

ITMO UNIVERSITY

NLP Basic and Selected Topics

A Practical and Easy Introduction to Selected Topics

Aug 31st, 2019

Overview of the Unit Today

- 1) Applications of NLP / Introduction (30min)
- 2) Practical NLP (NLTK / pythainlp) (45min)
- 3) Modern NLP with ML/DL (45min)**
- 4) Example: Word Similarity and WordNet (30min)
- 5) Modern NLP with fastAI / flair (30min)

Traditional NLP

- ✓ Traditional NLP often based on dictionaries and a large processing pipeline
- ✓ Often with rules and feature engineering
- ✓ **Example:** Sentiment
 - Dictionary
 - POS / NE-extraction – apply dict only for adjectives
 - Negation Rules
 - Maybe feature engineering for machine learning, rules/patterns

Modern NLP

- ✓ Often done end-to-end with deep learning
- ✓ Algorithm finds all the features, and what it needs by itself
- ✓ But: we need (lots of) training data
 - But for many problems we have a lot of training data. For example you can use online movie reviews and their numeric score.

Modern NLP

- ✓ Explain basic idea of a deep neural network

Modern NLP

- ✓ A neural network needs numbers as input – it does not understand strings!
- ✓ Every word needs to be a vector of numbers – **how can we make a word into a vector of numbers?**
- ✓ The numbers need to represent **the meaning** of a word somehow.
 - What does this mean?
 - **How could it be done?**

Count-based Word Vectors

- ✓ Explain word – document matrix
- ✓ Explain word-word matrix
- ✓ Similarity of 2 words in a word-word matrix

Term-Term Matrices

- In IR systems we typically use Term-Document matrices
- But in many NLP applications we are interested in **term-term co-occurrence** matrices
- Co-occurrence can be measured in different ways, for example
 - Within a unit like a **sentence** or **paragraph**
 - Within a **word window** (left and/or right) of the target word – eg. a word window of 5
 - Q: when could a co-occurrence matrix be useful?

Co-occurrence matrix - Example

1. I enjoy flying.
2. I like NLP.
3. I like deep learning.

The resulting counts matrix will then be:

$$X = \begin{array}{c} \begin{array}{l} I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} \begin{bmatrix} \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{array}{cccccccc} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array} \end{bmatrix} \end{array}$$

Co-occurrence Matrices

Distributional hypothesis – "a word is characterized by the company it keeps"

- Words are defined by their context (words)

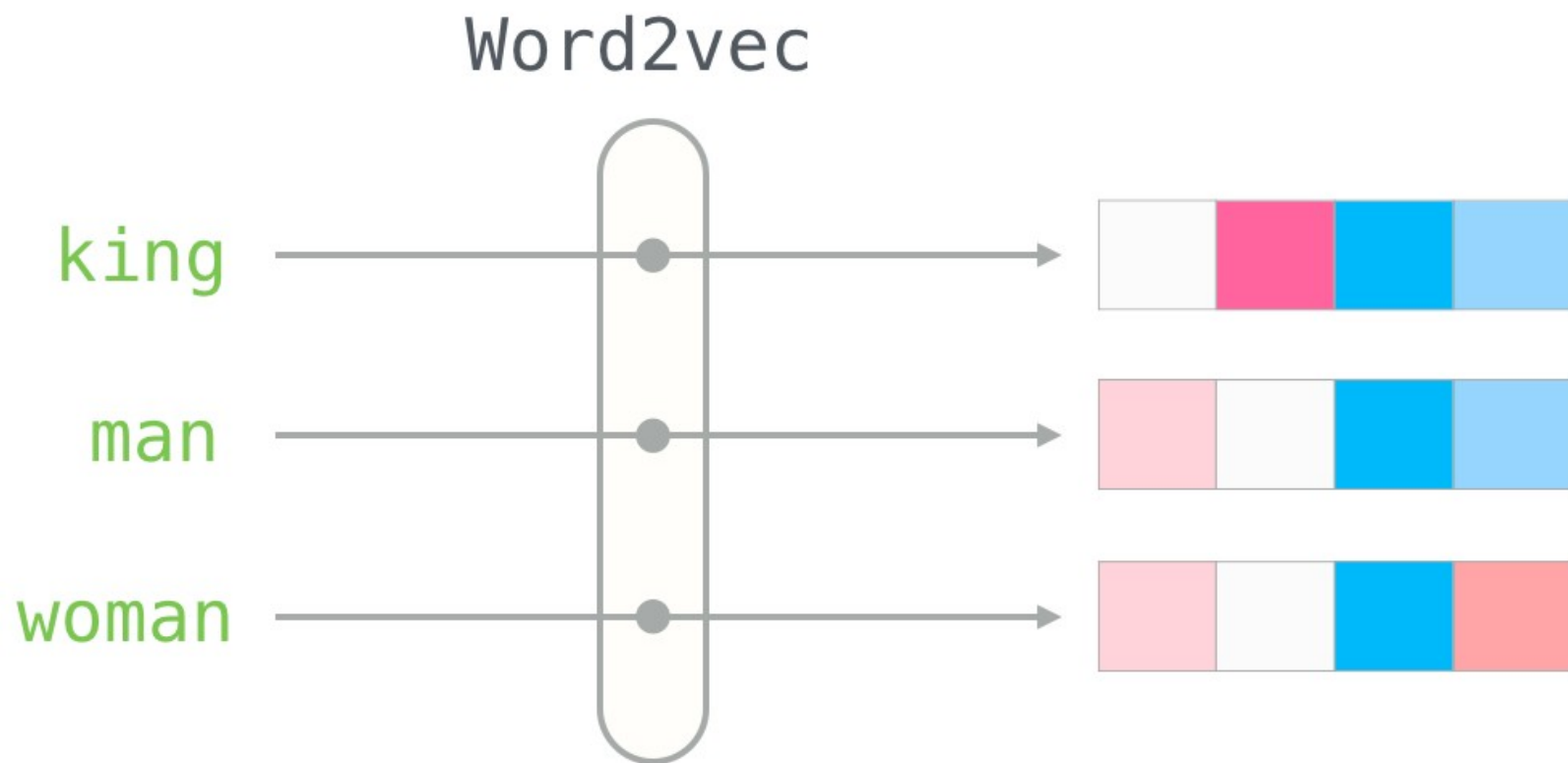
A bottle of **tesgüino** is on the table
Everybody likes **tesgüino**
Tesgüino makes you drunk
We make **tesgüino** out of corn.

