

Word2Vec/Glove/FastText

Hyopil Shin(Seoul National University)

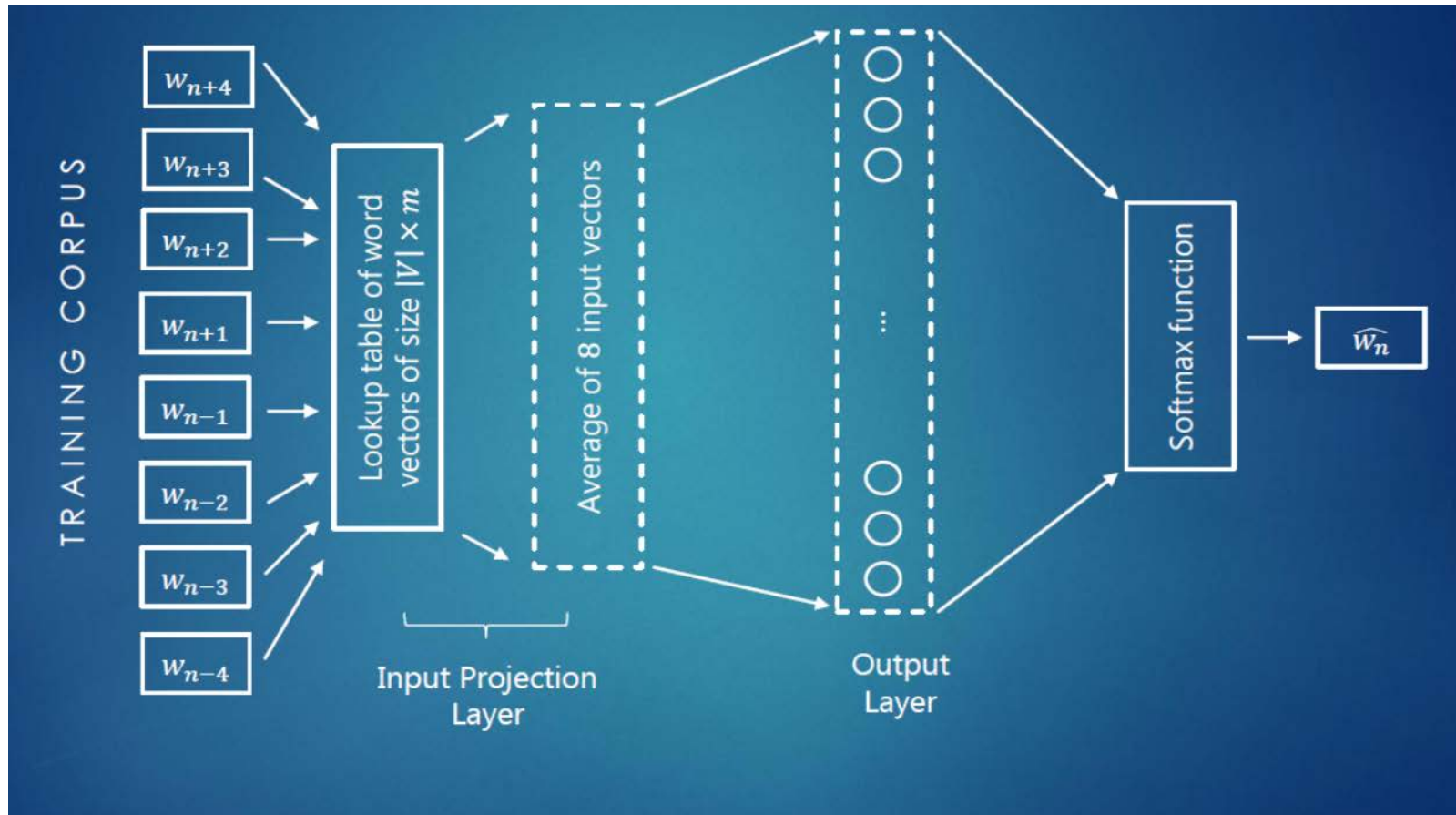
