Word Embeddings

Простой способ: one-hot encoding

One-hot вектор размерности длины словаря
(например, 500.000)

$$cat = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix}$$

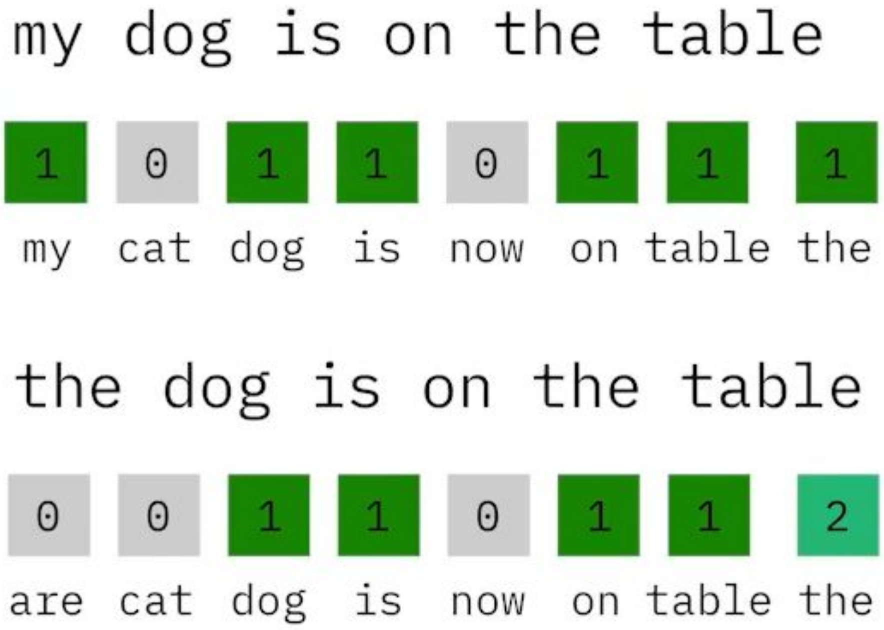
n (=500000) координат

$$mother = \begin{bmatrix} 0 \\ \dots \\ 1 \\ \dots \\ 0 \end{bmatrix}$$

i-я координата

Bag-of-Words

Сумма one-hot векторов слов



TF-IDF

$P(\mathrm{w}, \mathrm{d}, n_{\mathrm{dw}}) = (N_{\mathrm{w}}/N)^{n_{\mathrm{dw}}}$ - вероятность встретить n_{dw} раз слово w в документе d

$$-\log P(\mathrm{w}, \mathrm{d}, n_{\mathrm{dw}}) = n_{\mathrm{dw}} \cdot \log(N/N_{\mathrm{w}}) = TF(\mathrm{w}, \mathrm{d}) \cdot IDF(\mathrm{w})$$

$TF(\mathrm{w}, \mathrm{d}) = n_{\mathrm{dw}}$ - term frequency;

$IDF(\mathrm{w}) = \log(N/N_{\mathrm{w}})$ - inverted document frequency;

Context embeddings

Let’s take into account words meanings in some way:

$v(word_i)[\ j\ ] = \text{count(co-occurrences } word_i \text{ with } word_j \text{ in dataset)}$

$$
\begin{array}{ccccccc}
v(word_1) = & [12, & 1, & 0, & 10, & 5, & \ldots] \\
\uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \\
\text{horse} & \text{ride} & \text{wheel} & \text{roof} & \text{hair} & \text{breed} &
\end{array}
$$

$$
\begin{array}{ccccccc}
v(word_2) = & [20, & 10, & 0, & 0, & 1, & \ldots] \\
\uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \\
\text{car} & \text{ride} & \text{wheel} & \text{roof} & \text{hair} & \text{breed} &
\end{array}
$$

Pointwise mutual information

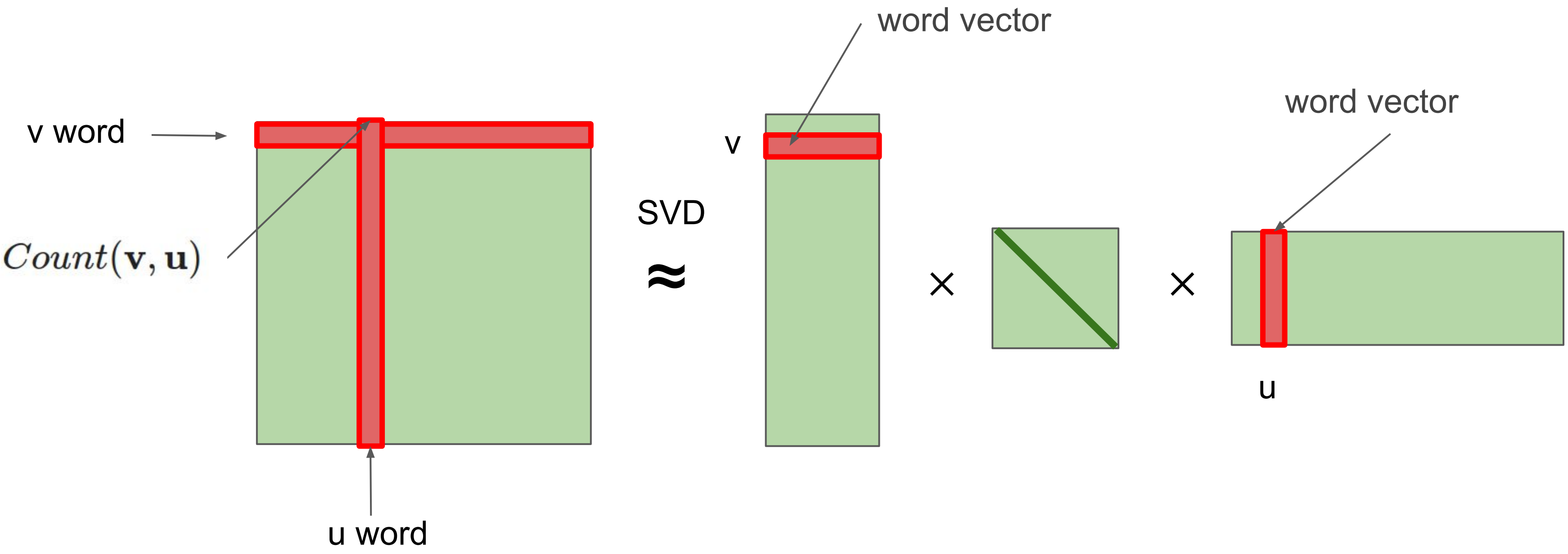
Скользящие окно фиксированной длины:

n_{uv} - встречаемость слова u и v вместе

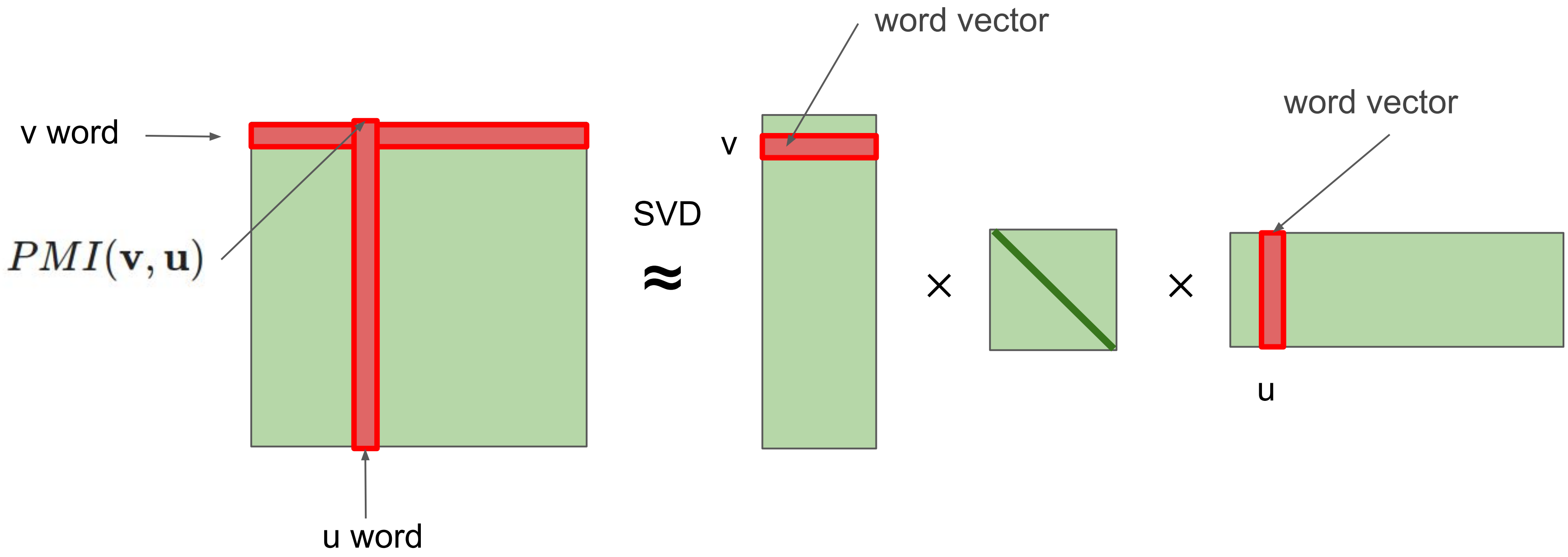
$$PMI = \log \frac{p(u,v)}{p(u)p(v)} = \log \frac{n_{uv}n}{n_u n_v}$$

