Deep Learning for Natural Language Processing

Training a word embedding model with the SGNS algorithm

UNIVERSITY OF GOTHENBURG

CHALMERS

WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM

Richard Johansson

richard.johansson@gu.se

training word embedding models on raw text

- we will now see how to train word embeddings from raw text
- according to the **distributional hypothesis**, word meaning is reflected in the distribution of contexts
- the key idea is to build a model of **cooccurrence**: for a word, what contexts are we likely to see?

training word embedding models on raw text

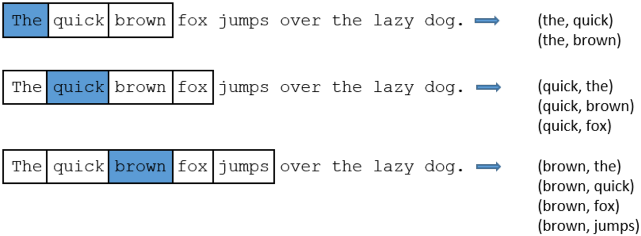
- we will now see how to train word embeddings from raw text
- according to the **distributional hypothesis**, word meaning is reflected in the distribution of contexts
- the key idea is to build a model of **cooccurrence**: for a word, what contexts are we likely to see?
- for instance: for ***coffee***
  - ***drink***, ***black***, ***cup*** are likely contexts
  - ***mosquito***, ***purple***, ***gradient*** are unlikely contexts

skip-gram with negative sampling (`word2vec`)

- the **skip-gram with negative sampling** (SGNS) model is a well-known training method (Mikolov et al., 2013)
- it is a simplification of several previous models
- in this model, we have one set of vectors V_T for the target words, and another set of vectors V_C for the contexts
- SGNS trains a model similar to logistic regression so that
  - $V_T(\mathit{coffee}) \cdot V_C(\mathit{drink})$ is high
  - $V_T(\mathit{coffee}) \cdot V_C(\mathit{gradient})$ is low

SGNS: the model

- collect **word–context pairs** occurring in the text data



[source]

- for each pair, randomly generate a number of **synthetic pairs**:

  fox: *mechanic*

  fox: *nitrogen*

  fox: *persuade*

SGNS: the model

- then fit the following model with respect to (V_T, V_C)

$$P(\text{true pair}|(w,c)) = \frac{1}{1+\exp\left(-V_T(w)\cdot V_C(c)\right)}$$

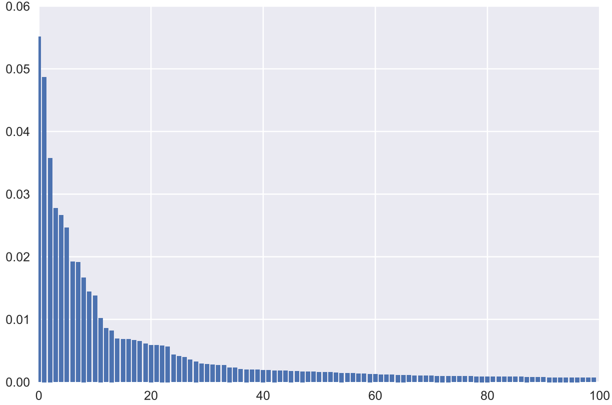
$$P(\text{synthetic pair}|(w,c)) = 1 - \frac{1}{1+\exp\left(-V_T(w)\cdot V_C(c)\right)}$$

- so the whole training objective becomes

$$\begin{aligned}\mathcal{L}(V_T, V_C) \quad = \quad & -\textstyle\sum_{(w,c)\in P} \log\sigma(V_T(w)\cdot V_C(c)) \\ & -\textstyle\sum_{(w,c)\in N} \log\sigma(-V_T(w)\cdot V_C(c))\end{aligned}$$

a challenge when training word embedding models

- word distributions are highly skewed



- remember **Zipf's law**

engineering tricks in SGNS: removing highly frequent words

- a word occurrence is removed from the input with a frequency-dependent probability:

$$P_{\text{remove}}(w) = \max(0, 1 - \sqrt{\frac{t}{\text{freq}(w)}})$$

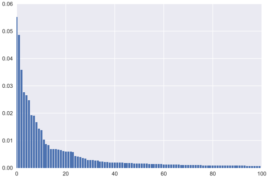
- reduces the influence of frequent words and speeds up training

engineering tricks in SGNS: negative sample distribution

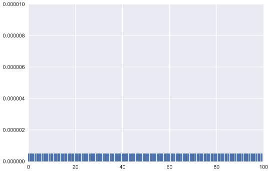
- SGNS uses the following distribution for drawing negative contexts:

$$P_{\text{neg}}(c) \propto \text{freq}(c)^{\alpha}$$

- α is set to a number between 0 and 1



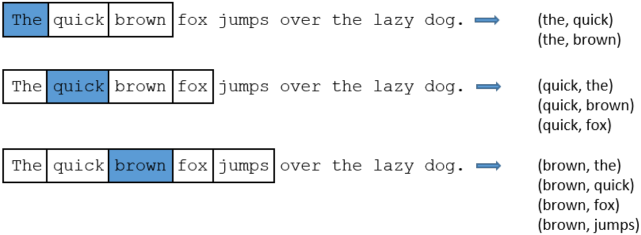
$\alpha = 1$



$\alpha = 0$

engineering tricks in SGNS: width of context window

- positive pairs are formed from words appearing in a window



- each time, a window width is drawn from a uniform distribution
- idea: distant words should be less influential

hyperparameters in SGNS

- **dimensionality** of embeddings: typically 50–300
- **number of negative samples**: typically 5
- **maximal width of the context window**: typically 5
- **smoothing constant** α for negative samples: typically 0.75
- **pruning threshold** t for frequent words: typically 0.0001

implementations

- `word2vec`: the software by Mikolov when he was at Google
  - implements the SGNS model (and a few others)
  - includes a model built by Google using a huge collection of news text
- `gensim`: a nice Python library by Řehůřek
  - includes a reimplementation of SGNS but also several other useful algorithms, such as LSA and LDA
  - the library also includes many pre-trained models

example notebooks

- using the Gensim library: training and loading models
- full implementation of the SGNS in PyTorch

references I

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.