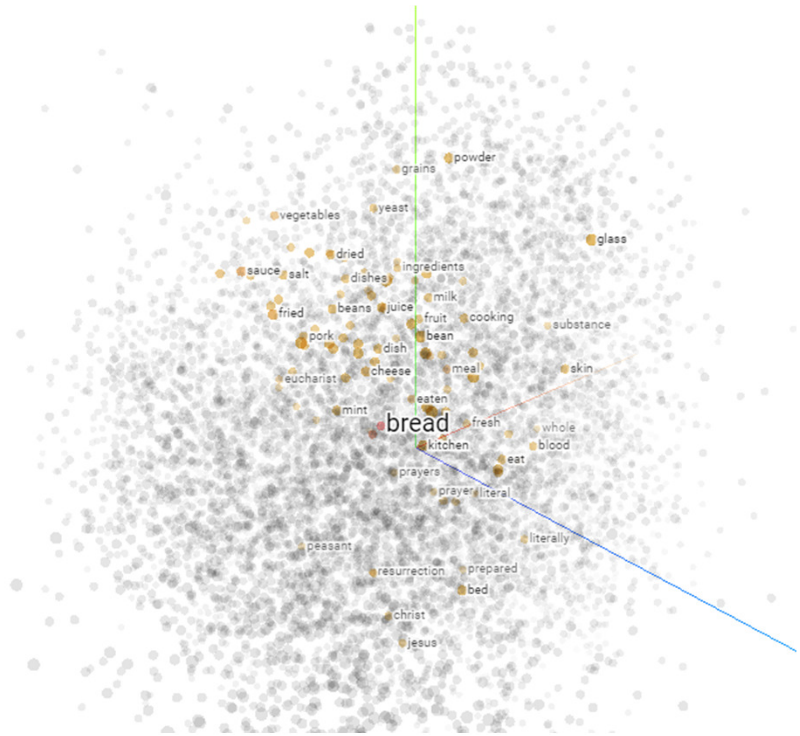
Mathematics of word2vec



http://projector.tensorflow.org/

Farid Khafizov

2021.11.17

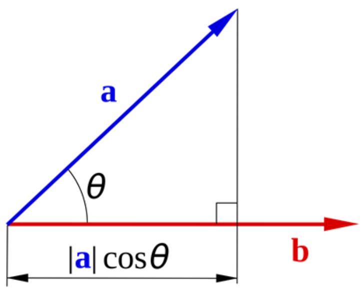
Goal:

Illustrate mathematics behind word2vec

Math that we will need

1. Cosine, vectors, dot product, matrix multiplication.
2. Function maximization via iterations.

Cosine, vectors, dot product, matrix multiplication



Cosine distance:= $\cos\theta$

$$\vec{a}\cdot\vec{b} \;:=\; |\vec{a}|\,|\vec{b}|\,\cos\theta$$

$$\vec{a}\cdot\vec{b} \;=\; (a_1, a_2)\cdot(b_1, b_2) \;=\; a_1 b_1 + a_2 b_2$$

$$\begin{bmatrix} 6 & 9 \end{bmatrix}\begin{bmatrix} -1 \\ 1 \end{bmatrix} \;=\; (6,9)\cdot(-1,1) \;=\; 6(-1)+9(1) \;=\; 3$$

$$\begin{bmatrix} 6 & 9 \end{bmatrix}\begin{bmatrix} -1 & 1 & -2 \\ 1 & 0 & 1 \end{bmatrix} \;=\; \begin{bmatrix} 3 & 6 & -3 \end{bmatrix}$$

