

Word Embeddings

SEPTEMBER 12, 2018

Lena Voita

 lena-voita@yandex-team.ru

 <https://lena-voita.github.io>

 @lena_voita

 lena-voita

Plan

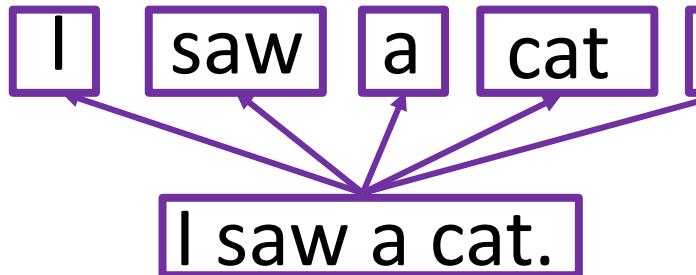
- Why do we need word representations?
- Distributional semantics
- Word2Vec in detail
- Glove overview
- Let's take a walk!
- Further directions and recent advances
- Hack of the day!

Why do we need word representation?

I saw a cat.

Text

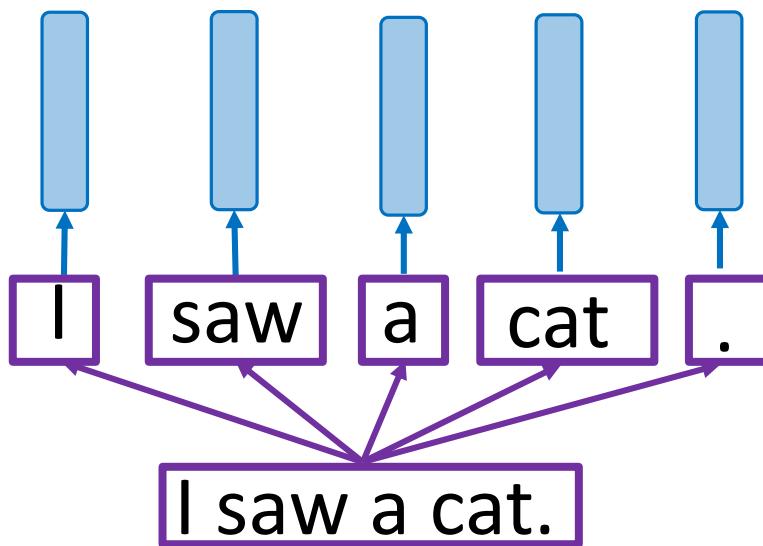
Why do we need word representation?



Sequence of tokens

Text

Why do we need word representation?

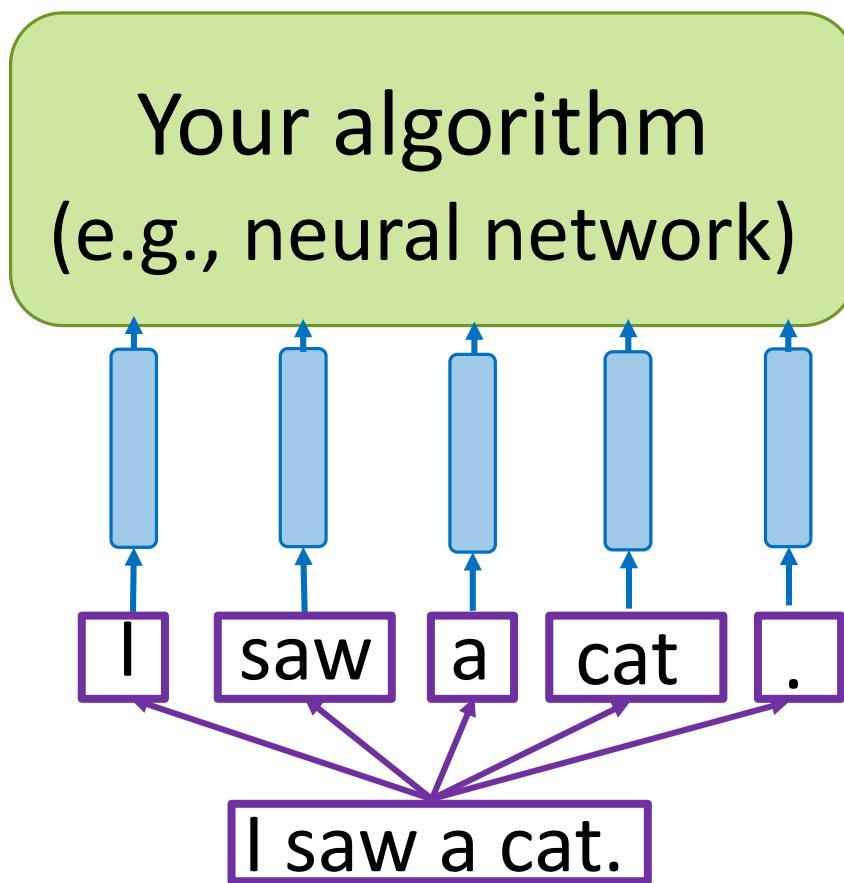


Word representation - vector
(word embedding)

Sequence of tokens

Text

Why do we need word representation?



Any algorithm for solving any task

Word representation - vector
(word embedding)

Sequence of tokens

Text

How it works: Look-up Table (Vocabulary)

I saw a cat .

39 1592 10 2548 5

Token index in
the vocabulary

Embedding dimension

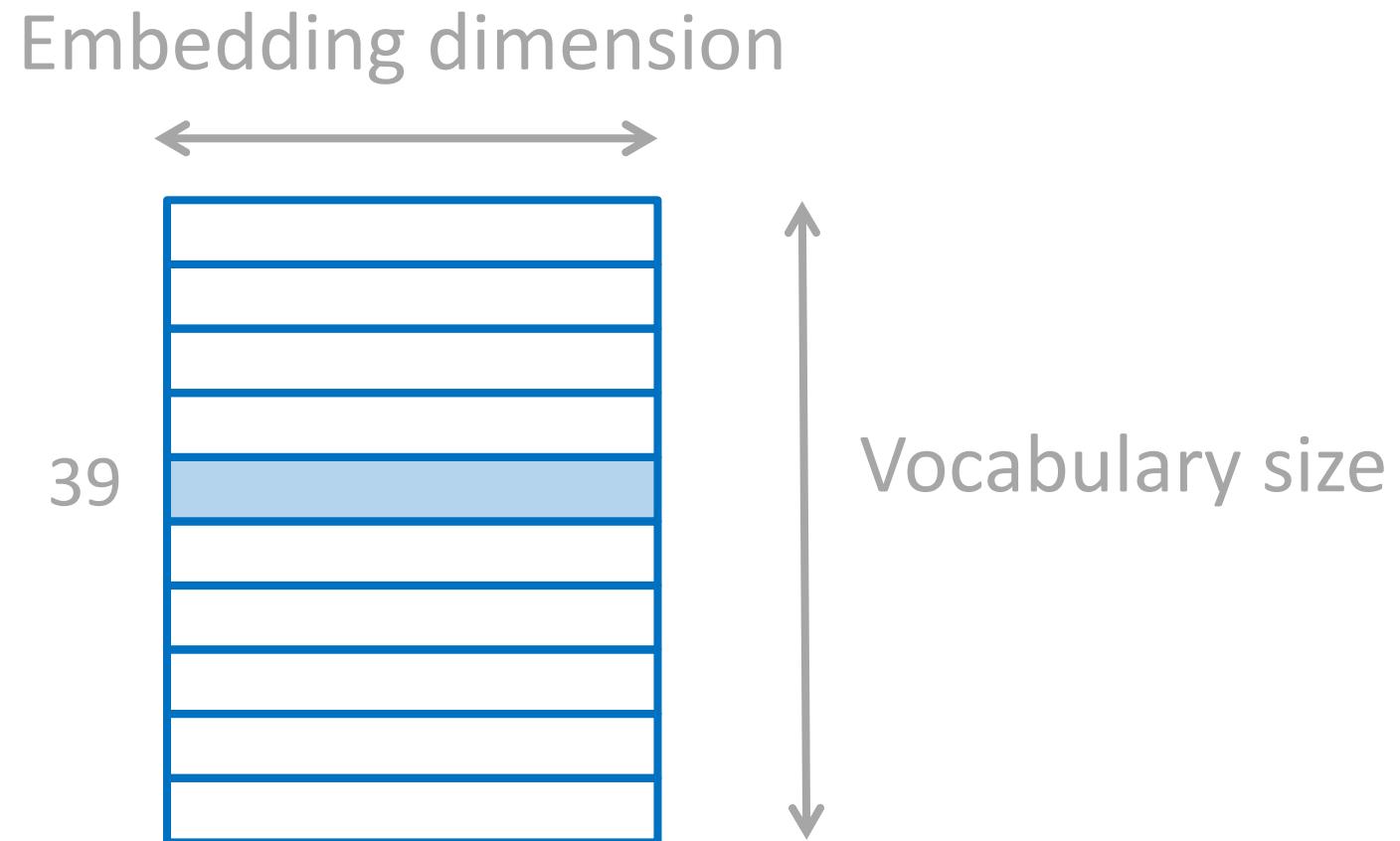


Vocabulary size

How it works: Look-up Table (Vocabulary)

I saw a cat .
39 1592 10 2548 5

Token index in
the vocabulary

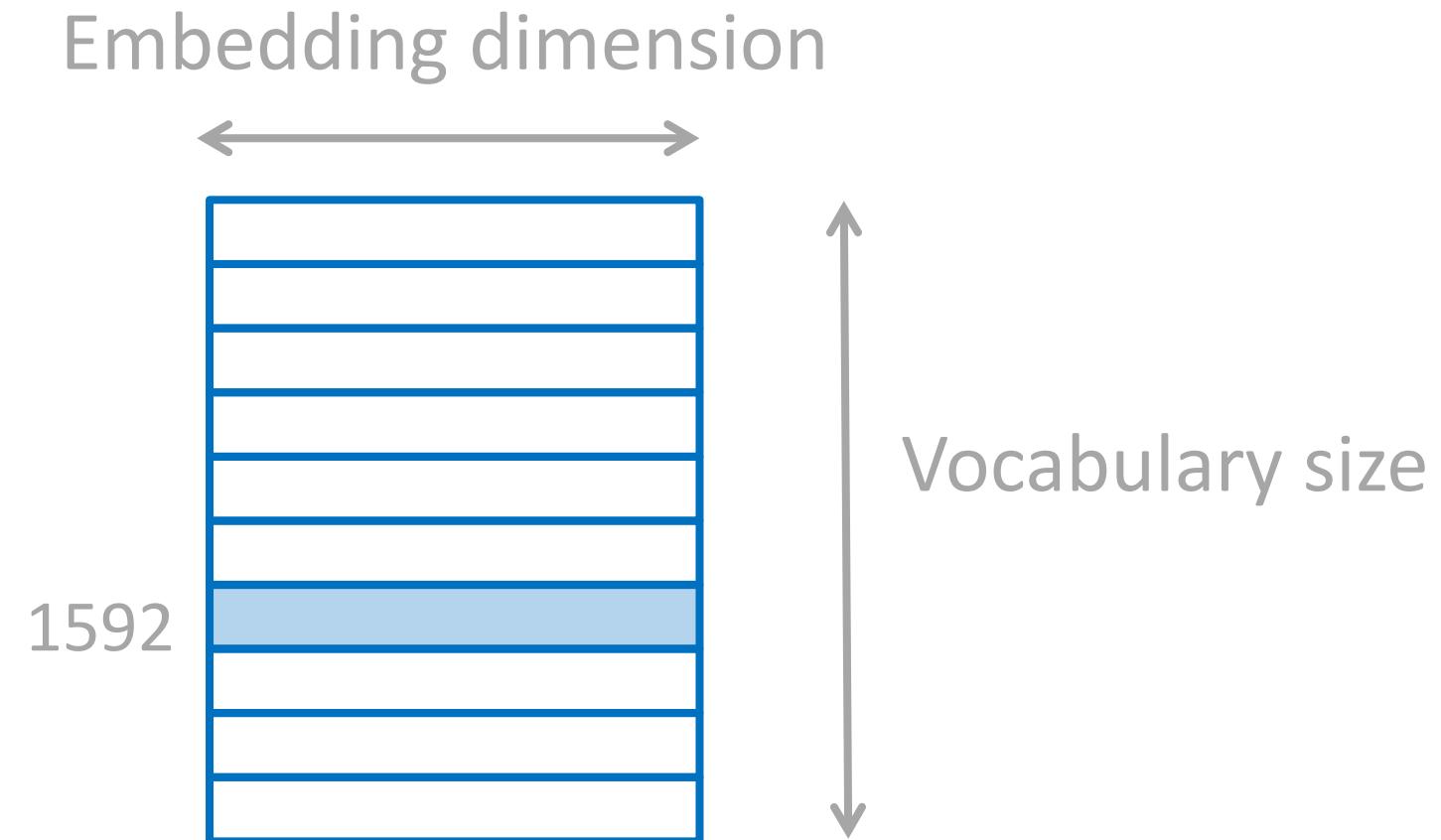


How it works: Look-up Table (Vocabulary)

I **saw** a cat .

39 1592 10 2548 5

Token index in
the vocabulary



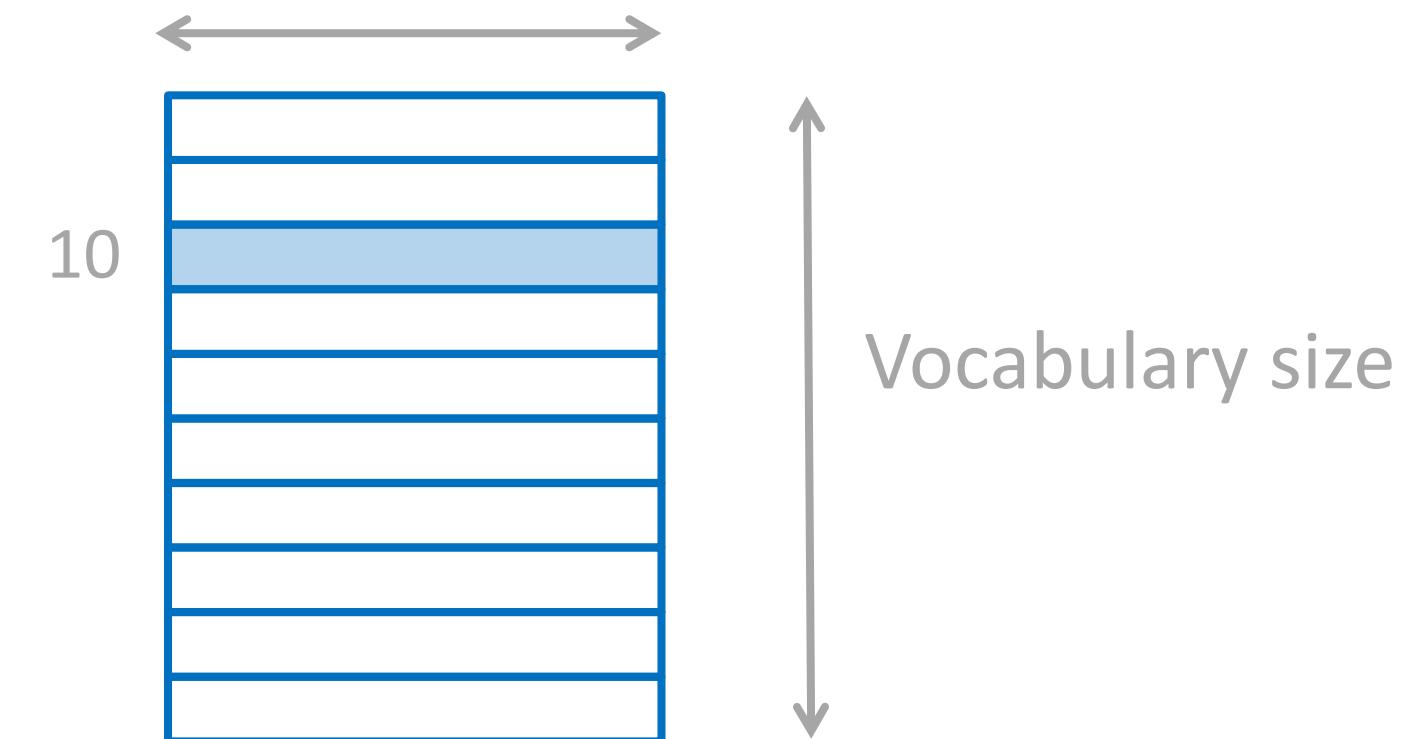
How it works: Look-up Table (Vocabulary)

I saw **a** cat .

39 1592 10 2548 5

Token index in
the vocabulary

Embedding dimension



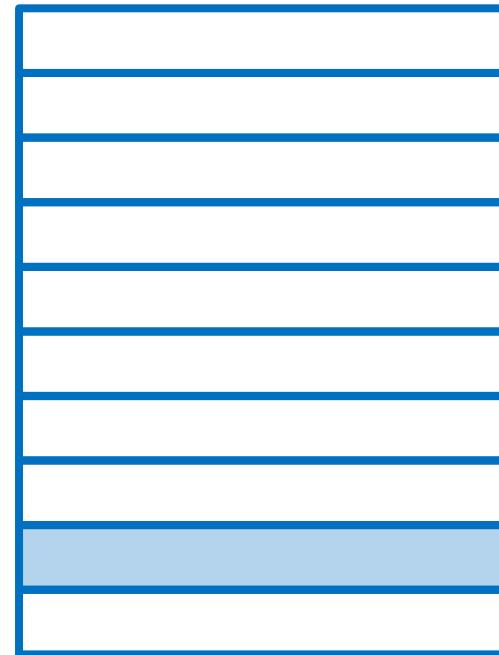
How it works: Look-up Table (Vocabulary)

I saw a **cat** .

39 1592 10 2548 5

Token index in
the vocabulary

Embedding dimension



Vocabulary size

2548

How it works: Look-up Table (Vocabulary)

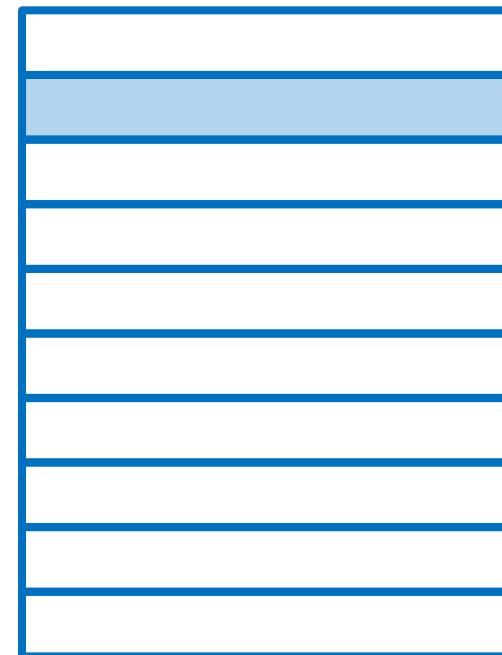
I saw a cat .

39 1592 10 2548 5

Token index in
the vocabulary

Embedding dimension

5



Vocabulary size

Representing words as discrete symbols

In traditional NLP, we regard words as discrete symbols: **hotel, conference, motel**

Means one 1, the rest 0s



Words can be represented by **one-hot** vectors:

motel = [0 0 0 0 0 0 0 0 0 1 0 0 0 0]

hotel = [0 0 0 0 0 0 1 0 0 0 0 0 0]

Vector dimension = number of words in vocabulary (e.g. 500,000)

Problem with words as discrete symbols

Example: in web search, if user searches for “**Seattle motel**”, we would like to match documents containing “**Seattle hotel**”.

But:

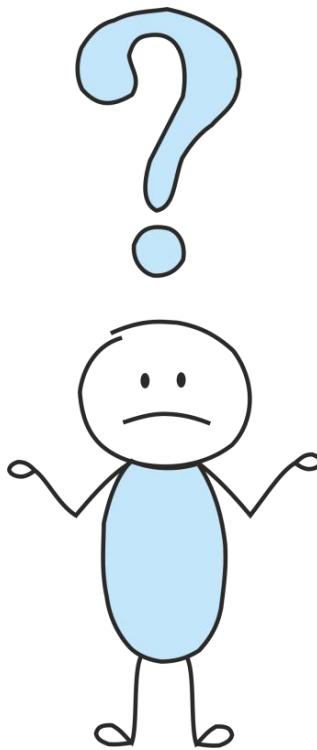
$$\begin{aligned}\text{motel} &= [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0] \\ \text{hotel} &= [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0]\end{aligned}$$

These two vectors are orthogonal.

There is no natural notion of **similarity** for one-hot vectors!

These vectors do not contain information about a **meaning** of a word.

But what is meaning?



But what is meaning?

What is **bardiwac**?



But what is meaning?

What is bardiwac?

- He handed her a glass of **bardiwac**.
- Beef dishes are made to complement the **bardiwac**.
- Nigel staggered to his feet, face flushed from too much **bardiwac**.
- Malbec, one of the lesser-known **bardiwac** grapes, responds well to Australia's sunshine.
- I dined off bread and cheese and this excellent **bardiwac**.
- The drinks were delicious: blood-red **bardiwac** as well as light, sweet Rhenish.

But what is meaning?

What is **bardiwac**?

- He handed her a glass of **bardiwac**.
- Beef dishes are made to complement the **bardiwac**.
- Nigel staggered to his feet, face flushed from too much **bardiwac**.
- Malbec, one of the lesser-known **bardiwac** grapes, responds well to Australia's sunshine.
- I dined off bread and cheese and this excellent **bardiwac**.
- The drinks were delicious: blood-red **bardiwac** as well as light, sweet Rhenish.



Bardiwac is a red
alcoholic beverage
made from grapes

Distributional semantics

- A bottle of bardiwac is on the table.
- Everybody likes bardiwac.
- Don't have bardiwac before you drive.
- We make bardiwac out of corn.

Distributional semantics

- A bottle of _____ is on the table.
- Everybody likes _____.
- Don't have _____ before you drive.
- We make _____ out of corn.

What other words fit into these contexts?

Distributional semantics

- A bottle of _____ is on the table. (1)
- Everybody likes _____. (2)
- Don't have _____ before you drive. (3)
- We make _____ out of corn. (4)

What other words fit into these contexts?

	(1)	(2)	(3)	(4)	...
bardiwac	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	
choices	0	1	0	0	

Distributional semantics

- A bottle of _____ is on the table. (1)
- Everybody likes _____. (2)
- Don't have _____ before you drive. (3)
- We make _____ out of corn. (4)

What other words fit into these contexts?

	(1)	(2)	(3)	(4)	...
bardiwac	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	
choices	0	1	0	0	

Distributional semantics

Does vector similarity imply semantic similarity?

Distributional semantics

Does vector similarity imply semantic similarity?



The **distributional hypothesis**, stated by Firth (1957):

“You shall know a word by the company it keeps.”

Idea: co-occurrence counts

Corpus sentences

He also found five fish swimming in murky water in an old **bathtub**.

We do abhor dust and dirt, and stains on the **bathtub**, and any kind of filth.

Above At the far end of the garden room a **bathtub** has been planted with herbs for the winter.

They had been drinking Cisco, a fruity, wine-based fluid that smells and tastes like a mixture of cough syrup and **bathtub** gin.

Science finds that a surface tension on the water can draw the boats together, like toy boats in a **bathtub**.

In fact, the godfather of gloom comes up with a plot that takes in Windsor Davies (the ghost of sitcoms past), a **bathtub** and a big box of concentrated jelly.

‘I’ll tell him,’ said the Dean from the bathroom above the sound of bathwater falling from a great height into the ample Edwardian **bathtub**.

Idea: co-occurrence counts

Corpus sentences

He also found five fish swimming in murky water in an old **bathtub**.

We do abhor dust and dirt, and stains on the **bathtub**, and any kind of filth.

Above At the far end of the garden room a **bathtub** has been planted with herbs for the winter.

They had been drinking Cisco, a fruity, wine-based fluid that smells and tastes like a mixture of cough syrup and **bathtub** gin.

Science finds that a surface tension on the water can draw the boats together, like toy boats in a **bathtub**.

In fact, the godfather of gloom comes up with a plot that takes in Windsor Davies (the ghost of sitcoms past), a **bathtub** and a big box of concentrated jelly.

