STOR566: Introduction to Deep Learning

Lecture 11: NLP Pre-training

Yao Li
UNC Chapel Hill

Sep 22, 2022

Materials are from *Deep Learning (UCLA)*

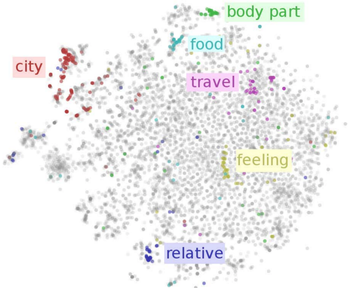
Unsupervised pretraining for NLP

Motivation

- Many unlabeled NLP data but very few labeled data
- Can we use large amount of unlabeled data to obtain meaningful representations of words/sentences?

Learning word embeddings

- Use large (unlabeled) corpus to learn a useful word representation
  - Learn a vector for each word based on the corpus
  - Hopefully the vector represents some semantic meaning
  - Can be used for many tasks
    - Replace the word embedding matrix for DNN models for classification/translation
- Two different perspectives but led to similar results:
  - Word2vec (Mikolov et al., 2013)
  - PPMI (Levy et al., 2014)
  - Glove (Pennington et al., 2014)



Context information

- Given a large text corpus, how to learn low-dimensional features to represent a word?
- For each word w_i, define the "contexts" of the word as the words surrounding it in an L-sized window:

$$w_{i-L-2}, w_{i-L-1}, \underbrace{w_{i-L}, \cdots, w_{i-1}}_{\text{contexts of } w_i}, w_i, \underbrace{w_{i+1}, \cdots, w_{i+L}}_{\text{contexts of } w_i}, w_{i+L+1}, \cdots$$

- Get a collection of (word, context) pairs, denoted by D.

Word pair

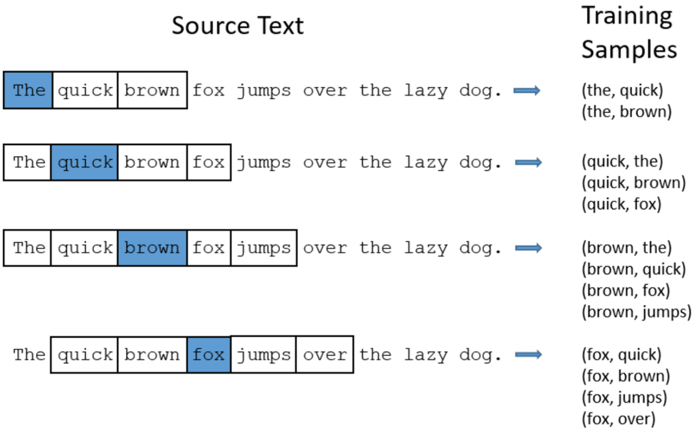


Figure from http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

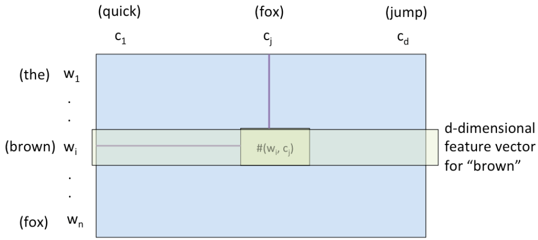
Use bag-of-word model

- Idea 1: Use the bag-of-word model to "describe" each word
- Assume we have context words $c_1, \cdots, c_d$ in the corpus, compute

$$\#(w, c_i) := \text{ number of times the pair } (w, c_i) \text{ appears in } D$$

- For each word w, form a d-dimensional (sparse) vector to describe w

$$\#(w, c_1), \cdots, \#(w, c_d),$$



PMI/PPMI Representation

- Similar to TF-IDF: Need to consider the frequency of each word and each context
- Instead of using co-ocurrent count $\#(w, c)$, we can define pointwise mutual information:

$$\text{PMI}(w, c) = \log\left(\frac{\hat{P}(w, c)}{\hat{P}(w)\hat{P}(c)}\right) = \log \frac{\#(w, c)|D|}{\#(w)\#(c)},$$

  - $\#(w) = \sum_c \#(w, c)$: number of pairs with word w
  - $\#(c) = \sum_w \#(w, c)$: number of pairs with word c
  - $|D|$: number of pairs in D

PMI/PPMI Representation

- Similar to TF-IDF: Need to consider the frequency of each word and each context
- Instead of using co-ocurrent count $\#(w, c)$, we can define pointwise mutual information:

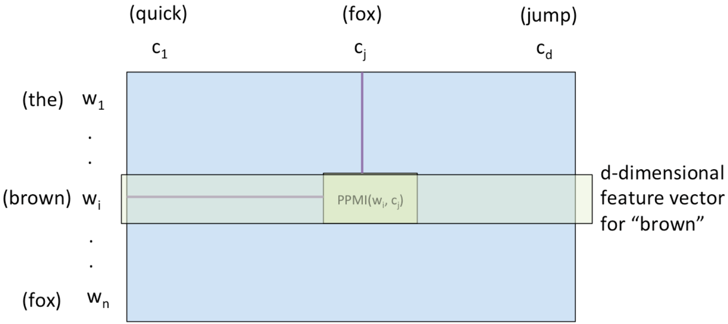
$$\text{PMI}(w, c) = \log(\frac{\hat{P}(w, c)}{\hat{P}(w)\hat{P}(c)}) = \log \frac{\#(w, c)|D|}{\#(w)\#(c)},$$

