Einführung in die Computerlinguistik

Distributional Semantics

Laura Kallmeyer

Heinrich-Heine-Universität Düsseldorf

Summer 2021

Introduction

- Vector semantics (= distributional semantics): We characterize words by the words that occur with them, i.e., by their distribution. The distributional properties tell a lot about the semantics of a word, therefore distributional *semantics*.
- The distribution of a word is captured in a vector, i.e., we compute an embedding of the word into a vector space, therefore the term *word embedding*, often used instead of *distributional vector*.

Jurafsky & Martin (2019), chapters 15, 16

Table of contents

Motivation

- Underlying idea: words with a similar meaning tend to occur in similar contexts.
- First formulated by Harris (1954), pointing out that "oculist and eye-doctor . . . occur in almost the same environment".
- Most famous formulation of this idea goes back to Firth (1957): "You shall know a word by the company it keeps".

Example from Nida (1975); Lin (1998); Jurafsky & Martin (2019)

(1) a. A bottle of *tesgüino* is on the table.
b. Everybody likes *tesgüino*.
c. *Tesgüino* makes you drunk.
d. We make *tesgüino* out of corn.

⇒ "The meaning of a word is thus related to the distribution of words around it." Jurafsky & Martin (2019)

Word vectors

Word-word matrix: Let V be our vocabulary. Then we use a $|V| \times |V|$ matrix where each row represents the distributional vector of a word.

The row i gives a vector of dimension $|V|$ that represents word v_i.

The cell i, j gives the frequency of v_j in the contexts of v_i. The context is generally a window around the word, i.e., k words to the left and k words to the right, for instance $k = 4$.

Word vectors

Example from Jurafsky & Martin (2019), chapter 19

Vectors for four words from the Brown corpus, showing only five of the dimensions:

	…	computer	data	pinch	result	sugar	…
apricot	…	0	0	1	0	1	…
pineapple	…	0	0	1	0	1	…
digital	…	2	1	0	1	0	…
information	…	1	6	0	4	0	…

The dimensions represent context words.
We usually consider only the n most frequent words as dimensions of our vectors with $10.000 \leq n \leq 50.000$.

The vectors are very sparse (i.e., contain a lot of zeros).

Word vectors

Syntactic dependencies connecting context words to the words we want to characterize play a role for the meaning.

(2) a. Hans' Ball rollt als erster ins Ziel.
    b. Hans rollt seinen Ball als erster ins Ziel.

Simple context word vectors cannot account for the difference between the two readings of *rollen*.

(3) a. Hans isst Kuchen.
    b. Kuchen isst Hans.

If the context window size is 1, we get

	essen
Hans	2
Kuchen	2

I.e., *Hans* and *Kuchen* have the same vector.

Word vectors

- Instead of using just words as context elements, one can also use words combined with syntactic information.
- Assume that we have a corpus with syntactic dependencies.
- Then, instead of context words $c_i \in V$, we use context elements $\langle dep, c_i \rangle$ as dimensions.

	subj-of, essen	*obj-of*, essen
Hans	2	0
Kuchen	0	2

I.e., *Hans* and *Kuchen* have cos similarity 0.

Pointwise mutual information

The raw frequency counts are not the best measures for associations between words. One common association measure used in stead is **pointwise mutual information (PMI)**. The PMI of two events x and y measures how often x and y occur together compared to what we would expect if they were independent:

$$PMI(x, y) = \log_2 \frac{P(x, y)}{P(x)P(y)}$$

Recall that $P(x, y) = P(x)P(y|x)$ and that for independent events we have $P(y|x) = P(y)$. I.e., for independent events x, y, we obtain $PMI(x, y) = \log_2 1 = 0$.

