

# word2vec

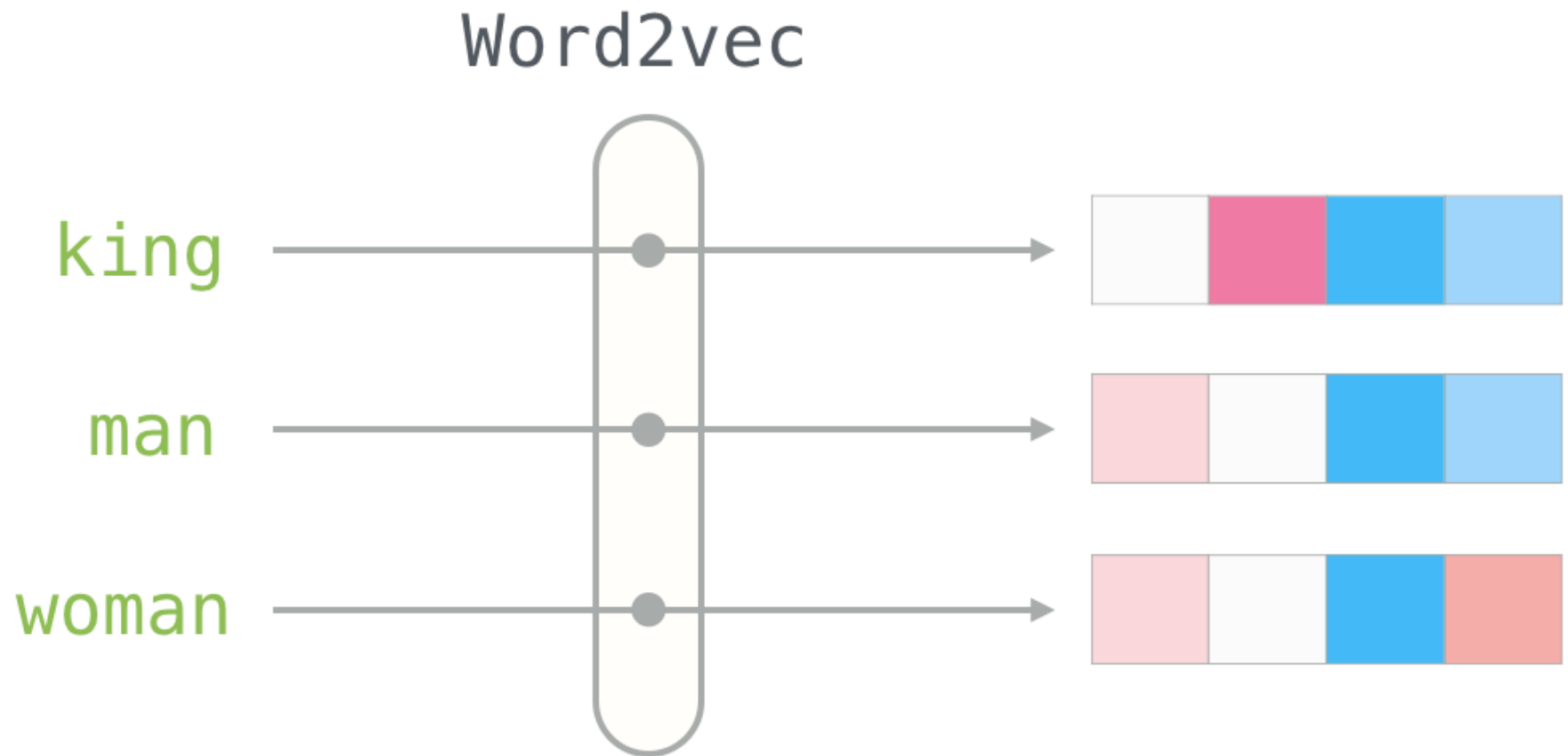
---

Jay Alammara The Illustrated Word2vec

2020

## **word2vec illustrated**

# Word2Vec illustrated



# 성격 임베딩: 당신의 성격은 어떠한가?

- 성격 임베딩의 예제

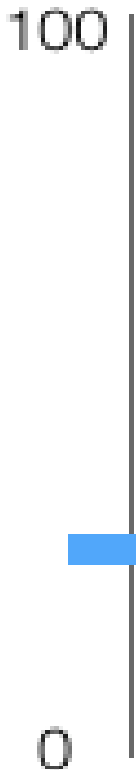
Openness to experience	79 out of 100
Agreeableness	75 out of 100
Conscientiousness	42 out of 100
Negative emotionality	50 out of 100
Extraversion	58 out of 100

# 성격 임베딩: 당신의 성격은 어떠한가?

- 내성적인지 외향적인지?

Extraversion

100



0

Introversion

Jay

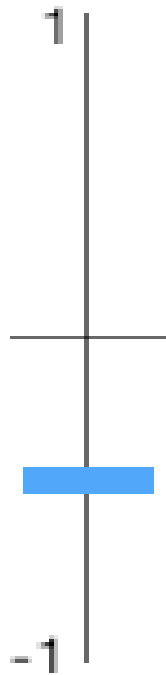
Extraversion



# 성격 임베딩: 당신의 성격은 어떠한가?

- 내성적인지 외향적인지? (표준화)

Extraversion



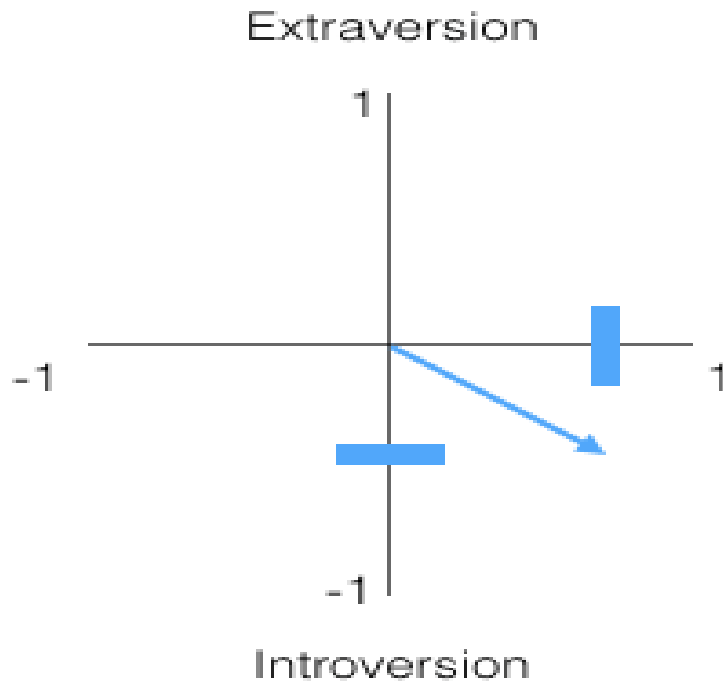
Introversion

Jay



# 성격 임베딩: 당신의 성격은 어떠한가?

- 여러 특성이 있으면 벡터로 표시할 때 더 잘 표현할 수 있다.

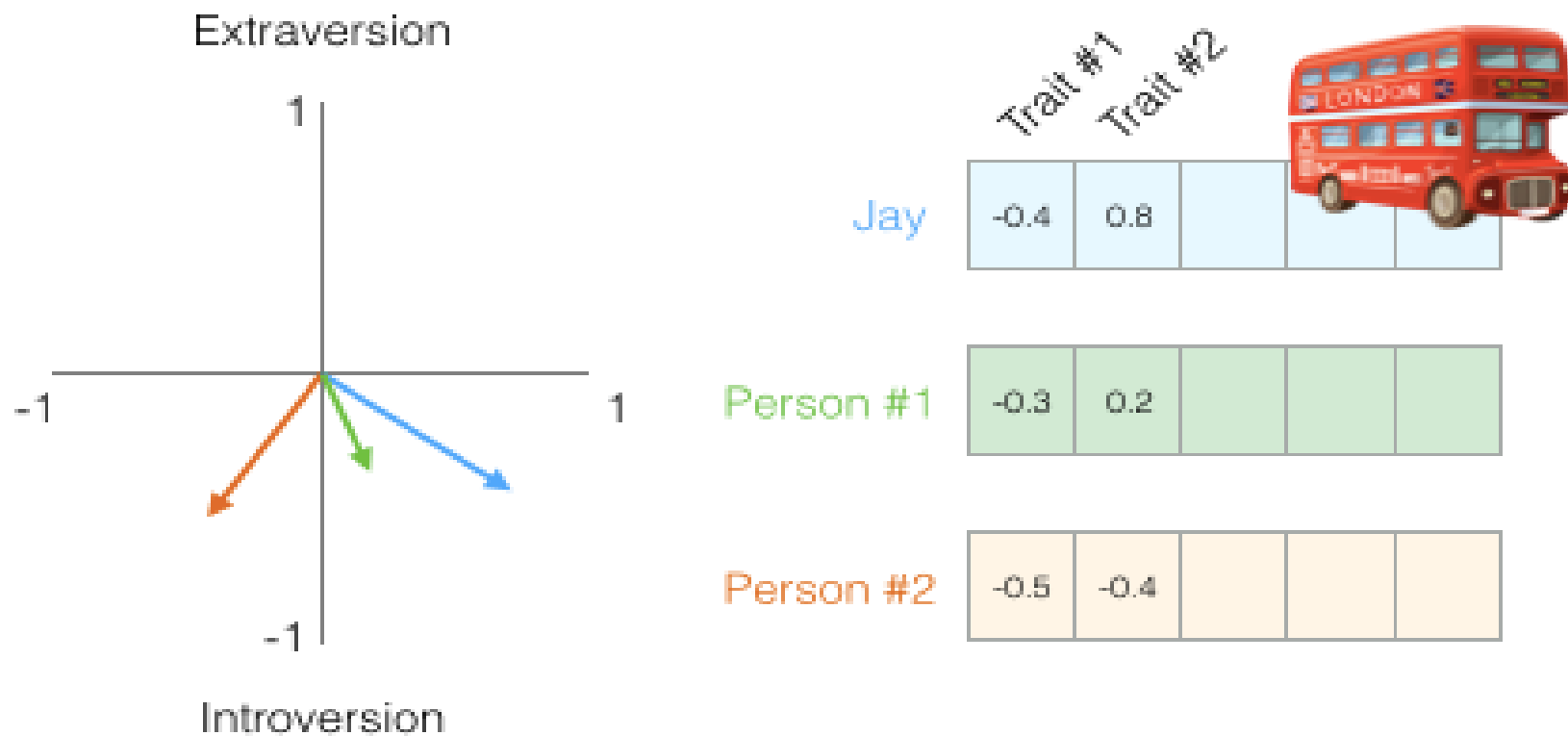


Jay

Trait #1	Trait #2			
-0.4	0.8			

# 성격 임베딩: 당신의 성격은 어떠한가?

- 벡터로 표시하면 비교하기 더 쉽다.





