Natural Language Processing (NLP)

Neural Network Embeddings

Equipping You with Research Depth and Industry Skills

By:

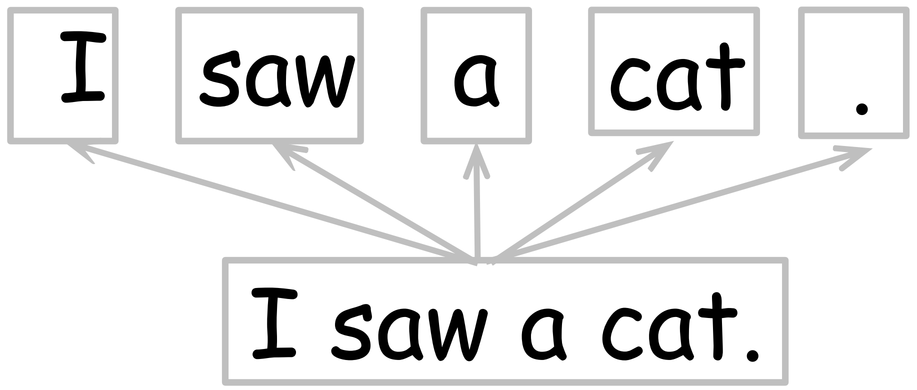
Dr. Zohair Ahmed

Why do we need word representations?

I saw a cat.

Text (your input)

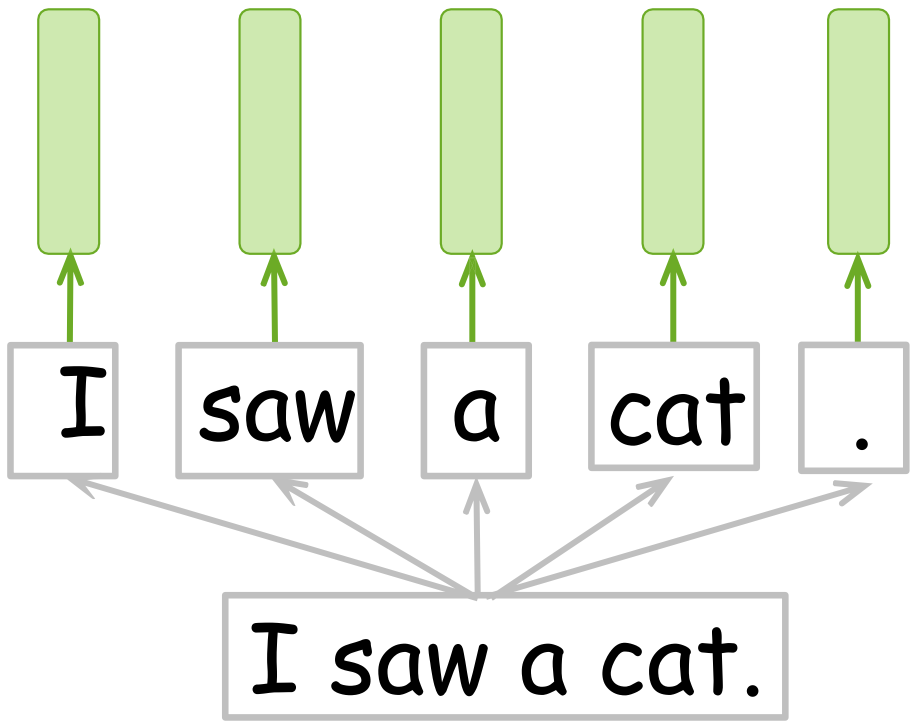
Why do we need word representations?



Sequence of tokens

Text (your input)

Why do we need word representations?

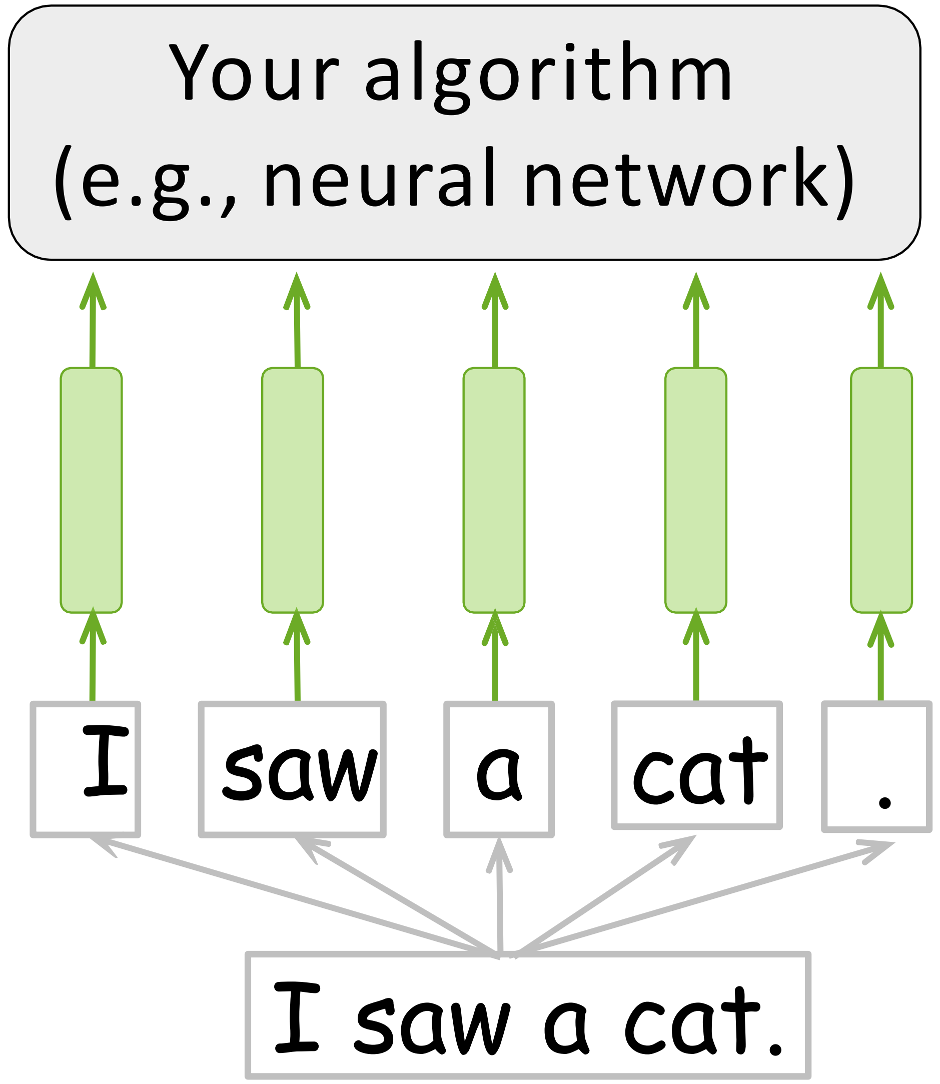


Word representation - vector
(input for your model/algorithm)

Sequence of tokens

Text (your input)

Why do we need word representations?



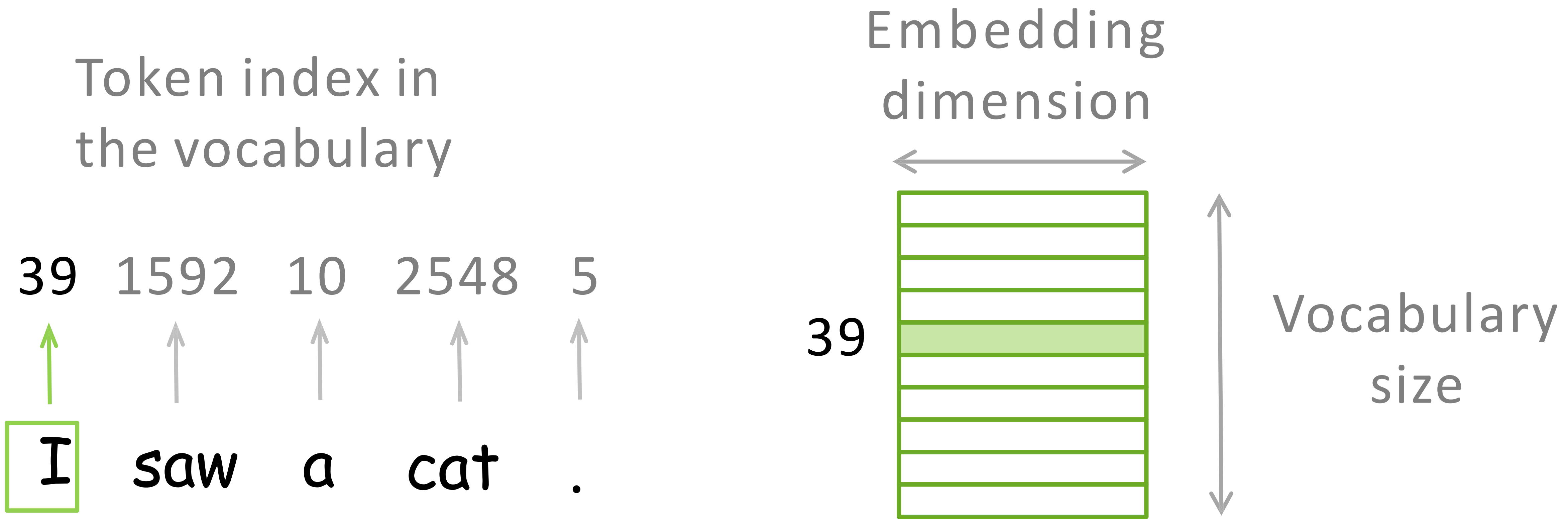
Any algorithm for solving a task

Word representation - vector
(input for your model/algorithm)

