Word to Vec
The Skip-gram Model

2021.02.28

Content

- Problem statement
- Quick demo
- Transforming a technique into a technology

Idea behind the Skip-gram Model

Текст:

- Балерина вдохновляет девочку.
- КИЧЛАМ сильный и ловкий.
- Штангист сильный мужчина.
- Балерина красивая женщина.
- Юниор ловкий штангист.
- Юниор молодой мужчина.
- Девочку учит балерина.

СЛОВАРЬ из 13 слов = ['балерина', 'вдохновляет', 'девочку', 'женщина', 'кичлам', 'красивая', 'ловкий', 'молодой', 'мужчина', 'сильный', 'учит', 'штангист', 'юниор']

Размер окна = 2
42 контекстные пары слов

	input	label	X_train	Y_train
0	балерина	вдохновляет	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
1	балерина	девочку	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
2	вдохновляет	балерина	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
3	вдохновляет	девочку	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4	девочку	балерина	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Постановка задачи:

Сгруппировать слова согласно их семантической близости

Подход:

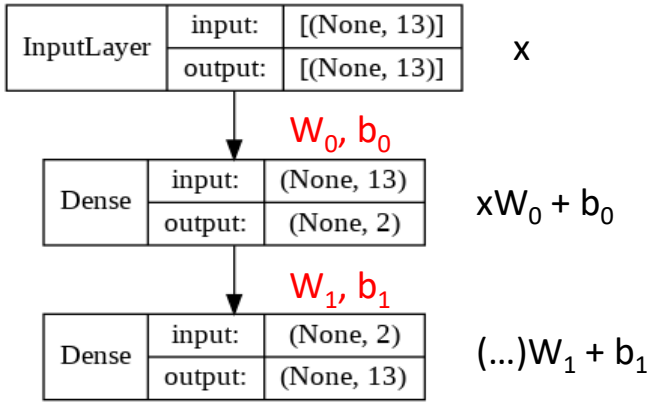
Использовать контекст

“You shall know a word by the company it keeps”
J. R. **Firth**, 1957

How does it work?

	word	x1	x2
0	балерина	-0.6	4.8
1	вдохновляет	0.5	2.2
2	девочку	0.9	3.4
3	женщина	0.6	3.5
4	кичлам	0.1	-2.0
5	красивая	0.8	2.5
6	ловкий	-0.2	-2.2
7	молодой	-1.2	-1.7
8	мужчина	-1.6	-1.6
9	сильный	1.1	-4.7
10	учит	-0.1	3.1
11	штангист	-1.1	-2.2
12	юниор	-0.1	-3.3

W_0 =
```
[[-0.6  4.8]
 [ 0.5  2.2]
 [ 0.9  3.4]
 [ 0.6  3.5]
 [ 0.1 -2. ]
 [ 0.8  2.5]
 [-0.2 -2.2]
 [-1.2 -1.7]
 [-1.6 -1.6]
 [ 1.1 -4.7]
 [-0.1  3.1]
 [-1.1 -2.2]
 [-0.1 -3.3]]
```



W_1 =
```
[[ 2.4  1.   0.6  1.1  1.4  1.1  1.1  0.2  0.6  0.4  1.1  0.9 -0.2]
 [ 0.4  0.4  0.4  0.3 -0.2  0.4 -0.3 -0.1 -0.3 -0.1  0.4 -0.3 -0.1]]
```

b_0= `[-0.3  0.0]`

b_1= `[-1.5 -2.9 -2.3 -2.8 -2.8 -3.0 -2.0 -3.1 -2.2 -2.1 -2.8 -2.1 -2.5]`

балерина: x = [1,0,0,…,0]

вдохновляет: y = [0,1,0,…,0]

