Word2Vec

[Paper]
Efficient Estimation of Word Representations in Vector Space
Distributed Representations of Words and Phrases and their Compositionality

OWOP season1 01
송민수

Paper

- Efficient Estimation of Word Representations in Vector Space
  - word2vec
- Distributed Representations of Words and Phrases and their Compositionality
  - word2vec + subsampling, negative sampling

- 두 논문 모두 2013년생입니다.
- 둘 다 word2vec에 관련한 내용이라 합쳐서 준비했습니다.
- 이번에는 논문 자체에 대한 집중보다 논문내용을 담아서 Word2Vec에 대해 전달하려고 합니다.

What Word2Vec does?

- 목적
  - 단어를 숫자로 나타내보자
    → 그렇다면 one-hot-encoding?
    → 하지만 high dimension, similarity 계산 불가(inner product하면 항상 0)
    → 적당한 차원을 가진 Semantically, Syntactically 유의미한 vector space를 만들자
    → 예를 들어, '왕 vector' - '남자 vector' + '여자 vector' = '여왕 vector'

- Distributional hypothesis
  - 단어의 의미는 주변 단어에 의해 형성된다

[예시]

나는 달콤한 배를 먹는다

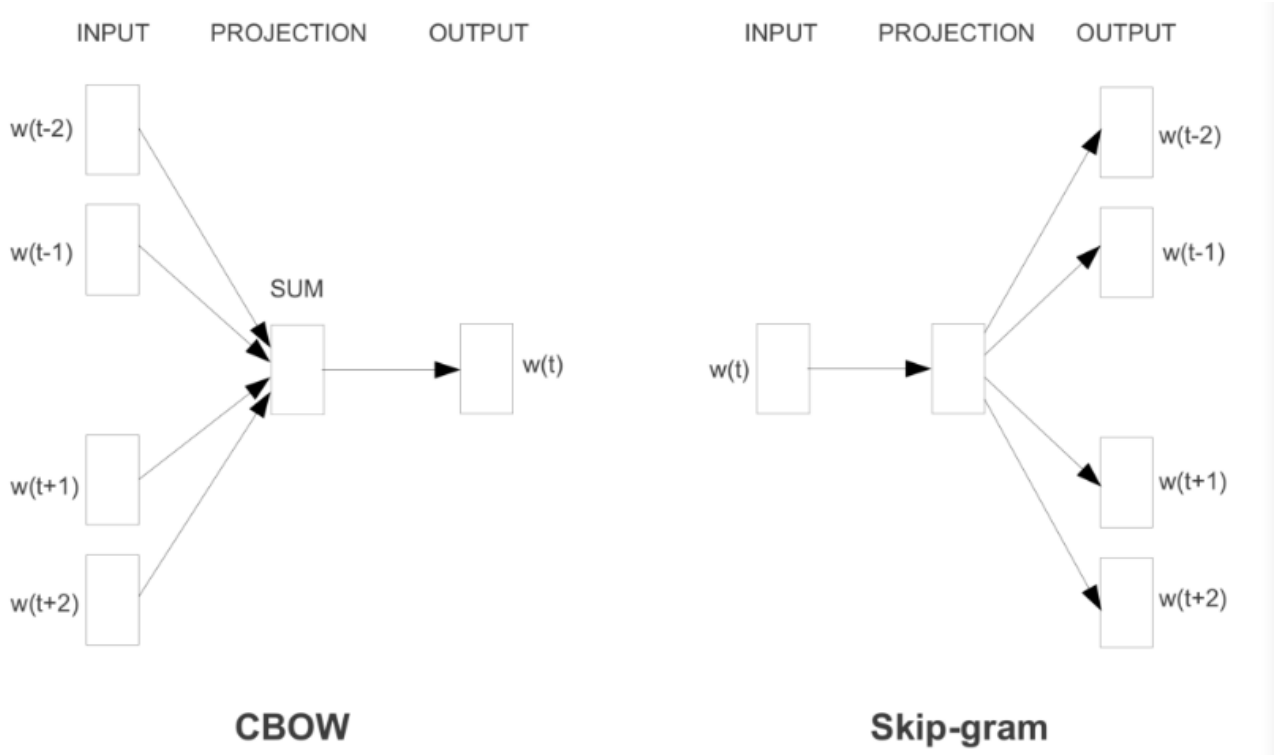
→ 우리는 '달콤한' 과 '먹는다' 라는 단어를 통해 배의 의미를 이해할 수 있다

