

# Автоматическая обработка текстов

## II. Векторное представление слов

Екатерина Артемова (на основе лекций М. Апишева)

# Векторное представление слова

- Счетные модели: сингулярное разложение
- Модель word2vec
- Модель GloVe
- Оценка качества векторных представлений слов
- Об исследованиях:
  - Определение семантического сдвига
  - Многоязычные модели
  - Разрешение многозначности

# Векторное представление слова

# Векторное представление слова

## Word embeddings

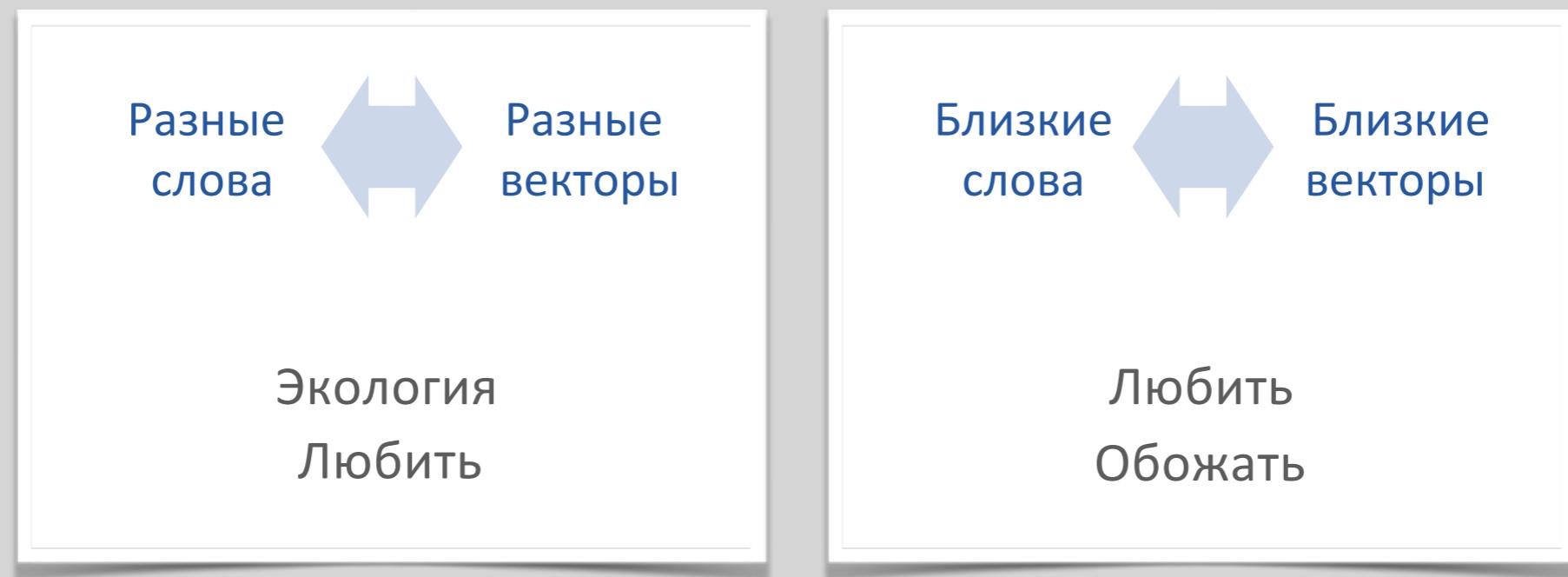
“Мой дядя самых честных правил”

“Мой” “самых” “правил”  
“дядя” “честных”

$$\begin{bmatrix} 123 \\ 456 \\ 12 \\ \dots \\ 89 \end{bmatrix} \quad \begin{bmatrix} 23 \\ 372 \\ 8 \\ \dots \\ 83 \end{bmatrix} \quad \begin{bmatrix} 16 \\ 124 \\ 76 \\ \dots \\ 29 \end{bmatrix} \quad \begin{bmatrix} 2 \\ 12 \\ 299 \\ \dots \\ 65 \end{bmatrix} \quad \begin{bmatrix} 177 \\ 6 \\ 504 \\ \dots \\ 304 \end{bmatrix}$$

# Векторное представление слова

## Word embeddings



# Близость между словами

## Близкие слова – синонимы?

### Семантическая близость слов

«Обычная» близость для слов

Например:

- компьютер
- ноутбук
- ПК
- лаптоп

### Близость $n$ -мерных векторов

$d(p, q)$ , где  $p$  и  $q$  — векторы:

$$p = (p_1, p_2, \dots, p_n)$$

$$q = (q_1, q_2, \dots, q_n)$$

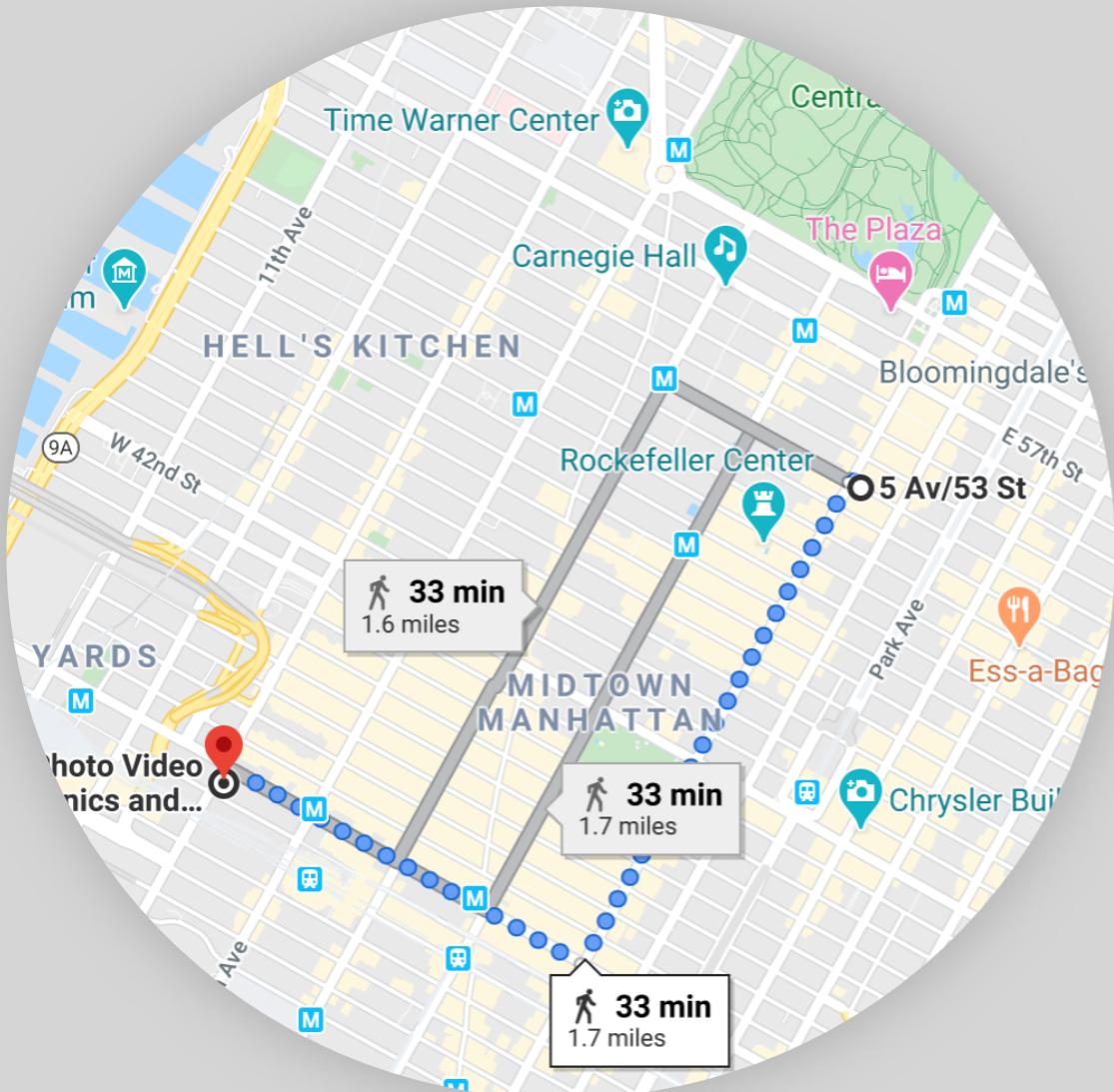
измеряется по-разному:

