

Natural Language Processing with Deep Learning

IFT6289, Winter 2022

Lecture 4: Word Meaning and Word Embedding
Bang Liu

2 Lecture outline

1. Represent the Meaning of a Word
2. Word Embeddings: Word2Vec and GloVe
3. Word Vector Evaluation

3 Certain Slides Adapted From or Referred To...

- ◎ **Stanford CS224n - Natural Language Processing with Deep Learning, Chris Manning**
 - Winter 2020: <http://web.stanford.edu/class/cs224n/>, lecture 1, 2
- ◎ **NTU S-108 Applied Deep Learning, Yun-Nung (Vivian) Chen**
 - Spring 2020: <https://www.csie.ntu.edu.tw/~miulab/s108-adl/syllabus>, lecture 3, 5
- ◎ <http://jalammar.github.io/illustrated-word2vec/>
- ◎ <https://ruder.io/word-embeddings-softmax/>

**Represent the
Meaning of a Word**



5 Meaning Representations

- ◎ Definition of “Meaning”
 - the idea that is represented by a word, phrase, etc.
 - the idea that a person wants to express by using words, signs, etc.
 - the idea that is expressed in a work of writing, art, etc.

6 Meaning Representations in Computers

Knowledge-Based Representation



Corpus-Based Representation

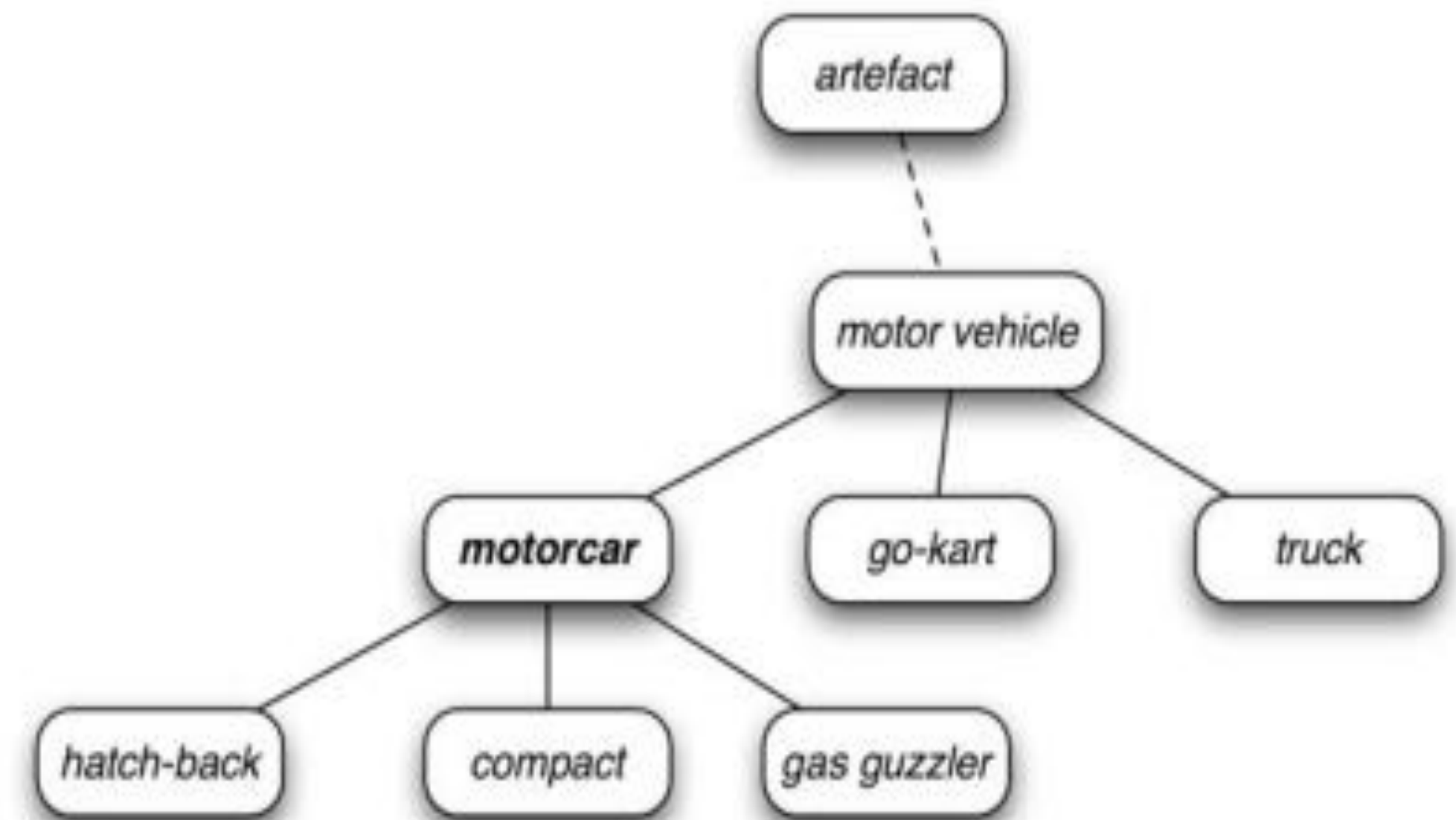


7 Knowledge-Based Representations

- Hypernyms (is-a) relationships of WordNet

```
from nltk.corpus import wordnet as wn
panda = wn.synset('panda.n.01')
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```



Issues:

- newly-invented words
- subjective
- annotation effort
- difficult to compute word similarity

8 Corpus-Based Representations

- Atomic symbols: **one-hot** representation

car [0 0 0 0 0 0 1 0 0 ... 0]
motorcycle [0 0 1 0 0 0 0 0 0 ... 0]

Issues: difficult to compute the similarity (i.e. comparing “car” and “motorcycle”)

Idea: words with similar meanings often have similar neighbors

9 Corpus-Based Representations

- ◎ Neighbor-based representation
 - Co-occurrence matrix constructed via neighbors
 - Neighbor definition: **full document** v.s. **windows**

full document

word-document co-occurrence matrix gives general topics
→ “Latent Semantic Analysis”

windows

context window for each word
→ capture syntactic (e.g. POS) and semantic information

10 Window-Based Co-occurrence Matrix

Example

- Window length=1
- Left or right context
- Corpus:

I love AI.
I love deep learning.
I enjoy learning.

similarity > 0

Counts	I	love	enjoy	AI	deep	learning
I	0	2	1	0	0	0
love	2	0	0	1	1	0
enjoy	1	0	0	0	0	1
AI	0	1	0	0	0	0
deep	0	1	0	0	0	1
learning	0	0	1	0	1	0

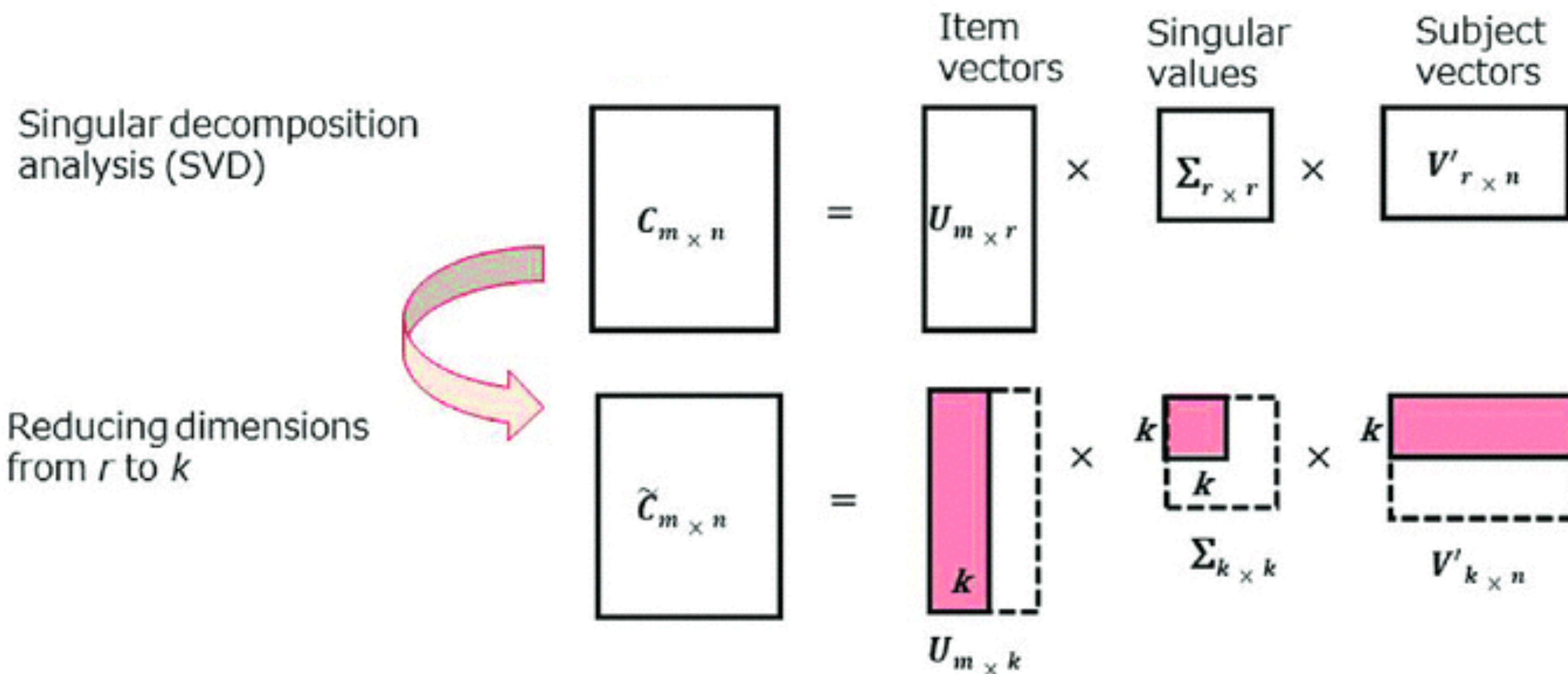
Issues:

- matrix size increases with vocabulary
- high dimensional
- sparsity → poor robustness

Idea: low dimensional word vector

