Word Embeddings

Einführung in die Informationsverarbeitung

Nils Reiter

January 26, 2023

INSTITUT FÜR
DIGITAL HUMANITIES
UNIVERSITÄT ZU KÖLN

Section 1

Word2Vec

Introduction

- Embeddings: Words are *embedded* into a high-dimensional vector space
  - (and not simply indexed any more)
- Word2Vec
  - A method to represent words in a (high-dimensional) vector space
  - No end-user task

Introduction

- Embeddings: Words are *embedded* into a high-dimensional vector space
  - (and not simply indexed any more)
- Word2Vec
  - A method to represent words in a (high-dimensional) vector space
  - No end-user task
- A vector representation for “köln”

```
0.0539 -0.0030 0.0203 -0.1084 -0.0099 0.0705 -0.0546 -0.0433 -0.0096 0.0561 -0.0095 0.0280 0.1726 0.0190 0.0369 0.0217 -0.0002
-0.0309 0.0347 -0.0749 -0.0202 0.0151 -0.0195 0.0001 0.0232 0.0243 -0.0170 -0.0090 -0.0108 -0.0943 0.0376 0.1118 -0.0324 0.0148
-0.0033 0.0537 -0.0681 -0.0733 -0.0201 -0.0329 0.1242 0.0324 -0.0744 -0.0149 -0.0047 -0.0484 -0.0483 0.0481 0.0107 0.0101 -0.0704
0.0500 0.0112 -0.0227 0.0499 -0.0259 -0.0441 0.0712 -0.0157 -0.1271 0.0407 -0.0495 -0.0359 0.0202 0.0024 0.0764 0.0196 0.0267
-0.0117 0.0026 0.0171 -0.0121 -0.1374 -0.0370 0.0247 -0.0113 -0.0094 0.0322 -0.0347 -0.0866 0.0042 -0.0014 0.0067 0.0591 0.0009
0.0085 0.0310 0.0479 -0.0511 0.0198 -0.0886 -0.0274 -0.1364 0.0322 -0.1638 -0.0689 0.0016 -0.1039 0.0059 0.0757 -0.0034 0.1013
-0.0034 -0.0065 -0.0468 0.1577 -0.0065 -0.0478 -0.0004 0.0682 0.0045 -0.0607 -0.0590 0.0343 0.0036 -0.1014 -0.0136 -0.0063 0.0801
0.0360 0.0579 -0.0039 0.0975 0.0500 -0.0558 -0.0095 0.0057 -0.0246 0.1070 -0.0186 0.0669 -0.0781 -0.0569 -0.1286 -0.0834 0.0106
-0.0672 -0.0205 0.0613 0.0290 -0.0545 -0.0481 -0.0882 -0.0489 0.0622 -0.0730 -0.0192 -0.0415 -0.0287 0.0218 -0.0427 -0.0046
0.0255 -0.1164 0.0077 -0.0546 -0.0786 0.0000 -0.0456 0.0943 0.0157 -0.0117 -0.0441 -0.0015 -0.0556 -0.0508 0.0088 0.0418 0.0030
-0.1450 -0.0663 0.0800 0.0172 -0.0289 0.1178 -0.0973 0.0888 0.0637 -0.0295 0.0212 0.0100 -0.0860 0.0035 0.0730 0.0425 -0.0080
0.0885 -0.0166 -0.0765 0.0004 -0.0118 0.0138 -0.0093 -0.0606 -0.0447 -0.0746 0.0131 -0.0447 -0.0763 0.0032 0.1181 0.0542 0.0431
-0.0273 0.0547 0.0135 0.0006 -0.0241 -0.0418 0.0278 -0.0821 -0.0572 -0.0039 0.0214 -0.0196 0.0449 -0.0286 0.0204 0.0681 -0.0901
-0.0266 -0.0287 -0.0874 0.0797 -0.0784 -0.0920 0.0380 0.0411 0.0859 0.0369 0.0595 0.0446 0.0363 -0.0353 -0.0044 -0.0061 0.1134
0.1420 -0.0026 -0.0013 0.0033 0.0508 0.0096 -0.0757 0.0085 -0.0099 -0.0384 0.0218 -0.0259 -0.0112 -0.0212 0.0273 0.0532 -0.0278
-0.0634 0.0317 -0.0022 0.0882 -0.0240 0.0031 -0.0370 0.0747 -0.0097 -0.0315 0.0405 0.0124 -0.1416 -0.0768 0.0363 -0.1248 -0.0134
0.0702 -0.0905 -0.0387 0.0683 -0.0784 0.0886 0.0640 0.0611 -0.0199 -0.0447 -0.1331 -0.1247 0.0540 0.0499 -0.0212 -0.0544 -0.1161
-0.0729 0.0894 0.0532 0.0164 -0.0039 -0.0108 -0.0248 -0.1021 -0.0549 -0.0318 0.0309 -0.0691
```

Embeddings

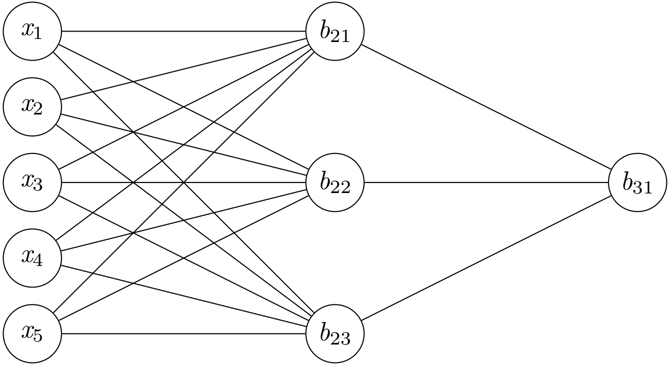
Why is that useful?

1. Input Representation for Neural Networks
   - Example Task: Sentiment Analysis
   - Take a sentence, classify it as positive or negative

Embeddings

Why is that useful?

