Efficient Estimation of Word Representations in Vector Space

https://arxiv.org/abs/1301.3781

발표에 앞서 해당 발표 자료는 아래 유튜브 영상의 자료를 캡쳐해 만들어졌음을 사전에 밝힙니다

자료출처

https://www.youtube.com/watch?v=sidPSG-EVDo

고려대학교 산업경영공학부 DSBA 연구실
[1] 발표자: 김지나
[2] 논문: Efficient Estimation of Word Representations in Vector Space (https://arxiv.org/abs/1301.3781)

- **Vector Representation of Words**
  - ✓ One-Hot Encoding
    - 한 단어에 대해, 표현하고자 하는 단어에 1, 나머지 단어에는 0을 부여한 vocabulary size 크기의 vector로 표현

Example

귤 →

사과	포도	귤	망고	…	오렌지	참외
0	0	1	0	…	0	0

오렌지 →

사과	포도	귤	망고	…	오렌지	참외
0	0	0	0	…	1	0

- 차원이 높고 sparse하기 때문에 이 vector로부터 정보를 추출하기 어렵다.
- 단어 간 similarity에 대한 정보를 담고 있지 않다.

- **Vector Representation of Words**
  - ✓ Distributed Representation
    - – 단어의 의미를 여러 차원에 분산하여 표현

Distributional Hypothesis

Words that occur in similar contexts tend to have similar meanings

Example

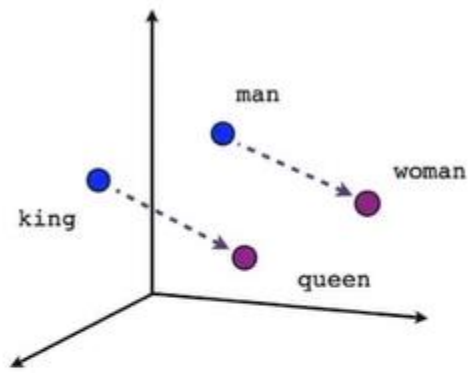
귤	→	0.2	0.3	−0.19	0.12	−0.2	0.4	0.2
오렌지	→	0.4	0.3	0.5	0.2	0.1	0.2	0.12

- **Vector Representation of Words**

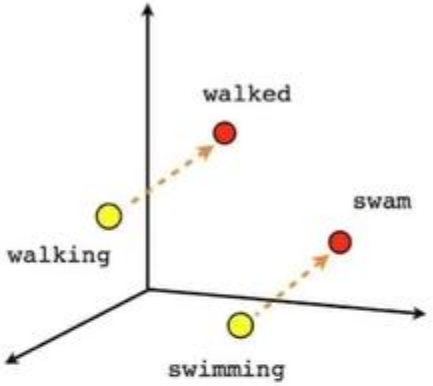
이미지 출처: https://towardsdatascience.com/creating-word-embeddings-coding-the-word2vec-algorithm-in-python-using-deep-learning-b337d0ba17a8

  - ✓ Distributed Representation의 이점
    - 단어 간 similarity 계산 가능
    - Algebraic operation 가능

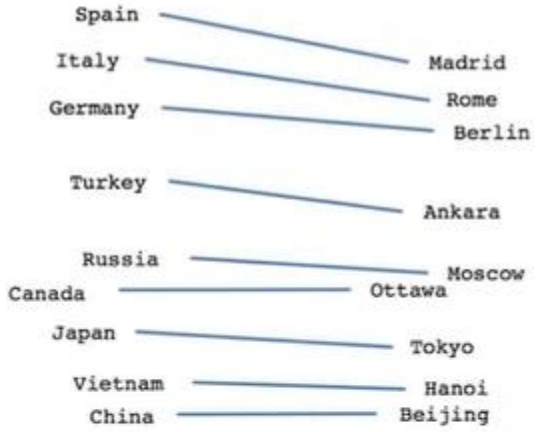
Example



Male-Female



Verb tense



Country-Capital

Vector('Madrid') − Vector('Spain') + Vector('Italy') = Vector('Rome')

• Goals of the Paper

❖ 대량의 data로부터 높은 수준의 word vector를 학습하는 기술 제안

❖ Vector representation의 결과의 quality를 평가하기 위한 technique 제안

→ 비슷한 단어가 서로 가까울 뿐만 아니라, multiple degrees of similarity를 갖게 한다.

❖ 학습 시간과 정확도가 word vector의 차원과 train data의 양에 어떻게 의존하는지 토의할 것이다.

