Statistical Learning Models for Text and Graph Data
Word Embeddings

Yangqiu Song

Hong Kong University of Science and Technology

yqsong@cse.ust.hk

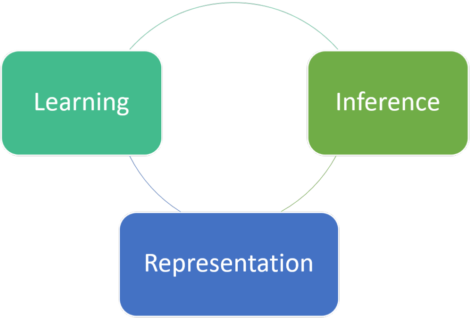
September 20, 2019

∗Contents are based on materials created by Noah Smith, Richard Socher, Percy Liang, Hongning Wang, David Jurgens, Mohammad Taher Pilehvar, Maneesh Sahani

Reference Content

- Noah Smith. CSE 517: Natural Language Processing https://courses.cs.washington.edu/courses/cse517/16wi/
- Richard Socher. CS224d: Deep Learning for Natural Language Processing. https://web.stanford.edu/class/cs224d/
- Percy Liang. ICML tutorial on Natural Language Understanding: Foundations and State-of-the-Art https://icml.cc/2015/tutorials/icml2015-nlu-tutorial.pdf
- Hongning Wang. CS6501 Text Mining. http://www.cs.virginia.edu/~hw5x/Course/Text-Mining-2015-Spring/_site/
- David Jurgens and Mohammad Taher Pilehvar. EMNLP 2015 Tutorial – Semantic Similarity Frontiers: From Concepts to Documents. http://www.emnlp2015.org/tutorials/34/34_OptionalAttachment.pdf
- Maneesh Sahani. Dimensionality Reduction. http://www.gatsby.ucl.ac.uk/~maneesh/dimred/dimred.pdf

Course Organization



- Representation: language models, word embeddings, topic models, knowledge graphs
- Learning: supervised learning, semi-supervised learning, distant supervision, indirect supervision, sequence models, deep learning, optimization techniques
- Inference: constraint modeling, joint inference, search algorithms

Overview

1. Overview
   - Language Models: Recap
   - Vector Space Model
2. Word Embeddings
   - Efficiency: Hierarchical Softmax
   - Efficiency: Negative Sampling
   - Evaluation

Paragraphs of Text

- A language model is a probability distribution over $\mathcal{V}^\dagger$
- Typically P decomposes into probabilities $P(x_i|\mathbf{h}_i)$
    - We considered n-gram, log-linear, and neural language models, etc.
- Today: probabilistic models that relate a word and its cotext (the linguistic environment of the word)
- This might help us learn to represent words, contexts, or both

Three Kinds of Cotext

If we consider a word token at a particular position i in text to be the observed value of a random variable X_i, what other random variables are predictive of/related to X_i?

- The words that occur within a small "window" around i (e.g., $x_{i-2}, x_{i-1}, x_{i+1}, x_{i+2}$, or maybe the sentence containing i) $\rightarrow$ distributional semantics
- The document containing i (a moderate-to-large collection of other words) $\rightarrow$ topic models
- A sentence known to be a translation of the one containing i $\rightarrow$ translation models

Overview

Context of Words

Example

Let's try to keep the kitchen ______.

Example

We used log-linear model to ______ the test data set.

What does ______ mean?

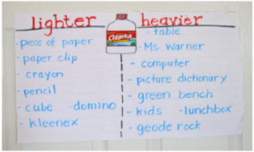
Let's try to keep the kitchen ____.

- Observation: context can tell us a lot about word meaning
- Context: local window around a word occurrence (for now)
- Roots in linguistics:
  - Distributional hypothesis: Semantically similar words occur in similar contexts (Harris (1954))
  - "You shall know a word by the company it keeps." (Firth (1957))
- Pros: data-driven, easy to implement
- Cons: ambiguity

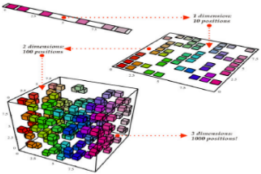
Corpus based Approach

1) Corpus

2) Preprocessing



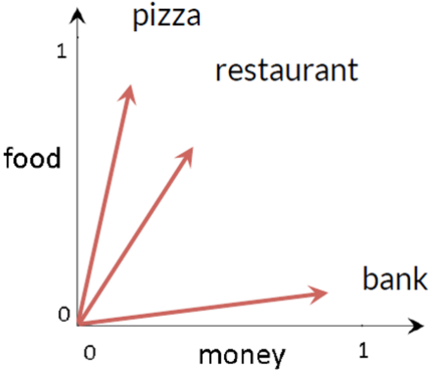
3) Dimensionality Reduction



4) Post Processing

Vector Space Model (VSM)

- Represent each word with its context words



Local Contexts: Distributional Semantics

- Within NLP, emphasis has shifted from topics to the relationship between $v \in \mathcal{V}$ and more local contexts
- These models are designed to "guess" a word at position i given a word at a position in $[i-c, i-1] \cup [i+1, i+c]$
- Sometimes such methods are used to "pre-train" word vectors used in other, richer models (like neural language models)

Context Vector Construction

- Form a word-context matrix of counts (data)

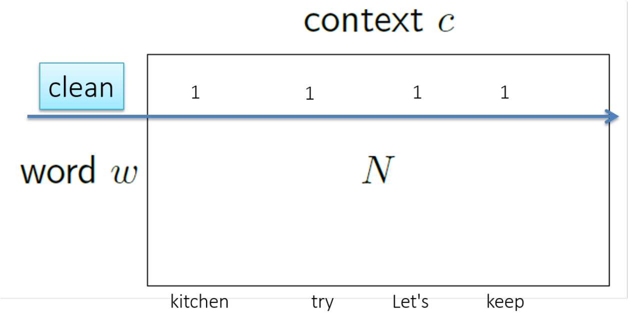


Figure: "Let's try to keep the kitchen clean."