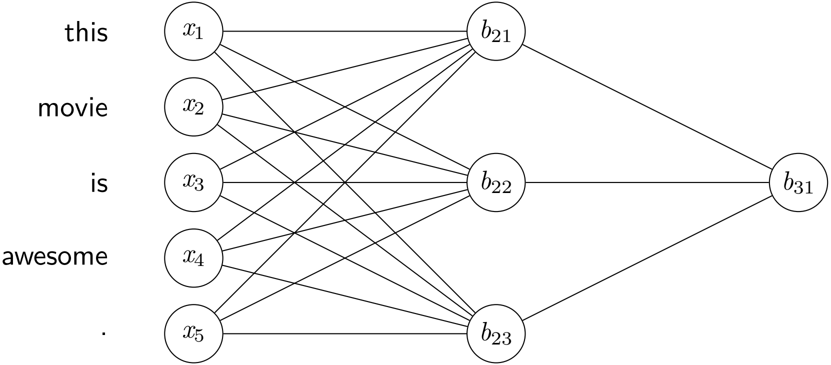
1. Input Representation for Neural Networks
   - Example Task: Sentiment Analysis
   - Take a sentence, classify it as positive or negative



Embeddings

Why is that useful?

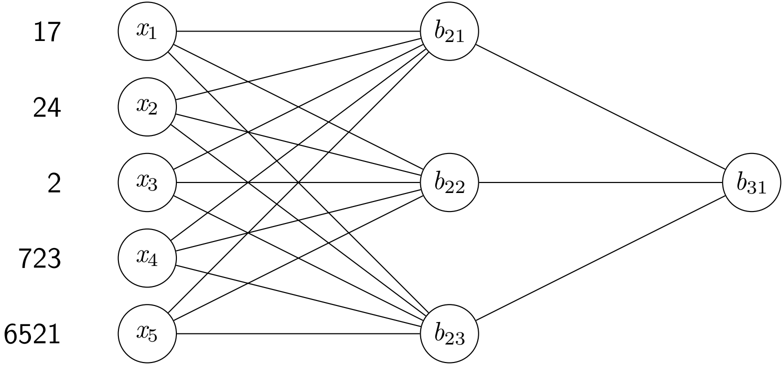
1. Input Representation for Neural Networks
   - Example Task: Sentiment Analysis
   - Take a sentence, classify it as positive or negative



Embeddings

Why is that useful?

1. Input Representation for Neural Networks
   - Example Task: Sentiment Analysis
   - Take a sentence, classify it as positive or negative



Embeddings

Why is that useful?

1. Input Representation for Neural Networks
   - Example Task: Sentiment Analysis
   - Take a sentence, classify it as positive or negative

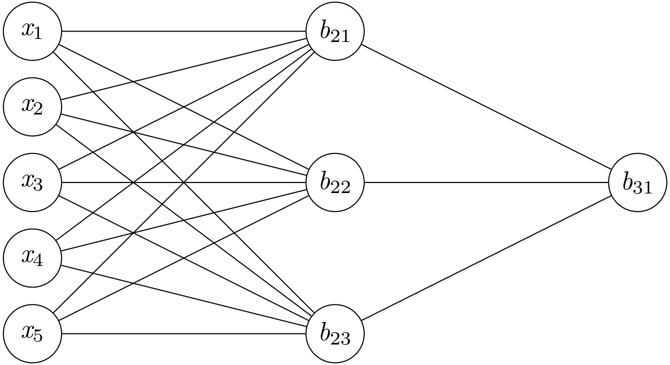
$$\langle 0.0088, 0.0418, 0.0030, -0.1450 \rangle$$

$$\langle 0.0683, -0.0784, 0.0886, 0.0640 \rangle$$

$$\langle -0.0353, -0.0044, -0.0061, 0.1134 \rangle$$

$$\langle -0.0278, -0.0634, 0.0317, -0.0022 \rangle$$

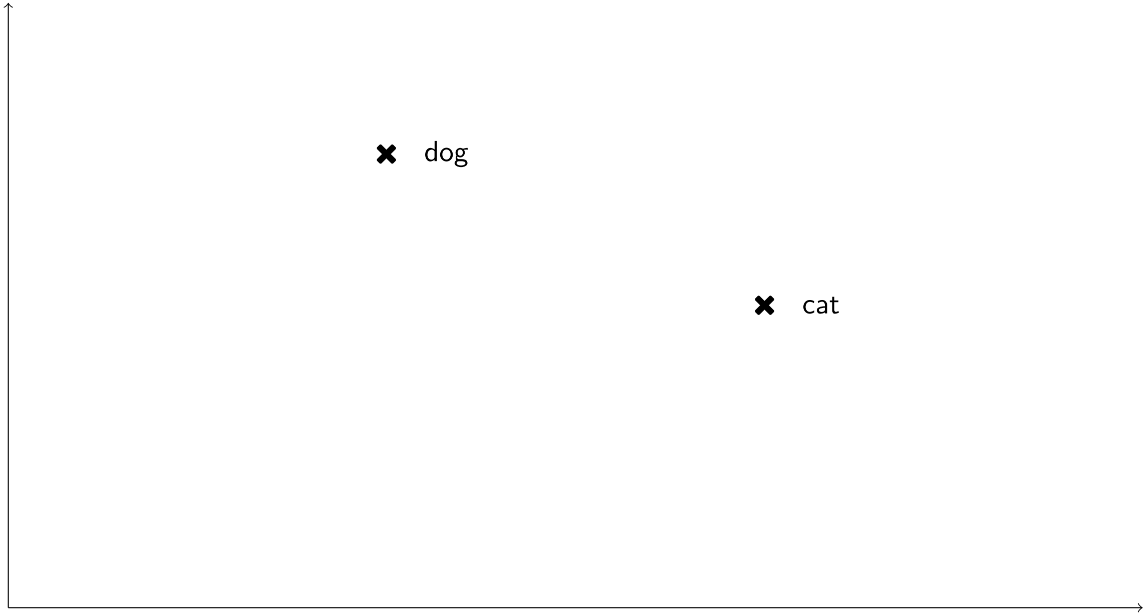
$$\langle -0.0689, 0.0016, -0.1039, 0.0059 \rangle$$

