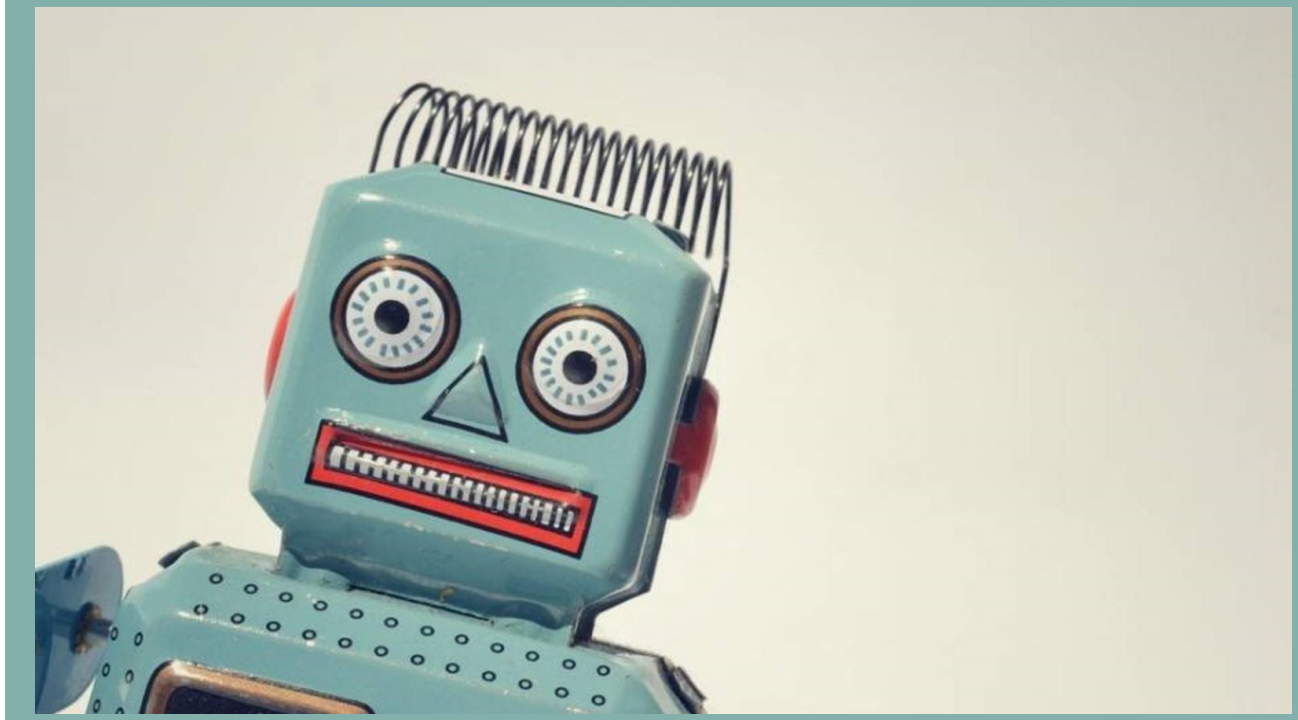
DIGITAL METHODS FOR ANALYSING TEXTS
//

03_Analysing words



Ana Valdivia
Research Associate
King's College London

BEFORE WE START...//

What have we learned?

- Encodings.
- Regular expressions.
- Term-document matrices.
- World clouds.