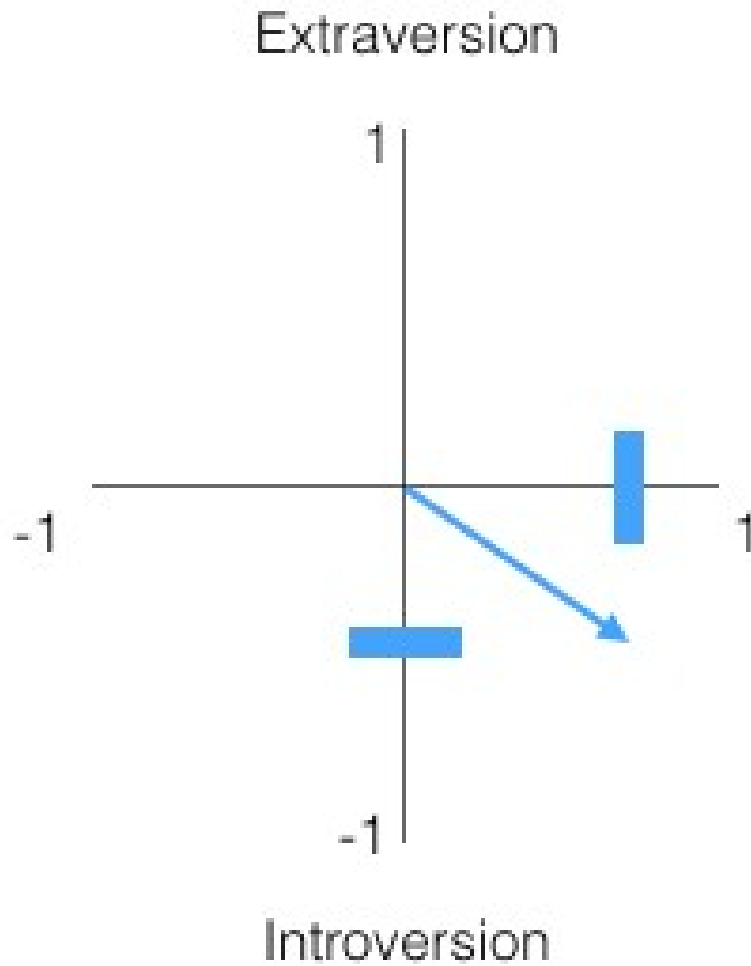
Count-based Word Vectors

- ✓ Once we have the **co-occurrence matrix**, we can apply PCA, SVD etc to **compress the matrix** to have eg only **300 dimensions per word**

