TECHNISCHE UNIVERSITÄT DRESDEN

SEMANTIC COMPUTING

Lecture 7: Introduction to Distributional and Distributed Semantics

Dagmar Gromann

International Center For Computational Logic

TU Dresden, 30 November 2018

Overview

- Distributional Semantics
- Distributed Semantics
  - Word Embeddings

Distributional Semantics

Distributional Semantics

Definition

- meaning of a word is the set of contexts in which it occurs
- no other information is used than the corpus-derived information about word distribution in contexts (co-occurrence information of words)
- semantic similarity can be inferred from proximity in contexts
- At the very core: Distributional Hypothesis

Distributional Hypothesis

“similarity of meaning correlates with similarity of distribution”

- Harris Z. S. (1954) “Distributional structure". Word, Vol. 10, No. 2-3, pp. 146-162

meaning = use = distribution in context => semantic distance

TECHNISCHE UNIVERSITÄT DRESDEN

Remember Lecture 1?

Types of Word Meaning

- **Encyclopaedic meaning**: words provide access to a large inventory of structured knowledge (world knowledge)
- **Denotational meaning**: reference of a word to object/concept or its “dictionary definition” (signifier <-> signified)
- **Connotative meaning**: word meaning is understood by its cultural or emotional association (positive, negative, neutral conntation; e.g. “She’s a dragon” in Chinese and English)
- **Conceptual meaning**: word meaning is associated with the mental concepts it gives access to (e.g. prototype theory)
- **Distributional meaning**: “You shall know a word by the company it keeps” (J.R. Firth 1957: 11)

John Rupert Firth (1957). "A synopsis of linguistic theory 1930-1955." In Special Volume of the Philological Society. Oxford: Oxford University Press.

Distributional Hypothesis in Practice

Study by McDonald and Ramscar (2001):

- The man poured from a balack into a handleless cup.
- One dispenses tea from a portable balack.
- ... it became a round-headed monster, something between a bat and a balack
- The common balack is a chubby marsupial... I once saw a sloth cuddle a balack

McDonald, S., & Ramscar, M. (2001, January). Testing the Distributional Hypothesis: The influence of Context on Judgments of Semantic Similarity. In Proceedings of the Annual Meeting of the Cognitive Science Society (Vol. 23, No. 23).

Distributional Semantic Models (DSMs)

- words are represented as real-valued vectors built from their distribution in contexts
- similarity between words is approximated in terms of their geometric distance between vectors
- build as general-purpose semantic models that can be applied to various tasks (resulting vectors are not task specific)
- rely on a co-occurrence matrix

Some Examples of DSMs

- Hyperspace Analogue to Language (Lund & Burgess 1996)
- Latent Semantic Analysis (LSA) (Landauer & Dumais 1997)
- Syntax-based DSMs using dependency relation (e.g. Baroni & Lenci 2010)

TECHNISCHE UNIVERSITÄT DRESDEN

Co-Occurrence Matrix

- Each row in the matrix M represents a word w_i
- Each column in the matrix M represents a context c_j (depending on window size: number of words in the context considered; frequently paragraphs or documents as below)
- Matrix M_{ij} represents the strength of association
- term-document matrices can get very large => dimensionality reduction
- compute similarity between two words based on row vectors

	d_1	d_2	d_3	d_4	d_5
dog	88	92	11	1	2
lion	57	28	3	0	0
bark	80	62	10	0	1
car	0	1	0	93	97
tire	2	0	2	80	72
drive	0	1	0	90	45

TECHNISCHE UNIVERSITÄT DRESDEN

Association and Similarity

Association refers to the value that is written into the matrix, whereas the similarity measure is used to calculate the similarity between two words based on the vectors (rows of the matrix).

