Word Embedding

인공지능 연구실 인공지능 세미나
석사 1기 설지우

Contents

[Word Embedding]

정의

- 단어(words)를 실수의 벡터(vectors of real numbers)로 매핑시키는 것
- 매핑된 벡터를 Word Representation이라고도 함

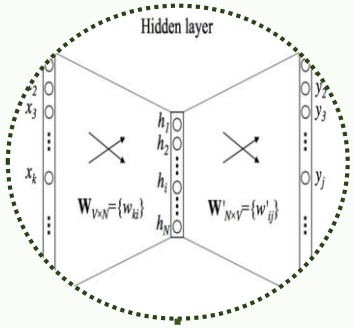
방법론

- Sparse Embedding
- Dense Embedding
- Contextual Embedding

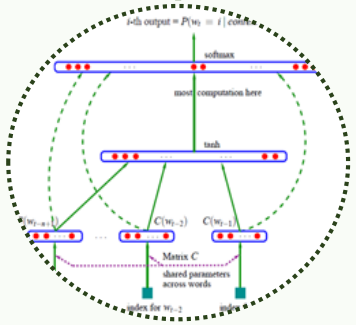
1 Overview

[Word Embedding 발전 흐름]

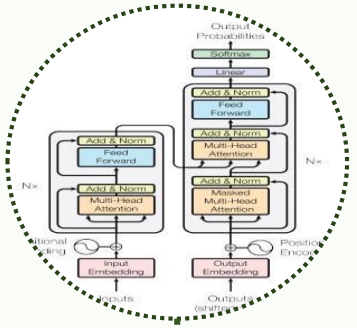
Feedforward Neural Language Model
2003



Word2Vec
2013

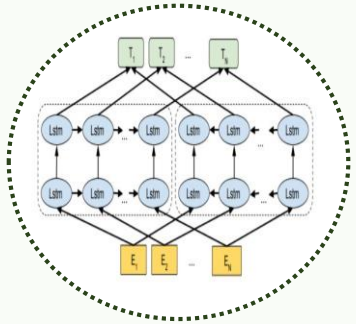


Glove
2014

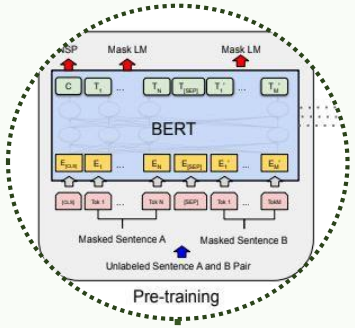


Transformer
2017

ELMo
2018



BERT
2018



2

Sparse Embedding

- One-hot Encoding

2 Sparse Embedding

[One-hot Encoding]

Text Corpus

지우는 마라샹궈보다
마라탕을 좋아해.
…
…

Vocab Dictionary

Vocab	Index
지우는	0
마라샹궈보다	1
마라탕을	2
좋아해	3
…	…

vocab size(k)

One-Hot Encoding

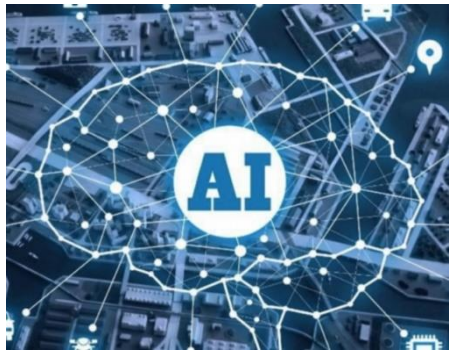
2 Sparse Embedding

[One-hot Encoding]

- 단어 간 관계를 파악할 수 없다 (두 단어 벡터 내적값 = 0)
- 단어 수에 비례하여 차원이 증가함 → 차원수 조정 불가, 지나치게 큰 연산 비용
- 약간의 값 변화도 의미상의 차이가 굉장히 클 수 있음

마라탕

0	0	1	0	0



인공지능

0	0	0	1	0

3

Dense Embedding

- NNLM
- Word2Vec

3 Dense Embedding

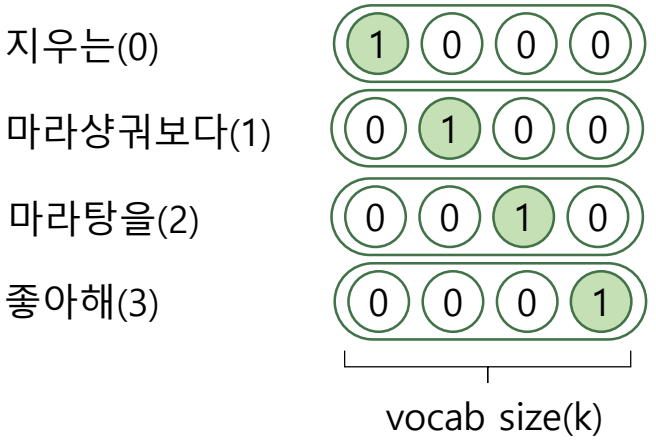
[Distributed Representation]

- **Distributional hypothesis(분포가정)** : 비슷한 의미의 단어는 비슷한 문맥에서 등장하는 경향이 있다.
- 단어의 특성을 보존하고 있는 representation을 특정 저차원 공간 상의 벡터로 표현 (d < k)
- Cosine similarity와 같은 지표로 벡터간 유사성 측정 가능함 → 단어 간 유사도 파악 가능
- 값의 변화에 덜 민감

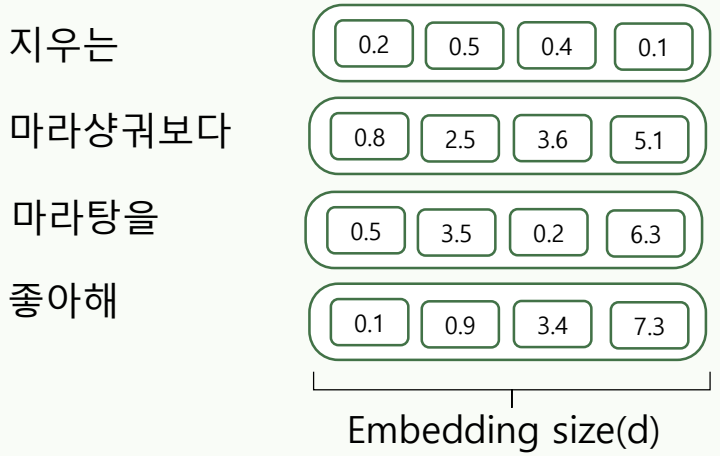
Vocab Dictionary

Vocab	Index
지우는	0
마라샹궈보다	1
마라탕을	2
좋아해	3
…	…

Sparse Embedding



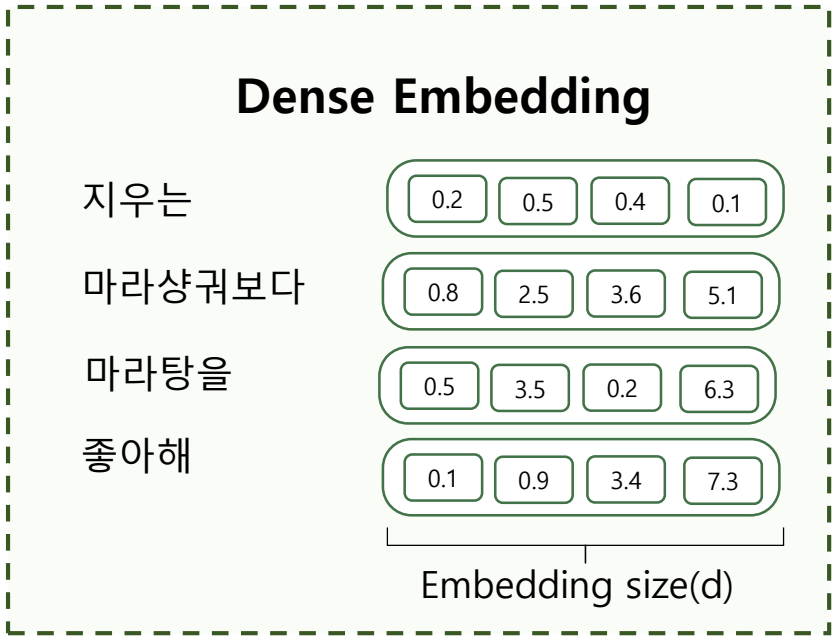
Dense Embedding



3 Dense Embedding

[Distributed Representation]