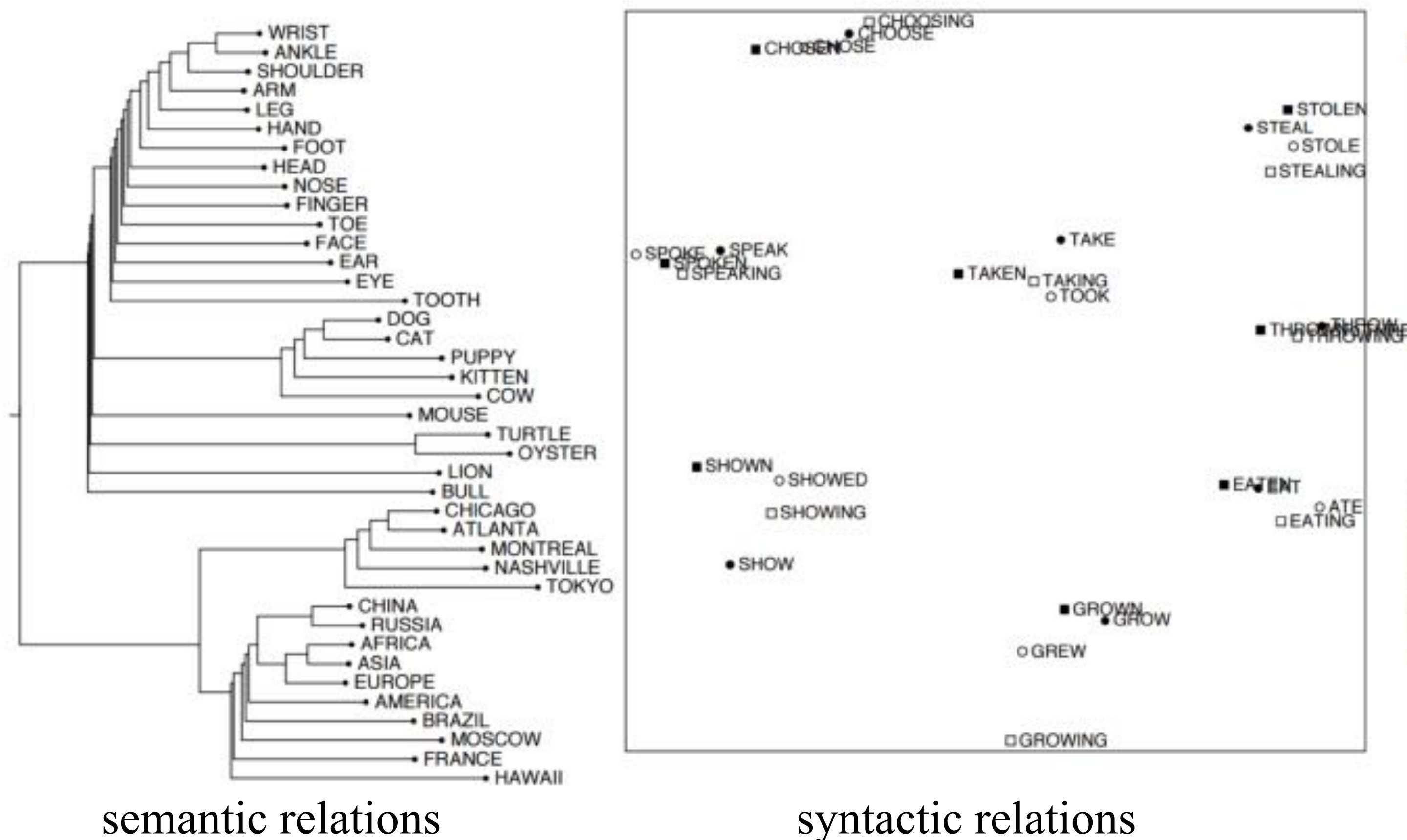
11 Low-Dimensional Dense Word Vector

- Method 1: dimension reduction on the matrix
- Singular Value Decomposition (SVD) of co-occurrence matrix X



12 Low-Dimensional Dense Word Vector

- Method 1: dimension reduction on the matrix
- Singular Value Decomposition (SVD) of co-occurrence matrix X



Issues:

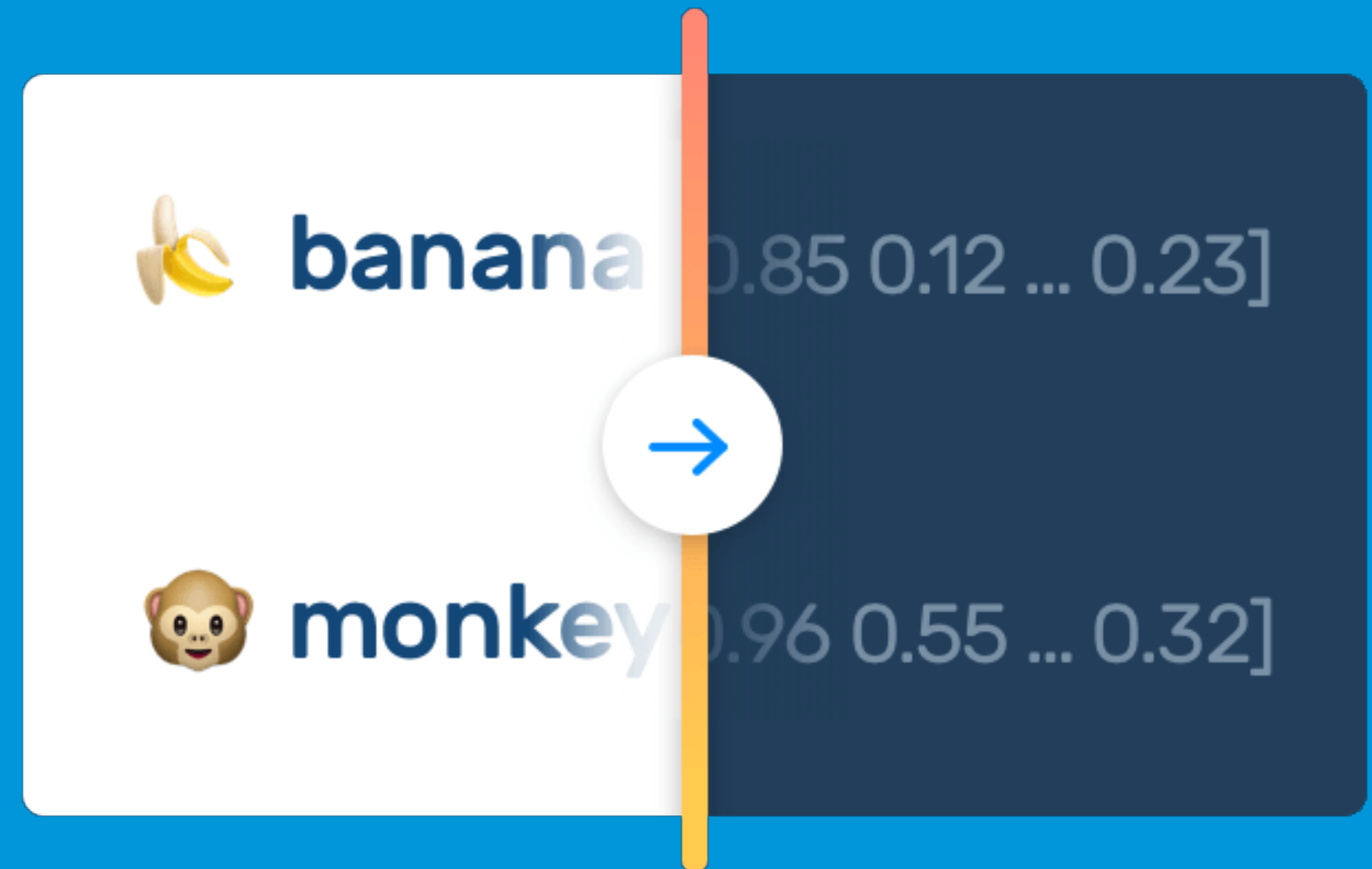
- computationally expensive: $O(mn^2)$ when $n < m$ for $n \times m$ matrix
- difficult to add new words

Idea: directly learn low-dimensional word vectors

13 Low-Dimensional Dense Word Vector

- ◎ Method 2: directly learn low-dimensional word vectors
 - Learning representations by back-propagation. (Rumelhart et al., 1986)
 - A neural probabilistic language model (Bengio et al., 2003)
 - NLP (almost) from Scratch (Collobert & Weston, 2008)
 - Most popular models: word2vec (Mikolov et al. 2013) and Glove (Pennington et al., 2014) (as known as “Word Embeddings ”)

Word Embedding



What are you like?
Personality Embedding

16 Big Five Personality Trait Test

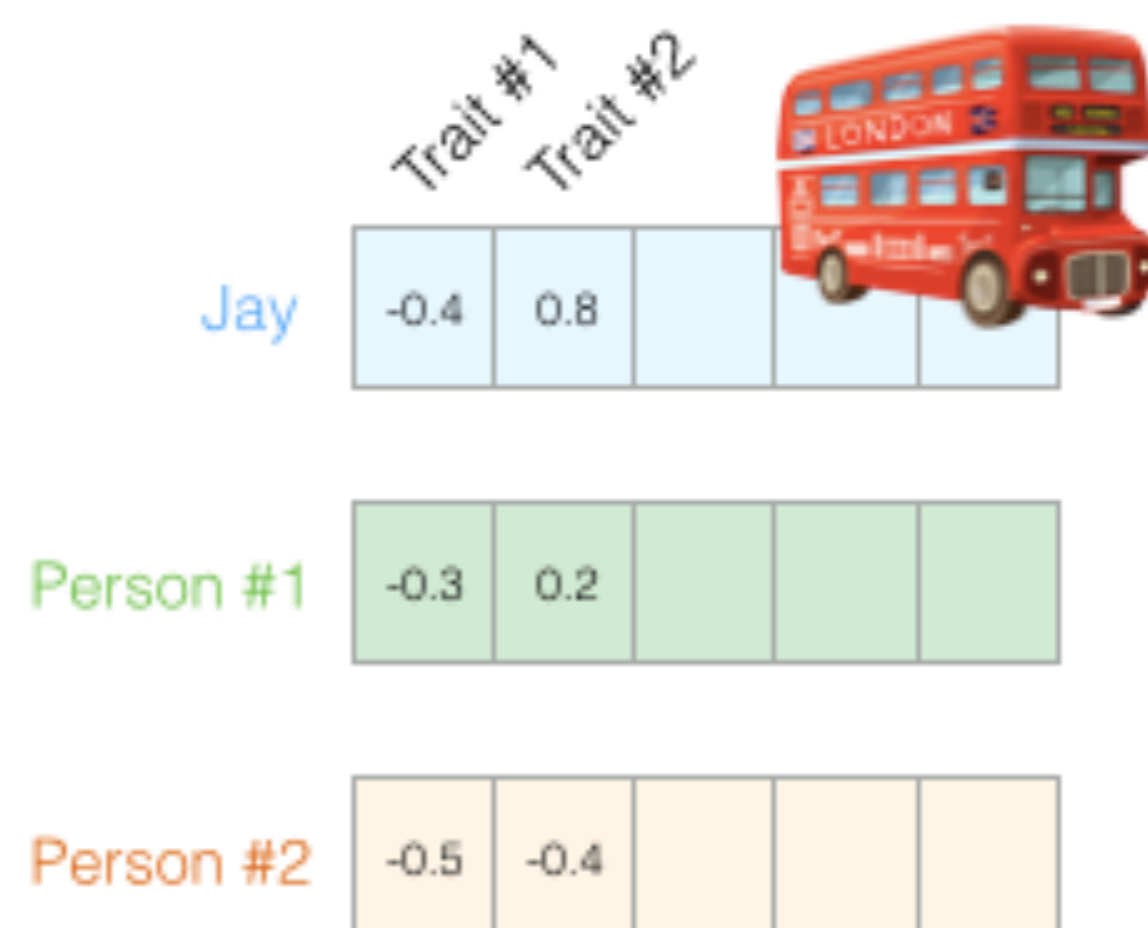
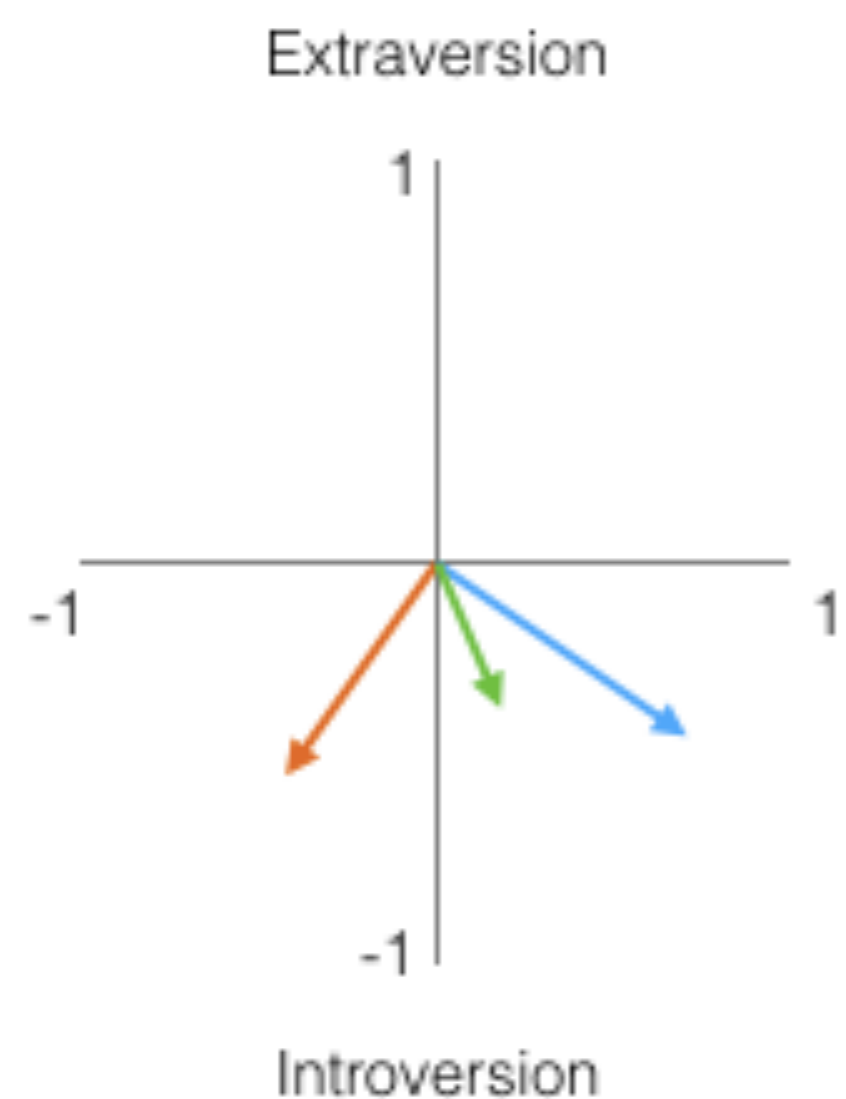
- On a scale of 0 to 100, how introverted/extraverted are you (where 0 is the most introverted, and 100 is the most extraverted)?

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

Example of the result of a Big Five Personality Trait test. It can really tell you a lot about yourself and is shown to have predictive ability in [academic](#), [personal](#), and [professional success](#).

17 Which Person is More Similar?

- Let's switch the range to be from -1 to 1. Say Jay get hit by a bus and Jay need to be replaced by someone with a similar personality. In the following figure, which of the two people is more similar to Jay?



$$\text{cosine_similarity}(\begin{matrix} \text{Jay} \\ -0.4 & 0.8 \end{matrix}, \begin{matrix} \text{Person \#1} \\ -0.3 & 0.2 \end{matrix}) = 0.87 \quad \checkmark$$

$$\text{cosine_similarity}(\begin{matrix} \text{Jay} \\ -0.4 & 0.8 \end{matrix}, \begin{matrix} \text{Person \#2} \\ -0.5 & -0.4 \end{matrix}) = -0.20$$

Which Person is More Similar?

- Let's use all five dimensions in our comparison. Which of the two people is more similar to Jay?

	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

$$\text{cosine_similarity}(\text{Jay}, \text{Person \#1}) = 0.66 \checkmark$$

$$\text{cosine_similarity}(\text{Jay}, \text{Person \#2}) = -0.37$$

19 Central Idea: Represent Things by Vectors

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

**Words can also be
Represented by Vectors:
Word Embedding**

Word Embedding