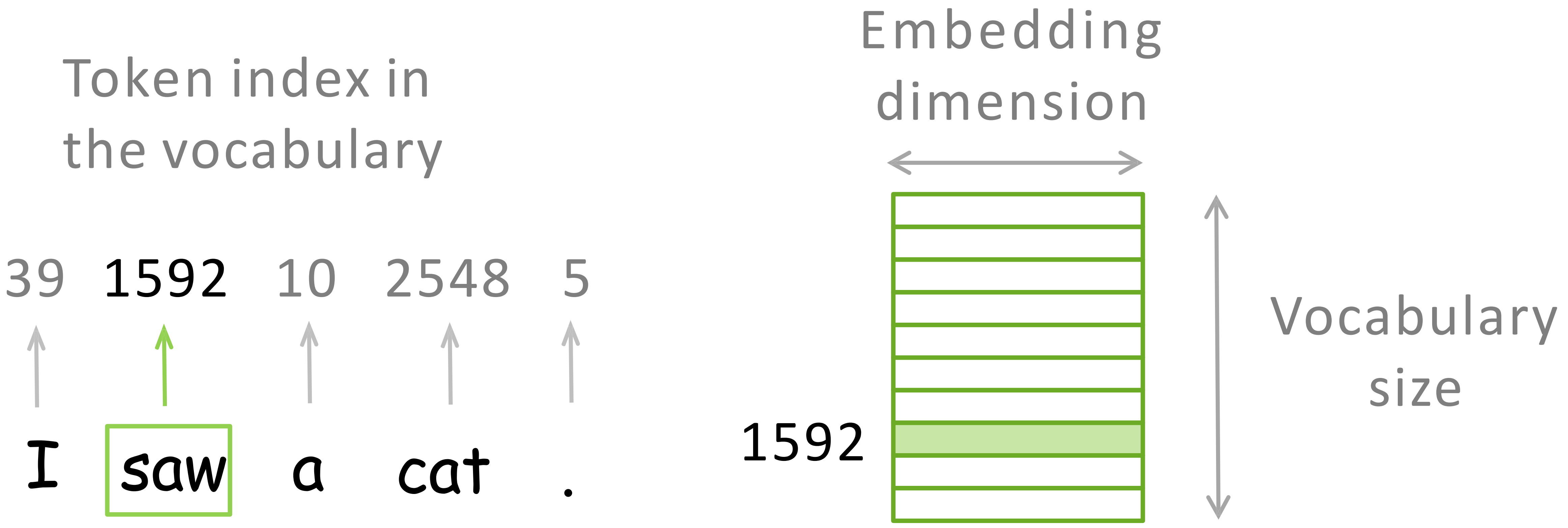
Sequence of tokens

Text (your input)