Computational Linguistics

#Semantics with Dense Vectors

Supplement Data

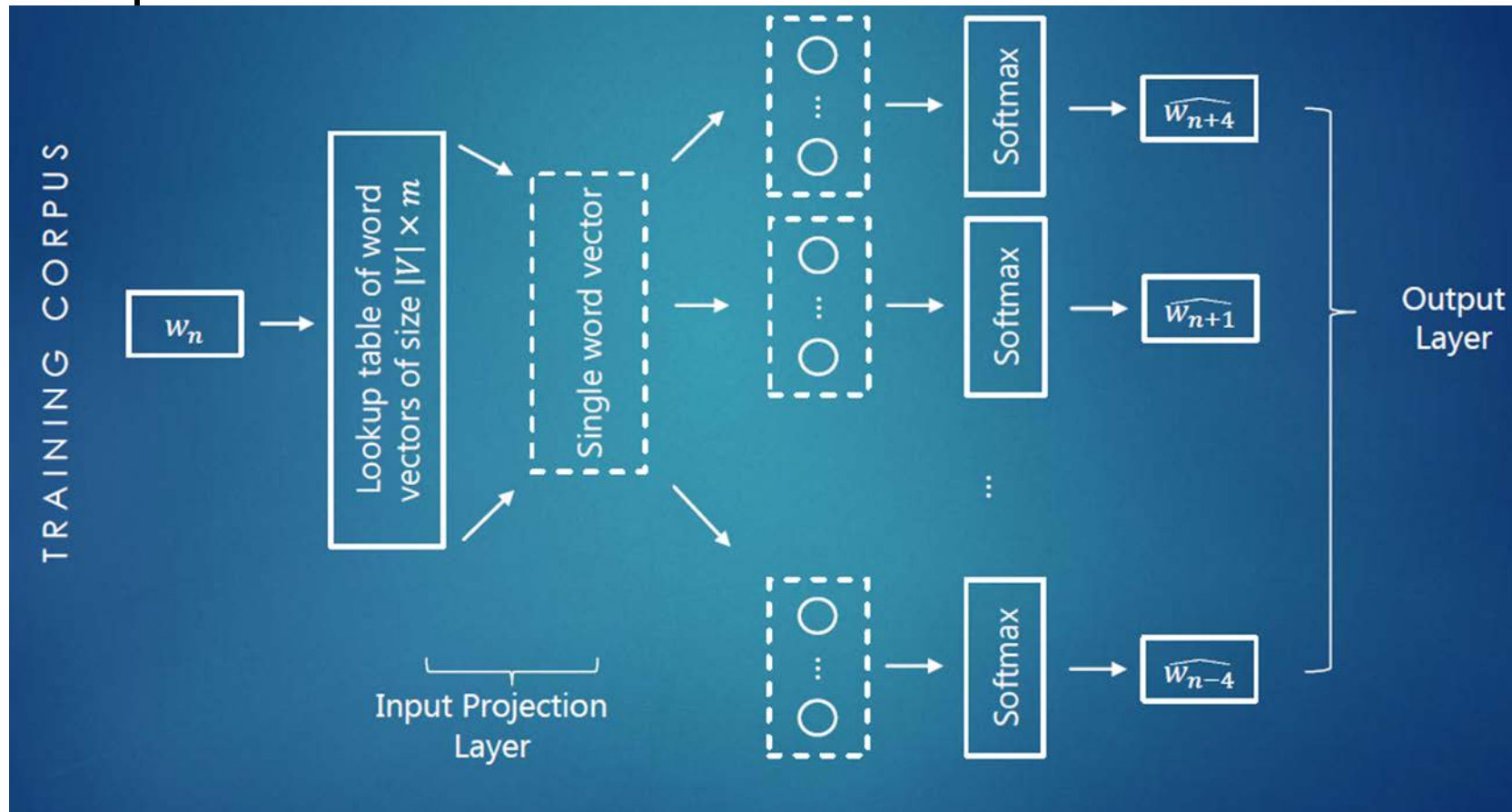
Word2Vec

- CBOW: Continuous Bag of words

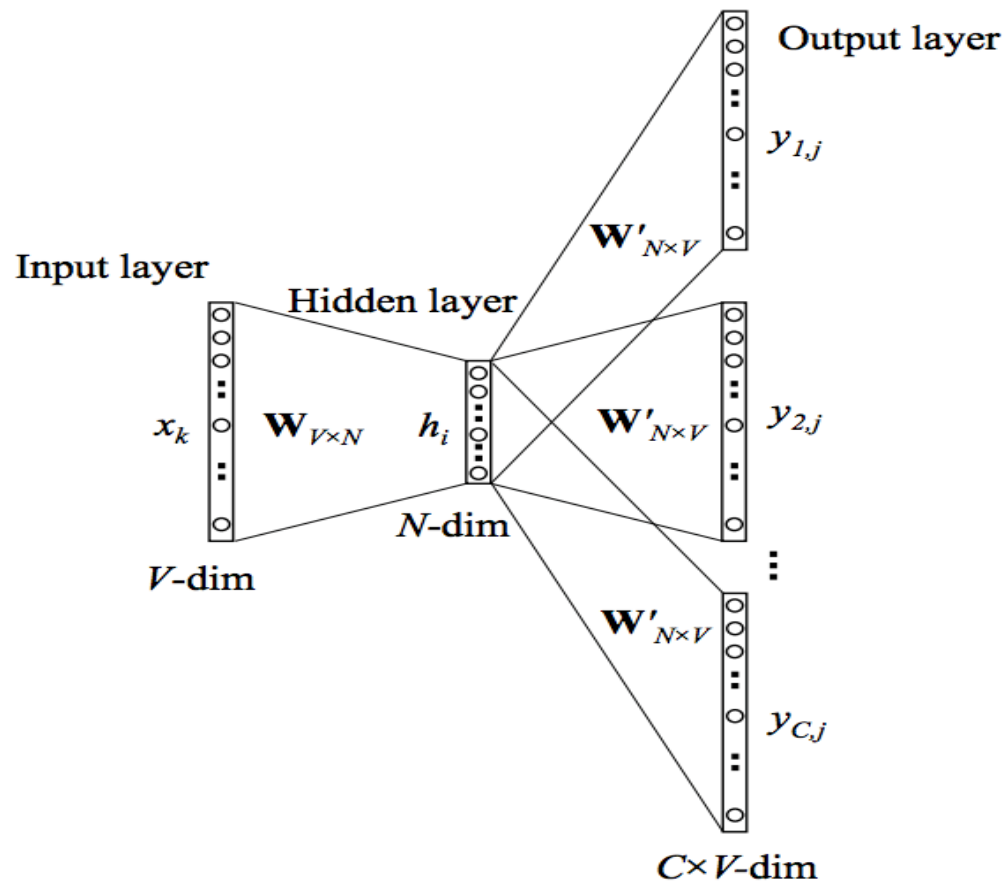