- Манхэттенское расстояние
- Евклидово расстояние
- Косинусное расстояние

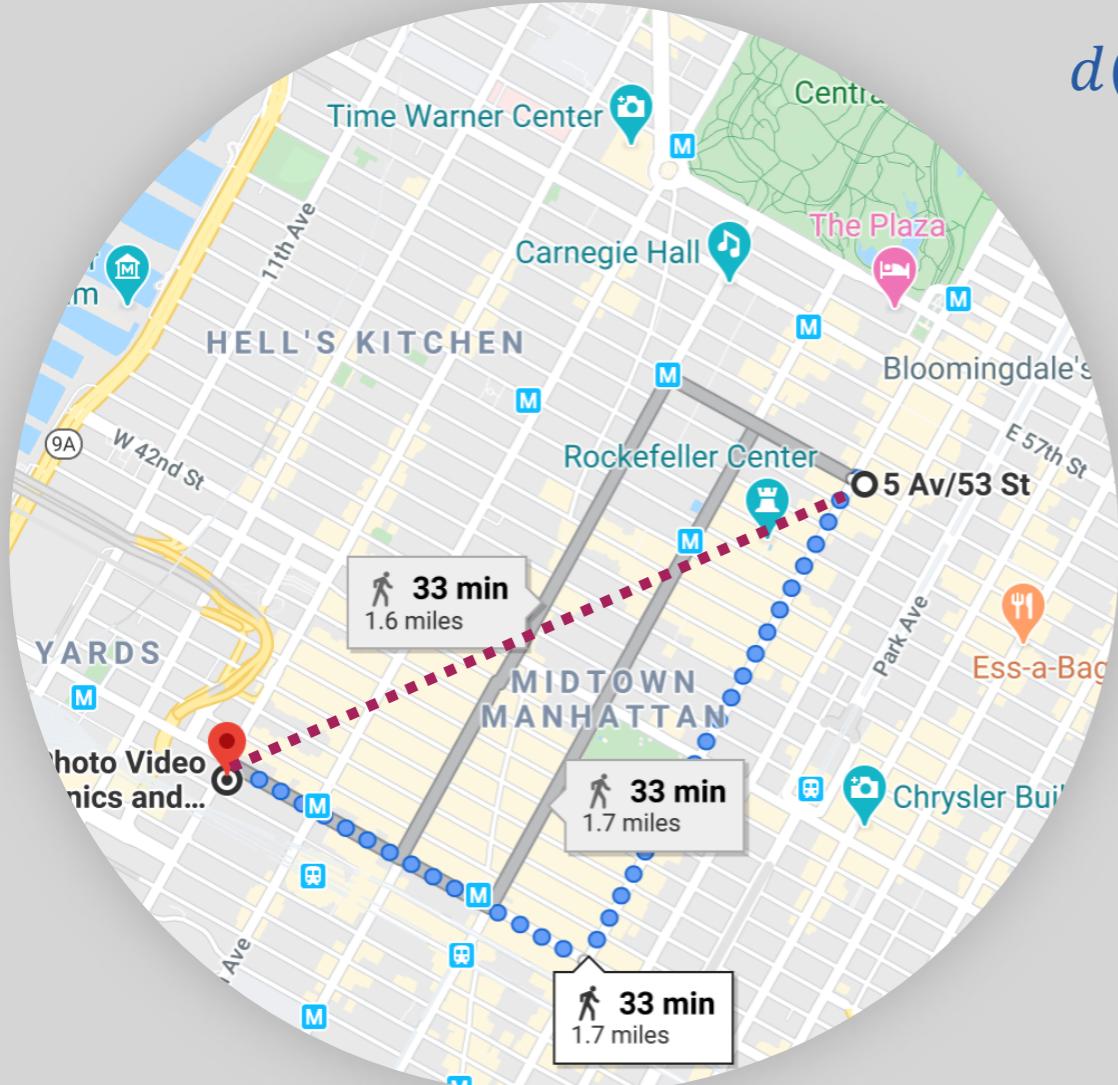
# Манхэттенское расстояние

## Manhattan distance, cityblock distance

$$d(p, q) = \sum_{i=1}^n |p_i - q_i|$$



# Евклидово расстояние



$$d(p, q) = \sqrt{\sum_{i=1}^n |p_i - q_i|^2}$$

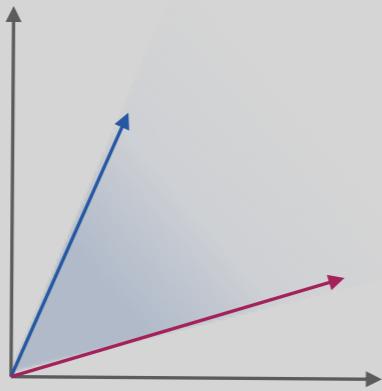
# Косинусное расстояние

$$p \cdot q = \| p \| \| q \| \cos(\theta)$$

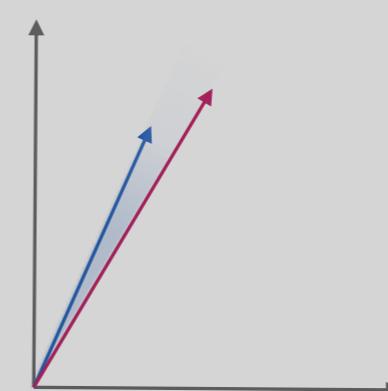
$$\cos(\theta) = \frac{p \cdot q}{\| p \| \| q \|} = \frac{(\sum_{i=1}^n p_i q_i)}{\sqrt{\sum_{i=1}^n p_i^2} \sqrt{\sum_{i=1}^n q_i^2}}$$

Косинусная близость —  $|\cos(\theta)|$

Косинусное расстояние —  $1 - |\cos(\theta)|$



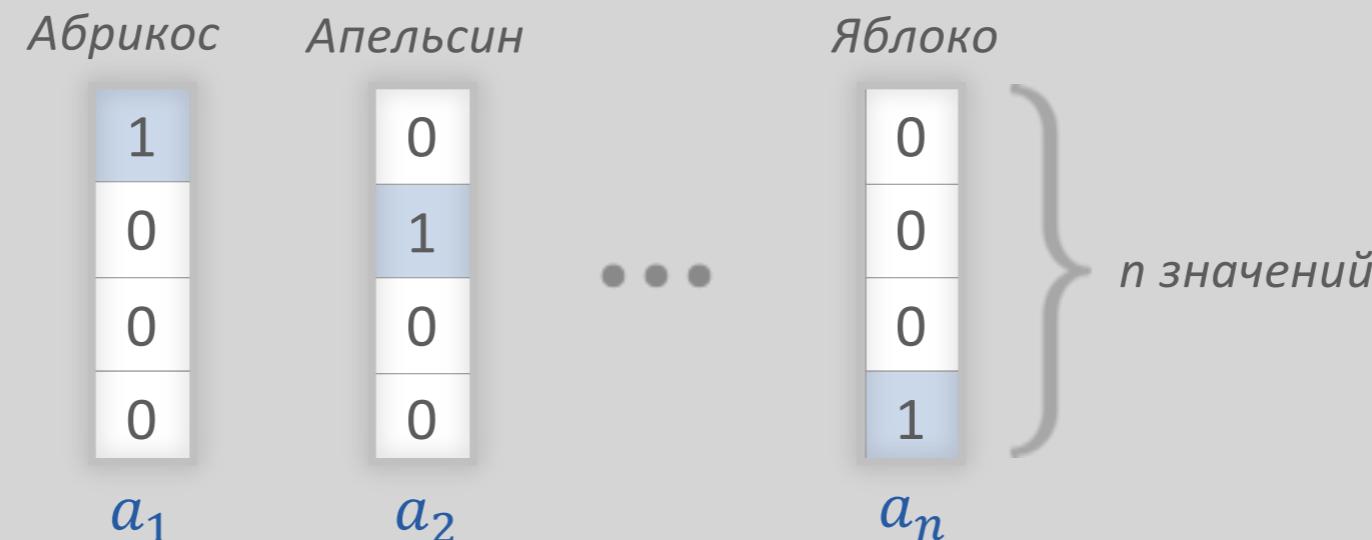
$$\theta \approx 45^\circ$$
$$\cos(\theta) \approx 0.7$$



$$\theta \approx 7^\circ$$
$$\cos(\theta) \approx 0.99$$

# One-hot кодирование

## “Мешок слов”



### Достоинства

- Простой способ

### Недостатки

- Длина векторов **не** фиксирована
- Вектора **ортогональны**

# Сингулярное разложение

## Singular value decomposition

- $d \in D, |D| = m$  – документы
- $w \in V, |V| = n$  – слова
- $M$  – мешок слов

$$\begin{matrix} & m \\ \text{---} & \boxed{\text{---}} \\ n & \end{matrix} = \begin{matrix} & n \\ \text{---} & \boxed{\text{---}} \\ n & \end{matrix} \times \begin{matrix} & m \\ \text{---} & \boxed{\text{---}} \\ n & \end{matrix} \times \begin{matrix} & m \\ \text{---} & \boxed{\text{---}} \\ m & \end{matrix}$$