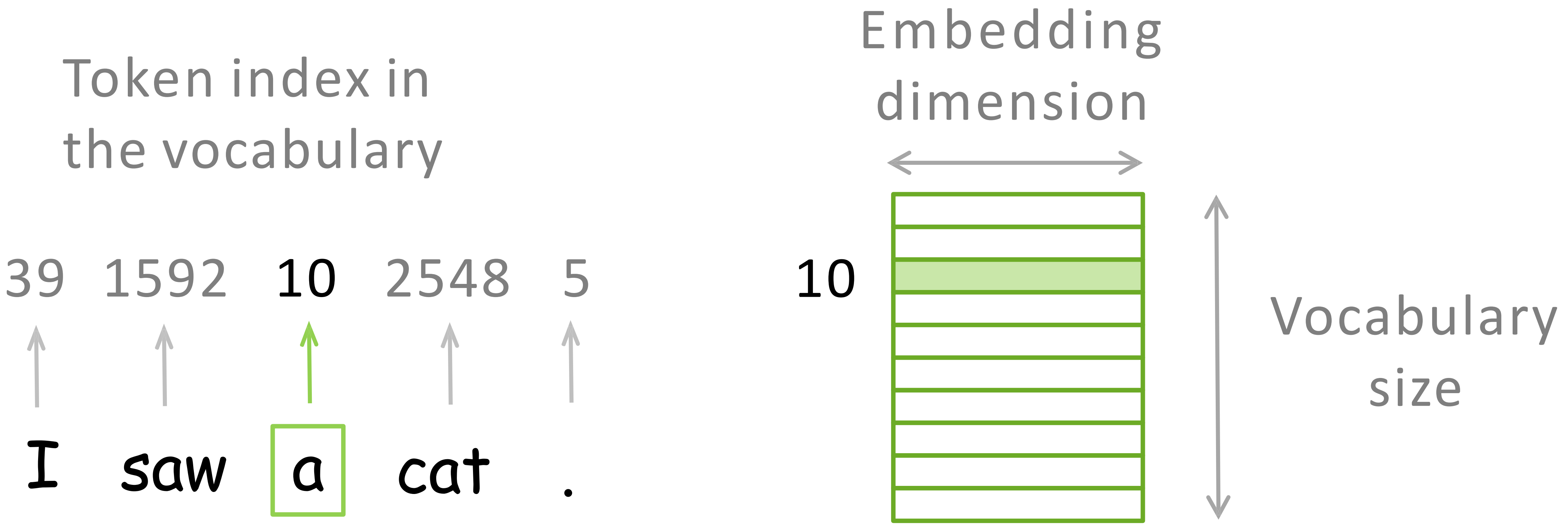
How it works: Look-up Table



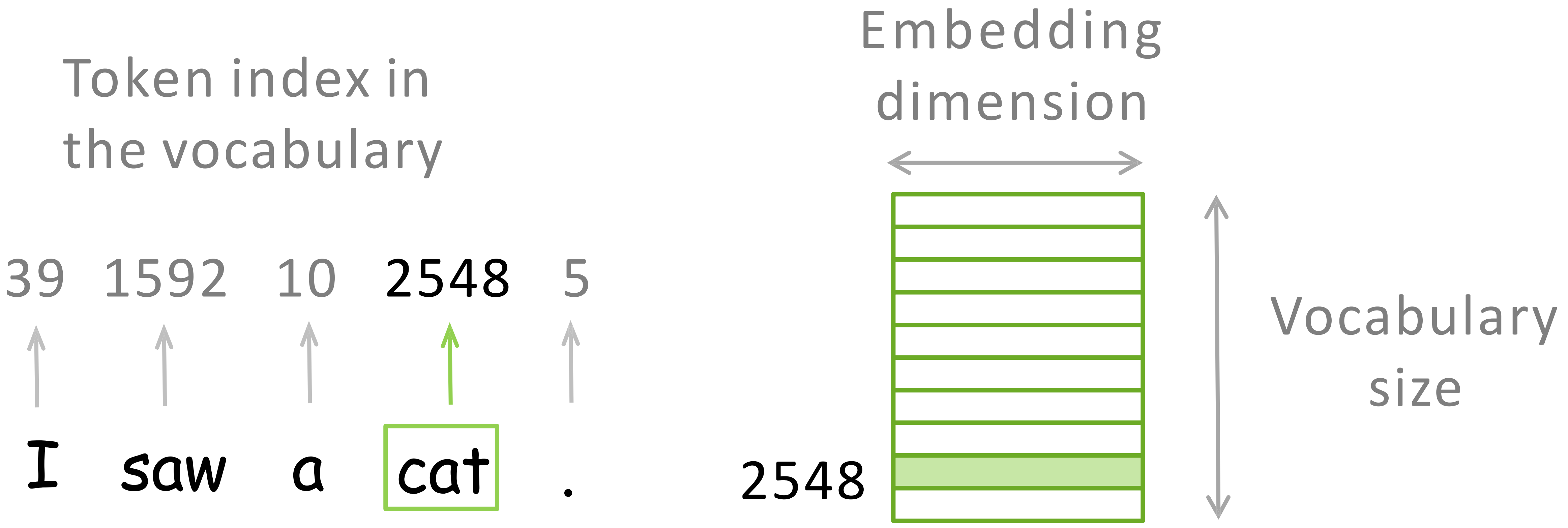
How it works: Look-up Table



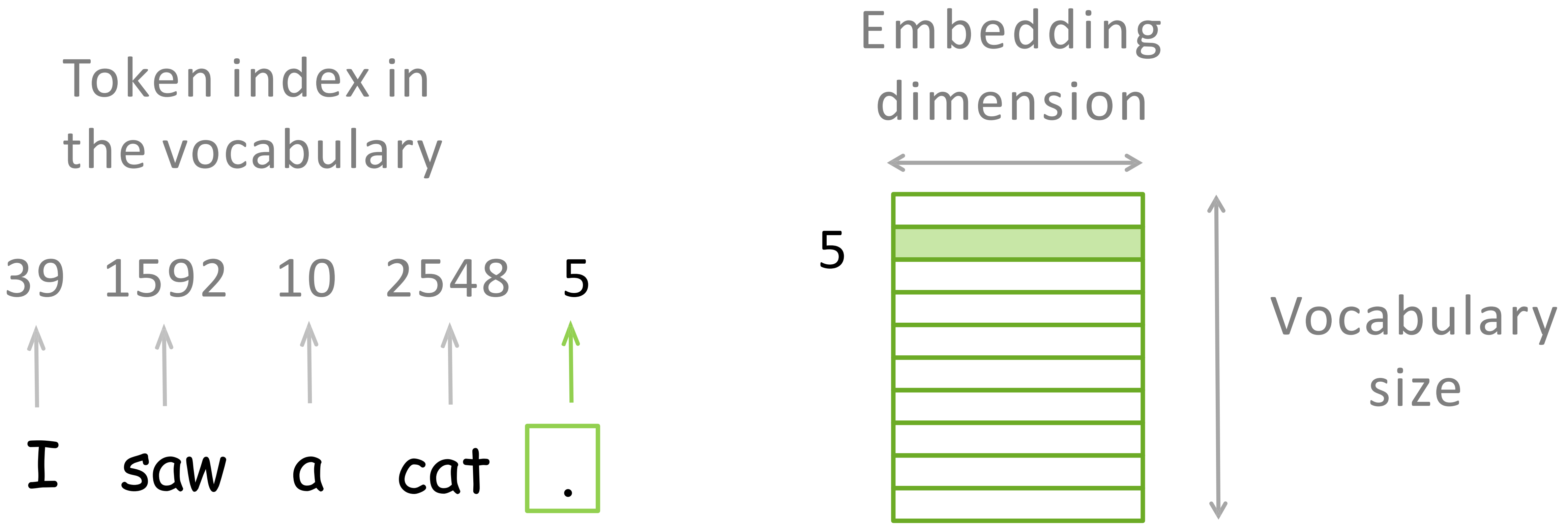
How it works: Look-up Table



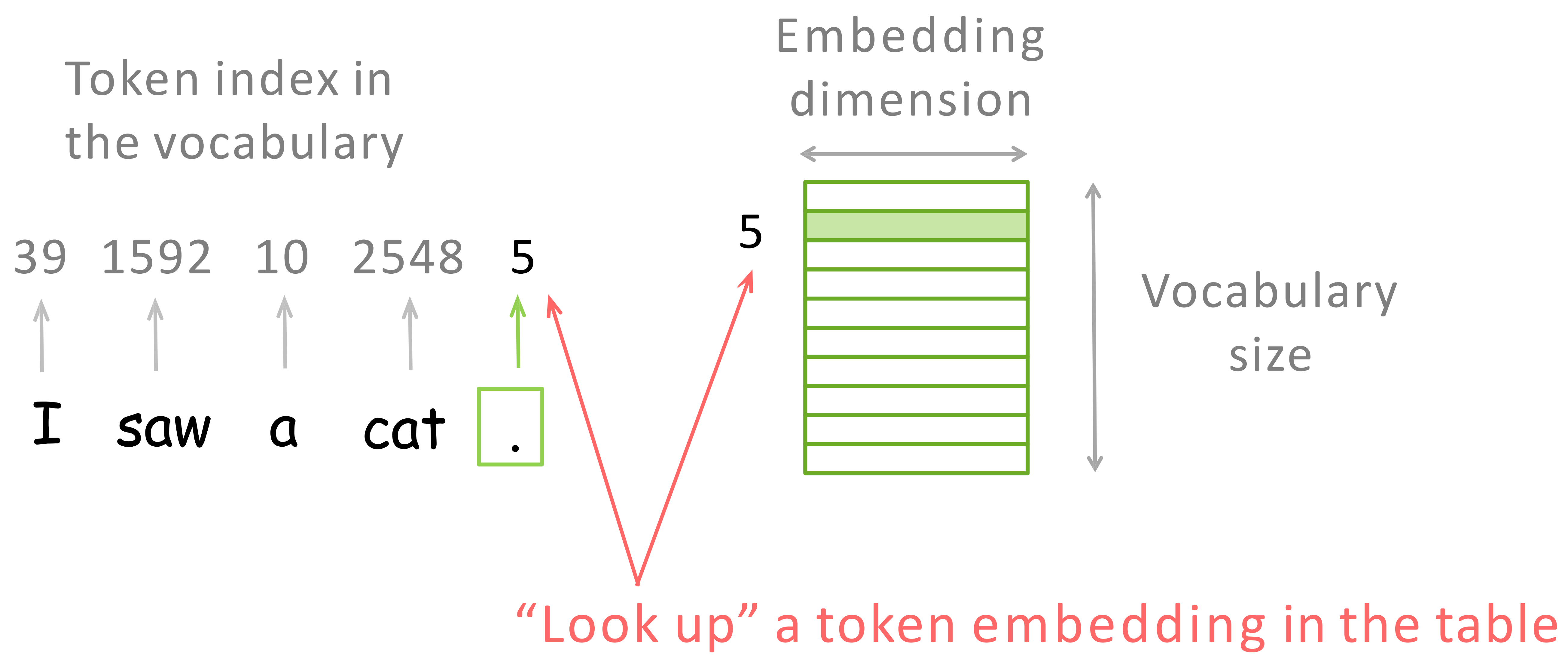
How it works: Look-up Table



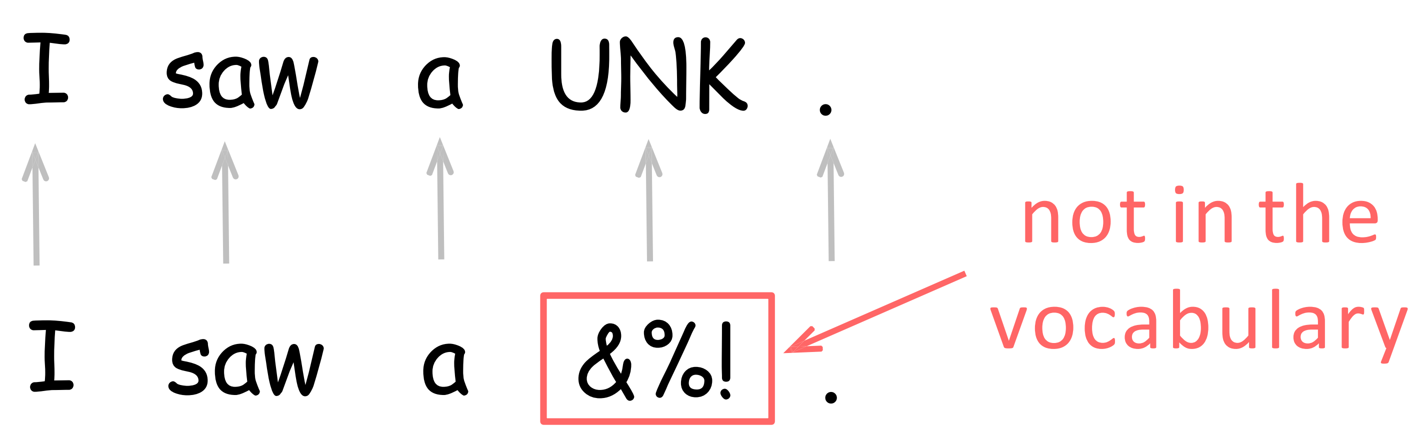
How it works: Look-up Table



How it works: Look-up Table



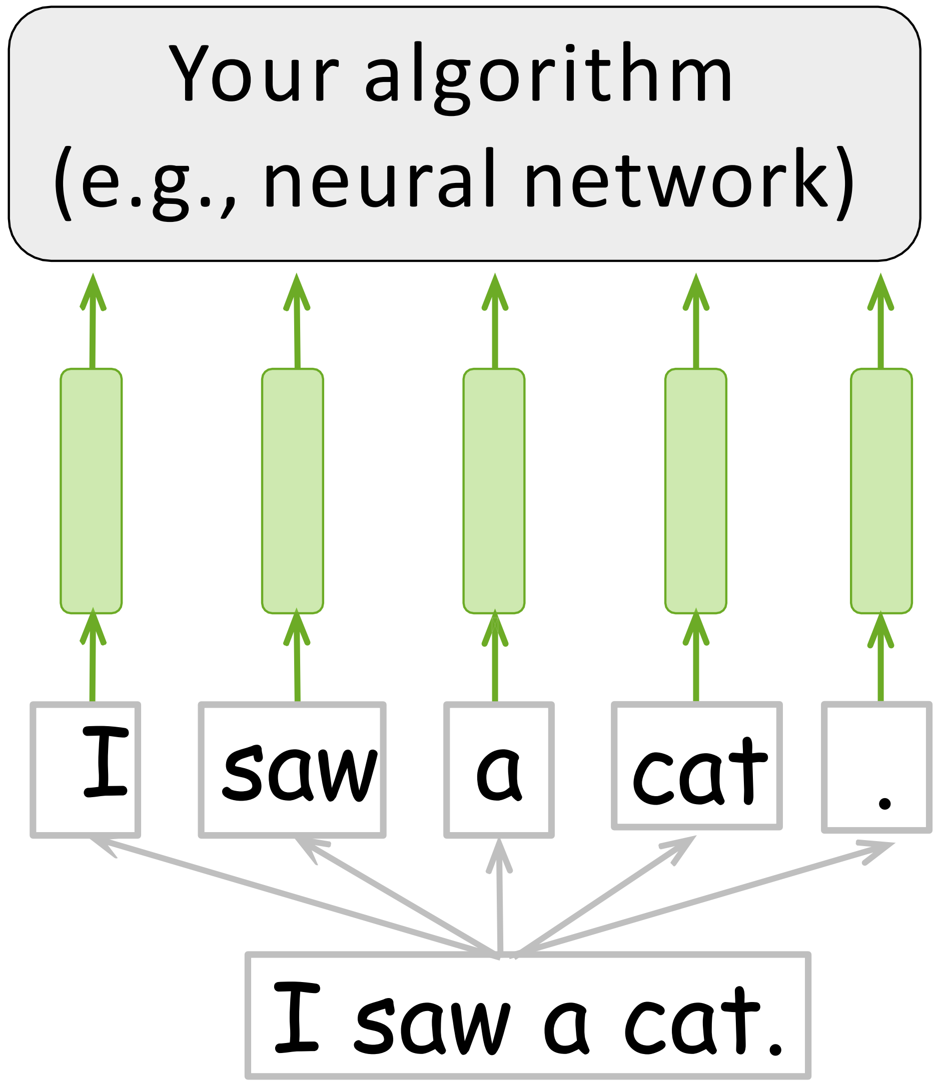
Note UNKs: Out-of-Vocabulary Tokens



Vocabulary is chosen in advance

Therefore, some tokens may be “unknown” – you can use a special token for them

How can we gat word representations?



<u>In the following</u>:

How can we get these representations?

One-hot Vectors

One-Hot Vectors: Represent Words as Discrete Symbols

One is 1, the rest are 0

dog	0…0…010….0…0
cat	0…010…0….0…0
table	0…0…0….0010…

Embedding dimension = vocabulary size

One-Hot Vectors: Represent Words as Discrete Symbols

One is 1, the rest are 0

dog	0…0…010….0…0
cat	0…010…0….0…0
table	0…0…0….0010…

Embedding dimension = vocabulary size

Any problems?

One-Hot Vectors: Represent Words as Discrete Symbols

One is 1, the rest are 0

dog 0…0…010….0…0

cat 0…010…0….0…0

table 0…0…0….0010…

Embedding dimension = vocabulary size

Problems:

- Vector size is too large
- Vectors know nothing about meaning

  e.g., cat is as close to dog as it is to table!

One-Hot Vectors: Represent Words as Discrete Symbols

One is 1, the rest are 0

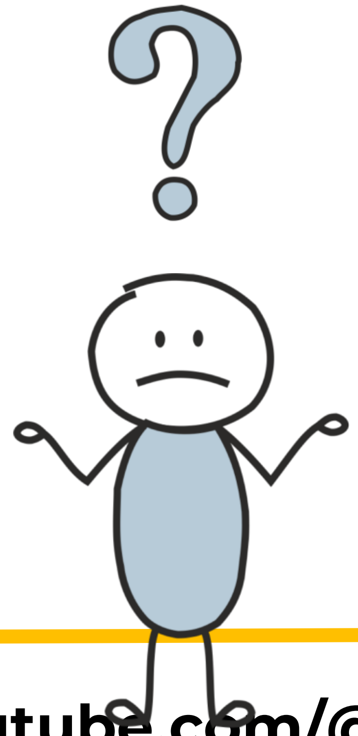
dog	0…0…010….0…0
cat	0…010…0….0…0
table	0…0…0….0010…

Embedding dimension = vocabulary size