Context Vector Construction

- Words on left, words on right

	cats_L	dogs_L	tails_R	have_L	have_R
cats	0	0	0	0	1
dogs	0	0	0	0	1
have	1	1	1	0	0
tails	0	0	0	1	0

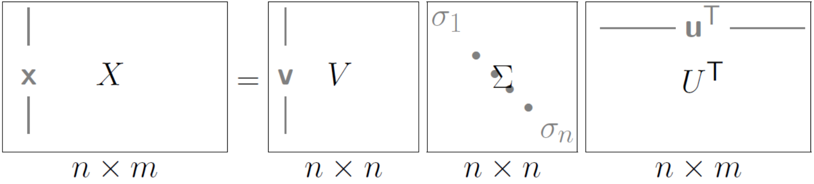
Figure: "Doc1: Cats have tails. Doc2: Dogs have tails."

- Usually used for part-of-speech induction

Dimensionality Reduction: SVD

Singular Value Decomposition (SVD):

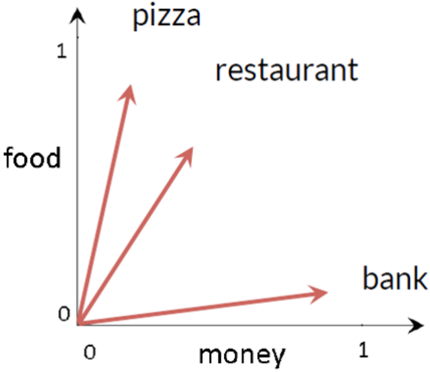
- Let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$, where $\mathbf{x}_i \in \mathbb{R}^n$, so $\mathbf{X} \in \mathbb{R}^{n \times m}$
- SVD computes $\mathbf{X} = \mathbf{V}\Sigma\mathbf{U}^\top$ with
  - $\mathbf{V}\mathbf{V}^\top = \mathbf{V}^\top\mathbf{V} = \mathbf{I}$, the orthonormal basis $\{\mathbf{v}_i\}$ for the columns of $\mathbf{X}$
  - $\mathbf{U}\mathbf{U}^\top = \mathbf{V}^\top\mathbf{V} = \mathbf{I}$, the orthonormal basis $\{\mathbf{u}_i\}$ for the rows of $\mathbf{X}$
  - Σ is a diagonal matrix containing singular values in decreasing order $\sigma_1 \geq \sigma_2 > \cdots > \sigma_n$ (if $n < M$)



Truncated at k: Approximating $\mathbf{X}$ by truncating $\sigma_i < \theta$ equates to a "low rank approximation"

Similarity between Words

$$cos(\theta) = \frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\|_2 \|\mathbf{b}\|_2}$$



Overview

Problems with SVD

- Computational cost scales quadratically for $n \times m$ matrix: $O(mn^2)$ flops (when $n < m$)
  - Could be less when the matrix is sparse, but still very inefficient in practice
  - Bad for millions of words or documents
- Hard to incorporate new words or documents
- Different learning regime than other DL models

Idea: Directly Learn Low-dimensional Word Vectors

- Old idea
  - Learning representations by back-propagating errors (Rumelhart et al. (1986))
  - A neural probabilistic language model (Bengio et al. (2003)
  - NLP (almost) from Scratch (Collobert et al. (2011))
- A recent, even simpler and faster model: word2vec
- Usually called distributed representations in the context of deep learning
  - Vector representation does not represent a distribution, but distributed over the space
  - Term widely used in connectionism (Hinton (1986)):
    - "In the componential approach each concept is simply a set of features and so a neural net can be made to implement a set of concepts by assigning a unit to each feature and setting the strengths of the connections between units so that each concept corresponds to a stable pattern of activity distributed over the whole network."

Main Idea of Word2vec

Mikolov et al. (2013a,b)

- Instead of capturing co-occurrence counts directly
- Predict surrounding words of every word
- Faster and can easily incorporate a new sentence/document or add a word to the vocabulary

Word2vec

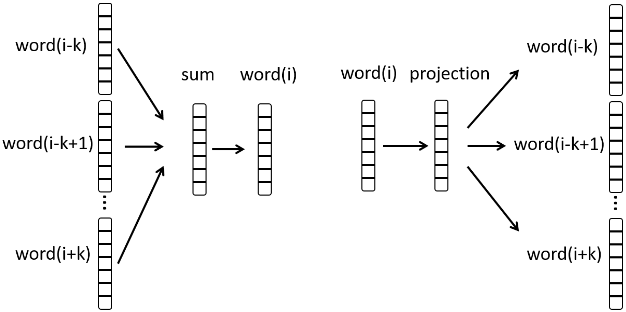
Mikolov et al. (2013a,b)