Model 비교를 위한 Computational Complexity 정의

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

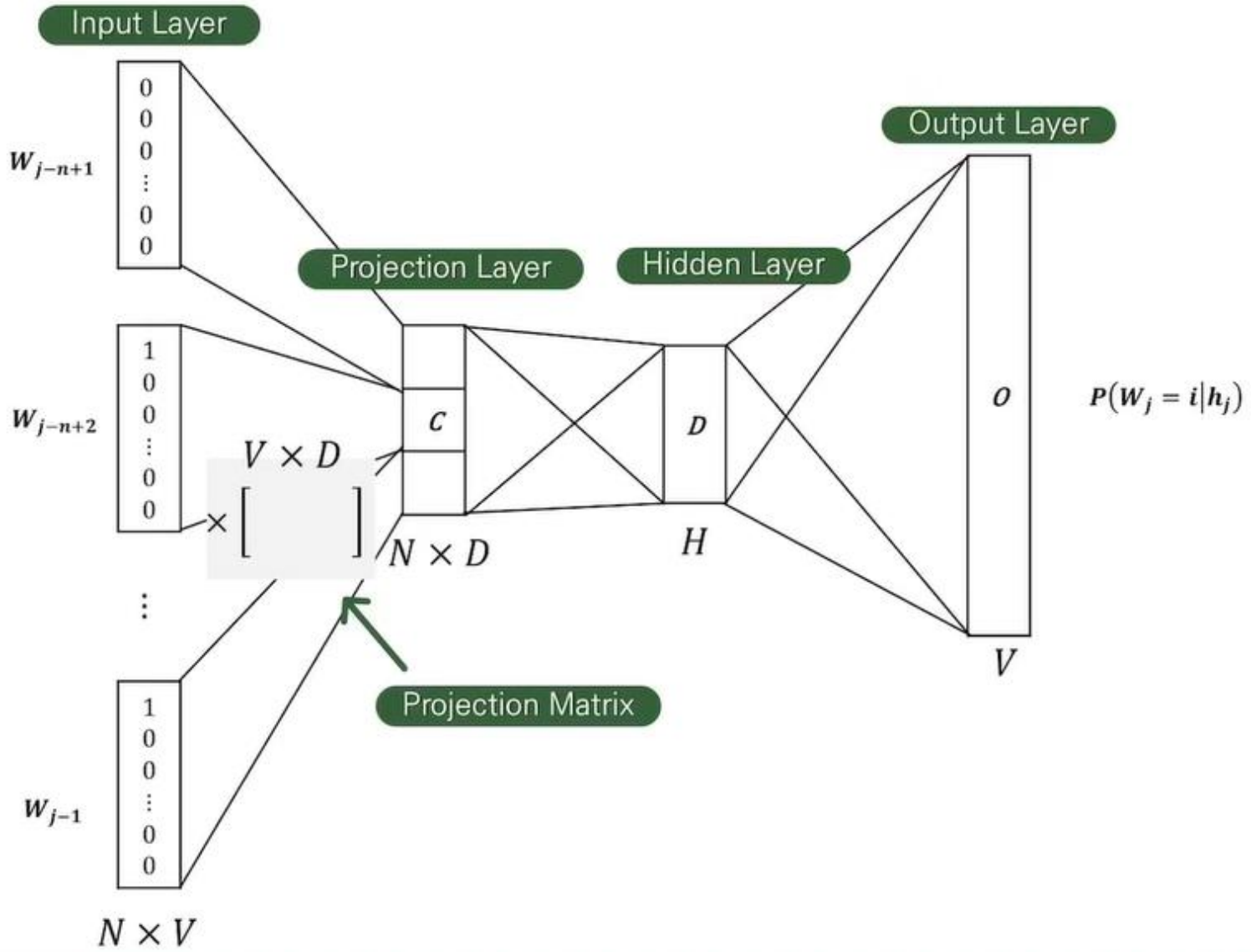
E: $number\ of\ the\ training\ epochs$
T: $number\ of\ the\ words\ in\ the\ training\ set$
Q: 각 $model$ 구조에 의해 정의됨