Problems:

- Vector size is too large
- Vectors know nothing about meaning

e.g., cat is as close to dog as it is to table!

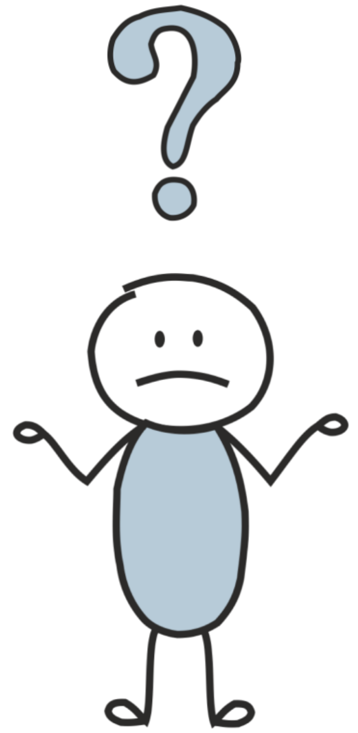


What is meaning?

What is meaning?

Do you know what the word tezgüino means ?

(We hope you do not)



What is meaning?

Now look how this word is used in different contexts:

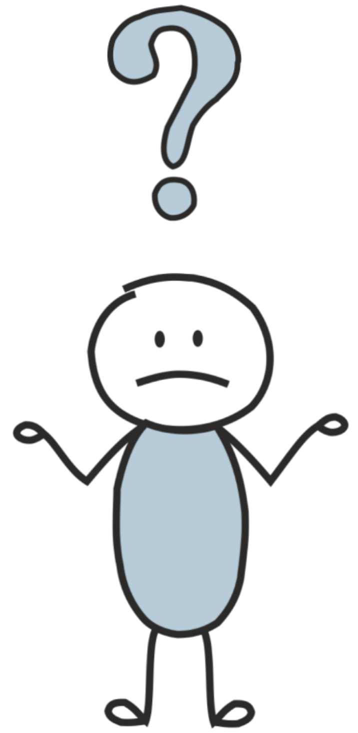
A bottle of tezgüino is on the table.

Everyone likes tezgüino.

Tezgüino makes you drunk.

We make tezgüino out of corn.

Can you understand what tezgüino means ?



What is meaning?

Now look how this word is used in different contexts:

A bottle of tezgüino is on the table.

Everyone likes tezgüino.

Tezgüino makes you drunk.

We make tezgüino out of corn.

Tezgüino is a kind of alcoholic beverage made from corn.

With context, you can understand the meaning!

What is meaning?

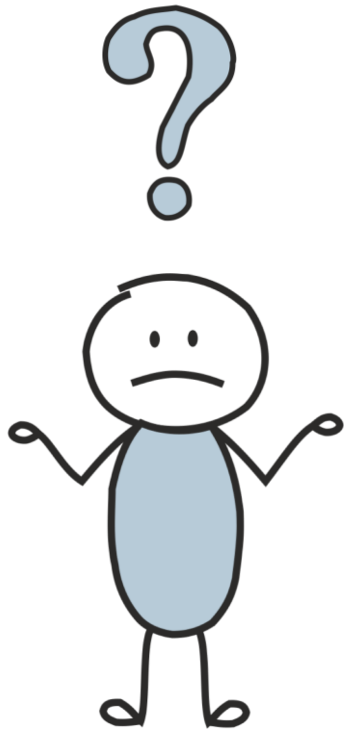
(1) A bottle of ______ is on the table.

(2) Everyone likes ______ .

(3) ______ makes you drunk.

(4) We make ______ out of corn.

What other words fit into these contexts ?



What is meaning?

(1) A bottle of ______ is on the table.

(2) Everyone likes ______ .

(3) ______ makes you drunk.

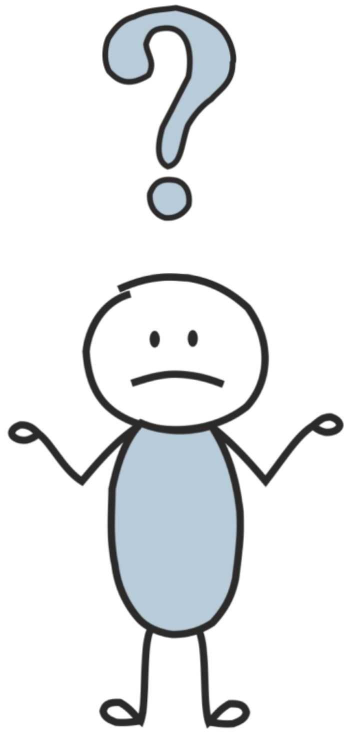
(4) We make ______ out of corn.

What other words fit into these contexts ?

	(1)	(2)	(3)	(4)	…
tezgüino	1	1	1	1	
loud	0	0	0	0	
motor oil	1	0	0	1	
tortillas	0	1	0	1	
wine	1	1	1	0	

← contexts

← rows show contextual properties: 1 if a word can appear in the context, 0 if not



What is meaning?