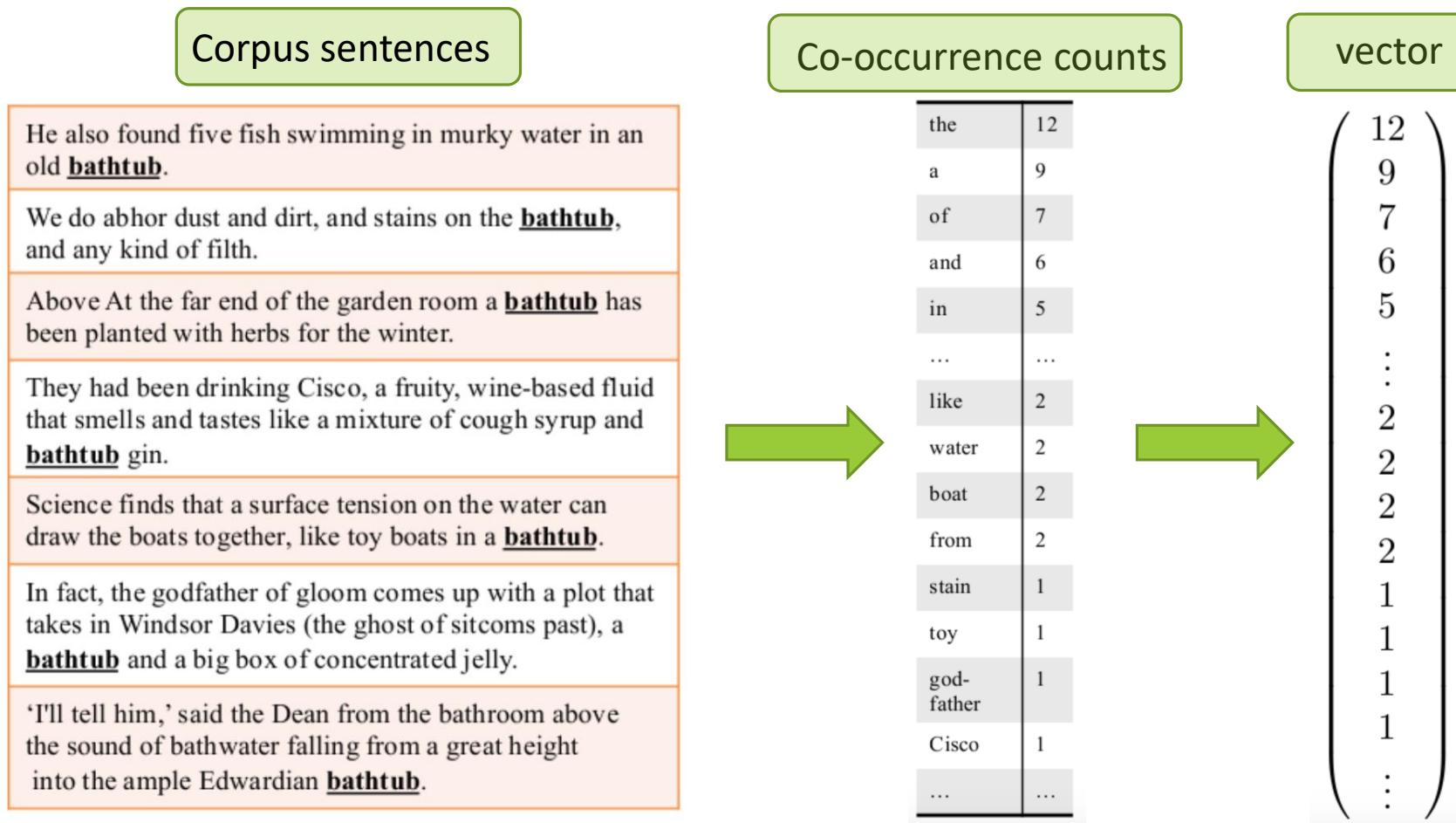
'I'll tell him,' said the Dean from the bathroom above the sound of bathwater falling from a great height into the ample Edwardian **bathtub**.

Co-occurrence counts

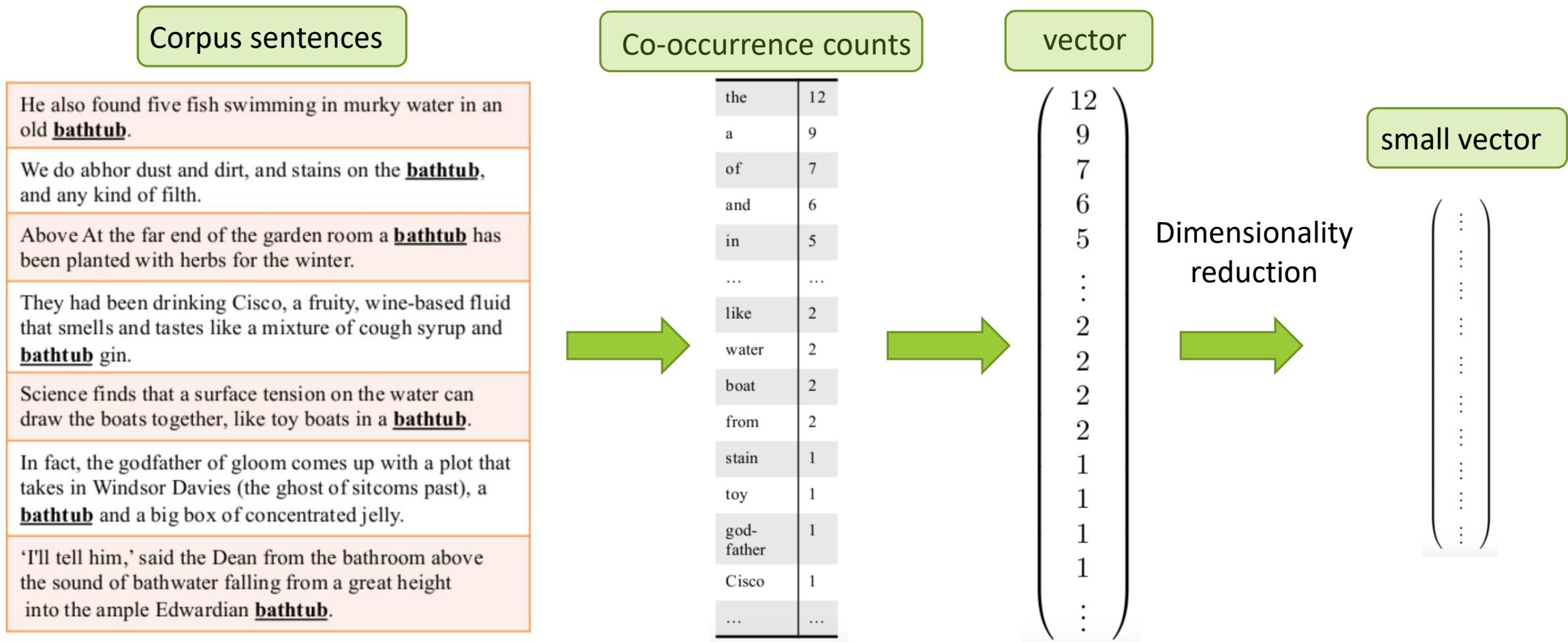


the	12
a	9
of	7
and	6
in	5
...	...
like	2
water	2
boat	2
from	2
stain	1
toy	1
god-father	1
Cisco	1
...	...

Idea: co-occurrence counts



Idea: co-occurrence counts



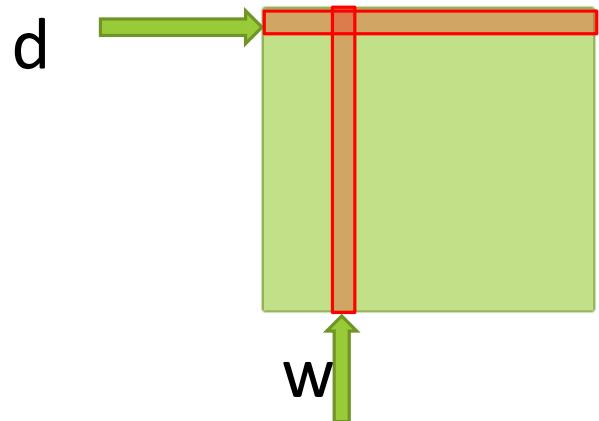
Latent semantic analysis (LSA)

X - document-term co-occurrence matrix

$$X \approx \hat{X} = U \Sigma V^T$$

- co-occurrence counts
- tf-idf
- filter stop-words
- lemmatize
- ...

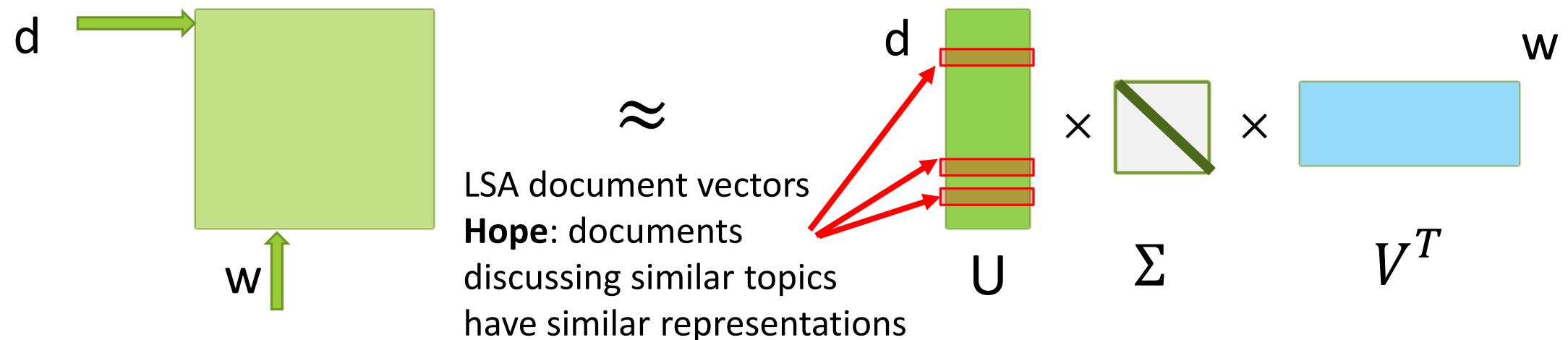
$$\begin{matrix} U & \Sigma & V^T \\ \text{d} & \times & \text{w} \end{matrix}$$



Latent semantic analysis (LSA)

X - document-term co-occurrence matrix

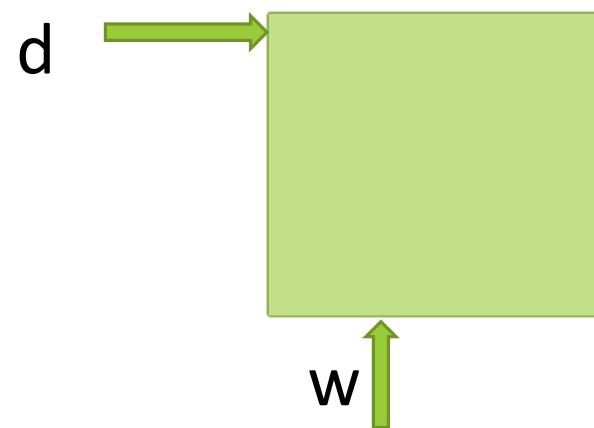
$$X \approx \hat{X} = U \Sigma V^T$$



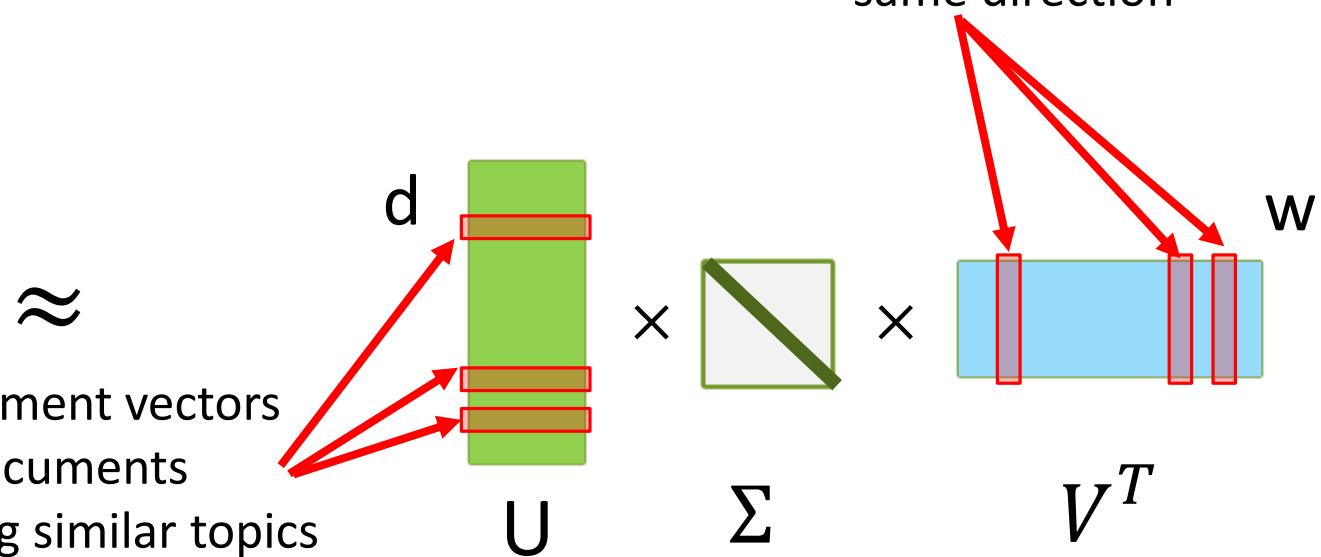
Latent semantic analysis (LSA)

X - document-term co-occurrence matrix

$$X \approx \hat{X} = U \Sigma V^T$$



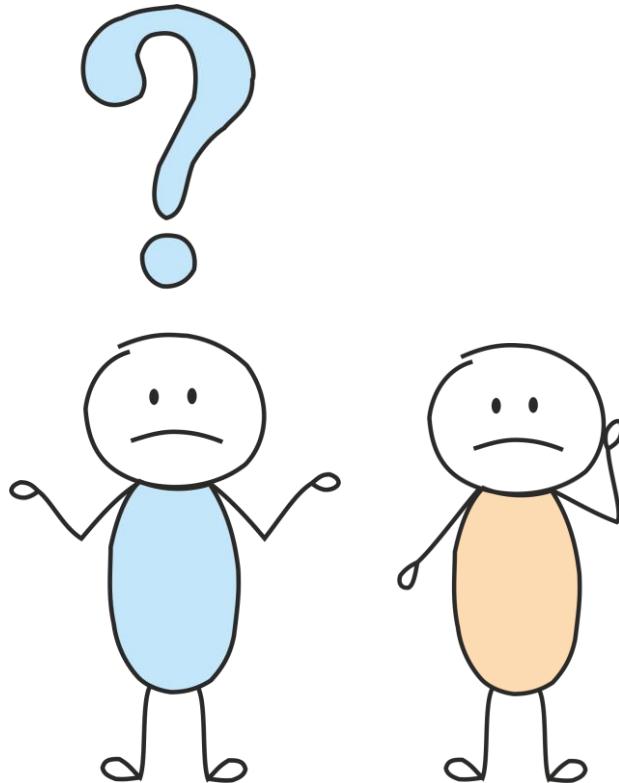
\approx
LSA document vectors
Hope: documents
discussing similar topics
have similar representations



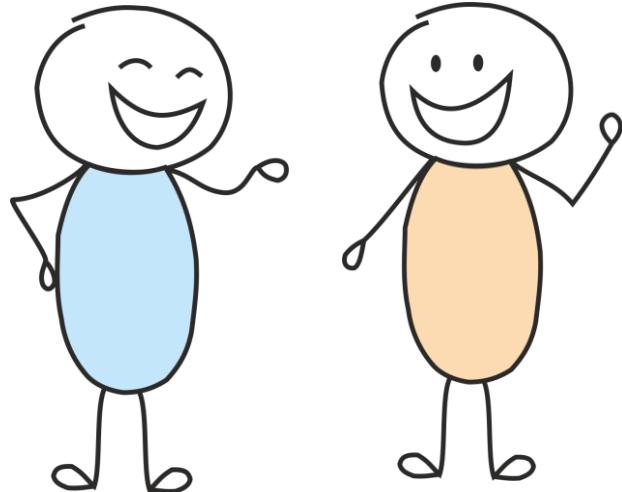
Count-based methods

However, this is not the only way to induce
distributional representations
(and not the best one)

Why not to learn distributed representations?



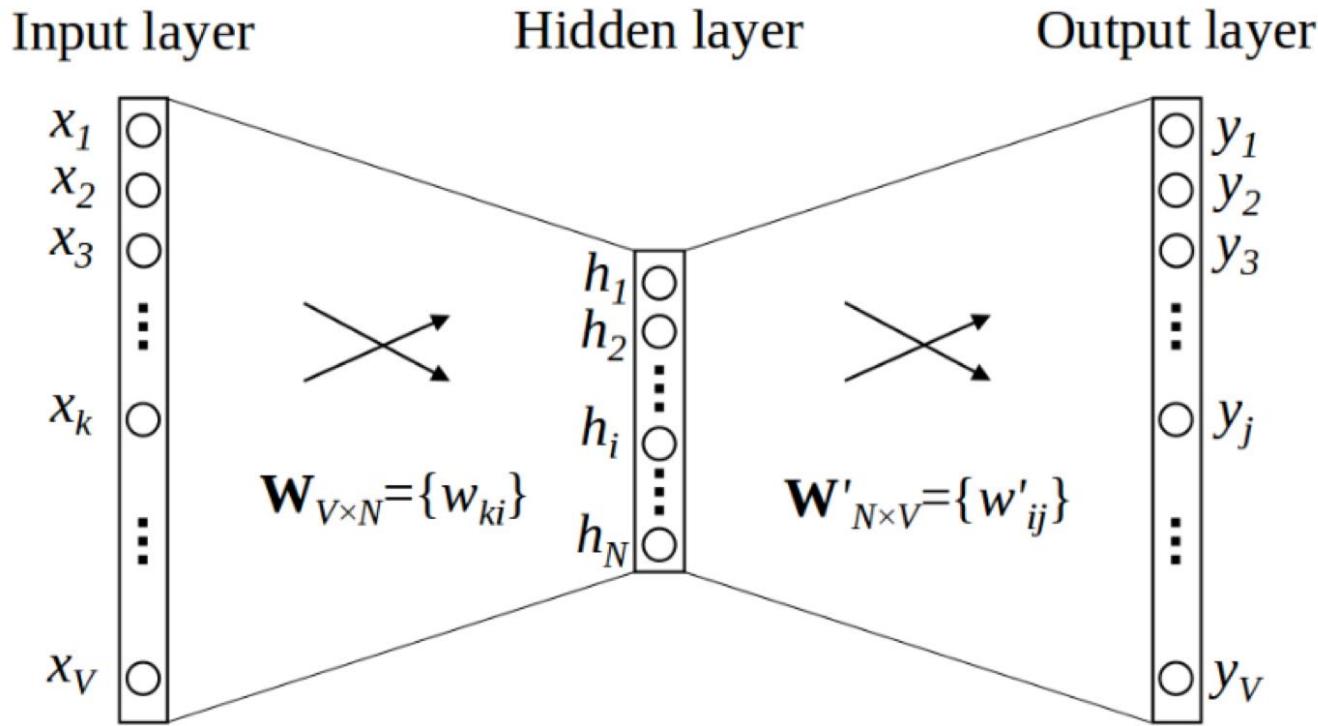
I mean, really, why?



We will learn a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts.

This **word vectors** are called **word embeddings** or word representations

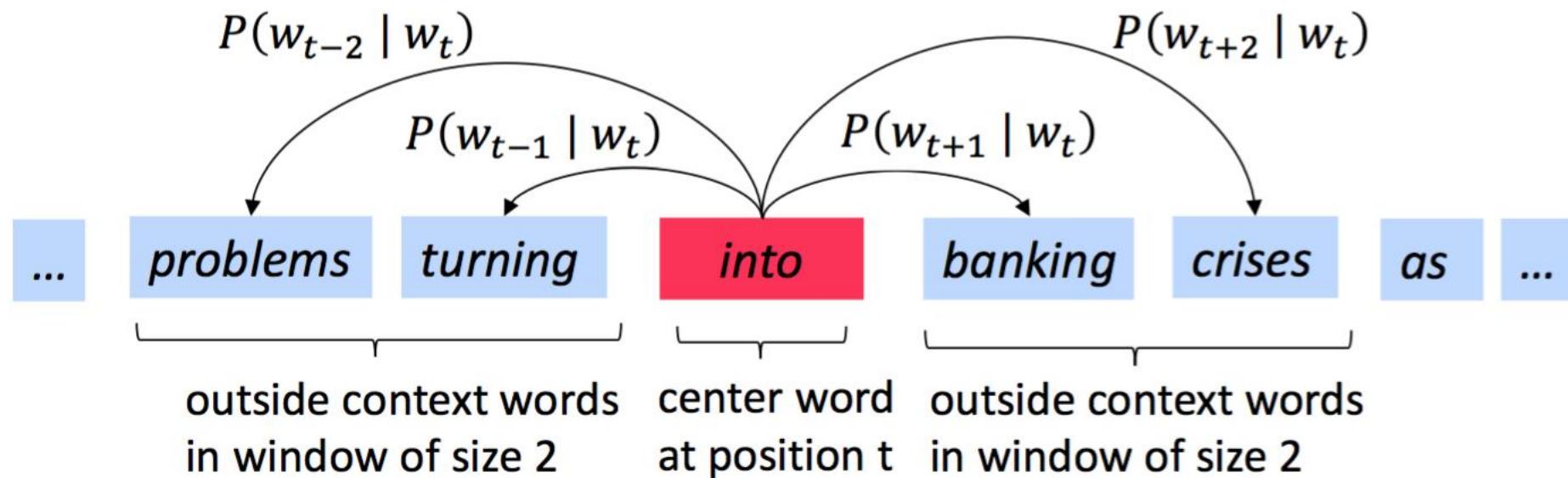
Word2Vec



- a large corpus of text
- Every word in a fixed vocabulary is represented by a vector
- Go through each position t in the text, which has a center word c and context (“outside”) words o
- Use the similarity of the word vectors for c and o to calculate the probability of o given c (or vice versa)
- Keep adjusting the word vectors to maximize this probability

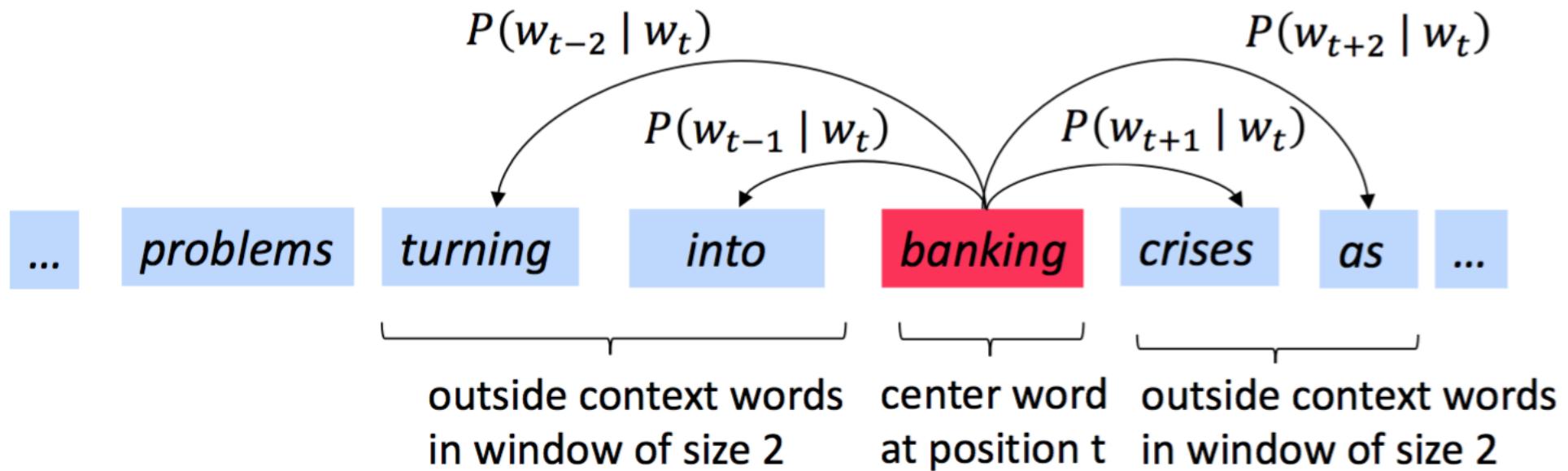
Word2Vec

- Examples windows and process for computing $P(w_{t+j} | w_j)$



Word2Vec

- Examples windows and process for computing $P(w_{t+j} | w_j)$



Word2Vec: objective function

For each position $t = 1, \dots, T$, predict context words within a window of fixed size m , given center word w_t .

$$L(\theta) = \prod_{t=1}^T \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j} | w_t; \theta)$$

Word2Vec: objective function

For each position $t = 1, \dots, T$, predict context words within a window of fixed size m , given center word w_t .

