NLP 2 - Word embeddings

SICSS Paris 2023

Julien Boelaert

CERAPS, Université de Lille

26/6/2023

Outline

Word embeddings: words as dense vectors

1/ Introduction: rationale

The technical argument for word embeddings is to progress from a **sparse** one-hot representation to a **dense** representation of terms.

Reminder: In a traditional DTM, each term is represented by a sparse one-hot vector: a lot of 0s (as many as size of dictionary), and a single 1 (in the dimension of the corresponding term).

- document-vector = sum of all its term vectors (still many 0s)
- problems: DTM = huge matrix, and no account taken of semantic proximity (*eg.* "cat" ≠ "cats")

	cat	hat	the	mad	and	on	mat
the	0	0	**1**	0	0	0	0
mad	0	0	0	**1**	0	0	0
cat	**1**	0	0	0	0	0	0
the mad cat	**1**	0	**1**	**1**	0	0	0

1/ Introduction: rationale

Intuition: it would be much more efficient to represent each word by a dense vector, of *eg.* 300 dimensions with no 0s.
Each word would live in this 300D **embedding space**.

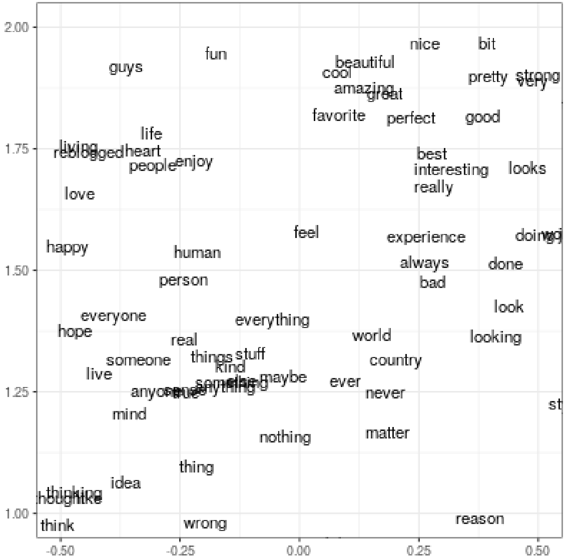
- no quality loss: one-hot representation is very inefficient
- quality gain, if embedding space captures semantic properties (*eg.* "cats" and "dog" closer to each other than to "liberty")

Typically, document-vector = mean of its word-vectors:

	D1	D2	D3	D4
the	2.24	0.10	0.11	-0.04
mad	-0.63	-0.04	1.92	0.73
cat	0.62	-0.85	0.05	0.83
the mad cat (sum)	2.23	-0.79	2.08	1.52
the mad cat (avg)	0.74	-0.26	0.69	0.51

NB: this is still BOW, but convnets and RNNs circumvent this

1/ Introduction: rationale



UMAP visualization of GloVe embeddings of some common words

1/ Introduction: history

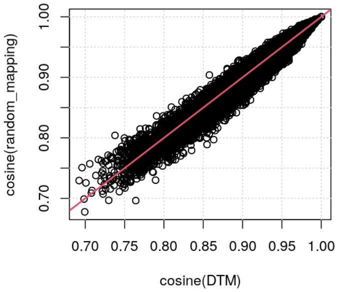
The history of word embeddings (and vector semantics) is longer than one might think:

- roots in structuralism, and the "distributionalist" linguistic theory of the 1950s:
  “A word is characterized by the company it keeps” (Firth, 1957)
- 1960s-1970s: "vector space model" for information retrieval
- 1990s: Random mapping
- 1990s: Latent Semantic Analysis
- 2000s: Neural network approaches
- 2010s: word2vec and co.

1/ Introduction: history

The most random way to do it: **random mapping**

- just map each word to a random dense vector
- doc-similarities are similar!
- even dim=90 is enough to *fingerprint* each term
- NB: this just approximates the DTM, the only benefit is lower dimensionality



SOTU: cosine sim, DTM vs random mapping (gaussian 90D)

S. Kaski, 1998, "Dimensionality reduction by random mapping: fast similarity computation for clustering", *Proc. IJCNN'98, Int. Joint Conference on Neural Networks*, Vol. 1, pp. 413—418.
S. Kaski, T. Honkela, K. Lagus, T. Kohonen, 1998, "WEBSOM — Self-organizing maps of document collections", *Neurocomputing* 21, pp. 101—117.