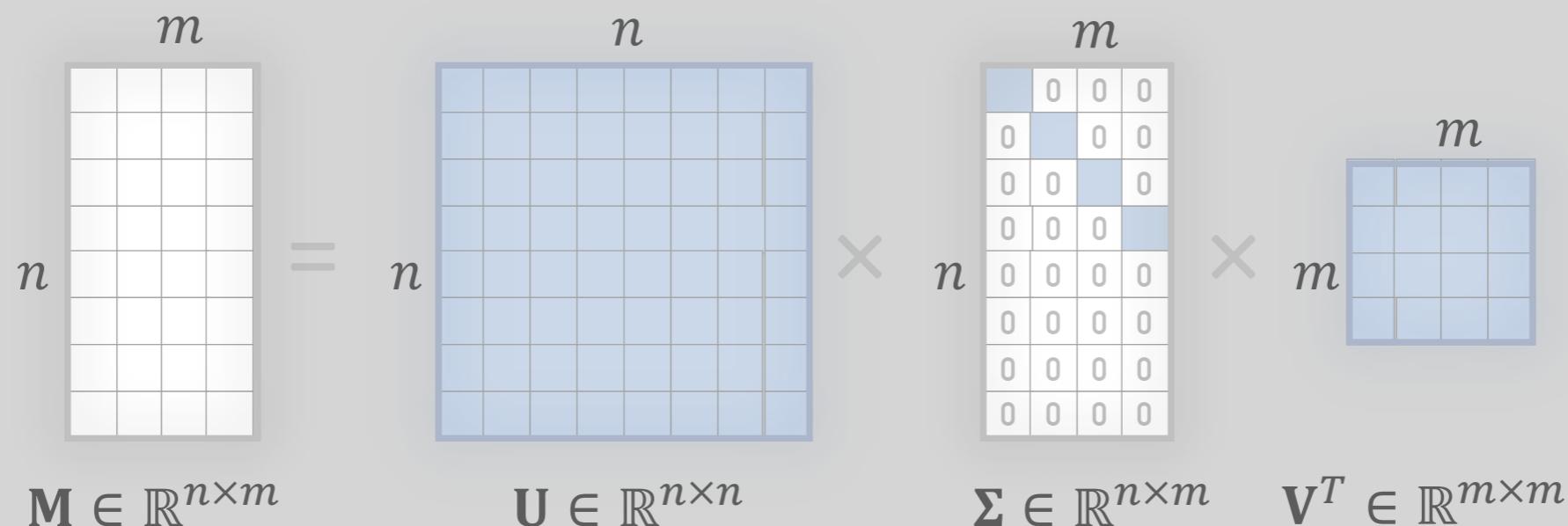
$\mathbf{M} \in \mathbb{R}^{n \times m}$        $\mathbf{U} \in \mathbb{R}^{n \times n}$        $\Sigma \in \mathbb{R}^{n \times m}$        $\mathbf{V}^T \in \mathbb{R}^{m \times m}$

# Сингулярное разложение

## Singular value decomposition

- Сингулярное разложение:

$$M = U\Sigma V^T$$



# Сингулярное разложение

## Singular value decomposition

- Снижение размерности: выберем  $k$  наибольших сингулярных чисел в  $\Sigma$
- Выберем  $k$  первых столбцов и строк в  $U$  и  $V^T$  –  $U_k$  и  $V_k^T$

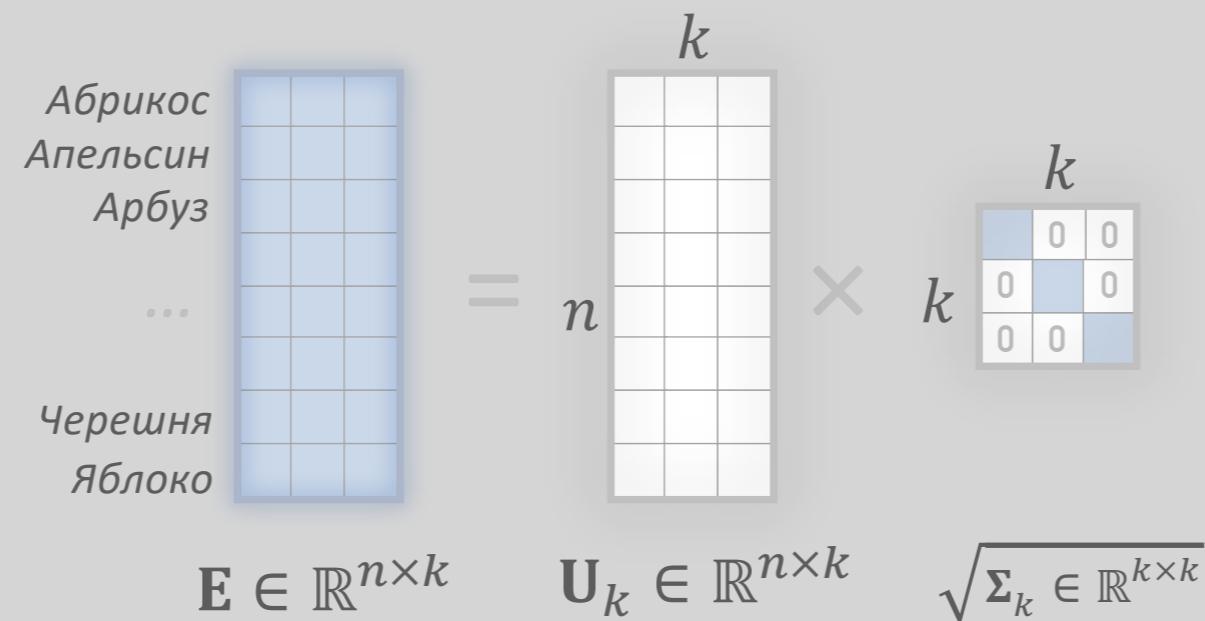
$$\begin{matrix} m \\ n \end{matrix} \begin{matrix} n \\ m \end{matrix} = \begin{matrix} k \\ n \end{matrix} \begin{matrix} U_k \\ n \end{matrix} \times \begin{matrix} k \\ n \end{matrix} \left\{ \begin{matrix} \Sigma_k \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{matrix} \right\} \times \begin{matrix} k \\ m \end{matrix} \left\{ \begin{matrix} V_k^T \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{matrix} \right\}$$

$M \in \mathbb{R}^{n \times m}$        $U \in \mathbb{R}^{n \times n}$        $\Sigma \in \mathbb{R}^{n \times m}$        $V^T \in \mathbb{R}^{m \times m}$

# Векторное представление слова

на основе сингулярного разложения матрицы “слово – документ”

- Каждому слову поставим в соответствие вектор  $U_k \sqrt{\Sigma_k}$



# Векторное представление слова

## на основе сингулярного разложения матрицы “слово – документ”

Что стало лучше:

- Векторы имеют фиксированную длину
- Они не взаимно ортогональны
- Семантическая близость как-то учитывается

Но:

- Добавление новых слов/документов потребует построение нового разложения
- Мы в явном виде должны работать с большой разреженной матрицей «мешка слов»

# Модель word2vec

# Дистрибутивная гипотеза

Близкие по смыслу слова встречаются в одном контексте

«Сегодня я съел **вкусный, сочный** апельсин»

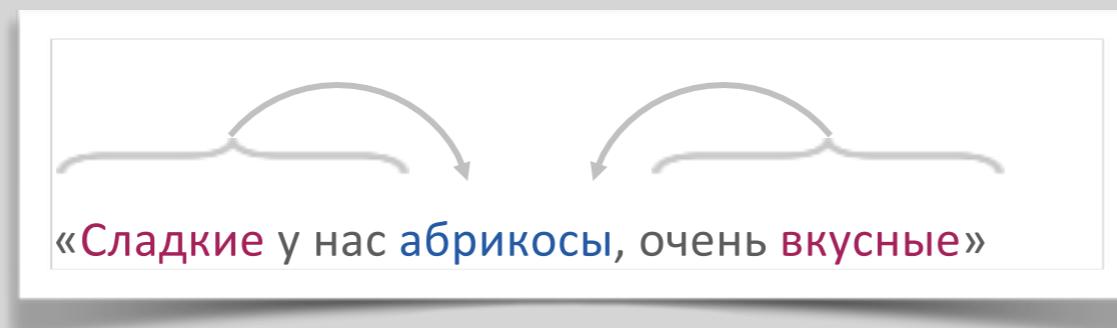
«Это **яблоко** очень сладкое и **сочное**»

