

# LECTURE 4

---

*Representation learning for text mining*

# WORD EMBEDDING

# GOAL OF WORD EMBEDDING

---

- Up until now we've represented words as an index in a vocabulary
- The goal of word embedding is to learn representations of words that capture their semantic
  - The vocabulary = a vector space  $U \in \mathbb{R}^{m \times d}$ 
    - Each word is represented by a vector  $u_i \in \mathbb{R}^d$
    - Similar words have similar vector representations, *i.e.* are close in this space
- Some linguistic operations = a linear function of word vectors

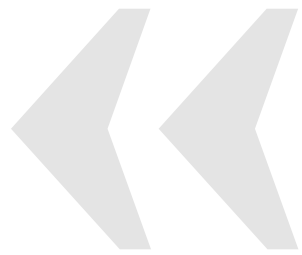
# SOME APPLICATIONS OF WORD EMBEDDING

---

- Supervised text classification
  - A document can be represented with a sequence of word vectors, which preserve word order and word meaning
  - A Long-Short Term Memory Recurrent Neural Network (LSTM) can deal with this kind of representation
- Query expansion
  - A query is a sequence of words
  - Add new relevant words to the query by interpreting the composition of the words

# DISTRIBUTIONAL SEMANTICS

---



You shall know a word by the company it keeps.

*-John Firth*

*Studies in Linguistic Analysis, 1957*

# DISTRIBUTIONAL SEMANTICS

---

- Distributional hypothesis
  - Words with similar meanings tend to appear in similar contexts
  - Word cooccurrence frequency
- Word embedding
  - Dense representations of words
  - Low-dimension vector space

# **SKIP-GRAM WITH NEGATIVE SAMPLING**

# MODEL

---

- Data

- $C$ : a raw textual corpus using a vocabulary of  $m$  words

- Embeddings

- $U \in \mathbb{R}^{m \times d}$  : target embeddings

- $V \in \mathbb{R}^{m \times d}$  : context embeddings

- Conditional probability of word  $i$  occurring, given word  $j$  is in its context

- $$p(w_i | w_j) = \frac{1}{1 + e^{-u_i^\top v_j}}$$
$$= \sigma(u_i^\top v_j)$$



# PARAMETER ESTIMATION

---

- Maximum likelihood estimation

- $\mathcal{L}(D; U, V) = \prod_{(i,j) \in D} \sigma(u_i^\top v_j)$

- Maximizing the log-likelihood is equivalent because the log is a monotonic, increasing function

- $U^*, V^* = \operatorname{argmax}_{U, V} \left( \prod_{(i,j) \in D} \sigma(u_i^\top v_j) \right) = \operatorname{argmax}_{U, V} \left( \sum_{(i,j) \in D} \log \sigma(u_i^\top v_j) \right)$

- This problem admits a trivial solution

- We can set all coefficients of  $U$  and  $V$  to a large enough constant, so that all the words have the same representations, because it leads to large dot-products, and thus conditional probabilities very close to 1

# PARAMETER ESTIMATION

---

- Maximum likelihood estimation
  - Add a negative sampling term
    - $U^*, V^* = \operatorname{argmax}_{U, V} \left( \sum_{(i, j) \in D} (\log \sigma(u_i^\top v_j) + k \mathbb{E}_{j' \sim q(j')} [\log \sigma(-u_i^\top v_{j'})]) \right)$ 
      - Where  $q$  is the noise distribution:  $q(j') \propto f(j')^{\frac{3}{4}}$
      - For every pair of co-occurring words  $(i, j)$ , randomly sample  $k$  negative, *i.e.* fake, context words  $j'$ 
        - Maximise  $1 - p(w_i | w_{j'})$
  - This is a supervised, binary, classification problem
    - SGNS aims at finding vectors that are good for distinguishing positive and negative pairs of co-occurring words

# **GLOBAL VECTORS**

## **AKA GLOVE**

# MODEL

---

- Data
  - $X$ , a square matrix that describes the number of co-occurrence between each of the  $m$  words of the vocabulary
- Embeddings
  - $U \in \mathbb{R}^{m \times d}$  : target embeddings
  - $V \in \mathbb{R}^{m \times d}$  : context embeddings
- GloVe specifies a matrix factorization problem
  - $\log(X) \simeq U^T V + B^U + B^V$
  - The more two words co-occur, the more their vectors should be similar

# PARAMETER ESTIMATION

---

- Weighted least-square

- $$J_{U,V,b^U,b^V} = \sum_{i=1}^m \sum_{j=1}^m f(x_{ij}) (u_i^\top v_j + b_i^U + b_j^V - \log(x_{ij}))^2$$

- Where  $f$  is the weighting function

- $$f(x_{ij}) = \begin{cases} \left(\frac{x_{ij}}{x_{max}}\right)^{\frac{3}{4}} & \text{if } x_{ij} \leq x_{max}, \\ 0 & \text{else.} \end{cases}$$

- There are two reasons for setting  $f(0) = 0$

- It zeros out elements of the sum where the log is undefined
- It allows for a lower time-complexity than SGNS due to Zipf's law (0 entries account for approx. 95% of  $X$ )

# IMPLEMENTATION OF SGNS AND GLOVE

---

- SGNS is implemented with stochastic gradient descent
  - Initialize randomly  $U$  and  $V$
  - For each positive ( $\gamma = 1$ ) or negative ( $\gamma = -1$ ) pair of words, update the vectors for words  $i$  and  $j$  in the direction of the gradient of  $\log(\sigma(\gamma u_i^\top v_j))$ , according to a fixed (per iteration) learning rate
- GloVe is implemented with AdaGrad
  - Variant of the stochastic gradient descent, where the learning rate is automatically adapted, for each word and each dimension, throughout learning, which helps converging faster