Before Word2Vec

- Bag of words
  - 단어의 순서 상관없이 오직 빈도만 이용한 방법
- TF-IDF
  - Term-Document 형태로 문서에서 단어빈도를 weight로 이용한 방법
- 시소러스
  - 사전 만들기, Wordnet
- 통계기반
  - 동시발생행렬, 차원축소 →
    - Word2Vec이 만들어지고 시간이 지나 Word2vec이 통계기반 동시발생행렬의 방법론과 거의 유사하다는 것이 밝혀졌다
    - 성능도 의미론적으로는 비슷한 경우가 많다고 한다
- NNLM
  - Word2vec 이전 NN을 이용한 방법

Word2Vec

- 두 가지 방법이 존재
  - CBOW, Skip-gram
  - Skip-gram이 조금 더 나은 성능을 보인다



Word2Vec

- 두 가지 방법이 존재
  - CBOW, Skip-gram
  - Skip-gram이 조금 더 나은 성능을 보인다

- Skip-gram 예시
  - Window size = 1
  - 파란색 글씨로 빨간색 글씨가 무엇인지 맞춘다
  - 즉, Word2Vec은 사실 classification하는 것!

나는 ?? 배를 먹는다

?? 달콤한 ?? 먹는다

나는 ?? 배를 ??

나는 달콤한 ?? 먹는다

Skip-gram

- NN에 input으로 넣기 위해 단어들을 one-hot-encoding 한다
- Train data는 아래의 모습이다
  - 일단 4개의 단어밖에 없다고 생각

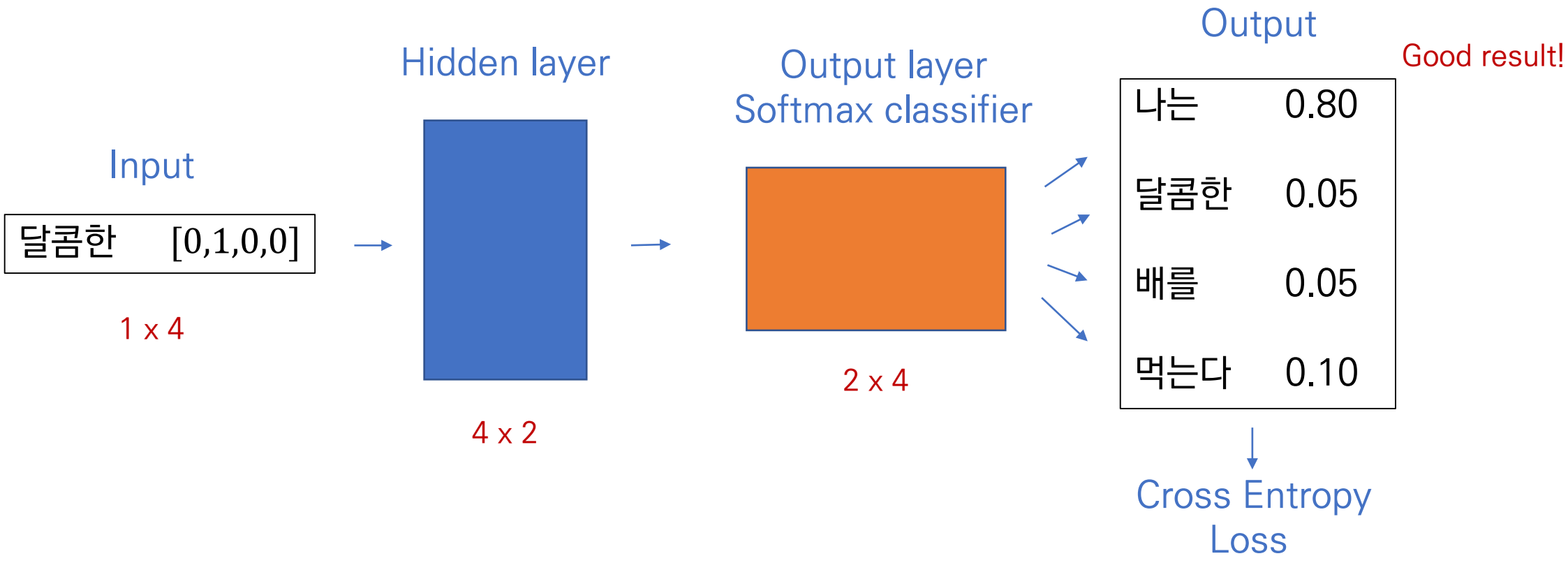
나는	$[1,0,0,0]^T$
달콤한	$[0,1,0,0]^T$
배를	$[0,0,1,0]^T$
먹는다	$[0,0,0,1]^T$

Train data (x,y)

(나는, 달콤한)
(달콤한, 나는), (달콤한, 배를)
(배를, 달콤한), (배를, 먹는다)
(먹는다, 배를)

