Introduction to Word Embeddings

Maximilian Mozes
Department of Security and Crime Science
University College London

maximilian.mozes@ucl.ac.uk

Agenda

1. Introduction

2. word2vec

3. GloVe

Introduction

Introduction

- Word embeddings capture the semantics of individual words

Introduction

- Word embeddings capture the semantics of individual words
- However, a single word can have
  - different meanings (*parts of speech*)

*Death in Venice is my favorite **novel**.*

*He came up with a **novel** idea.*

  - different meanings in different contexts

*Don't use the trackpad, just use your **mouse**.*

*My cat just caught a **mouse**.*

Introduction

- Word embeddings capture the semantics of individual words
- However, a single word can have
  - different meanings (*parts of speech*)

*Death in Venice is my favorite **novel**.*

*He came up with a **novel** idea.*

  - different meanings in different contexts

*Don't use the trackpad, just use your **mouse**.*
*My cat just caught a **mouse**.*

Language is very complex!

Representing words

- To capture a word's meaning, we need to store information associated with it
- In computational settings, information are represented as numbers
- Thus, to represent a word we assign it to a **list of numbers**

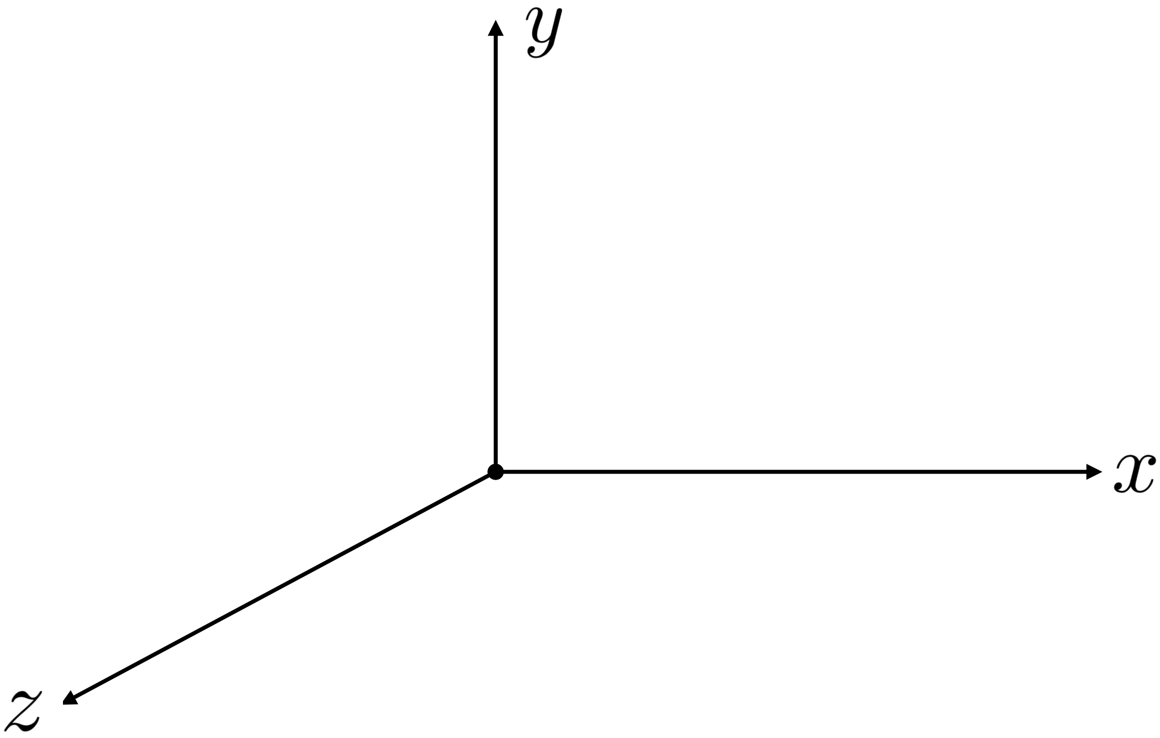
Representing words

- To capture a word's meaning, we need to store information associated with it
- In computational settings, information are represented as numbers
- Thus, to represent a word we assign it to a **list of numbers**

$$\text{mouse} \mapsto \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$$

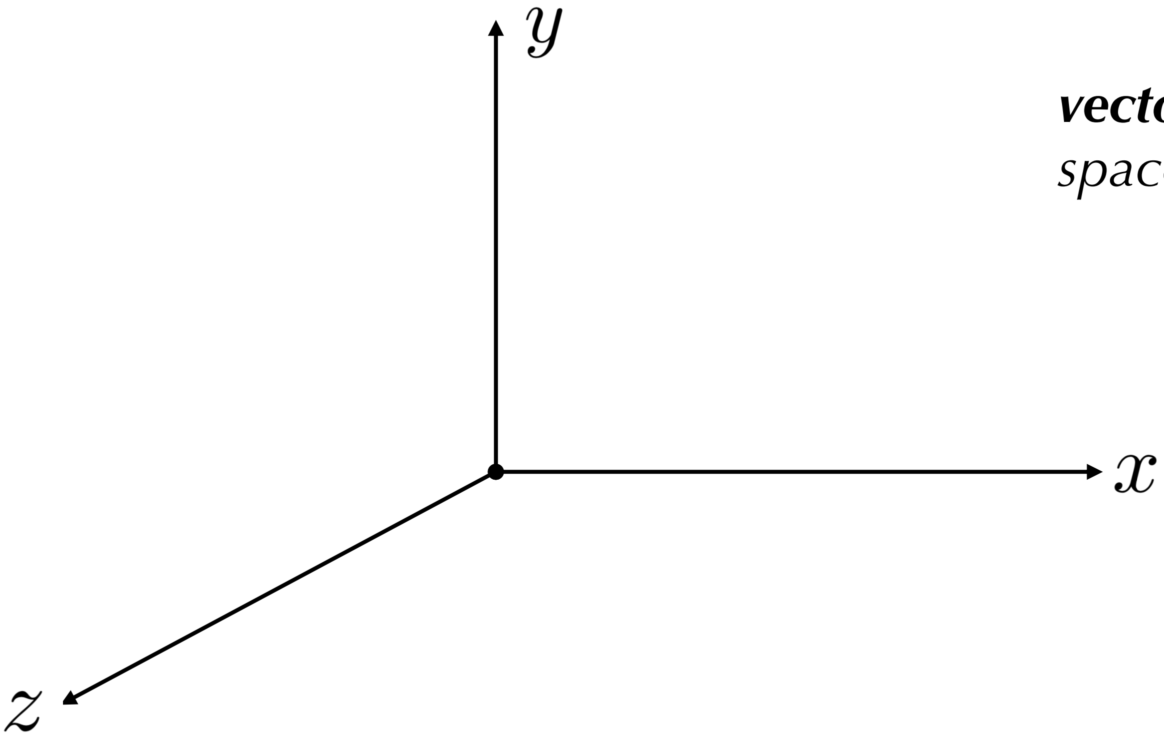
Representing words

Word embeddings map words to lists of numbers (**vectors**)



Representing words

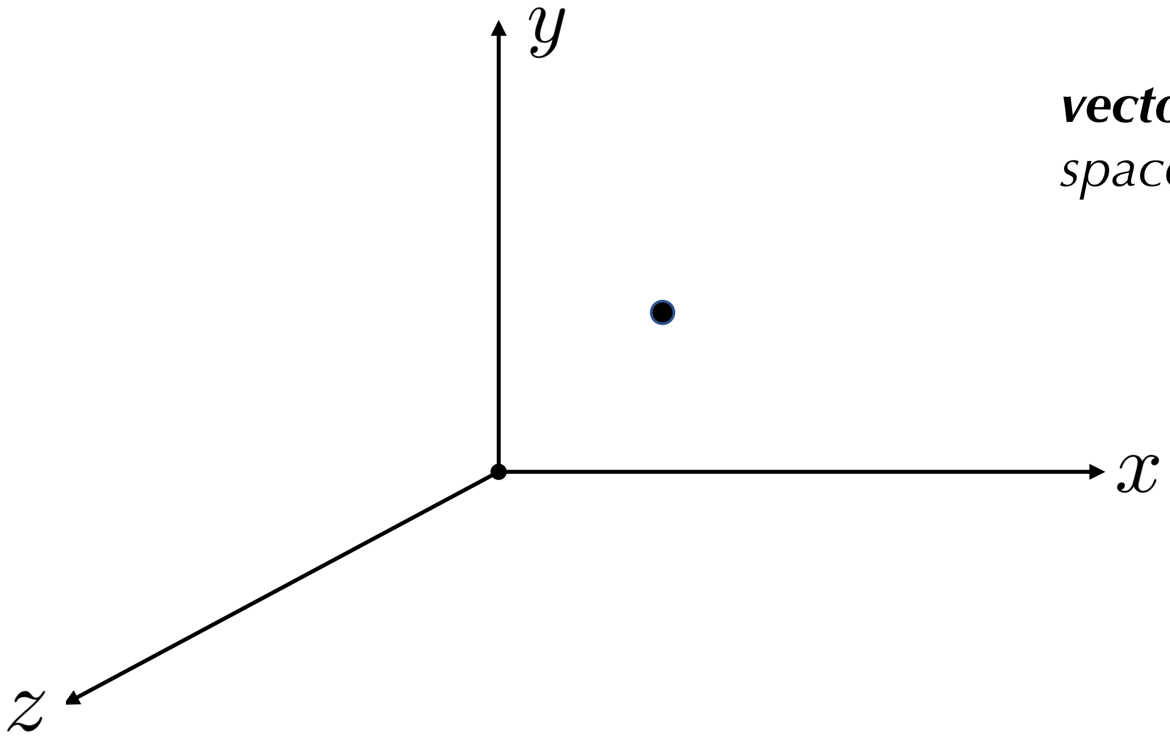
Word embeddings map words to lists of numbers (**vectors**)



vector space:
space where vectors live

Representing words

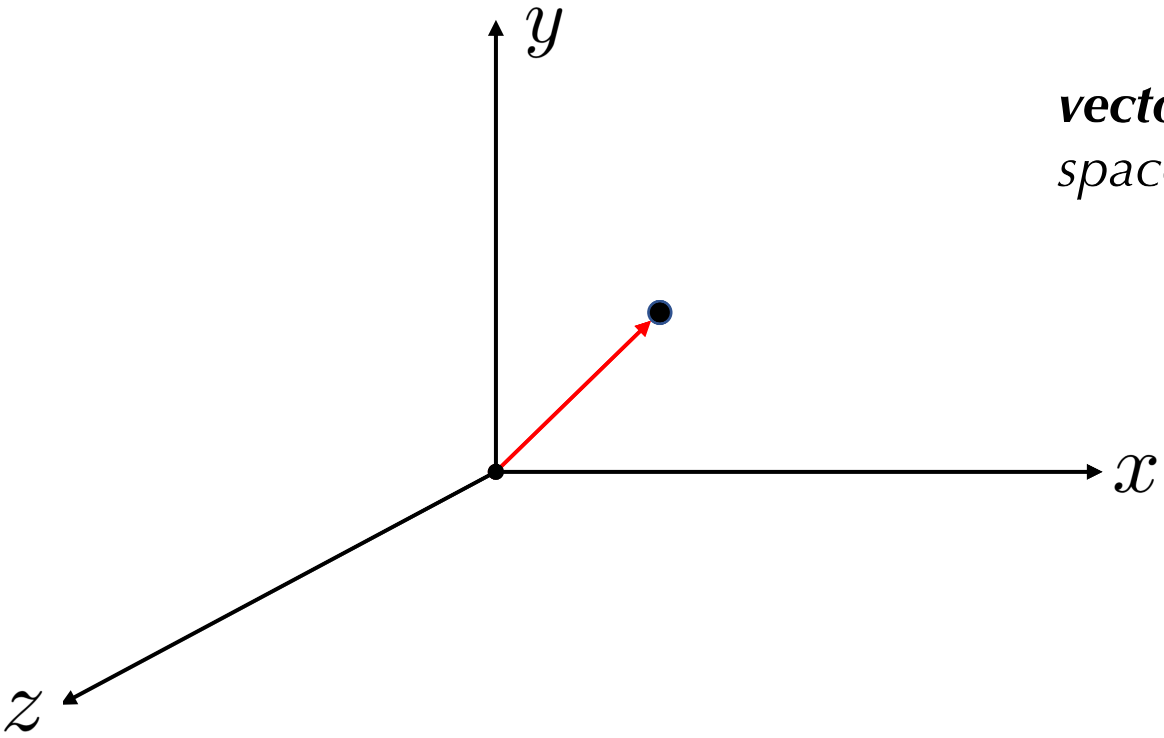
Word embeddings map words to lists of numbers (**vectors**)