2

Previous work

- **Feed-Forward Neural Net Language Model (NNLM)**

✓ Structure



Notation

N: $input$으로 들어가는 이전 단어의 개수

D: $projection$ 후의 $dimension$

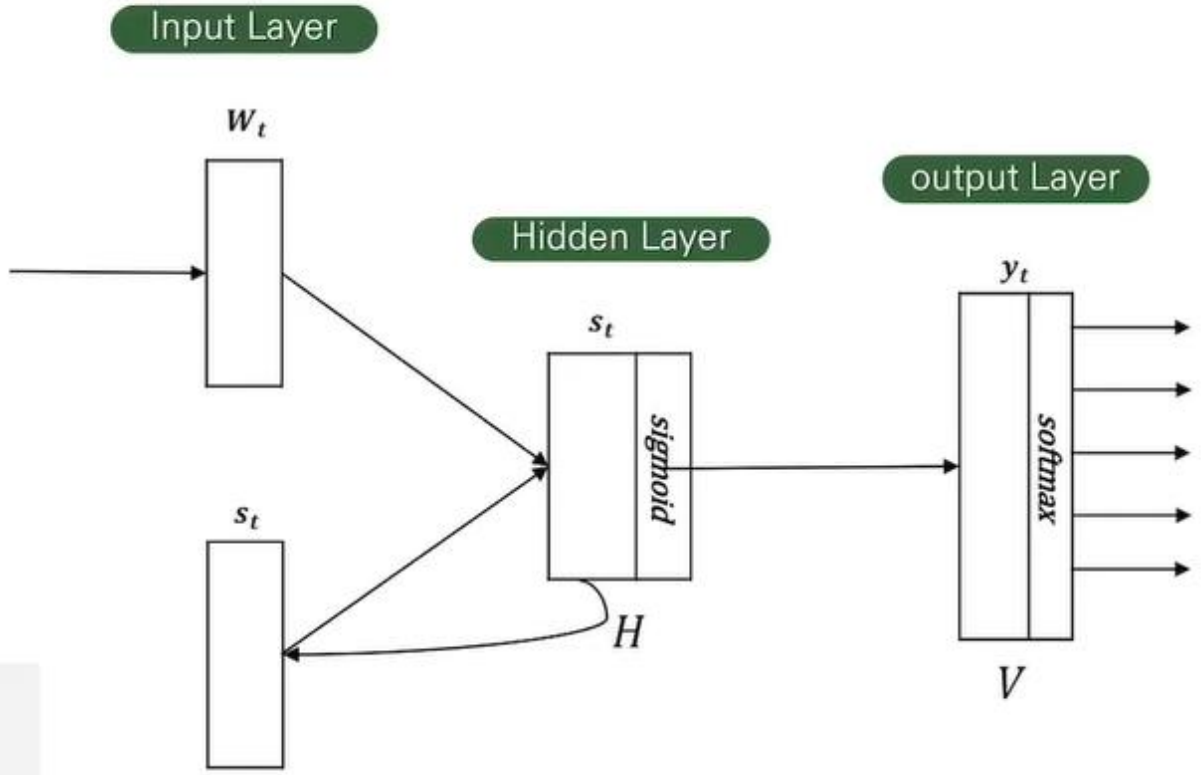
H: $hidden\ layer\ size$

V: $Vocabulary\ size$

- **Feed-Forward Neural Net Language Model (NNLM)**
  - ✓ 한계
    - History로 사용할 단어의 개수를 고정해주어야 한다.
    - History만을 보고 예측하기 때문에 미래 시점의 단어들을 고려하지 않는다.
    - Computational complexity:

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

- **Recurrent Neural Net Language Model (RNNLM)**
  - ✓ Structure



Notation

H: *hidden layer size*

V: *Vocabulary size*

- **Recurrent Neural Net Language Model (RNNLM)**
  - ✓ 이점
    - Input으로 사용할 단어의 수를 정해줄 필요 없이 학습할 단어를 순차적으로 입력하면 된다.
    - Computational complexity:

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

    - Word representation D는 hidden layer H와 같은 dimension을 가짐

3

Word2Vec

- **Model Structure**

  ✓ Distributed representation of words

  ✓ Complexity의 대부분이 model의 non-linear hidden layer로부터 야기된다.