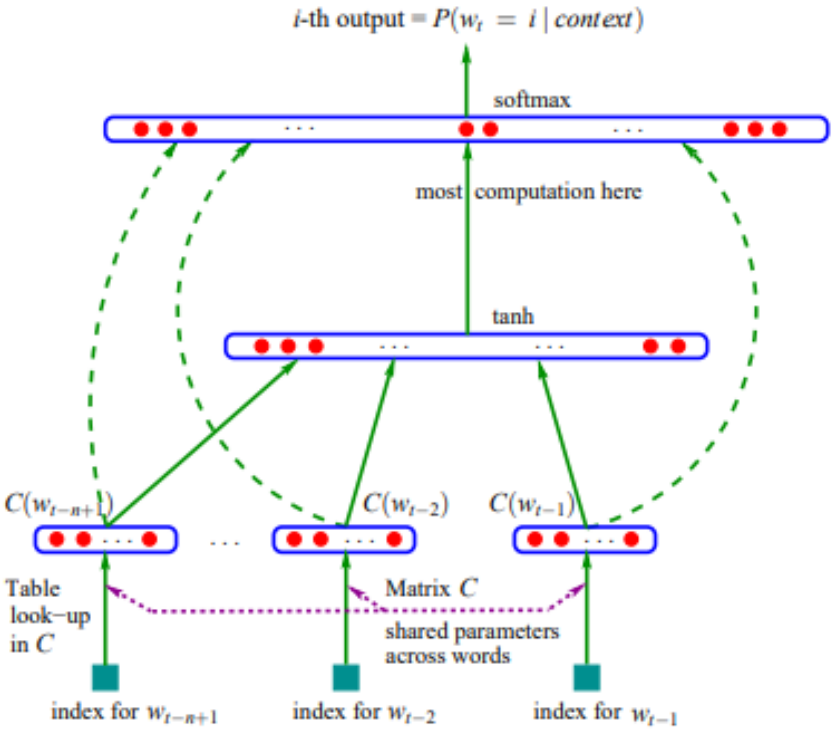
- **문맥을 잘 반영하고 있는 임베딩 벡터(Distributed Representation) 생성이 핵심!**



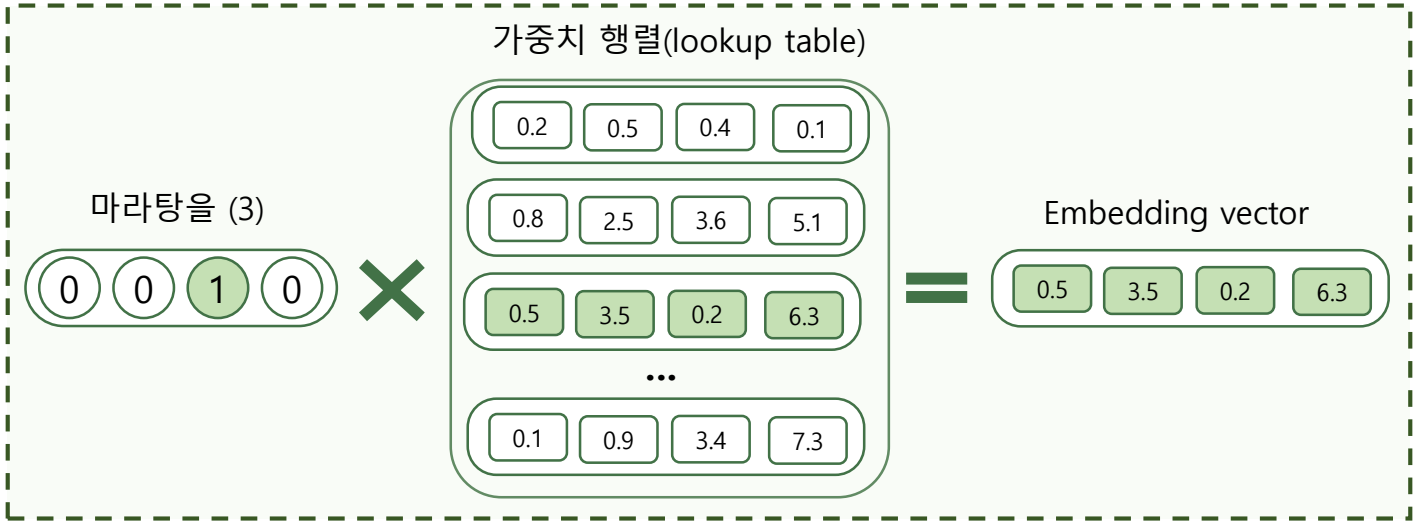
3 Dense Embedding

[NNLM]

A neural probabilistic language model (Bengio et al., 2003)



- 미리 지정한 window size 내 단어를 통해 그 다음에 위치할 단어를 예측
- Vocab 내 각 단어의 embedded feature(distributed representation)을 학습함

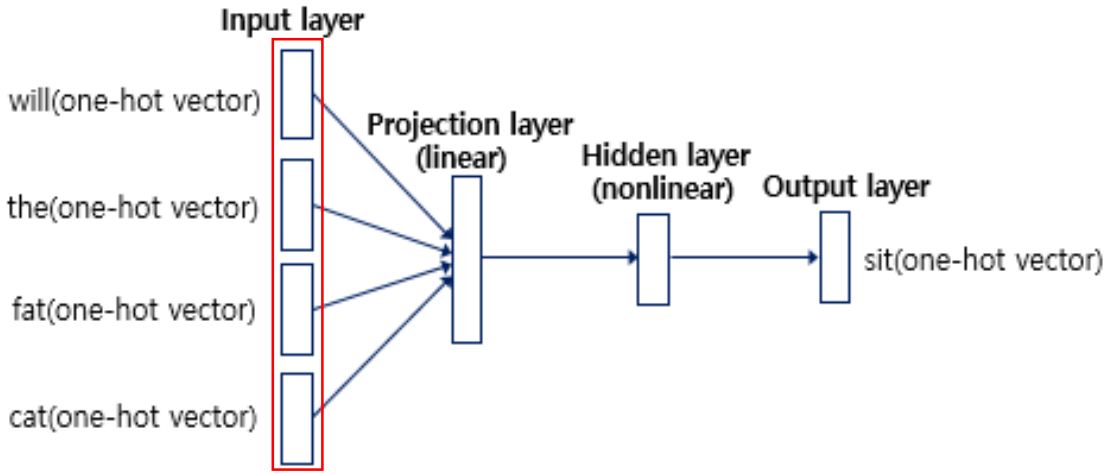


[NNLM]

A neural probabilistic language model (Bengio et al., 2003)

sit을 예측할 예정!

- e.g.) “what will the fat cat sit on mat”, window size=4



4개의 word에 대한 one-hot encoding을 input으로 받음

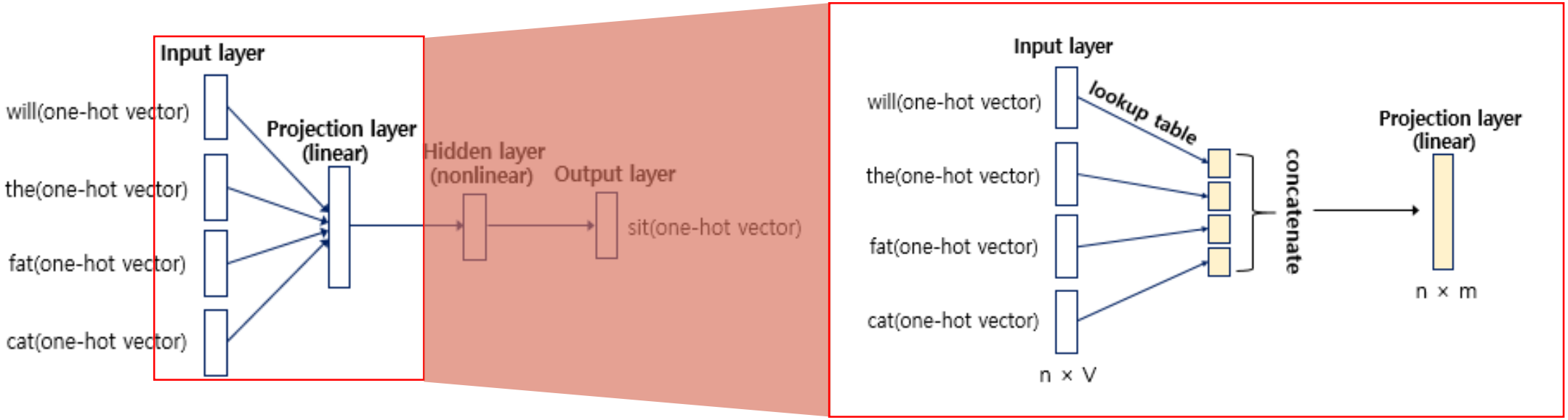
```
what = [1, 0, 0, 0, 0, 0, 0]
will = [0, 1, 0, 0, 0, 0, 0]
the = [0, 0, 1, 0, 0, 0, 0]
fat = [0, 0, 0, 1, 0, 0, 0]
cat = [0, 0, 0, 0, 1, 0, 0]
sit = [0, 0, 0, 0, 0, 1, 0]
on = [0, 0, 0, 0, 0, 0, 1]
```

[NNLM]

A neural probabilistic language model (Bengio et al., 2003)

sit을 예측할 예정!