vector space:
space where vectors live

Representing words

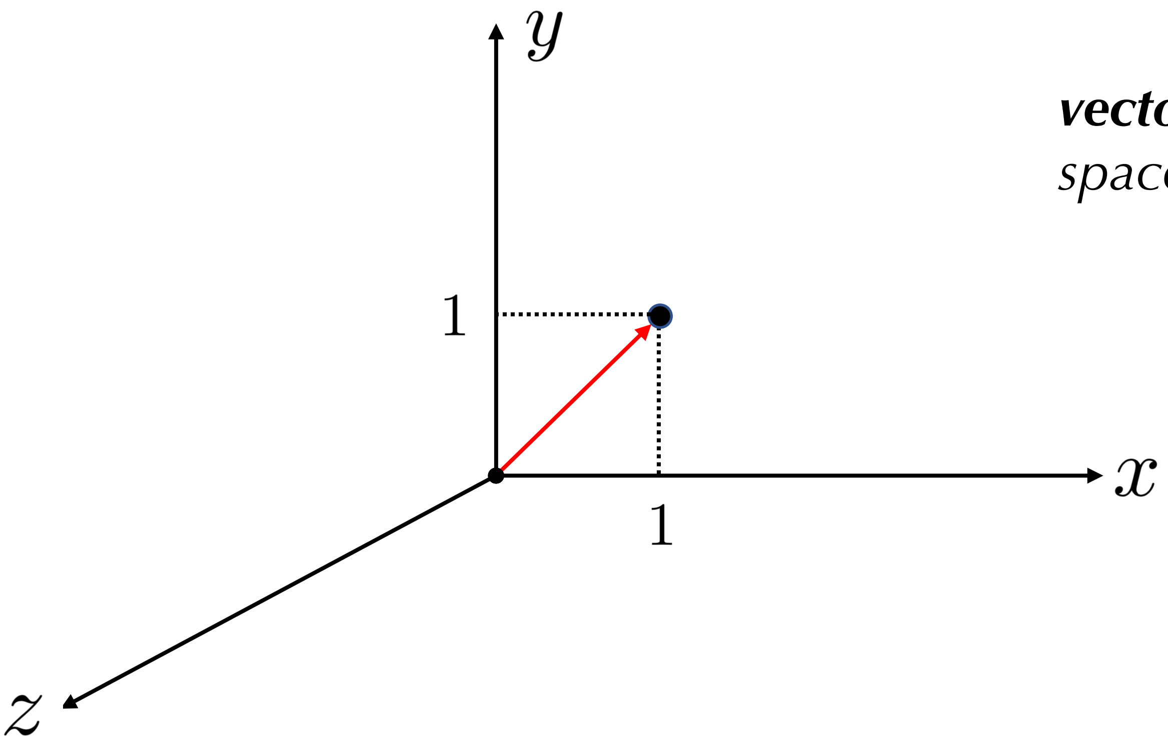
Word embeddings map words to lists of numbers (**vectors**)



vector space:
space where vectors live

Representing words

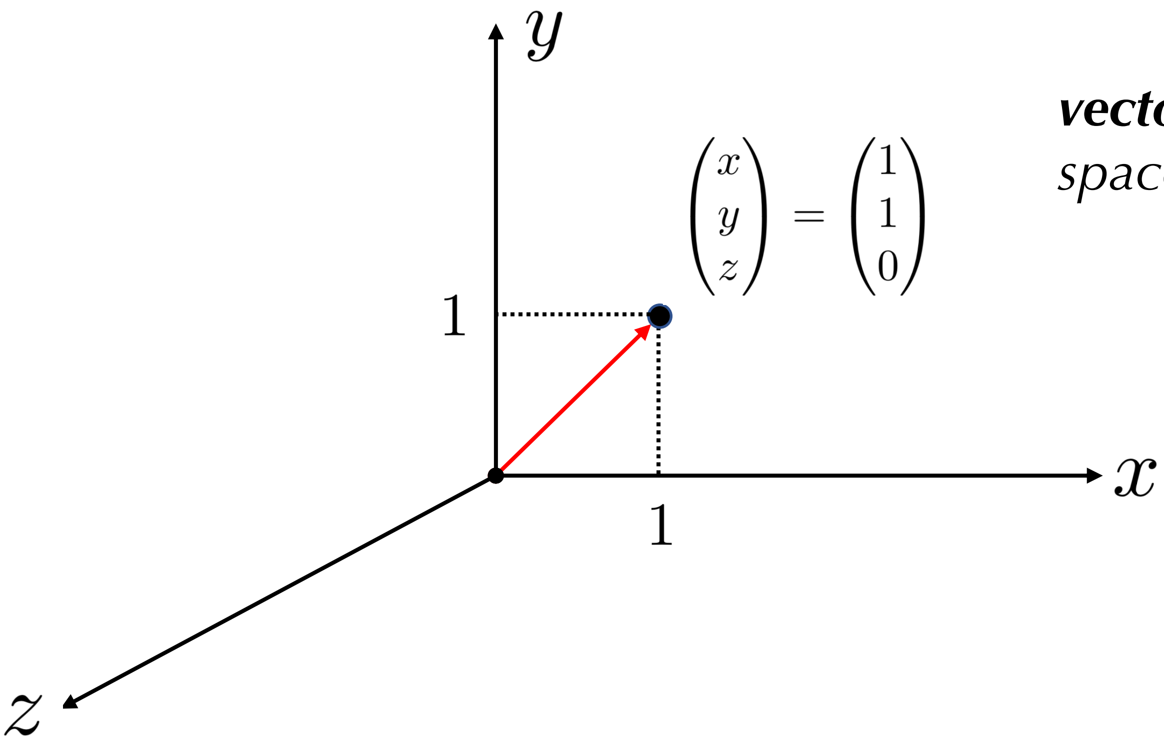
Word embeddings map words to lists of numbers (**vectors**)



vector space:
space where vectors live

Representing words

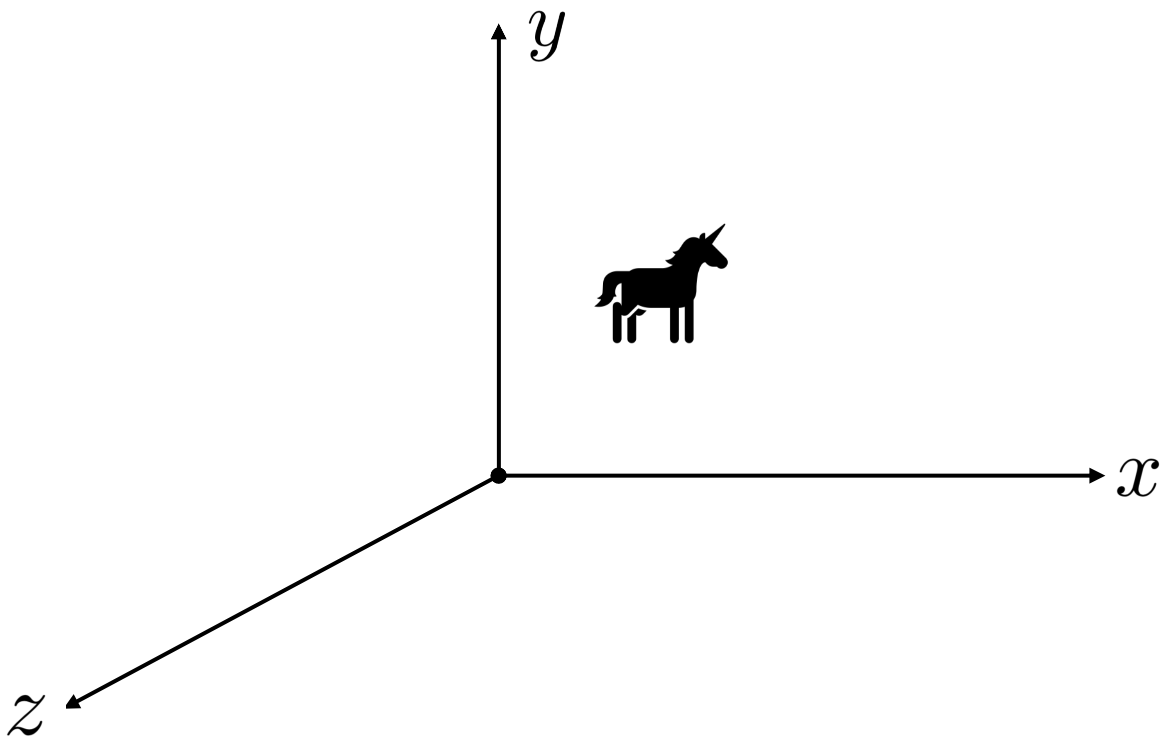
Word embeddings map words to lists of numbers (**vectors**)



vector space:
space where vectors live

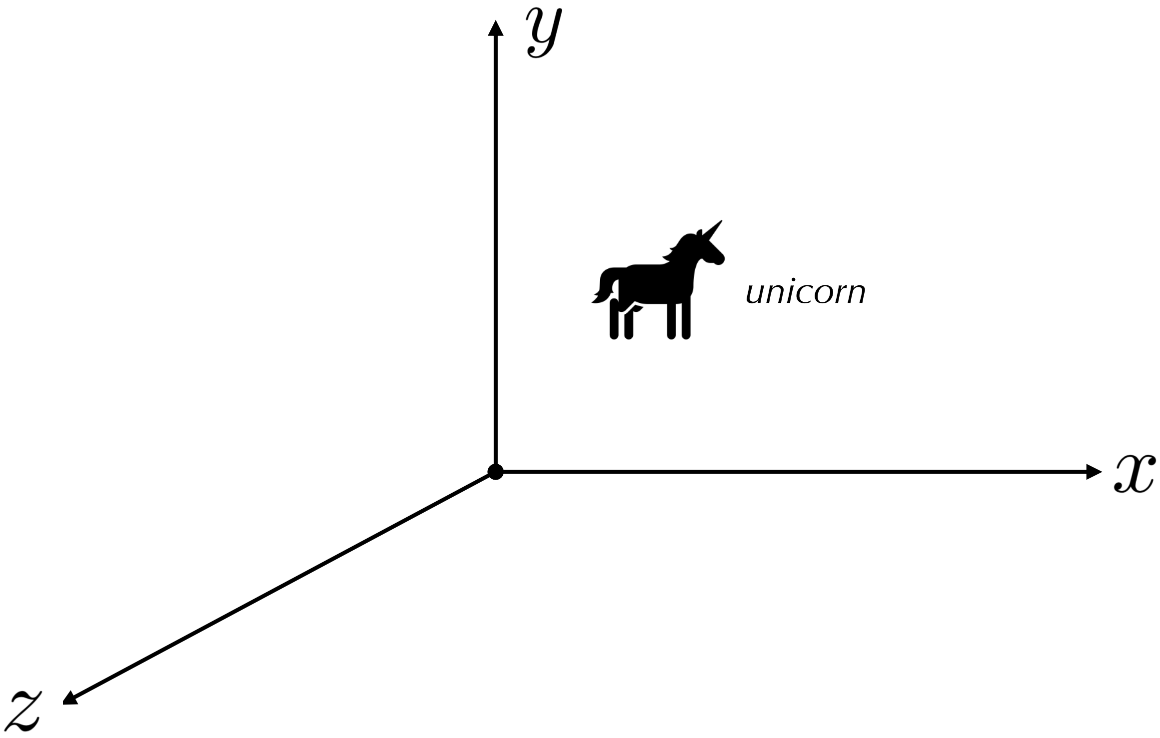
Representing words

Word embeddings map words to lists of numbers (**vectors**)