## 성격 임베딩: 당신의 성격은 어떠한가?

- 유사도를 구할 수 있다.

$$\text{cosine\_similarity}\left(\begin{array}{c|c} \text{Jay} \\ \hline -0.4 & 0.8 \end{array}, \begin{array}{c|c} \text{Person \#1} \\ \hline -0.3 & 0.2 \end{array}\right) = 0.87 \quad \checkmark$$

$$\text{cosine\_similarity}\left(\begin{array}{c|c} \text{Jay} \\ \hline -0.4 & 0.8 \end{array}, \begin{array}{c|c} \text{Person \#2} \\ \hline -0.5 & -0.4 \end{array}\right) = -0.20$$

# 성격 임베딩: 당신의 성격은 어떠한가?

	Trait #1	Trait #2	Trait #3	Trait #4	Trait #5
Jay	-0.4	0.8	0.5	-0.2	0.3
Person #1	-0.3	0.2	0.3	-0.4	0.9
Person #2	-0.5	-0.4	-0.2	0.7	-0.1

## 성격 임베딩: 당신의 성격은 어떠한가?

Jay                      Person #1

$$\text{cosine\_similarity}(\begin{bmatrix} -0.4 & 0.8 & 0.5 & -0.2 & 0.3 \end{bmatrix}, \begin{bmatrix} -0.3 & 0.2 & 0.3 & -0.4 & 0.9 \end{bmatrix}) = 0.66 \quad \checkmark$$

Jay                      Person #2

$$\text{cosine\_similarity}(\begin{bmatrix} -0.4 & 0.8 & 0.5 & -0.2 & 0.3 \end{bmatrix}, \begin{bmatrix} -0.5 & -0.4 & -0.2 & 0.7 & -0.1 \end{bmatrix}) = -0.37$$

# 성격 임베딩: 당신의 성격은 어떠한가?

- 사람의 성격을 벡터로 표시할 수 있고, 유사도를 구할 수 있다.

1- We can represent things  
(and people) as vectors of  
numbers  
(Which is great for machines!)

Jay

-0.4	0.8	0.5	-0.2	0.3
------	-----	-----	------	-----

2- We can easily calculate how  
similar vectors are to each other

The people most similar to Jay are:

cosine\_similarity ▼

Person #1	0.86
Person #2	0.5
Person #3	-0.20

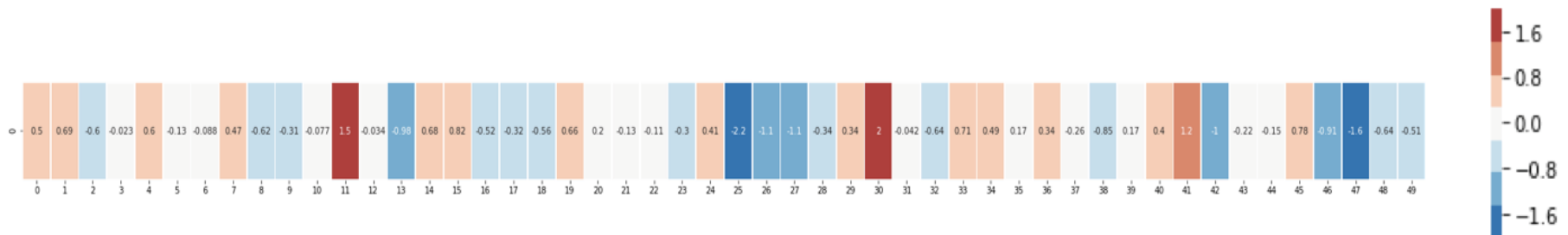
# Word Embedding

- 사전학습된 word-vector 예 (word-embedding이라고도 불린다)

- 예를 들면 "king" (GloVe vector trained on Wikipedia)은 다음과 같이 표현된다.

[ 0.50451 , 0.68607 , -0.59517 , -0.022801, 0.60046 , -0.13498 , -0.08813 , 0.47377 , -0.61798 , -0.31012 , -0.076666, 1.493 , -0.034189, -0.98173 , 0.68229 , 0.81722 , -0.51874 , -0.31503 , -0.55809 , 0.66421 , 0.1961 , -0.13495 , -0.11476 , -0.30344 , 0.41177 , -2.223 , -1.0756 , -1.0783 , -0.34354 , 0.33505 , 1.9927 , -0.04234 , -0.64319 , 0.71125 , 0.49159 , 0.16754 , 0.34344 , -0.25663 , -0.8523 , 0.1661 , 0.40102 , 1.1685 , -1.0137 , -0.21585 , -0.15155 , 0.78321 , -0.91241 , -1.6106 , -0.64426 , -0.51042 ]

- 이는 50개의 숫자 리스트이다. 알아보기 힘들지만, 다른 단어벡터와 비교하기 위해 시각화할 수 있다.



# Word Embedding

---

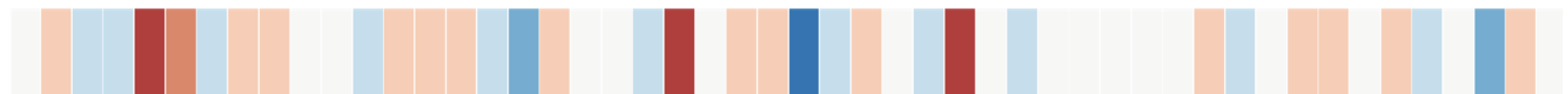
“king”



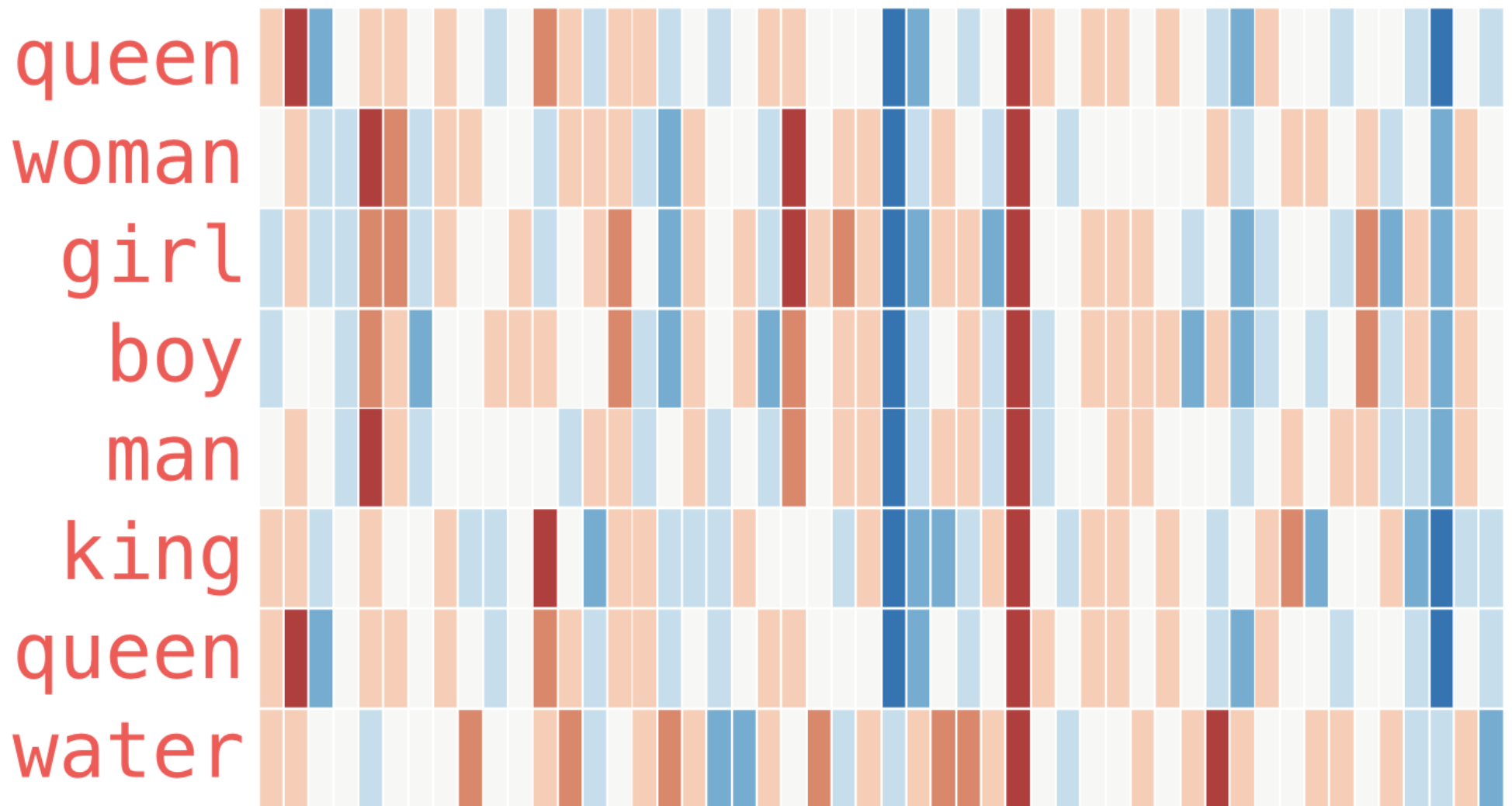
“Man”