«**Сладкие** у нас абрикосы, очень **вкусные**»

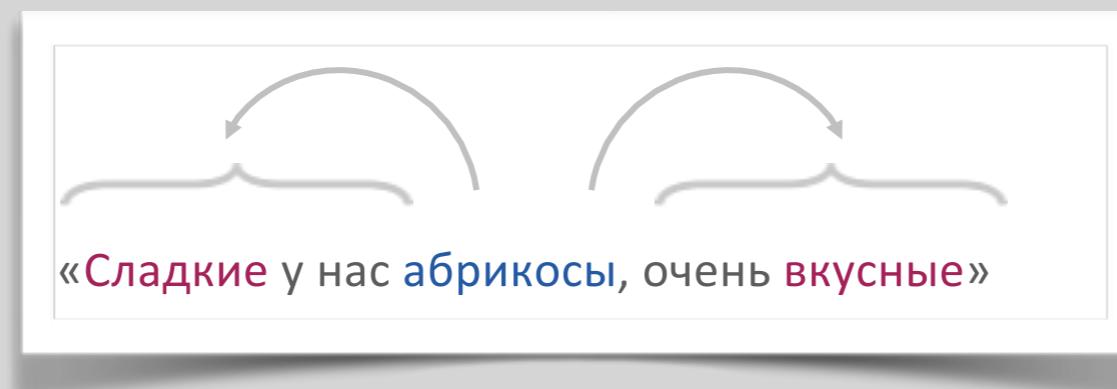
# Модель word2vec

## Постановки задач СВОУ и skip-gram

- Continues Bag of Word – предсказать по контексту слово



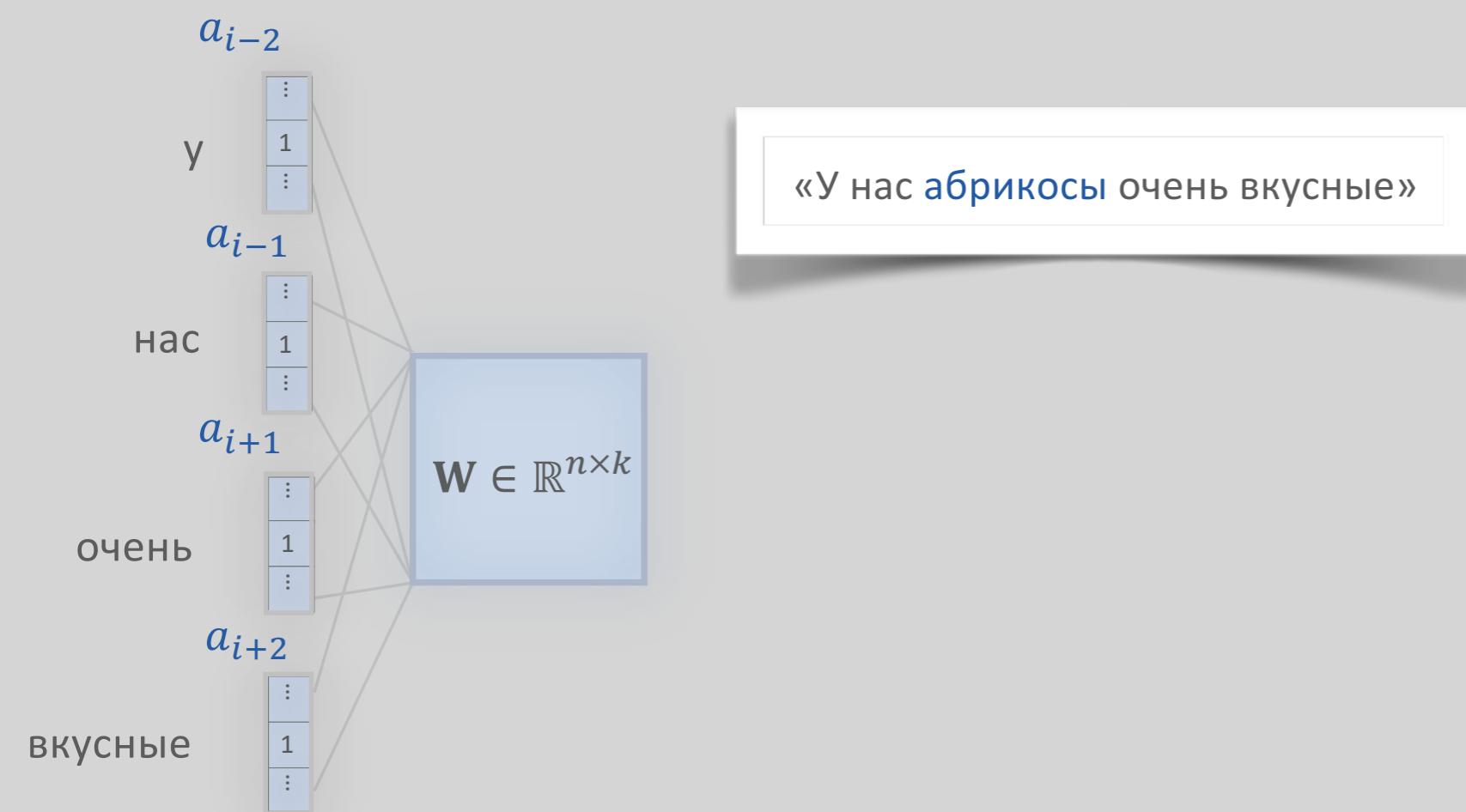
- Skip-gram – предсказать контекст по слову



# Модель СВOW

## Нейронная сеть с одним скрытым слоем

- Разобьем корпус на контексты размера  $2C + 1$
- Вход нейронной сети:  $2C$  векторов контекста, размерность вектора –  $n$



# Модель СВOW

## Нейронная сеть с одним скрытым слоем

- Применим ко входу преобразование  $W$  и усредним результаты – получим  $k$ -мерный вектор скрытого

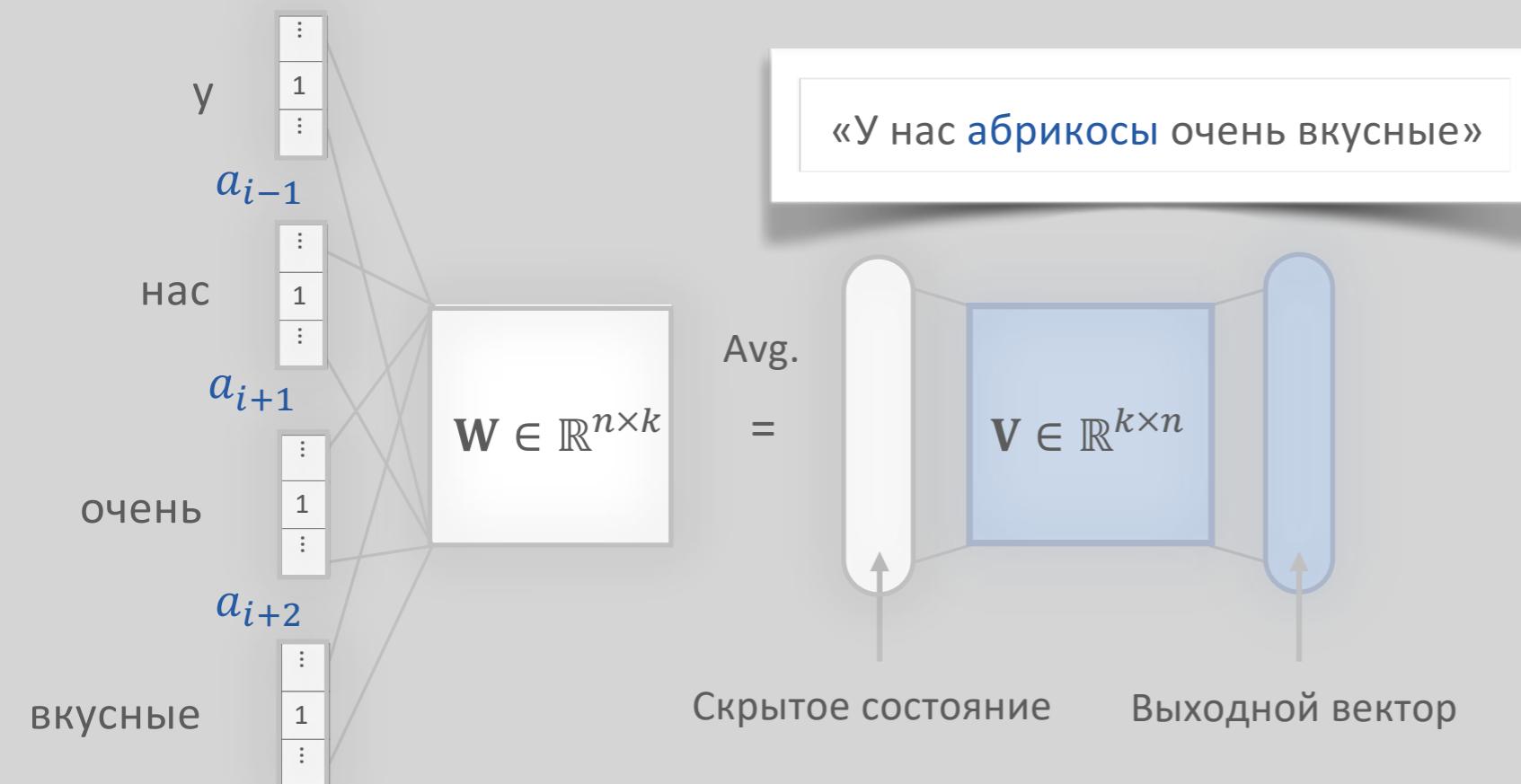
$$\text{состояния: } h = \frac{1}{4} W^T (a_{i-2} + a_{i-1} + a_{i+1} + a_{i+2})$$



