Word Embeddings

Lena Voita

Lecture-blog and lots of additional materials are here:
https://lena-voita.github.io/nlp_course/word_embeddings.html

What is going to happen:

- Why do we need word representations?
- One-hot Vectors
- Distributional Semantics
- Count-Based Methods
- Word2Vec (Prediction-based Method)
- GloVe
- Evaluation
-  Analysis and Interpretability

What is going to happen:

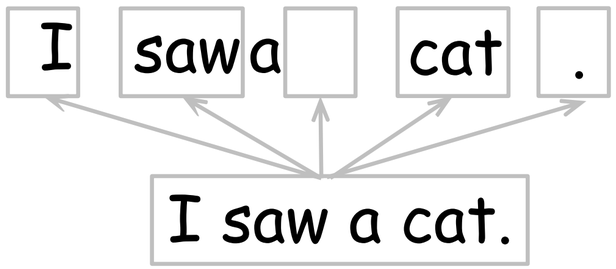
- Why do we need word representations?
- One-hot Vectors
- Distributional Semantics
- Count-Based Methods
- Word2Vec (Prediction-based Method)
- GloVe
- Evaluation
-  Analysis and Interpretability

Why do we need word representations?

I saw a cat.

Text (your input)

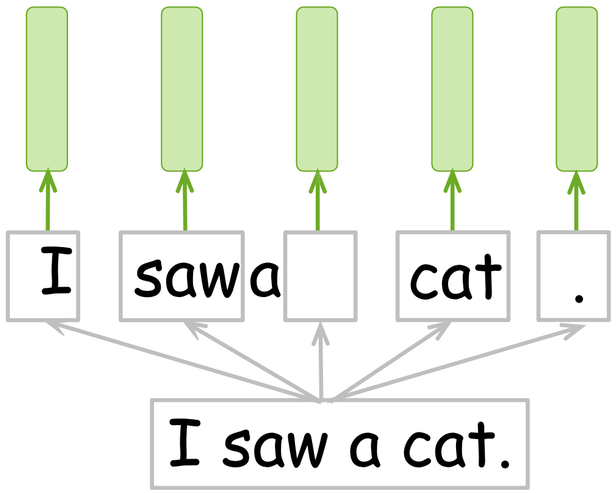
Why do we need word representations?



Sequence of tokens

Text (your input)

Why do we need word representations?

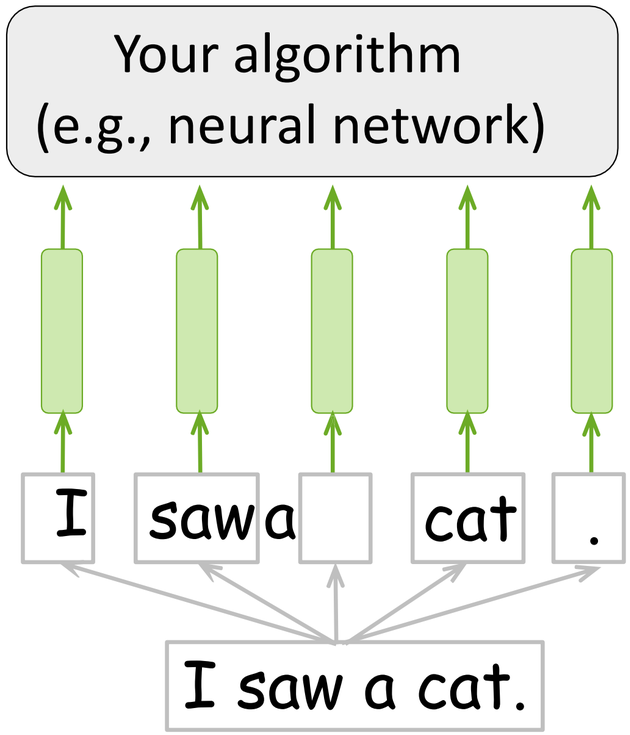


Word representation - vector
(input for your model/algorithm)

Sequence of tokens

Text (your input)

Why do we need word representations?



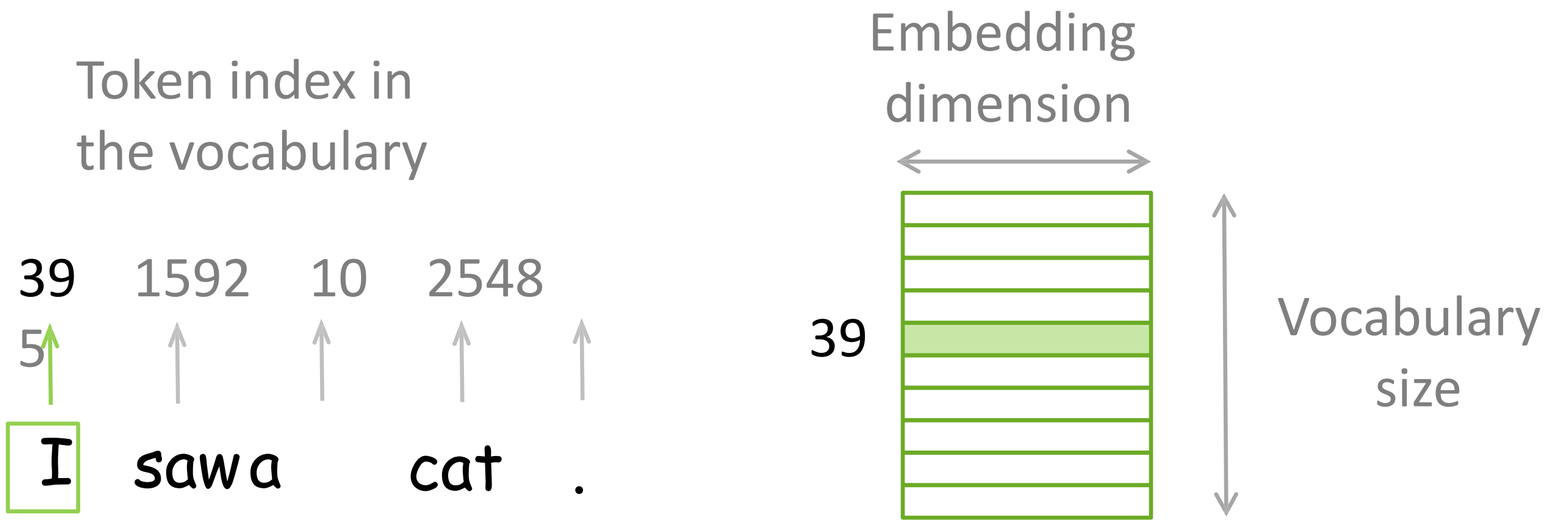
Any algorithm for solving a task

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

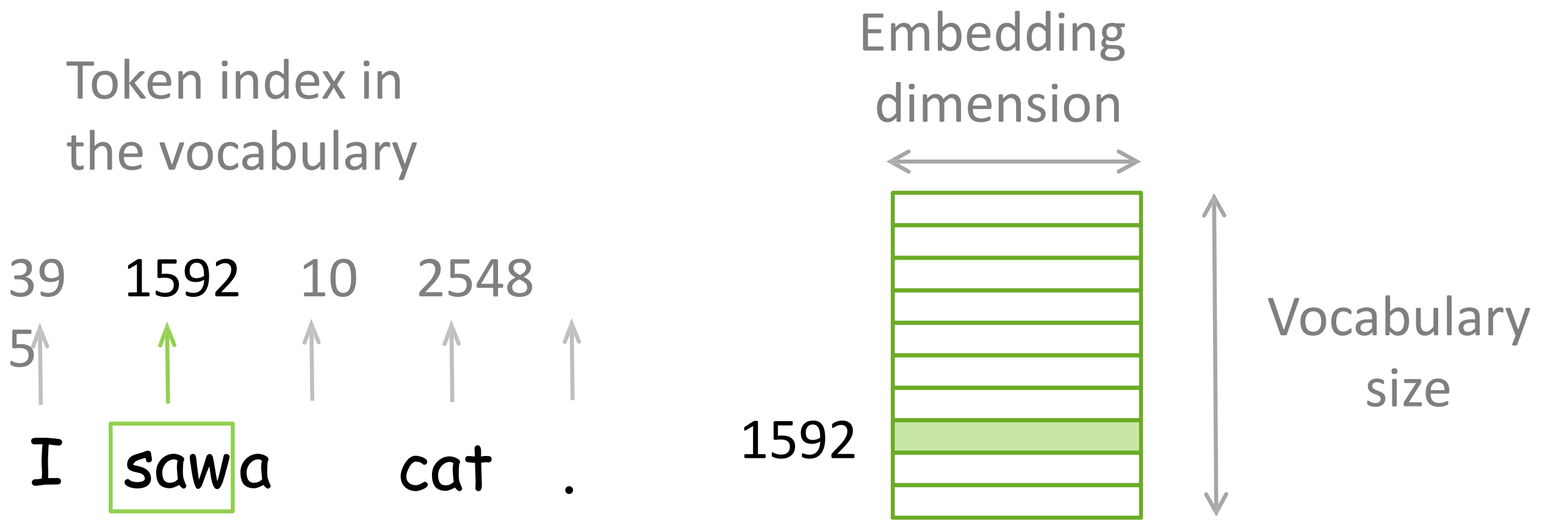
Sequence of tokens

Text (your input)

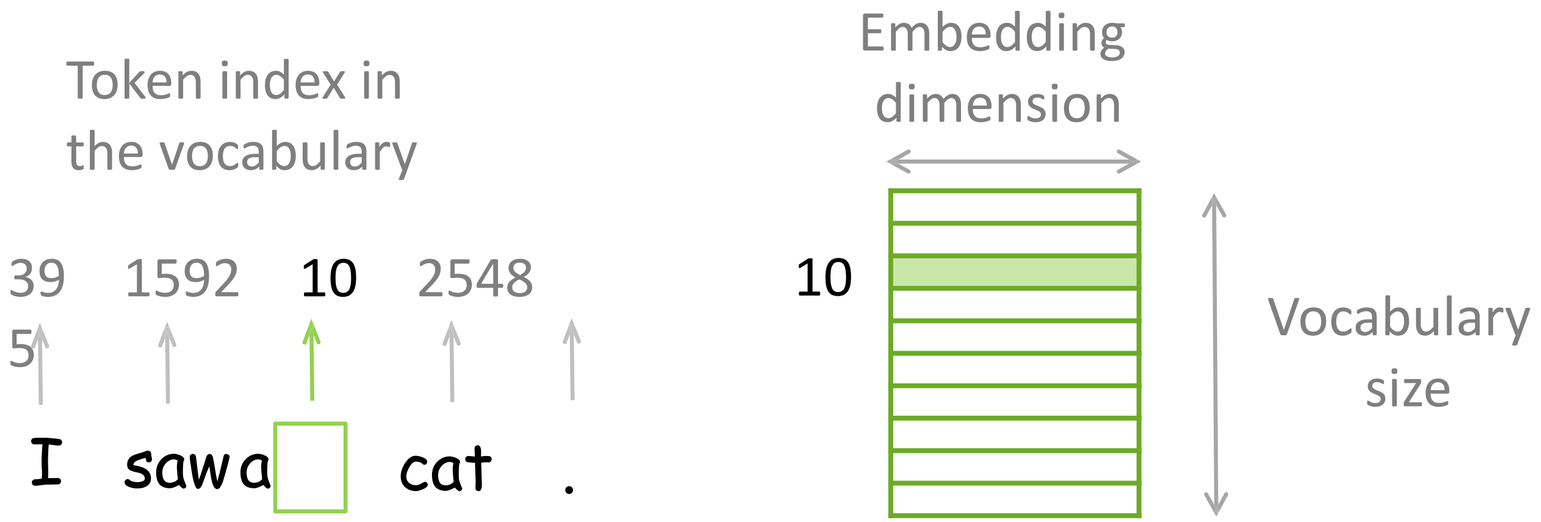
How it works: Look-up Table



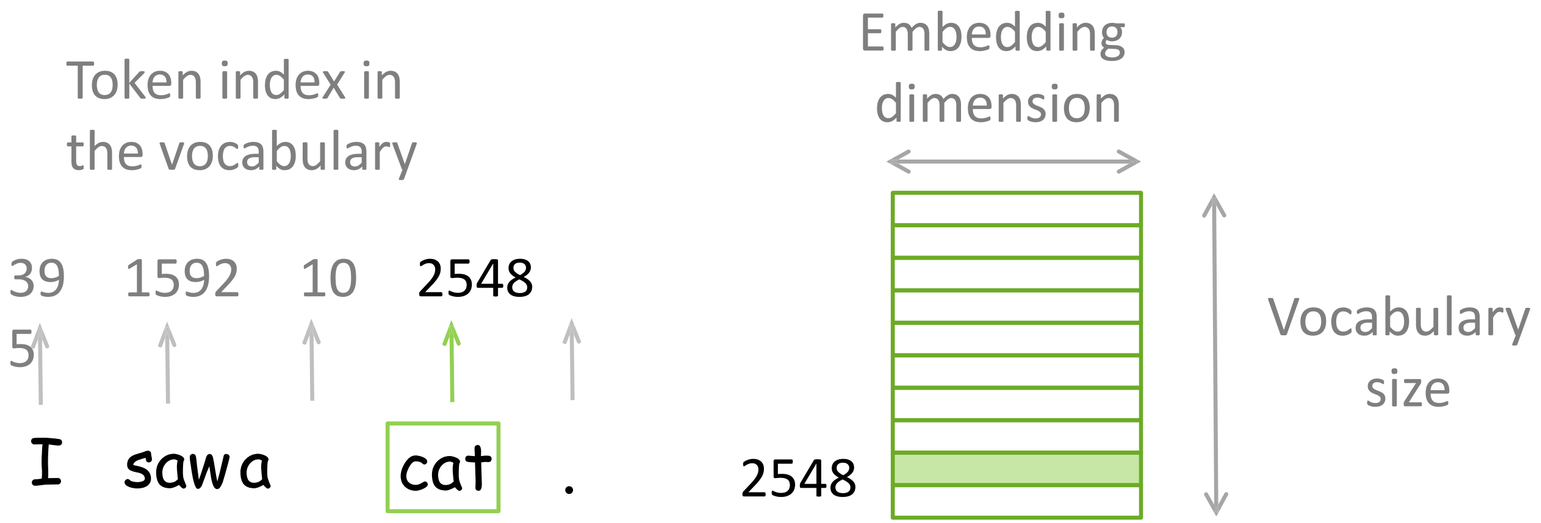
How it works: Look-up Table



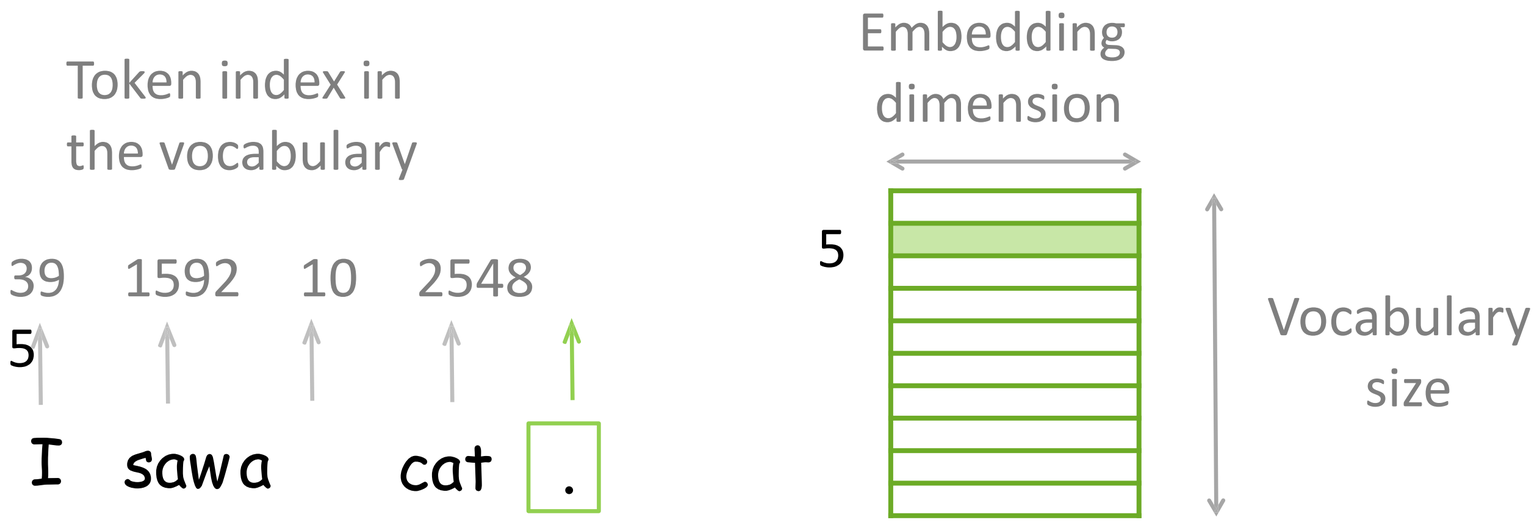
How it works: Look-up Table



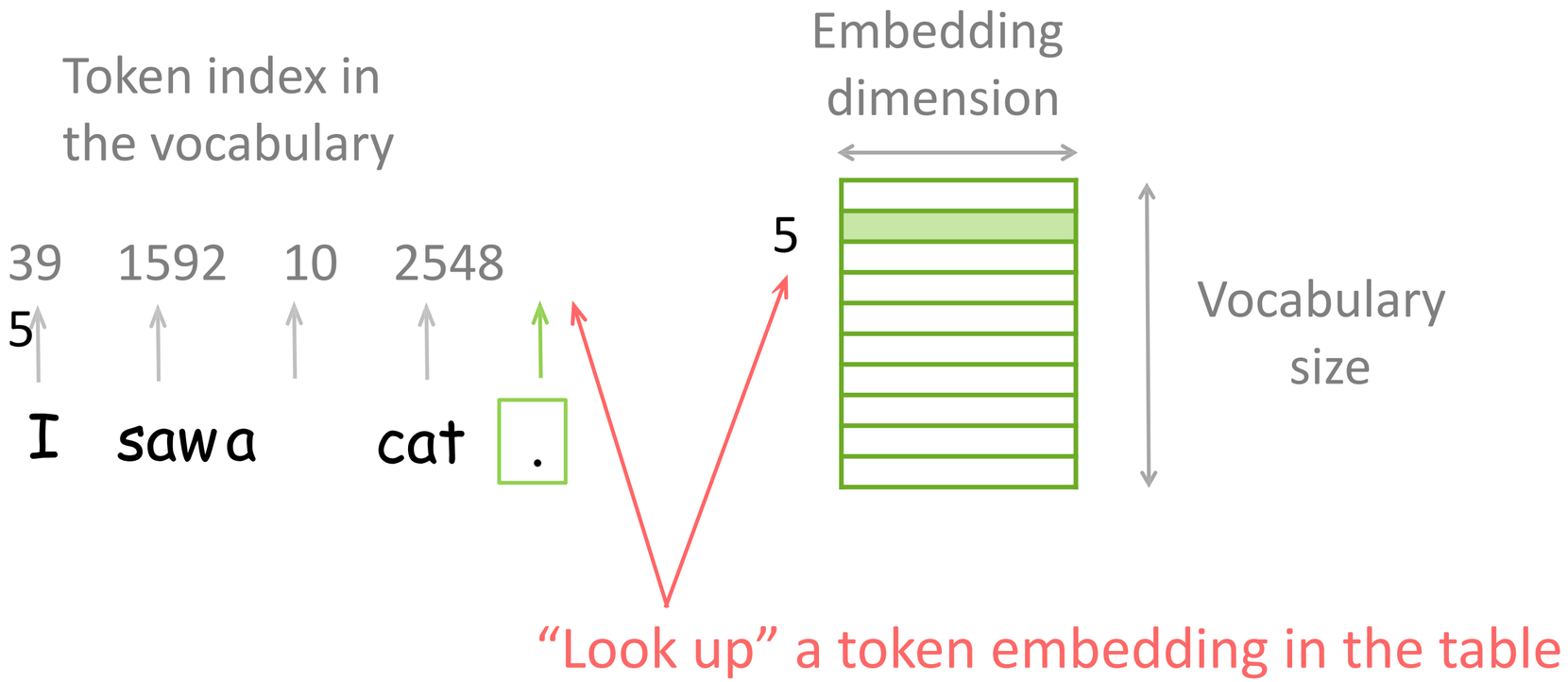
How it works: Look-up Table



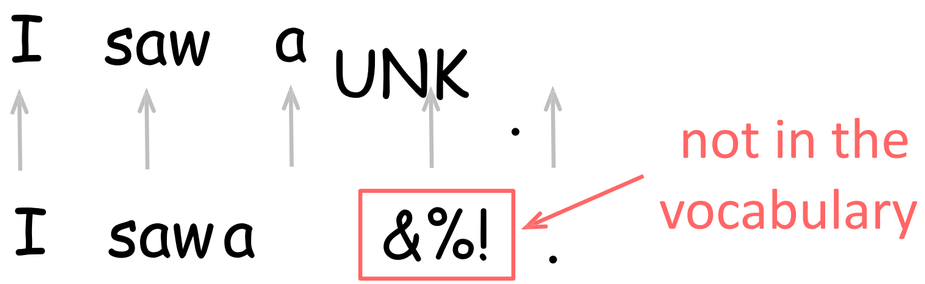
How it works: Look-up Table



How it works: Look-up Table



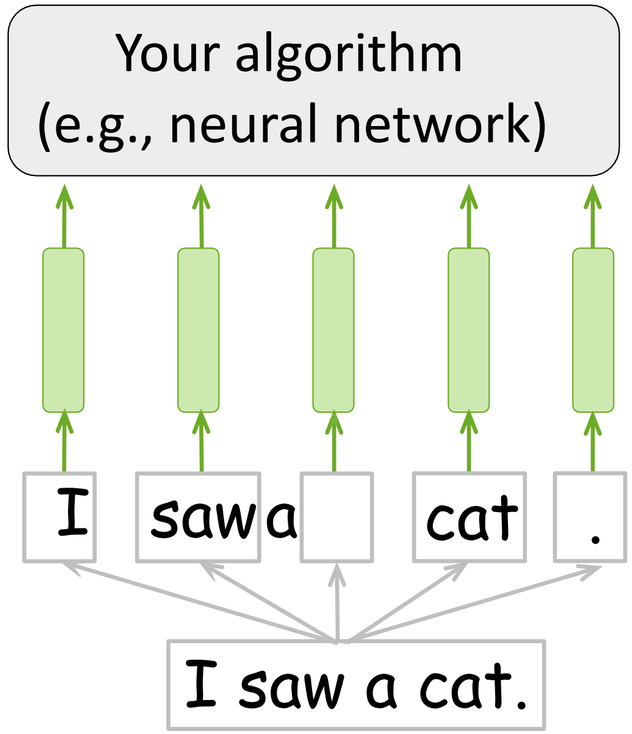
Note UNKs: Out-of-Vocabulary Tokens



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?



In the following:

← How can we get these representations?

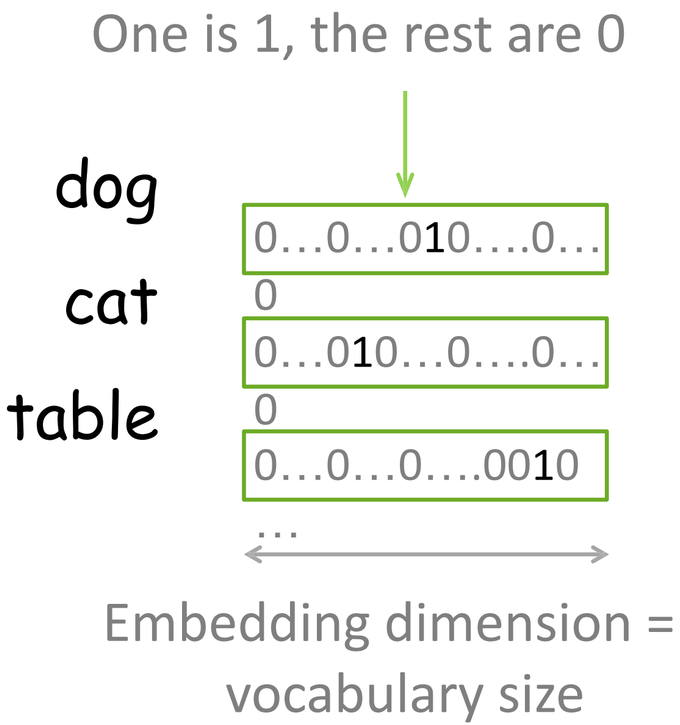
What is going to happen:

- Why do we need word representations?
- One-hot Vectors
- Distributional Semantics
- Count-Based Methods
- Word2Vec (Prediction-based Method)
- GloVe
- Evaluation
-  Analysis and Interpretability

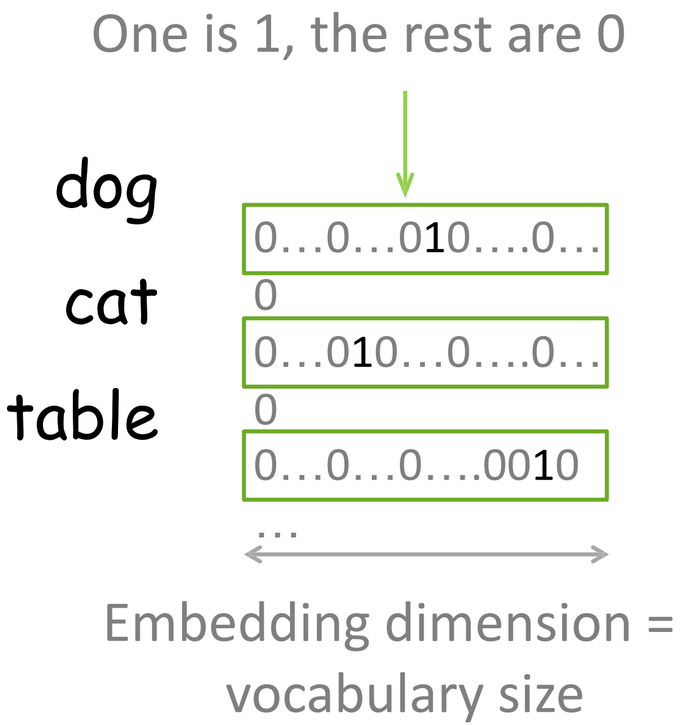
What is going to happen:

- Why do we need word representations?
- One-hot Vectors
- Distributional Semantics
- Count-Based Methods
- Word2Vec (Prediction-based Method)
- GloVe
- Evaluation
-  Analysis and Interpretability

One-Hot Vectors: Represent Words as Discrete Symbols

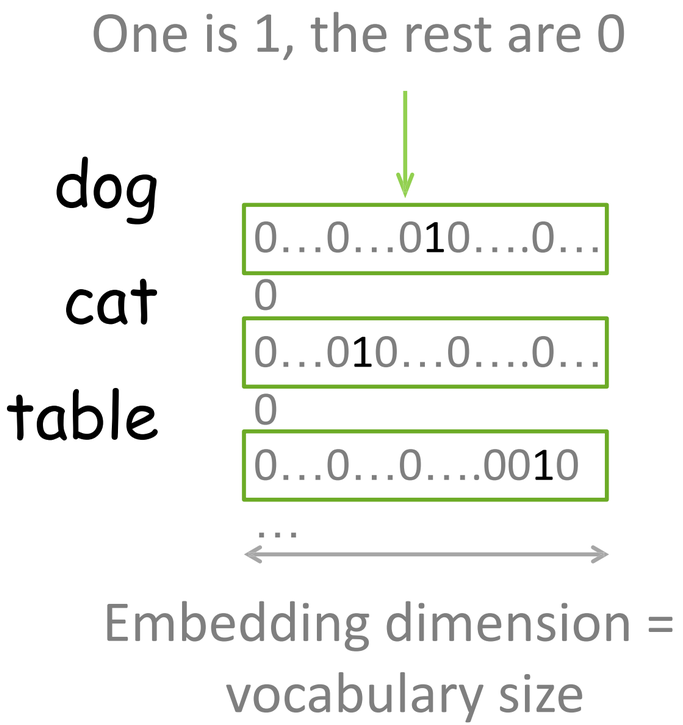


One-Hot Vectors: Represent Words as Discrete Symbols



Any problems?