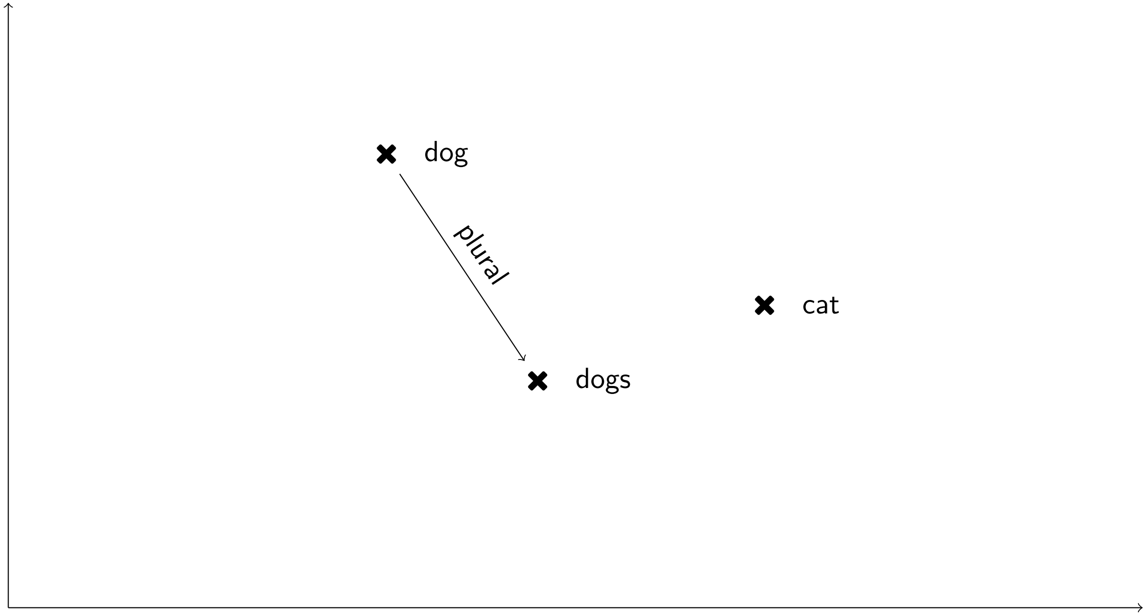


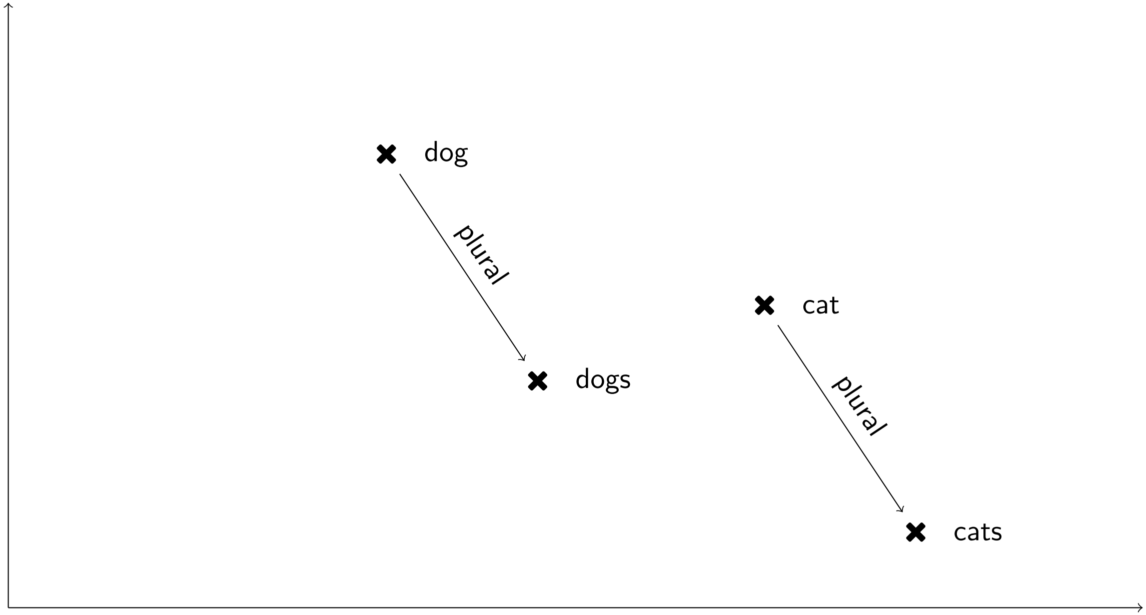
Embeddings

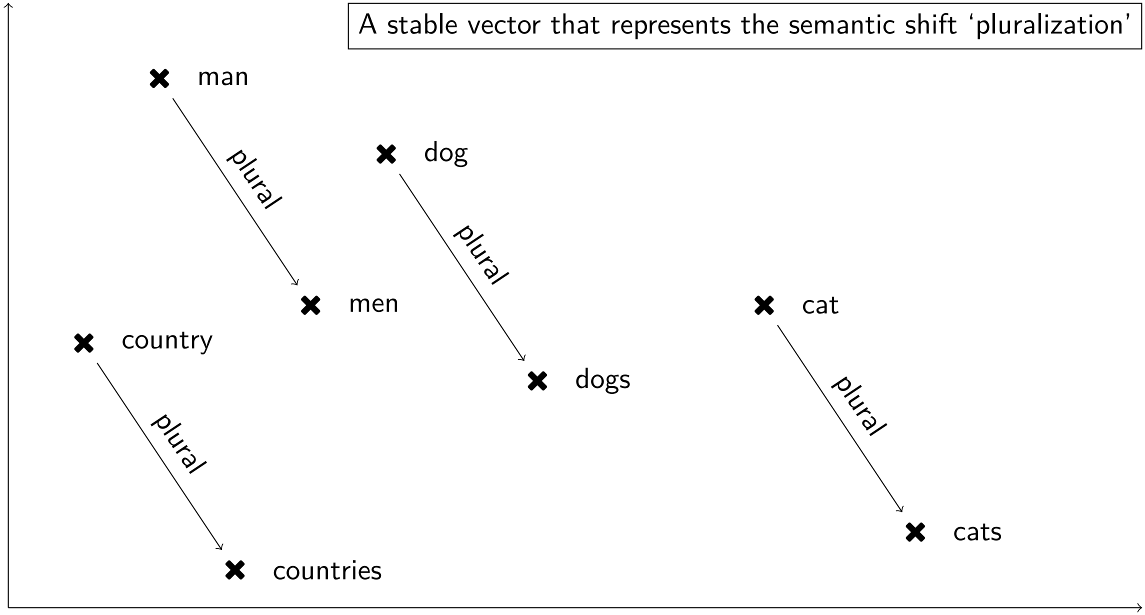
Why is that useful?

