# Learning distributed word representations

Elad Hoffer

NLP Class
January 2016

# Outline

# Introduction

- A very successful approach in Machine Learning is that of *representation learning* - learning a transformation of raw data input to a representation that can be effectively exploited.
- A popular set of models for distributed representation learning are *Neural networks* - this has culminated in current *Deep Learning* trend
- In language domain, a notable recent example of representation learning was brought forth in the *Word2Vec* framework - *"Distributed Representations of Words and Phrases and their Compositionality"* by Mikolov et al.

# Motivation

Why should we be interested in learning a *distributed* word-level representations?

- Reducing representation dimensionality - replacing *one-hot* and *BOW* which have a vocabulary-sized dimensionality
- Achieving a metric that corresponds to semantic relations - e.g word similarity as embedded distance (similar words are closer in space)
- Providing better word-level features for subsequent tasks

# Learning objective

What should be our objective for learning word representation?

Our guide - the "Distributional hypothesis" (Harris 54', Firth 57'):

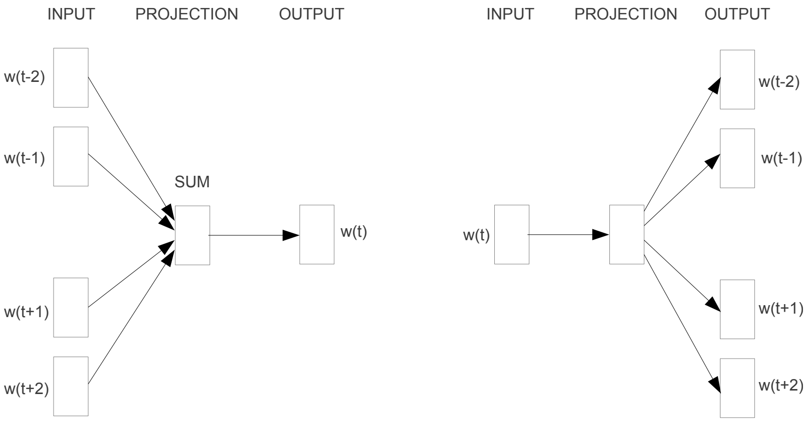
"A word is characterized by the company it keep"

We will cover two models suggested by Mikolov:

- Continuous bag-of-words (CBOW)
- Continuous Skip-gram

# Outline

# CBOW and Skip-gram models




**CBOW** **Skip-gram**

# CBOW and Skip-gram models

The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

- They are both a single-layer neural networks that can be efficiently trained using SGD
- These are essentially *Log-linear* models (using a SoftMax layer will provide us with a probability measure)

# Outline

# Training word representation

- The training objective of the Skip-gram model is to find word representations that are useful for predicting the surrounding words in a sentence or a document.
- More formally, given a sequence of training words $w_1, w_2, w_3, \ldots, w_T$, the objective of the Skip-gram model is to maximize the average log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t) = \frac{1}{T}\sum_{t=1}^{T}\log p(Context|w_t) \tag{1}$$

where $c$ is the size of the training context (which can be a function of the center word $w_t$).

# Training word representation

- The basic Skip-gram formulation defines $p(w_{t+j}|w_t)$ using the softmax function:

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

  where $v_w$ and $v'_w$ are the "input" and "output" vector representations of $w$, and $W$ is the number of words in the vocabulary.
- This formulation is impractical because the cost of computing $\nabla \log p(w_O|w_I)$ is proportional to $W$, which is often large ($10^5$–$10^7$ terms).

# Outline

# Hierarchical SoftMax

A computationally efficient approximation of the full softmax is the hierarchical softmax.

- Uses a binary tree representation of the output layer with the $W$ words as its leaves and, for each node, explicitly represents the relative probabilities of its child nodes.
- Instead of evaluating $W$ output nodes, it is needed to evaluate only about $\log_2(W)$.
- Accuracy is very dependent on the choice of tree structure formation (Huffman, frequency)

# Hierarchical SoftMax

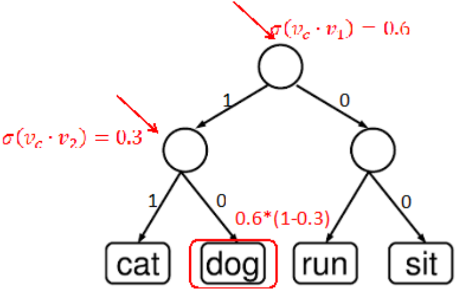



Figure: Hierarchical SoftMax

# Noise Contrastive Estimation

An alternative to the hierarchical softmax is NCE - Noise Contrastive Estimation (Gutmann 10', Mnih 13').