- e.g.) “what will the fat cat sit on mat”, window size=4



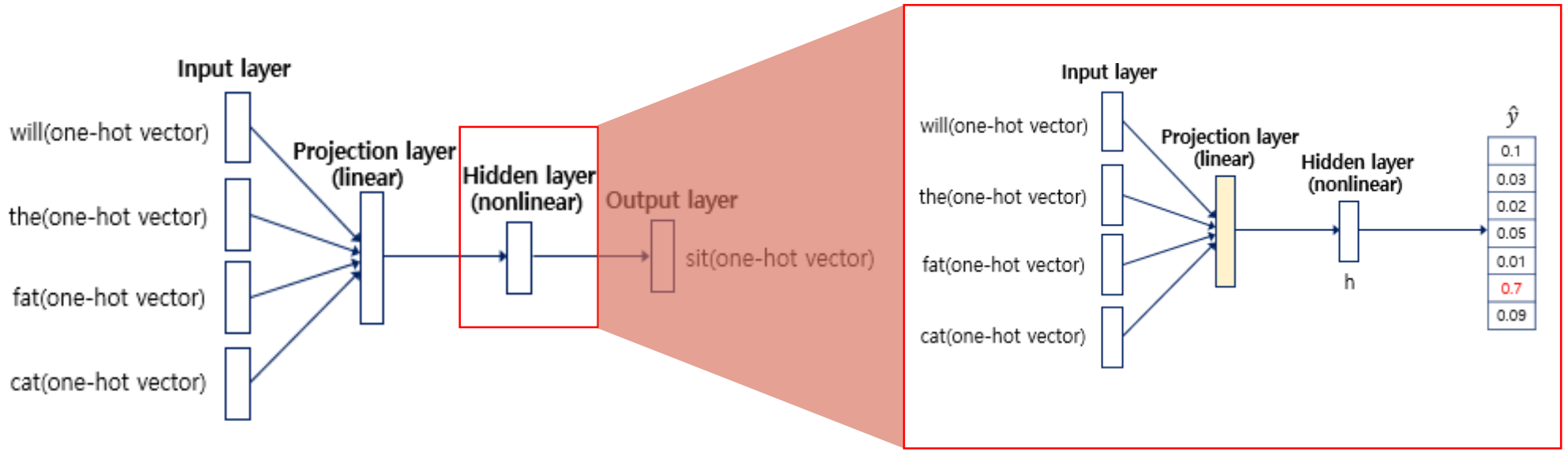
$$p^{layer} = (lookup(x_{t-n}); \ldots; lookup(x_{t-2}); lookup(x_{t-1})) = (e_{t-n}; \ldots; e_{t-2}; e_{t-1})$$

[NNLM]

A neural probabilistic language model (Bengio et al., 2003)

sit을 예측할 예정!

- e.g.) “what will the fat cat sit on mat”, window size=4



$$h^{layer} = tanh(W_h p^{layer} + b_h)$$

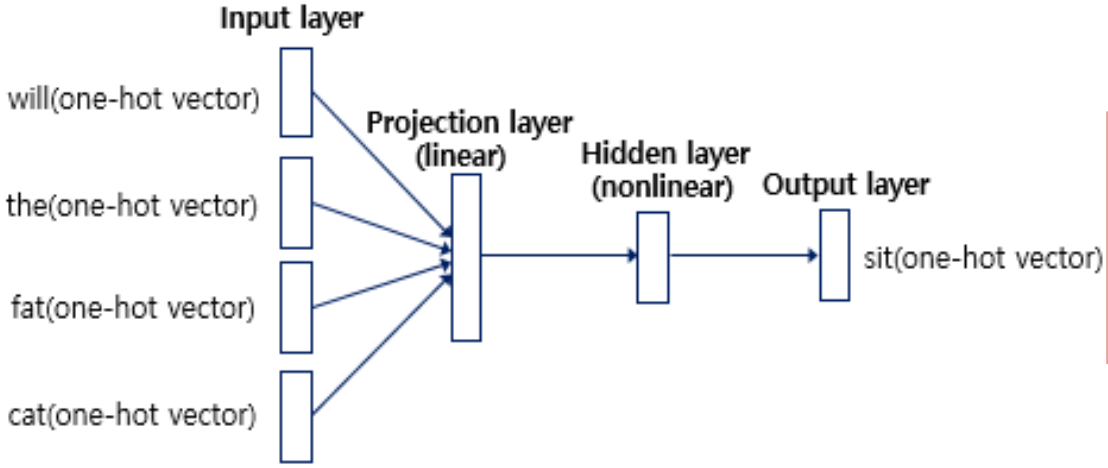
$$\hat{y} = softmax(W_y h^{layer} + b_y)$$

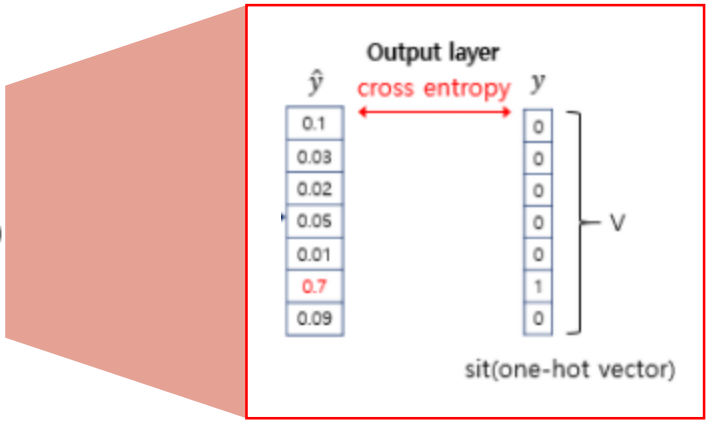
[NNLM]

A neural probabilistic language model (Bengio et al., 2003)

sit을 예측할 예정!

- e.g.) “what will the fat cat sit on mat”, window size=4





$Lost\ function:\ cross-entropy$

3 Dense Embedding

```python
# Model
class NNLM(nn.Module):
    def __init__(self):
        super(NNLM, self).__init__()
        # nn.Embedding(num_embeddings, embedding_dim)
        self.C = nn.Embedding(n_class, m)

        # nn.Linear(input_dim, output_dim, bias=T or F(bias 학습 X))
        self.H = nn.Linear(n_step*m, n_hidden, bias=False)
        self.d = nn.Parameter(torch.ones(n_hidden))

        self.U = nn.Linear(n_hidden, n_class, bias=False)
        self.W = nn.Linear(n_step*m, n_class, bias=False)
        self.b = nn.Parameter(torch.ones(n_class))

    def forward(self, X):
        # 입력 : [n_step, m]
        # Embedding layer
        X = self.C(X)   # [batch_size, n_step, m]

        # Projection Layer
        X = X.view(-1, n_step*m)    # [batch_size, n_step*m]

        # Hidden Layer
        tanh = torch.tanh(self.d + self.H(X))   # [batch_size, n_hidden]

        # Output Layer
        output = self.b + self.W(X) + self.U(tanh)  # [batch_size, n_class]

        return output
```

```python
# Training
for epoch in range(5000):
    optimizer.zero_grad()
    output = model(input_batch)     # 순전파

    # output : [batch_size, n_class]
    # target : [batch_size]
    loss = criterion(output, target_batch)  # 손실함수
    if (epoch+1)%1000==0:
        print('Epoch:', '%04d' % (epoch + 1), 'cost =', '{:.6f}'.format(loss))

    loss.backward()     # 역전파
    optimizer.step()    # learning rate만큼 이동
```

✓ 2.7s

```
Epoch: 1000 cost = 0.000024
Epoch: 2000 cost = 0.000013
Epoch: 3000 cost = 0.000007
Epoch: 4000 cost = 0.000004
Epoch: 5000 cost = 0.000002
```

[NNLM]

A neural probabilistic language model (Bengio et al., 2003)

[한계점]

- 고정된 window size 내의 앞의 시점 단어들만 고려하기 때문에 다음 시점의 단어들을 고려하지 않음
- Complexity의 대부분이 model의 non-linear hidden layer로부터 야기된다.

[NNLM]

A neural probabilistic language model (Bengio et al., 2003)

training complexity

$$O = E \times T \times Q$$

- E : the number of training epochs (30~50)
- T : the number of words in the training set (최대 10억개)
- Q : each model architecture에 대해 추가로 정의됨!

