Word embedding

intuition, concepts, and methods

Intuition

how to learn a word’s meaning

- guessing an unseen word’s meaning is hard but context helps
- guessing blanked-out words let’s you discover how word embedding methods work

How to learn a word’s meaning

Example

What does the word **tezgüino** mean?
(please don’t google it 😅)

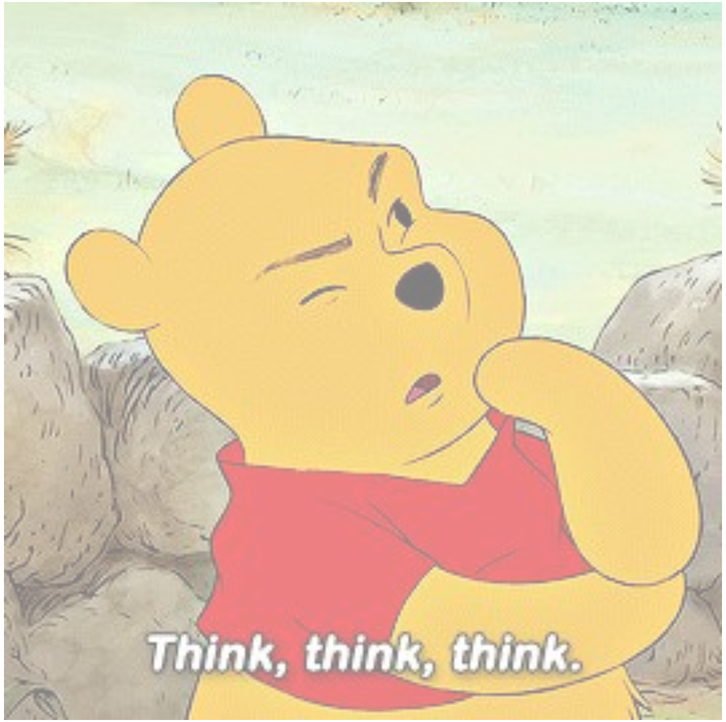
How to learn a word’s meaning

Example

What does the word **tezgüino** mean?

Examples how it’s used in a sentences:

1. A bottle of **tezgüino** is on the table.
2. Everyone likes **tezgüino**.
3. **Tezgüino** makes you drunk.
4. We make **tezgüino** out of corn.



How to learn a word’s meaning

Example

How did you guess?!?!

A ***fill-the-blank*** approach (masked LM)

1. A bottle of _______ is on the table.
2. Everyone likes _______.
3. _______ makes you drunk.
4. We make _______ out of corn.

Exercise

- take a list of words
- check for each in which of the sentences on the left it’d make sense

Hint vary the words in your list in how similar they are to your current guess of what “tezgüino” means

Source: Lin (1994) (from Lena Voita's NLP course)

How to learn a word’s meaning

Example

What did you guess?!?!

A *fill-the-blank* approach (masked LM)

1. A bottle of _______ is on the table.
2. Everyone likes _______.
3. _______ makes you drunk.
4. We make _______ out of corn.

	Sentence			
	(1.)	(2.)	(3.)	(4.)
tezgüino	✔	✔	✔	✔
loud				
motor oil	✔			✔
tortillias		✔		✔
wine	✔	✔	✔	

Inference The blanked-out word is similar to wine, but its made from corn!

Source: Lin (1994) (from Lena Voita's NLP course)

How to learn a word's meaning

Nice!!!

You just discovered the **distributional hypothesis**:

> *Words which frequently appear in similar contexts have similar meaning.*

> "You shall know a word by the company it keeps" (Firth 1957)

How to learn a word’s meaning

Nice!!!

You just discovered the **distributional hypothesis**:

> *Words which frequently appear in similar contexts have similar meaning.*

> “You shall know a word by the company it keeps” (Firth 1957)

Implication

- Words with **similar meaning** appear in **similar context** (word windows or sentences)
- To **capture a word’s meaning** with numbers, its *numeric representation* should summarize in which word contexts it occurs

Word embedding

properties, characteristics, and methods

- word embedding methods learn word vectors
- word vectors “summarize” in which contexts words occur

Word embeddings

Properties

- summarize in which contexts words occur (“distributional patterns”)
- they thus capture word **meaning** *and* **function**

Figure 6.1 A two-dimensional (t-SNE) projection of embeddings for some words and phrases, showing that words with similar meanings are nearby in space. The original 60-dimensional embeddings were trained for sentiment analysis. Simplified from Li et al. (2015) with colors added for explanation.