One-Hot Vectors: Represent Words as Discrete Symbols



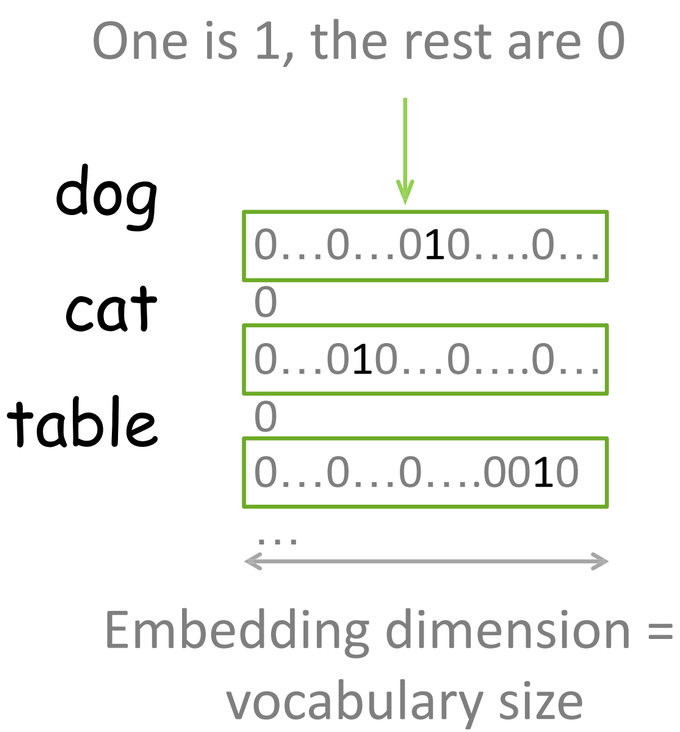
Problems:

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

e.g., cat is as close to

dog as it is to table!

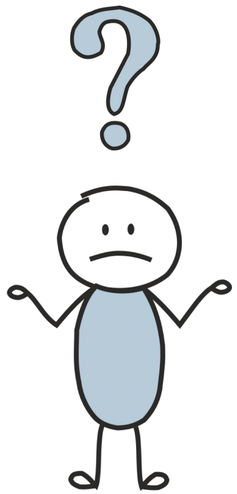
One-Hot Vectors: Represent Words as Discrete Symbols



Problems:

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

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

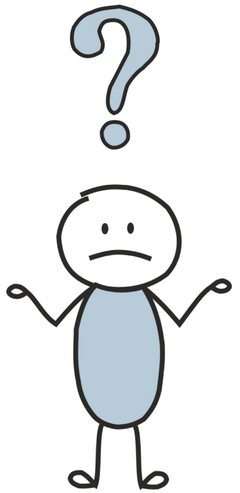


What is meaning?

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:

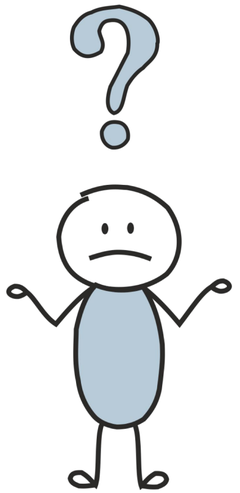
A bottle of tezgüino is on the table.

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:

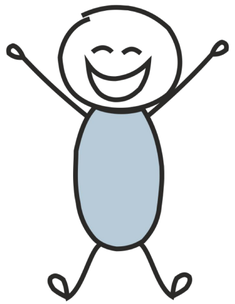
A bottle of tezgüino is on the table.

Everyone likes tezgüino.

Tezgüino makes you drunk.

We make tezgüino out of corn.

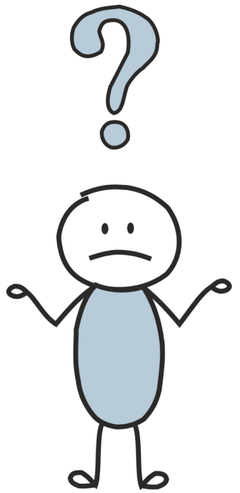
Tezgüino is a kind of alcoholic beverage made from corn.



With context, you can understand the meaning!

What is meaning?

How did you do this?



What is meaning?

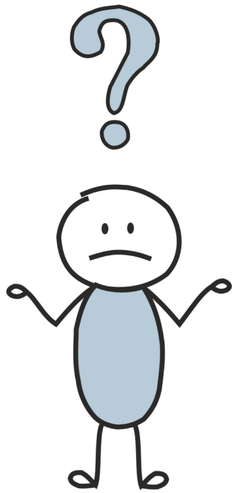
(1) A bottle of _____ is on the table.

(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.

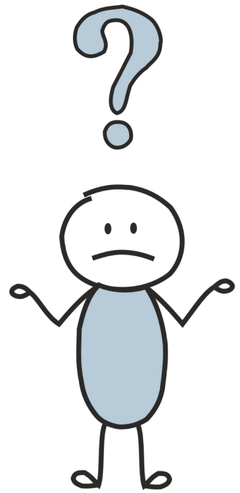
(4) We make out of corn.

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?

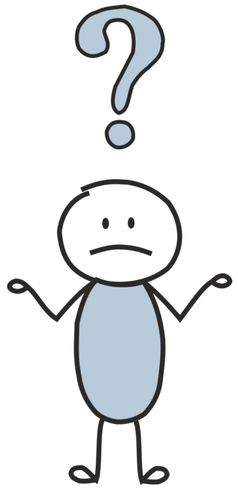
(1) A bottle of _____ is on the table.

(2) Everyone likes _____.

(3) _____ makes you drunk.

(4) We makeout 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 → meanings 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 makeout 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

<u>rows</u> are similar → <u>meanings</u> 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)

Main idea:

We have to put information about contexts into word vectors.

What comes next: different ways to do this

What is going to happen:

- Why do we need word representations?
- One-hot Vectors
- Distributional Semantics
- Count-Based Methods
- Word2Vec (Prediction-based Method)
- GloVe
- Evaluation
-  Analysis and Interpretability

What is going to happen:

- Why do we need word representations?
- One-hot Vectors
- Distributional Semantics
- Count-Based Methods
- Word2Vec (Prediction-based Method)
- GloVe
- Evaluation
-  Analysis and Interpretability

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

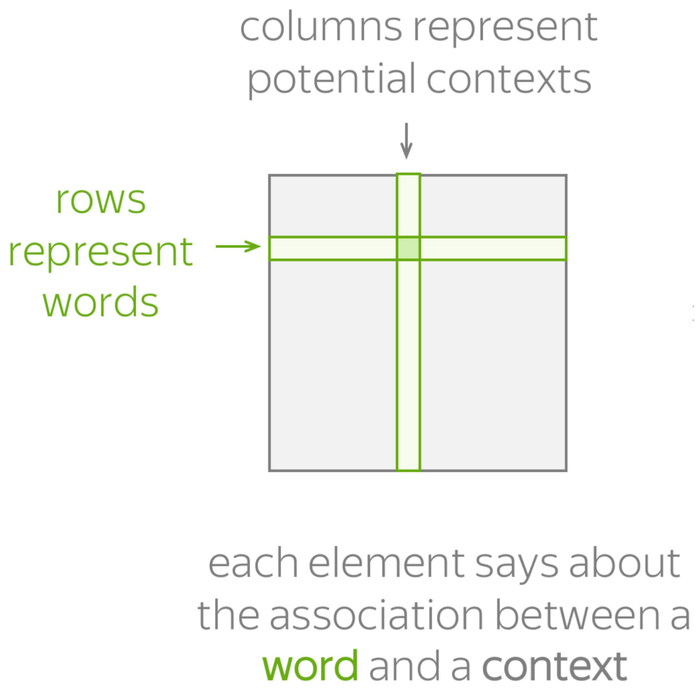
Let’s remember our main idea:

We have to put information about contexts into word vectors.

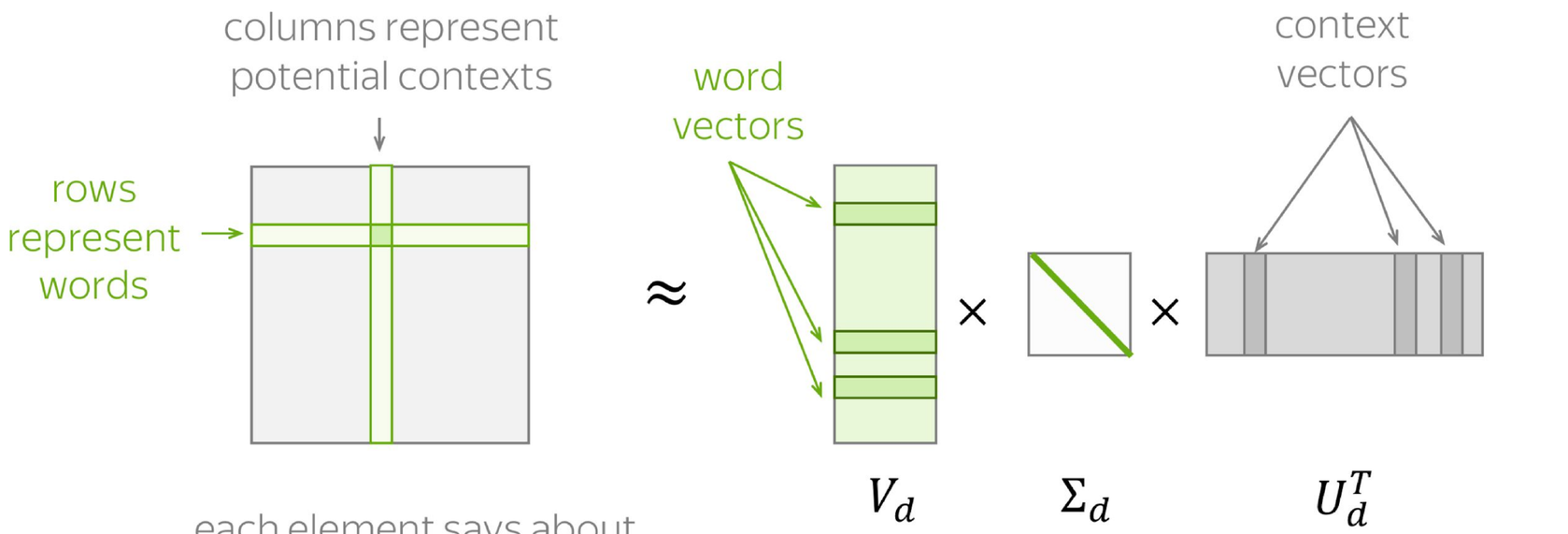
Count-based methods take this idea quite literally :)

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

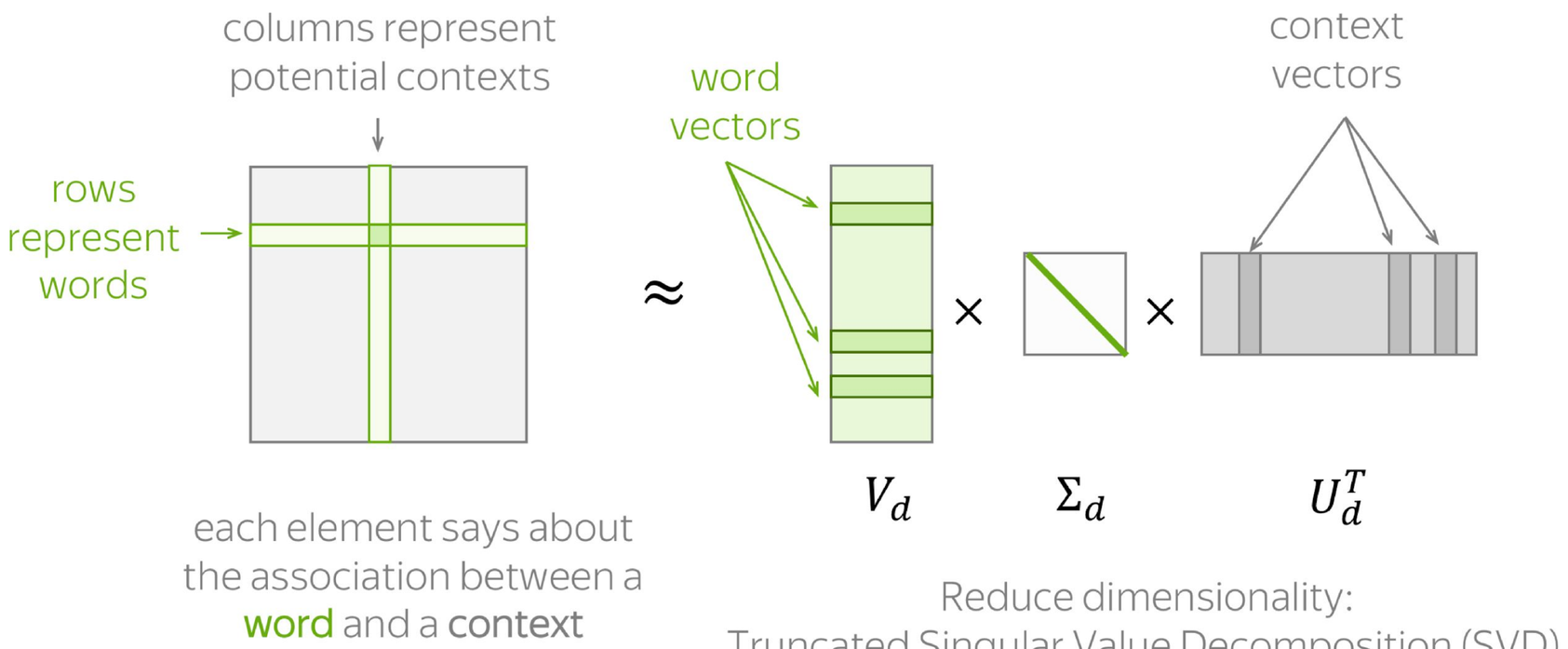
Count-Based Methods: The General Pipeline



Count-Based Methods: The General Pipeline



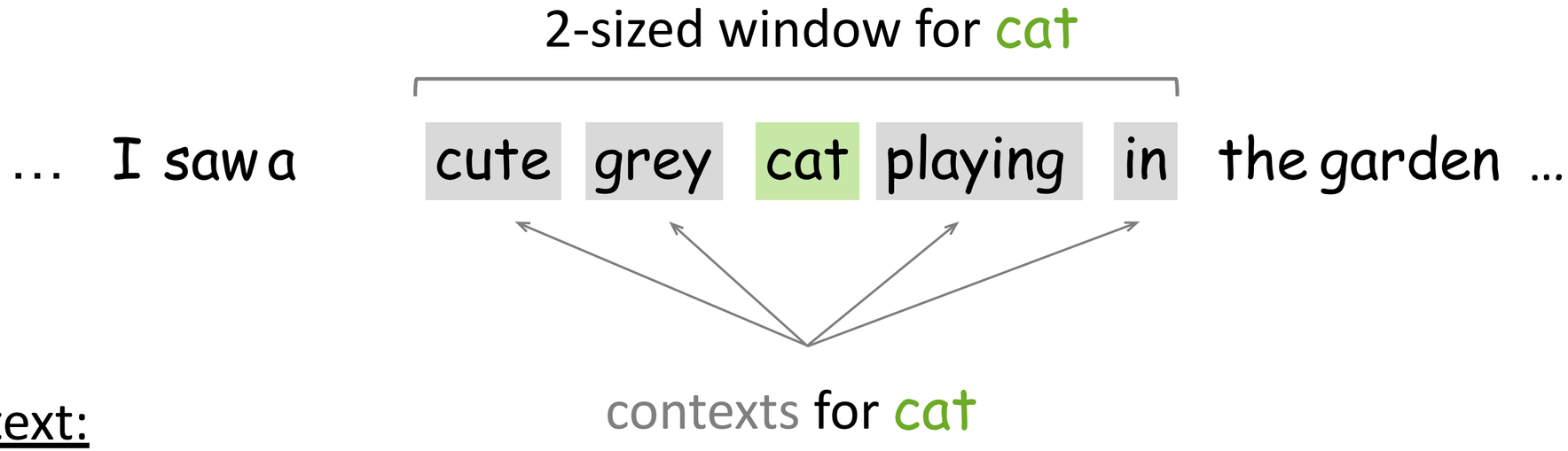
Count-Based Methods: The General Pipeline



Need to define:

- what is context
- how to compute matrix elements

Simple: Co-Occurrence Counts



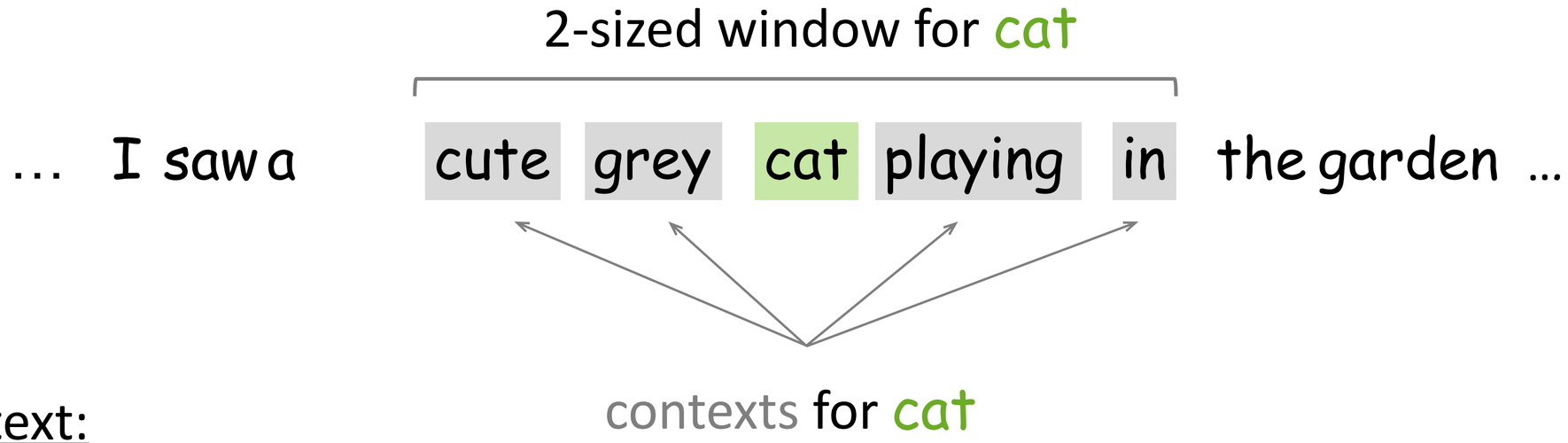
Context:

- surrounding words
  in a L-sized window

Matrix element:

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

Simple: Co-Occurrence Counts



Context:

- surrounding words
  in a L-sized window

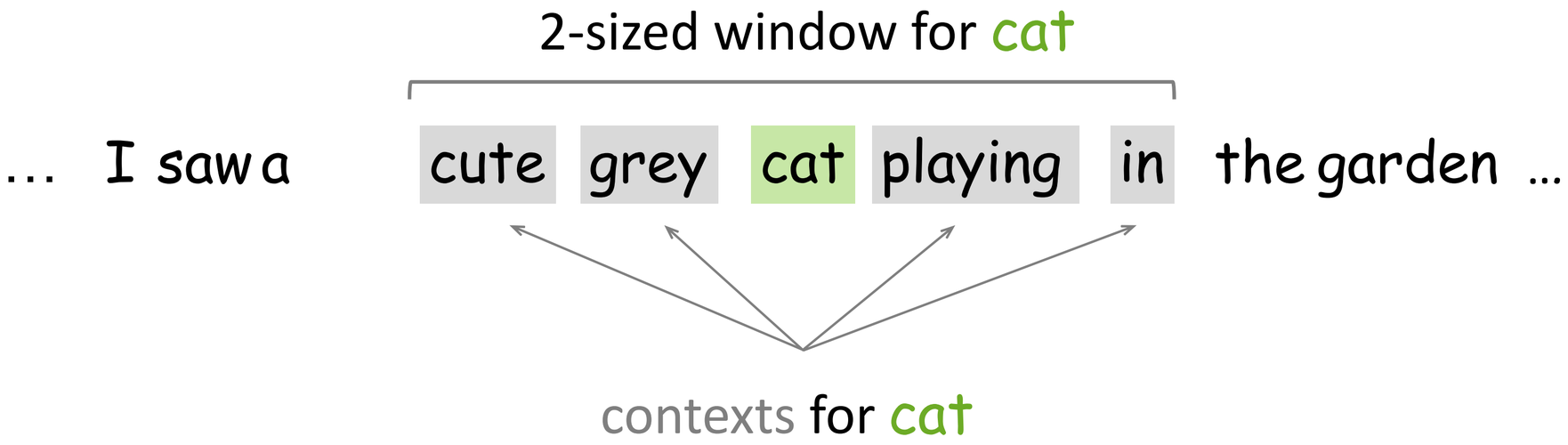
Matrix element:

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

NLP Course For You

The (once) famous HAL model (1996) is also a modification of this approach. Learn more from this exercise in the Research Thinking section.

Positive Pointwise Mutual Information (PPMI)



Context:

- surrounding words in a L-sized window

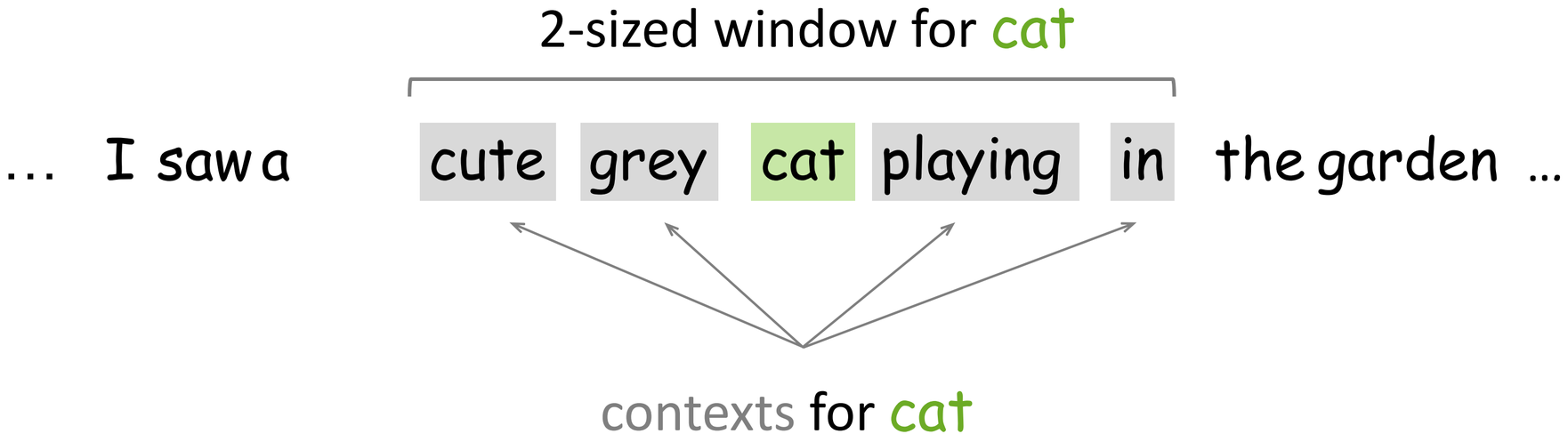
Matrix element:

- PPMI(w, c) = max(0, PMI(w, c)),

where

$$\mathrm{PMI}(w, c) = \log \frac{P(\&, ()}{P(\&)P(()} = \log \frac{|(\&, ()|}{+(\&)+(()}$$

Positive Pointwise Mutual Information (PPMI)



Context:

- surrounding words in a L-sized window

Matrix element:

- PPMI(w, c) = max(0, PMI(w, c)),

where

$$\text{PMI}(w, c) = \log \frac{P(\&, ())}{P(\&)P(())} = \log \frac{|(\&, ())| \cdot +(\&, ())}{+(\&)\ +(())}$$

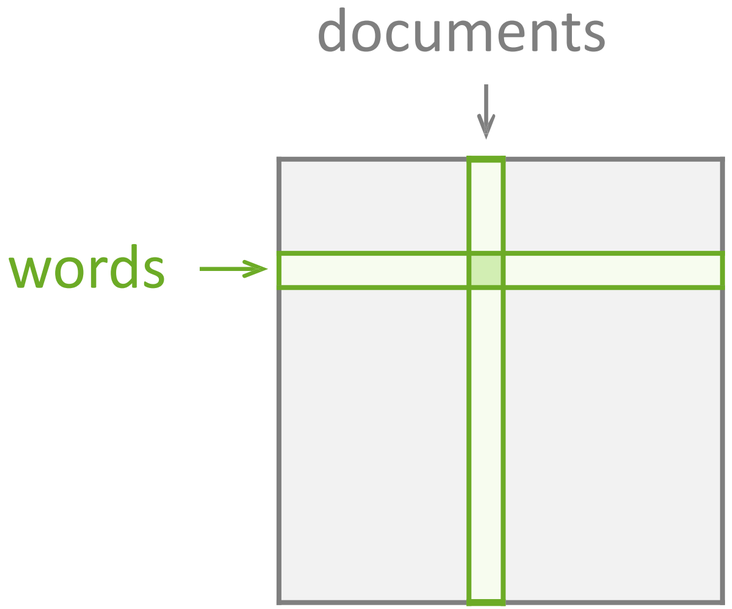
How much one variable tells about another

Latent Semantic Analysis (LSA):

Understanding Documents

<u>Context:</u>

- document d (from a collection D)



Each element is the association between a word and a document

Latent Semantic Analysis (LSA): Understanding Documents

<u>Context:</u>

- document d (from a collection D)

<u>Matrix element:</u>

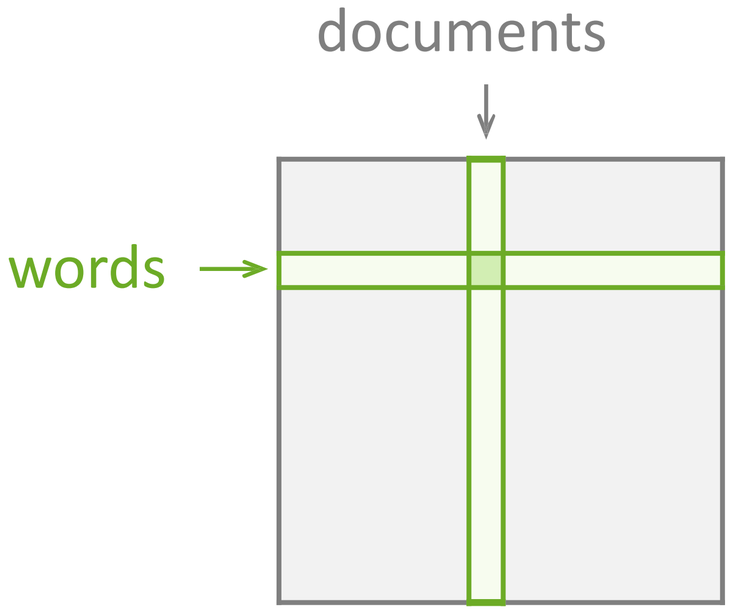
- tf-idf(w, d, D) = tf(w, d) ! idf(w, D)

N(w, d)

term frequency

$$\log \frac{|D|}{|\{d \in D:\ w \in D\}|}$$

inverse document frequency



Each element is the association between a word and a document

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.