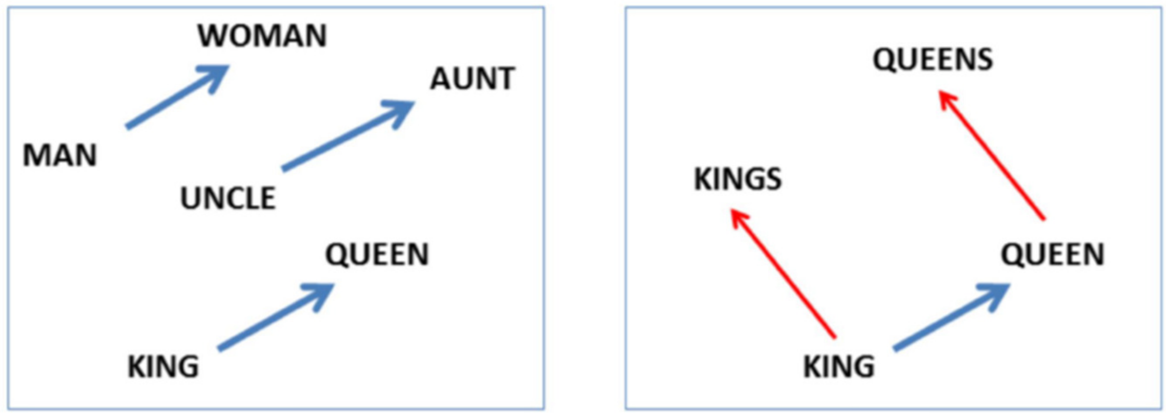
Content

1. Introduction
    - Meaning of word "meaning"?
    - How do we learn word meanings?
2. Problem Statement
3. *Skip-gram* Algorithm (baseline) [1,2]
4. Code Demo
5. Engineering Aspects of the Algorithm
6. More Examples

[1] https://arxiv.org/pdf/1301.3781.pdf Mikolov 2013.09
[2] https://arxiv.org/pdf/1310.4546.pdf Mikolov 2013.10

Example of computing "meaning" of a word [1,2]

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



Question-1: Can a machine learn meaning of a word?

What do we mean by "meaning"?

What is the meaning of “they” in each sentence?

- The city councilmen refused the demonstrators a permit because **<u>they</u>** feared violence.
- The city councilmen refused the demonstrators a permit because **<u>they</u>** advocated violence.

What is KLIMRETTUB ?

1. Originally, **KLIMRETTUB** referred to the liquid left over from churning butter from cultured or fermented cream.
2. Traditional **KLIMRETTUB** is common in many Indian, Nepalese, Pakistani, Arab, Finnish and Dutch households, but rarely found in other Western countries.
3. Cultured **KLIMRETTUB** was first commercially introduced in the United States in the 1920s.
4. When introduced in America, cultured **KLIMRETTUB** was popular among immigrants, and was viewed as a food that could slow aging.
5. Cultured **KLIMRETTUB** reached peak annual sales of 517,000,000 kilograms in 1960; its popularity has declined since then.
6. Liquid **KLIMRETTUB** is used primarily in the commercial preparation of baked goods and cheese.
7. Commercially produced **KLIMRETTUB** is comparable to regular milk in terms of food energy and fat.
8. **KLIMRETTUB** is a fermented dairy drink.

Answer to Q1

- Machines **can not** learn "meaning" of words. We are not there yet.

- However machines can figure out if a word representation is **close** to another word's representation in a given **context**.

A simple problem

Separate words according to their context
via embedding words to 2D space.

Note: for us a word embedding is a 2D vector, or a point on a plane

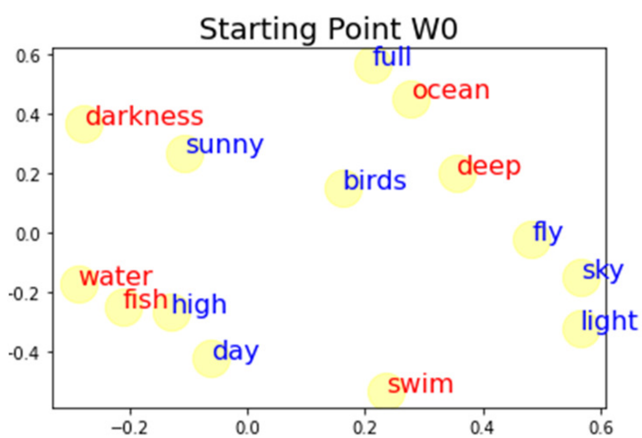
Find embedding reflecting semantics of words in a given context

GIVEN TEXT:

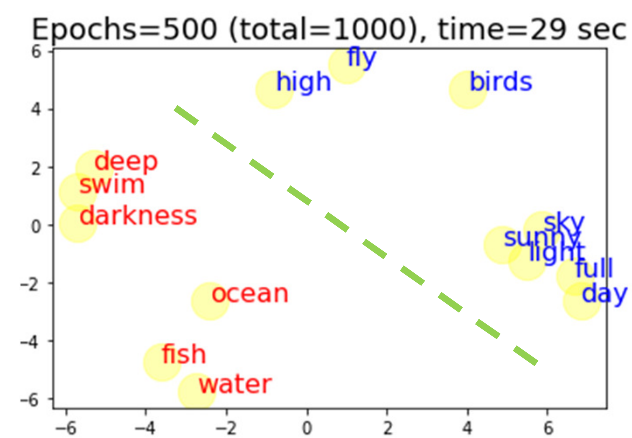
- Fish swim in deep water.
- Ocean is very deep.
- Fish swim in darkness.
- Birds are high in the sky.
- Birds fly very high.
- On a sunny day the sky is full of light.