- Two models for word vectors designed to be computationally efficient
  - Continuous bag of words (CBOW): $P(v|\mathcal{C}_w)$
    - Similar in spirit to the feedforward neural language model we saw before (Bengio et al. (2003))
  - Skip-gram: : $P(\mathcal{C}_w|v)$
- It turns out these are closely related to matrix factorization as in LSI/A (Levy and Goldberg (2014))

Word2vec: Overview of Two Models

Mikolov et al. (2013a,b)

- Continuous bag of words (CBOW): $P(v|\mathcal{C}_w)$
  - Use the context words (average) to predict the center word
- Skip-gram: : $P(\mathcal{C}_w|v)$
  - Use the center word to predict each of the context words



(a) CBOW (b) Skipgram

CBOW

Skipgram can be derived similarly

- Given a sequence of training words $w_1, w_2, \dots, w_N$ in $\mathcal{W}$, the training objective is to maximize the average negative log-likelihood function

$$\mathcal{J}_{CBOW} = -\sum_{w \in \mathcal{W}} \log P(w|\mathcal{C}_w)$$
$$= -\sum_{w \in \mathcal{W}} \log \frac{\exp(\mathbf{v}_w^\top \mathbf{h}_c)}{\sum_{w' \in \mathcal{V}} \exp(\mathbf{v}_{w'}^\top \mathbf{h}_c)}$$

where $\mathbf{h}_c = \frac{1}{|\mathcal{C}_w|} \sum_{c \in \mathcal{C}_w} \mathbf{u}_c$, $\mathcal{C}_w$ contains all words shown in a small window around w

  - For all $w, c \in \mathcal{V}$, $\mathbf{v}_w$ (parameters) and $\mathbf{u}_c$ (word embedding) are two sets of vectors

- When performing SGD to this cost function
  - Non-convex: optimize $\mathbf{v}_w$ and $\mathbf{u}_c$ simultaneously
  - Inefficient to compute $\sum_{c \in \mathcal{V}} \exp(\mathbf{v}_c^\top \mathbf{h}_c)$ for large vocabulary $\mathcal{V}$

Overview

How to Improve Efficiency?

Approach 1: a surrogate loss, hierarchical softmax

- Class based model was first proposed by Goodman (2001)

$$P(w|\mathcal{C}_w) = \sum_{l} P(w, class(w) = l|\mathcal{C}_w) = P(w, class(w) = l|\mathcal{C}_w)$$

  since only one class label l is compilable with the hard clustering. So

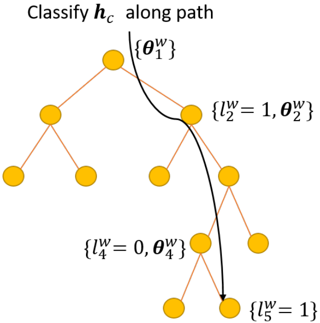
$$P(w|\mathcal{C}_w) = P(w|class(w) = l, \mathcal{C}_w)P(class(w) = l|\mathcal{C}_w)$$

- Although any $l(\cdot)$ would yield correct probabilities, generalization could be better for choices of word classes that "make sense," i.e., those for which it easier to learn the $P(class(w) = l|\mathcal{C}_w)$ (Morin and Bengio (2005))

Approach 1: Hierarchical Softmax

A generalization of class based model is to apply it for a tree of words, using

- l_i^w label of i-th node in path, $i \in \{2, \ldots, L^w\}$
- $\boldsymbol{\theta}_i^w$ as the vector representation of the i-th node in path



If we use a binary tree $l_i^w \in \{-1, 1\}$, then the classification sequence along a path consists of a sequence of logistic regression:

$$P(w|\mathcal{C}_w) = \sum_{i=2}^{L^w} P(l_i^w|\mathbf{h}_c, \boldsymbol{\theta}_{i-1}^w) = \sum_{i=2}^{L^w} \sigma(l_i^w \mathbf{h}_c^\top \boldsymbol{\theta}_{i-1}^w)$$

where $\sigma(x) = 1/(1 + \exp(-x))$

Approach 1: Hierarchical Softmax (Cont'd)

Then the objective function can be written as:

$$\mathcal{J}_{CBOW}^{HS} = -\sum_{w \in \mathcal{W}} \sum_{i=2}^{L^w} \log[\sigma(l_i^w \mathbf{h}_c^\top \boldsymbol{\theta}_{i-1}^w)]$$

Options to build the tree of words

- Wordnet (Morin and Bengio (2005))
- Hierarchical clustering (Mnih and Hinton (2008))
- Huffman tree based on word frequrencies (Mikolov et al. (2013a,b))
  - Complexity reduced from V to $\log_2 V$
  - If consider lower-frequency words are deeper, the practical performance is further improved (higher frequency words are accessed more frequently)

Hierarchical Softmax: Optimization

- For each word w at position i along the path, we denote

$$\mathcal{J}_{CBOW}^{HS}(w,i) = -\log[\sigma(l_i^w \mathbf{h}_c^\top \boldsymbol{\theta}_{i-1}^w)] = \log[1+\exp(-l_i^w \mathbf{h}_c^\top \boldsymbol{\theta}_{i-1}^w)]$$

- So we have:

$$\frac{\partial \mathcal{J}_{CBOW}^{HS}(w,i)}{\partial \boldsymbol{\theta}_{i-1}^w} = -l_i^w \sigma(-l_i^w \mathbf{h}_c^\top \boldsymbol{\theta}_{i-1}^w)\mathbf{h}_c$$