```
[ 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 ]
```

“King”

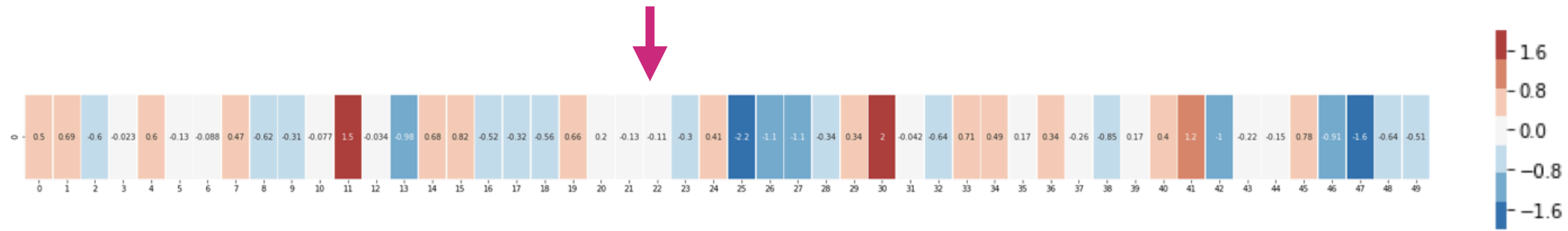
This is a word embedding for the word “king” (GloVe vector trained on Wikipedia).

Visualize Word Embedding

- Let's color code the cells based on their values (red if they're close to 2, white if they're close to 0, blue if they're close to -2)

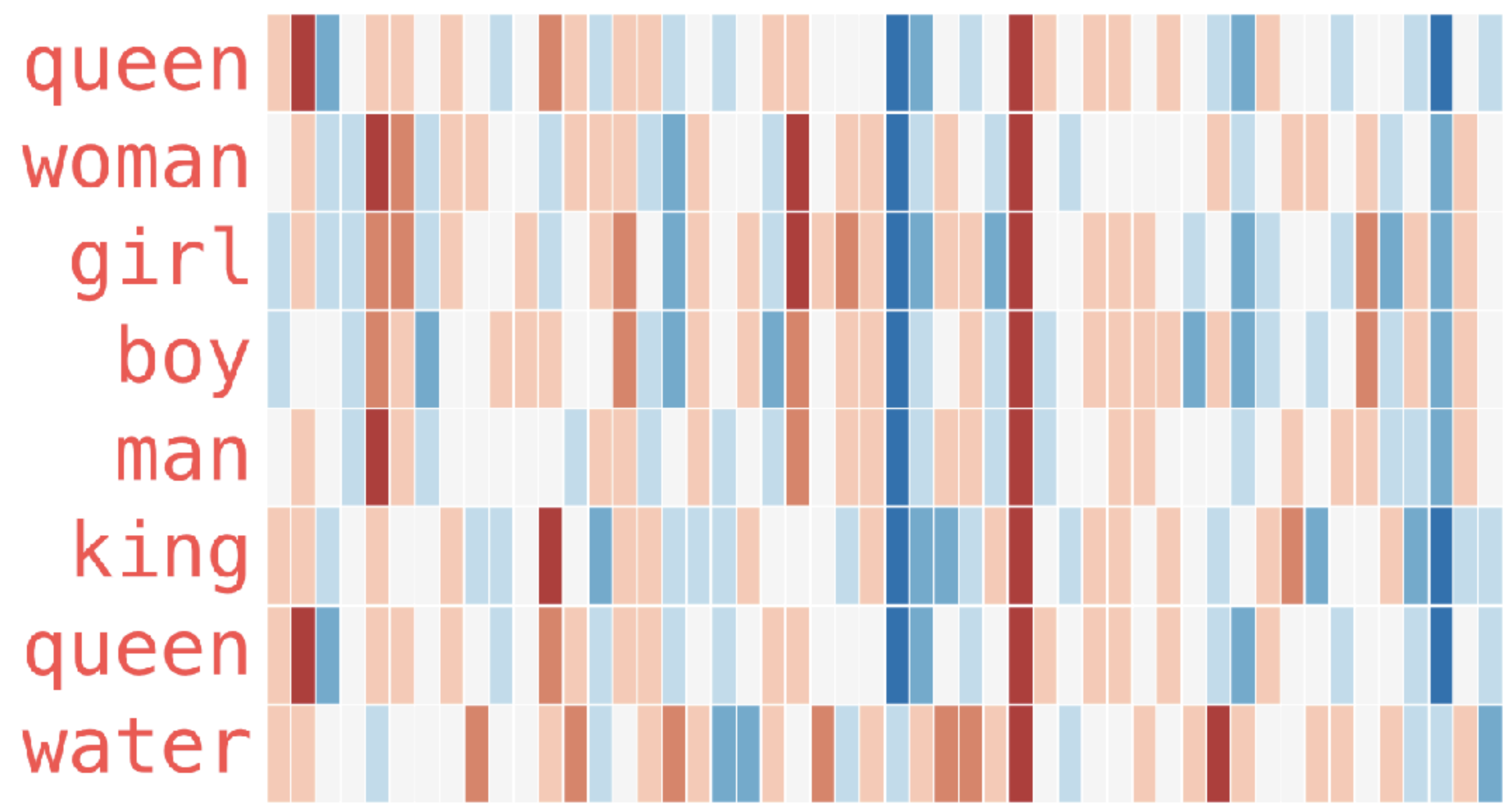
```
[ 0.50451 , 0.68607 , -0.59517 , -0.022801, 0.60046 ,
-0.13498 , -0.08813 , 0.47377 , -0.61798 , -0.31012 ,
-0.076666, 1.493 , -0.034189, -0.98173 , 0.68229 ,
0.81722 , -0.51874 , -0.31503 , -0.55809 , 0.66421 ,
0.1961 , -0.13495 , -0.11476 , -0.30344 , 0.41177 ,
-2.223 , -1.0756 , -1.0783 , -0.34354 , 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 ]
```

“King”

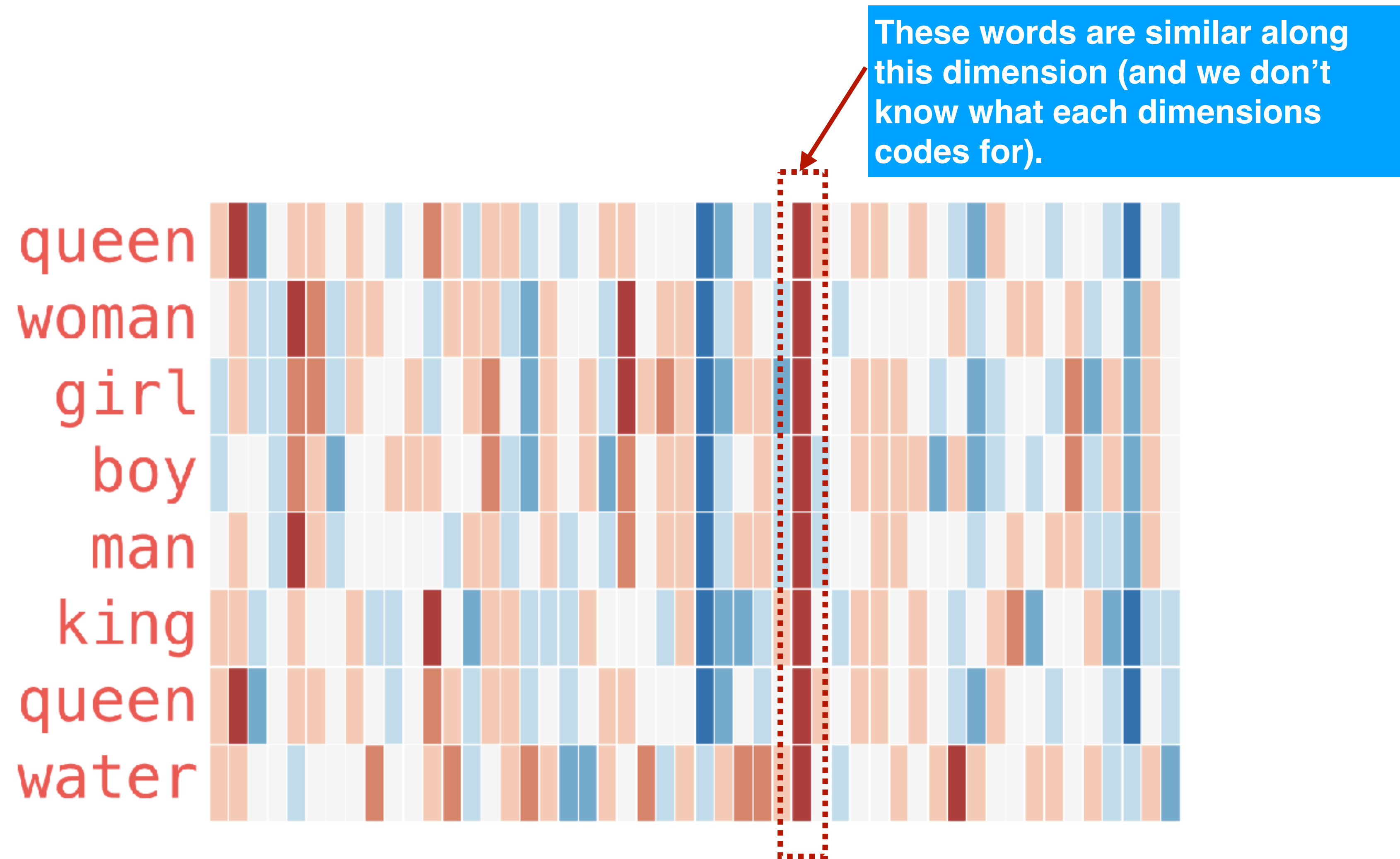


Compare Word Embeddings

- A list of examples (compare by vertically scanning the columns looking for columns with similar colors).

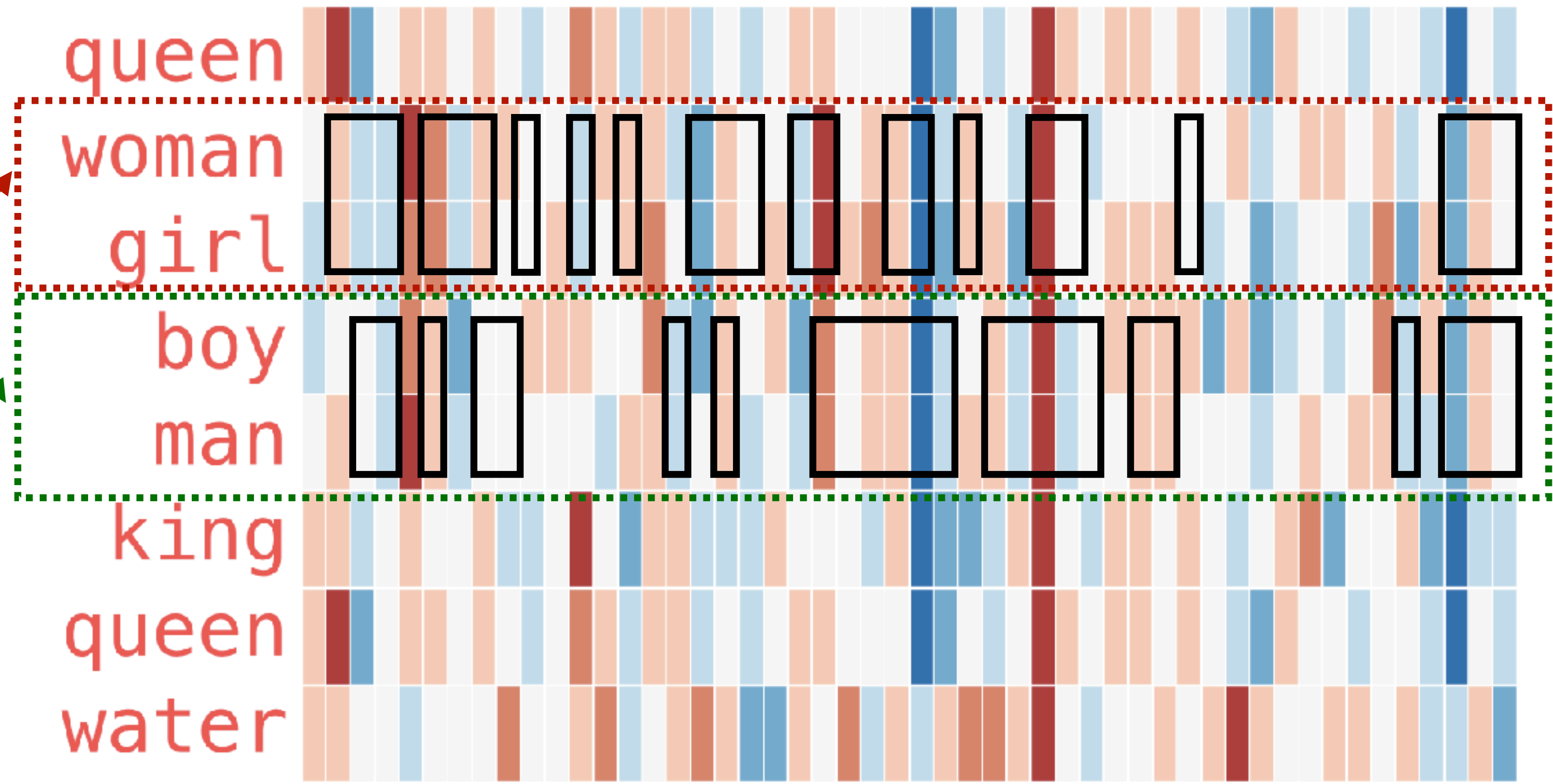


Compare Word Embeddings



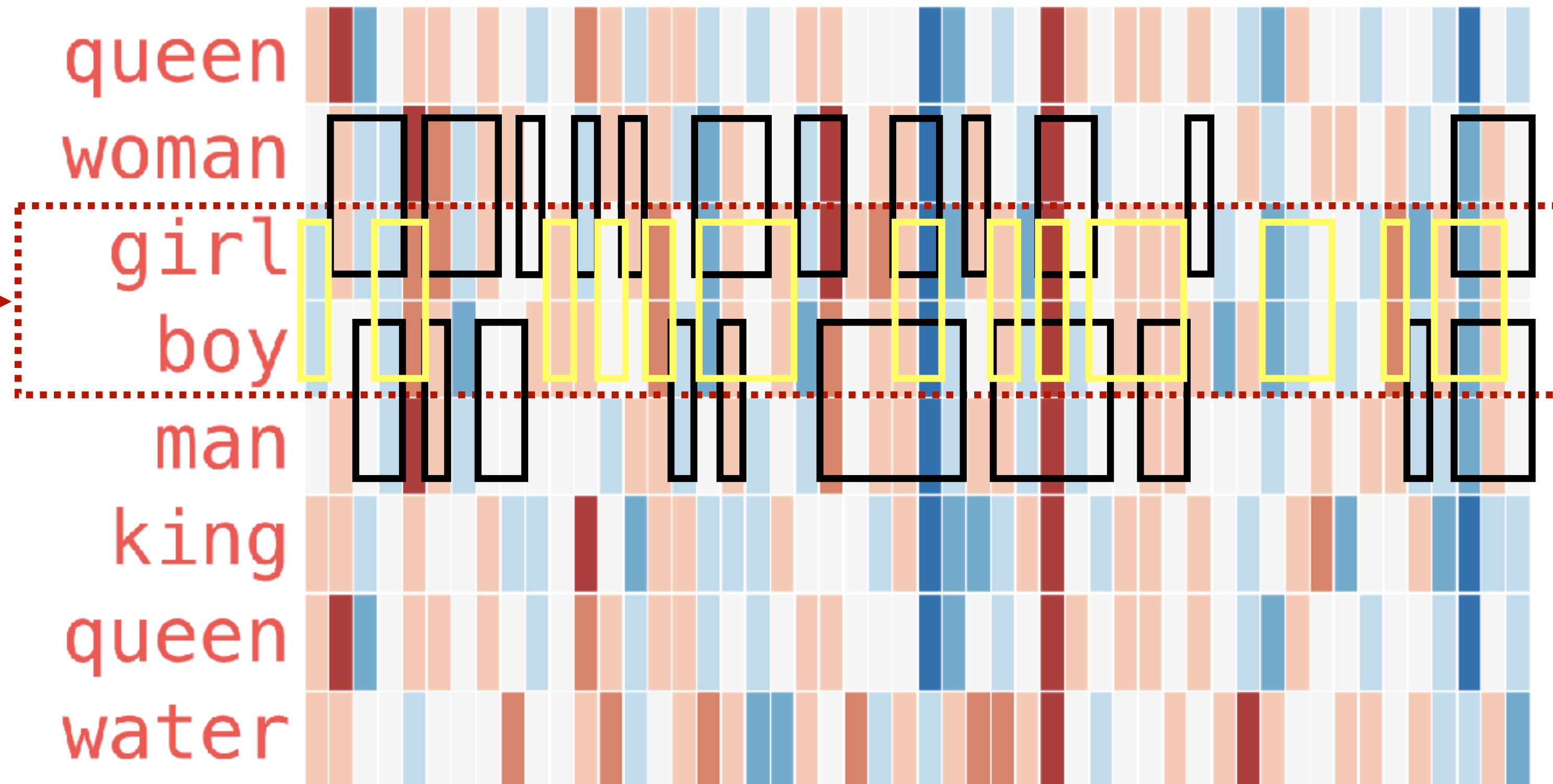
Compare Word Embeddings

“woman” and “girl” are similar to each other in a lot of places. The same with “man” and “boy”.



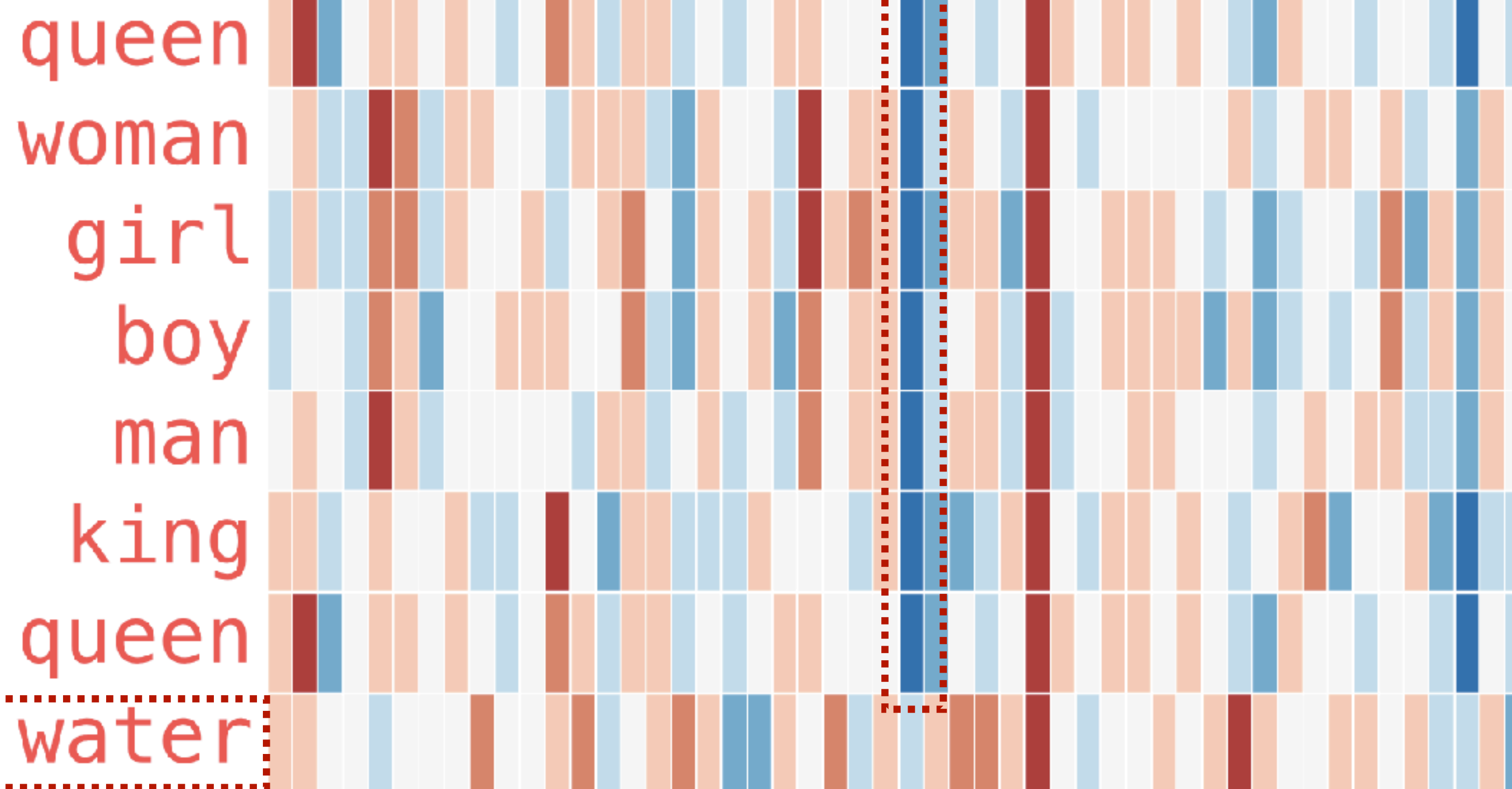
26 Compare Word Embeddings

“boy” and “girl” also have places where they are similar to each other, but different from “woman” or “man”. Could these be coding for a vague conception of youth? possible.



27 Compare Word Embeddings

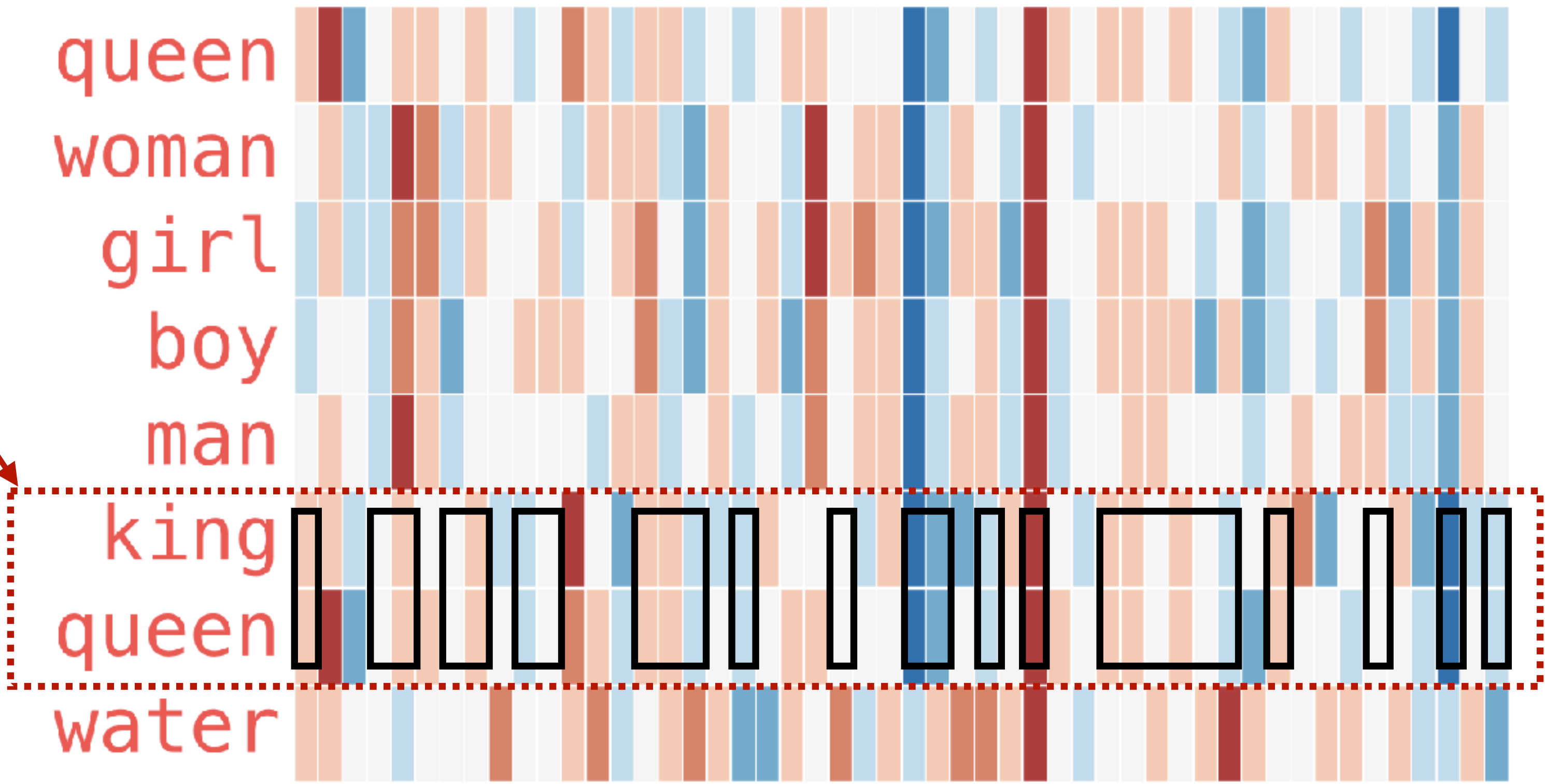
Different category!



This column goes all the way down and stops before the embedding for "water".

28 Compare Word Embeddings

There are clear places where “king” and “queen” are similar to each other and distinct from all the others. Could these be coding for a vague concept of royalty?



Compare Word Embeddings

$$\text{king} - \text{man} + \text{woman} \approx \text{queen}$$



The resulting vector from "king-man+woman" doesn't exactly equal "queen", but "queen" is the closest word to it from the 400,000 word embeddings we have in this collection.

Compare Word Embeddings

