Discovering Embeddings

SICSS Paris '25

Julien Boelaert

CERAPS, Université de Lille

23/06/2025

Outline

1. Introduction: embedding text
2. Principles and history
3. Word embeddings in some detail
4. Uses

1. Introduction

Motivation: do better than document-term or term-cooccurence matrices

- Short and dense vs. Long and sparse
- Encode meaning / semantics better
- Semantic algebra? "Paris - France + Spain = ?"
- Transfer learning: benefit from others' big data

1. Introduction

Principle: **Words / documents as vectors**

- Similar words/docs have similar vectors
- Word embeddings (word2vec, fastText, GloVe)
- Sentences as mean(word embbedings)
- Sentence embeddings (SBERT family)

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" $\neq$ "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 dense **embedding space** (instead of a much higher-dimension sparse tokens 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")

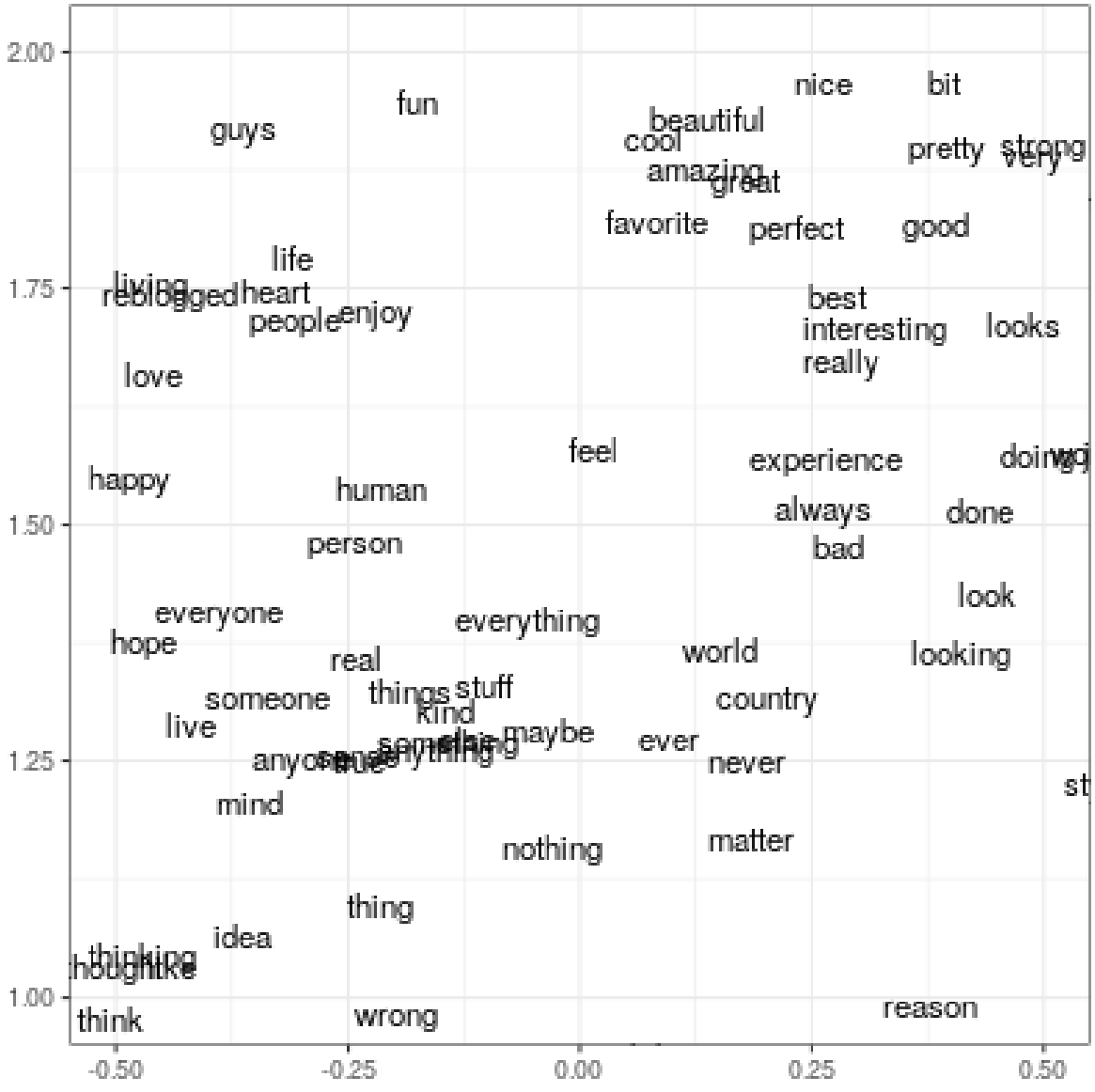
1/ Introduction: rationale

From word embeddings to **document embeddings**:
document-vector = sum (or 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 (order doesn't matter)

1/ Introduction: rationale



UMAP visualization of GloVe embeddings of some common words

1. Introduction: uses

Uses:

- General NLP: better retrieval, clustering, prediction

1. Introduction: uses

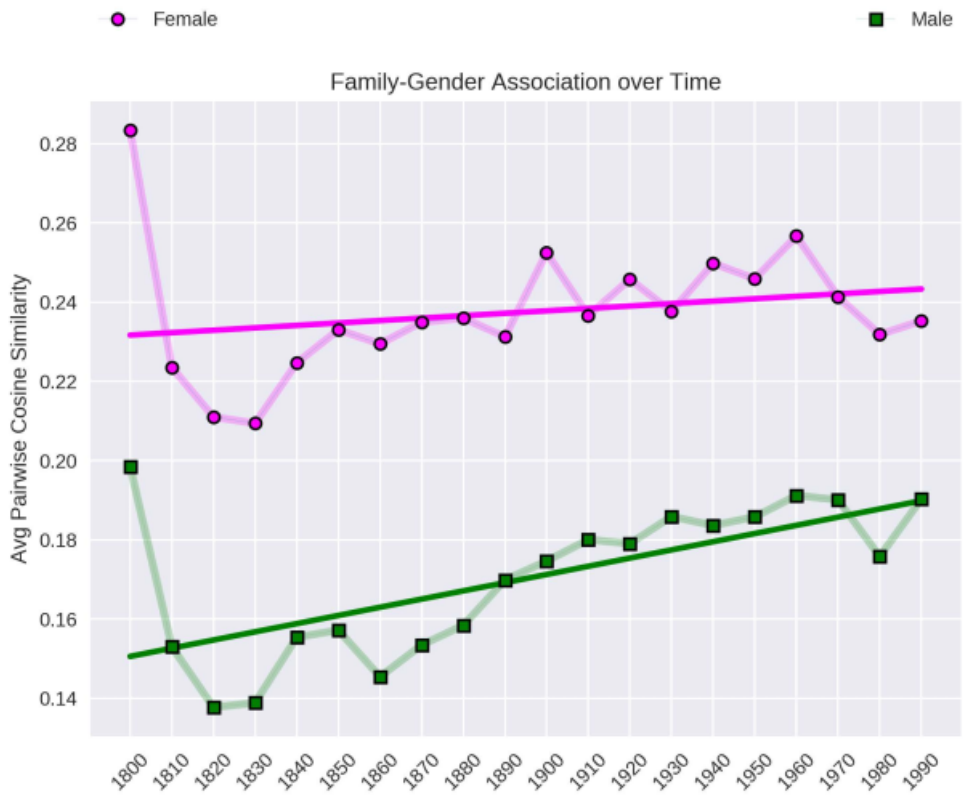
Uses:

- General NLP: better retrieval, clustering, prediction
- (Soc. sci) Induce social phenomena from word embeddings
  - Biases: "Doctor - Man + Woman = ?"

1. Introduction: uses

Uses:

- General NLP: better retrieval, clustering, prediction
- (Soc. sci) Induce social phenomena from word embeddings
  - Biases: "Doctor - Man + Woman = ?"
  - Word associations over time



(Jones *et al*, 2019)

2. Principles and History

2. Principles and History

- **Similarity**
  - What's in a similarity?
  - Measuring similarities
- **History**
  - Random vectors
  - Linear reductions
  - Word2vec and friends
  - SBERT and friends

2a. Similarity

What's in a similarity? (Jurafski and Martin, 6.1)

- **Synonyms**: car/automobile, friend/buddy
- **Antonyms**: friend/enemy, wet/dry
- **Relatedness / Association**: student/university, pan/cook
- **Semantic field**: tractor/corn/farm vs. trading/securities/bank
- **Semantic frame**: "Sam bought the book from Ling" = "Ling sold the book to Sam"
- **Connotations / Sentiment**: innocent vs. naive ; (great, love) vs. (terrible, hate)

2a. What's in a similarity?

Word embedding approach:

define a word by its context
turn this into numbers

2a. What's in a similarity?

Word embedding approach:

define a word by its context
turn this into numbers

word	embedding	context
"lemon"	(0.3, 1.4, -0.9, ..., -0.7, 0.1)	("juice", "tree", "car", ...)
"orange"	(0.2, 1.5, -0.8, ..., 1.2, -0.5)	("juice", "tree", "clockwork", ...)

similar word $\leftarrow$ similar embedding $\leftarrow$ similar context

2a. What's in a similarity?

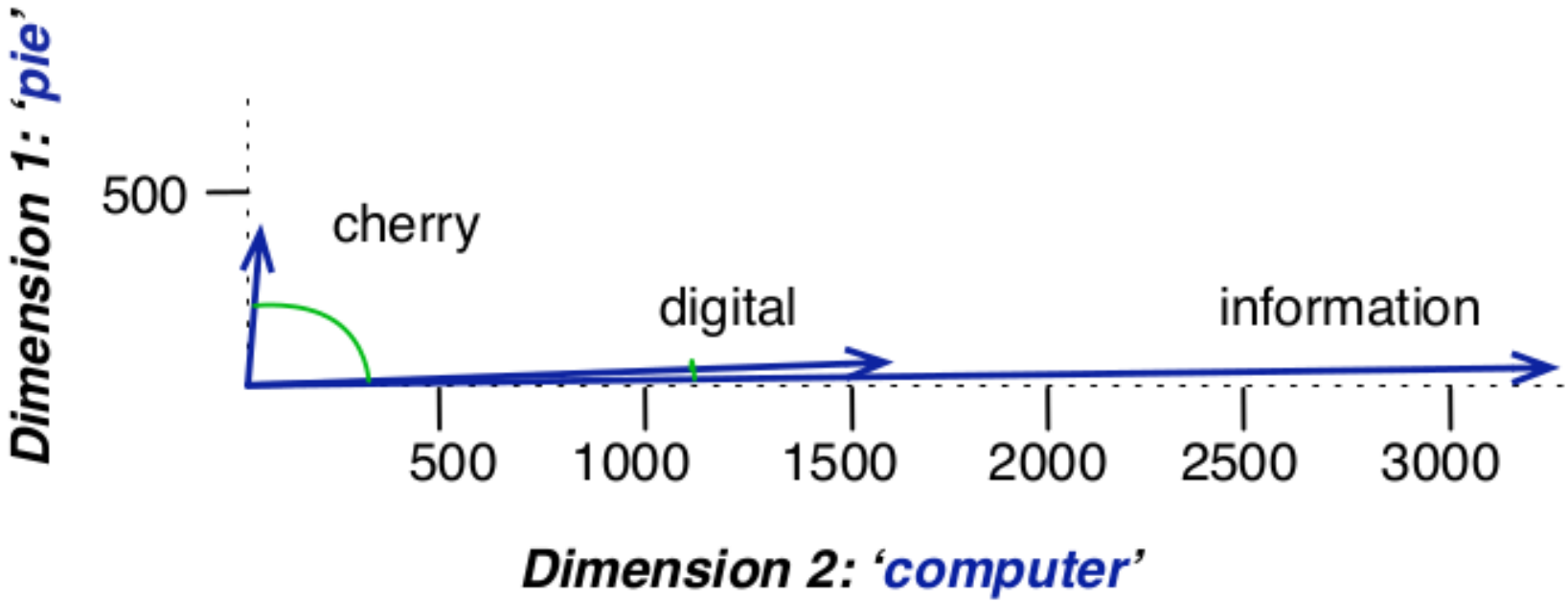
Theoretical roots of "word = context":