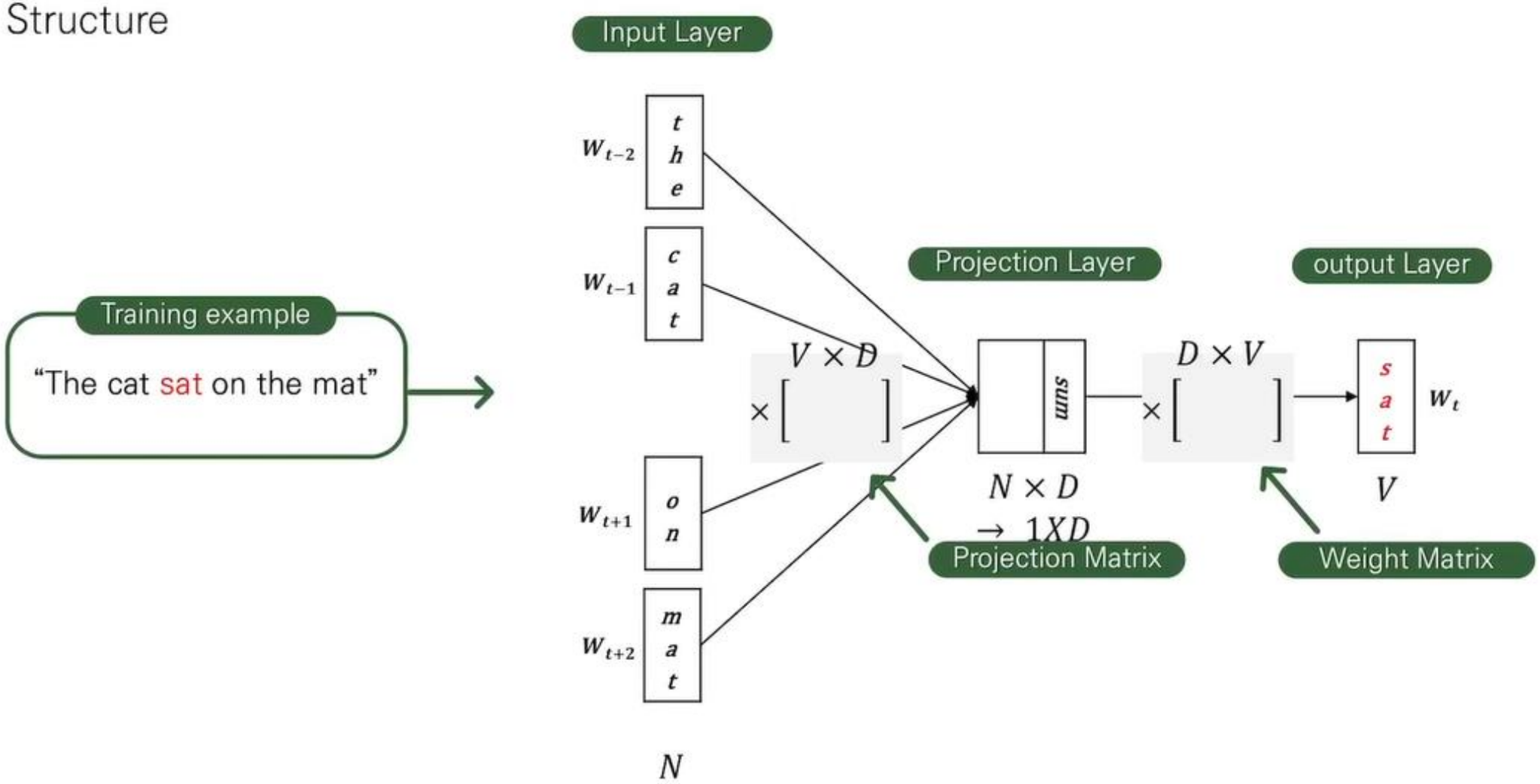
  → feedforward NNLM의 structure를 따름

  ✓ Structure

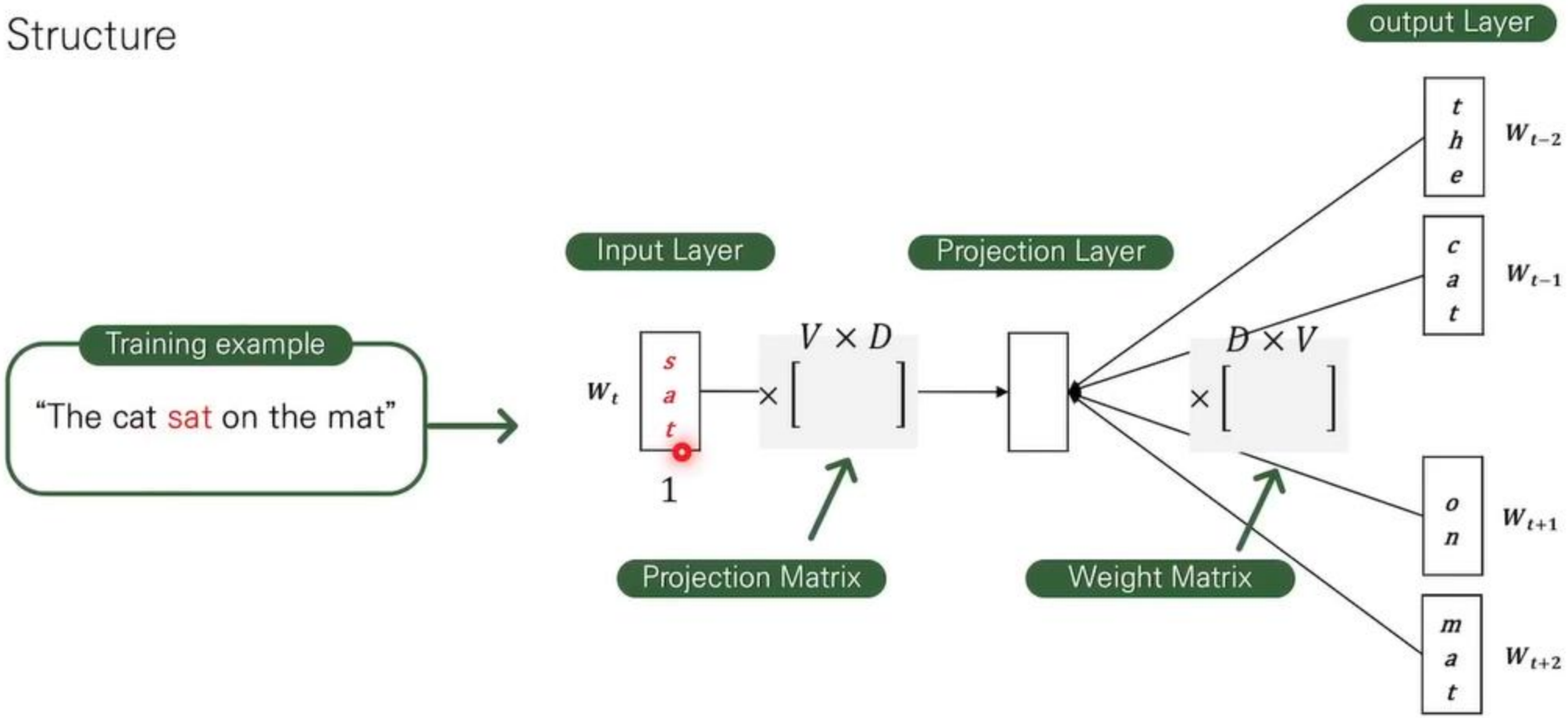
  – Continuous Bag-of-Words Model (CBOW)

  – Continuous Skip-gram Model (Skip-gram)