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

θ is all variables
to be optimized

Word2Vec: objective function

The **objective function (or loss, or cost function)** $J(\theta)$ is the (average) negative log likelihood

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j} | w_t; \theta)$$

Word2Vec: objective function

The **objective function (or loss, or cost function)** $J(\theta)$ is the (average) negative log likelihood

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j} | w_t; \theta)$$

Minimizing objective function  Maximizing predictive accuracy

Word2Vec: objective function

- We want to minimize objective function

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j} | w_t; \theta)$$

Word2Vec: objective function

- We want to minimize objective function

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j} | w_t; \theta)$$

- Question: **How to calculate $P(w_{t+j}|w_j, \theta)$?**

Word2Vec: objective function

- We want to minimize objective function

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j} | w_t; \theta)$$

- Question: **How to calculate $P(w_{t+j}|w_j, \theta)$?**

- Answer: We will use two vectors per word w
 - v_w is a center word
 - u_w is a context word

Word2Vec: objective function

- We want to minimize objective function

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j} | w_t; \theta)$$

- Question: How to calculate $P(w_{t+j} | w_j, \theta)$?

- Answer: We will use two vectors per word w

- v_w is a center word
- u_w is a context word

- Then for a center word c and a context word o :

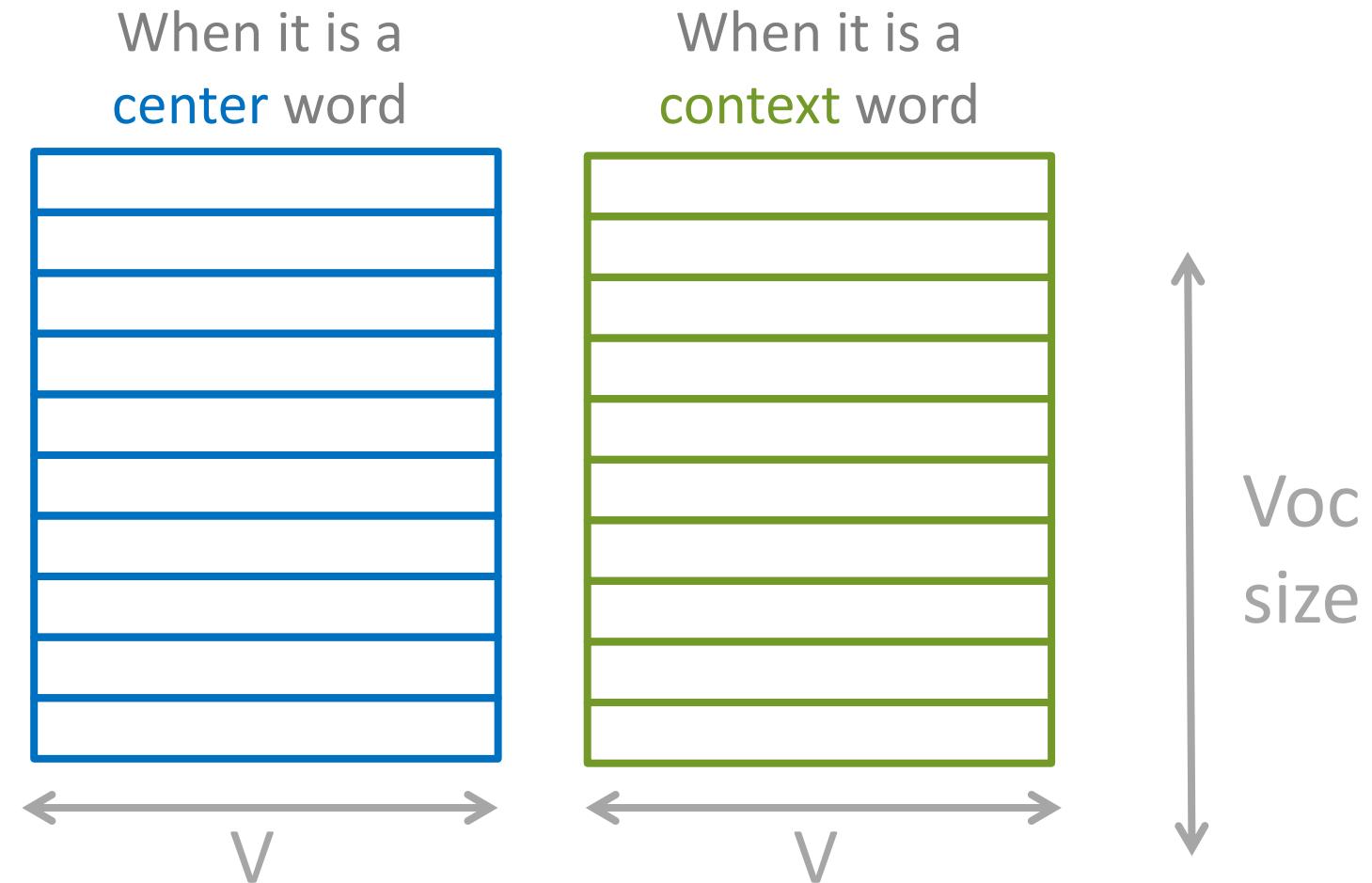
$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Two vectors for each word

I saw a cat .

39 1592 10 2548 5

Token index in
the vocabulary



Two vectors for each word

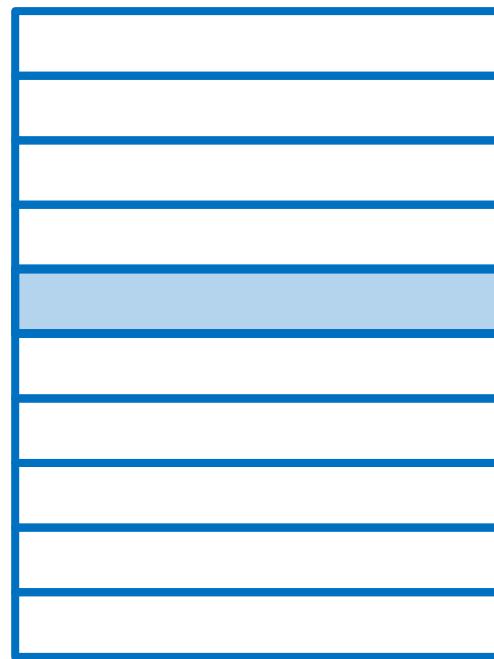
I saw a cat .

39 1592 10 2548 5

Token index in
the vocabulary

39

When it is a
center word



When it is a
context word



Two vectors for each word

I **saw** a cat .

39 1592 10 2548 5

Token index in
the vocabulary

1592

When it is a
center word



When it is a
context word



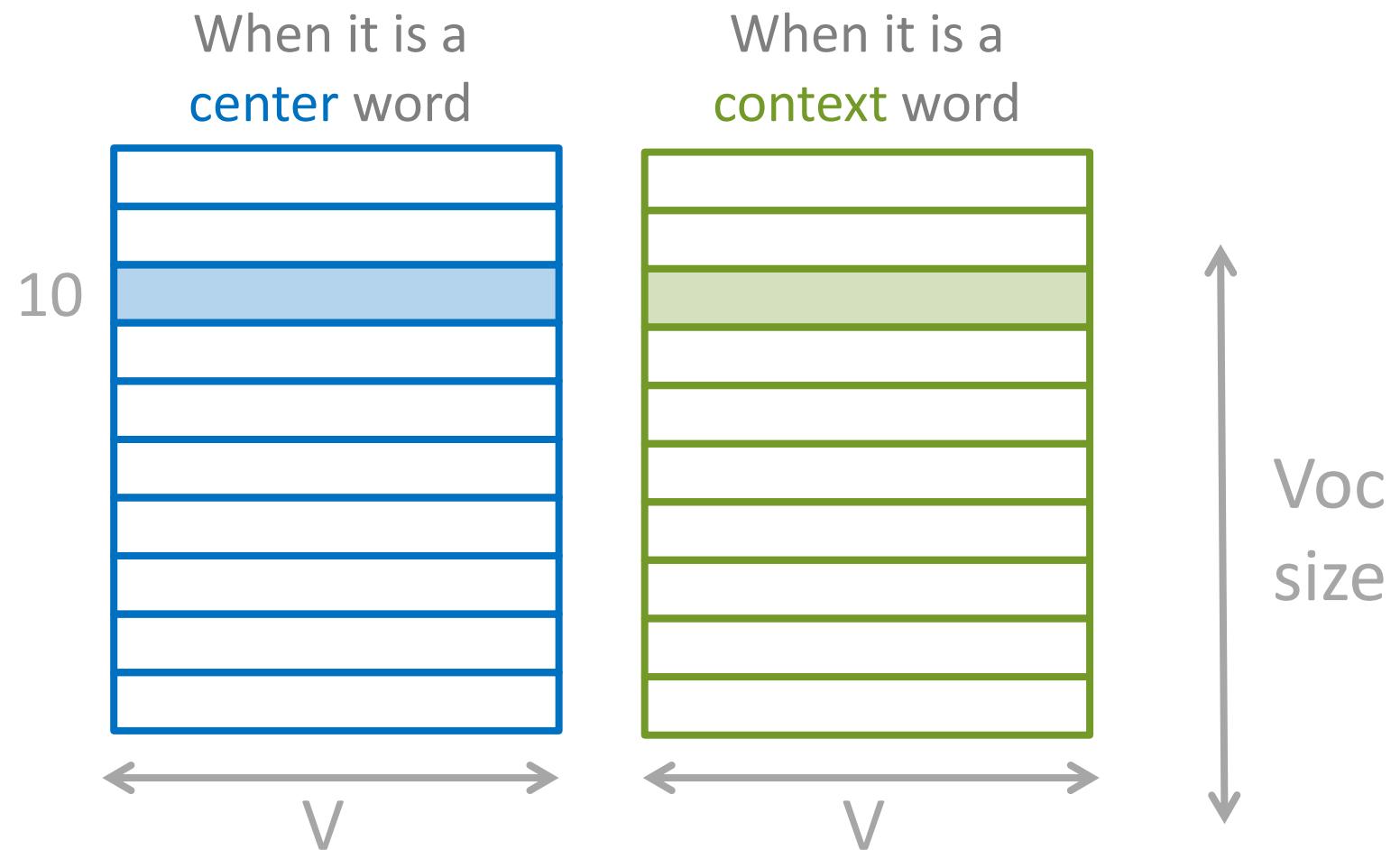
Voc
size

Two vectors for each word

I saw **a** cat .

39 1592 10 2548 5

Token index in
the vocabulary

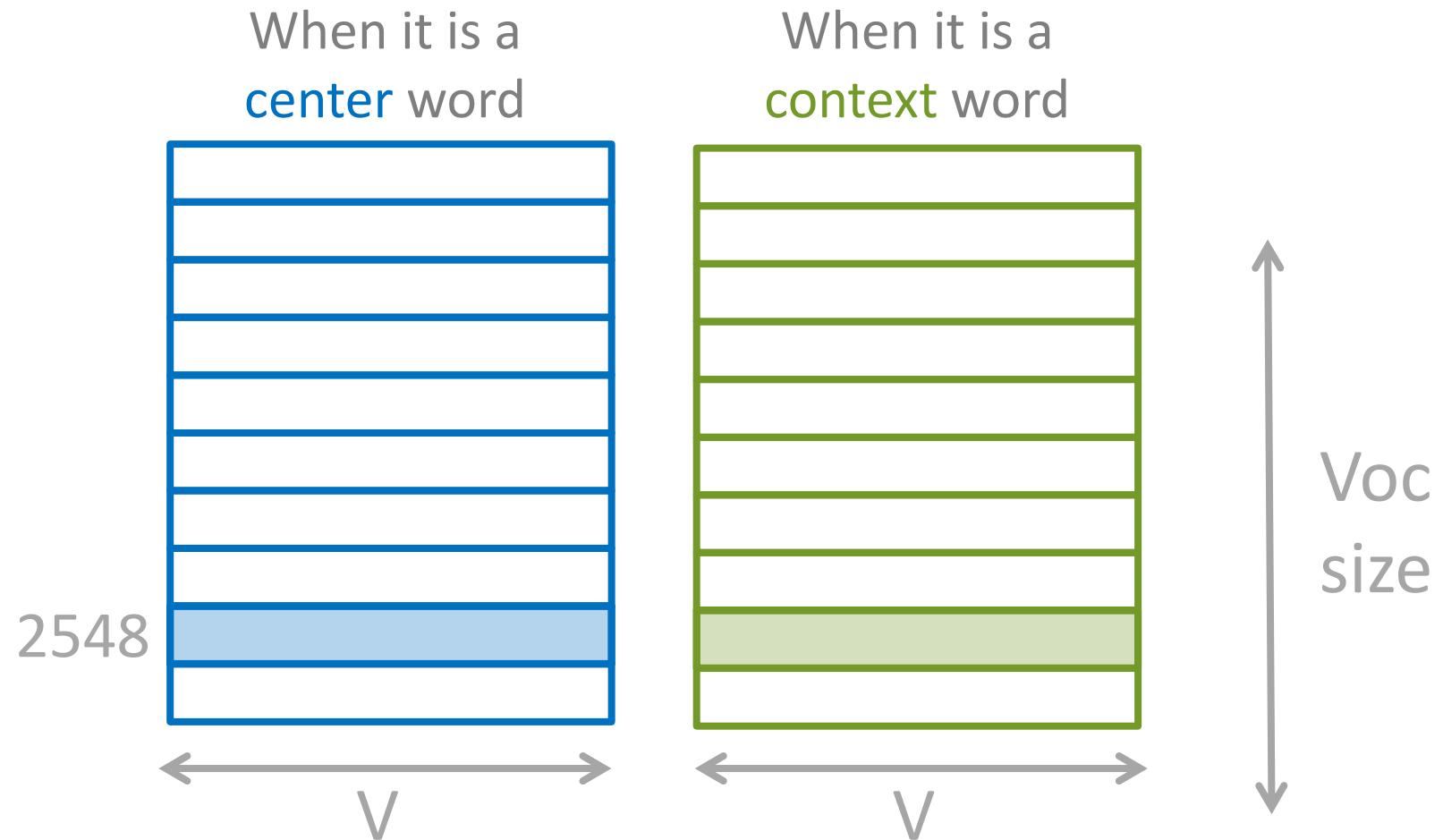


Two vectors for each word

I saw a **cat**.

39 1592 10 2548 5

Token index in
the vocabulary

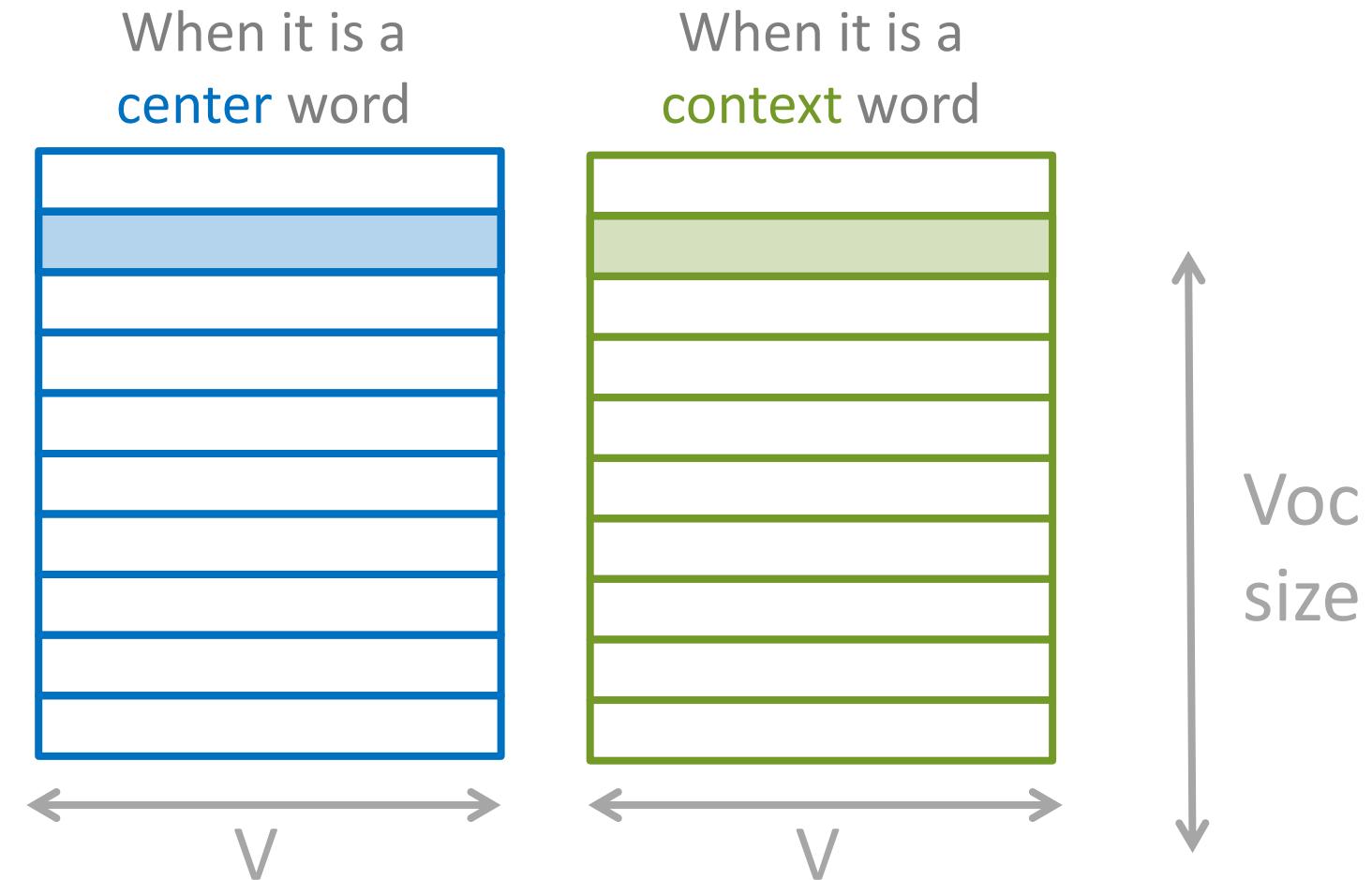


Two vectors for each word

I saw a cat .

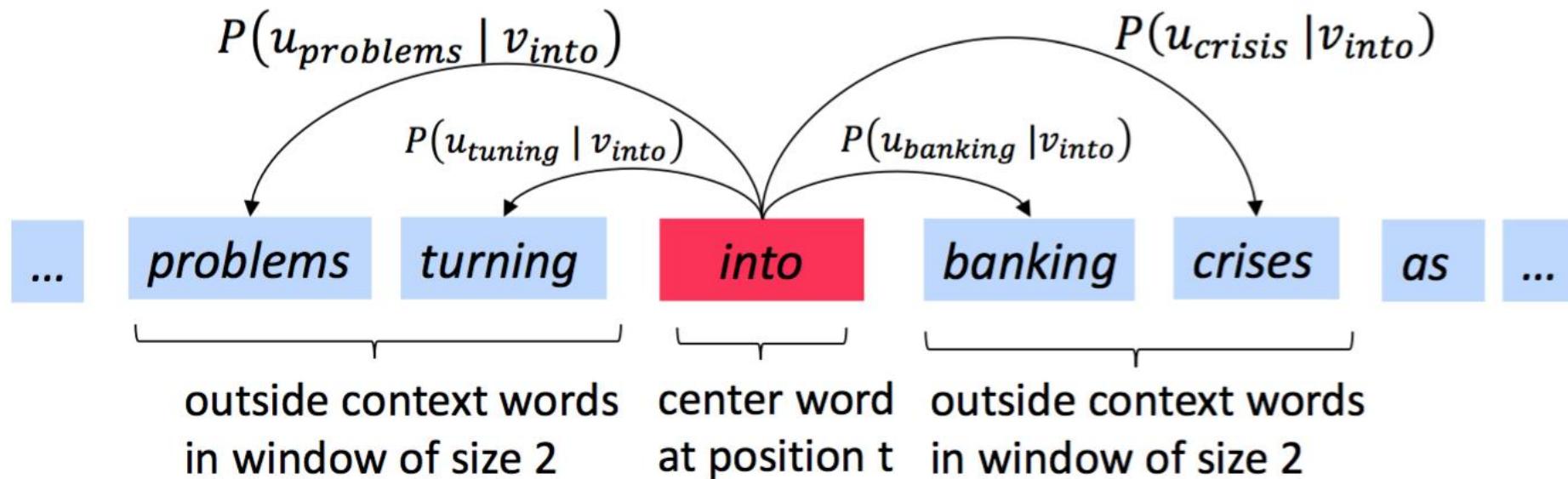
39 1592 10 2548 5

Token index in
the vocabulary



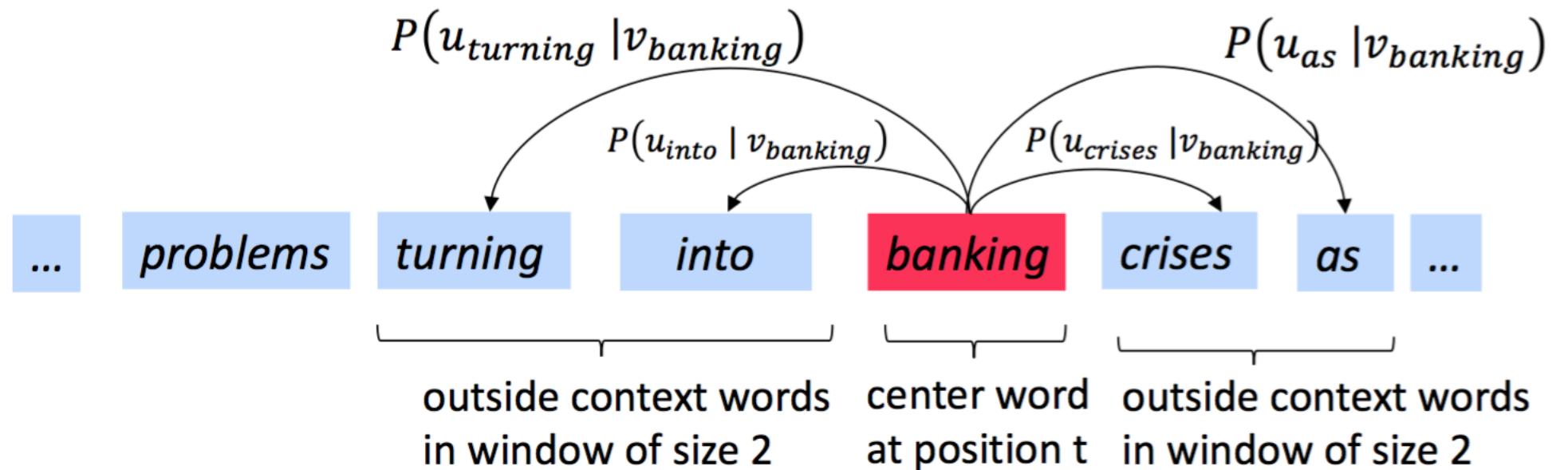
Word2Vec

- Examples windows and process for computing $P(w_{t+j} | w_j)$
- $P(u_{problems} | v_{into})$ is short for $P(problems|into; u_{problems}, v_{into}, \theta)$

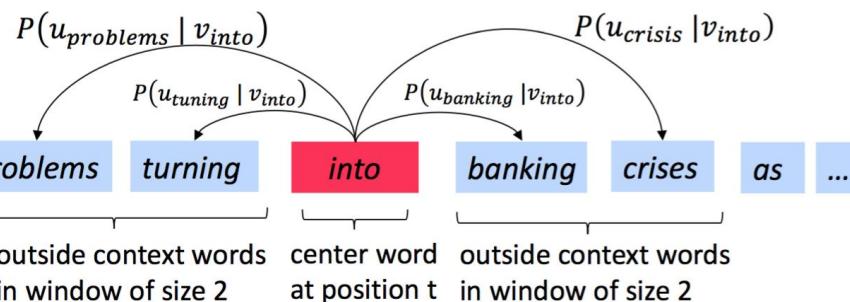


Word2Vec

- Examples windows and process for computing $P(w_{t+j}|w_j)$
- $P(u_{problems}|v_{into})$ is short for $P(problems|into; u_{problems}, v_{into}, \theta)$