(1) A bottle of _______ is on the table.

(2) Everyone likes _______ .

(3) _______ makes you drunk.

(4) We make _______ out of corn.

	(1)	(2)	(3)	(4)	…
tezgüino	1	1	1	1	
loud	0	0	0	0	
motor oil	1	0	0	1	
tortillas	0	1	0	1	
wine	1	1	1	0	

rows are similar

What is meaning?

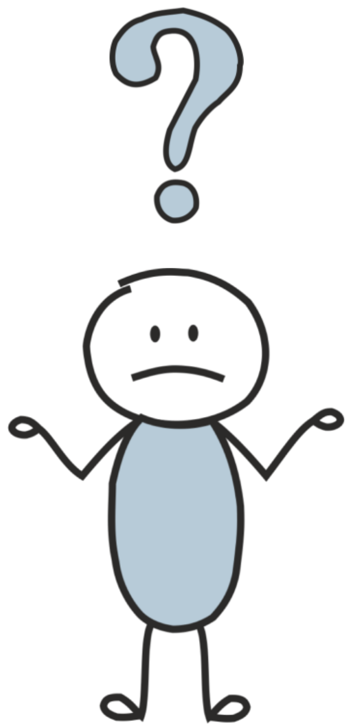
(1) A bottle of _______ is on the table.

(2) Everyone likes _______ .

(3) _______ makes you drunk.

(4) We make _______ out of corn.

	(1)	(2)	(3)	(4)	...
tezgüino	1	1	1	1	
loud	0	0	0	0	
motor oil	1	0	0	1	
tortillas	0	1	0	1	
wine	1	1	1	0	



<u>rows</u> are similar → <u>meanings</u> of the words are similar

Is this true?

What is meaning?

(1) A bottle of _______ is on the table.

(2) Everyone likes _______ .

(3) _______ makes you drunk.

(4) We make _______ out of corn.

	(1)	(2)	(3)	(4)	…
tezgüino	1	1	1	1	
loud	0	0	0	0	
motor oil	1	0	0	1	
tortillas	0	1	0	1	
wine	1	1	1	0	

This is the distributional hypothesis

rows are similar → meanings of the words are similar

Distributional Hypothesis

> Words which frequently appear in similar contexts have similar meaning.

(Harris 1954, Firth 1957)

Distributional Hypothesis

> Words which frequently appear in similar contexts have similar meaning.

(Harris 1954, Firth 1957)

This can be used in practice to build word vectors!

Distributional Hypothesis

> Words which frequently appear in similar contexts have similar meaning.

(Harris 1954, Firth 1957)

<u>Main idea:</u>

> We have to put information about contexts into word vectors.

What comes next: different ways to do this

Count-Based Methods

Count-Based Methods Idea

Let’s remember our main idea:

We have to put information about contexts into word vectors.

Count-Based Methods: Idea

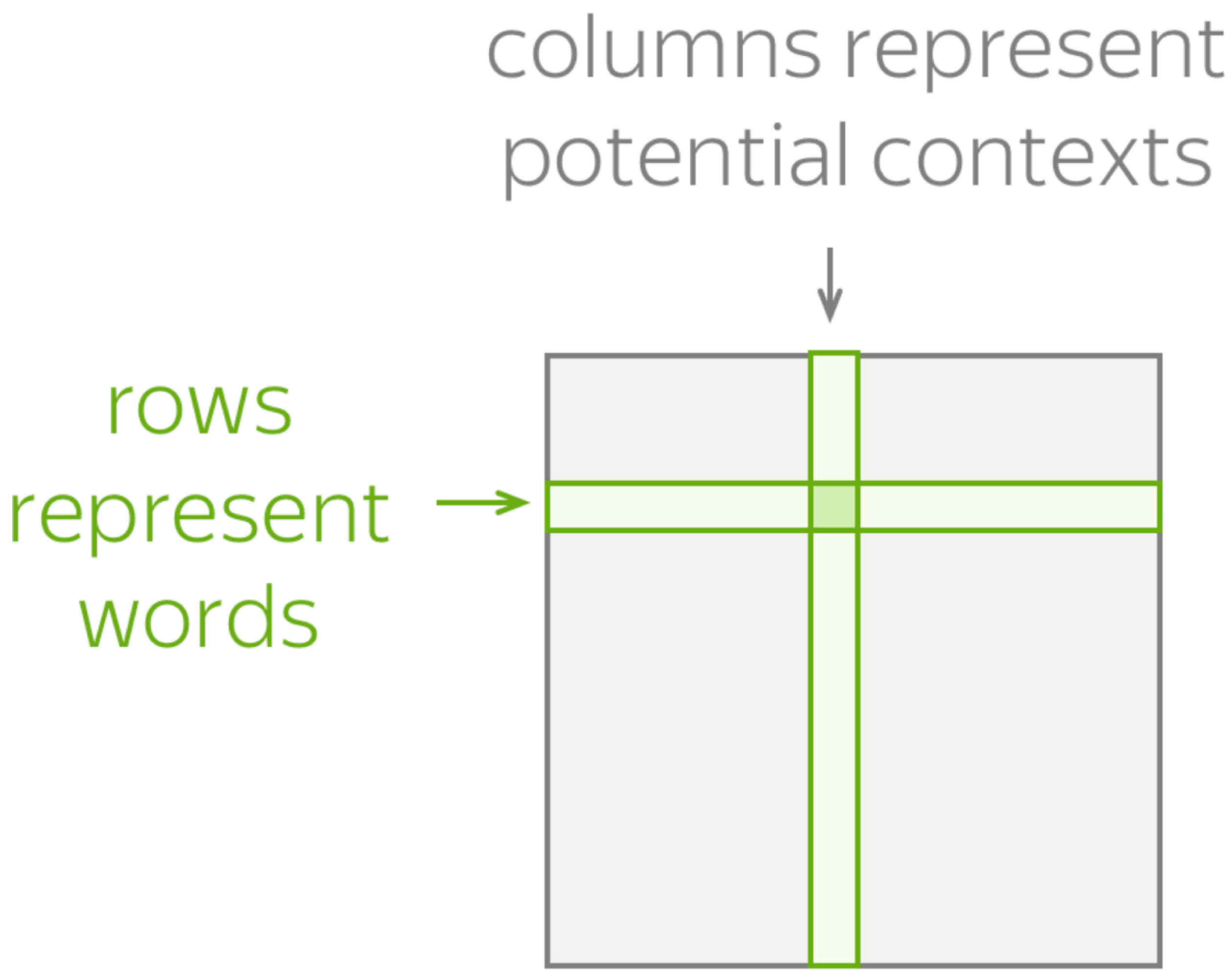
Let's remember our main idea:

We have to put information about contexts into word vectors.

Count-based methods take this idea quite literally :)

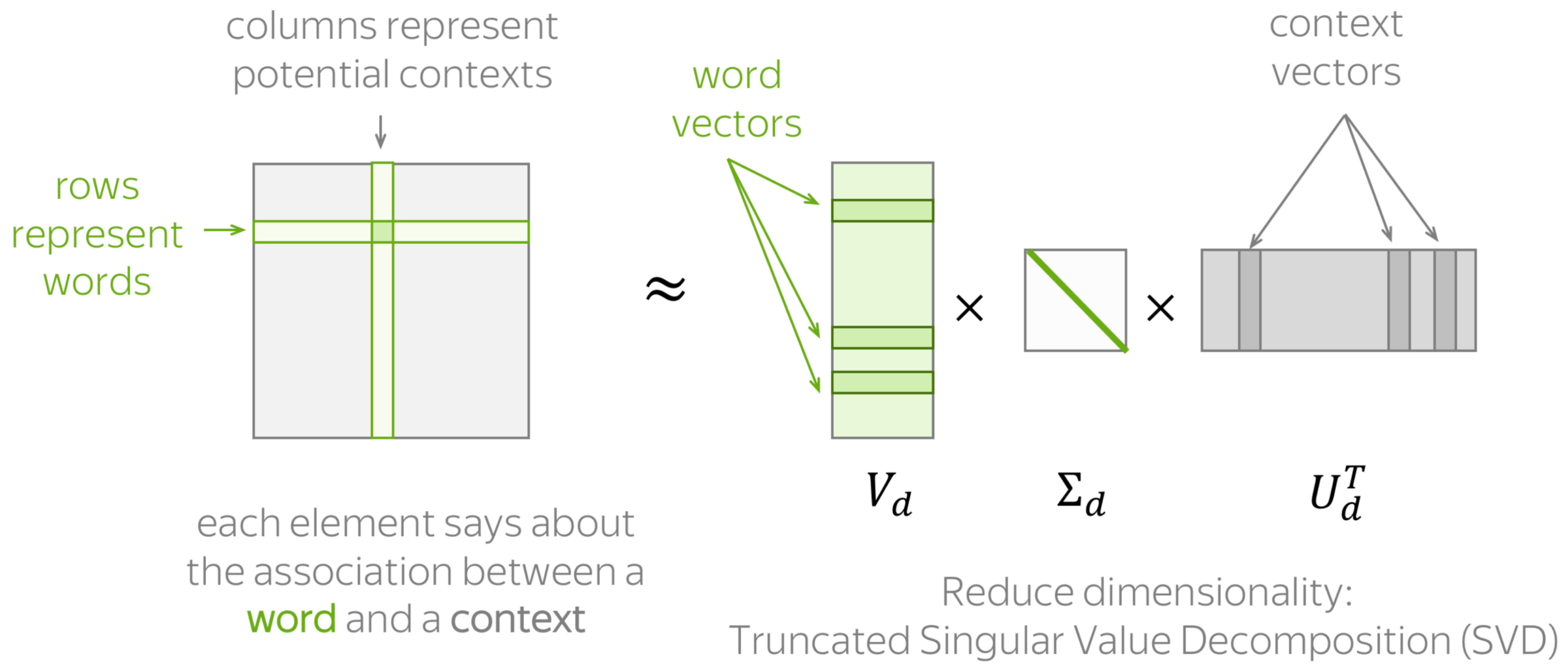
How: Put this information manually based on global corpus statistics.