- So we have SGD for $\boldsymbol{\theta}_{i-1}^w$

$${\boldsymbol{\theta}_{i-1}^w}^{(t+1)} = {\boldsymbol{\theta}_{i-1}^w}^{(t)} + \eta l_i^w \sigma(-l_i^w \mathbf{h}_c^\top \boldsymbol{\theta}_{i-1}^w)\mathbf{h}_c$$

Hierarchical Softmax: Optimization

- Similarly, we have:

$$\frac{\partial \mathcal{J}_{CBOW}^{HS}(w,i)}{\partial \mathbf{h}_c} = -l_i^w \sigma(-l_i^w \mathbf{h}_c^\top \boldsymbol{\theta}_{i-1}^w)\boldsymbol{\theta}_{i-1}^w$$

which leads to

$$\mathbf{u}_c^{(t+1)} = \mathbf{u}_c^{(t)} + \eta \sum_{i=2}^{L^w} \frac{1}{|\mathcal{C}_w|} l_i^w \sigma(-l_i^w \mathbf{h}_c^\top \boldsymbol{\theta}_{i-1}^w)\boldsymbol{\theta}_{i-1}^w$$

since $\mathbf{h}_c = \frac{1}{|\mathcal{C}_w|}\sum_{c\in\mathcal{C}_w} \mathbf{u}_c$ and $\mathcal{C}_w$ contains all words shown in a small window around w

Overview

How to Improve Efficiency?

Approach 2: transforming the computationally expensive learning problem into a binary classification *proxy problem* that uses the same parameters but requires statistics that are easier to compute

- Recall the CBOW objective is

$$\mathcal{J}_{CBOW} = -\sum_{w \in \mathcal{W}} \log P(w|\mathcal{C}_w)$$
$$= -\sum_{w \in \mathcal{W}} \log \frac{\exp(\mathbf{v}_w^\top \mathbf{h}_c)}{\sum_{c \in \mathcal{V}} \exp(\mathbf{v}_{w'}^\top \mathbf{h}_c)}$$

  where $\mathbf{h}_c = \frac{1}{|\mathcal{C}_w|} \sum_{c \in \mathcal{C}_w} \mathbf{u}_c$, $\mathcal{C}_w$ contains all words shown in a small window around w
- We simplify the probability to be $P(w|\mathcal{C}_w) = P_\theta(w|c)$
- We denote the parameters as $\theta = \{\mathbf{v}_w, w \in \mathcal{V}\}$

Noise Contrastive Estimation (NCE)

- We denote the empirical distributions as $\tilde{P}(w|c)$ and $\tilde{P}(c)$
- We use the parameterized distribution $P_\theta(w|c)$ to approximate $\tilde{P}(w|c)$
- To avoid costly summations, a "noise" distribution, $Q(w)$, is used
  - In practice Q is a uniform, empirical unigram, or flattened empirical unigram distribution
- NCE reduces the estimation problem to the problem of estimating the parameters of a probabilistic binary classifier that
  - uses the same parameters to distinguish samples from the empirical distribution from samples generated by the noise distribution

$$P(d, w|c) = \begin{cases} \frac{k}{k+1} Q(w) & \text{if } d = 0 \\ \frac{1}{k+1} \tilde{P}(w|c) & \text{if } d = 1 \end{cases}$$

1. Sample a c from $\tilde{P}(c)$ and given c
2. Sample a w from $\tilde{P}(w|c)$ (true distribution) and k of w from $Q(w)$ (noise)

Noise Contrastive Estimation (NCE) (Cont'd)

- From the joint conditional probability

$$P(d, w|c) = \begin{cases} \frac{k}{k+1} Q(w) & \text{if } d = 0 \\ \frac{1}{k+1} \tilde{P}(w|c) & \text{if } d = 1 \end{cases}$$

  - Using definition of conditional probability

$$P(d|c, w) = \begin{cases} \frac{\frac{k}{k+1} Q(w)}{\frac{k}{k+1} Q(w) + \frac{1}{k+1} \tilde{P}(w|c)} = \frac{kQ(w)}{\tilde{P}(w|c) + kQ(w)} & \text{if } d = 0 \\ \frac{\tilde{P}(w|c)}{\tilde{P}(w|c) + kQ(w)} & \text{if } d = 1 \end{cases}$$

- NCE replaces the empirical distribution $\tilde{P}(w|c)$ with the model distribution $P_\theta(w|c)$, and θ is chosen to maximize the conditional likelihood of the "proxy corpus" created as described above

Noise Contrastive Estimation (NCE) (Cont'd)

- So we have

$$P_\theta(d|c,w) = \begin{cases} \frac{kQ(w)}{P_\theta(w|c)+kQ(w)} & \text{if } d=0 \\ \frac{P_\theta(w|c)}{P_\theta(w|c)+kQ(w)} & \text{if } d=1 \end{cases}$$

- Recall the original (simplified) negative log likelihood is

$$\mathcal{J}_{CBOW} = -\sum_{w\in\mathcal{W}} \log P_\theta(w|c) = -\mathbb{E}_{\tilde{P}(w|c)} \log P_\theta(w|c)$$

- With NCE, θ can be trained to maximize the expectation of $\log P_\theta(d|c,w)$ under the mixture of the data and noise samples

$$\mathcal{J}_{NCE_k} = -\mathbb{E}_{\tilde{P}(w|c)}\left[\log \frac{P_\theta(w|c)}{P_\theta(w|c)+kQ(w)}\right] - k\mathbb{E}_{Q(w)}\left[\log \frac{kQ(w)}{P_\theta(w|c)+kQ(w)}\right]$$