ROAD MAP//

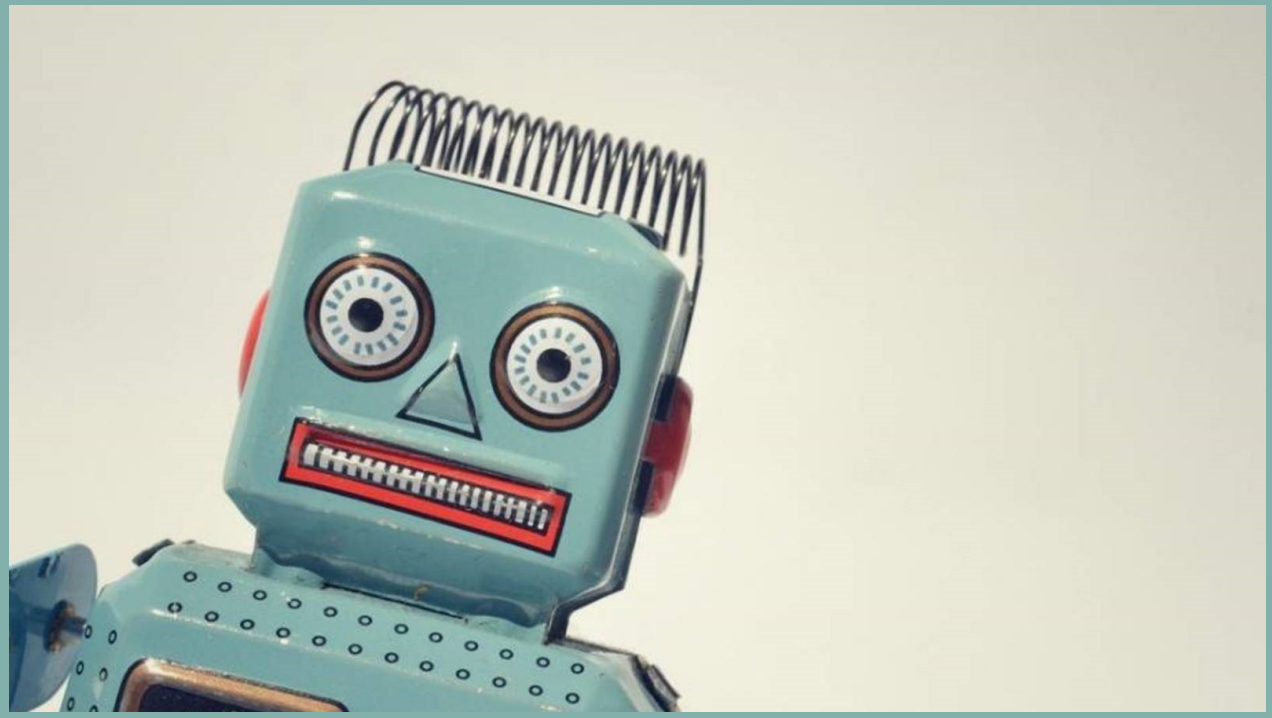
1. WORD VECTORIZATION

1.1. One-hot-encoding

1.2. Word-embeddings

2. WORD PREPROCESSING

TEXT VECTORIZATION
//



TEXT VECTORIZATION//

How would you numerically represent a word?

Click here

TEXT VECTORIZATION//

One-hot encoding

judge = [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]

appellant = [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

TEXT VECTORIZATION//

One-hot encoding

Lack of information:

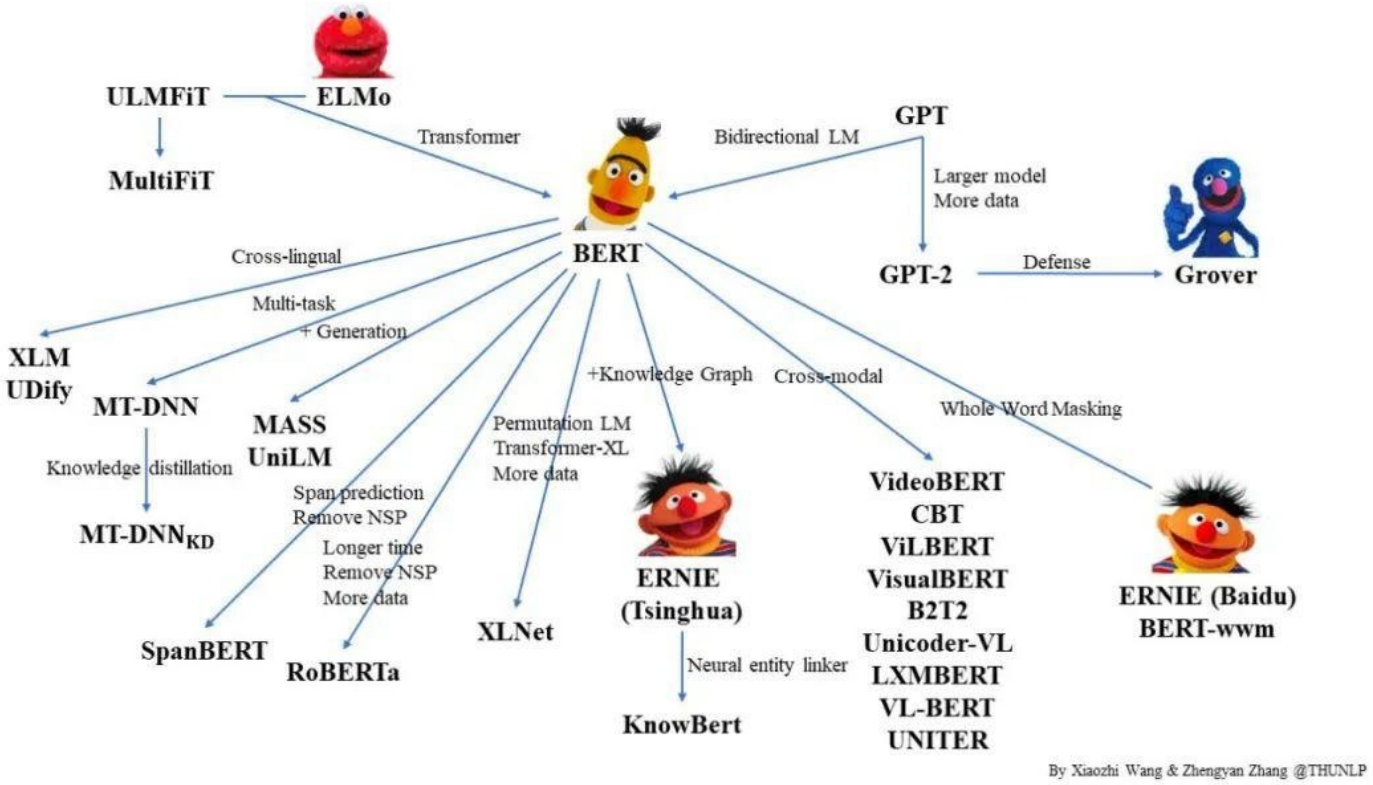
How will a machine know that these two words are related/similar?

```
judge = [ 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 ]
Appellant   = [ 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
```