Skip-gram

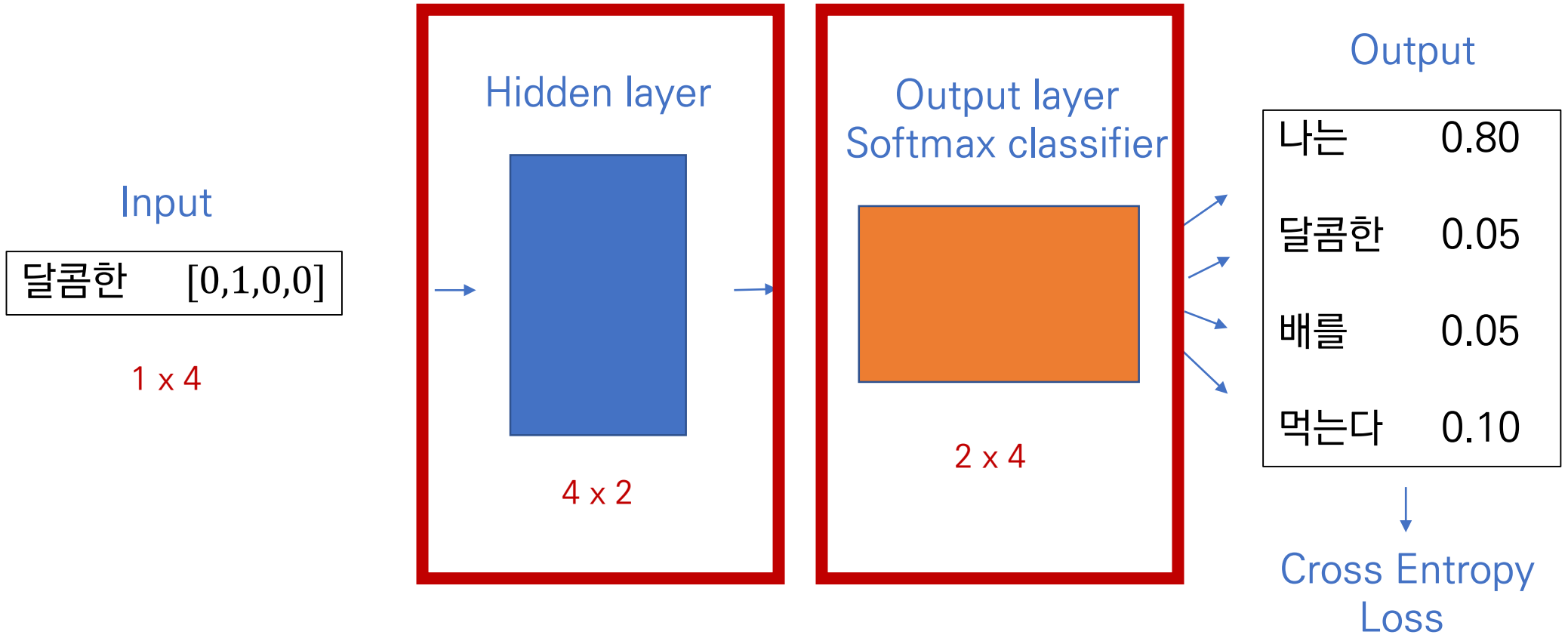
- Classification을 잘 하도록 parameters를 업데이트한다
- 각 layer들은 fully connected layer이다
- Activation function은 없다



Skip-gram

- 그렇다면 도대체 단어를 나타내는 Vector는 무엇?

