텐서플로와 머신러닝으로 시작하는 자연어처리

Lecture 2
2019. 06. 08

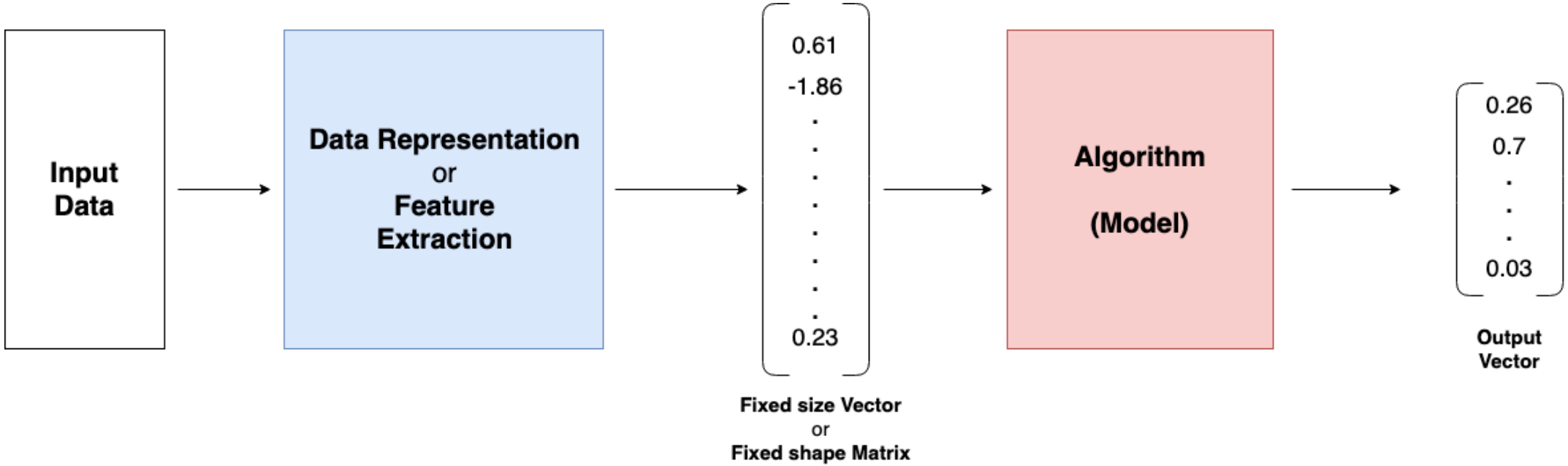
Word Representation

Word Representation

- Data format
  - Tabular data
  - Image data
  - Text data
- One - hot
- Bag of Words
- Neural Text Representation
  - Word2Vec
  - GloVe
  - FastText

Data Representation

Data Representation



Data Representation

	Age	Salary	Type1	Type2
0	22	3200	2	7
1	53	6500	0	2
2	32	4300	1	2
3	42	3400	3	3
4	37	5600	5	6

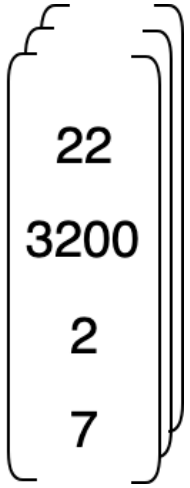
Tabular Data

Data Representation

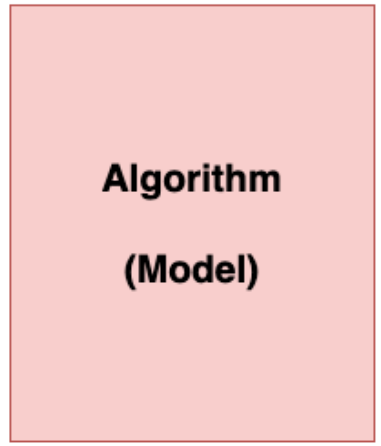
	Age	Salary	Type1	Type2
0	22	3200	2	7
1	53	6500	0	2
2	32	4300	1	2
3	42	3400	3	3
4	37	5600	5	6

Tabular Data

Data Representation



Fixed size Vector



Data Representation

Image Data

Data Representation

0	0	0	0	0	0	0
0	**1**	**1**	**1**	**1**	**1**	0
0	**1**	0	0	0	0	0
0	0	**1**	**1**	0	0	0
0	0	0	0	**1**	**1**	0
0	0	0	0	**1**	0	0
0	**1**	**1**	**1**	0	0	0

7 x 7
Matrix

Algorithm

(Model)

Data Representation

나는 학교에 간다.

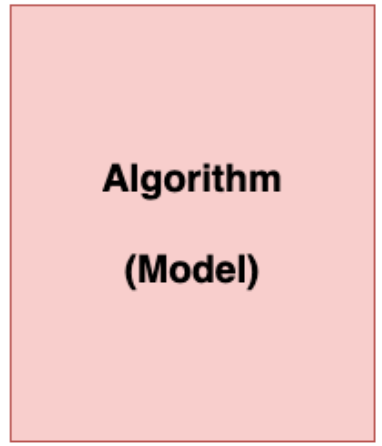
Text Data

Data Representation

나는 학교에 간다.

Text Data

Data Representation



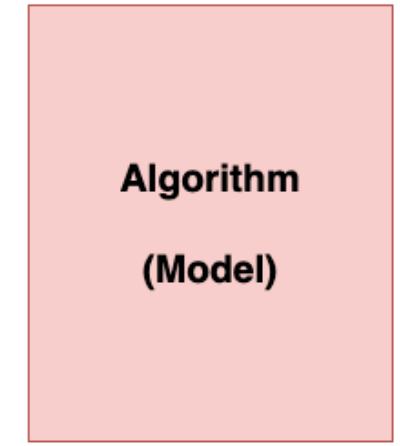
- Text Representation
- Word Representation

Data Representation

나는 학교에 간다.

Text Data

Data Representation



- **Text Representation**
  - Word Representation
  - Character Representation
  - Sentence Representation
  - Document Representation

Text Representation

1. One-hot Encoding (1-of-N encoding)
2. Bag of Words
3. TF-IDF
4. Neural Text Representation
    - Word2Vec
    - GLoVe
    - FastText
    - ELMo

One-hot Encoding

One-hot Encoding (1-of-N encoding)

One-hot

One-hot Encoding (1-of-N encoding)

One-hot

하나만 hot, 나머지는 cool

하나만 1, 나머지는 0

One-hot Encoding (1-of-N encoding)

One-hot

하나만 hot, 나머지는 cool

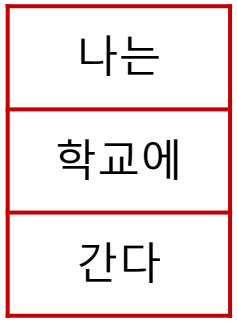
하나만 1, 나머지는 0

나는 학교에 간다.

One-hot Encoding (1-of-N encoding)

나는 학교에 간다.

One-hot Encoding (1-of-N encoding)



Vocabulary
(단어 사전)

나는 학교에 간다.

One-hot Encoding (1-of-N encoding)

나는
학교에
간다
매일
늦는다

Vocabulary
(단어 사전)

1
0
0

나는 학교에 간다.

One-hot Encoding (1-of-N encoding)

나는
학교에
간다
매일
늦는다

Vocabulary
(단어 사전)

0
1
0

나는 **학교에** 간다.

One-hot Encoding (1-of-N encoding)

나는
학교에
간다
매일
늦는다

Vocabulary
(단어 사전)

0
0
1

나는 학교에 **간다.**

One-hot Encoding (1-of-N encoding)

나는
학교에
간다
매일
늦는다

Vocabulary
(단어 사전)

1	0	0
0	1	0
0	0	1

나는 학교에 간다.

One-hot Encoding (1-of-N encoding)

나는
학교에
간다
매일
늦는다

Vocabulary
(단어 사전)

나는 학교에 간다.
나는 매일 아침을 먹는다.

One-hot Encoding (1-of-N encoding)

나는
학교에
간다
매일
늦는다

Vocabulary
(단어 사전)

1
0
0
0
0

나는 학교에 간다.
나는 매일 학교에 늦는다.

One-hot Encoding (1-of-N encoding)

나는
학교에
간다
매일
늦는다

Vocabulary
(단어 사전)

0
1
0
0
0

나는 **학교에** 간다.
나는 매일 학교에 늦는다.

One-hot Encoding (1-of-N encoding)

나는
학교에
간다
매일
늦는다

Vocabulary
(단어 사전)

0
0
1
0
0

나는 학교에 **간다.**
나는 매일 학교에 늦는다.

One-hot Encoding (1-of-N encoding)

나는
학교에
간다
매일
늦는다

Vocabulary
(단어 사전)

1
0
0
0
0

나는 학교에 간다.
나는 매일 학교에 늦는다.

One-hot Encoding (1-of-N encoding)

나는
학교에
간다
매일
늦는다

Vocabulary
(단어 사전)

0
0
0
1
0

나는 학교에 간다.
나는 **매일** 학교에 늦는다.

One-hot Encoding (1-of-N encoding)

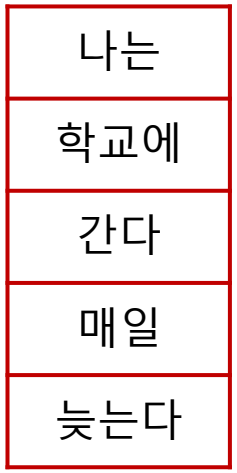
나는
학교에
간다
매일
늦는다

Vocabulary
(단어 사전)

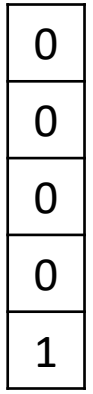
0
1
0
0
0

나는 학교에 간다.
나는 매일 **학교에** 늦는다.

One-hot Encoding (1-of-N encoding)



Vocabulary
(단어 사전)



나는 학교에 간다.
나는 매일 학교에 **늦는다.**

One-hot Encoding (1-of-N encoding)

나는
학교에
간다
매일
늦는다

Vocabulary
(단어 사전)

1	0	0
0	1	0
0	0	1
0	0	0
0	0	0

나는 학교에 간다.

나는 매일 학교에 늦는다.

One-hot Encoding (1-of-N encoding)

나는
학교에
간다
매일
늦는다

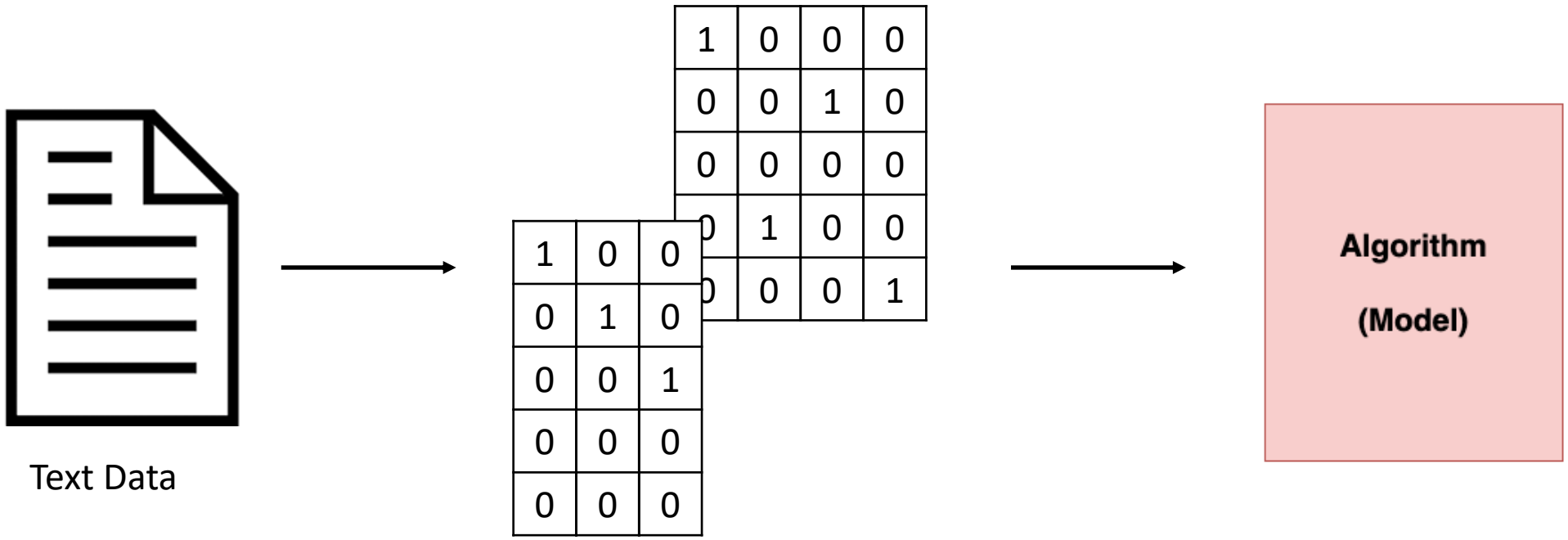
Vocabulary
(단어 사전)

1	0	0	0
0	0	1	0
0	0	0	0
0	1	0	0
0	0	0	1

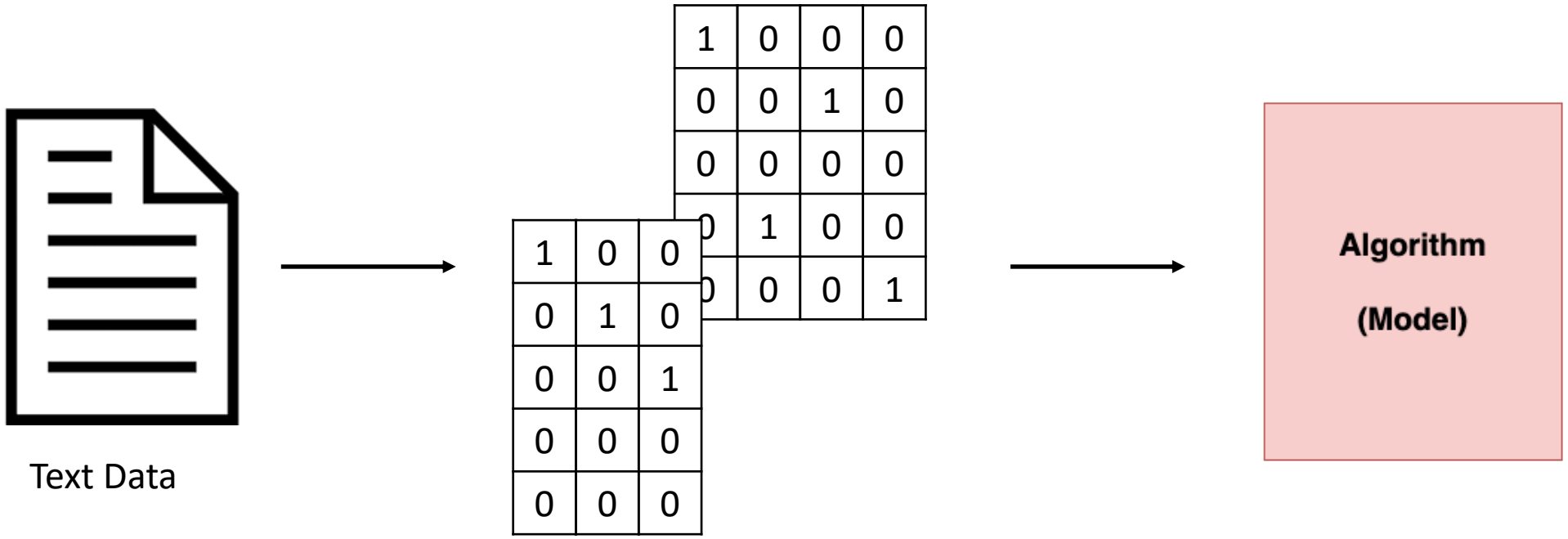
나는 학교에 간다.

나는 매일 학교에 늦는다.

One-hot Encoding (1-of-N encoding)



One-hot Encoding (1-of-N encoding)



Input: Sentence

One-hot Encoding (1-of-N encoding)

Text Data

1	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	0	0	0	0
0	0	0	0	1	0	0
0	0	0	0	0	0	1

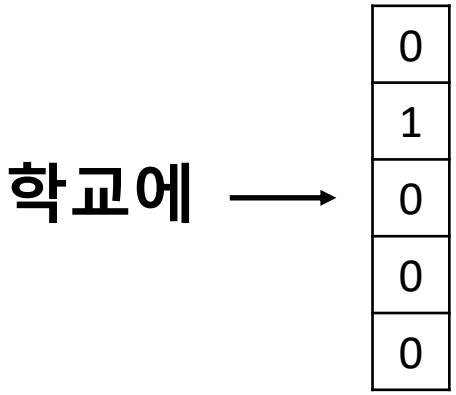
Algorithm

(Model)

Input: Document

One-hot Encoding (1-of-N encoding)

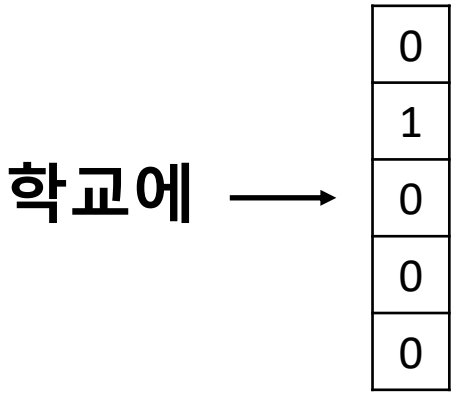
In One-Hot Encoding



- 벡터 하나당 하나의 단어(Word) or 문자(Character)
- 매우 큰 벡터의 크기에 비해 벡터가 Sparse함.

One-hot Encoding (1-of-N encoding)

In One-Hot Encoding



- 벡터 하나당 하나의 단어(Word) or 문자(Character)
- 매우 큰 벡터의 크기에 비해 벡터가 Sparse함.

Bag-of-Words

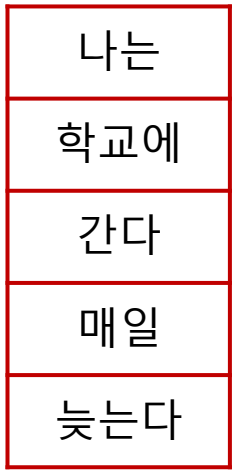
- 벡터 하나당 하나의 문장(Sentence) or 문서(Document)
- 벡터의 크기는 같지만 상대적으로 덜 Sparse 함

Bag-of-Words

Bag-of-Words (BoW)

나는 학교에 간다.
나는 매일 아침을 먹는다.

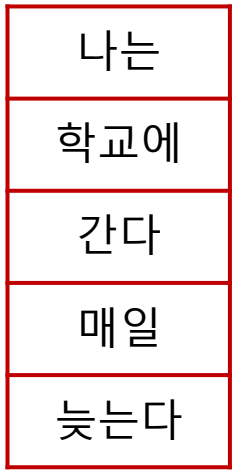
Bag-of-Words (BoW)



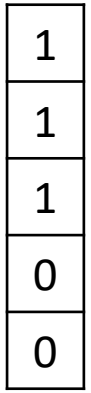
Vocabulary
(단어 사전)

나는 학교에 간다.
나는 매일 아침을 먹는다.

Bag-of-Words (BoW)



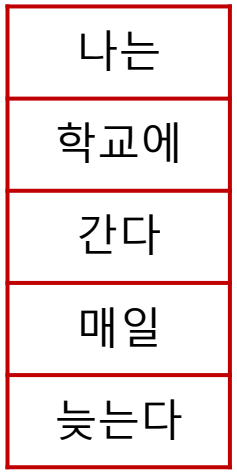
Vocabulary
(단어 사전)



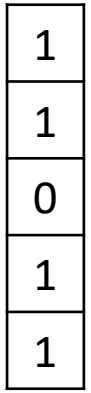
나는 학교에 간다.

나는 매일 학교에 늦는다.

Bag-of-Words (BoW)



Vocabulary
(단어 사전)



나는 학교에 간다.

나는 매일 학교에 늦는다.