TEXT VECTORIZATION//

Word embeddings

During the last years, NLP community has designed several word embeddings:



TEXT VECTORIZATION//

Word embeddings

But this has brought some concerns that we will discuss in session 5th

AI me to the Moon... Carbon footprint for 'training GPT-3' same as driving to our natural satellite and back

Get ready for Energy Star stickers on your robo-butlers, maybe?

TEXT VECTORIZATION//

Word embeddings

word2vec

The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text.

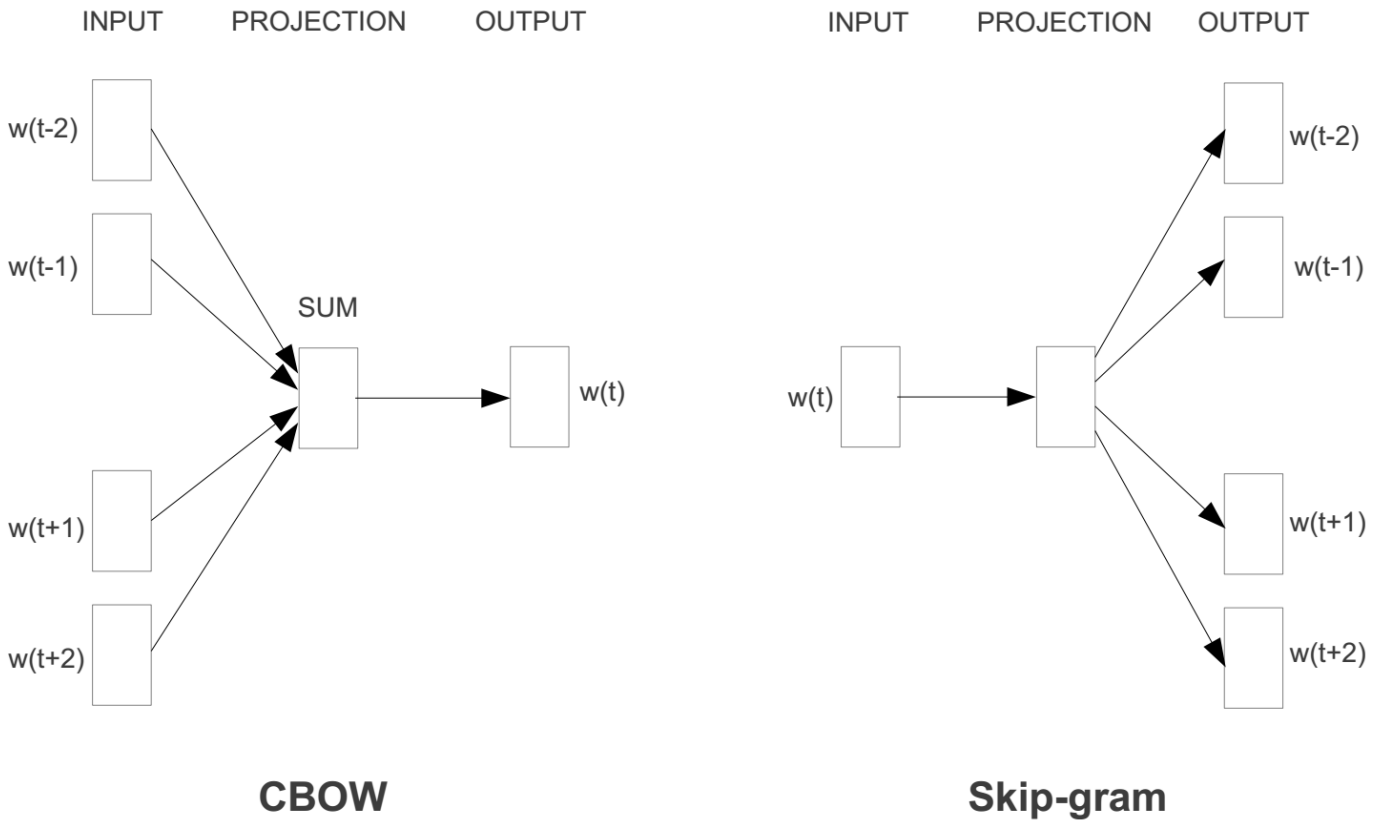
TEXT VECTORIZATION//

Word embeddings

word2vec

The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text.

word2vec (Mikolov et. al. 2013)