- Structuralist? Saussure : "linguistic signs are unrelated to what they designate and that, therefore, 'a' cannot designate anything without the the aid of 'b' and vice versa, or, in other words, that both have value only by the difference between them."
- Wittgenstein "the meaning of a word is its use in the language" (1953)
- "Distributionalist" linguistic theory of the 1950s: "A word is characterized by the company it keeps" (Firth, 1957)

2a. Measuring similarity

But how do we measure the similarities of embedding vectors?

Ex. on term co-occurence matrix



(Jurafsky and Martin, 2023, 6.2)

$\rightarrow$ **similarity = direction**

2a. Measuring similarity

But how do we measure the similarities of embedding vectors?

- Euclidean distance? too sensitive to magnitude
- Better measure angles $\rightarrow$ **cosine similarity**

$$cos(u, v) = \frac{u \cdot v}{|u| \times |v|}$$

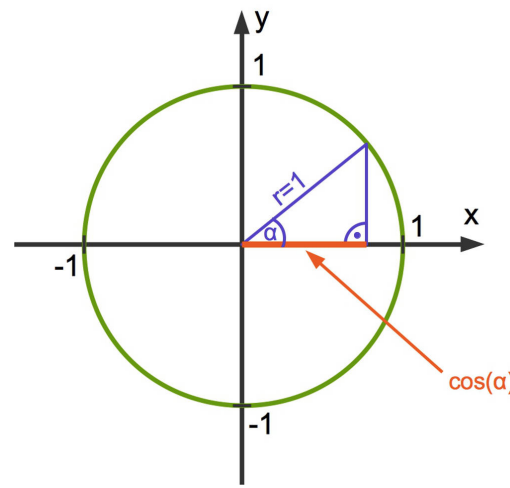
2a. Measuring similarity

But how do we measure the similarities of embedding vectors?

- Euclidean distance? too sensitive to magnitude
- Better measure angles $\rightarrow$ **cosine similarity**

$$cos(u, v) = \frac{u \cdot v}{|u| \times |v|}$$

$\cos = 1$	perfectly aligned
$0 \leq \cos \leq 1$	similar general direction
$\cos = 0$	orthogonal
$-1 \leq \cos \leq 0$	different general direction
$\cos = -1$	opposite direction



2a. Measuring similarity

At the heart of the cosine: the **dot product**

$$u \cdot v = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_D \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_D \end{bmatrix} = u_1 \times v_1 + u_2 \times v_2 + \cdots + u_D \times v_D$$

2a. Measuring similarity

At the heart of the cosine: the **dot product**

$$u \cdot v = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_D \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_D \end{bmatrix} = u_1 \times v_1 + u_2 \times v_2 + \cdots + u_D \times v_D$$

- big when u and v aligned
- 0 when orthogonal
- small (negative) when aligned in opposite directions

$\rightarrow$ cosine similarity = normalized dot product

2a. Measuring similarity

Recap

- word $\rightarrow$ embedding vector
- embeddings are a representation of the contexts
- similar words =
  - similar context
  - similar vector direction
  - large dot product
  - cosine close to 1

NB: **cosine distance = 1 - similarity**

2a. Measuring similarity

Caution! cosine(embeddings) can be very counter-intuitive
Don't necessarily conform with semantic intuition of association, especially synonymity.

- Antonyms may have high similarity: ex on GloVe CC (850B)
  - cos("always", "never") = 0.85 (2d out of 2M)
  - cos("good", "bad") = 0.74 (12th out of 2M)
- Cosine similarities on word embeddings can be negative
  - sim=1: always in the same context
    sim=0: never
    sim<0: unclear...
  - averaging cosine similarities is dangerous

→ **Beware of over-interpretation!**

2b. History

2b. History

Semantic features (1940s-50s): hand-crafted features representing concepts

hen	+female, +chicken, +adult
rooster	-female, +chicken, +adult
chick	+chicken, -adult

(Jurafsky and Martin, 2023, 6)

2b. History

Semantic features (1940s-50s): hand-crafted features representing concepts

hen	+female, +chicken, +adult
rooster	-female, +chicken, +adult
chick	+chicken, -adult

(Jurafsky and Martin, 2023, 6)

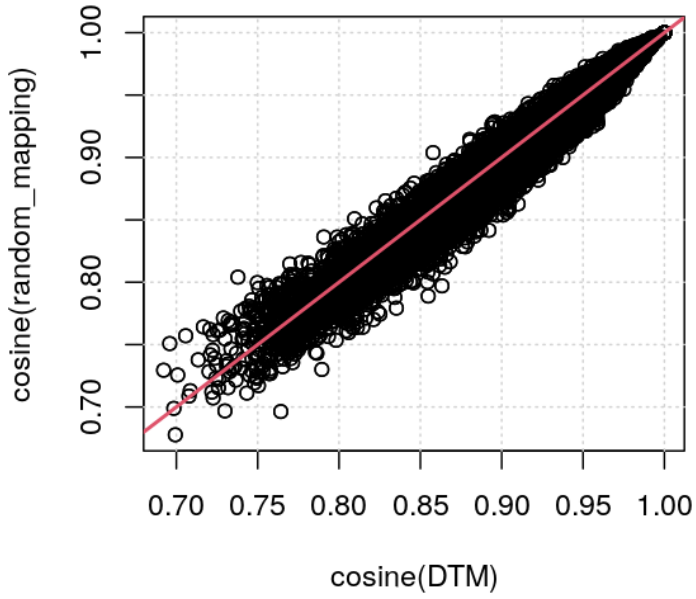
Vector space model (1950s-60s)

- Made for information retrieval (best documents for a query)
- Based on DTMs, word similarity from statistical association
- Documents in the same vector spaces used for words

2b. History

The most random way to do it: **random mapping** (1980s)

- 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.