# Модель СВOW

## Нейронная сеть с одним скрытым слоем

- Умножим скрытое состояние на вторую матрицу и получим  $n$ -мерный выходной вектор:  $b = V^T h$



# Модель CBOW

## Оптимизационная задача

- Для  $i$ -того окна по контексту  $c_i$  предсказать центральное слово  $w_i$ :

$$\sum_{i=1}^N \log p(w_i | c_i) \rightarrow \max_{W, V}$$

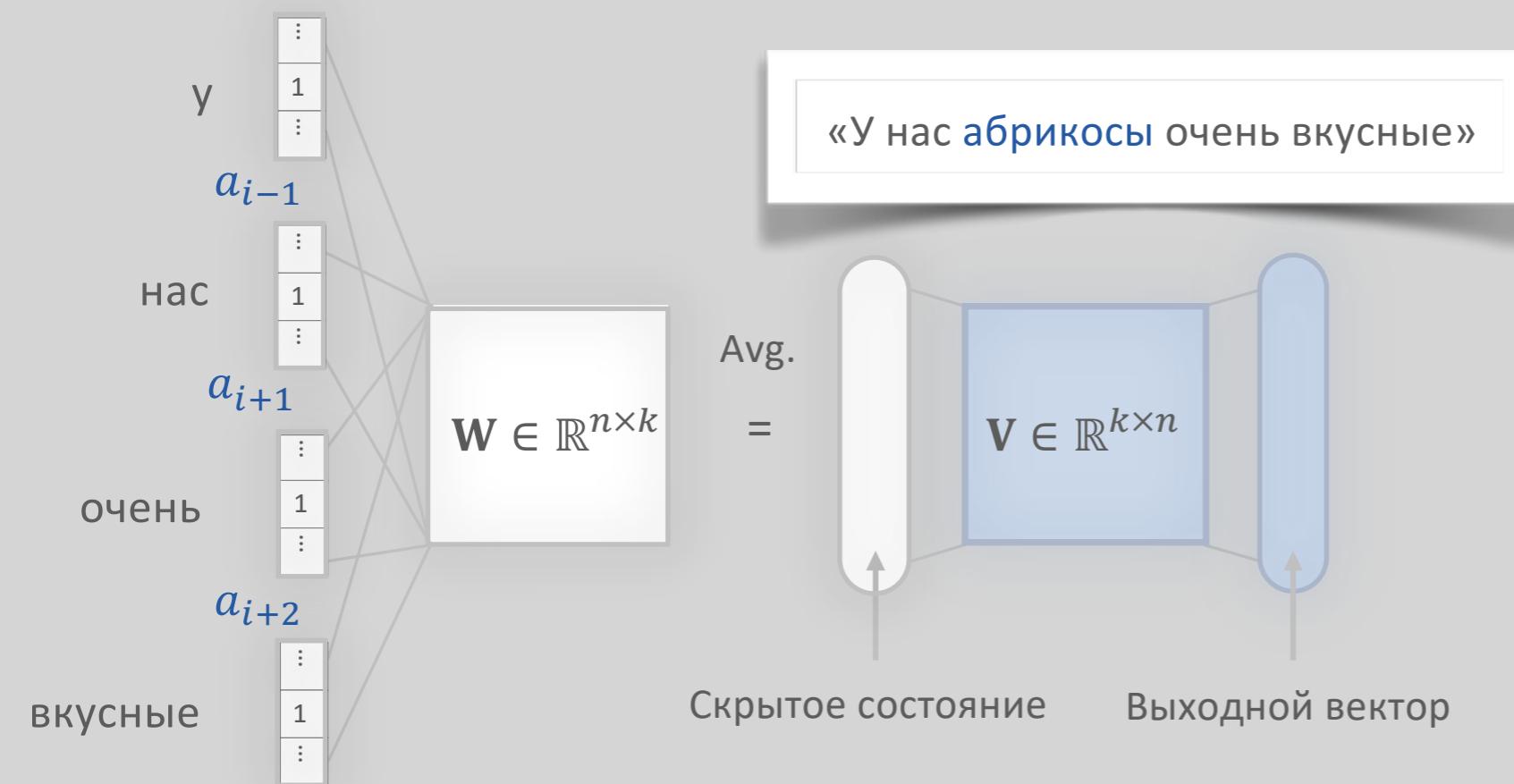
- Выход последнего слоя сети:  $\text{softmax}(b) = (z_i, \dots, z_j)$ , где

$$z_j = p(w_j | c_i) = \frac{e^{b_j}}{\sum_{k=1}^n e^{b_k}}$$

# Модель СВОУ

## Искомые вектора

- Получаем две матрицы весов:  $W$  и  $V$
- Обычно матрицу  $W$  считают искомой матрицей векторов



# Модель Skip-gram

## Нейронная сеть с одним скрытым слоем

- Разобьем корпус на контексты размера  $2C + 1$
- Вход нейронной сети:  $n$ -мерный one-hot вектор центрального слова



# Модель Skip-gram

## Нейронная сеть с одним скрытым слоем

- Выход нейронной сети: вероятностное распределение слова контекста при условии центрального слова



# Модель Skip-gram

## Оптимизационная задача

- Для  $i$ -того центрального слова  $w_i$  предсказать слово контекста  $c_i$ :

$$\sum_{i=1}^N \sum_{j=-C, j \neq 0}^C \log(c_j | w_i) \rightarrow \max_{W, V}$$

# Обучение моделей word2vec

## Функции потерь

- Проблема: вычисление softmax на выходном слое требует прохода по всему словарю –  $O(n)$  операций, вычислительно трудная задача
- Методы сокращения перебора:
  - Иерархический softmax
  - Noise Contrastive Estimation (NCE)
  - Negative sampling (NS)

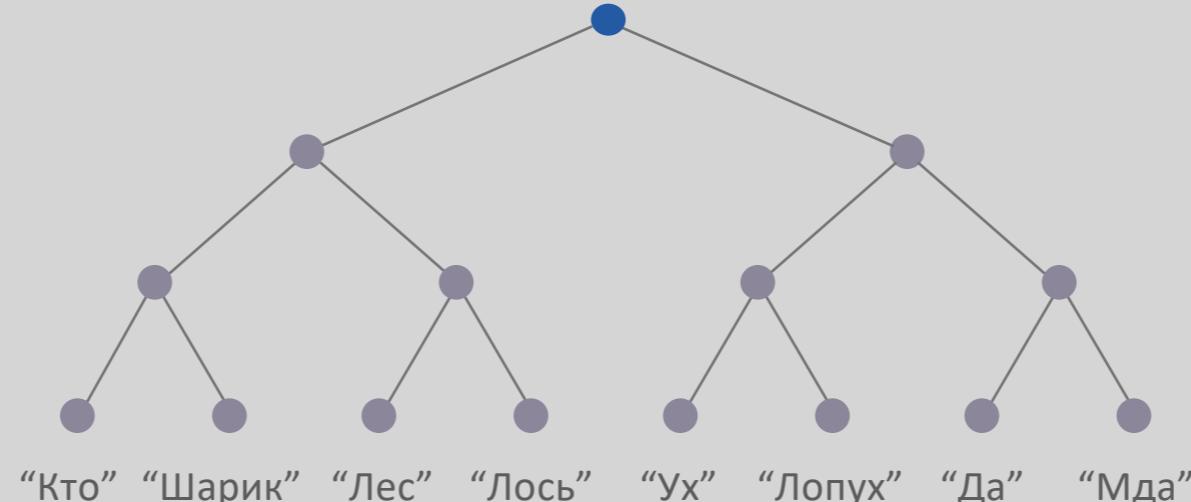
# Обучение моделей word2vec

## Иерархический softmax

- Будем оценивать только значения, стоящие в позициях предсказываемых слов контекста