What is going to happen:

- Why do we need word representations?
- One-hot Vectors
- Distributional Semantics
- Count-Based Methods
- Word2Vec (Prediction-based Method)
- GloVe
- Evaluation
-  Analysis and Interpretability

What is going to happen:

- Why do we need word representations?
- One-hot Vectors
- Distributional Semantics
- Count-Based Methods
- Word2Vec (Prediction-based Method)
- GloVe
- Evaluation
-  Analysis and Interpretability

Word2Vec
(Prediction-Based Method)

What’s inside:

Word2Vec

- Idea
- Objective Function
- Training Procedure
- Faster Training: Negative Sampling
- Word2Vec versions: Skip-Gram vs CBOW
- Final Notes

What’s inside:

Word2Vec

- Idea
- Objective Function
- Training Procedure
- Faster Training: Negative Sampling
- Word2Vec versions: Skip-Gram vs CBOW
- Final Notes

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)

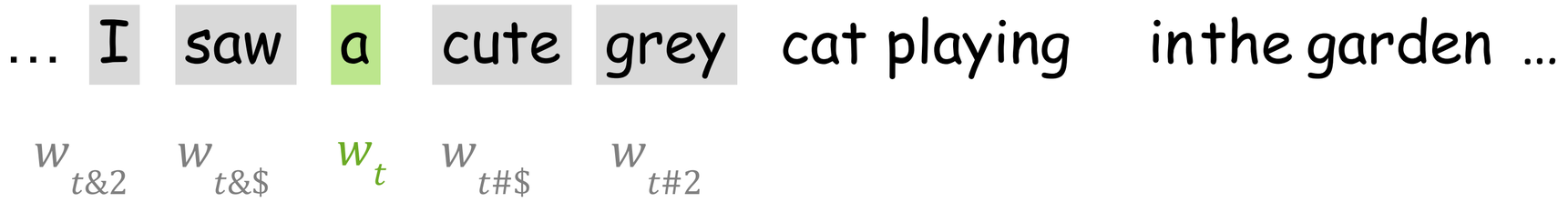
Word2Vec: High-Level pipeline

- take a huge text corpus

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

Word2Vec: High-Level pipeline

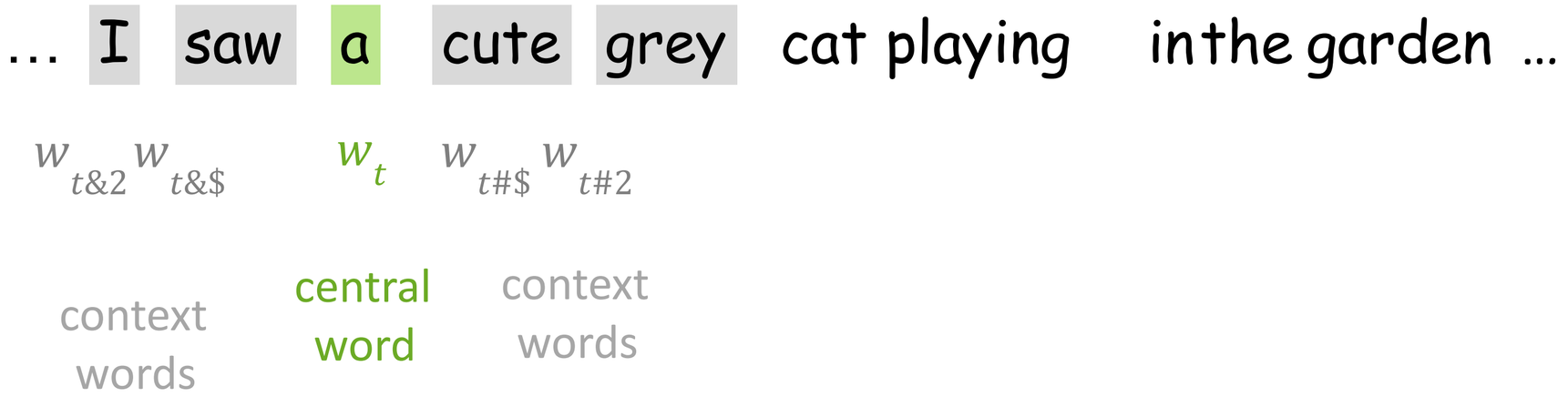
- take a huge text corpus
- go over the text with a sliding window, moving one word at a time.



Lena: Visualization idea is from the Stanford CS224n course.

Word2Vec: High-Level pipeline

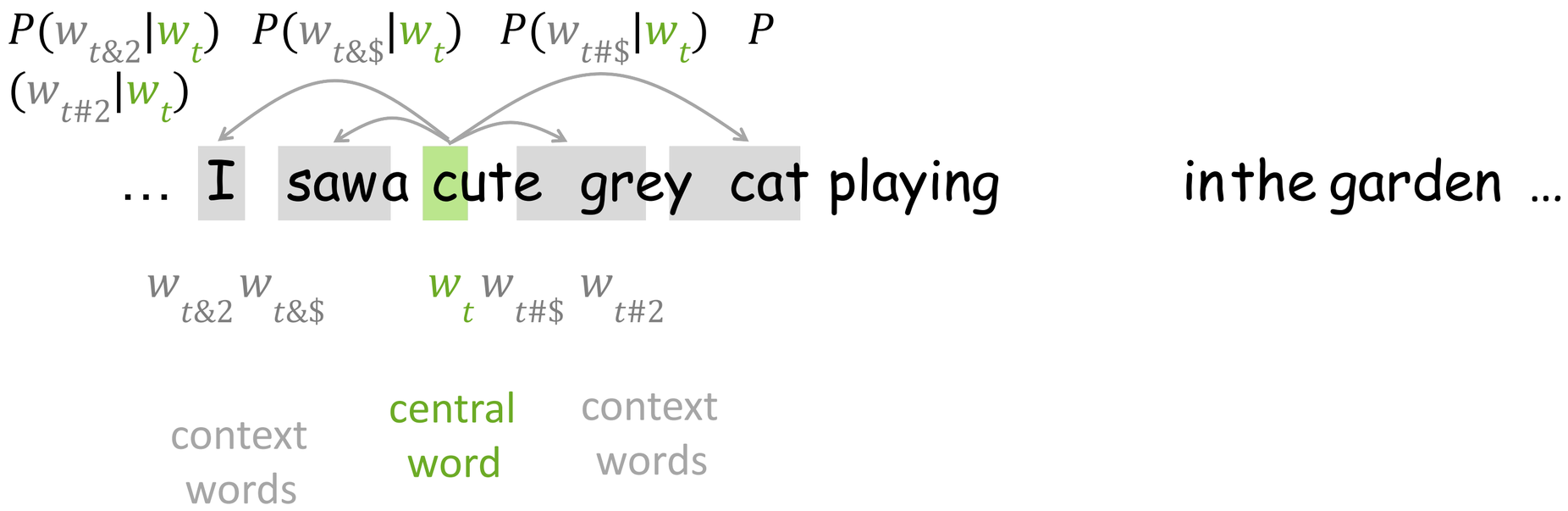
- take a huge text corpus
- go over the text with a sliding window, moving one word at a time.



Lena: Visualization idea is from the Stanford CS224n course.

Word2Vec: High-Level pipeline

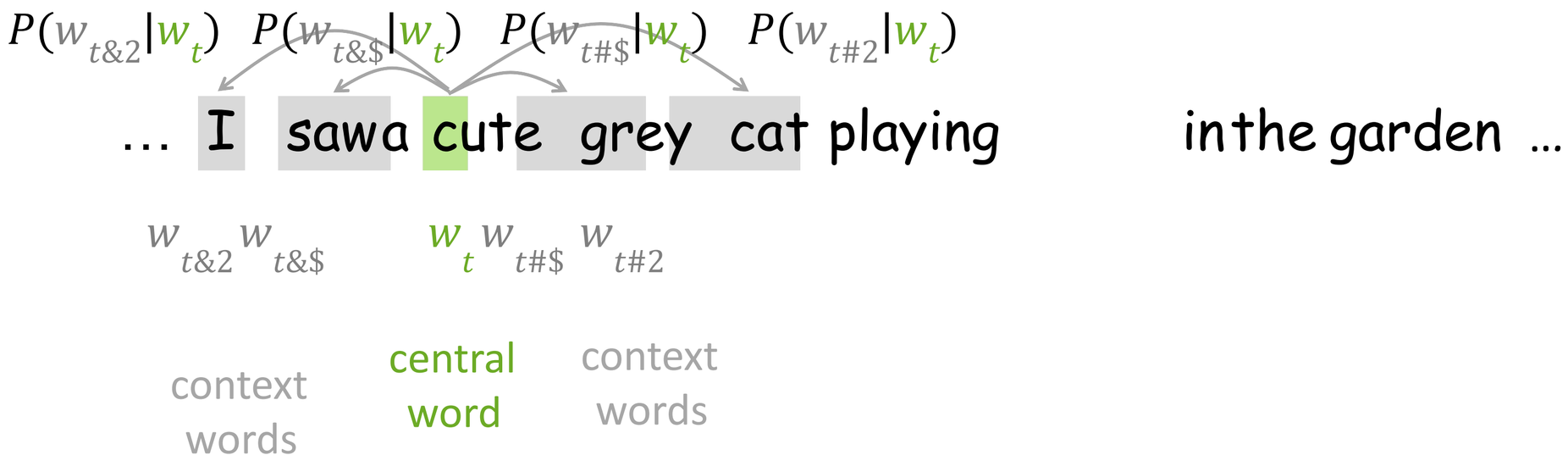
- take a huge text corpus
- go over the text with a sliding window, moving one word at a time.
- for the central word, compute probabilities of context words;



Lena: Visualization idea is from the Stanford CS224n course.

Word2Vec: High-Level pipeline

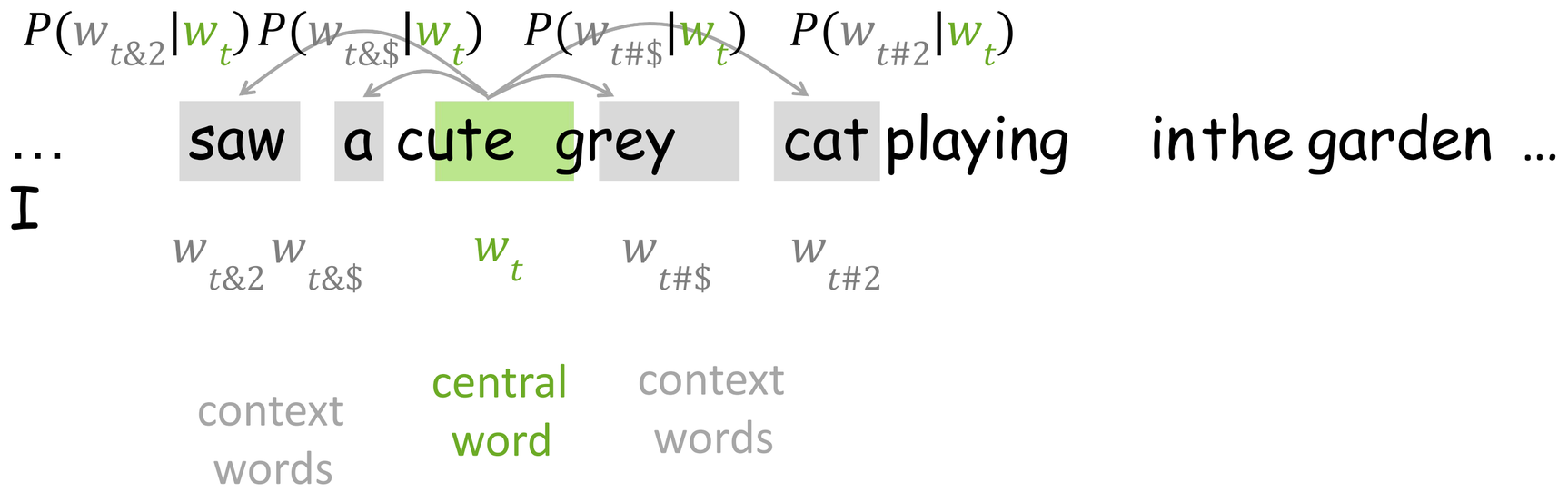
- take a huge text corpus
- go over the text with a sliding window, moving one word at a time.
- for the central word, compute probabilities of context words;
- adjust the vectors to increase these probabilities.



Lena: Visualization idea is from the Stanford CS224n course.

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.



Lena: Visualization idea is from the Stanford CS224n course.

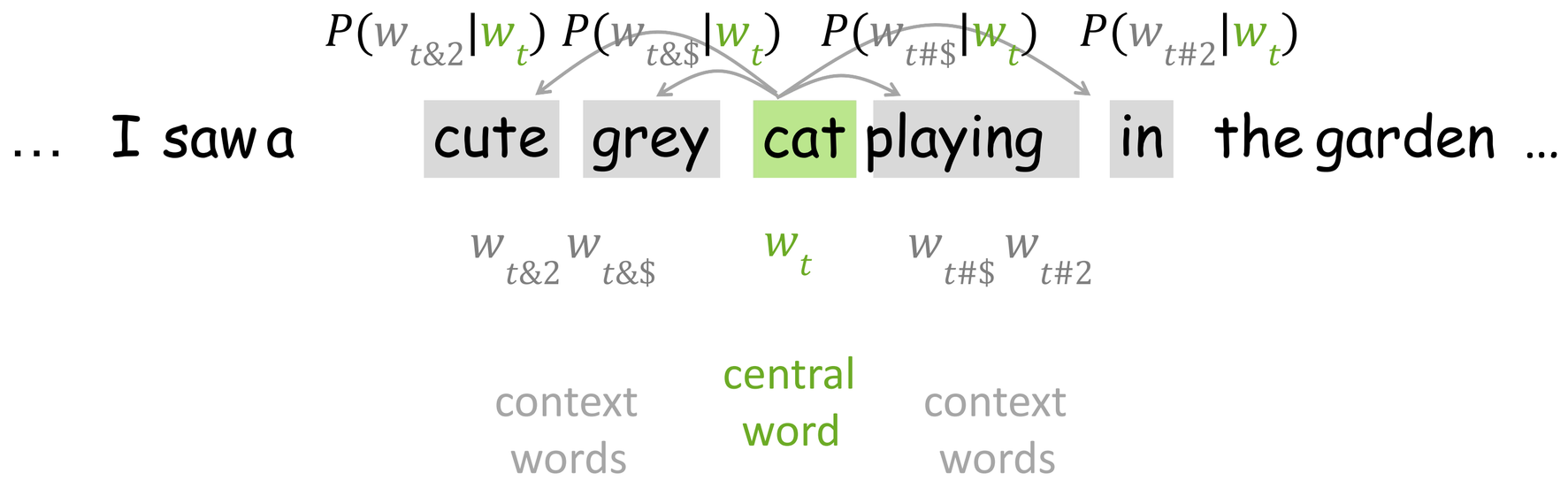
Word2Vec: High-Level pipeline

- 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.

Lena: Visualization idea is from the Stanford CS224n course.

Word2Vec: High-Level pipeline

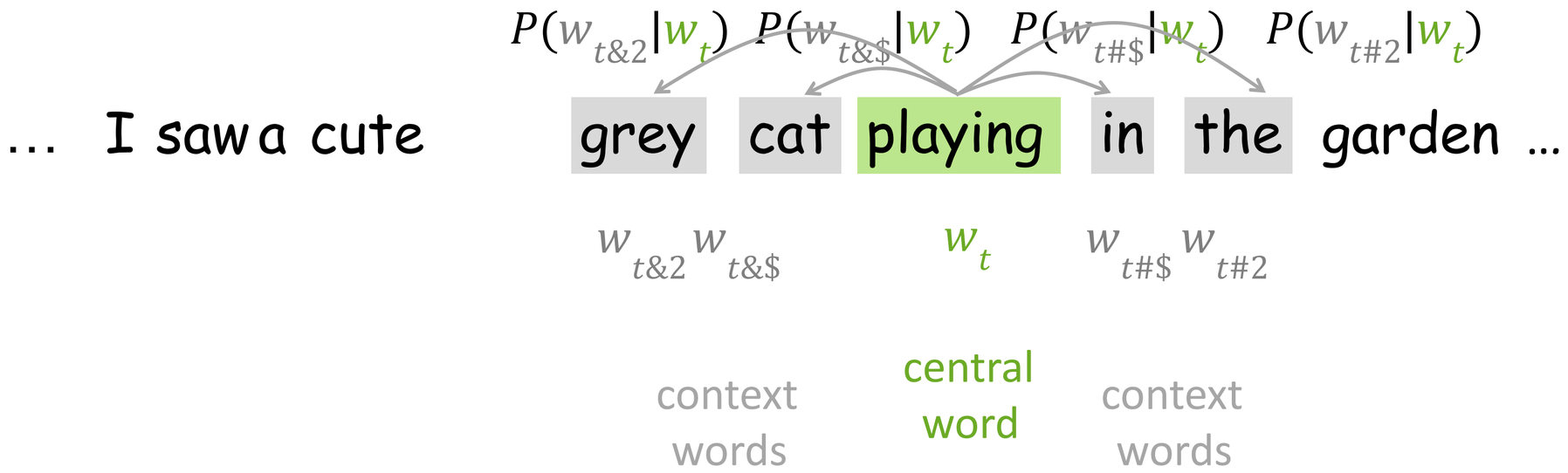
- take a huge text corpus
- go over the text with a sliding window, moving one word at a time.
- for the central word, compute probabilities of context words;
- adjust the vectors to increase these probabilities.



Lena: Visualization idea is from the Stanford CS224n course.

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.



Lena: Visualization idea is from the Stanford CS224n course.

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.

Lena: Visualization idea is from the Stanford CS224n course.

What’s inside:

Word2Vec

- Idea
- Objective Function
- Training Procedure
- Faster Training: Negative Sampling
- Word2Vec versions: Skip-Gram vs CBOW
- Final Notes

What’s inside:

Word2Vec

- Idea
- Objective Function
- Training Procedure
- Faster Training: Negative Sampling
- Word2Vec versions: Skip-Gram vs CBOW
- Final Notes

Objective Function: Negative Log-Likelihood

Word2Vec tries to find the parameters that maximize the data likelihood:

Likelihood = $L\ "(\ \ ) = \mathsf{G}\ \mathsf{G}^{T}_{t\&1}\ \mathsf{G}^{)m+\mathrm{j}+m,}_{\mathrm{j}./} P(w_{t3\mathrm{j}}|w_t, ")$

Objective Function: Negative Log-Likelihood

Word2Vec tries to find the parameters that maximize the data likelihood:

Likelihood = $L\ "(\ \ ) = \mathsf{G}\ \mathsf{G}^{T}_{t\&1}\ \mathsf{G}_{)m+\mathrm{j}+m,\ \mathrm{j}./} P(w_{t3\mathrm{j}}|w_t, ")$

We want our model to think that the training datais “likely”

Objective Function: Negative Log-Likelihood

Word2Vec tries to find the parameters that maximize the data likelihood:

Likelihood = $L\ "(\ ) = \mathrm{G}\ \mathrm{G}^{T}_{t\&1}\ P(w_{t3\mathrm{j}}|w_t, ")^{)m+\mathrm{j}+m,}_{\mathrm{j./}}$

" are all variables to optimize

We want our model to think that the training datais “likely”

Objective Function: Negative Log-Likelihood

Word2Vec tries to find the parameters that maximize the data likelihood:

Likelihood = $L\ "(\ \ ) = \mathsf{G}\ \mathsf{G}^{T}_{t\&1\ )m+\mathrm{j}+m,\ \mathrm{j}./} P(w_{t3\mathrm{j}}|w_t, ")$

We want our model to think that the training datais “likely”

To do this, it uses negative (log-)likelihood as its loss function:

Loss = $J\ (" \ ) = -\frac{1}{T}\log L(") = -\frac{1}{T}\overline{=}^{T}_{t\&1\ )m+\mathrm{j}+m,\ \mathrm{j}./} = \log P(w_{t3\mathrm{j}t}|w\ ,\ ")$

Objective Function: Negative Log-Likelihood

Word2Vec tries to find the parameters that maximize the data likelihood:

Likelihood = $L\ "(\ \ ) = \mathsf{G}\ \mathsf{G}^{T}_{t\&1\ )m+\mathrm{j}+m,\ \mathrm{j}./} P(w_{t3\mathrm{j}}|w_t, ")$

We want our model to think that the training datais “likely”

To do this, it uses negative (log-)likelihood as its loss function:

Loss = $J\ (" \ ) = -\frac{1}{T}\log L(") = -\frac{1}{T}\overline{=}^{T}_{t\&1\ )m+\mathrm{j}+m,\ \mathrm{j}./} = \log P(w_{t3\mathrm{j}t}|w\ ,\ ")$

agrees with our plan above ⟼

Objective Function: Negative Log-Likelihood

Word2Vec tries to find the parameters that maximize the data likelihood:

Likelihood = $L\ "(\ \ ) = \mathsf{G}\ \mathsf{G}^{T}_{t\&1\ )m+\mathrm{j}+m,\ \mathrm{j}./} P(w_{t3\mathrm{j}}|w_t, ")$

We want our model to think that the training datais “likely”

To do this, it uses negative (log-)likelihood as its loss function:

Loss = $J\ (" \ ) = -\frac{1}{T}\frac{1}{}\log L(") = -\frac{}{T}\overline{=}\ ^{T}_{t\&1\ )m+\mathrm{j}+m,\ \mathrm{j}./} = \log P(w_{t3\mathrm{j}t}|w\ ,\ ")$

agrees with our plan above

⟼ go over text

Objective Function: Negative Log-Likelihood

Word2Vec tries to find the parameters that maximize the data likelihood:

Likelihood = $L\ "(\ \ ) = \mathsf{G}\ \mathsf{G}^{T} \quad P(w_{t3\mathrm{j}}|w_t, ")$

$t\&1\ )m+\mathrm{j}+m,$
$\mathrm{j}./$

We want our model to think that the training datais “likely”

To do this, it uses negative (log-)likelihood as its loss function:

Loss = $J\ (\text{"}\ ) = -\frac{1}{T}\log L(") = -\frac{1}{T}\overline{=}\ = \log P(w^{T}\quad {}_{t3\mathrm{j}t}|w\ ,$
$")$

$t\&1\ )m+\mathrm{j}+m,$
$\mathrm{j}./$

agrees with our plan above ⟼ go over text with a sliding window

Objective Function: Negative Log-Likelihood

Word2Vec tries to find the parameters that maximize the data likelihood:

Likelihood = $L\ "(\ \ ) = \mathsf{G}\ \mathsf{G}^{T}_{t\&1}\ \mathsf{G}_{)m+\mathrm{j}+m,\ \mathrm{j}./} P(w_{t3\mathrm{j}}|w_t, ")$

We want our model to think that the training datais “likely”

To do this, it uses negative (log-)likelihood as its loss function:

Loss = $J\ (" \ ) = -\frac{1}{T}\log L(") = -\frac{1}{T}\sum_{t\&1}^{T}\ \sum_{)m+\mathrm{j}+m,\ \mathrm{j}./} \log P(w_{t3\mathrm{j}t}|w\ \ , ")$

agrees with our plan above

↦ go over text

with a sliding window

compute probability of the context word given the central

Objective Function: Negative Log-Likelihood

Word2Vec tries to find the parameters that maximize the data likelihood:

Likelihood = $L\ "(\ \ ) = \mathsf{G}\ \mathsf{G}^{T}_{t\&1}\ \mathsf{G}^{)m+j+m,}_{j./} P(w_{t3j}|w_t, ")$

We want our model to think that the training datais “likely”

To do this, it uses negative (log-)likelihood as its loss function:

Loss = $J\ (" \ ) = -\frac{1}{T}\log L(") = -\frac{1}{T}\sum_{t\&1}^{T}\sum_{j./}^{)m+j+m,} \log P(w_{t3j}|w_t, ")$

$P(w_{t3j}|w_t, ")$

How to compute this?

agrees with our plan above

↦ go over text

with a sliding window

compute probability of the context word given the central

How to compute $P(w_{t\%\&} | w_t,))$?