2. For semantic calculations

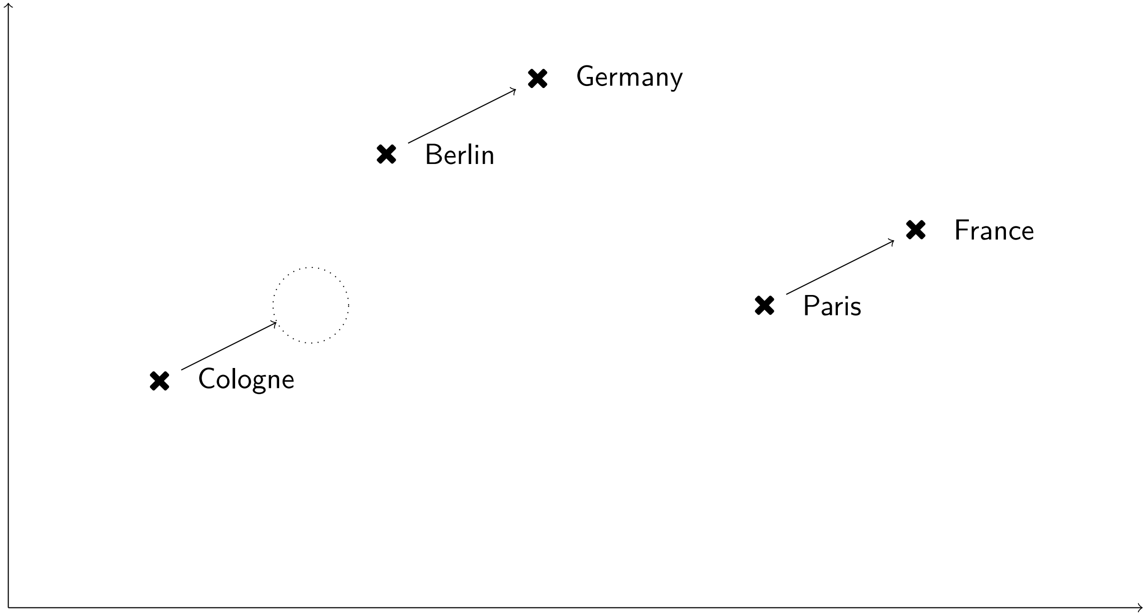

dog
cat


dog
plural
cat
dogs


dog
plural
dogs
cat
plural
cats

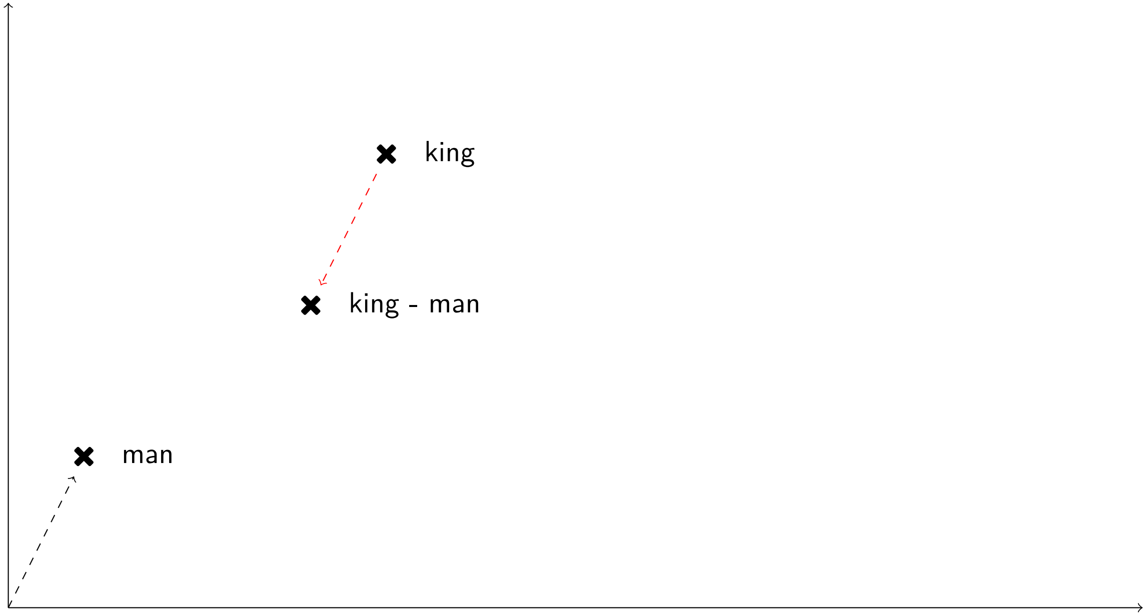

A stable vector that represents the semantic shift 'pluralization'
man
plural
men
dog
plural
dogs
cat
plural
cats
country
plural
countries