Bag-of-Words (BoW)

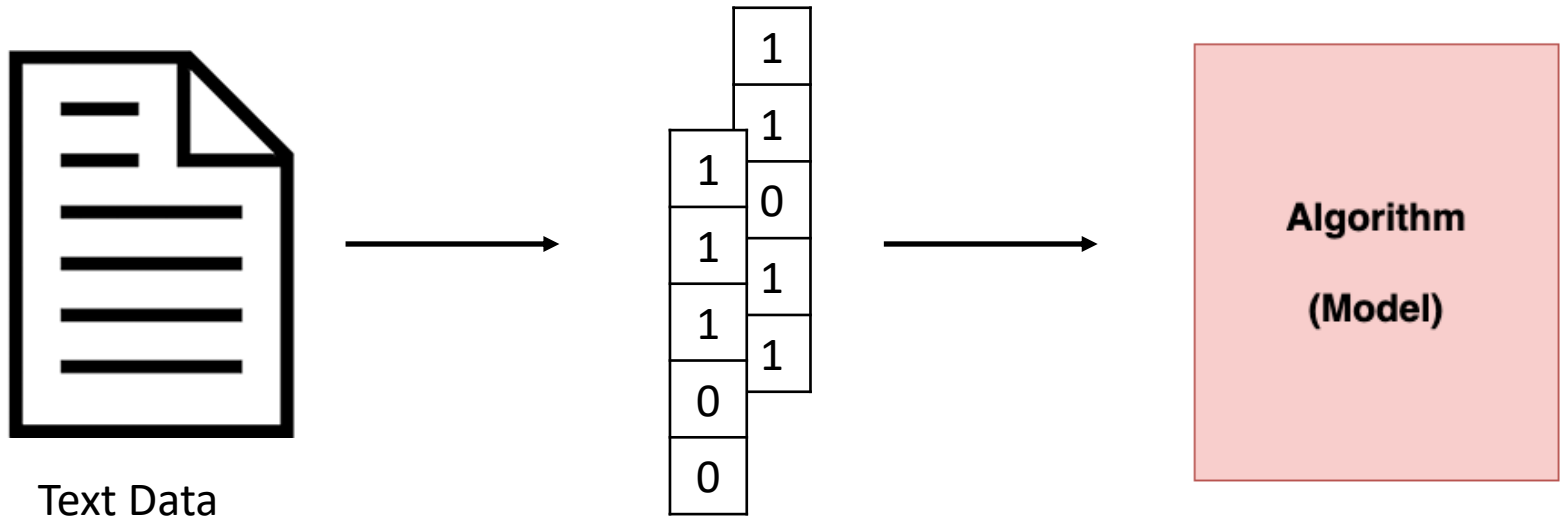
나는
학교에
간다
매일
늦는다

Vocabulary
(단어 사전)

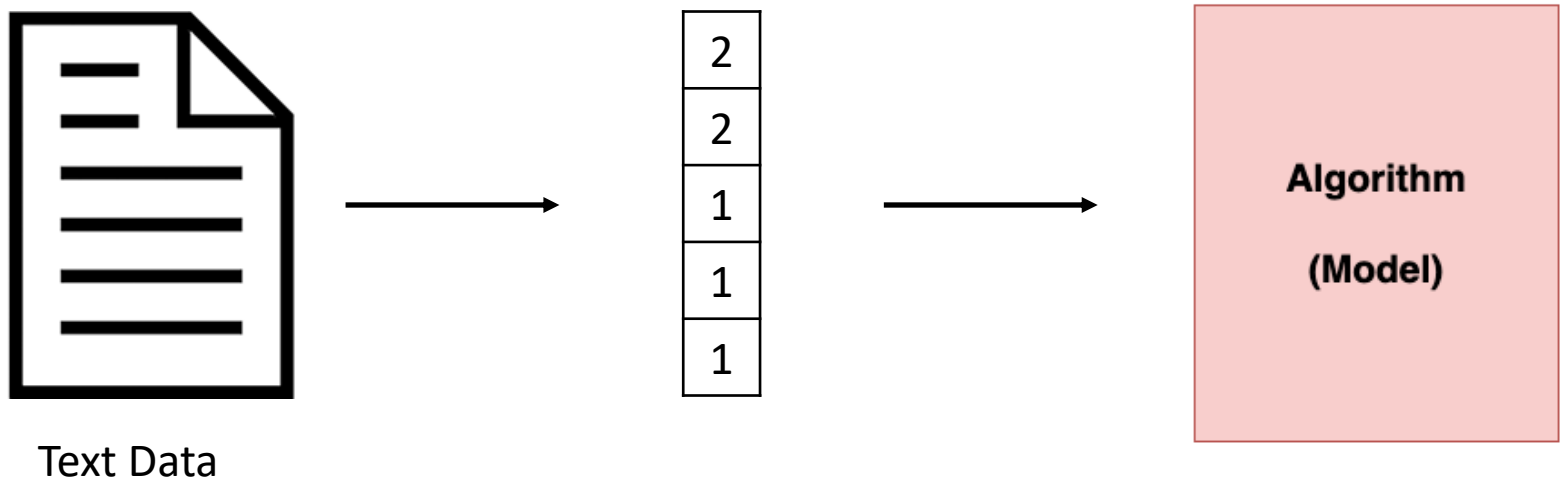
2
2
1
1
1

나는 학교에 간다. 나는 매일 학교에 늦는다.

Bag-of-Words (BoW)



Bag-of-Words (BoW)



1-Hot & BoW

Advantage

- 간단하고 매우 직관적이다.
- 쉽게 구현이 가능하다.

Disadvantage

- 벡터의 크기가 단어 사전의 크기에 비례해서 벡터가 매우 크고 Sparse하다.
- 벡터가 단어 혹은 문장의 의미를 담고 있지 않다.

1-Hot & BoW

Advantage

- 간단하고 매우 직관적이다.
- 쉽게 구현이 가능하다.

Disadvantage

- 벡터의 크기가 단어 사전의 크기에 비례해서 벡터가 매우 크고 Sparse하다.
- 벡터가 단어 혹은 문장의 의미를 담고 있지 않다.

Neural Text Representation

Neural Text Representation

Neural Text Representation

- Word2Vec
- GLoVe
- FastText

Neural Text Representation

- Word2Vec
- GLoVe
- FastText

단어의 의미를 capture한다.

Neural Text Representation

- Word2Vec
- GLoVe
- FastText

단어의 의미를 capture한다.

How?

Neural Text Representation

1. 운동을 마치고 난 후 _____에서 몸을 씻었다.
2. 나는 _____ 청소를 거의 매일 한다.
3. 철수는 _____에 들어가기만 하면 한 시간이다.
4. 찜질방이 한창일 때 _____ 손님이 많이 줄었다고 한다.
5. 나는 지난 휴일에 _____에 가서 때를 밀었다.

Neural Text Representation

1. 운동을 마치고 난 후 목욕탕에서 몸을 씻었다.
2. 나는 목욕탕 청소를 거의 매일 한다.
3. 철수는 목욕탕에 들어가기만 하면 한 시간이다.
4. 찜질방이 한창일 때 목욕탕 손님이 많이 줄었다고 한다.
5. 나는 지난 휴일에 목욕탕에 가서 때를 밀었다.

Neural Text Representation

1. 운동을 마치고 난 후 목욕탕에서 몸을 씻었다.
2. 나는 목욕탕 청소를 거의 매일 한다.
3. 철수는 목욕탕에 들어가기만 하면 한 시간이다.
4. 찜질방이 한창일 때 목욕탕 손님이 많이 줄었다고 한다.
5. 나는 지난 휴일에 목욕탕에 가서 때를 밀었다.

주변 단어를 보면 해당 단어의 뜻을 알 수 있다.

Neural Text Representation

1. 운동을 마치고 난 후 목욕탕에서 몸을 씻었다.
2. 나는 목욕탕 청소를 거의 매일 한다.
3. 철수는 목욕탕에 들어가기만 하면 한 시간이다.
4. 찜질방이 한창일 때 목욕탕 손님이 많이 줄었다고 한다.
5. 나는 지난 휴일에 목욕탕에 가서 때를 밀었다.

주변 단어를 보면 해당 단어의 뜻을 알 수 있다.

Distributional Hypothesis
(분산 가정)

Neural Text Representation

1. 운동을 마치고 난 후 목욕탕에서 몸을 씻었다.
2. 나는 목욕탕 청소를 거의 매일 한다.
3. 철수는 목욕탕에 들어가기만 하면 한 시간이다.
4. 찜질방이 한창일 때 목욕탕 손님이 많이 줄었다고 한다.
5. 나는 지난 휴일에 목욕탕에 가서 때를 밀었다.

주변 단어를 보면 해당 단어의 뜻을 알 수 있다.

Distributional Hypothesis
(분산 가정)

(Key idea of Neural Text Representation)

Word2Vec

- Mikorov et al. (2013)
- Neural Network를 사용한 Word Representation 방법 중 대표적인 방법.
- 기존의 NNLM, RNNLM 방식의 모델에 비해 속도가 대폭 개선됨
- CBOW, Skip-Gram 두 개의 모델 포함

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

1 Introduction

Many current NLP systems and techniques treat words as atomic units - there is no notion of similarity between words, as these are represented as indices in a vocabulary. This choice has several good reasons - simplicity, robustness and the observation that simple models trained on huge amounts of data outperform complex systems trained on less data. An example is the popular N-gram model used for statistical language modeling - today, it is possible to train N-grams on virtually all available data (trillions of words [3]).

However, the simple techniques are at their limits in many tasks. For example, the amount of relevant in-domain data for automatic speech recognition is limited - the performance is usually dominated by the size of high quality transcribed speech data (often just millions of words). In machine translation, the existing corpora for many languages contain only a few billions of words or less. Thus, there are situations where simple scaling up of the basic techniques will not result in any significant progress, and we have to focus on more advanced techniques.

With progress of machine learning techniques in recent years, it has become possible to train more complex models on much larger data set, and they typically outperform the simple models. Probably the most successful concept is to use distributed representations of words [10]. For example, neural network based language models significantly outperform N-gram models [1, 27, 17].

Word2Vec

핵심 Idea

CBOW

매일 늦게 일어나서 _____ 안먹고 출근한다.

Skip-Gram

____ ____ _____ 아침을 _____ _____

Word2Vec

핵심 Idea

CBOW

매일 늦게 일어나서 _____ 안먹고 출근한다.

주변의 여러 단어들을 통해 하나의 단어를 예측

Skip-Gram

____ ____ _____ 아침을 _____ _____

하나의 단어를 통해 주변 여러 단어들을 예측

Word2Vec

Objective function

CBOW

$$\sum_{t=1}^{T} P(w_t | w_{t-m}, \dots, w_{t-1}, w_{t-2}, \dots, w_{t+m})$$

Skip-Gram

$$\sum_{t=1}^{T} \sum_{-m \le j \le m, j \neq 0} P(w_{t-j} | w_t)$$

- T: 전체 단어 개수
- m: window size
- w: 단어

Word2Vec (CBOW)

Inputs

매일 늦게 일어나서 _____ 안먹고 출근한다.

Word2Vec (CBOW)

Inputs

매일 늦게 일어나서 _____ 안먹고 출근한다.

Output

아침을

Word2Vec (CBOW)

Inputs

매일 늦게 일어나서 _____ 안먹고 출근한다.

Output

아침을

Label

아침을

Word2Vec (CBOW)

Inputs

매일 늦게 일어나서 _____ 안먹고 출근한다.

Output

False 회사를

Label

아침을

Word2Vec (CBOW)

Inputs

매일 늦게 일어나서 _____ 안먹고 출근한다.

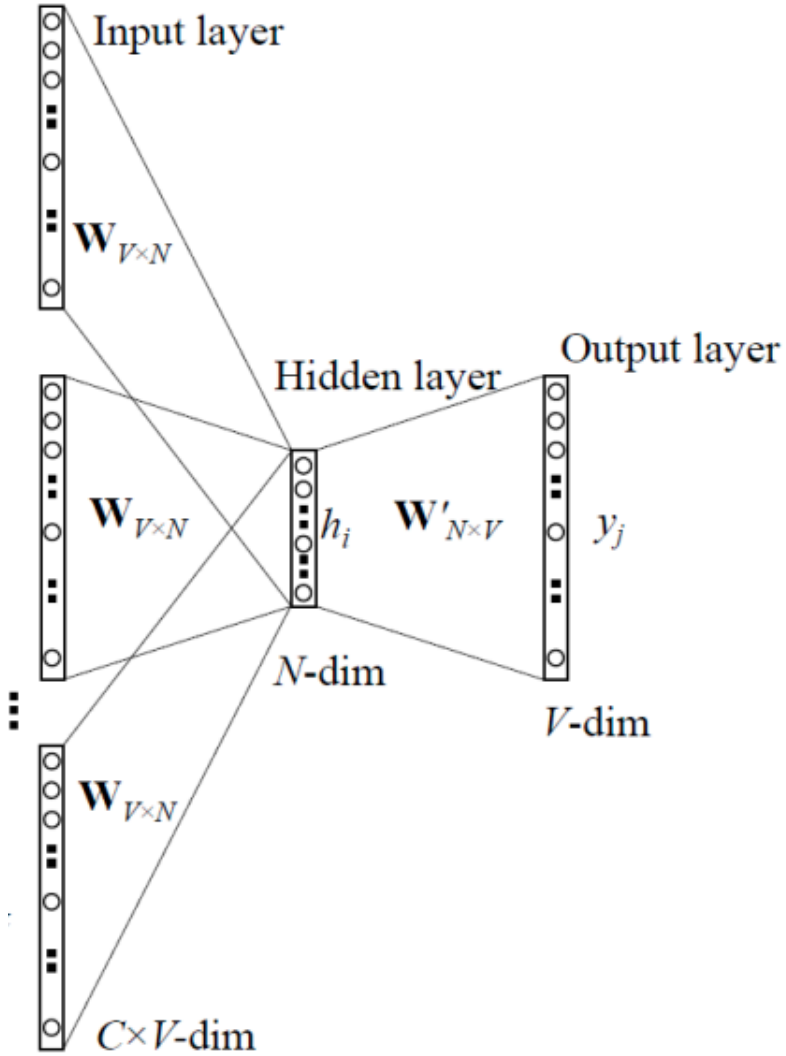
Output

False 회사를

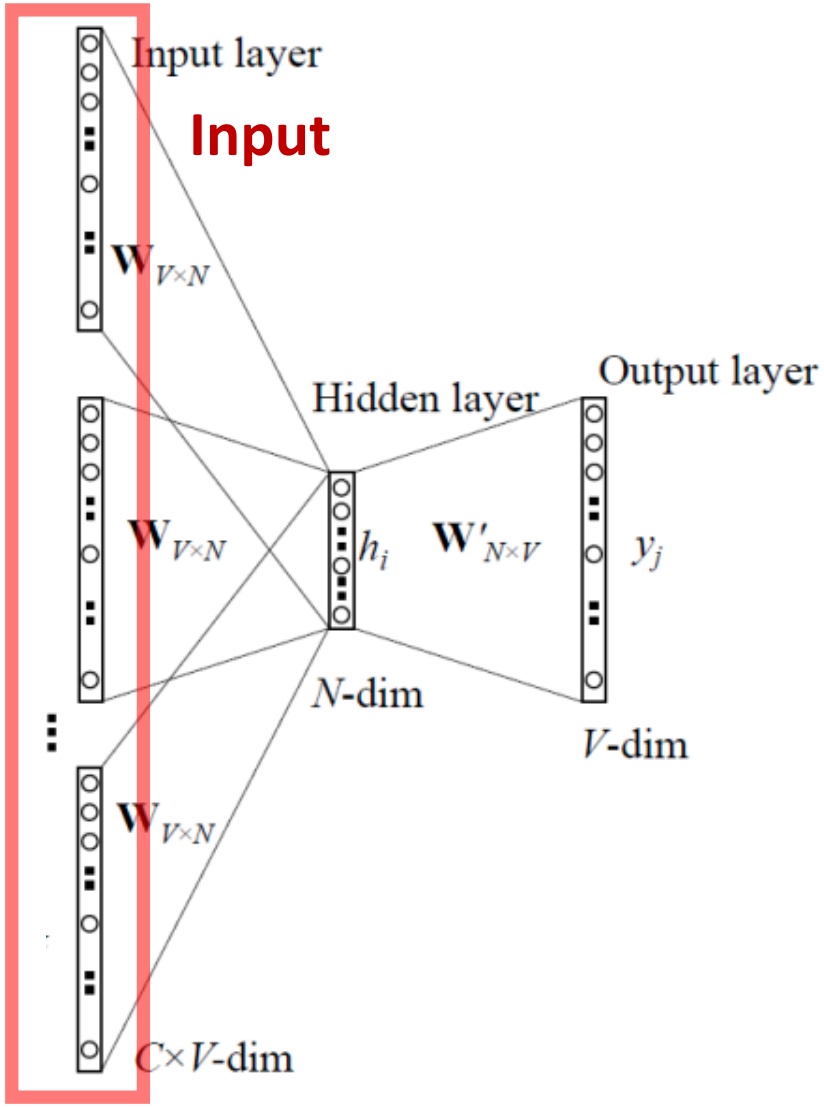
Label

아침을

Word2Vec (CBOW)



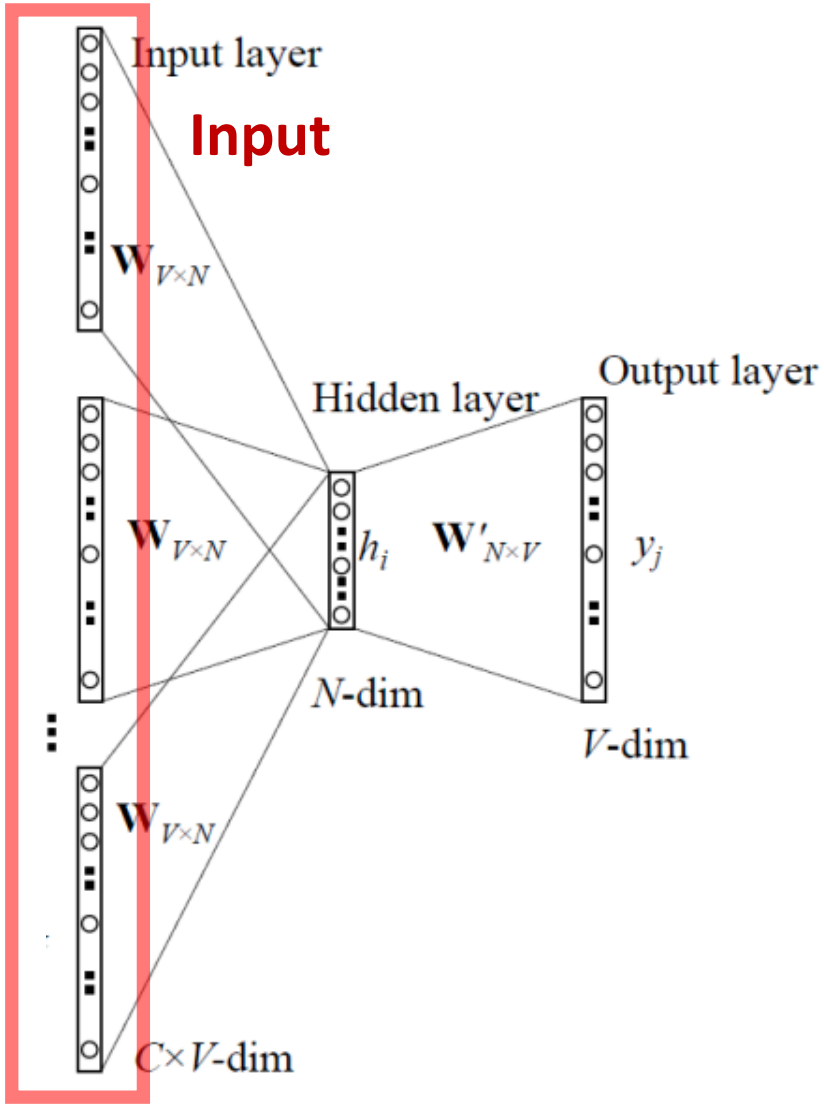
Word2Vec (CBOW)



매일 늦게 일어나서 _____ 안먹고 출근한다.