NNLM training complexity

$$Q = N \times D + N \times D \times H + H \times V$$

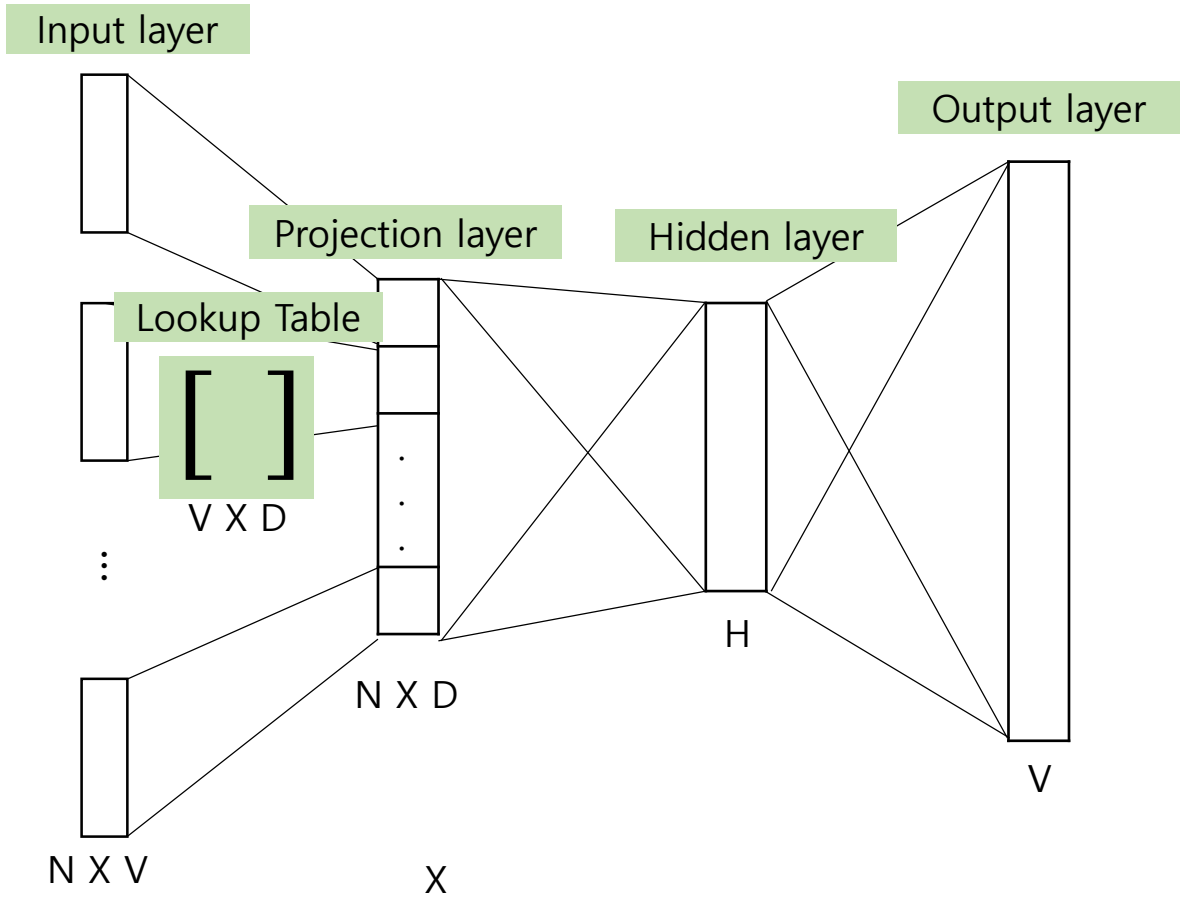
- N : the number of input words
- D : embedded size
- V : size of the vocabulary
- H : hidden layer size

3 Dense Embedding

Most of the complexity is caused by here!

$$Q = N \times D + N \times D \times H + H \times V$$

- N : the number of input words
- D : embedded size
- V : size of the vocabulary
- H : hidden layer size



[Word2Vec]

Efficient estimation of word representations in vector space (Mikolov et al., 2013)

- 대량의 data로부터 Complexity를 최소화하고 고품질의 distributed representation(embedded feature)를 학습하는 기술을 제안
- Shallow Neural Network : Input Layer – Projection Layer(hidden layer) – Output Layer
- Word2Vec의 2가지 모델을 제시
  - **CBOW** (Continuous Bag-of-Words)
  - **Skip-gram** (Continuous Skip-gram)

[Word2Vec]

Efficient estimation of word representations in vector space (Mikolov et al., 2013)

[CBOW]

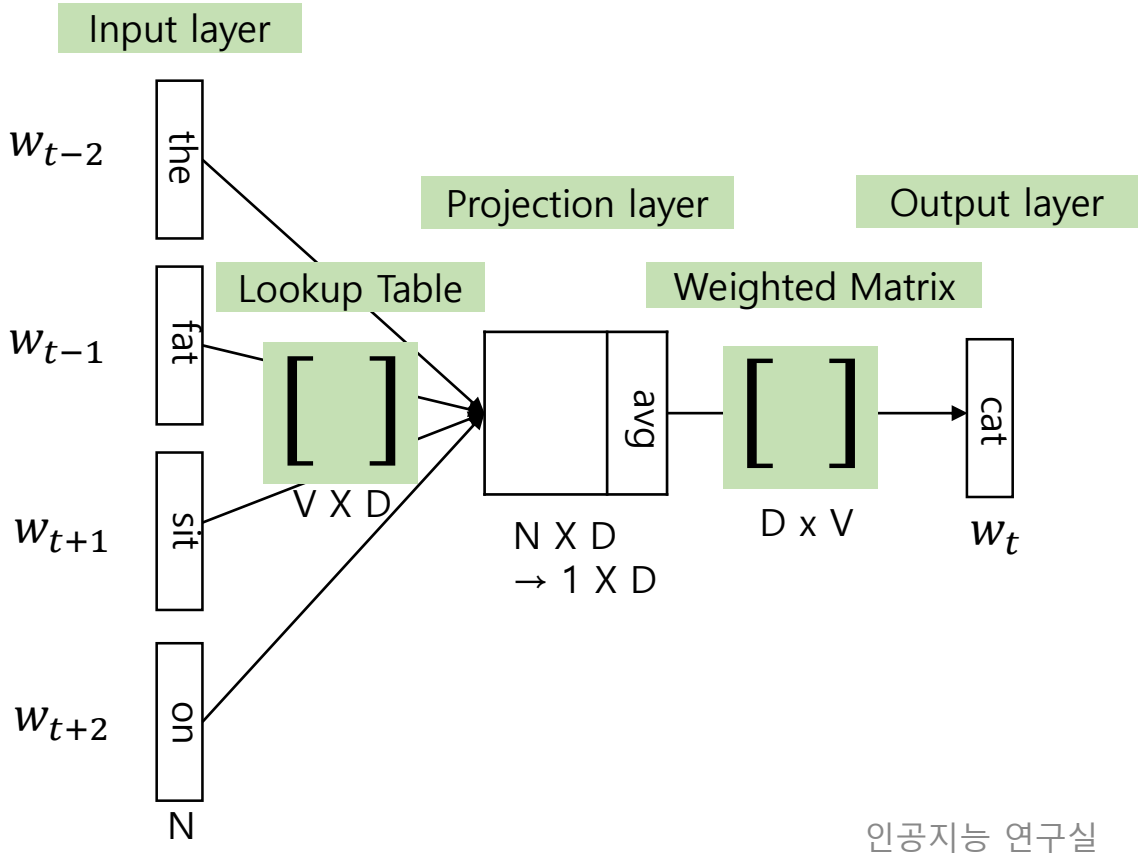
- 주변 context 단어들로부터 하나의 target 단어를 예측하는 모델
- NNLM과 비슷한 구조이지만 non-linear hidden layer가 제거됨 & 모든 단어가 같은 position으로 projected됨!
- Bag-of-words 모델과 동일하게 단어의 순서를 고려하지 않기 때문에 CBOW라고 부름
  - 단어의 순서를 고려하지 않는 이유는 Projection Layer에서 input word에 대한 embedded feature를 평균을 냄

[Word2Vec]

Efficient estimation of word representations in vector space (Mikolov et al., 2013)

[CBOW]

- Training example) "what will the fat cat sit on mat."



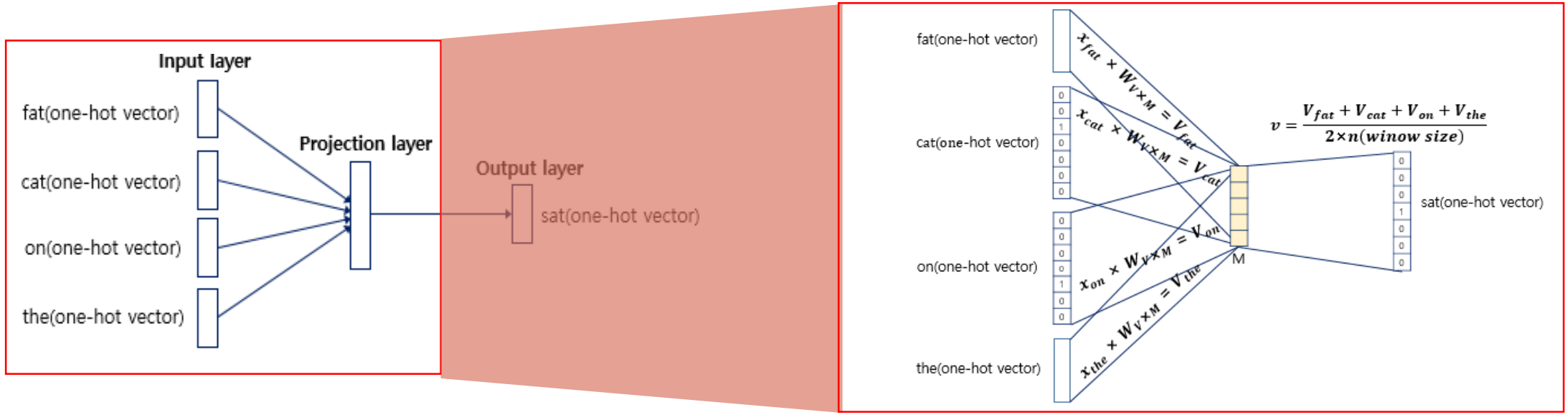
- N : the number of input words
- D : embedded size
- V : size of the vocabulary

[Word2Vec]

Efficient estimation of word representations in vector space (Mikolov et al., 2013)

[CBOW]