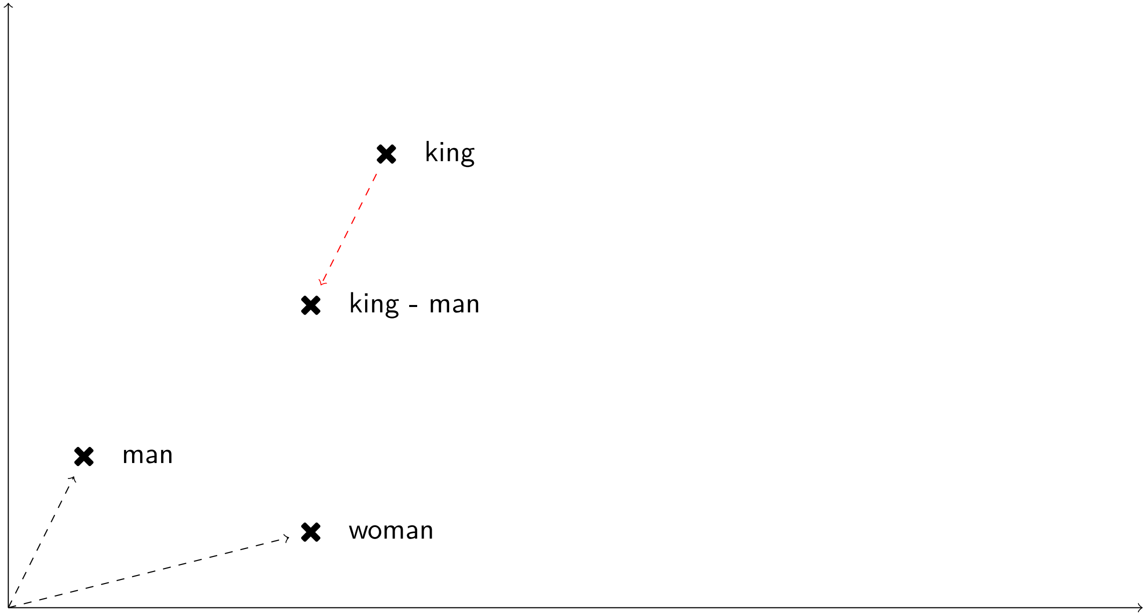

Berlin
Paris
Cologne

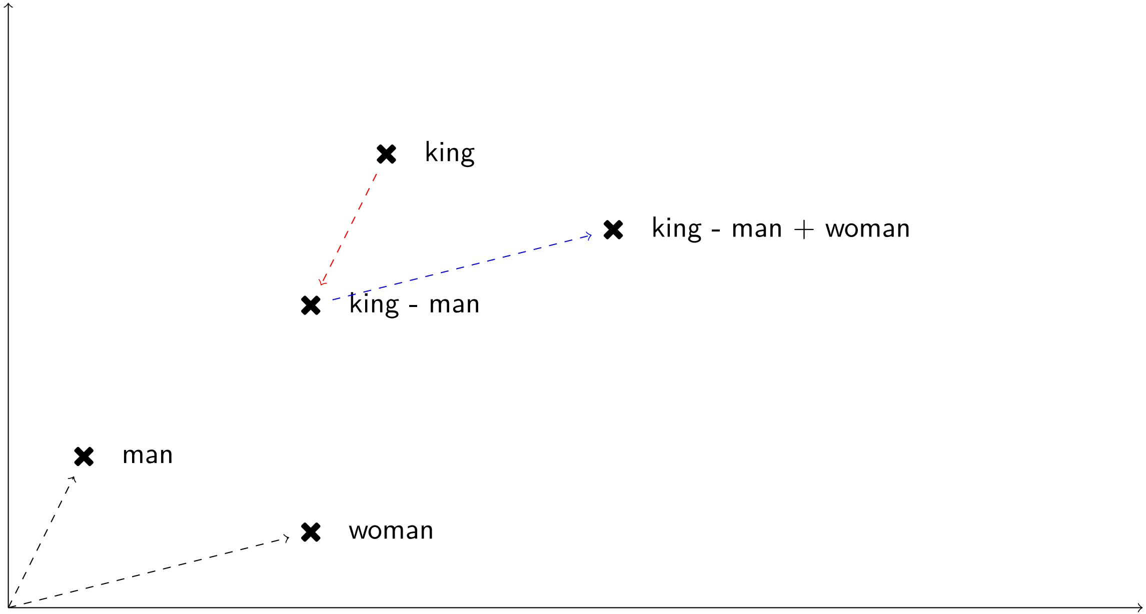


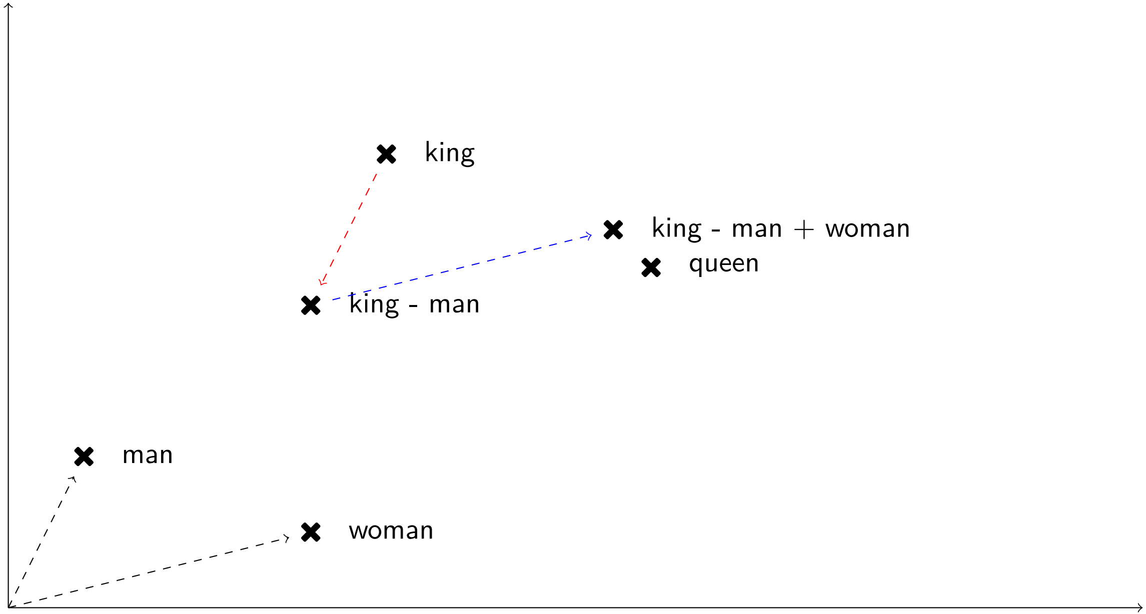
Germany
Berlin
France
Paris
Cologne


king
man


king
king - man
man


king
king - man
man
woman


king
king - man + woman
king - man
man
woman


king
king - man + woman
queen
king - man
man
woman

demo

Subsection 1

Generating Word Embeddings with Word2Vec

Literature basis

Two very influential papers by Mikolov et al.

Tomáš Mikolov/Kai Chen/Greg Corrado/Jeffrey Dean (2013). "Efficient Estimation of Word Representations in Vector Space". In: *arXiv cs.CL*. URL: https://arxiv.org/pdf/1301.3781.pdf

Tomáš Mikolov/Ilya Sutskever/Kai Chen/Greg S Corrado/Jeff Dean (2013). "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges/L. Bottou/M. Welling/Z. Ghahramani/K. Q. Weinberger. Curran Associates, Inc., pp. 3111–3119. URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

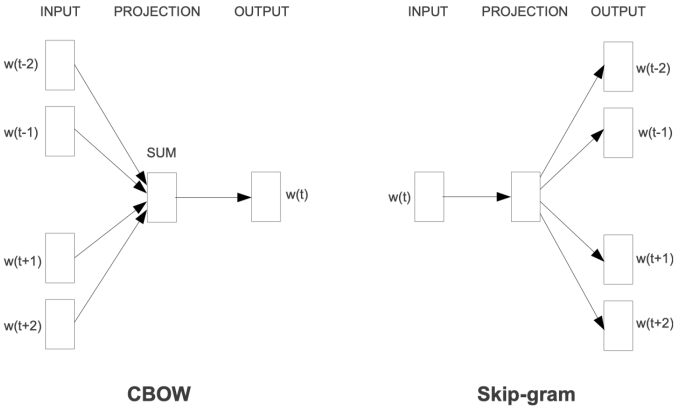
Software package

word2vec – https://github.com/tmikolov/word2vec
(other implementations do exist)

