

# Word Embeddings: Beyond word2vec

Kostas Perifanos

*@k\_perifanos*

April 28, 2018

# Overview

Topic: An overview of word embedding models and their properties

- ▶ What are word embeddings?
- ▶ Why are they so important?

# Word Embeddings

Word embeddings: Mapping words (or more generally, entities from vocabularies) to vectors

- ▶ word2vec
- ▶ GloVe
- ▶ fastText
- ▶ StarSpace
- ▶ RAND-WALK
- ▶ ... and more

# Why Word Embeddings?

Typical text representation: 1-hot encoding

Given a collection of texts with a vocabulary of size  $V$ , word  $i$  in the vocabulary is represented with  $V$ -dimensional vector with zeros everywhere, 1 in position  $i$ .

Problems:

- ▶ all words in the vocabulary have the same distance, given a distance measure (eg cosine distance)
- ▶ We would like to use distance (or similarity) as a measure of semantic similarity
- ▶ Ideally, we would like a compact  $D$ -dimensional representation such that  $D \ll V$

# Word Analogies

Semantics as linear algebra

$$v(\textit{king}) - v(\textit{man}) + v(\textit{woman}) \approx v(\textit{queen})$$

# Distributional Hypothesis

*'Words that occur in similar contexts tend to have similar meanings'*  
Harris, 1954

*'You shall know a word by the company it keeps'*  
Firth, 1957

# Neural Language Models

Core idea: Approximate a Statistical Language Model with a neural network (Bengio et.al. 2003)

Statistical Language Model: Conditional Probability of the next word given the previous ones.

$$\hat{P}(w_i^T) = \prod_{i=1}^T \hat{P}(w_i | w_1^{t-1})$$

where  $w_i$  the  $i$ -th word and  $w_i^j = (w_i, w_{i+1}, \dots, w_{j-1}, w_j)$

Simplification (n-gram model):

$$\hat{P}(w_t | w_1^{t-1}) \approx P(w_t | w_{t-n+1}^{t-1})$$

Use a neural net to approximate the model (and also avoid assigning zero probability to sequences never appearing in a training corpus)

# word2vec

Fast forward 20 years later:

2013, Mikolov et. al. introduce word2vec. A very fast implementation of neural language models

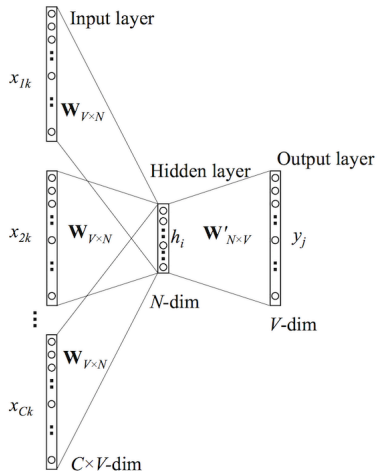
- ▶ CBOW : Predict the word given context
- ▶ SkipGram: Predict context given word

Dealing with performance:

- ▶ negative sampling
- ▶ hierarchical softmax
- ▶ async SGD



# word2vec - CBOW

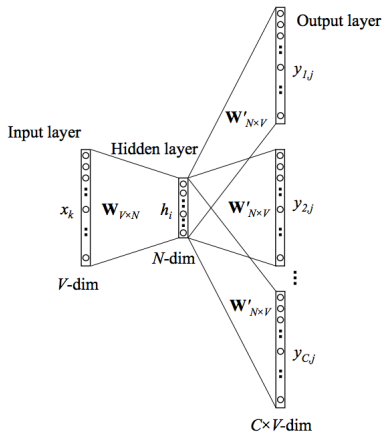


1

---

<sup>1</sup>Image Source: Wikipedia

# word2vec - SkipGram



2

<sup>2</sup>Image Source: Wikipedia

# GloVe

## Matrix Factorization

2014, Pennington et.al. introduce GloVe (Global Vectors).

Given a corpus, build a the co-occurrence matrix of the words in the vocabulary in a given context size.