Count-Based Methods: The General Pipeline

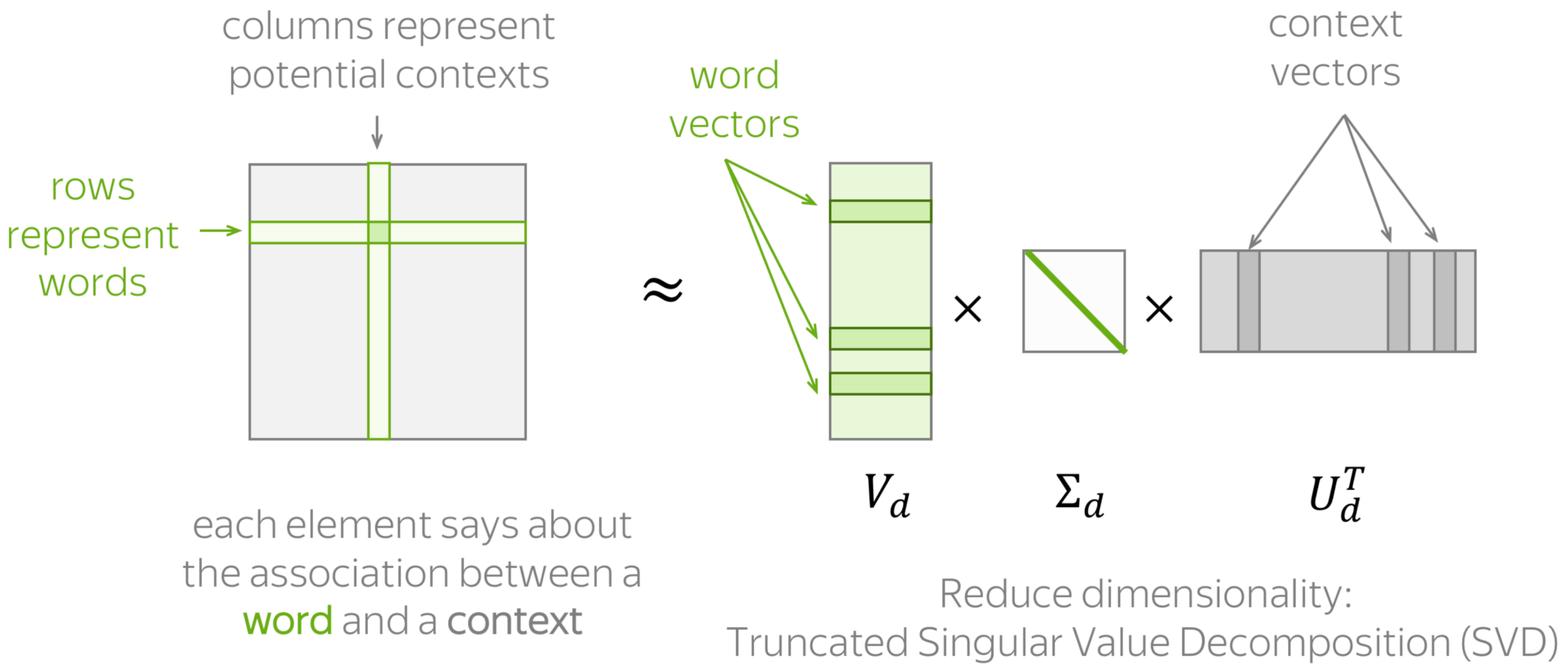


each element says about the association between a word and a context

Count-Based Methods: The General Pipeline



Count-Based Methods: The General Pipeline



Need to define:

- what is context
- how to compute matrix elements

Simple: Co-Occurrence Counts

2-sized window for cat

… I saw a cute grey cat playing in the garden …

contexts for cat

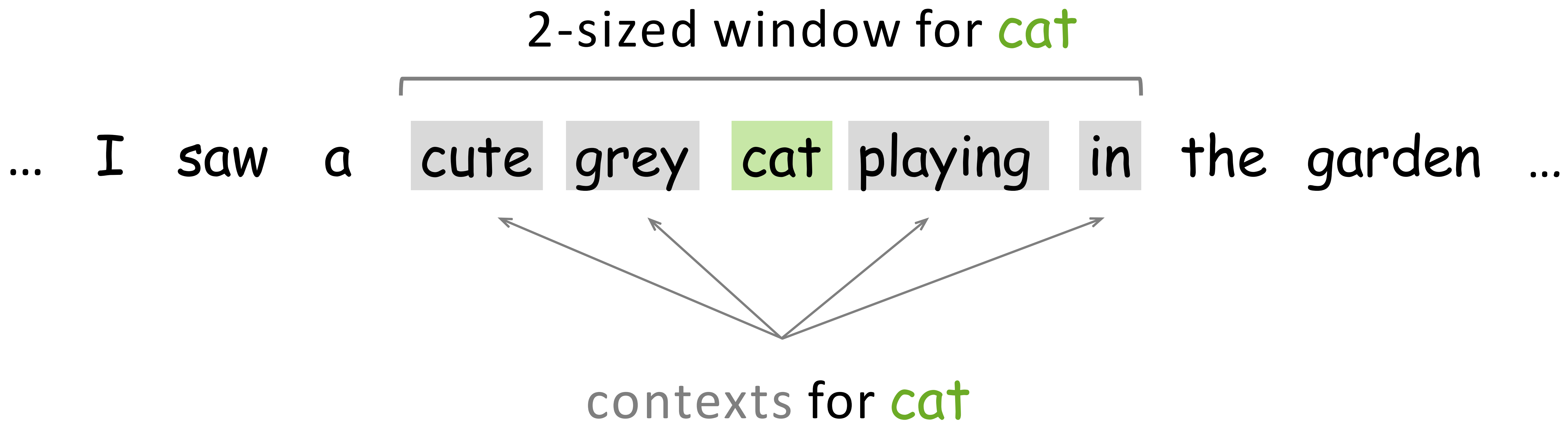
Context:

- surrounding words
  in a L-sized window

Matrix element:

- N(w, c) – number of times
  word w appears in context c

Point Wise Mutual Information (PMI)



Context:

- surrounding words in a L-sized window

$$PMI(w, c) = \log \frac{P(w, c)}{P(w) \cdot P(c)}$$

$P(w,c)$ = probability that w and c occur together (co-occurrence).

$P(w) \cdot P(c)$ = probability that they would occur together if they were independent.

Ratio > 1 (positive PMI) → words occur together more often than chance.

Ratio < 1 (negative PMI) → words occur together less often than chance.

Point wise Mutual Information (PMI)

- "The king rules the kingdom. The queen rules the empire."
- **Step 1: Word probabilities**
  - P(king) = 1/10 = 0.1
  - P(queen) = 1/10 = 0.1
  - P(rules) = 2/10 = 0.2
  - P(the) = 3/10 = 0.3
- Step 2: Co-occurrence (window size = 1)
  - Pair (king, rules) appears 1 time → P(king, rules) = 1/10 = 0.1
  - Pair (king, empire) appears 0 times → P = 0
  - Pair (queen, rules) appears 1 time → P(queen, rules) = 1/10 = 0.1

Point wise Mutual Information (PMI)

- **Step 3: Compute PMI**
- Formula: PMI(w, c) = log(P(w,c) / (P(w) * P(c)))
- PMI(king, rules) = log(0.1 / (0.1 * 0.2))
  - = log(0.1 / 0.02)
  - = log(5) ≈ 1.61 → positive (they co-occur more than chance)
- PMI(king, empire) = log(0 / (0.1 * 0.1))
- = log(0) = negative infinity → strong negative
- **Step 4: Apply PPMI**
- PPMI(king, rules) = 1.61
- PPMI(king, empire) = 0 (we clip negative values to zero)

Point wise Mutual Information (PMI)

- **Interpretation:**
- "king" and "rules" have a high positive PPMI → they are related.
- "king" and "empire" have zero PPMI → no useful relationship.