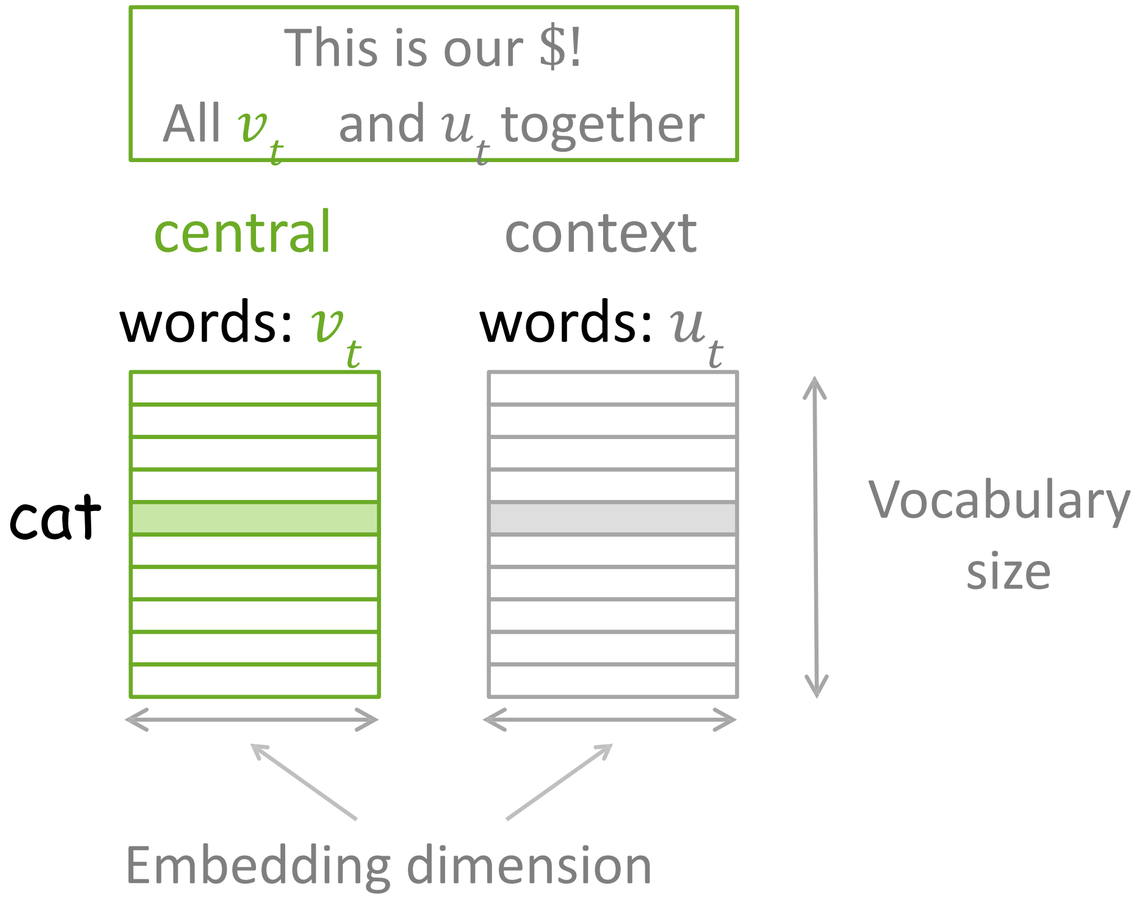
For each word w, we will have two vectors:

- v_w when it is a central word
- u_w when it is a context word

How to compute $P(w_{t)^*} | w_t, \$)$?

For each word w, we will have two vectors:

- v_w when it is a central word
- u_w when it is a context word

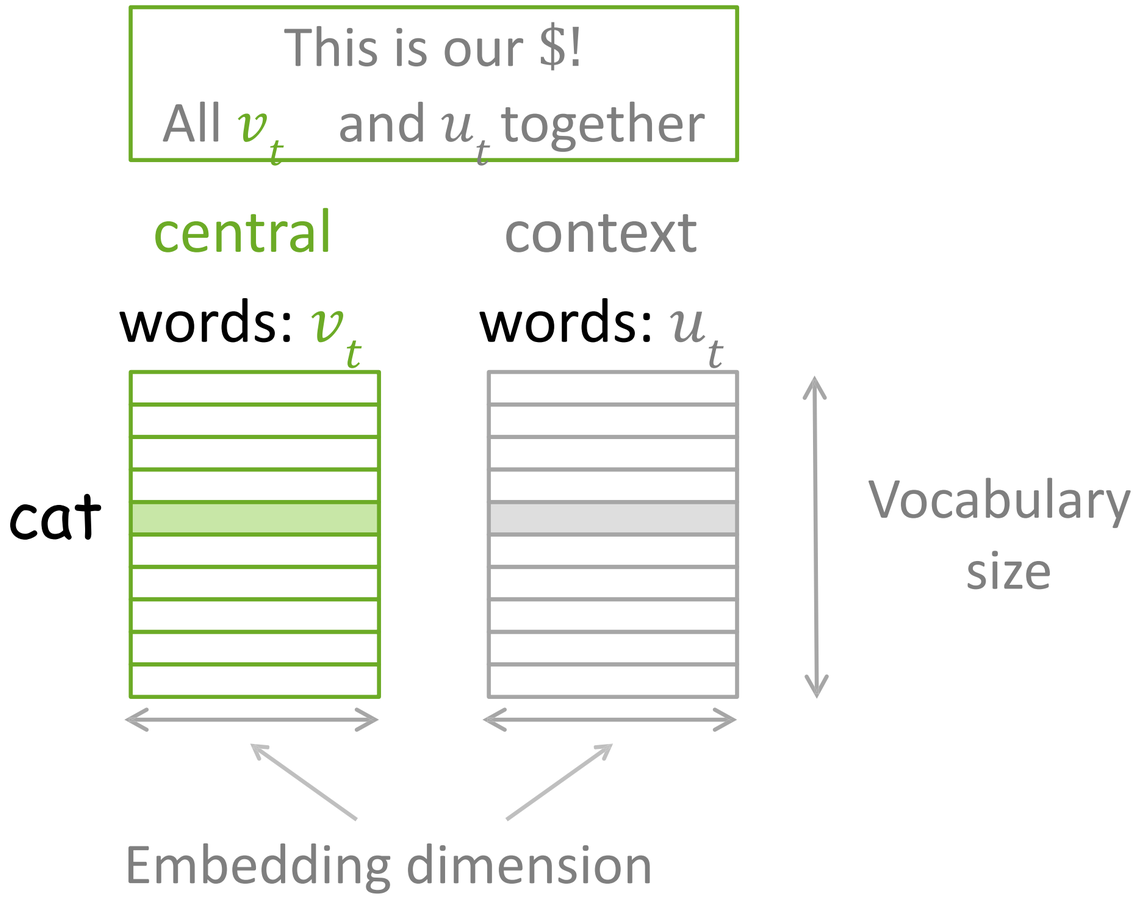


How to compute $P(w_{t)^*} | w_t, \$)$?

For each word w, we will have two vectors:

- v_w when it is a central word
- u_w when it is a context word

Once the vectors are trained, usually we throw away context vectors and use only word vectors.



How to compute $P(w_{t56} | w_t, 9)$?

For the central word c and context word o (o - outside):

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{0 \in V} \exp(u_0^T v_c)}$$

How to compute $P(w_{t56} | w_t, 9)$?

For the central word c and context word o (o - outside):

Dot product: measures similarity of o and c
Larger dot product = larger probability

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{0 \in V} \exp(u_0^T v_c)}$$

How to compute $P(w_{t56} | w_t, 9)$?

For the central word c and context word o (o - outside):

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{0 \in V} \exp(u_0^T v_c)}$$

Dot product: measures similarity of o and c
Larger dot product = larger probability

Normalize over entire vocabulary
to get probability distribution

How to compute $P(w_{t56} | w_t, 9)$?

For the central word c and context word o (o - outside):

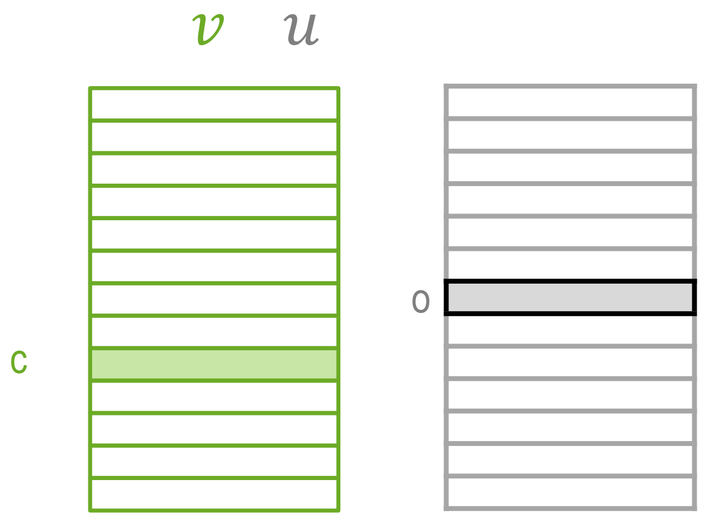
$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{0 \in V} \exp(u_0^T v_c)}$$

Dot product: measures similarity of o and c
Larger dot product = larger probability

Normalize over entire vocabulary
to get probability distribution

Let us recall our plan:

- …
- adjust the vectors to increase these probabilities.



How to compute $P(w_{t56} | w_t, 9)$?

For the central word c and context word o (o - outside):

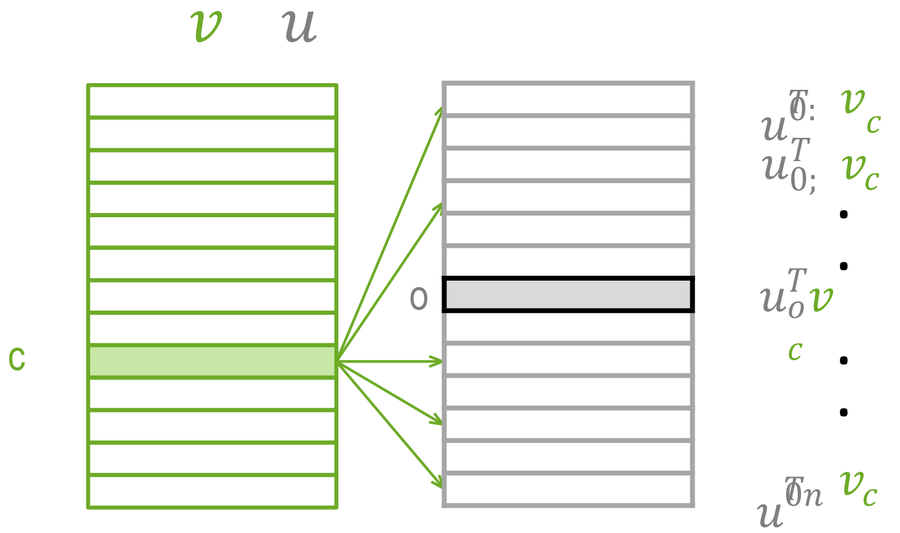
$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{0 \in V} \exp(u_0^T v_c)}$$

Dot product: measures similarity of o and c
Larger dot product = larger probability

Normalize over entire vocabulary
to get probability distribution

Let us recall our plan:

- …
- adjust the vectors to increase these probabilities.



How to compute $P(w_{t56} | w_t, 9)$?

For the central word c and context word o (o - outside):

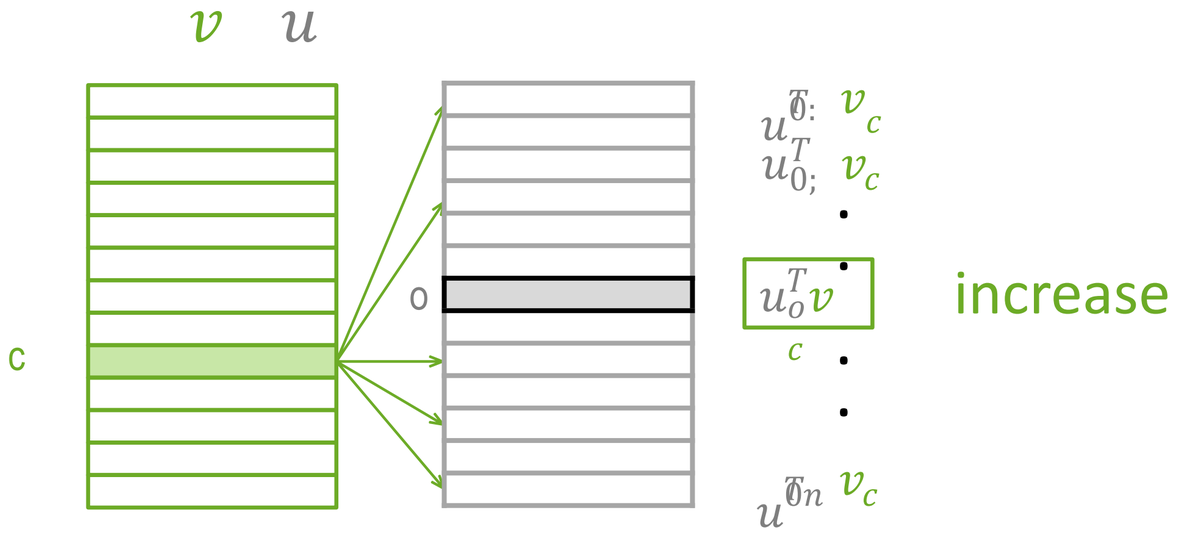
$$P(o|c) = \frac{\exp(u^T v)}{\sum_{0 \in V} \exp(u_0^T v_c)}$$

Dot product: measures similarity of o and c
Larger dot product = larger probability

Normalize over entire vocabulary
to get probability distribution

Let us recall our plan:

- …
- adjust the vectors to increase these probabilities.



How to compute $P(w_{t56} | w_t, 9)$?

For the central word c and context word o (o - outside):

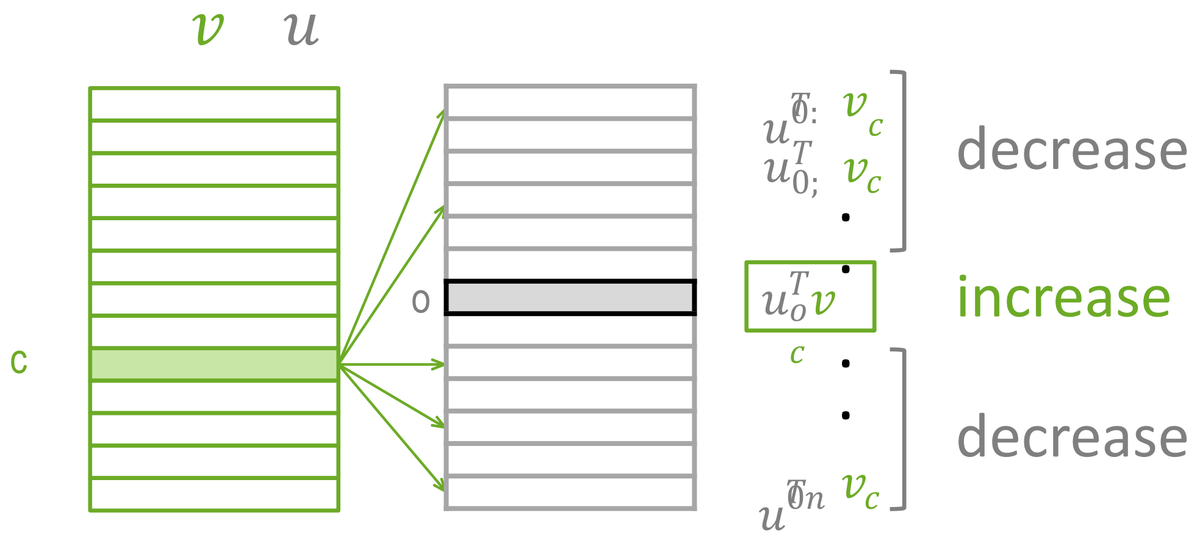
$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{0 \in V} \exp(u_0^T v_c)}$$

Dot product: measures similarity of o and c
Larger dot product = larger probability

Normalize over entire vocabulary
to get probability distribution

Let us recall our plan:

- …
- adjust the vectors to increase these probabilities.



Note: We used the softmax function

$$P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{0 \in V} \exp(u_0^T v_c)}$$ - the probability we used

Note: We used the softmax function

$$P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{0 \in V} \exp(u_0^T v_c)}$$ - the probability we used

---

Softmax function $\mathbb{R}^n \rightarrow \mathbb{R}^n$:

$$softmax(x)_i = \frac{\exp(x_i)}{\sum_{j \in A} \exp(x_j)} = p_i$$

Note: We used the softmax function

$$P(o \mid c) = \frac{\exp(u^T v)}{\sum_{0 \in V} \exp(u_0^T v_c)}$$ - the probability we used

---

Softmax function $\mathbb{R}^n \rightarrow \mathbb{R}^n$:

$$softmax(x)_> = \frac{\exp(x_>)}{\sum_{j \in A} \exp(x_j)} = p_>$$

- maps arbitrary values $x_>$ to a probability distribution $p_>$

Note: We used the softmax function

$$P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{0 \in V} \exp(u_0^T v_c)}$$ - the probability we used

---

Softmax function $\mathbb{R}^n \rightarrow \mathbb{R}^n$:

$$softmax(x_i) = \frac{\exp(x_i)}{\sum_{j \in A} \exp(x_j)} = p_i$$

- maps arbitrary values x_i to a probability distribution p_i
- ”max” because amplifies probability of largest x_i

Note: We used the softmax function

$$P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{0 \in V} \exp(u_0^T v_c)}$$ - the probability we used

---

Softmax function $\mathbb{R}^n \rightarrow \mathbb{R}^n$:

$$softmax(x_i) = \frac{\exp(x_i)}{\sum_{j \in A} \exp(x_j)} = p_i$$

- maps arbitrary values x_i to a probability distribution p_i
- ”max” because amplifies probability of largest x_i
- “soft” because still assigns some probability to smaller x_i

Note: We used the softmax function

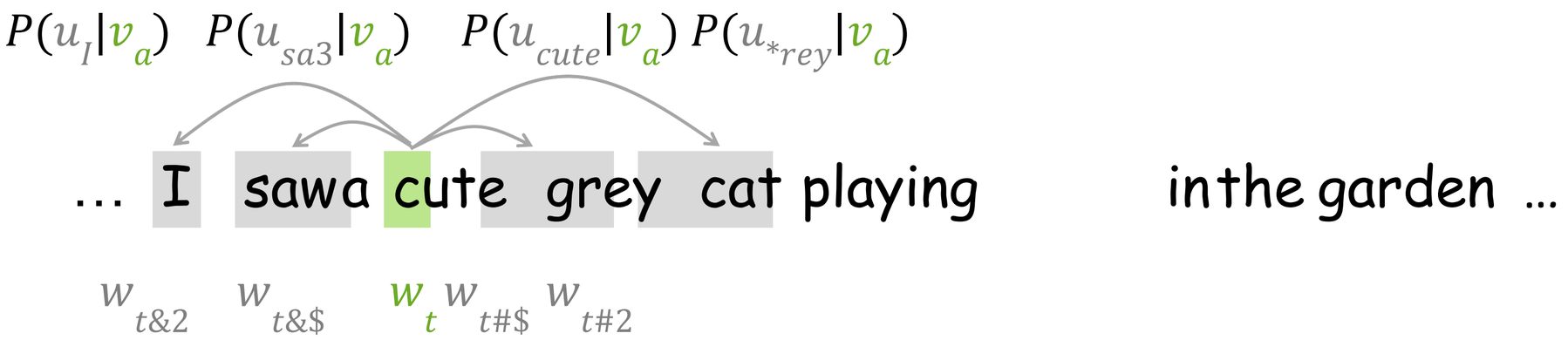
$$P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{0 \in V} \exp(u_0^T v_c)}$$ - the probability we used

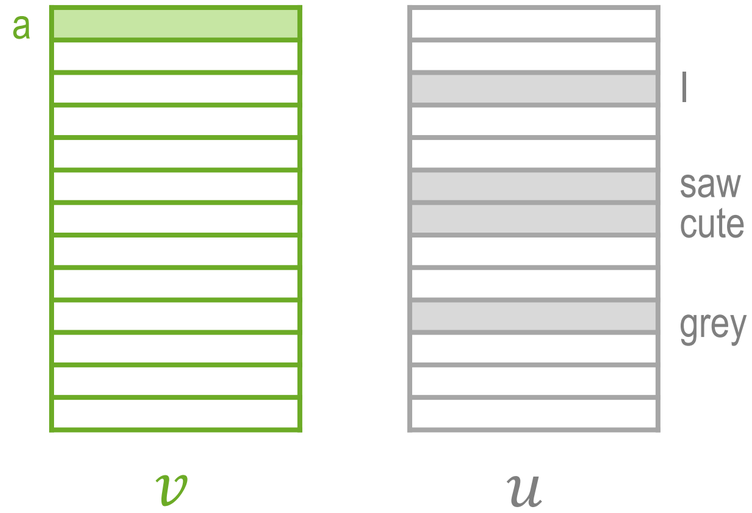
Softmax function $\mathbb{R}^n \rightarrow \mathbb{R}^n$:

$$softmax(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

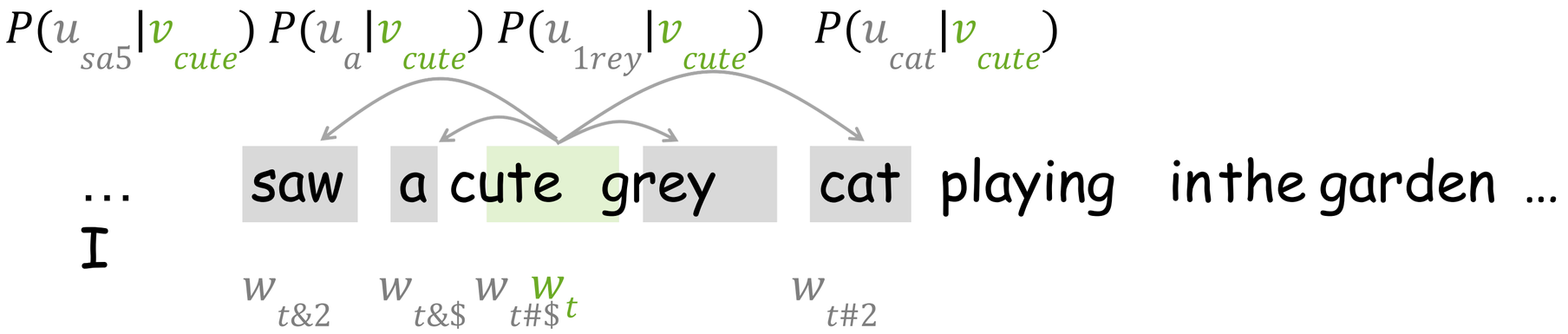
- maps arbitrary values x_i to a probability distribution p_i
- ”max” because amplifies probability of largest x_i
- “soft” because still assigns some probability to smaller x_i
- often used in Deep Learning!