```
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)]
```

Using the Gensim library in python, we can add and subtract word vectors, and it would find the most similar words to the resulting vector. The image shows a list of the most similar words with “king+woman-man”, each with its cosine similarity.

How to Train Word Embeddings?

Recall Language Modeling

32 Recall Language Modeling

- A **language model** can take a list of words (let's say two words), and attempt to predict the word that follows them.



input/feature #1

input/feature #2

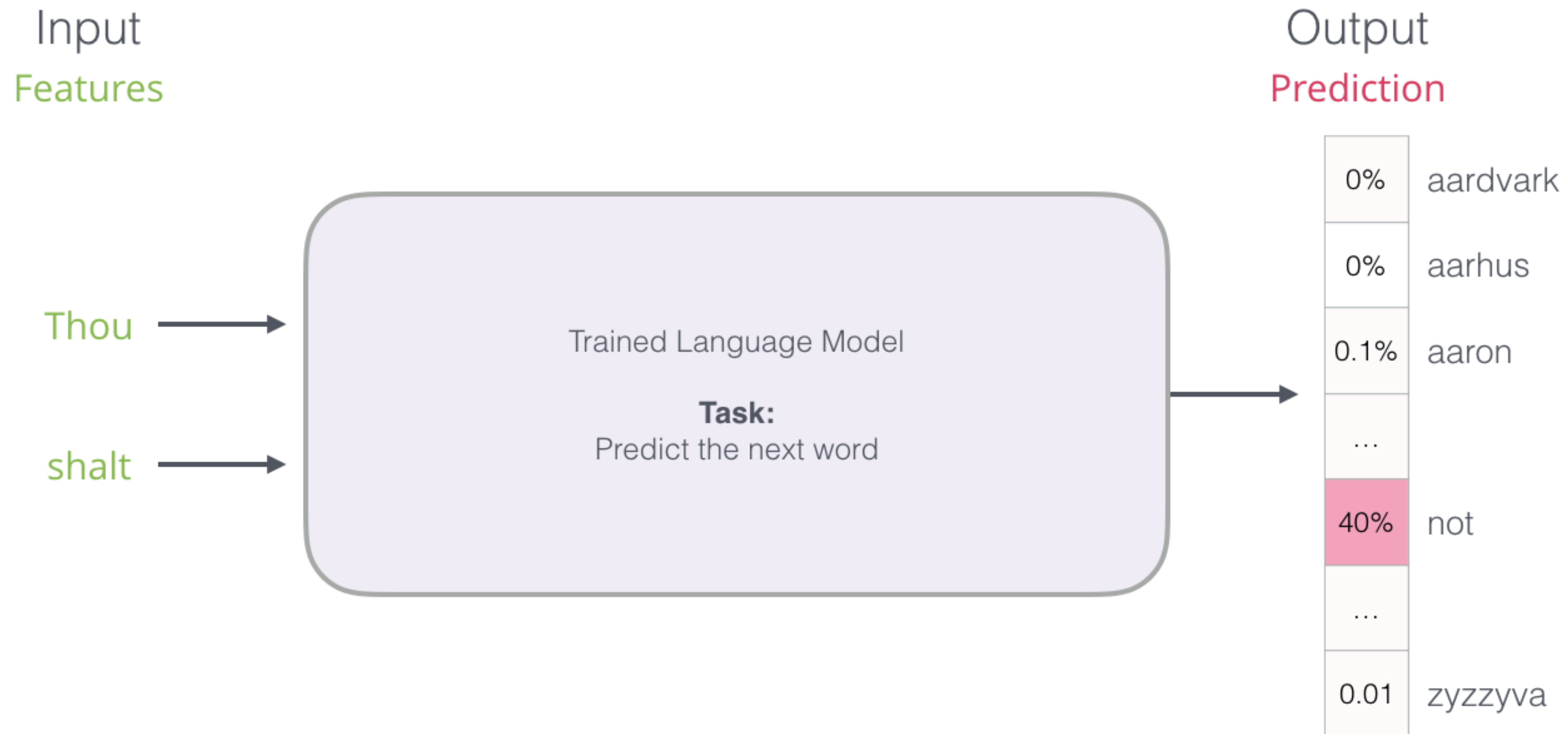
output/label

Thou shalt



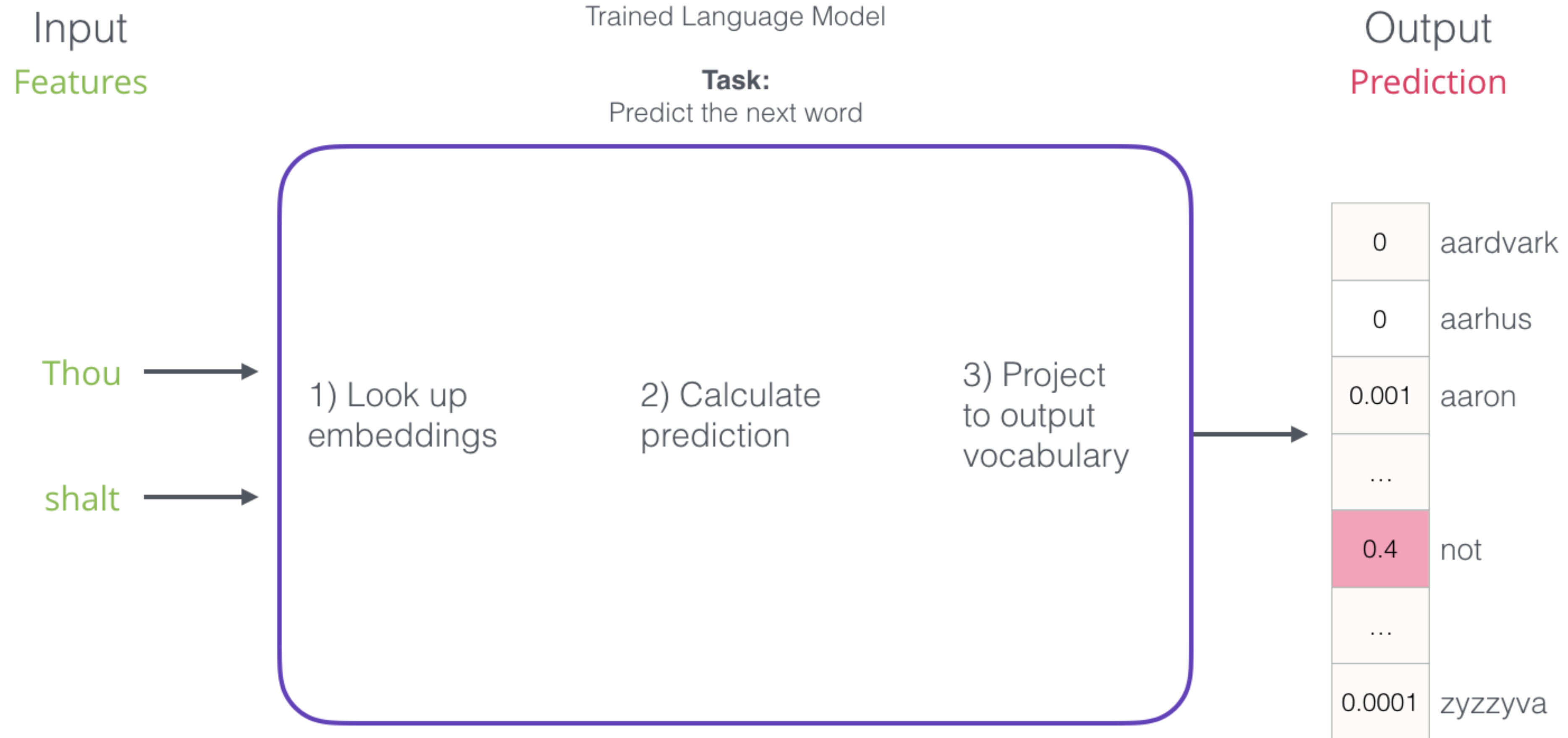
33 Recall Language Modeling

- A language model actually outputs a probability score for all the words it knows (the model's "vocabulary")



Recall Language Modeling

- After being trained, early neural language models (Bengio 2003) would calculate a prediction in three steps:



35 Recall Language Modeling

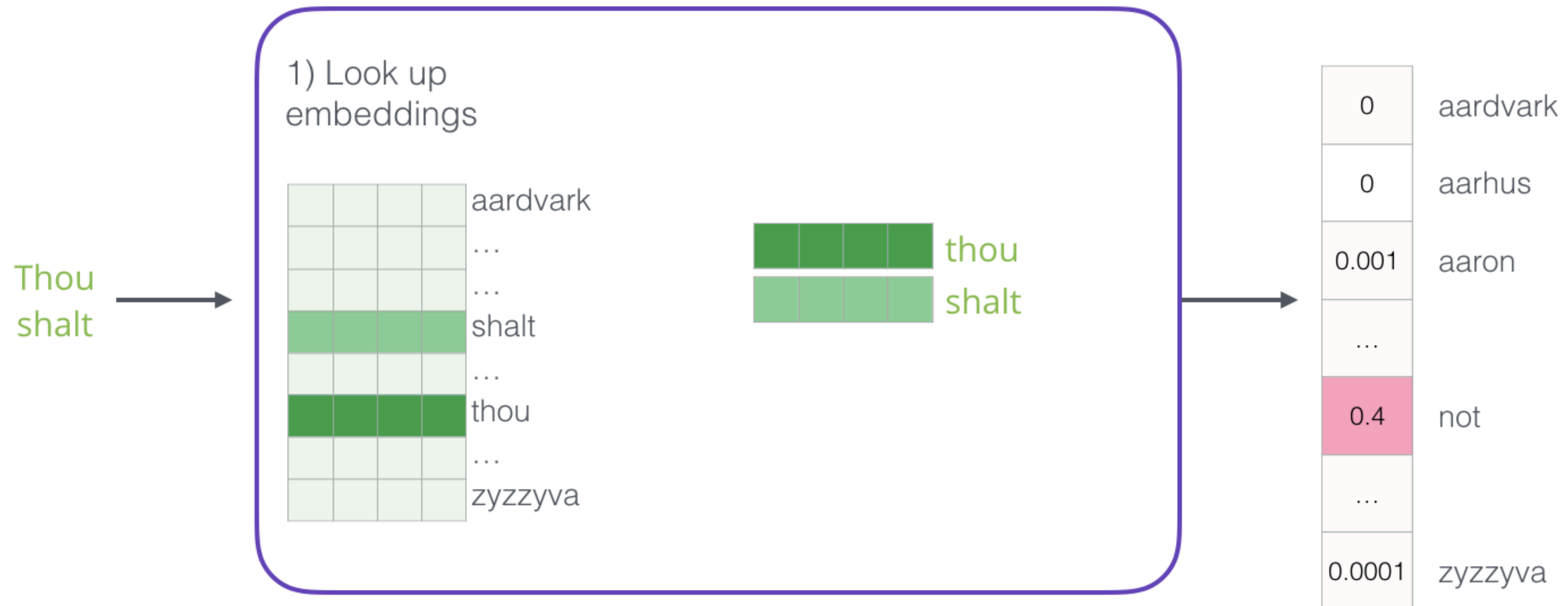
- In the first step, we get a **matrix that contains an embedding for each word** in our vocabulary.

Input
Features

Trained Language Model

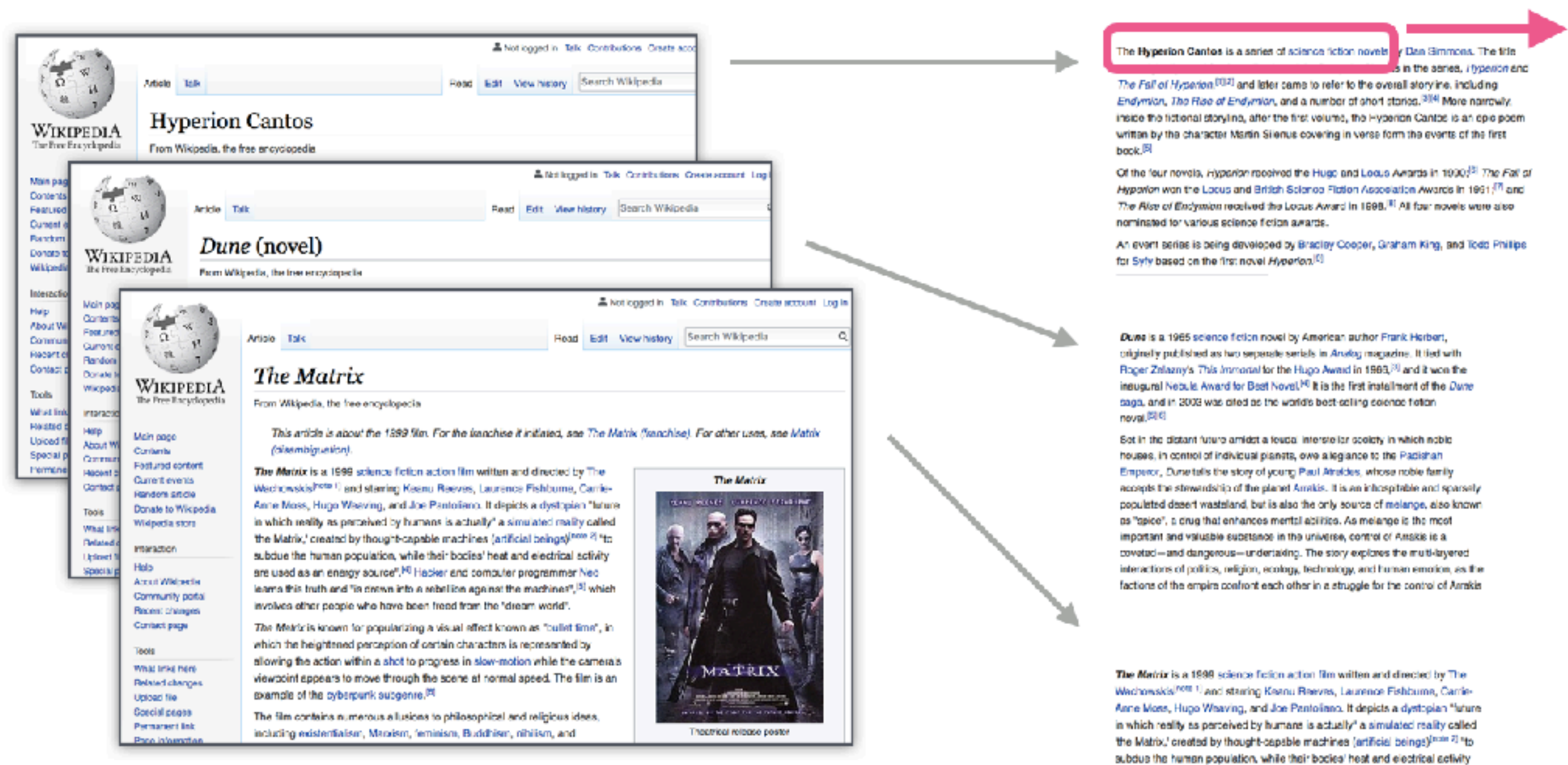
Output
Prediction

Task:
Predict the next word



Language Model Training

- Words get their embeddings by looking at which other words they tend to appear next to.
 - We get a lot of text data (say, all Wikipedia articles, for example).
 - We have a window (say, of three words) that we slide against all of that text.
 - The sliding window generates training samples for our model



We take the first two words to be features, and the third word to be a label:



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

We then slide our window to the next position and create a second sample:



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

And pretty soon we have a larger dataset of which words tend to appear after different pairs of words:



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

**From Language Modeling to
Word Embedding:
Look Both Ways**

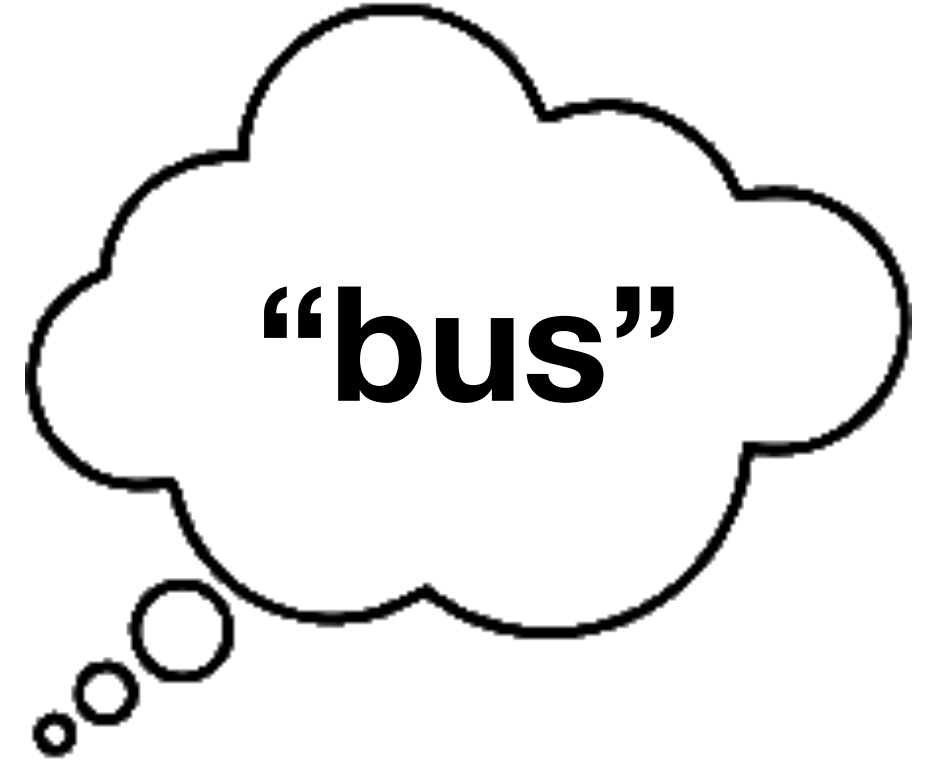
39 Language Model Training

Jay was hit by a _____



Look Both Ways

Jay was hit by a _____



Jay was hit by a _____ bus



41 Word2Vec: CBOW and Skip-gram

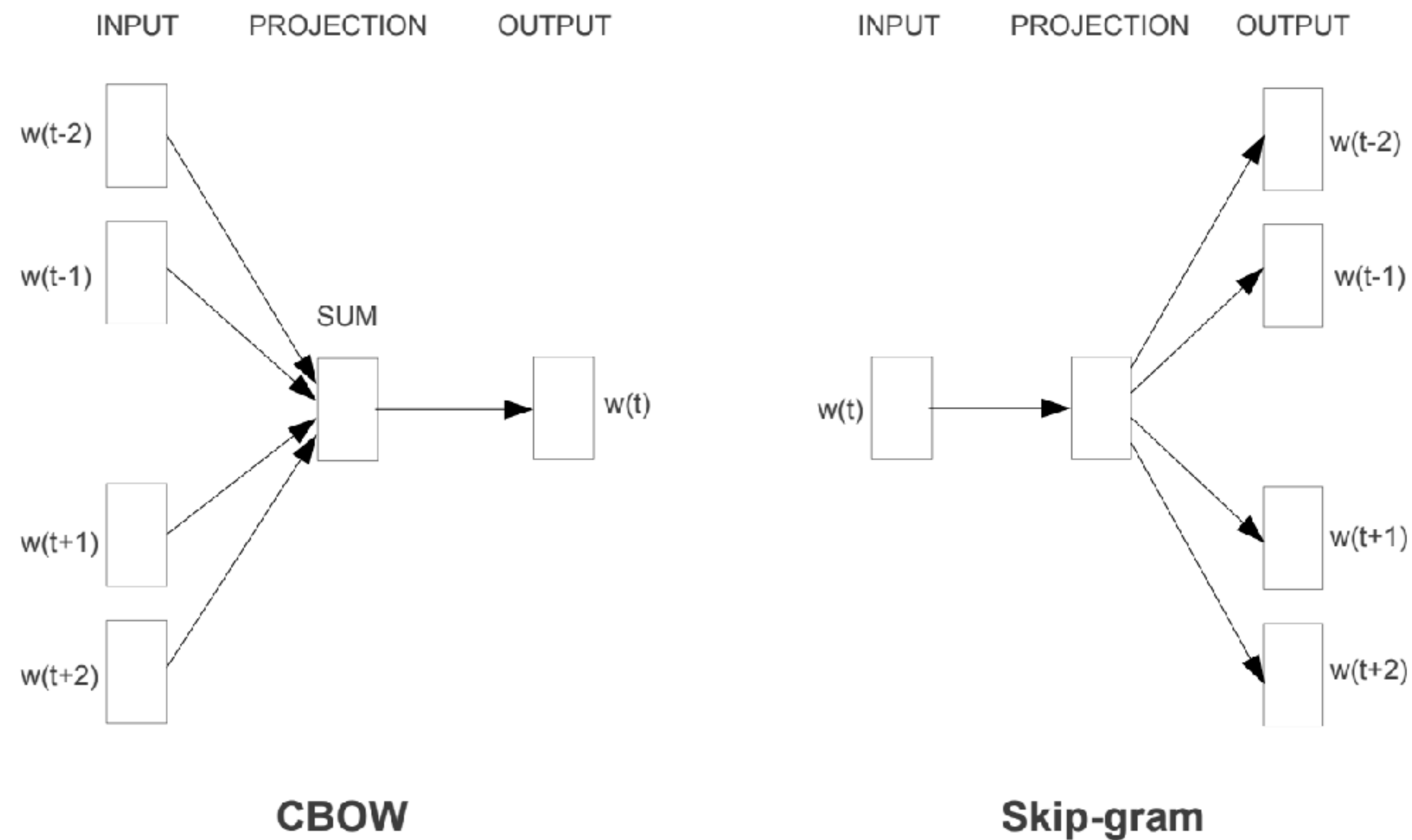


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

42 CBOW: Continuous Bag of Words

- Instead of only looking at words before the target word, we can also look at words after it.

“You shall know a word by the company it keeps” — J.R. Firth

Jay was hit by a _____ bus in...

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



Build training dataset

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

43 Skip-gram

- Instead of guessing a word based on its context (the words before and after it), this other architecture tries to guess neighboring words using the current word.

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

The word in the green slot would be the input word, each pink box would be a possible output. The pink boxes are in different shades because this sliding window actually creates four separate samples in our training dataset.

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

By sliding our window to the next positions, a couple of positions later, we will have a lot more training examples.

Step 1: grab an example from the dataset. Feed it into an untrained model asking it to predict an appropriate neighbour word.