Prediction-based word vectors

- ✓ Using neural nets
- ✓ **Most famous: word2vec**
- ✓ Very good explanation <http://jalammar.github.io/illustrated-word2vec/>

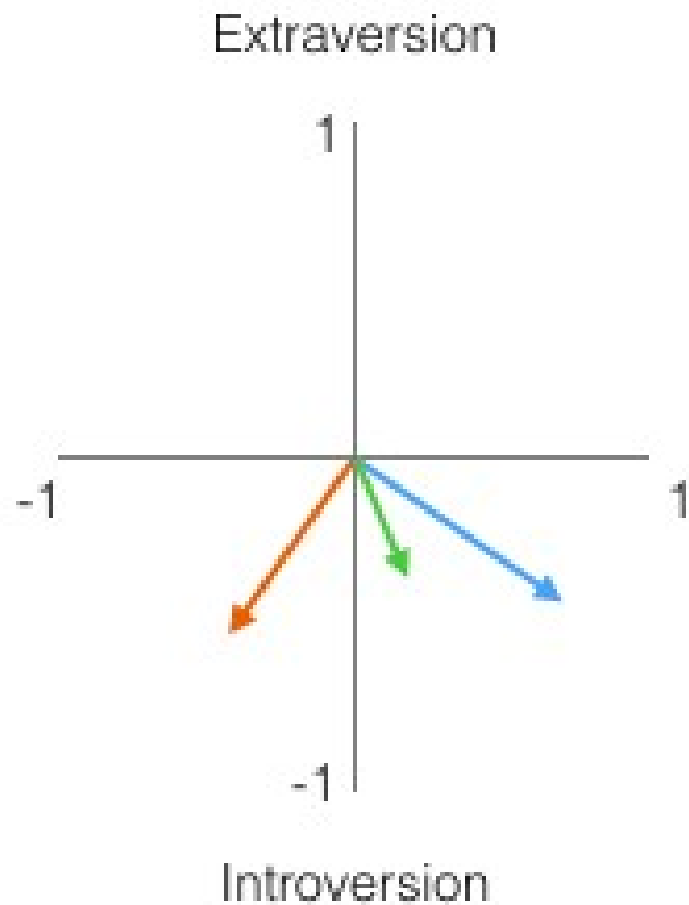




Jay

Trait #1
Trait #2

Trait #1	Trait #2			
-0.4	0.8			



Jay

Trait #1		Trait #2			
-0.4	0.8				



Person #1

-0.3	0.2				
------	-----	--	--	--	--

Person #2

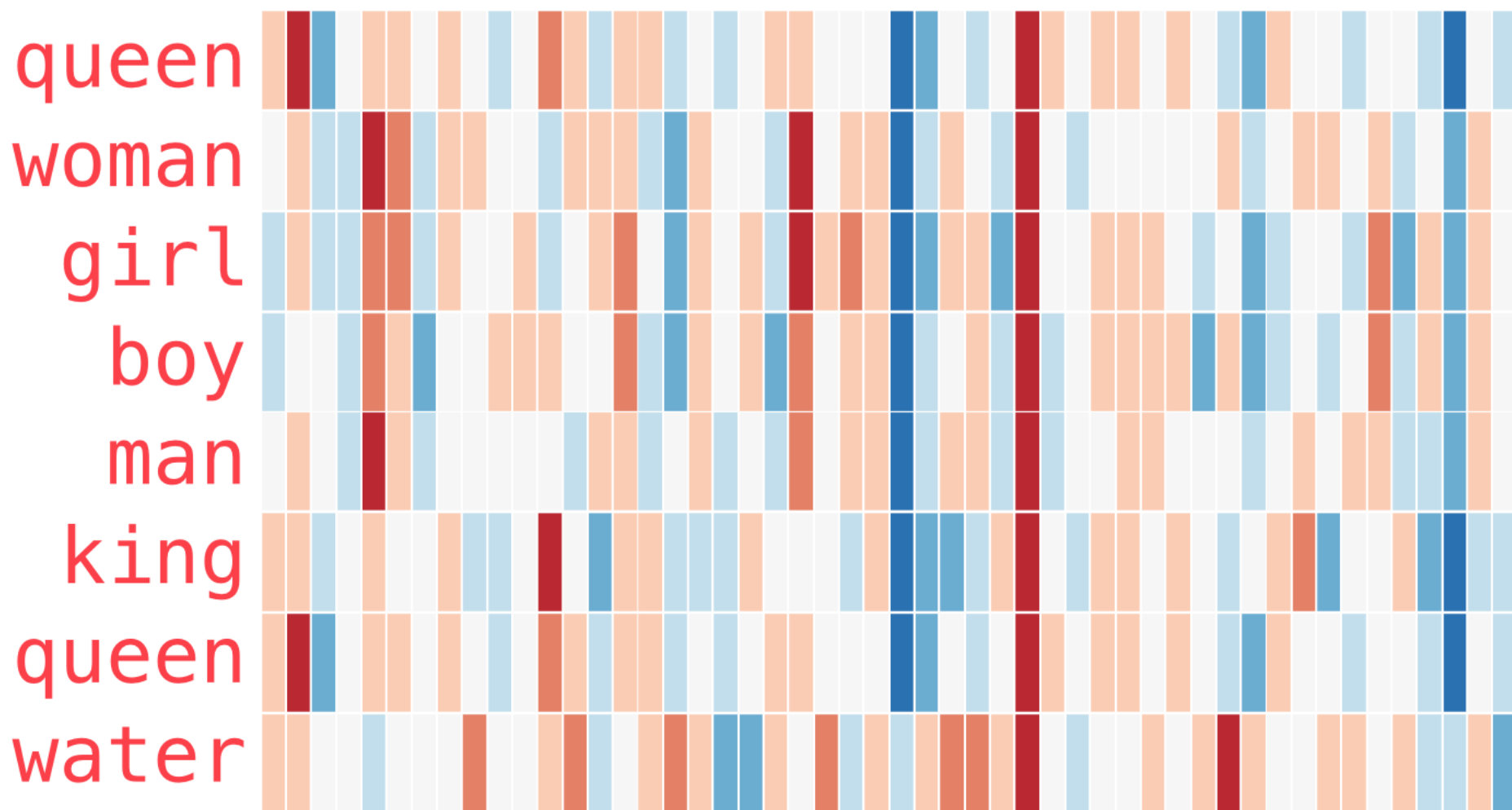
-0.5	-0.4				
------	------	--	--	--	--

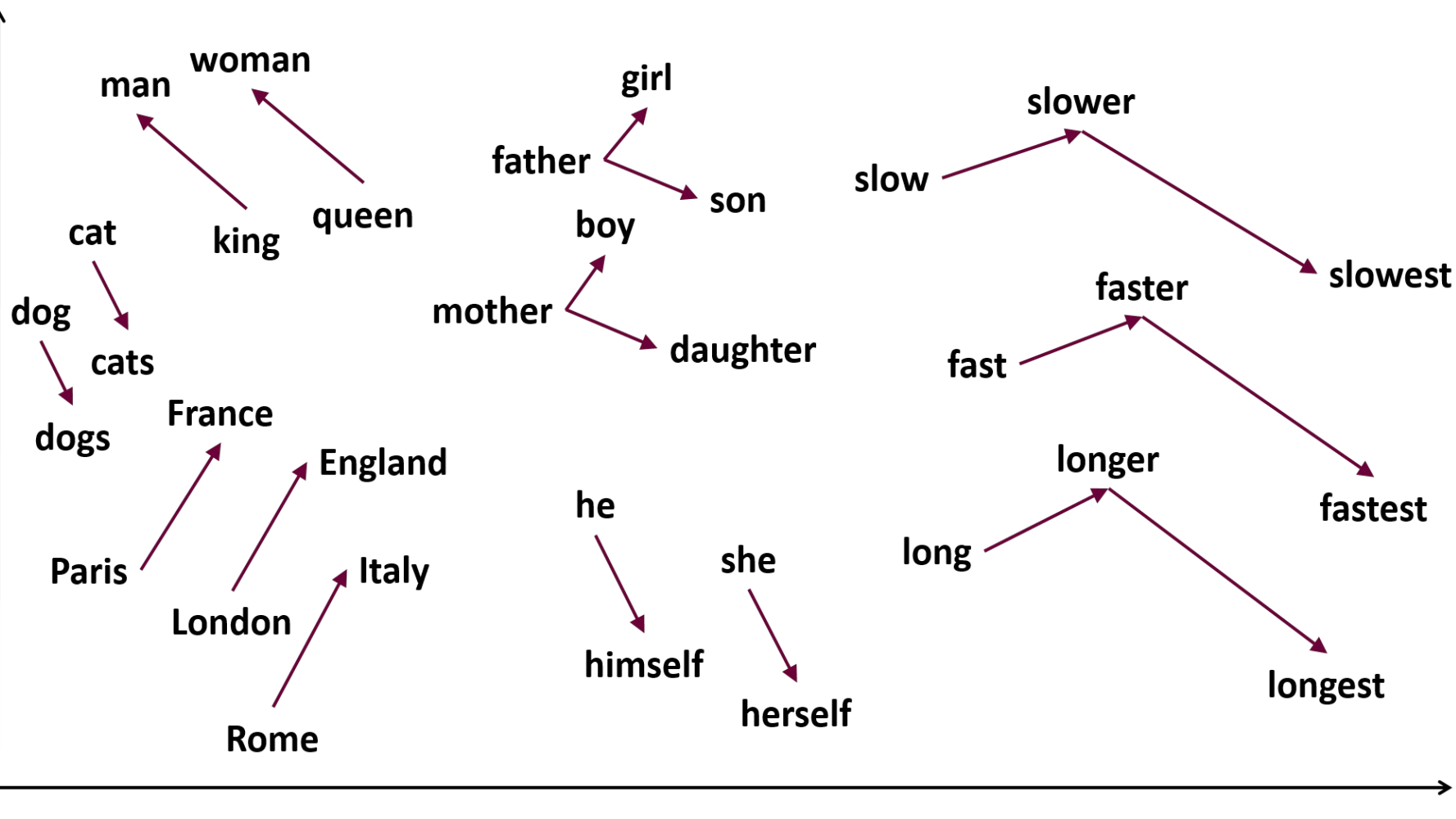
Similarity in Vector Space

$$\text{cosine_similarity}\left(\begin{array}{|c|c|} \hline \text{Jay} & \\ \hline -0.4 & 0.8 \\ \hline \end{array}, \begin{array}{|c|c|} \hline \text{Person \#1} & \\ \hline -0.3 & 0.2 \\ \hline \end{array} \right) = 0.87 \quad \checkmark$$

$$\text{cosine_similarity}\left(\begin{array}{|c|c|} \hline \text{Jay} & \\ \hline -0.4 & 0.8 \\ \hline \end{array}, \begin{array}{|c|c|} \hline \text{Person \#2} & \\ \hline -0.5 & -0.4 \\ \hline \end{array} \right) = -0.20$$

Word vectors