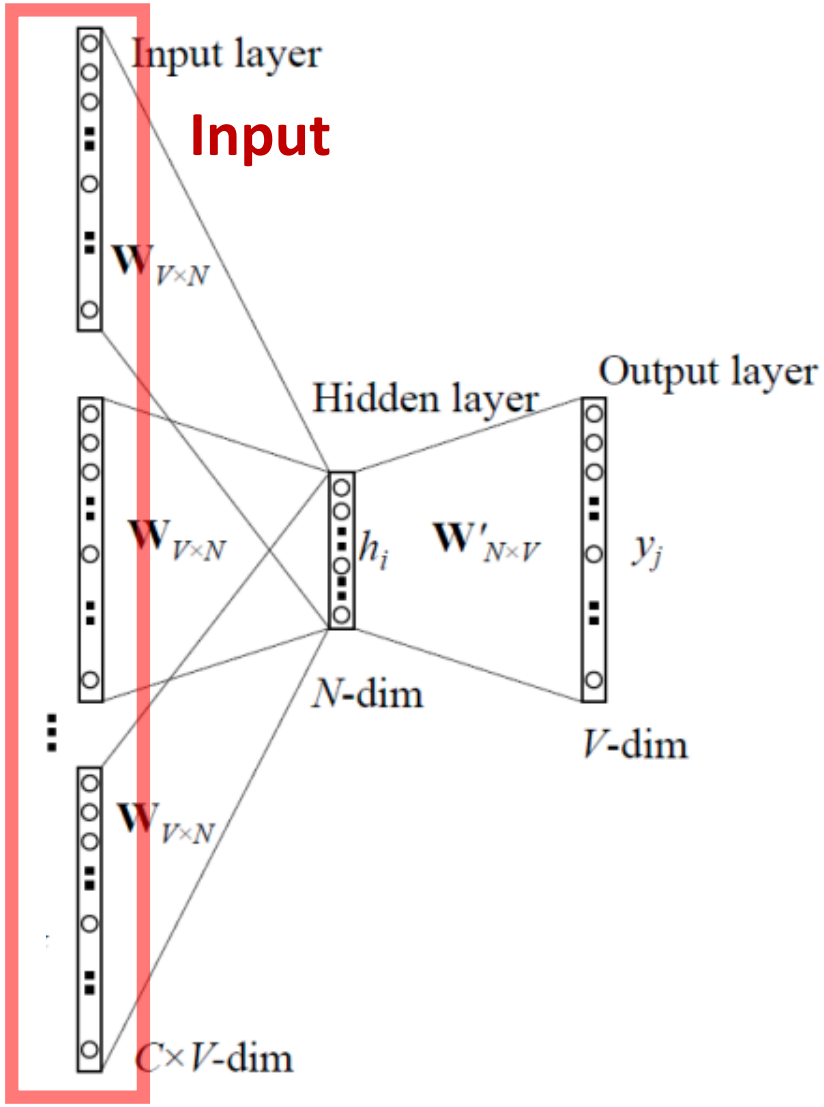
Word2Vec (CBOW)

Window size = 2



매일 늦게 일어나서 _____ 안먹고 출근한다.

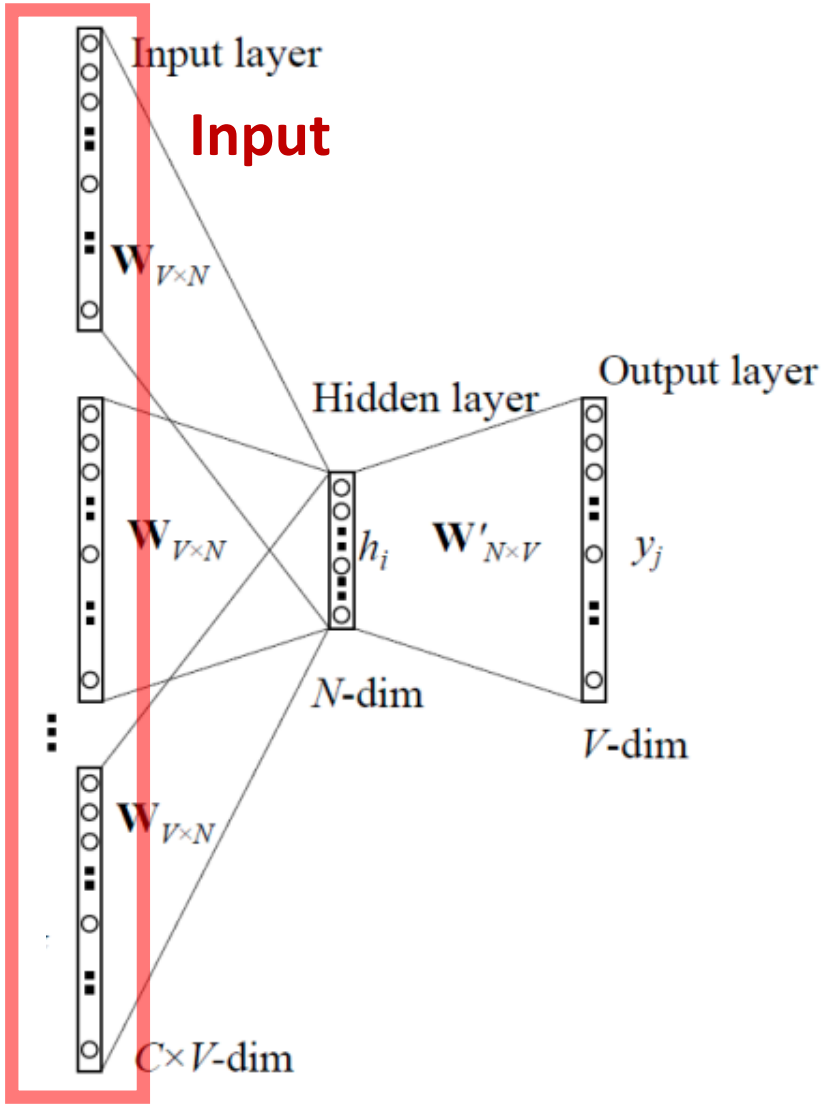
Word2Vec (CBOW)



매일 늦게 일어나서 _____ 안먹고 출근한다.

Text Representation

Word2Vec (CBOW)

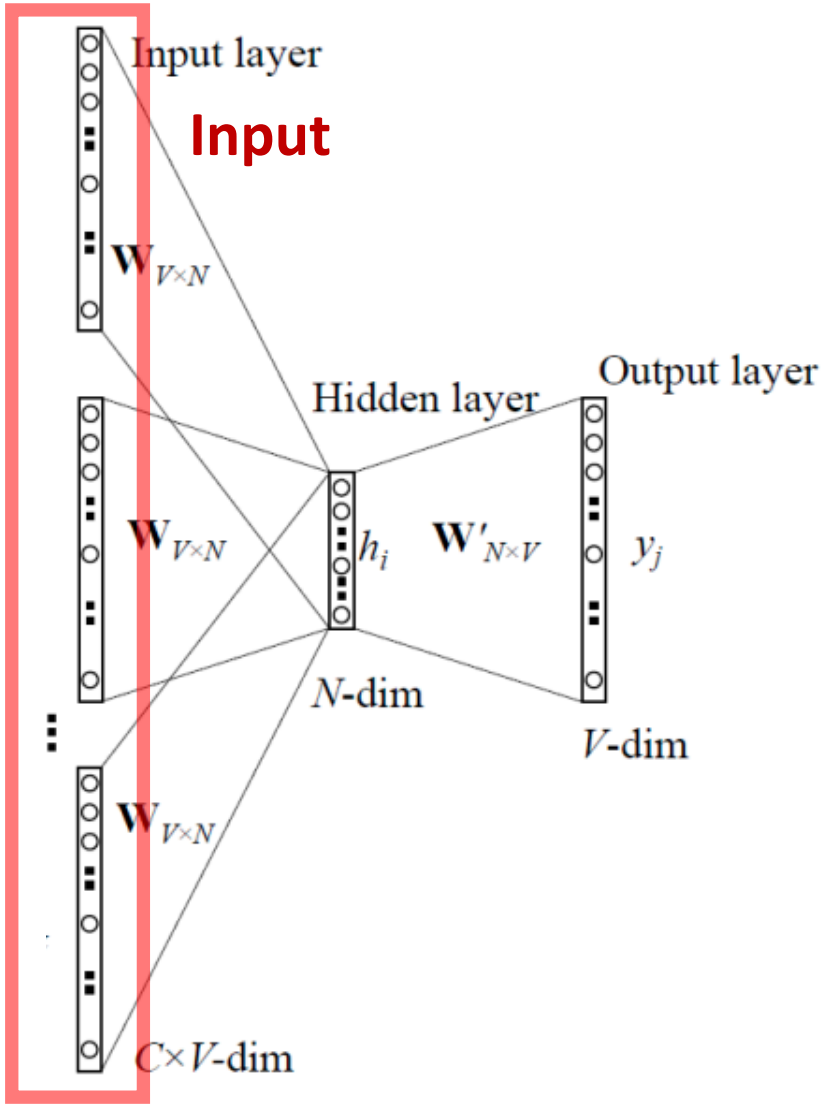


매일 늦게 일어나서 _____ 안먹고 출근한다.

Text Representation

0	0	0	1
1	0	0	0
0	0	0	0
⋮	⋮	⋮	⋮
0	1	0	0
⋮	⋮	⋮	⋮
0	0	1	0
0	0	0	0

Word2Vec (CBOW)



매일 늦게 일어나서 _____ 안먹고 출근한다.

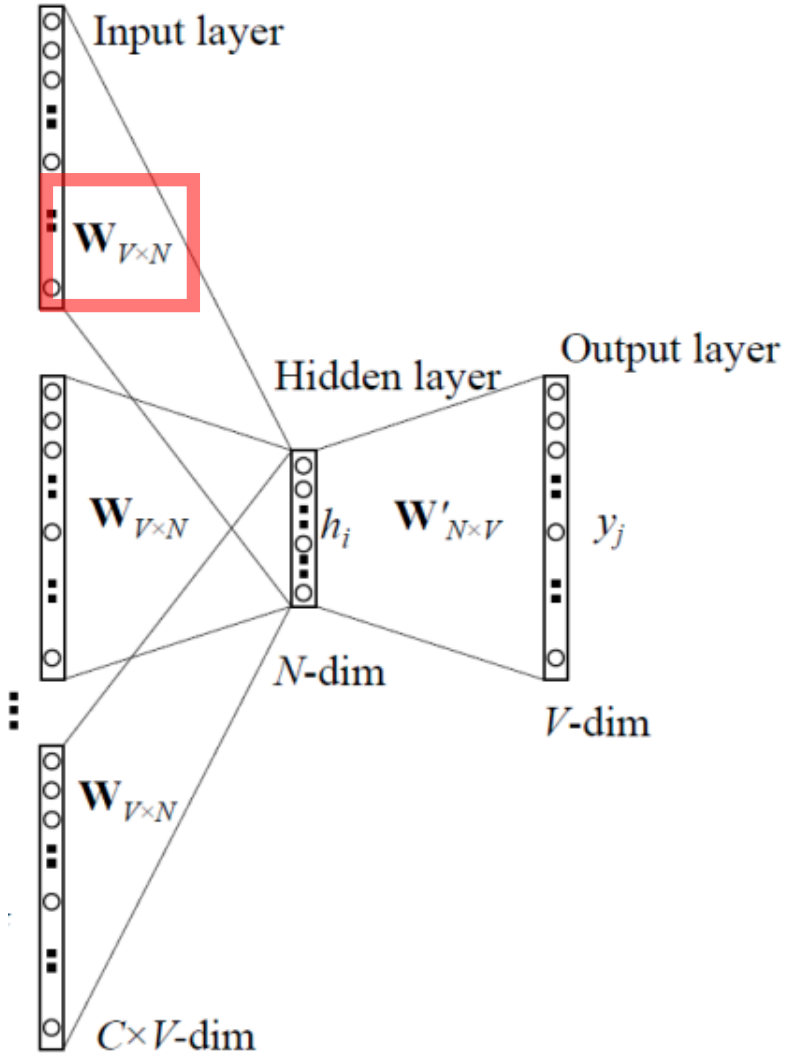
Text Representation

0	0	0	1
1	0	0	0
0	0	0	0
⋮	⋮	⋮	⋮
0	1	0	0
⋮	⋮	⋮	⋮
0	0	1	0
0	0	0	0

Shape = (1, V)

V: Vocabulary size

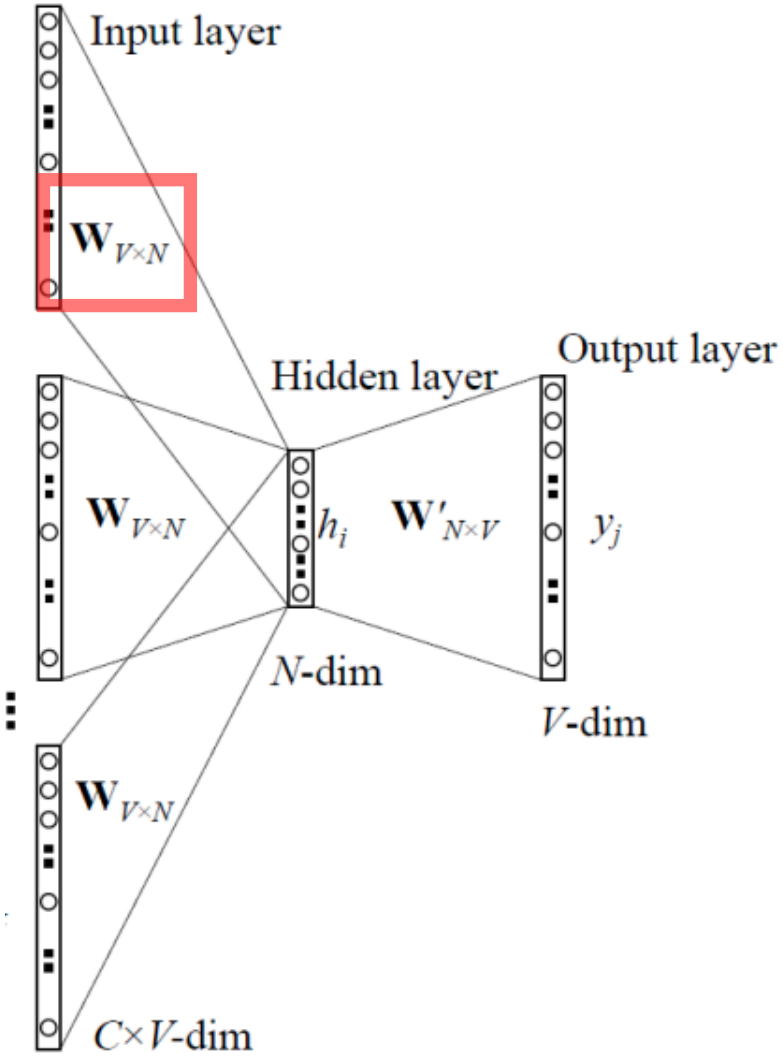
Word2Vec (CBOW)



$$W \in R^{V\times N}$$

- V: Vocabulary size
- N: Embedding 하고자 하는 dimension 크기

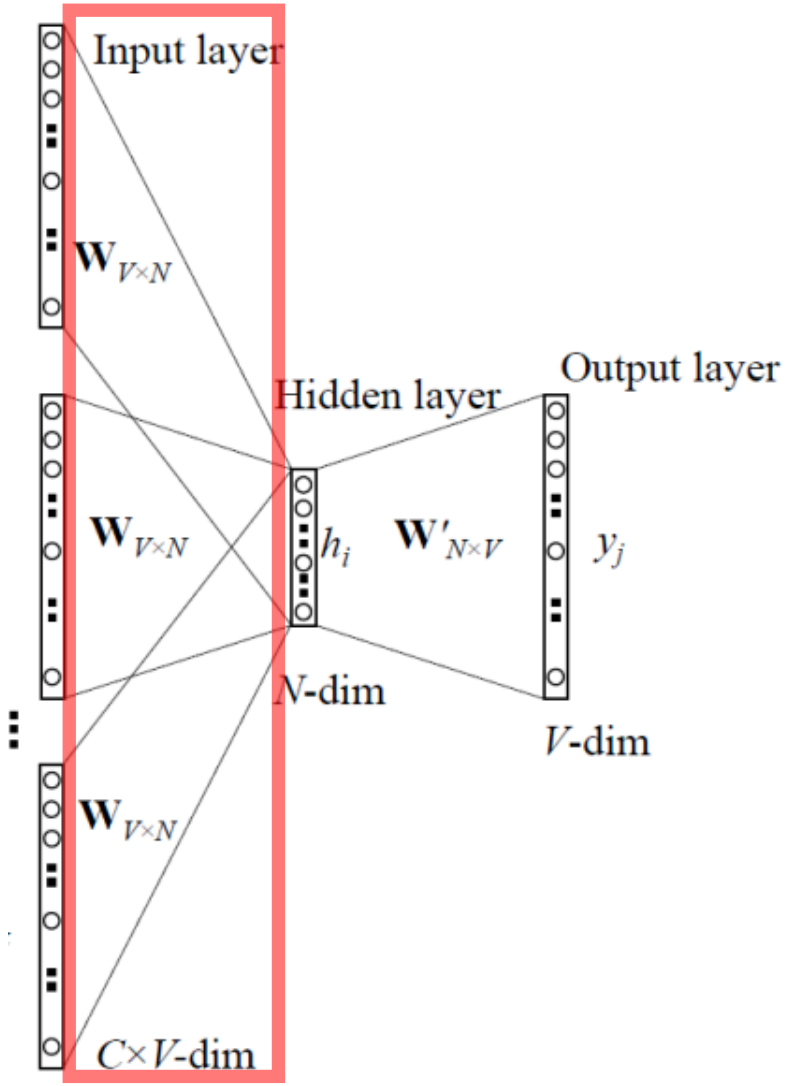
Word2Vec (CBOW)



$W \in R^{V\times N}$ **=> shape of (V , N)**

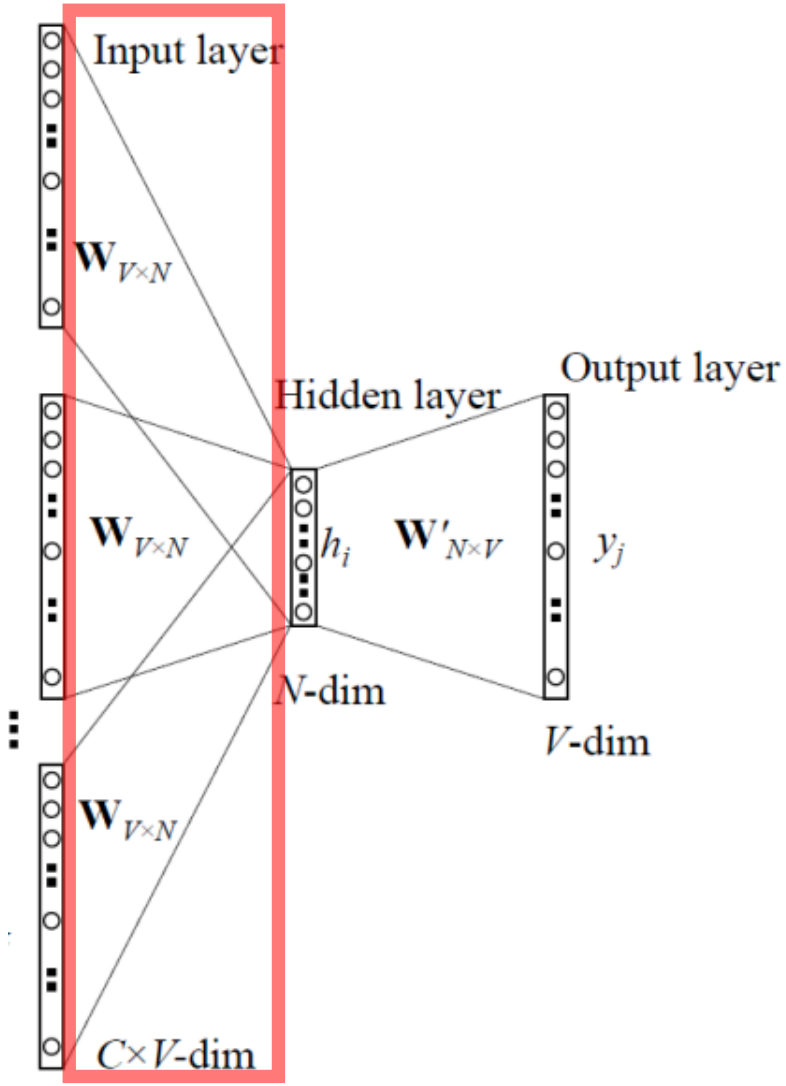
- V: Vocabulary size
- N: Embedding 하고자 하는 dimension 크기