Step 2: The model conducts the three steps and outputs a prediction vector (with a probability assigned to each word in its vocabulary)

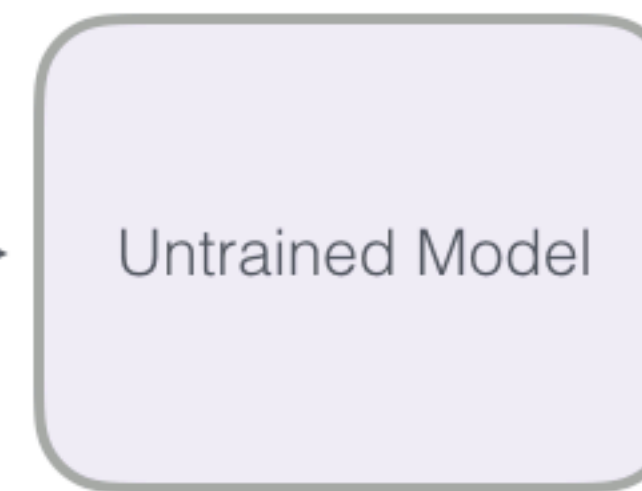
input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine
a	not
a	make
a	machine
a	in
machine	make
machine	a
machine	in
machine	the
in	a
in	machine
in	the
in	likeness

Actual Target

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

- 1) Look up embeddings
- 2) Calculate prediction
- 3) Project to output vocabulary

not



Model Prediction

0	aardvark	=	0
0	aarhus		0
0.001	aaron		-0.001
...			...
0.4	taco		-0.4
0.001	thou		0.999
...			...
0.0001	zyzzyva		-0.0001

Error

Update Model Parameters

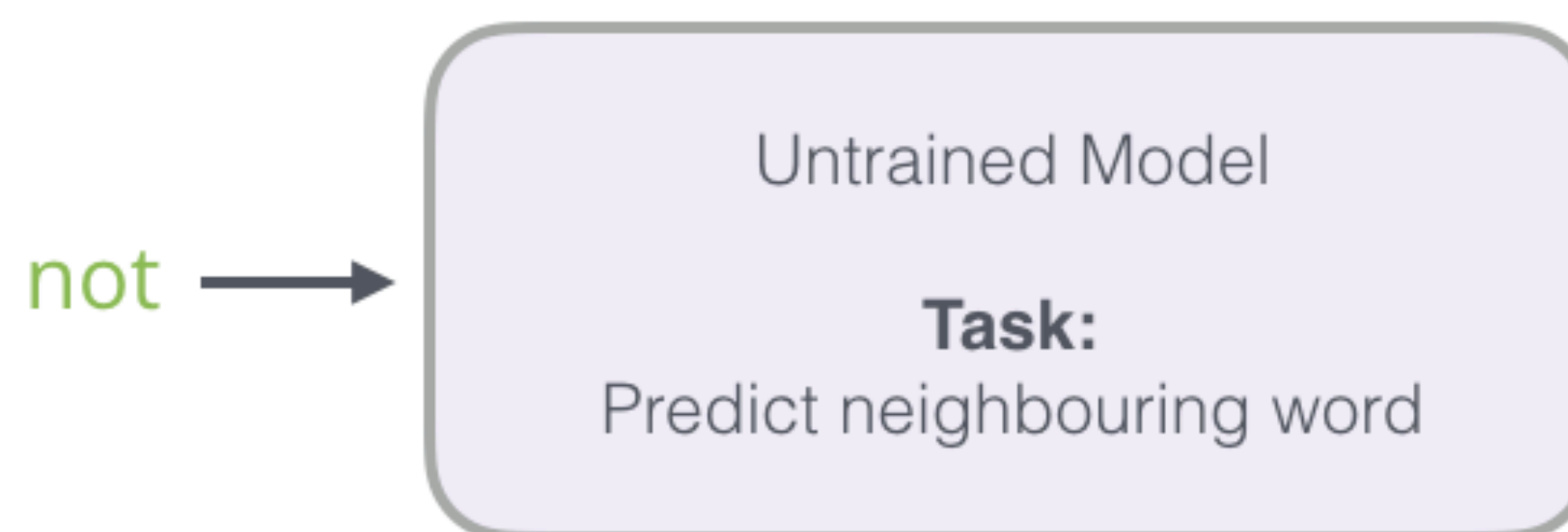
Step 4: We proceed to do the same process with the next sample in our dataset, and then the next, until we've covered all the samples in the dataset.

Step 3: This error vector can now be used to update the model so the next time, it's a little more likely to guess thou when it gets not as input.

It's still not how word2vec is actually trained. We're missing a couple of key ideas.

47 Negative Sampling

- The third step is very expensive from a computational point of view. **How to improve the performance?**



1) Look up embeddings

2) Calculate prediction

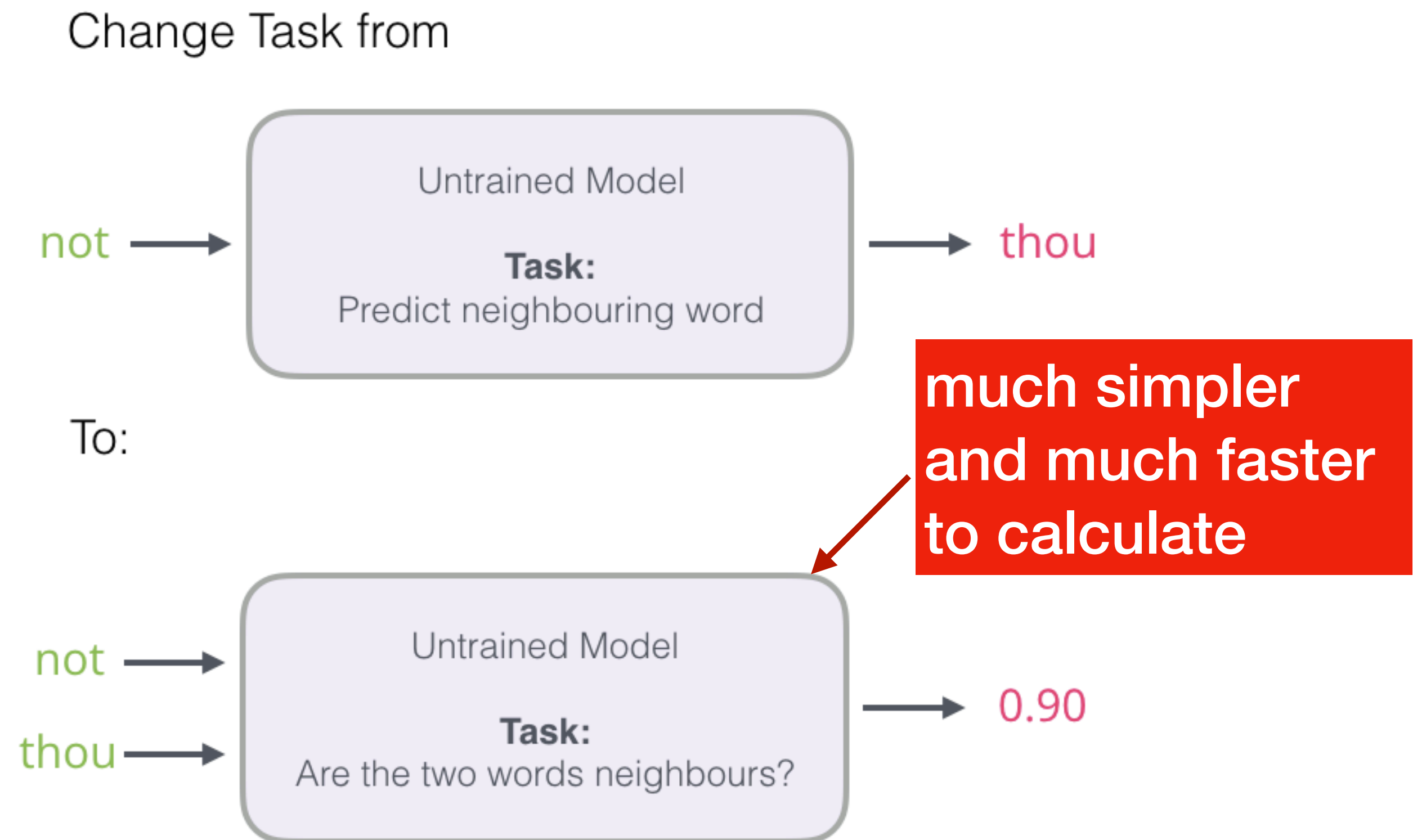
3) Project to output vocabulary

[Computationally Intensive]

Negative Sampling

- One way is to split our target into two steps:
 1. Generate high-quality word embeddings (Don't worry about next-word prediction);
 2. Use these high-quality embeddings to train a language model (to do next-word prediction).

To **generate high-quality embeddings** using a high-performance model, we can switch the model's task from **predicting a neighboring word** to **takes the input and output word, and outputs a score indicating if they're neighbors or not** (0 for "not neighbors", 1 for "neighbors").



Negative Sampling

- Switch the structure of our dataset

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



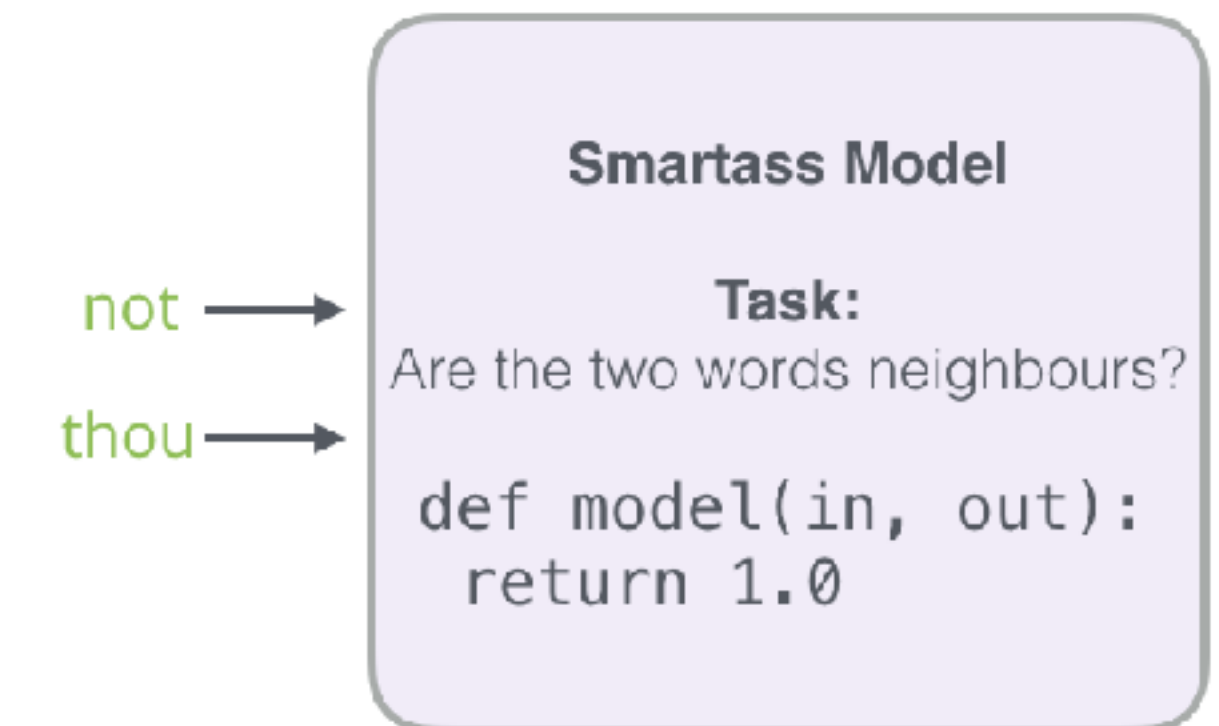
input word	output word	target
not	thou	1
not	shalt	1
not	make	1
not	a	1
make	shalt	1
make	not	1
make	a	1
make	machine	1

50 Negative Sampling

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

No negative sample



So a smartass model that always returns 1 – achieving 100% accuracy, but learning nothing and generating garbage embeddings.

51 Skip-gram with Negative Sampling

- Randomly selected words that are not neighbors from the vocabulary to get **negative examples**.

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

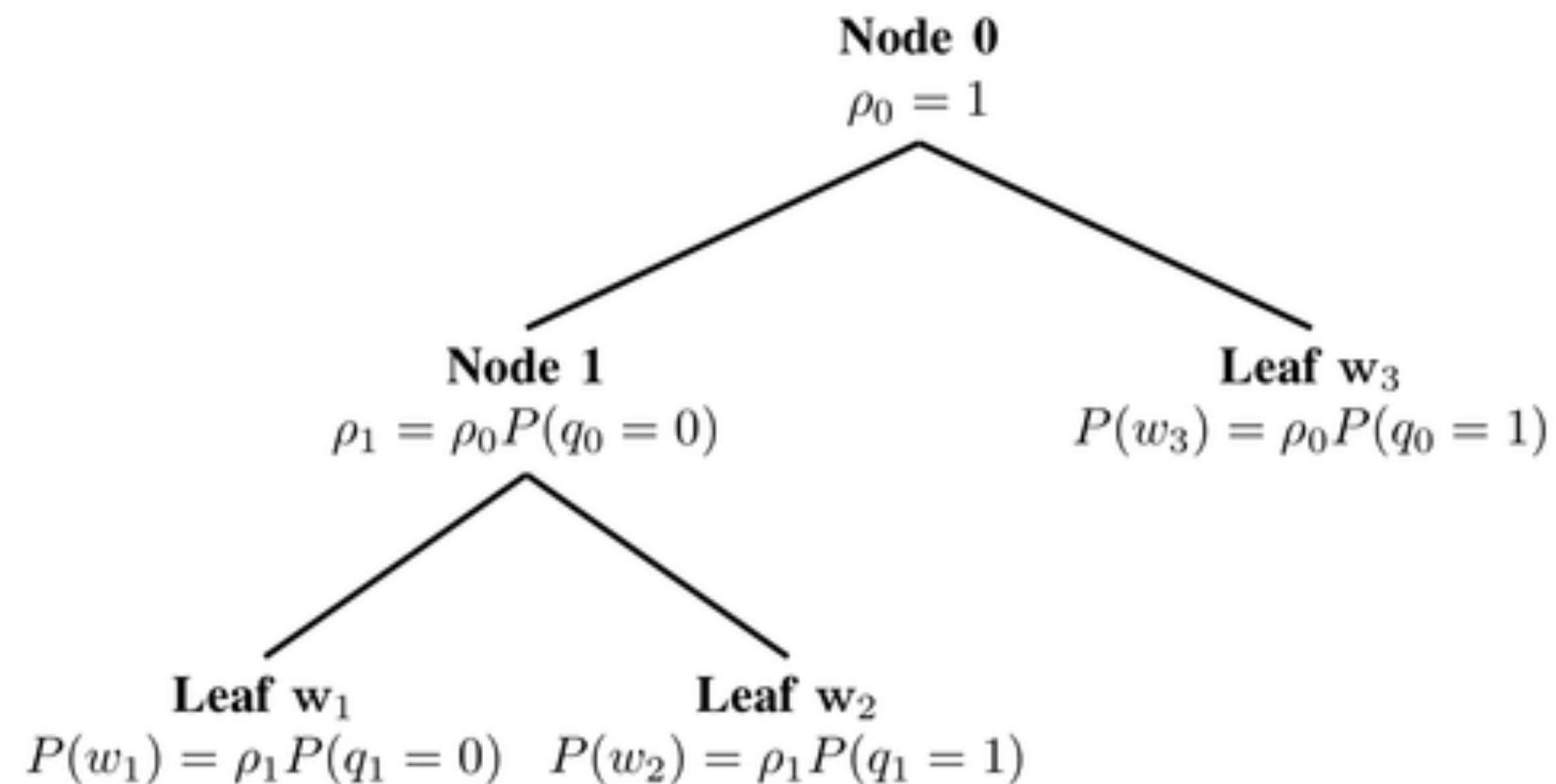
input word	output word	target
not	thou	1
not	aaron	0
not	taco	0
not	shalt	1
not	make	1

Pick randomly from vocabulary
(random sampling)

Word	Count	Probability
aardvark		
aarhus		
aaron		
taco		
thou		
zyzzyva		

Hierarchical Softmax

- Another way to accelerate model training is H-Softmax.
- Idea: compute the probability of leaf nodes using the paths



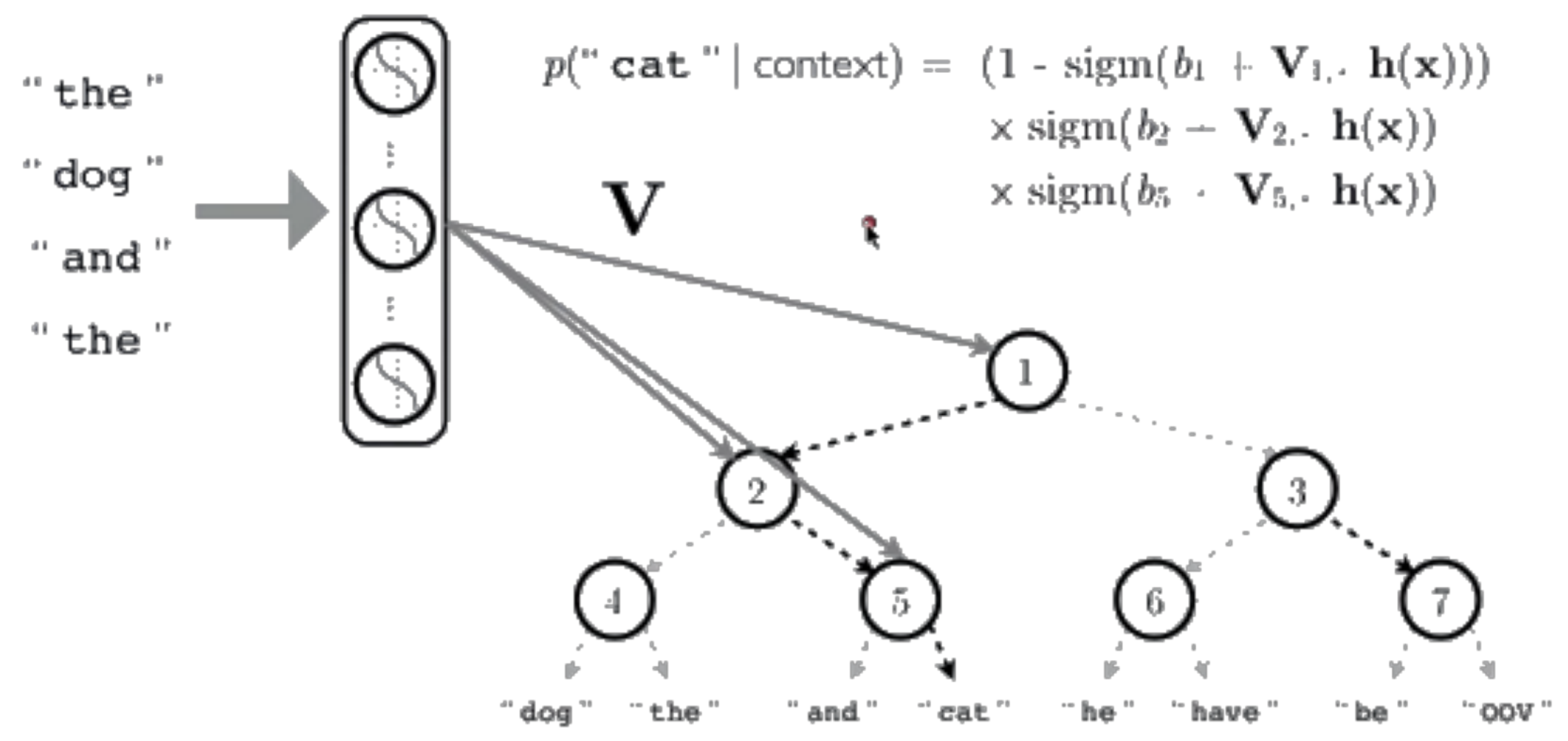
$$O(|V|) \rightarrow O(\log |V|)$$

Hierarchical Softmax

- Idea: compute the probability of leaf nodes using the paths
- Obviously, the structure of the tree is of significance. Mikolov et al. (2013) utilized a **Huffman tree** for their hierarchical softmax, (generates such a coding by assigning fewer bits to more common symbols.)