$$pPMI = \max(0, PMI)$$

Co-Occurrence Counts



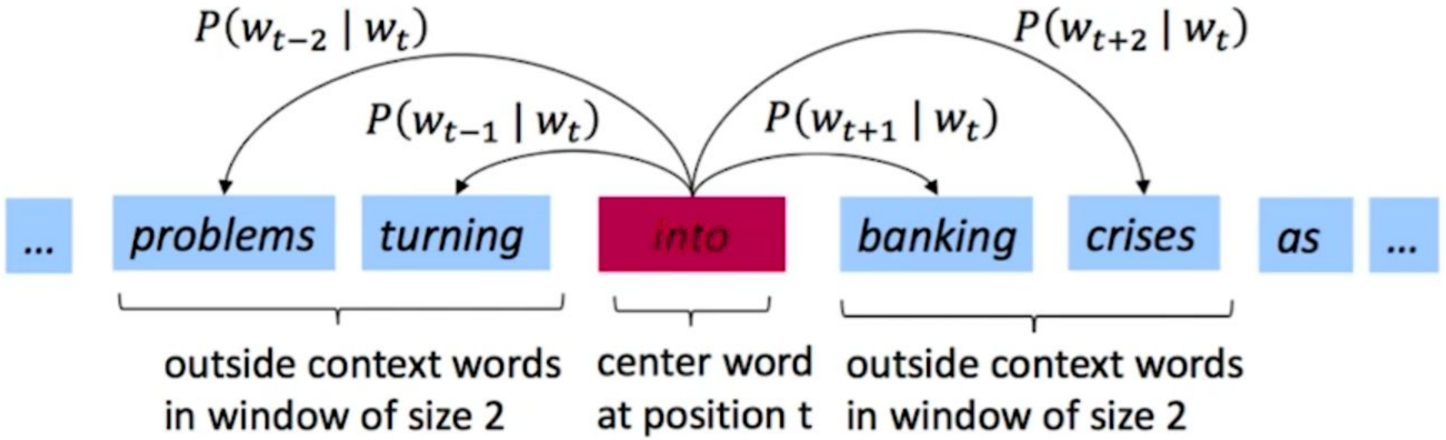
Pointwise mutual information



Word2Vec

We want to maximize probabilities of seeing a surrounding word based on center words.

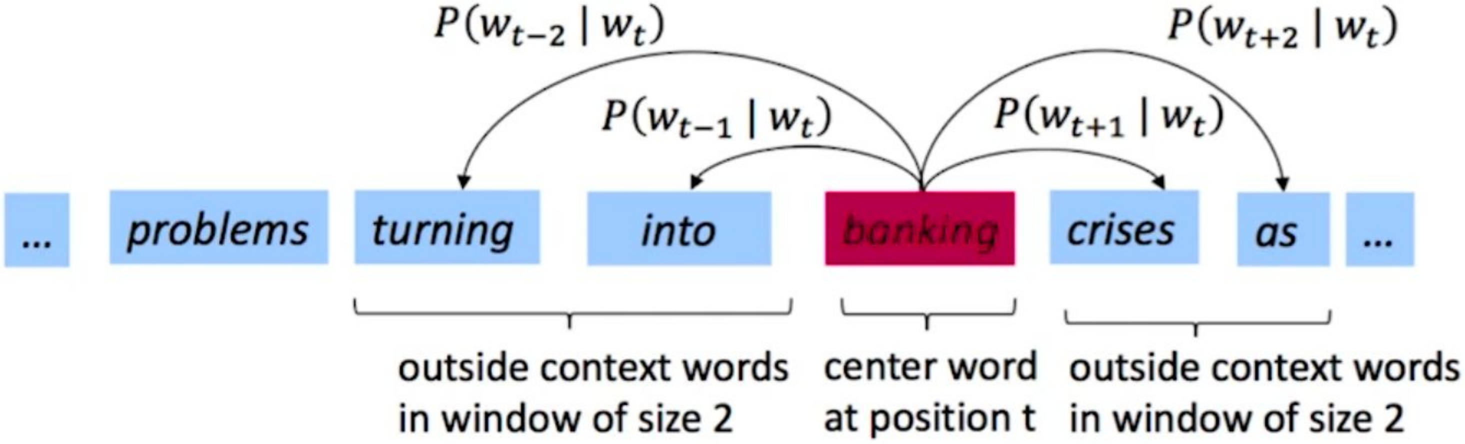
Going through text corpus by sliding windows.



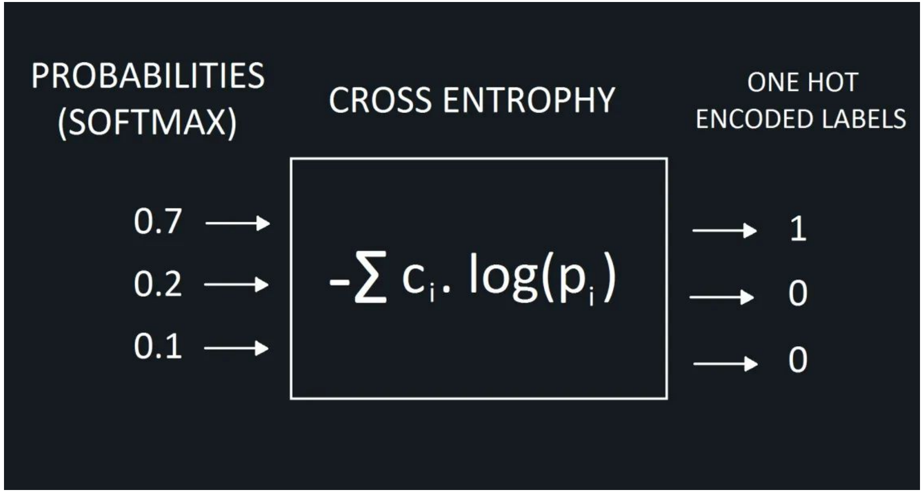
Word2Vec

We want to maximize probabilities of seeing a surrounding word based on center words.

Going through text corpus by sliding windows.