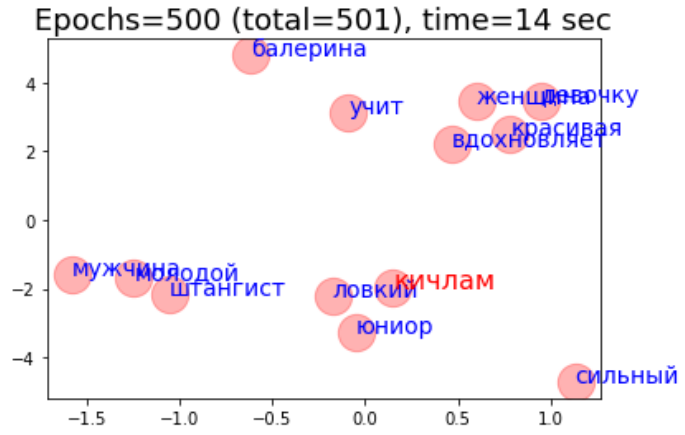
softmax((x*W_0 +b_0) * W_1 + b_1) → y

Наша цель: вычислить W_0 , группирующую слова согласно их семантическому смыслу

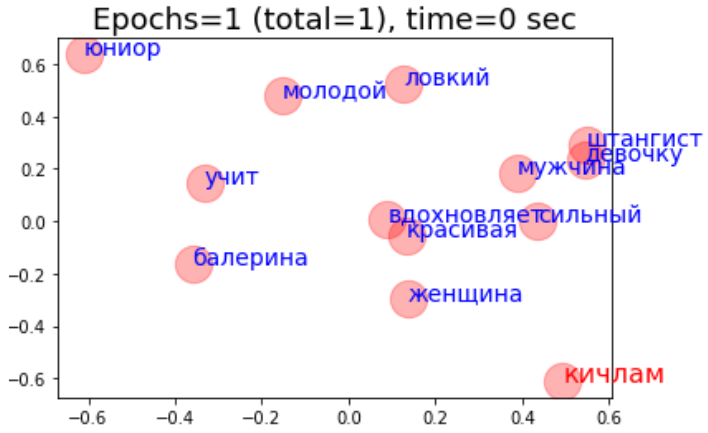
	word	x1	x2
0	балерина	-0.6	4.8
1	вдохновляет	0.5	2.2
2	девочку	0.9	3.4
3	женщина	0.6	3.5
4	кичлам	0.1	-2.0
5	красивая	0.8	2.5
6	ловкий	-0.2	-2.2
7	молодой	-1.2	-1.7
8	мужчина	-1.6	-1.6
9	сильный	1.1	-4.7
10	учит	-0.1	3.1
11	штангист	-1.1	-2.2
12	юниор	-0.1	-3.3

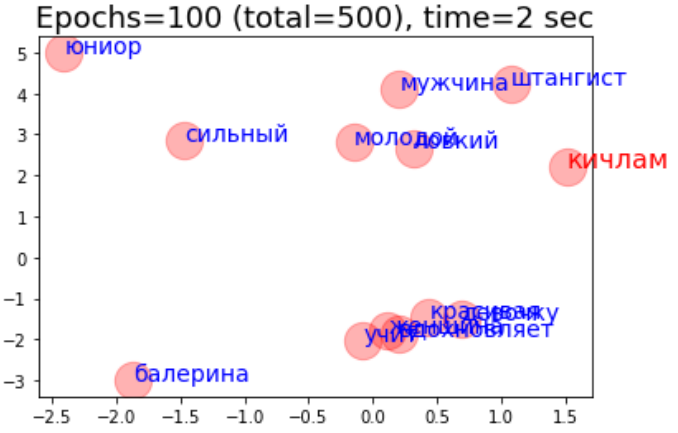
$W_0 =$

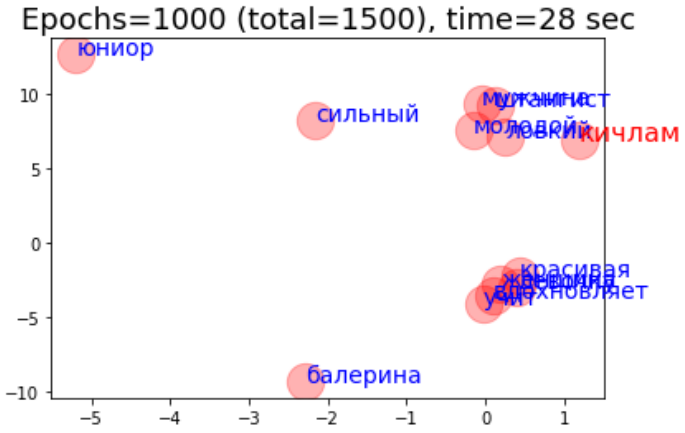
```
[[-0.6  4.8]
 [ 0.5  2.2]
 [ 0.9  3.4]
 [ 0.6  3.5]
 [ 0.1 -2. ]
 [ 0.8  2.5]
 [-0.2 -2.2]
 [-1.2 -1.7]
 [-1.6 -1.6]
 [ 1.1 -4.7]
 [-0.1  3.1]
 [-1.1 -2.2]
 [-0.1 -3.3]]
```