Core Idea

- Define a classification task for which we have huge training data sets
  - Given a word, predict predict possible context words
  - Training data: Any text collection (e.g., Wikipedia)
- Train a neural network
- Extract learned weights and use as embeddings ☑

Two tasks



Continuous Bag of Words (CBOW)

Context words used to predict a single word

Skip-Gram

One word used to predict its context

Skip-gram

- Context: ± 2 words around target word t

```
... lemon, a [tablespoon of apricot jam, a] pinch ...
              c1          c2 t          c3   c4
```

Skip-gram

- Context: ± 2 words around target word t

```
... lemon, a [tablespoon of apricot jam, a] pinch ...
               c1         c2 t       c3   c4
```

- Classifier:
  - Predict for (t, c) wether c are *really* context words for t
  - Probability of $\vec{t}$ and $\vec{c}$ being positive examples: $p(+|\vec{t}, \vec{c})$

Skip-gram

- Context: ± 2 words around target word t

```
... lemon, a [tablespoon of apricot jam, a] pinch ...
              c1         c2 t       c3  c4
```

- Classifier:
  - Predict for (t, c) wether c are *really* context words for t
  - Probability of $\vec{t}$ and $\vec{c}$ being positive examples: $p(+|\vec{t}, \vec{c})$
- Probability is based on similarity
  - "a word is likely to occur near the target if its embedding is similar to the target embedding" Jurafsky/Martin (JM19, 112)

Skip-gram

- Context: ± 2 words around target word t

```
... lemon, a [tablespoon of apricot jam, a] pinch ...
              c1          c2 t       c3   c4
```

- Classifier:
  - Predict for (t, c) wether c are *really* context words for t
  - Probability of $\vec{t}$ and $\vec{c}$ being positive examples: $p(+|\vec{t}, \vec{c})$
- Probability is based on similarity
  - "a word is likely to occur near the target if its embedding is similar to the target embedding" Jurafsky/Martin (JM19, 112)
  - Similarity of vectors? Dot product / cosine!

Skip-gram

- Context: ± 2 words around target word t

```
... lemon, a [tablespoon of apricot jam, a] pinch ...
              c1         c2 t       c3   c4
```

- Classifier:
  - Predict for (t, c) wether c are *really* context words for t
  - Probability of $\vec{t}$ and $\vec{c}$ being positive examples: $p(+|\vec{t}, \vec{c})$
- Probability is based on similarity
  - "a word is likely to occur near the target if its embedding is similar to the target embedding" Jurafsky/Martin (JM19, 112)
  - Similarity of vectors? Dot product / cosine! ➋
  - Similarity $\rightarrow$ probability? Sigmoid / logistic function! ➋

When are vectors similar?

- Metric that takes two vectors and returns a similarity score
- Linear algebra: dot product ("Skalarprodukt")

When are vectors similar?

- Metric that takes two vectors and returns a similarity score
- Linear algebra: dot product ("Skalarprodukt")

$$\begin{aligned}\vec{a} \cdot \vec{b} &= \sum_{i=1}^{N} a_i b_i \\ &= |\vec{a}||\vec{b}| \cos \sphericalangle(\vec{a}, \vec{b})\end{aligned}$$

Dot product

Example

$$\begin{aligned} \vec{a} &= [0, 0, 1, 1] \\ \vec{b} &= [0, 0, 1, 0.95] \\ \vec{a} \cdot \vec{b} &= 1.95 \end{aligned}$$

Dot product

Example

$$\begin{aligned}
\vec{a} &= [0, 0, 1, 1] \\
\vec{b} &= [0, 0, 1, 0.95] \\
\vec{a} \cdot \vec{b} &= 1.95 \\
\vec{a}' &= 10\vec{a} = [0, 0, 10, 10] \\
\vec{b}' &= 10\vec{b} = [0, 0, 10, 9.5]
\end{aligned}$$

Dot product

Example

$$
\begin{aligned}
\vec{a} &= [0, 0, 1, 1] \\
\vec{b} &= [0, 0, 1, 0.95] \\
\vec{a} \cdot \vec{b} &= 1.95 \\
\vec{a}' &= 10\vec{a} = [0, 0, 10, 10] \\
\vec{b}' &= 10\vec{b} = [0, 0, 10, 9.5] \\
\vec{a}' \cdot \vec{b}' &= 195
\end{aligned}
$$

Dot product as similarity metric?

- Favours high frequent words
  - For the word ‘Cologne’, it’s easier to be similar to ‘the’ than to ‘Düsseldorf’
  - Because ‘the’ is more frequent (= has more higher numbers in its vector) than ‘Cologne’

Dot product as similarity metric?

- Favours high frequent words
  - For the word ‘Cologne’, it’s easier to be similar to ‘the’ than to ‘Düsseldorf’
  - Because ‘the’ is more frequent (= has more higher numbers in its vector) than ‘Cologne’
- Missing: Normalisation for vector length
  - Normalisation can mostly done by dividing by something

Dot product as similarity metric?

- Favours high frequent words
  - For the word ‘Cologne’, it’s easier to be similar to ‘the’ than to ‘Düsseldorf’
  - Because ‘the’ is more frequent (= has more higher numbers in its vector) than ‘Cologne’
- Missing: Normalisation for vector length
  - Normalisation can mostly done by dividing by something
- Normalised dot product: Divide by vector lengths

$$\vec{a} \cdot \vec{b} \quad = \quad |\vec{a}||\vec{b}| \cos \sphericalangle(\vec{a}, \vec{b})$$

Dot product as similarity metric?

- Favours high frequent words
  - For the word ‘Cologne’, it’s easier to be similar to ‘the’ than to ‘Düsseldorf’
  - Because ‘the’ is more frequent (= has more higher numbers in its vector) than ‘Cologne’
- Missing: Normalisation for vector length
  - Normalisation can mostly done by dividing by something
- Normalised dot product: Divide by vector lengths

$$\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}| \cos \sphericalangle(\vec{a}, \vec{b})$$

$$\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} = \frac{|\vec{a}||\vec{b}| \cos \sphericalangle(\vec{a}, \vec{b})}{|\vec{a}||\vec{b}|}$$

Dot product as similarity metric?