- A good measure for association: Pointwise Mutual Information (PMI)

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

- Most common similarity measure between two vectors: cosine similarity

$$cos(\theta) = \frac{\sum_i u_i v_i}{\sqrt{\sum_i (u_i)^2}\,\sqrt{\sum_i (v_i)^2}}$$

Dimensionality Reduction

One available method is Singular Value Decomposition (SVD):

- Factorizes matrix M into three matrices: $M_{mxn} = U_{mxm}\Sigma_{mxn}V_{nxn}^{\top}$
- U and V represent orthogonal matrices and Σ is a diagonal matrix
- we want to select the k top singular values to obtain lower dimensional vectors that account for the maximum proportion of the original variance
- we want to get the best rank-d approximation of M
- computing the truncated projection: $M_{reduced} = U_{mxk}\Sigma_{kxk}V_{nxk}^{\top}$

LSA-SVD Example

Matrix M_{mxn}:

	d_1	d_2	d_3	d_4	d_5
dog	88	92	11	1	2
lion	57	28	3	0	0
bark	80	62	10	0	1
car	0	1	0	93	97
tire	2	0	2	80	72
drive	0	1	0	90	45

Matrix U_{mxm} (word-to-concept):

	1	2	3	4	5	6
dog	-0.059	0.73	0.0038	-0.58	-0.26	0.25
lion	-0.023	0.35	0.0069	0.77	-0.45	0.28
bark	-0.042	0.58	-0.0072	0.25	0.59	-0.49
car	-0.67	-0.053	-0.52	-0.034	-0.33	-0.41
tire	-0.54	-0.037	-0.13	0.063	0.51	0.65
drive	-0.49	-0.039	0.85	-0.013	-0.11	-0.16

Matrix Σ_{mxn}

197.6	0	0	0	0
0	**173.9**	0	0	0
0	0	27.78	0	0
0	0	0	20.93	0
0	0	0	0	2.744
0	0	0	0	0

Matrix $V^{\top}_{nxn}$ (document-to-concept):

	d_1	d_2	d_3	d_4	d_5
1	-0.055	-0.05	-0.011	-0.76	-0.64
2	0.75	0.65	0.085	-0.061	-0.043
3	-0.0037	0.015	-0.0097	0.64	-0.77
4	0.66	-0.75	-0.065	0.01	-0.0092
5	-0.022	-0.11	0.99	0.0036	-0.012

LSA-SVD Example

Matrix $M_{reduced}$:

	d_1	d_2	d_3	d_4	d_5
dog	95.85	83.10	10.92	1.12	2.00
lion	45.90	39.79	5.22	-0.26	0.29
bark	76.10	65.98	8.66	0.15	0.97
car	0.37	0.63	0.67	101.18	85.13
tire	1.04	1.15	0.63	81.49	68.57
drive	0.24	0.43	0.49	74.00	62.26

Matrix U_{mxk} (word-to-concept):

	1	2	3	4	5	6
dog	-0.059	0.73	0	0	0	0
lion	-0.023	0.35	0	0	0	0
bark	-0.042	0.58	0	0	0	0
car	-0.67	-0.053	0	0	0	0
tire	-0.54	-0.037	0	0	0	0
drive	-0.49	-0.039	0	0	0	0

Matrix Σ_{kxk}:

197.6	0	0	0	0
0	**173.9**	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Matrix V^T_{nxk} (document-to-concept):

	d_1	d_2	d_3	d_4	d_5
1	-0.055	-0.05	-0.011	-0.76	-0.64
2	0.75	0.65	0.085	-0.061	-0.043
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

LSA-SVD Example

- Cosine similarity between d_1 and d_3 originally $\approx$ 0.96
- Cosine similarity between d_1 and d_3 reduced $\approx$ 0.99
- Cosine similarity between lion and car originally $\approx$ 0.0033
- Cosine similarity between lion and car reduced $\approx$ 0.0054
- etc.

Distributional vs. Distributed Representation

Distributional Representation

- captures linguistic distribution of each word in form of a high-dimensional numeric vector
- typically based on co-occurrence counts (count models)
- based on distributional hypothesis: similar distribution ~ similar meaning (similar distribution = similar representation)

Distributed Representation

- sub-symbolic, compact representation of words as dense numeric vectors
- meaning is captured in different dimensions and it is used to predict words (predict models)
- similarity of vectors corresponds to similarity of the words
- word embeddings