Word2Vec (CBOW)



$$h_{\mathrm{t-m}} = x_{t-m}W$$

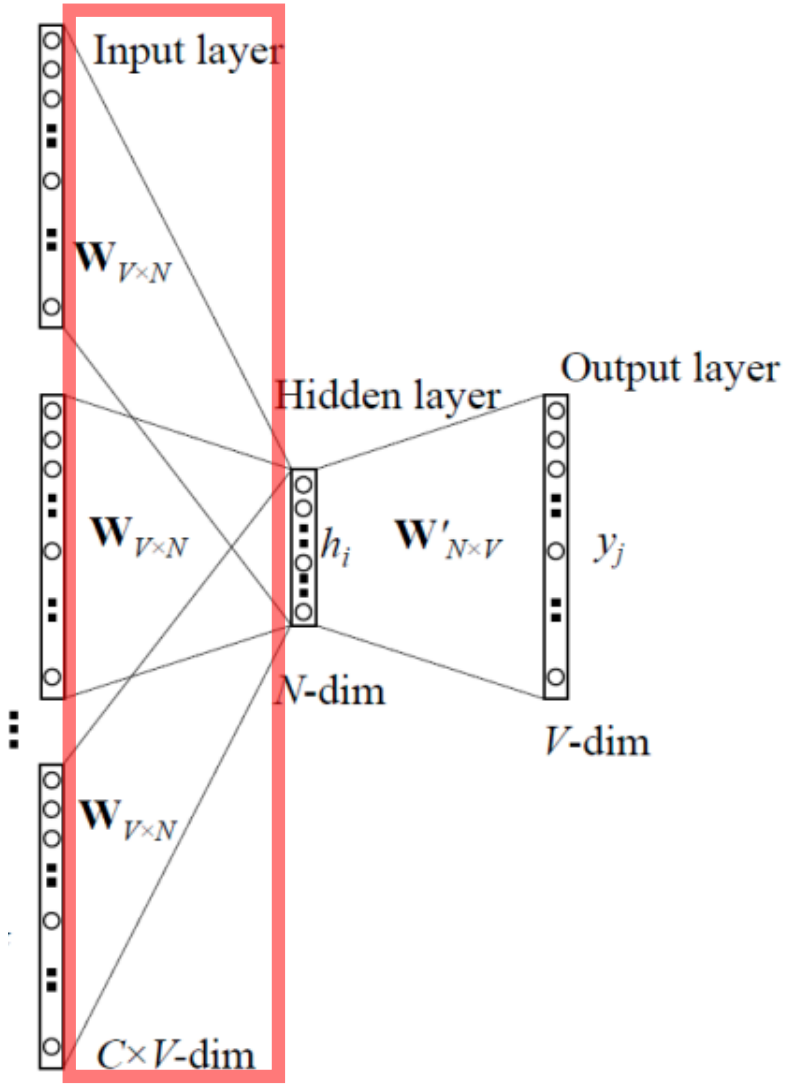
Word2Vec (CBOW)



$$h_{\mathrm{t-m}} = x_{t-m}W$$
$$\vdots$$
$$h_{\mathrm{t+m}} = x_{t+m}W$$

} 2m개

Word2Vec (CBOW)



$$h_{t-2} = x_{t-2}W$$

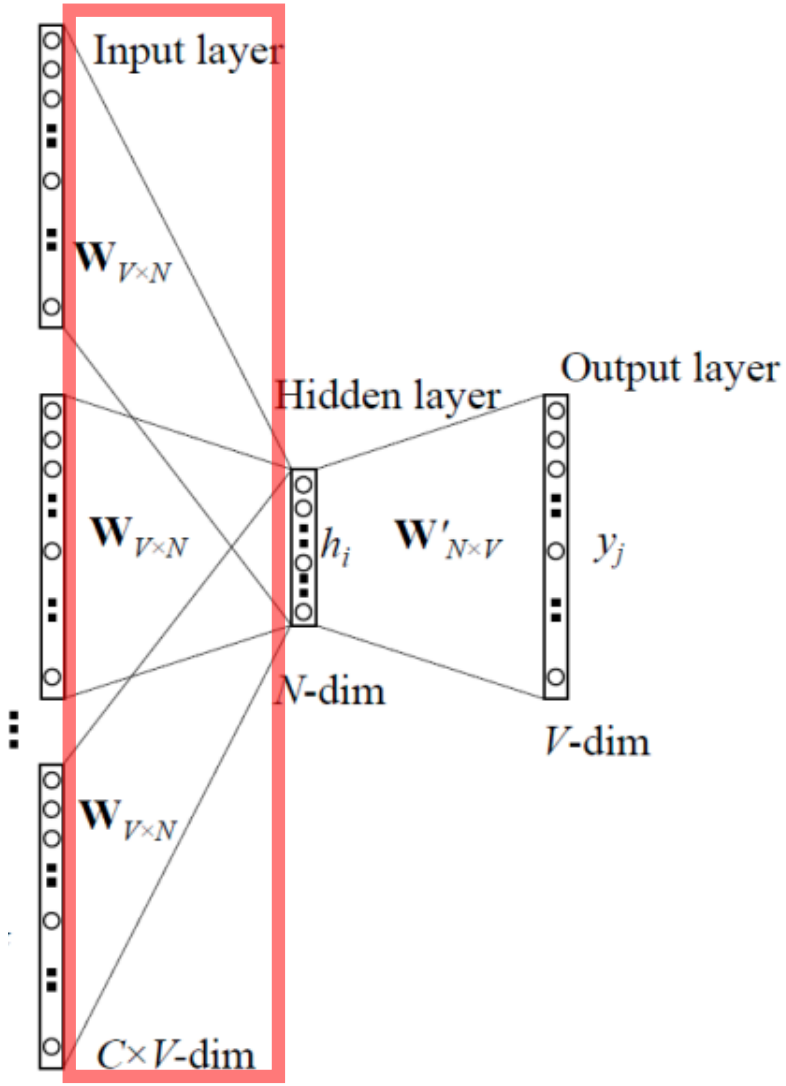
$$\vdots$$

$$h_{t+2} = x_{t+2}W$$

4개

매일 늦게 일어나서 _____ 안먹고 출근한다.

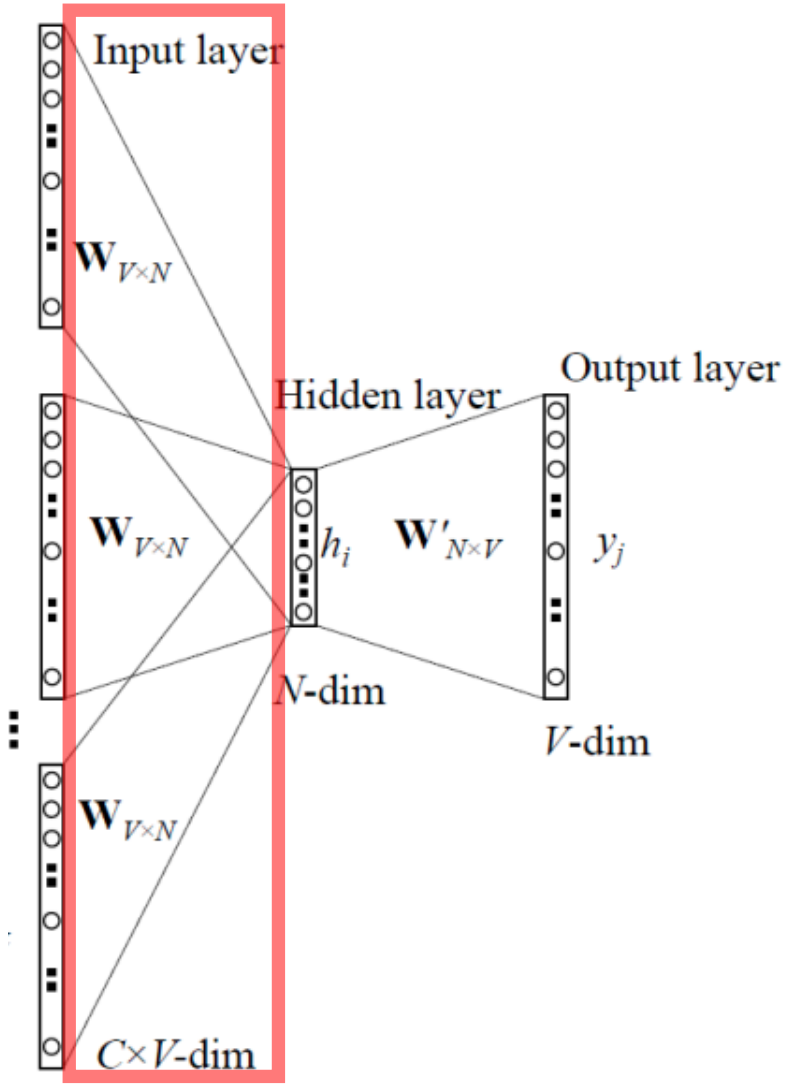
Word2Vec (CBOW)



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

x_{t-i} W h_{t-i}

Word2Vec (CBOW)

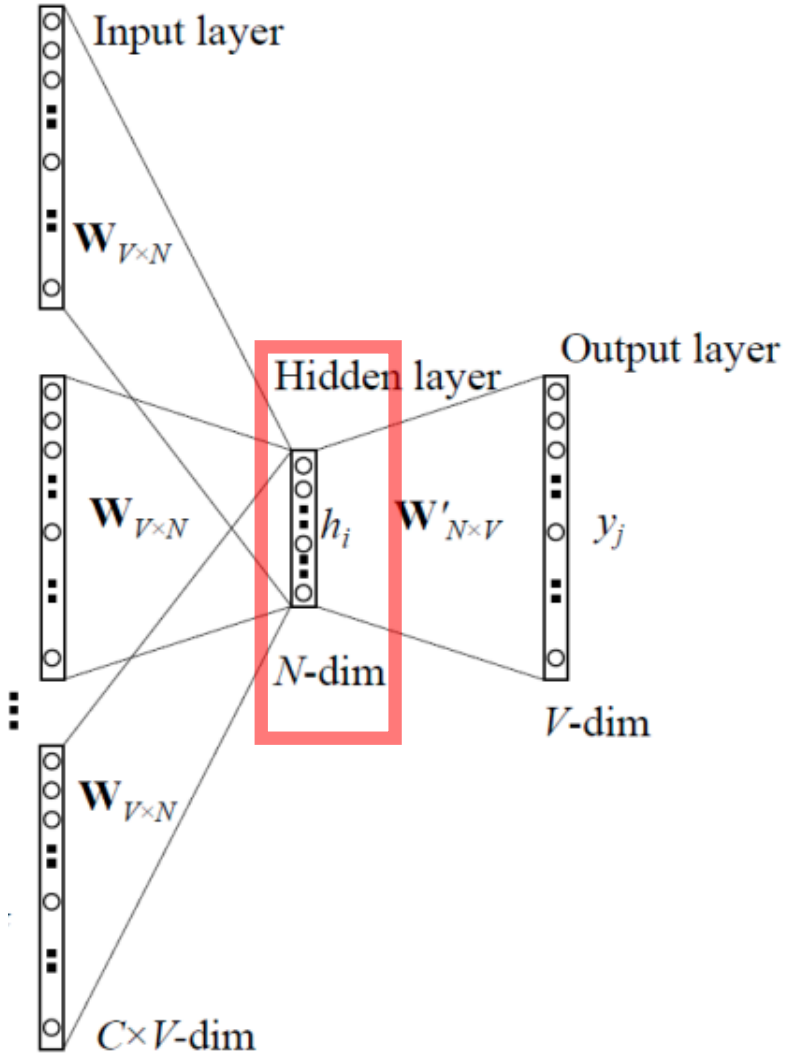


$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

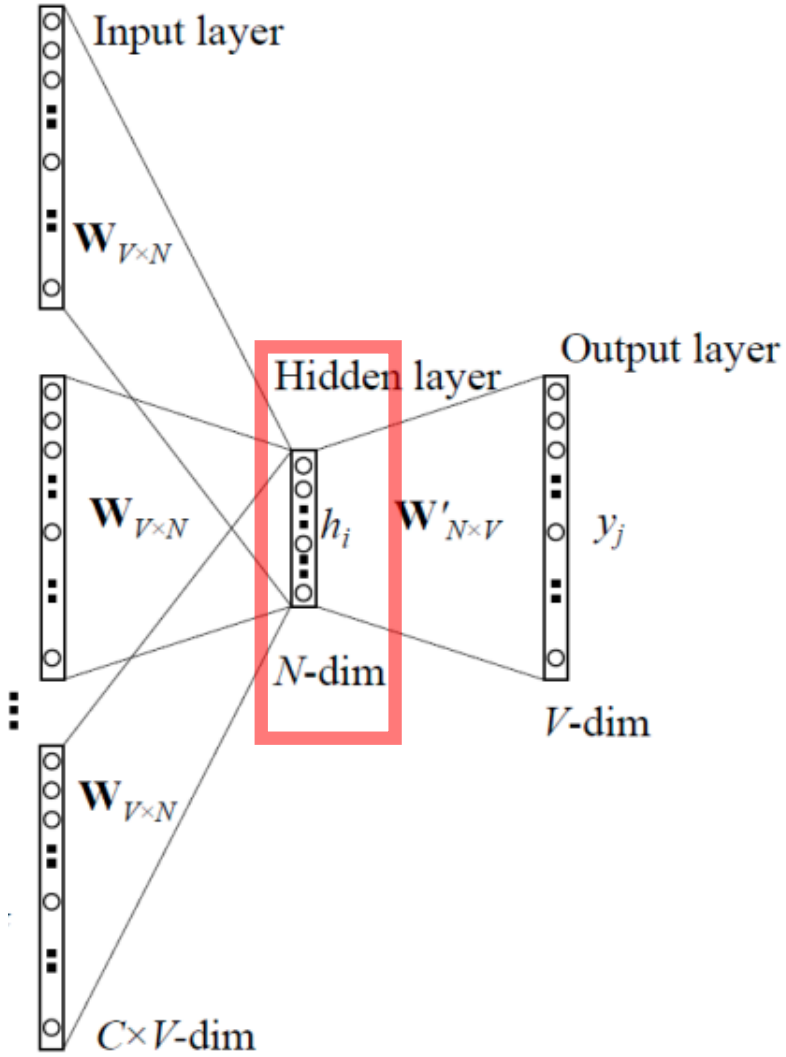
x_{t-i} W h_{t-i}

$(1, V)$ (V, N) $(1, N)$

Word2Vec (CBOW)



Word2Vec (CBOW)

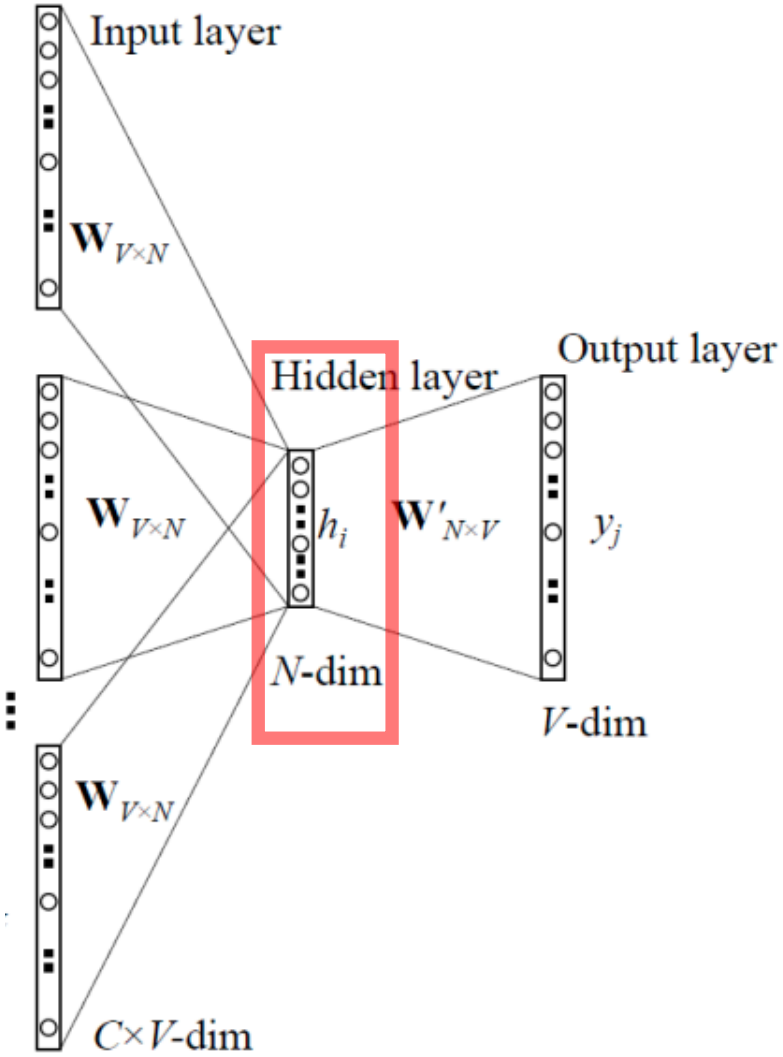


$$h_{t-m}, \ldots, h_{t-1}, h_{t+1}, \ldots, h_{t+n}$$

↓

하나의 vector $\boldsymbol{v}$

Word2Vec (CBOW)

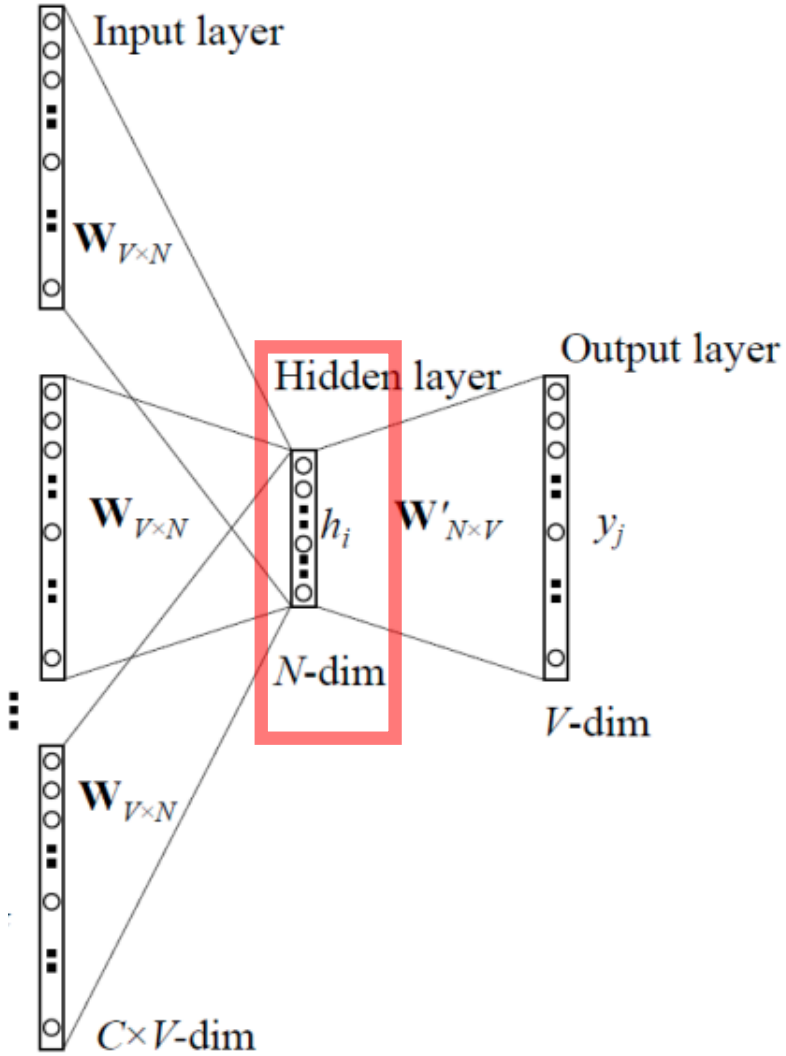


$$h_{t-m}, \ldots, h_{t-1}, h_{t+1}, \ldots, h_{t+n}$$

↓ 평균

하나의 vector $\boldsymbol{v}$

Word2Vec (CBOW)