$$\sum_{i=1}^N \sum_{j=-C, j \neq 0}^C \log(c_j | w_i) \rightarrow \max_{W, V}$$

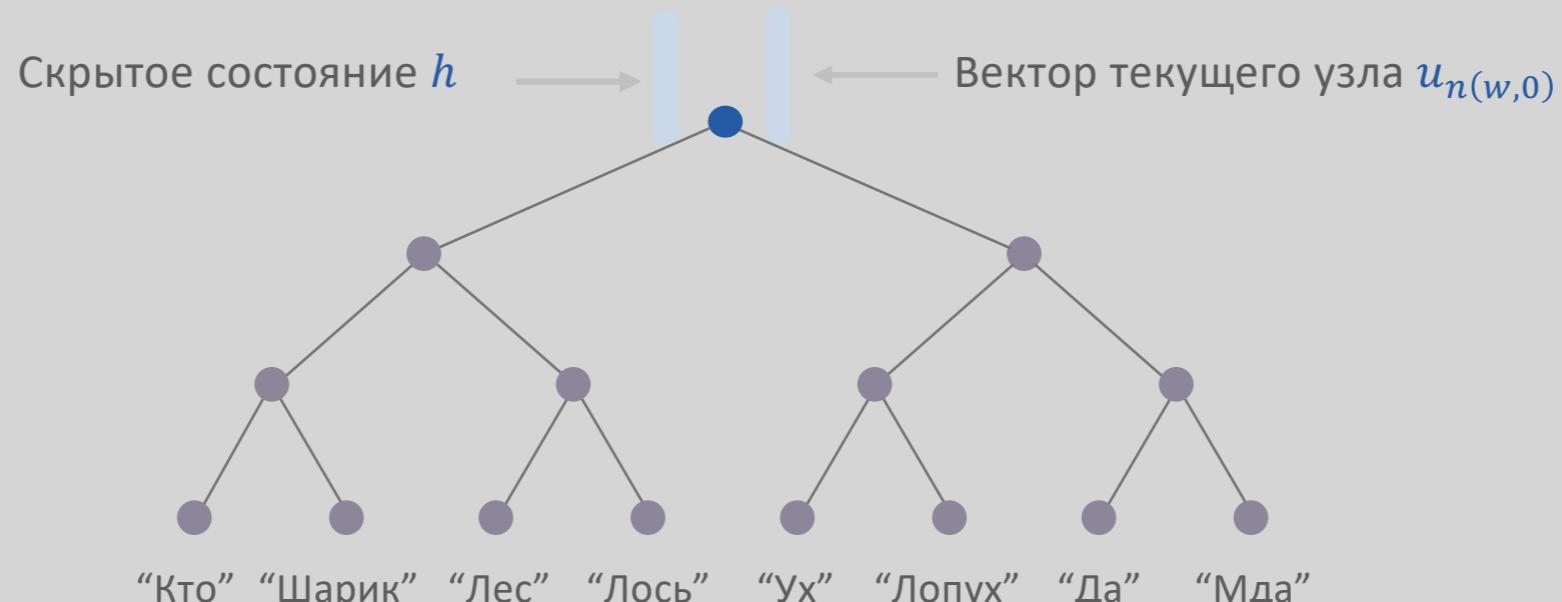
- Заменим второй слой сети на дерево Хаффмана:



# Обучение моделей word2vec

## Иерархический softmax

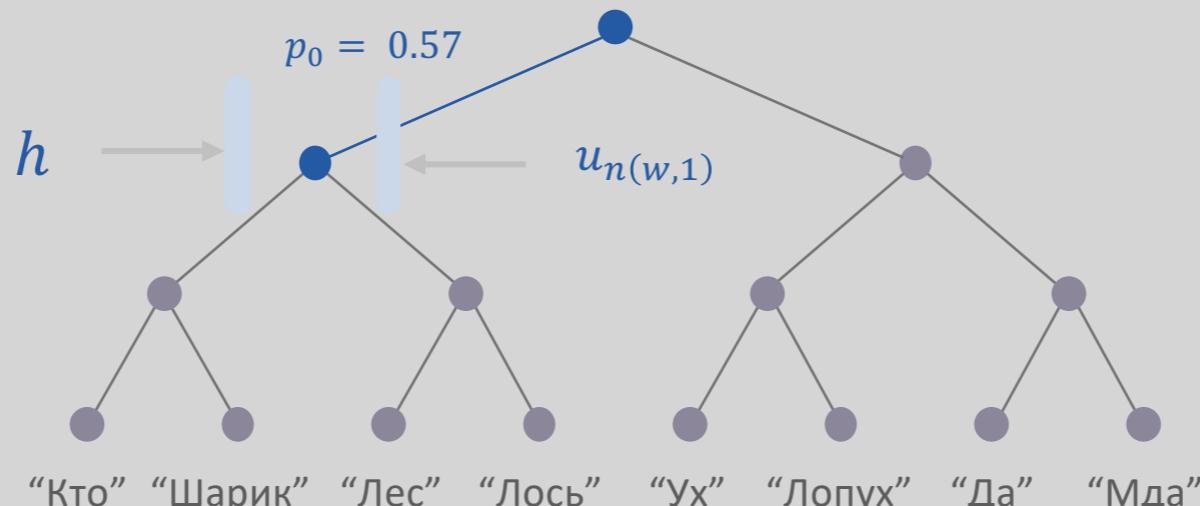
- Пусть мы хотим оценить вероятность слова “лось”
- Вероятность перехода из текущий вершины в поддеревья:
  - $p_0 = \sigma(u_{n(w,0)}^T h)$ , если спускаемся по левой ветке
  - $p_0 = \sigma(-u_{n(w,0)}^T h)$ , если спускаемся по правой ветке



# Обучение моделей word2vec

## Иерархический softmax

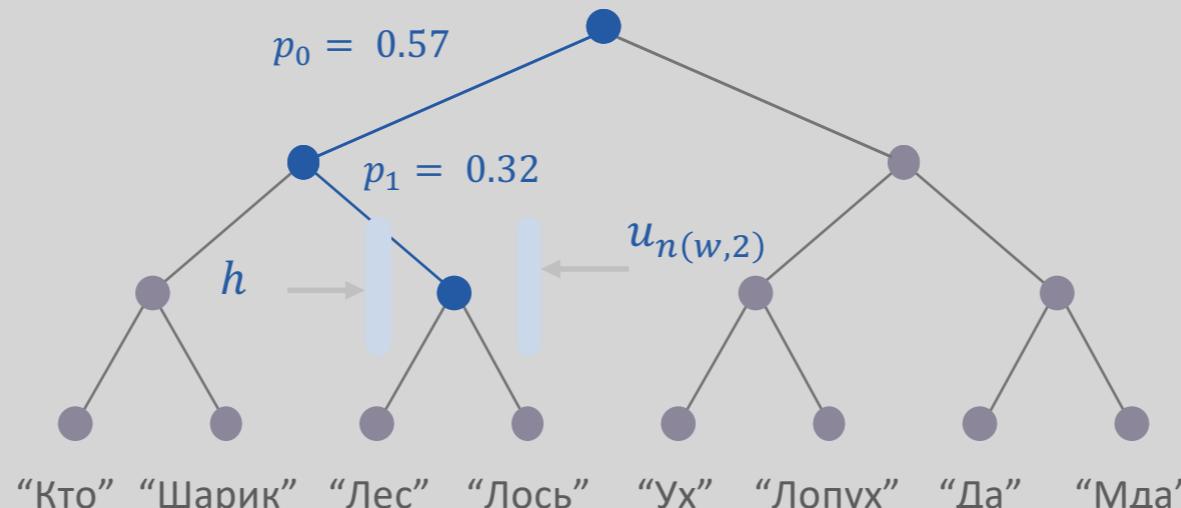
- Вероятность перехода из текущий вершины в поддеревья на первом уровне:
  - $p_0 = \sigma(u_{n(w,1)}^T h)$ , если спускаемся по левой ветке
  - $p_0 = \sigma(-u_{n(w,1)}^T h)$ , если спускаемся по правой ветке