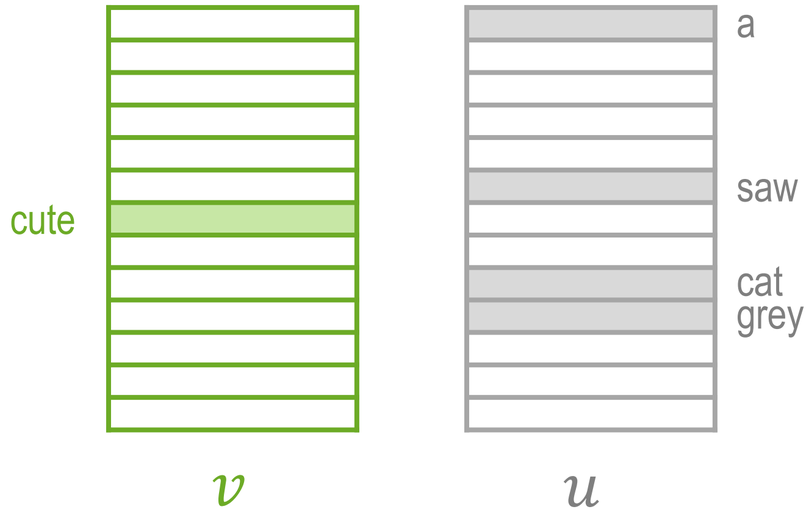
Two vectors for each word



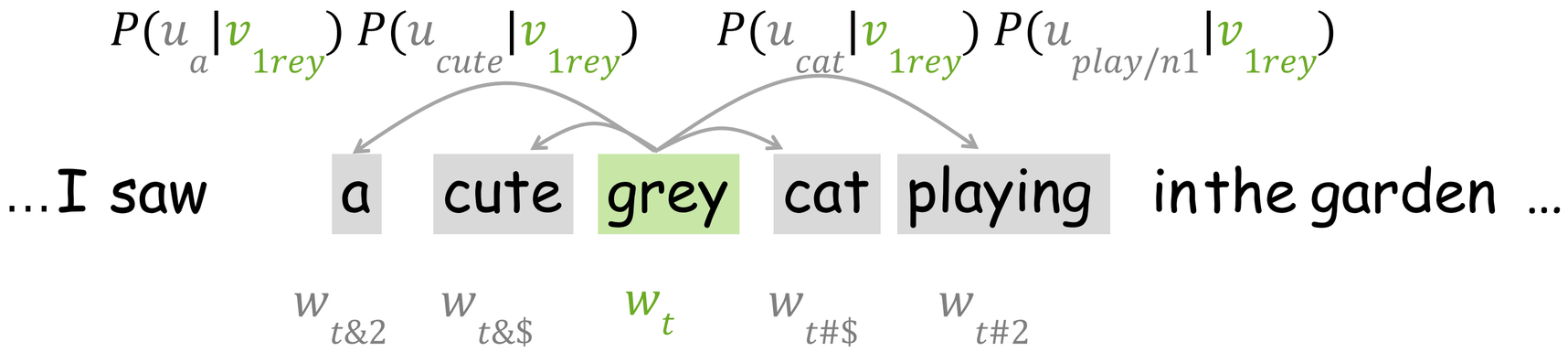


Two vectors for each word

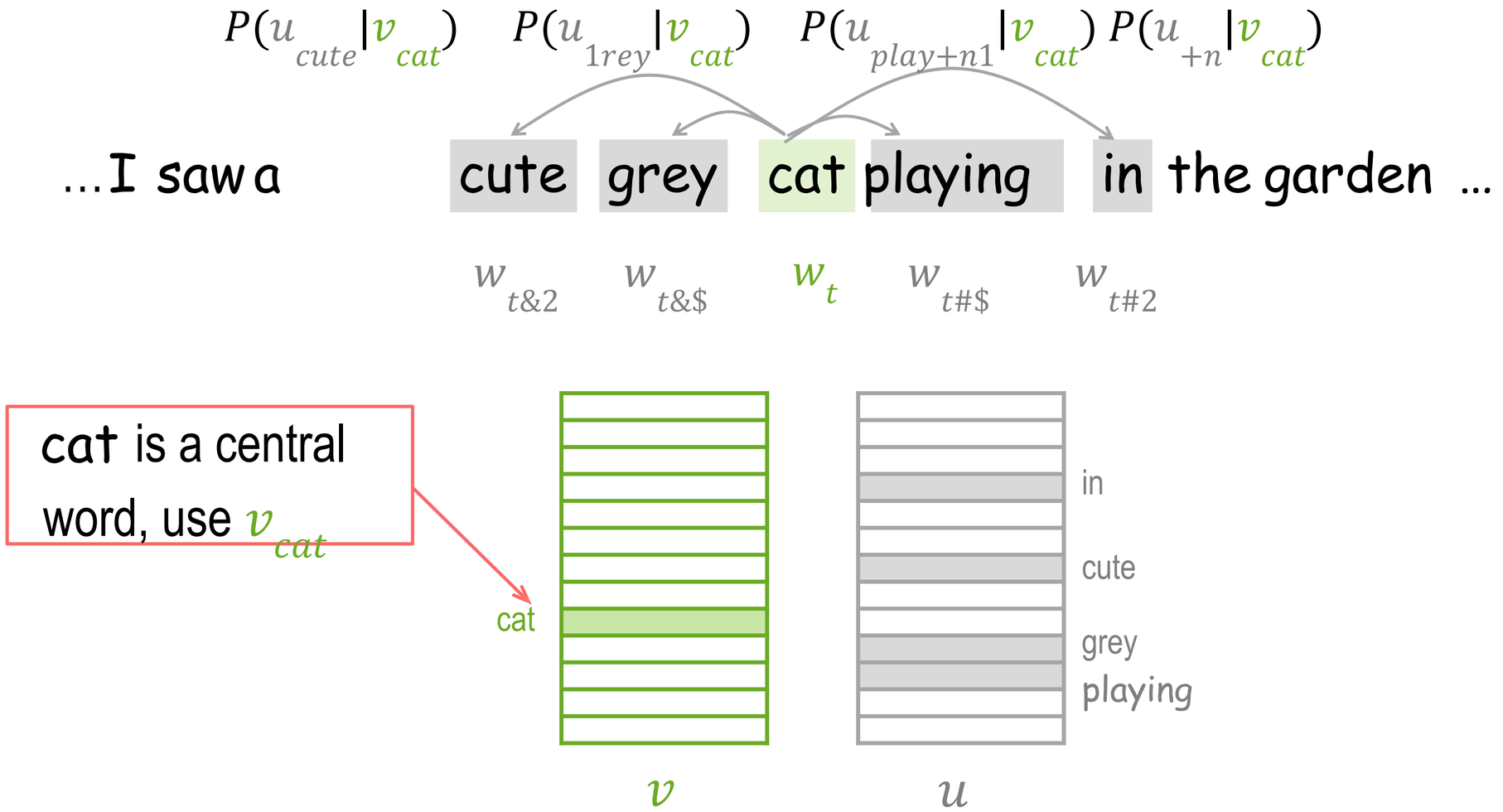




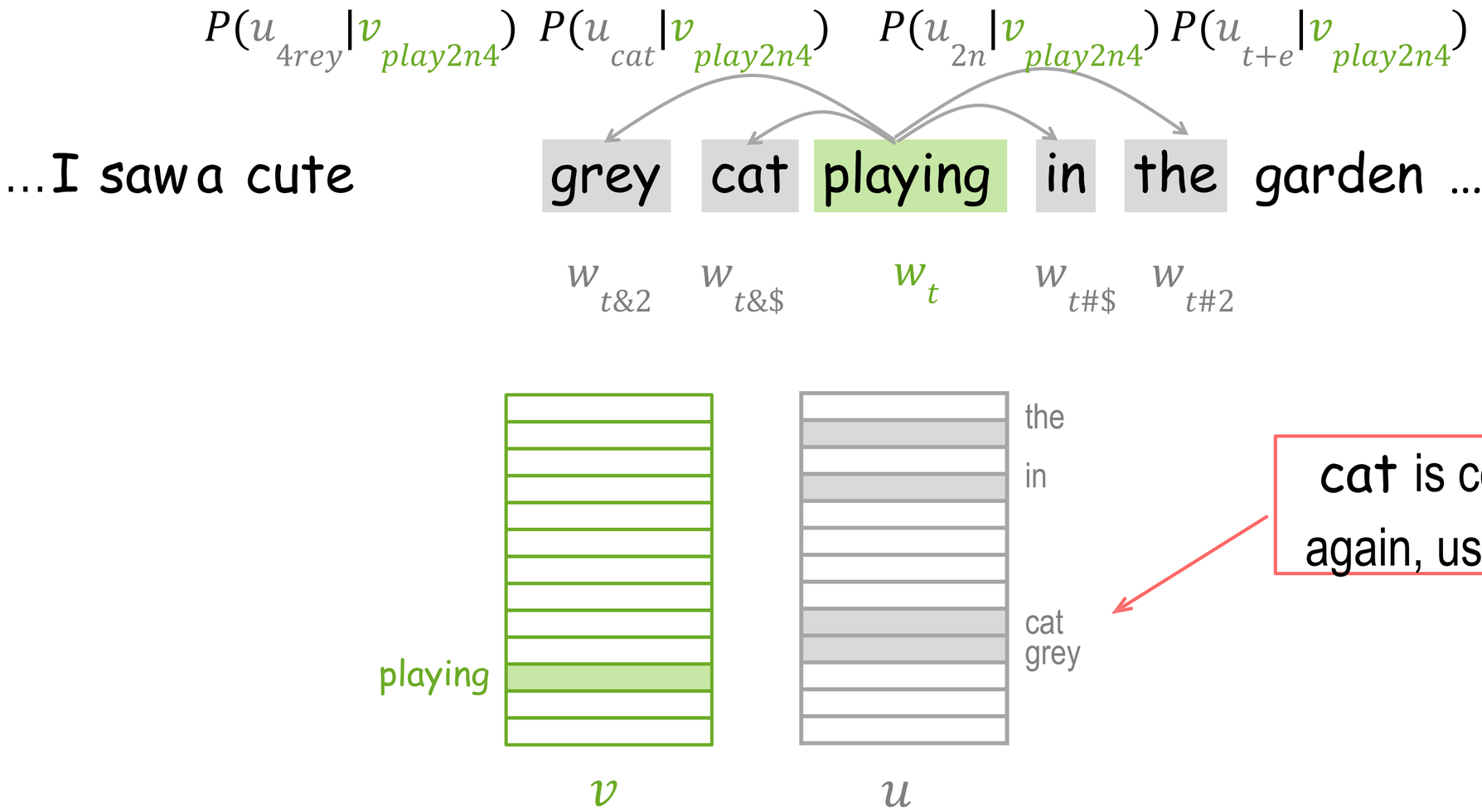
Two vectors for each word



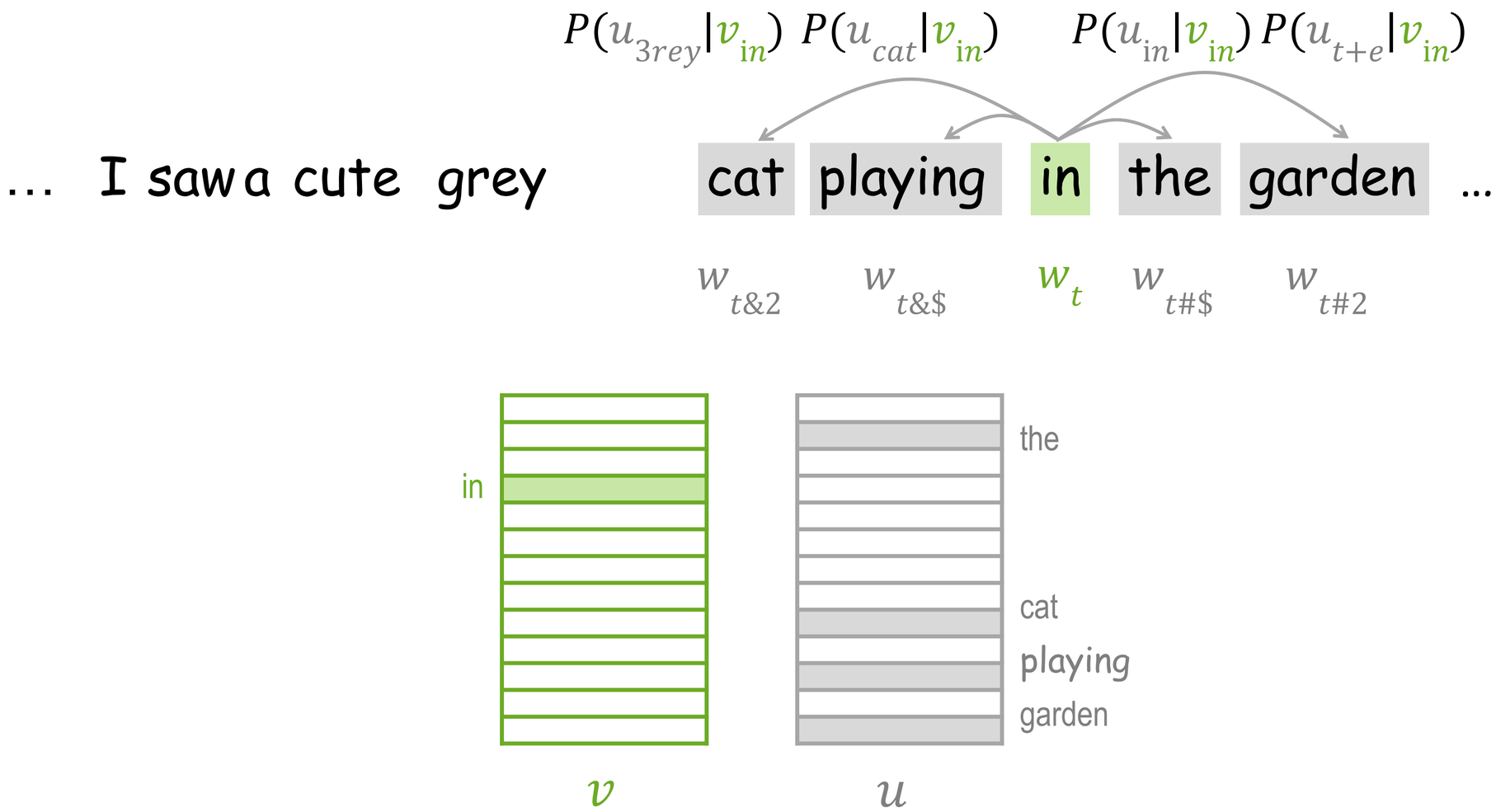
Two vectors for each word



Two vectors for each word



Two vectors for each word



What’s inside:

Word2Vec

- Idea
- Objective Function
- Training Procedure
- Faster Training: Negative Sampling
- Word2Vec versions: Skip-Gram vs CBOW
- Final Notes

What’s inside:

Word2Vec

- Idea
- Objective Function
- Training Procedure
- Faster Training: Negative Sampling
- Word2Vec versions: Skip-Gram vs CBOW
- Final Notes

One training step in detail

Loss = $J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\log P(w_t \mid w_{t-1}, \theta)$

One training step in detail

Loss for word $<$ in window t

Loss = $J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m \le j \le m, j \neq 0} \log P(w_{t+j} \mid w_t, \theta)$

$P(w_{t+j} \mid w_t, \theta) = -\frac{1}{T}\sum_{t=1}^{T} J_{t,j}(\theta)$

One training step in detail

Loss for word = in window t

Loss = $J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m \le j \le m, j \ne 0} \log P(w_{t+j} \mid w_t, \theta) = -\frac{1}{T}\sum_{t=1}^{T} J_{t,j}(\theta)$

pick one window

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

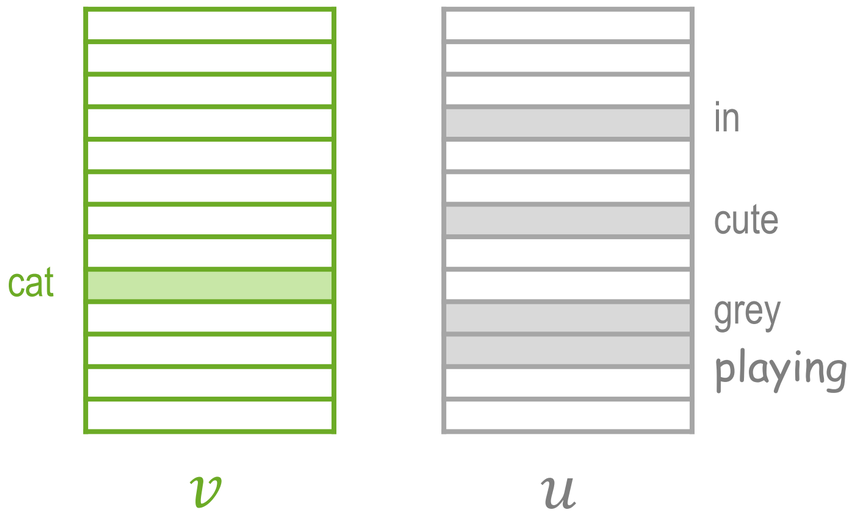
w_{t-2} w_{t-1} w_t w_{t+1} w_{t+2}

One training step in detail

$$P(u_{cute}|v_{cat})\,P(u_{1rey}|v_{cat})\,P(u_{play+n1}|v_{cat})\,P(u_{+n}|v_{cat})$$

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

$w_{t\&2}$ $w_{t\&\$}$ w_t $w_{t\#\$}$ $w_{t\#2}$

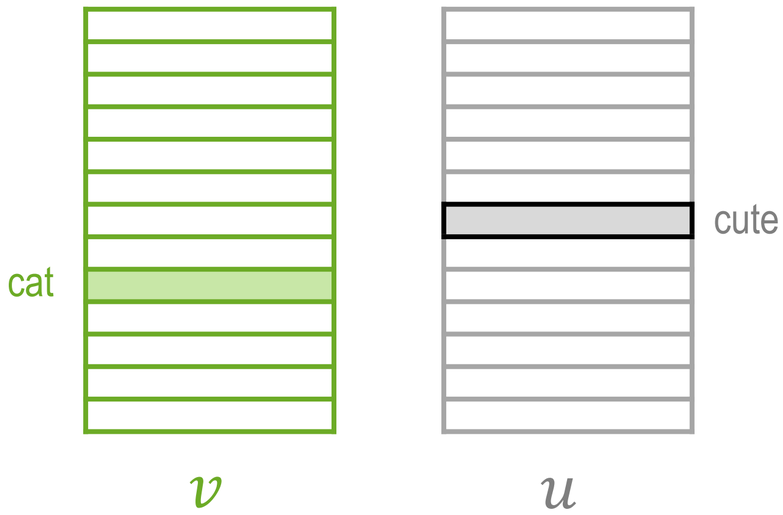


One training step in detail

$P(u_{cute}|v_{cat})P(u_{1rey}|v_{cat})P(u_{play+n1}|v_{cat})P(u_{+n}|v_{cat})$

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

$w_{t\&2}$ $w_{t\&\$}$ w_{t} $w_{t\#\$}$ $w_{t\#2}$



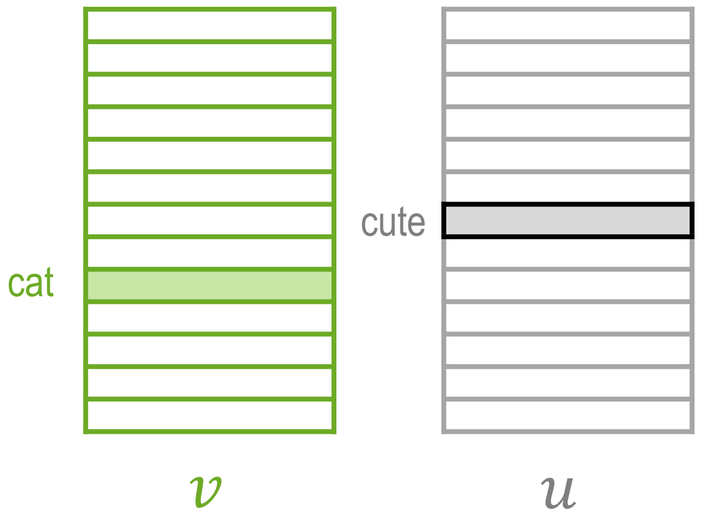
Pick one of the context words

One training step in detail

$P(u_{cut3}|v_{cat})P(u_{1r3y}|v_{cat})$ $P(u_{play*n1}|v_{cat})$ $P(u_{*n}|v_{cat})$

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

$w_{t\&2}$ $w_{t\&\$}$ w_t $w_{t\#\$}$ $w_{t\#2}$

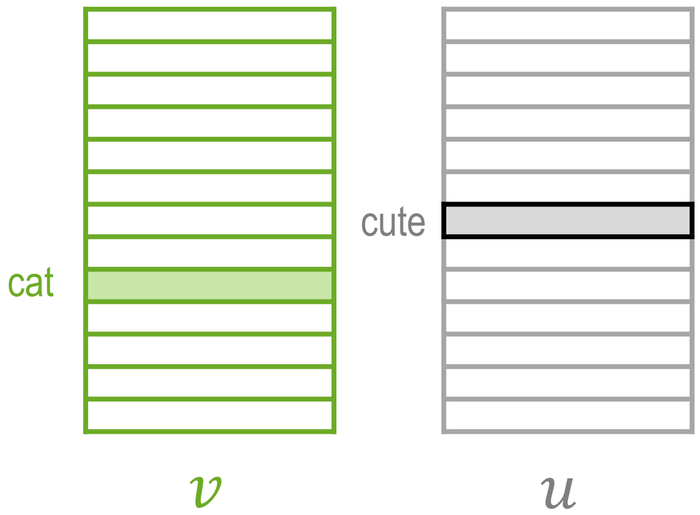


Look at the loss component for this step:

$$-\log P(cut|cat) = -\log \frac{\exp(u_{cut3}^T v_{cat})}{\sum_{F \in V} \exp(u_F^T v_{cat})}$$

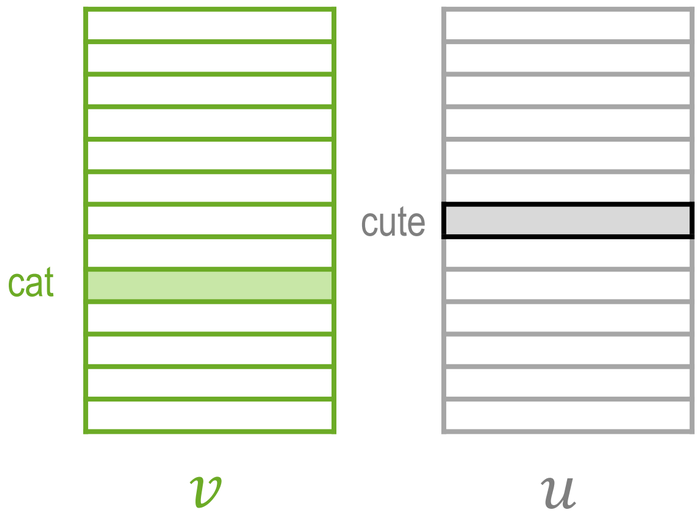
One training step in detail

$$-\log P(cute \mid cat) = -\log \frac{\exp(u_{cute}^T v_{cat})}{\sum_{7 \in V} \exp(u_7^T v_{cat})}$$



One training step in detail

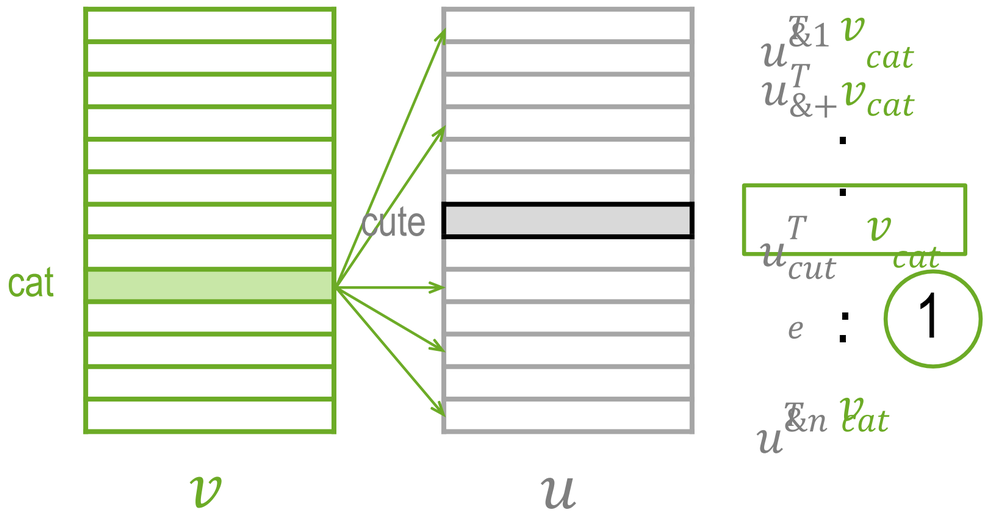
$$-\log P(cute \mid cat) = -\log \frac{\exp(u_{cute}^T v_{cat})}{\sum_{7 \in V} \exp(u_7^T v_{cat})} = -u_{cute}^T v_{cat} + \log \sum_{7 \in V} \exp(u_7^T v_{cat})$$



One training step in detail

$$-\log P(cute \mid cat) = -\log \frac{\exp(u_{cute}^T v_{cat})}{\sum_{\&\in V} \exp(u_{\&}^T v_{cat})} = -u_{cute}^T v_{cat} + \log \sum_{\&\in V} \exp(u_{\&}^T v_{cat})$$

1. Take dot product of v_{cat} with all u

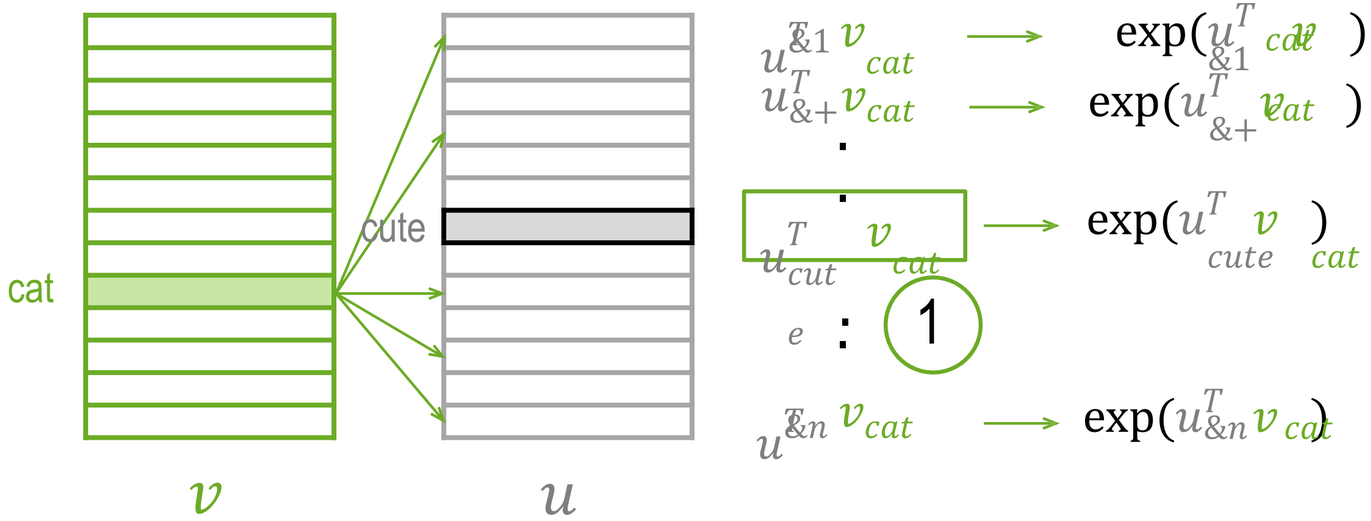


One training step in detail

$$-\log P(cute \mid cat) = -\log \frac{\exp(u_{cute}^T v_{cat})}{\sum_{\&\in V} \exp(u_{\&}^T v_{cat})} = -u_{cute}^T v_{cat} + \log \sum_{\&\in V} \exp(u_{\&}^T v_{cat})$$