Cross-Entropy



$$H(q, p) = -\sum_x q(x) \log p(x)$$

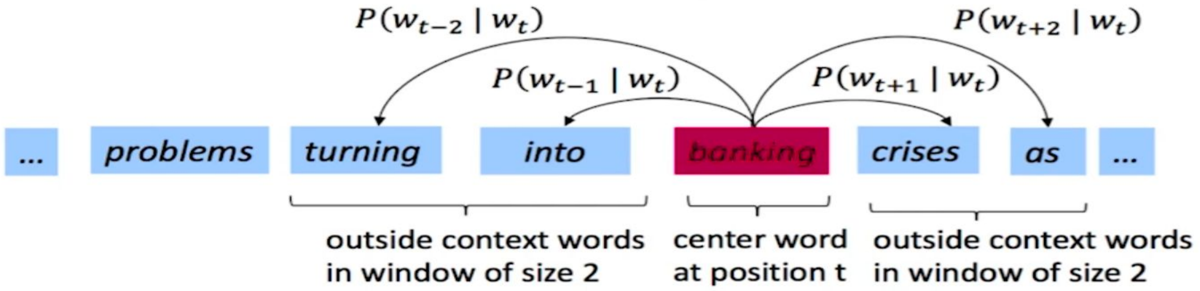
Word2Vec

Target function:

$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} P\left(w_{t+j} \mid w_t; \theta\right)$$

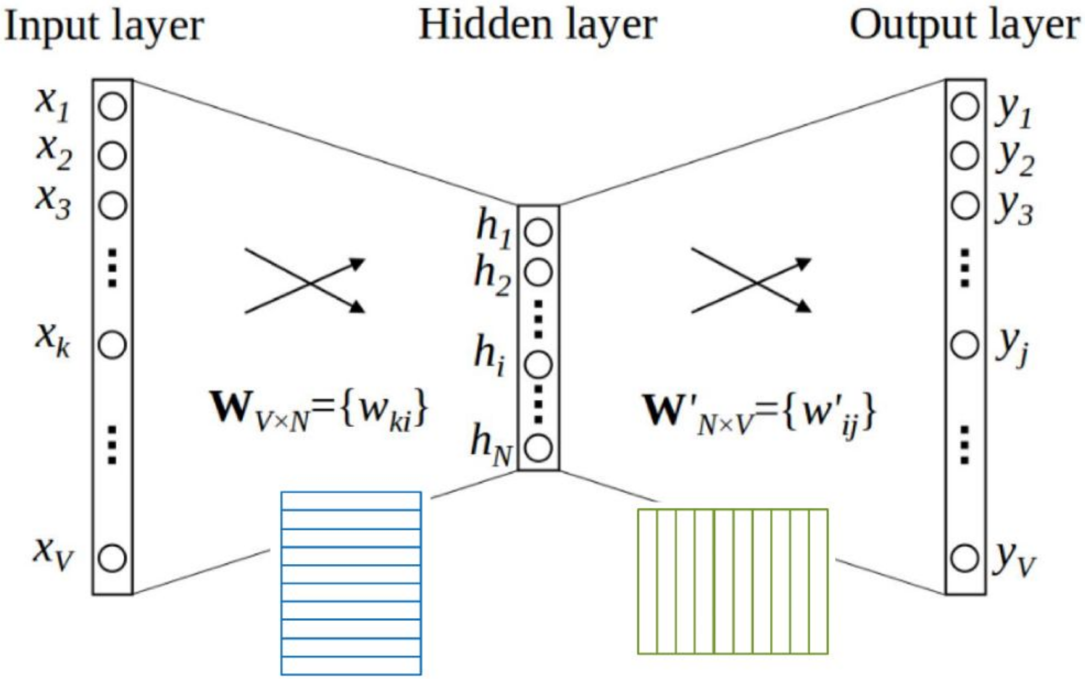
Or in log-likelihood point of view:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

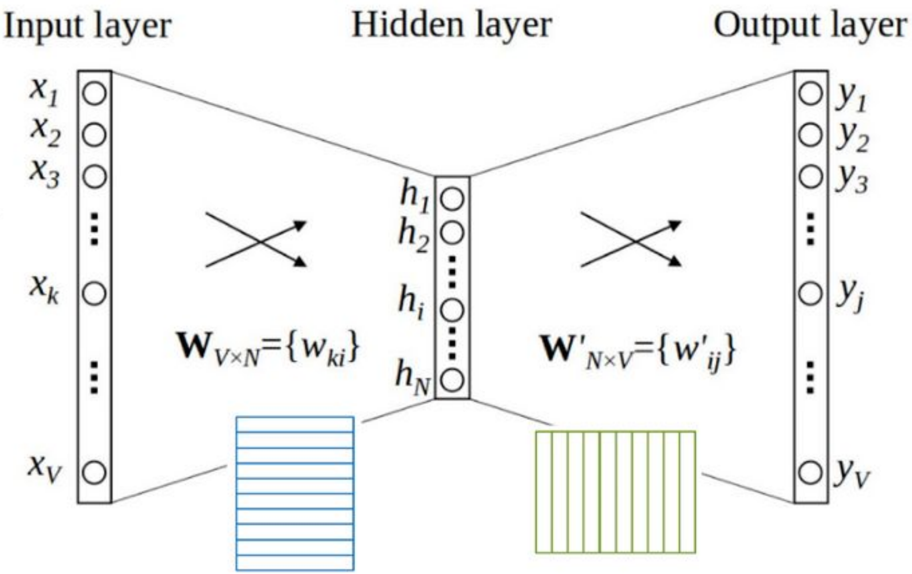


Word2Vec

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2NV}$$



Word2Vec

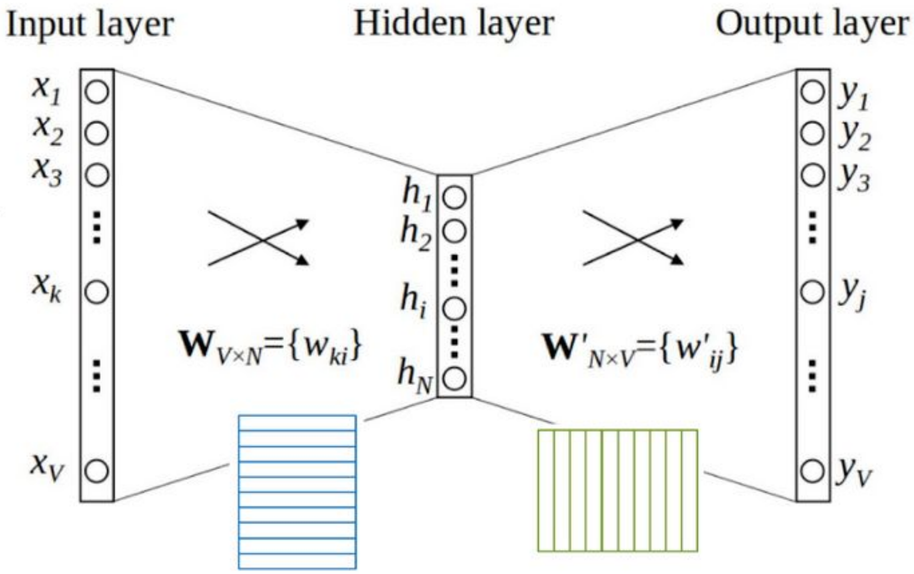


$$\mathbf{W} - V \times N$$
$$\mathbf{W}' - N \times V$$

1. $\mathbf{W}^T \cdot \mathbf{x} = h \Longrightarrow (N \times V) \cdot (V \times 1) = N \times 1$

$$\mathbf{W}^T = [w_1^T w_2^T \cdots w_V^T] \Rightarrow \sum_{i=1}^{V} w_i^T \mathbf{x}_i = w_k^T = h$$

