Machine Learning for natural language processing

Distributional Semantics

Laura Kallmeyer

Heinrich-Heine-Universität Düsseldorf

Summer 2016

Introduction

- Vector classification: characterize a document by a vector that captures its bag-of-words, i.e., that tells about the words occurring in the document and about their frequencies.
- Vector semantics (= distributional semantics) is very similar: We characterize words by the words that occur with them. This vector representation tells a lot about the semantics of the word, therefore distributional *semantics*.
- Many notions from the session on k nearest neighbors will be relevant for vector semantics.

Jurafsky & Martin (2015), 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 (2015)

(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 (2015)

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. (Note that in the term-document matrix, each column was one of our vectors, this is different now!)

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 (2015), 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

As in the kNN case, 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

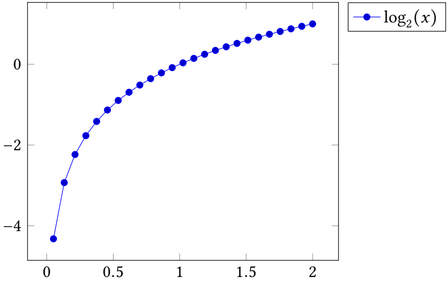
For our specific case of vector semantics, we measure the association between a target word w and a context word c as

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

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 get these probabilities by MLE 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 (2015) 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 MLE $P(c_j)$ 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:

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

From sparse vectors to dense embeddings

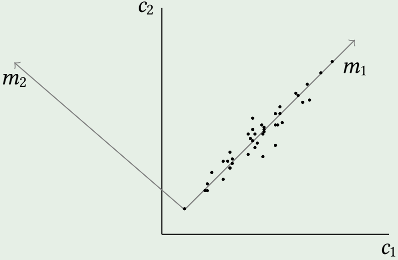
So far, our vectors are high-dimensional and sparse. **Singular value decomposition (SVD)** is a classic method for generating dense vectors.

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.

From sparse vectors to dense embeddings

- Other popular methods for generating dense embeddings are **skip-gram** and **continuous bag of words (CBOW)**.
- Both of them are implemented in the **word2vec** package Mikolov et al. (2013).

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.

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. 2015. Speech and language processing. an introduction to natural language processing, computational linguistics, and speech recognition. Draft of the 3rd edition.

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.