“Woman”



# Word Embedding

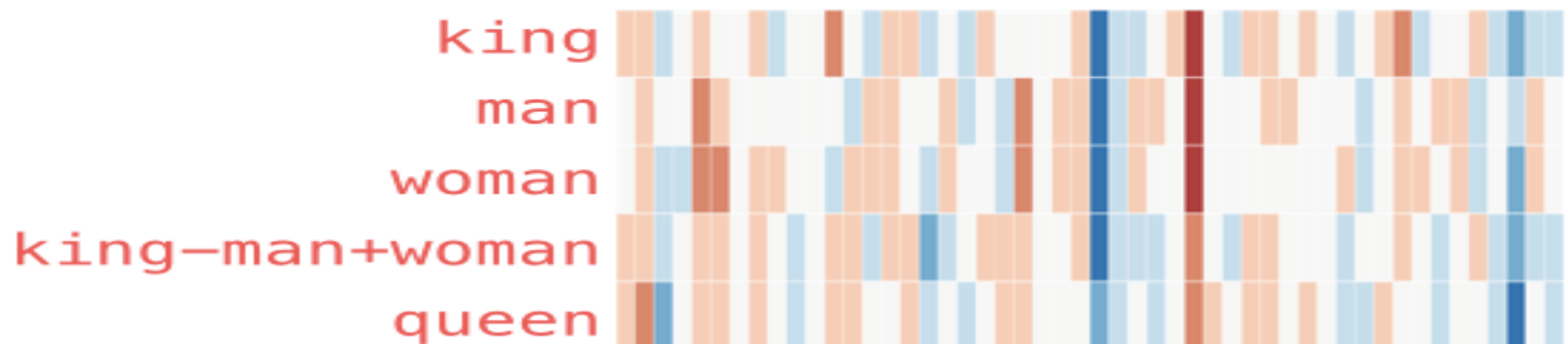


# Analogy

```
model.most_similar(positive=["king", "woman"], negative=["man"])
```

```
[('queen', 0.8523603677749634),  
 ('throne', 0.7664333581924438),  
 ('prince', 0.7592144012451172),  
 ('daughter', 0.7473883032798767),  
 ('elizabeth', 0.7460219860076904),  
 ('princess', 0.7424570322036743),  
 ('kingdom', 0.7337411642074585),  
 ('monarch', 0.721449077129364),  
 ('eldest', 0.7184862494468689),  
 ('widow', 0.7099430561065674)]
```

king - man + woman ≈ queen





# 언어 모델

- 다음 단어 예측(Next-word prediction)



input/feature #1

input/feature #2

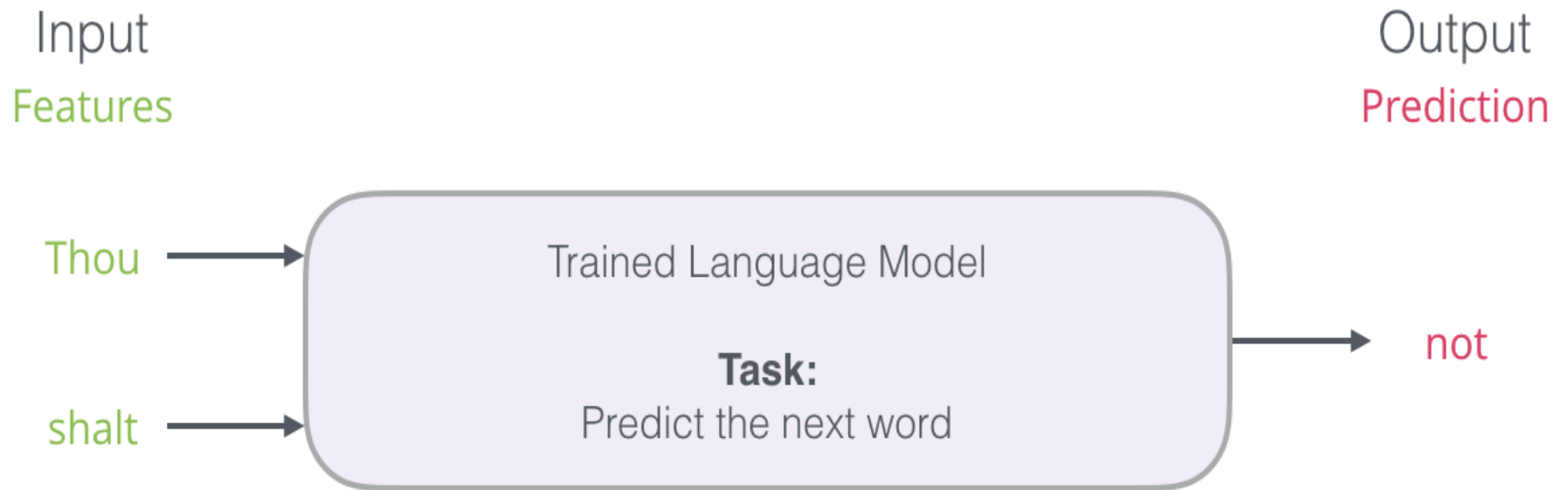
output/label

Thou shalt

\_\_\_\_\_

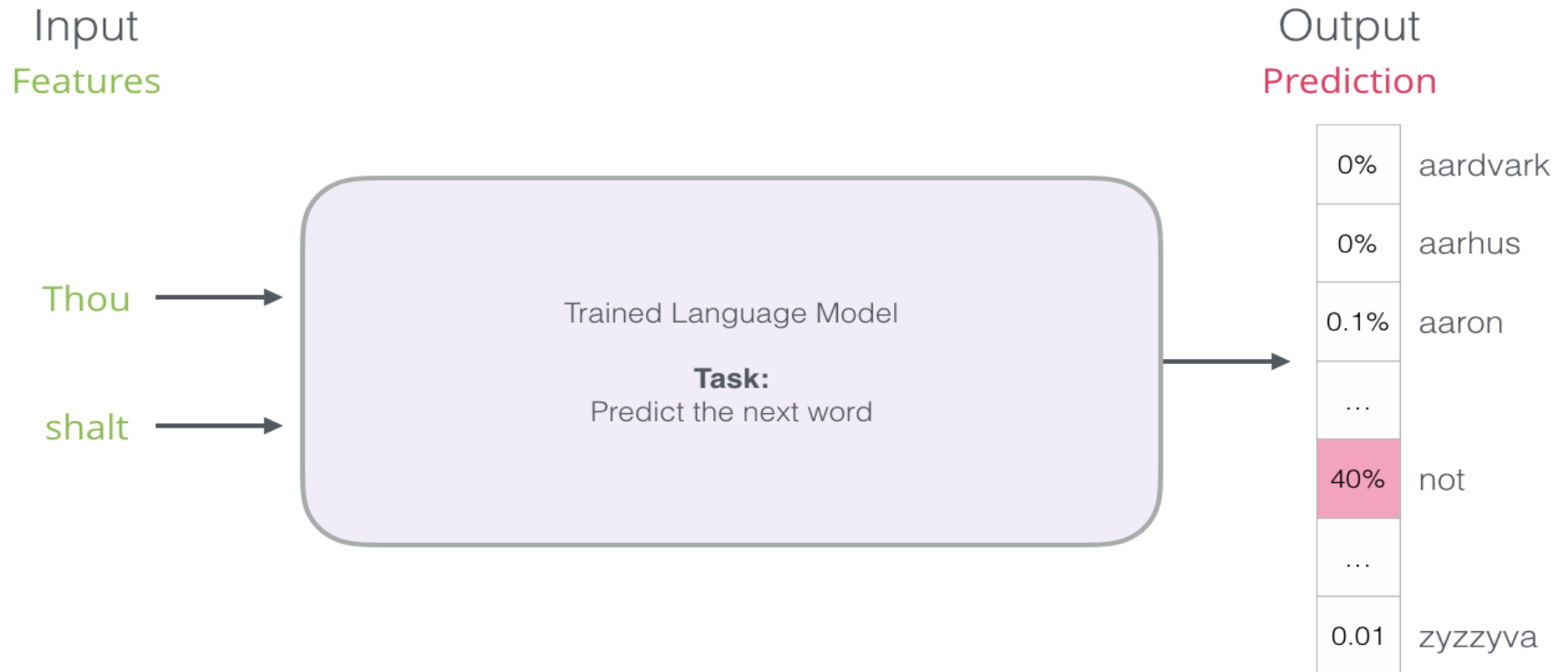
# 언어 모델

- 다음 단어 예측(Next-word prediction)