Word2Vec

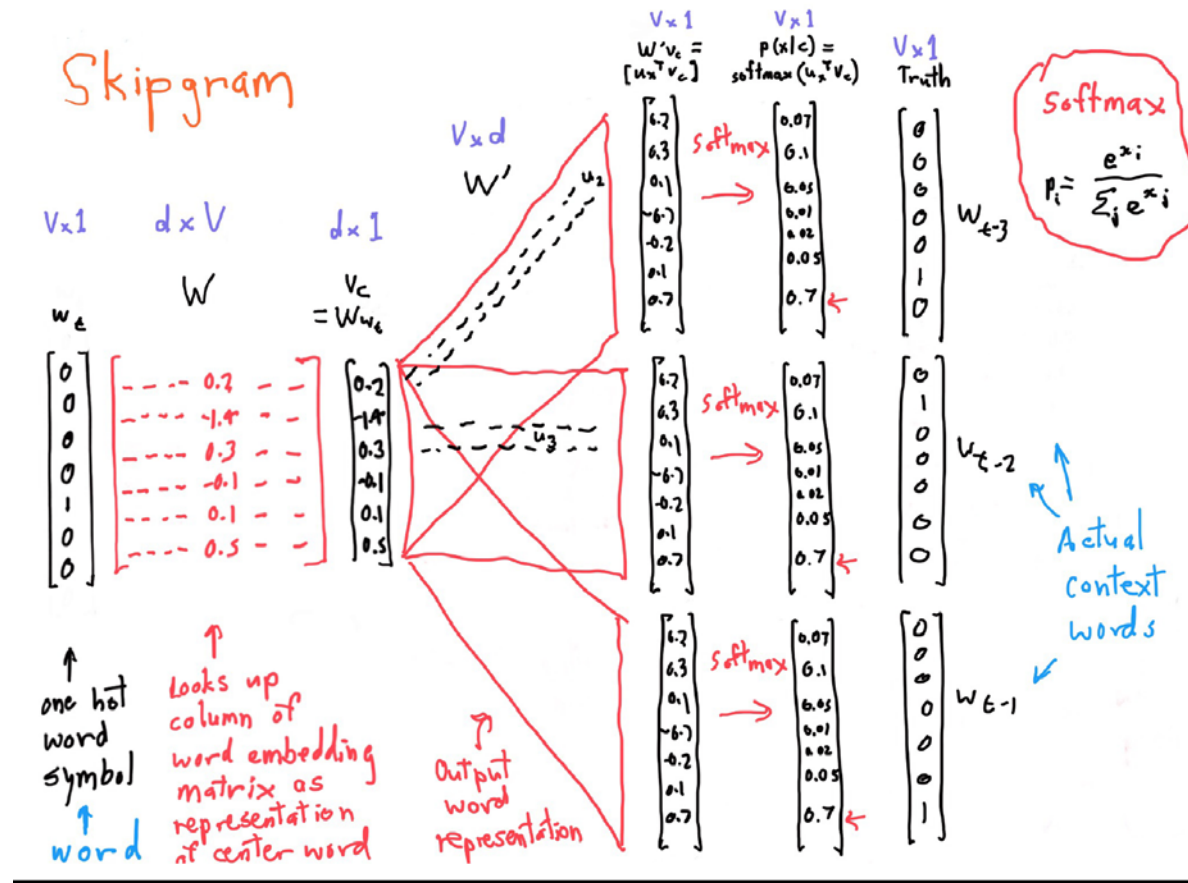
- Skip-Gram



Word2Vec 학습 패러미터(skip-gram)