Word embeddings

Properties

- summarize in which contexts words occur (“distributional patterns”)
- they thus capture word **meaning** *and* **function**
- are ***geometrically*** **meaningful**
  - similar words are close by
  - captures semantic, syntactic, and conceptual relationships

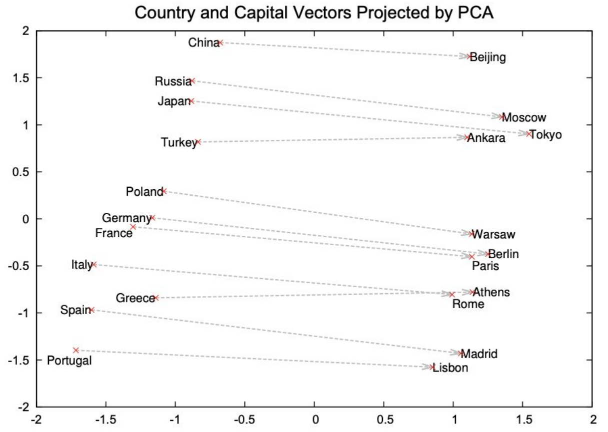


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised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

Word embeddings

Characteristics

- **fixed-length** & **low-dimensional**
- **real-valued** ("dense") ⇒ word vectors have no zero entries
- **distributed**, i.e. information about words semantic properties and syntactic functions distributed across dimensions

	d001	*d002*	…	*d299*	*d300*
word1	-0.833	0.036	…	-1.027	0.761
word2	0.180	-0.667	…	2.515	-2.165
...					
***V*'s word**	-1.595	0.350	…	-0.759	0.981

Word embeddings

Added value for CSS

Word embeddings are valuable for *computational social science* research

- for quantifying **cultural biases** and connotations
- for scoring document on **conceptual dimensions**
- for keyword expansion and dictionary construction
- …

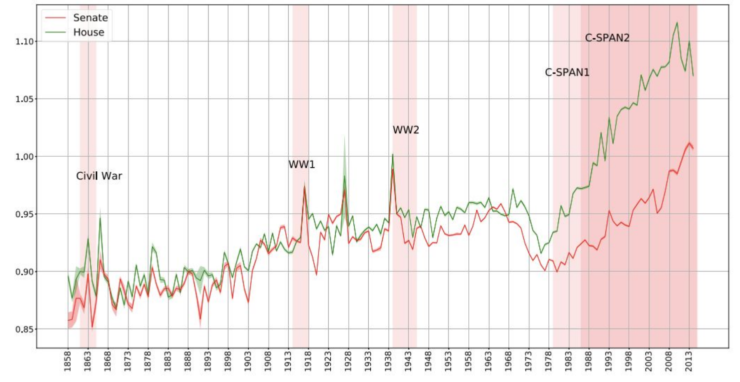


Fig. 2. *Emotionality in U.S. Congress by Chamber, 1858–2014.*
Notes: Time series of emotionality in the Senate (red) and the House of Representatives (green).

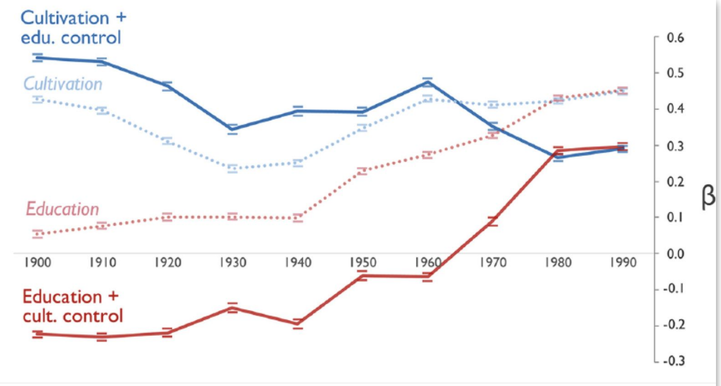
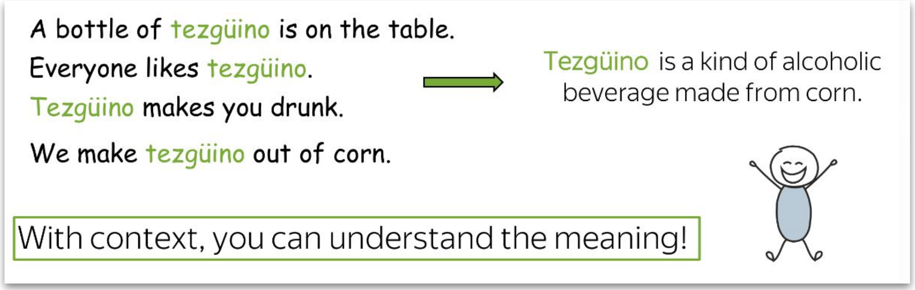