1/ Introduction: history

Latent Semantic Analysis (LSA):

- Compute SVD on TDM, to (linearly) compress its information
- Keep the first D dimensions (*eg.* 300 most important components) $\rightarrow$ embedding space
- Each term is now a D-dimensional vector
- Terms that appear in similar contexts have similar embeddings
- Many uses: language modeling, spell checking, essay grading...

NB: Alternatively, the SVM can be applied to a TCM.

S. C. Deerwester, S. T. Dumais, G. W. Furnas, R. A. Harshman, T. K. Landauer, K. E. Lochbaum, and L. Streeter, 1988, "Computer information retrieval using latent semantic structure", US Patent 4,839,853.
S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, 1990, "Indexing by latent semantics analysis", *JASIS* 41(6):391–407.
H. Schütze, 1992b, "Dimensions of meaning", *Proceedings of Supercomputing '92*, IEEE Press.

1/ Introduction: history

In the 2000s, neural network approaches yielded word embeddings as a by-product of language modeling (predicting the next word).

- **Transfer learning**: word embeddings computed on a big corpus, and re-used for different tasks
- word2vec and co. grew out of this literature, as a more efficient way to compute word embeddings, using simpler models that allow scaling to massive corpora.

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, 2003, "A neural probabilistic language model", *Journal of Machine Learning Research*, 3:1137–1155.
Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain, 2006, "Neural probabilistic language models", In *Innovations in Machine Learning*, pages 137–186. Springer.
R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, 2011, Natural language processing (almost) from scratch, *Journal of Machine Learning Research*, 12:2493–2537.

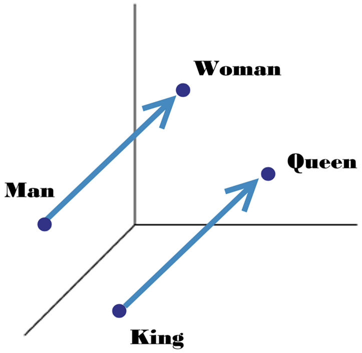
1/ Introduction: benefits

Benefits of modern embeddings:

- more efficient than DTM
- process synonyms (*eg.* "car" $\approx$ "automobile")
- multiple degrees of similarity
- embeddings analyzed for themselves (*eg.* gender bias)

Drawbacks:

- Vector semantics actually not so powerful
- Static embeddings $\rightarrow$ no polysemy



Analogy by vector semantics:
King + Woman - Man = Queen

1/ Introduction: benefits?

Vector semantics are actually not all-powerful! Bad at:

- lexicography: hypernyms (cat:feline), substance (sea:water), part-whole (car:engine), ...
- derivations: noun+less (life:lifeless), un+adj. (able:unable), verb+able (allow:allowable), ...

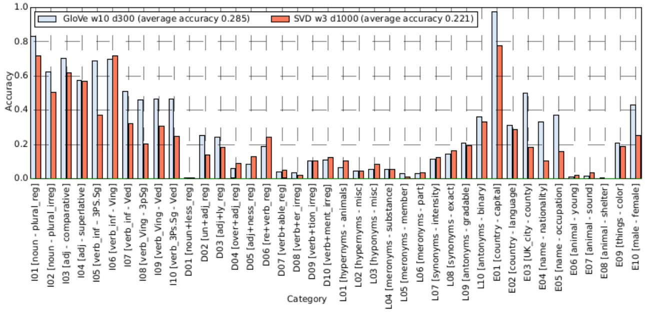


Figure 1: GloVe and SVD: accuracy on different types of relations

A. Gladkova, A. Drozd and S. Matsuoka, 2016, "Analogy-Based Detection of Morphological and Semantic Relations with Word Embeddings: What Works and What Doesn't", *Proceedings of the NAACL-HLT SRW*, 47–54. https://doi.org/10.18653/v1/N16-2002

2/ Embedding models

Three main algorithms:

- **word2vec** (v1 and v2, 2013, Google): made embedding learning efficient
- **fastText** (2017, Facebook): incorporates sub-word information
- **GloVe** (2014, Stanford): based on global corpus statistics

New word embedding models are still being developed, *eg.* to explicitly take into account a time variable.