- Training example) "what will the fat cat sit on mat.", window size = 2

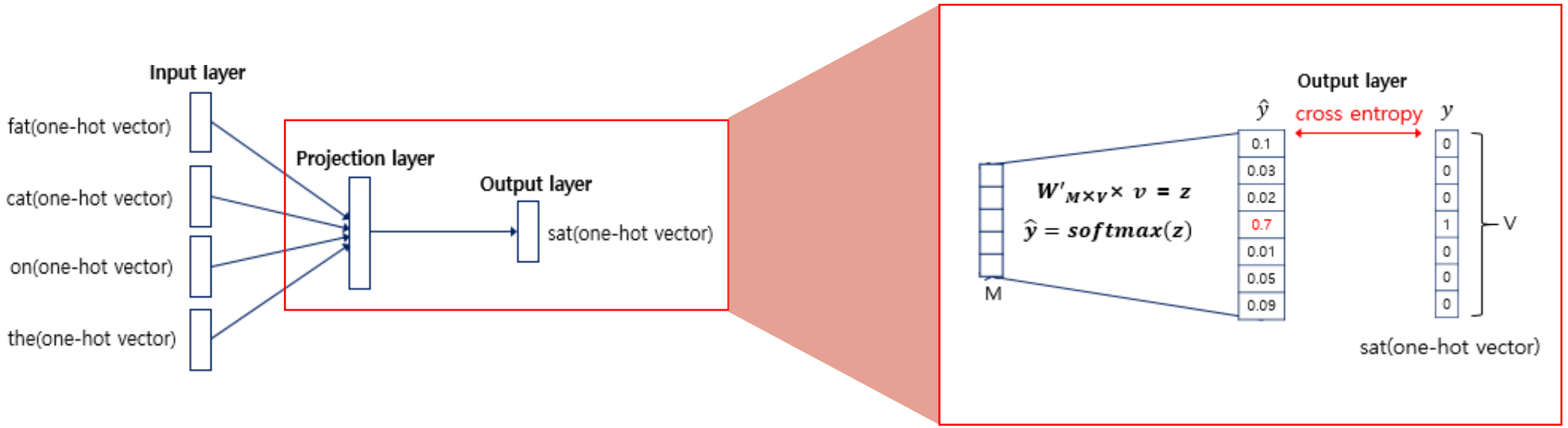


[Word2Vec]

Efficient estimation of word representations in vector space (Mikolov et al., 2013)

[CBOW]

- Training example) "what will the fat cat sit on mat.", window size = 2



[Word2Vec]

Efficient estimation of word representations in vector space (Mikolov et al., 2013)

[Skip-gram]

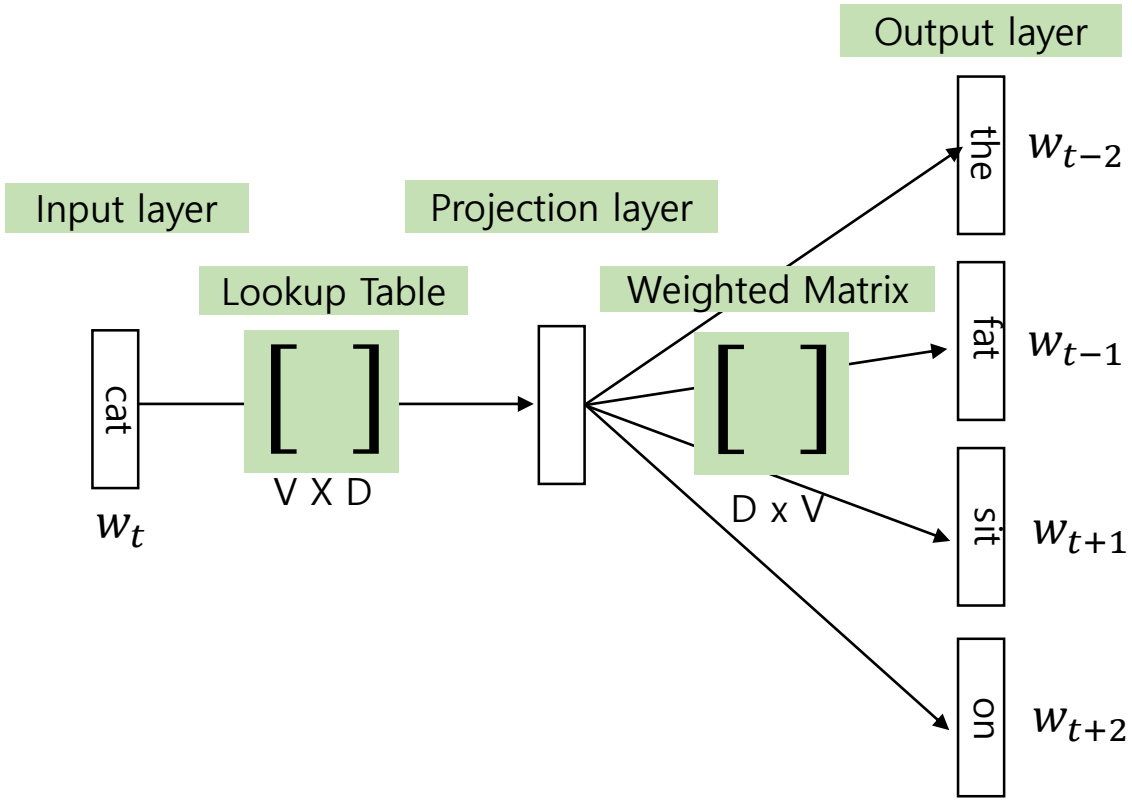
- 하나의 target 단어로부터 주변 context 단어들을 예측하는 모델
- 중심 target 단어가 입력으로 사용되며, window size만큼 전후 단어들 즉 총 2*window_size개의 context 단어를 예측함
- 본 논문 외에도 Skip-gram을 효율적으로 학습시키고 성능을 높이는 학습 방법들이 제시됨
  - Distributed Representations of Words and Phrases and their Compositionality (Mikolov et al., 2013)

[Word2Vec]

Efficient estimation of word representations in vector space (Mikolov et al., 2013)

[Skip-gram]

- Training example) "what will the fat cat sit on mat.", window size = 2



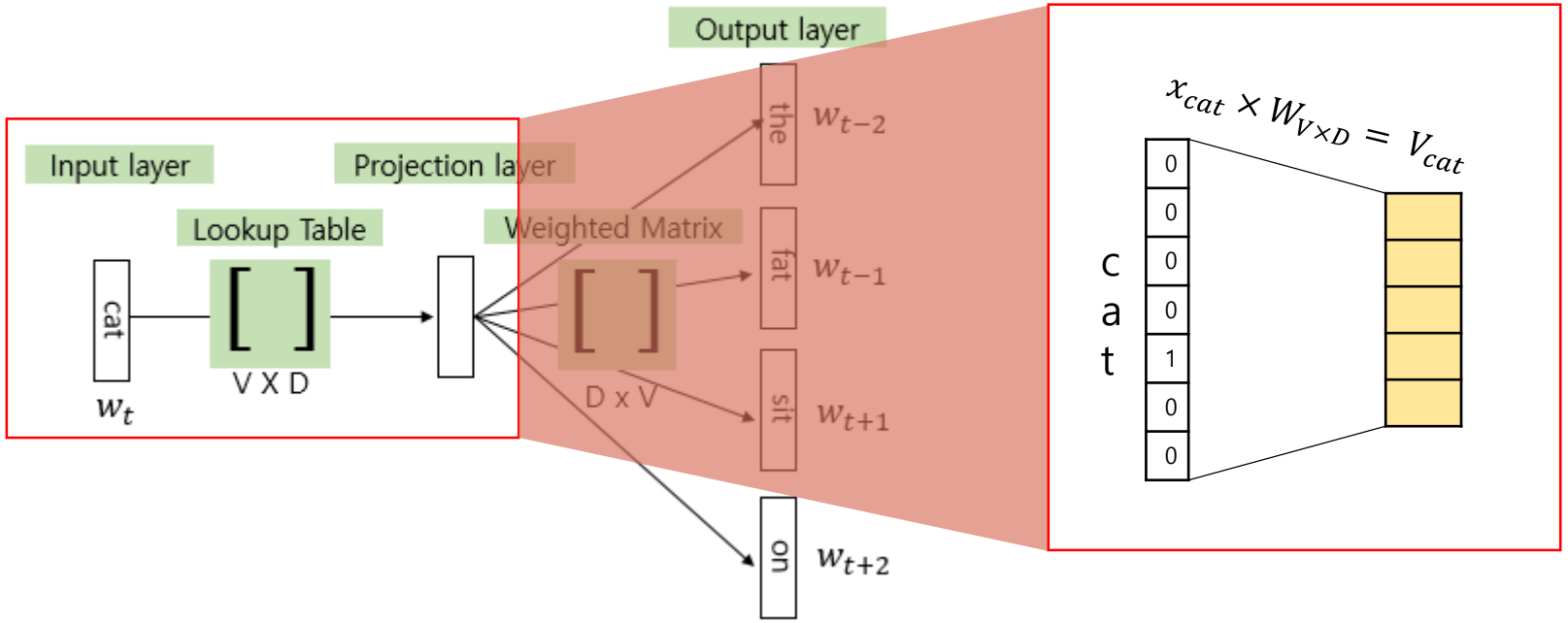
- N : the number of input words
- D : embedded size
- V : size of the vocabulary

[Word2Vec]

Efficient estimation of word representations in vector space (Mikolov et al., 2013)

[Skip-gram]

- Training example) "what will the fat cat sit on mat."