Learning: Minimize the objective function

$$J = \sum_{i,j=1}^V f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log(X_{ij}))^2$$

# GloVe

## GloVe notation

where

$X$  the co-occurrence (usually symmetric) matrix

$w_i, \tilde{w}_j$  word vectors (left, right context)

$b_i, \tilde{b}_j$  bias for words  $i, j$

$$f(x) = \begin{cases} (x/x_{max})^a, & x < x_{max} \\ 1, & x \geq x_{max} \end{cases}$$

and finally, the embedding of the word  $i$  is defined as

$$e_i = w_i + \tilde{w}_j$$

$$a = 3/4, x_{max} = 100.$$

## Bag of Tricks

- ▶ Logic similar to word2vec / SkipGram model
- ▶ Word  $\rightarrow$  subword information  $\rightarrow$  n-grams  $\rightarrow$  embeddings for n-grams
- ▶ Word Embedding: average of the n-gram embeddings

## Comparing entities

Optimisation objective:

$$J = \sum L^{batch}(sim(a, b), sim(a, b_1^-), \dots, sim(a, b_n^-))$$

- ▶ Generator of positive pairs  $(a, b)$ , from the set  $E^+$
- ▶ Generator of negative entities  $b_j^-$  from the set  $E^-$  (negative sampling as in word2vec)
- ▶  $sim$ : Similarity function (cosine similarity, dot product)
- ▶  $L^{batch}$  comparison of the positive pair with the negative pairs. (hinge, softmax)

# Bonus Round: RAND-WALK

## A random walk over words

- ▶ A generative model
- ▶ Discourse direction vector  $c$
- ▶ Choose words sampling

$$P[\text{word emitted at time } t] \propto \exp(\langle w_t, c \rangle)$$

## Objective functions

$$J_{rw1} = \sum_{w, w'} X_{ij} (\log(X_{ij}) - \|u_w + u_{w'}\|^2 - C)^2$$

$$J_{rw2} = \sum_{w, w'} X_{ij} (PMI(w, w') - \langle w, w' \rangle)^2$$

$$PMI(w, w') = \log \frac{p(w, w')}{p(w)p(w')}$$

# Features / Sentence Representation

## Building features

Word embeddings and "traditional" ML, eg features for my SVM

- ▶ the average of  $n$  vectors
- ▶ average of the unit norm of the vectors [fastText]
- ▶ Smooth Inverse Frequency (SIF): First principal component of the weighted average of the vectors



## C/C++ implementations

- ▶ word2vec : <https://code.google.com/p/word2vec/>
- ▶ GloVe : <https://nlp.stanford.edu/projects/glove/>
- ▶ fastText : <https://github.com/facebookresearch/fastText>
- ▶ StarSpace : <https://github.com/facebookresearch/StarSpace>
- ▶ RAND-WALK : TBA [work in progress]

## Python

- ▶ gensim : <https://radimrehurek.com/gensim/>
- ▶ glove-python <https://github.com/maciejkula/glove-python>

# Applications in NLP

## Metaphor detection

- ▶ train (or download) an embedding model
- ▶ label metaphor and literal examples
- ▶ locate the verb in a sentence, get the context, average embeddings
- ▶ train a model with LogisticRegression or SVC

## Automating Curation

- ▶ news stream: approved vs non-approved
- ▶ use pre-tagged articles, extract document embeddings
- ▶ feed to classifier

# Pre-trained Embedding Layers

```
from keras.layers import Embedding








embedding_layer = Embedding(len(word_index) + 1,
                             EMBEDDING_DIM,
                             weights=[embedding_matrix],
                             input_length=MAX_SEQUENCE_LENGTH,
                             trainable=False)
```

# Questions and properties

## Interesting properties

- ▶ relation of word2vec with pairwise mutual information (PMI) [Levy et. al.]
- ▶ relatively small dimensions work better ( 500)
- ▶ See RAND-WALK [Arora et. al.]

# References

-  Efficient Estimation of Word Representations in Vector Space, Mikolov et. al. 2013
-  GloVe: Global Vectors for Word Representation, Pennington et. al. 2014
-  Bag of Tricks for Efficient Text Classification, Joulin, et. al. 2016
-  StarSpace: Embed All The Things!, Joulin, et. al. 2017
-  A Simple but Tough-to-Beat Baseline for Sentence Embeddings, Arora et. al. 2016
-  RAND-WALK: A Latent Variable Model Approach to Word Embeddings, Arora et. al. 2017
-  Neural Word Embedding as Implicit Matrix Factorization, Levy et. al. 2014

The End (and thanks!)