Figure 6. Standardized Coefficients from OLS Regression Models in Which Word Projections on Cultivation and Education Dimensions Predict Projection on the Affluence Dimension; 1900 to 1999 Google Ngrams Corpus

Word embedding methods

Intuition

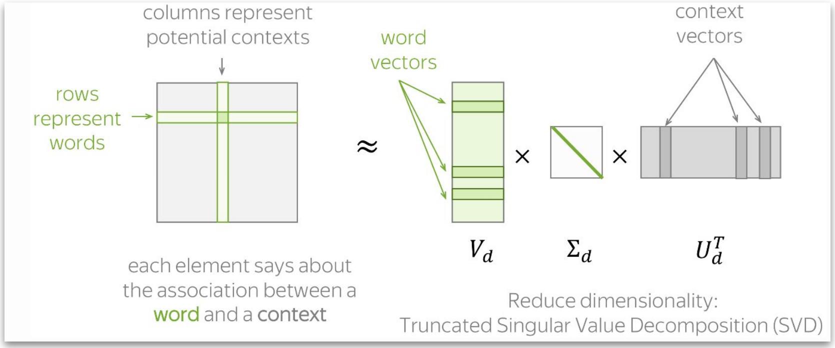
- co-occurrence patterns and/or word context information summarizes a word's meaning and functions
- embedding methods condense these distributional patterns into low-dimensional vectors



Word embedding methods

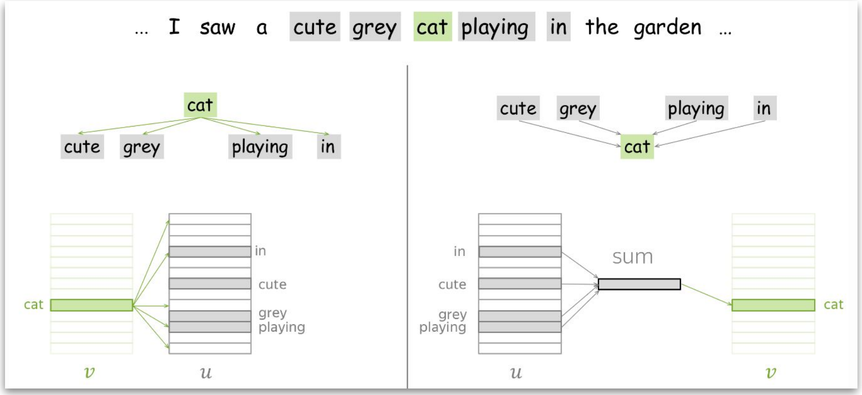
Count-based methods

learn word vectors by reducing the dimensionality of word context representations