Asymptotic Property

- Given that

$$\mathcal{J}_{NCE_k} = -\mathbb{E}_{\tilde{P}(w|c)}\left[\log \frac{P_\theta(w|c)}{P_\theta(w|c) + kQ(w)}\right] - k\mathbb{E}_{Q(w)}\left[\log \frac{kQ(w)}{P_\theta(w|c) + kQ(w)}\right]$$

- The gradient is

$$\begin{aligned}\frac{\partial \mathcal{J}_{NCE_k}}{\partial \theta} =& -\mathbb{E}_{\tilde{P}(w|c)}\left[\frac{kQ(w)}{P_\theta(w|c) + kQ(w)} \frac{\partial}{\partial \theta} \log P_\theta(w|c)\right] \\ &+ k\mathbb{E}_{Q(w)}\left[\frac{P_\theta(w|c)}{P_\theta(w|c) + kQ(w)} \frac{\partial}{\partial \theta} \log P_\theta(w|c)\right] \\ =& -\sum_w \frac{kQ(w)}{P_\theta(w|c) + kQ(w)}\left[\tilde{P}(w|c) - P_\theta(w|c)\right] \frac{\partial}{\partial \theta} \log P_\theta(w|c)\end{aligned}$$

- As $k \to \infty$, we have

$$\frac{\partial \mathcal{J}_{NCE_k}}{\partial \theta} \to -\sum_w \left[\tilde{P}(w|c) - P_\theta(w|c)\right] \frac{\partial}{\partial \theta} \log P_\theta(w|c)$$

which is the maximum likelihood gradient (the gradient is 0 when the model distribution matches the empirical distribution)

Practical Issues

- In practice, we do random sampling (Monte Carlo approximation) to generate k noise samples to perform estimation

$$
\begin{aligned}
\mathcal{J}_{NCE_k} &= -\sum_{w,c\in\mathcal{D}} [\log P(d=1|c,w) + k\mathbb{E}_{w'\sim Q(w)} \log P(d=0|c,w')] \\
&\approx -\sum_{w,c\in\mathcal{D}} [\log P(d=1|c,w) + k \sum_{i=1,w'\sim Q(w)}^{k} \frac{1}{k} \log P(d=0|c,w')] \\
&= -\sum_{w,c\in\mathcal{D}} [\log P(d=1|c,w) + \sum_{i=1,w'\sim Q(w)}^{k} \log P(d=0|c,w')]
\end{aligned}
$$

- Then the stochastic gradient is

$$
\begin{aligned}
\frac{\partial \mathcal{J}^{w}_{NCE_k}}{\partial\theta} = &-[\frac{kQ(w)}{P_\theta(w|c)+kQ(w)} \frac{\partial}{\partial\theta} \log P_\theta(w|c) \\
&- \sum_{i=1,w'\sim Q(w)}^{k} \frac{P_\theta(w'|c)}{P_\theta(w'|c)+kQ(w')} \frac{\partial}{\partial\theta} \log P_\theta(w'|c)]
\end{aligned}
$$

Practical Issues

- NCE replaces the empirical distribution $\tilde{P}(w|c)$ with the model distribution $P_\theta(w|c)$
  - But $P_\theta(w|c) = \frac{f_\theta(w,c)}{\sum_{w'} f_\theta(w',c)} = \frac{f_\theta(w,c)}{Z_\theta(c)}$ still requires evaluating the partition function $Z_\theta(c)$
  - Original NCE introduce a parameter to estimate $Z_\theta(c)$ for every possible c, which is still huge in language models (Mnih and Teh (2012))
    - This approach is, however, not possible for Maximum Likelihood Estimation since the likelihood can be made arbitrarily large by making Z go to zero.(Gutmann and Hyvärinen (2012))
  - For neural networks, original paper simply set $Z_\theta(c) = 1$ (Mnih and Teh (2012)), which result in

$$P(d|c,w) = \begin{cases} \frac{kQ(w)}{f_\theta(w,c)+kQ(w)} & \text{if } d = 0 \\ \frac{f_\theta(w,c)}{f_\theta(w,c)+kQ(w)} & \text{if } d = 1 \end{cases}$$

  and claimed they found comparable results
  - Intuitively, by parameterizing $\log P_\theta(w,c)$, $\log Z_\theta$ can be considered as a bias term in addtition to the parameters of $\log f_\theta(w,c)$

Negative Sampling for Word2vec

Mikolov et al. (2013a,b)

- Now we derive negative sampling for word2vec and compare with general NCE strategy
- The proposed proxy distribution of negative sampling is

$$P(d|c,w) = \begin{cases} \frac{1}{f_\theta(w,c)+1} & \text{if } d = 0 \\ \frac{f_\theta(w,c)}{f_\theta(w,c)+1} & \text{if } d = 1 \end{cases}$$

Compared to NCE:

$$P(d|c,w) = \begin{cases} \frac{kQ(w)}{f_\theta(w,c)+kQ(w)} & \text{if } d = 0 \\ \frac{f_\theta(w,c)}{f_\theta(w,c)+kQ(w)} & \text{if } d = 1 \end{cases}$$