- Assume you want to distinguish words coming from true data according to context $P_d(w \mid c)$, from words sampled from noise distribution $P_n(w)$ (e.g unigram distribution)
- Noise samples are $k$ more probable than true data samples

$$p(d, w \mid c) = \begin{cases} \frac{k}{1+k} \times P_n(w) & \text{if } d = 0 \\ \frac{1}{1+k} \times P_d(w \mid c) & \text{if } d = 1 \end{cases}.$$

- So that the conditional probability of $d$ having observed $w$ and $c$:

$$p(D = 1 \mid c, w) = \frac{P_d(w \mid c)}{P_d(w \mid c) + k \times P_n(w)}.$$

# Noise Contrastive Estimation

- We can now write the conditional likelihood in terms of a $\theta$ parameterized model:

$$p(D = 1 \mid c, w) = \frac{s_\theta(w, c)}{s_\theta(w, c) + k \times P_n(w)}.$$

- This is a binary classification problem with parameters $\theta$ that can be trained to maximize conditional log-likelihood of $\mathcal{D}$, with $k$ negative samples chosen:

$$\mathcal{L}_{\mathrm{NCE}_k} = \sum_{(w,c)\in\mathcal{D}} \left(\log p(D = 1 \mid c, w) + k\mathbb{E}_{\overline{w}\sim P_n} \log p(D = 0 \mid c, \overline{w})\right)$$

- The expectation over noise can be approximated by:

$$\frac{1}{k} \sum_{i=1, \overline{w}\sim P_n}^{k} \log p(D = 0 \mid c, \overline{w})$$

# Negative-Sampling

While NCE can be shown to approximately maximize the log probability of the softmax, the Skip-gram model is only concerned with learning high-quality vector representations.

- We are free to simplify NCE as long as the vector representations retain their quality.
- This leads to "Negative sampling" (NEG), with the objective

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[ \log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right] \qquad (2)$$

  where $\sigma(x)$ is the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$
- Noise distribution is heuristically chosen to be $P(w)_{Unigram}^{3/4}$

# Outline

# Subsampling of Frequent Words

- In very large corpora, the most frequent words can easily occur hundreds of millions of times (e.g., “in”, “the”, and “a”). Such words usually provide less information value than the rare words.
- To counter the imbalance between the rare and frequent words, each word $w_i$ in the training set is discarded with probability computed by the formula

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \tag{3}$$

  where $f(w_i)$ is the frequency of word $w_i$ and $t$ is a chosen threshold, typically around $10^{-5}$.
- This aggressively subsamples words whose frequency is greater than $t$ while preserving the ranking of the frequencies.

# Outline

# Additive Compositionality

Somewhat surprisingly, word vectors capture many linguistic regularities as linear translations.

- For example, the result of a vector calculation vec(“king”) - vec(“man”) + vec(“woman”) is closer to vec(“queen”) than to any other word

| Czech + currency | Vietnam + capital | German + airlines | Russian + river | French + actress |
|---|---|---|---|---|
| koruna | Hanoi | airline Lufthansa | Moscow | Juliette Binoche |
| Check crown | Ho Chi Minh City | carrier Lufthansa | Volga River | Vanessa Paradis |
| Polish zolty | Viet Nam | flag carrier Lufthansa | upriver | Charlotte Gainsbourg |
| CTK | Vietnamese | Lufthansa | Russia | Cecile De |

Table: Vector compositionality using element-wise addition.