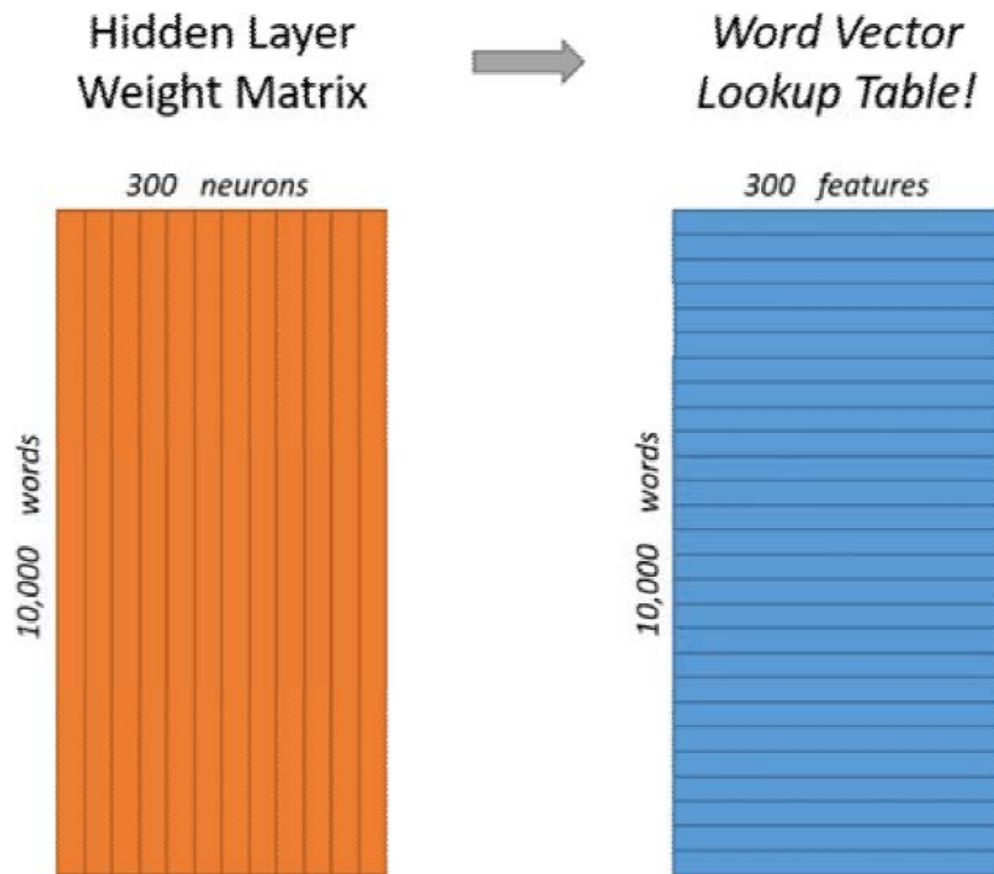


From (cs224n Natural Language Processing with Deep Learning at Stanford University)



Word2Vec

- V-임베딩 단어수
- N-은닉층의 노드 개수
- 중심단어로 주변단어를 맞추거나, 주변단어로 중심단어를 맞추기 위해 가중치 행렬인 W, W' 을 조금씩 업데이트 하면서 이루어지는 구조



Word2Vec

- 학습이 마무리되면 W의 행벡터들이 각 단어에 해당하는 임베딩이 됨

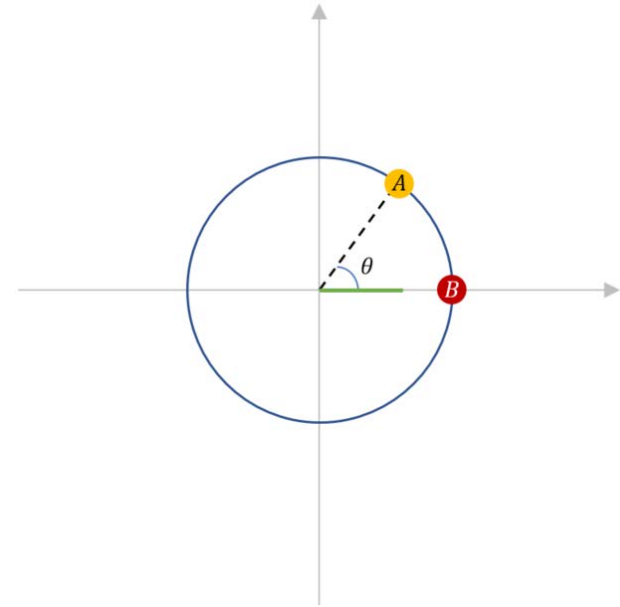
$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

Word2Vec 입력과 Skip-Gram

Source Text	Training Samples			
<table><tr><td>The</td><td>quick</td><td>brown</td></tr></table> fox jumps over the lazy dog. ➡	The	quick	brown	(the, quick) (the, brown)
The	quick	brown		
The <table><tr><td>quick</td><td>brown</td><td>fox</td></tr></table> jumps over the lazy dog. ➡	quick	brown	fox	(quick, the) (quick, brown) (quick, fox)
quick	brown	fox		
The quick <table><tr><td>brown</td><td>fox</td><td>jumps</td></tr></table> over the lazy dog. ➡	brown	fox	jumps	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
brown	fox	jumps		
The quick brown <table><tr><td>fox</td><td>jumps</td><td>over</td></tr></table> the lazy dog. ➡	fox	jumps	over	(fox, quick) (fox, brown) (fox, jumps) (fox, over)
fox	jumps	over		