Word2Vec (Prediction-Based Method)

Word2Vec: Idea

Let's remember our main idea:

We have to put information about contexts into word vectors.

Word2Vec: Idea

Let's remember our main idea:

We have to put information about contexts into word vectors.

Word2Vec uses this idea differently from count-based methods:

How: Learn word vectors by teaching them to predict contexts.

Word2Vec: Idea

How: Learn word vectors by teaching them to predict contexts.

- Learned parameters: word vectors
- Goal: make each vector “know” about the contexts of its word
- How: train vectors to predict possible contexts from words (or, alternatively, words from contexts)

Word2Vec: High-Level pipeline

- take a huge text corpus

… I saw a cute grey cat playing in the garden …

Word2Vec: High-Level pipeline

- take a huge text corpus
- go over the text with a sliding window, moving one word at a time.

…	I	saw	a	cute	grey	cat	playing	in	the	garden	…
	w_{t-2}	w_{t-1}	w_t	w_{t+1}	w_{t+2}						

Word2Vec: High-Level pipeline

- take a huge text corpus
- go over the text with a sliding window, moving one word at a time.

… I saw a cute grey cat playing in the garden …

w_{t-2} w_{t-1} w_t w_{t+1} w_{t+2}

context words | central word | context words

Word2Vec: High-Level pipeline

- take a huge text corpus
- go over the text with a sliding window, moving one word at a time.
- for the central word, compute probabilities of context words;
- adjust the vectors to increase these probabilities.

$P(w_{t-2}|w_t)$ $P(w_{t-1}|w_t)$ $P(w_{t+1}|w_t)$ $P(w_{t+2}|w_t)$

… I saw a cute grey cat playing in the garden …

w_{t-2} w_{t-1} w_t w_{t+1} w_{t+2}

context words | central word | context words

Word2Vec: High-Level pipeline

- take a huge text corpus
- go over the text with a sliding window, moving one word at a time.
- for the central word, compute probabilities of context words;
- adjust the vectors to increase these probabilities.

$P(w_{t-2}|w_t)$ $P(w_{t-1}|w_t)$ $P(w_{t+1}|w_t)$ $P(w_{t+2}|w_t)$

… I saw a cute grey cat playing in the garden …

w_{t-2} w_{t-1} w_t w_{t+1} w_{t+2}

context words | central word | context words

Word2Vec: High-Level pipeline

- take a huge text corpus
- go over the text with a sliding window, moving one word at a time.
- for the central word, compute probabilities of context words;
- adjust the vectors to increase these probabilities.

$P(w_{t-2}|w_t)$ $P(w_{t-1}|w_t)$ $P(w_{t+1}|w_t)$ $P(w_{t+2}|w_t)$

… I saw a cute grey cat playing in the garden …

w_{t-2} w_{t-1} w_t w_{t+1} w_{t+2}

context words | central word | context words

Word2Vec: High-Level pipeline

- take a huge text corpus
- go over the text with a sliding window, moving one word at a time.
- for the central word, compute probabilities of context words;
- adjust the vectors to increase these probabilities.

$P(w_{t-2}|w_t)$ $P(w_{t-1}|w_t)$ $P(w_{t+1}|w_t)$ $P(w_{t+2}|w_t)$

… I saw a cute grey cat playing in the garden …

w_{t-2} w_{t-1} w_t w_{t+1} w_{t+2}

context words | central word | context words

Word2Vec: High-Level pipeline

- take a huge text corpus
- go over the text with a sliding window, moving one word at a time.
- for the central word, compute probabilities of context words;
- adjust the vectors to increase these probabilities.

$P(w_{t-2}|w_t)$ $P(w_{t-1}|w_t)$ $P(w_{t+1}|w_t)$ $P(w_{t+2}|w_t)$

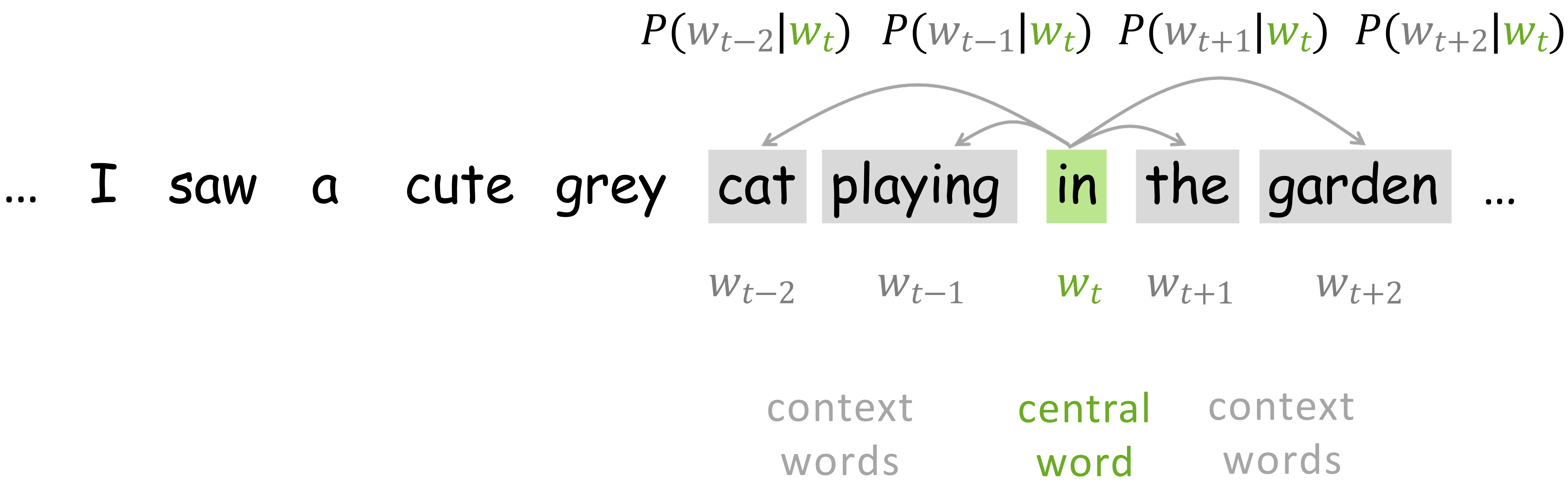
… I saw a cute grey cat playing in the garden …

w_{t-2} w_{t-1} w_t w_{t+1} w_{t+2}

context words | central word | context words

Word2Vec: High-Level pipeline

- take a huge text corpus
- go over the text with a sliding window, moving one word at a time.
- for the central word, compute probabilities of context words;
- adjust the vectors to increase these probabilities.



How Does Word2Vec Work?

- Train NN on a **fake problem**: predict missing word
- Example:
  - "The emperor ordered his ___."
- Correct word could be *king* or *minister*
- Goal: **not prediction**, but to **learn embeddings** (side effect)

Context Is Key

- Meaning of a word depends on **context**
- Example:
  - “Eating ___ is healthy.” → [apple, walnut] ✅
  - Not [pizza, truck] ❌
- Context words = training input

Objective Function: Negative Log-Likelihood

- **Word2Vec tries to find the parameters that maximize the data likelihood:**
- **wt** → → the current (center) word at position t.
- **wt+j** → a context word around wt.
- **m** → the size of the context window.
- **T** → total number of words in the corpus.
- P(wt+j|wt,θ) → the probability of seeing context word wt+j given center word wt, with parameters θ (neural network weights).
- So:

  The likelihood is the product of probabilities of predicting the context words for every center word in the training corpus.

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m, \\ j \ne 0}} P(w_{t+j} | w_t, \theta)$$

Objective Function: Intuition (Why It Works)

- Training Word2Vec = making the model assign **high probability** to actual context words.
- If the model predicts wrong context words, probability is low → log-likelihood decreases → NLL increases.
- So by minimizing NLL, the model **learns embeddings that bring context words closer in vector space**.

Objective Function: Intuition (Why It Works)

- **Simple Analogy**
- Imagine the sentence: **"The king ruled the empire."**
- Center word = *king*
- Context words (window=2): *the*, *ruled*
- If the model says: P(ruled|king)=0.7 and P(the|king)=0.6
- High probabilities → good embeddings.

  If probabilities were low (e.g. 0.05), NLL would be large, pushing the model to adjust embeddings.

