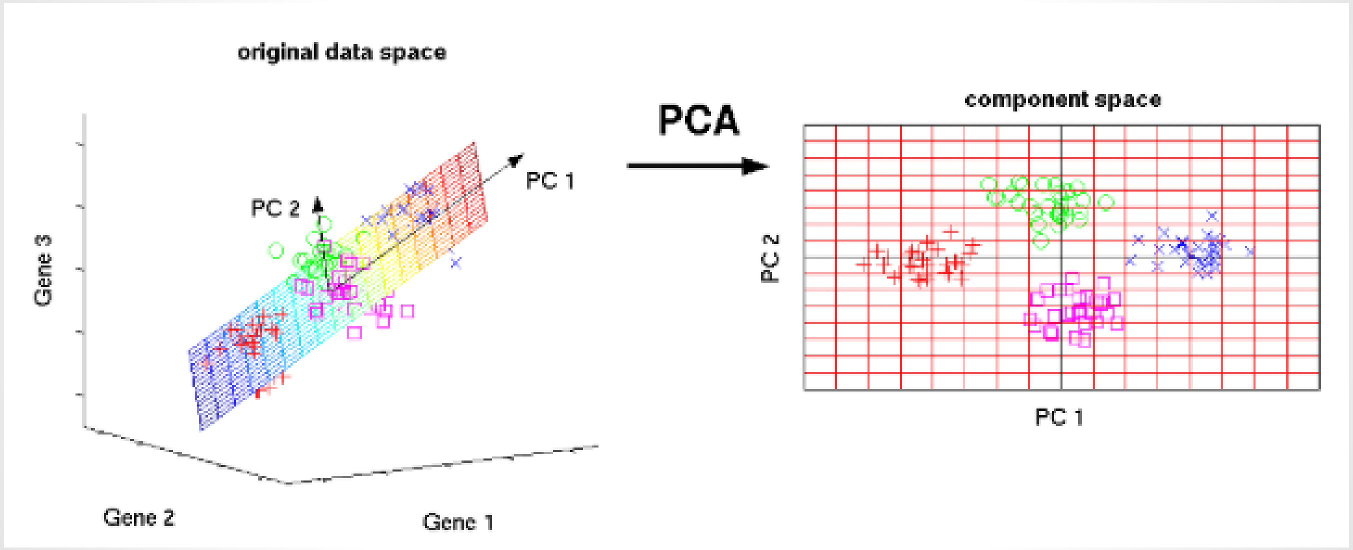
- 임의의 TDM 대상

doc1 : 나,는,학교,에,가,ㄴ,다
doc2 : 학교,에,가,는,영희
doc3 : 나,는,영희,는,좋,다

-	doc1	doc2	doc3
나	1	0	0
는	1	1	2
학교	1	1	0
에	1	1	0
가	1	1	0
ㄴ	1	0	0
다	1	0	1
영희	0	1	1
좋	0	0	1

Latent Semantic Analysis (LSA)

- PCA



  - A technique for converting samples from high-dimensional space into low-dimensional space by finding a new basis (axis) that is orthogonal to each other while preserving the variance of data as much as possible

Latent Semantic Analysis (LSA)