Two vectors for each word

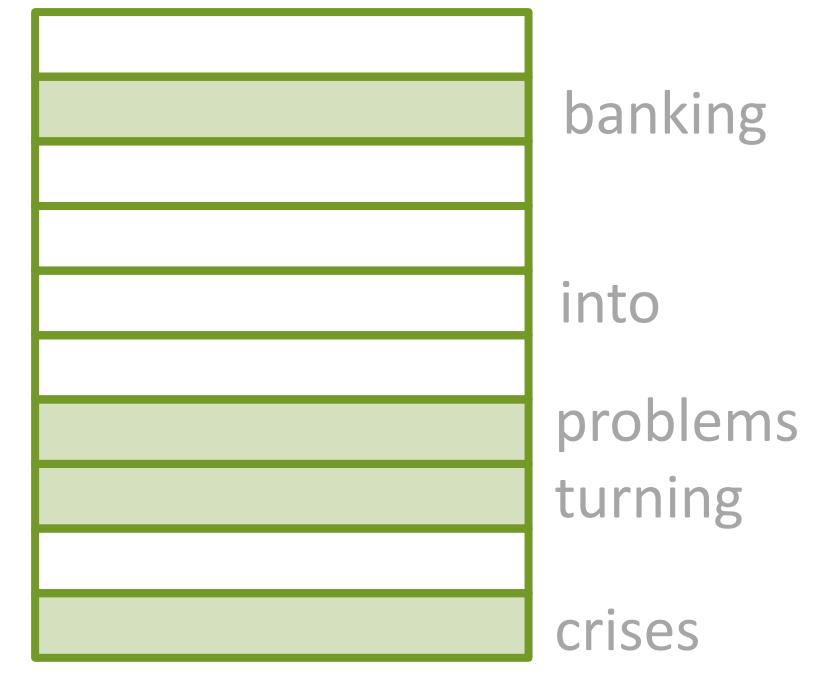


When it is a
center word

into
problems
turning
crises



When it is a
context word



banking

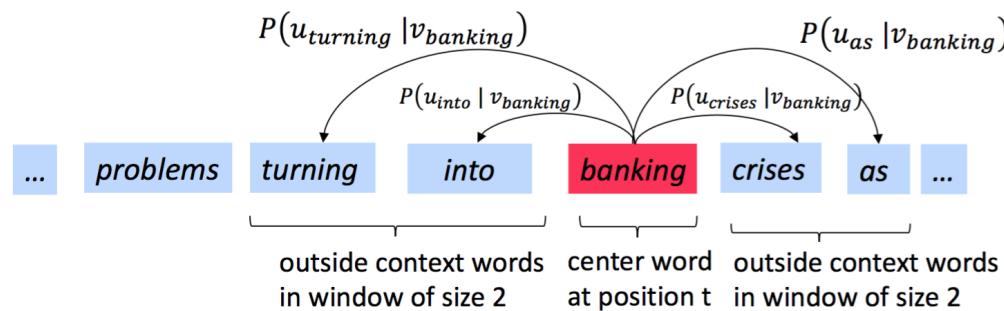
into

problems

turning

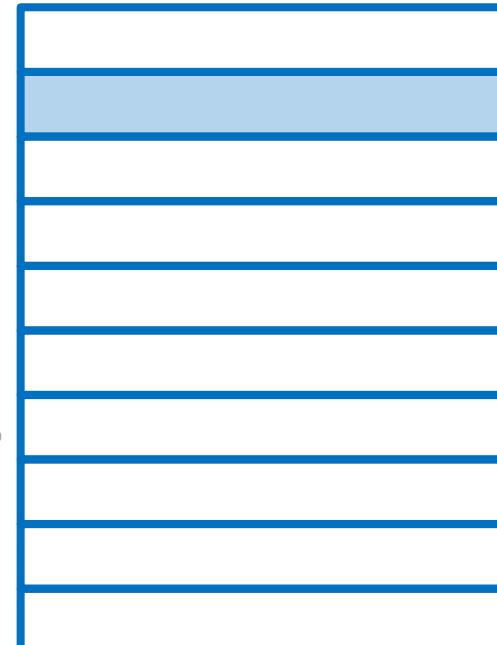
crises

Two vectors for each word



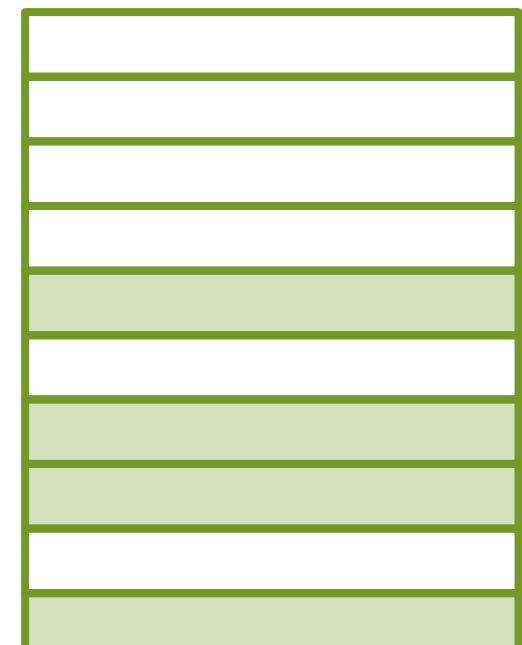
When it is a
center word

into
problems
turning
crises



↔ V ↔

When it is a
context word



↔ V ↔

banking
into
problems
turning
crises

Word2Vec: prediction function

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

Word2Vec: prediction function

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

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



Word2Vec: prediction function

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

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

After taking exponent, normalize over entire vocabulary

This is softmax!

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

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

- 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!**

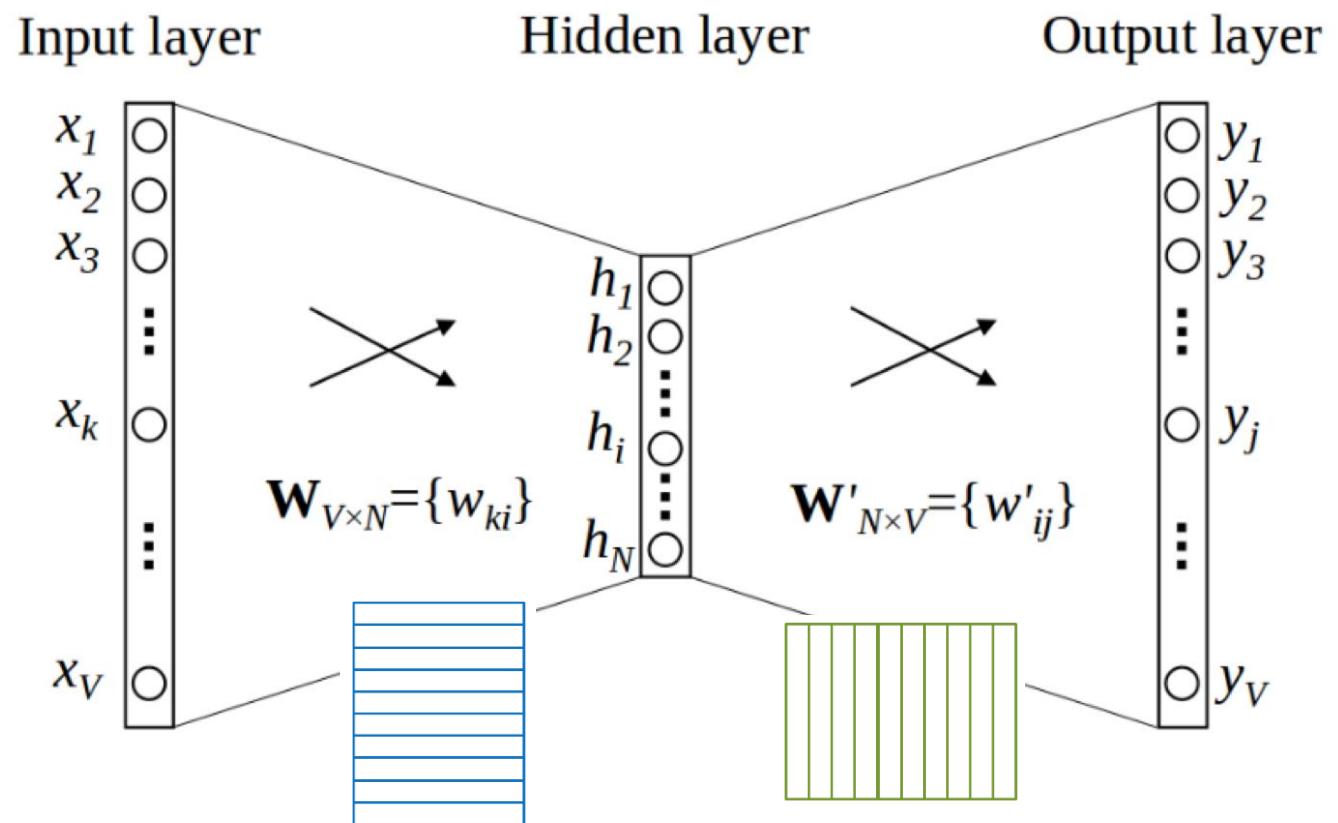
Where is θ ?

$$\theta = \begin{matrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{matrix} \in \mathbb{R}^{2dV}$$

- θ - d-dimensional vectors for V words
- every word has two vectors!
- we optimize these parameters

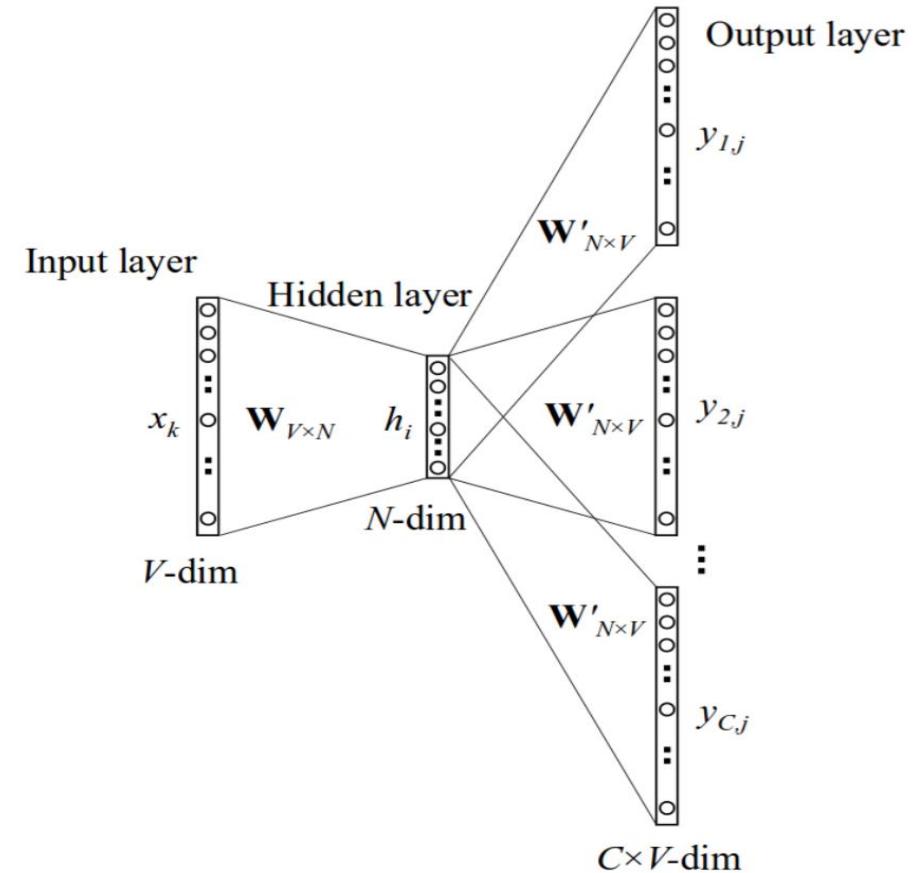
Where is θ ?

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$



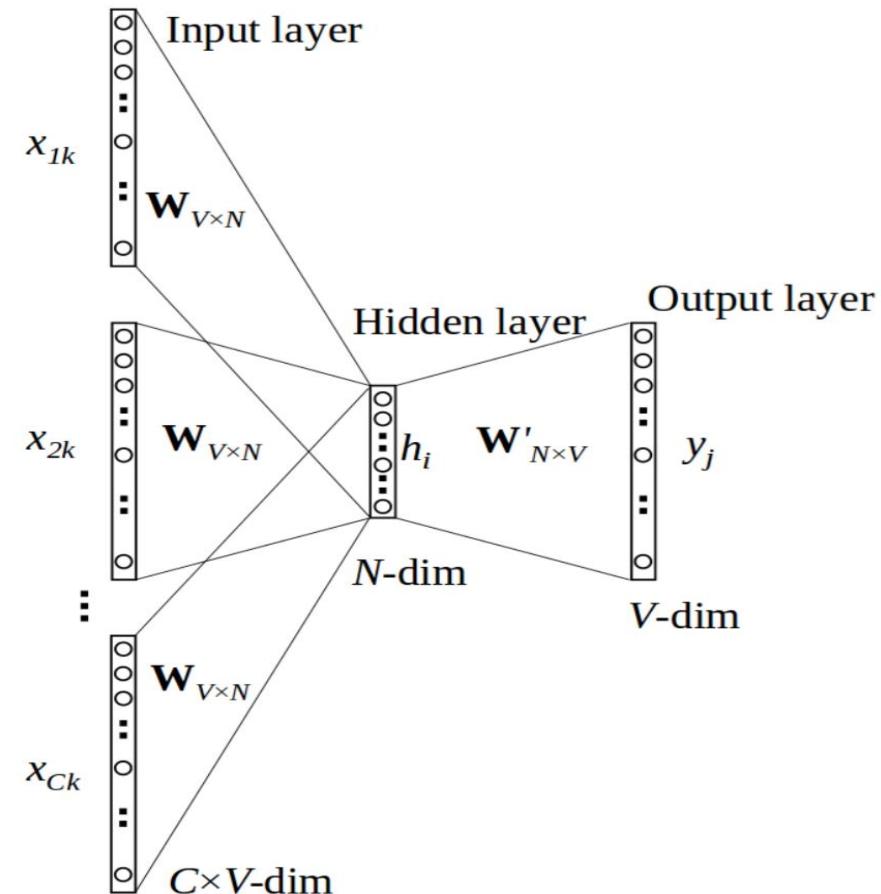
Word2Vec: Skip-gram (SG)

- Predict context ("outside") words (position independent) given center word



Word2Vec: Continuous Bag of Words (CBOW)

- Predict center word from (bag of) context words



Word2Vec: Additional efficiency in training

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

Huge sum! Time for calculating gradients is proportional to $|V|$

Word2Vec: Additional efficiency in training

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

Huge sum! Time for calculating gradients is proportional to $|V|$

Possible solutions:

- Hierarchical softmax
- Negative sampling

Word2Vec: Additional efficiency in training

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

Huge sum! Time for calculating gradients is proportional to $|V|$

Possible solutions:

- Hierarchical softmax
- Negative sampling

$$\sum_{w \in V} \exp(u_w^T v_c) \xrightarrow{\text{green arrow}} \sum_{w \in \{o\} \cup S_k} \exp(u_w^T v_c)$$

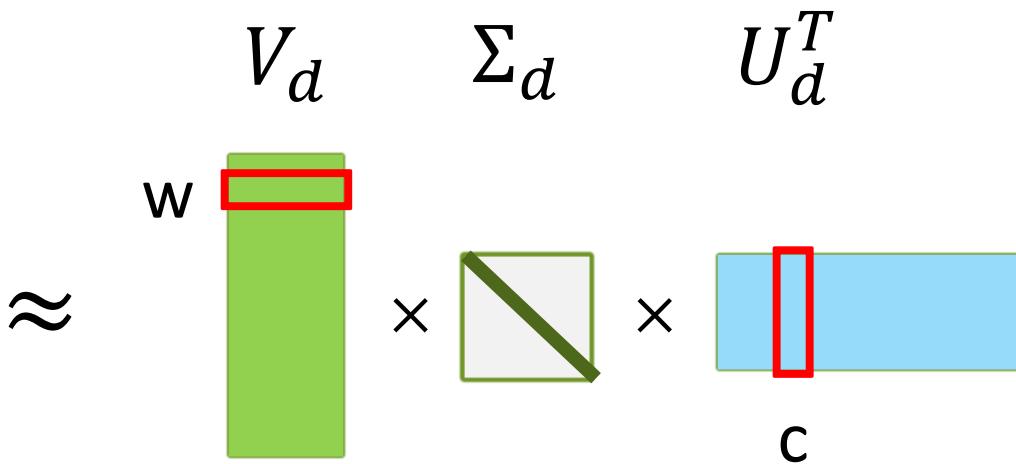
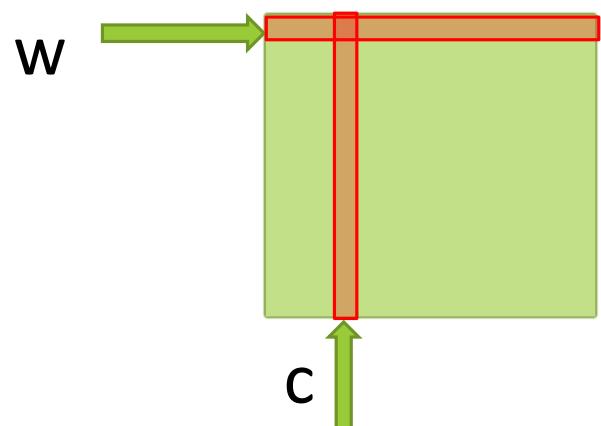
Sum over a small subset: *negative sample*, $|S_k|=k$

Mikolov et al, 2013, <https://arxiv.org/pdf/1310.4546.pdf>

Word2Vec: (Near) equivalence to matrix factorization

$$PMI(w, c) = \log \frac{N(w, c) \times |V|}{N(w)N(c)}$$

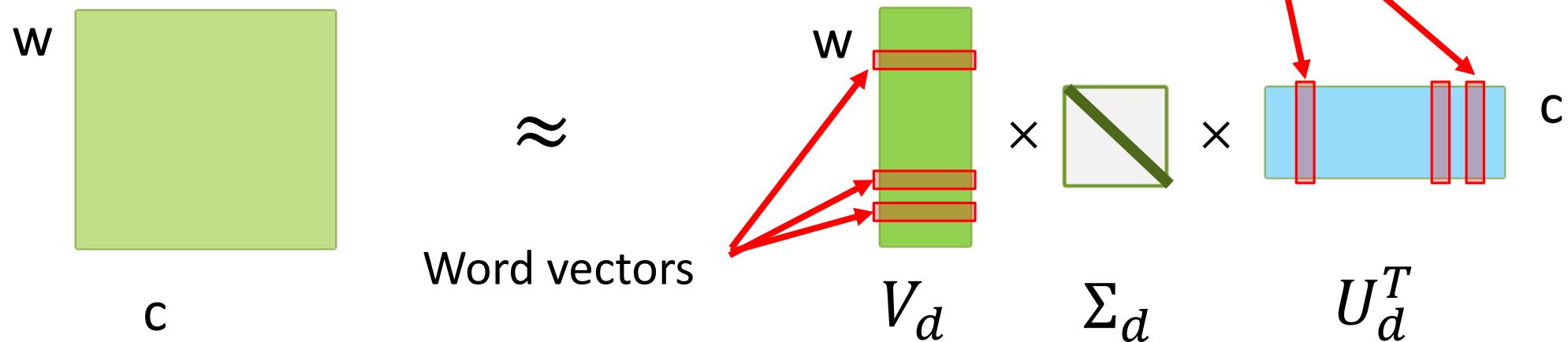
$$PMI = X \approx \hat{X} = V_d \Sigma_d U_d^T$$



Word2Vec: (Near) equivalence to matrix factorization

$$PMI(w, c) = \log \frac{N(w, c) \times |V|}{N(w)N(c)}$$

$$PMI = X \approx \hat{X} = V_d \Sigma_d U_d^T$$

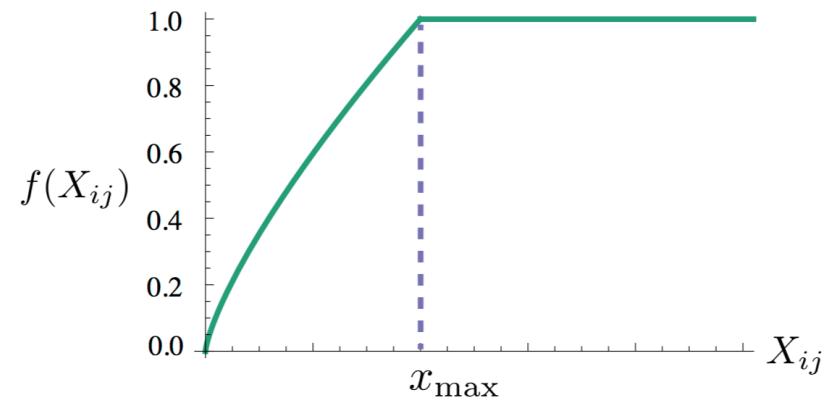


Levy et al, TACL 2015 <http://www.aclweb.org/anthology/Q15-1016>

GloVe: combine count-based and direct prediction methods

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^W f(P_{ij})(u_i^T v_j - \log P_{ij})^2$$

$$X_{final} = U + V$$

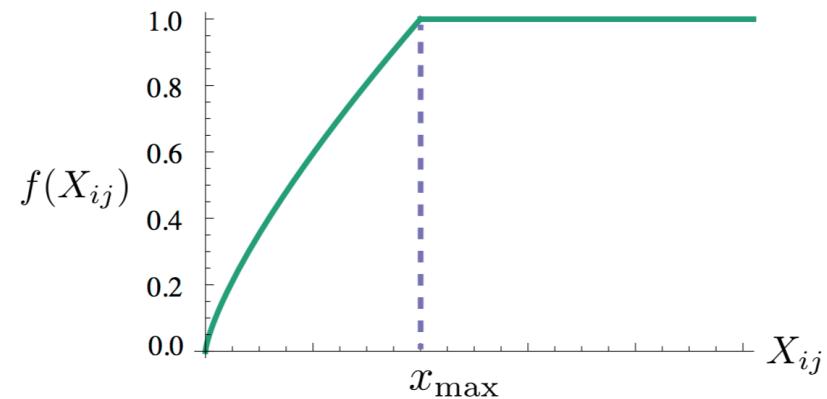


GloVe: combine count-based and direct prediction methods

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^W f(P_{ij})(u_i^T v_j - \log P_{ij})^2$$

$$X_{final} = U + V$$

probability that word j
appears in the context
of word i

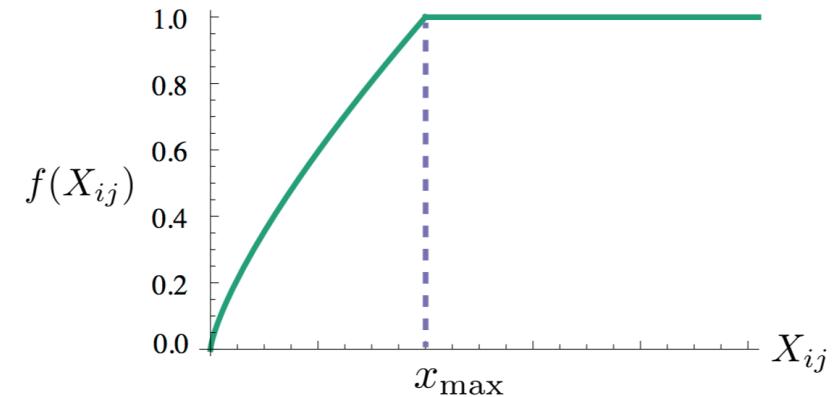


GloVe: combine count-based and direct prediction methods

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^W f(P_{ij}) (u_i^T v_j - \log P_{ij})^2$$

$$X_{final} = U + V$$

the idea is close to
factorizing the log of the co-
occurrence matrix (closely
related to LSA)