Word2vec 학습 - 코사인 유사도

- A와 B가 겹칠 때($\Theta = 0$),
 $\cos(\Theta) = 1$
- $\cos(\Theta)$ 는 단위원 내 벡터
들끼리의 내적과 같음
- 내적이 커진다는 것은
두 벡터가 이루는 Θ 가
작아지고, 유사도는 높아
진다
- Cos인과 내적을 목적함
수에 활용



Word2Vec 학습(skip-gram): 목적함수

- Skip-gram: 중심단어(c)가 주어졌을 때 주변단어 (o)가 나타날 확률을 최대화 하는 식으로 학습이 이루어짐

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

- v : 입력층-은닉층을 잇는 가중치 행렬 W 의 행벡터
- u : 은닉층-출력층을 잇는 가중치 행렬 W 의 열벡터
- 최대화:
 - 분자 지수를 키움-중심단어(c)에 대한 주변단어(o) 벡터의 내적값을 높임. Θ 를 줄여 유사도를 높임
 - 분모는 중심단어 c 와 학습말뭉치 내 모든 단어를 각각 내적한 것의 총합. 분모를 줄이기 위해서는 주변에 등장하지 않은 단어와 중심단어의 내적값을 줄여야 함. Θ 를 크게함
 - 윈도우 내에 등장하지 않는 단어에 해당하는 벡터는 중심단어 벡터와 벡터 공간상에서 멀어지게(내적을 줄임), 등장하는 주변단어 벡터는 중심단어 벡터와 가까워지게(내적을 늘림) 하는 방식으로 벡터값을 업데이트 하면서 이루어짐

Word2Vec: 목적함수

$$\begin{aligned}\frac{\partial}{\partial v_c} \ln P(o|c) &= \frac{\partial}{\partial v_c} \ln \frac{\exp(u_o^T v_c)}{\sum_{w=1}^W \exp(u_w^T v_c)} \\&= \frac{\partial}{\partial v_c} u_o^T v_c - \frac{\partial}{\partial v_c} \ln \sum_{w=1}^W \exp(u_w^T v_c) \\&= u_o^T - \frac{1}{\sum_{w=1}^W \exp(u_w^T v_c)} \left(\sum_{w=1}^W \exp(u_w^T v_c) \cdot u_w \right) \\&= u_o^T - \sum_{w=1}^W \frac{\exp(u_w^T v_c)}{\sum_{w=1}^W \exp(u_w^T v_c)} \cdot u_w \\&= u_o^T - \sum_{w=1}^W P(w|c) \cdot u_w\end{aligned}$$

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

Word2Vec: Complexity Reduction

- 10만개의 V와 $N=100$ 일 경우에, W, W' 의 계산량은 2000만($2 \times 10\text{만} \times 100$)
- Subsampling frequent words: 자주 등장하는 단어의 학습량을 확률적인 방식으로 줄임.
 - $f(w_i)$ 는 해당 단어가 말뭉치에 등장한 비율 (해당단어빈도/전체단어수)
 - t 는 사용자 지정값. 보통 0.00001
 - $f(w_i)$ 가 0.01이라면 $p(w_i)$ 는 0.9684로 100번의 학습 중 96번 정도는 학습에서 제외
 - Subsampling은 학습량을 효과적으로 줄여 계산량을 감소

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

Word2Vec: Complexity Reduction

- Negative sampling:
 - 출력층의 softmax 함수는 중심단어와 나머지 모든 단어들의 내적을 한 뒤 다시 exp를 취해야함. -> 계산량이 큼
 - Softmax를 구할 때 전체 단어를 대상으로 하지 않고 일부만 뽑아 계산
 - 지정한 윈도우 사이즈 내에 등장하지 않는 단어(negative sample)를 5-20개 정도 뽑은 후 이를 청답단어와 합쳐 이것만으로 softmax를 계산.
 - 윈도우 사이즈가 5일 경우 최대 25개 단어를 대상으로만 softmax 확률을 계산하고, 패러미터 업데이트도 25개만 함
 - 윈도우 내에 등장하지 않는 어떤 단어(w_i)가 negative sample로 뽑힐 확률
- 예) is: $0.9^{(3/4)} = 0.92$ (2.2% 증가)
Constitution: $0.09^{(3/4)} = 0.16$
bombastic: $0.01^{(3/4)} = 0.032$ (220% 증가)