- PCA
  - For data matrix A
  - Co-variance matrix

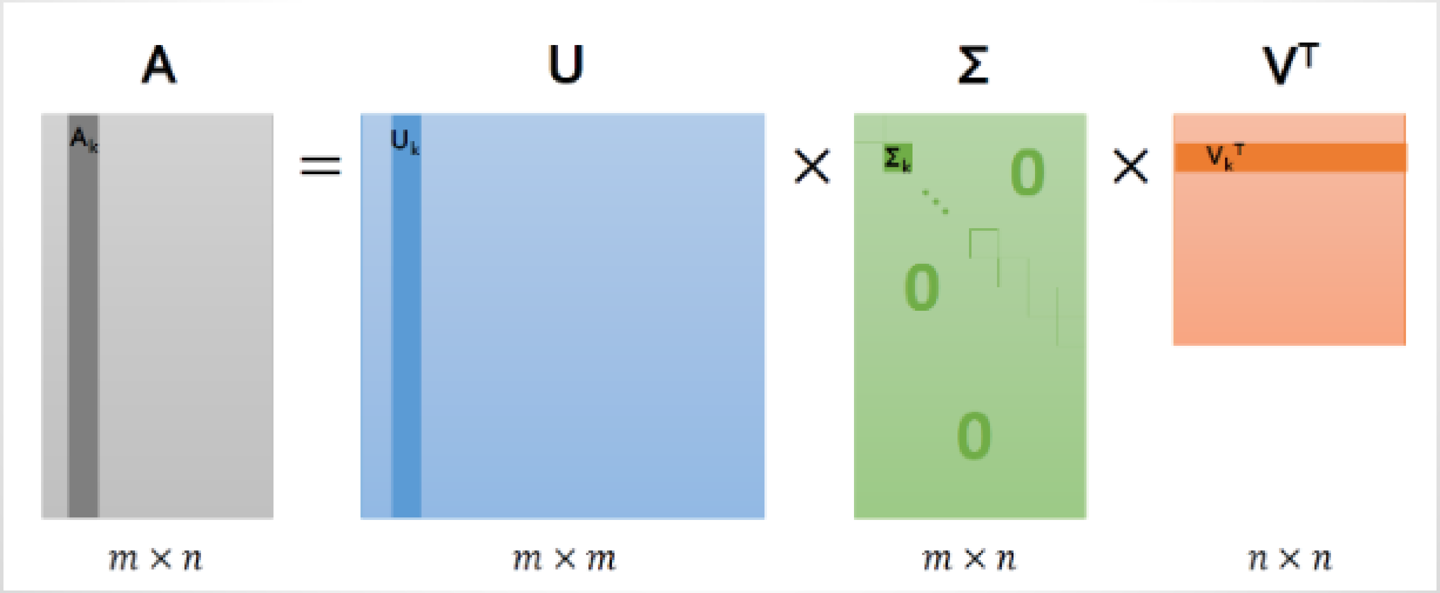
$$cov(A) = \frac{1}{n-1} A^T A \propto A^T A$$

  - Eigen value decomposition
    - Λ : diagonal elements are eigen value
    - V : column elements are eigen vector

$$A^T A = V \Lambda V^T$$

Latent Semantic Analysis (LSA)

- SVD
  - Decomposing the data matrix of arbitrary size into orthogonal components and sizes



Latent Semantic Analysis (LSA)

- SVD

$$A = U\Sigma V^T$$

  - Column vector of matrix U and V : singular vector, orthogonal

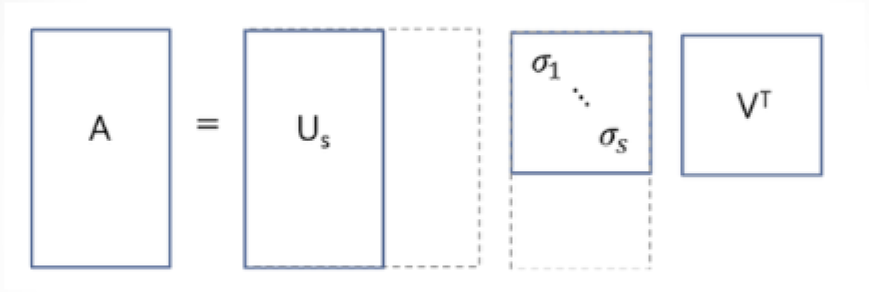
$$U = \begin{bmatrix} \overrightarrow{u_1} & \overrightarrow{u_2} & \dots & \overrightarrow{u_m} \end{bmatrix}$$
$$V = \begin{bmatrix} \overrightarrow{v_1} & \overrightarrow{v_2} & \dots & \overrightarrow{v_n} \end{bmatrix}$$
$$\overrightarrow{u_k} = \begin{bmatrix} u_{k1} \\ u_{k2} \\ \dots \\ u_{km} \end{bmatrix} \quad \overrightarrow{v_k} = \begin{bmatrix} v_{k1} \\ v_{k2} \\ \dots \\ v_{kn} \end{bmatrix}$$
$$\overrightarrow{u_k}^T \overrightarrow{u_k} = 1, \quad U^T U = I$$
$$\overrightarrow{v_k}^T \overrightarrow{v_k} = 1, \quad V^T V = I$$

$$\begin{aligned} A^T A &= (U\Sigma V^T)^T U\Sigma V^T \\ &= V\Sigma U^T U\Sigma V^T \\ &= V\Sigma^2 V^T \end{aligned}$$