- Notably, **we are only able to obtain this speed-up during training**, when we know the word we want to predict (and consequently its path) in advance. During testing, when we need to find the most likely prediction, we still need to calculate the probability of all words, although narrowing down the choices in advance helps here.

$$p(\text{right} | n, c) = \sigma(h^\top v'_n).$$

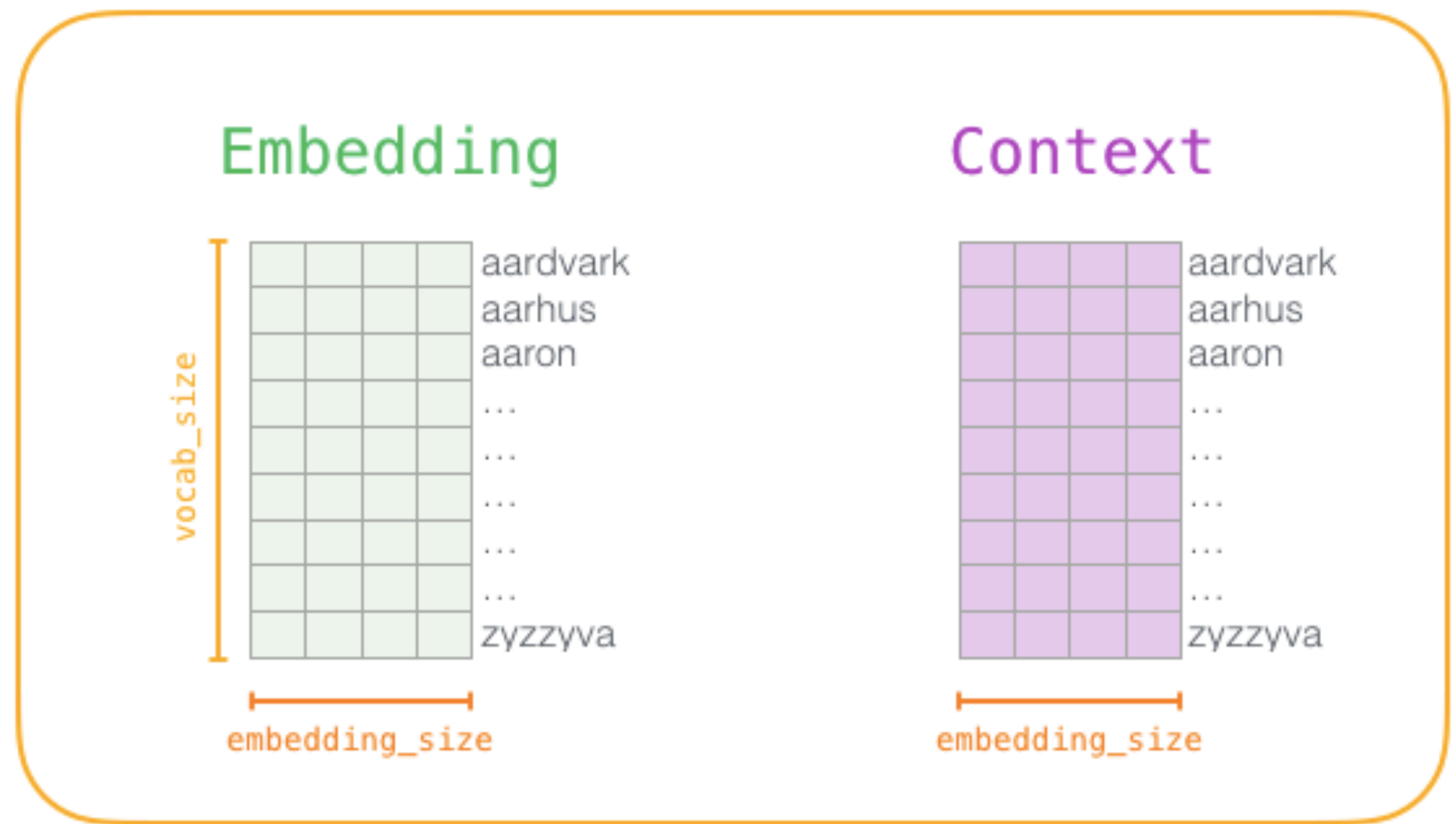
• Example: [" the ", " dog ", " and ", " the ", " cat "]



Word2Vec Training Process

Model initialization

- At the start of the training phase, we create two matrices – an **Embedding** matrix and a **Context** matrix. These two matrices have an embedding for each word in our vocabulary. We initialize these matrices with random values. (Why two vectors? Easier optimization.)

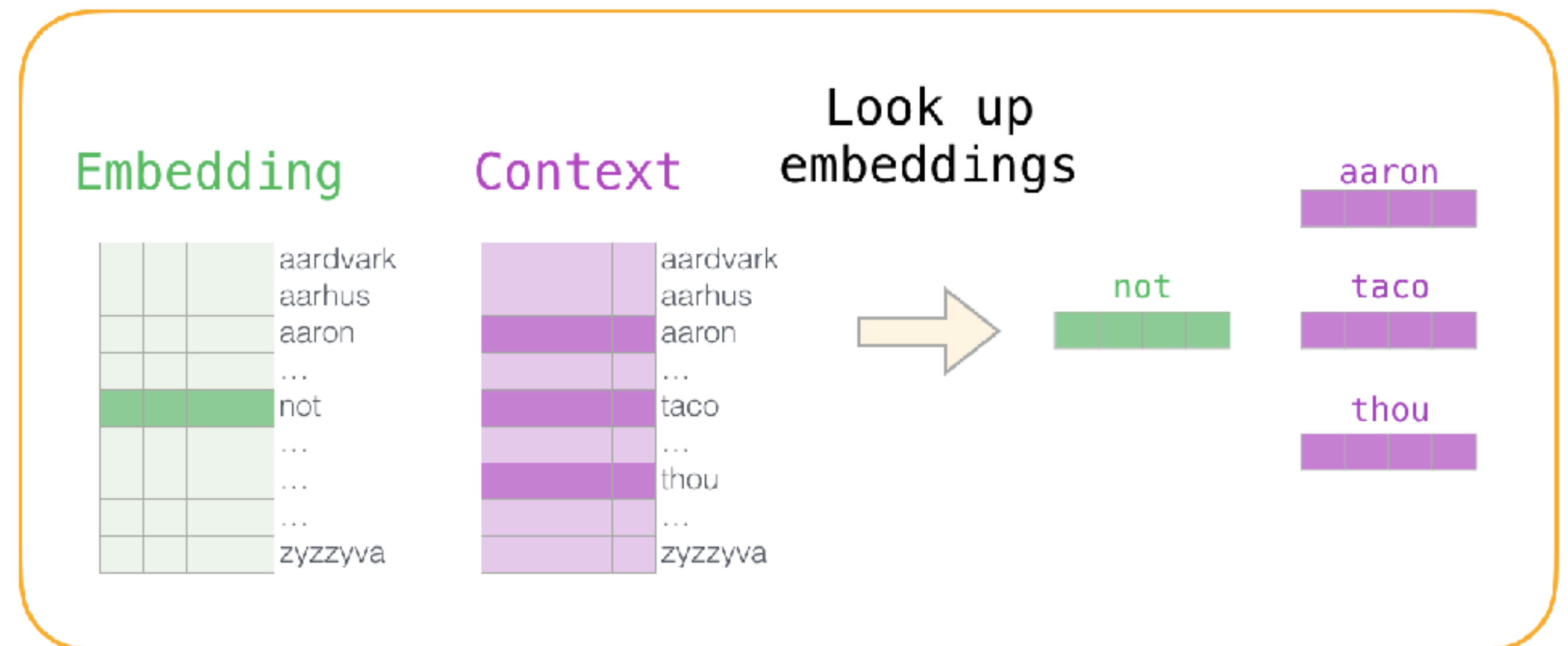


56 Feed Data

- For the input word, we look in the Embedding matrix. For the context words, we look in the Context matrix

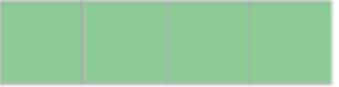

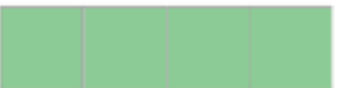

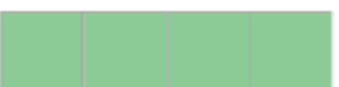

dataset

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
...



57 Forward Propagation

- Take the dot product of the input embedding with each of the context embeddings.
- Calculate probability by Sigmoid().
- Calculate error.

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

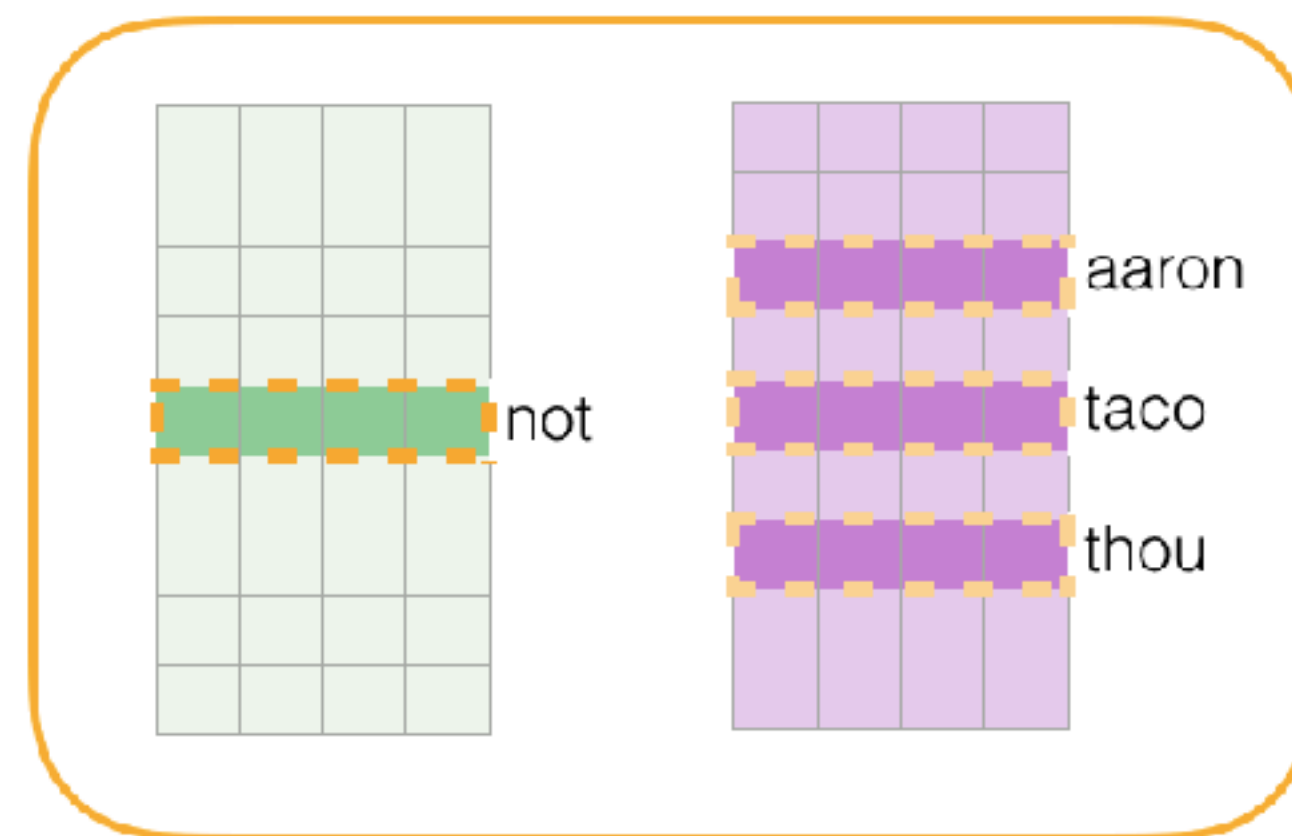
$$\text{error} = \text{target} - \text{sigmoid_scores}$$

Back-propagation

- We can now use this error score to adjust the embeddings of **not**, **thou**, **aaron**, and **taco** so that the next time we make this calculation, the result would be closer to the target scores.

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

$$\text{error} = \text{target} - \text{sigmoid_scores}$$



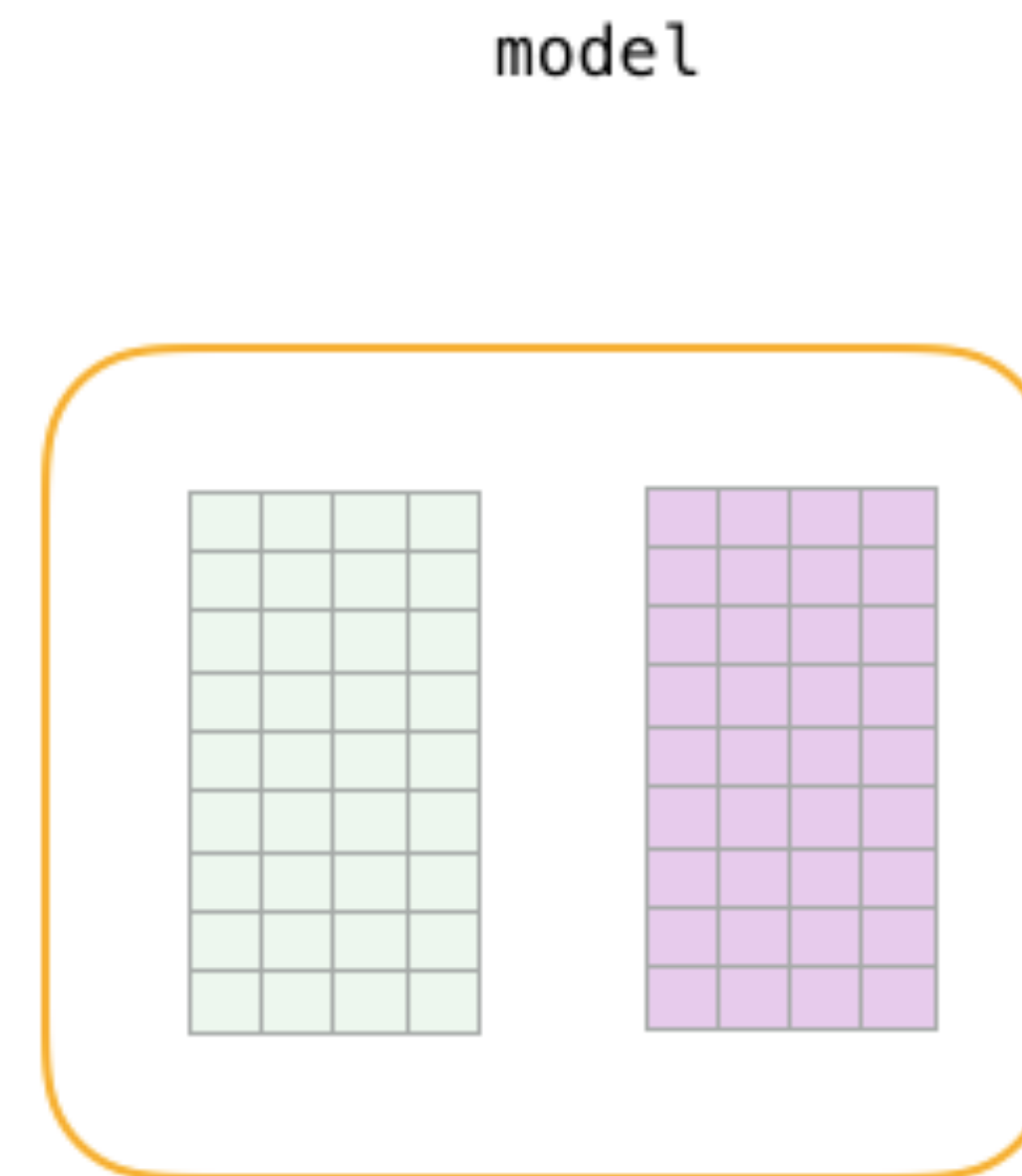
Update Model Parameters

Proceed to Next Batch

- Then we proceed to our next step (the next positive sample and its associated negative samples) and do the same process again.

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
...



Glove

61 Comparison

Count-based

- LSA, HAL (Lund & Burgess), COALS (Rohde et al), Hellinger-PCA (Lebret & Collobert)
- Pros
 - ✓ Fast training
 - ✓ Efficient usage of statistics
- Cons
 - ✓ Primarily used to capture word similarity
 - ✓ Disproportionate importance given to large counts

Direct prediction

- NNLM, HLBL, RNN, Skipgram/CBOW (Bengio et al; Collobert & Weston; Huang et al; Mnih & Hinton; Mikolov et al; Mnih & Kavukcuoglu)
- Pros
 - ✓ Generate improved performance on other tasks
 - ✓ Capture complex patterns beyond word similarity
- Cons
 - ✓ Benefits mainly from large corpus
 - ✓ Inefficient usage of statistics

Combining the benefits from both worlds → GloVe

62 GloVe

- Idea: **ratio of co-occurrence probability** can encode meaning
- P_{ij} is the probability that word w_j appears in the context of word w_i

$$P_{ij} = P(w_j | w_i) = X_{ij} / X_i$$

- Relationship between the words w_i and w_j

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k ice)/P(k steam)$	8.9	8.5×10^{-2}	1.36	0.96

63 GloVe

- ⊙ Idea: **ratio of co-occurrence probability** can encode meaning
- ⊙ P_{ij} is the probability that word w_j appears in the context of word w_i

$$P_{ij} = P(w_j | w_i) = X_{ij} / X_i$$

- ⊙ Relationship between the words w_i and w_j

Probability and Ratio	$k = \textit{solid}$	$k = \textit{gas}$	$k = \textit{water}$	$k = \textit{fashion}$
$P(k \textit{ice})$	large	small	large	small
$P(k \textit{steam})$	small	large	large	small
$P(k \textit{ice})/P(k \textit{steam})$	large	small	$\sim=1$	$\sim=1$