2/ Embedding models: word2vec

word2vec version 1: embeddings computed by optimizing a **log-linear word prediction model**.

Training procedure (simplified):

- for each word in the vocabulary, create a vector (randomly initialized; dimension D is a hyperparameter)
- create a training sample by drawing tuplets of [target word, context words] (size of context is a hyperparameter)
- train a log-linear classifier: embedding linearly predicts word (output size = vocabulary size)

Example of *target* and **context** for window size 4:
"Meanwhile, **the mad** *cat* **was sitting** on the red mat."

T. Mikolov, K. Chen, G. S. Corrado and J. Dean, 2013, "Efficient estimation of word representations in vector space", *ICLR 2013*.

2/ Embedding models: word2vec

Two flavors of word2vec:

- **CBOW** (continuous bag of words): use (summed) vectors of context words to predict target word
- **Skip-gram**: use vector of target word to predict context words (actually 1-to-1, with words closer in context sampled more often)
- Not clear from the original results which flavor is best overall (skip-gram preferred in the subsequent literature?)

Original motivation: be more efficient than neural embeddings:

- much faster training
- scalable to massive corpora (billions of words)
- reusable embeddings $\rightarrow$ transfer learning

2/ Embedding models: word2vec

word2vec version 2: **negative sampling** for even faster training

Based on **data corruption**: recognize data from noise

- positive example: target-context couples that appear in corpus
- negative example: draw random context word from the corpus
- train logistic regression to recognize positive from negative (around $5\times$ more negative examples; more for small corpora)

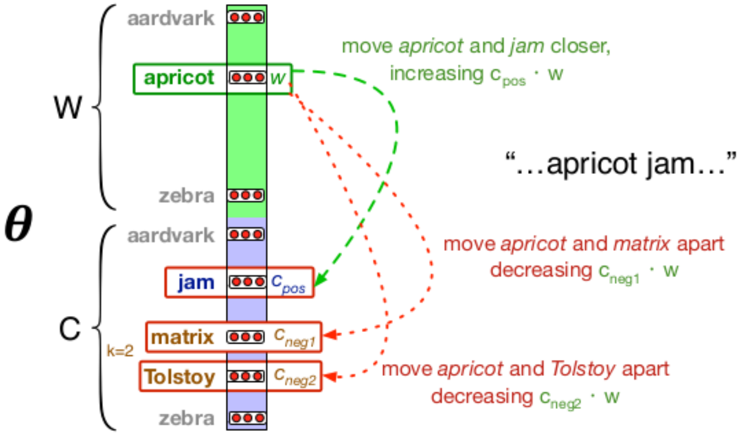
Example:

- Positive: "Meanwhile, the mad *cat* was **sitting** on the red mat."
- Negative: "Meanwhile, the mad *cat* was **constitution** on the red mat."

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, 2013, "Distributed representations of words and phrases and their compositionality", NeurIPS.

2/ Embedding models: word2vec

Negative sampling illustration:



W
aardvark
apricot
w
zebra
θ
C
aardvark
jam
c_{pos}
k=2
matrix
c_{neg1}
Tolstoy
c_{neg2}
zebra
move *apricot* and *jam* closer, increasing $c_{pos} \cdot w$
"…apricot jam…"
move *apricot* and *matrix* apart decreasing $c_{neg1} \cdot w$
move *apricot* and *Tolstoy* apart decreasing $c_{neg2} \cdot w$

2/ Embedding models: word2vec

word2vec v2 technical details:

- for each word i, two embeddings: target $\mathbf{v}_i$ and context $\mathbf{c}_i$
- $P(positive|i, j) = logistic(\mathbf{v_i} \cdot \mathbf{c}_j)$
- in the end, either sum them or discard context vectors.
- to accelerate training, under-sample very frequent words in training set

Overall:

- much **simpler** model: binary outcome instead of predicting word (out of thousands/millions possibilities)
- thus, **faster** to train and more scalable

2/ Embedding models: fastText

fastText: improve on word2vec by using sub-word information

- Each word is transformed into a **bag of character ngrams** (with additional boundary symbols $< >$)
  Example for "where" with 3-grams:
  "where" -> ("<where>", "<wh", "whe", "her", "ere", "re>")
- Each ngram has an embedding vector; word's embedding $=$ sum of its ngram embeddings
- In the paper, use all n-grams with $3 \leq n \leq 6$
- For the rest, training procedure is the same as word2vec with negative sampling (CBOW or skip-gram)

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, 2017, "Enriching word vectors with subword information", *TACL*, 5:135–146.