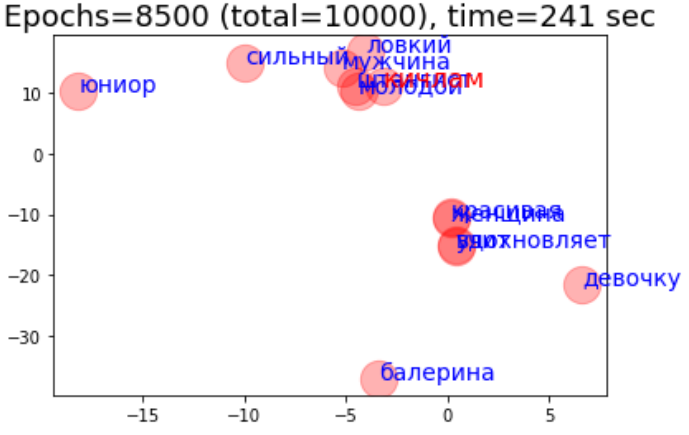


w2v_demo


Epochs=1 (total=1), time=0 sec
юниор
молодой
ловкий
штангист
девочку
учит
мужчина
вдохновляет
сильный
красивая
балерина
женщина
кичлам


Epochs=100 (total=500), time=2 sec
юниор
мужчина
штангист
сильный
молодой
ловкий
кичлам
балерина


Epochs=1000 (total=1500), time=28 sec
юниор
сильный
кичлам
балерина


Epochs=8500 (total=10000), time=241 sec
ловкий
сильный
мужчина
юниор
молодой
девочку
балерина

Transforming a technique into a technology

The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of precise syntactic and semantic word relationships. In this paper we present several extensions that improve both the quality of the vectors and the training speed. By subsampling of the frequent words we obtain significant speedup and also learn more regular word representations. We also describe a simple alternative to the hierarchical softmax called negative sampling.

Distributed Representations of Words and Phrases and their Compositionality
Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean
https://arxiv.org/abs/1310.4546

Utility Function

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

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

where c is the size of the training context (which can be a function of the center word w_t). Larger c results in more training examples and thus can lead to a higher accuracy, at the expense of the training time. The basic Skip-gram formulation defines $p(w_{t+j}|w_t)$ using the softmax function:

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^{\top} v_{w_I}\right)} \tag{2}$$

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).

Negative Sampling

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

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

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

BAD SOLUTION: w_I=x='балерина' w_O=y='штангист' w_n=n='девочку'

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

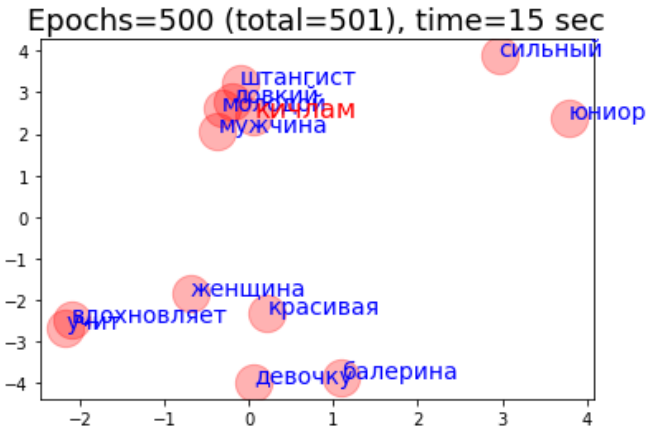
x= [1.1 -3.9] y= [-0.1 3.2] xy= -12.6, sig(xy)=3.4e-06, log(sig(uv))= -12.6 << 0

GOOD SOLUTION: w_I=x='балерина' w_O=y='девочку' w_n=n='штангист'