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

Glove(Global Vectors for Word Representation)

- GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations show case interesting linear substructures of the word vector space.
 - While methods like LSA efficiently leverage statistical information, they do relatively poorly on the word analogy task, indicating a sub-optimal vector space structure. Methods like skip-gram may do better on the analogy task, but they poorly utilize the statistics of the corpus since they train on separate local context windows instead of on global co-occurrence counts.
 - <https://nlp.stanford.edu/projects/glove/>

Glove(Global Vectors for Word Representation)

- GloVe is essentially a log-bilinear model with a weighted least-squares objective. The main intuition underlying the model is the simple observation that ratios of **word-word co-occurrence probabilities** have the potential for encoding some form of meaning

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

- 특정 문맥 단어가 주어졌을 때 임베딩된 두 단어벡터의 내적이 두 단어의 동시등장확률간 비율에 되게끔 임베딩
- solid*라는 문맥 단어가 주어졌을 때 *ice*와 *steam* 벡터 사이의 내적값이 8.9가 되도록
- 단어 상호간 비율 정보를 말뭉치 전체를 놓고 한꺼번에 반영하면 좀 더 정확한 임베딩이 될 것이라고 가정

Glove(Global Vectors for Word Representation)

- Objective Function

- $P_{ik} = P(k|i)$: i 번째 단어가 등장했을 때 k 라는 문맥 단어(context word)가 등장할 조건부 확률,
 $P(\text{solid}|\text{ice})$
- $P_{ik}/P_{jk} : P(\text{solid}|\text{ice})/ P(\text{solid}|\text{stream}) = 8.9$

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

- Three conditions

- w_i, w_k 를 서로 바꾸어도 같은 값을 가져야 함
- 코퍼스 전체에서 구한 co-occurrence matrix X 는 대칭행렬(symmetric matrix)이므로 함수는 이런 특징을 포함해야 함
- Homomorphism조건을 만족해야 함

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

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

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

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

$$w_i \longleftrightarrow \tilde{w}_k$$

$$X \longleftrightarrow X^T$$

$$F(X - Y) = \frac{F(X)}{F(Y)}$$

Glove(Global Vectors for Word Representation)

- 이러한 조건을 만족하는 함수는 지수함수, F를 exp 로 치환하고 정리
- w_j, w_k 를 서로 바꾸어도 식이 성립해야 하나 $\log(P_{ik})$ 가 $\log(P_{kj})$ 와 같아야 하나 $\log(X_{ik}) - \log(X_j)$ 와 $\log(X_{kj}) - \log(X_i)$ 로 달라짐.
- 따라서 $\log(X_j)$ 를 상수항(b_j, b_k)로 하여 이 조건을 만족시킴
- $\log(X_{ik})$ 는 co-occurrence matrix에 로그값을 취한것.
- 이 차이를 최소한으로 하는 것이 목적함수

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

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

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

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

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

Glove(Global Vectors for Word Representation) : weighted least squares regression model

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

1. $f(0) = 0$. If f is viewed as a continuous function, it should vanish as $x \rightarrow 0$ fast enough that the $\lim_{x \rightarrow 0} f(x) \log^2 x$ is finite.
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.