Pointwise mutual information

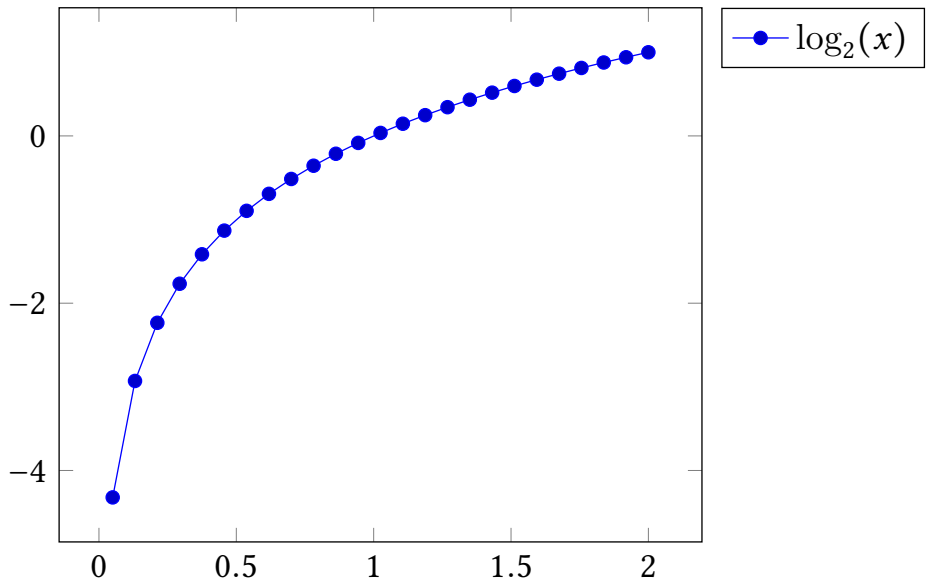
For our specific case of vector semantics, we measure the association between a target word w and a context word c as

$$PMI(w, c) = \log_2 \frac{P(w, c)}{P(w)P(c)}$$

PMI gives us an estimate of how much more the word w and context word c co-occur than we would expect by chance.

Pointwise mutual information

Reminder:



In particular, $\log_2(1) = 0$ (events are completely independent, therefore there is no need to consider the value in the vector), and $\log_2(0)$ is not defined ($-\infty$), i.e., PMI has a problem for pairs w, c that never occur together.

Pointwise mutual information

Negative PMI values (w and c occur together less often than by chance) tend to be unreliable. Therefore, one usually uses **positive PMI (PPMI)**:

$$PPMI(w, c) = \max(\log_2 \frac{P(w, c)}{P(w)P(c)}, 0)$$

We can estimate these probabilities using the frequencies: Let $W = \{w_1, \ldots, w_{|W|}\}$ be our set of words, $C = \{c_1, \ldots, c_{|C|}\}$ our set of context words, f_{ij} the frequency of c_j in the context of w_i. Then

- $P(w_i, c_j) = \frac{f_{ij}}{\sum_{n=1}^{|W|} \sum_{m=1}^{|C|} f_{nm}}$
- $P(w_i) = \frac{\sum_{m=1}^{|C|} f_{im}}{\sum_{n=1}^{|W|} \sum_{m=1}^{|C|} f_{nm}}$
- $P(c_j) = \frac{\sum_{n=1}^{|W|} f_{nj}}{\sum_{n=1}^{|W|} \sum_{m=1}^{|C|} f_{nm}}$

Pointwise mutual information

Example from Jurafsky & Martin (2019) continued

Counts replaced with joint probabilities:

	computer	data	pinch	result	sugar	$p(w)$
apricot	0	0	0.05	0	0.05	0.11
pineapple	0	0	0.05	0	0.05	0.11
digital	0.11	0.05	0	0.05	0	0.21
information	0.05	0.32	0	0.21	0	0.58
$p(c)$	0.16	0.37	0.11	0.26	0.11	

PPMI matrix:

	computer	data	pinch	result	sugar
apricot	0	0	2.25	0	2.25
pineapple	0	0	2.25	0	2.25
digital	1.66	0	0	0	0
information	0	0.57	0	0.47	0

Pointwise mutual information

(P)PMI has a bias towards infrequent events. Therefore one sometimes replaces the above estimate $P(c_j)$ with

$$P_\alpha(c_j) = \frac{(\sum_{n=1}^{|W|} f_{nj})^\alpha}{\sum_{m=1}^{|C|}(\sum_{n=1}^{|W|} f_{nm})^\alpha}$$

for example with $\alpha = 0.75$ (Levy et al., 2015).

To avoid the 0 entries, one can also apply Laplace smoothing before computing PMI: add a constant k to all counts (usually $0.1 \leq k \leq 3$).

Another association measure sometimes used in vector semantics:

$$t - test(w, c) = \frac{P(w, c) - P(w)P(c)}{\sqrt{P(w)P(c)}}$$

Measuring vector distance/similarity

In order to compare word embeddings, we need some metric for the distance of two vectors. One possibility is the **Euclidian distance**.