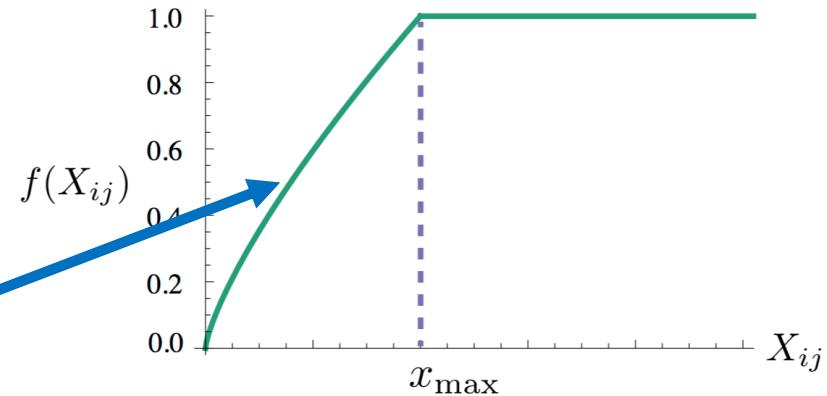


GloVe: combine count-based and direct prediction methods

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^W f(P_{ij})(u_i^T v_j - \log P_{ij})^2$$

$$X_{final} = U + V$$

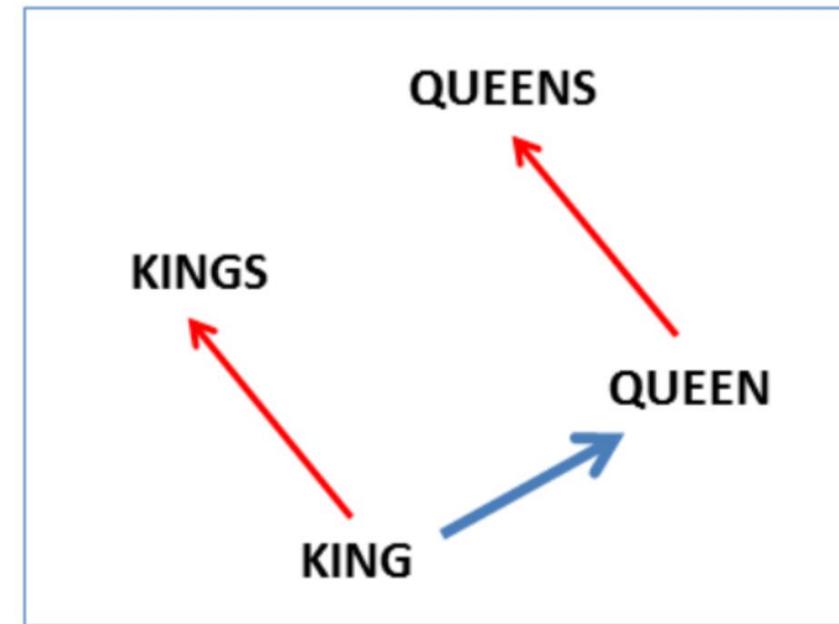
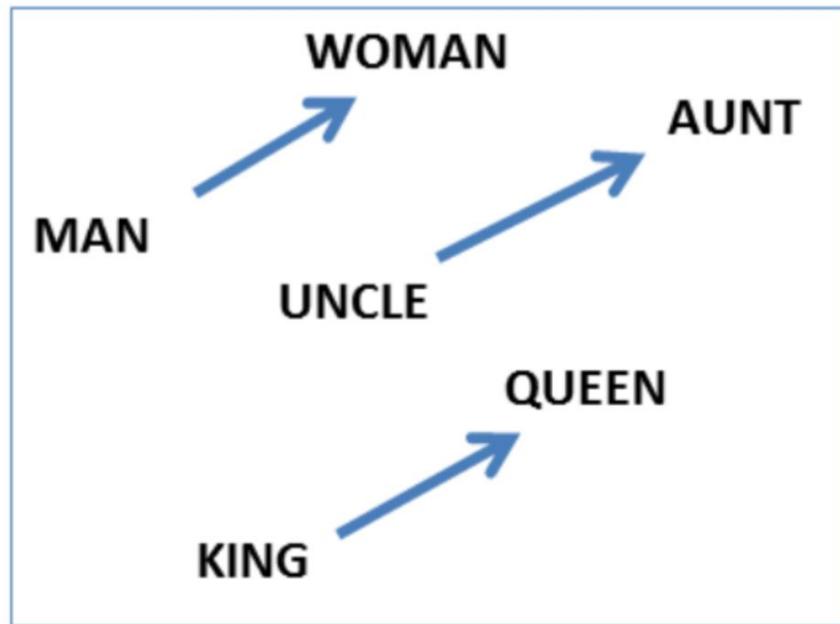
discard rare noisy
co-occurrences



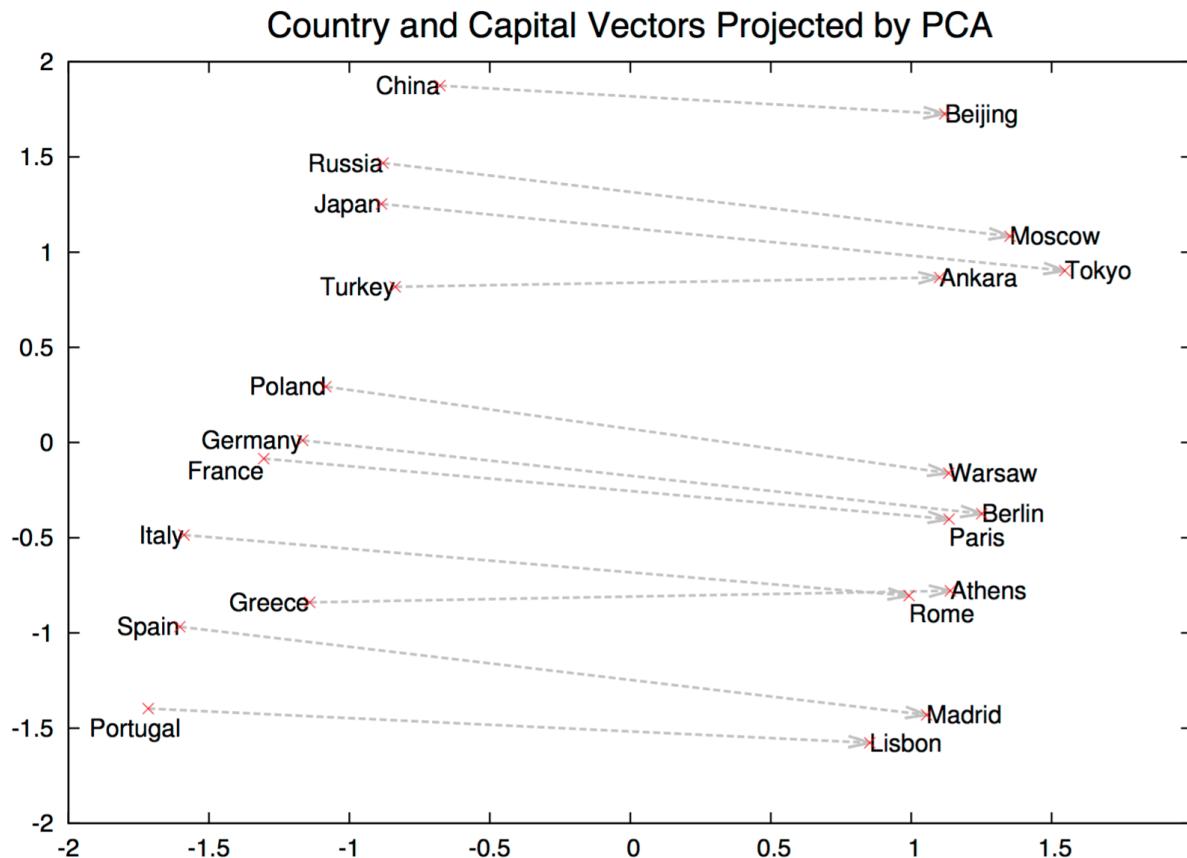
And now the fun starts!

Word2Vec: what are relations between vectors?

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

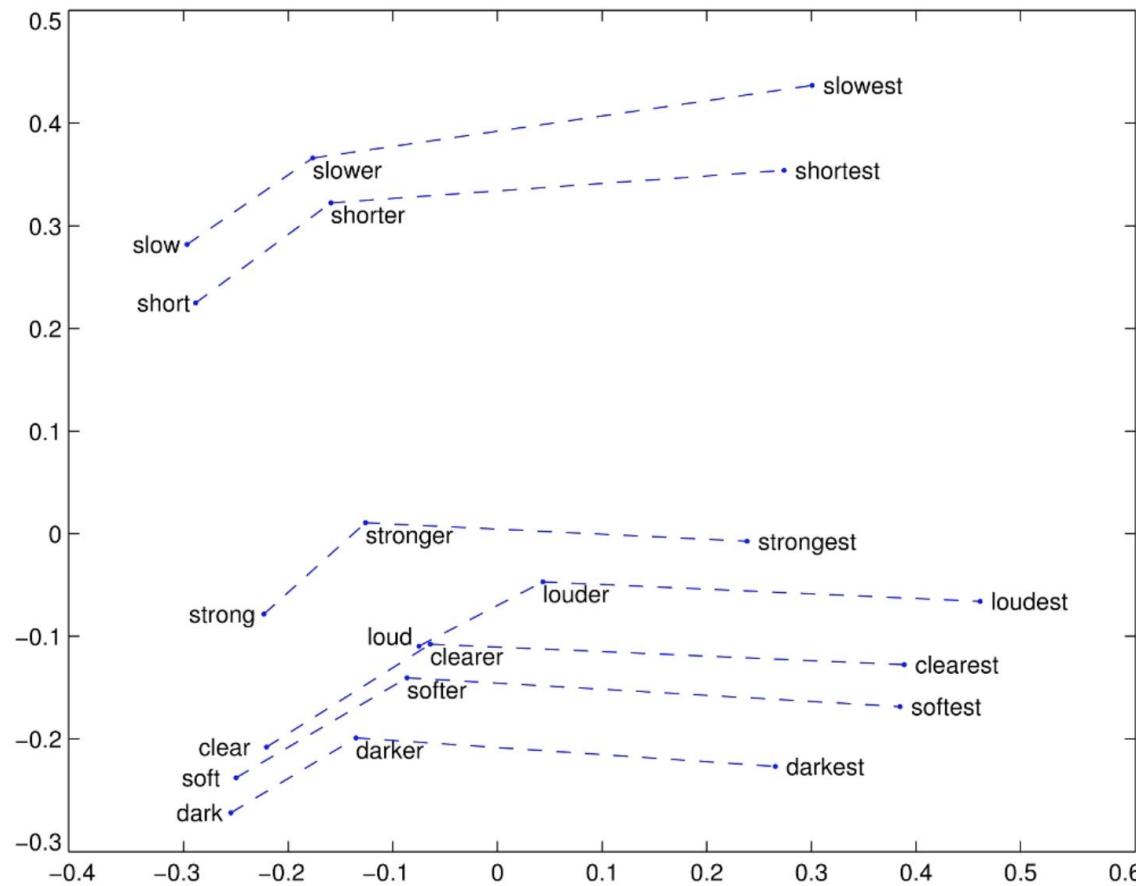


What are relations between vectors?



Mikolov et al, 2013, <https://arxiv.org/pdf/1310.4546.pdf>

What are relations between vectors?



(<http://nlp.stanford.edu/projects/glove/>)

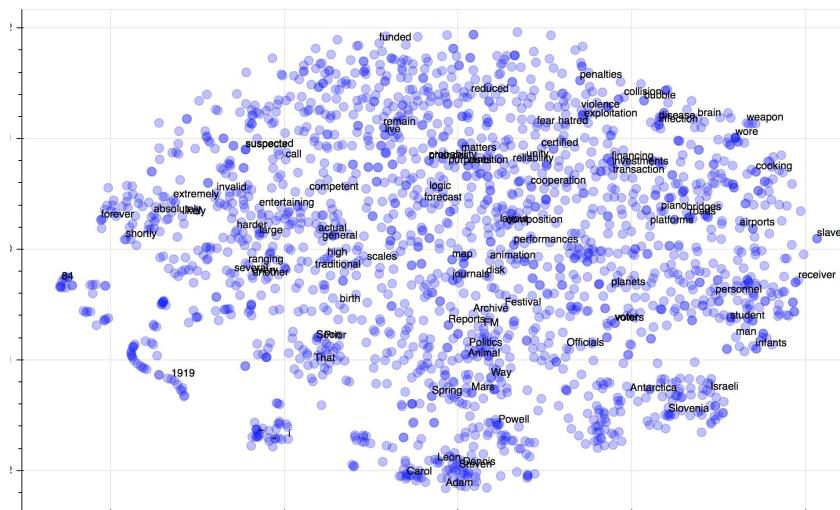
What are relations between vectors?

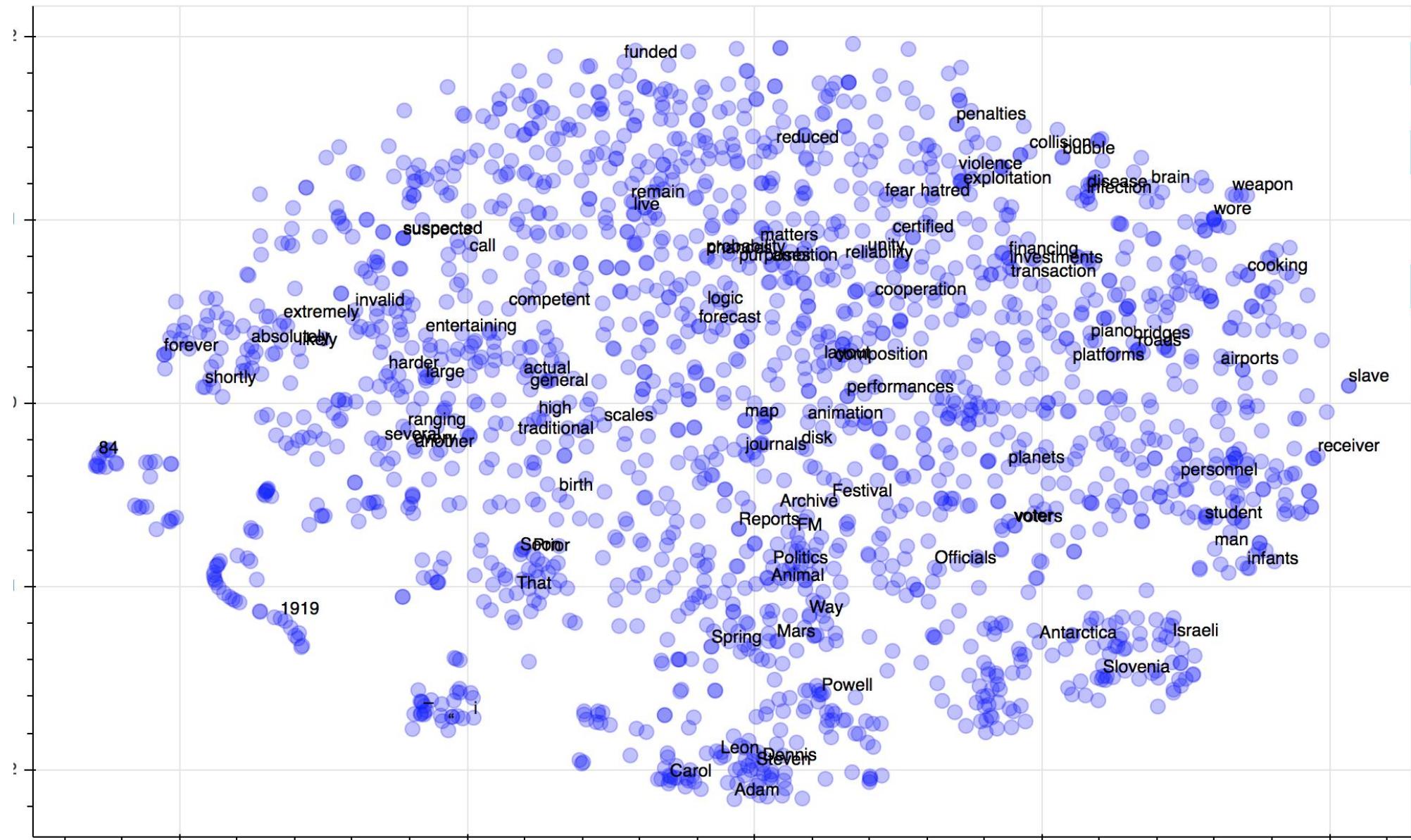
nearest neighbors of <i>frog</i>	Litoria	Leptodactylidae	Rana	Eleutherodactylus
Pictures				

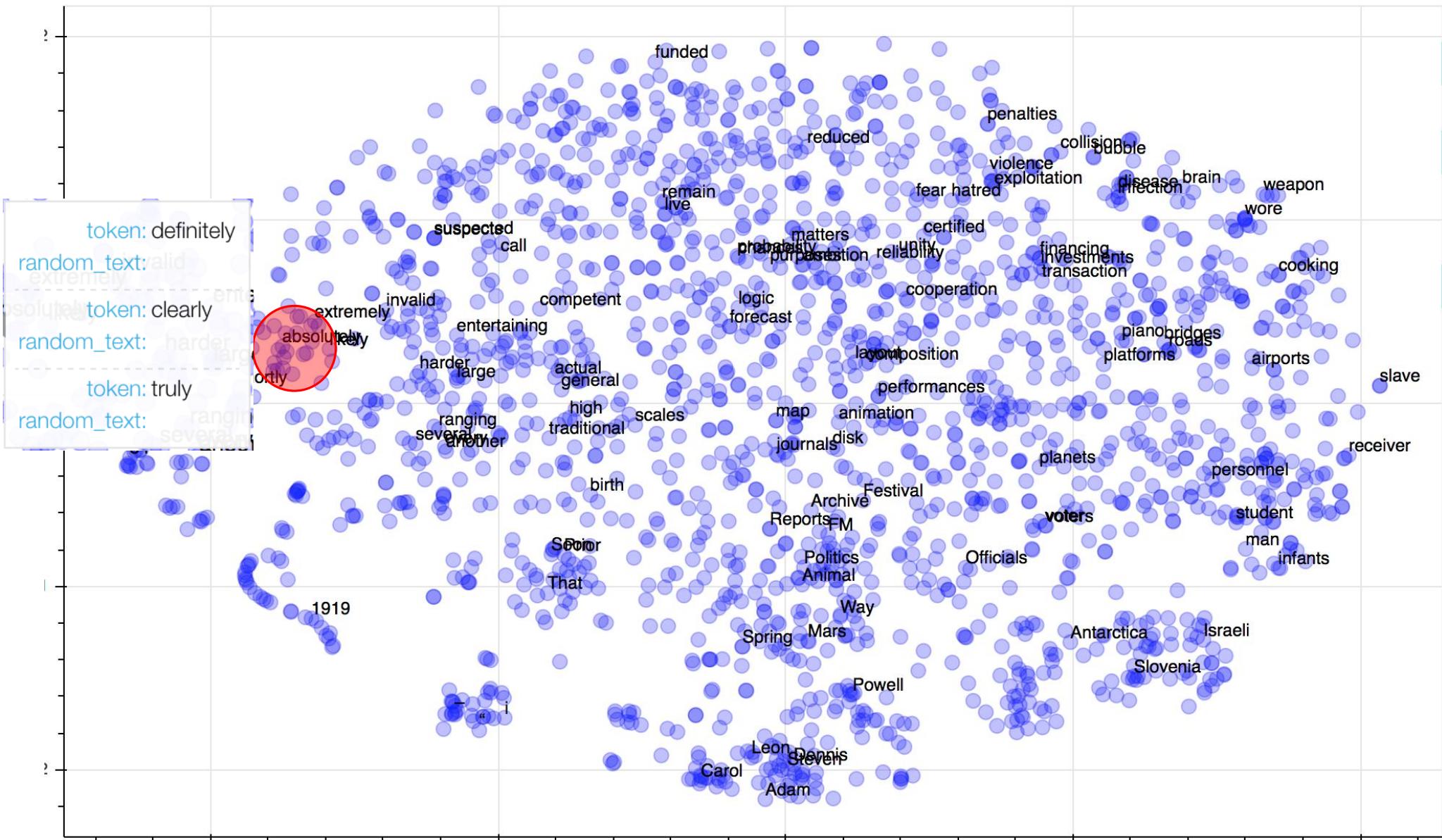
(<http://nlp.stanford.edu/projects/glove/>)

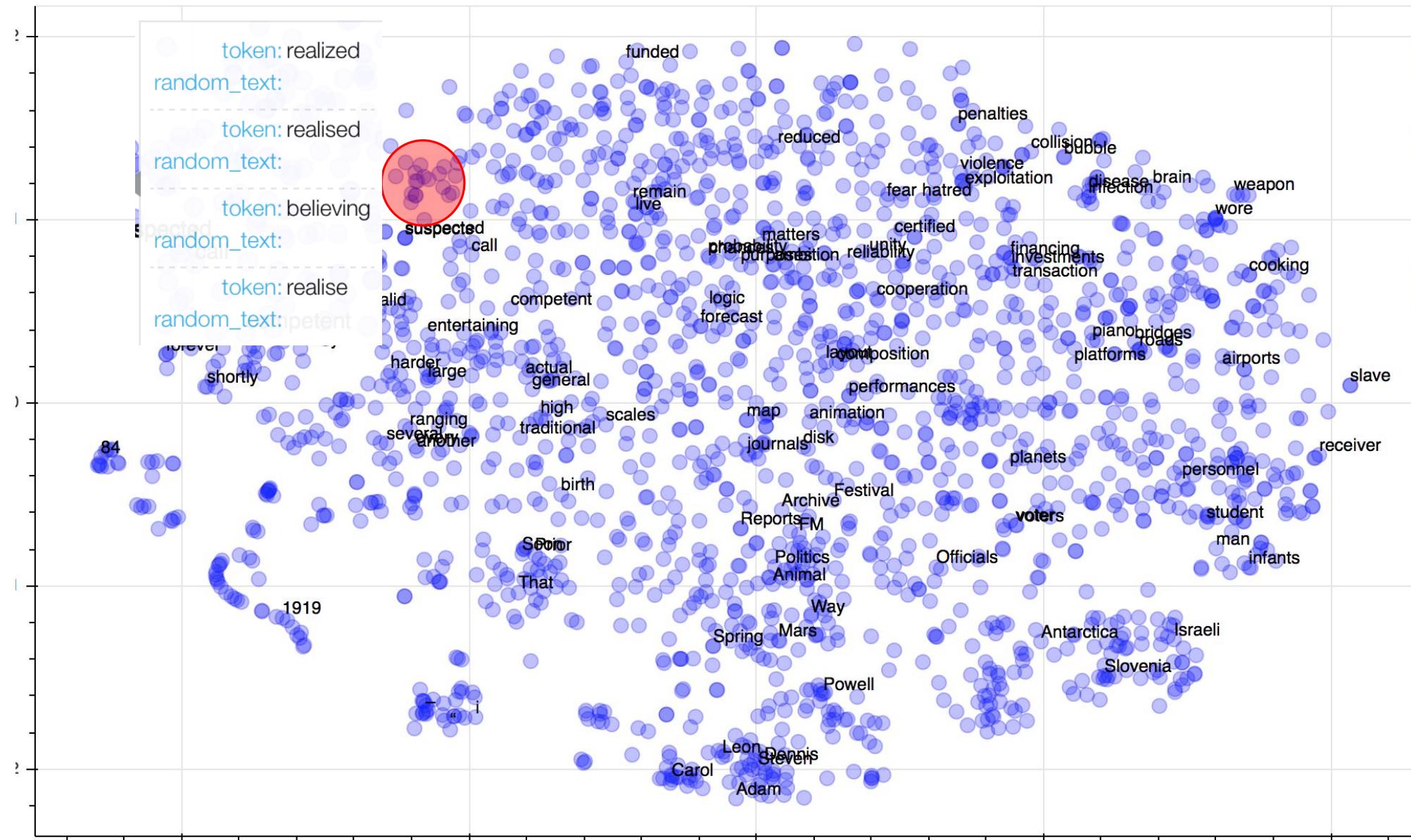
Let's walk through space...