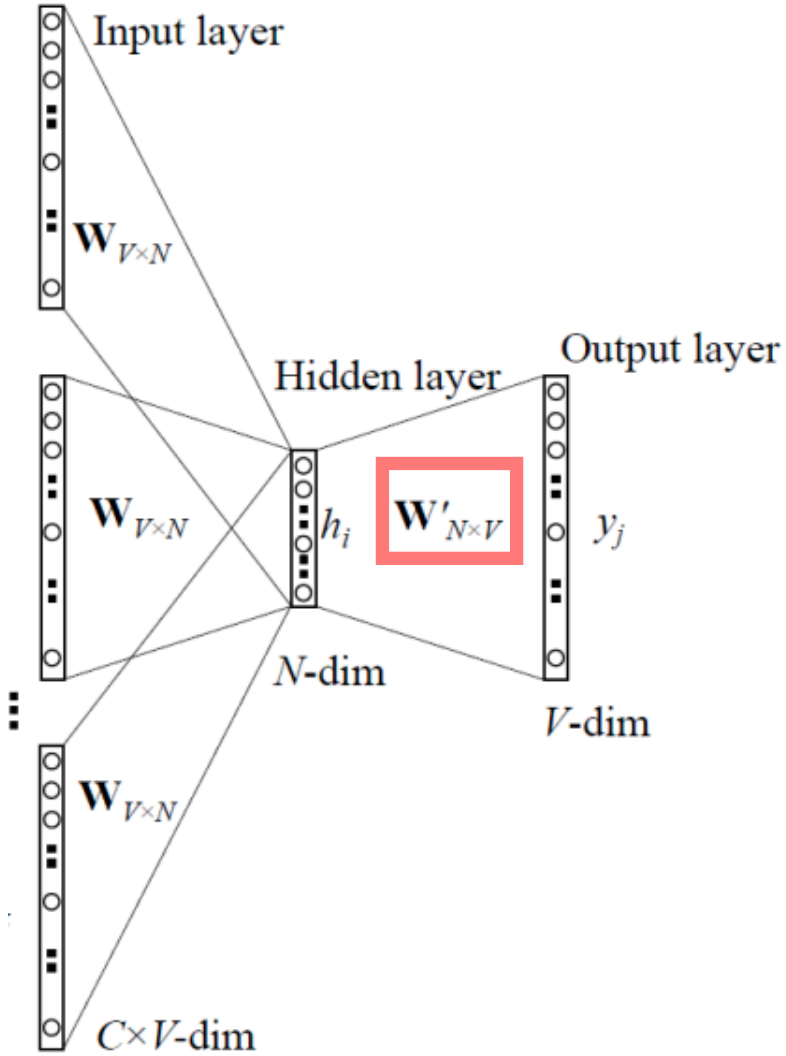


$$h_{t-m}, \dots, h_{t-1}, h_{t+2}, \dots, h_{t+n}$$

↓ 평균

$$h_t = \frac{h_{t-m} + \cdots + h_{t-1} + h_{t+1} + \cdots + h_{t+m}}{2m}$$

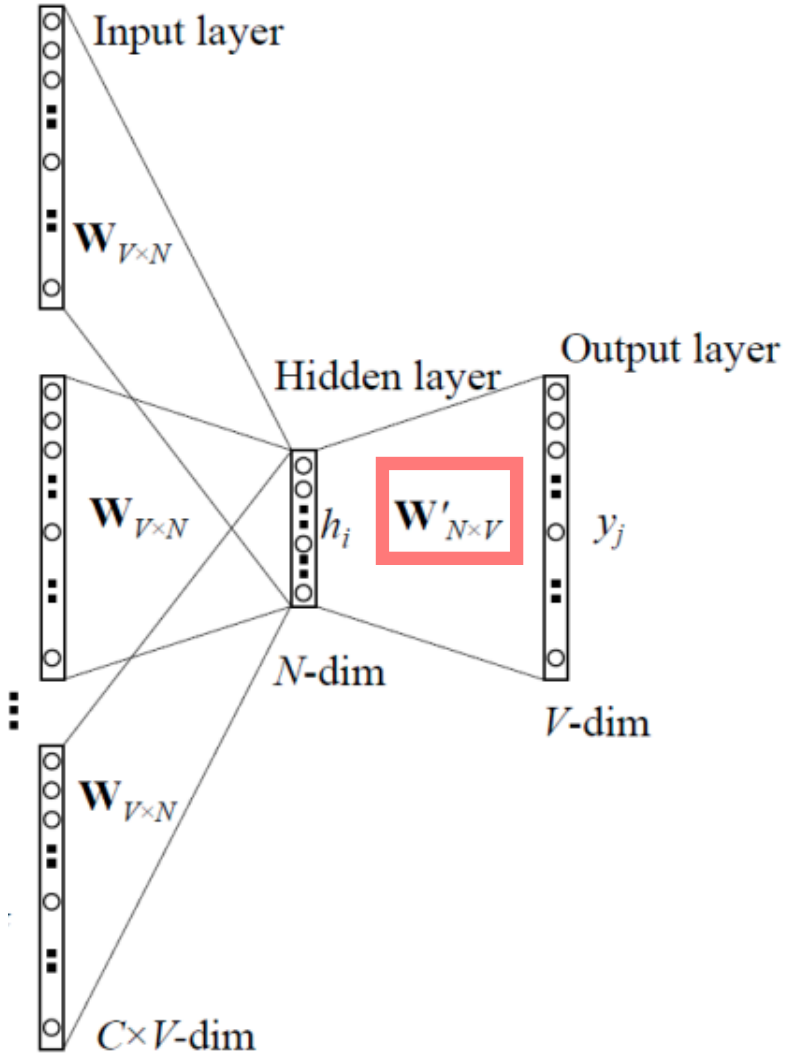
Word2Vec (CBOW)



$W' \in R^{V\times N}$ **=> shape of (N , V)**

- V: Vocabulary size
- N: Embedding 하고자 하는 dimension 크기

Word2Vec (CBOW)

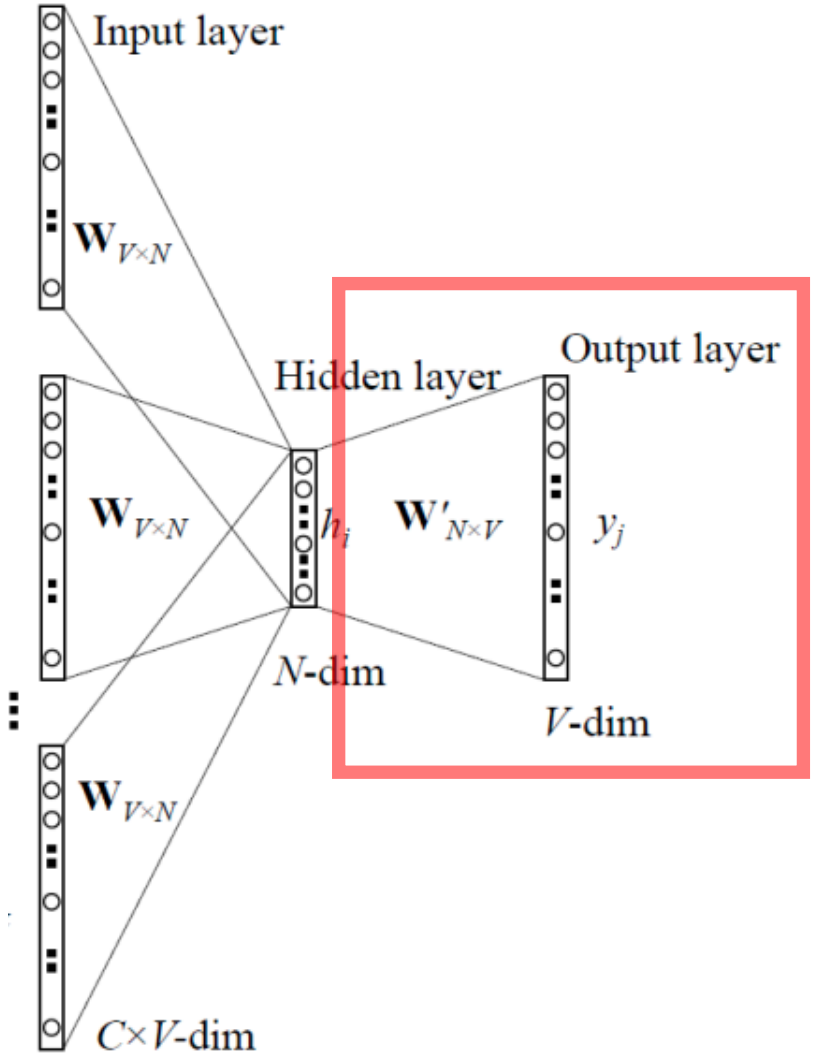


$W' \in R^{V\times N}$ **=> shape of (N , V)**

- V: Vocabulary size
- N: Embedding 하고자 하는 dimension 크기

Output으로 나올 단어 예측을 위해 v 크기의 벡터로 만들어줌.

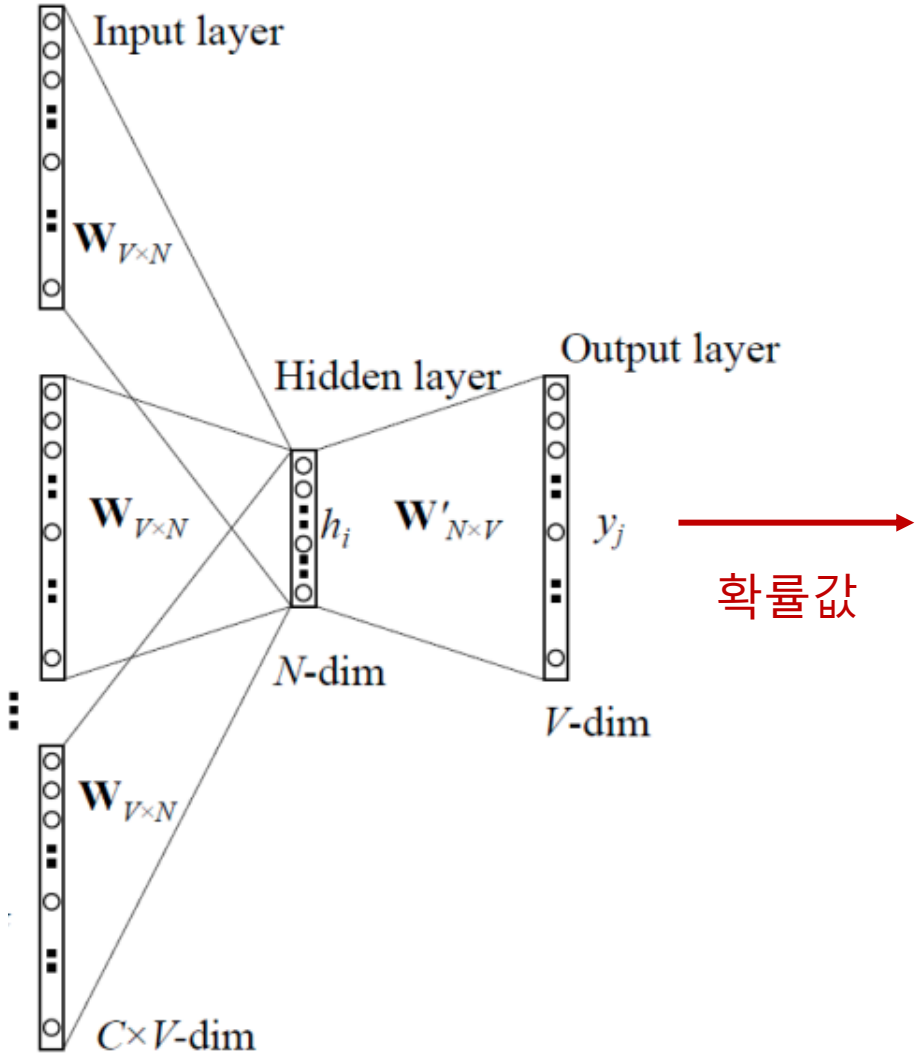
Word2Vec (CBOW)



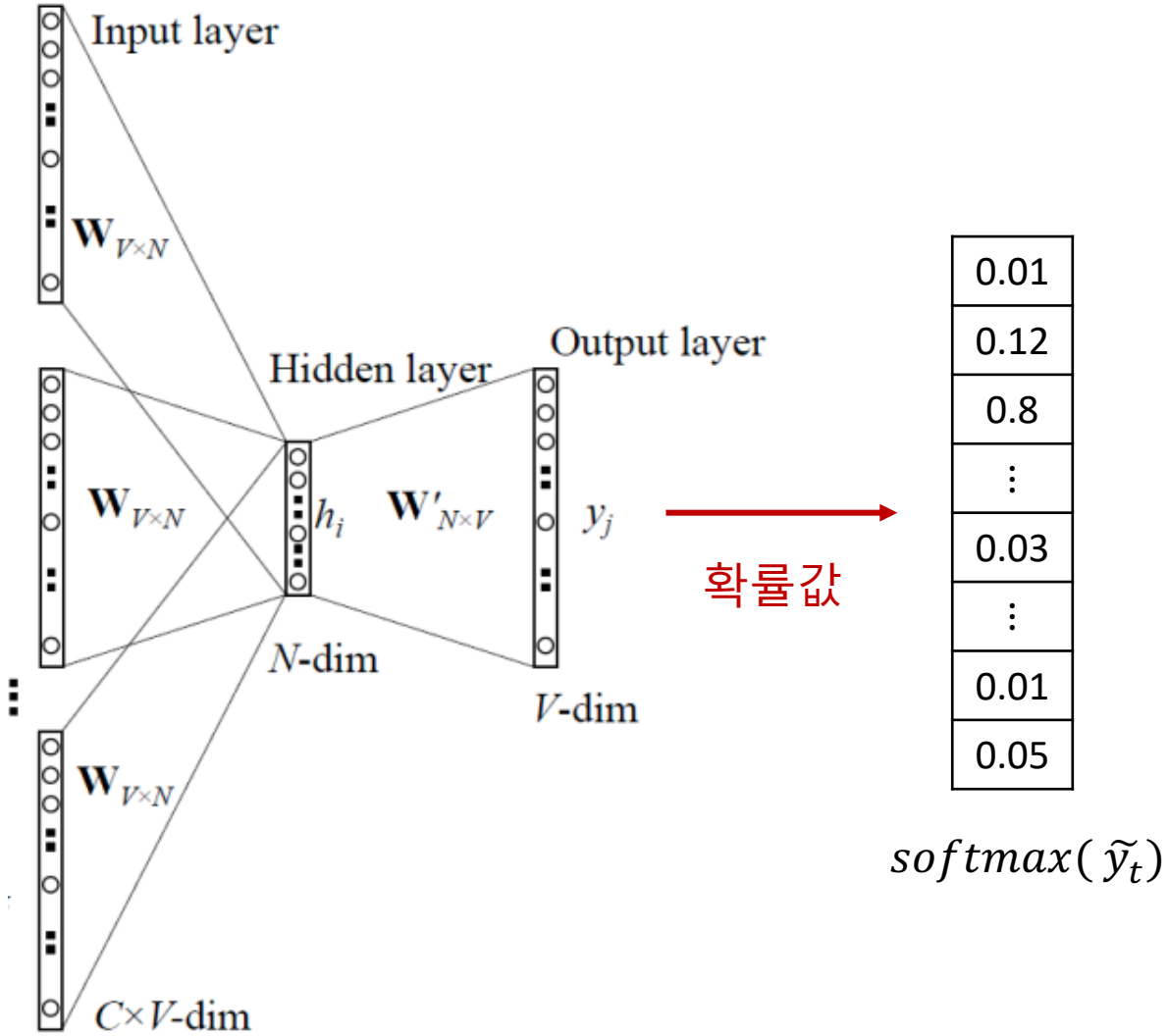
$$\tilde{y}_t = h_t W'$$

$$(1, V) = (1, N) \times (N, V)$$

Word2Vec (CBOW)



Word2Vec (CBOW)

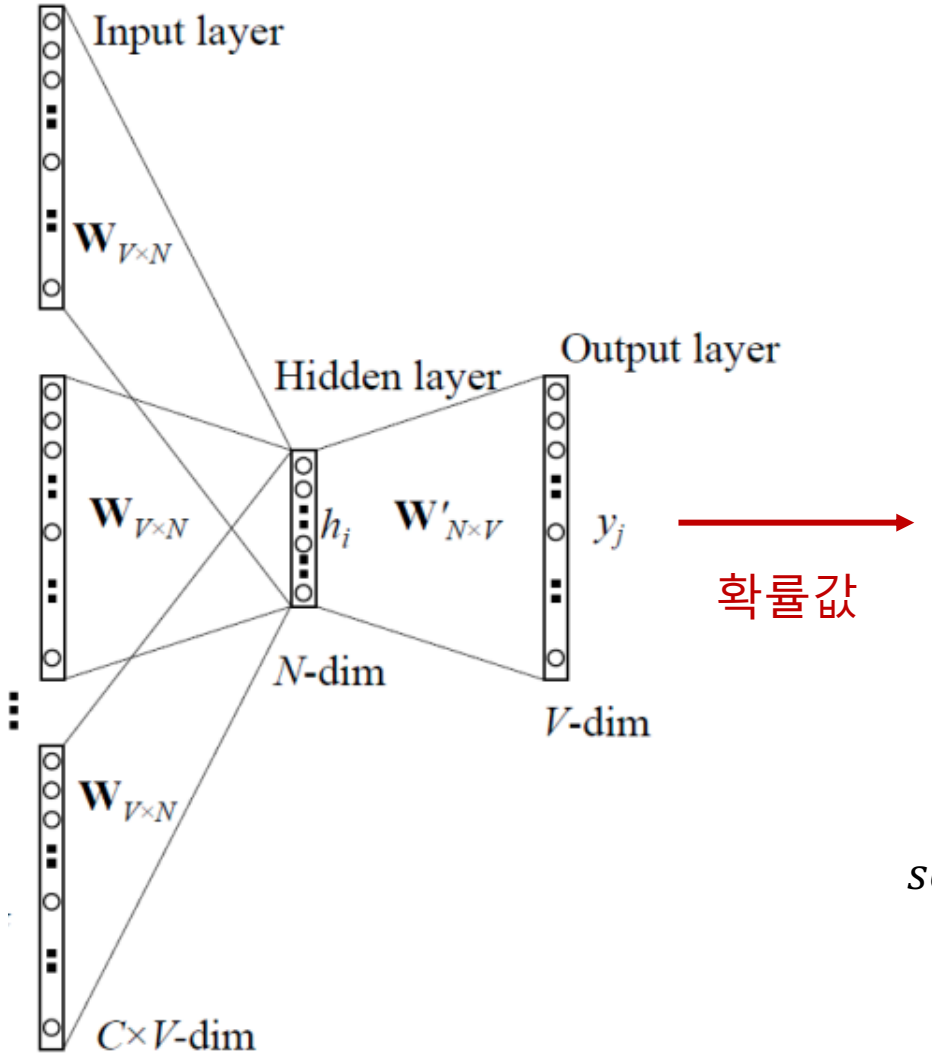


확률값

0.01
0.12
0.8
⋮
0.03
⋮
0.01
0.05

$softmax(\tilde{y}_t)$

Word2Vec (CBOW)



확률값

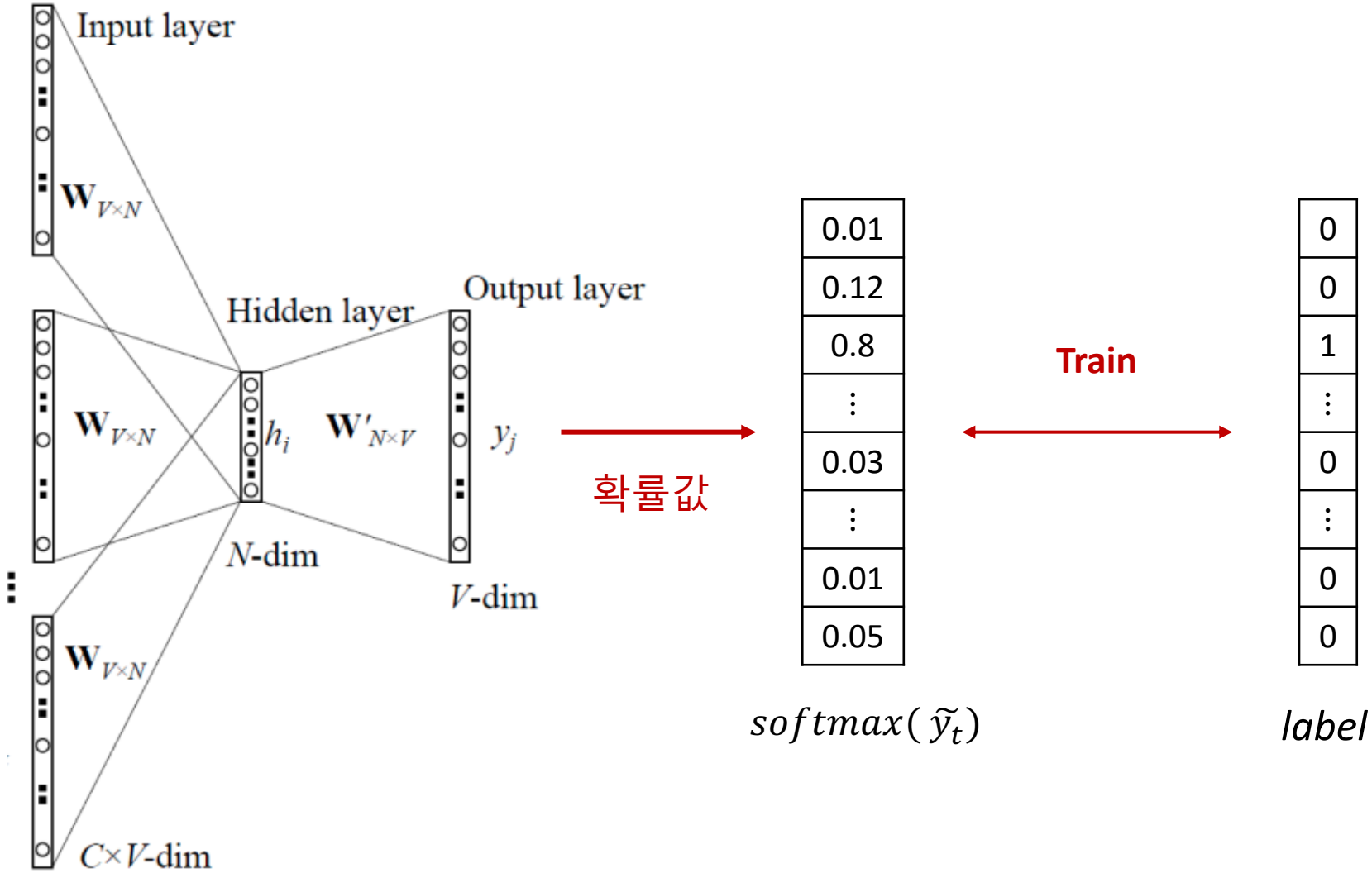
0.01
0.12
0.8
⋮
0.03
⋮
0.01
0.05

$softmax(\tilde{y}_t)$

0
0
1
⋮
0
⋮
0
0

label

Word2Vec (CBOW)