Training models

- ✓ Word2vec models can be easily trained
- ✓ You just need an input corpus
- ✓ For example Wikipedia
- ✓ But also domain specific, for example a book
- ✓ The result will be a model file like this:

Word vectors

the 0.418 0.24968 -0.41242 0.1217 0.34527 -0.044457 -0.49688 -0.17862
-0.00066023 -0.6566 0.27843 -0.14767 -0.55677 0.14658 -0.0095095 0.011658
0.10204 -0.12792 -0.8443 -0.12181 -0.016801 -0.33279 -0.1552 -0.23131
-0.19181 -1.8823 -0.76746 0.099051 -0.42125 -0.19526 4.0071 -0.18594
-0.52287 -0.31681 0.00059213 0.0074449 0.17778 -0.15897 0.012041 -0.054223
-0.29871 -0.15749 -0.34758 -0.045637 -0.44251 0.18785 0.0027849 -0.18411
-0.11514 -0.78581
, 0.013441 0.23682 -0.16899 0.40951 0.63812 0.47709 -0.42852 -0.55641
-0.364 -0.23938 0.13001 -0.063734 -0.39575 -0.48162 0.23291 0.090201
-0.13324 0.078639 -0.41634 -0.15428 0.10068 0.48891 0.31226 -0.1252
-0.037512 -1.5179 0.12612 -0.02442 -0.042961 -0.28351 3.5416 -0.11956
-0.014533 -0.1499 0.21864 -0.33412 -0.13872 0.31806 0.70358 0.44858
-0.080262 0.63003 0.32111 -0.46765 0.22786 0.36034 -0.37818 -0.56657
0.044691 0.30392

Training models

- ✓ But: we don't want to train models now.
- ✓ There are many pretrained models

Use models

- ✓ You can use the Gensim library for Python
- ✓ **See Colab!**
- ✓ <https://rare-technologies.com/word2vec-tutorial/>

Exercise

- ✓ Open Colab
- ✓ Load a model (copy !wget etc code)
- ✓ Test the `most_similar` function
- ✓ Test the `doesnt_match` function
- ✓ Look at the vector of some word

What to do with those embeddings?

- ✓ **Can we do sentence / document classification with the embeddings? How?**
- ✓ How to improve (tf-idf)
- ✓ Use as input in for example machine translation with an RNN – for example on POS tagging
 - Show on whiteboard

What to do with those embeddings?

- ✓ **How can we use embedding in a search system to get more results?!**
- ✓ Example search query: “Bangkok animal shelter”

What to the with those embeddings?

Classification with CNN