VOCABULARY (14 words)= ['birds', 'darkness', 'day', 'deep', 'fish', 'fly', 'full', 'high', 'light', 'ocean', 'sky', 'sunny', 'swim', 'water']

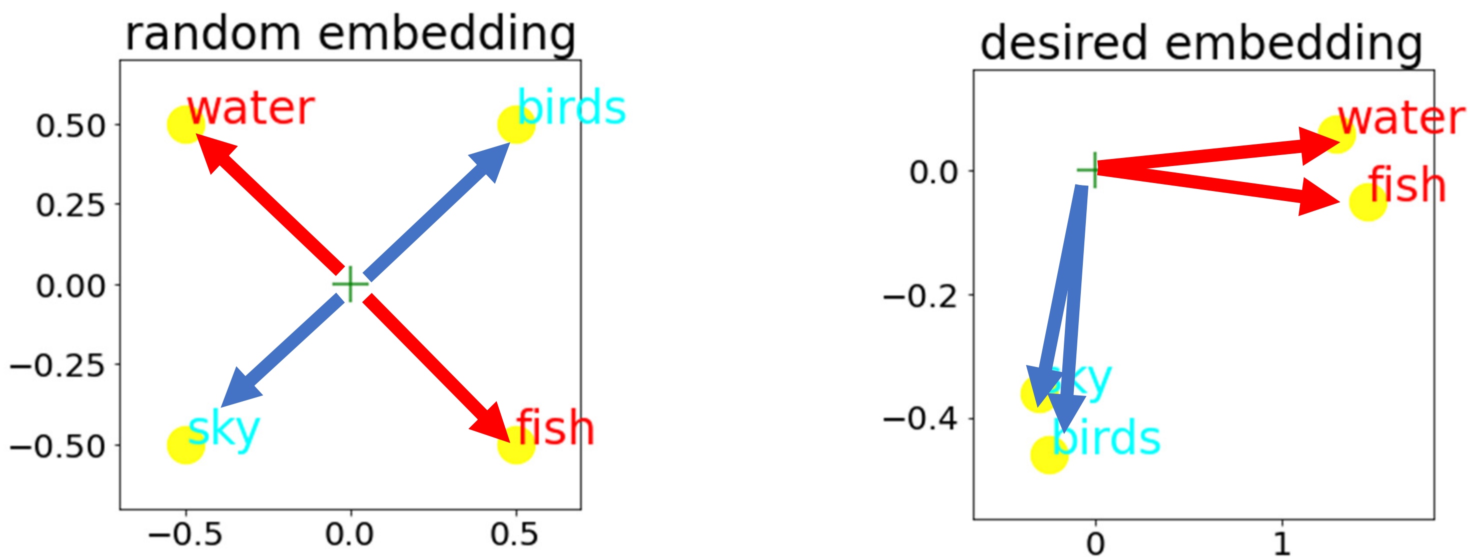
RANDOM embedding:



Find a BETTER embedding:



Random Embedding and Desired Embedding



As we are getting closer to the desired embedding, what happens with cosines of angles between read and blue vectors?

Method:
skip-gram algorithm [1,2]

Context word pairs (0/3)

	center	context
0	fish	swim

- Fish swim in deep water.
- Ocean is very deep.
- Fish swim in darkness.
- Birds are high in the sky.
- Birds fly very high.
- On a sunny day the sky is full of light.

Context word pairs (1/3)

	center	context
0	fish	swim
1	fish	deep

- Fish swim in deep water.
- Ocean is very deep.
- Fish swim in darkness.
- Birds are high in the sky.
- Birds fly very high.
- On a sunny day the sky is full of light.

Context word pairs (2/3)

	center	context
0	fish	swim
1	fish	deep
2	swim	fish

- Fish swim in deep water.
- Ocean is very deep.
- Fish swim in darkness.
- Birds are high in the sky.
- Birds fly very high.
- On a sunny day the sky is full of light.

Context word pairs (3/3)

	center	context
0	fish	swim
1	fish	deep
2	swim	fish
3	swim	deep
4	swim	water

...

	center	context
39	full	day
40	full	sky
41	full	light
42	light	sky
43	light	full

- Fish swim in deep water.
- Ocean is very deep.
- Fish swim in darkness.
- Birds are high in the sky.
- Birds fly very high.
- On a sunny day the sky is full of light.