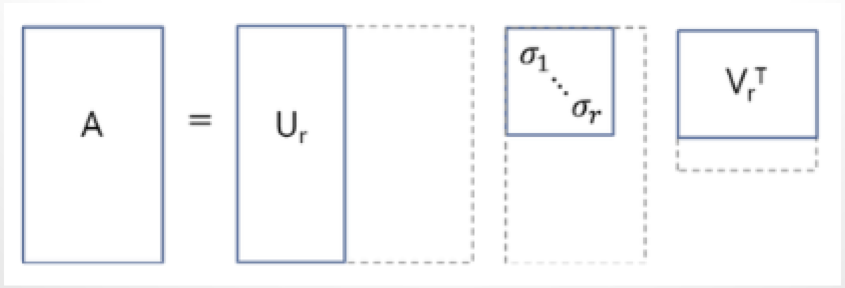
$$A^T A = V\Lambda V^T = V\Sigma^2 V^T$$

Latent Semantic Analysis (LSA)

- SVD
  - Thin SVD

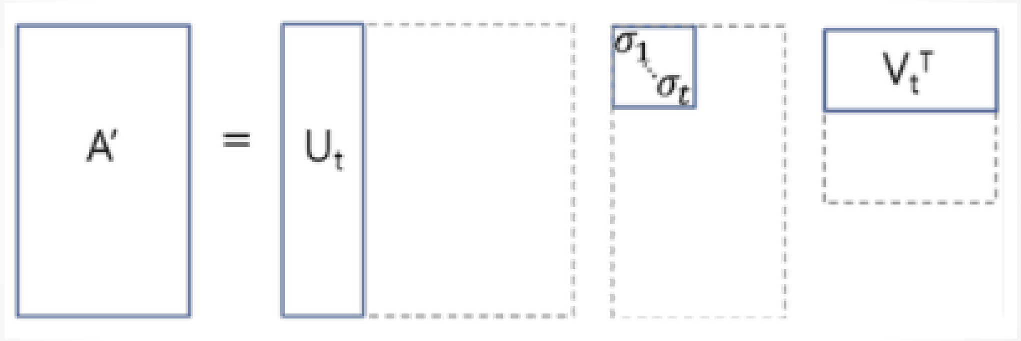


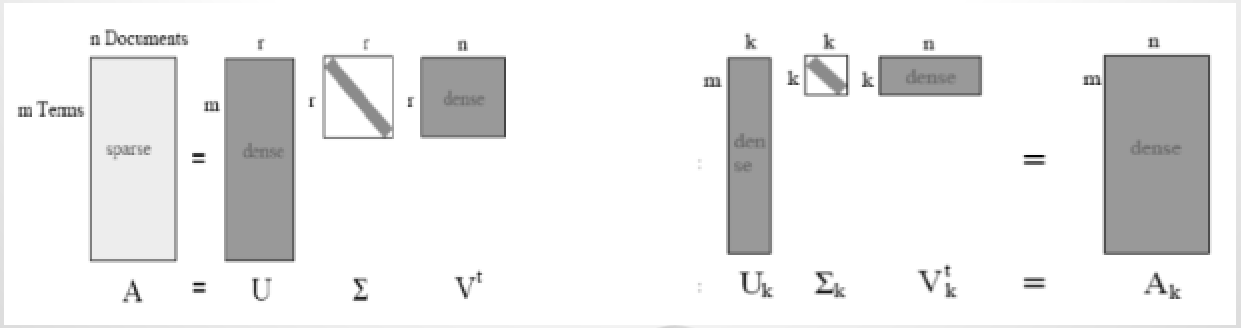
  - Compact SVD



Latent Semantic Analysis (LSA)

- SVD
  - Truncated SVD





Latent Semantic Analysis (LSA)

- SVD for TDM

-	doc1	doc2	doc3
나	1	0	0
는	1	1	2
학교	1	1	0
에	1	1	0
가	1	1	0
ㄴ	1	0	0
다	1	0	1
영희	0	1	1
좋	0	0	1

$$A = U\Sigma V^T$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 2 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -0.17 & 0.27 & -0.40 \\ -0.63 & -0.41 & -0.03 \\ -0.32 & 0.37 & 0.21 \\ -0.32 & 0.37 & 0.21 \\ -0.32 & 0.37 & 0.21 \\ -0.17 & 0.27 & -0.40 \\ -0.33 & -0.12 & -0.52 \\ -0.30 & -0.29 & 0.49 \\ -0.15 & -0.39 & -0.13 \end{bmatrix} \begin{bmatrix} 3.61 & 0 & 0 \\ 0 & 2.04 & 0 \\ 0 & 0 & 1.34 \end{bmatrix} \begin{bmatrix} -0.63 & -0.53 & -0.57 \\ 0.56 & 0.20 & -0.80 \\ -0.54 & 0.83 & -0.17 \end{bmatrix}$$