  - $\#(w) = \sum_c \#(w, c)$: number of pairs with word w
  - $\#(c) = \sum_w \#(w, c)$: number of pairs with word c
  - $|D|$: number of pairs in D
- Positive PMI (PPMI) usually achieves better performance:

$$\text{PPMI}(w, c) = \max(\text{PMI}(w, c), 0)$$

- M^{PPMI}: word feature matrix with $\text{PPMI}(w, c)$ as element

PPMI Matrix


(quick)
(fox)
(jump)
c1
cj
cd
(the) w1
(brown) wi
(fox) wn
PPMI(wi, cj)
d-dimensional
feature vector
for "brown"

Low-dimensional embedding

- Perform PCA/SVD on the sparse feature matrix:

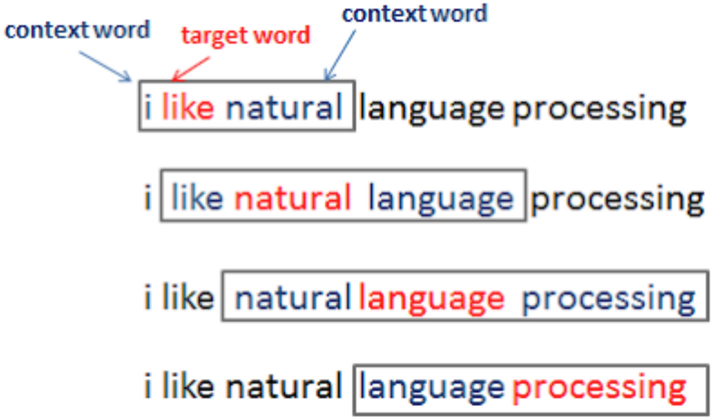
$$M^{\text{PPMI}} \approx U_k \Sigma_k V_k^T$$

  Then $W^{\text{SVD}} = U_k \Sigma_k$ is the context representation of each word (Each row is a k-dimensional feature for a word)

- $k << d$

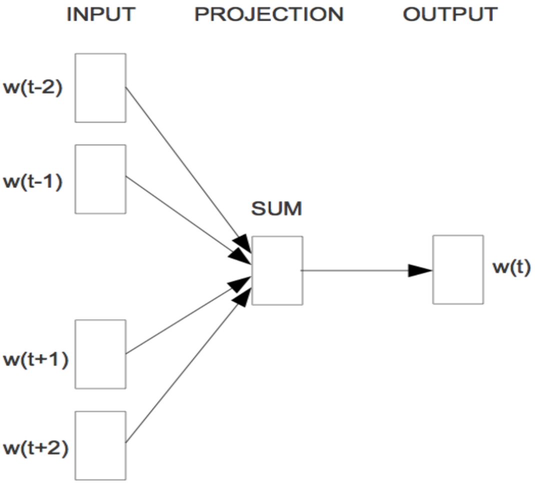
Word2vec (Mikolov et al., 2013)

- A neural network model for learning word embeddings
- Main idea:
  - Predict the target words based on the neighbors (CBOW)
  - Predict neighbors given the target words (Skip-gram)



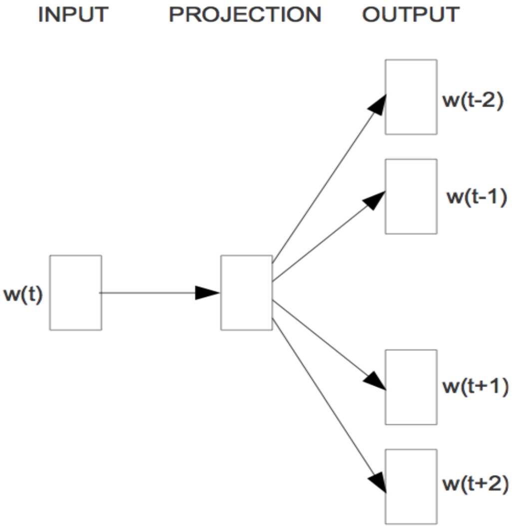
CBOW

- Predict the target words based on the neighbors



Skip-gram

- Predict neighbors using target word



More on skip-gram

- Learn the probability $P(w_{t+j}|w_t)$: the probability to see w_{t+j} in target word w_t's neighborhood
- Every word has two embeddings:
  - v_i serves as the role of target
  - u_i serves as the role of context
- Model probability as softmax:

$$P(o|c) = \frac{e^{u_o^T v_c}}{\sum_{w=1}^{W} e^{u_w^T v_c}}$$

Results

The low-dimensional embeddings are (often) meaningful:

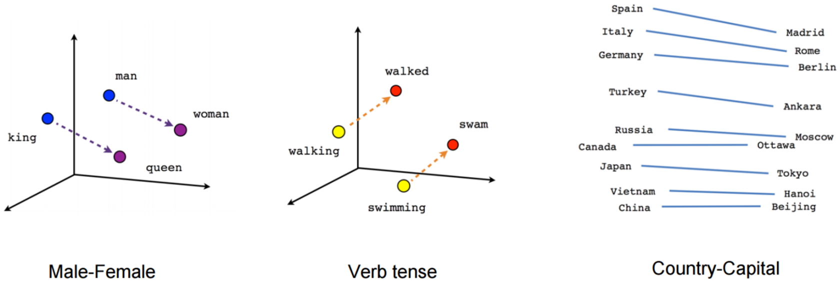


Figure from https://www.tensorflow.org/tutorials/word2vec

Conclusions

- PPMI
- Word2vec

Questions?