2b. History

Latent Semantic Analysis (LSA, 1990s):

- Compute SVD (singular value decomposition) 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 SVD 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.

2b. History

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

- **Self-supervized learning**: supervized, but on labels that are already in the base corpus (next word, context words)
  $\rightarrow$ no need for costly human annotation
- **Transfer learning**: embeddings computed on a big corpus, and re-used for different tasks on other corpora

2b. History

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

- **Self-supervized learning**: supervized, but on labels that are already in the base corpus (next word, context words)
  $\rightarrow$ no need for costly human annotation
- **Transfer learning**: embeddings computed on a big corpus, and re-used for different tasks on other corpora

In the 2010s, **word2vec** and co. grew out of this literature: .

- simpler models, more efficient, scalable to massive corpora
- specifically designed to produce word embeddings

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.

2b. History

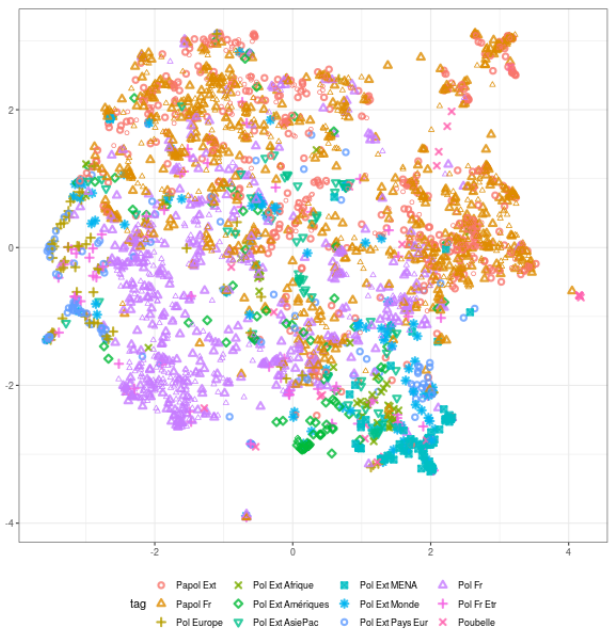
Sentence embeddings: SBERT family (2020s)

- Transformer age: process sentences/paragraphs, not words
- Static embeddings for documents
- Finally free from bag-of-words

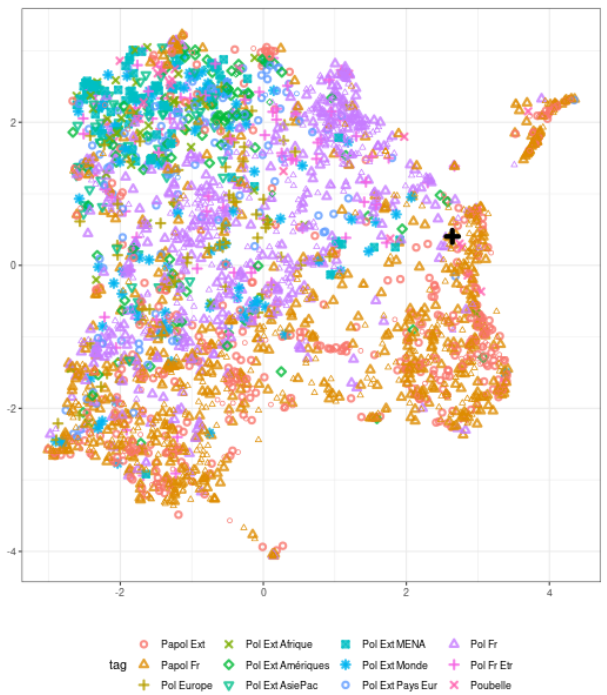
2b. History

Sentence embeddings: SBERT family (2020s)

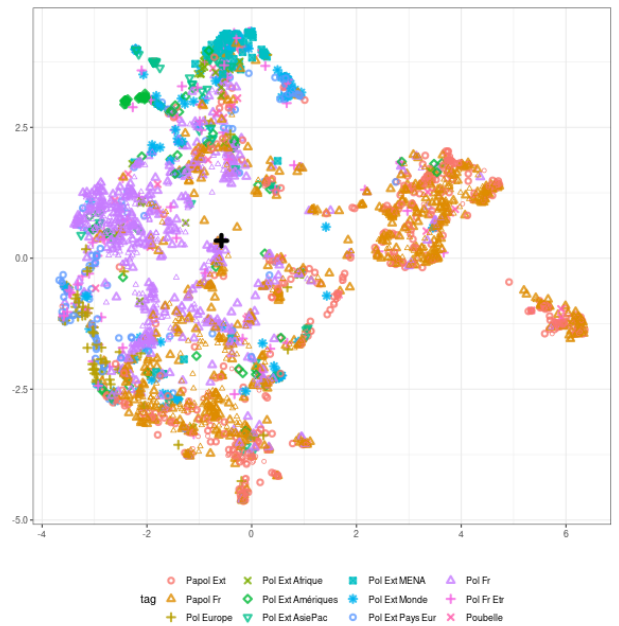
- Transformer age: process sentences/paragraphs, not words
- Static embeddings for documents
- Finally free from bag-of-words