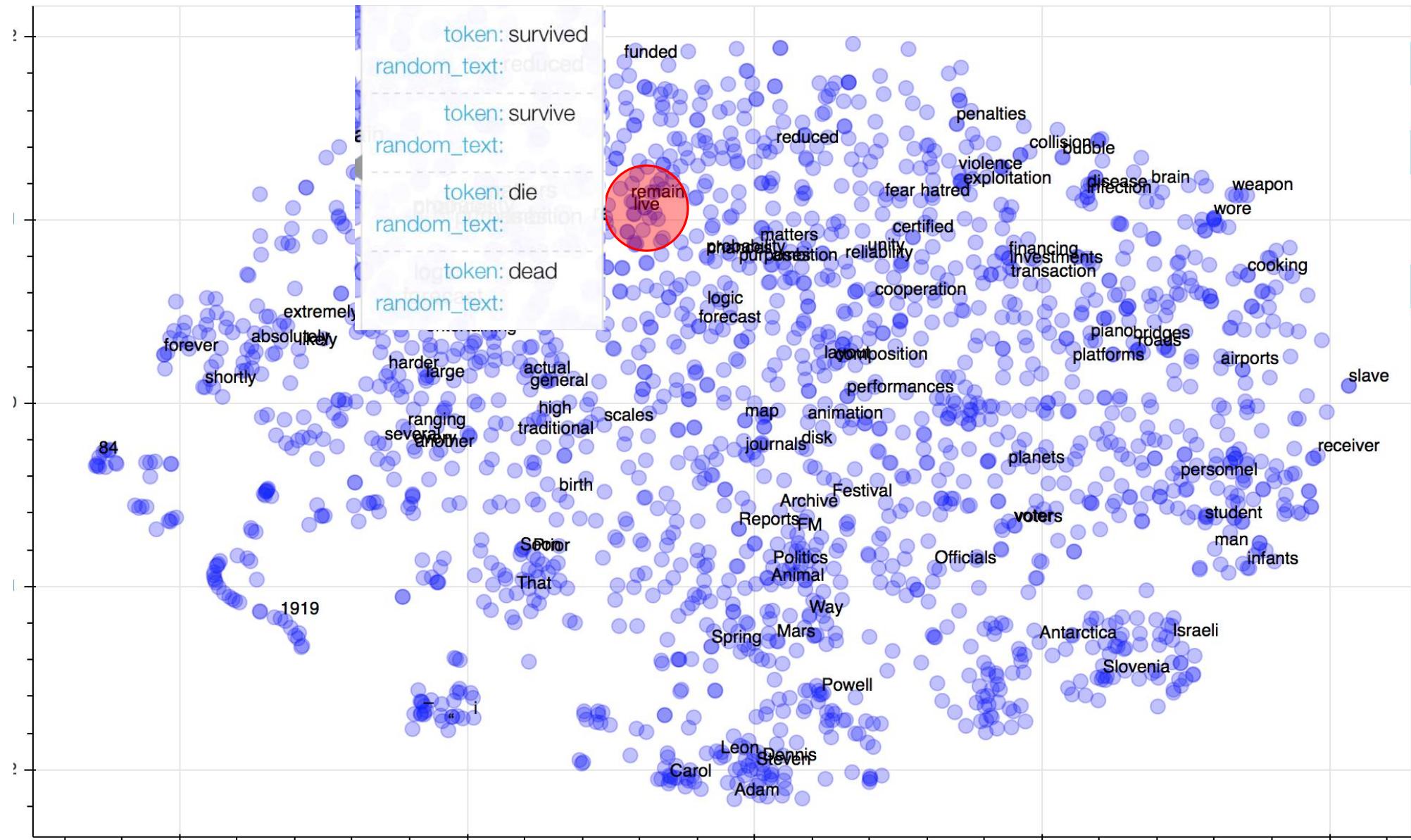
Let's walk through space...
Semantic space!