# Word vectors algebra

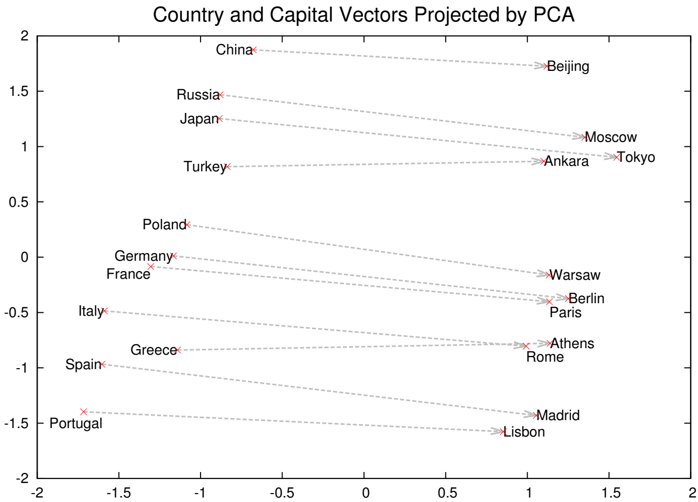



Figure: Word Representation

# Additive Compositionality

The additive property of the vectors can be explained by inspecting the training objective:

- The word vectors are in a linear relationship with the inputs to the softmax nonlinearity.
- As the word vectors are trained to predict the surrounding words in the sentence, the vectors can be seen as representing the distribution of the context in which a word appears.
- These values are related logarithmically to the probabilities computed by the output layer, so the sum of two word vectors is related to the product of the two context distributions.

# Outline

# Analogical reasoning task

The analogical reasoning task consists of analogies such as “Germany” : “Berlin” :: “France” : ?,

- Solved by finding a vector **x** such that vec(**x**) is closest to vec(“Berlin”) - vec(“Germany”) + vec(“France”).
- Correctly answered if **x** is “Paris”.
- The task has two broad categories: the syntactic analogies (such as “quick” : “quickly” :: “slow” : “slowly”) and the semantic analogies, such as the country to capital city relationship.

| Method | Time [min] | Syntactic [%] | Semantic [%] | Total accuracy [%] |
|---|---|---|---|---|
| NEG-5 | 38 | 63 | 54 | 59 |
| NEG-15 | 97 | 63 | 58 | **61** |
| HS-Huffman | 41 | 53 | 40 | 47 |
| NCE-5 | 38 | 60 | 45 | 53 |
| The following results use $10^{-5}$ subsampling | | | | |
| NEG-5 | 14 | 61 | 58 | 60 |
| NEG-15 | 36 | 61 | 61 | **61** |
| HS-Huffman | 21 | 52 | 59 | 55 |

# Analogical reasoning task for phrases

Examples of the analogical reasoning task for phrases.

| | | | |
|---|---|---|---|
| Newspapers | | | |
| New York | New York Times | Baltimore | Baltimore Sun |
| San Jose | San Jose Mercury News | Cincinnati | Cincinnati Enquirer |
| NHL Teams | | | |
| Boston | Boston Bruins | Montreal | Montreal Canadiens |
| Phoenix | Phoenix Coyotes | Nashville | Nashville Predators |
| NBA Teams | | | |
| Detroit | Detroit Pistons | Toronto | Toronto Raptors |
| Oakland | Golden State Warriors | Memphis | Memphis Grizzlies |
| Airlines | | | |
| Austria | Austrian Airlines | Spain | Spainair |
| Belgium | Brussels Airlines | Greece | Aegean Airlines |
| Company executives | | | |
| Steve Ballmer | Microsoft | Larry Page | Google |
| Samuel J. Palmisano | IBM | Werner Vogels | Amazon |

Table: analogical reasoning task for phrases

# Phrase Skip-Gram Results

| Method | Dimensionality | No subsampling [%] | $10^{-5}$ subsampling [%] |
|---|---|---|---|
| NEG-5 | 300 | 24 | 27 |
| NEG-15 | 300 | 27 | 42 |
| HS-Huffman | 300 | 19 | **47** |

Table: Accuracies of the Skip-gram models on phrase analogy.

| | NEG-15 with $10^{-5}$ subsampling | HS with $10^{-5}$ subsampling |
|---|---|---|
| Vasco de Gama | Lingsugur | Italian explorer |
| Lake Baikal | Great Rift Valley | Aral Sea |
| Alan Bean | Rebbeca Naomi | moonwalker |
| Ionian Sea | Ruegen | Ionian Islands |
| chess master | chess grandmaster | Garry Kasparov |

Table: Examples of the closest entities to the given short phrases, using two different models.