- Favours high frequent words
  - For the word 'Cologne', it's easier to be similar to 'the' than to 'Düsseldorf'
  - Because 'the' is more frequent (= has more higher numbers in its vector) than 'Cologne'
- Missing: Normalisation for vector length
  - Normalisation can mostly done by dividing by something
- Normalised dot product: Divide by vector lengths

$$
\begin{aligned}
\vec{a} \cdot \vec{b} &= |\vec{a}||\vec{b}| \cos \sphericalangle(\vec{a}, \vec{b}) \\
\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} &= \frac{|\vec{a}||\vec{b}| \cos \sphericalangle(\vec{a}, \vec{b})}{|\vec{a}||\vec{b}|} \\
&= \cos \sphericalangle(\vec{a}, \vec{b})
\end{aligned}
$$

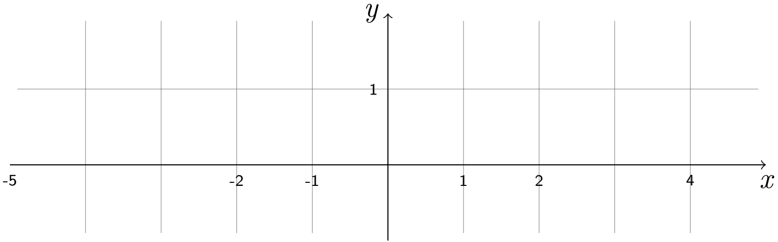
Cosine Similarity Metric

$$\cos \sphericalangle(\vec{a}, \vec{b}) = \frac{\sum_{i=1}^{N} a_i b_i}{\sum_{i=1}^{N} a_i^2 \sum_{i=1}^{N} b_i^2}$$

- Independent of length (measures the angle between the vectors)
- Simple to calculate

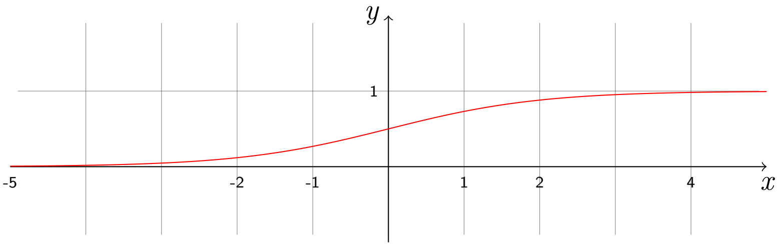
The Logistic Function

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = 2.71828 = \text{Euler's Number}$$



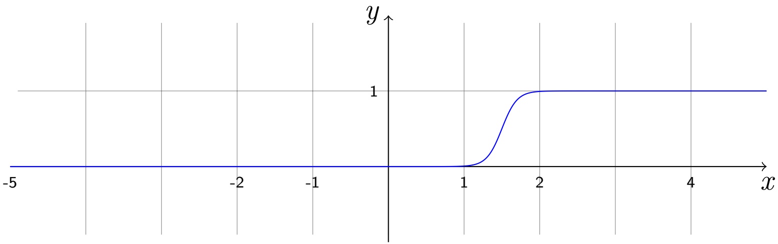
$$y = \frac{1}{1+e^{-x}} = \frac{1}{1+e^{-(ax+b)}}$$

The Logistic Function



$$y = \frac{1}{1+e^{-x}} = \frac{1}{1+e^{-(ax+b)}} = \frac{1}{1+e^{-(1*x+0)}}$$

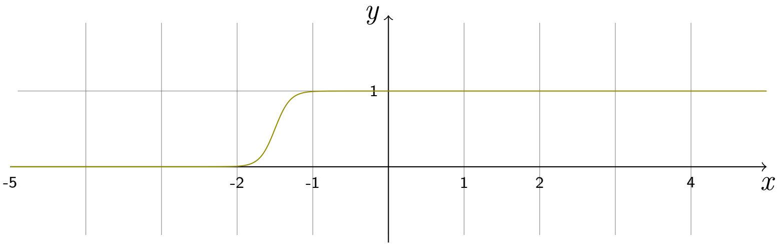
The Logistic Function



$$y = \frac{1}{1+e^{-x}} = \frac{1}{1+e^{-(ax+b)}} = \frac{1}{1+e^{-(1*x+0)}}$$

$$y = \frac{1}{1+e^{-(10*x-15)}}$$

The Logistic Function

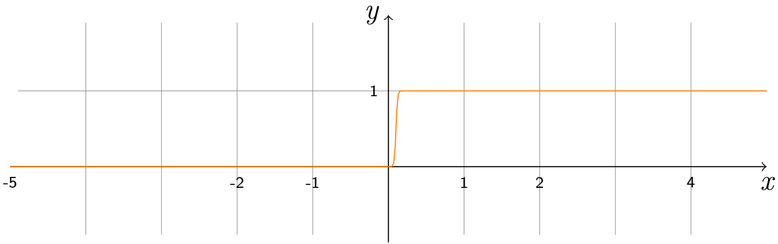


$$y = \frac{1}{1+e^{-x}} = \frac{1}{1+e^{-(ax+b)}} = \frac{1}{1+e^{-(1*x+0)}}$$

$$y = \frac{1}{1+e^{-(10*x-15)}}$$

$$y = \frac{1}{1+e^{-(10*x+15)}}$$

The Logistic Function



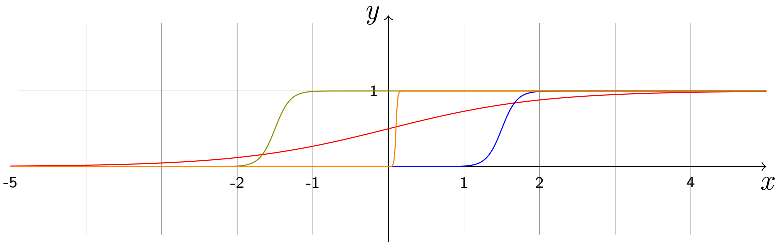
$$y = \frac{1}{1+e^{-x}} = \frac{1}{1+e^{-(ax+b)}} = \frac{1}{1+e^{-(1*x+0)}}$$

$$y = \frac{1}{1+e^{-(10*x-15)}}$$

$$y = \frac{1}{1+e^{-(10*x+15)}}$$

$$y = \frac{1}{1+e^{-(100*x-10)}}$$

The Logistic Function



$$y = \frac{1}{1+e^{-x}} = \frac{1}{1+e^{-(ax+b)}} = \frac{1}{1+e^{-(1*x+0)}}$$
$$y = \frac{1}{1+e^{-(10*x-15)}}$$
$$y = \frac{1}{1+e^{-(10*x+15)}}$$
$$y = \frac{1}{1+e^{-(100*x-10)}}$$

Skip-gram

Notation

t, c: words

$\vec{t}$, $\vec{c}$: vectors for the words

(this is different from JM19)

$$p(+|t,c) = \frac{1}{1+e^{-\vec{t}\cdot\vec{c}}}$$

$$p(-|t,c) = 1 - \frac{1}{1+e^{-\vec{t}\cdot\vec{c}}} = \frac{e^{-\vec{t}\cdot\vec{c}}}{1+e^{-\vec{t}\cdot\vec{c}}}$$

Skip-gram

Notation

t, c: words

$\vec{t}$, $\vec{c}$: vectors for the words

(this is different from JM19)

$$p(+|t, c) = \frac{1}{1 + e^{-\vec{t}\cdot\vec{c}}}$$

$$p(-|t, c) = 1 - \frac{1}{1 + e^{-\vec{t}\cdot\vec{c}}} = \frac{e^{-\vec{t}\cdot\vec{c}}}{1 + e^{-\vec{t}\cdot\vec{c}}}$$

but the context consists of more than one word!