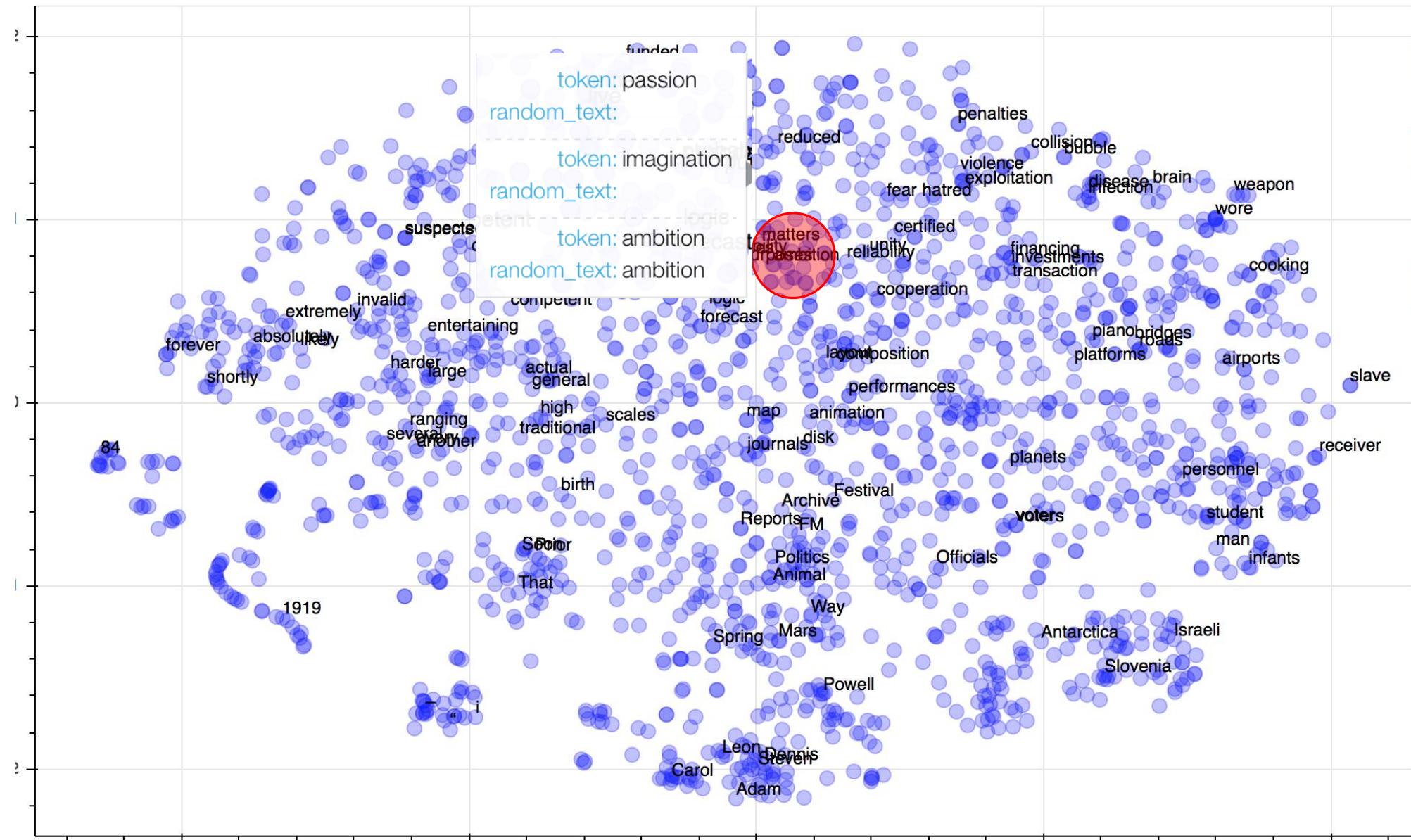


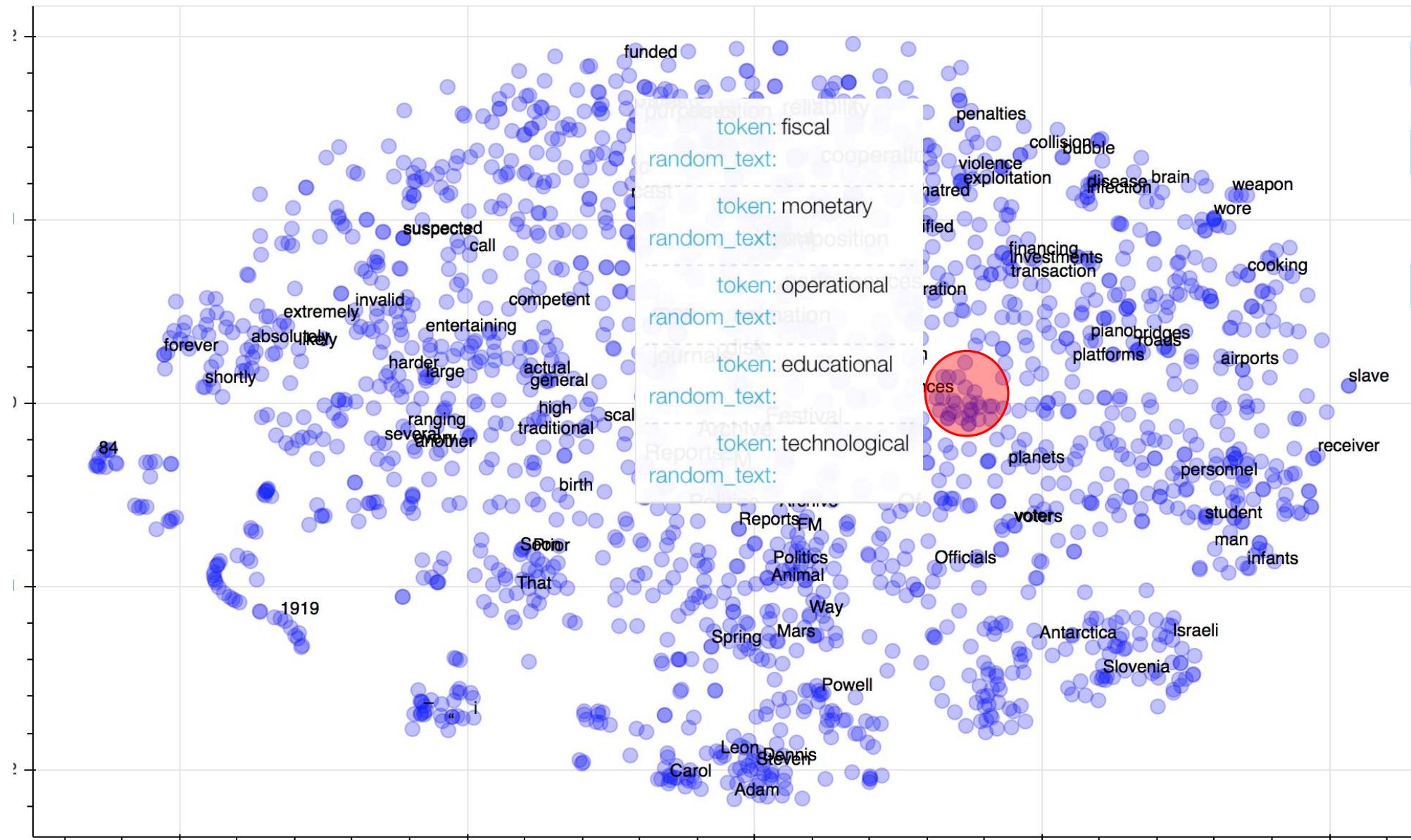




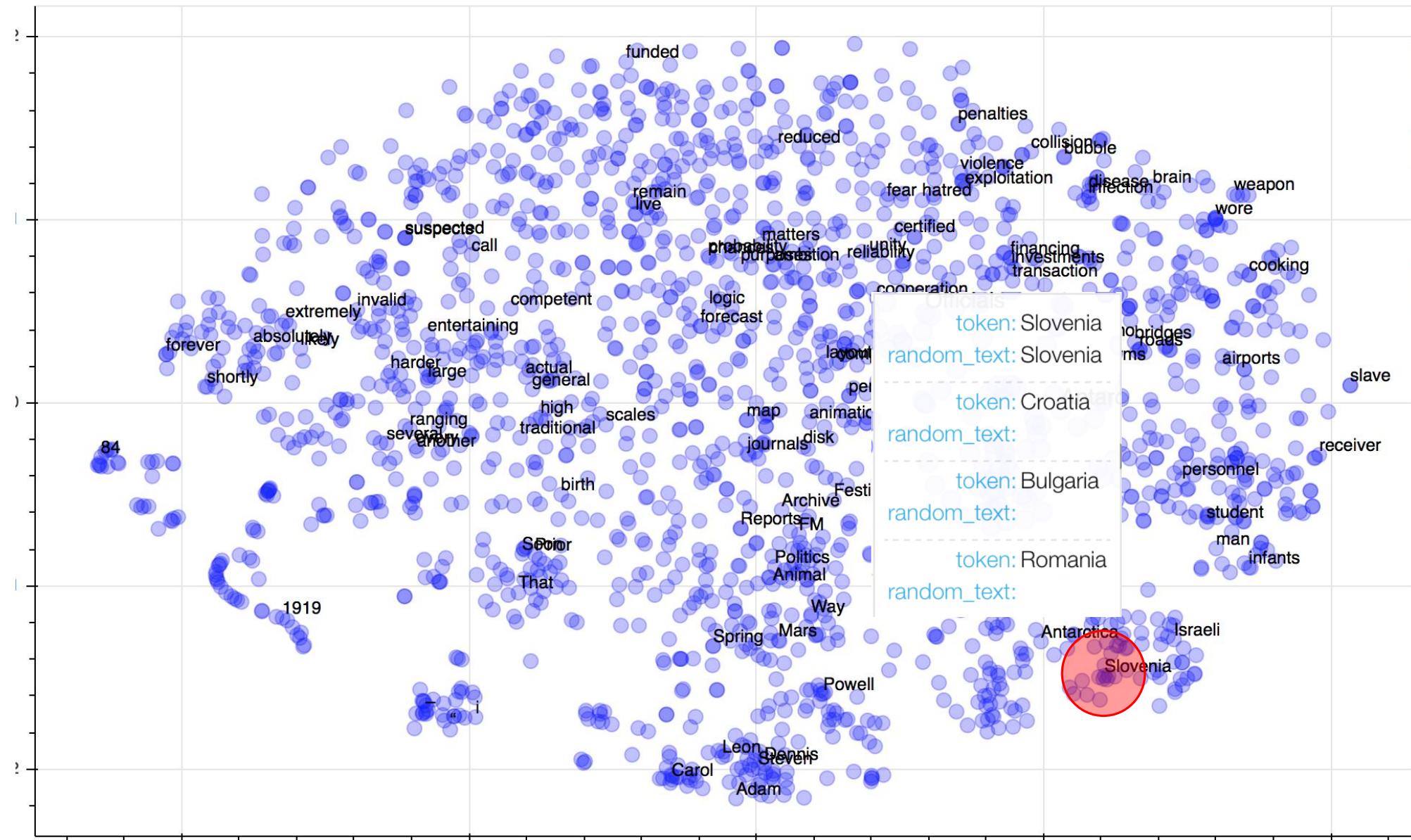


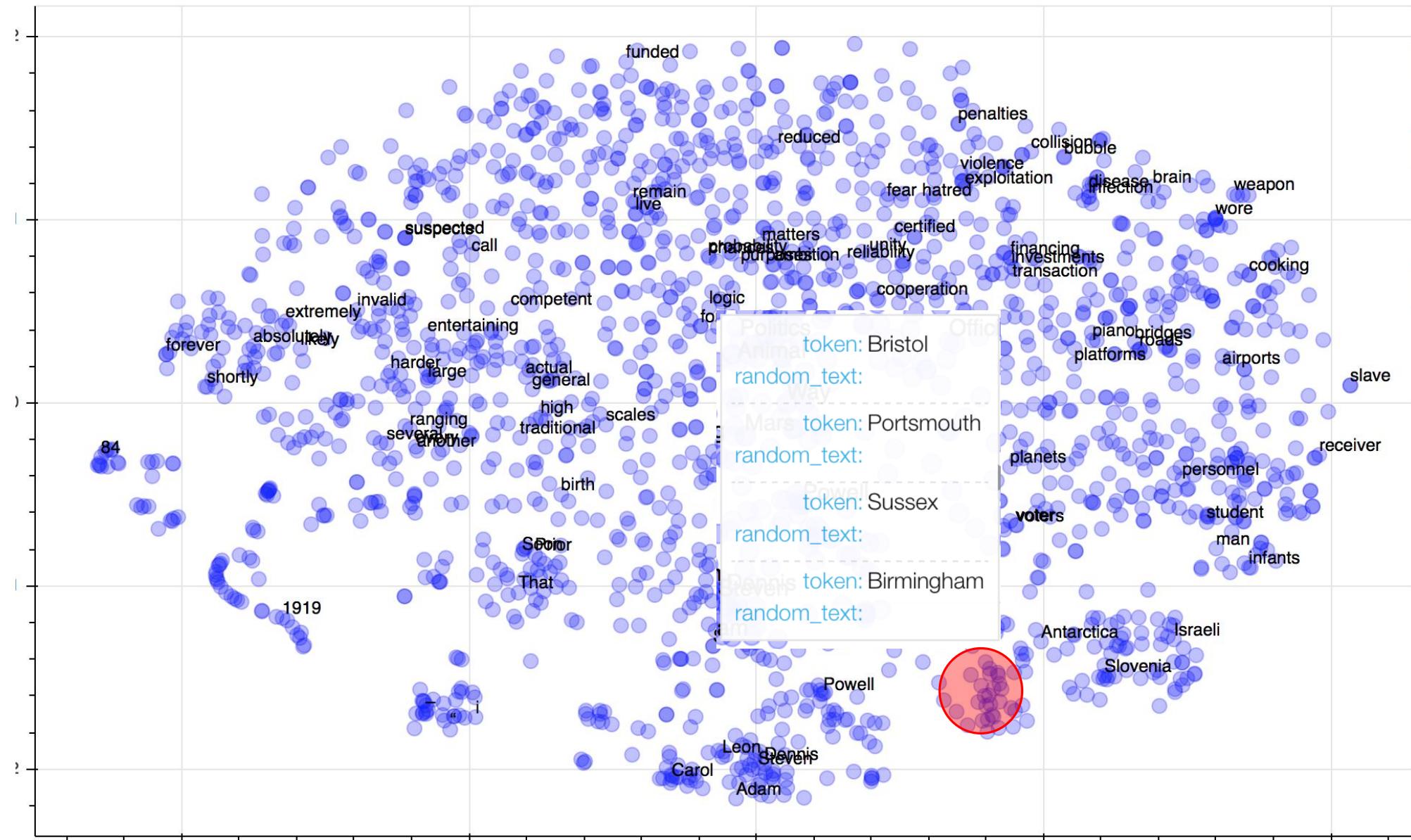


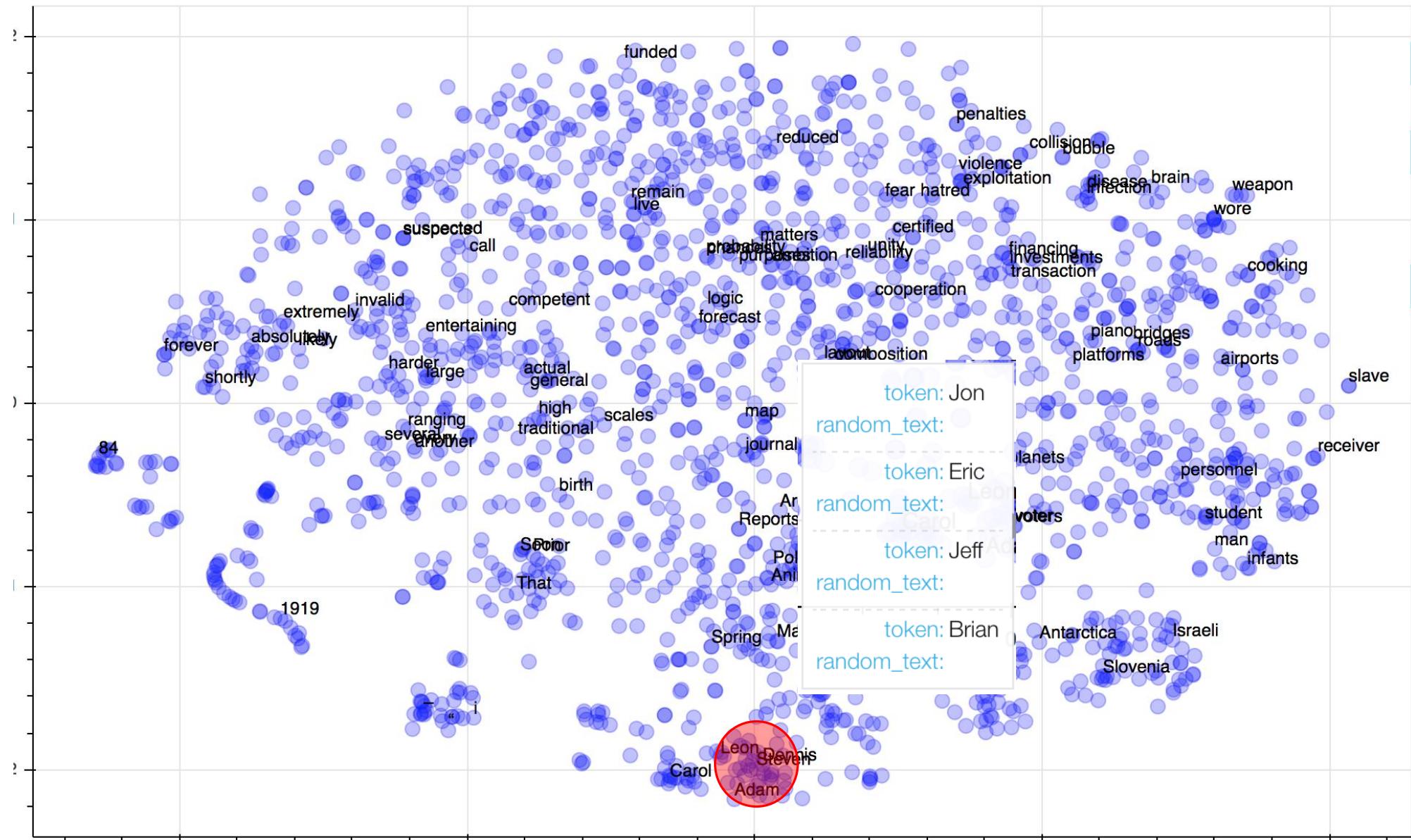


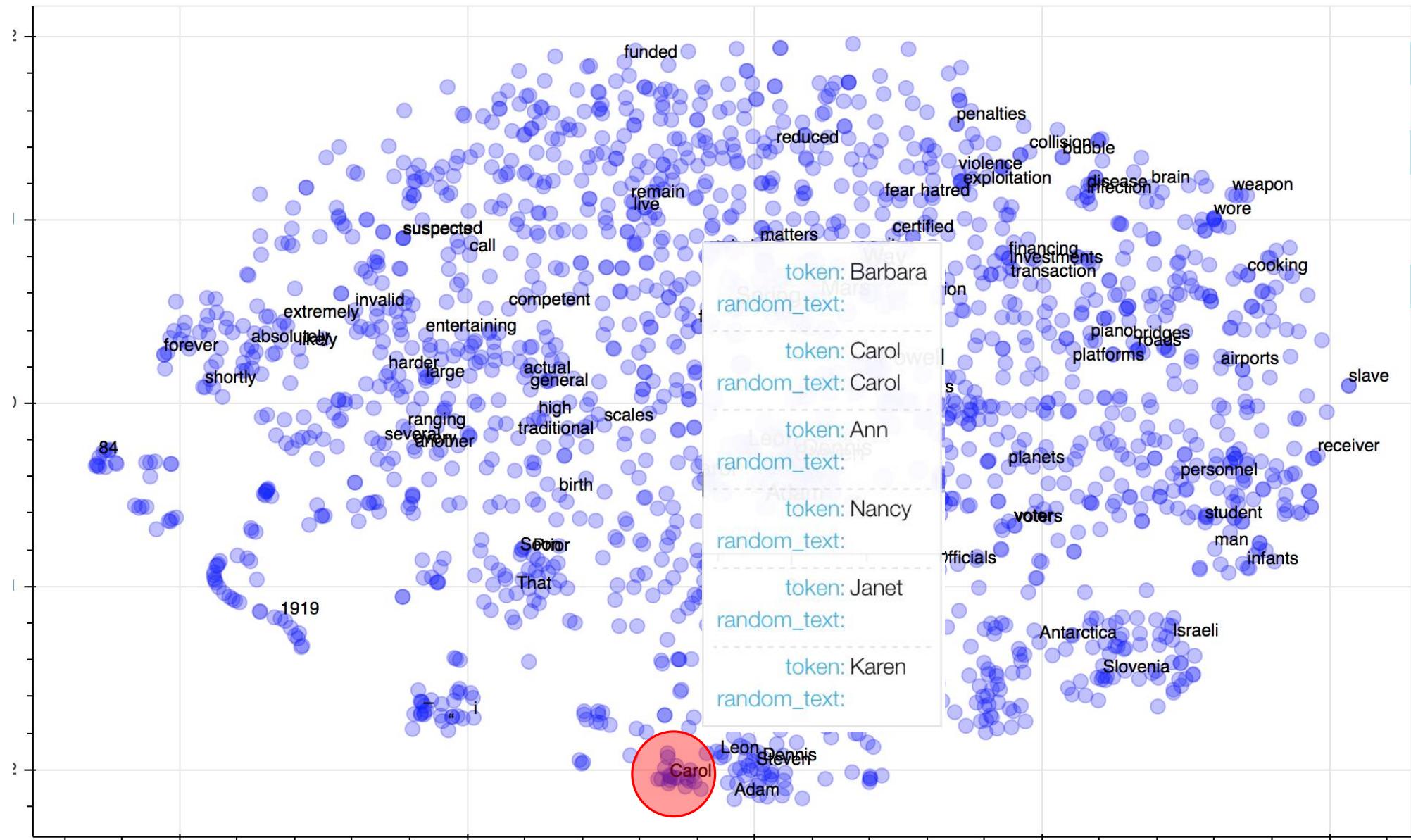


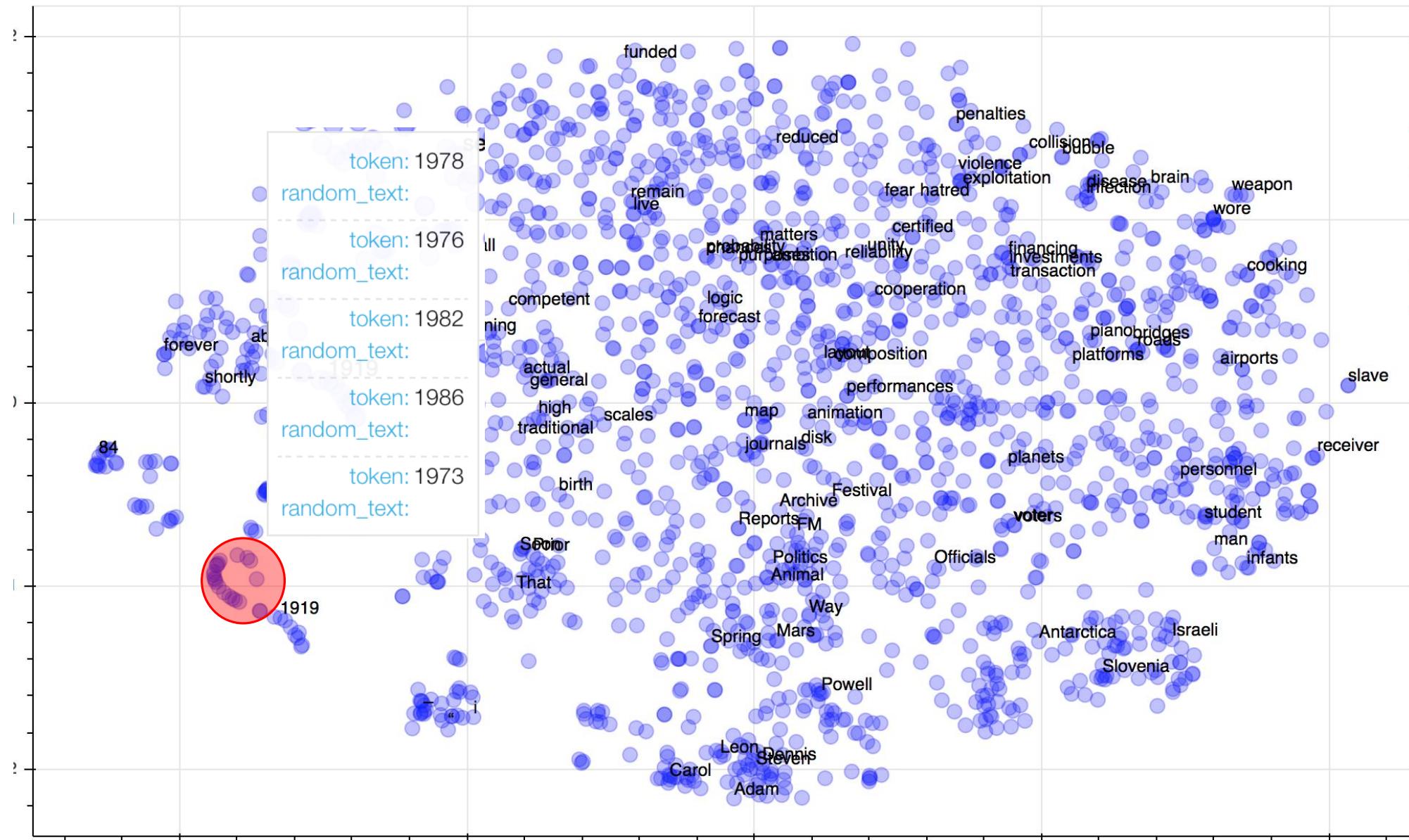












How to evaluate embeddings

Intrinsic: evaluation on a specific/intermediate subtask

- word analogies: “ a is to b as c is to ____?”
- word similarity: correlation of the rankings
- ...

Extrinsic: evaluation on a real task

- take some task (MT, NER, coreference resolution, ...) or several tasks
- train with different pretrained word embeddings
- if the task quality is better -> win!

What if..

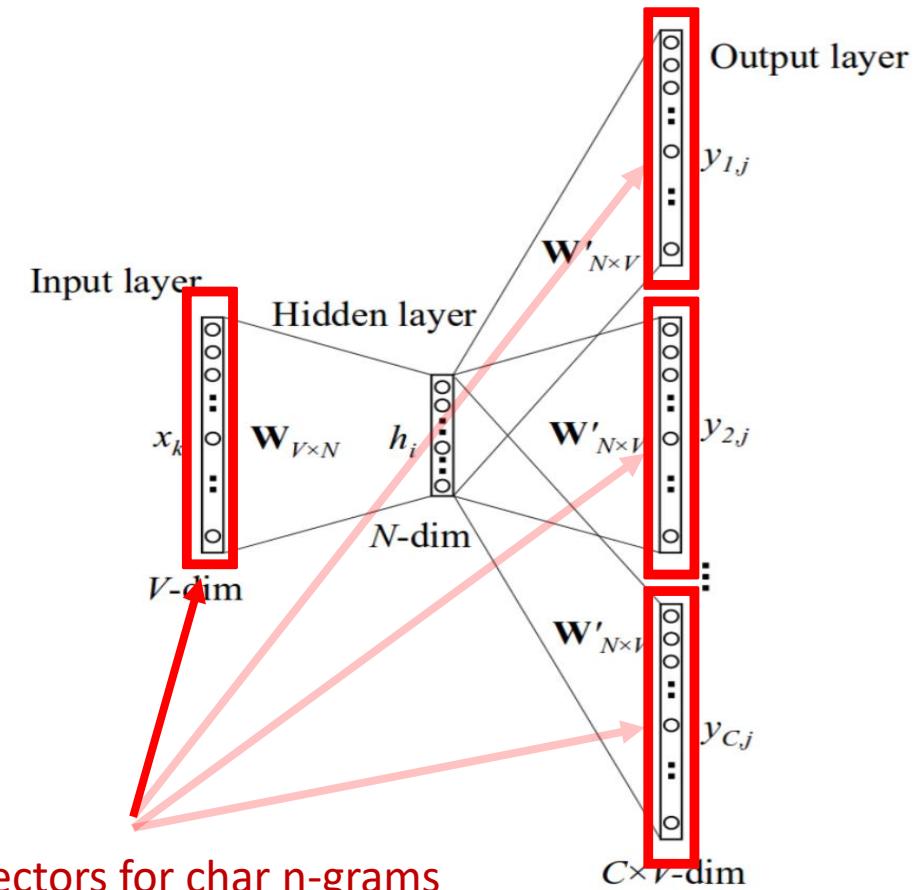
We want to use subword
information?

Adding subword information: FastText

Model: SG-NS (skip-gram with negative sampling)

Change the way word vectors are formed:

- each word represented as a bag of character n-gram `<wh, whe, her, ere, re> <where>`
- associate a vector representation to each n-gram
- represent a word by the sum of the vector representations of its n-grams

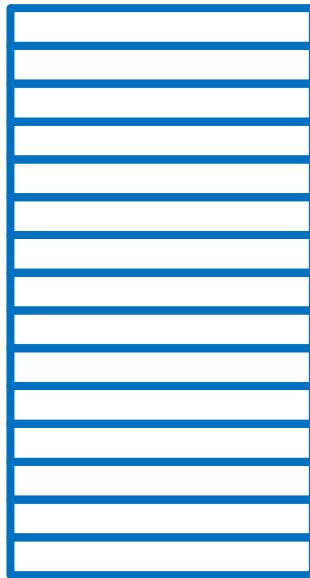


Word2Vec vs FastText

Word2Vec

where

FastText

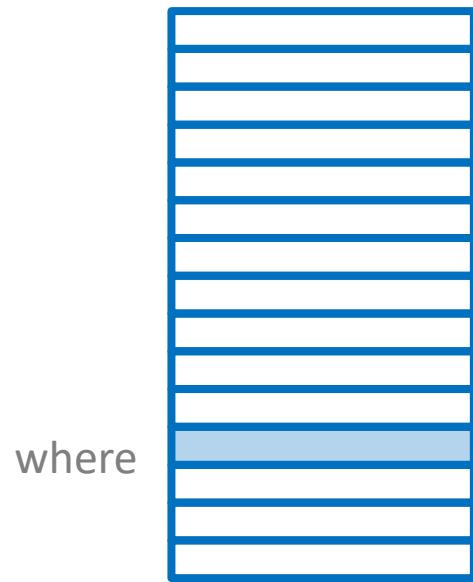


Word2Vec vs FastText

Word2Vec

where

FastText



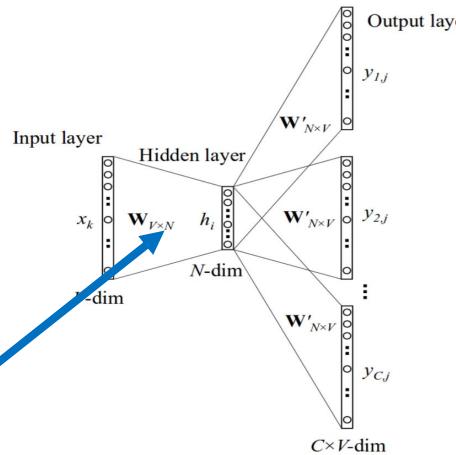
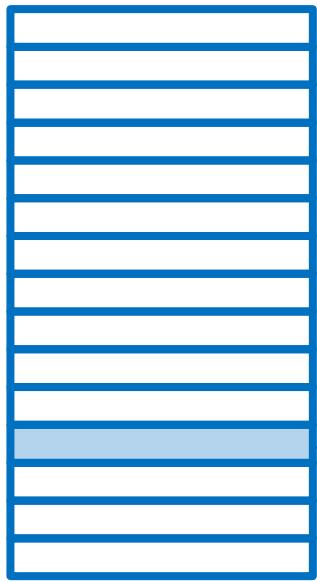
Word2Vec vs FastText

Word2Vec

where

FastText

where



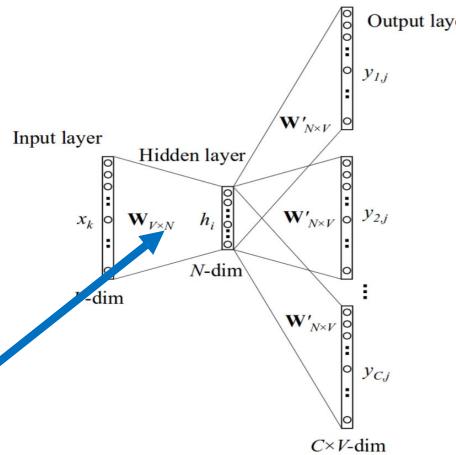
Word2Vec vs FastText

Word2Vec

where

FastText

where



<wh, whe, her, ere, re> <where>

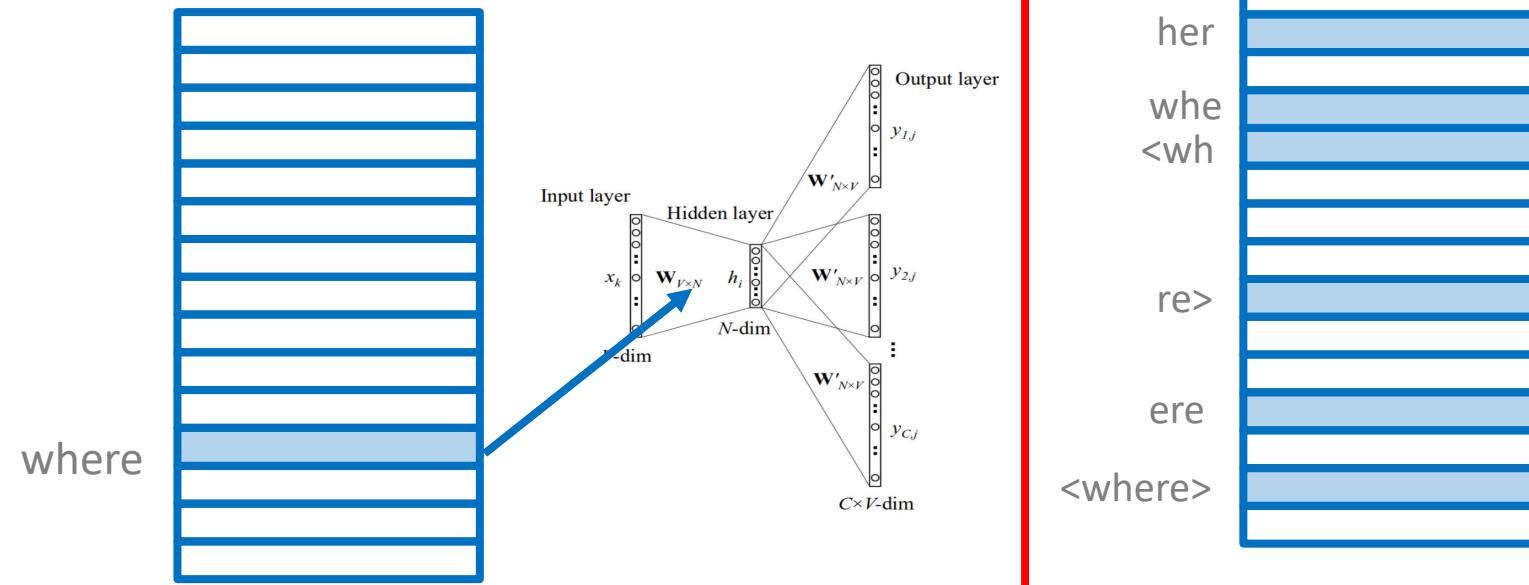


Word2Vec vs FastText

Word2Vec

where

FastText

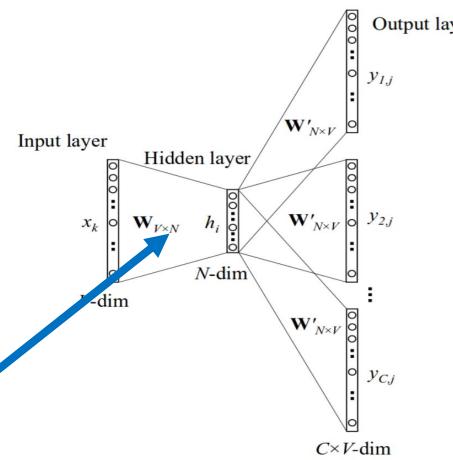
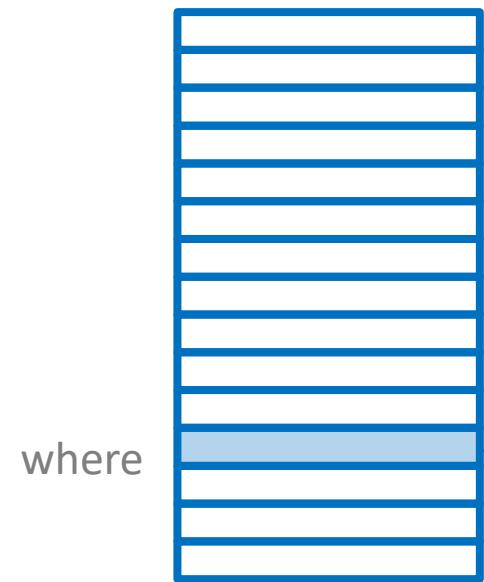


Word2Vec vs FastText

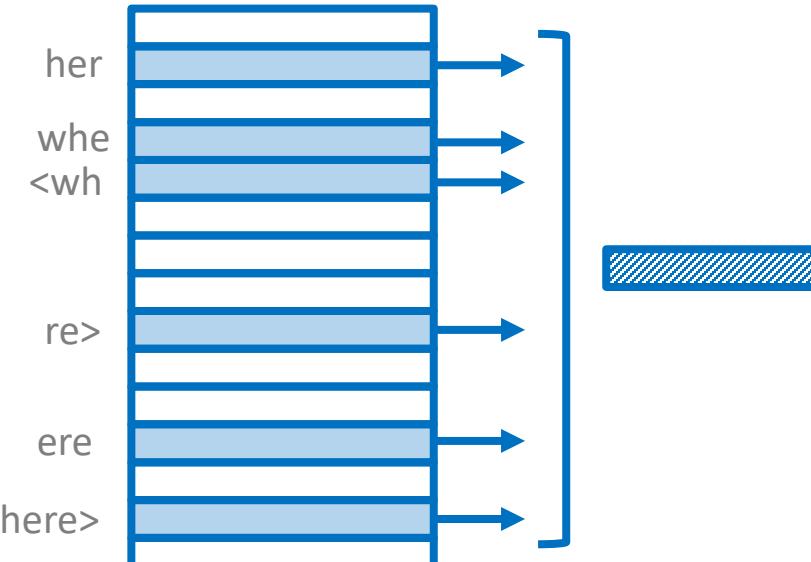
Word2Vec

where

FastText



<wh, whe, her, ere, re> <where>



sum

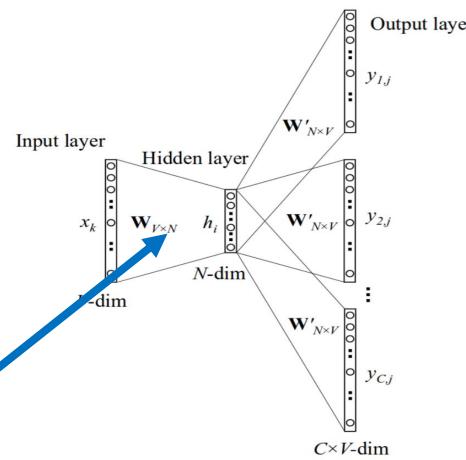
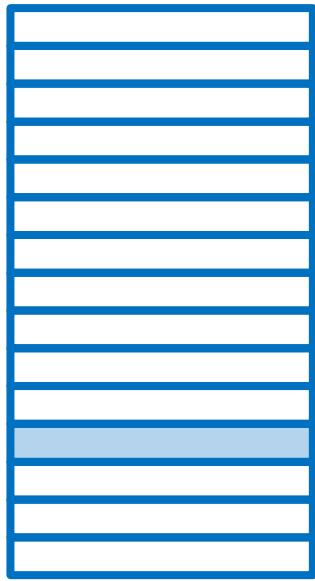
Word2Vec vs FastText

Word2Vec

where

FastText

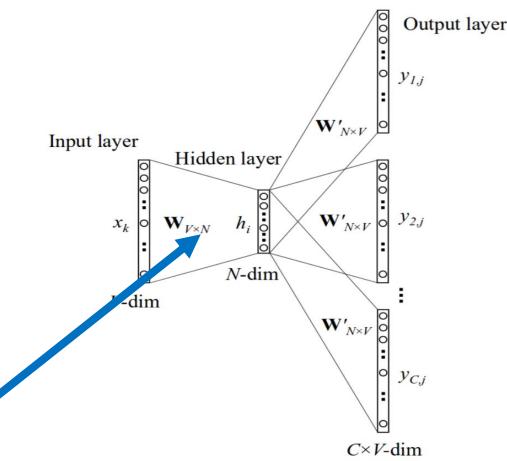
where



<wh, whe, her, ere, re> <where>

her
whe
<wh
re>
ere
<where>

sum



What if..

We abstract the skip-gram
model to the sentence level?

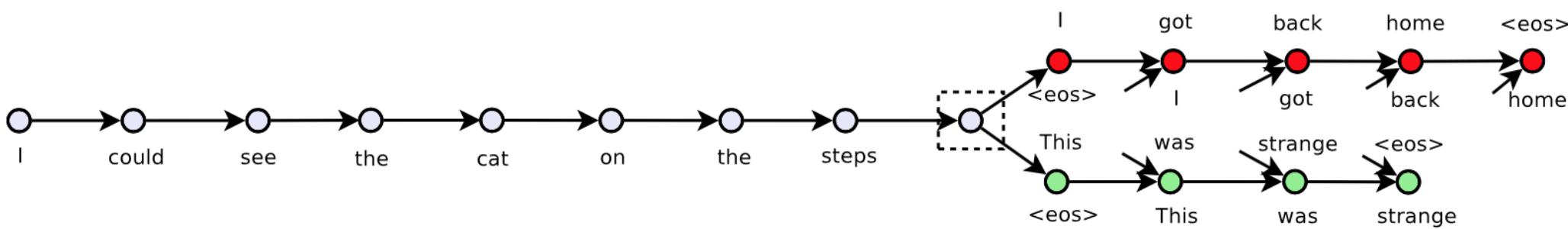
Skip-Thought Vectors

Before:

- use a word to predict its surrounding context

Now:

- encode a sentence to predict the sentences around it



What if..

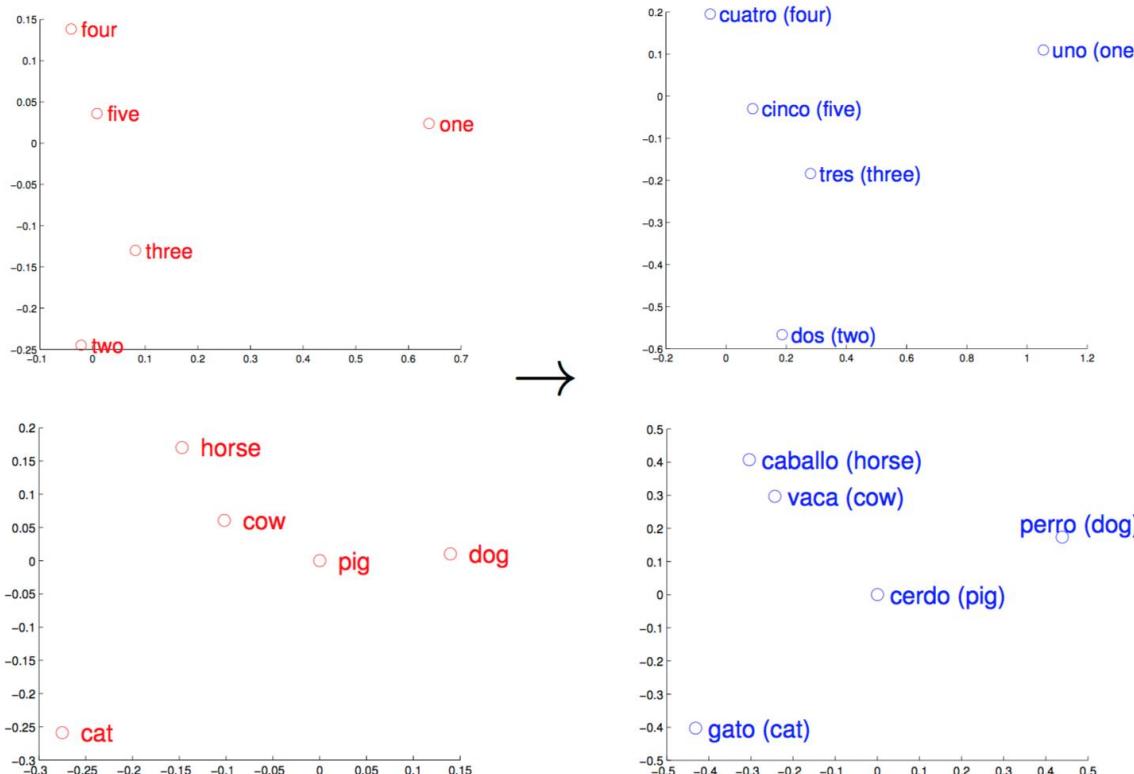
We exploit the structure of
semantic space?

Exploiting Similarities among Languages for Machine Translation

- we are given a set of word pairs and their associated vector representations
- find a transformation matrix W

$$\min_W \sum_{i=1}^n \|Wx_i - z_i\|^2$$

- for any given new word we can map it to the other language space



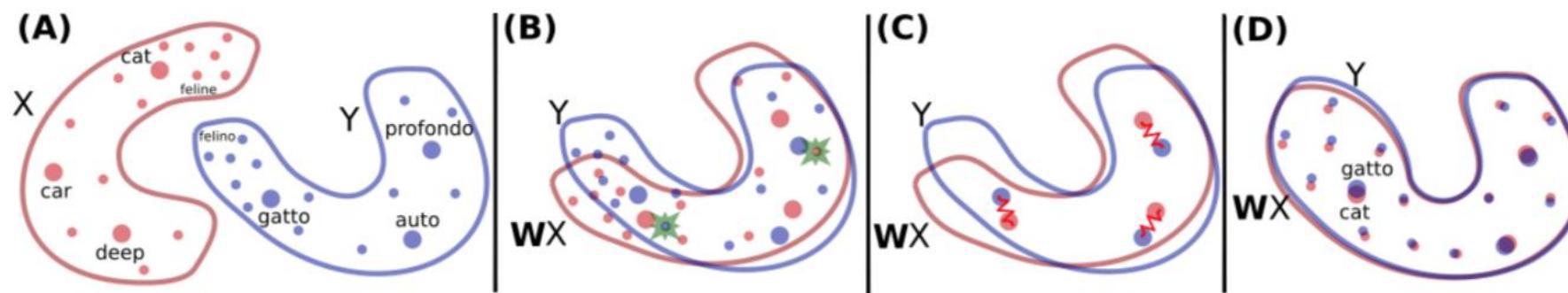
Mikolov et al, 2013, <https://arxiv.org/pdf/1309.4168.pdf>

Here we supposed that we already know
some word pairs.