# Comparison to Published Word Representations

Examples of the closest tokens given various well known models and the Skip-gram model trained on phrases using over 30 billion training words. An empty cell means that the word was not in the vocabulary.

| Model (training time) | Redmond | Havel | ninjutsu | graffiti | capitulate |
|---|---|---|---|---|---|
| Collobert (50d) (2 months) | conyers<br>lubbock<br>keene | plauen<br>dzerzhinsky<br>osterreich | reiki<br>kohona<br>karate | cheesecake<br>gossip<br>dioramas | abdicate<br>accede<br>rearm |
| Turian (200d) (few weeks) | McCarthy<br>Alston<br>Cousins | Jewell<br>Arzu<br>Ovitz | -<br>-<br>- | gunfire<br>emotion<br>impunity | -<br>-<br>- |
| Mnih (100d) (7 days) | Podhurst<br>Harlang<br>Agarwal | Pontiff<br>Pinochet<br>Rodionov | -<br>-<br>- | anaesthetics<br>monkeys<br>Jews | Mavericks<br>planning<br>hesitated |
| Skip-Phrase (1000d, 1 day) | Redmond Wash.<br>Redmond Washington<br>Microsoft | Vaclav Havel<br>president Vaclav Havel<br>Velvet Revolution | ninja<br>martial arts<br>swordsmanship | spray paint<br>grafitti<br>taggers | capitulation<br>capitulated<br>capitulating |

# Word Embedding as Implicit Matrix Factorization (Levy, Goldberg 14')

Why is Skip-gram with negative sampling working so well?

- Lets revisit the NEG objective, with $w$ as current word, and $c$ as context

$$\log \sigma({v'_c}^\top v_w) + \sum_{i=1}^{k} \mathbb{E}_{c_i \sim P_n(w)} \left[ \log \sigma(-{v'_c}^\top v_w) \right]$$

- Denoting $\#(w, c)$ as number of occurrences of a pair $w$ and $c$, and $\#c$ is the count of specific context, $P_n(w)$ is the unigram distribution, the loss for a specific pair $(w, c)$ is:

$$\#(w, c) \log \sigma({v'_c}^\top v_w) + k \cdot \#(w) \cdot \frac{\#(c)}{|D|} \left[ \log \sigma(-{v'_c}^\top v_w) \right]$$

# Word Embedding as Implicit Matrix Factorization (Levy, Goldberg 14')

- We can now mark $x = {v'_c}^\top v_w$ and optimize the objective by comparing the partial derivative to zero:

$$\frac{\partial L}{\partial x} = \#(w,c) \cdot \sigma(-x) - k \cdot \#(w) \cdot \frac{\#(c)}{|D|}\sigma(x) = 0$$

- Turns out, that after doing the algebra (see paper), we get that the optimal solution satisfies:

$${v'_c}^\top v_w = \log\left(\frac{\#(w,c) \cdot |D|}{\#(w) \cdot \#(c)} \cdot \frac{1}{k}\right) = \log\left(\frac{\#(w,c) \cdot |D|}{\#(w) \cdot \#(c)}\right) - \frac{1}{k}$$

- This is actually the Pointwise Mutual Information measure with an additional constant

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

# Word Embedding as Implicit Matrix Factorization (Levy, Goldberg 14')

So the word2vec framework is actually doing a matrix factorization of the shifted *PMI*

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

by finding word and context representations $W$ and $C$ so that

$$W_i \cdot C_i = PMI(w_i, c_i)$$

- Levy and Goldberg showed that similar results can be attained by using *SVD* to decompose the shifted PPMI
- word2vec still provides better results, probably due to the weighted nature of the algorithm

# Current research and applications

- The word2vec embeddings have been widely used as preliminary features for a wide variety of language-related tasks such as language models, image captioning and mode
- It has also inspired similar theme research such as "Skip-Thought Vectors" (Kirks et al. 15')

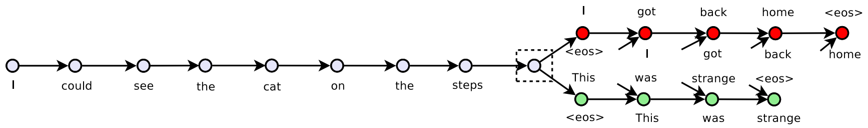



Figure: Skip-thought objective

## Summary

In this work we've seen

- The word2vec work has demonstrated how learning distributed representation of word can provide superior embeddings applicable to language tasks
- It requires the use of carefully chosen techniques and heuristics. Most notable are likelihood function approximations - the hierarchical softmax, noise-contrastive estimation and negative-sampling
- Its success can partially be attributed to the classical NLP objective of decomposing the *PMI* measure