# Обучение моделей word2vec

## Иерархический softmax

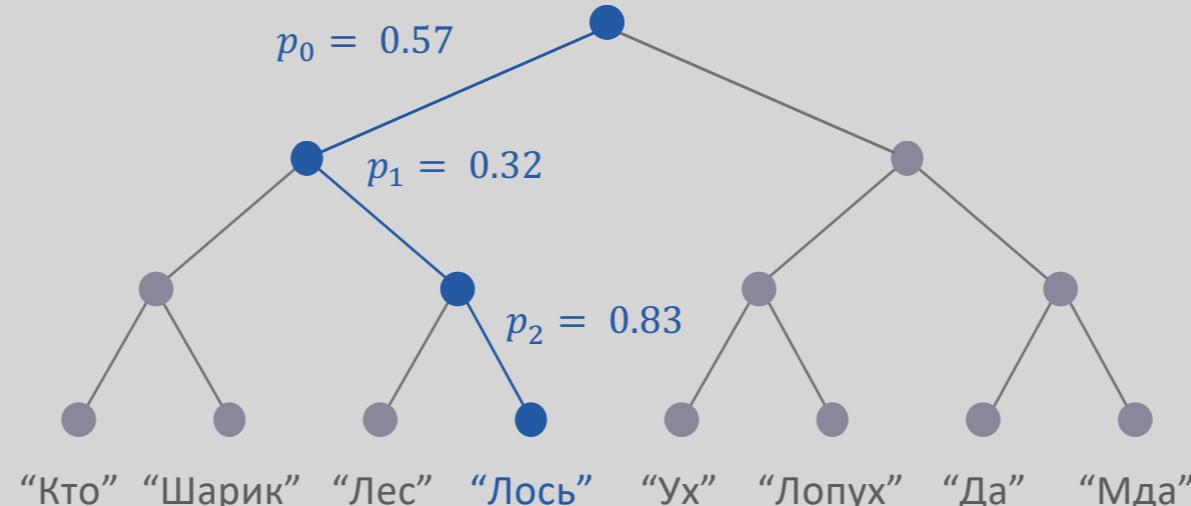
- Вероятность перехода из текущий вершины в поддеревья на втором уровне:
  - $p_0 = \sigma(u_{n(w,2)}^T h)$ , если спускаемся по левой ветке
  - $p_0 = \sigma(-u_{n(w,2)}^T h)$ , если спускаемся по правой ветке



# Обучение моделей word2vec

## Иерархический softmax

- В результате оказались в целевом листе
- Каждый  $i$ -тый спуск происходил с вероятностью  $p_i$
- Итоговая вероятность спуска составляет  $\prod_i p_i$

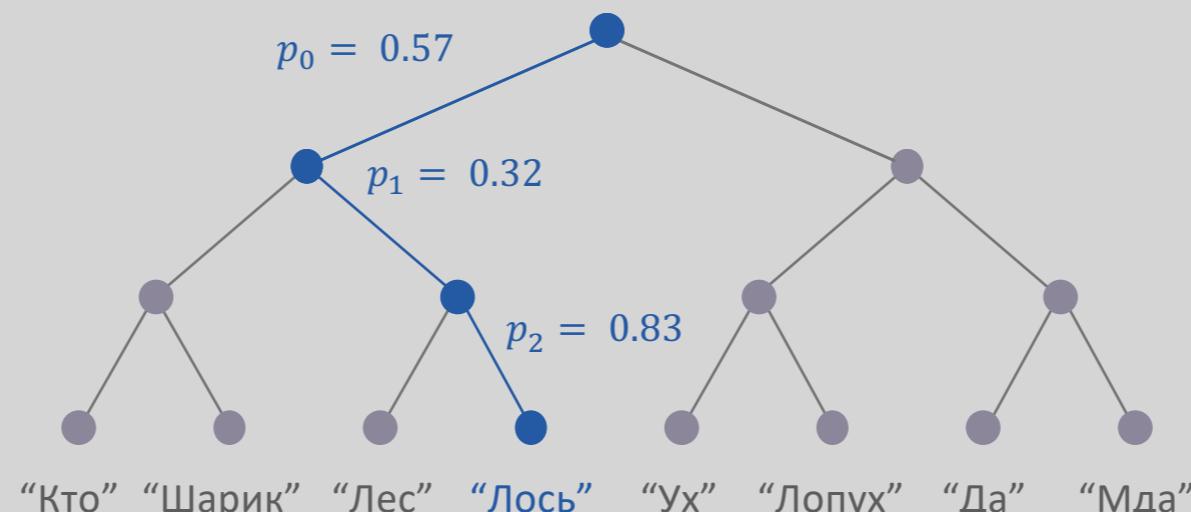


# Обучение моделей word2vec

## Иерархический softmax

- В результате оказались в целевом листе
- Каждый  $i$ -тый спуск происходил с вероятностью  $p_i$
- Итоговая вероятность спуска составляет  $\prod_i p_i$

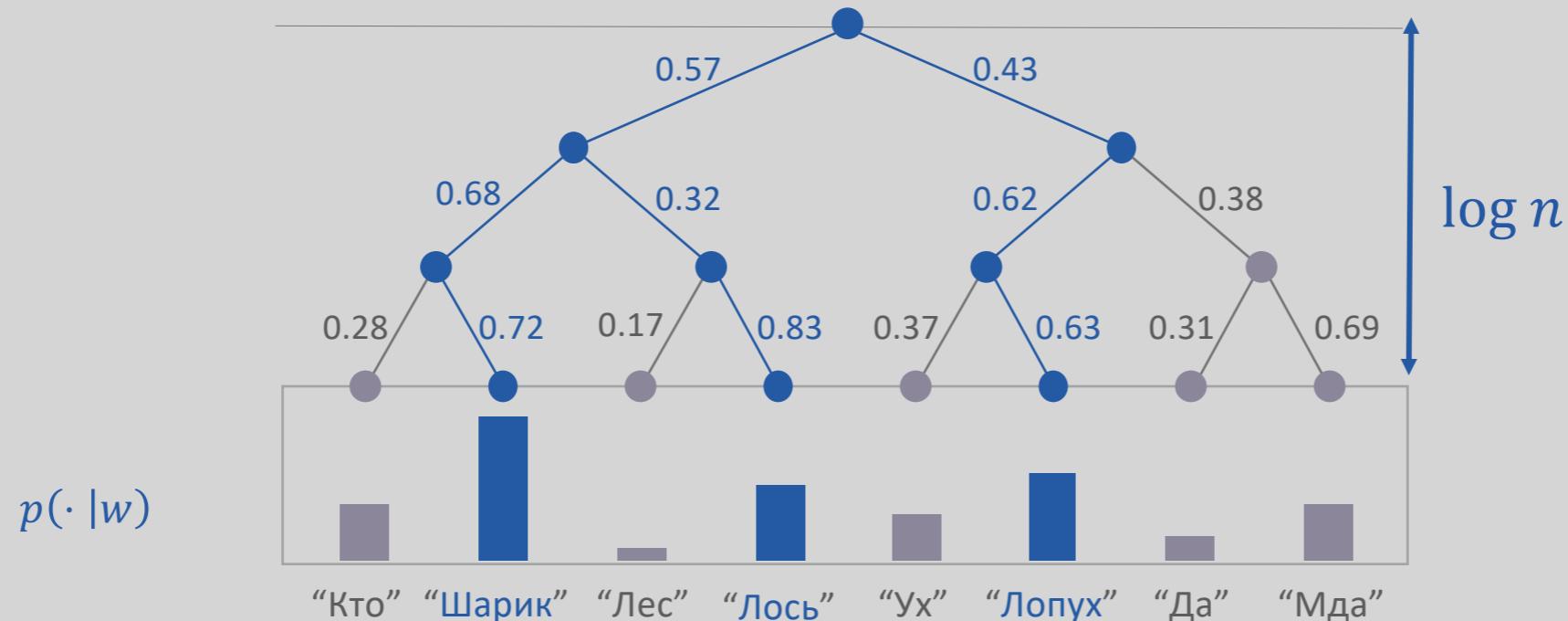
$$p(\text{Лось}|w) = 0.57 \cdot 0.32 \cdot 0.83 \approx 0.15$$



# Обучение моделей word2vec

## Иерархический softmax

- Пройдем по дереву  $2C$  раз, получим все вероятности, необходимые для подсчёта ошибки и настройки весов
- Сложность спуска по дереву:  $O(n \log n)$