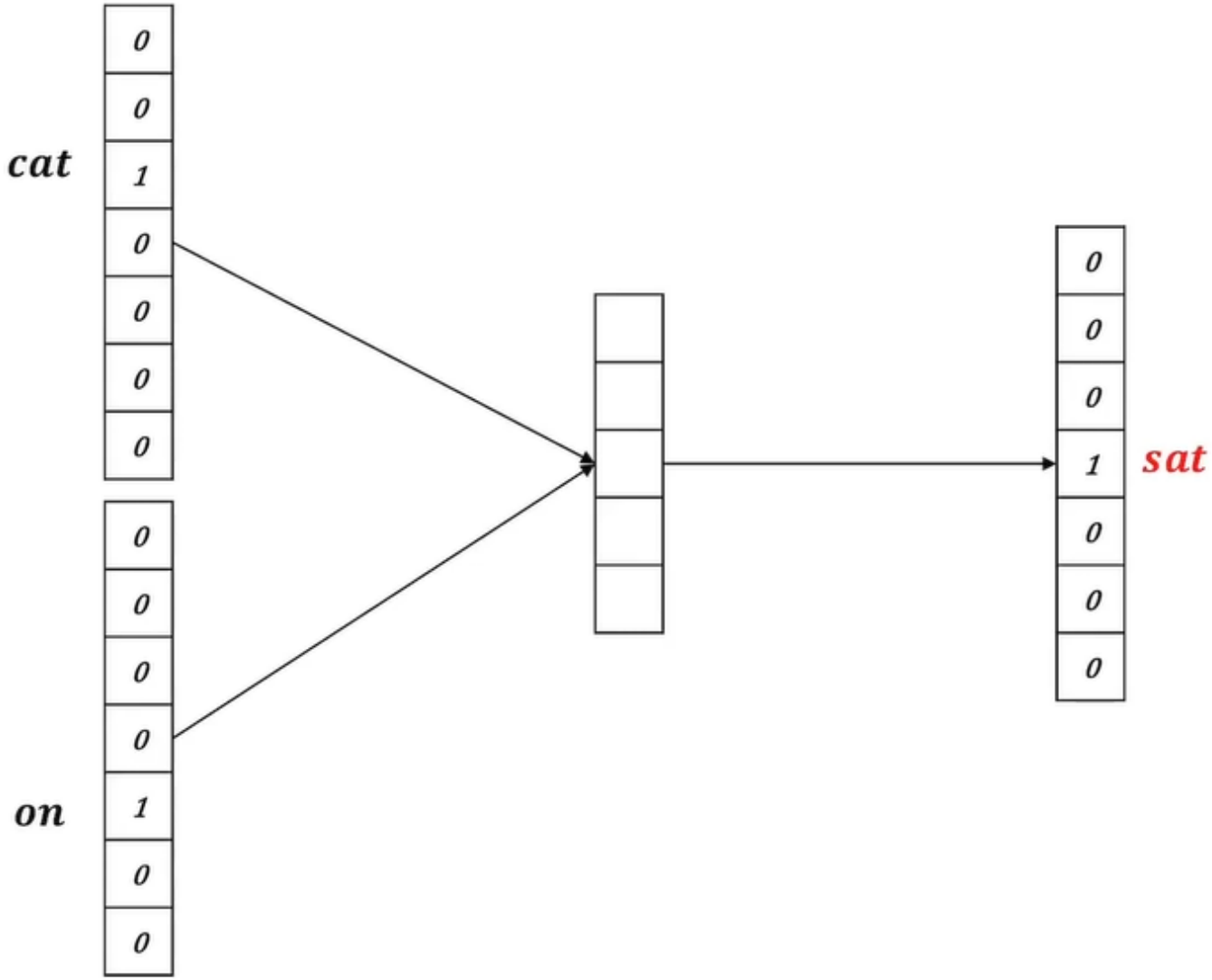
- **Continuous Bag-of-Words Model**
  - ✓ Structure



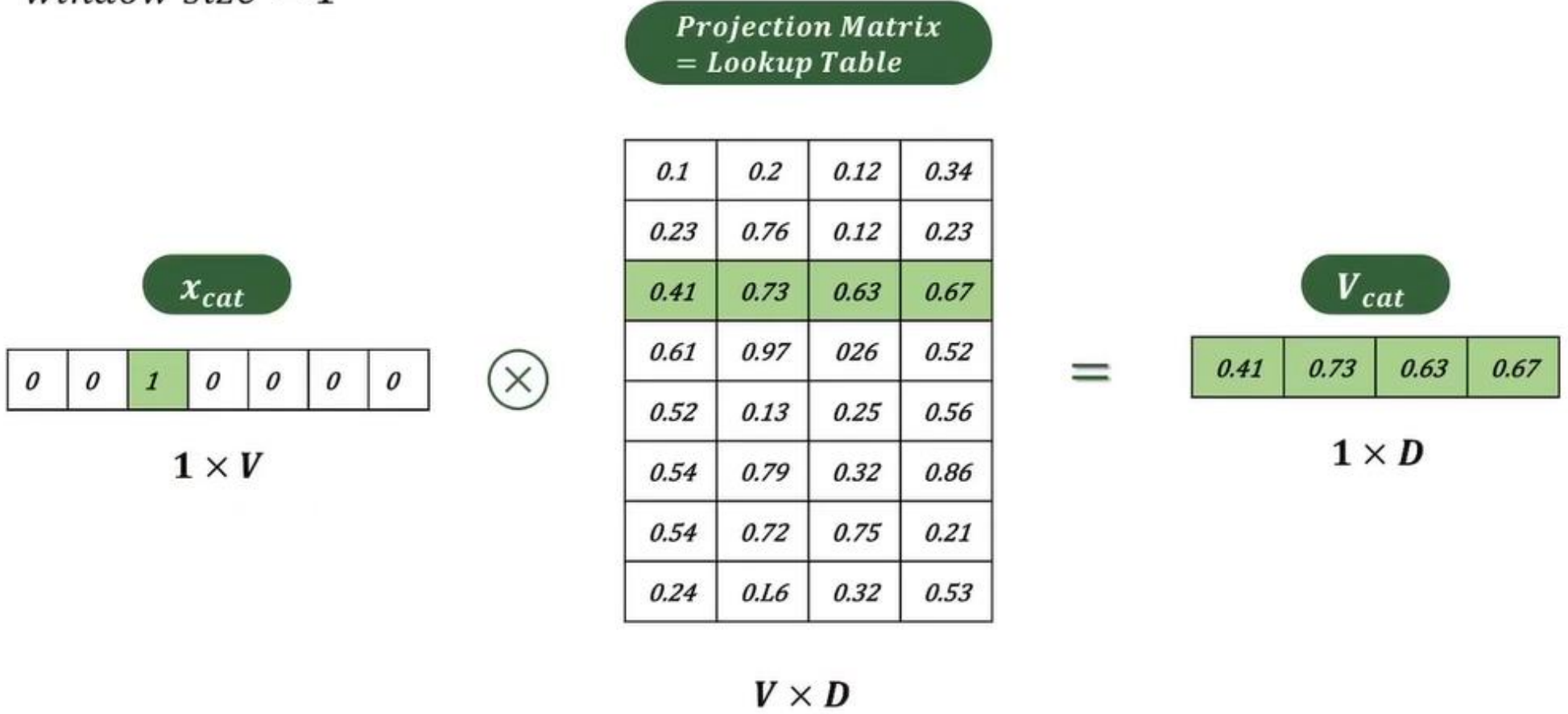
- **Continuous Skip-gram Model**
  - ✓ Structure