Word2Vec



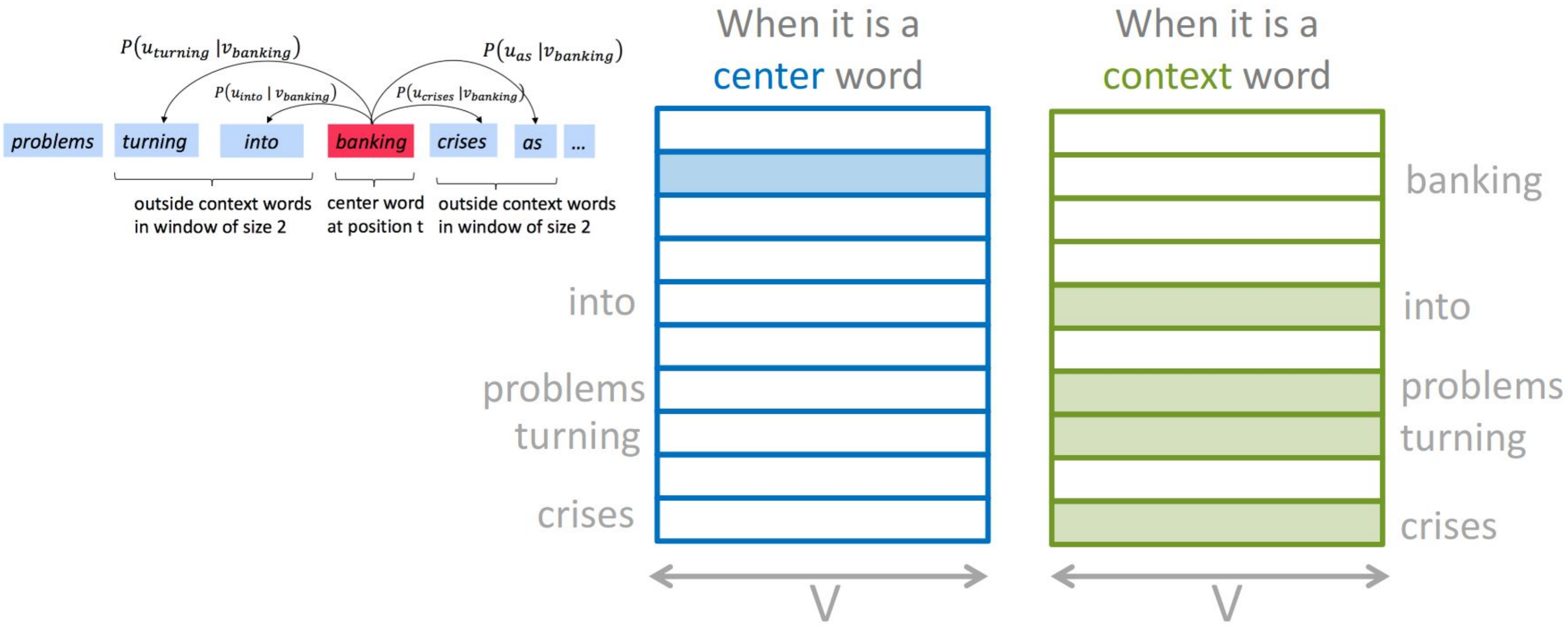
$\mathbf{W} - V \times N$

$\mathbf{W}' - N \times V$

2. $\mathbf{W}'^T \cdot h = y \Longrightarrow (V \times N) \cdot (N \times 1) = V \times 1$

$$\mathbf{W}' = [w'_1 w'_2 \cdots w'_V] \Rightarrow y_j = (\mathbf{W}'^T \cdot h)_j = (w'_j)^T w_k^T = \langle\, w_k, w'_j \rangle$$

Word2Vec

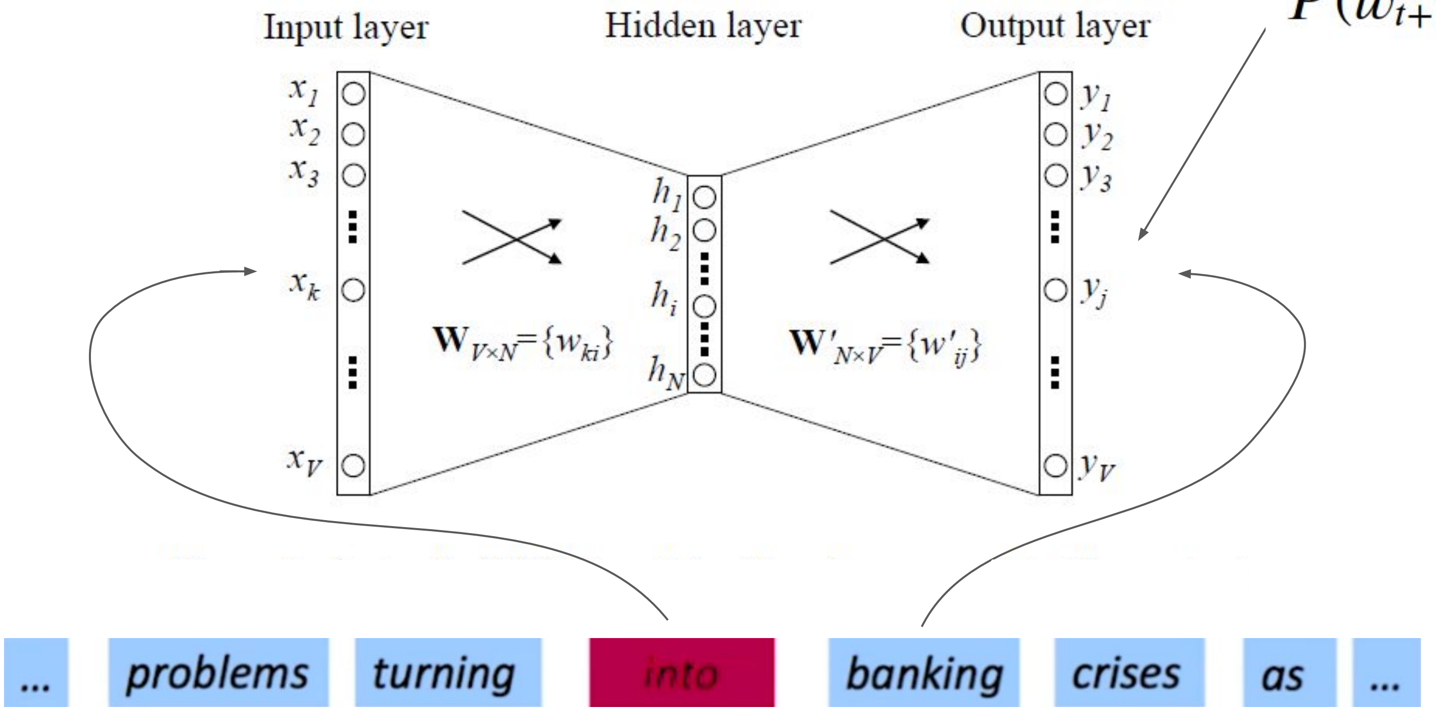


$P(u_{turning} \mid v_{banking})$
$P(u_{into} \mid v_{banking})$
$P(u_{crises} \mid v_{banking})$
$P(u_{as} \mid v_{banking})$
problems
turning
into
banking
crises
as
...
outside context words in window of size 2
center word at position t
outside context words in window of size 2
When it is a center word
When it is a context word
banking
into
problems
turning
crises
into
problems
turning
crises
V
V