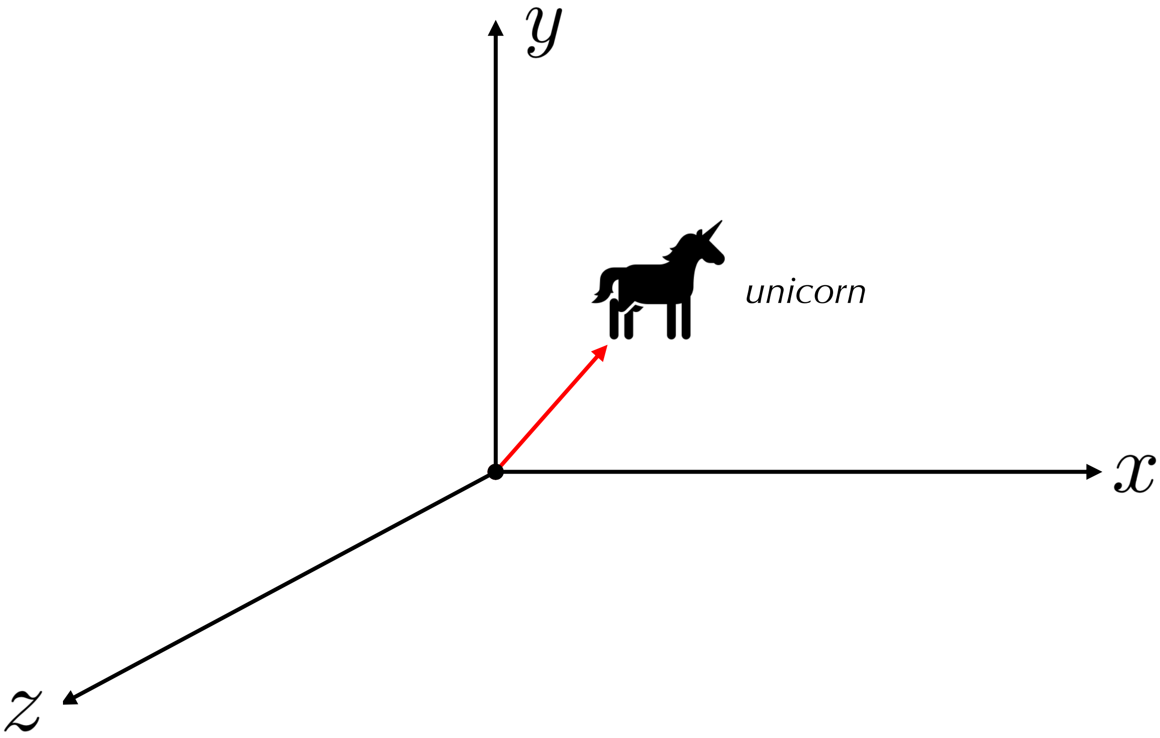
Representing words

Word embeddings map words to lists of numbers (**vectors**)



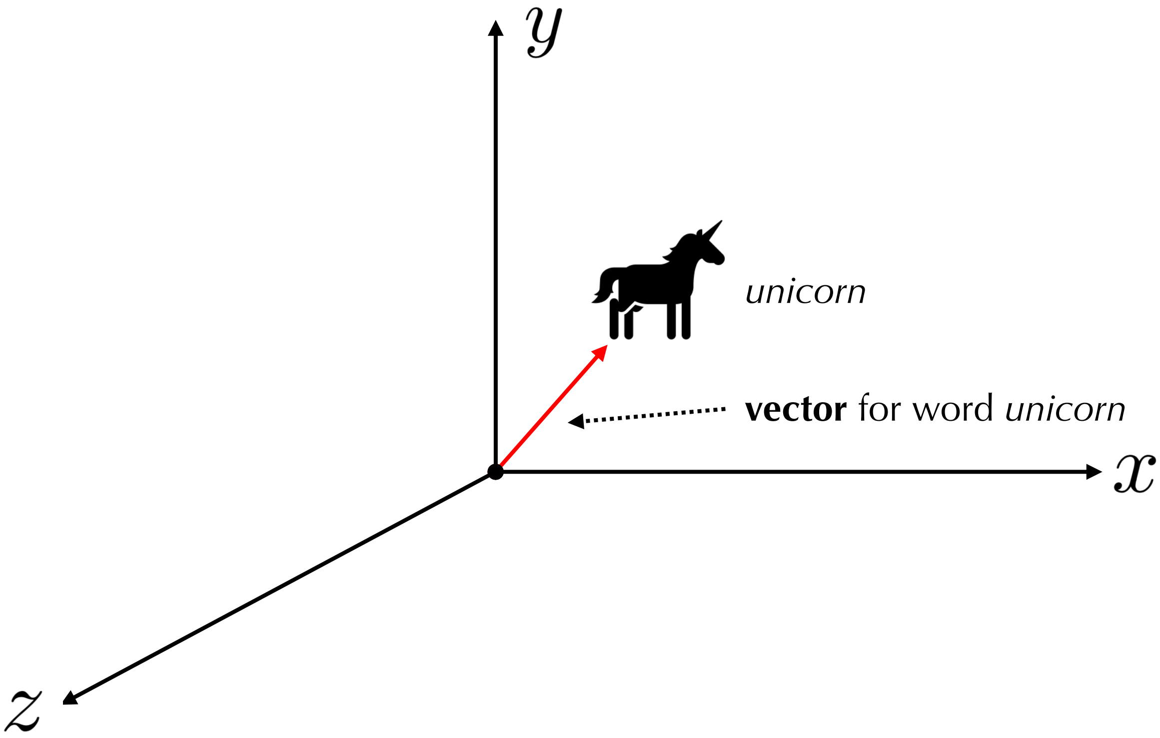
Representing words

Word embeddings map words to lists of numbers (**vectors**)



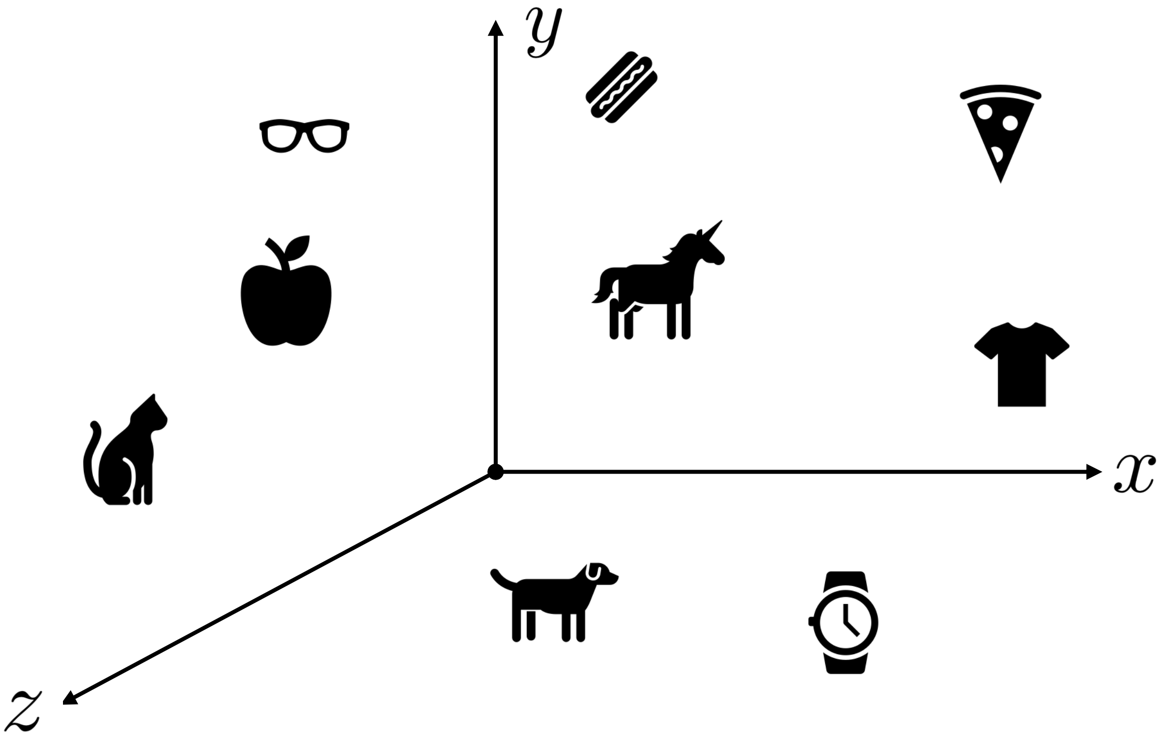
Representing words

Word embeddings map words to lists of numbers (**vectors**)



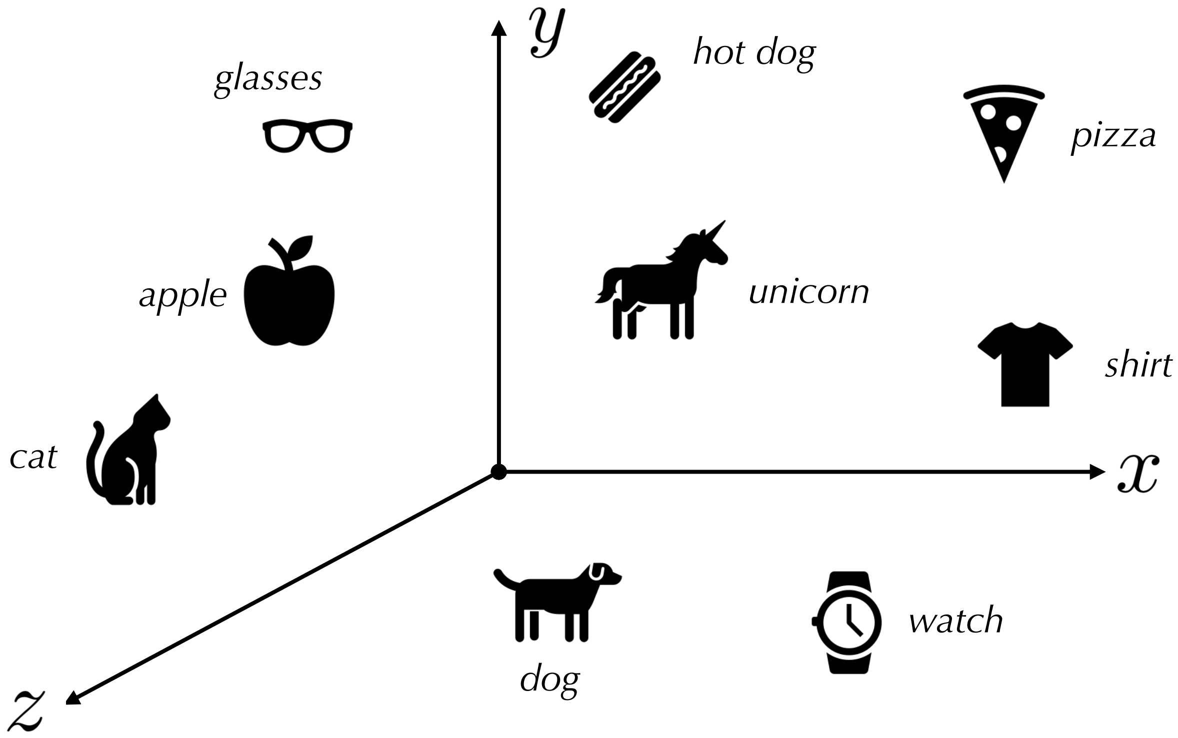
Representing words

Word embeddings map words to lists of numbers (**vectors**)