The length of a vector is defined as

$$|\vec{v}| = \sqrt{\sum_{i=1}^{n} v_i^2}$$

We define the Euclidian distance of two vectors $\vec{v}$, $\vec{w}$ as

$$|\vec{v} - \vec{w}| = \sqrt{\sum_{i=1}^{n} (v_i - w_i)^2}$$

Measuring vector distance/similarity

Using the Euclidian distancs as such does not make sense yet since words that occur in exactly the same contexts but with different frequencies would be considered different.

We therefore first normalize the vectors, i.e., we use

$$|\frac{\vec{v}}{|\vec{v}|} - \frac{\vec{w}}{|\vec{w}|}| = \sqrt{\sum_{i=1}^{n}(\frac{v_i}{|\vec{v}|} - \frac{w_i}{|\vec{w}|})^2}$$

Alternatively, we can also use a metric for the similarity of vectors.

The most common metric used in NLP for vector similarity is the **cosine** of the angle between the vectors.

Measuring vector distance/similarity

Underlying idea: We use the dot product from linear algebra as a similarity metric: For vectors $\vec{v}$, $\vec{w}$ of dimension n,

$$\vec{v} \cdot \vec{w} = \sum_{i=1}^{n} v_i w_i$$

Example: dot product

Consider the following word-word matrix:

	c_1	c_2	c_3
w_1	1	2	30
w_2	2	8	120
w_3	8	3	0
w_4	50	20	2

- w_1, w_2: $\vec{v}_{w_1} \cdot \vec{v}_{w_2} = \langle 1, 2, 30 \rangle \cdot \langle 2, 8, 120 \rangle = 3618$
- w_1, w_3: $\vec{v}_{w_1} \cdot \vec{v}_{w_4} = 200$
- But: $\vec{v}_{w_2} \cdot \vec{v}_{w_4} = 500$ and $\vec{v}_{w_3} \cdot \vec{v}_{w_4} = 460$

Measuring vector distance/similarity

The dot product favors long vectors. Therefore, we use the **normalized dot product**, i.e., we divide by the lengths of the two vectors.