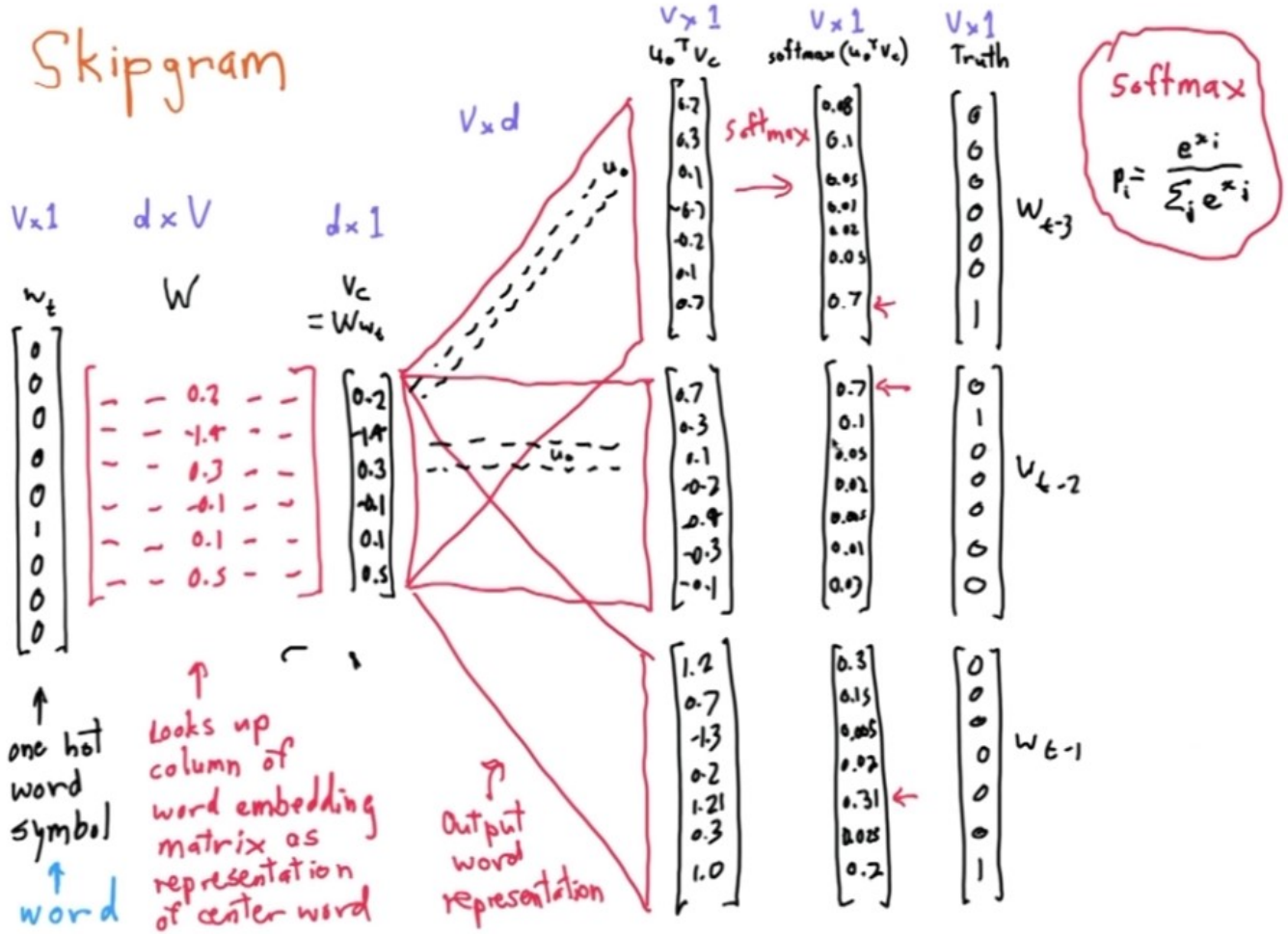


TEXT VECTORIZATION//

Word embeddings

word2vec

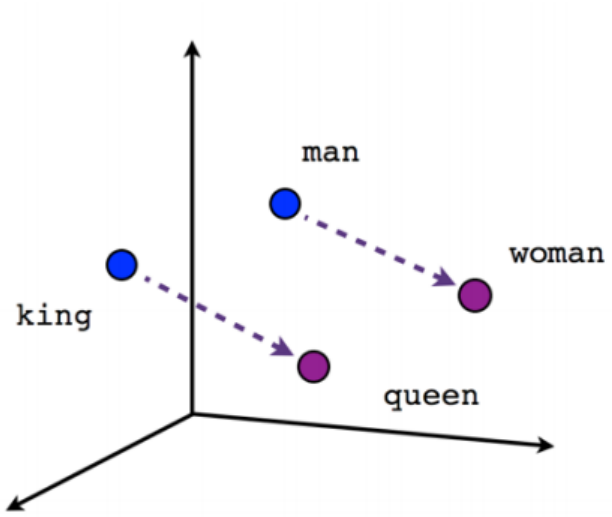
The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text.