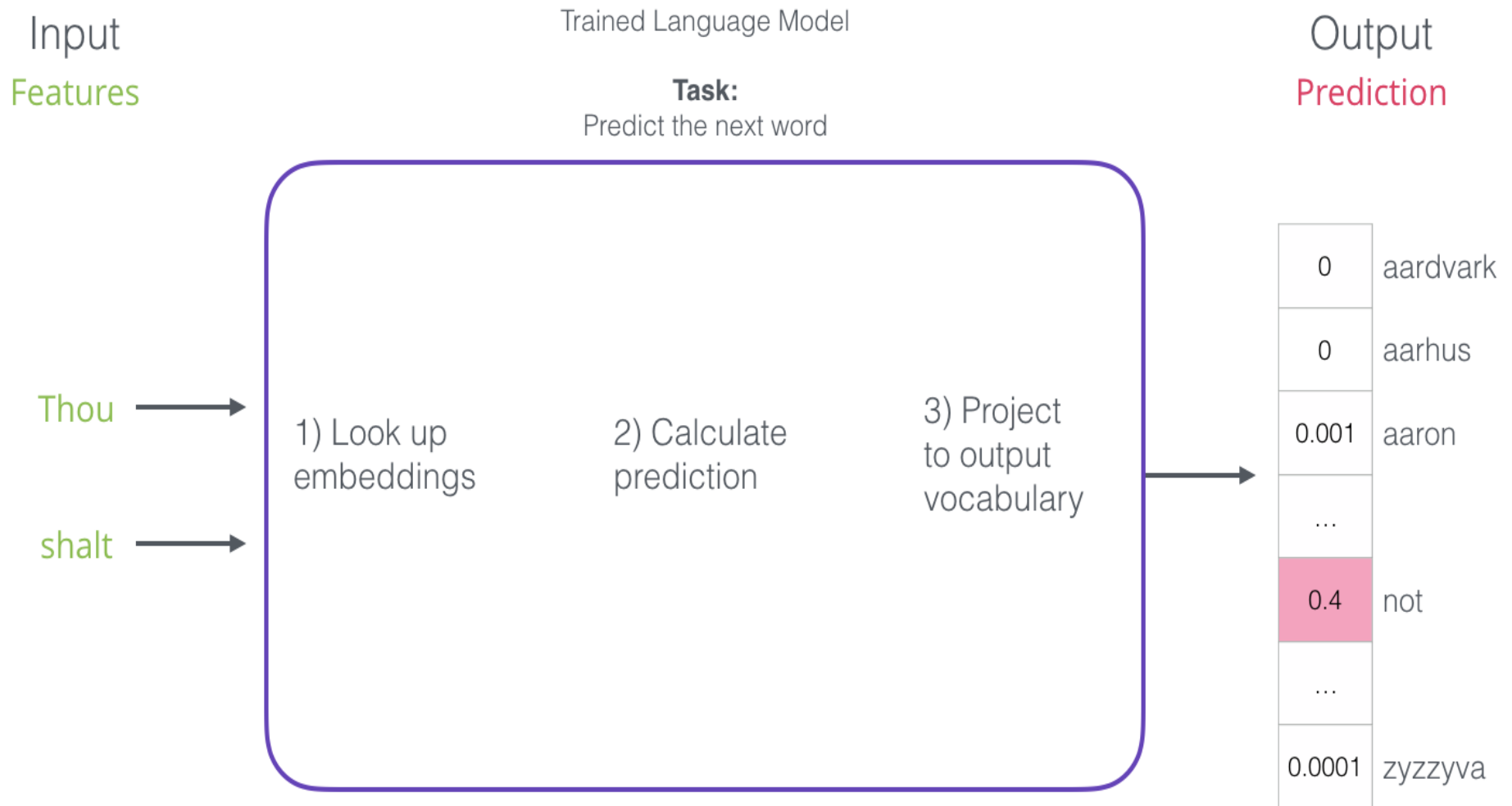
# 언어 모델

- 다음 단어 예측(Next-word prediction)



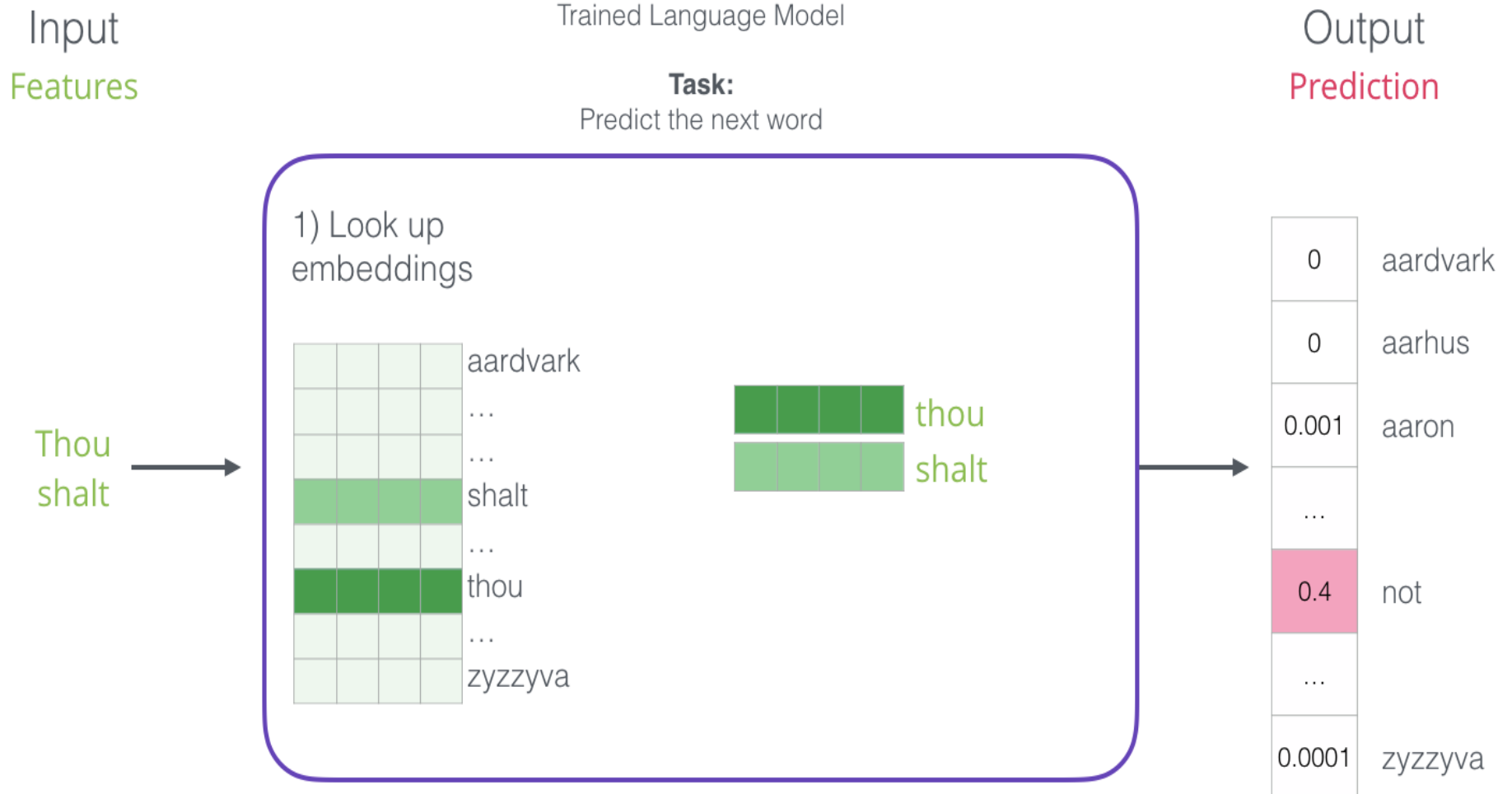
# 언어 모델

- 다음 단어 예측(Next-word prediction)



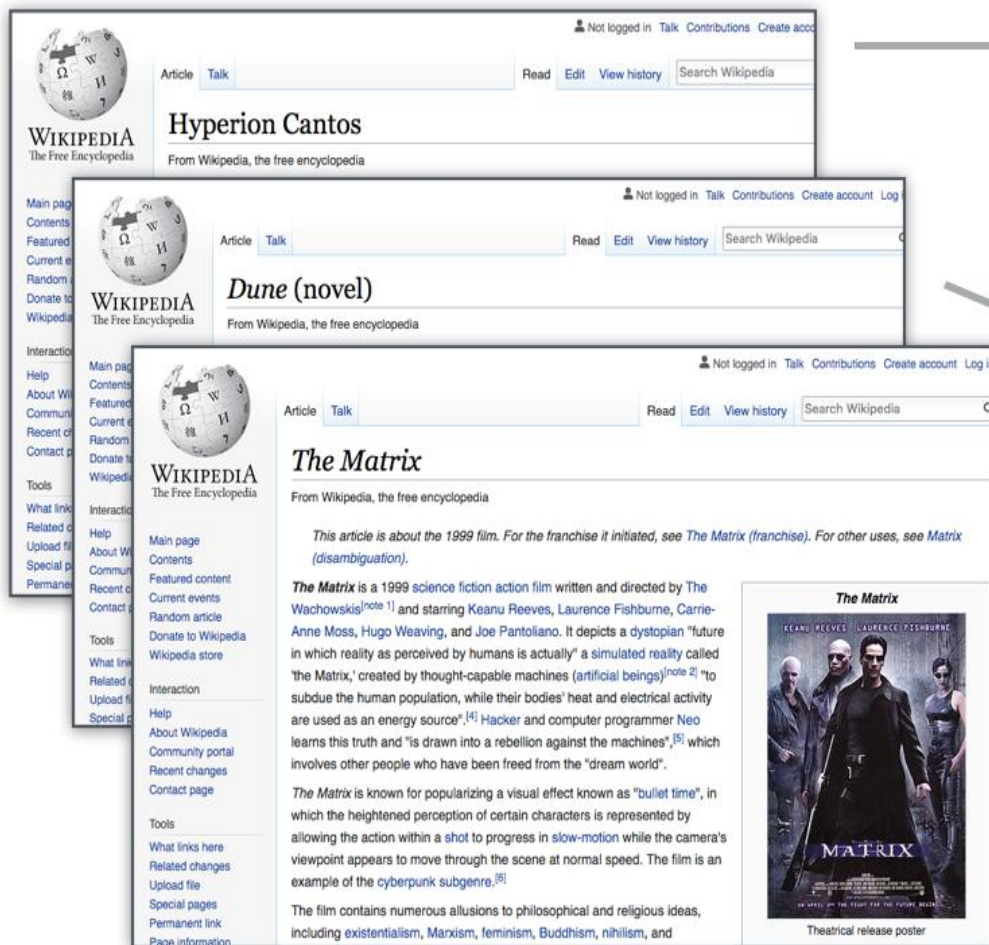
# 언어 모델

- 다음 단어 예측(Next-word prediction)