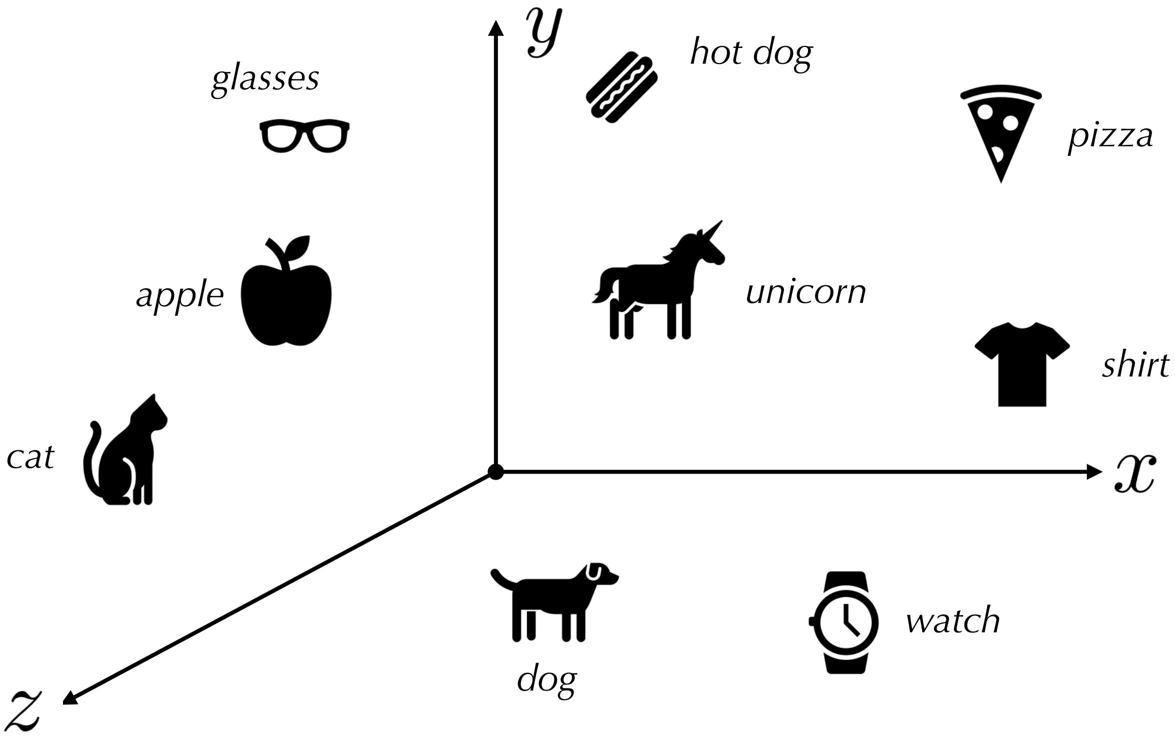
Representing words

Word embeddings map words to lists of numbers (**vectors**)