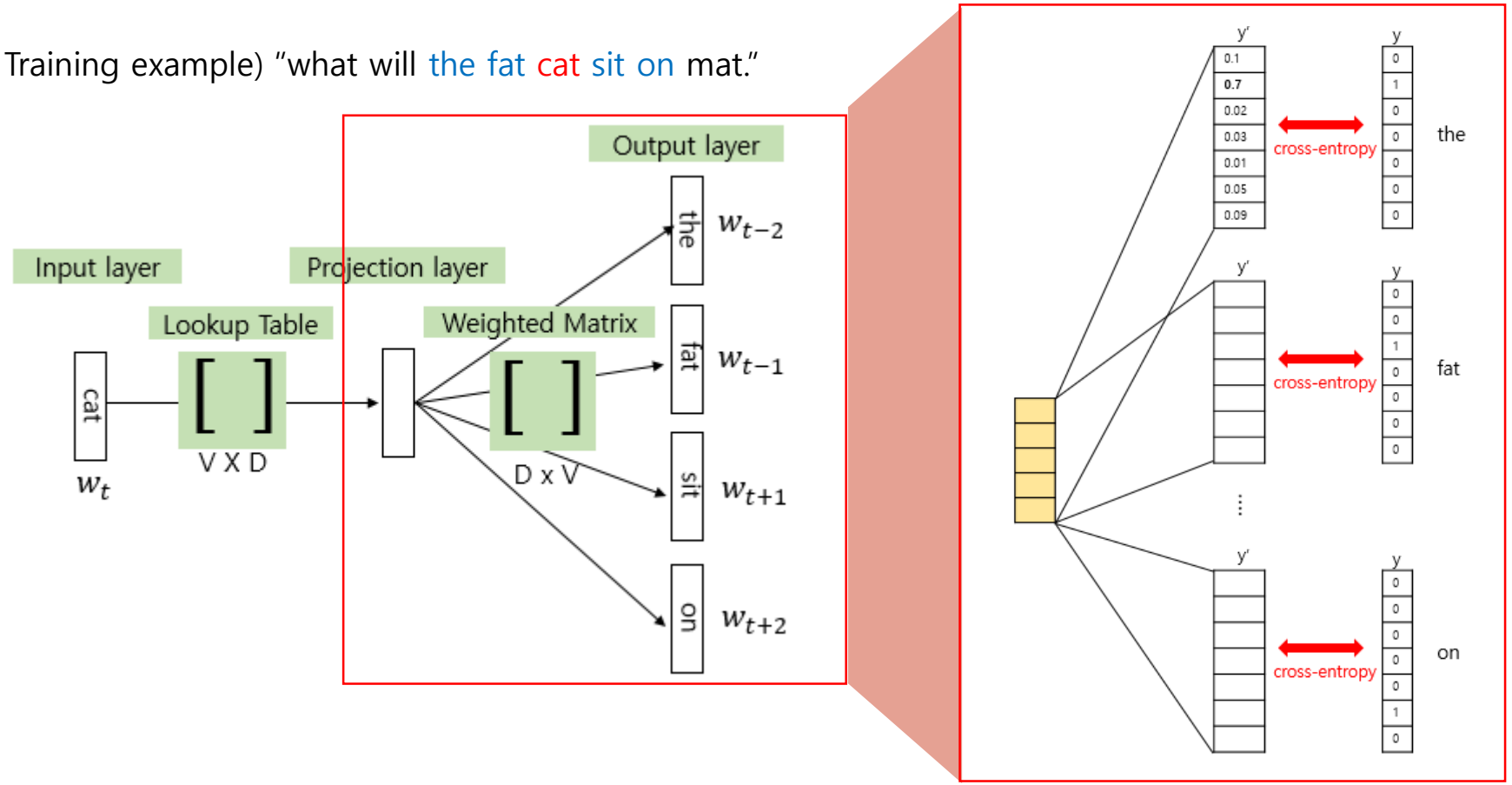


[Word2Vec]

Efficient estimation of word representations in vector space (Mikolov et al., 2013)

[Skip-gram]

- Training example) "what will the fat cat sit on mat."



4 Summary

[Training Complexity : NNLM vs Word2Vec]

	NNLM	Word2Vec
Complexity	$Q = N \times D + N \times D \times H + H \times V$	**[CBOW] :** $Q = N \times D + D \times V$ **[Skip-gram] :** $Q = C \times (D + D \times V)$
Architecture	Input layer – Projection layer – Hidden Layer – Output layer	Input layer – Projection layer – Output layer

- C : Maximum distance of the words
- N : the number of input words
- D : embedded size
- V : size of the vocabulary
- H : hidden layer size

4 Summary

[Semantic-Syntactic Word Relationship test set]

Table 1: *Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set.*

	Type of relationship	Word Pair 1		Word Pair 2	
semantic	Common capital city	Athens	Greece	Oslo	Norway
	All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
	Currency	Angola	kwanza	Iran	rial
	City-in-state	Chicago	Illinois	Stockton	California
	Man-Woman	brother	sister	grandson	granddaughter
syntactic	Adjective to adverb	apparent	apparently	rapid	rapidly
	Opposite	possibly	impossibly	ethical	unethical
	Comparative	great	greater	tough	tougher
	Superlative	easy	easiest	lucky	luckiest
	Present Participle	think	thinking	read	reading
	Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
	Past tense	walking	walked	swimming	swam
	Plural nouns	mouse	mice	dollar	dollars
	Plural verbs	work	works	speak	speaks

1) $X = vector(Athens) - vector(Greece) + vector(Oslo)$
2) cosine distance를 통해 vector space상에서 X와 가까운 word 구함
3) word가 *Norway* 와 같다면 정답!

4 Summary

[NNLM vs Word2Vec]

Training Dataset
- Google News (6B=60억)

Test Dataset
- Semantic-Syntactic Word Relationship test set (new!)
- 8869 semantic questions & 10675 syntactic questions

Evaluation

Table 6: *Comparison of models trained using the DistBelief distributed framework. Note that training of NNLM with 1000-dimensional vectors would take too long to complete.*

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days x CPU cores]
			Semantic	Syntactic	Total	
NNLM	100	6B	34.2	64.5	50.8	14 x 180
CBOW	1000	6B	57.3	68.9	63.7	2 x 140
Skip-gram	1000	6B	66.1	65.1	65.6	2.5 x 125

4 Summary

[Conclusion]

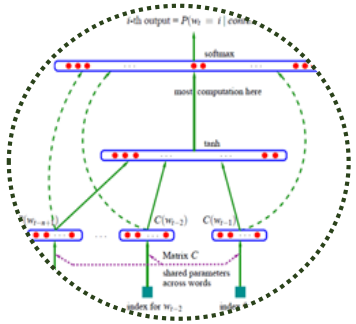
- 간단한 모델 구조를 사용해 NNLM에 비교해서 높은 quality의 word vector를 학습할 수 있음
- 새로운 Syntactic, Semantic task를 제시해 word vector가 다양한 의미들에 대한 여러 similarity를 반영하는가를 평가함
- 이전 모델보다 낮은 complexity 덕분에 훨씬 많은 dataset으로부터 고차원의 word vector를 계산하는 것이 가능

[개인적인 생각]

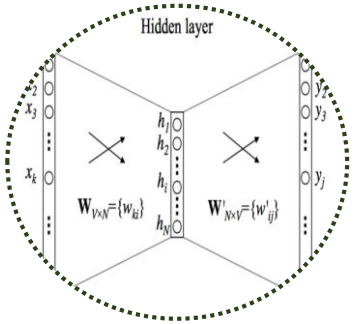
- 본 논문의 저자가 Sec3 부분에서는 상당히 불친절하게 작성하였다고 생각함 → 모델 이해의 어려움..!

[Dense Embedding 방법론의 한계]

Feedforward Neural Language Model
2003



Word2Vec
2013



고정된 window size 내에 위치하지 않은 단어들을 고려하지 못함
문맥에 따른 단어의 의미 차이를 고려하지 못함 (e.g. 배가 고프다, 배가 뜬다)

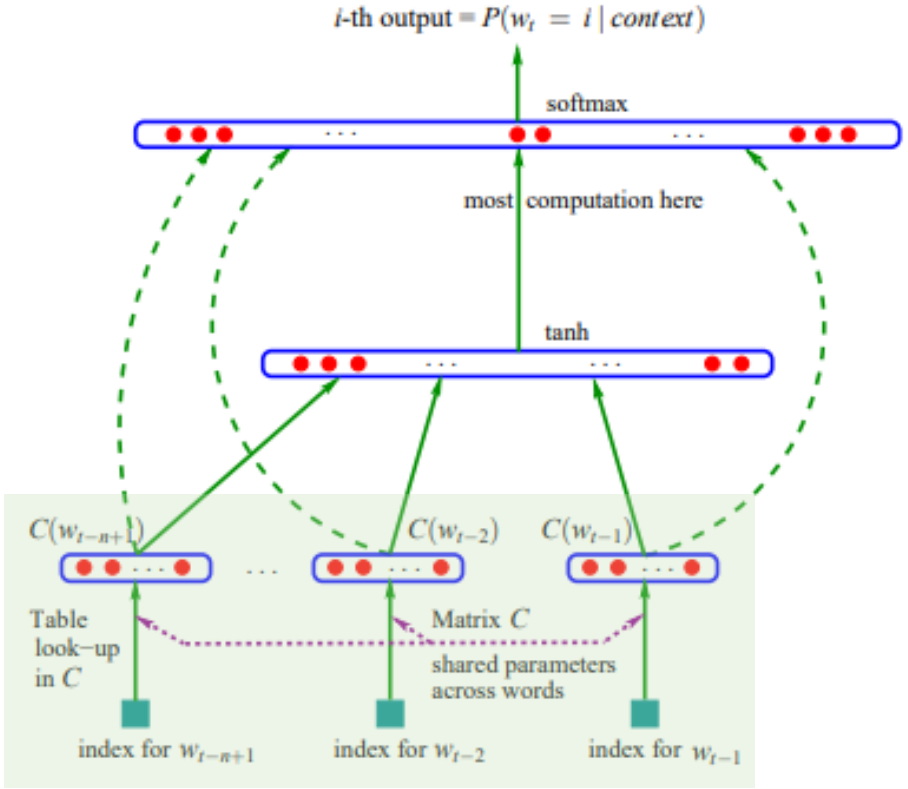
END

[NNLM]

A neural probabilistic language model (Bengio et al., 2003)

sit을 예측할 예정!