Word2Vec

SoftMax models

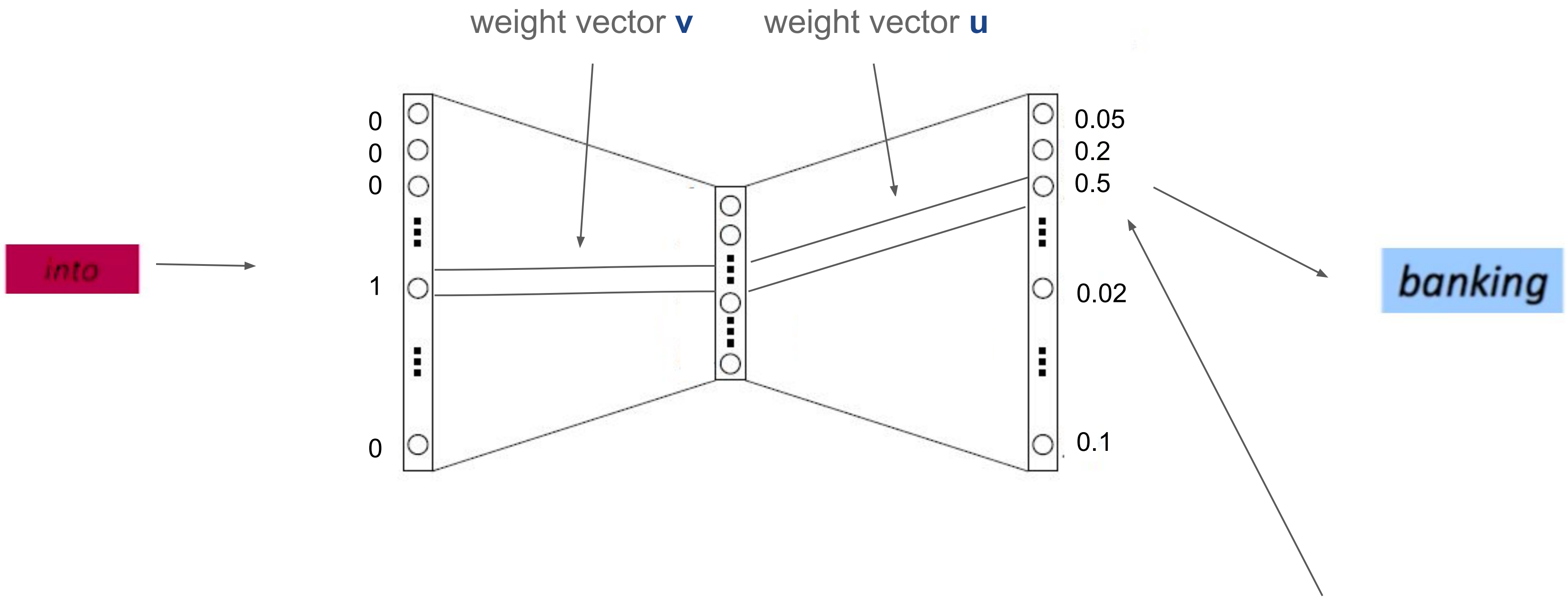
$$P(w_{t+1} \mid w_t; \theta)$$



Softmax

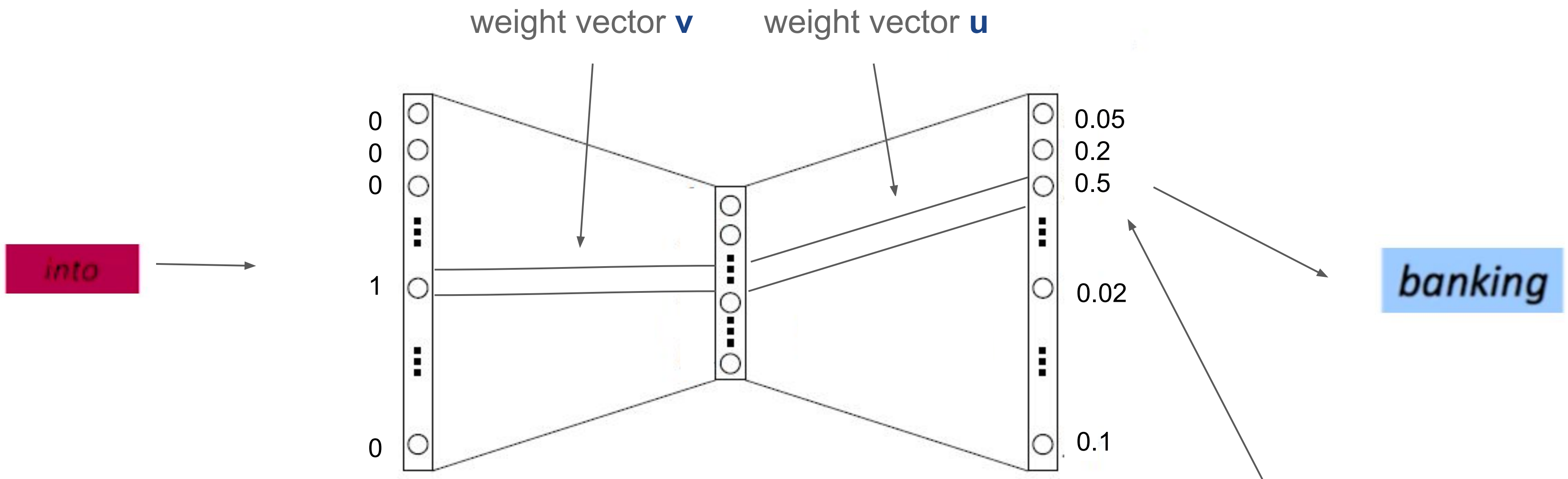
$$softmax(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

Word2Vec



$$p(\text{banking} \mid \text{into}, v, u) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Word2Vec



- u is a context word vector
- v is a center word vector

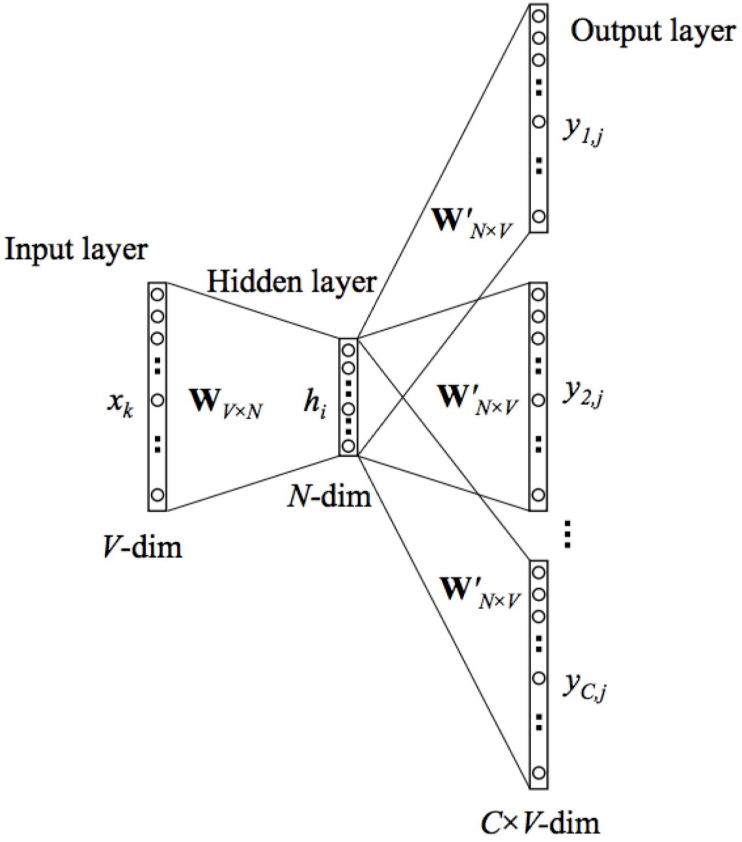
"Scalar product between me and my neighbour must be as big as possible"

p(banking | into,v, u) = $\frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$

Word2Vec: Skip-Gram

Maximize probabilities of seeing a surrounding word based on center words.

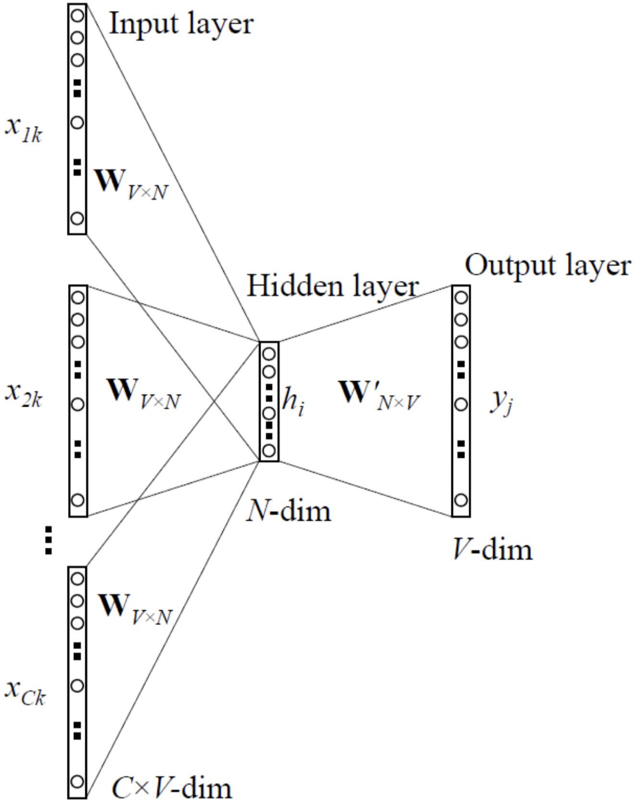
$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P\left(w_{t+j} \mid w_t; \theta\right)$$



Word2Vec: CBOW

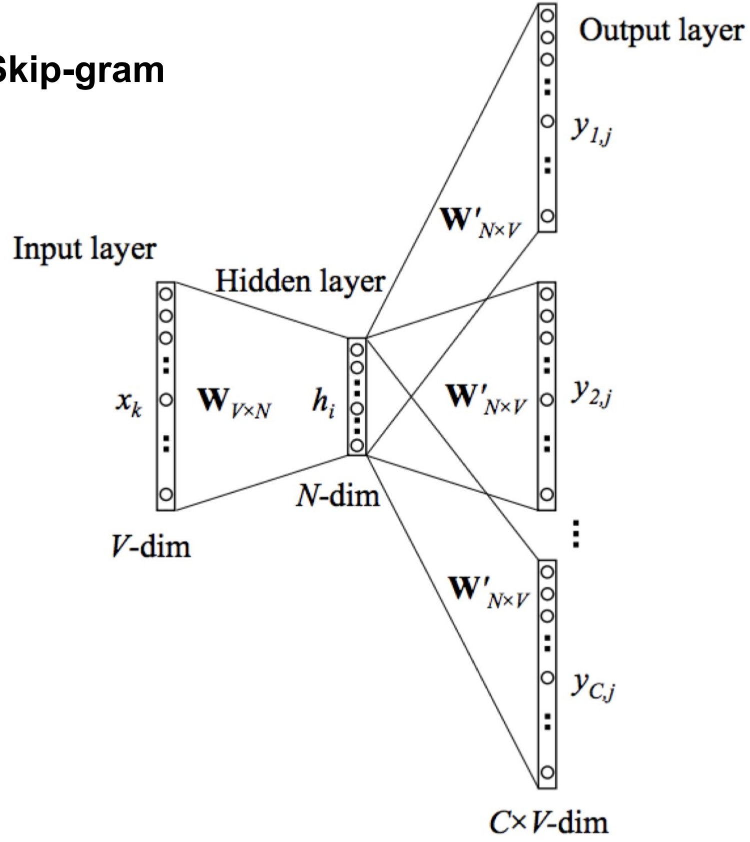
Maximize probabilities of seeing a center word based on surrounding words.