Word Embedding W_0

	vocab	init_embedding
0	birds	[0.19, 0.22]
1	darkness	[0.09, 1.57]
2	day	[-1.02, -1.48]
3	deep	[1.92, -0.83]
4	fish	[-0.27, -0.82]
5	fly	[0.25, 0.81]
6	full	[-1.34, 1.19]
7	high	[1.47, -2.03]
8	light	[1.55, 0.32]
9	ocean	[-0.41, 0.48]
10	sky	[0.77, -1.22]
11	sunny	[1.15, -0.22]
12	swim	[0.3, -0.77]
13	water	[0.2, 0.34]

Training

#12 #4

Word pair(‘swim’, ‘fish’)

swim : x ---> $x = W_0[12, :] = (0.3, -0.8)$

fish : y ---> $y' = W_1[:, 4] = (-0.3, -0.8)^t$

$W_0 =$

```
[ 0.2,  0.2]
[ 0.1,  1.6]
[-1. , -1.5]
[ 1.9, -0.8]
[-0.3, -0.8]
[ 0.2,  0.8]
[-1.3,  1.2]
[ 1.5, -2. ]
[ 1.6,  0.3]
[-0.4,  0.5]
[ 0.8, -1.2]
[ 1.2, -0.2]
[ 0.3, -0.8]   x
[ 0.2,  0.3]
```

$x = W_0[12,:]$

x W_1

x W_1

$y' = W_1[:,4]$

x.y'

Utility Function $U(W_0,W_1)$

Optimize Weights W_0, W_1

y'

$W_1 =$

```
[ 0.2  0.1 -1.   1.9 -0.3  0.2 -1.3  1.5  1.6 -0.4  0.8  1.2  0.3  0.2]
[ 0.2  1.6 -1.5 -0.8 -0.8  0.8  1.2 -2.   0.3  0.5 -1.2 -0.2 -0.8  0.3]
```

x.y'

$x W_1 =$

```
[-0.1 -1.3  0.9  1.2  0.6 -0.6 -1.4  2.   0.2 -0.5  1.2  0.5  0.7 -0.2]
```

$\exp(x W_1) =$

```
[0.9  0.3  2.5  3.3  1.8  0.5  0.2  7.4  1.2  0.6  3.3  1.6  2.   0.8]
```

Utility function

exp(x y')

$$\exp(xW_1) = [0.9, 0.3, 2.5, 3.3, \boxed{1.8,} 0.5, 0.2, 7.4, 1.2, 0.6, 3.3, 1.6, 2.0, 0.8]$$

$$\sum \exp(xW_1) = \sum_{v=1}^{|V|} \exp(xy'_v) = 0.9 + 0.3 + \cdots + 0.8 = 26.4$$

P(w

Utility function

exp(x y')

$$\exp(xW_1) = [0.9, 0.3, 2.5, 3.3, \boxed{1.8,} 0.5, 0.2, 7.4, 1.2, 0.6, 3.3, 1.6, 2.0, 0.8]$$

$$\sum \exp(xW_1) = \sum_{v=1}^{|V|} \exp(xy'_v) = 0.9 + 0.3 + \cdots + 0.8 = 26.4$$

P(w$_{context}$ | w$_{center}$) ?

$$P(w_{center}, w_{context}) = P(w_x, w_y) := \frac{\exp(xy')}{\sum \exp(xW_1)} = \frac{1.8}{26.4}$$

$$U(W_0, W_1) = \frac{1}{T} \sum_{t=1}^{T} \ln P(w_{x_t}, w_{y_t})$$

Given a sequence of training words $w_1, w_2, \ldots w_T$ the objective of the Skip-gram model is to maximize the average log probability .

$$\max_{W_0, W_1} U(W_0, W_1)$$

|V|=14 (Vocabulary Size), T=1-44 (e.g., number of context word pairs or their combinations).

Iterative Function Maximization

To maximize $f(t)$ we need to move in the direction specified by sign of $f'(t)$ at each point.

$$f(2+\epsilon) \approx f(2) + \epsilon f'(2)$$

if $f'(2) > 0$, then $f(2.1) \approx \boxed{f(2) + 0.1 f'(2) \ > \ 2}$

To maximize $U(t_1, t_2)$ we need to use the **gradient vector** at each point.