UMAP on DFM



UMAP on fasttext



UMAP on SBERT

3. Word embeddings

3. Word embeddings

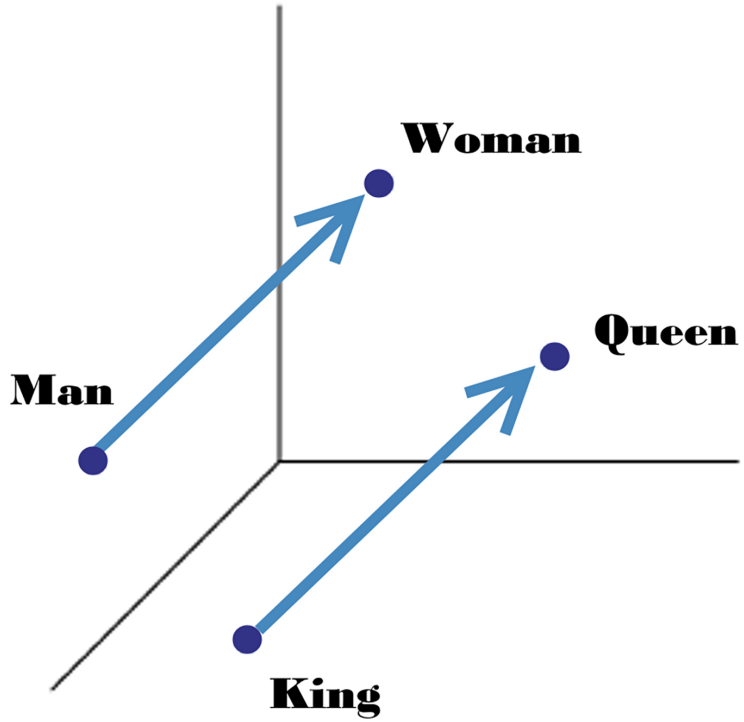
word2vec and friends:

- embeddings as solutions to classification tasks
- self-supervised learning, from huge corpora
- models specifically tailored for efficient embedding
- built-in dot-product similarity

3. Word embeddings

The word2vec hype:
"Algebraic semantics"

"Somewhat surprisingly, these [word similarity] questions can be answered by performing simple algebraic operations with the vector representation of words."



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

3. Word embeddings

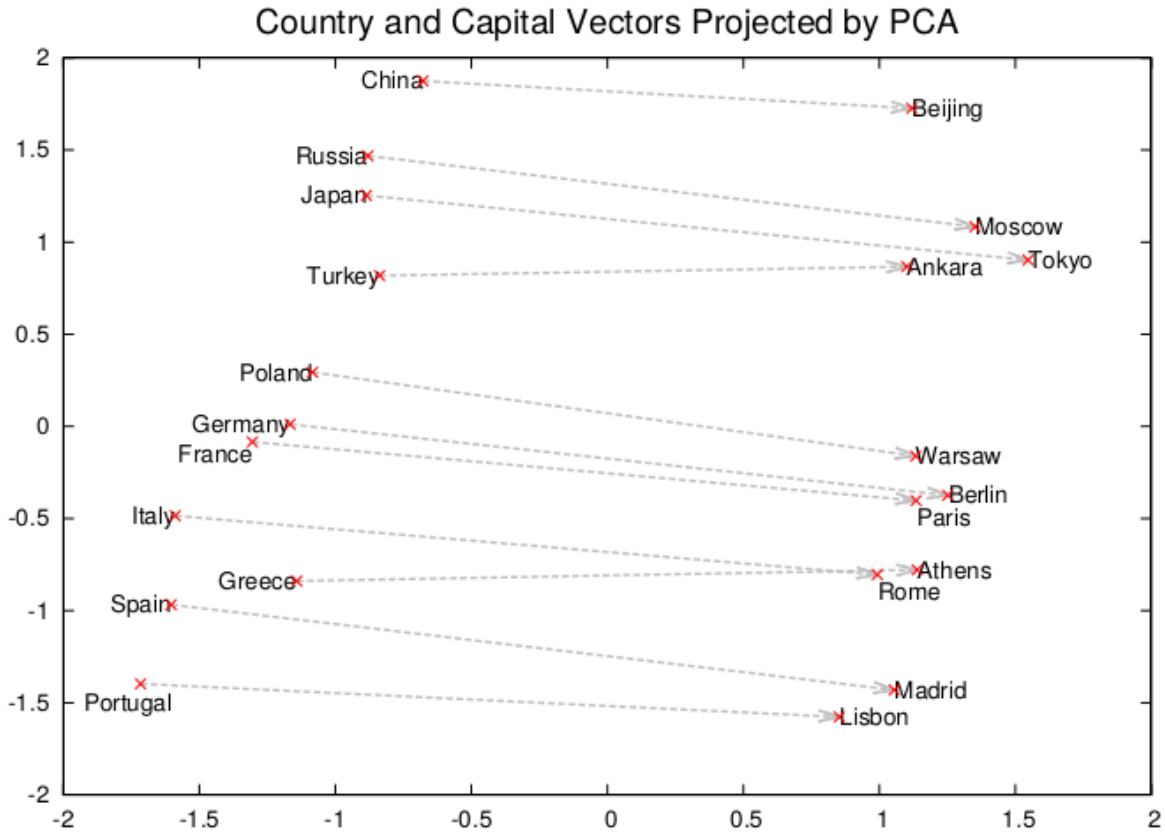


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

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

3. Word embeddings

Table 1: *Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set.*

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

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

3. Word embeddings

BUT: Algebraic semantics are actually 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), ...