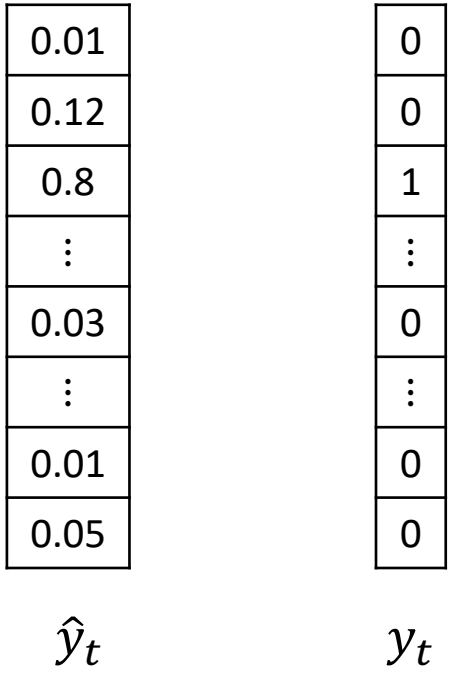


Word2Vec (CBOW)

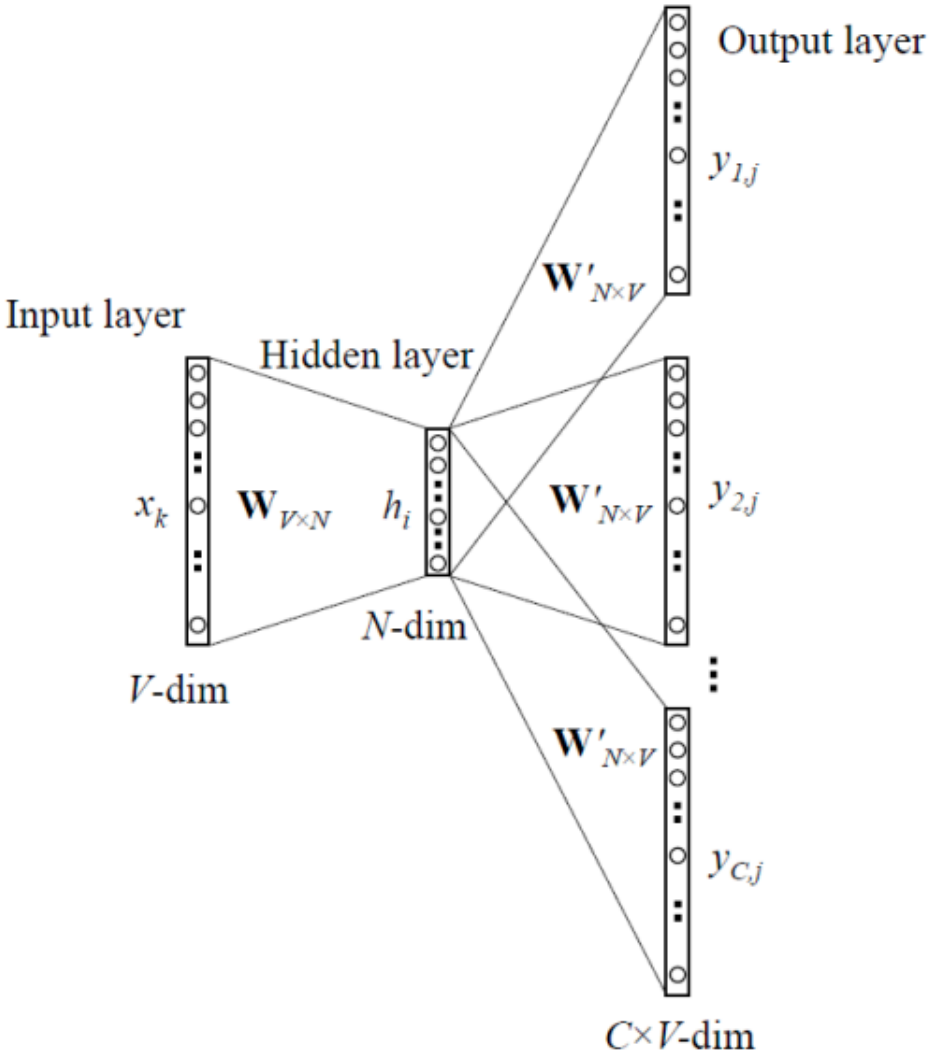
Cross Entropy Loss

$$H(y, \hat{y}) = -\sum_{t}^{T} y_t \log \hat{y}_t$$

$$= -(0 \cdot \log 0.01 + 0 \cdot \log 0.12 + 1 \cdot \log 0.8 + \cdots + 0 \cdot \log 0.05)$$



Word2Vec (Skip-Gram)



Word2Vec

핵심 Idea

CBOW

매일 늦게 일어나서 _____ 안먹고 출근한다.

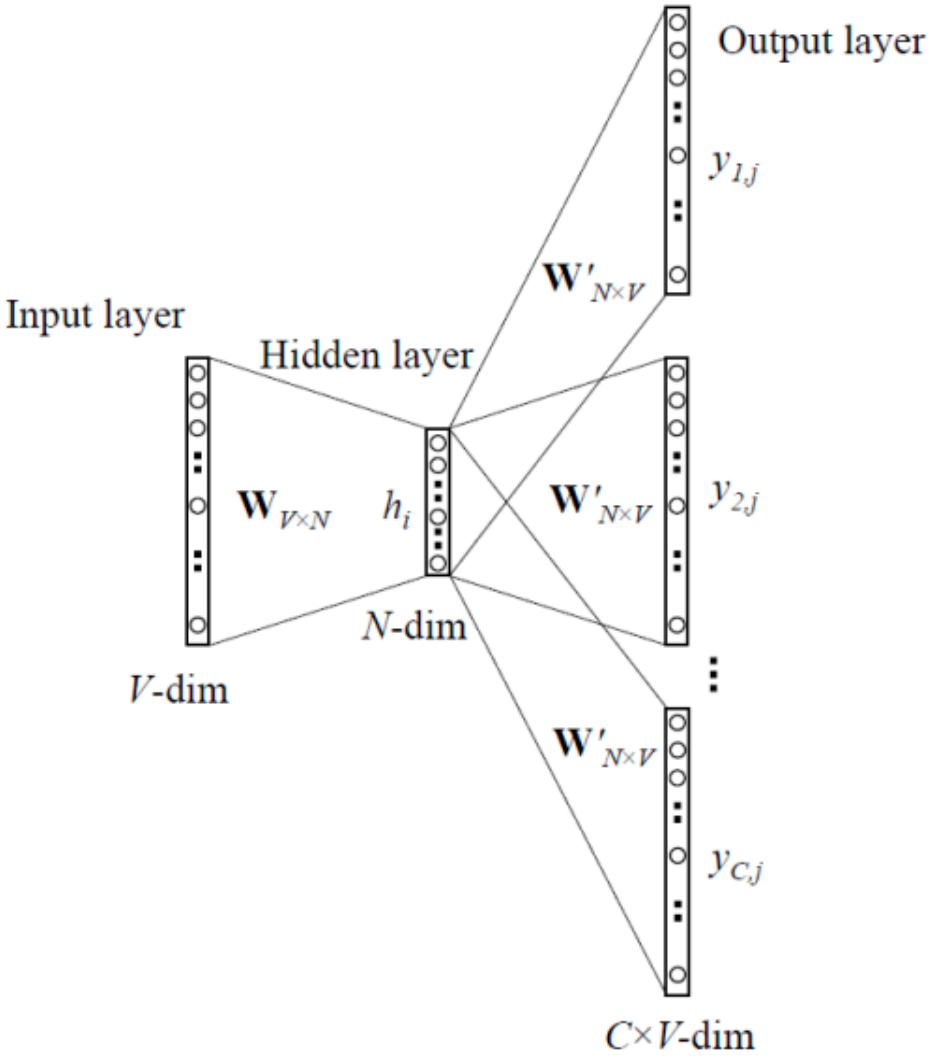
주변의 여러 단어들을 통해 하나의 단어를 예측

Skip-Gram

____ ____ _____ 아침을 _____ _____

하나의 단어를 통해 주변 여러 단어들을 예측

Word2Vec (Skip-Gram)



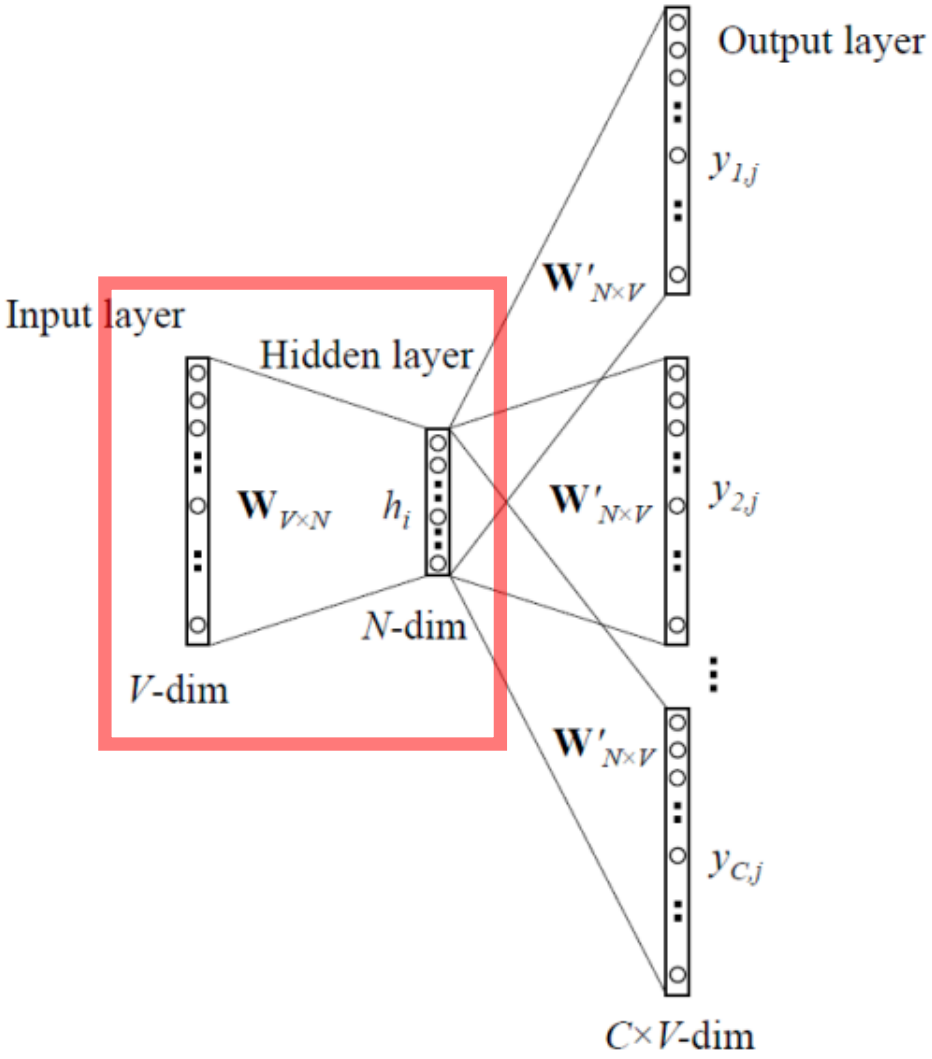
Input

- CBOW: 2m개
- Skip-Gram: 1개

Output

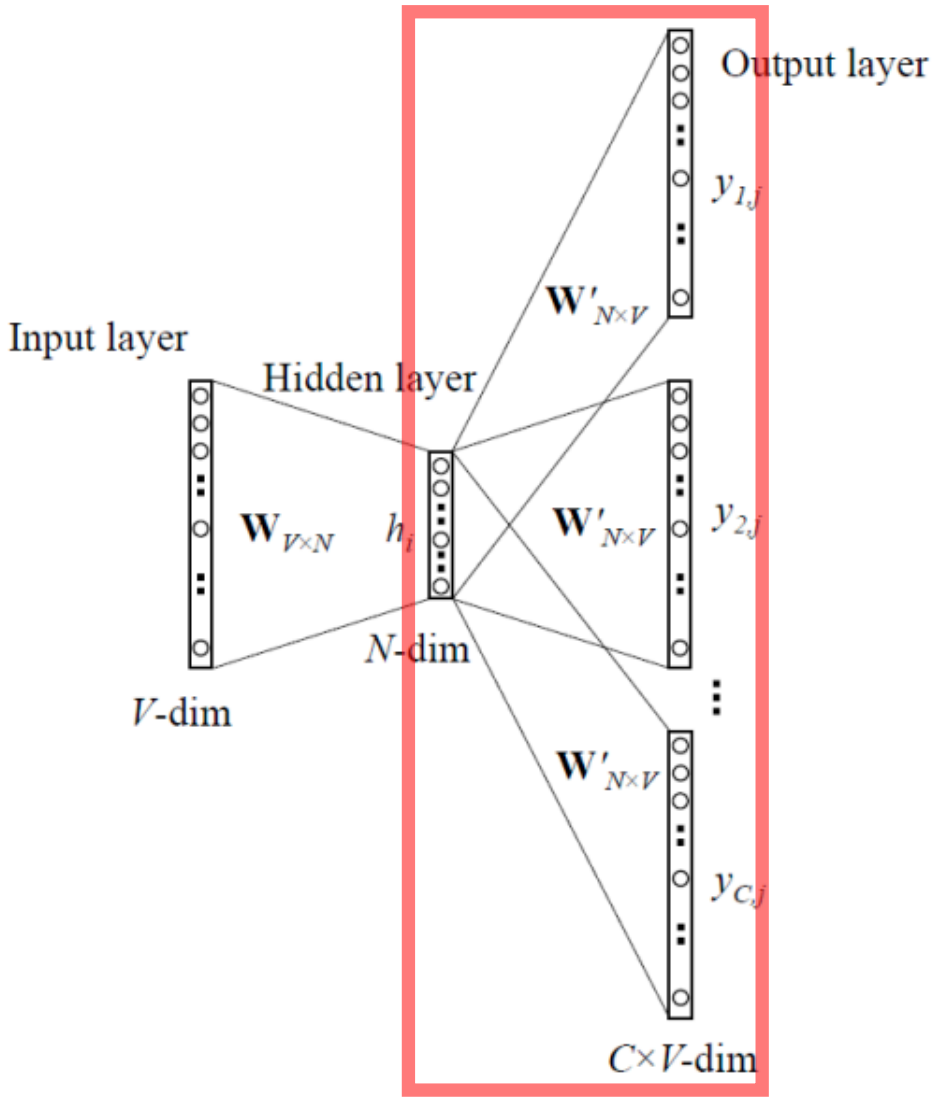
- CBOW: 1개
- Skip-Gram: 2m개

Word2Vec (Skip-Gram)



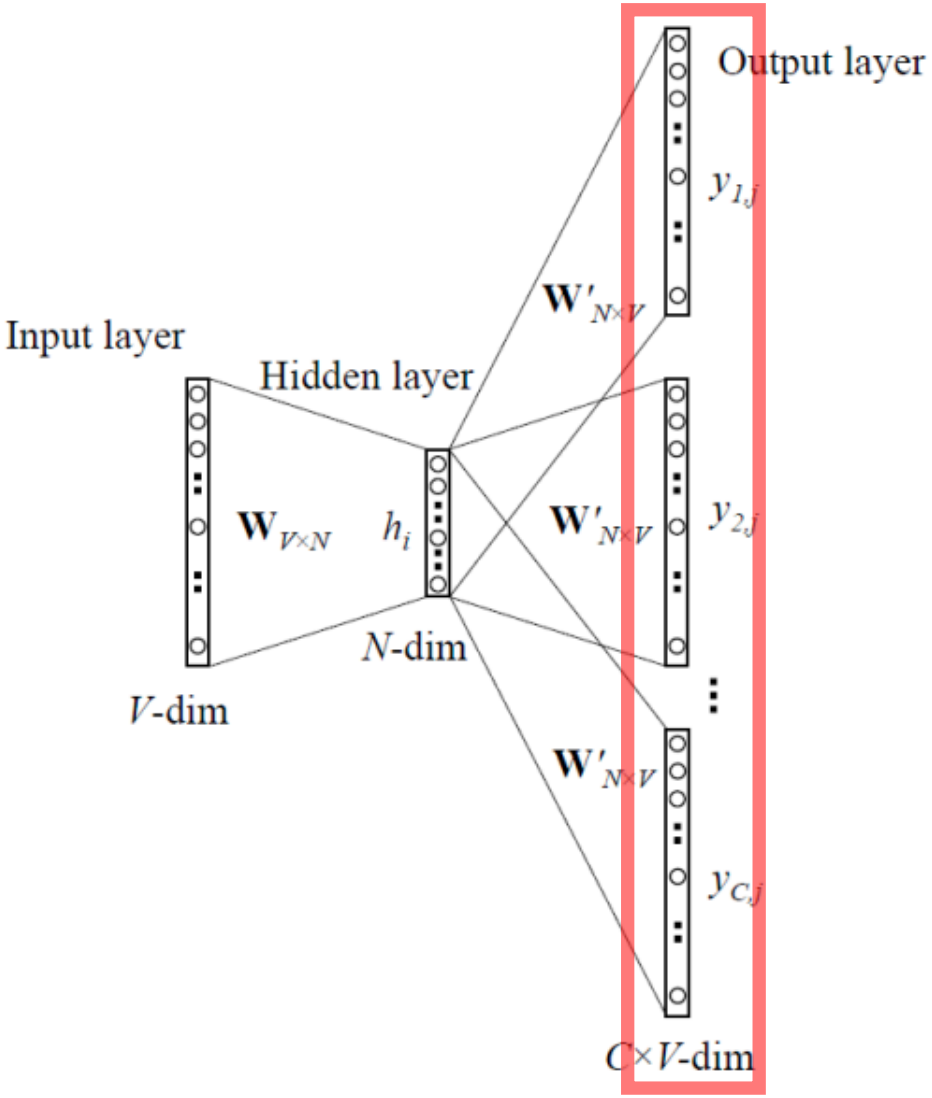
1. $h_{\mathrm{t-m}} = x_{t-m}W$

Word2Vec (Skip-Gram)