  - Negative sampling is equivalent to NCE when $k = |\mathcal{V}|$ and $Q(w)$ is uniform
  - Aside from the $k = |\mathcal{V}|$ and uniform $Q(w)$ case, the conditional probabilities of d given (w, c) are not consistent with the language model probabilities of $P_\theta(w|c)$
    - It does not have the same asymptotic consistency guarantees that NCE has (when $k \to \infty$)

Negative Sampling for CBOW

- Recall the CBOW objective is

$$\mathcal{J}_{CBOW} = -\sum_{w \in \mathcal{W}} \log P(w|\mathcal{C}_w) = -\sum_{w \in \mathcal{W}} \log \frac{\exp(\mathbf{v}_w^\top \mathbf{h}_c)}{\sum_{w' \in \mathcal{V}} \exp(\mathbf{v}_{w'}^\top \mathbf{h}_c)}$$

  where $\mathbf{h}_c = \frac{1}{|\mathcal{C}_w|} \sum_{c \in \mathcal{C}_w} \mathbf{u}_c$, $\mathcal{C}_w$ contains all words shown in a small window around w
- By introducing the proxy distribution:

$$P(d|c,w) = \begin{cases} \frac{1}{f_\theta(w,c)+1} & \text{if } d = 0 \\ \frac{f_\theta(w,c)}{f_\theta(w,c)+1} & \text{if } d = 1 \end{cases}$$

- We have the following objective function for (CBOW) word embedding with negative sampling:

$$\mathcal{J}_{CBOW}^{NS} = -\sum_{w \in \mathcal{W}} [\log \frac{\exp(\mathbf{v}_w^\top \mathbf{h}_c)}{\exp(\mathbf{v}_w^\top \mathbf{h}_c) + 1} + \sum_{w' \in Q(w)}^{k} \log \frac{1}{\exp(\mathbf{v}_{w'}^\top \mathbf{h}_c) + 1}]$$

Learning with Negative Sampling

- Starting from the objective of CBOW using negative sampling

$$
\begin{aligned}
\mathcal{J}_{CBOW}^{NS} &= -\sum_{w \in \mathcal{W}} [\log \frac{\exp(\mathbf{v}_w^\top \mathbf{h}_c)}{\exp(\mathbf{v}_w^\top \mathbf{h}_c) + 1} + \sum_{w' \in Q(w)}^{k} \log \frac{1}{\exp(\mathbf{v}_{w'}^\top \mathbf{h}_c) + 1}] \\
&= -\sum_{w \in \mathcal{W}} [\log \sigma(\mathbf{v}_w^\top \mathbf{h}_c) + \sum_{w' \in Q(w)}^{k} \log \sigma(-\mathbf{v}_{w'}^\top \mathbf{h}_c)] \\
&\doteq -\sum_{u \in \mathcal{W} \cup \mathcal{N}(w)} \log \sigma(l^u \mathbf{v}_u^\top \mathbf{h}_c) \\
&= \sum_{u \in \mathcal{W} \cup \mathcal{N}(w)} \log[1 + \exp(-l^u \mathbf{v}_u^\top \mathbf{h}_c)]
\end{aligned}
$$

where $\mathcal{N}(w)$ is the set of negative sampling, l^u is a binary label:

  - $l^u = 1$ represents the word is from empirical distribution and $u = w$
  - $l^u = -1$ represents the word is from the proxy distribution and $u = w'$

Learning with Negative Sampling (Cont'd)

- So the gradient of $\mathcal{J}_{CBOW}^{NS}$ w.r.t. $\boldsymbol{\theta}_w = \mathbf{v}_u$ is

$$\frac{\partial \mathcal{J}_{CBOW}^{NS}}{\partial \mathbf{v}_u} = -l^u \sigma(-l^u \mathbf{v}_u^\top \mathbf{h}_c)\mathbf{h}_c$$

and w.r.t. $\mathbf{h}_c$ is

$$\frac{\partial \mathcal{J}_{CBOW}^{NS}}{\partial \mathbf{h}_c} = -l^u \sigma(-l^u \mathbf{v}_u^\top \mathbf{h}_c)\mathbf{v}_u$$

- So SGD for $\boldsymbol{\theta}_w = \mathbf{v}_u$ is

$$\mathbf{v}_u^{t+1} = \mathbf{v}_u^t + \eta l^u \sigma(-l^u \mathbf{v}_u^\top \mathbf{h}_c)\mathbf{h}_c$$

- and SGD for $\mathbf{u}_c$ is

$$\mathbf{u}_c^{t+1} = \mathbf{u}_c^t + \eta \frac{1}{|\mathcal{C}_w|} l^u \sigma(-l^u \mathbf{v}_u^\top \mathbf{h}_c)\mathbf{v}_u$$

Summary of Negative Sampling

- NCE is an effective way of learning parameters for an arbitrary locally normalized language model
- Negative sampling should be thought of as an alternative task for generating representations of words for use in other tasks
  - It is not a method for learning parameters in a generative model of language
- If your goal is language modeling, you should use NCE
- If your goal is word representation learning, you should consider both NCE and negative sampling

Overview

Word Vector Evaluations

See `http://wordvectors.org` for a suite of examples.