# 언어 모델 학습

## ● 다음 단어 예측(Next-word prediction)



The **Hyperion Cantos** is a series of science fiction novels by Dan Simmons. The title is a reference to the *Cantos* in the series, *Hyperion* and *The Fall of Hyperion*,<sup>[1][2]</sup> and later came to refer to the overall storyline, including *Endymion*, *The Rise of Endymion*, and a number of short stories.<sup>[3][4]</sup> More narrowly, inside the fictional storyline, after the first volume, the Hyperion Cantos is an epic poem written by the character Martin Silenus covering in verse form the events of the first book.<sup>[5]</sup>

Of the four novels, *Hyperion* received the Hugo and Locus Awards in 1990;<sup>[6]</sup> *The Fall of Hyperion* won the Locus and British Science Fiction Association Awards in 1991;<sup>[7]</sup> and *The Rise of Endymion* received the Locus Award in 1998.<sup>[8]</sup> All four novels were also nominated for various science fiction awards.

An event series is being developed by Bradley Cooper, Graham King, and Todd Phillips for Syfy based on the first novel *Hyperion*.<sup>[9]</sup>

**Dune** is a 1965 science fiction novel by American author Frank Herbert, originally published as two separate serials in *Analog* magazine. It tied with Roger Zelazny's *This Immortal* for the Hugo Award in 1966,<sup>[3]</sup> and it won the inaugural Nebula Award for Best Novel.<sup>[4]</sup> It is the first installment of the *Dune* saga, and in 2003 was cited as the world's best-selling science fiction novel.<sup>[5][6]</sup>

Set in the distant future amidst a feudal interstellar society in which noble houses, in control of individual planets, owe allegiance to the Padishah Emperor, *Dune* tells the story of young Paul Atreides, whose noble family accepts the stewardship of the planet Arrakis. It is an inhospitable and sparsely populated desert wasteland, but is also the only source of *melange*, also known as "spice", a drug that enhances mental abilities. As melange is the most important and valuable substance in the universe, control of Arrakis is a coveted—and dangerous—undertaking. The story explores the multi-layered interactions of politics, religion, ecology, technology, and human emotion, as the factions of the empire confront each other in a struggle for the control of Arrakis

**The Matrix** is a 1999 science fiction action film written and directed by The Wachowskis<sup>[note 1]</sup> and starring Keanu Reeves, Laurence Fishburne, Carrie-Anne Moss, Hugo Weaving, and Joe Pantoliano. It depicts a dystopian "future in which reality as perceived by humans is actually" a simulated reality called 'the Matrix,' created by thought-capable machines (artificial beings)<sup>[note 2]</sup> "to subdue the human population, while their bodies' heat and electrical activity

# 언어모델 학습

- 다음 단어 예측(Next-word prediction)

Thou shalt not make a machine in the likeness of a human mind

Sliding window across running text

thou	shalt	not	make	a	machine	in	the	...
------	-------	-----	------	---	---------	----	-----	-----

Dataset

input 1	input 2	output

# 언어모델 학습

- 다음 단어 예측(Next-word prediction)

Thou shalt not make a machine in the likeness of a human mind

Sliding window across running text

thou	shalt	not	make	a	machine	in	the	...
------	-------	-----	------	---	---------	----	-----	-----

Dataset

input 1	input 2	output
thou	shalt	not



# 언어모델 학습

- 다음 단어 예측(Next-word prediction)

Thou shalt not make a machine in the likeness of a human mind

Sliding window across running text

thou	shalt	not	make	a	machine	in	the	...
thou	shalt	not	make	a	machine	in	the	

Dataset

input 1	input 2	output
thou	shalt	not
shalt	not	make

# 언어모델 학습

- 다음 단어 예측(Next-word prediction)

Thou shalt not make a machine in the likeness of a human mind

Sliding window across running text

thou	shalt	not	make	a	machine	in	the	...
thou	shalt	not	make	a	machine	in	the	
thou	shalt	not	make	a	machine	in	the	
thou	shalt	not	make	a	machine	in	the	
thou	shalt	not	make	a	machine	in	the	

Dataset

input 1	input 2	output
thou	shalt	not
shalt	not	make
not	make	a
make	a	machine
a	machine	in

## 언어모델 학습

---

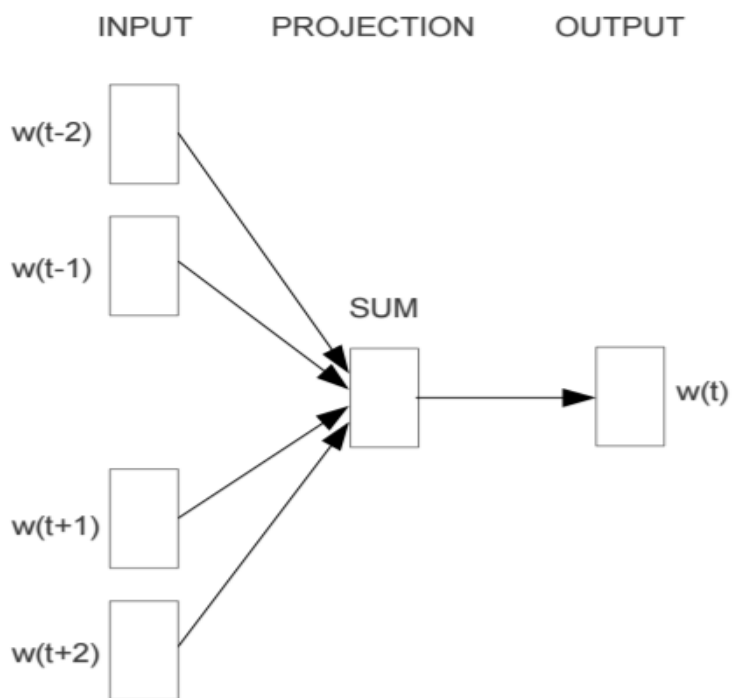
- 양방향으로 볼 필요가 있다. (Look both ways)

Jay was hit by a \_\_\_\_\_

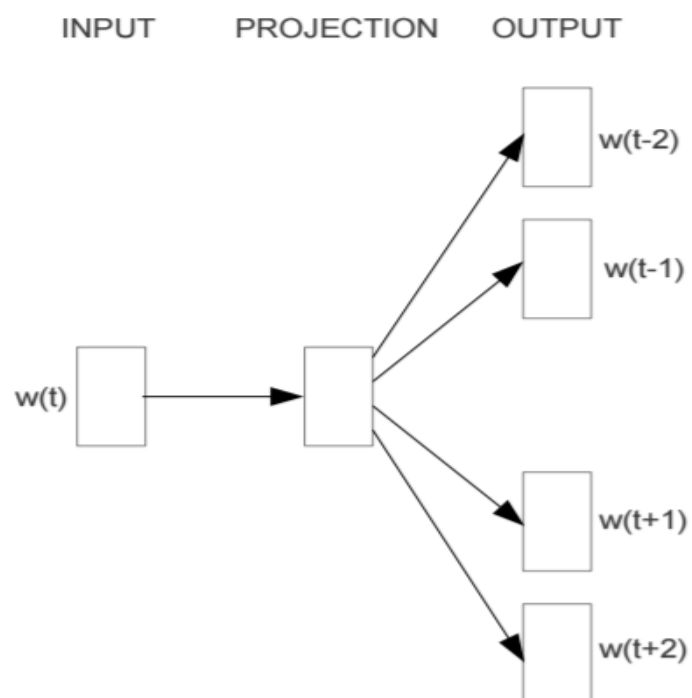
Jay was hit by a \_\_\_\_\_ bus

# 언어모델 학습

- 양방향으로 볼 필요가 있다. (Look both ways)



**CBOW**



**Skip-gram**

# 언어모델 학습

- 연속 단어주머니 (CBOW: Continuous Bag of Words) :이전 두 단어만 보는 것이 아니라 다음 두 단어도 본다.

Jay was hit by a \_\_\_\_\_ bus in...

by	a	red	bus	in
----	---	-----	-----	----

input 1	input 2	input 3	input 4	output
by	a	bus	in	red

# 언어모델 학습

- 스킵 그램 학습 Skip gram

Jay was hit **by a red bus in...**



Jay was hit **by a red bus in...**



input	output
red	by
red	a
red	bus
red	in

- 녹색이 입력 핑크색이 출력이다.