- e.g.) “what will the fat cat sit on”, window size=4



4개의 word에 대한 one-hot encoding을 input으로 받음

```
what = [1, 0, 0, 0, 0, 0, 0]
will = [0, 1, 0, 0, 0, 0, 0]
the = [0, 0, 1, 0, 0, 0, 0]
fat = [0, 0, 0, 1, 0, 0, 0]
cat = [0, 0, 0, 0, 1, 0, 0]
sit = [0, 0, 0, 0, 0, 1, 0]
on = [0, 0, 0, 0, 0, 0, 1]
```

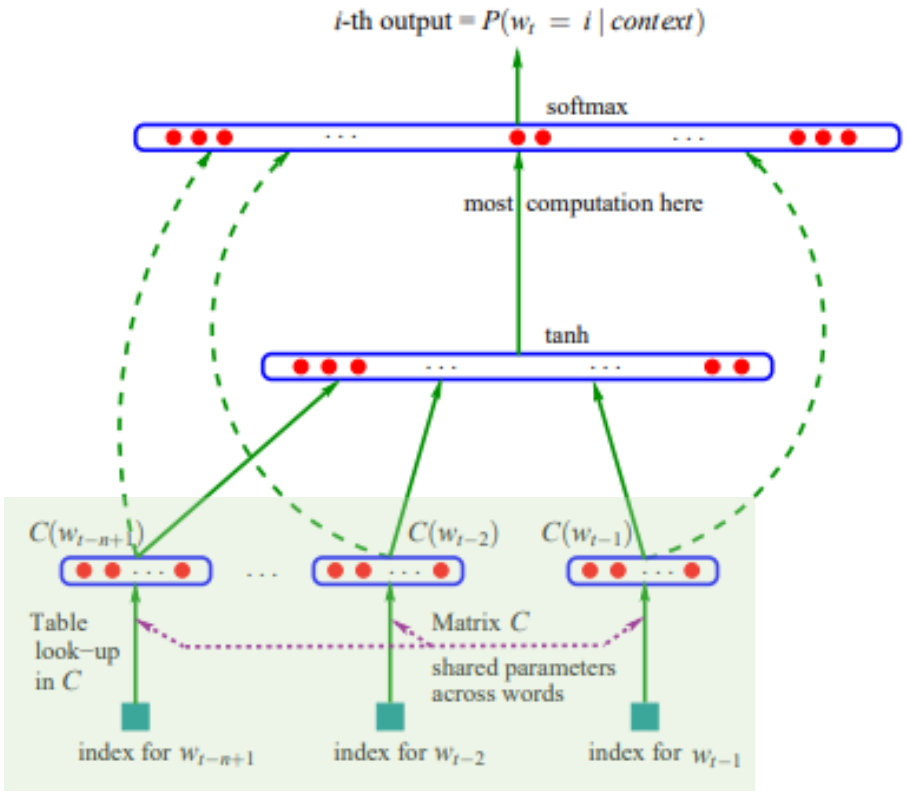
Vocab size = 7
Embed size = 3
Hidden size = 6

[NNLM]

A neural probabilistic language model (Bengio et al., 2003)

sit을 예측할 예정!

- e.g.) “what will the fat cat sit on”, window size=4



Lookup Table (Shared Parameter)

INDEX	F1	F2	F3
0	1.11	1.00	0.89
1	0.12	-0.98	2.54
2	2.10	0.22	1.56
3	1.28	0.69	1.57
4	-1.92	2.22	1.09
5	1.82	0.93	1.91
6	2.11	1.91	3.10

0.12	-0.98	2.54

$= x_i$

0	1	0	0	0	0	0

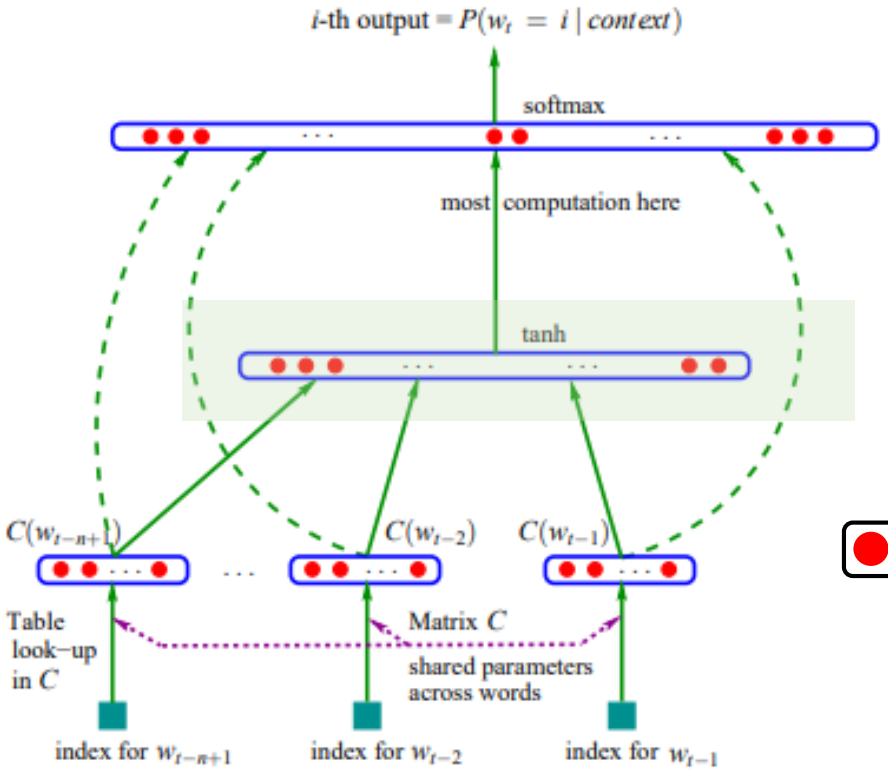
Vocab size = 7
Embed size = 3
Hidden size = 6

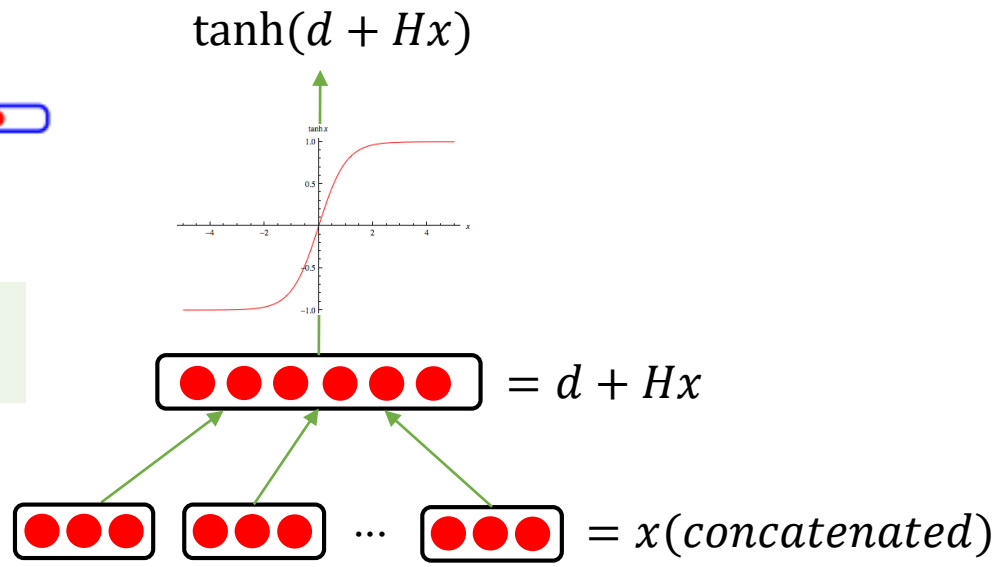
[NNLM]

A neural probabilistic language model (Bengio et al., 2003)

sit을 예측할 예정!