2/ Embedding models: fastText

fastText: word2vec with sub-word information

- Improves results on downstream tasks
- Using character n-grams is more important for Arabic, German and Russian than for English, French or Spanish (because of grammatical declensions, and compound words)
- Solves the out-of-vocabulary problem: unknown word is represented as the sum of known character n-grams

Pre-trained word vectors available for 157 languages: trained on Common Crawl and Wikipedia, CBOW with position-weights, 300-dim, character n-grams length 5, a window size 5, 10 negatives.
`https://fasttext.cc/docs/en/crawl-vectors.html`

2/ Embedding models: GloVe

GloVe: instead on relying on local context windows, use corpus-wide (global) word co-occurrence statistics

- First compute a term-term co-occurrence matrix (with a given context window size, *eg.* 10, and vocabulary size).
- Then learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence.
- Objective: $\sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - logX_{ij})^2$ where f is a weighting function that prevents very frequent co-occurrences from weighing too much, and infrequent ones too little. Final vectors $= w + \tilde{w}$.
- The result is computed slightly faster than word2vec, and yields better results on the word analogy task.

J. Pennington, R. Socher, and C. D. Manning, 2014, "GloVe: Global vectors for word representation", *EMNLP*.

2/ Embedding models

General technical remarks:

- each model has several hyperparameters, whose optimal values depend on the training corpus and the downstream task
- word vector sizes vary in 50-1000, typical size = 300
- training of word embeddings is a stochastic process: two consecutive runs (identical data and hyperpars) will yield different embeddings!
- fastText and GloVe perform similarly (and better than word2vec), with fastText having the additional benefit of solving the OOV problem

3/ Embeddings in practice

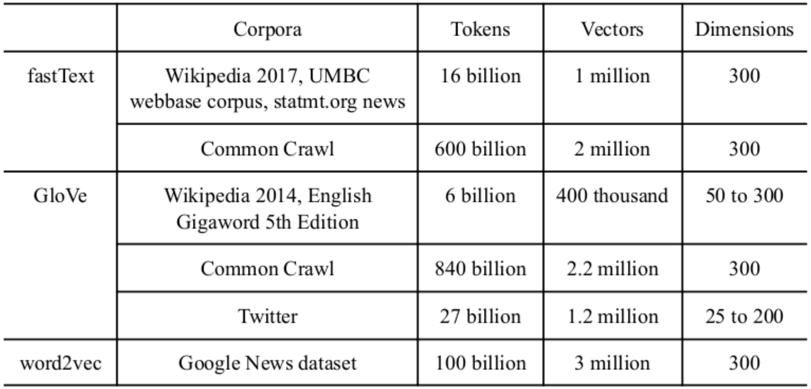
How to obtain word vectors?

- Either download vectors pre-trained (on wikipedia and CC)
  - GloVe (en): `https://nlp.stanford.edu/projects/glove/`
  - fastText (157 languages, vectors or model): `https://fasttext.cc/docs/en/crawl-vectors.html`
  - fastText aligned (44 languages): `https://fasttext.cc/docs/en/aligned-vectors.html`

- Or use open-source tools to compute them from your corpus
  - fastText (CLI or python): `https://fasttext.cc`
  - GloVe (CLI, linux or mac): `https://nlp.stanford.edu/projects/glove/`
  - NB: careful with the pre-processing (best to first tokenize with a powerful tool like spacy)

3/ Embeddings in practice

Details on common pre-trained vectors



	Corpora	Tokens	Vectors	Dimensions
fastText	Wikipedia 2017, UMBC webbase corpus, statmt.org news	16 billion	1 million	300
	Common Crawl	600 billion	2 million	300
GloVe	Wikipedia 2014, English Gigaword 5th Edition	6 billion	400 thousand	50 to 300
	Common Crawl	840 billion	2.2 million	300
	Twitter	27 billion	1.2 million	25 to 200
word2vec	Google News dataset	100 billion	3 million	300

(D. Stoltz and M. Taylor, "Cultural cartography with word embeddings", *Poetics*, Vol. 88, 2021)

3/ Embeddings in practice

What to do with word embeddings?