1. Take dot product of v_{cat} with all u

2. exp



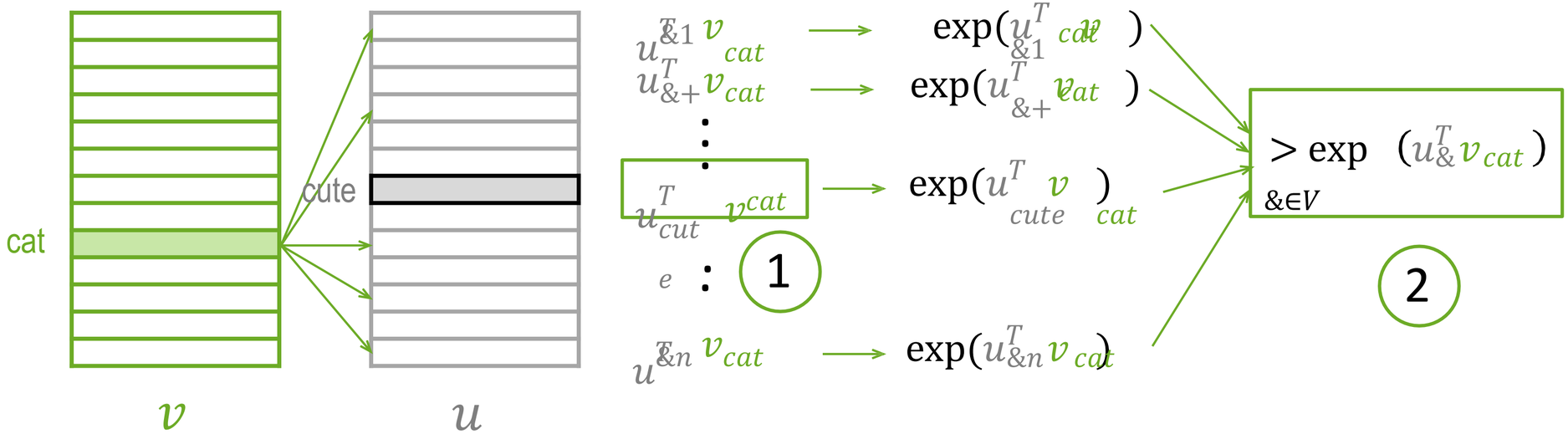
One training step in detail

$$-\log P(cute \mid cat) = -\log \frac{\exp(u_{cute}^T v_{cat})}{\sum_{\&\in V} \exp(u_{\&}^T v_{cat})} = -u_{cute}^T v_{cat} + \log \sum_{\&\in V} \exp(u_{\&}^T v_{cat})$$

1. Take dot product of v_{cat} with all u

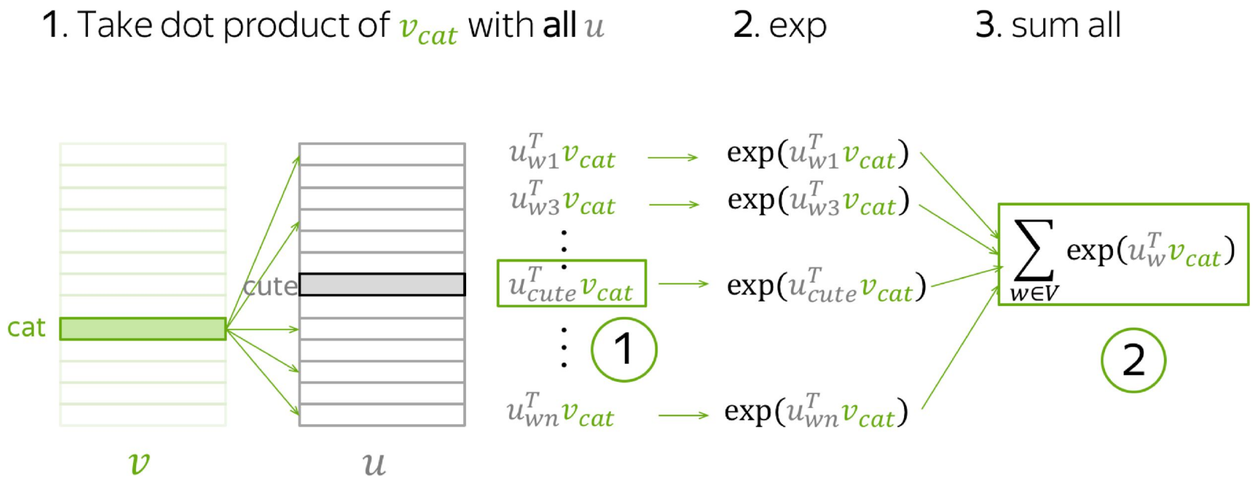
2. exp

3. sum all



One training step in detail

$$-\log P(cute \mid cat) = -u^T_{cute} v_{cat} + \log \sum_{4 \in V} \exp(u^T_4 v_{cat})$$



4. get loss (for this one step)

$$J_{t,\$}(\ ) = -u^T_{cute} v_{cat} + \log \sum_{4 \in V} \exp(u^T_4 v_{cat})$$

1 2

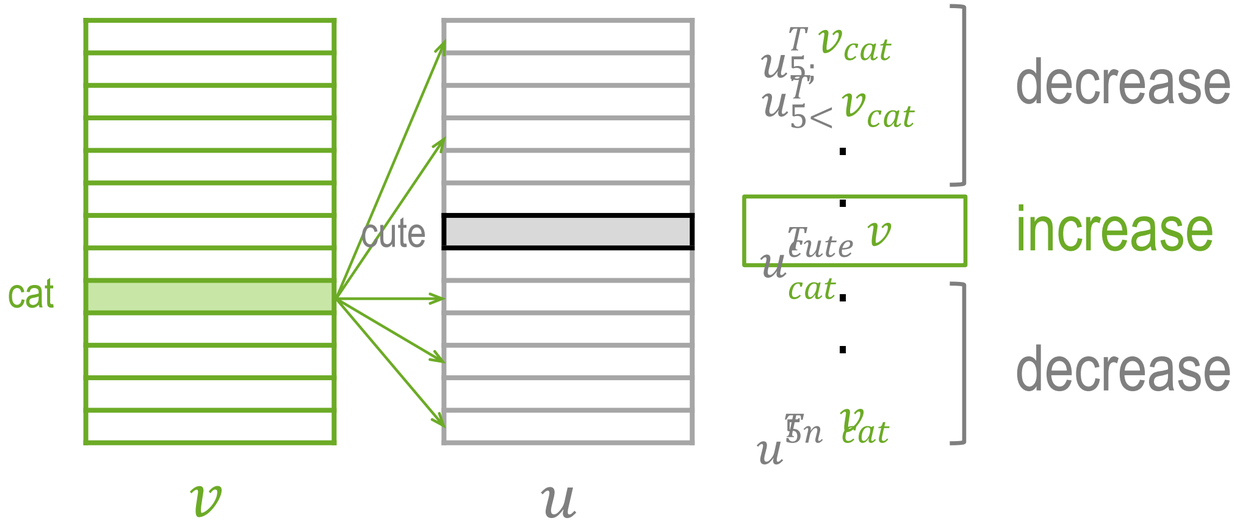
5. evaluate the gradient, make an update

$$v_{cat} := v_{cat} - \alpha \frac{\partial J_{t,\$}(\ )}{\partial v_{cat}}$$

$$u_4 := u_4 - \alpha \frac{\partial J_{t,\$}(\ )}{\partial u_4} \quad \forall w \in V$$

Recap: One Update Intuition

$$-\log P(\text{cute} \mid \text{cat}) = -u_{cute}^{T} v_{cat} + \log \sum_{5 \in V} 4 \exp\left(u_{5}^{T} v_{cat}\right)$$



What’s inside:

Word2Vec

- Idea
- Objective Function
- Training Procedure
- Faster Training: Negative Sampling
- Word2Vec versions: Skip-Gram vs CBOW
- Final Notes

What’s inside:

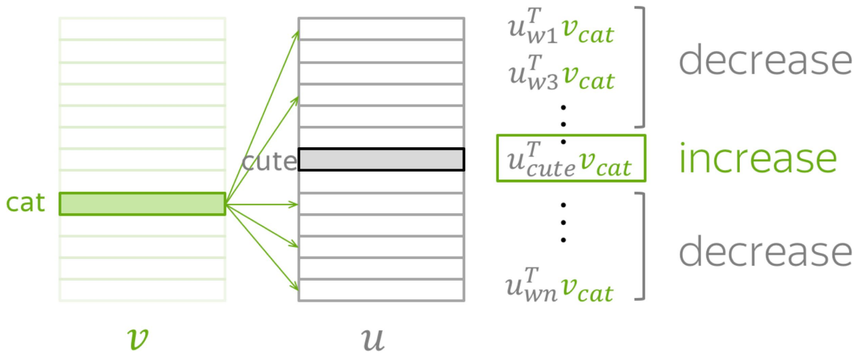
Word2Vec

- Idea
- Objective Function
- Training Procedure
- Faster Training: Negative Sampling
- Word2Vec versions: Skip-Gram vs CBOW
- Final Notes

What happens at each training step

Dot product of v_{cat}:

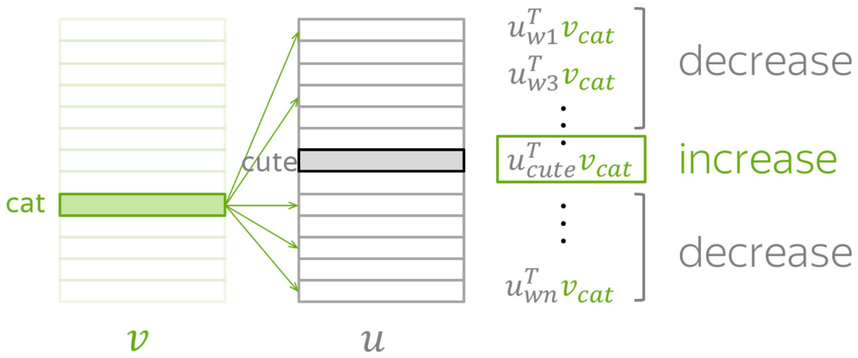
- with u_{cute} - increase,
- with <u>all other</u> u - decrease



What happens at each training step

Dot product of v_{cat}:

- with u_{cute} - increase,
- with <u>all other</u> u - decrease

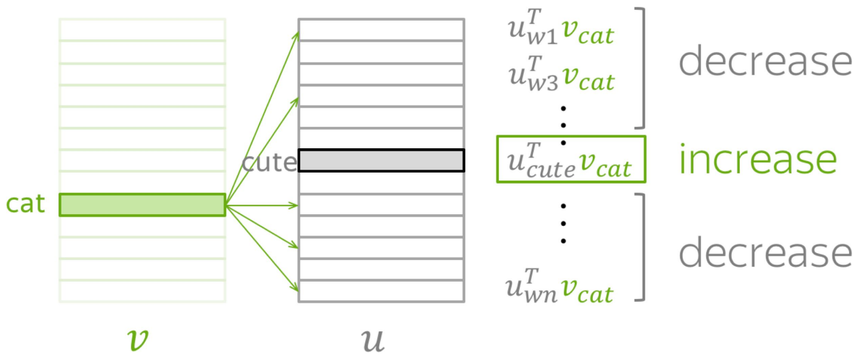


Parameters to be updated:

What happens at each training step

Dot product of v_{cat}:

- with u_{cute} - increase,
- with <u>all other</u> u - decrease



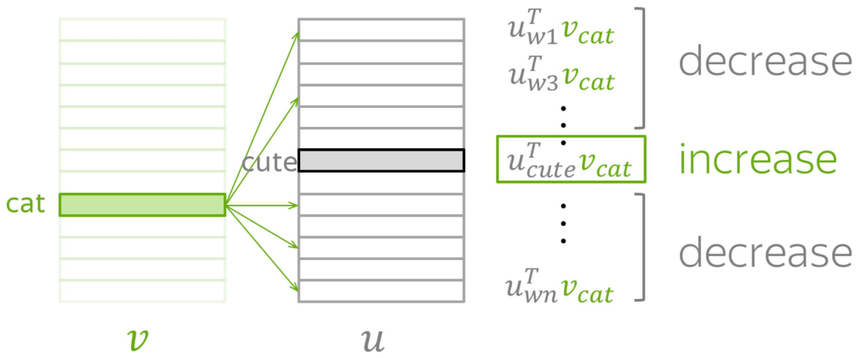
Parameters to be updated:

- v_{cat}

What happens at each training step

Dot product of v_{cat}:

- with u_{cute} - increase,
- with <u>all other</u> u - decrease



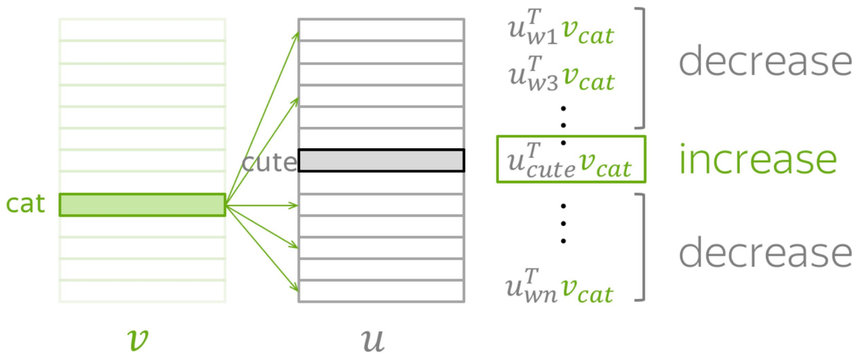
Parameters to be updated:

- v_{cat}
- u_w for all w in the vocabulary

What happens at each training step

Dot product of v_{cat}:

- with u_{cute} - increase,
- with <u>all other</u> u - decrease



Parameters to be updated:

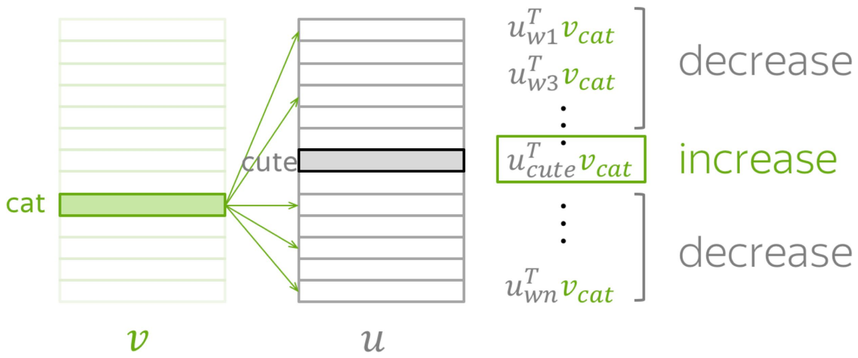
- v_{cat}
- u_w for all w in the vocabulary

|V| + 1 vectors

What happens at each training step

Dot product of v_{cat}:

- with u_{cute} - increase,
- with <u>all other</u> u - decrease



Parameters to be updated:

- v_{cat}
- u_w for all w in the vocabulary

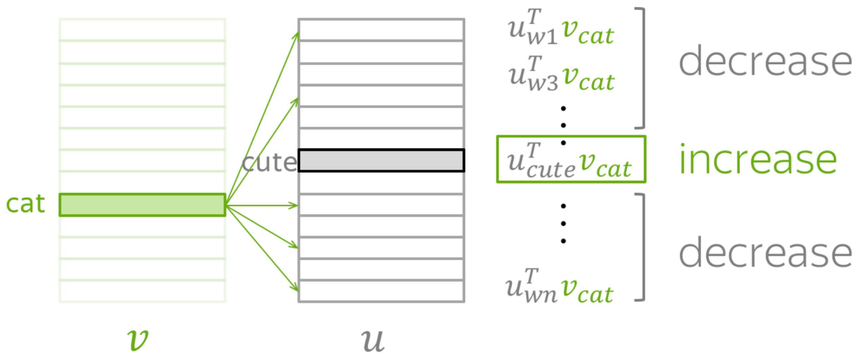
|V| + 1 vectors

Many parameters at each step – slow training

What happens at each training step

Dot product of v_{cat}:

- with u_{cute} - increase,
- with <u>all other</u> u - decrease



Parameters to be updated:

- v_{cat}
- $u_{(}$ for all w in the vocabulary

bad

|V| + 1 vectors

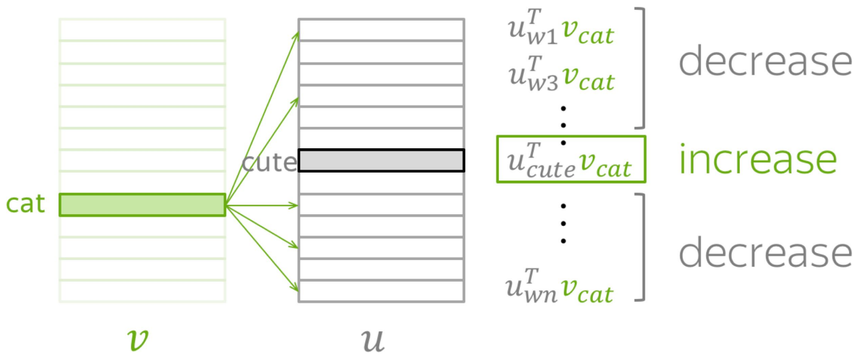
Let’s do better: Negative Sampling

Dot product of v_{cat}:

- with u_{cute} - increase,
- with <u>all other</u> u - decrease

Dot product of v_{cat}:

- with u_{cute} - increase,
- with <u>a subset of other</u> u - decrease



Parameters to be updated:

- v_{cat}
- $u_{(}$ for all w in the vocabulary

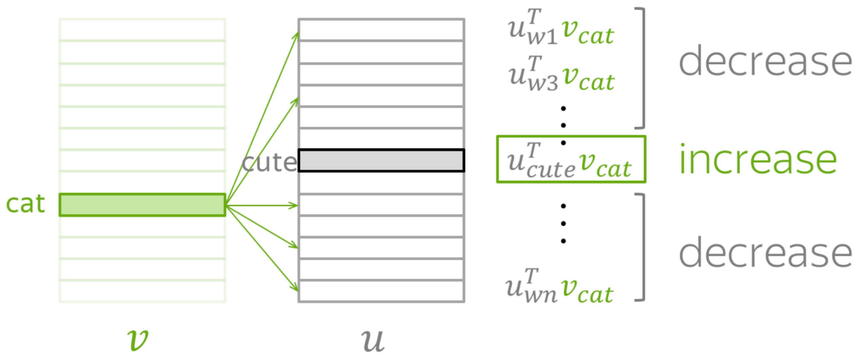
bad

|V| + 1 vectors

Let's do better: Negative Sampling

Dot product of v_{cat}:

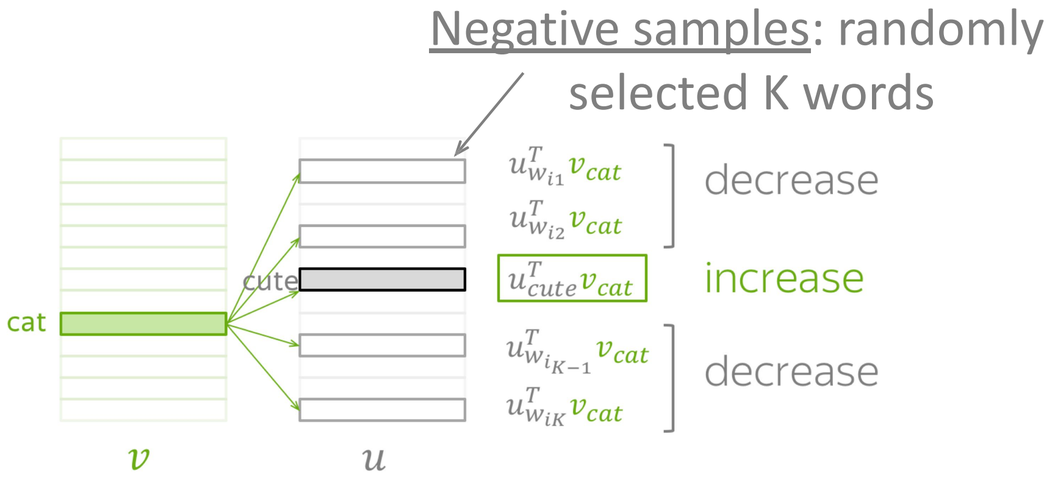
- with u_{cute} - increase,
- with <u>all other</u> u - decrease



Dot product of v_{cat}:

- with u_{cute} - increase,
- with <u>a subset of other</u> u - decrease

<u>Negative samples</u>: randomly selected K words



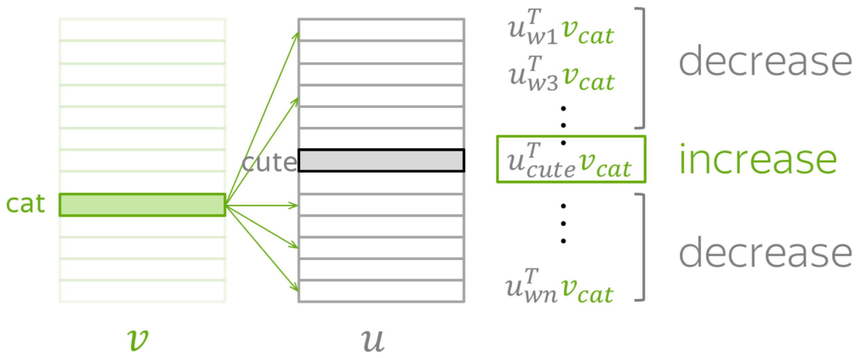
Parameters to be updated: bad

- v_{cat}
- $u_{(}$ for all w in the vocabulary |V| + 1 vectors

Let's do better: Negative Sampling

Dot product of v_{cat}:

- with u_{cute} - increase,
- with <u>all other</u> u - decrease



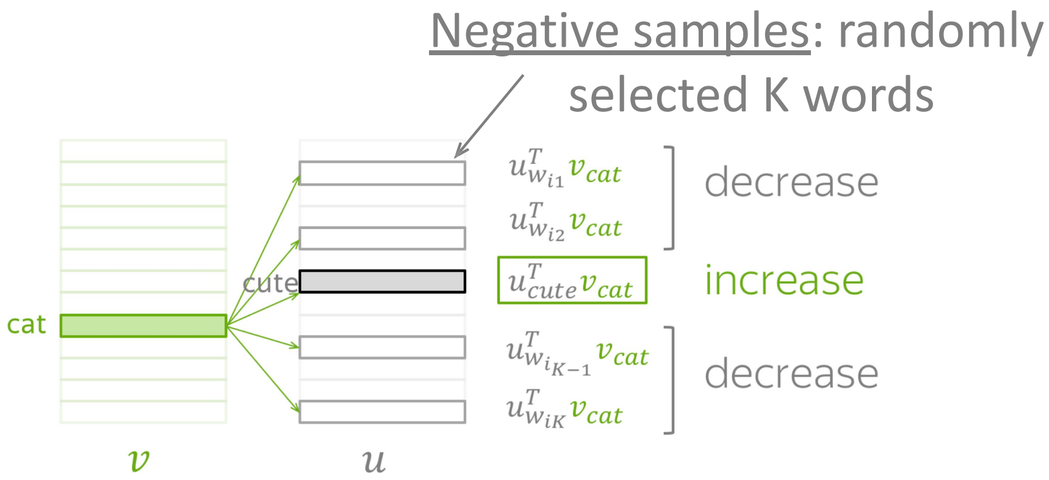
Parameters to be updated: bad

- v_{cat}
- $u_{\cdot}$ for all w in the vocabulary

|V| + 1 vectors

Dot product of v_{cat}:

- with u_{cute} - increase,
- with <u>a subset of other</u> u - decrease



Parameters to be updated:

- v_{cat}
- u_{cute} and $u_{\cdot}$ for w in K negative examples

K + 2 vectors

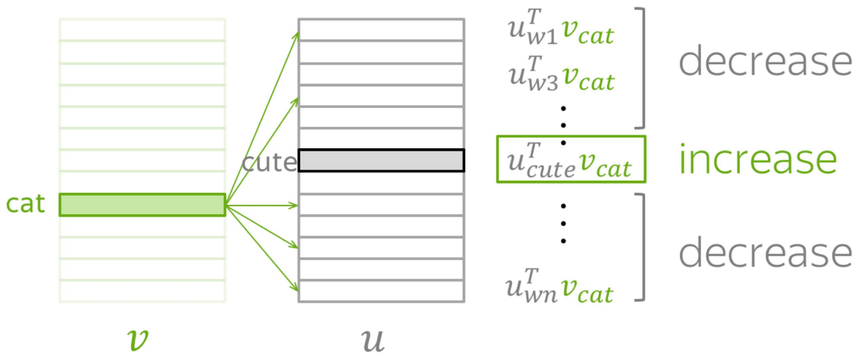
Let’s do better: Negative Sampling

Dot product of v_{cat}:

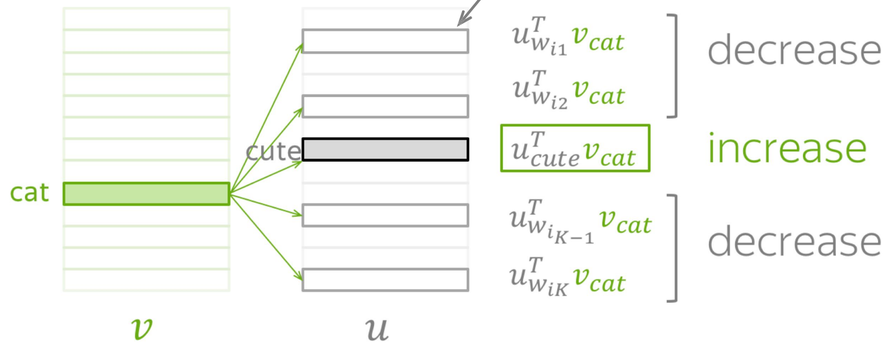
- with u_{cute} - increase,
- with <u>all other</u> u - decrease

Dot product of v_{cat}:

- with u_{cute} - increase,
- with <u>a subset of other</u> u - decrease



<u>Negative samples</u>: randomly selected K words



Parameters to be updated: bad

- v_{cat}
- $u_{\_}$ for all w in the vocabulary

|V| + 1 vectors

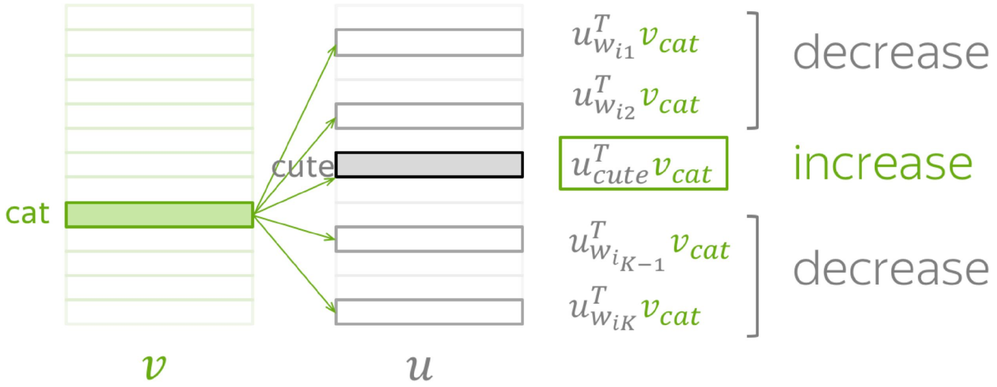
Parameters to be updated: good

- v_{cat}
- u_{cute} and $u_{\_}$ for w in K negative examples

K + 2 vectors

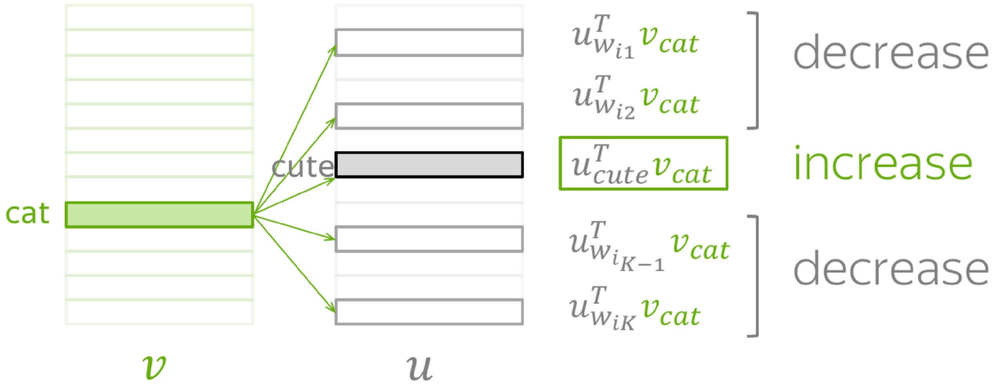
Negative Sampling: Loss Function

We want to do this:



Negative Sampling: Loss Function

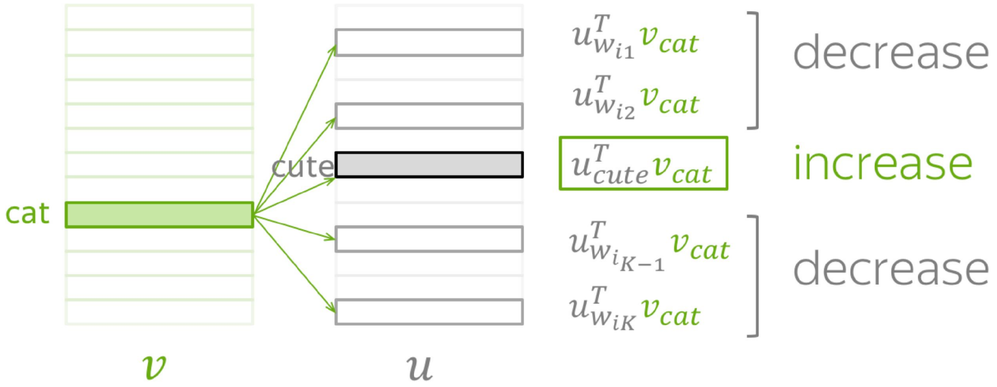
We want to do this: How we do this:



$$J_{t,j}(\theta) = -\log \sigma(u^T_{cute} v_{cat}) - \sum_{w \in \{w_{i_1}, \ldots, w_{i_K}\}} \log \sigma(-u^T_w v_{cat})$$

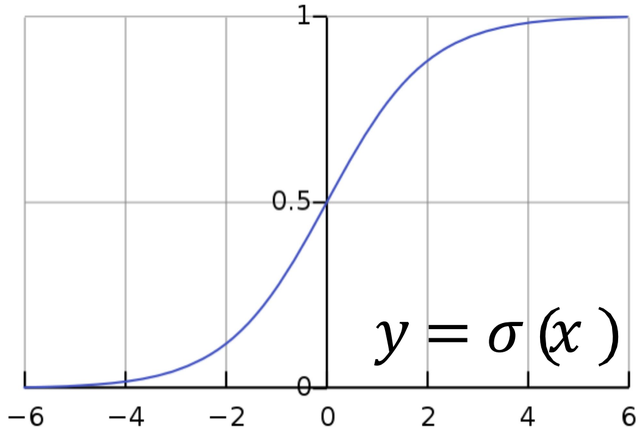
Negative Sampling: Loss Function

We want to do this: How we do this:



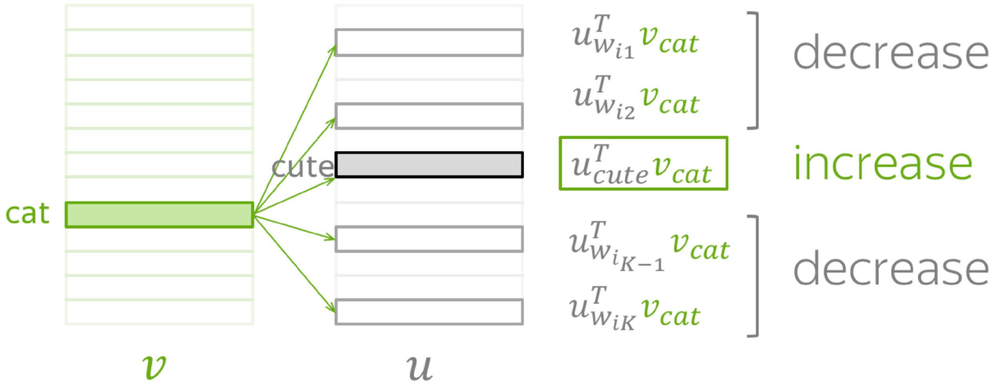
$$J_{t,j}(\theta) = -\log \sigma(u_{cute}^T v_{cat}) - \sum_{w \in \{w_{i_1}, \ldots, w_{i_K}\}} \log \sigma(-u_w^T v_{cat})$$

$\sigma(x) = \frac{\$}{\$\%e^{'x}}$ - the sigmoid function



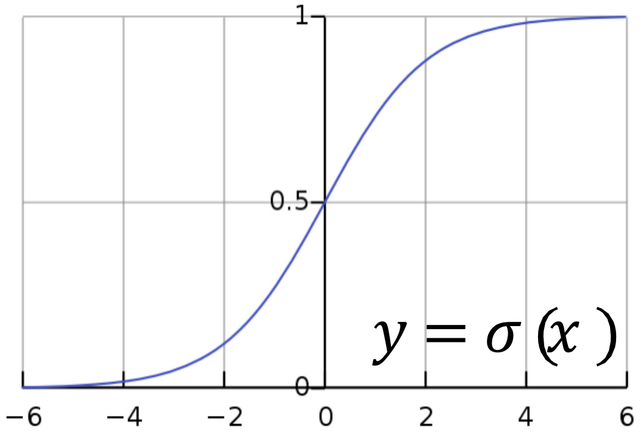
Negative Sampling: Loss Function

We want to do this: How we do this:



$$J_{t,j}(\theta) = -\log \sigma(u^T_{cute} v_{cat}) - \sum_{w \in \{w_{i_1}, \ldots, w_{i_K}\}} \log \sigma(-u^T_w v_{cat})$$

$\sigma(x) = \frac{\$}{\$\%e^{'x}}$ - the sigmoid function

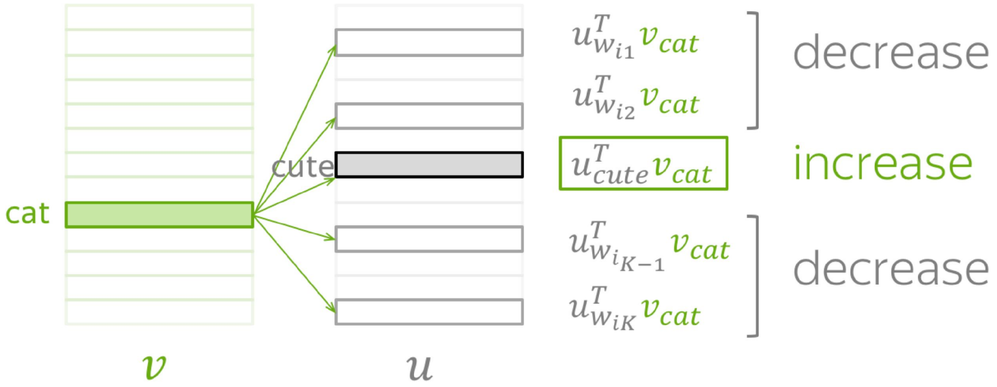


Note: $\sigma(-x) = \frac{1\ 1 / e^{-x} 1}{1 + e^x} = \frac{}{(1 + e^x) / e^{-x}} = \frac{e^{-x}}{1 + e^{-x}} = 1 - \frac{}{1 + e^{-x}} = 1 - \sigma$

(x)

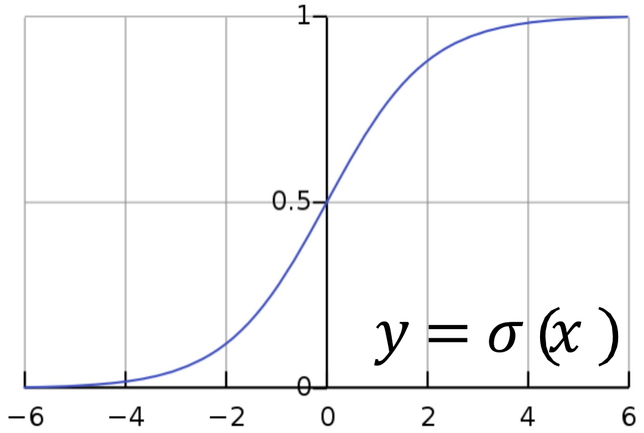
Negative Sampling: Loss Function

We want to do this: How we do this:



$$J_{t,j}(\theta) = -\log \sigma(u^T_{cute} v_{cat}) - \sum_{w \in \{w_{i_1}, \ldots, w_{i_K}\}} \log \sigma(-u^T_w v_{cat})$$

$\sigma(x) = \frac{\$}{\$\%e^{'x}}$ - the sigmoid function



Note: $\sigma(-x) = \frac{1\ 1 / e^{-x} 1}{1 + e^x} = \frac{}{(1 + e^x) / e^{-x}} = \frac{e^{-x}}{1 + e^{-x}} = 1 - \frac{}{1 + e^{-x}} = 1 - \sigma$

Then: $$J_{t,j}(\theta) = -\log \sigma(u^T_{cute} v_{cat}) - \sum_{w \in \{w_{i_1}, \ldots, w_{i_K}\}} \log(1 - \sigma(u^T_w v_{cat}))$$

What’s inside:

Word2Vec

- Idea
- Objective Function
- Training Procedure
- Faster Training: Negative Sampling
- Word2Vec versions: Skip-Gram vs CBOW
- Final Notes

What’s inside:

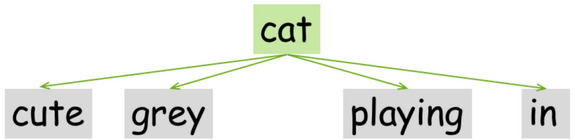
Word2Vec

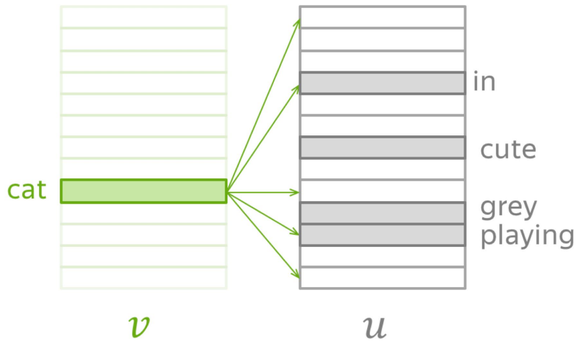
- Idea
- Objective Function
- Training Procedure
- Faster Training: Negative Sampling
- Word2Vec versions: Skip-Gram vs CBOW
- Final Notes

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)





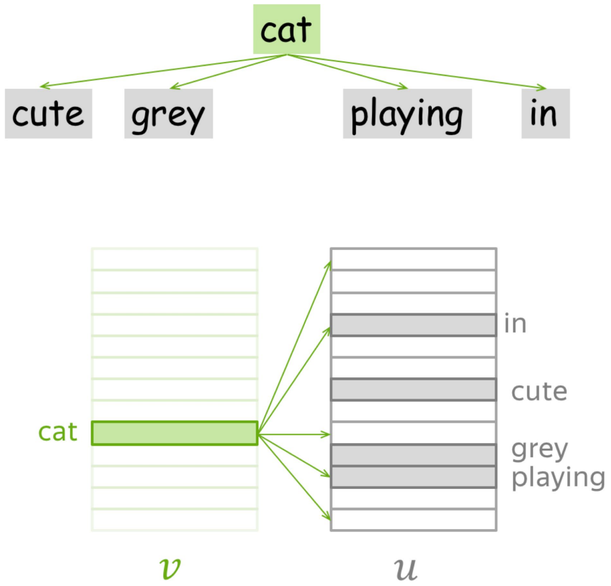
(this is what we did so far)

Word2Vec Variants: Skip-Gram and CBOW

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

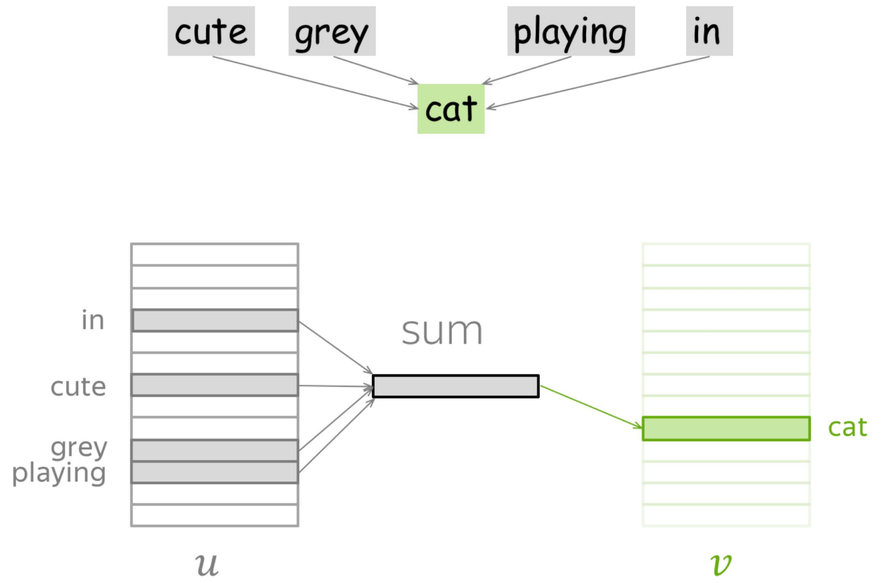
Skip-Gram: from central predict context

(one at a time)



(this is what we did so far)

CBOW: from sum of context predict central



(Continuous Bag of Words)

What’s inside:

Word2Vec

- Idea
- Objective Function
- Training Procedure
- Faster Training: Negative Sampling
- Word2Vec versions: Skip-Gram vs CBOW
- Final Notes

What’s inside:

Word2Vec

- Idea
- Objective Function
- Training Procedure
- Faster Training: Negative Sampling
- Word2Vec versions: Skip-Gram vs CBOW
- Final Notes

(Somewhat) Standard Hyperparameters

- Model: Skip-Gram with negative sampling;

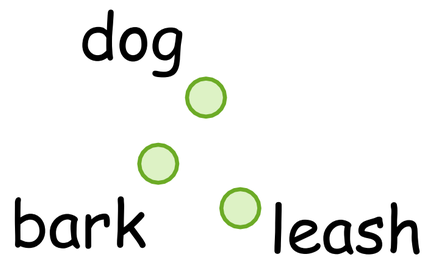
- Number of negative examples: for smaller datasets, 15-20; for huge datasets (which are usually used) it can be 2-5.

- Embedding dimensionality: frequently used value is 300, but other variants (e.g., 100 or 50) are also possible.

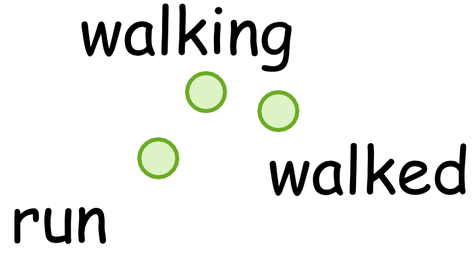
- Sliding window (context) size: 5-10.

The Effect of Context Window Size

- Larger windows – more topical similarities

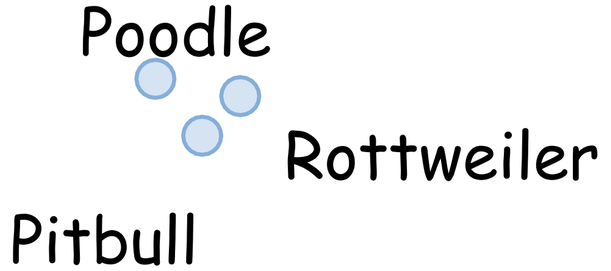


(grouped together)



(grouped together)

- Smaller windows – more functional and syntactic similarities



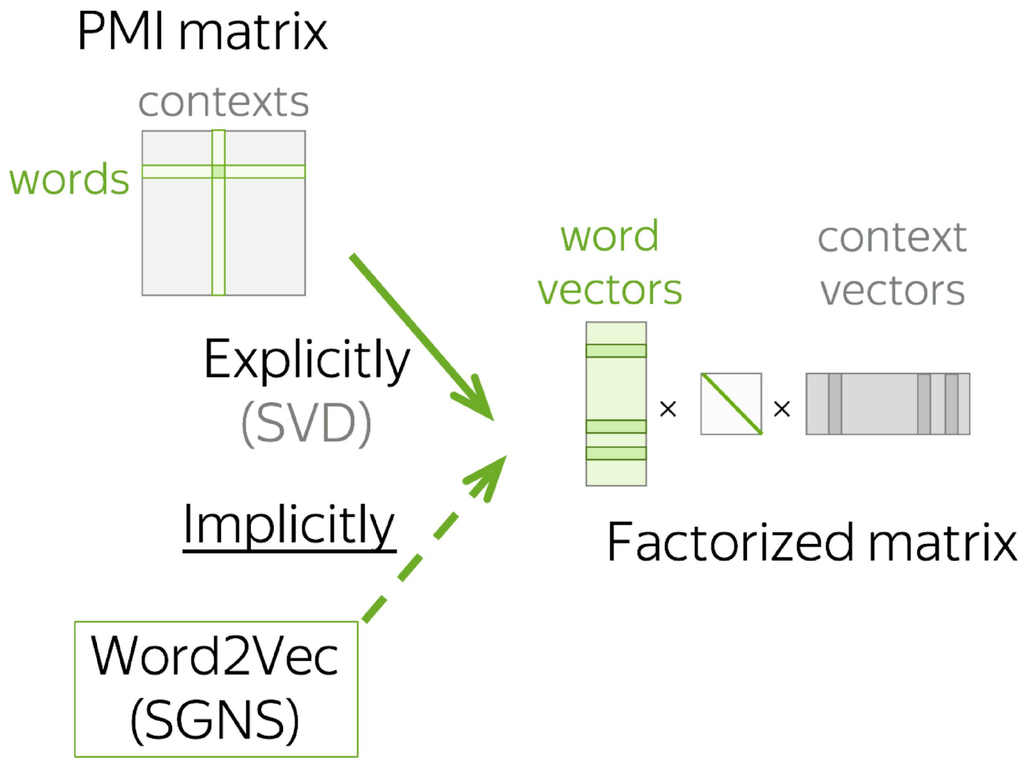
(grouped together)

(grouped together)

Relation to PMI Matrix Factorization

NLP Course For You

Word2Vec SGNS (Skip-Gram with Negative Sampling) implicitly approximates the factorization of a (shifted) PMI matrix. Learn more here.



Relation to PMI Matrix Factorization

Related Papers

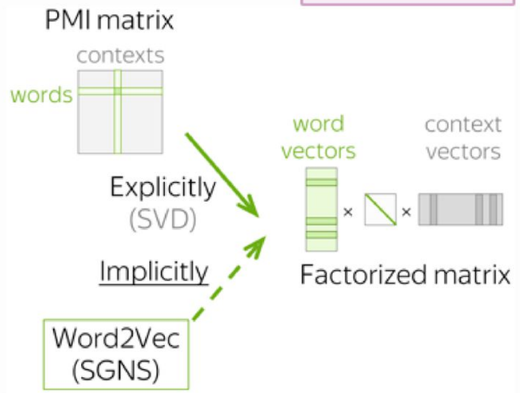
Good Old Classics

Neural Word Embedding as Implicit Matrix Factorization

Omer Levy, Yoav Goldberg

NeurIPS 2014

Theoretically, Word2Vec is not so different from matrix factorization approaches! Skip-gram with negative-sampling (SGNS) implicitly factorizes the shifted pointwise mutual information (PMI) matrix: $PMI(w, c) - \log k$, where k is the number of negative examples in negative sampling.



▶ More details

More Hacks in Training

Research Thinking

Word2Vec

Are all context words equally important for training?

During Word2Vec training, we make an update for each of the context words. For example, for the central word **cat** we make an update for each of the words **cute**, **grey**, **playing**, **in**.

? Are all context words equally important?

Which word types give more/less information than others? Think about some characteristics of words that can influence their importance. Do not forget the previous exercise!

▶ Possible answers

? How can we use this to modify training?

▶ Tricks from the original Word2Vec

NLP Course For You

https://lena-voita.github.io/nlp_course/word_embeddings.html

What is going to happen:

- Why do we need word representations?
- One-hot Vectors
- Distributional Semantics
- Count-Based Methods
- Word2Vec (Prediction-based Method)
- GloVe
- Evaluation
-  Analysis and Interpretability

What is going to happen:

- Why do we need word representations?
- One-hot Vectors
- Distributional Semantics
- Count-Based Methods
- Word2Vec (Prediction-based Method)
- GloVe
- Evaluation
-  Analysis and Interpretability

GloVe

(kind of both count-based and prediction-based)

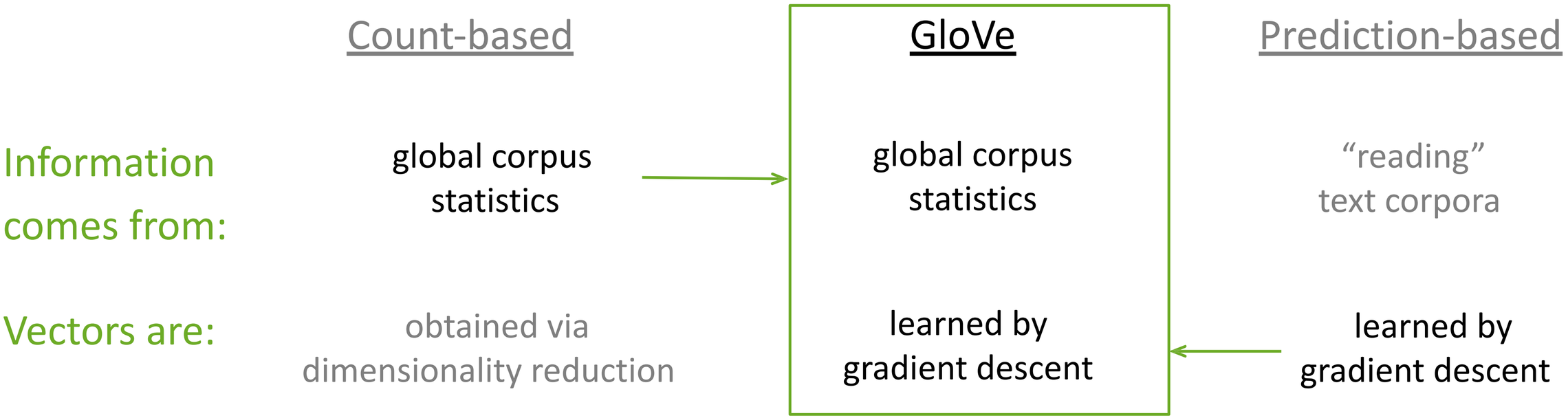
GloVe: Global Vectors for Word Representation

	Count-based
Information comes from:	global corpus statistics
Vectors are:	obtained via dimensionality reduction

GloVe: Global Vectors for Word Representation

	Count-based	Prediction-based
Information comes from:	global corpus statistics	"reading" text corpora
Vectors are:	obtained via dimensionality reduction	learned by gradient descent

GloVe: Global Vectors for Word Representation



GloVe: Global Vectors for Word Representation

context vector

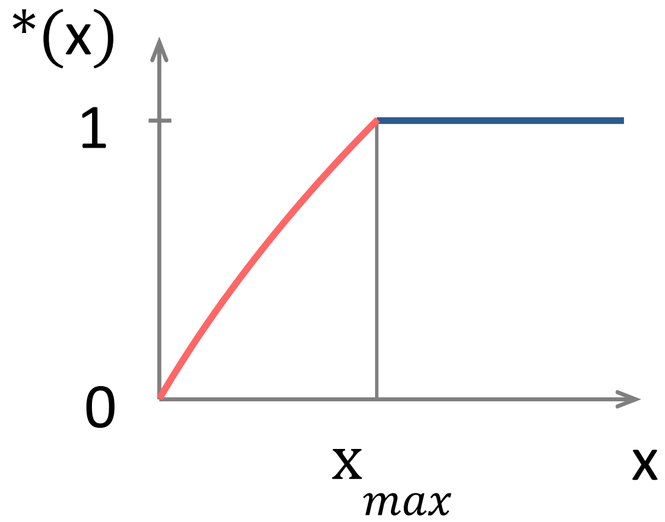
word vector

bias terms (also learned)

$$J(\text{"}) = \sum_{w,c \in V} *(\mathrm{N}(w, c)) - (u^T v_c + b_c + \overline{b}_w - \log \mathrm{N}(w, c))^2$$

Weighting function to:

- penalize rare events
- not to over-weight frequent events



$$*(x) = \begin{cases} (x/x_{max})^{\alpha} \text{ if } x < x_{max}, \\ 1 \text{ otherwise.} \end{cases}$$

$\alpha = 0.75$, $x_{max} = 100$

What is going to happen:

- Why do we need word representations?
- One-hot Vectors
- Distributional Semantics
- Count-Based Methods
- Word2Vec (Prediction-based Method)
- GloVe
- Evaluation
-  Analysis and Interpretability

What is going to happen:

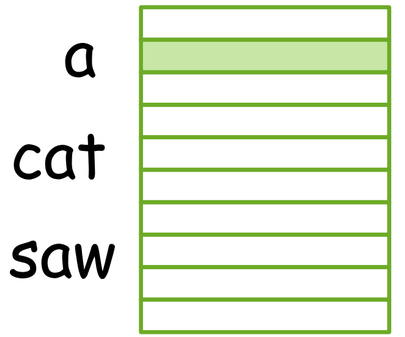
- Why do we need word representations?
- One-hot Vectors
- Distributional Semantics
- Count-Based Methods
- Word2Vec (Prediction-based Method)
- GloVe
- Evaluation
-  Analysis and Interpretability

Evaluation: Intrinsic and Extrinsic

Intrinsic Evaluation: Based on Internal Properties

How well do embeddings capture meaning?