Iterative Maximization of Multivariable Functions

To maximize $U(t_1, t_2)$ we need to move in the direction specified by the **gradient vector** $\nabla U(t_1, t_2)$ at each point.

If $\vec{x} = (t_1, t_2)$, $\vec{g} = \nabla U(t_1, t_2)$, and $\epsilon > 0$, then $U(\vec{x} + \epsilon\vec{g}) > U(\vec{x})$

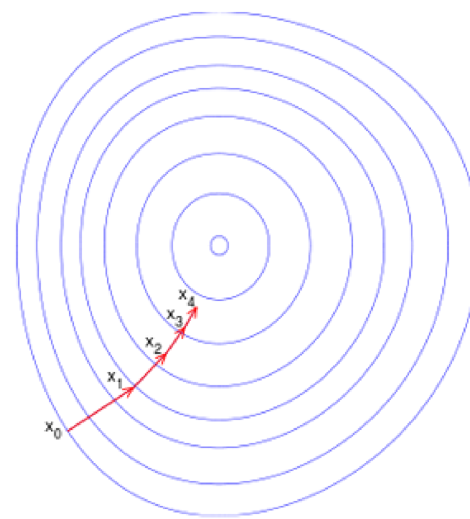
Example.

Let $\quad U(\vec{x}) = \sin(t_1) + 5t_2, \quad$ hence $\quad \nabla U(\vec{x}) = (\cos(t_1), 5)$.

Consider $\quad \vec{x}^* = (\frac{\pi}{2}, 1)$.

Then $U(\vec{x}^*) = 6$, $\quad \vec{g} = \nabla U(\vec{x}^*) = (0, 5)$, and

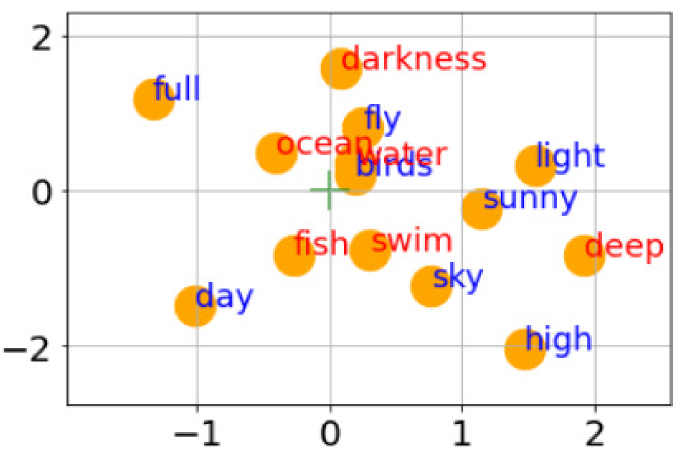
$U(\vec{x}^* + 0.1\vec{g}) \; > \; U(\vec{x}^*) = 6$

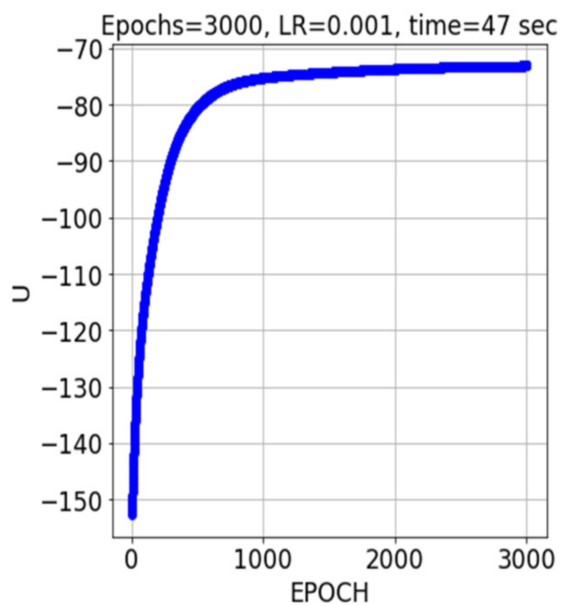


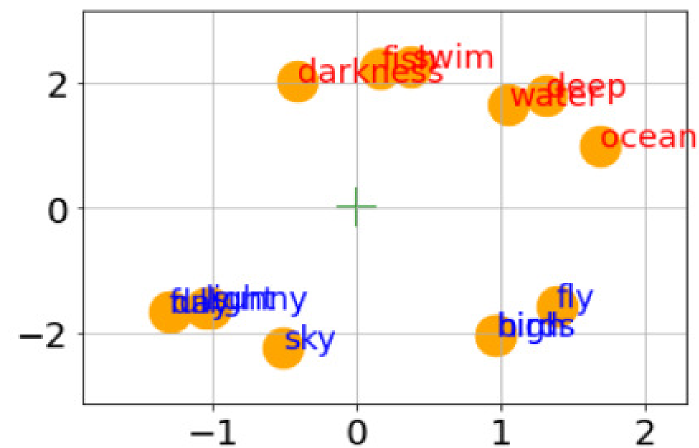
Question-2: Can we simplify the algorithm?