- e.g.) “what will the fat cat sit on”, window size=4





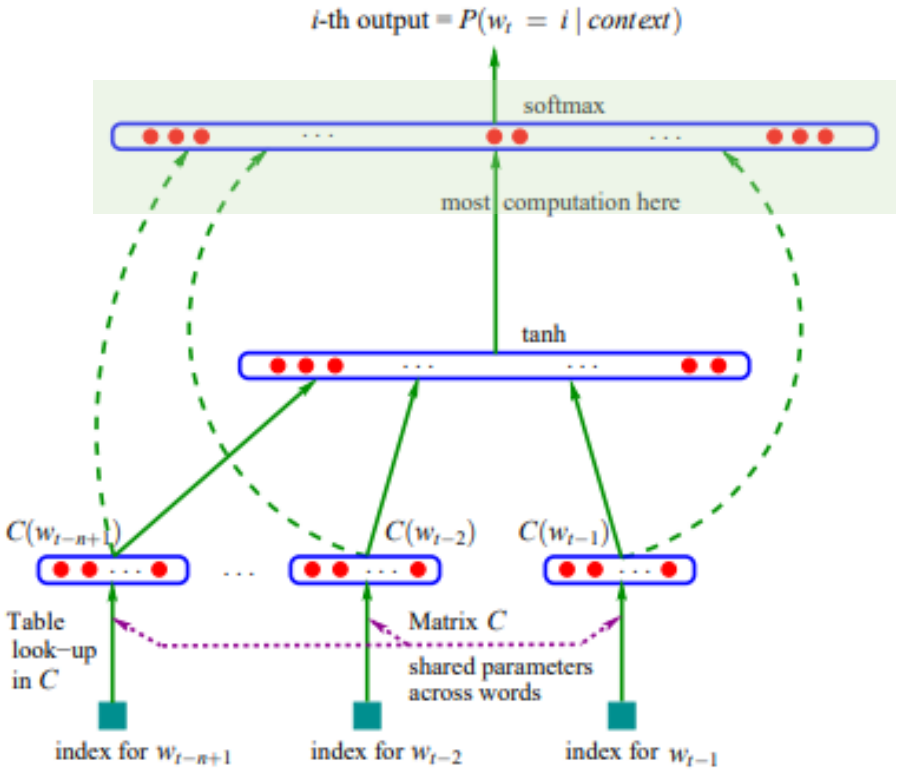
Vocab size = 7
Embed size = 3
Hidden size = 6

[NNLM]

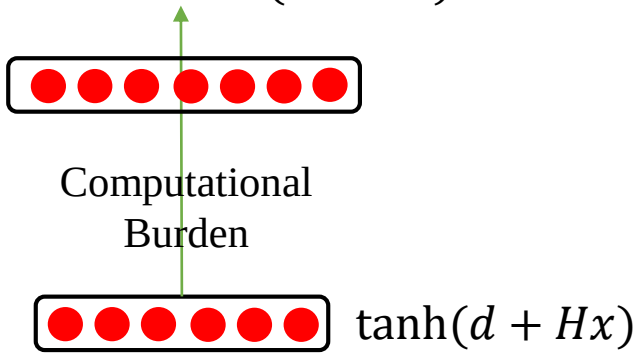
A neural probabilistic language model (Bengio et al., 2003)

sit을 예측할 예정!

- e.g.) “what will the fat cat sit on”, window size=4



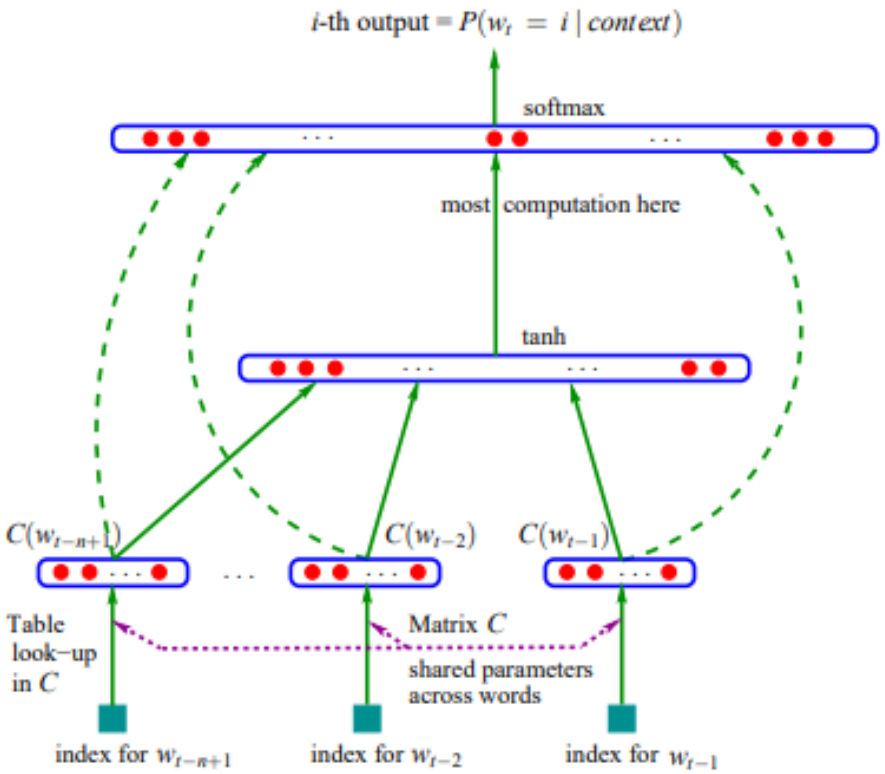
$$y = b + U \cdot \tanh(d + Hx)$$



Vocab size = 7
Embed size = 3
Hidden size = 6

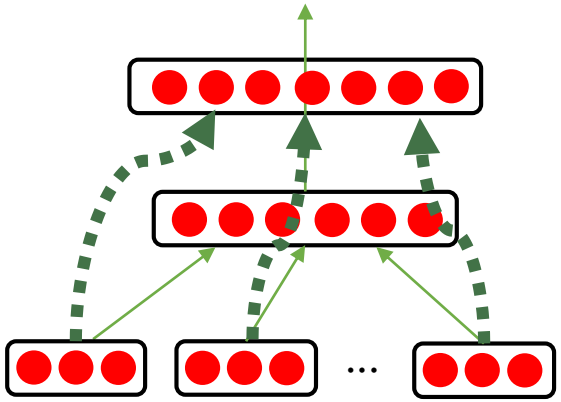
[NNLM]

A neural probabilistic language model (Bengio et al., 2003)



[optional]

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



Vocab size = 7
Embed size = 3
Hidden size = 6

[NNLM]

A neural probabilistic language model (Bengio et al., 2003)

Parameters (θ)

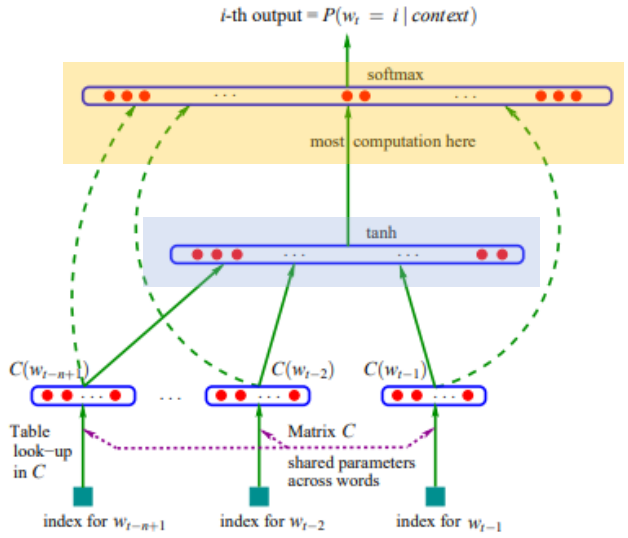
1. Lookup Table

Lookup Table (Shared Parameter)

INDEX	*m*		
|V|	1.11	1.00	0.89
	0.12	-0.98	2.54
	2.10	0.22	1.56
	1.28	0.69	1.57
	-1.92	2.22	1.09
	1.82	0.93	1.91
	2.11	1.91	3.10

2. Network Parameters

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



[Semantic relationships]

Word vector에 대한 간단한 대수 연산(algebraic operations)으로 단어 간 복잡한 유사도 질문에 답할 수 있었음

Q) "What is the word that is similar to small in the same sense as biggest is similar to big?"

A) $X = vector(biggest) - vector(big) + vector(small)$
$cosine\ distance$로 X와 가장 가까운 단어를 vector space에서 찾음 → X is smallest.

[Semantic and Syntactic Evaluation of word vectors]

Table 1: *Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set.*

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Parameters
- 640 embedded dim

Training Dataset
- Google News corpus (6B)

Test Dataset
- Semantic-Syntactic Word Relationship test set (new!)
- 8869 semantic questions & 10675 syntactic questions

Evaluation

Table 3: *Comparison of architectures using models trained on the same data, with 640-dimensional word vectors. The accuracies are reported on our Semantic-Syntactic Word Relationship test set, and on the syntactic relationship test set of [20]*

Model Architecture	Semantic-Syntactic Word Relationship test set		MSR Word Relatedness
	Semantic Accuracy [%]	Syntactic Accuracy [%]	Test Set [20]
RNNLM	9	36	35
NNLM	23	53	47
CBOW	24	64	61
Skip-gram	55	59	56

syntactic