But what if we know nothing about the languages?

Word Translation Without Parallel Data

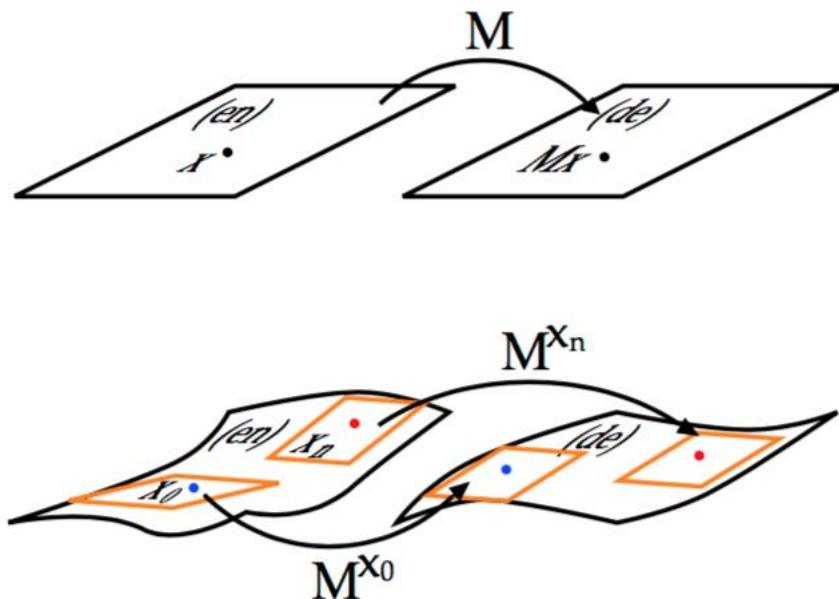
Map semantic spaces so that their samples are indistinguishable



Are the underlying maps really linear?

Look at the local linear approximations:

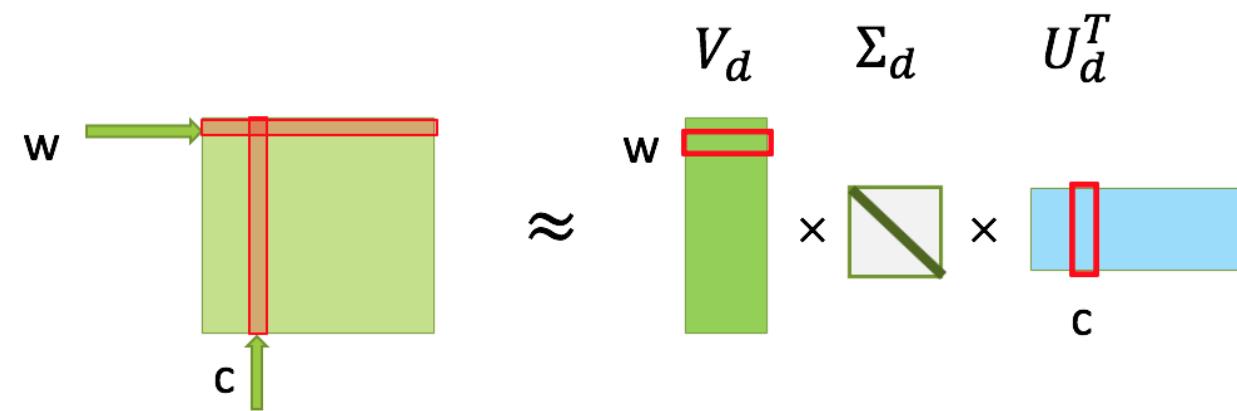
- If they're identical, then the mapping is indeed linear
- If they are not, probably not
(actually, they're not)



What happed over the last year

On the Dimensionality of Word Embedding

- Word embeddings are obtained by factorizing PMI matrix
- Dimensionality – how many largest singular values are taken



On the Dimensionality of Word Embedding

M – true signal matrix

← you DO NOT have it

\tilde{M} - estimated signal matrix

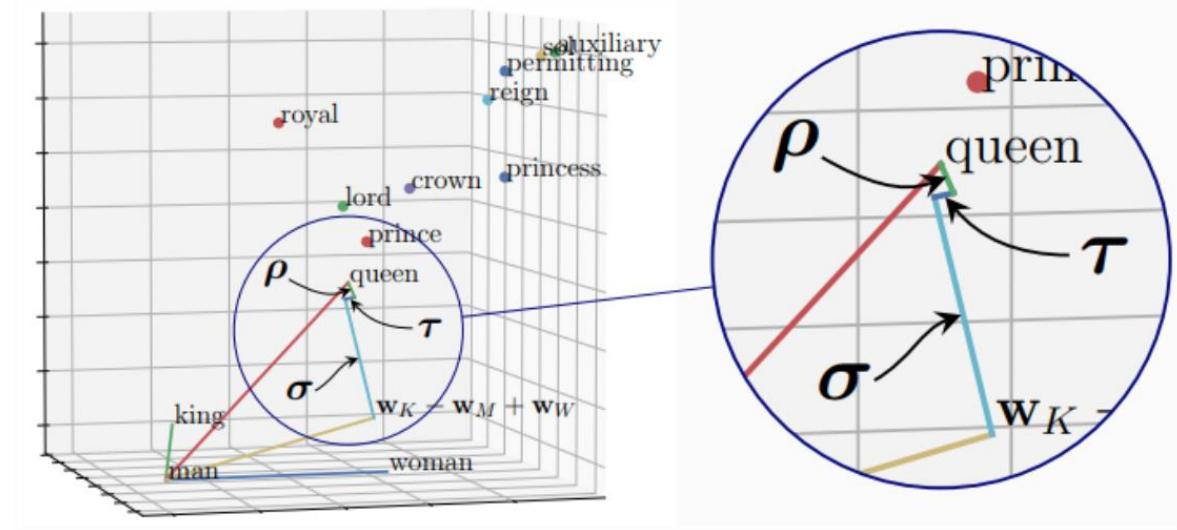
← this is what you have

The Bias-Variance Trade-off:

- Small dimension – throw away signal
- Large dimension – take noise

$$w \approx V_d \times \Sigma_d \times U_d^T$$

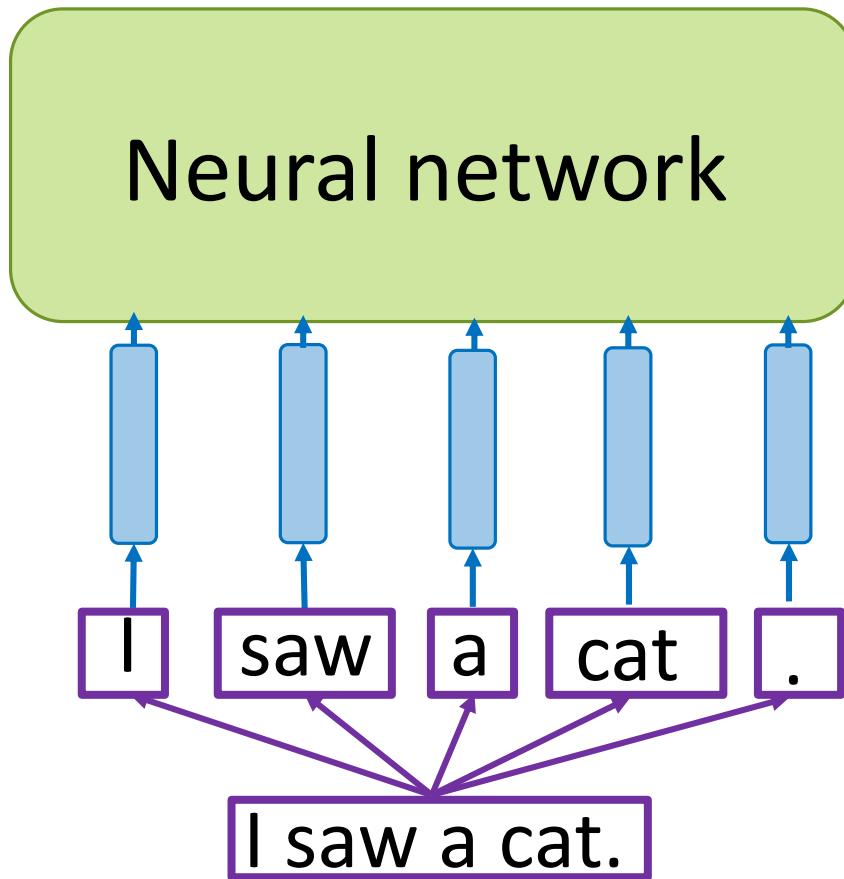
Analogies Explained: Towards Understanding Word Embeddings



Allen & Hospedales, ICML 2019, Best Paper Honourable Mention, <https://arxiv.org/abs/1901.09813>
Official blog: <https://carl-allen.github.io/nlp/2019/07/01/explaining-analogies-explained.html>

A piece of practice:
When we really need to learn word
representations?

Why do we need word representation?



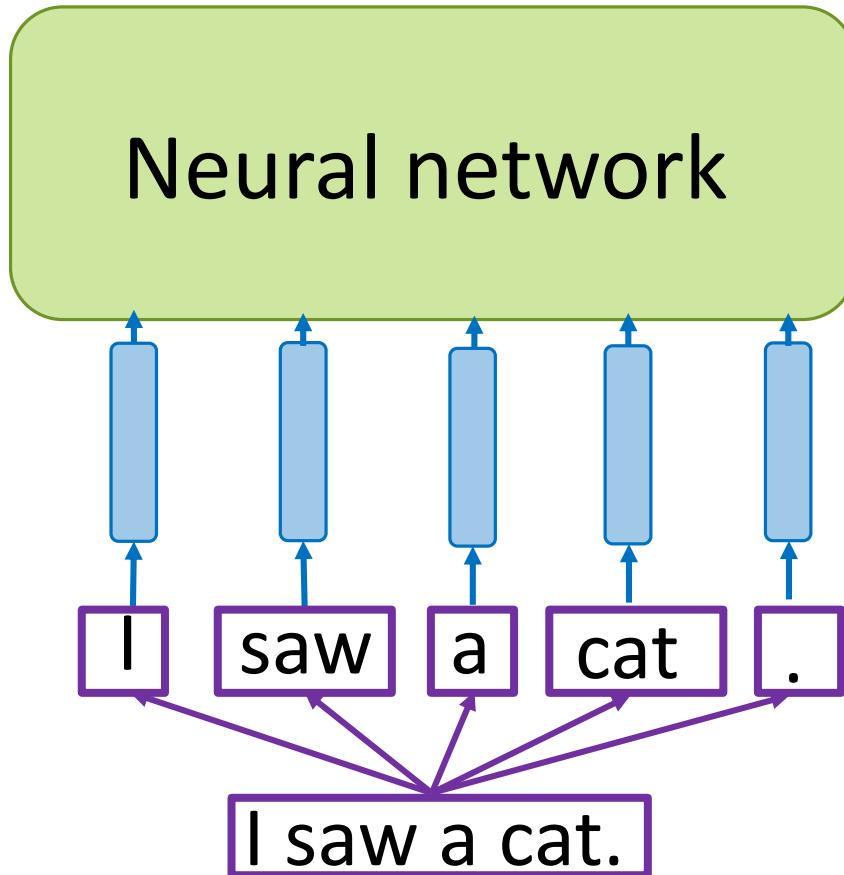
Any NN for solving any task

Word representation - vector
(word embedding)

Sequence of tokens

Text

Do we REALLY need to learn word representations in advance?



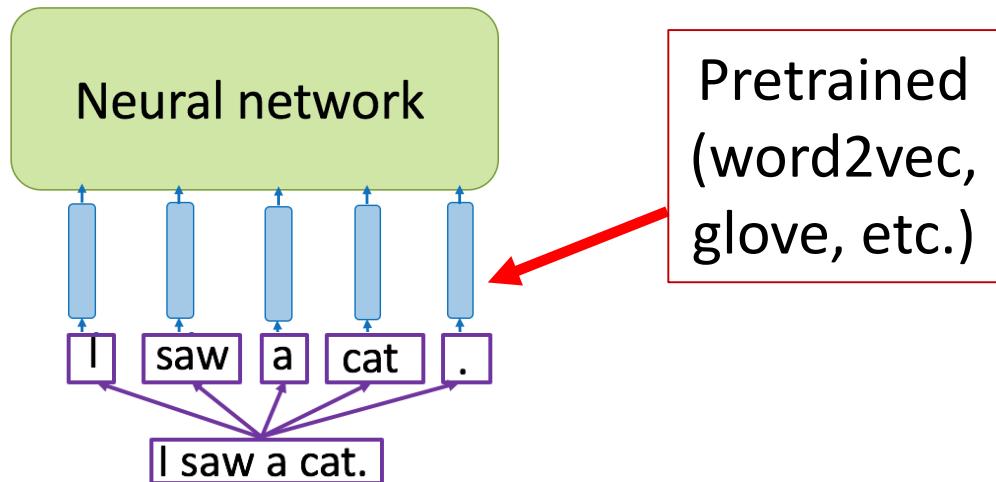
Ok, but if we **already have an NN** for our task, why do we have to learn parameters for word embeddings using **some other NN**?

When to use pretrained embeddings?

Not enough data or the task is too simple



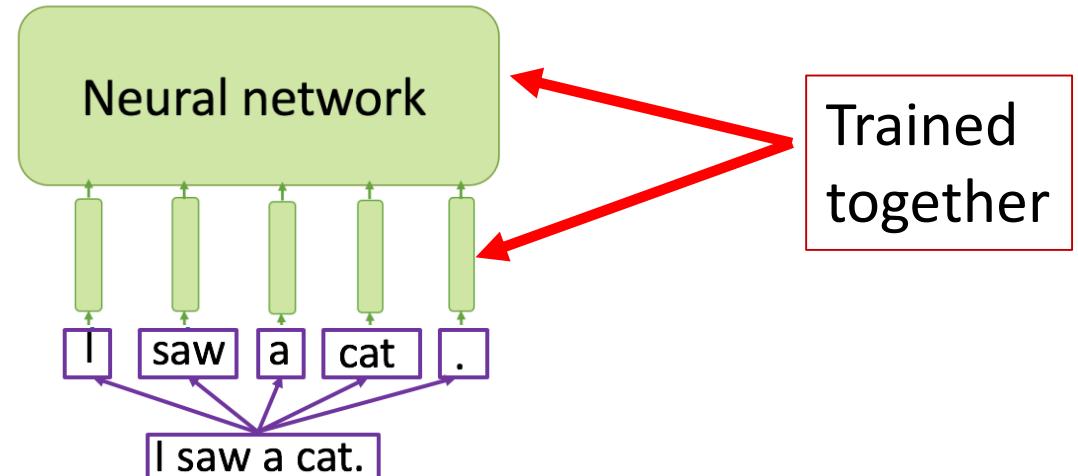
Use pretrained on the other task



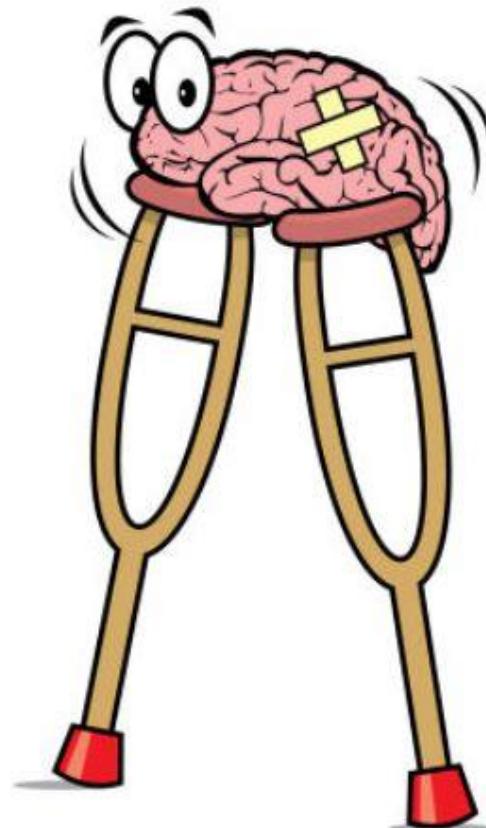
Enough data and a hard task (LM, MT, ...)



Train with the model

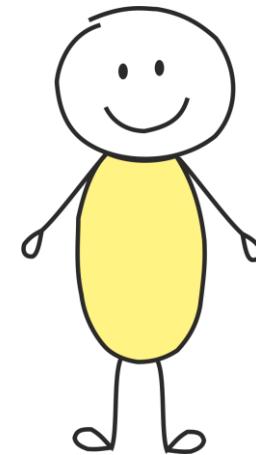
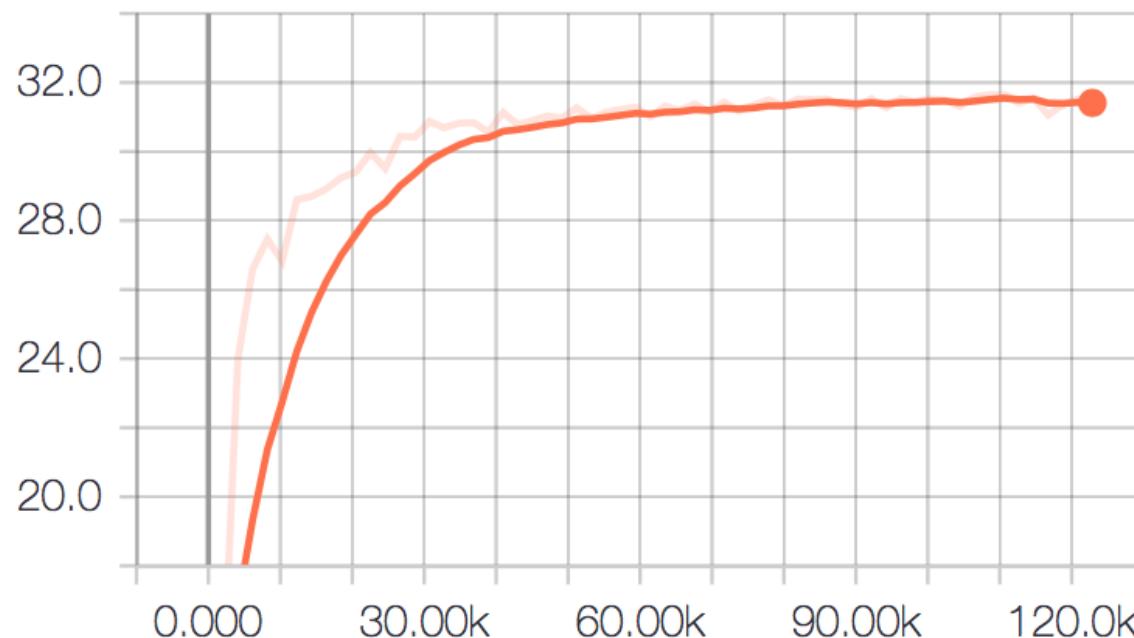


Hack of the day

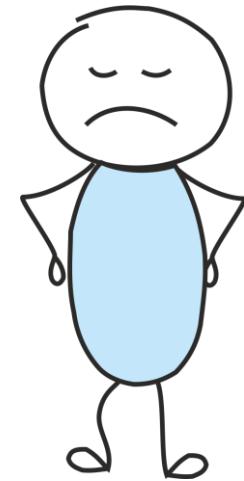
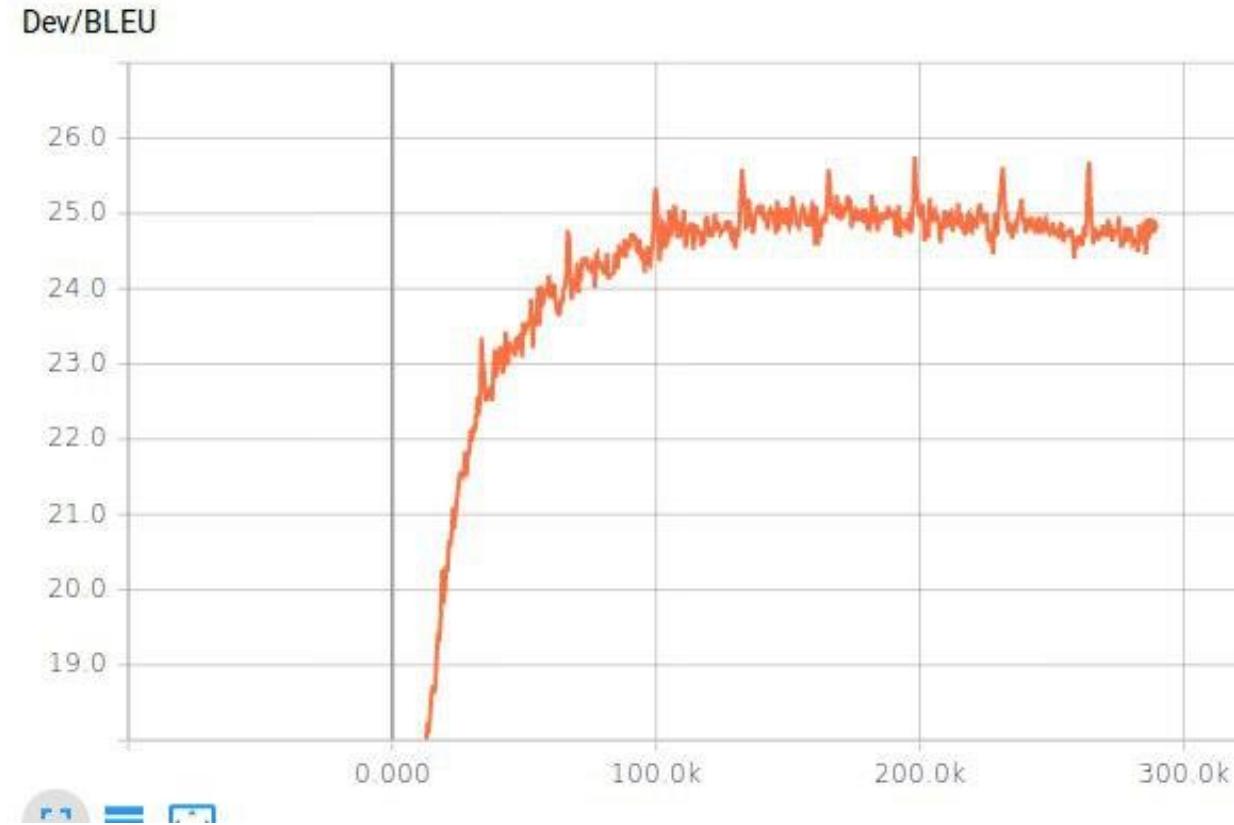


Tensorboard of a healthy man

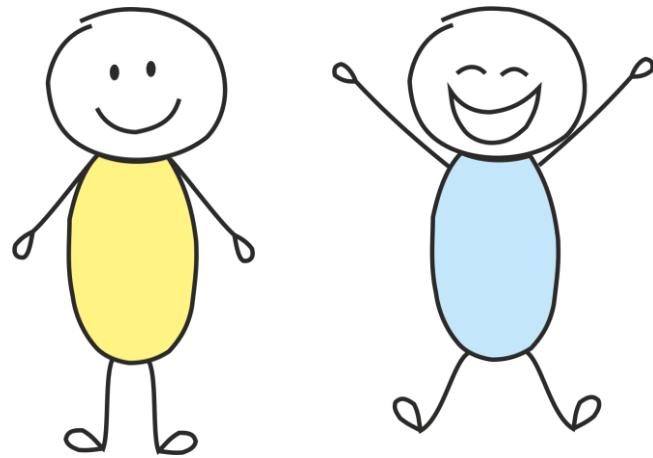
Dev/BLEU



Tensorboard of a man who doesn't shuffle his data



Shuffle your data!



Congratulations, you've just
survived the first NLP lecture!

Looking forward to the next week's episode...

Sincerely yours,
Yandex Research