- The relationship of w_i and w_j approximates the ratio of their co-occurrence probabilities with various w_k

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

$$F((v_{w_i} - v_{w_j})^T v'_{\tilde{w}_k}) = \frac{P_{ik}}{P_{jk}} \quad F(\cdot) = \exp(\cdot)$$

$$v_{w_i} \cdot v'_{\tilde{w}_k} = v_{w_i}^T v'_{\tilde{w}_k} = \log P(w_k | w_i)$$

$$\begin{aligned} v_{w_i} \cdot v'_{\tilde{w}_j} &= v_{w_i}^T v'_{\tilde{w}_j} = \log P(w_j | w_i) \\ &= \log P_{ij} = \log(X_{ij}) - \log(X_i) \end{aligned}$$

$$P_{ij} = X_{ij}/X_i$$

$$v_{w_i}^T v'_{\tilde{w}_j} + b_i + \tilde{b}_j = \log(X_{ij})$$

$$C(\theta) = \sum_{i,j=1}^V f(P_{ij}) (v_{w_i} \cdot v'_{\tilde{w}_j} - \log P_{ij})^2$$

$$C(\theta) = \sum_{i,j=1}^V f(X_{ij}) (v_{w_i}^T v'_{\tilde{w}_j} + b_i + \tilde{b}_j - \log X_{ij})^2$$

1. As $P(w_j|w_i) = X_{ij}/X_i$, we get first equation
2. In neural networks, we have some bias terms. we can merge the constant $\log(X_i)$ with the learnable bias term b_i , so that we get the second equation
3. So our Cost function to learn word vectors are the third equation
4. If we take bias terms into account, we get the forth equation.
5. Note we weight the cost by taking the frequency of co-occurrence into account. The weight is some function over the co-occurrence frequency X_{ij} or P_{ij} .

66 GloVe – Weighted Least Squares Regression

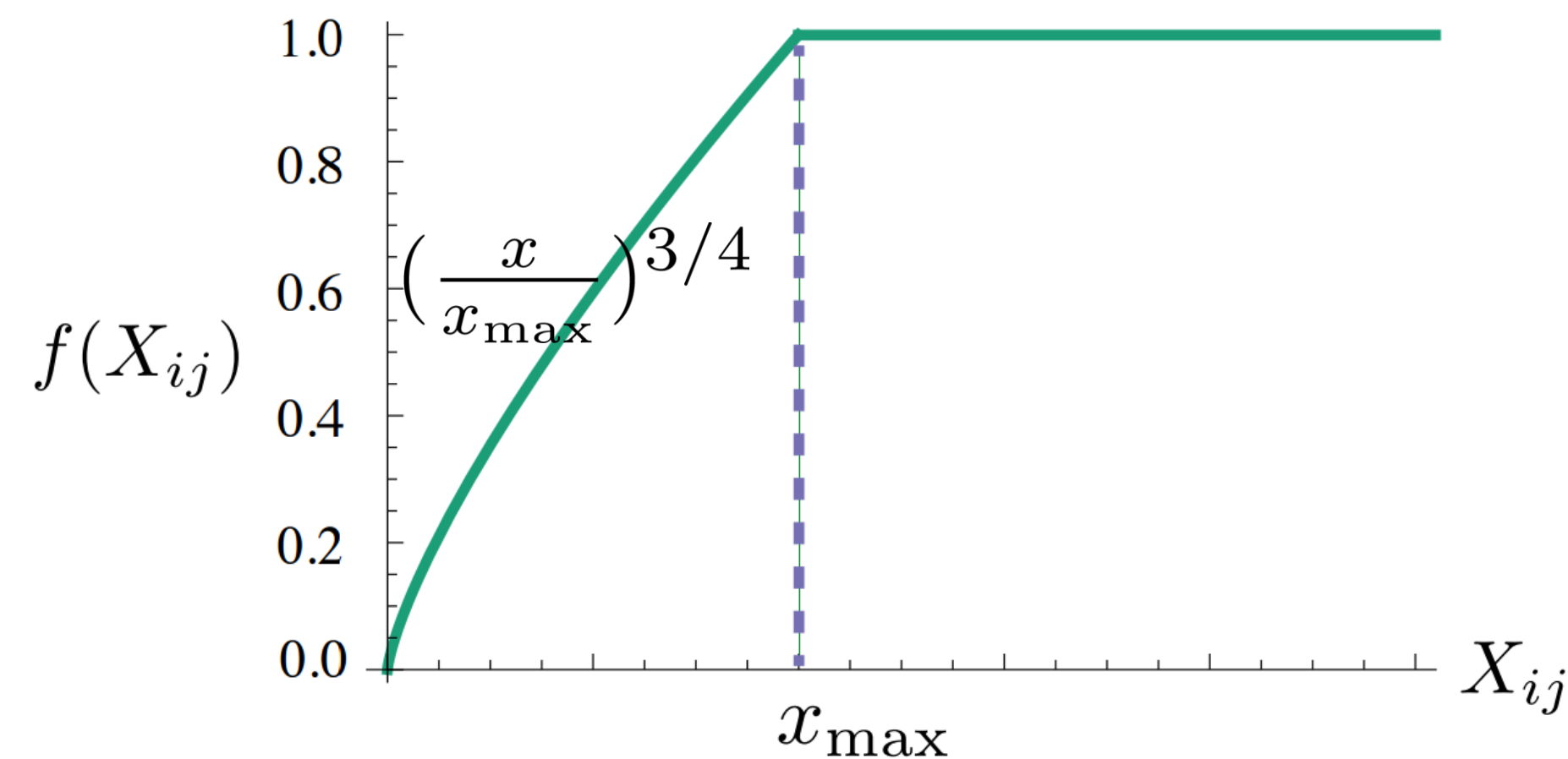
$$C(\theta) = \sum_{i,j=1}^V f(X_{ij}) (v_{w_i}^T v'_{\tilde{w}_j} + b_i + \tilde{b}_j - \log X_{ij})^2$$

Weighting function should obey

- $f(0) = 0$

- $f(x)$ should be non-decreasing so that *rare co-occurrences* are not overweighted

- $f(x)$ should be relatively small for large values of x , so that *frequent co-occurrences* are not overweighted



fast training, scalable, good performance even with small corpus, and small vectors

GloVe Results

Nearest words to frog:

- 1. frogs
- 2. toad
- 3. litoria
- 4. leptodactylidae
- 5. rana
- 6. lizard
- 7. eleutherodactylus



litoria



leptodactylidae

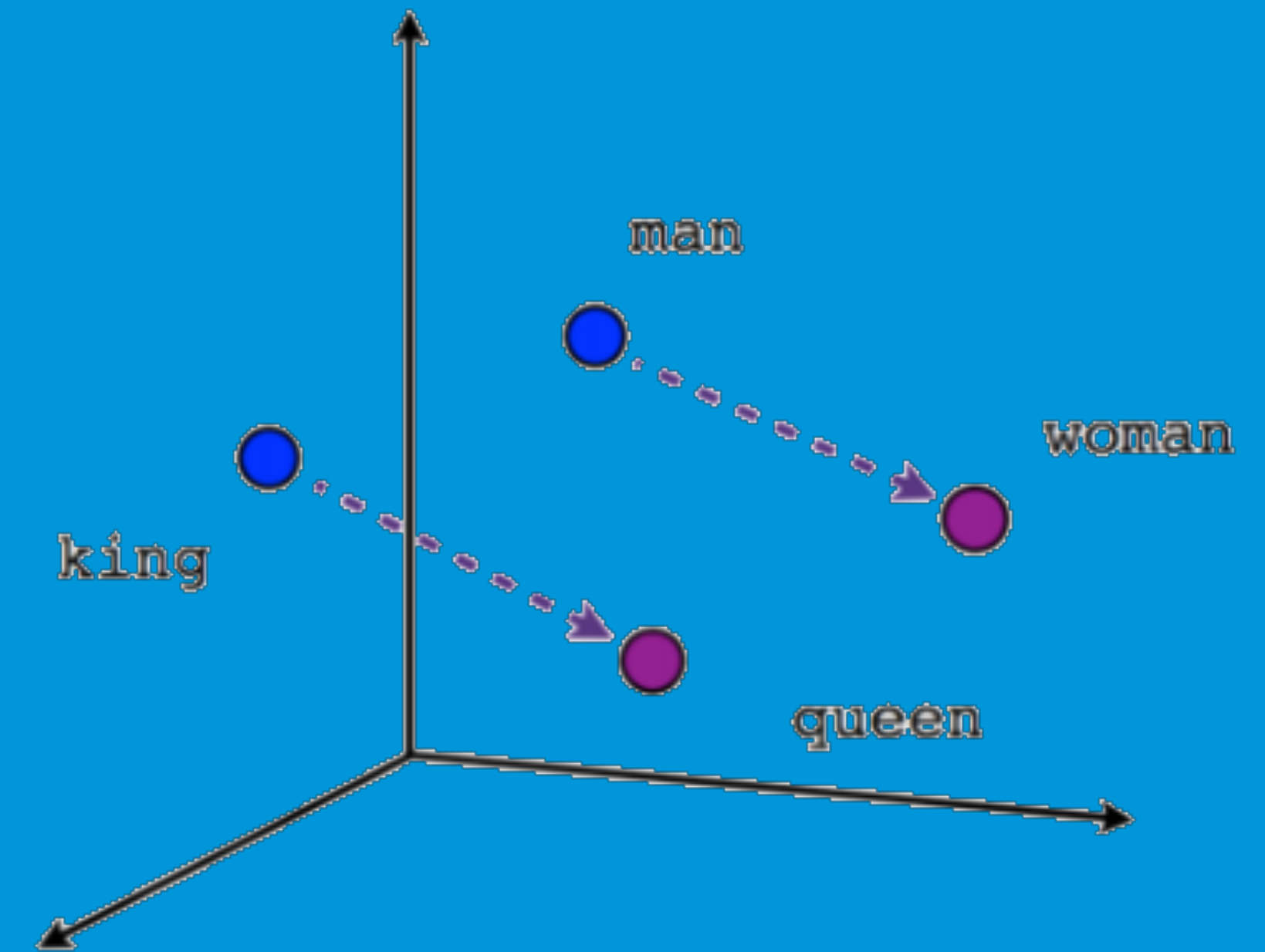


rana



eleutherodactylus

Word Vector Evaluation

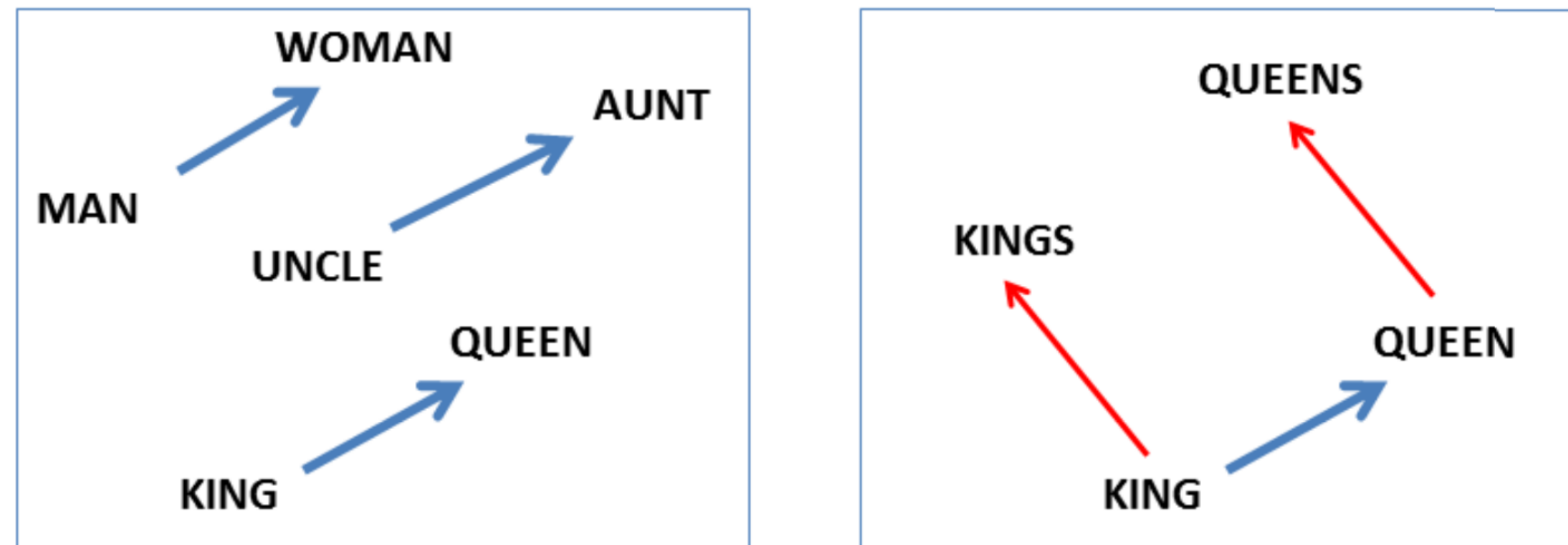


69 Intrinsic Evaluation – Word Analogies

- Word linear relationship $w_A : w_B = w_C : w_x$

$$x = \arg \max_x \frac{(v_{w_B} - v_{w_A} + v_{w_C})^T v_{w_x}}{\|v_{w_B} - v_{w_A} + v_{w_C}\|}$$

- Syntactic and Semantic example questions [\[link\]](#)



Issue: what if the information is there but not linear

70 Intrinsic Evaluation – Word Analogies

Word linear relationship $w_A : w_B = w_C : w_x$

Syntactic and **Semantic** example questions [[link](#)]

city---in---state

Chicago : Illinois = Houston : Texas

Chicago : Illinois = Philadelphia : Pennsylvania

Chicago : Illinois = Phoenix : Arizona

Chicago : Illinois = Dallas : Texas

Chicago : Illinois = Jacksonville : Florida

Chicago : Illinois = Indianapolis : Indiana

Chicago : Illinois = Austin : Texas

Chicago : Illinois = Detroit : Michigan

Chicago : Illinois = Memphis : Tennessee

Chicago : Illinois = Boston : Massachusetts

capital---country

Abuja : Nigeria = Accra : Ghana

Abuja : Nigeria = Algiers : Algeria

Abuja : Nigeria = Amman : Jordan

Abuja : Nigeria = Ankara : Turkey

Abuja : Nigeria = Antananarivo : Madagascar

Abuja : Nigeria = Apia : Samoa

Abuja : Nigeria = Ashgabat : Turkmenistan

Abuja : Nigeria = Asmara : Eritrea

Abuja : Nigeria = Astana : Kazakhstan

Issue: different cities may have same name

Issue: can change with time

71 Intrinsic Evaluation – Word Analogies

- Word linear relationship $w_A : w_B = w_C : w_x$