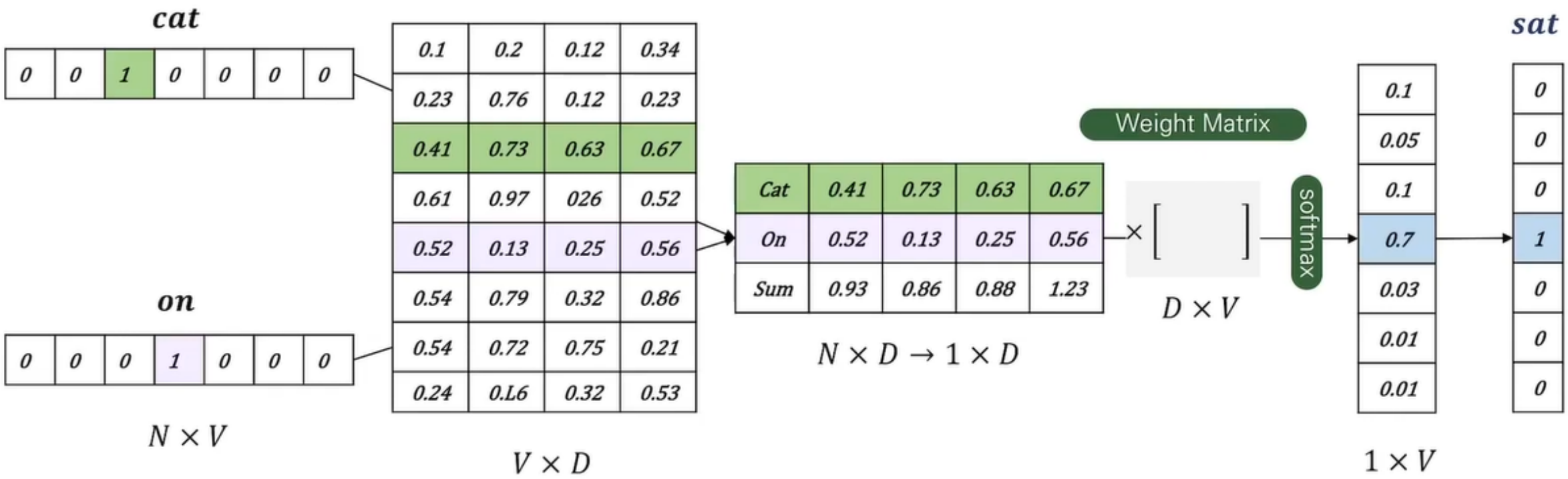
- **Example**
  - ✓ "The fat cat sat on the mat"
    - $window\ size = 1$



- **Example**
  - ✓ "The fat cat sat on the mat"
    - $window\ size = 1$



- **Example**
  - ✓ "The fat cat sat on the mat"
    - $window\ size = 1$



- **Computational Complexity**

✓ CBOW

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

1. $N \times D$: 현재 단어를 중심으로 N개의 단어 projection
2. $D \times V$: projection layer에서 output layer 계산

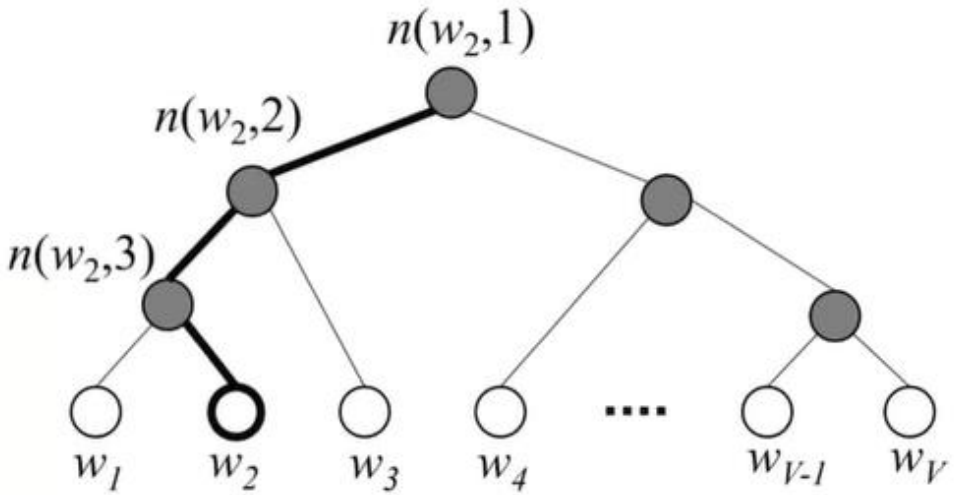
✓ Skip-gram

- $Q = C \times (D + D \times V)$

1. $D + D \times V$: 현재 단어 projection + output 계산
2. $\times C$: C 개의 단어에 대해 진행해야 하므로 총 C배