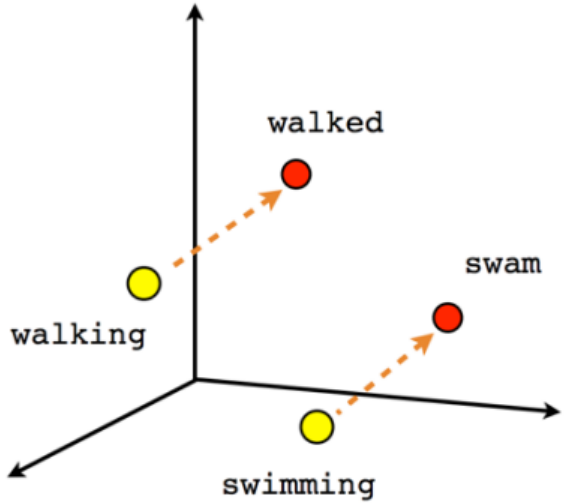
TEXT VECTORIZATION//

Word embeddings

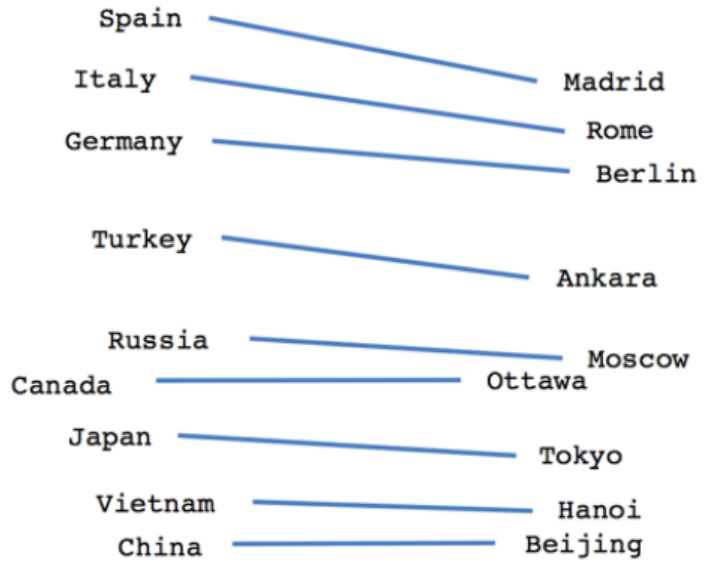
word2vec



Male-Female



Verb tense



Country-Capital

TEXT VECTORIZATION//

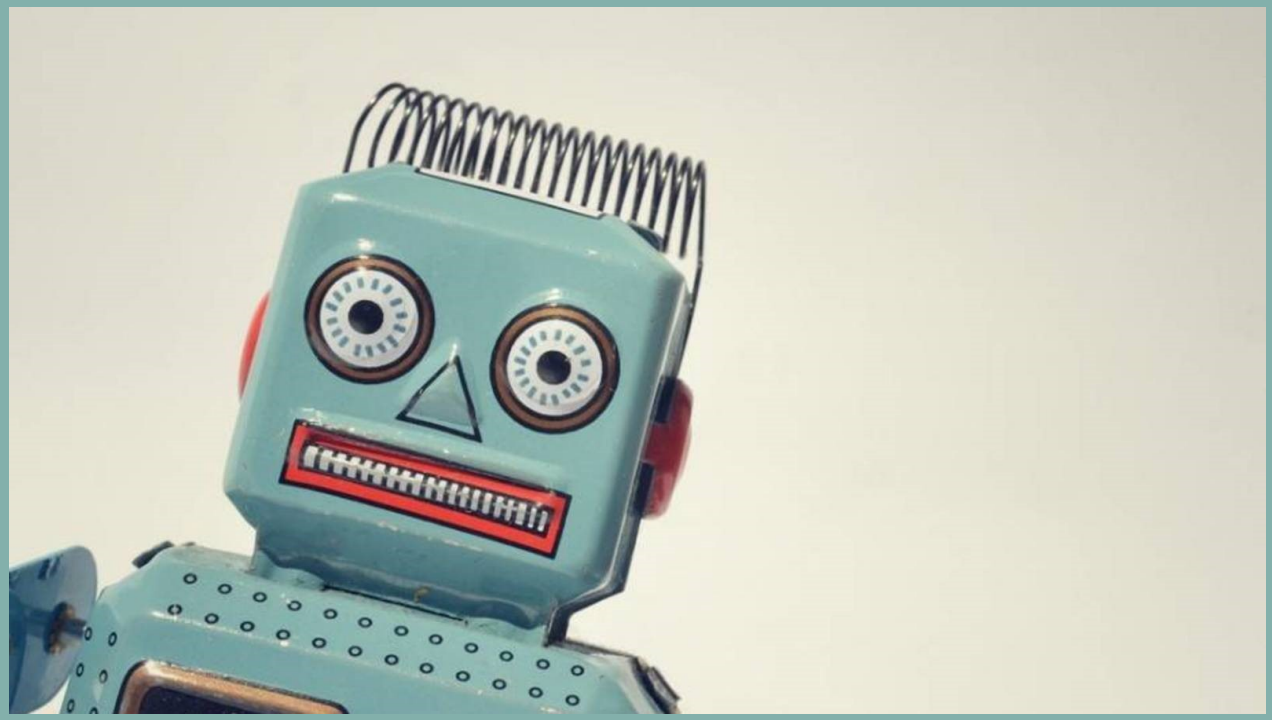
Word embeddings

Word embedding is the collective name for a set of language modeling and feature learning techniques in NLP where words from the vocabulary are mapped to vectors of real numbers.

- "The **judge** noted how the **Appellant** in his screening interview?"
- "During his **asylum** interview, however, the **Appellant** stated that he was a danger also because he was a **homosexual**."
- "The **Appellant** had not mentioned this at his **screening interview**."

- **Appellant** is related with **judge, asylum, homosexual, screening interview,** etc.

WORD PREPROCESSING //



WORD PREPROCESSING//

Tokenization

How many words are there in

“She went to Berlin”

and in

“She went to San Luis Obispo”?

WORD PREPROCESSING//

Tokenization

The process of separate symbols by introducing extra white space is called **tokenization**.

```
import spacy
nlp = spacy.load('en')

documents = "I've been 2 times to New York in 2011, but did not