Latent Semantic Analysis (LSA)

- SVD for TDM
  - Applying Truncated SVD to utilize only the top two distributed information

$$\begin{bmatrix} -0.17 & 0.27 \\ -0.63 & -0.41 \\ -0.32 & 0.37 \\ -0.32 & 0.37 \\ -0.32 & 0.37 \\ -0.17 & 0.27 \\ -0.33 & -0.12 \\ -0.30 & -0.29 \\ -0.15 & -0.39 \end{bmatrix} \begin{bmatrix} 3.61 & 0 \\ 0 & 2.04 \end{bmatrix} \begin{bmatrix} -0.63 & -0.53 & -0.57 \\ 0.56 & 0.20 & -0.80 \end{bmatrix} = \begin{bmatrix} 0.71 & 0.44 & -0.09 \\ 0.97 & 1.04 & 1.99 \\ 1.15 & 0.76 & 0.04 \\ 1.15 & 0.76 & 0.04 \\ 1.15 & 0.76 & 0.04 \\ 0.71 & 0.45 & -0.09 \\ 0.62 & 0.58 & 0.88 \\ 0.36 & 0.45 & 1.11 \\ -0.09 & 0.14 & 0.97 \end{bmatrix}$$

$$A = U\Sigma V^T$$

Latent Semantic Analysis (LSA)

- SVD for TDM
  - Applying Truncated SVD to utilize only the top two distributed information

$$\begin{bmatrix} -0.17 & 0.27 \\ -0.63 & -0.41 \\ -0.32 & 0.37 \\ -0.32 & 0.37 \\ -0.32 & 0.37 \\ -0.17 & 0.27 \\ -0.33 & -0.12 \\ -0.30 & -0.29 \\ -0.15 & -0.39 \end{bmatrix} \begin{bmatrix} 3.61 & 0 \\ 0 & 2.04 \end{bmatrix} \begin{bmatrix} -0.63 & -0.53 & -0.57 \\ 0.56 & 0.20 & -0.80 \end{bmatrix} = \begin{bmatrix} 0.71 & 0.44 & -0.09 \\ 0.97 & 1.04 & 1.99 \\ 1.15 & 0.76 & 0.04 \\ 1.15 & 0.76 & 0.04 \\ 1.15 & 0.76 & 0.04 \\ 0.71 & 0.45 & -0.09 \\ 0.62 & 0.58 & 0.88 \\ 0.36 & 0.45 & 1.11 \\ -0.09 & 0.14 & 0.97 \end{bmatrix}$$

$$U_2 \Sigma_2 {V_2}^T = A'$$

Latent Semantic Analysis (LSA)

- SVD for TDM

$$\begin{bmatrix} 0.71 & 0.44 & -0.09 \\ 0.97 & 1.04 & 1.99 \\ 1.15 & 0.76 & 0.04 \\ 1.15 & 0.76 & 0.04 \\ 1.15 & 0.76 & 0.04 \\ 0.71 & 0.45 & -0.09 \\ 0.62 & 0.58 & 0.88 \\ 0.36 & 0.45 & 1.11 \\ -0.09 & 0.14 & 0.97 \end{bmatrix}$$

$$round(A') = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 2 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 2 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Latent Semantic Analysis (LSA)

- SVD for TDM

$$U_k^T A_k = U_k^T U_k \Sigma_k V_k^T = I \Sigma_k V_k^T = \Sigma_k V_k^T = X_1$$

$$A_k V_k = U_k \Sigma_k V_k^T V_k = U_k \Sigma_k \mathrm{I} = U_k \Sigma_k = X_2$$

$$X_1 = \begin{bmatrix} -2.28 & -1.90 & -2.07 \\ 1.14 & 0.42 & -1.64 \end{bmatrix}$$

document

$$X_2 = \begin{bmatrix} -0.63 & 0.56 \\ -2.30 & -0.84 \\ -1.16 & 0.76 \\ -1.16 & 0.76 \\ -1.16 & 0.76 \\ -0.63 & 0.56 \\ -1.20 & -0.24 \\ -1.10 & -0.60 \\ -0.57 & -0.80 \end{bmatrix}$$

word