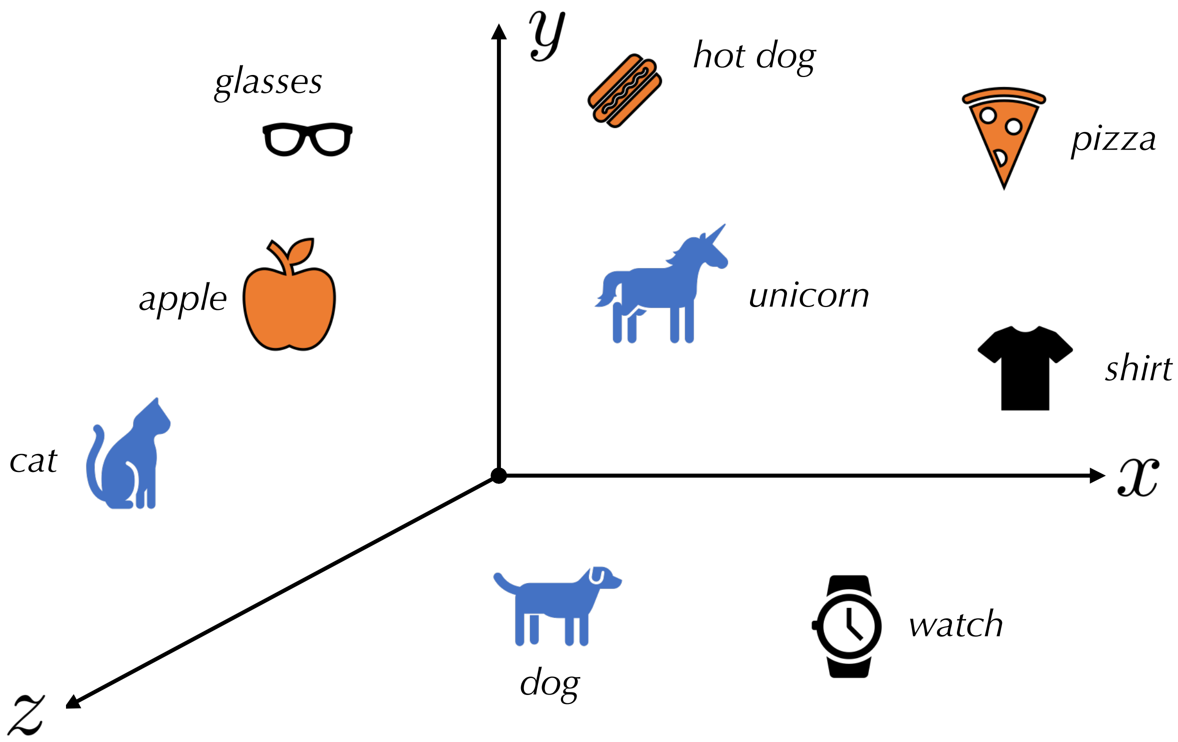
Representing words

However, different words have different semantic relations



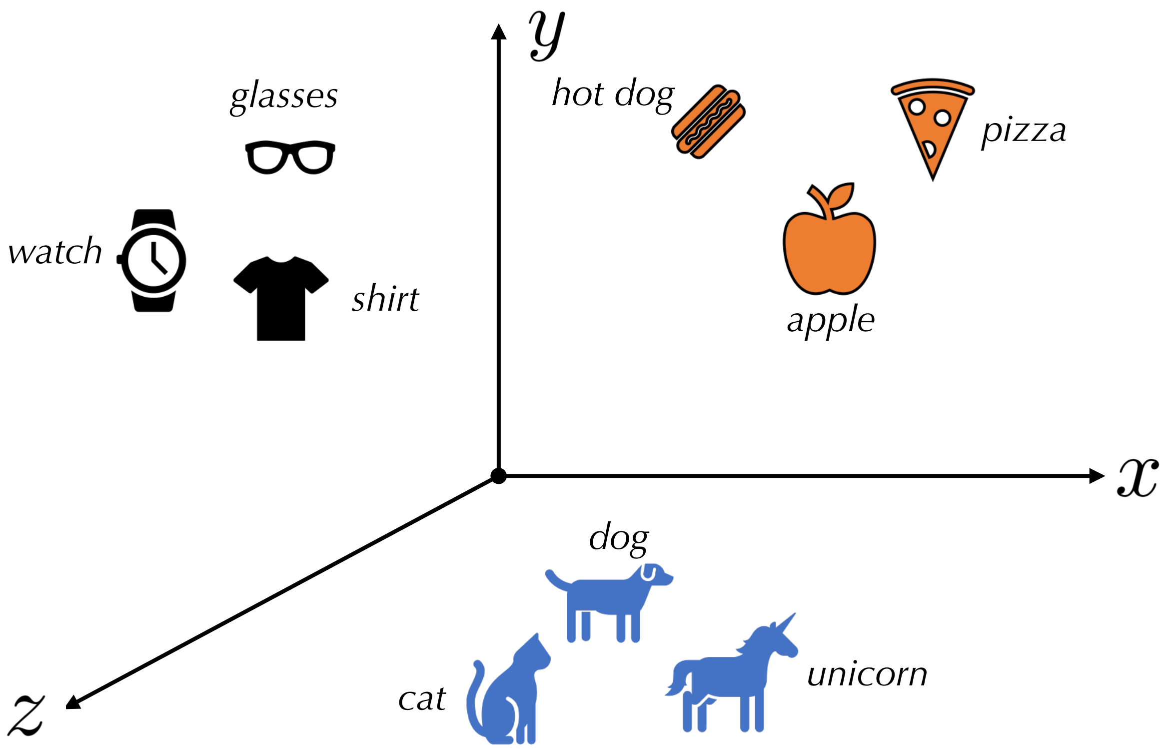
Representing words

However, different words have different semantic relations



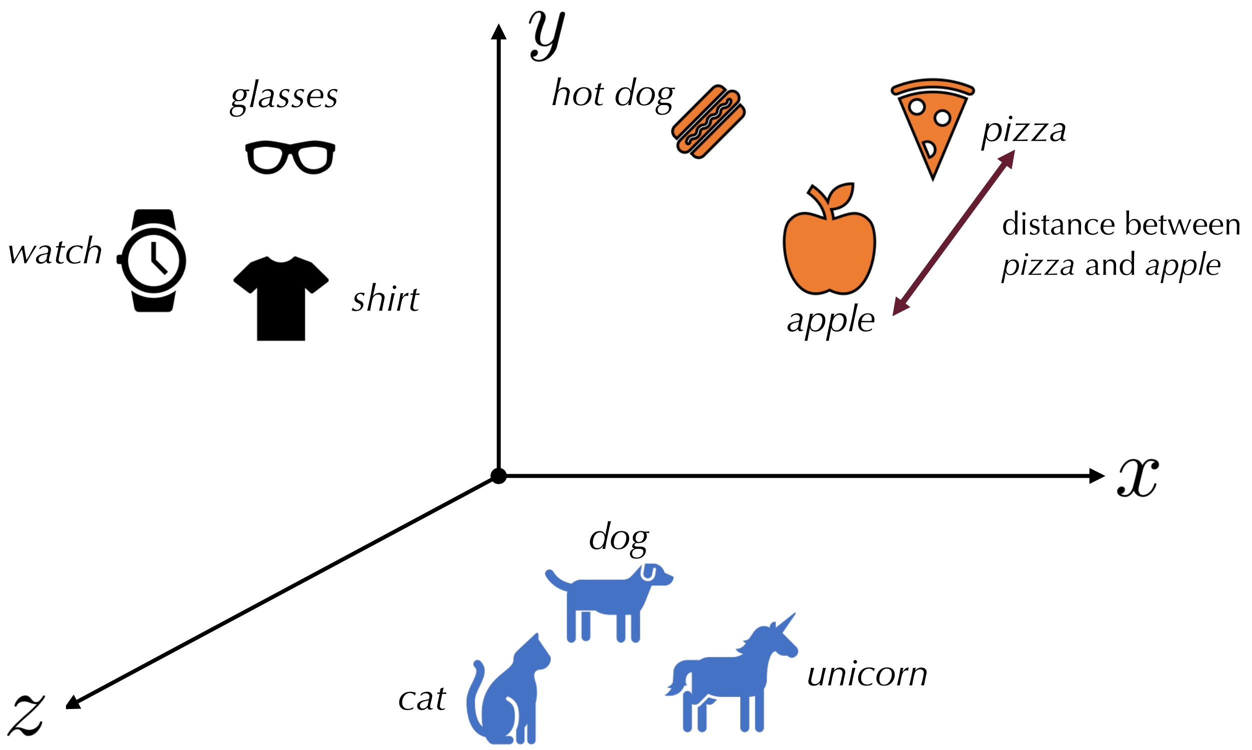
Representing words

Word embeddings aim to bring related words together in this space



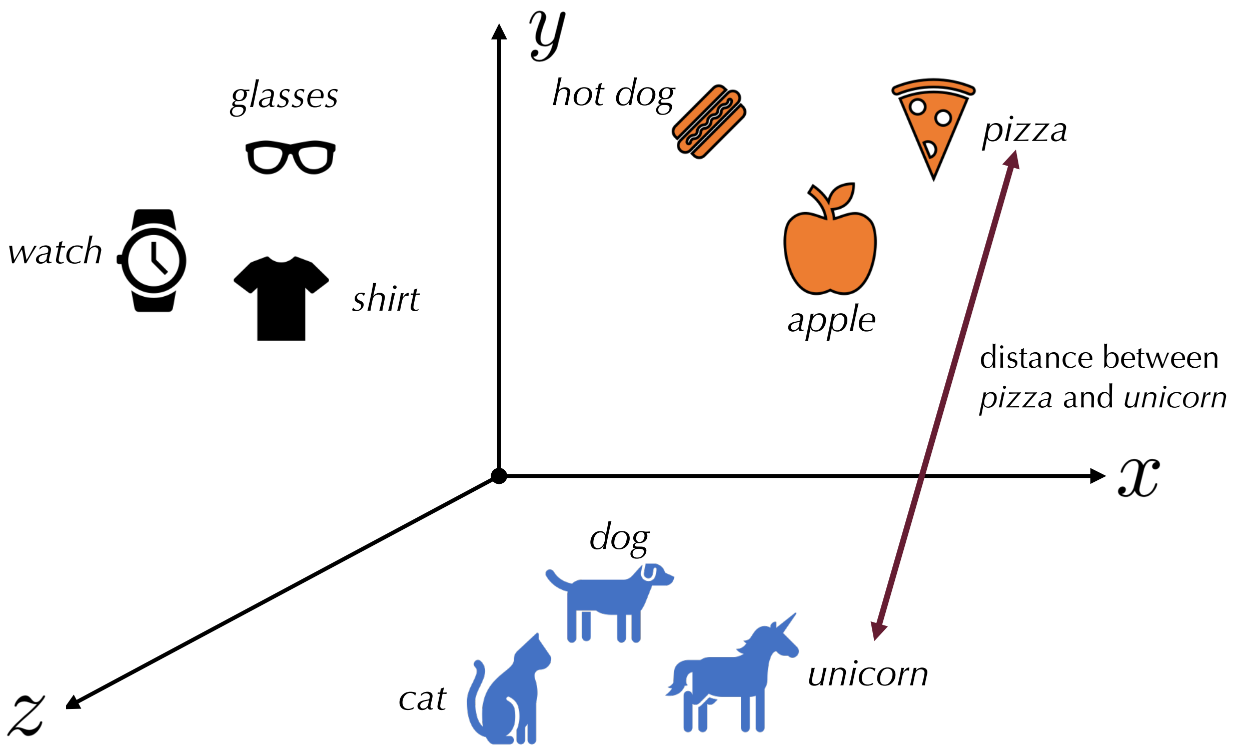
Representing words

Word embeddings aim to bring related words together in this space



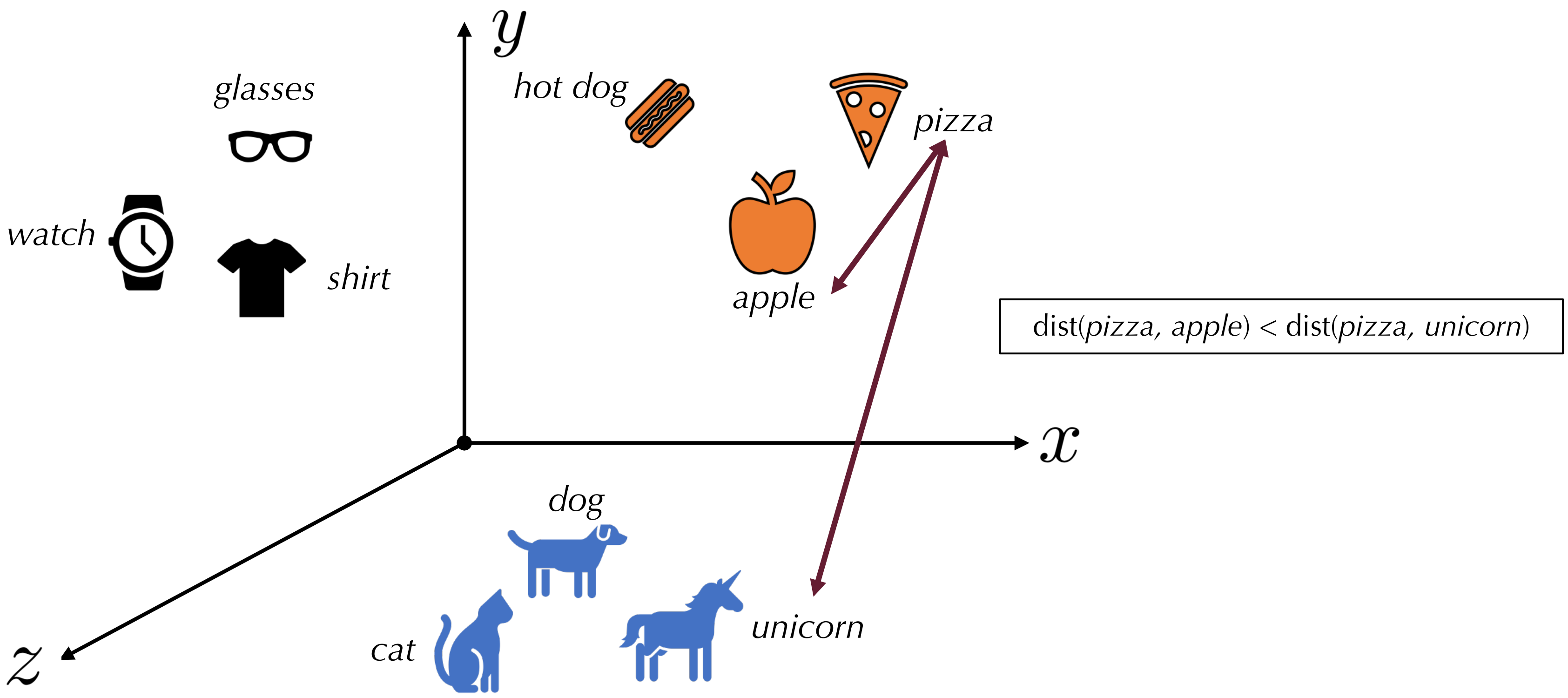
Representing words

Word embeddings aim to bring related words together in this space



Representing words

Word embeddings aim to bring related words together in this space



Terminology

- A **word** w is a concatenation of characters
- A **document** $\mathcal{D} = (w_1, \ldots, w_n)$ is a sequence of words
- The **vocabulary** V is the set of all words of consideration
- N is the **size of our vocabulary**, i.e. $|V| = N$
- A **word vector** $\mathbf{v}_w \in \mathbb{R}^D$ for a word w is an ordered list of D real numbers
- D is the **dimension** of a word vector $\mathbf{v}_w$

Representing words as vectors

- Assume we have a finite set of words $w_i \in V, i \in [N]$
- We assign each word to an N- dimensional "zero-vector" with a non-zero entry at index i

Representing words as vectors

- Assume we have a finite set of words $w_i \in V, i \in [N]$
- We assign each word to an N- dimensional "zero-vector" with a non-zero entry at index i
- Example: for word w_7 , we define a vector $\mathbf{v}_{w_7}$ such that

$$\mathbf{v}_{w_7} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}0\\ \vdots\\ 0\end{matrix}}\right\} 1-6 \\ \\ \left.\vphantom{\begin{matrix}0\\ \vdots\\ 0\end{matrix}}\right\} 8-N \end{matrix}$$

Representing documents as vectors

- If we now have an entire document $\mathcal{D} = (w_1, \ldots, w_n)$, we can compute a frequency vector $\mathbf{v}_{\mathcal{D}}$ of this document by adding the word vectors for each w_i

Representing documents as vectors

- If we now have an entire document $\mathcal{D} = (w_1, \ldots, w_n)$, we can compute a frequency vector $\mathbf{v}_{\mathcal{D}}$ of this document by adding the word vectors for each w_i
- Example: $\mathcal{D} = (\text{what}, \text{do}, \text{vectors}, \text{have}, \text{to}, \text{do}, \text{with}, \text{words})$

$V = \{\text{what}, \text{do}, \text{vectors}, \text{have}, \text{to}, \text{with}, \text{words}\}, |V| = 7$

$$\{\text{what},\ \text{do},\ \text{vectors},\ \text{have},\ \text{to},\ \text{with},\ \text{words}\}$$

1 2 3 4 5 6 7

Representing documents as vectors

- If we now have an entire document $\mathcal{D} = (w_1, \ldots, w_n)$, we can compute a frequency vector $\mathbf{v}_\mathcal{D}$ of this document by adding the word vectors for each w_i
- Example: $\mathcal{D} = (\text{what}, \text{do}, \text{vectors}, \text{have}, \text{to}, \text{do}, \text{with}, \text{words})$

$V = \{\text{what}, \text{do}, \text{vectors}, \text{have}, \text{to}, \text{with}, \text{words}\}, |V| = 7$

$$\mathbf{v}_\mathcal{D} = \sum_{i=1}^{8} \mathbf{v}_{w_i} = \begin{pmatrix}1\\0\\0\\0\\0\\0\\0\end{pmatrix} + \begin{pmatrix}0\\1\\0\\0\\0\\0\\0\end{pmatrix} + \begin{pmatrix}0\\0\\1\\0\\0\\0\\0\end{pmatrix} + \begin{pmatrix}0\\0\\0\\1\\0\\0\\0\end{pmatrix} + \begin{pmatrix}0\\0\\0\\0\\1\\0\\0\end{pmatrix} + \begin{pmatrix}0\\1\\0\\0\\0\\0\\0\end{pmatrix} + \begin{pmatrix}0\\0\\0\\0\\0\\1\\0\end{pmatrix} + \begin{pmatrix}0\\0\\0\\0\\0\\0\\1\end{pmatrix} = \begin{pmatrix}1\\2\\1\\1\\1\\1\\1\end{pmatrix}$$

Representing documents as vectors

- This is called *bag-of-words* approach
- $\mathbf{v}_{\mathcal{D}}$ now represents a term-frequency vector that can be used as a feature (e.g. for a text classifier)

Our task seems to be solved! We can efficiently model entire documents with a single vector.

There are problems…

Representing words as vectors

Problems with this approach:

1. For large N, the individual vectors become very large and inefficient
2. The word vectors are semantically **unrelated** to each other

Unrelated here means, that we cannot say anything about the semantic relationships between words by looking at their vectors.