- either the **same things** you would do with a DTM (dim reduction, prediction, retrieval), hopefully better because good embeddings capture more meaning than one-hot encodings
- or **analyze them for themselves**: word embeddings mirror the stereotypical racial, ethnic, and gender related biases found in the texts they are trained on
  (NB: accurately reflect bias or exaggerate it? open debate)

3/ Embeddings in practice

How do we go from word embeddings to **documents**?

- standard CS approach: document vector = average of its word vectors (lots of information lost, BOW)
- use special model architectures (convolutional or recurrent neural networks) to treat each document as a sequence of word embeddings, and no longer as BOW (usually for prediction ; now superseded by transformer models)
- cloud approach: document = cloud of its word vectors
  Analyzed with Word Mover's Distance: doc-doc distances = "cost" of moving one document's cloud of vectors to another's

3/ Embeddings in practice

How are word embeddings typically used in social sciences ?

Interesting dichotomy proposed by D. Stoltz and M. Taylor ("Cultural cartography with word embeddings", *Poetics*, Vol. 88, 2021):

- "**variable embedding space** methods: (...) holding terms constant to measure how the meaning space moves around them (...) these methods entail splitting a corpus by a covariate — e.g., time periods or authors — and training word embeddings on each subset of the corpus"
  "much like early astronomers measured how the positions of celestial bodies changed across the seasons"

- "**fixed embedding space** methods use the same map of meaning space to measure how documents or authors change in relation to it"
  "just as ships use the stars at a given time to determine their location."

3/ Embeddings in practice

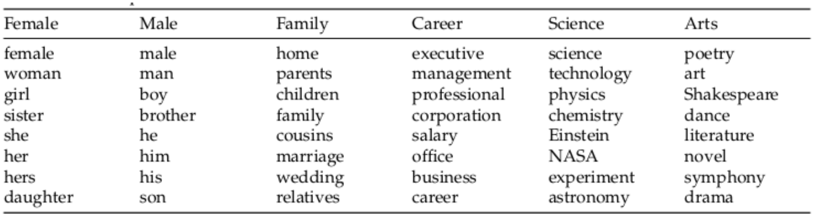
Example of use in sociology: (variable embedding space)
Jones, Jason J., Mohammad Ruhul Amin, Jessica Kim, and Steven Skiena. 2019. "Stereotypical Gender Associations in Language Have Decreased Over Time." *Sociological Science* 7: 1-35.

- Goal: "quantify the association between gender and stereotypically gendered domains (career, family, science, and arts) latent within language patterns"
- Data: published books from every decade from 1800 to 2000 (from Google N-Grams "All English" corpus)
- Embeddings: word2vec, computed by decade by the HistWords team (Hamilton, Leskovec and Jurafsky, `https://nlp.stanford.edu/projects/histwords/`)

3/ Embeddings in practice

Method:

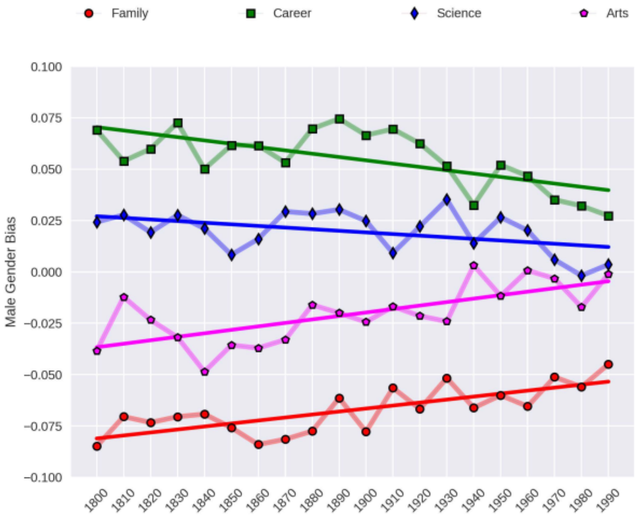
- Choose list of words representing each gender and each theme
- Compute mean cosine similarity between all combinations of words between two lists $\rightarrow$ association between a gender and a theme (for each decade)
- Gender bias = association(M, theme) - association(F, theme)