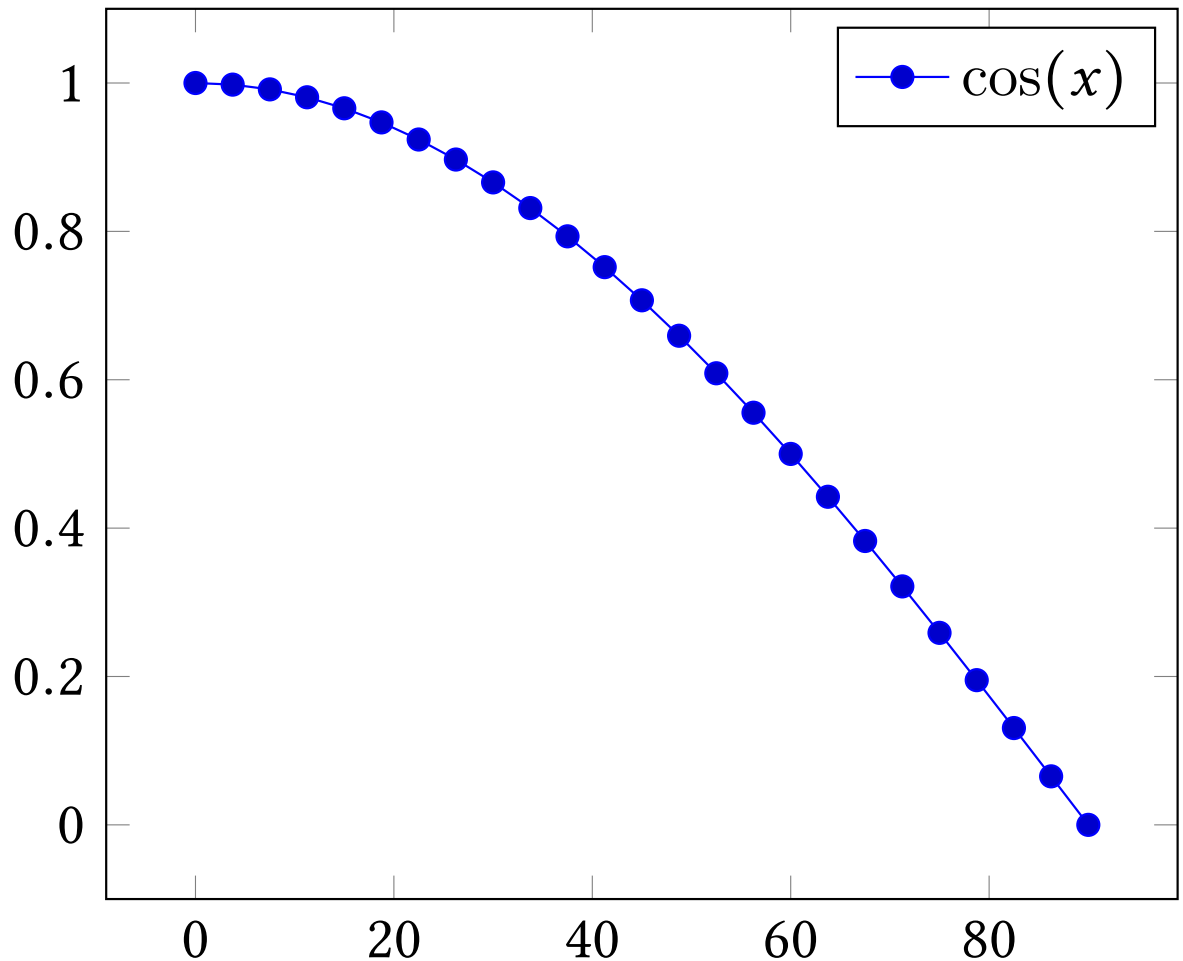
Cosine similarity metric

$$CosSim(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}|\,|\vec{w}|} = \frac{\sum_{i=1}^{n} v_i w_i}{\sqrt{\sum_{i=1}^{n} v_i^2}\sqrt{\sum_{i=1}^{n} w_i^2}} = \cos\phi$$

where ϕ is the angle between $\vec{v}$ and $\vec{w}$.

Measuring vector distance/similarity

Reminder: This is how the cosine looks like for angles between 0 and 90 (with all vector components ≥ 0, other angles cannot occur):



Measuring vector distance/similarity

Example: cosine similarity

Consider again the following term-document matrix:

	c_1	c_2	c_3	$\lvert\vec{v}_w\rvert$
w_1	1	2	30	30,08
w_2	2	8	120	120,28
w_3	8	3	0	8,54
w_4	50	20	2	53,89

Cosine values:

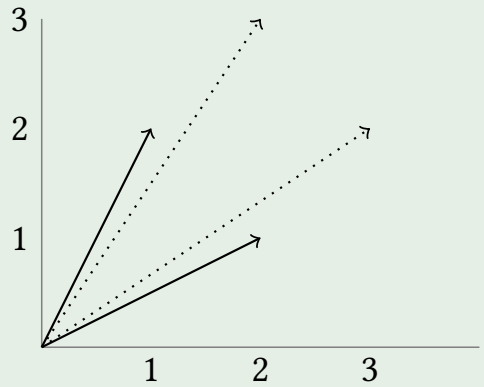
	w_1	w_2	w_3
w_2	1		
w_3	0.05	0.04	
w_4	0.09	0.08	1

(The cosine values for w_1, w_2 and for w_3, w_4 are rounded.)

Measuring vector distance/similarity

Cosine similarity is not invariant to shifts.

Cosine



Adding 1 to all components increases the cosine similarity.

The Euclidian distance, however, remains the same (if used without normalization).

An alternative is the **Pearson correlation**:

Pearson correlation

Let $\overline{v} = \frac{\sum_{i=1}^{n} v_i}{n}$ be the means of the vector components.

$$Corr(\vec{v}, \vec{w}) = \frac{\sum_{i=1}^{n} (v_i - \overline{v})(w_i - \overline{w})}{|\vec{v} - \overline{v}|\,|\vec{w} - \overline{w}|} = CosSim(\vec{v} - \overline{v}, \vec{w} - \overline{w})$$

where $\langle v_1, \ldots, v_n \rangle - c = \langle v_1 - c, \ldots, v_n - c \rangle$.