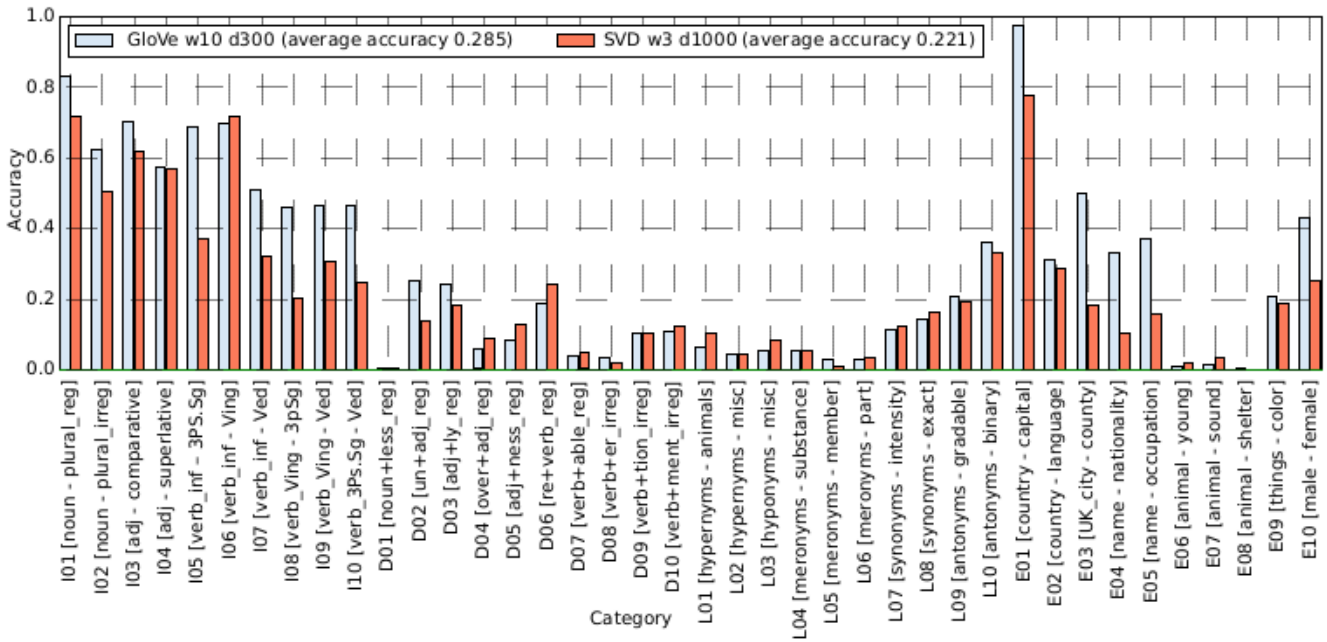


Figure 1: GloVe and SVD: accuracy on different types of relations

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

3. Word embeddings

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.

3. Word embeddings: word2vec

word2vec version 1: which word is likely to be next to "cat"?

Log-linear word prediction:

- predict target word from context (or reverse)
- input: embedding, output: probabilities of all words
- core: dot-product of embeddings
- data: context-target pairs drawn from raw text

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

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*.

3. Word embeddings: word2vec

Two flavors of word2vec (and most word embedding methods):

- **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?)

3. Word embeddings: word2vec

word2vec version 2: do “cat” and “mat” do well together?

Binary classifier: simpler (1 output, not V), dot-product core.

Recognize true from false target-context couples:

- true couple from the data:

  ("cat", "mat")

- false couples by random corruption (**negative sampling**)

  ("cat", "constitution"), ("cat", "logistic") ...

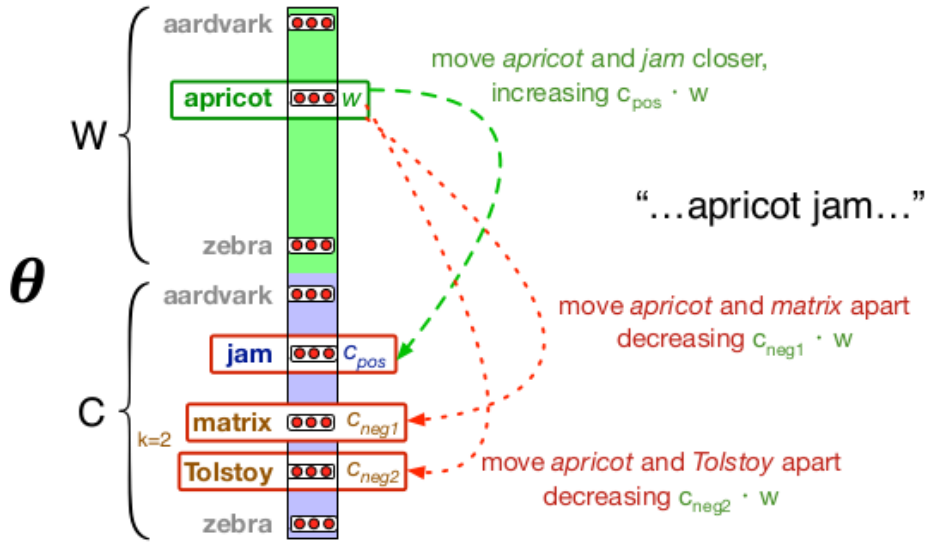
$$P(+|w,c) = \sigma(\mathbf{c}\cdot\mathbf{w}) = \frac{1}{1+\exp(-\mathbf{c}\cdot\mathbf{w})}$$

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

3. Word embeddings: word2vec

word2vec version 2 details:

- two vectors for each word (target, context), as in v1
- after training, either sum them or discard context
- 5x negatives for each positive sample (more for small corpora)
- under-sample very frequent words



3. Word embeddings: 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, same procedure as word2vec v2

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, 2017, "Enriching word vectors with subword information", *Transactions of the Association for Computational Linguistics*, 5:135–146.