Similarity between vectors

- In Euclidean N- space, we can measure the similarity between different vectors
- This can be done by considering the dot product $\langle \mathbf{v}, \mathbf{w} \rangle$ between two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{R}^N$, defined as

$$\langle \mathbf{v}, \mathbf{w} \rangle = \sum_{i=1}^{N} v_i \cdot w_i$$

Similarity between vectors

- In Euclidean N- space, we can measure the similarity between different vectors
- This can be done by considering the dot product $\langle \mathbf{v}, \mathbf{w} \rangle$ between two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{R}^N$, defined as

$$\langle \mathbf{v}, \mathbf{w} \rangle = \sum_{i=1}^{N} v_i \cdot w_i$$

- Geometric interpretation:

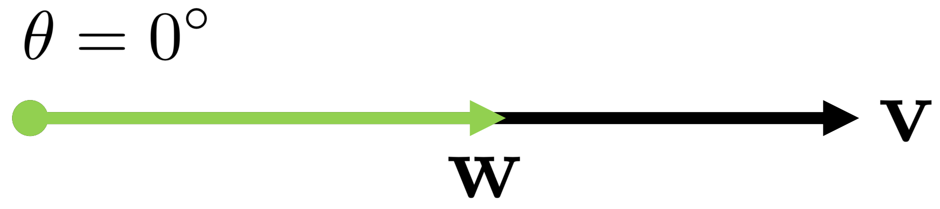
$$\cos(\theta) = \frac{\langle \mathbf{v}, \mathbf{w} \rangle}{||\mathbf{v}|| \cdot ||\mathbf{w}||} \qquad \text{hence} \qquad \langle \mathbf{v}, \mathbf{w} \rangle = \cos(\theta) \cdot ||\mathbf{v}|| \cdot ||\mathbf{w}||$$

Similarity between vectors

$$\langle \mathbf{v}, \mathbf{w} \rangle = \cos(\theta) \cdot ||\mathbf{v}|| \cdot ||\mathbf{w}||$$

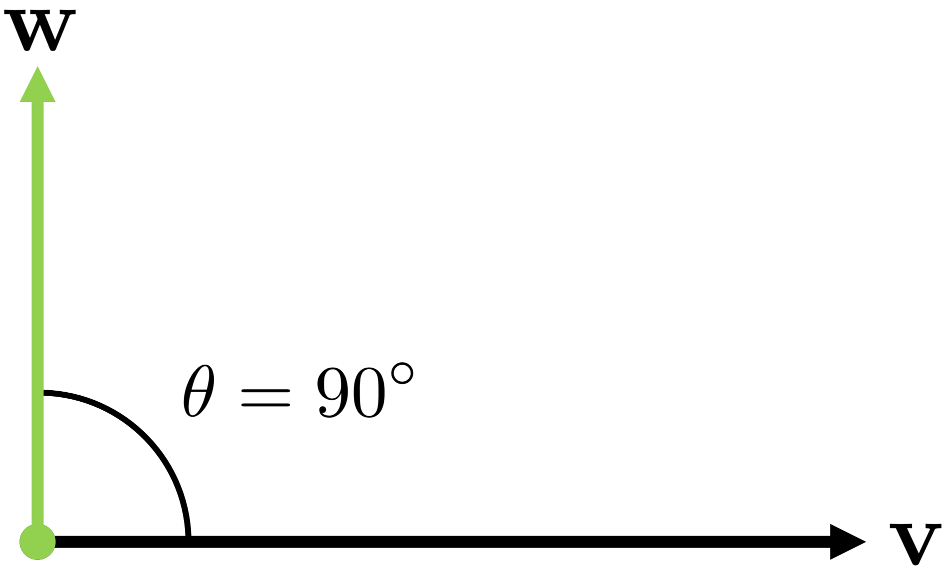
Note: $||\mathbf{v}|| = \sqrt{\sum_{i=1}^{N} v_i^2} \geq 0$

Similarity between vectors



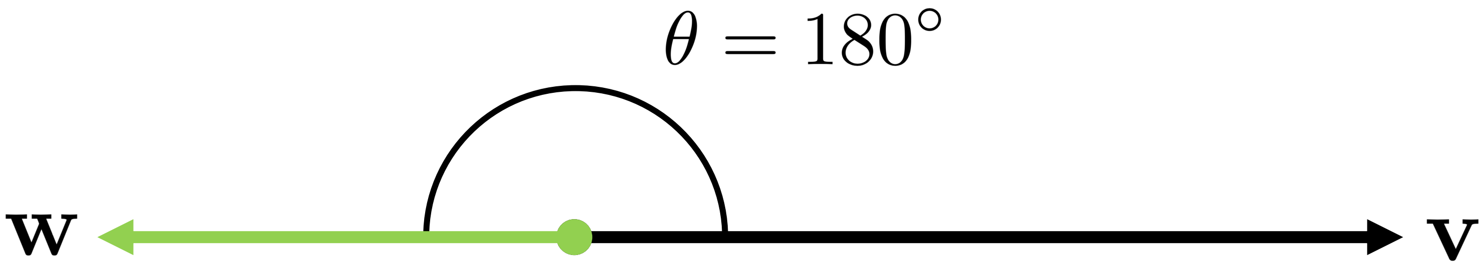
$$\langle \mathbf{v}, \mathbf{w} \rangle = \cos(\theta) \cdot ||\mathbf{v}|| \cdot ||\mathbf{w}|| = \cos(0^\circ) \cdot ||\mathbf{v}|| \cdot ||\mathbf{w}|| = 1 \cdot ||\mathbf{v}|| \cdot ||\mathbf{w}|| \geq 0$$

Similarity between vectors



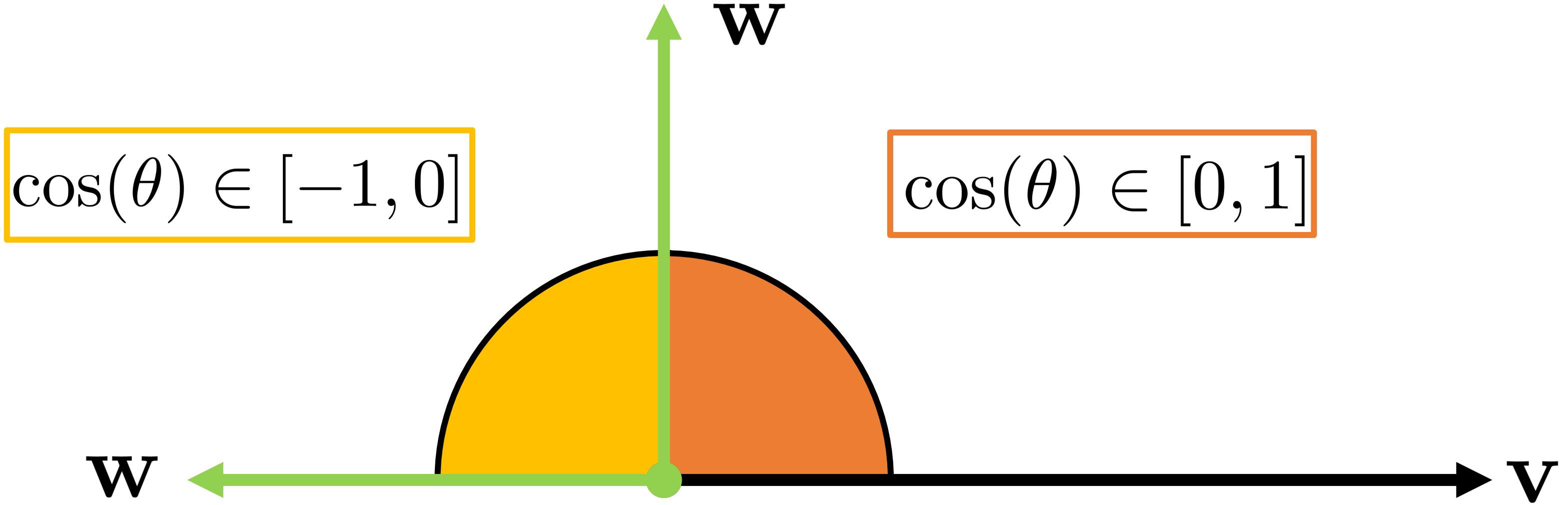
$$\langle \mathbf{v}, \mathbf{w} \rangle = \cos(\theta) \cdot ||\mathbf{v}|| \cdot ||\mathbf{w}|| = \cos(90^\circ) \cdot ||\mathbf{v}|| \cdot ||\mathbf{w}|| = 0 \cdot ||\mathbf{v}|| \cdot ||\mathbf{w}|| = 0$$

Similarity between vectors



$$\langle \mathbf{v}, \mathbf{w} \rangle = \cos(\theta) \cdot ||\mathbf{v}|| \cdot ||\mathbf{w}|| = \cos(180^\circ) \cdot ||\mathbf{v}|| \cdot ||\mathbf{w}|| = -1 \cdot ||\mathbf{v}|| \cdot ||\mathbf{w}|| \leq 0$$

Similarity between vectors



Similarity between vectors

$$\cos(\theta) \in \begin{cases} [0,1] & \text{if } 0^\circ \leq \theta \leq 90^\circ \\ [-1,0) & \text{if } 90^\circ < \theta \leq 180^\circ \\ (-1,0] & \text{if } 180^\circ < \theta \leq 270^\circ \\ (0,1] & \text{if } 270^\circ < \theta \leq 360^\circ \end{cases}$$

Similarity between vectors

The **distributional hypothesis** states that words occurring in the same context tend to have similar meanings (Harris, 1954).

Idea: use word vectors that capture the semantic similarity between words

One-hot vectors fail at this task, since the dot product for two different words is always zero!

Similarity between vectors

If we use the dot product for two different one-hot vectors, the result will always be 0!

$$\langle \mathbf{v}_{\text{vectors}}, \mathbf{v}_{\text{words}} \rangle = \left\langle \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \right\rangle = 0 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 0 \cdot 0 + 0 \cdot 0 + 0 \cdot 1 = 0$$

We need a different representation!

Word embeddings

- Enter word embeddings!
- Instead of using one-hot vectors, assign each word in the vocabulary to a dense vector representation
- If two words are semantically similar, their dot product should be high

How do we find those embeddings?

word2vec

word2vec

- Set of embedding models introduced by Mikolov et al. (2013a, 2013b)
- Given a large corpus of text, we compute a dense word embedding for each word by looking at the context in which it appears

- To do so, word2vec provides two methods:

1. **Skip-gram**: given a specific word, estimate probability of context words
2. **Continuous-bag-of-words**: given a set of context words, predict the center word

Skip-gram

Moscow is the capital of Russia, Paris is the capital of France

Skip-gram

Moscow is the capital of Russia, Paris is the capital of France

w_t

Skip-gram

w_{t-3} w_{t-1} w_{t+1} w_{t+3}

Moscow is the capital of Russia, Paris is the capital of France

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

Skip-gram

w_{t-3} w_{t-1} w_{t+1} w_{t+3}

Moscow is the capital of Russia, Paris is the capital of France

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

Maximise probability $p(\text{the, capital, of, Paris, is, the} \,|\, \text{Russia})$

Skip-gram

w_{t-3} w_{t-1} w_{t+1} w_{t+3}

Moscow is the capital of Russia, Paris is the capital of France

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

Maximise probability $p(w_{t-3}, w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}, w_{t+3} \mid w_t)$

Skip-gram

w_{t-3} w_{t-1} w_{t+1} w_{t+3}

Moscow is the capital of Russia, Paris is the capital of France

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

Maximise probability $$\prod_{-c \leq k \leq c, k \neq 0} p(w_{t+k} \mid w_t)$$

Skip-gram

w_{t-3} w_{t-1} w_{t+1} w_{t+3}

Moscow is the capital of Russia, Paris is the capital of France

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

For all words in all texts (let them be indexed by $t = 1, \ldots, T$):

Maximise objective $$\prod_{t=1}^{T} \prod_{-c \leq k \leq c, k \neq 0} p(w_{t+k} \mid w_t)$$

Skip-gram

We denote our objective with

$$J(\mathbf{W}) = \prod_{t=1}^{T} \prod_{-c \leq k \leq c, k \neq 0} p(w_{t+k} \mid w_t; \mathbf{W})$$

where $\mathbf{W}$ is a matrix of the embeddings for all words in the vocabulary.

Skip-gram

We denote our objective with

$$J(\mathbf{W}) = \prod_{t=1}^{T} \prod_{-c \le k \le c, k \ne 0} p(w_{t+k} \mid w_t; \mathbf{W})$$

where $\mathbf{W}$ is a matrix of the embeddings for all words in the vocabulary.