$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} P\left(w_t \mid w_{t+j} ; \theta\right)$$

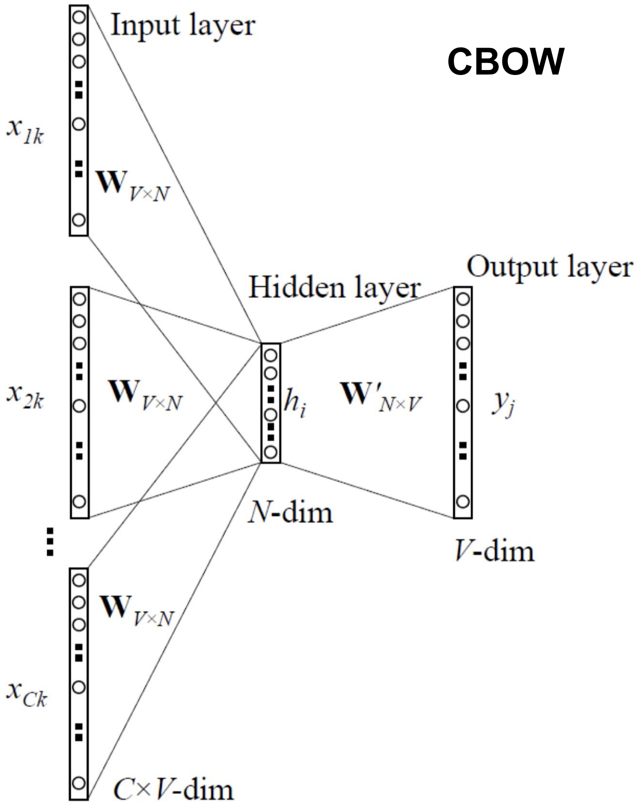


Word2Vec

Skip-gram



CBOW



Word2Vec

Problem:

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\boxed{\sum_{w\in V} \exp(u_w^T v_c)}}$$

Still big sum to compute

Word2Vec

Problem:

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\boxed{\sum_{w \in V} \exp(u_w^T v_c)}}$$

Still big sum to compute

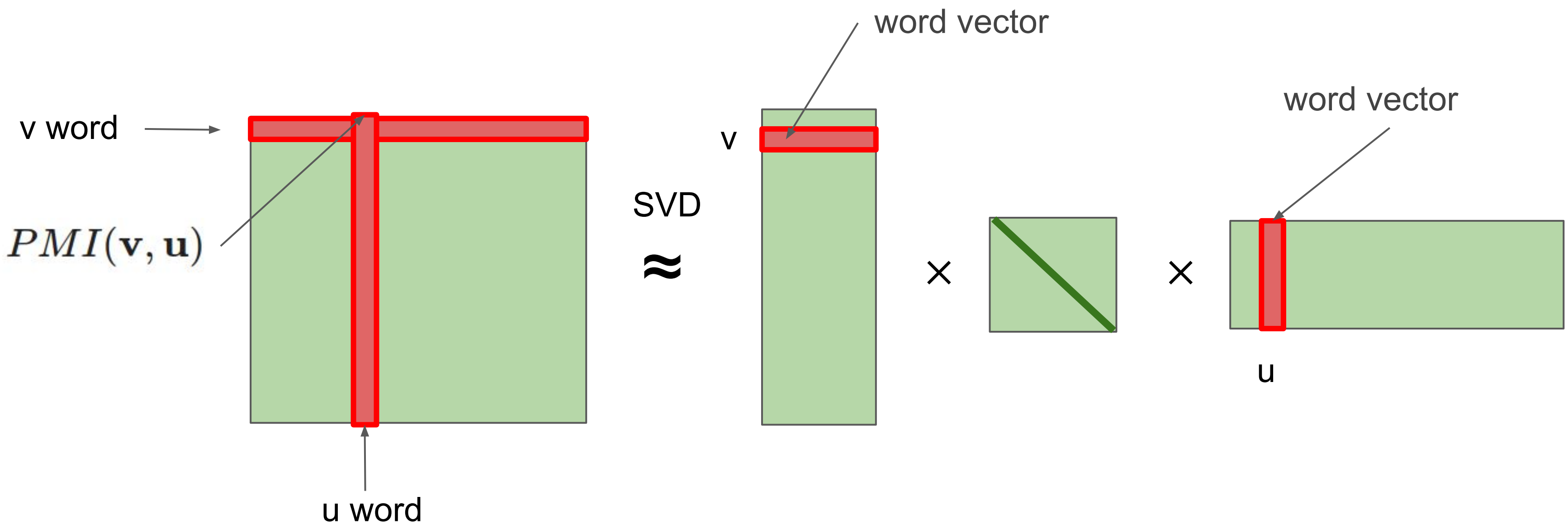
Possible solution:

- Negative sampling

$$P(w_i) = \frac{f(w_i)}{\sum_{j=0}^{n} \left(f(w_j)\right)}$$

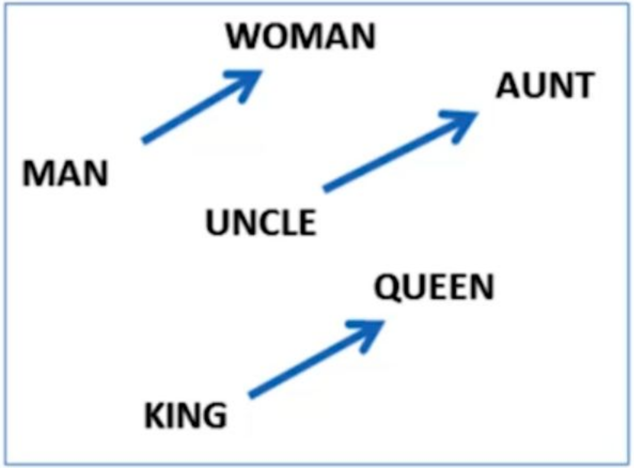
Word2Vec vs SVD

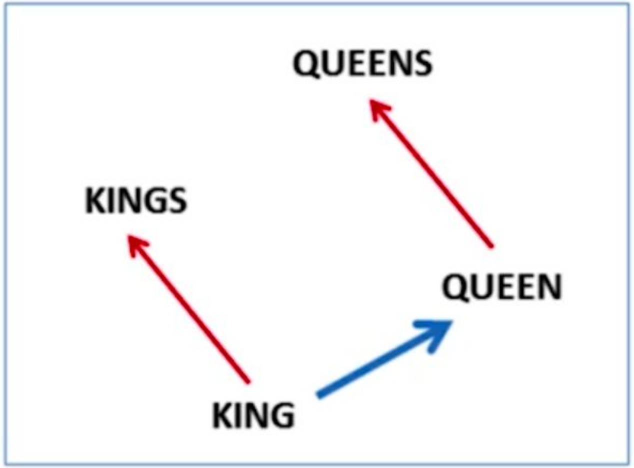
Word2Vec with negative sampling ≈ matrix factorization - [Link]



Word2Vec

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





Word2Vec

Target function:

$$L(\theta) = \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log P\left(w_{t+j} \mid w_t, \theta\right)$$

$$L_{\log}(\theta) = \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log P\left(w_{t+j} \mid w_t, \theta\right) = \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log \frac{\exp\left(u_{t+j}^{T} v_t\right)}{\sum_{w \in V} \exp\left(u_w^{T} v_t\right)}$$

$$L_{\log}(\theta) = \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \left(u_{t+j}^{T} v_t - \log \sum_{w \in V} \exp\left(u_w^{T} v_t\right)\right)$$