**Thou shalt not make a** machine in the likeness of a human mind



input word	target word

# 언어모델 학습

- 스킵 그램 학습 Skip gram

Thou shalt not make a machine in the likeness of a human mind

thou	shalt	not	make	a	machine	in	the	...
------	-------	-----	------	---	---------	----	-----	-----

input word	target word
not	thou
not	shalt
not	make
not	a

# 언어모델 학습

## ● 스킵 그램 학습 Skip gram

Thou shalt not make a machine in the likeness of a human mind

thou	shalt	not	make	a	machine	in	the	...
------	-------	-----	------	---	---------	----	-----	-----

thou	shalt	not	make	a	machine	in	the	...
------	-------	-----	------	---	---------	----	-----	-----

input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine



# 언어모델 학습

## ● 스킵 그램 학습 Skip gram

Thou shalt not make a machine in the likeness of a human mind

thou	shalt	not	make	a	machine	in	the	...
------	-------	-----	------	---	---------	----	-----	-----

thou	shalt	not	make	a	machine	in	the	...
------	-------	-----	------	---	---------	----	-----	-----

thou	shalt	not	make	a	machine	in	the	...
------	-------	-----	------	---	---------	----	-----	-----

thou	shalt	not	make	a	machine	in	the	...
------	-------	-----	------	---	---------	----	-----	-----

thou	shalt	not	make	a	machine	in	the	...
------	-------	-----	------	---	---------	----	-----	-----

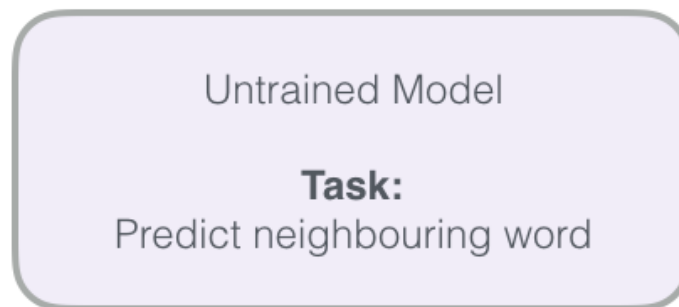
input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine
a	not
a	make
a	machine
a	in
machine	make
machine	a
machine	in
machine	the
in	a
in	machine
in	the
in	likeness

# 학습 프로세스

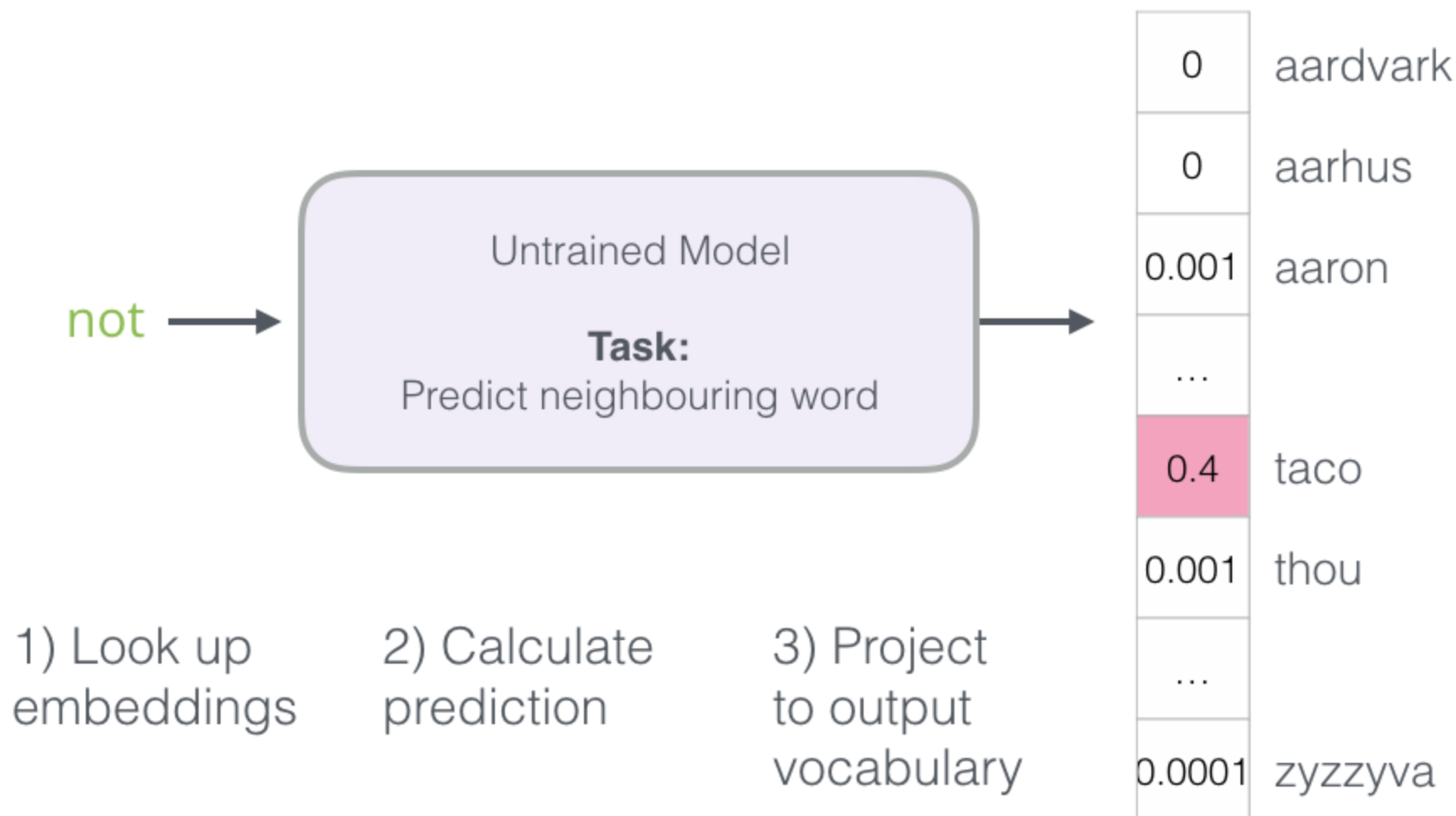
- 스킵 그램 학습 Skip gram

input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine
a	not
a	make
a	machine
a	in
machine	make
machine	a
machine	in
machine	the
in	a
in	machine
in	the
in	likeness

not →



# 학습 프로세스



# 학습 프로세스

Actual  
Target

0
0
0
...
0
1
...
0

-

Model  
Prediction

0	aardvark
0	aarhus
0.001	aaron
...	
0.4	taco
0.001	thou
...	
0.0001	zyzzyva

# 학습 프로세스

Actual  
Target

0
0
0
...
0
1
...
0

-

Model  
Prediction

0	aardvark
0	aarhus
0.001	aaron
...	...
0.4	taco
0.001	thou
...	...
0.0001	zyzzyva

=

Error

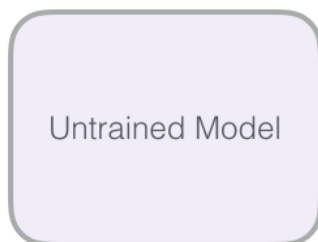
0
0
-0.001
...
-0.4
0.999
...
-0.0001

# 학습 프로세스

Actual  
Target

0
0
0
...
0
1
...
0

not



Model  
Prediction

0	aardvark
0	aarhus
0.001	aaron
...	...
0.4	taco
0.001	thou
...	...
0.0001	zyzzyva