Since we are interested in finding the optimal values for these embeddings, we instead maximise

$$\log J(\mathbf{W}) = \sum_{t=1}^{T} \sum_{-c \le k \le c, k \ne 0} \log p(w_{t+k} \mid w_t; \mathbf{W})$$

The embedding matrix $\mathbf{W}$

- $\mathbf{W}$ is a matrix of dimensions $N \times D$ (vocab size $\times$ embedding dim)

$$\mathbf{W} = \begin{pmatrix} -18.123 & 0.981 & 0.000787 & \cdots & 1.324 \\ -0.005323 & 64.35 & 0.9013 & \cdots & 7.7534 \\ -0.002321 & 8.544 & 5.23 & \cdots & 65.75665 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 3.000434 & 4.34232 & \cdots & \cdots & 0.0002132 \end{pmatrix}$$

The embedding matrix $\mathbf{W}$

- $\mathbf{W}$ is a matrix of dimensions $N \times D$ (vocab size $\times$ embedding dim)

$$
\mathbf{W} = \overbrace{\begin{pmatrix} -18.123 & 0.981 & 0.000787 & \cdots & 1.324 \\ -0.005323 & 64.35 & 0.9013 & \cdots & 7.7534 \\ -0.002321 & 8.544 & 5.23 & \cdots & 65.75665 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 3.000434 & 4.34232 & \cdots & \cdots & 0.0002132 \end{pmatrix}}^{D} \left.\vphantom{\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}}\right\} N
$$

The embedding matrix $\mathbf{W}$

- $\mathbf{W}$ is a matrix of dimensions $N \times D$ (vocab size $\times$ embedding dim)

$$\mathbf{W} = \begin{array}{cl} \begin{pmatrix} -18.123 & 0.981 & 0.000787 & \cdots & 1.324 \\ -0.005323 & 64.35 & 0.9013 & \cdots & 7.7534 \\ -0.002321 & 8.544 & 5.23 & \cdots & 65.75665 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 3.000434 & 4.34232 & \cdots & \cdots & 0.0002132 \end{pmatrix} & \begin{array}{l} \textit{Russia} \\ \\ \\ \\ \\ \end{array} \end{array}$$

The embedding matrix $\mathbf{W}$

- $\mathbf{W}$ is a matrix of dimensions $N \times D$ (vocab size $\times$ embedding dim)

$$\mathbf{W} = \begin{pmatrix} -18.123 & 0.981 & 0.000787 & \cdots & 1.324 \\ \boxed{-0.005323 & 64.35 & 0.9013 & \cdots & 7.7534} \\ -0.002321 & 8.544 & 5.23 & \cdots & 65.75665 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 3.000434 & 4.34232 & \cdots & \cdots & 0.0002132 \end{pmatrix}$$

France

The embedding matrix $\mathbf{W}$

- $\mathbf{W}$ is a matrix of dimensions $N \times D$ (vocab size $\times$ embedding dim)

$$
\mathbf{W} = \begin{pmatrix}
-18.123 & 0.981 & 0.000787 & \cdots & 1.324 \\
-0.005323 & 64.35 & 0.9013 & \cdots & 7.7534 \\
\boxed{-0.002321 \quad 8.544 \quad 5.23 \quad \cdots \quad 65.75665} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
3.000434 & 4.34232 & \cdots & \cdots & 0.0002132
\end{pmatrix} \quad \textit{capital}
$$

The embedding matrix $\mathbf{W}$

- $\mathbf{W}$ is a matrix of dimensions $N \times D$ (vocab size $\times$ embedding dim)

$$\mathbf{W} = \begin{array}{cl} \begin{pmatrix} -18.123 & 0.981 & 0.000787 & \cdots & 1.324 \\ -0.005323 & 64.35 & 0.9013 & \cdots & 7.7534 \\ -0.002321 & 8.544 & 5.23 & \cdots & 65.75665 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 3.000434 & 4.34232 & \cdots & \cdots & 0.0002132 \end{pmatrix} & \begin{array}{l} \\ \\ \\ \\ \textit{Moscow} \end{array} \end{array}$$

Retrieving a vector from $\mathbf{W}$

- Recall our N- dimensional one-hot vector
- For *Russia*, this would be $(1, 0, \ldots, 0)^\top \in \{0, 1\}^N$

Retrieving a vector from $\mathbf{W}$

- Recall our N- dimensional one-hot vector
- For *Russia*, this would be $(1, 0, \ldots, 0)^\top \in \{0, 1\}^N$
- Hence

$$
\begin{aligned}
\mathbf{W}^\top \cdot (1, 0, ..., 0)^\top &= \begin{pmatrix} -18.123 & 0.981 & 0.000787 & \cdots & 1.324 \\ -0.005323 & 64.35 & 0.9013 & \cdots & 7.7534 \\ -0.002321 & 8.544 & 5.23 & \cdots & 65.75665 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 3.000434 & 4.34232 & \cdots & \cdots & 0.0002132 \end{pmatrix}^\top \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} -18.123 \\ 0.981 \\ 0.000787 \\ \vdots \\ 1.324 \end{pmatrix}
\end{aligned}
$$

Retrieving a vector from $\mathbf{W}$

- Recall our N- dimensional one-hot vector
- For *Russia*, this would be $(1, 0, \ldots, 0)^\top \in \{0, 1\}^N$
- Hence

$$\mathbf{W}^\top \cdot (1, 0, ..., 0)^\top = \begin{pmatrix} -18.123 & 0.981 & 0.000787 & \cdots & 1.324 \\ -0.005323 & 64.35 & 0.9013 & \cdots & 7.7534 \\ -0.002321 & 8.544 & 5.23 & \cdots & 65.75665 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 3.000434 & 4.34232 & \cdots & \cdots & 0.0002132 \end{pmatrix}^\top \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} -18.123 \\ 0.981 \\ 0.000787 \\ \vdots \\ 1.324 \end{pmatrix}$$

This is our word embedding for *Russia*

Modelling probabilities

- Initially, the system assigns each word in the vocabulary a random vector
- For words w_t, w_{t+k}, vectors are denoted with $\mathbf{v}_{w_{t+k}}, \mathbf{v}'_{w_t}$

Modelling probabilities

- Initially, the system assigns each word in the vocabulary a random vector
- For words w_t, w_{t+k}, vectors are denoted with $\mathbf{v}_{w_{t+k}}, \mathbf{v}'_{w_t}$
- Then

$$p(w_{t+k} \mid w_t; \mathbf{W}) = \frac{\exp(\langle \mathbf{v}_{w_{t+k}}, \mathbf{v}'_{w_t} \rangle)}{\sum_{j=1}^{N} \exp(\langle \mathbf{v}_{w_j}, \mathbf{v}'_{w_t} \rangle)}$$

- Note: computationally very expensive (for large N)

Modelling probabilities

- Initially, the system assigns each word in the vocabulary a random vector
- For words w_t, w_{t+k}, vectors are denoted with $\mathbf{v}_{w_{t+k}}, \mathbf{v}'_{w_t}$
- Then

$$p(w_{t+k} \mid w_t; \mathbf{W}) = \frac{\exp(\langle \mathbf{v}_{w_{t+k}}, \mathbf{v}'_{w_t} \rangle)}{\sum_{j=1}^{N} \exp(\langle \mathbf{v}_{w_j}, \mathbf{v}'_{w_t} \rangle)}$$

softmax function

- Note: computationally very expensive (for large N)

Putting it all together

- We aim to maximise

$$\log J(\mathbf{W}) = \sum_{t=1}^{T} \sum_{-c \leq k \leq c, k \neq 0} \log \left(\frac{\exp(\langle \mathbf{v}_{w_{t+k}}, \mathbf{v}'_{w_t} \rangle)}{\sum_{j=1}^{N} \exp(\langle \mathbf{v}_{w_j}, \mathbf{v}'_{w_t} \rangle)} \right)$$

- Since this quickly becomes intractable, this optimisation does not find many applications

How wo we train word2vec?

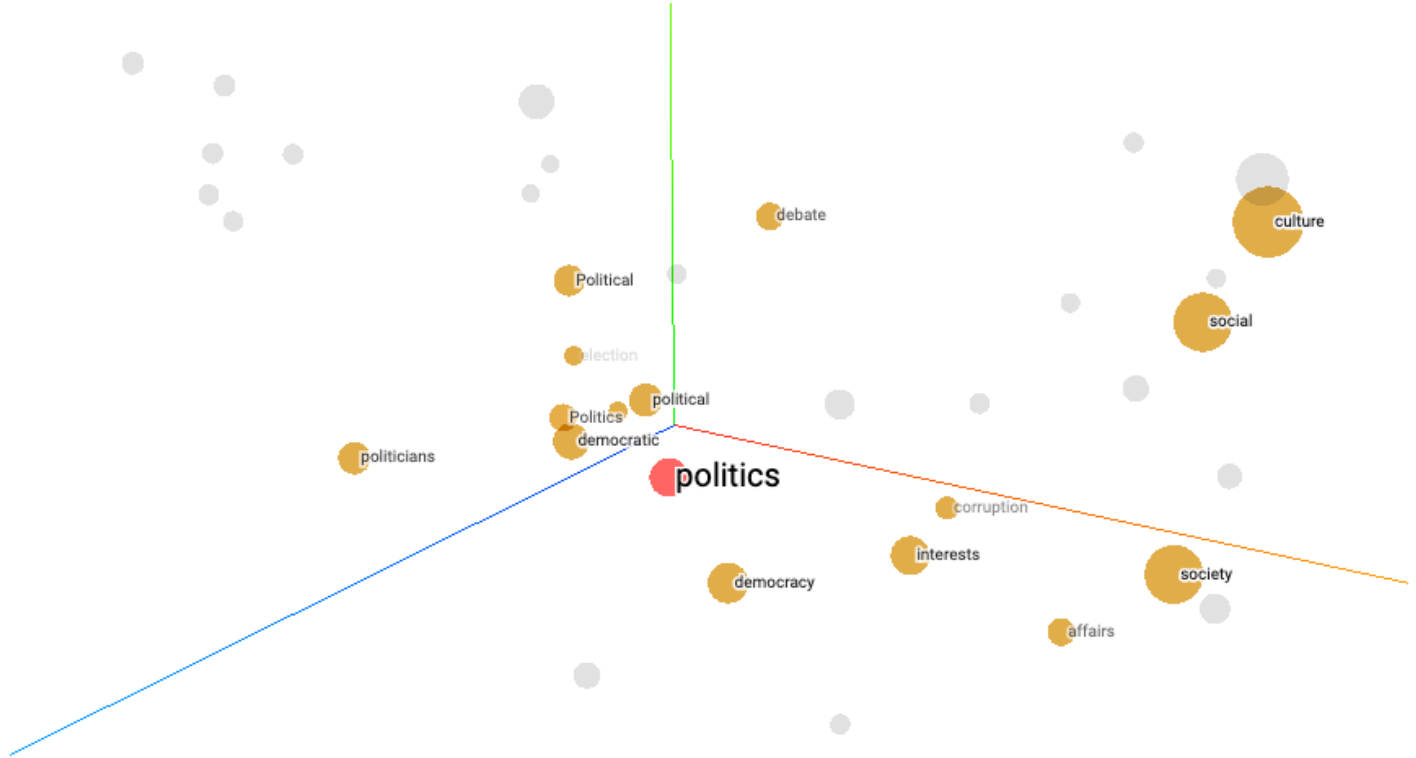
Training wor2vec models

- Mikolov et al. (2013b) propose two approximative training methods for word2vec

1. **Hierarchical Softmax**: computes an approximative softmax instead of the full one
2. **Negative Sampling**: approximation of the real objective with a logistic regression model to differentiate target words from noise words

(We will not discuss these techniques in detail here.)