    have the constitution for it. It DIDN'T appeal to me. I
    preferred Los Angeles."
tokens = [[token.text for token in sentence] for sentence in nlp
    (documents).sents]
```

```
[['I', "'ve", 'been', '2', 'times', 'to', 'New', 'York', 'in', '
    2011', ',', 'but', 'did', 'not', 'have', 'the', '
    constitution', 'for', 'it', '.'],
 ['It', "DIDN'T", 'appeal', 'to', 'me', '.'],
 ['I', 'preferred', 'Los', 'Angeles', '.']]
```

WORD PREPROCESSING//

Tokenization

How many words are there in

“她去了柏林”

and in

“她去了圣路易斯奥比斯波”?

WORD PREPROCESSING//

Lemmatization

The process of reducing words to its dictionary based (lemma) is called **lemmatization**.

```
[['I', "'ve", 'been', '2', 'times', 'to', 'New', 'York', 'in', '
    2011', ',', 'but', 'did', 'not', 'have', 'the', '
    constitution', 'for', 'it', '.'],
 ['It', "DIDN'T", 'appeal', 'to', 'me', '.'],
 ['I', 'preferred', 'Los', 'Angeles', '.']]
```

```
[['-PRON-', 'have', 'be', '2', 'time', 'to', 'new', 'york', 'in'
    , '2011', ',', 'but', 'do', 'not', 'have', 'the', '
    constitution', 'for', '-PRON-', '.'],
 ['-PRON-', "didn't", 'appeal', 'to', '-PRON-', '.'],
 ['-PRON-', 'prefer', 'los', 'angeles', '.']]
```