GloVe

Before training count occurrences of pairs [word$_i$, word$_j$] in corpus

Compute probabilities: $$P_{ij} = \frac{Count(v_i, v_j)}{Count(v_i)}, Count(v_i) = \sum_k Count(v_i, v_k)$$

Objective function:

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^{W} \boxed{f(P_{ij})} (u_i^T v_j - \log P_{ij})^2$$

Discount factor for rare words

FastText

- Divide word into bag of n-grams: apple = <ap, ppl, ple, le>
- Compute vector for each n-gram
- Vector for a word = sum of vectors for word n-grams
- Hash table
- Some n-grams have the same vector

FastText

- Divide word into bag of n-grams: apple = <ap, ppl, ple, le> (**BPE**)
- Compute vector for each n-gram
- Vector for a word = sum of vectors for word n-grams
- Hash table
- Some n-grams have the same vector

Advantages:

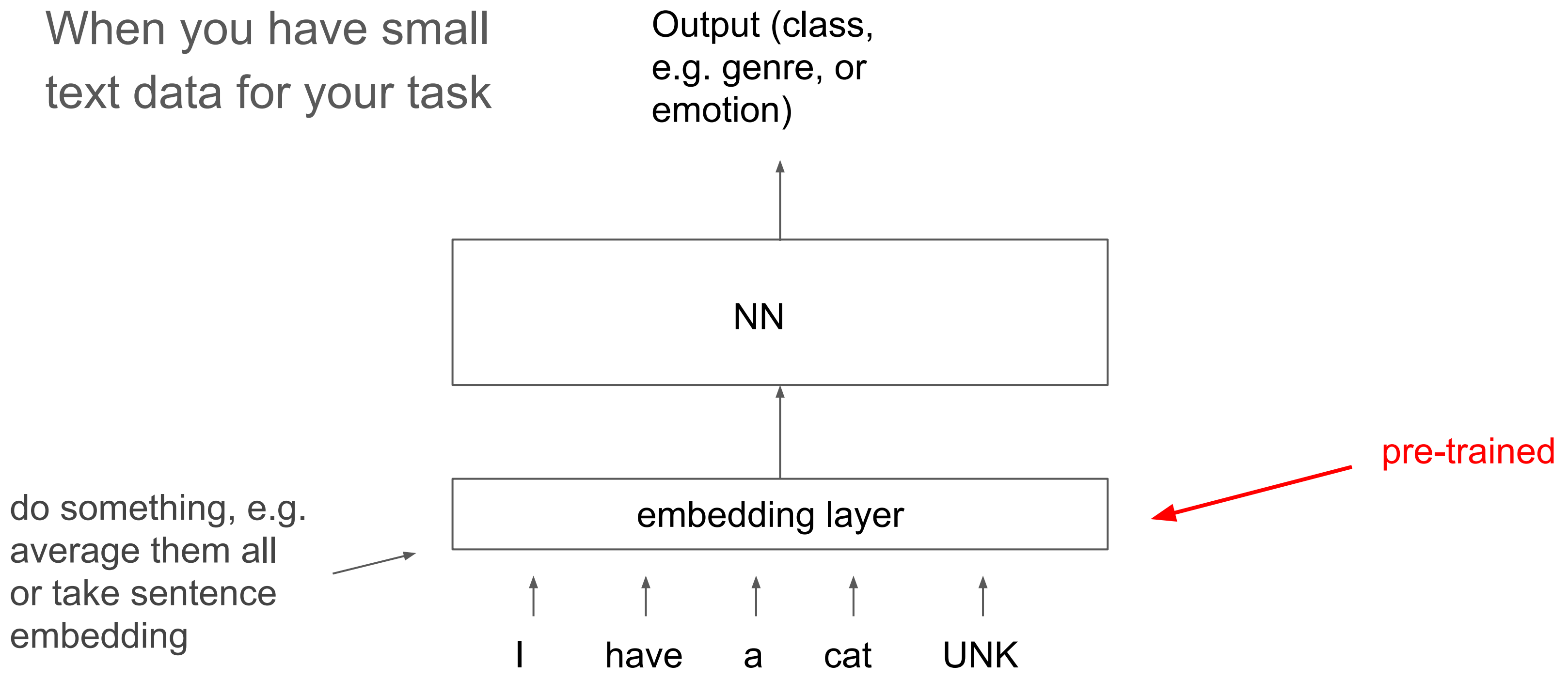
- Reasonable embeddings for rare words and words with mistakes
- Model is the same as before, we can even use model trained on words to train it further on n-grams!

Quality of embeddings

- Оценка семантической близости между словами
  - поиск близких слов
  - поиск лишних слов
- Поиск аналогий
- Решение почти любой NLP задачи
  - ранжирование
  - классификация

Classification task

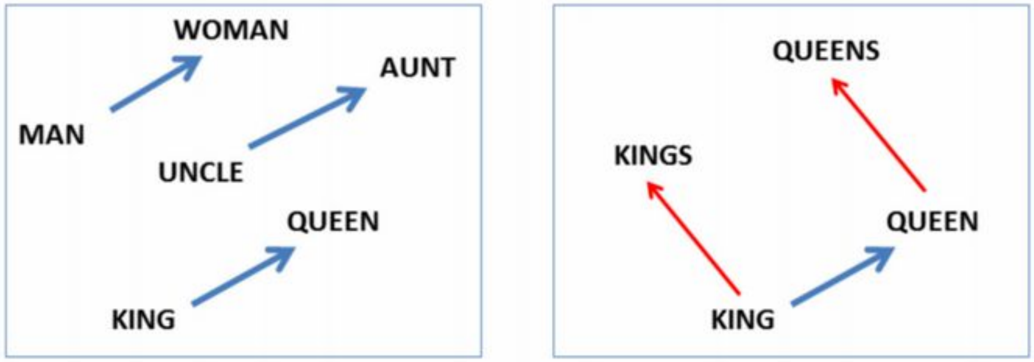
When you have small text data for your task



Поиск аналогий

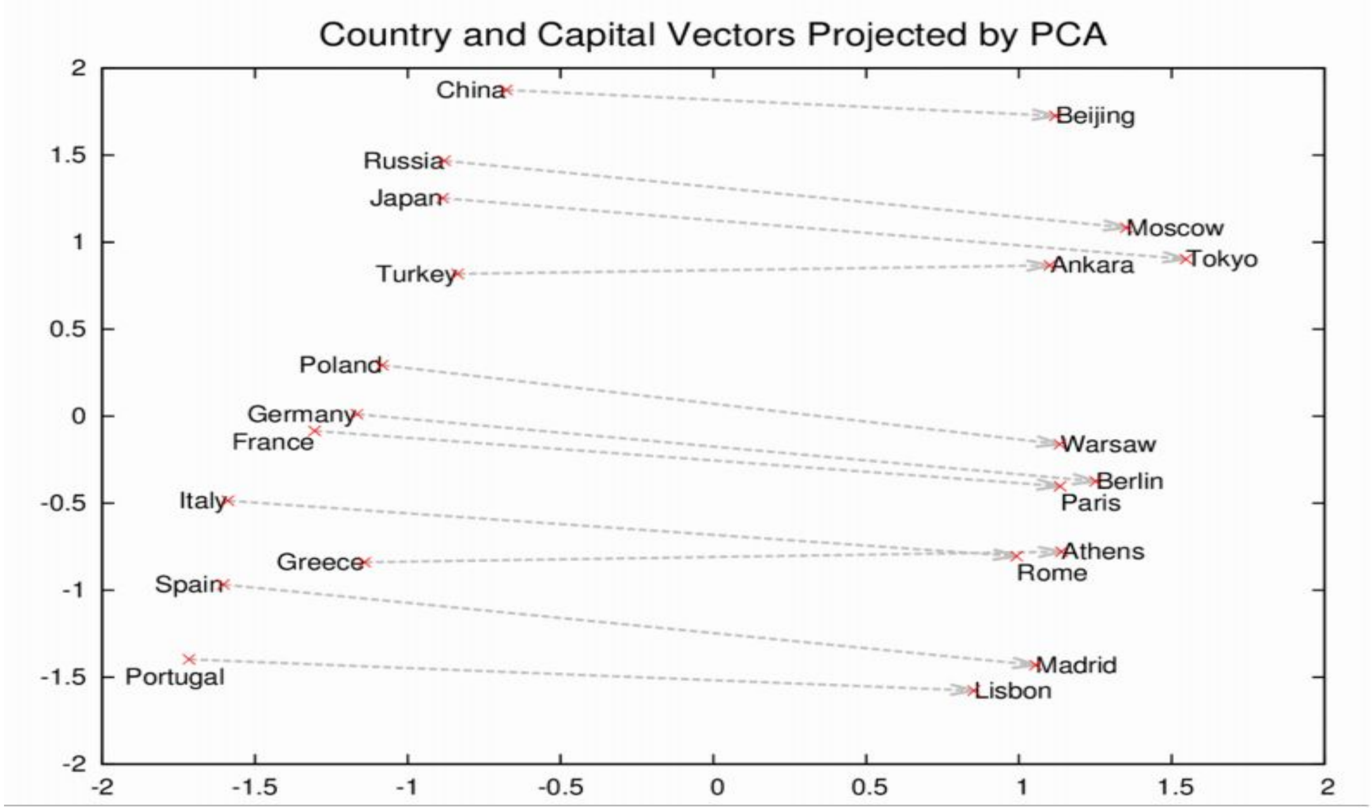
semantic: $v(\mathbf{king}) - v(\mathbf{man}) + v(\mathbf{woman}) \approx v(\mathbf{queen})$