Skip-gram

Notation

t, c: words

$\vec{t}$, $\vec{c}$: vectors for the words

(this is different from JM19)

$$p(+|t, c) = \frac{1}{1 + e^{-\vec{t} \cdot \vec{c}}}$$

$$p(-|t, c) = 1 - \frac{1}{1 + e^{-\vec{t} \cdot \vec{c}}} = \frac{e^{-\vec{t} \cdot \vec{c}}}{1 + e^{-\vec{t} \cdot \vec{c}}}$$

but the context consists of more than one word!

Assumption: They are independent, allowing multiplication

Skip-gram

Notation
t, c: words
$\vec{t}$, $\vec{c}$: vectors for the words
(this is different from JM19)

$$p(+|t, c) = \frac{1}{1 + e^{-\vec{t} \cdot \vec{c}}}$$

$$p(-|t, c) = 1 - \frac{1}{1 + e^{-\vec{t} \cdot \vec{c}}} = \frac{e^{-\vec{t} \cdot \vec{c}}}{1 + e^{-\vec{t} \cdot \vec{c}}}$$

but the context consists of more than one word!
Assumption: They are independent, allowing multiplication

$$p(+|t, c_{1:k}) = \prod_{i=1}^{k} \frac{1}{1 + e^{-\vec{t} \cdot \vec{c}_i}}$$

$$\log p(+|t, c_{1:k}) = \sum_{i=1}^{k} \log \frac{1}{1 + e^{-\vec{t} \cdot \vec{c}_i}}$$

Skip-gram

- So far, we have assumed that we have vector $\vec{t}$ for word t, but where do they come from?
- Basic gradient descent: We start randomly, and iteratively improve

Skip-gram

Negative sampling

- Negative examples
  - Training a classifier needs negative examples, i.e., words that are not in the context of each other

Skip-gram

Negative sampling

- Negative examples
  - Training a classifier needs negative examples, i.e., words that are not in the context of each other
- Negative sampling
  - For every positive tuple (t, c), we add k negative tuples
  - Negative tuple (t, c_n), with c_n randomly selected (and $t \neq c_n$)

Skip-gram

Negative sampling

- Negative examples
  - Training a classifier needs negative examples, i.e., words that are not in the context of each other
- Negative sampling
  - For every positive tuple (t, c), we add k negative tuples
  - Negative tuple (t, c_n), with c_n randomly selected (and $t \neq c_n$)
  - Select noise words according to their weighted frequency
  - $p_\alpha(w) = \frac{count(w)^\alpha}{\sum_{w'} count(w')^\alpha}$
    - This leads to rare words being more frequently selected, frequent words less

Skip-gram

Negative sampling

- Negative examples
  - Training a classifier needs negative examples, i.e., words that are not in the context of each other
- Negative sampling
  - For every positive tuple (t, c), we add k negative tuples
  - Negative tuple (t, c_n), with c_n randomly selected (and $t \neq c_n$)
  - Select noise words according to their weighted frequency
  - $p_\alpha(w) = \frac{count(w)^\alpha}{\sum_{w'} count(w')^\alpha}$
    - This leads to rare words being more frequently selected, frequent words less
- Two new ‘parameters’ on this slide: k and α
  - They have a different status than θ (the parameters we want to learn)
  - Therefore: Hyperparameters

Word2Vec

Loss

- We also need a loss function
- Idea:
  - Maximize $p(+|t, c)$ (positive samples)
  - Minimize $p(+|t, c_n)$ (negative samples)

Word2Vec

Loss

- We also need a loss function
- Idea:
  - Maximize $p(+|t, c)$ (positive samples)
  - Minimize $p(+|t, c_n)$ (negative samples)

$$L(\theta) = \sum_{(t,c)} \log p(+|t, c) + \sum_{(t,c_n)} \log p(-|t, c_n)$$

Word2Vec

Loss

- We also need a loss function
- Idea:
  - Maximize $p(+|t, c)$ (positive samples)
  - Minimize $p(+|t, c_n)$ (negative samples)

$$L(\theta) = \sum_{(t,c)} \log p(+|t, c) + \sum_{(t,c_n)} \log p(-|t, c_n)$$

θ: Concatenation of all $\vec{t}$, $\vec{c}$, $\vec{c}_n$

Remarks and observations

- Each word is used twice, with different roles
  - As target word (for predicting its context)
  - As context word (to be predicted from another target word)
  - Different options: Only use one embedding, combine them by addition or concatenation

Remarks and observations

- Each word is used twice, with different roles
  - As target word (for predicting its context)
  - As context word (to be predicted from another target word)
  - Different options: Only use one embedding, combine them by addition or concatenation
- Matrices
  - Conceptually, it is not hugely important how the embeddings are stored in detail
  - But for the implementation because of efficiency
  - All target vectors are stored in matrix W (word matrix)
  - All context vectors are stored in matrix C (context matrix)
  - $\theta = (W, C)$

Zum Schluss

- Einführung in die Informationsverarbeitung ✔
  - Nächste Woche: Studienleistung, anschließend Referenzlösung in Ilias

Zum Schluss

- Einführung in die Informationsverarbeitung ✔
  - Nächste Woche: Studienleistung, anschließend Referenzlösung in Ilias
- Als nächstes: ~~Semesterferien~~vorlesungsfreie Zeit ☺
- Sommersemester 2023
  - Lehrveranstaltungen in Klips: `https://klips2.uni-koeln.de`
  - Welche soll/muss ich nehmen? → Modulhandbuch!
    - `https://phil-fak.uni-koeln.de/studium/bachelor/bachelor-faecher`


Danke für's Zuhören und eine gute Zeit!