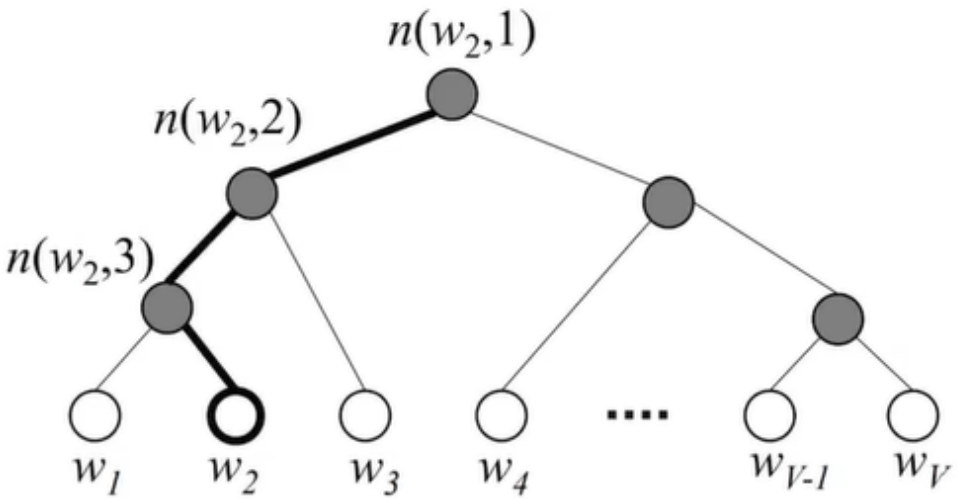
- **Hierarchical Softmax**

$$P(w = w_O) = \prod_{j=1}^{L(w)-1} \sigma([\![n(w, j+1) = ch(n(w, j))]\!] \cdot \mathrm{v'}_{n(w,j)}{}^{T} \mathrm{h})$$



Notation	
$L(w)$	$leaf\ w$에 도달하기까지의 $path$의 길이
$n(w,i)$	$root$에서부터 $leaf\ w$에 도달하는 $path$에서 만나는 i번째 $node$ 예) $n(w,1) = root, n(w, L(w)) = w$
$ch(n)$	$node$의 왼쪽 자식
$\mathrm{v'}_{n(w,j)}$	$W^{'} matrix$ 대신 사용하는 길이 D인 $weight\ vector$
h	$Input\ word$에 대한 $projection\ matrix\ row.$ 길이 D
$[[x]] = \begin{cases} 1 & if\ x\ is\ true \\ -1 & otherwise \end{cases}$	

- Hierarchical Softmax



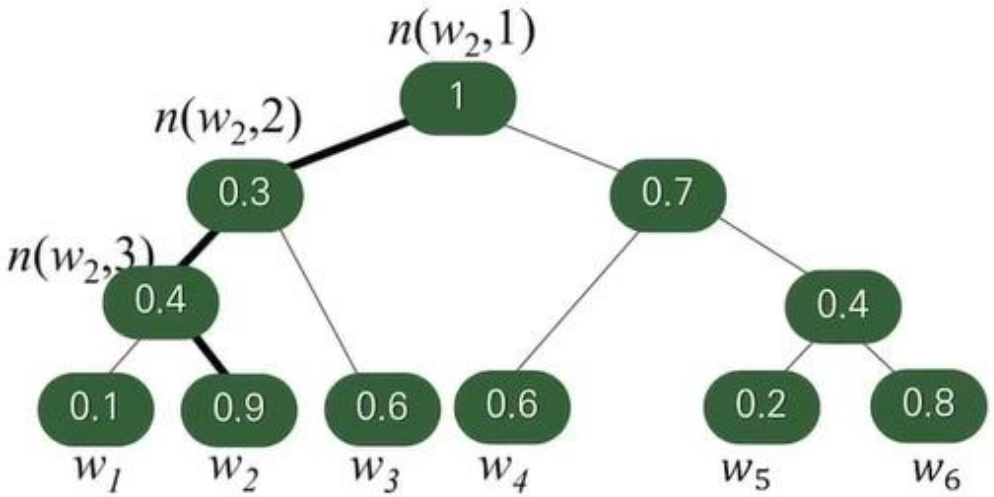
$$P(n, left) = \sigma\left({v'_n}^T \cdot h\right)$$
$: node\ n$이 $left$로 갈 확률

$$P(n, right) = \sigma\left(-{v'_n}^T \cdot h\right) = 1 - \sigma\left({v'_n}^T \cdot h\right)$$
$: sigmoid$이므로 위의 식이 성립

$$P(w_2 = w_O)$$
$$= P(n(w_2, 1), left) \cdot P(n(w_2, 2), left) \cdot P(n(w_2, 3), right)$$
$$= \sigma\left({v'_{n(w_2,1)}}^T \cdot h\right) \cdot \sigma\left({v'_{n(w_2,2)}}^T \cdot h\right) \cdot \sigma\left(-{v'_{n(w_2,3)}}^T \cdot h\right)$$

$$P(w = w_O) = \prod_{j=1}^{L(w)-1} \sigma([\![n(w, j+1) = ch(n(w, j))]\!] \cdot {v'_{n(w,j)}}^T h)$$