Error

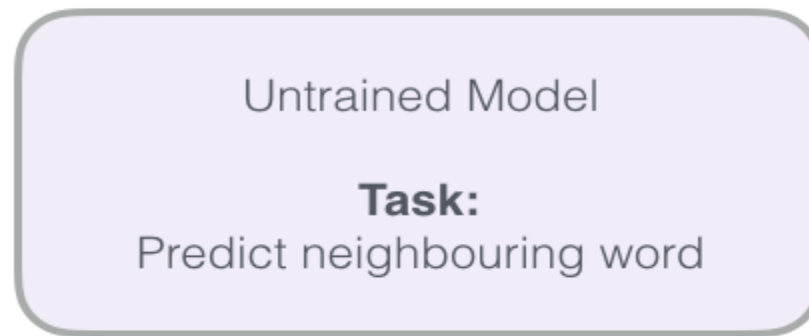
0
0
-0.001
...
-0.4
0.999
...
-0.0001

Update  
Model  
Parameters

# 부정 샘플링 (Negative Sampling)

- 단어집의 모든 단어에 대해서 오차를 계산해야 하는 이 방식은 계산비용이 큼

not



1) Look up embeddings

2) Calculate prediction

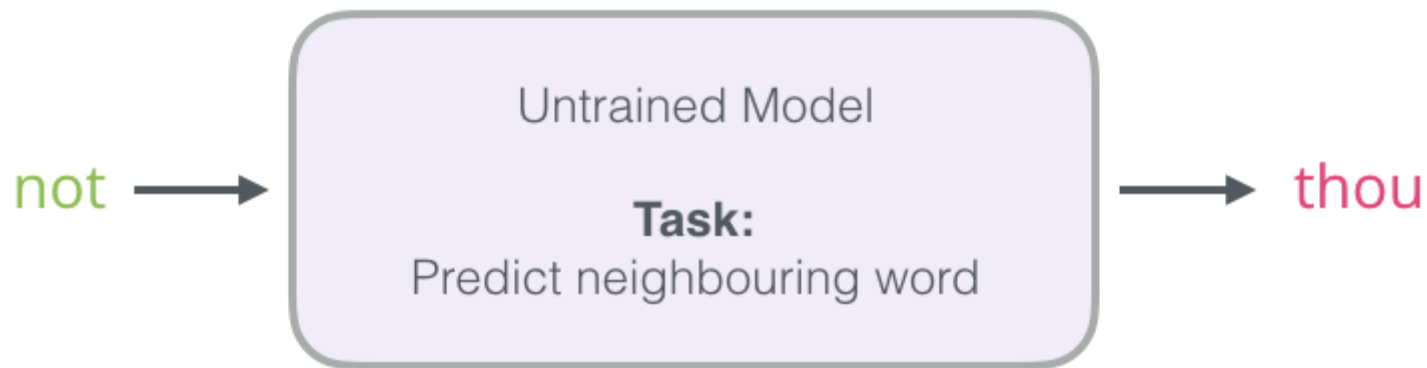
**3) Project to output vocabulary**

**[Computationally Intensive]**

# 부정 샘플링 (Negative Sampling)

- 작업을 좀 바꿔보자: 신경망 -> 로지스틱 회귀로

Change Task from

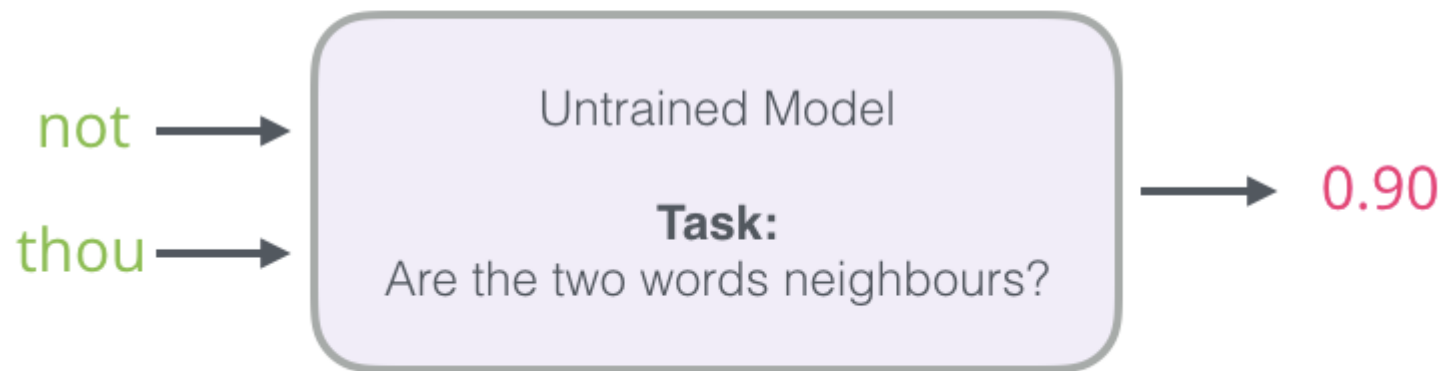




## 부정 샘플링 (Negative Sampling)

- 작업을 좀 바꿔보자: 신경망 -> 로지스틱 회귀로

To:



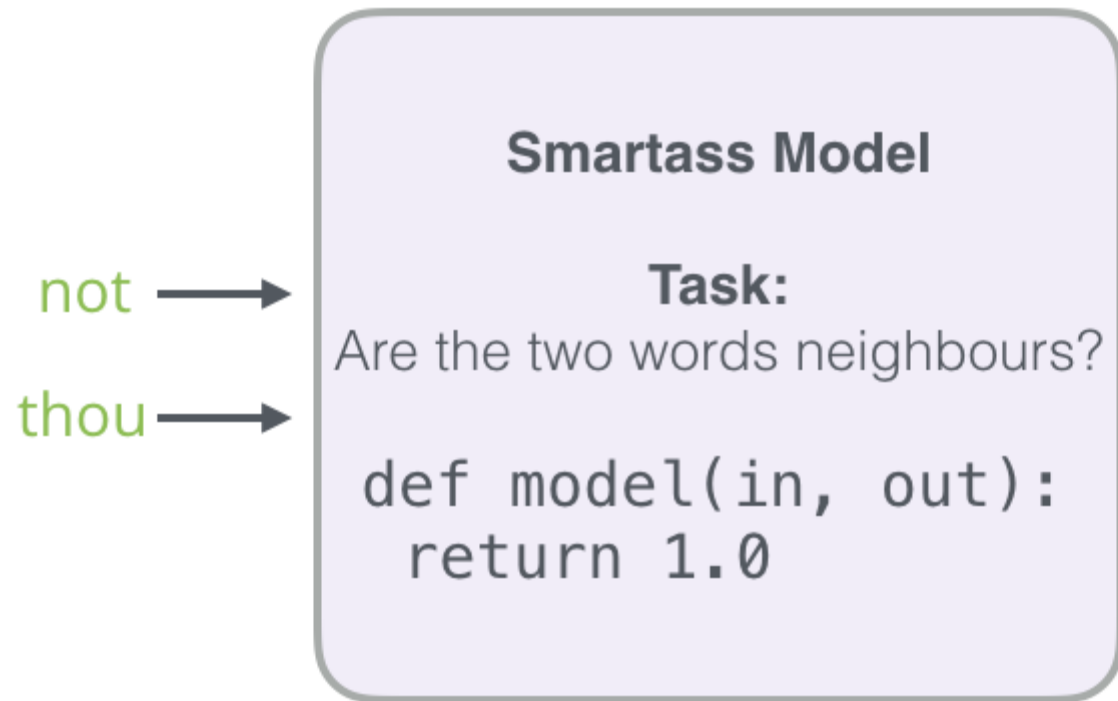
# 부정 샘플링 (Negative Sampling)

- 이웃에 있는 단어에 대해서 1

input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine

input word	output word	target
not	thou	1
not	shalt	1
not	make	1
not	a	1
make	shalt	1
make	not	1
make	a	1
make	machine	1

# 부정 샘플링 (Negative Sampling)



## 부정 샘플링 (Negative Sampling)

- 이웃이 아닌 단어에 대해서 0를 할당

input word	output word	target
not	thou	1
not		0
not		0
not	shalt	1
not	make	1

 Negative examples

# 부정 샘플링 (Negative Sampling)

- 이웃이 아닌 단어에 대해서 0를 할당
  - 이웃이 아닌 단어를 단어집에서 랜덤추출

Pick randomly from vocabulary  
(random sampling)

input word	output word	target
not	thou	1
not	aaron	0
not	taco	0
not	shalt	1
not	make	1