Measuring vector distance/similarity

Two other widely used similarity measures are **Jaccard** and **Dice**. The underlying idea is that we measure the overlap in the features. Therefore in both metrics, we include

$$\sum_{i=1}^{n} \min(v_i, w_i)$$

Jaccard similarity measure

$$simJaccard(\vec{v}, \vec{w}) = \frac{\sum_{i=1}^{n} \min(v_i, w_i)}{\sum_{i=1}^{n} \max(v_i, w_i)}$$

Dice is very similar except that it does not take the max but the average in the denominator:

Dice similarity measure

$$simDice(\vec{v}, \vec{w}) = \frac{2 \sum_{i=1}^{n} \min(v_i, w_i)}{\sum_{i=1}^{n} v_i + w_i}$$

From sparse vectors to dense embeddings

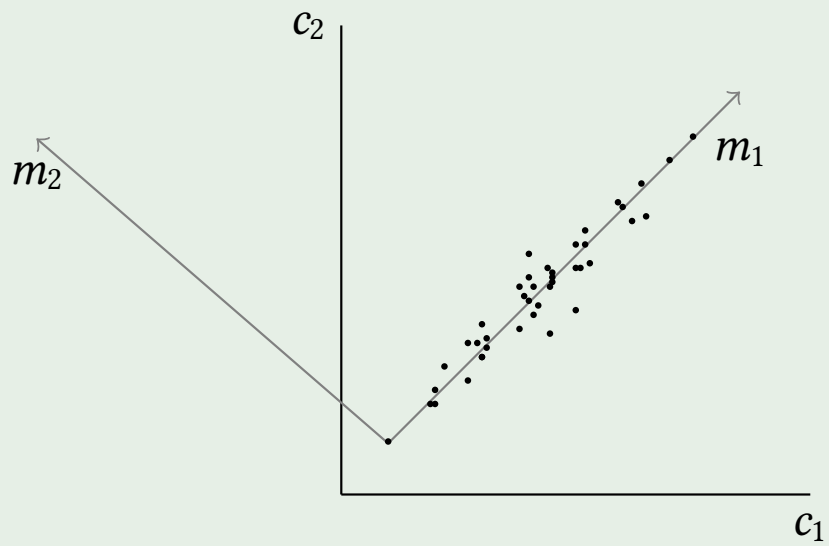
So far, our vectors are high-dimensional and sparse. **Singular value decomposition (SVD)** is a classic method for generating dense vectors.

Idea:

- Change the dimensions such that they are still orthogonal to each other.
- The new dimensions are such that the first describes the largest amount of variance in the data, the second the second large variance amount etc.
- Then, instead of keeping all the m dimensions resulting from this, we only keep the first k.

From sparse vectors to dense embeddings

Example with 2 dimensions



The original dimensions c_1 and c_2 get replaced with m_1 and m_2. Then we could truncate and keep only the dimension m_1.

After truncation, we obtain context vectors of dimension k for each word. These are dense **embeddings**.

From sparse vectors to dense embeddings

Assume that we have w words and c context words. Then, in gneral, SVD decomposes the $w \times c$ word-context matrix X into a product of three matrices W, Σ, C:

$$\underset{w \times c}{\begin{bmatrix} \\ X \\ \\ \end{bmatrix}} = \underset{w \times m}{\begin{bmatrix} \\ W \\ \\ \end{bmatrix}} \underset{m \times m}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ & & \dots & & \\ 0 & 0 & 0 & \dots & \sigma_m \end{bmatrix}} \underset{m \times c}{\begin{bmatrix} \\ C \\ \\ \end{bmatrix}}$$

Each row in X is a PPMI context word vector of a word. Each row in W is a word embedding of a word in a new m-dimensional vector space.

From sparse vectors to dense embeddings

Example

$$\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} \sqrt{5} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ -\frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{bmatrix}$$

- matrix W: first dimension (i.e., vector $\langle 1, 0 \rangle$) corresponds to $\langle 1, 2 \rangle$ in original matrix X;
- matrix Σ: multiply length 1 with the length of $\langle 1, 2 \rangle$;
- matrix C: rotation from x-axis to the axis along $\langle 1, 2 \rangle$;

Last step: truncation.The second dimension in W can be left out, which leads to $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ instead of the original $\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$, giving 1-dimensional embedding vectors for each word.

Neural word embeddings

- Two popular methods for generating dense embeddings are **skip-gram** and **continuous bag of words (CBOW)**.
- Both learn embeddings from pairs of context words and target word via a neural network.
- Both of them are implemented in the **word2vec** package Mikolov et al. (2013).
- skip-gram learns embeddings by predicting (single) context words from a given target word while CBOW learns embedding by predicting a target word from a set of context words.
- In both cases, the activations of a hidden layer are used as the word embedding.