Word Embeddings

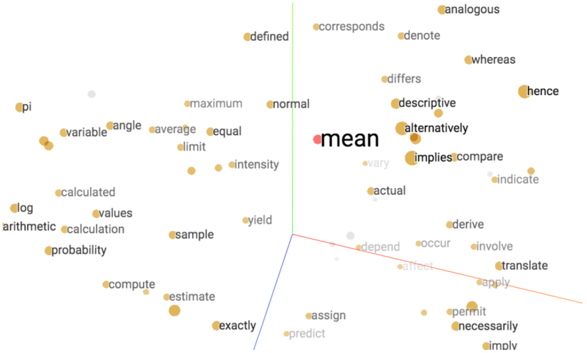
Word Embeddings

Definition

Real-valued and sub-symbolic representations of words as dense numeric vectors.

- distributed representation of word meanings (not count-based)
- usually learned with neural networks
- specific dimensions of the resulting vectors cannot bet directly mapped to symbolic representation
- models that seek to predict between a center word and context words (predict models)
- key elements of deep learning models

Visualization



$$\begin{bmatrix} -0.061 \\ -0.068 \\ 0.120 \\ 0.191 \\ -0.029 \\ 0.096 \\ 0.043 \\ \dots \end{bmatrix}$$

Visualization: `http://projector.tensorflow.org/`

Most Similar Words

Words similar to “wolf”:

- bear
- hunters
- zeus
- ferrari
- maltese

Ermanno Wolf-Ferrari (Italian composer)

Maltese dogs

Methods to Train Word Embeddings

- First and most used: word2vec (see below)
- FastText: similar to word2vec but trained on character n-grams instead of words
- GloVe: Global Vectors - first uses co-occurrence matrix, calculates ratios of probabilities; trained with log-bilinear regression model
- Many other methods...

TECHNISCHE UNIVERSITÄT DRESDEN

word2vec

Framework for learning word embeddings; main idea:

- takes words from a very large corpus of text as input (unsupervised)
- learn a vector representation for each word to predict between every word and its context
- fully connected feed-forward neural network with one hidden layer
- Two main algorithms:
  - **Continuous Bag of Words (CBOW)**: predicts center word from the given context (sum of surrounding words vectors)
  - **Skip-gram**: predicts context taking the center word as input

Seminal paper: Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

TECHNISCHE UNIVERSITÄT DRESDEN

Center word and context

In contrast to language models we have talked about, embedding models consider the history (previous words) **and** the future (following words) of a center word. The number of words considered is called the window size (standard size = 5).

Importance of Window Size

“Australian scientists discover stars with telescopes."
context window size 2 center word context window size 2

Note: different meaning of “stars" with and without telescope

Example from: Levy, O., & Goldberg, Y. (2014). Dependency-based word embeddings. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (Vol. 2, pp. 302-308).

One-Hot Encodings

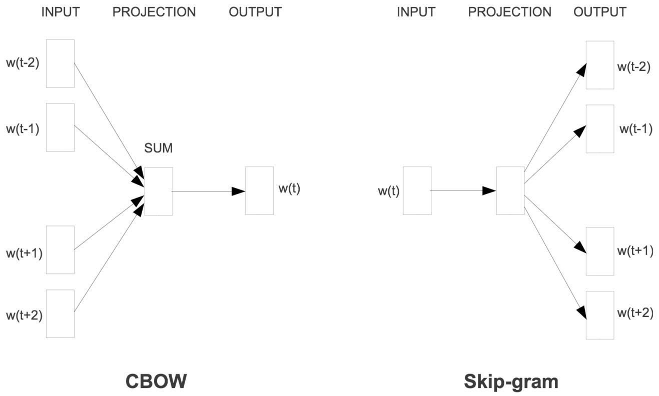
To train embeddings, words need to be converted to one-hot encodings so that they can be used as input to a neural network.