Word Count Probability

aardvark

aarhus

aaron

taco

thou

zyzzyva

# 부정 샘플링 (Negative Sampling) Skipgram (SGNS)

- 부정 샘플링 + 스킵그램

## Skipgram

shalt	not	make	a	machine
-------	-----	------	---	---------

input	output
make	shalt
make	not
make	a
make	machine

## Negative Sampling

input word	output word	target
make	shalt	1
make	aaron	0
make	taco	0

**word2vec**

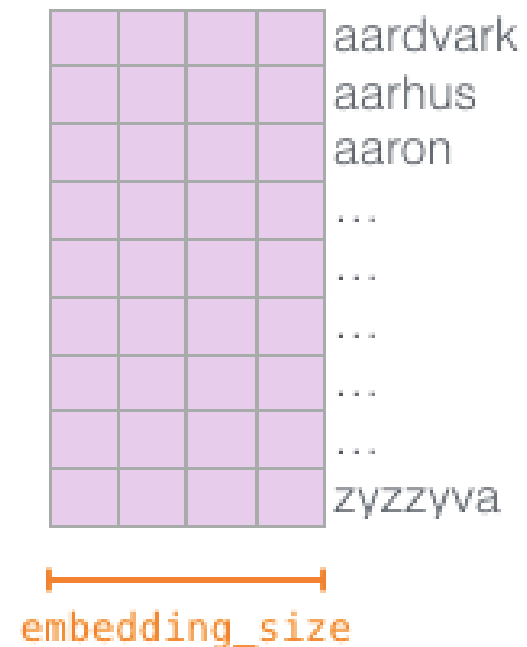
# Word2Vec 학습 프로세스

- 2개의 임베딩 행렬을 생성한다. embedding\_size는 일반적으로 300

## Embedding



## Context





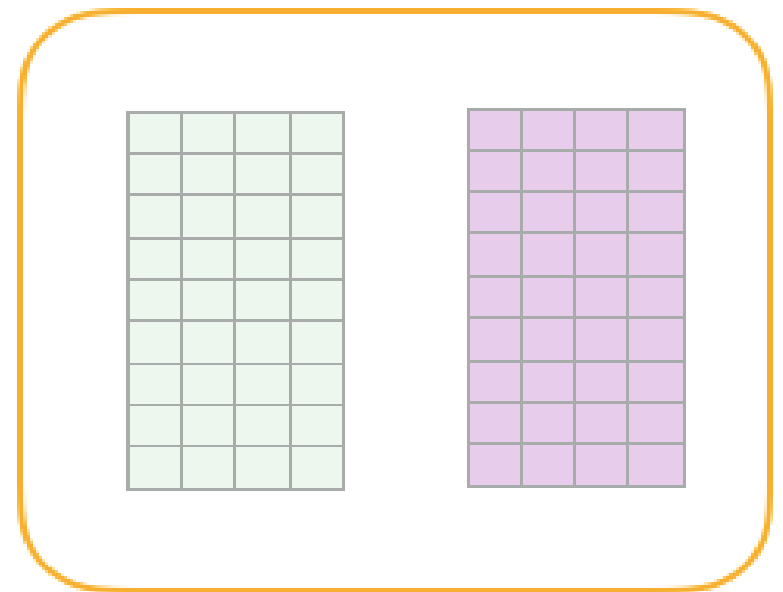
# Word2Vec 학습 프로세스

- 시작은 랜덤넘버로 시작

dataset

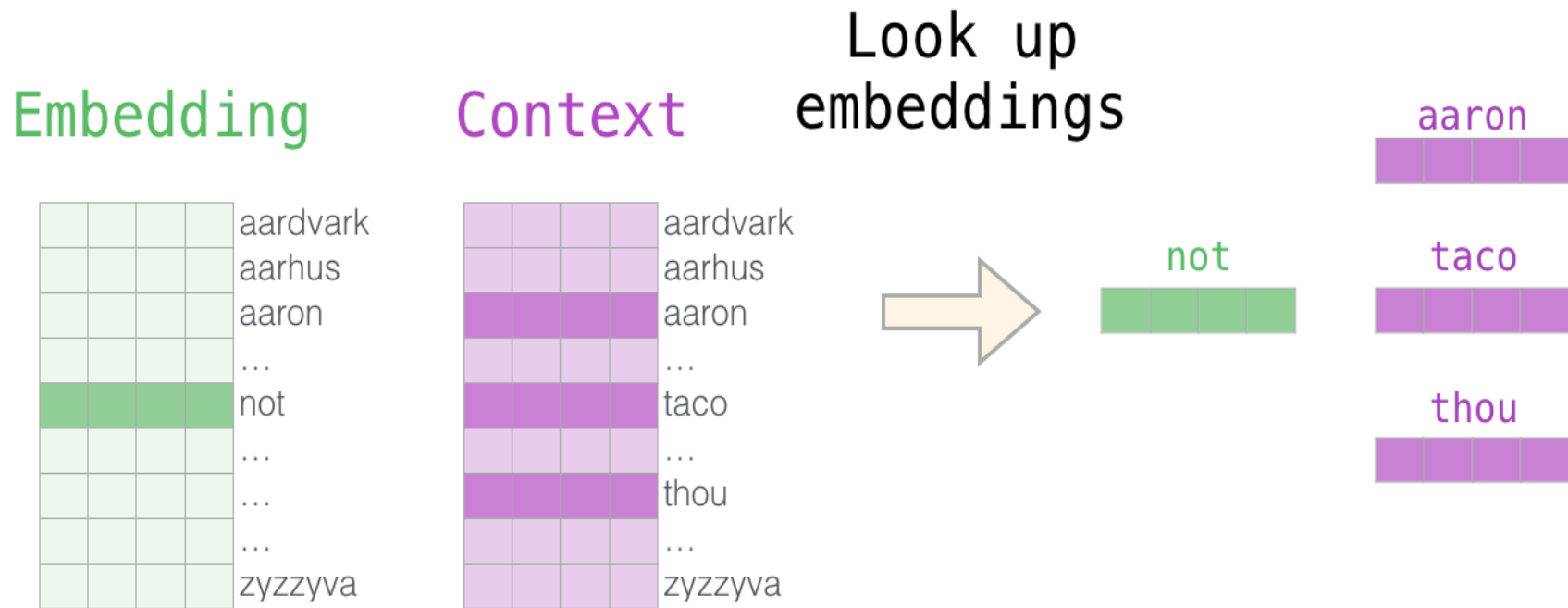
input word	output word	target
not	thou	1
not	aaron	0
not	taco	0
not	shalt	1
not	mango	0
not	finglonger	0
not	make	1
not	plumbus	0
...	...	...

model









# Word2Vec 학습프로세스

- 임베딩 행렬에서 입력단어 1개, 콘텍스트 행렬에서 타겟단어 3개 추출



## Word2Vec 학습 프로세스

- 유사도를 측정하기 위해 점곱을 계산 (코사인유사도를 상기하라.)

input word	output word	target	input • output
not 	thou 	1	0.2
not 	aaron 	0	-1.11
not 	taco 	0	0.74

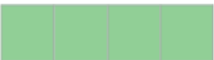
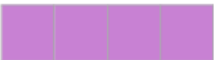
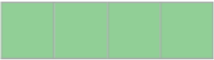

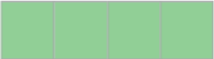

## Word2Vec 학습 프로세스

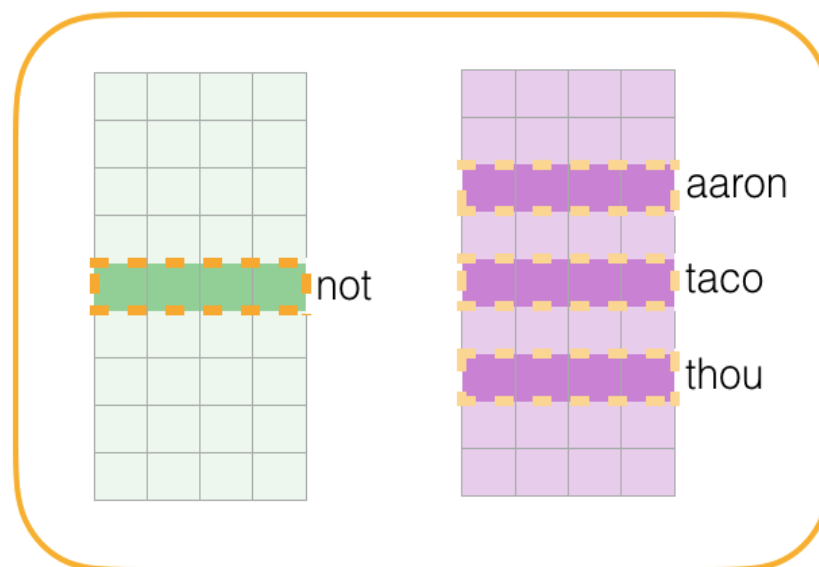
- 유사도 결과에 시그모이드를 적용해 확률을 계산하고, 다음 오차를 계산한다.
  - 오차는 타겟 - 시그모이드이다.

input word	output word	target	input • output	sigmoid()
not 	thou 	1	0.2	0.55
not 	aaron 	0	-1.11	0.25
not 	taco 	0	0.74	0.68

# Word2Vec 학습 프로세스

- 오차를 최소화하기 위해 선택된 단어들의 임베딩을 조정한다.

input word	output word	target	input • output	sigmoid()	Error
not 	thou 	1	0.2	0.55	0.45
not 	aaron 	0	-1.11	0.25	-0.25
not 	taco 	0	0.74	0.68	-0.68



Update  
Model  
Parameters

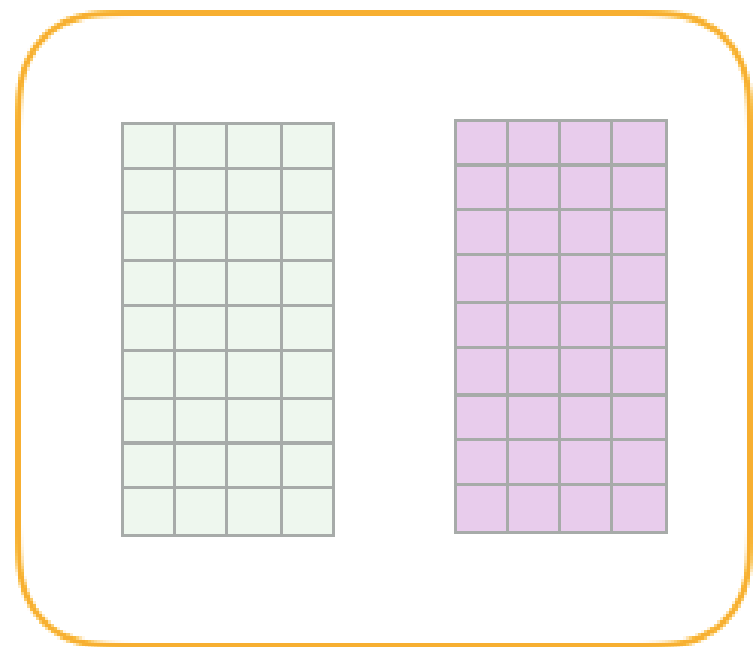
# Word2Vec 학습 프로세스

- 다음 단어셋(not shalt mango, finglonger)에 대해서 이 프로세스를 반복한다.

dataset

input word	output word	target
not	thou	1
not	aaron	0
not	taco	0
not	shalt	1
not	mango	0
not	finglonger	0
not	make	1
not	plumbus	0
...	...	...

model



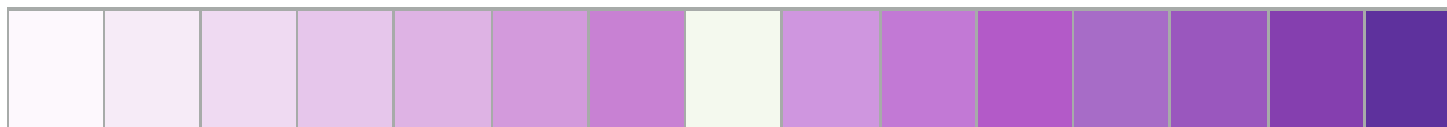
# 윈도우 크기와 부정 샘플링 갯수

- 중요한 하이퍼 파라미터가 윈도우크기와 부정 샘플링 갯수이다.
- 윈도우 크기 (5-15) 15-50도 사용하지만, 젠심은 5을 기본으로 한다.  
윈도우 크기가 작으면 인접 단어가 교환가능한 단어(동의어 뿐 아니라 반대말도) 찾게 되며, 큰 윈도우는 관련언어를 찾게 된다.

Window size: 5



Window size: 15



## 위도우 크기와 부정 샘플링 갯수

- 부정 샘플링 갯수 (5-20) 하지만 2-5도 나쁘지 않으며 Gensim은 5를 기본으로 한다.

Negative samples: 2

input word	output word	target
make	shalt	1
make	aaron	0
make	taco	0

Negative samples: 5

input word	output word	target
make	shalt	1
make	aaron	0
make	taco	0
make	finglonger	0
make	plumbus	0
make	mango	0



# 참고문헌

---

- [Distributed Representations of Words and Phrases and their Compositionality](#) [pdf]
- [Efficient Estimation of Word Representations in Vector Space](#) [pdf]
- [A Neural Probabilistic Language Model](#) [pdf]
- [Speech and Language Processing](#) by Dan Jurafsky and James H. Martin is a leading resource for NLP. Word2vec is tackled in Chapter 6.
- [Neural Network Methods in Natural Language Processing](#) by [Yoav Goldberg](#) is a great read for neural NLP topics.
- [Chris McCormick](#) has written some great blog posts about Word2vec. He also just released [The Inner Workings of word2vec](#), an E-book focused on the internals of word2vec.
- Want to read the code? Here are two options:
  - Gensim's [python implementation](#) of word2vec
  - Mikolov's original [implementation in C](#) – better yet, this [version with detailed comments](#) from Chris McCormick.
- [Evaluating distributional models of compositional semantics](#)
- [On word embeddings, part 2](#)
- [Dune](#)