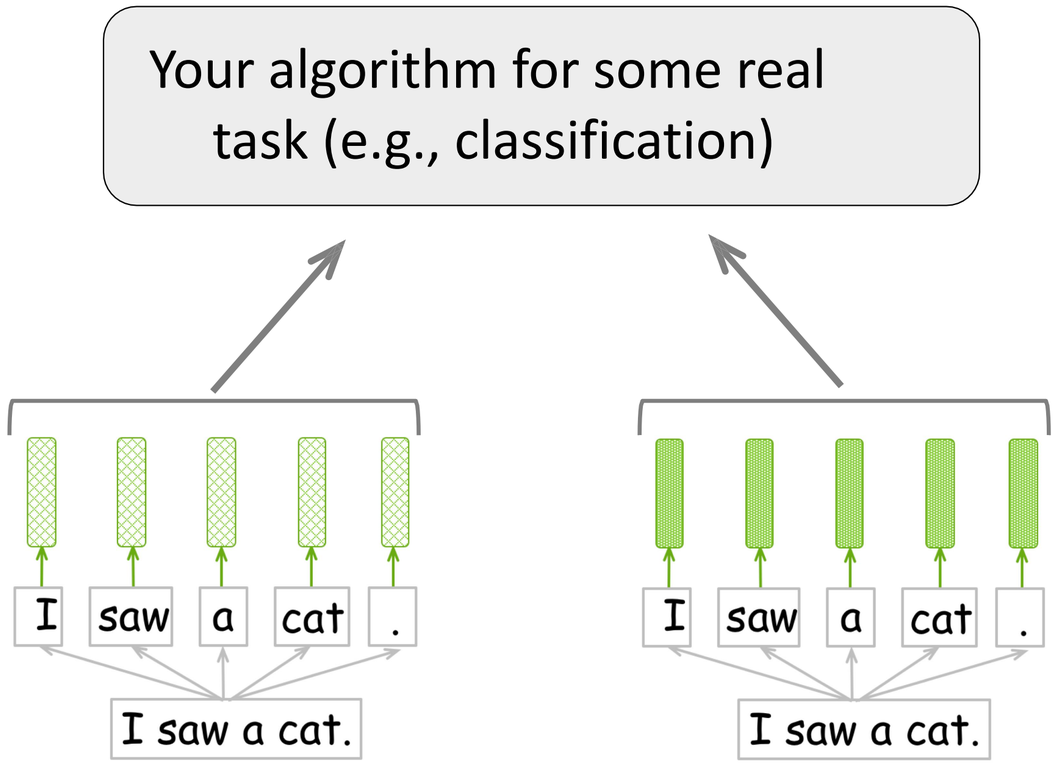
- word similarity
- word analogy
- …



Look-up table

We'll look at this in detail in the Analysis and Interpretability part!

Extrinsic Evaluation: On a Real Task



some embeddings

some other embeddings

Model with which embeddings performs better?

Train the same model several times: one model for each embedding set

For the same dataset, you can get representations using different word embeddings

Intrinsic vs Extrinsic: How to Choose?

Intrinsic:

- usually fast
- does not tell what is better in practice

Extrinsic:

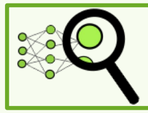
- tells directly what is better in practice
- training several real-task models is expensive

…in the end, this is up to you

What is going to happen:

- Why do we need word representations?
- One-hot Vectors
- Distributional Semantics
- Count-Based Methods
- Word2Vec (Prediction-based Method)
- GloVe
- Evaluation
-  Analysis and Interpretability

What is going to happen:

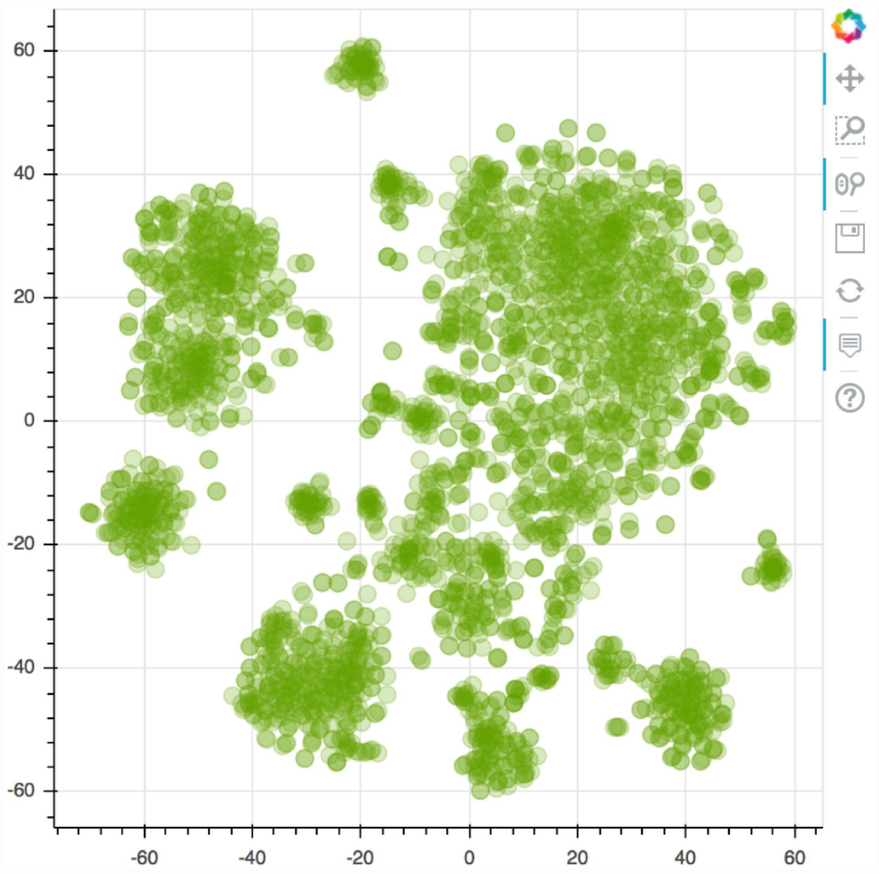
- Why do we need word representations?
- One-hot Vectors
- Distributional Semantics
- Count-Based Methods
- Word2Vec (Prediction-based Method)
- GloVe
- Evaluation
-  Analysis and Interpretability

Analysis and Interpretability

Take a Walk Through Space… Semantic Space!

How to: Walk through semantic space and try to find:

- language clusters: Spanish, Arabic, Russian, English. Can you find more languages?
- clusters for: food, family, names, geographical locations. What else can you find?



Open right now and take your time!

(come back in 5-10 minutes)

NLP Course For You

https://lena-voita.github.io/nlp_course/word_embeddings.html

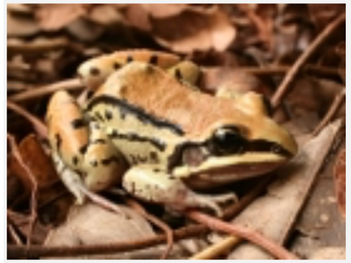
Nearest Neighbors

Closest to `frog`:

- frogs
- toad
- litoria
- leptodactylidae
- rana
- lizard
- eleutherodactylus

litoria

leptodactylidae



rana

eleutherodactylus

The example is from the GloVe project page.

Word Similarity Benchmarks

The benchmarks that estimate the quality of embeddings:

- Contain word pairs with similarity scores from humans
- For evaluation, use correlation between the two similarity scores: from humans and from embeddings

word	pair	score
vulgarism	profanity	9.62
subdividing	separate	8.67
friendships	brotherhood	7.5
exceedance	probability	5.0
assigned	allow	3.5
marginalize	interact	2.5
misleading	beat	1.25
radiators	beginning	0

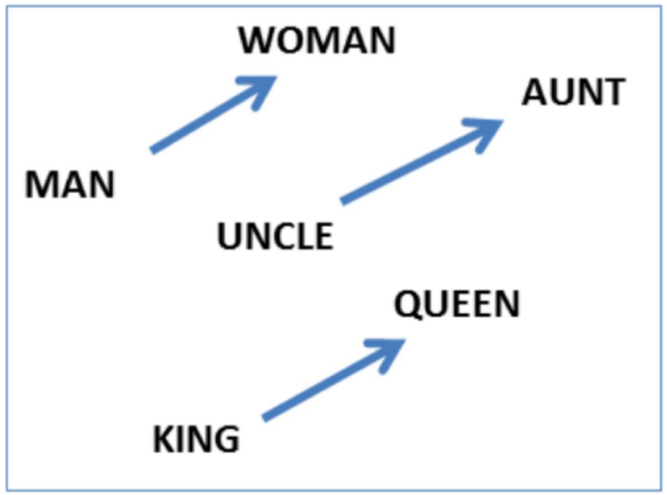
Several pairs from the Rare Words similarity benchmark.

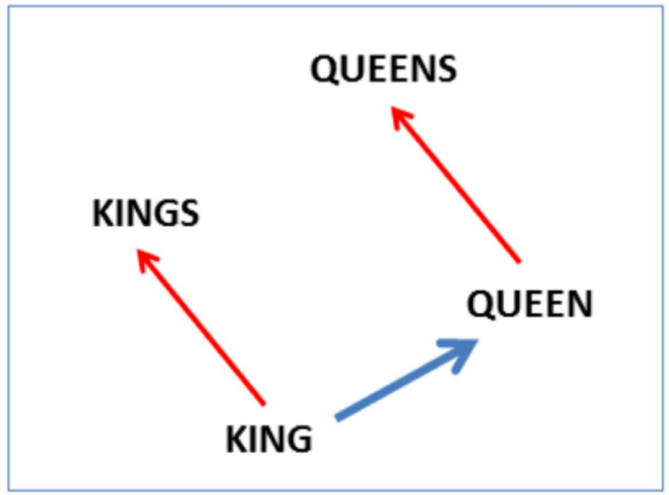
Linear Structure

Many semantic and syntactic relationship between words are (almost) linear in the embedding space!

semantic: $v(king) - v(man) + v(woman) \approx v(queen)$

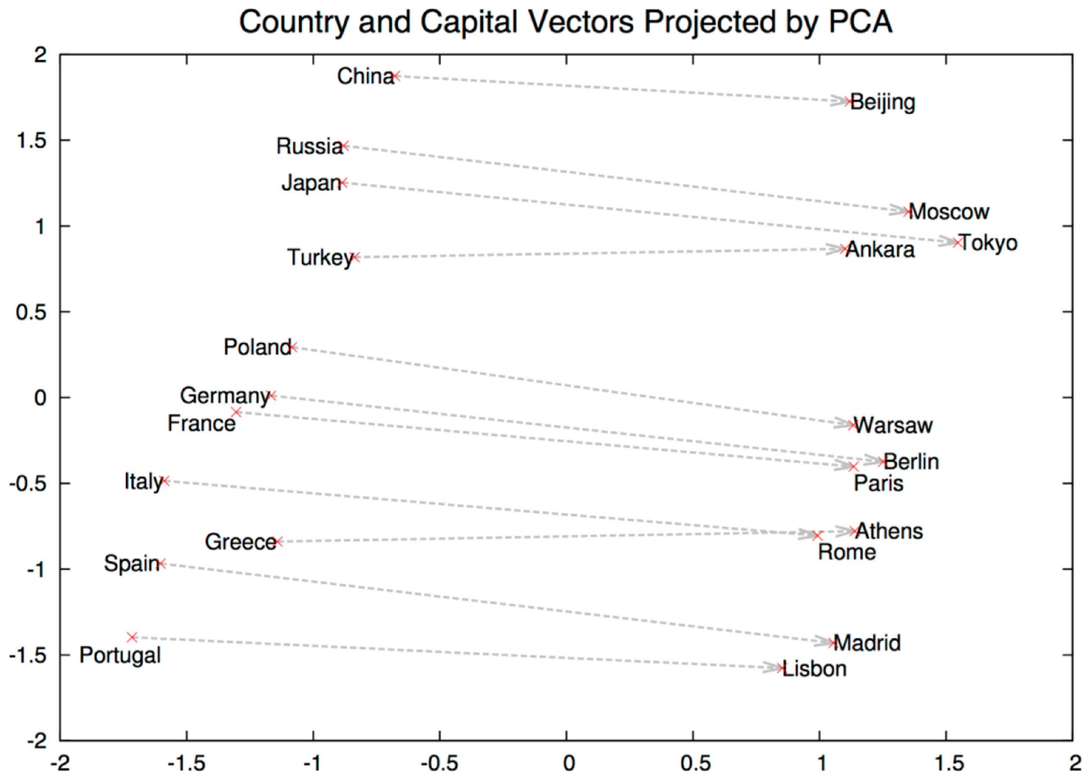
syntactic: $v(kings) - v(king) + v(queen) \approx v(queens)$



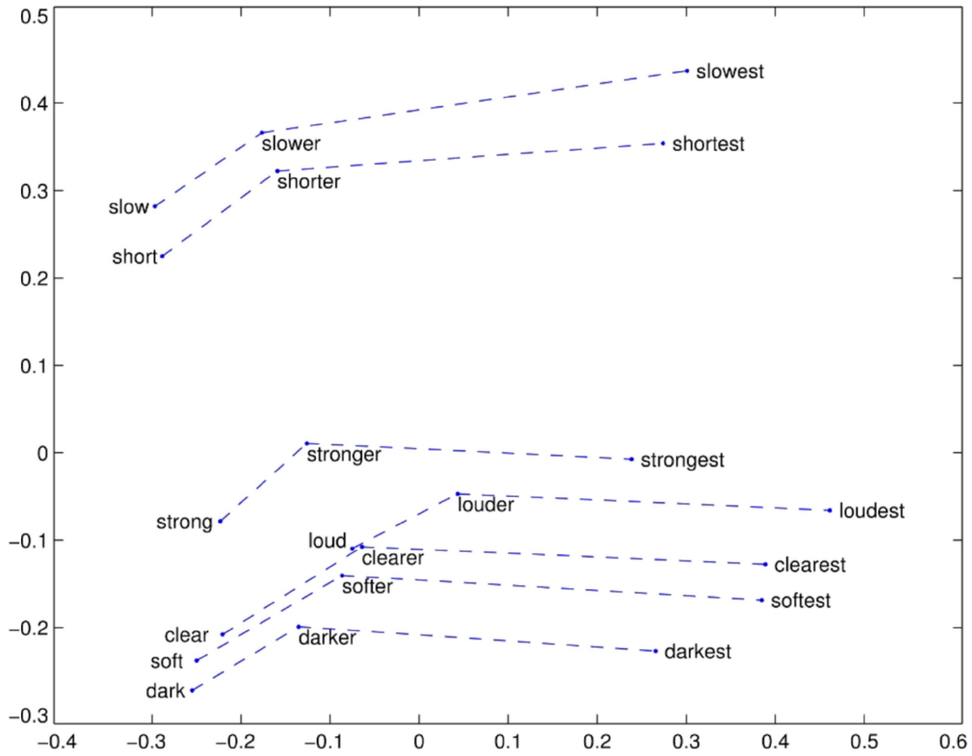


Linear Structure

Many semantic and syntactic relationship between words are (almost) linear in the embedding space!



Word2Vec



GloVe

Word Similarity Benchmarks

Analogy: a is to a^* as b is to

Task: $v(a^*) - v(a) + v(b) \approx ?$ ⟶

- find the closest vector
- check if it corresponds to the correct word

relation	word pair 1		word pair 2	
man-woman	brother	sister	grandson	granddaughter
currency	Angola	kwanza	Iran	rial
opposite	possibly	impossibly	ethical	unethical
past tense	walking	walked	swimming	swam
superlative	easy	easiest	lucky	luckiest

Examples of relations and word pairs from the Google analogy test set.

Similarities Across Languages

The recipe for building large dictionaries from small ones

Ingredients:

- corpus in one language
  (e.g., English)
- corpus in another language
  (e.g., Spanish)

- very small dictionary

  cat ↔ gato
  cow ↔ vaca
  dog ↔ perro
  fox ↔ zorro
  …

Similarities Across Languages

The recipe for building large dictionaries from small ones

Ingredients:

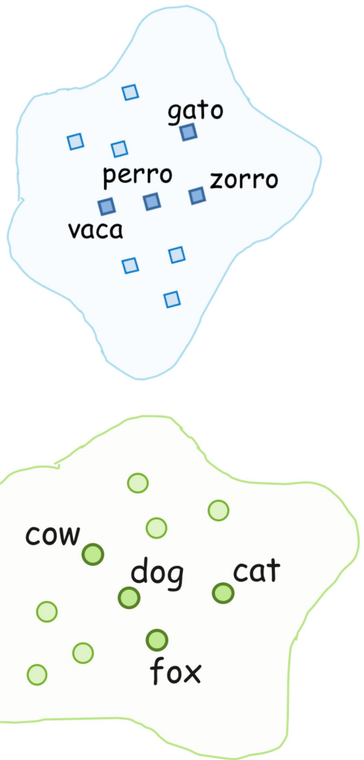
- corpus in one language (e.g., English)
- corpus in another language (e.g., Spanish)

- very small dictionary

cat ↔ gato
cow ↔ vaca
dog ↔ perro
fox ↔ zorro
…

Step 1:

- train embeddings for each language



Similarities Across Languages

The recipe for building large dictionaries from small ones

Ingredients:

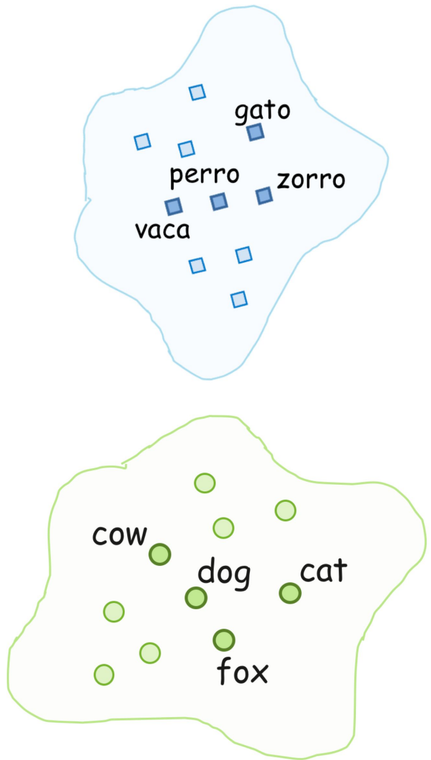
- corpus in one language (e.g., English)
- corpus in another language (e.g., Spanish)

- very small dictionary

cat ↔ gato
cow ↔ vaca
dog ↔ perro
fox ↔ zorro
…

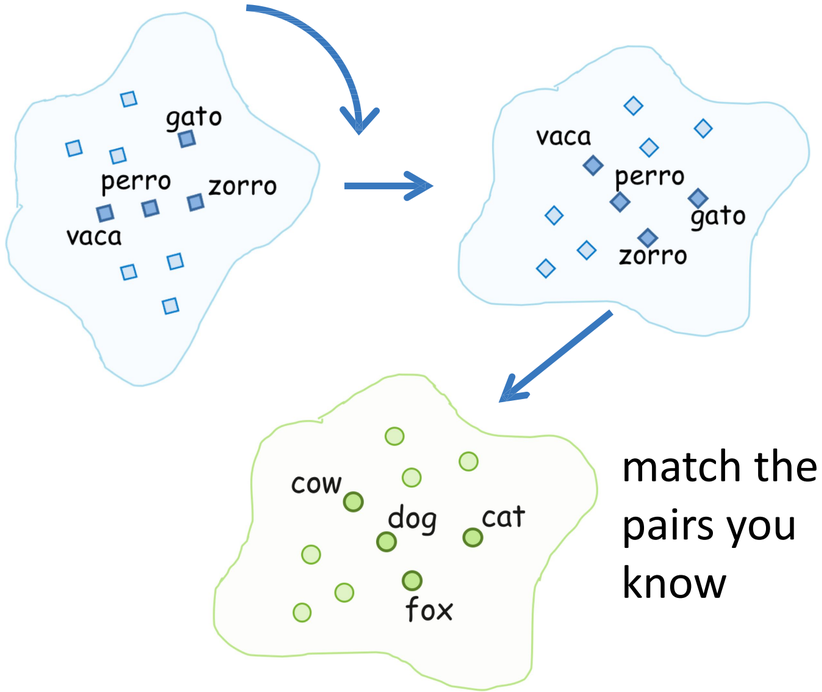
Step 1:

- train embeddings for each language



Step 2:

- linearly map one embeddings to the other to match words from the dictionary



match the pairs you know

Similarities Across Languages

The recipe for building large dictionaries from small ones

Ingredients:

- corpus in one language (e.g., English)
- corpus in another language (e.g., Spanish)

- very small dictionary

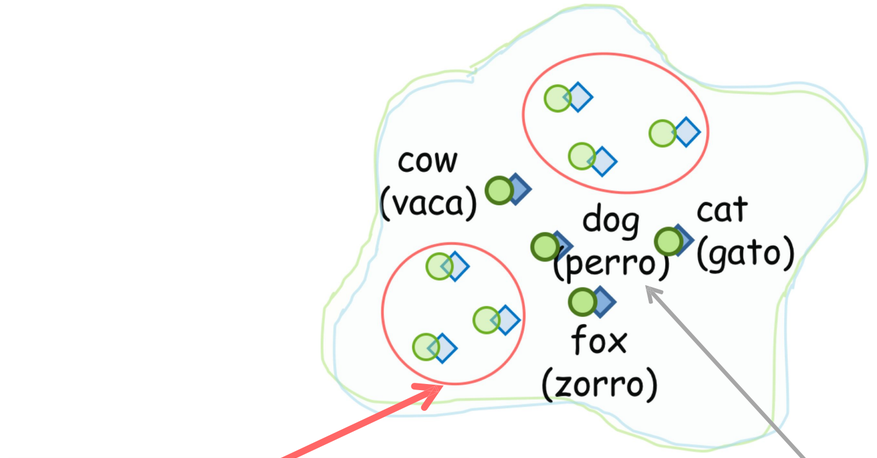
cat ↔ gato
cow ↔ vaca
dog ↔ perro
fox ↔ zorro
…

Steps1-2:

- match words from the vocabulary

Step 3:

- after matching the two spaces, get new pairs from the new matches



new translations – you learned them!

old translations – the ones you knew

More in NLP Course For You

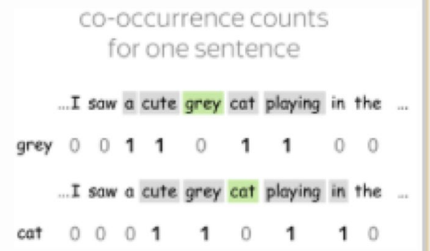
NLP Course For You

Research Thinking

Count-Based Methods

Improve Simple Co-Occurrence Counts

The simplest co-occurrence counts treat context words equally, although these words are at different relative positions from the central word. For example, from one sentence the central word **cat** will get a co-occurrence count of 1 for each of the words **cute**, **grey**, **playing**, **in** (look at the example to the right).



? Are context words at different distances equally important? If not, how can we modify co-occurrence counts?

▶ Possible answers

? In language, word order is important; specifically, left and right contexts have different meanings. How can we distinguish between the left and right contexts?

▶ One of the existing approaches

NLP Course For You

Research Thinking

Related Papers

What's inside:

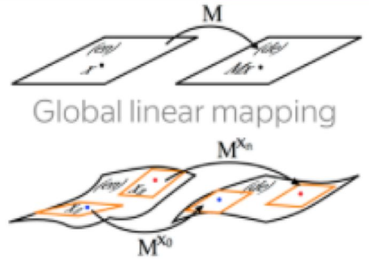
- Good Old Classics
- Analyzing Geometry
- Biases in Word Embeddings
- Semantic Change
- Theory to the Rescue! - coming soon
- Cross-Lingual Embeddings - coming soon
- ... to be updated

ACL 2018

Characterizing Departures from Linearity in Word Translation

Ndapa Nakashole, Raphael Flauger

We learned that we can (almost) match semantic spaces for different languages linearly. But is the "true" underlying mapping between languages indeed linear? If it is linear globally, then all local linear mappings have to be similar (to the global linear mapping, and hence to each other). Well, they are not.



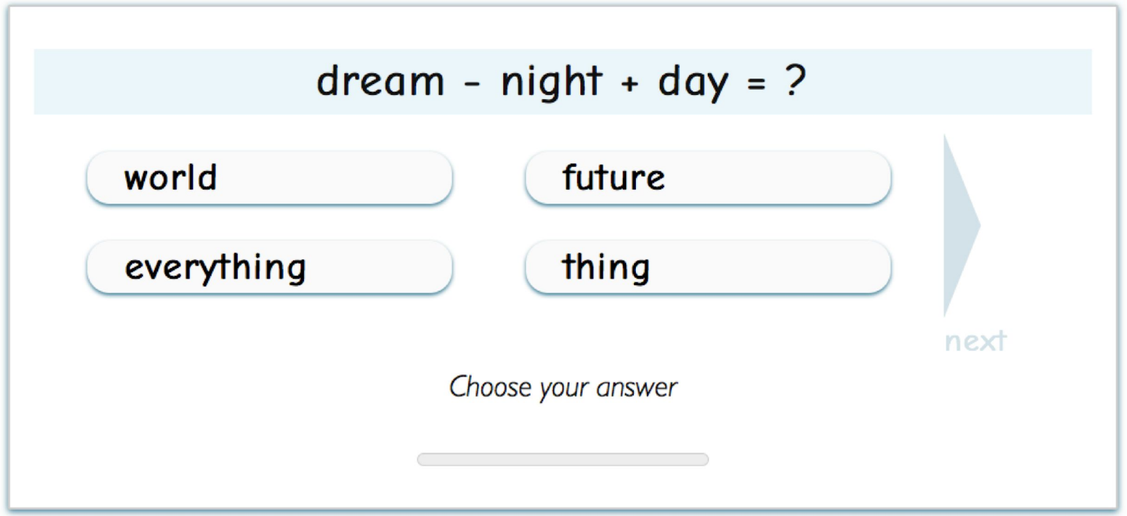
▶ More details

NLP Course For You

Research Thinking

Related Papers

Have Fun!



Thank you!

Lena Voita

PhD student, Uni Edinburgh & Uni Amsterdam

lena-voita@hotmail.com

https://lena-voita.github.io

@lena_voita

lena-voita