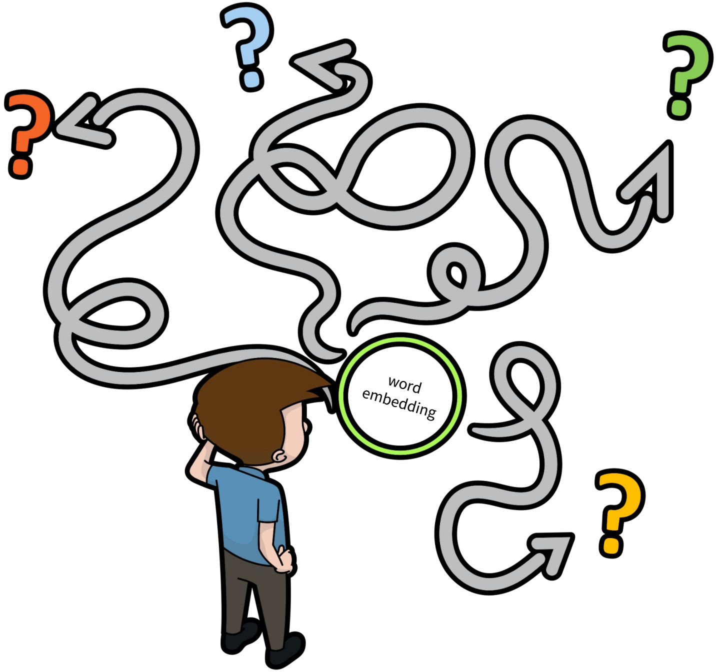
Prediction-based methods

learn good word vectors by predicting their context (or *vice versa*)



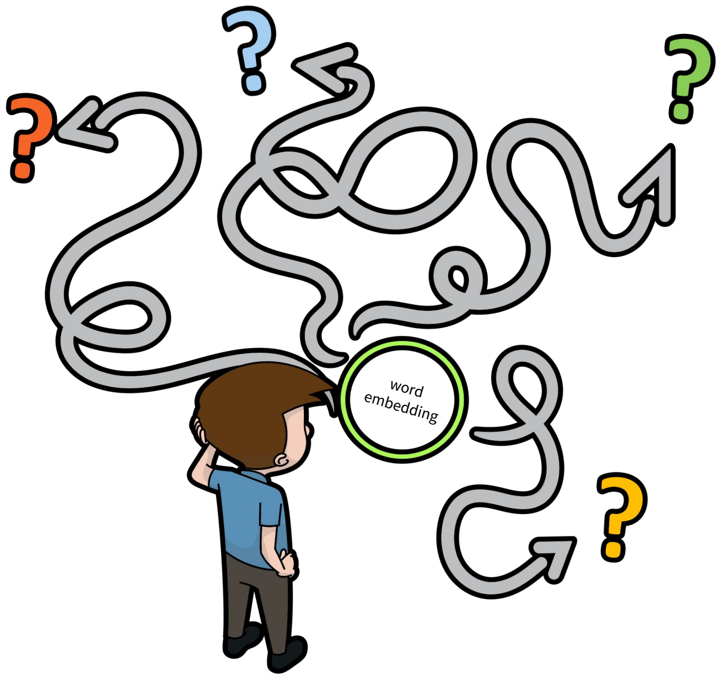
Word embeddings – *some terminology*

- ***a* word embedding** (singular, noun): the *vector* representing a word in a *d*-dimensional, real-valued space
- **the embedding space**: the *d*-dimensional space discovered/induced while training an embedding model on text data



Word embedding methods – *some terminology*

- **word embedding model**: a model train with a specific algorithm (the learned word embeddings are model parameters)
- **word embedding**: can refer to
  a. an individual word embedding, and
  b. the process of "embedding" words in a corpus through machine learning



Advantages

properties and c

- word embedding methods learn word vectors
- word vectors “summarize” in which contexts words occur

Why word embedding (Why should we care?)

Limitations of bag-of-word (BoW) representations

only possible word (~~document~~) representation is the "*one-hot* encoding"

- each vector contains zeros with only one entry set to 1
  ⇒ extremely **sparse**
  ⇒ no info about **words' relations**
- length of vectors = vocabulary size
  ⇒ "curse of dimensionality"

	d00001	*d00002*	…	*d29999*	*d30000*
word1	1	0	…	0	0
word2	0	1	…	0	0
...					
***V*'s word**	0	0	…	0	1

Why word embedding (Why should we care?)

Limitations of bag-of-word (BoW) representations

representing *documents* as bag-of-words (i.e., in a document-term matrix)

- discards word order ⇒ no information about **meaning**
- length of vectors = vocabulary size ⇒ “curse of dimensionality”