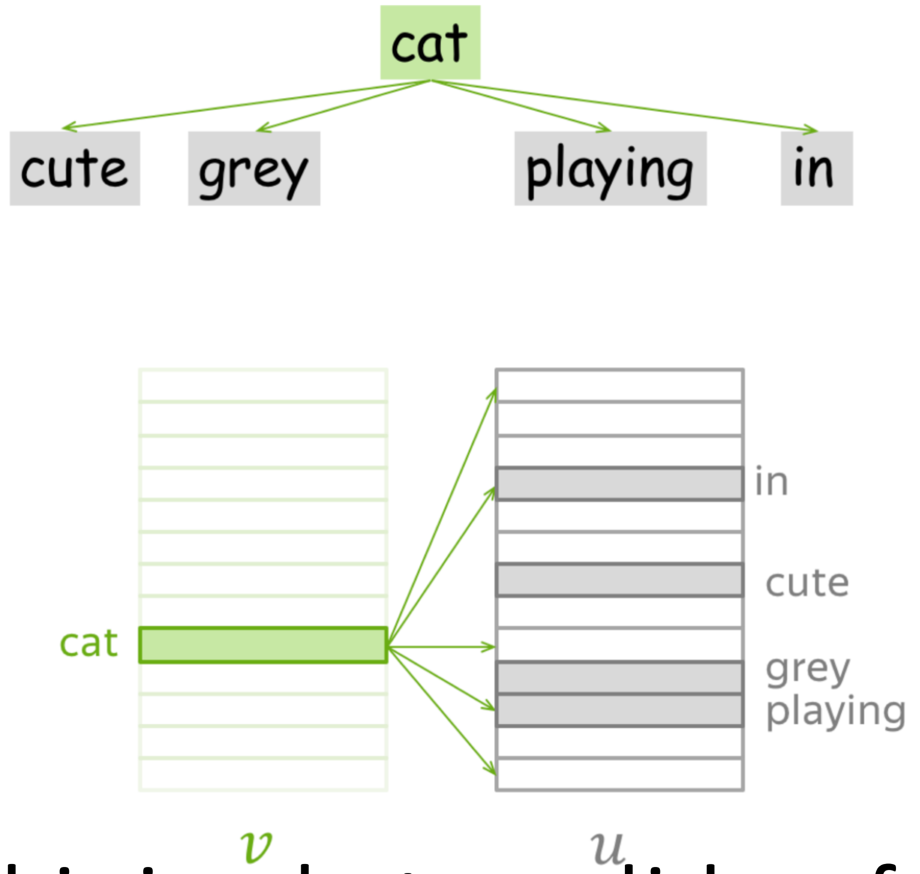
Objective Function: Intuition (Why It Works)

- Likelihood = probability of seeing all context words given their center words.
- Log-likelihood = makes computation manageable.
- Negative log-likelihood = the loss function we minimize, so embeddings improve.

Word2Vec Variants: Skip-Gram and CBOW

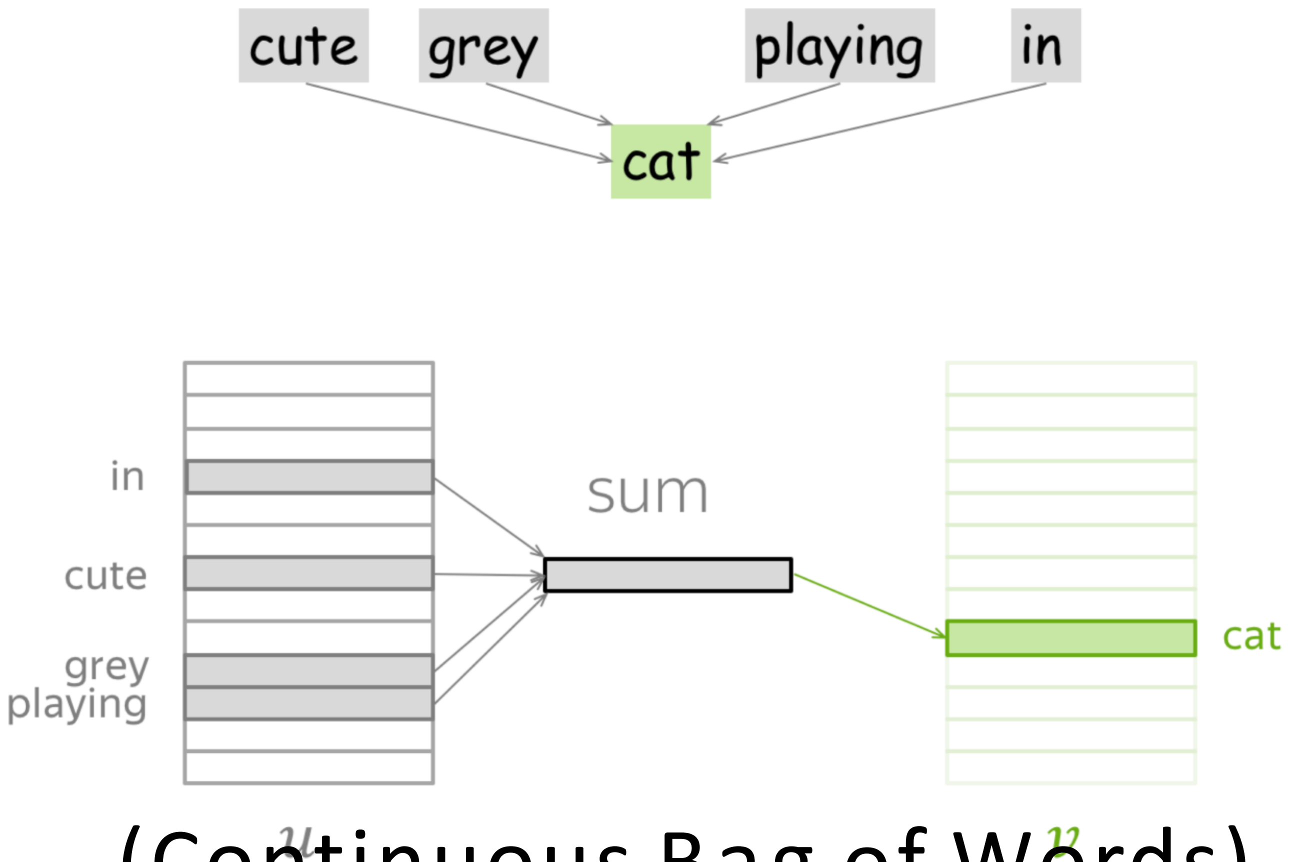
… I saw a cute grey cat playing in the garden …

Skip-Gram: from central predict context
(one at a time)



(this is what we did so far)

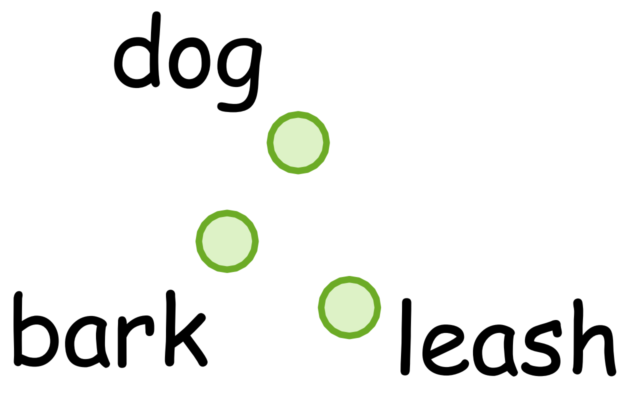
CBOW: from sum of context predict central



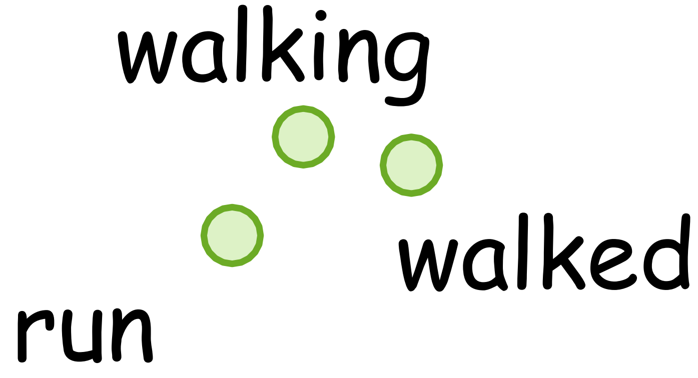
(Continuous Bag of Words)

The Effect of Context Window Size

- Larger windows – more topical similarities

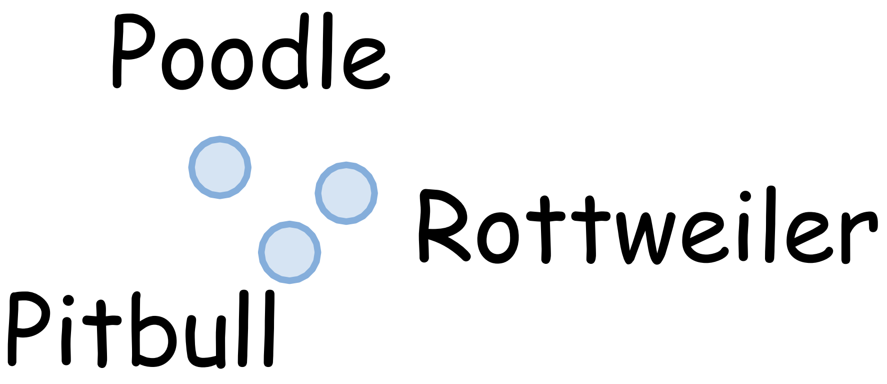


(grouped together)



(grouped together)

- Smaller windows – more functional and syntactic similarities



(grouped together)

(grouped together)