SEMANTIC COMPUTING

Lecture 7: Introduction to Distributional Semantics

Dagmar Gromann
International Center For Computational Logic

TU Dresden, 30 November 2018
Overview

- Distributional 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”


meaning = use = distribution in context => semantic distance

Dagmar Gromann, 30 November 2018
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 connotation; 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)

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.

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)
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

<table>
<thead>
<tr>
<th></th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
<th>$d_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>88</td>
<td>92</td>
<td>11</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>lion</td>
<td>57</td>
<td>28</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bark</td>
<td>80</td>
<td>62</td>
<td>10</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>car</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>93</td>
<td>97</td>
</tr>
<tr>
<td>tire</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>80</td>
<td>72</td>
</tr>
<tr>
<td>drive</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>90</td>
<td>45</td>
</tr>
</tbody>
</table>
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: $M_{mxn} = U_{mxm} \Sigma_{mxn} V_{nxn}^T$
- $U$ and $V$ represent orthogonal matrices and $\Sigma$ 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: $M_{reduced} = U_{mxk} \Sigma_{kxk} V_{nxk}^T$
LSA-SVD Example

Matrix $M_{mxn}$:

<table>
<thead>
<tr>
<th></th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
<th>$d_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>88</td>
<td>92</td>
<td>11</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>lion</td>
<td>57</td>
<td>28</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bark</td>
<td>80</td>
<td>62</td>
<td>10</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>car</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>93</td>
<td>97</td>
</tr>
<tr>
<td>tire</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>80</td>
<td>72</td>
</tr>
<tr>
<td>drive</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>90</td>
<td>45</td>
</tr>
</tbody>
</table>

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

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

Matrix $\Sigma_{mxn}$

<table>
<thead>
<tr>
<th></th>
<th>197.6</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>173.9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>27.78</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>20.93</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.744</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
<th>$d_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.055</td>
<td>-0.05</td>
<td>-0.011</td>
<td>-0.76</td>
<td>-0.64</td>
</tr>
<tr>
<td>2</td>
<td>0.75</td>
<td>0.65</td>
<td>0.085</td>
<td>-0.061</td>
<td>-0.043</td>
</tr>
<tr>
<td>3</td>
<td>-0.0037</td>
<td>0.015</td>
<td>-0.0097</td>
<td>0.64</td>
<td>-0.77</td>
</tr>
<tr>
<td>4</td>
<td>0.66</td>
<td>-0.75</td>
<td>-0.065</td>
<td>0.01</td>
<td>-0.0092</td>
</tr>
<tr>
<td>5</td>
<td>-0.022</td>
<td>-0.11</td>
<td>0.99</td>
<td>0.0036</td>
<td>-0.012</td>
</tr>
</tbody>
</table>
LSA-SVD Example

Matrix $M_{\text{reduced}}$:

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

Matrix $\Sigma_{kk}$:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>-0.059</td>
<td>0.73</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lion</td>
<td>-0.023</td>
<td>0.35</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bark</td>
<td>-0.042</td>
<td>0.58</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>car</td>
<td>-0.67</td>
<td>-0.053</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>tire</td>
<td>-0.54</td>
<td>-0.037</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>drive</td>
<td>-0.49</td>
<td>-0.039</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>-0.059</td>
<td>0.73</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lion</td>
<td>-0.023</td>
<td>0.35</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>bark</td>
<td>-0.042</td>
<td>0.58</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>car</td>
<td>-0.67</td>
<td>-0.053</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>tire</td>
<td>-0.54</td>
<td>-0.037</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>drive</td>
<td>-0.49</td>
<td>-0.039</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Matrix $V_{nxk}^T$ (document-to-concept):

<table>
<thead>
<tr>
<th></th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
<th>$d_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.055</td>
<td>-0.05</td>
<td>-0.011</td>
<td>-0.76</td>
<td>-0.64</td>
</tr>
<tr>
<td>2</td>
<td>0.75</td>
<td>0.65</td>
<td>0.085</td>
<td>-0.061</td>
<td>-0.043</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Dagmar Gromann, 30 November 2018  Semantic Computing 12
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 $\sim$ 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 be 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

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...
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

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."

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

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.

<table>
<thead>
<tr>
<th>Original words</th>
<th>id</th>
<th>red</th>
<th>blue</th>
<th>green</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>blue</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>green</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>blue</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
word2vec Algorithms

CBOW

Skip-gram

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
- \( W_{V \times N} \) ... weight matrix connecting input layer and hidden layer
- \( W_{N \times V} \) ... 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 \( V \)
\( J_\theta \) ... objective function
\( \theta \) ... 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 $V$
$J_{\theta}$ ... objective function
$\theta$ ... 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
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 ≈ queen

How many word analogies can the trained embeddings predict correctly?

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?