syntactic: $v(\mathbf{kings}) - v(\mathbf{king}) + v(\mathbf{queen}) \approx v(\mathbf{queens})$



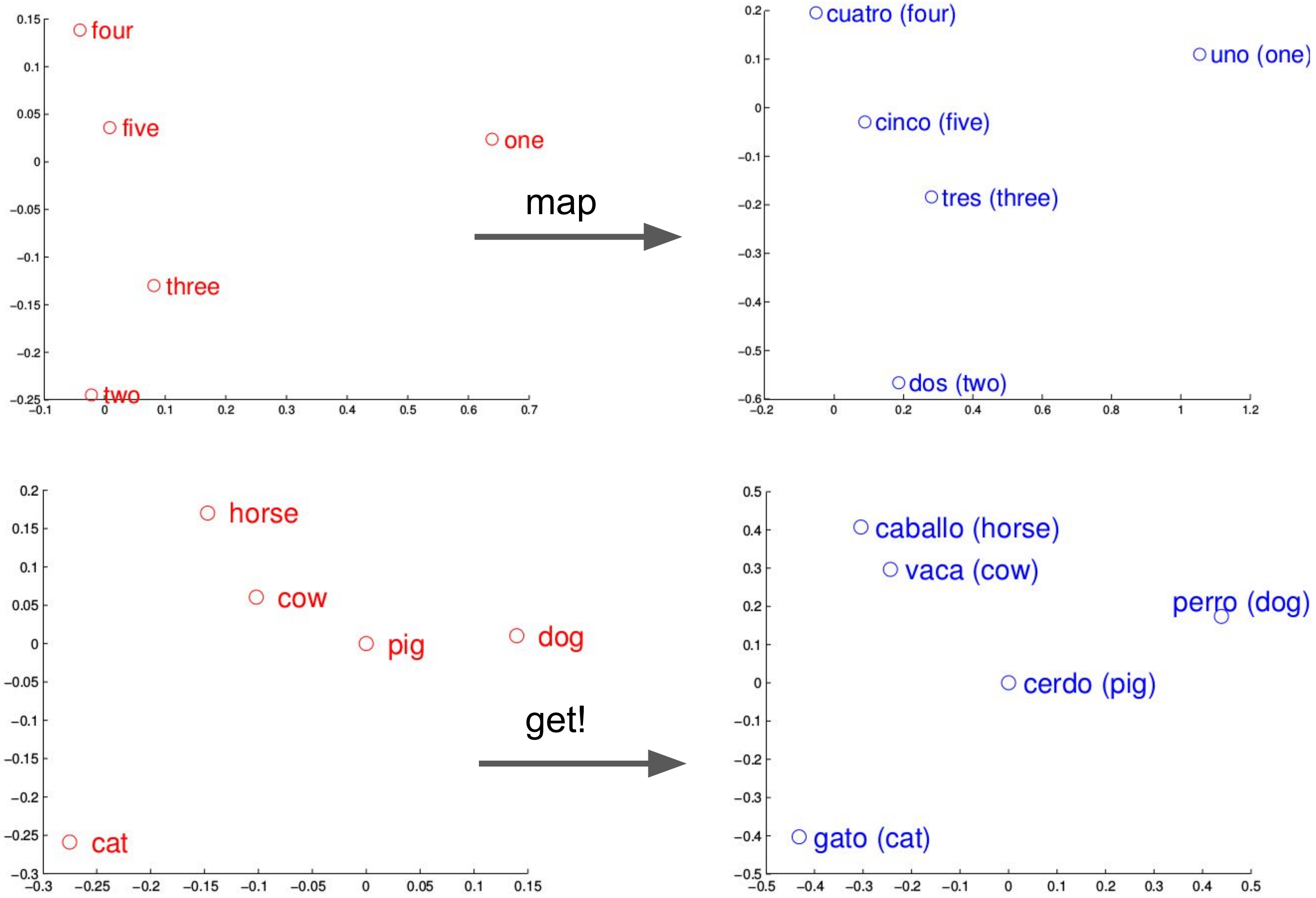
https://drive.google.com/file/d/1y2GKIKBzie7I8iycBO6gTKGiTTfJc4Dr/view

Поиск аналогий



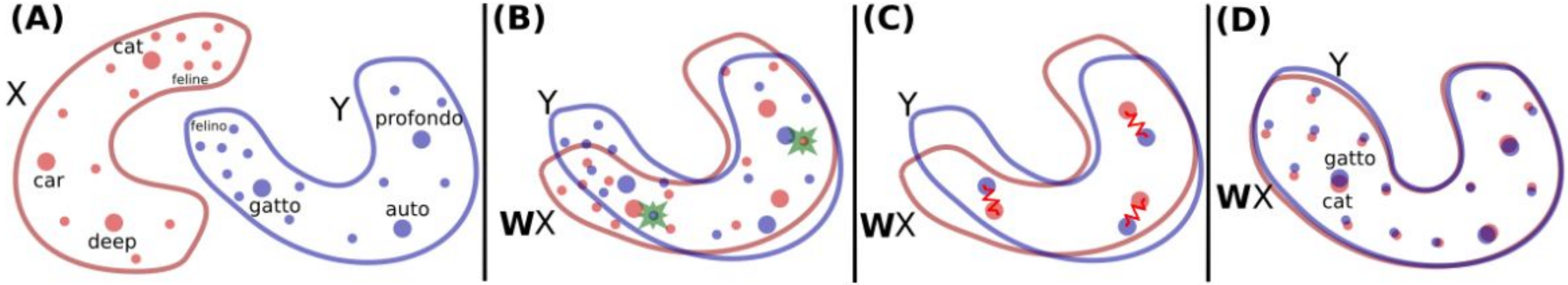
Language similarities

- train embeddings for english
- find mapping f() from english to spanish
- get new english word -- use f() to compute translation!



Language similarities

$$W^\star = \underset{W \in M_d(\mathbb{R})}{\operatorname{argmin}} \|WX - Y\|_{\mathrm{F}}$$



https://arxiv.org/pdf/1710.04087.pdf

Finally...

Okay, that’s great, but why do we actually need embeddings?

Timeline

January 16, 2013

WORD2VEC

Word2Vec Paper by Tomas Mikolov et al

January 2, 2014

GLOVE

GloVe: Global Vectors for Word Representation by Jeffrey Pennington et al.

July 15, 2016

FASTTEXT

Enriching Word Vectors with Subword Information by Piotr Bojanowski et al

June 12, 2017

TRANSFORMER

Attention Is All You Need by Ashish Vaswani et al

February 15, 2018

ELMO

Deep contextualized word representations by Matthew E. Peters et al

October 11, 2018

BERT

BERT: Pre-training of Deep Bidirectional Transformers for...

https://towardsdatascience.com/2019-year-of-bert-and-transformer-f200b53d05b9