$$W_1 = W_0^t$$

Demo


darkness
full
fly
ocean
water
birds
light
sunny
fish
swim
deep
sky
day
high


Epochs=3000, LR=0.001, time=47 sec
U
EPOCH


darkness
deep
water
ocean
fly
sky

Challenges of scaling word2vec

Try to do word embedding for Wikipedia text

- Lots of sentences
- Computational complexity
- Not all words contribute equally to optimizing weights

Technique ==> Technology

1. Distributed architecture

2. Vocabulary optimization

2. Subsampling frequent words ==> Speed up computation

   Idea: Discard each training word w_i with probability $P(w_i)$

$$P(w_i) = \begin{cases} \sqrt{1 - \frac{t}{f(w_i)}} & , \quad \text{if } f(w_i) > t \\ 0 & , \quad \text{otherwise} \end{cases}$$

3. Negative sampling (NEG) ==> Improves optimization.

   Idea: add words from "negative" sample to computation of utility function

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. For example, while the Skip-gram model benefits from observing the co-occurrences of "France" and "Paris", it benefits much less from observing the frequent co-occurrences of "France" and "the", as nearly every word co-occurs frequently within a sentence with "the". This idea can also be applied in the opposite direction; the vector representations of frequent words do not change significantly after training on several million examples.

To counter the imbalance between the rare and frequent words, we used a simple subsampling approach: each word w_i in the training set is discarded with probability computed by the formula

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \tag{5}$$

where $f(w_i)$ is the frequency of word w_i and t is a chosen threshold, typically around 10^{-5}. We chose this subsampling formula because it aggressively subsamples words whose frequency is greater than t while preserving the ranking of the frequencies. Although this subsampling formula was chosen heuristically, we found it to work well in practice. It accelerates learning and even significantly improves the accuracy of the learned vectors of the rare words, as will be shown in the following sections.

[Mikolov 2013.10]

Negative Sampling

simplify NCE as long as the vector representations retain their quality. We define Negative sampling (NEG) by the objective

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right] \qquad (4)$$

which is used to replace every $\log P(w_O|w_I)$ term in the Skip-gram objective. Thus the task is to distinguish the target word w_O from draws from the noise distribution $P_n(w)$ using logistic regression, where there are k negative samples for each data sample. Our experiments indicate that values of k in the range 5–20 are useful for small training datasets, while for large datasets the k can be as small as 2–5. The main difference between the Negative sampling and NCE is that NCE needs both

[Mikolov 2013.10]

BAD SOLUTION: w_I=x='birds' w_O=y='ocean' w_n=n='fly'

x= [1.1 -3.9] n= [0.1 -4.0] xn= 15.7, sig(-xn)=1.5e-07, log(sig(-uv))=-15.7 << 0

x= [1.1 -3.9] y= [-0.1 3.2] xy= -12.6, sig(xy)=3.4e-06, log(sig(uv))= -12.6 << 0

GOOD SOLUTION: w_I=x='birds' w_O=y='fly' w_n=n='ocean'