아래 두가지 Layer중 하나!를 사용하면 된다



Skip-gram

- 학습된 parameters 자체가 해당 단어를 나타내는 vector가 된다

Hidden layer

4 x 2

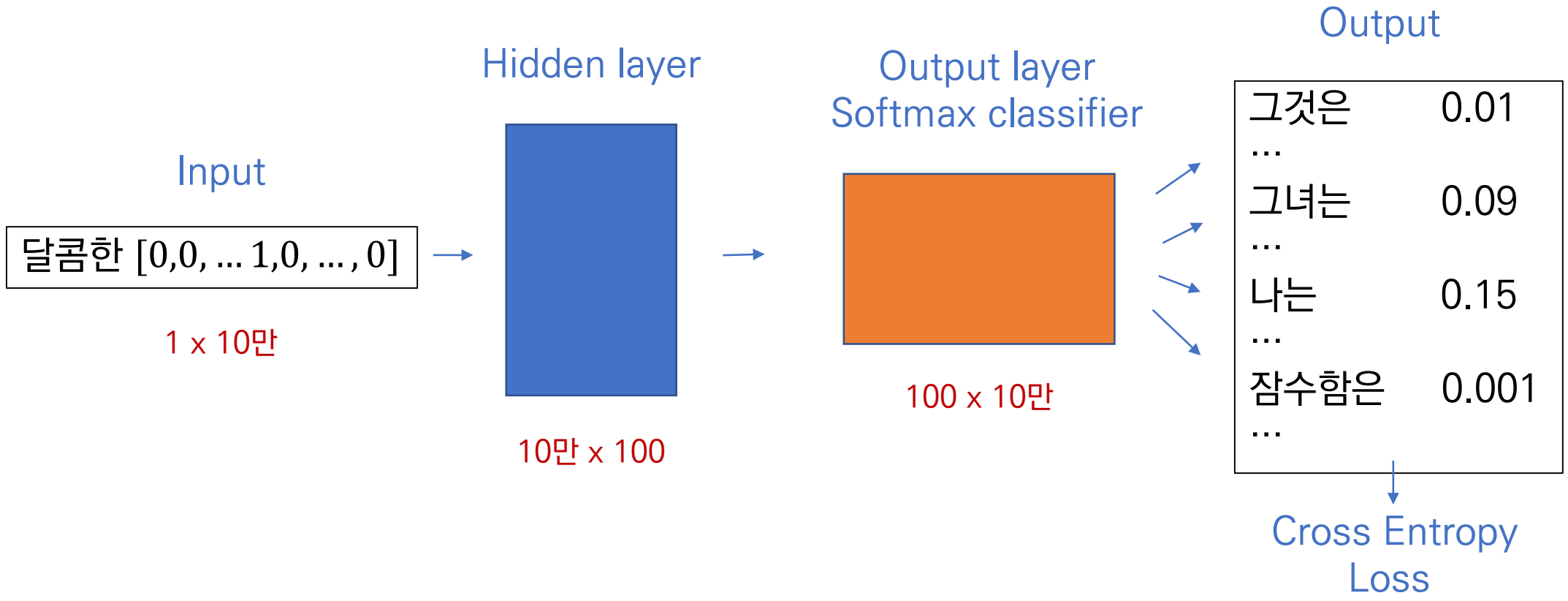
나는 :	0.42	0.5
달콤한 :	0.21	0.2
배를 :	0.92	0.88
먹는다 :	0.12	0.24

각 단어를 2차원의 vector로 표현할 수 있는 것이다

여기서 Embedding dimension은 연구자 맘대로 정할 수 있으며 단어를 나타내는 vector의 차원이 된다

Skip-gram

- 이제는 단어의 수가 10만개 라고 생각해보자



Skip-gram

- 굳이 matrix multiplication을 하지 않아도 되지 않을까?
  - Indexing을 하자!
  - 실제로 nlp task에서 indexing을 통해 진행한다

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

For better Word2Vec

- Word Pairs and Phrase
  - 예를 들어, ‘연세대학교’ 같은 단어의 경우 ‘연세’ 와 ‘대학교’ 로 나누지 않고 ‘연세대학교’로 학습
- Subsampling
  - 자주 등장하는 단어의 경우는 학습에서 확률적으로 제외
  - 예를 들어, ‘은/는’ 과 같은 경우
- Negative Sampling
  - Softmax 계산을 줄이기 위해 사용하는 방법
  - Target이 0인 단어를 모두 학습하지 않고 일부만 뽑아서 backpropagation 진행

Subsampling

- 말뭉치에서 자주 등장하는 단어는 학습량을 확률적인 방법으로 줄이기로 하였다
- 예를 들어, 등장빈도가 많은 '그' , '우리는' 등의 단어는 parameter update 기회가 많다

- 아래의 확률로 해당단어를 training에서 제외한다
- t는 논문에서 0.00001이고 $f(w_i)$는 해당단어의 출현확률 (해당단어 등장 수/전체단어 수)

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

Negative Sampling

- 마지막에 softmax를 계산할 때 너무 부담이 크다

- 이를 해결하기 위해 window size에 등장하지 않는 단어(negative sample)을 5개~20개를 뽑는다
- 그리고 parameter update도 해당 뽑은 단어들만 진행한다
- Negative sample로 뽑힐 확률은 아래처럼 정의된다

$$P(w_i) = \frac{f(w_i)^{\frac{3}{4}}}{\sum f(w_j)^{\frac{3}{4}}}$$

Additional Paper

- GloVe: Global Vectors for Word Representation
- FastText： Enriching Word Vectors with Subword Information
- Improving Distributional Similarity with Lessons Learned from Word Embeddings
- Evaluation methods for unsupervised word embeddings
- A Latent Variable Model Approach to PMI-based Word Embeddings
- Linear Algebraic Structure of Word Senses, with Applications to Polysemy
- On the Dimensionality of Word Embedding

Reference

- Efficient Estimation of Word Representations in Vector Space
- Distributed Representations of Words and Phrases and their Compositionality

- http://jalammar.github.io/illustrated-word2vec/
- https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/03/30/word2vec/
- 밑바닥부터 시작하는 딥러닝 2