WORD PREPROCESSING//

Stemming

The process of reducing words to its stem is called **stemming**. This process is **more radical** than lemmatization.

```
[['I', "'ve", 'been', '2', 'times', 'to', 'New', 'York', 'in', '
    2011', ',', 'but', 'did', 'not', 'have', 'the', '
    constitution', 'for', 'it', '.'],
 ['It', "DIDN'T", 'appeal', 'to', 'me', '.'],
 ['I', 'preferred', 'Los', 'Angeles', '.']]
```

```
[['i', 've', 'been', '2', 'time', 'to', 'new', 'york', 'in', '
    2011', ',', 'but', 'did', 'not', 'have', 'the', 'constitut',
     'for', 'it', '.'],
 ['it', "didn't", 'appeal', 'to', 'me', '.'],
 ['i', 'prefer', 'los', 'angel', '.']]
```

WORD PREPROCESSING//

Tokenization, Lemmatization, and Stemming

T

```
[['I', "'ve", 'been', '2', 'times', 'to', 'New', 'York', 'in', '
    2011', ',', 'but', 'did', 'not', 'have', 'the', '
    constitution', 'for', 'it', '.'],
 ['It', "DIDN'T", 'appeal', 'to', 'me', '.'],
 ['I', 'preferred', 'Los', 'Angeles', '.']]
```

L

```
[['-PRON-', 'have', 'be', '2', 'time', 'to', 'new', 'york', 'in'
    , '2011', ',', 'but', 'do', 'not', 'have', 'the', '
    constitution', 'for', '-PRON-', '.'],
 ['-PRON-', "didn't", 'appeal', 'to', '-PRON-', '.'],
 ['-PRON-', 'prefer', 'los', 'angeles', '.']]
```

S

```
[['i', 've', 'been', '2', 'time', 'to', 'new', 'york', 'in', '
    2011', ',', 'but', 'did', 'not', 'have', 'the', 'constitut',
     'for', 'it', '.'],
 ['it', "didn't", 'appeal', 'to', 'me', '.'],
 ['i', 'prefer', 'los', 'angel', '.']]
```

WORD PREPROCESSING//

Part of speech (POS)

Part of speech corresponds to the process of classifying words to its category: nouns, verbs, adjectives, etc.

```
[['I', "'ve", 'been', '2', 'times', 'to', 'New', 'York', 'in', '
    2011', ',', 'but', 'did', 'not', 'have', 'the', '
    constitution', 'for', 'it', '.'],
 ['It', "DIDN'T", 'appeal', 'to', 'me', '.'],
 ['I', 'preferred', 'Los', 'Angeles', '.']]
```

```
[['PRON', 'VERB', 'VERB', 'NUM', 'NOUN', 'ADP', 'PROPN', 'PROPN'
    , 'ADP', 'NUM', 'PUNCT', 'CCONJ', 'VERB', 'ADV', 'VERB', '
    DET', 'NOUN', 'ADP', 'PRON', 'PUNCT'],
 ['PRON', 'PUNCT', 'VERB', 'ADP', 'PRON', 'PUNCT'],
 ['PRON', 'VERB', 'PROPN', 'PROPN', 'PUNCT']]
```

WORD PREPROCESSING//

Stopwords

Stopwords is the process of removing words that cannot be beneficial for the analysis, like determiners.

```
[['I', "'ve", 'been', '2', 'times', 'to', 'New', 'York', 'in', '
    2011', ',', 'but', 'did', 'not', 'have', 'the', '
    constitution', 'for', 'it', '.'],
 ['It', "DIDN'T", 'appeal', 'to', 'me', '.'],
 ['I', 'preferred', 'Los', 'Angeles', '.']]
```

```
[["'ve", 'times', 'New', 'York', 'constitution'],
 ['appeal'],
 ['preferred', 'Los', 'Angeles']]
```

WORD PREPROCESSING//

Parsing

Parsing is the process of classifying words in a sentence based on its syntax.

```
[['I', "'ve", 'been', '2', 'times', 'to', 'New', 'York', 'in', '
    2011', ',', 'but', 'did', 'not', 'have', 'the', '
    constitution', 'for', 'it', '.'],
 ['It', "DIDN'T", 'appeal', 'to', 'me', '.'],
 ['I', 'preferred', 'Los', 'Angeles', '.']]
```

```
[('I', 'been', 'nsubj'),
  ("'ve", 'been', 'aux'),
  ('been', 'been', 'ROOT'),
  ('2', 'been', 'npadvmod'),
  ('times', '2', 'quantmod'),
  ('to', '2', 'prep'),
  ('New', 'York', 'compound'),
  ('York', 'to', 'pobj'),
```

```
  ('in', 'been', 'prep'),
  ('2011', 'in', 'pobj'),
  (',', 'been', 'punct'),
  ('but', 'been', 'cc'),
  ('did', 'have', 'aux'),
  ('not', 'have', 'neg'),
  ('have', 'been', 'conj'),
  ('the', 'constitution', 'det'),
  ('constitution', 'have', 'dobj'),
  ('for', 'have', 'prep'),
  ('it', 'for', 'pobj'),
  ('.', 'been', 'punct')]
```