Visualisation



Nearest neighbours for word **politics** (more visualisations in practical session)

Qualitative results

Nearest neighbours

Word	1	2	3	4	5
dog	*dogs*	*puppy*	*pit_bull*	*pooch*	*cat*
cat	*cats*	*dog*	*kitten*	*feline*	*beagle*
computer	*computers*	*laptop*	*laptop_computer*	*Computer*	*com_puter*
russia	*russians*	*russian*	*korea*	*germany*	*ukraine*
france	*spain*	*french*	*germany*	*europe*	*italy*
paris	*heidi*	*london*	*france*	*dubai*	*samuel*
moscow	*russian*	*russia*	*norway*	*iranian*	*munich*

Continuous-bag-of-words (CBOW)

- Similar approach, but predict center word instead, i.e.

$$p(w_t \,|\, w_{t-3}, w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}, w_{t+3})$$

- Order of context words irrelevant
- Computationally more efficient

Remarks

- word2vec learns two vector representations for each word (one for center, one for context)
- Multiple options to obtain single embedding for each word:
    1. Just use center word representation
    2. Add both embedding representations up
    3. Concatenate both vectors

- Model parameters:
    - dimension of embedding
    - context size
    - Initialisation of embedding vectors

GloVe

GloVe

- Word embedding model proposed by Pennington et al. (2014)
- Makes use of word co-occurrences in a given corpus
- Recall that we can represent documents with word frequency vectors
- Likewise, we can represent word co-occurrences with word co-occurrence matrices
- Each row and column corresponds to a word in the vocabulary
- Each cell entry i, j denotes how often word i co-occurs with word j

Word co-occurrences

- Let N denote the vocabulary size
- We define a co-occurrence matrix $\mathbf{M}_c \in (\mathbb{N} \cup \{0\})^{N \times N}$ such that

$$\mathbf{M}_c[i, j] = \text{number of occurrences of word } j \text{ in the context } c \text{ of word } i$$

- Context c is a hyperparameter of the model

Word co-occurrences

- Example:

$$\mathcal{D} = (\text{what}, \text{do}, \text{vectors}, \text{have}, \text{to}, \text{do}, \text{with}, \text{words})$$

$$V = \{\text{what}, \text{do}, \text{vectors}, \text{have}, \text{to}, \text{with}, \text{words}\}, |V| = 7$$

$$\mathbf{M}_3 = \begin{array}{c} \begin{array}{ccccccc} \textit{what} & \textit{do} & \textit{vectors} & \textit{have} & \textit{to} & \textit{with} & \textit{words} \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 2 & 2 & 2 & 1 & 1 \\ 1 & 2 & 0 & 1 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 & 1 & 1 & 0 \\ 0 & 2 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array} \begin{array}{l} \\ \textit{what} \\ \textit{do} \\ \textit{vectors} \\ \textit{have} \\ \textit{to} \\ \textit{with} \\ \textit{words} \end{array}$$

GloVe

- Idea: align semantic similarity of words with their observed co-occurrence frequencies
- Compare all words in vocabulary with each other
- Minimise difference between their dot product and their co-occurrence

GloVe

- Idea: align semantic similarity of words with their observed co-occurrence frequencies
- Compare all words in vocabulary with each other
- Minimise difference between their dot product and their co-occurrence
- Objective then is

$$\langle \mathbf{v}_i, \mathbf{v}'_j \rangle \qquad\qquad - \log\left(\mathbf{M}_c[i, j]\right)$$

GloVe

- Idea: align semantic similarity of words with their observed co-occurrence frequencies
- Compare all words in vocabulary with each other
- Minimise difference between their dot product and their co-occurrence
- Objective then is

$$\langle \mathbf{v}_i, \mathbf{v}'_j \rangle + b_i + b'_j - \log\left(\mathbf{M}_c[i,j]\right)$$

GloVe

- Idea: align semantic similarity of words with their observed co-occurrence frequencies
- Compare all words in vocabulary with each other
- Minimise difference between their dot product and their co-occurrence
- Objective then is

$$\left(\langle \mathbf{v}_i, \mathbf{v}'_j \rangle + b_i + b'_j - \log\left(\mathbf{M}_c[i,j]\right)\right)^2$$

GloVe

- Idea: align semantic similarity of words with their observed co-occurrence frequencies
- Compare all words in vocabulary with each other
- Minimise difference between their dot product and their co-occurrence
- Objective then is

$$f(\mathbf{M}_c[i,j])\left(\langle \mathbf{v}_i, \mathbf{v}'_j\rangle + b_i + b'_j - \log\left(\mathbf{M}_c[i,j]\right)\right)^2$$

GloVe

- Idea: align semantic similarity of words with their observed co-occurrence frequencies
- Compare all words in vocabulary with each other
- Minimise difference between their dot product and their co-occurrence
- Objective then is

$$\sum_{j=1}^{N} f(\mathbf{M}_c[i,j]) \left(\langle \mathbf{v}_i, \mathbf{v}'_j \rangle + b_i + b'_j - \log\left(\mathbf{M}_c[i,j]\right)\right)^2$$

GloVe

- Idea: align semantic similarity of words with their observed co-occurrence frequencies
- Compare all words in vocabulary with each other
- Minimise difference between their dot product and their co-occurrence
- Objective then is

$$\sum_{i=1}^{N}\sum_{j=1}^{N} f(\mathbf{M}_c[i,j])\left(\langle \mathbf{v}_i, \mathbf{v}'_j\rangle + b_i + b'_j - \log\left(\mathbf{M}_c[i,j]\right)\right)^2$$

GloVe

- Idea: align semantic similarity of words with their observed co-occurrence frequencies
- Compare all words in vocabulary with each other
- Minimise difference between their dot product and their co-occurrence
- Objective then is

$$J(\mathbf{W}) = \sum_{i=1}^{N} \sum_{j=1}^{N} f(\mathbf{M}_c[i,j]) \left(\langle \mathbf{v}_i, \mathbf{v}'_j \rangle + b_i + b'_j - \log\left(\mathbf{M}_c[i,j]\right) \right)^2$$

GloVe

- Idea: align semantic similarity of words with their observed co-occurrence frequencies
- Compare all words in vocabulary with each other
- Minimise difference between their dot product and their co-occurrence
- Objective then is

$$J(\mathbf{W}) = \sum_{i=1}^{N} \sum_{j=1}^{N} \boxed{f(\mathbf{M}_c[i,j])} \left(\langle \mathbf{v}_i, \mathbf{v}'_j \rangle + b_i + b'_j - \log\left(\mathbf{M}_c[i,j]\right) \right)^2$$

Weighting function

GloVe

- Idea: align semantic similarity of words with their observed co-occurrence frequencies
- Compare all words in vocabulary with each other
- Minimise difference between their dot product and their co-occurrence
- Objective then is

$$J(\mathbf{W}) = \sum_{i=1}^{N} \sum_{j=1}^{N} f(\mathbf{M}_c[i,j]) \left(\boxed{\langle \mathbf{v}_i, \mathbf{v}'_j \rangle} + b_i + b'_j - \log\left(\mathbf{M}_c[i,j]\right)\right)^2$$

Dot product

GloVe

- Idea: align semantic similarity of words with their observed co-occurrence frequencies
- Compare all words in vocabulary with each other
- Minimise difference between their dot product and their co-occurrence
- Objective then is

$$J(\mathbf{W}) = \sum_{i=1}^{N}\sum_{j=1}^{N} f(\mathbf{M}_c[i,j]) \left(\langle \mathbf{v}_i, \mathbf{v}'_j \rangle + \underbrace{\boxed{b_i + b'_j}}_{\text{Bias terms}} - \log\left(\mathbf{M}_c[i,j]\right)\right)^2$$

GloVe

- Idea: align semantic similarity of words with their observed co-occurrence frequencies
- Compare all words in vocabulary with each other
- Minimise difference between their dot product and their co-occurrence
- Objective then is

$$J(\mathbf{W}) = \sum_{i=1}^{N} \sum_{j=1}^{N} f(\mathbf{M}_c[i,j]) \left(\langle \mathbf{v}_i, \mathbf{v}'_j \rangle + b_i + b'_j - \boxed{\log\left(\mathbf{M}_c[i,j]\right)} \right)^2$$

Word co-occurrence frequency

GloVe

- Idea: align semantic similarity of words with their observed co-occurrence frequencies
- Compare all words in vocabulary with each other
- Minimise difference between their dot product and their co-occurrence
- Objective then is

$$J(\mathbf{W}) = \sum_{i=1}^{N} \sum_{j=1}^{N} f(\mathbf{M}_c[i,j]) \left(\langle \mathbf{v}_i, \mathbf{v}'_j \rangle + b_i + b'_j - \boxed{\log\left(\mathbf{M}_c[i,j]\right)} \right)^2$$

Note: not defined for $\mathbf{M}_c[i,j] = 0\,!$

In practice, shift $\log(\mathbf{M}_c[i,j]) \rightarrow \log(1 + \mathbf{M}_c[i,j])$

Weighting function $f(\cdot)$

Pennington et al. (2014) propose the following characteristics for $f(\cdot)$:

1. $f(0) = 0$, i.e. if two words do not co-occur, we neglect them
2. The function should be non-decreasing
3. The function should not overweight frequent co-occurrences

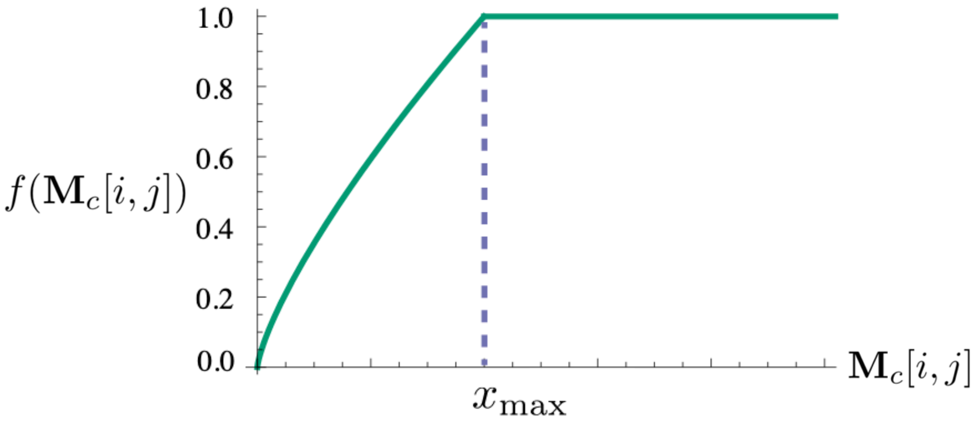
Weighting function $f(\cdot)$

Pennington et al. (2014) propose the following characteristics for $f(\cdot)$:

1. $f(0) = 0$, i.e. if two words do not co-occur, we neglect them
2. The function should be non-decreasing
3. The function should not overweight frequent co-occurrences

Author's choice:

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



Source: Pennington et al. (2014)

Qualitative results

Nearest neighbours

Word	1	2	3	4	5
dog	*dogs*	*puppy*	*pet*	*cat*	*puppies*
cat	*cats*	*kitten*	*dog*	*kitty*	*pet*
computer	*computers*	*Computer*	*laptop*	*software*	*desktop*
russia	*ukraine*	*russian*	*moscow*	*poland*	*soviet*
france	*paris*	*europe*	*italy*	*spain*	*germany*
paris	*france*	*hilton*	*montreal*	*lyon*	*prague*
moscow	*russia*	*prague*	*ukraine*	*poland*	*kiev*

Remarks

- GloVe also computes two vectors per word
- The final representation can be obtained similar to word2vec (addition, concatenation)

- Model parameters:
  - dimension of embedding
  - context size
  - initialisation of embedding vectors
  - (…)

word2vec vs. GloVe

word2vec	GloVe
• Aims to capture local contexts many different times • Computationally intractable (requires approximative methods)	• Aims to capture global context once • Computationally feasible

References

- Harris, Z.S., 1954. Distributional structure. Word, 10(2-3), pp.146-162.
- Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013a. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).
- Pennington, J., Socher, R. and Manning, C., 2014. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

Recommended resources

- Lecture video "Word Vector Representations: word2vec" by Christopher Manning. Available at **`https://www.youtube.com/watch?v=ERibwqs9p38`**.
- Lecture video "GloVe: Global Vectors for Word Representation" by Richard Socher. Available at **`https://www.youtube.com/watch?v=ASn7ExxLZws`**.
- Textbook "Speech and Language Processing" by Dan Jurafsky. Available at **`https://web.stanford.edu/~jurafsky/slp3/`**.

These slides are inspired by the above resources.

Thank you

Any questions?

Next up: *Applications and Limitations of Word Embeddings in CSS*

(Laura, Bennett)