Original words
red
blue
green
blue

index	red	blue	green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

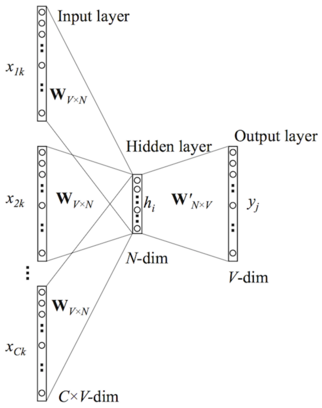
TECHNISCHE UNIVERSITÄT DRESDEN

word2vec Algorithms



Source: Mikolov et al. (2013) see previous slide

Continuous Bag of Words (CBOW)



C... window size
V... vocabulary size (e.g. 10,000 most frequent words)
$x_1, \ldots, x_C$ ... one-hot encoded input context vectors
h ... N-dimensional vector of the hidden layer
N ... predefined number of dimensions (e.g. 300)
y_j ... output vector
W_{VxN} ... weight matrix connecting input layer and hidden layer

W_{NxV} ... weight matrix connecting hidden layer and output layer

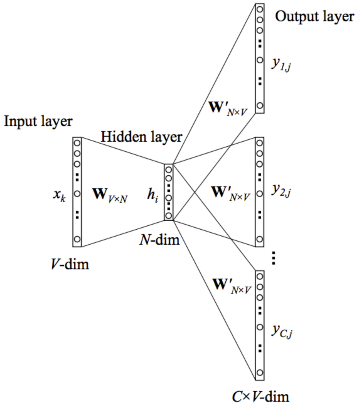
Objective function CBOW

Maximize the conditional probability of the output word given the input context (= minimize objective function; average negative log likelihood):

$$J_\theta = -\frac{1}{T}\sum_{t=1}^{T} log\ p(w_t | w_{t-n}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+n})$$

T ... training words in vocabulary V
J_θ ... objective function
θ ... all variables to be optimized
w_t ... target word to be predicted
n .... window size (number of words before and after w_t to be considered)

Skip-gram



C... window size
V... vocabulary size (e.g. 10,000 most frequent words)
x_k ... one-hot encoded input context vector
h ... N-dimensional vector of the hidden layer
N ... predefined number of dimensions (e.g. 300)
$y_1, \ldots, y_C$ ... output vectors
W_{VxN} ... weight matrix connecting input layer and hidden layer

W_{NxV} ... weight matrix connecting hidden

layer and output layer

Objective function SG

Maximize the conditional probability of the context words given a certain input word (= minimize objective function; average negative log likelihood):

$$J_\theta = -\frac{1}{T}\sum_{t=1}^{T} log\ p(w_{t-n}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+n}|w_t)$$

T ... training words in vocabulary V
J_θ ... objective function
θ ... all variables to be optimized
w_t ... center word provided as input
n .... window size (number of words before and after w_t to be considered)

Softmax function

Turn the obtained scores at the output layer into probabilities (softmax puts arbitrary values into a probability distribution):

$$P(o|c) = \frac{exp(u_o{}^\top v_c)}{\sum_{w \in V} exp(u_w{}^\top v_c)}$$

c ... center word

o ... context word

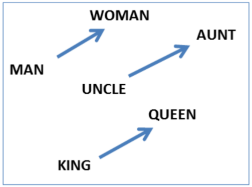
TECHNISCHE UNIVERSITÄT DRESDEN

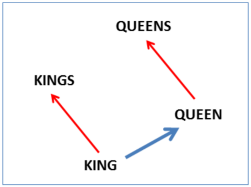
Hyperparameters

- corpus size / source / type
- context window
- vector dimensions
- context symmetry / position
- sampling frequent words / delete rare words

Example implementation in tensorflow: `https://github.com/tensorflow/tensorflow/blob/r1.8/tensorflow/examples/tutorials/word2vec/word2vec_basic.py`

Evaluation of Word Embeddings





Word Analogy:
a is to b as c is to ?

Characteristic: gender

$king - man + woman \approx queen$

How many word analogies can the trained embeddings predict correctly?

Image Source: Mikolov, T., Yih, W. T., & Zweig, G. (2013). Linguistic regularities in continuous space word representations.

Review of Lecture 8

- What is distributional semantics?
- What is the distributional hypothesis?
- What is SVD and how is it used in LSA?
- Can you explain the difference between distributional and distributed representations of words?
- What are word embeddings? How are they obtained?
- What are one-hot encodings and what are they good for?
- Which architectures does the word2vec model have and what is their difference?
- What is an analogy and how can it be used to evaluate word embeddings?