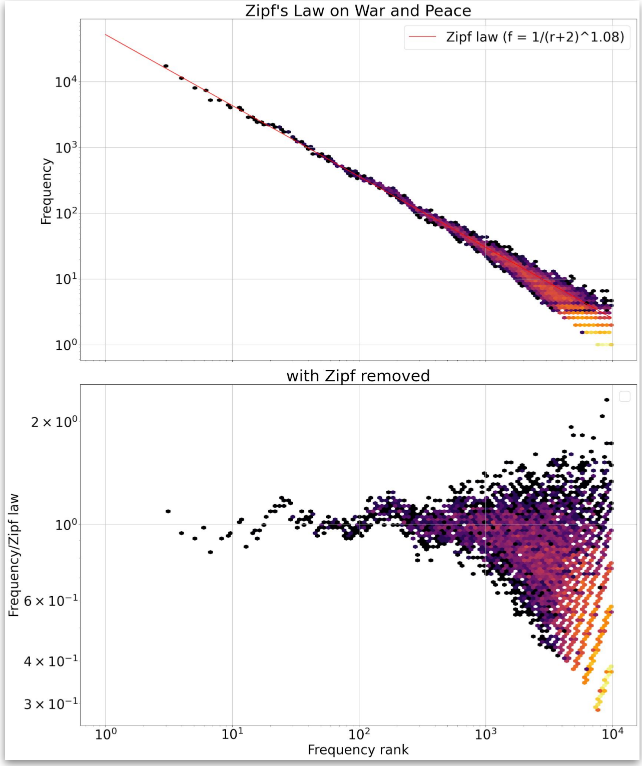
	d00001	*d00002*	…	*d29999*	*d30000*
doc1	1	1	…	0	0
doc2	0	1	…	1	0
...					
***N*'s doc**	1	0	…	0	1

Why word embedding (Why should we care?)

Curse of dimensionality

Zipf's Law: most documents only contain a small subset of the entire vocabulary

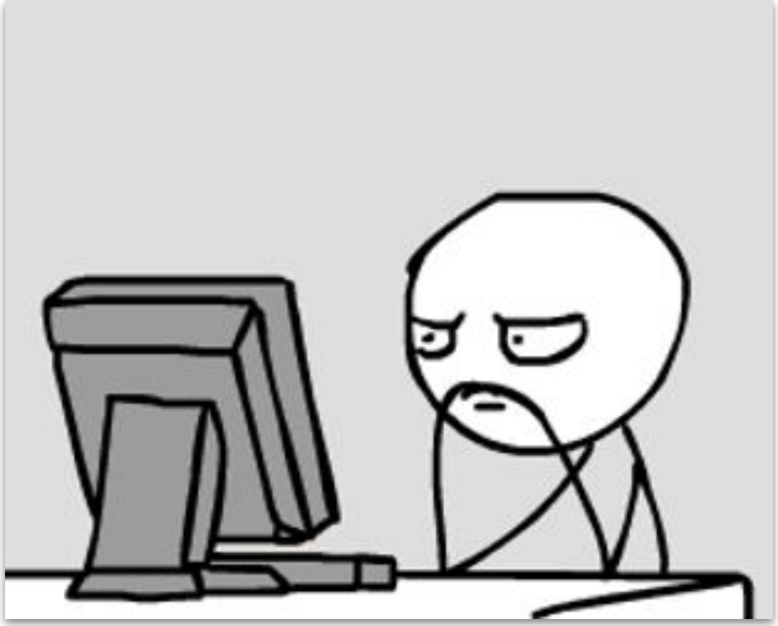
- leads to extremely high sparsity because
  - most entries in BoW vectors are 0, and
  - dimensionality of BoW vectors increases with corpus size



Why word embedding (Why should we care?)

High-*d* and sparsity are bad for quant. analysis and machine learning

- sparsity in high-D spaces can cause overfitting ⇒ limits generalization
- leads to high computational complexity
  - higher d ⇒ more RAM needed
  - higher d ⇒ higher computing time



Why word embedding (Why should we care?)

Characteristics (recap)

- **fixed-length** & **low-dimensional**
- **real-valued** ("dense") ⇒ word vectors have no zero entries
- **distributed**: information about words semantic properties and syntactic functions distributed across dimensions

	d001	*d002*	…	*d299*	*d300*
word1	-0.833	0.036	…	-1.027	0.761
word2	0.180	-0.667	…	2.515	-2.165
...					
***V*'s word**	-1.595	0.350	…	-0.759	0.981