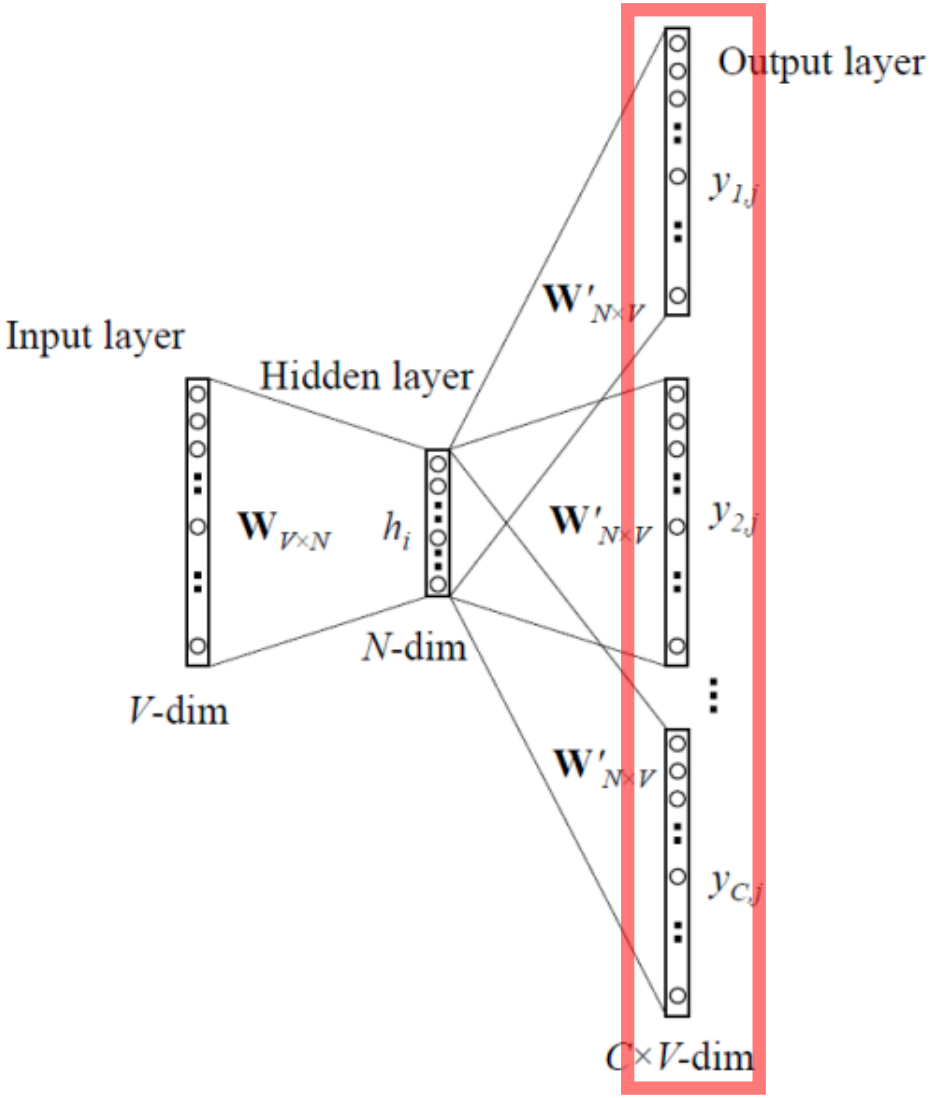
1. $h_t = x_t W$
2. $\tilde{y}_t = h_t W'$

Word2Vec (Skip-Gram)



1. $h_t = x_t W$
2. $\tilde{y}_t = h_t W'$
3. $\hat{y}_t = softmax(\tilde{y}_t)$

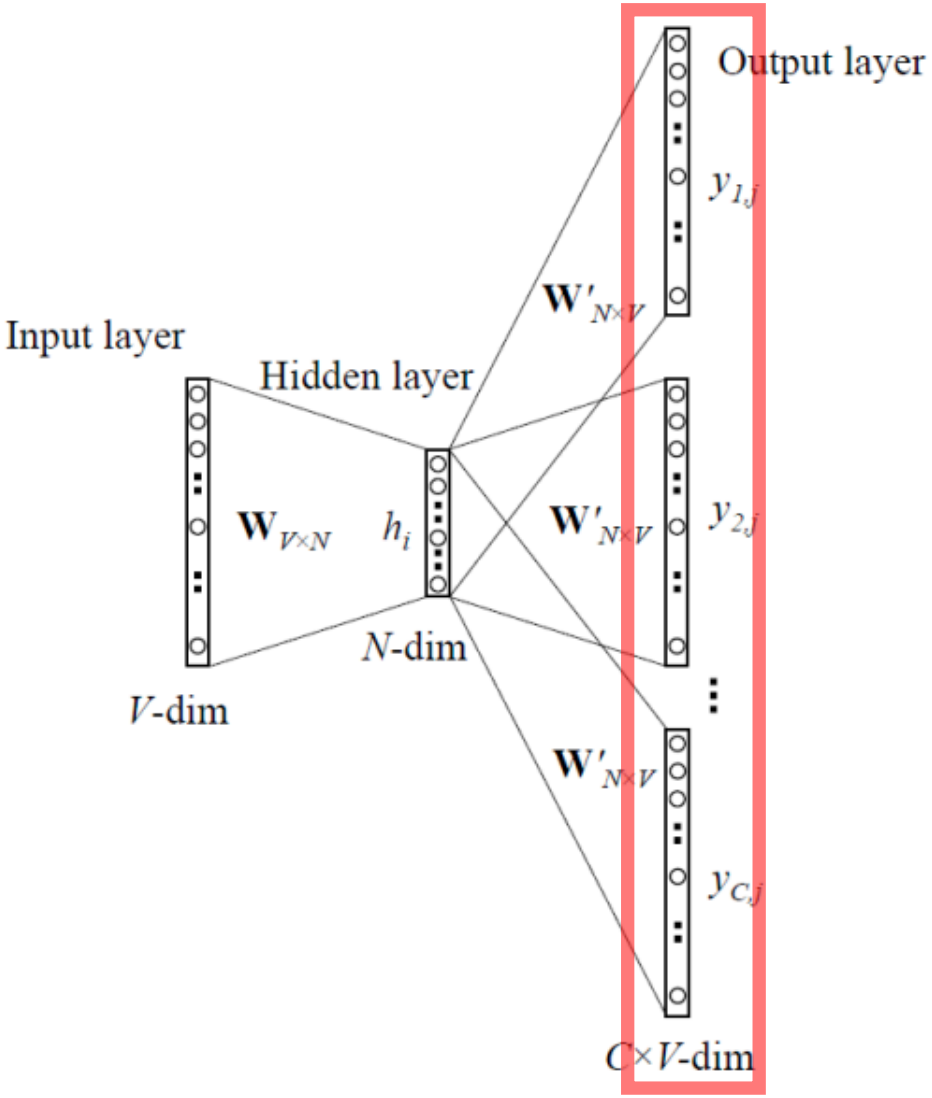
Word2Vec (Skip-Gram)



1. $h_t = x_t W$
2. $\tilde{y}_t = h_t W'$
3. $\hat{y}_t = softmax(\tilde{y}_t)$

Label 이 2m개

Word2Vec (Skip-Gram)

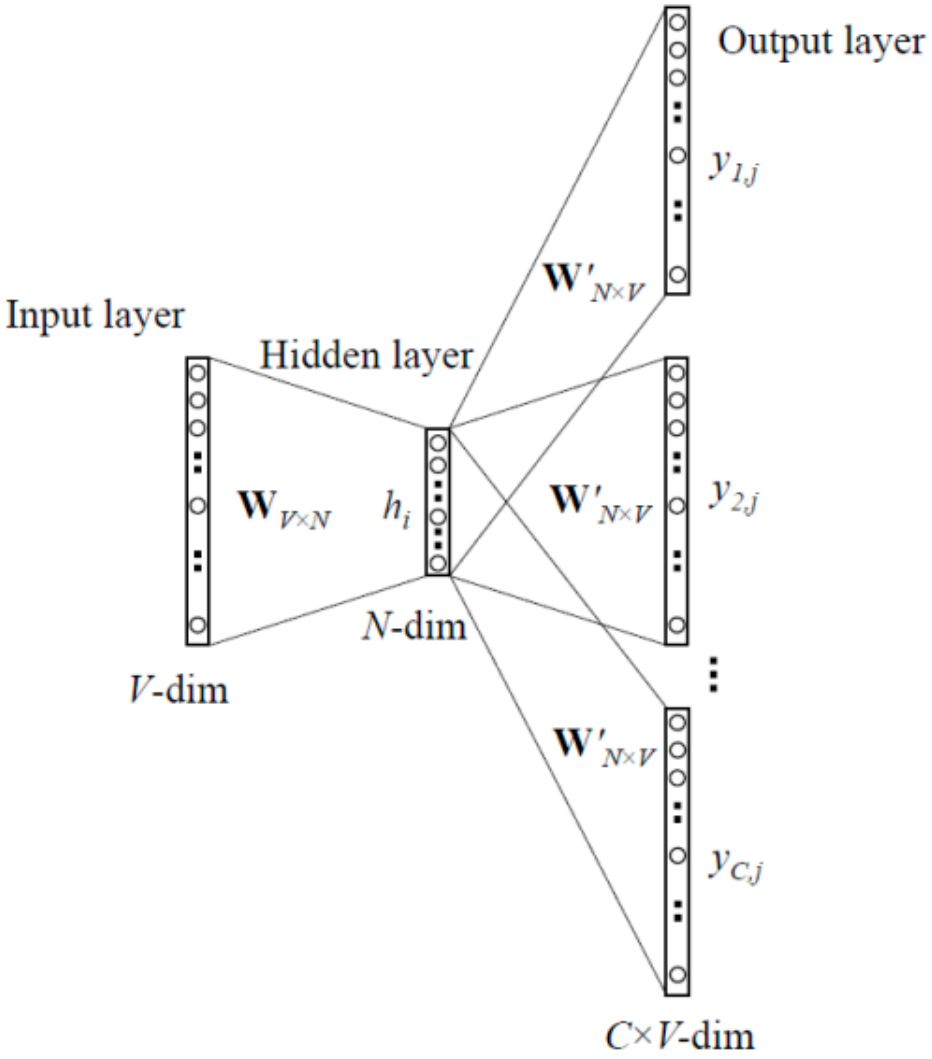


1. $h_t = x_t W$
2. $\tilde{y}_t = h_t W'$
3. $\hat{y}_t = softmax(\tilde{y}_t)$

Label 이 2m개

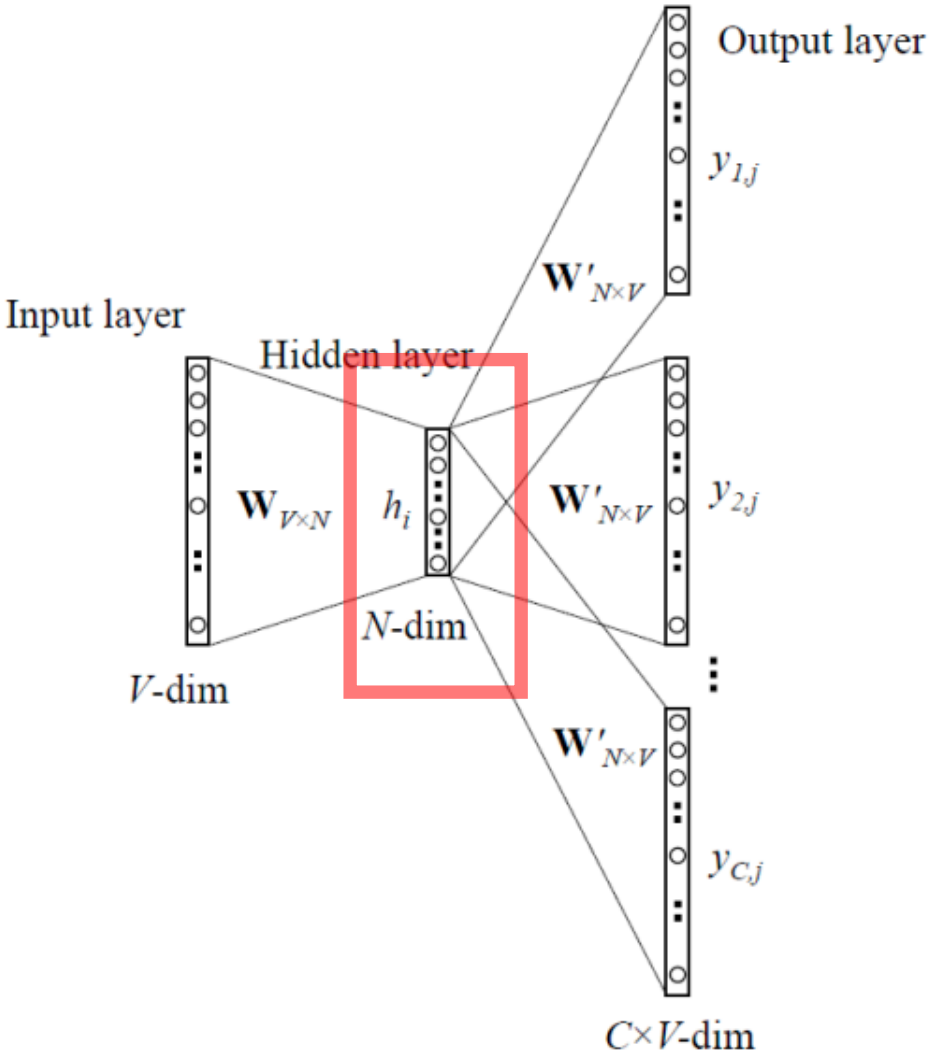
=> 총 2m번 학습

Word2Vec (Skip-Gram)



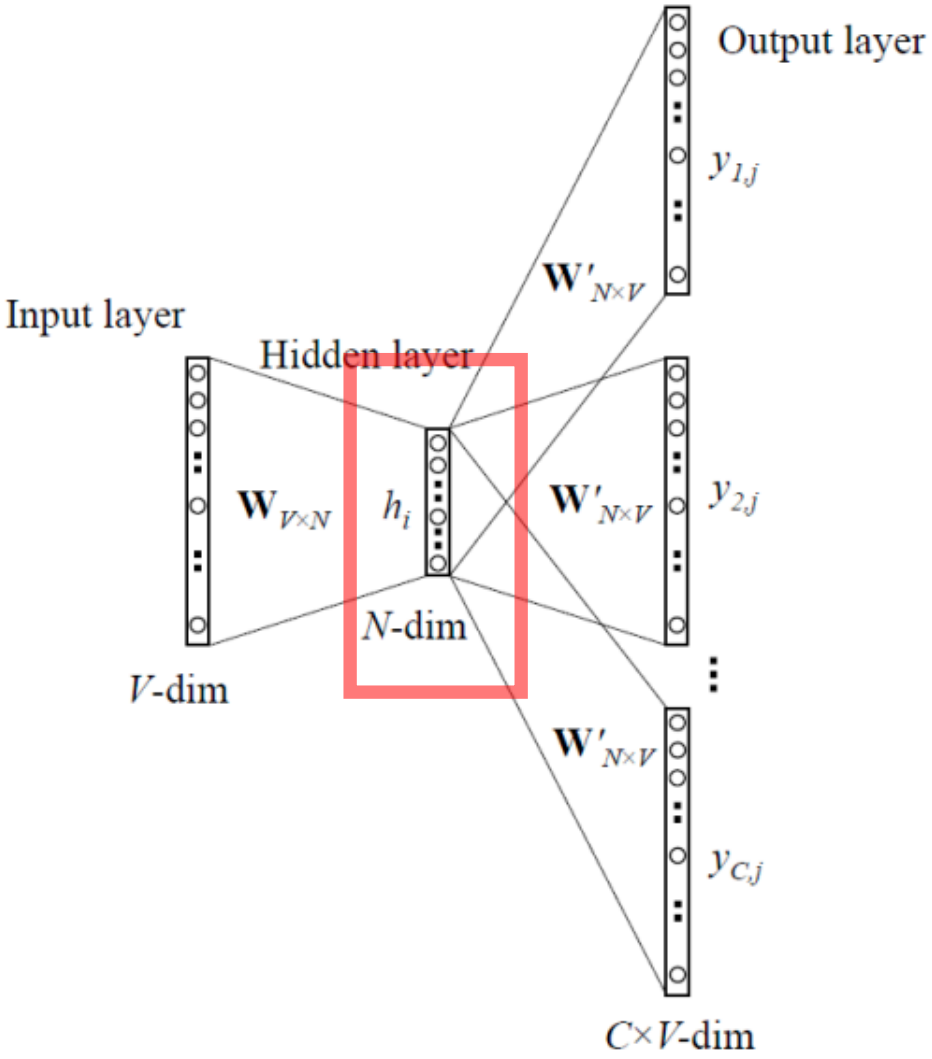
Embedding vector?

Word2Vec (Skip-Gram)



Embedding vector?

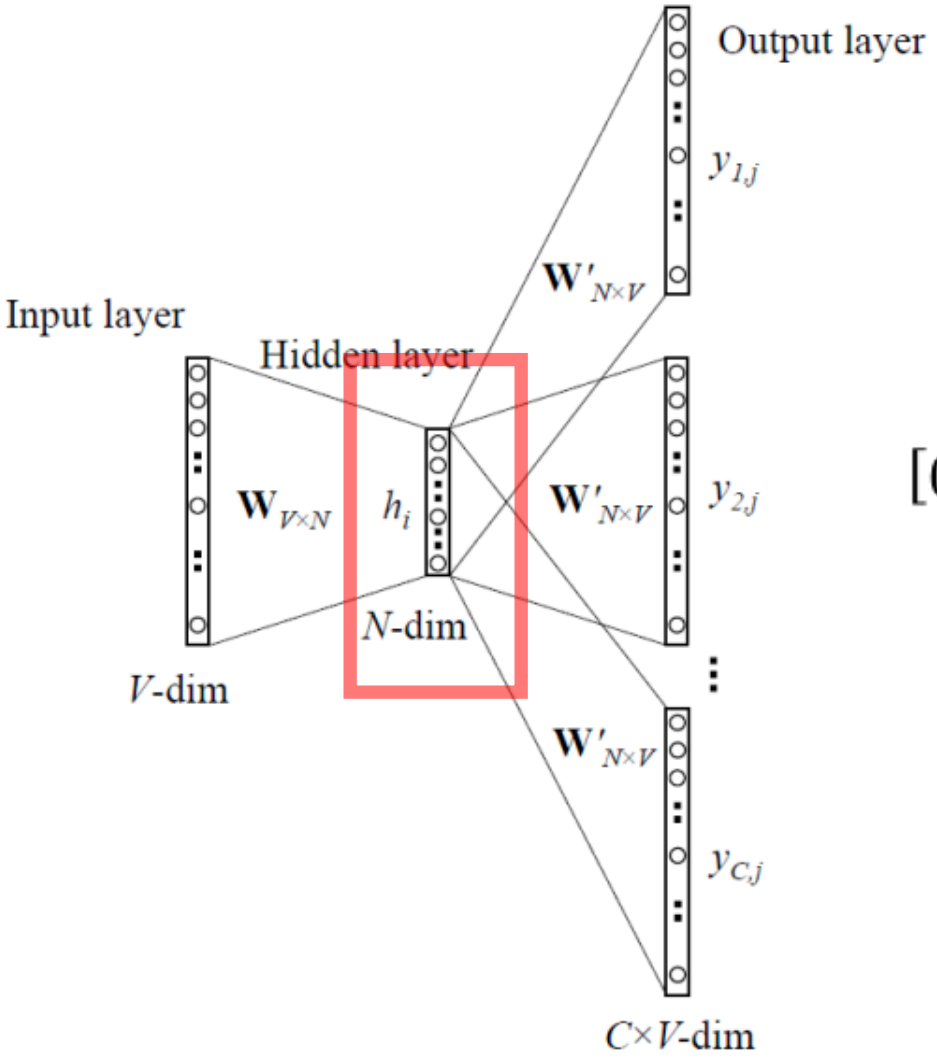
Word2Vec (Skip-Gram)



Embedding vector?

$$x_t\, W = h_t$$

Word2Vec (Skip-Gram)

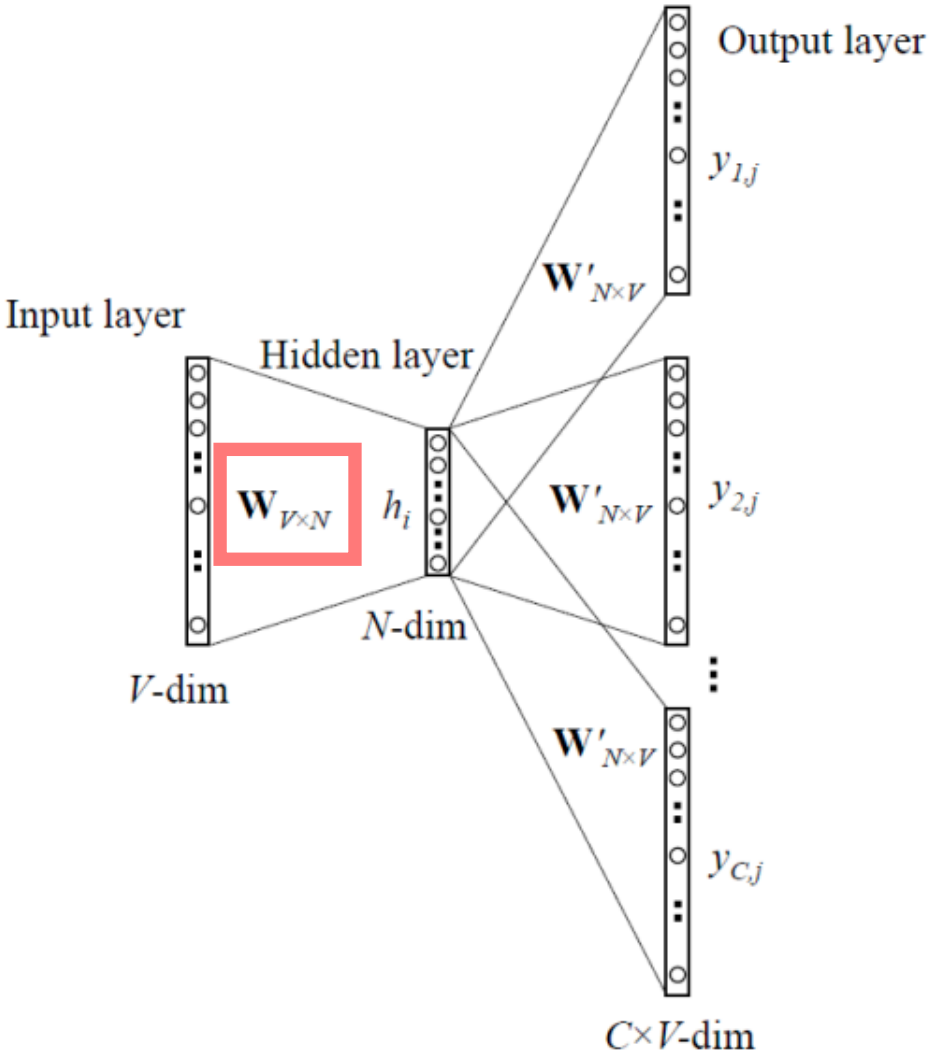


Embedding vector?