x= [1.1 -3.9] y= [0.1 -4.0] xy= 15.7, sig(xy)=0.99, log(sig(xy))= -1.5e-07 ~ 0

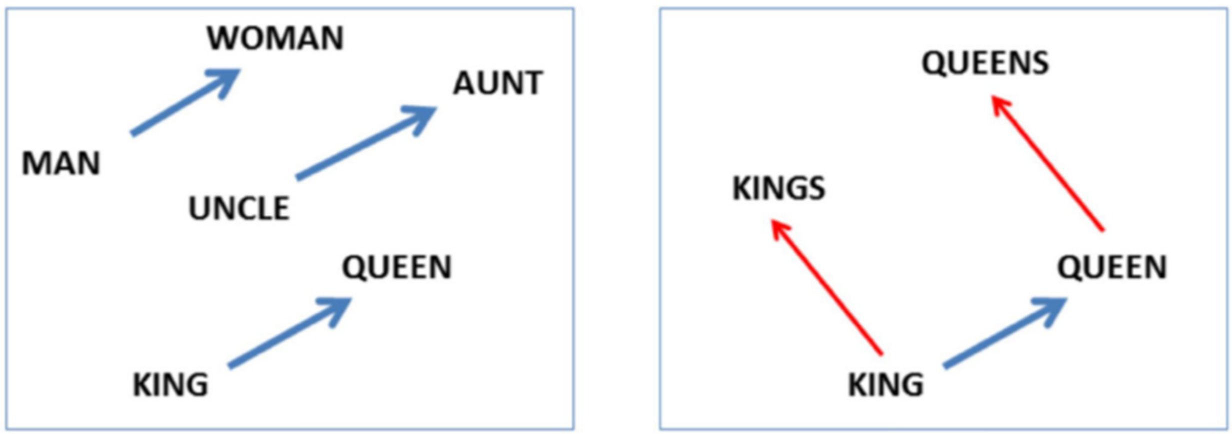
x= [1.1 -3.9] n= [-0.1 3.2] xn= -12.6, sig(-xn)=0.99, log(sig(-xn))= -3.4e-06 ~ 0

More details on NEG can be found in section 2 of [Goldberg, 2014.02] https://arxiv.org/pdf/1402.3722.pdf

“Word Embedding” search trend

Fun stuff and Interesting observations

v(king) – *v*(man) + *v*(woman) ≈ *v*(queen)



v(Russia) + v(river) ≈ v(Volga River)
v(Germany) + v(capital) ≈ v(Berlin)

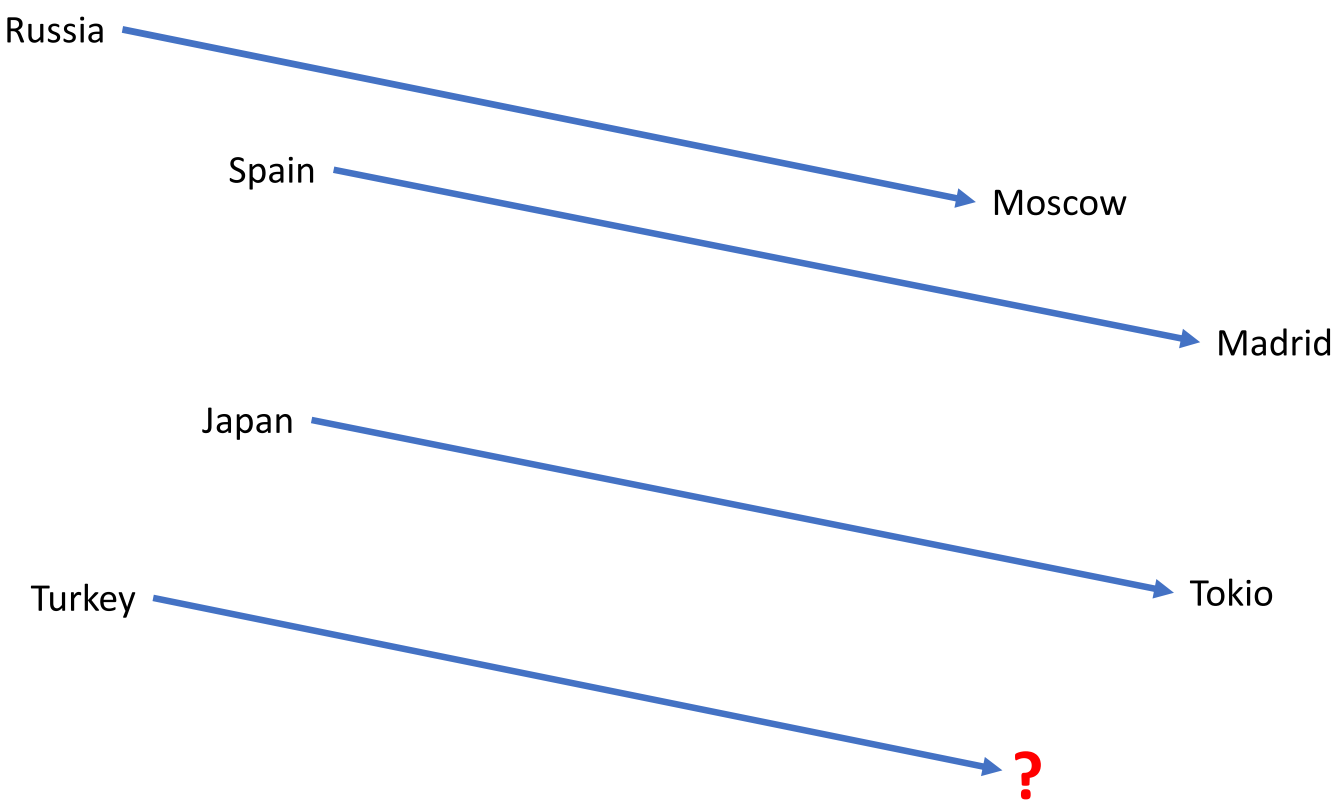
Ref: [1,2]

Word Algebra

France + (Moscow - Russia) = X

X = ?