3. Word embeddings: 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`

3. Word embeddings: GloVe

GloVe: predict co-occurence by embeddings dot product

$$\mathbf{w_i} \cdot \tilde{\mathbf{w}}_\mathbf{j} \sim log(cooccurrence(i, j))$$

- Based on global corpus-wide statistics (instead of target-context samples): term-term co-occurrence matrix with a given context window size, *eg.* 10
- Again, two embeddings per word ($\mathbf{w_i}$ target, $\tilde{\mathbf{w}}_\mathbf{i}$ context); final vectors $= \mathbf{w_i} + \tilde{\mathbf{w}}_\mathbf{i}$
- In practice, weighting function prevents very frequent co-occurrences from weighing too much, and infrequent ones too little
- Slightly faster than word2vec, better results on the word analogy task.

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

3. Word embeddings

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. Word embeddings

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 vectors (44 languages): `https://fasttext.cc/docs/en/aligned-vectors.html`
  - HistWords: word2vec by decade (1800-2000) on 4 languages (English, French, German, and Mandarin), diverse corpora: `https://nlp.stanford.edu/projects/histwords/`
- 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. Word embeddings

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. Word embeddings

Benefits of modern word embeddings:

- More efficient than DTM
- Process synonyms ("car" $\approx$ "automobile")
- Multiple degrees of similarity
  ("mouse" close to "cat" but also to "keyboard" and "Mickey")
- Embeddings analyzed for themselves (*eg.* gender bias)

Drawbacks:

- Vector semantics actually not so powerful
- Similarity can be confusing (not only synonymity)
- Static embeddings $\rightarrow$ no context awareness
- In most cases, documents still processed as bag-of-words

Hence the need for transformer-based methods (incl. SBERT).

4/ Working with embeddings

4/ Working with embeddings

- NLP uses
  - information retrieval
  - supervized: cheap prediction
  - unsupervized: visualization, clustering
- Interpretation of similarities

4/ Working with embeddings

Information retrieval: in a corpus of documents, find the ones most similar to a given "query" document ($\sim$ search engine)

1. embed query document and all the corpus ("keys")

2. retrieve the documents with the highest similarity to query

The better the embeddings, the better the results
($\rightarrow$ use SBERT-type model).

4/ Working with embeddings

Supervized: based on a limited set of annotated data, **predict** the class of new data ("does this abstract contain a gender angle?")

1. embed documents (annotated and new)
2. train classifier to predict labels from embeddings
3. predict on new data, better than with DTM

"Cheap" wrt to BERT: no GPU required, trained in a few seconds

4/ Working with embeddings

Unsupervized: visualize and/or cluster texts

1. embed documents (annotated and new)
2. run visualization and/or clustering algorithm

BERTopic: SBERT + UMAP + DBClust (+ tf-idf interpretation)

- faster than LDA
- no underlying statistical model, but better underlying semantic treatment
- python package bertopic

M. Grootendorst (2022), "BERTopic: Neural topic modeling with a class-based TF-IDF procedure", https://arxiv.org/abs/2203.05794

4/ Working with embeddings

In social sciences: **interpretation of similarities**

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."

4/ Working with embeddings

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/`)

4/ Working with embeddings

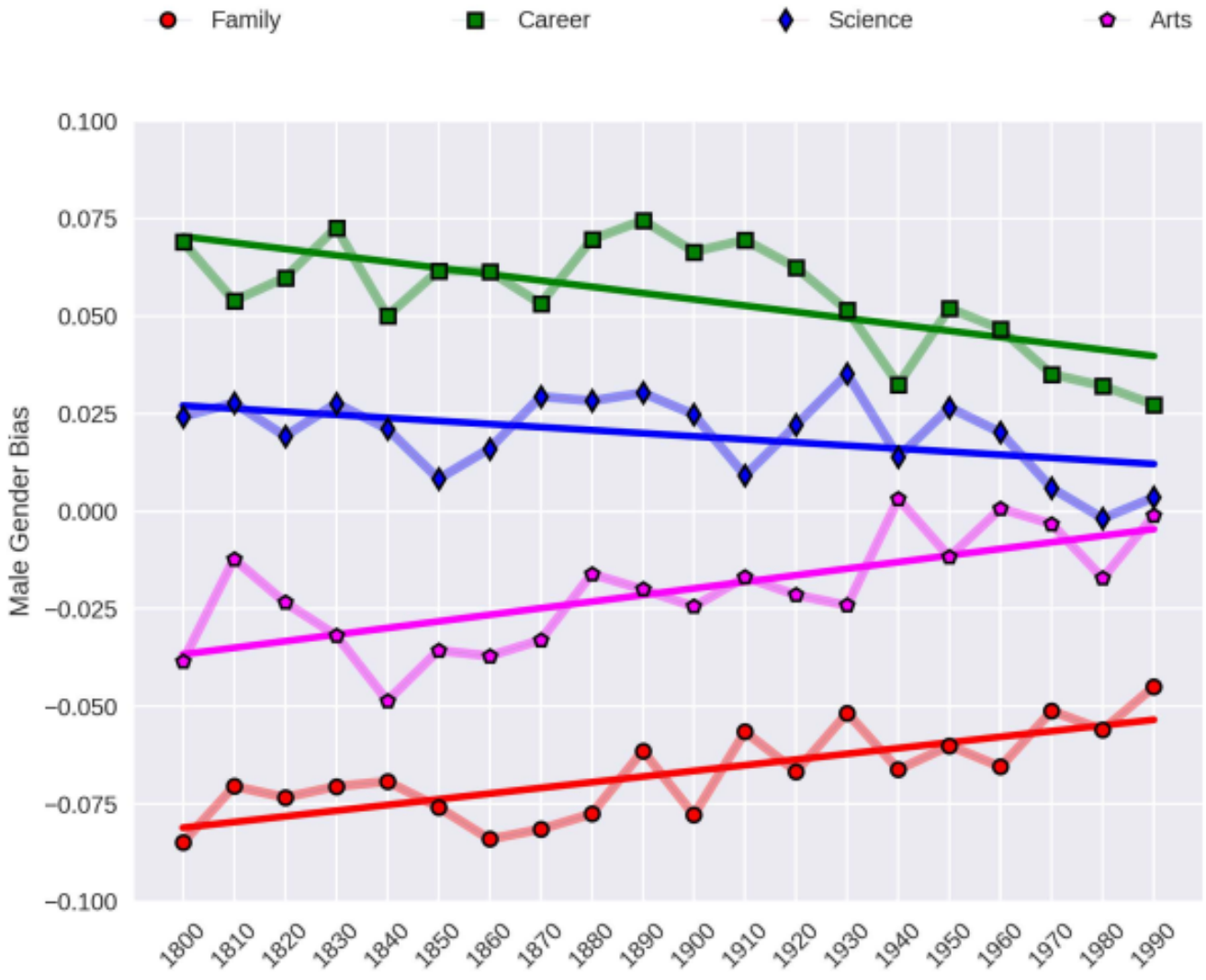
Method:

- 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)

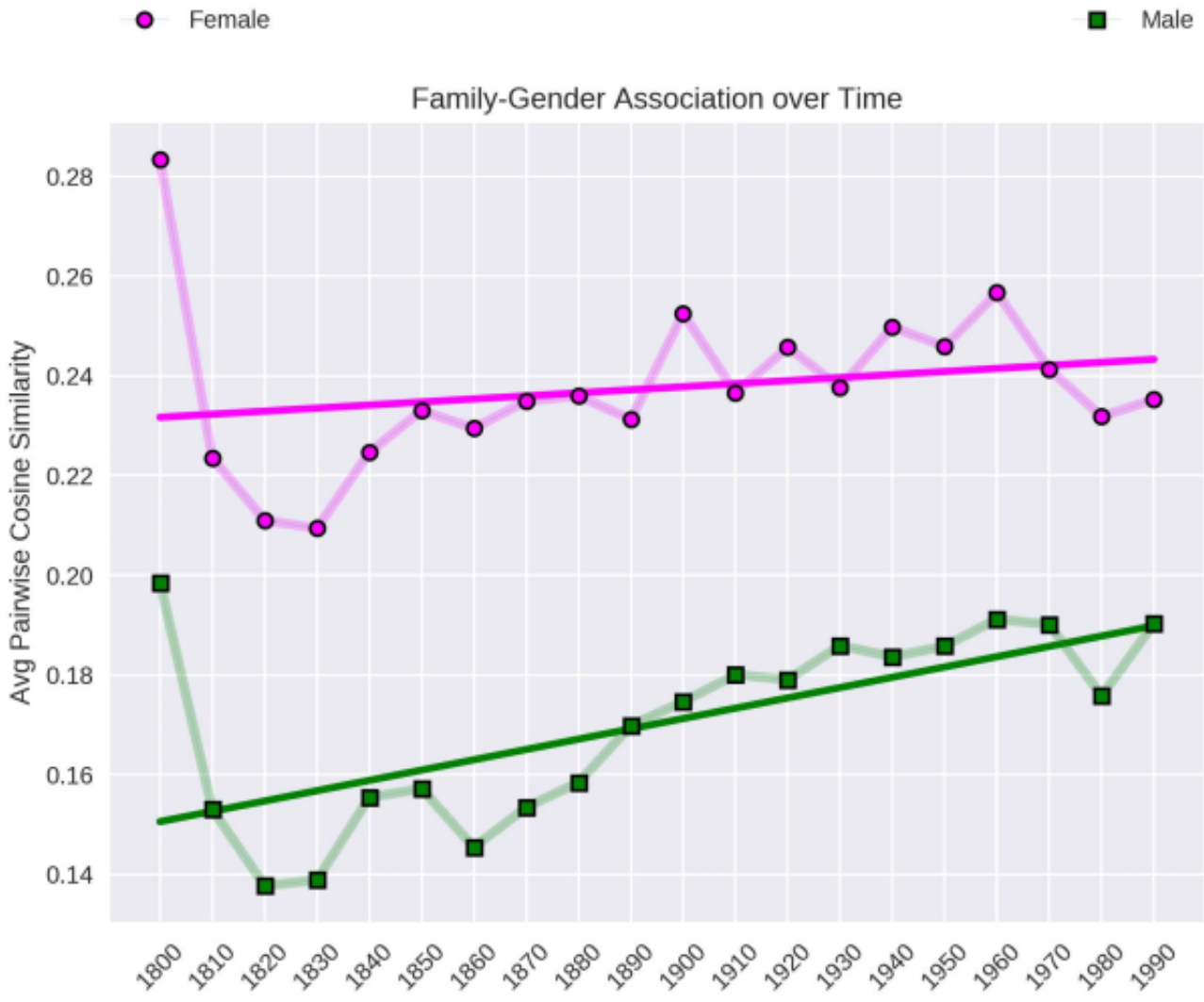
Female	Male	Family	Career	Science	Arts
female	male	home	executive	science	poetry
woman	man	parents	management	technology	art
girl	boy	children	professional	physics	Shakespeare
sister	brother	family	corporation	chemistry	dance
she	he	cousins	salary	Einstein	literature
her	him	marriage	office	NASA	novel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

4/ Working with embeddings

Result: gender bias has decreased over time



4/ Working with embeddings



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
- Dustin S. Stoltz and Marshall A. Taylor, *Mapping Texts: Computational Text Analysis for the Social Sciences* (New York, 2024; online edn, Oxford Academic, 14 Dec. 2023), https://doi.org/10.1093/oso/9780197756874.001.0001
- Yoav Goldberg, 2016, "A primer on neural network models for natural language processing", *The Journal of Artificial Intelligence Research*, 57, 345–420.