- Several popular methods for *intrinsic* evaluations:
  - Do (cosine) similarities of pairs of words' vectors correlate with judgments of similarity by humans?
  - TOEFL-like synonym tests, e.g., rug $\rightarrow$ {sofa, ottoman, carpet, hallway}
  - Syntactic analogies, e.g., "walking is to walked as eating is to what?" Solved via:

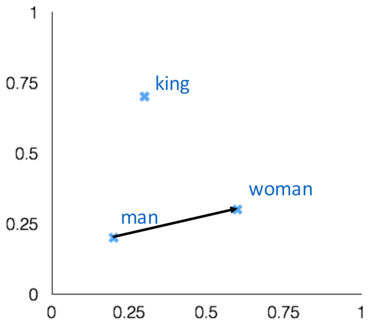
$$\min_{\mathbf{v}\in\mathcal{V}} \cos(\mathbf{v}_v, \mathbf{v}_{walking} - \mathbf{v}_{walked} + \mathbf{v}_{eating})$$

- Also: *extrinsic* evaluations on NLP tasks that can use word vectors (e.g., sentiment analysis)

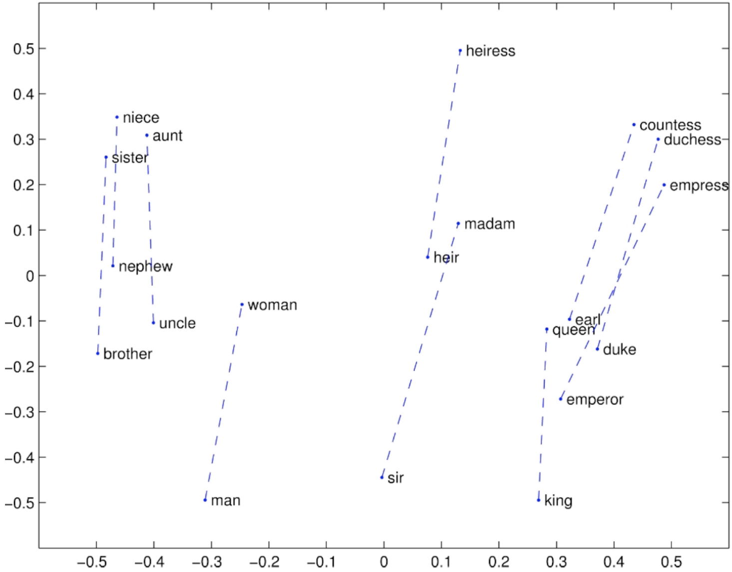
Word Analogy

- Evaluate word vectors by how well their cosine distance after addition captures intuitive semantic and syntactic analogy questions
- Discarding the input words from the search!
- Problem: What if the information is there but not linear?

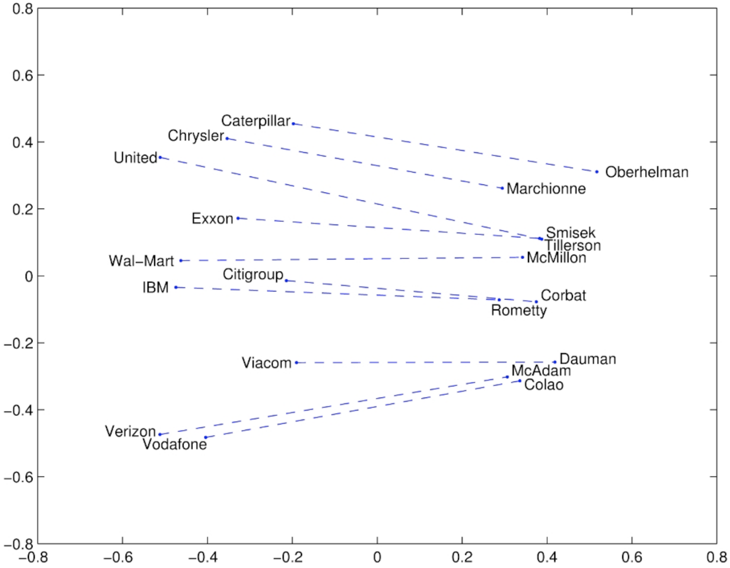
$$\min_{\mathbf{v} \in \mathcal{V}} \cos(\mathbf{v}_v, \mathbf{v}_{man} - \mathbf{v}_{womon} + \mathbf{v}_{king})$$



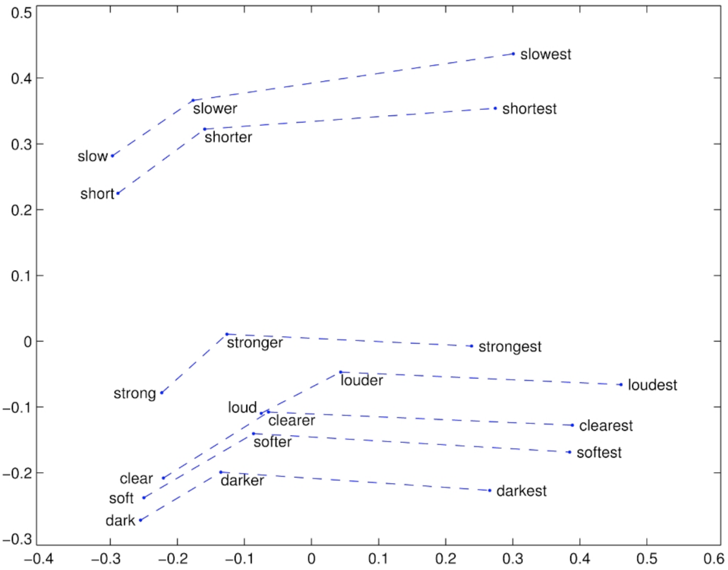
Word Analogy: Glove (Pennington et al. (2014)