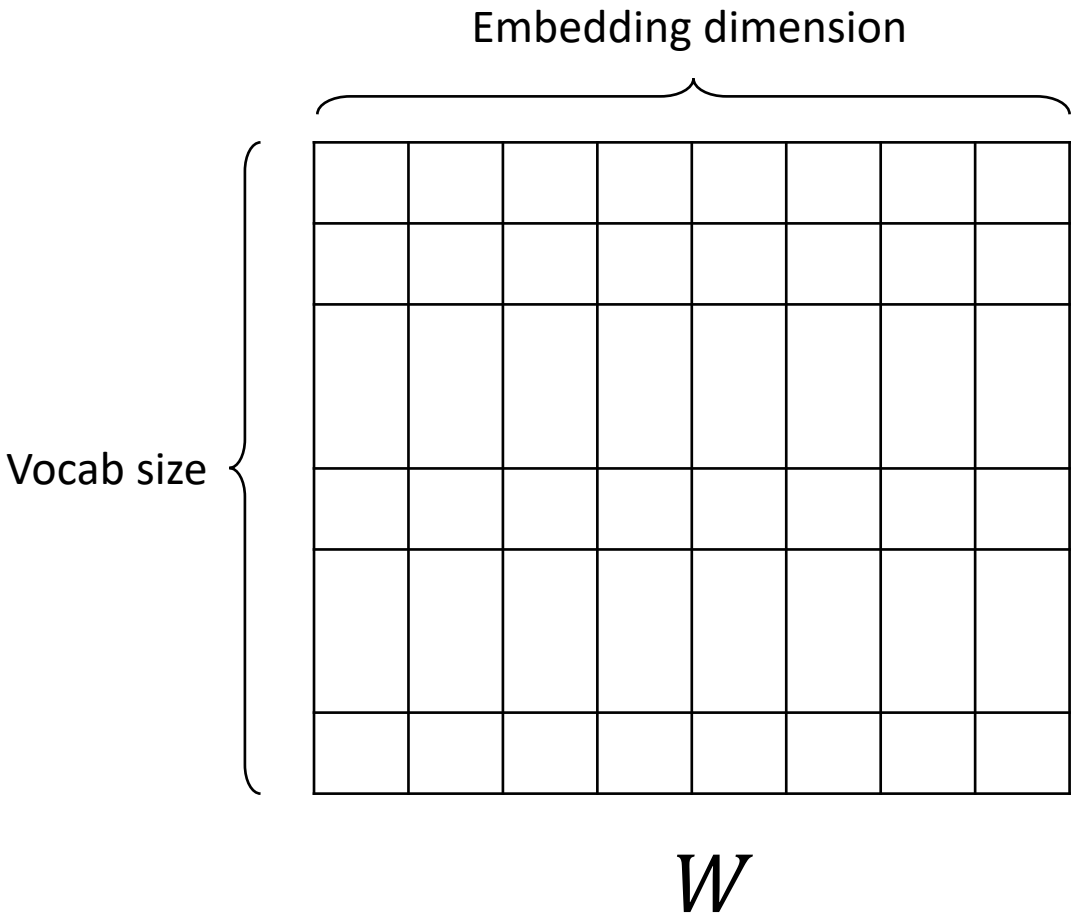
$$[0 \quad 0 \quad 0 \quad \boxed{1} \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ \boxed{10} & \boxed{12} & \boxed{19} \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

x_t W h_t

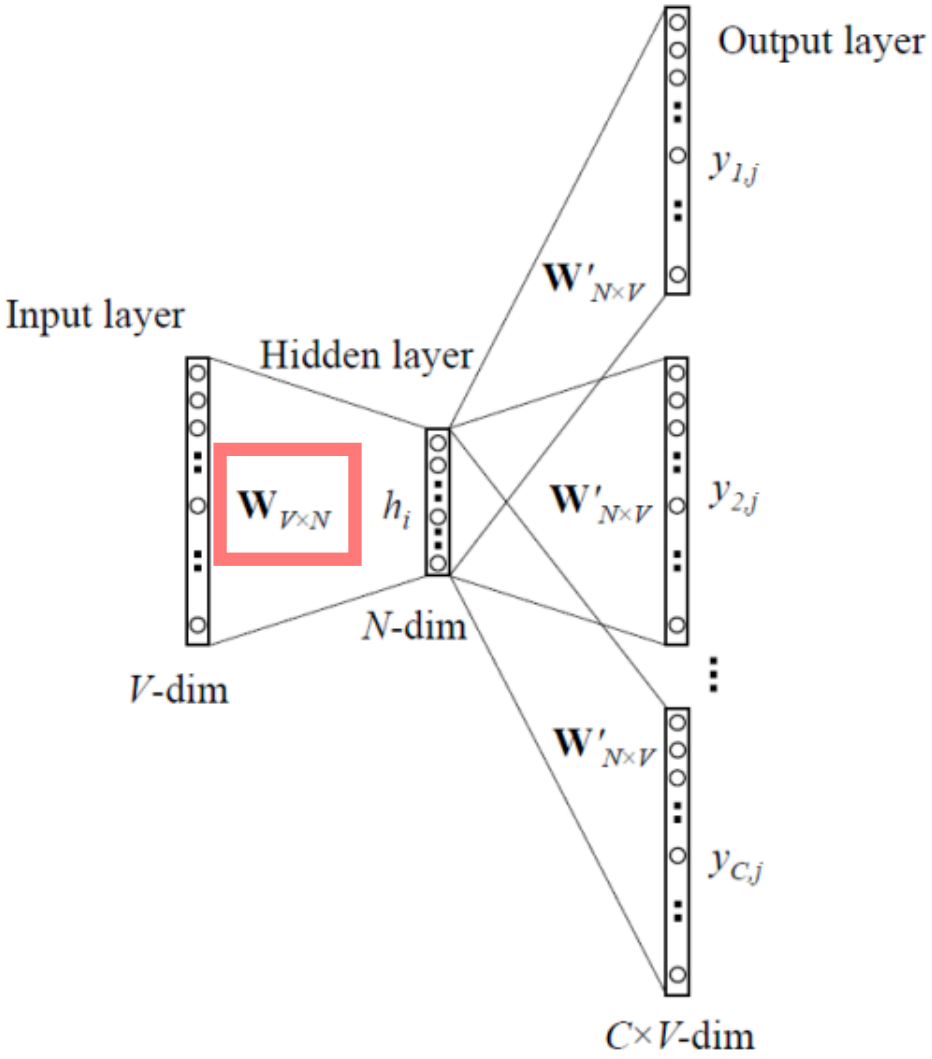
Word2Vec (Skip-Gram)



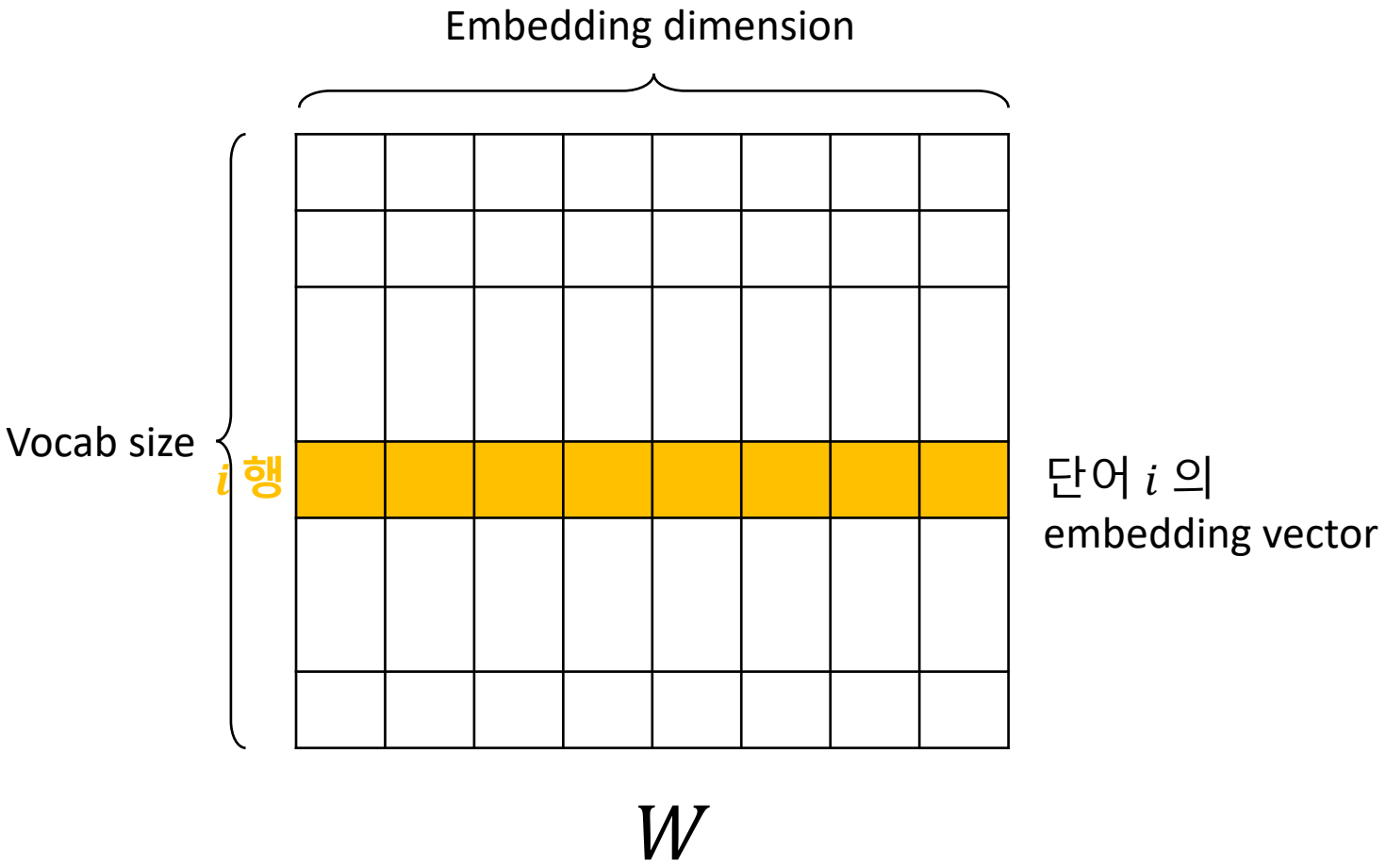
Embedding vector?



Word2Vec (Skip-Gram)



Embedding vector?



Word2Vec

Word embedding visual inspector

https://ronxin.github.io/wevi/

Word2Vec

Word2Vec의 단점

$$\hat{y}_t = softmax(\tilde{y}_t) = \frac{\exp(u_t^T \cdot v_t)}{\sum_{w \in |V|} \exp(u_k v_t)}$$

Word2Vec

Word2Vec의 단점

$$\hat{y}_t = softmax(\tilde{y}_t) = \frac{\exp(u_t^T \cdot v_t)}{\sum_{w \in |V|} \exp(u_k v_t)}$$

계산량이 $|V|$ 에 비례함!

($|V| \approx 10k \sim 1000k$)

Word2Vec

Word2Vec의 단점

$$\hat{y}_t = softmax(\tilde{y}_t) = \frac{\exp(u_t^T \cdot v_t)}{\sum_{w \in |V|} \exp(u_k v_t)}$$

계산량이 $|V|$ 에 비례함!

($|V| \approx 10k \sim 1000k$)

Solutions

- Hierarchical softmax
- Negative Sampling

Word2Vec

Word2Vec의 단점

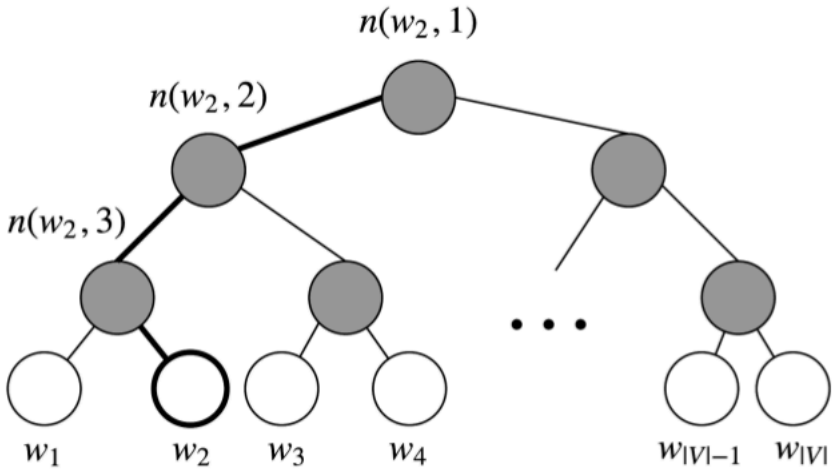
$$\hat{y}_t = softmax(\tilde{y}_t) = \frac{\exp(u_t^T \cdot v_t)}{\sum_{w \in |V|} \exp(u_k v_t)}$$

계산량이 $|V|$ 에 비례함!

($|V| \approx 10k \sim 1000k$)

Solutions

- Hierarchical softmax
- Negative Sampling



$\boldsymbol{O(N) \Rightarrow O(\log N)}$

Word2Vec

Word2Vec의 단점

$$\hat{y}_t = softmax(\tilde{y}_t) = \frac{\exp(u_t^T \cdot v_t)}{\sum_{w \in |V|} \exp(u_k v_t)}$$

계산량이 $|V|$ 에 비례함!

($|V| \approx 10k \sim 1000k$)

Solutions

- Hierarchical softmax
- Negative Sampling

$$\sum_{w \in |V|} \exp(u_k v_t) \Rightarrow \sum_{w \in \{o\} \cap S} \exp(u_k v_t)$$

S: Negative Sample

GloVe

- Jeffrey et al.(2014) - Stanford
- Word2Vec이 통계적인 정보(Statistical Information) 을 잘 이용하지 못한다는 단점을 해결하기 위해 나옴
- Co-Occurrence matrix를 기반으로 word vector 만듬

GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning
Computer Science Department, Stanford University, Stanford, CA 94305
jpennin@stanford.edu, richard@socher.org, manning@stanford.edu

Abstract

Recent methods for learning vector space representations of words have succeeded in capturing fine-grained semantic and syntactic regularities using vector arithmetic, but the origin of these regularities has remained opaque. We analyze and make explicit the model properties needed for such regularities to emerge in word vectors. The result is a new global log-bilinear regression model that combines the advantages of the two major model families in the literature: global matrix factorization and local context window methods. Our model efficiently leverages statistical information by training only on the nonzero elements in a word-word co-occurrence matrix, rather than on the entire sparse matrix or on individual context windows in a large corpus. The model produces a vector space with meaningful substructure, as evidenced by its performance of 75% on a recent word analogy task. It also outperforms related models on similarity tasks and named entity recognition.

1 Introduction

Semantic vector space models of language represent each word with a real-valued vector. These vectors can be used as features in a variety of applications, such as information retrieval (Manning et al., 2008), document classification (Sebastiani, 2002), question answering (Tellex et al., 2003), named entity recognition (Turian et al., 2010), and parsing (Socher et al., 2013).

Most word vector methods rely on the distance or angle between pairs of word vectors as the primary method for evaluating the intrinsic quality of such a set of word representations. Recently, Mikolov et al. (2013c) introduced a new evaluation scheme based on word analogies that probes the finer structure of the word vector space by examining not the scalar distance between word vectors, but rather their various dimensions of difference. For example, the analogy "king is to queen as man is to woman" should be encoded in the vector space by the vector equation *king* − *queen* = *man* − *woman*. This evaluation scheme favors models that produce dimensions of meaning, thereby capturing the multi-clustering idea of distributed representations (Bengio, 2009).

The two main model families for learning word vectors are: 1) global matrix factorization methods, such as latent semantic analysis (LSA) (Deerwester et al., 1990) and 2) local context window methods, such as the skip-gram model of Mikolov et al. (2013c). Currently, both families suffer significant drawbacks. 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.

In this work, we analyze the model properties necessary to produce linear directions of meaning and argue that global log-bilinear regression models are appropriate for doing so. We propose a specific weighted least squares model that trains on global word-word co-occurrence counts and thus makes efficient use of statistics. The model produces a word vector space with meaningful substructure, as evidenced by its state-of-the-art performance of 75% accuracy on the word analogy dataset. We also demonstrate that our methods outperform other current methods on several word similarity tasks, and also on a common named entity recognition (NER) benchmark.

We provide the source code for the model as well as trained word vectors at `http://nlp.stanford.edu/projects/glove/`.

FastText

- Bojanowski et al.(2017)
- Word2Vec을 만든 Mikolov가 연구에 참여
- 대부분의 Idea는 word2vec과 유사
- Sub-word를 이용해 오타, 유사어 모두 의미를 잘 잡아냄.

Enriching Word Vectors with Subword Information

Piotr Bojanowski* and **Edouard Grave*** and **Armand Joulin** and **Tomas Mikolov**
Facebook AI Research
{bojanowski,egrave,ajoulin,tmikolov}@fb.com

Abstract

Continuous word representations, trained on large unlabeled corpora are useful for many natural language processing tasks. Popular models that learn such representations ignore the morphology of words, by assigning a distinct vector to each word. This is a limitation, especially for languages with large vocabularies and many rare words. In this paper, we propose a new approach based on the skipgram model, where each word is represented as a bag of character n-grams. A vector representation is associated to each character n-gram; words being represented as the sum of these representations. Our method is fast, allowing to train models on large corpora quickly and allows us to compute word representations for words that did not appear in the training data. We evaluate our word representations on nine different languages, both on word similarity and analogy tasks. By comparing to recently proposed morphological word representations, we show that our vectors achieve state-of-the-art performance on these tasks.

1 Introduction

Learning continuous representations of words has a long history in natural language processing (Rumelhart et al., 1988). These representations are typically derived from large unlabeled corpora using co-occurrence statistics (Deerwester et al., 1990; Schütze, 1992; Lund and Burgess, 1996). A large body of work, known as distributional semantics, has studied the properties of these methods (Turney et al., 2010; Baroni and Lenci, 2010). In the neural network community, Collobert and Weston (2008) proposed to learn word embeddings using a feedforward neural network, by predicting a word based on the two words on the left and two words on the right. More recently, Mikolov et al. (2013b) proposed simple log-bilinear models to learn continuous representations of words on very large corpora efficiently.

Most of these techniques represent each word of the vocabulary by a distinct vector, without parameter sharing. In particular, they ignore the internal structure of words, which is an important limitation for morphologically rich languages, such as Turkish or Finnish. For example, in French or Spanish, most verbs have more than forty different inflected forms, while the Finnish language has fifteen cases for nouns. These languages contain many word forms that occur rarely (or not at all) in the training corpus, making it difficult to learn good word representations. Because many word formations follow rules, it is possible to improve vector representations for morphologically rich languages by using character level information.

In this paper, we propose to learn representations for character n-grams, and to represent words as the sum of the n-gram vectors. Our main contribution is to introduce an extension of the continuous skipgram model (Mikolov et al., 2013b), which takes into account subword information. We evaluate this model on nine languages exhibiting different morphologies, showing the benefit of our approach.

*The two first authors contributed equally.