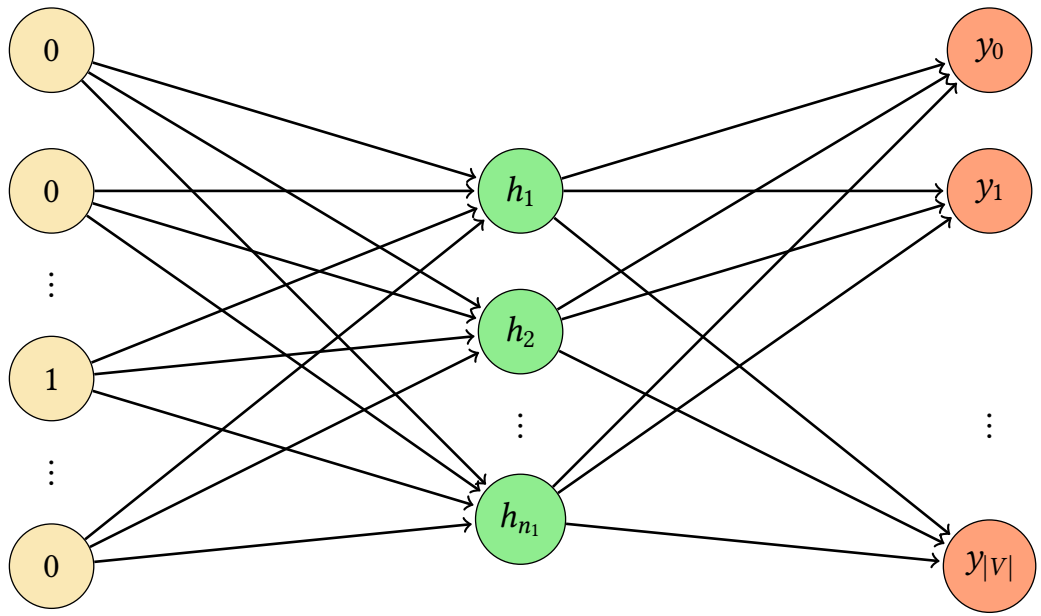
Neural word embeddings

Skip-gram:

- As training data, we use all token pairs of target word x_i and context word y_j where y_j occurs in a window of size n around x_i.
- Both input and output layers have $|V|$ components. The input is a one-hot-vector for a target word. The output layer is trained to be a distribution over context words.
- Once the network is trained, the activations of the hidden middle layer for a one-hot encoding of x_i as input give the word embedding for x_i.
- These are the weights on the outgoing edges from the ith input component.

Neural word embeddings

Skip-gram:



input layer:
one-hot
encoding of x_i

hidden layer:
embedding of x_i

output layer:
context word
distribution for x_i

Evaluating vector models

One common way to test distributional vector models is to evaluate their performance on **similarity**. Some datasets one can evaluate on:

- **WordSim-353**, a set of ratings from 0 to 10 of the similarity of 353 noun pairs
- **SimLex** includes both concrete and abstract noun and verb pairs.
- The **TOEFL dataset** is a set of 80 questions, each consisting of a target word and 4 word choices. E.g., *Levied is closest in meaning to: imposed, believed, requested, correlated*
- The **Stanford Contextual Word Similarity (SCWS)** dataset gives human judgements on 2,003 pairs of words in their sentential context.

References

Firth, J. R. 1957. A synopsis of linguistic theory 1930–1955. In *Studies in linguistic analysis*, Philological Society. Reprinted in Palmer, F. (ed.) 1968. *Selected Papers of J. R. Firth*. Longman, Harlow.

Goldberg, Yoav. 2017. *Neural network methods in natural language processing*. Morgan & Claypool Publishers.

Harris, Z. S. 1954. Distributional structure. *Word* 10. 146–162. Reprinted in J. Fodor and J. Katz, *The Structure of Language*, Prentice Hall, 1964 and in Z. S. Harris, *Papers in Structural and Transformational Linguistics*, Reidel, 1970, 775–794.

Jurafsky, Daniel & James H. Martin. 2019. Speech and language processing. an introduction to natural language processing, computational linguistics, and speech recognition. Draft of the 3rd edition. Available here: `https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf`.

Levy, Omer, Yoav Goldberg & Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics* 3. 211–225. `https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/570`.

Lin, Dekang. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on computational linguistics - volume 2* COLING '98, 768–774. Stroudsburg, PA, USA: Association for Computational Linguistics. doi:10.3115/980432.980696. http://dx.doi.org/10.3115/980432.980696.

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

Nida, E. A. 1975. *Componential analysis of meaning: An introduction to semantic structures*. The Hague: Mouton.