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

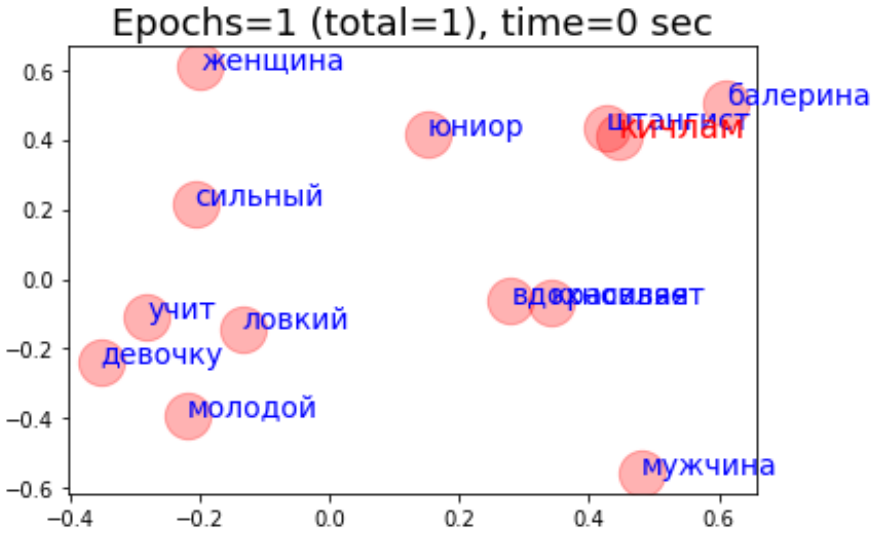
Пример для тестирования NS

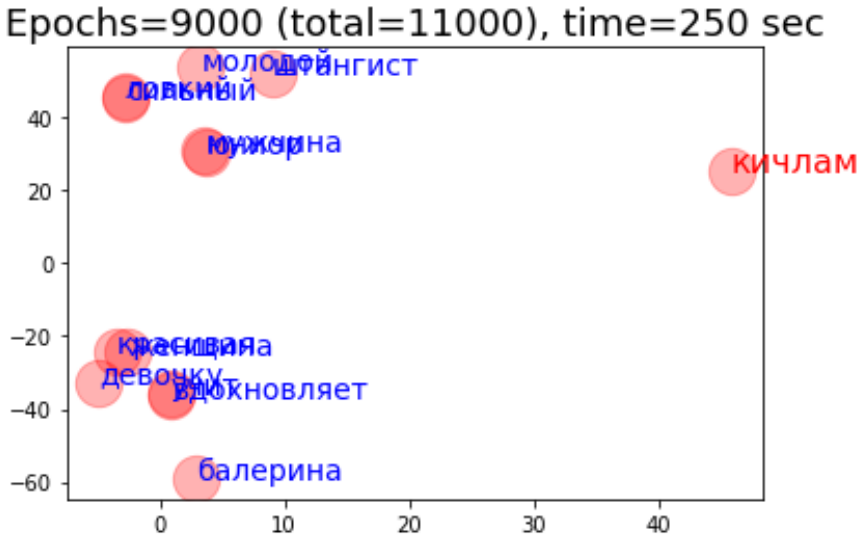


	word	x1	x2
0	балерина	1.1	-3.9
1	вдохновляет	-2.1	-2.5
2	девочку	0.1	-4.0
3	женщина	-0.7	-1.8
4	кичлам	0.1	2.4
5	красивая	0.2	-2.3
6	ловкий	-0.2	2.8
7	молодой	-0.3	2.6
8	мужчина	-0.4	2.1
9	сильный	3.0	3.9
10	учит	-2.2	-2.7
11	штангист	-0.1	3.2
12	юниор	3.8	2.4

```
WEIGHT MATRICES:
(13, 2)
[[ 1.0908686  -3.8614712 ]
 [-2.0932245  -2.4768493 ]
 [ 0.05848308 -3.9837399 ]
 [-0.69571817 -1.8264005 ]
 [ 0.06139259  2.4187555 ]
 [ 0.21810865 -2.3058703 ]
 [-0.195345    2.7684455 ]
 [-0.32613984  2.627697  ]
 [-0.36819074  2.0505943 ]
 [ 2.9649143   3.8852904 ]
 [-2.171909   -2.6725817 ]
 [-0.10905603  3.212555  ]
 [ 3.783006    2.3649342 ]]
```

Гарантирован ли успех?





References

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

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

[3] Statistical Language Models Based on Neural Networks, T Mikolov, Ph. D. thesis, Brno University of Technology, https://www.fit.vutbr.cz/~imikolov/rnnlm/thesis.pdf

[4] Language modeling of Czech using neural networks, T Mikolov, Proc. 13th Conference STUDENT EEICT 2007, 1-3

[5] Firth, John R. “A synopsis of linguistic theory, 1930–1955.” Studies in linguistic analysis. 1957.

[6] Levy, Omer, and Yoav Goldberg. “Neural word embedding as implicit matrix factorization.” NeurIPS. 2014.