Code

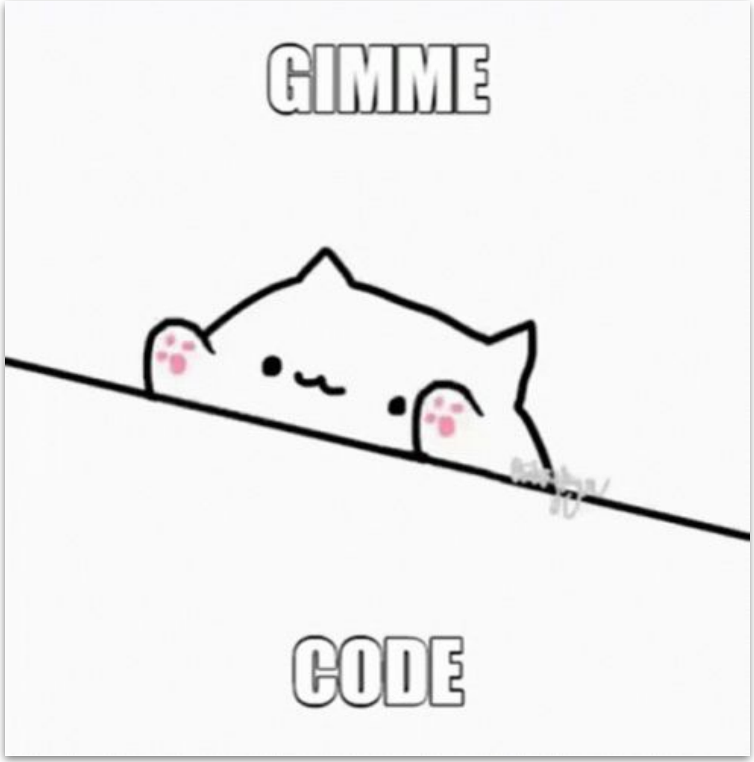
how to compute with word embeddings

- load pre-trained model from `gensim`'s models API
- how to compute similarities, find nearest neighbors, and solve analogies

Computing with word embeddings in `gensim`

We have prepared two jupyter notebooks

- one with illustrations, examples, and explanations
- another one with in-class exercises



Summary

- because similar words tend to occur in similar textual contexts (i.e., co-occur), word embeddings “store” a lot of information about words’ similarities
    - in terms of words’ semantic relations (synonyms, opposites, etc.)
    - in terms of words’ (grammatical) functions
-

Relevance

Things we can do with word vectors

- Computating
    - similarities
    - analogies
- Measurement
    - induce conceptual dimensions (e.g., emotional–rational)
    - analyze lower dimensionality
- Other uses
    - use as knowledge base, e.g., for dictionary expansion
    - use as input feature representations in deep/machine learning models/algorithms
    -

Exercise

- hands-on
  - loading gensim
  - loading pre-trained models from the API
  - computing
    - similarities
    - todo: illustrate cosine similarity, and explain difference to dot product
    - exercise: re-implement odd-one (`doesnt_match`) computation out to build intuition for computing with embeddings
    - nearest neighbors
    - show dimensionality
      - based on capital–country pairs (with 2d PCA) see https://chat.openai.com/share/8ffcfab5-a18b-4485-b30c-38c491b6aad3
    - computing analogies
      - note how this is just simple arithmetic combined with nearest neighbor search

word2vec

learning word vectors by predicting words in context

- motivation: simple idea, clever algorithm, illustrates lots of key ideas in deep learning-based NLP
- building blocks
    - target/focal/focus words and context/neighboring words
    - self-supervised learning
    - skipgram and CBOW
        - CBOW: predict target word from its context.
            - goal: max. probability of the target word given its context
            - implementation: sum/average context words' embeddings into 1d vector; use it to predict target word (as in nominal regression).
        - skip-gram: predict context words given target word.
            - goal: max. probability of context words given target word
            - take one context word at a time; an predict it given the target word's vector
    - Pr(y | **x**, *ß*)
    - predicting target/context word as classification task (with large label space)
        - costly
        - like a nominal/categorical regression, but need non-linearities for learning "good" word embeddings
    - softmax and probability distribution over the vocabulary
    -
    - negative sampling
    - *stochatstic gradient decent* and back propagation (show explainer video)

Social Science Applications

- for measurement purposes
-
- features in downstream tasks

Different ways of using word embeddings in CSS research

1. as primary quantity of interest
   a. to compute associations (Kozlowski, WEAT)
   b.
2. as a tool
   a. scale documents (e.g., Gennaro and Ash, 2022)
   b. to compare language use (Rodriguez, Spirling, and Stewart)