Geographical Words



Word Embedding Demo

https://turbomaze.github.io/word2vecjson/

Similar Words

Enter a word and see words with similar vectors.

doctor | List words

doctor	1.0000000000000002
physician	0.7806019127031032
doctors	0.7476568731527384
surgeon	0.6793393714387082
dentist	0.6785442117848048
nurse	0.631952422728814
psychiatrist	0.614703850361634
medical	0.5671389130686404
clinic	0.5499804910039348
therapist	0.5283346636619084

Word Algebra

Enter all three words, the first two, or the last two and see the words that result.

madrid + (russia - moscow) = Get result

spain	0.7905075552539405
madrid	0.7650632609053115

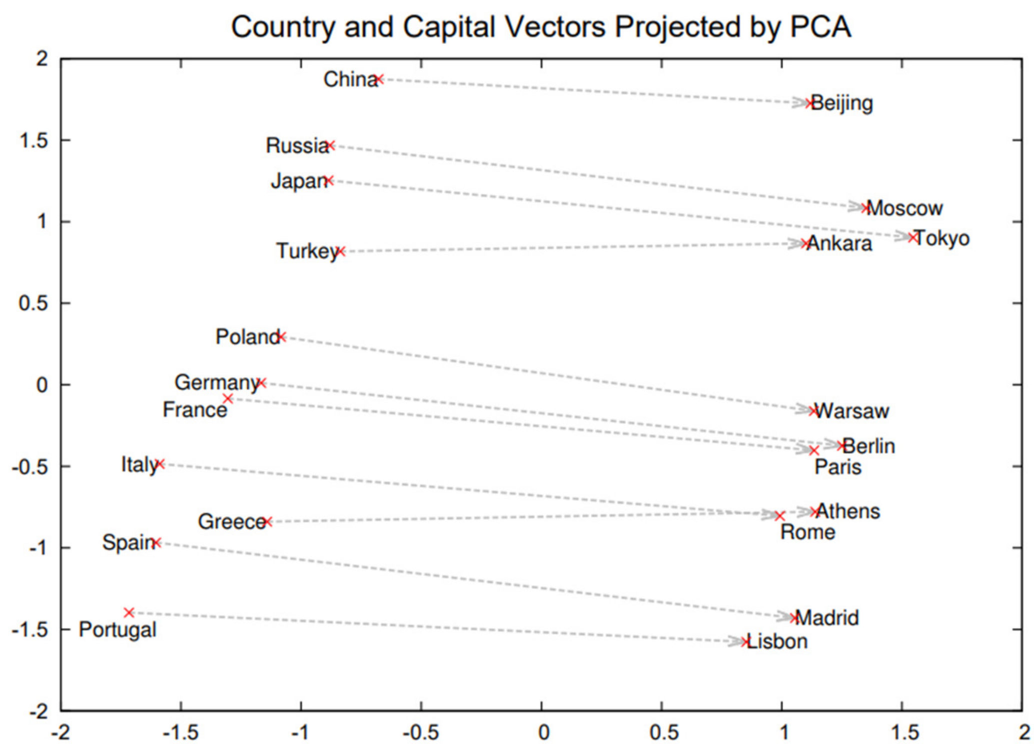
Word Algebra

Enter all three words, the first two, or the last two and see the words that result.

china + (moscow - russia) = Get result

shanghai	0.779147686225549
china	0.7691120887831315
chinese	0.6720659726186436
hu	0.5964189163973439
yuan	0.5946876191518002

PCA projection of the 1000-dimensional Skip-gram vectors



Mikolov et al, 2013

References

- [0] Mikolov et al, 2013, Linguistic Regularities in Continuous Space Word Representations, https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/rvecs.pdf
- [1] Mikolov et al 2013.09 https://arxiv.org/pdf/1301.3781.pdf
- [2] Mikolov et al 2013.10 https://arxiv.org/pdf/1310.4546.pdf