- Syntactic** and Semantic example questions

superlative

bad : worst = big : biggest
bad : worst = bright : brightest
bad : worst = cold : coldest
bad : worst = cool : coolest
bad : worst = dark : darkest
bad : worst = easy : easiest
bad : worst = fast : fastest
bad : worst = good : best
bad : worst = great : greatest

past tense

dancing : danced = decreasing : decreased
dancing : danced = describing : described
dancing : danced = enhancing : enhanced
dancing : danced = falling : fell
dancing : danced = feeding : fed
dancing : danced = flying : flew
dancing : danced = generating : generated
dancing : danced = going : went
dancing : danced = hiding : hid
dancing : danced = hiding : hit

72 Intrinsic Evaluation – Word Correlation

- Comparing word correlation with human-judged scores
- Human-judged word correlation

Word 1	Word 2	Human-Judged Score
tiger	cat	7.35
tiger	tiger	10.00
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62

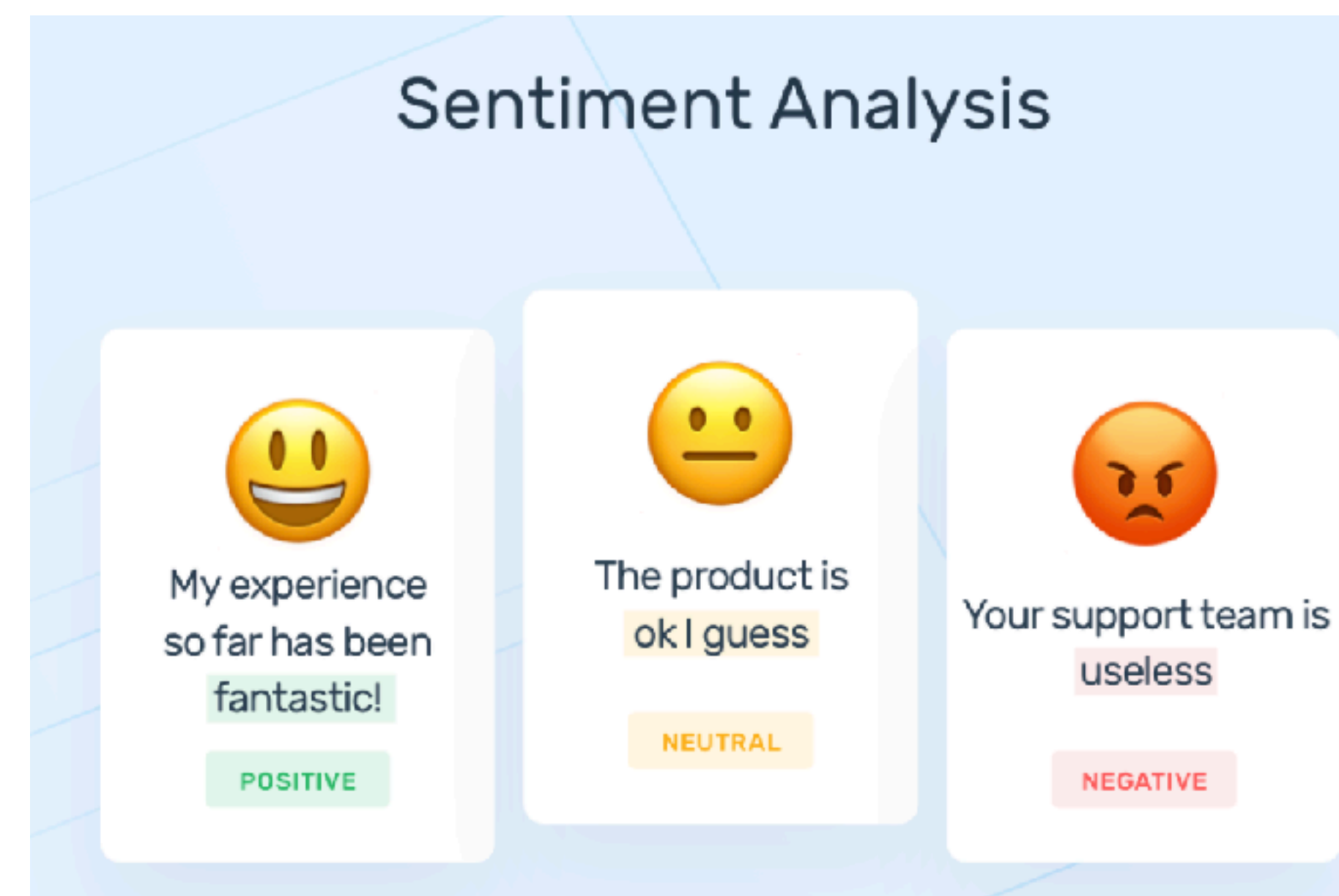
Ambiguity: synonym or same word with different POSs

73 Extrinsic Evaluation – Subsequent Task

- Goal: use word vectors in neural net models built for subsequent tasks

In fact, the **Chinese** market has the **three** most influential names of the retail and tech space – **Alibaba**, **Baidu**, and **Tencent** (collectively touted as **BAT**), and is betting big in the global **AI** in retail industry space. The **three** giants which are claimed to have a cut-throat competition with the **U.S.** (in terms of resources and capital) are positioning themselves to become the 'future **AI** platforms'. The trio is also expanding in other **Asian** countries and investing heavily in the **U.S.** based **AI** startups to leverage the power of **AI**. Backed by such powerful initiatives and presence of these conglomerates, the market in APAC AI is forecast to be the fastest-growing **one**, with an anticipated **CAGR** of **45%** over **2018 - 2024**.

To further elaborate on the geographical trends, **North America** has procured **more than 50%** of the global share in **2017** and has been leading the regional landscape of **AI** in the retail market. The **U.S.** has a significant credit in the regional trends with **over 65%** of investments (including M&As, private equity, and venture capital) in artificial intelligence technology. Additionally, the region is a huge hub for startups in tandem with the presence of tech titans, such as **Google**, **IBM**, and **Microsoft**.



If two word vectors are similar, they may share the same NER tag or sentiment information.

⦿ **Reading assignment 2: (due date: February 1st, 2022 11:59 pm EST timezone)**

- **Distributed Representations of Words and Phrases and their Compositionality (negative sampling paper)**
<https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf>

⦿ **Suggested readings:**

- Efficient Estimation of Word Representations in Vector Space (original word2vec paper)
- GloVe: Global Vectors for Word Representation (original GloVe paper)
- Improving Distributional Similarity with Lessons Learned from Word Embeddings
- Evaluation methods for unsupervised word embeddings
- A Latent Variable Model Approach to PMI-based Word Embeddings
- Linear Algebraic Structure of Word Senses, with Applications to Polysemy
- On the Dimensionality of Word Embedding

Next lecture: Sentence Representation & Text Classification

Thanks! Q&A

Bang Liu

Email: bang.liu@umontreal.ca

Homepage: <http://www-labs.iro.umontreal.ca/~liubang/>