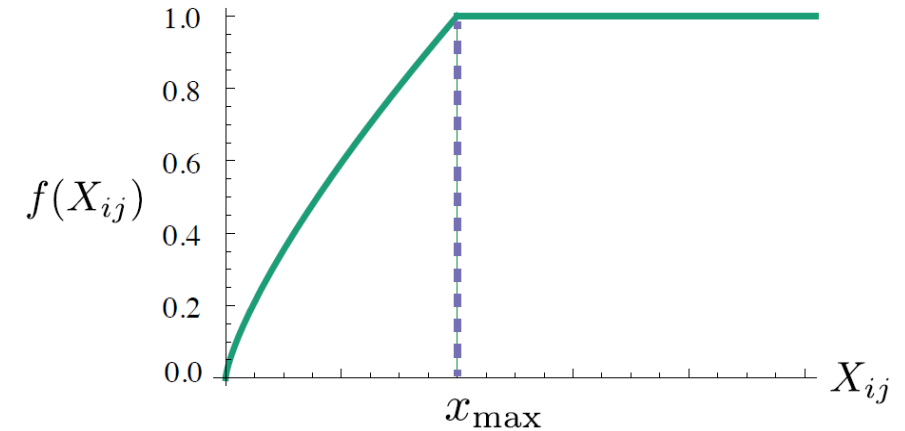


Figure 1: Weighting function f with $\alpha = 3/4$.

FastText

- Facebook AI Research lab에 의한 open source
 - <https://research.fb.com/fasttext/>
 - FastText combines some of the most successful concepts introduced by the natural language processing and machine learning communities in the last few decades. These include representing sentences with **bag of words** and **bag of n-grams**, as well as using **subword information**, and **sharing information across classes** through a hidden representation.
 - It also employs a hierarchical softmax that takes advantage of the unbalanced distribution of the classes to speed up computation. These different concepts are being used for two different tasks: efficient text classification and learning word vector representations.

General Model

- Given a word vocabulary size W , where a word is identified by its index $w \in \{1, \dots, W\}$, the goal is to learn a vectorial representation for each word w .
- Skip-gram model is to maximize the following log-likelihood:

$$\sum_{t=1}^T \sum_{c \in \mathcal{C}_t} \log p(w_c \mid w_t),$$

- where the context \mathcal{C}_t is the set of indices of words surrounding word w_t
- one possible choice to define the probability of a context word is the softmax:

- d

$$p(w_c \mid w_t) = \frac{e^{s(w_t, w_c)}}{\sum_{j=1}^W e^{s(w_t, j)}}.$$

General Model

- Negative Sampling

- The problem of predicting context words can be framed as a set of independent binary classification tasks
- For the word at position t we consider all context words as positive examples and sample negatives at random from the dictionary
- For a chosen context position c , using the binary logistic loss, we obtain the following negative log-likelihood:

$$\log \left(1 + e^{-s(w_t, w_c)} \right) + \sum_{n \in \mathcal{N}_{t,c}} \log \left(1 + e^{s(w_t, n)} \right)$$

- Where $\mathcal{N}_{t,c}$ is a set of negative examples sampled from the vocabulary
- By denoting the logistic loss function $\ell : x \mapsto \log(1 + e^{-x})$, we can rewrite the objective as:

$$\sum_{t=1}^T \left[\sum_{c \in \mathcal{C}_t} \ell(s(w_t, w_c)) + \sum_{n \in \mathcal{N}_{t,c}} \ell(-s(w_t, n)) \right]$$

Subword Model

- By using a distinct vector representation for each word, the skipgram model ignores the internal structure of words.
 - Each word w is represented as a bag of character n -gram. FastText adds special symbols $<$ and $>$ at the beginning and end of words, allowing to distinguish prefixes and suffixes from other character sequences
 - Also includes w itself in the set of its n -grams, to learn a representation for each word
 - Taking the word where and $n=3$ as an example, it will be represented by the character n -grams:
 - $<wh, whe, her, ere, re>$ and the special sequence
 - $<where>$
 - Note that the sequence $<her>$, corresponding to her is different from the tri-gram her from the word where.
 - In practice, they extract all the n -grams for n greater or equal 2 and smaller or equal to 6

Subword Model

- Suppose that you are given a dictionary of *n-grams* of size G . Given a word w , let us denote by $\mathcal{G}_w \subset \{1, \dots, G\}$, the set of *n-grams* appearing in w .
 - Then associate a vector representation \mathbf{z}_g to each n-gram g
 - The scoring function is

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

- This simple model allows sharing the representations across words, thus allowing to learn reliable representation for rare words