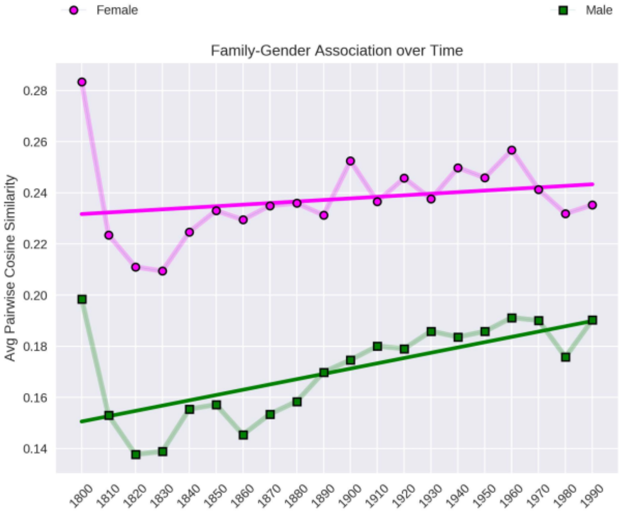
Female	Male	Family	Career	Science	Arts
female	male	home	executive	science	poetry
woman	man	parents	management	technology	art
girl	boy	children	professional	physics	Shakespeare
sister	brother	family	corporation	chemistry	dance
she	he	cousins	salary	Einstein	literature
her	him	marriage	office	NASA	novel
hers	his	wedding	business	experiment	symphony
daughter	son	relatives	career	astronomy	drama

3/ Embeddings in practice

Result: gender bias has decreased over time



3/ Embeddings in practice



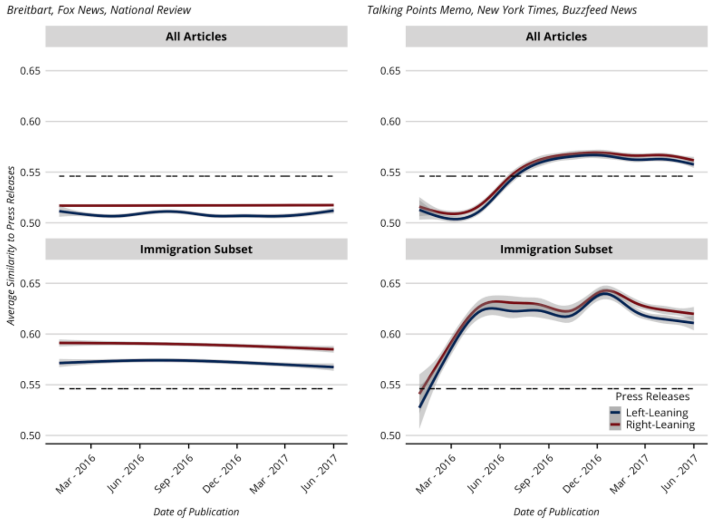
3/ Embeddings in practice

Example of use in sociology: (fixed embedding space)
D. Stoltz and M. Taylor, "Cultural cartography with word embeddings", *Poetics*, Vol. 88, 2021

- Goal: quantify the similarity between news outlets and immigration-focus advocacy organizations, according to political leaning
- Data: news articles from 2013-2018, right-leaning (Breitbart, Fox News, and National Review) and more left-leaning (Talking Points Memo, New York Times, and Buzzfeed News); 986 press releases from two extreme-right and two left-wing organizations (1998-2020).
- Embeddings: unspecified (presumably pre-trained)
- Method: WMD from news articles per year to press releases (pooled by organization)

3/ Embeddings in practice

Result: right-leaning press closer to extreme-right organizations; left-leaning press closer to both left and right press releases



Further reading

- Dan Jurafsky and James H. Martin, 2023, *Speech and Language Processing* (3rd ed. draft), chapter I.6 `https://web.stanford.edu/~jurafsky/slp3`
- Yoav Goldberg, 2016, "A primer on neural network models for natural language processing", *The Journal of Artificial Intelligence Research*, 57, 345–420.

Questions and comments: `julien.boelaert@univ-lille.fr`