WORD PREPROCESSING//

Named Entity Recognition (NER)

Named Entity Recognition is the process of classifying words in a sentence based on its noun category (PERSON, FACILITY, ORGANIZATION, GEOPOLITICAL ENTITY, etc.).

```
[['I', "'ve", 'been', '2', 'times', 'to', 'New', 'York', 'in', '
    2011', ',', 'but', 'did', 'not', 'have', 'the', '
    constitution', 'for', 'it', '.'],
 ['It', "DIDN'T", 'appeal', 'to', 'me', '.'],
 ['I', 'preferred', 'Los', 'Angeles', '.']]
```

```
[[('2', 'CARDINAL'), ('New York', 'GPE'), ('2011', 'DATE')],
 [],
 [('Los Angeles', 'GPE')]]
```

WORD PREPROCESSING//

What if it's not English?

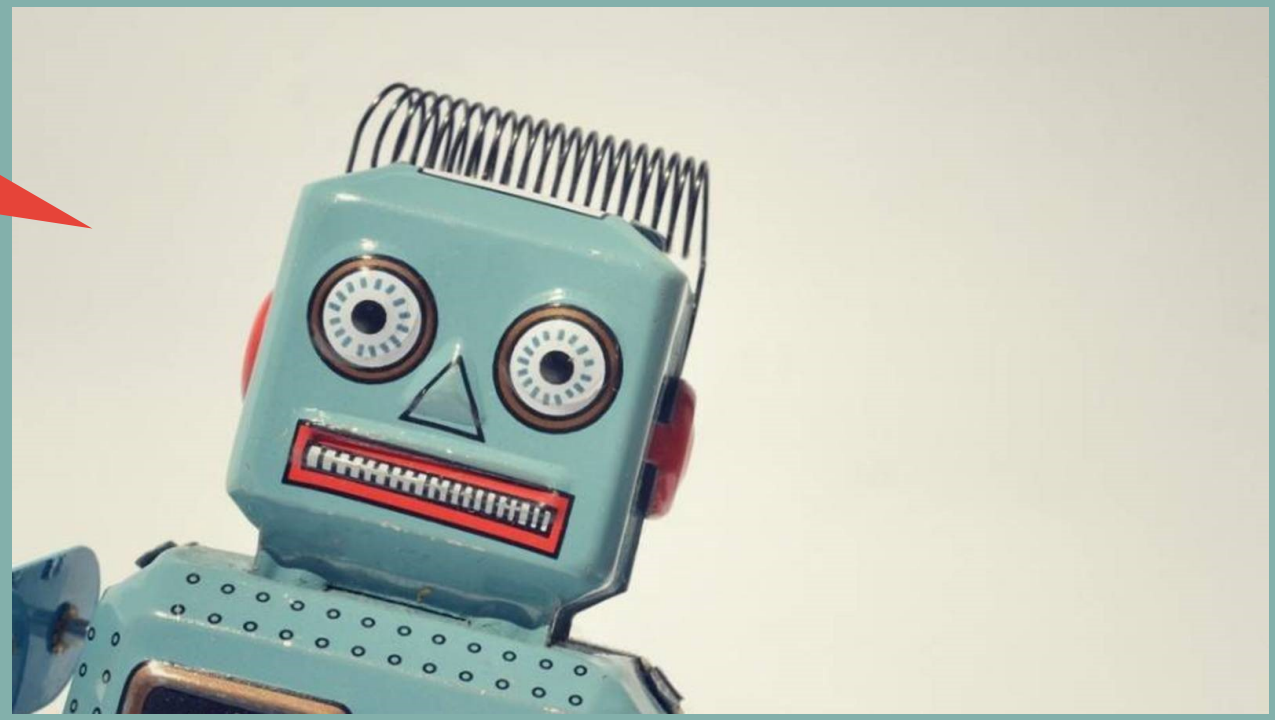
Say you have a whole load of Italian data that you want to work with, doing some of the things we have done in the previous sections. What are your options?

> spacy comes with support for a number of other languages, including German (de), Spanish (es), French (fr), Italian (it), Dutch (nl), and Portuguese (pt). All you have to do is load the correct library:

```
import spacy
nlp = spacy.load('it')
```

However, NLP is highly English-centered.

LET'S CODE!



WE'LL BE BACK IN 15 MIN...