Glove Visualizations: Company - CEO

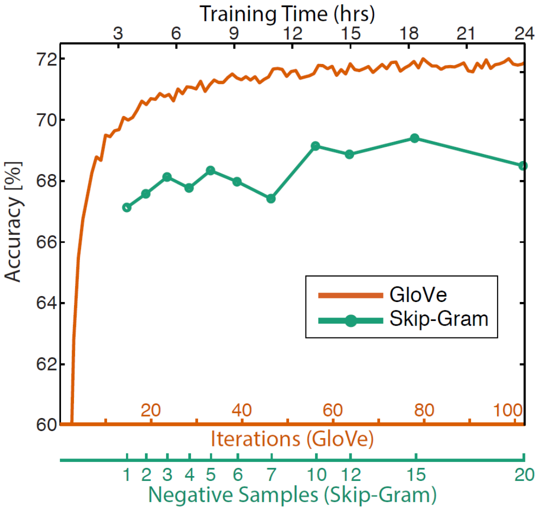


Glove Visualizations: Superlatives



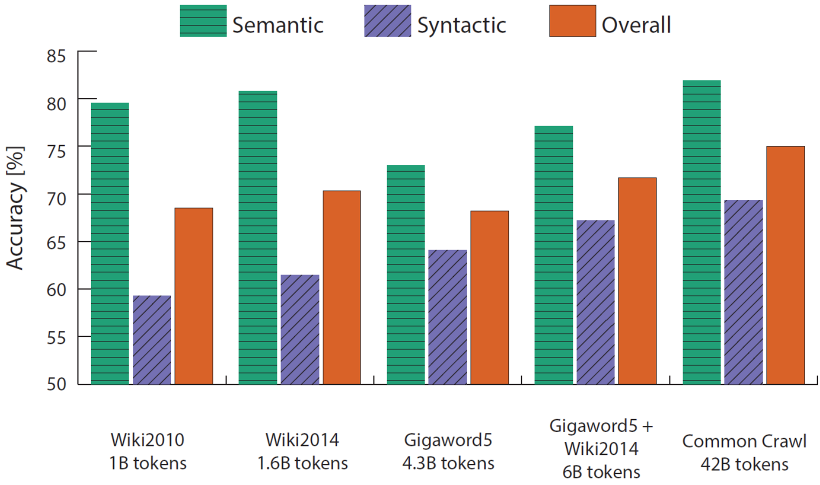
Analogy Evaluation and Hyperparameters

- More training time helps



Analogy Evaluation and Hyperparameters

- More data helps, Wikipedia is better than news text!



Arguments from Yoav GoldBurg

http://u.cs.biu.ac.il/~yogo/cvsc2015.pdf

- Nothing magical about embeddings
- It is just the same old distributional word similarity in a shiny new dress
- "But word2vec is still better, isn't it?"
  - Plenty of evidence that word2vec outperforms traditional methods (In particular: "Don't count, predict!" (Baroni et al. (2014))
  - How does this fit with our story?
- The Big Impact of "Small" Hyperparameters
  - word2vec is more than just an algorithm
  - Introduces many engineering tweaks and hyperpararameter settings
    - May seem minor, but make a big difference in practice
    - Their impact is often more significant than the embedding algorithm's
  - These modifications can be ported to distributional methods

Arguments from Yoav GoldBurg

http://u.cs.biu.ac.il/~yogo/cvsc2015.pdf

- There is no single downstream task
  - Different tasks require different kinds of similarity
  - Different vector-inducing algorithms produce different similarity functions
  - No single representation for all tasks
- "but my algorithm works great for all these different word-similarity datasets! doesn't it mean something?"
  - Sure it does
  - It means these datasets are not diverse enough
  - They should have been a single dataset
  - (alternatively: our evaluation metrics are not discriminating enough)

Further Reading

- Potts (2013). Distributional approaches to word meanings. Ling 236/Psych 236c: Representations of meaning, Spring 2013
- Goldberg and Levy (2014). word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method
- Dyer (2014). Notes on Noise Contrastive Estimation and Negative Sampling.

References I

Baroni, M., Dinu, G., and Kruszewski, G. (2014). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*, pages 238–247. The Association for Computer Linguistics.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Dyer, C. (2014). Notes on noise contrastive estimation and negative sampling. *CoRR*, abs/1410.8251.

Firth, J. R. (1957). *A synopsis of linguistic theory 1930-55.*, volume 1952-59, pages 1–32. The Philological Society, Oxford.

Goldberg, Y. and Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722.

Goodman, J. (2001). Classes for fast maximum entropy training. In *ICASSP*, pages 561–564.

References II

Gutmann, M. and Hyvärinen, A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361.

Harris, Z. (1954). Distributional structure. *Word*, 10(23):146–162.

Hinton, G. E. (1986). Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 1–12.

Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *NIPS*, pages 2177–2185.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *ICLR*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Mnih, A. and Hinton, G. E. (2008). A scalable hierarchical distributed language model. In *NIPS*, pages 1081–1088.

Mnih, A. and Teh, Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. In *ICML*. icml.cc / Omnipress.

References III

Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *AISTATS*. Society for Artificial Intelligence and Statistics.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543. ACL.

Potts, C. (2013). Distributional approaches to word meanings, ling 236/psych 236c: Representations of meaning, spring 2013. Technical report, Stanford University.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.