- Hierarchical Softmax



$$\sum_{i=1}^{6} P(w_i = w_O) = 1 \times 0.3 \times 0.4 \times 0.1 + 1 \times 0.3 \times 0.4 \times 0.9 + 1 \times 0.3 \times 0.6 + 1 \times 0.7 \times 0.6 + 1 \times 0.7 \times 0.4 \times 0.2 + 1 \times 0.7 \times 0.4 \times 0.8 = 1$$

$P(n, left) = \sigma\left({v'_n}^T \cdot h\right)$
: $node\ n$의 $left$로 갈 확률

$P(n, right) = \sigma\left(-{v'_n}^T \cdot h\right)$
: $node\ n$의 $right$로 갈 확률

$P(n, left) + P(n, right) = 1$
: $sigmoid$이므로 위의 식이 성립

$$\sum_{i=1}^{V} P(w_i = w_O) = 1$$

: 모든 단어의 값에 대한 계산 없이

전체 합이 1이 되는 확률값을 구할 수 있다.

$Computational\ complexity \rightarrow D \times \ln(V)$

D: 각 $step$마다 길이 D의 $vector$ 내적

$ln(V)$: $binary\ tree$의 $depth$는 $ln(V)$에 비례한다.

- **Task Description**
  - ✓ Semantic question 5가지
  - ✓ Syntactic question 9가지
  - ✓ Example)

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

- **Maximization of Accuracy**

  ✓ CBOW

  ✓ Data: Google News corpus

  – 약 60억 개의 token 포함

  – 이 중 frequency 기준으로 단어를 선정해 dataset size를 조절하여 실험

  ✓ Word vector dimensionality 변경

Dimensionality / Training words	24M	49M	98M	196M	391M	783M
50	13.4	15.7	18.6	19.1	22.5	23.2
100	19.4	23.1	27.8	28.7	33.4	32.2
300	23.2	29.2	35.3	38.6	43.7	45.9
600	24.0	30.1	36.5	40.8	46.6	50.4

✓ Word vector의 차원과 data size를 동시에 크게 가질 때 성능이 향상된다.

✓ $Q = N \times D + D \times log_2(V)$

→ training data의 양을 두 배로 늘렸을 때$(2Q)$, vector 차원(D)을 두 배 늘렸을 때 증가하는 계산 복잡도와 거의 비슷하게 증가

- **Comparison of Model Architecture**

  - ✓ Word vector dimensionality=640 고정
  - ✓ Training data 동일
    - 3만개의 어휘 test set에 있는 semantic-syntactic word relationship 질문 모두 사용

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

- ✓ NNLM이 RNNLM보다 성능이 좋다. 이는 NNLM이 projection layer를 거쳐 hidden layer로 들어가기 때문이다.
- ✓ CBOW는 syntactic task에서 NNLM보다 큰 성능 향상을 보였다.
- ✓ Skip-gram은 syntactic 질문에 대해서는 CBOW보다 약간 성능이 떨어지지만, semantic에서는 가장 좋은 성능을 보였다.

- **Comparison of Model Architecture**

  ✓ 동일한 training dataset에 대해, epoch=1 or 3으로 실험한 결과

  ✓ Full Semantic-Syntactic dataset 사용

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days]
			Semantic	Syntactic	Total	
3 epoch CBOW	300	783M	15.5	53.1	36.1	1
3 epoch Skip-gram	300	783M	50.0	55.9	53.3	3
1 epoch CBOW	300	783M	13.8	49.9	33.6	0.3
1 epoch CBOW	300	1.6B	16.1	52.6	36.1	0.6
1 epoch CBOW	600	783M	15.4	53.3	36.2	0.7
1 epoch Skip-gram	300	783M	45.6	52.2	49.2	1
1 epoch Skip-gram	300	1.6B	52.2	55.1	53.8	2
1 epoch Skip-gram	600	783M	56.7	54.5	55.5	2.5

✓ 같은 data size로 3번의 epoch을 도는 것보다, 두 배 이상의 data로 epoch 한번 도는 것이 수행 시간이 더 짧고, 비슷하거나 더 좋은 결과를 낸다.

✓ Training data size를 두 배 이상 늘리는 것보다 vector 차원을 두 배 늘리는 것이 더 큰 성능 향상을 가져왔다.

- **Word Relationships**

✓ Best word vector들을 사용해 도출한 word pair relationship 예시

– Skip-gram model trained on 783M words with 300 dimensionality

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

✓ 두 vector를 뺌으로써 관계를 정의할 수 있다. 예) Paris-France+Italy=Rome

✓ accuracy = 약 60%

✓ Relationship example 10개 이상 주어지면, semantic-syntactic test에서 accuracy 약 10%까지 향상

7. Conclusion

- ❖ 매우 간단한 model 구조를 사용해 feedforward NNLM, RNNLM 같은 NN model과 비교해서 높은 quality의 word vector를 train할 수 있다는 것을 발견
- ❖ 다양한 syntactic, semantic language task를 제시해 word vector가 다양한 의미들에 의한 여러 similarity를 반영하는가를 평가.
- ❖ 이전 model 보다 낮은 계산 복잡도 덕분에 훨씬 많은 dataset으로부터 높은 차원의 정확한 word vector를 계산하는 것이 가능
- ❖ Pre-trained embedding model로써 다양한 NLP task에 사용될 수 있음