# Обучение моделей word2vec

## Иерархический softmax

- Обучение с softmax:  $p(w_O | w_I) = \text{softmax}(V^T h)$
- Из вектора вероятностей  $p$  используем только  $2C$  компонент
- Обучение с иерархическим softmax:

$$p(w_O | w_I) = \prod_{k=1}^L \sigma(I_{turn}(n(w_O, k), n(w_O, k+1))) v_{n(w_O, k)}^T w_{w_I}$$

- Получаем вероятности только нужных слов

# Обучение моделей word2vec

## Skip-gram negative sampling

- Сформулируем задачу skip-gram как задачу бинарной классификации
- Рассмотрим пару слов  $(w, s)$ :
  - Класс 0:  $s$  не входит в контекст  $w$
  - Класс 1:  $s$  входит в контекст  $w$
- Вероятность класса:  $p(1 | (w, s)) = \frac{1}{1 + e^{-v_w^T v_s}} = \sigma(v_w^t, v_s)$

# Обучение моделей word2vec

## Skip-gram negative sampling

- Пусть  $D_1$  – множество наблюдаемых пар  $(w, s)$
- Пусть  $D_0$  – множество ненаблюдаемых пар  $(w, s)$ , для этого выбираем случайные пары слов на каждой итерации обучения
- Функционал правдоподобия:

$$L = \sum_{(w,s) \in D_1} \log \sigma(v_w^T v_s) + \sum_{w,s \in D_0} \log \sigma(-v_w^T v_s)$$

# Модели word2vec

- Ускорение обучения за счет иерархического softmax и negative sampling
- Вариант по умолчанию: SGNS (skip-gram с negative sampling)
- Другие эвристики:
  - Мягкое скользящее окно
  - Снижение веса частых слов
  - Учет биграм

# Neural Word Embedding as Implicit Matrix Factorization

---

**Omer Levy**

Department of Computer Science  
Bar-Ilan University  
omerlevy@gmail.com

**Yoav Goldberg**

Department of Computer Science  
Bar-Ilan University  
yoav.goldberg@gmail.com

## Abstract

We analyze skip-gram with negative-sampling (SGNS), a word embedding method introduced by Mikolov et al., and show that it is implicitly factorizing a word-context matrix, whose cells are the pointwise mutual information (PMI) of the respective word and context pairs, shifted by a global constant. We find that another embedding method, NCE, is implicitly factorizing a similar matrix, where each cell is the (shifted) log conditional probability of a word given its context. We show that using a sparse *Shifted Positive PMI* word-context matrix to represent words improves results on two word similarity tasks and one of two analogy tasks. When dense low-dimensional vectors are preferred, exact factorization with SVD can achieve solutions that are at least as good as SGNS's solutions for word similarity tasks. On analogy questions SGNS remains superior to SVD. We conjecture that this stems from the weighted nature of SGNS's factorization.

# Модель GloVe

# Модель GloVe

## Global vectors

*j*

<i>i</i>	$x_{ij}$		

$$\mathbf{X} \in \mathbb{R}^{n \times n}$$

- Строим матрицу слово–слово
- Компоненты матрицы  $x_{ij}$  показывают, сколько раз слово *i* встретилось в контексте *j*
- Вероятность *j*-того слова в контексте *i*-того:

$$p_{ij} = \frac{x_{ij}}{X_i}, X_i = \sum_j x_{ij}$$

# Модель GloVe

## Global vectors

i	$x_{ij}$		

$$\mathbf{X} \in \mathbb{R}^{n \times n}$$

- Пусть  $v_i$  – векторное представление слова  $i$
- Допустим, что счетчики  $x_{ij}$  нам неизвестны
- Определим функцию, показывающую, какое из слов,  $i$ -ое или  $j$ -ое вероятнее встретить в контексте  $k$ -ого слова

$$F(v_i, v_j, \tilde{v}_k) = \frac{P_{ik}}{P_{jk}}$$

# Модель GloVe

## Global vectors

i

 $x_{ij}$ 

		$x_{ij}$	

$$\mathbf{X} \in \mathbb{R}^{n \times n}$$

- Функция  $F$  должна удовлетворять условиям:

$$F((v_i - v_j)^T v_k) = \frac{F(v_i^T v_k)}{F(v_j^T v_k)} = \frac{P_{ik}}{P_{ij}}$$

- На роль функции  $F$  подойдет  $\exp$
- Отсюда  $F(v_i^T v_k) = P_{ik}$  и  $v_i^T v_k = \log(x_{ik}) - \log(X_i)$

*j*

# Модель GloVe

## Оптимизационная задача

*i*

		$x_{ij}$	

$$\mathbf{X} \in \mathbb{R}^{n \times n}$$

- Оптимизируем следующий функционал

$$\sum_i \sum_k F(x_{ik})(v_i^T v_k + b_i + b_k - \log(x_{ik}))^2 \rightarrow \min_{v_i, b_i, i \in \{1, n\}}$$

- Функция  $F(x_{ik})$  уменьшает вес редких слов
- Положим  $\log(X_i) = b_i + b_k$  для симметрии пары слов в функционале

# Векторное представление слова

## Word2vec VS GloVe

### word2vec

- Модель векторного представления слова
- Одно слово – один вектор
- Проблема OOV слов
- Инкрементальное обучение

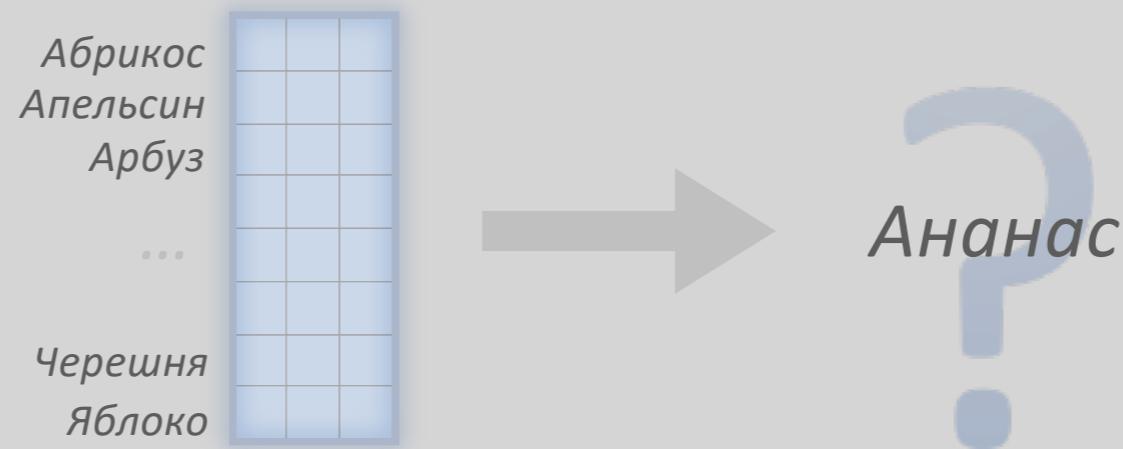
### GloVe

- Модель векторного представления слова
- Одно слово – один вектор
- Проблема OOV слов
- Использует матрицу слово–слово

# Модель FastText

# Проблемы word2vec и GloVe

## 1. Проблема OOV слов



## 2. Не учитывается морфология



# Модель FastText

## Векторное представление N-грамм

- Разобьем слова на последовательности символов – символные N-граммы
- Обучим вектора для каждой N-граммы с помощью СВОУ / Skip-gram
- Вектор слова получим усреднением векторов N-грамм

N-граммы слова «боцман»

N=3: ^бо, боц, оцм, цма, ман, ан\$

N=4: ^боц, боцм, оцма, цман, ман\$

N=5: ^боцм, боцма, оцман, цман\$

# Модель FastText

## Проблема: N-грамм очень много

Hashing trick оптимизирует потребление памяти

- Зафиксируем максимальное число векторов, которые мы хотим обучить
- Сопоставим им хэш-таблицу
- Несколько N-грамм используют один и тот же вектор

# Модель FastText

## Достоинства

- Решается проблема OOV слов
- Инкрементальное обучение

## Недостатки

- Одно слово – один вектор
- Путаем похожие по написанию, но неродственный слова (“спор”, “спорт”, “споры”)

# Качество векторных представлений слов

# Оценки качества векторов слов

## Типы метрик

### Внешние

- Качество решения любой downstream задач

### Внутренние

- Поиск близких слов
- Поиск аналогий
- Поиск лишнего слова

# Внутренние [Intrinsic] меры качества

## Близость слов

- Пары слов и оценки семантической близости между ними
  - WordSim353
  - SimLex-999
- Проверяем, коррелирует ли близость, получаемая с помощью косинусной меры близости между векторами, с оценками, данными людьми

$\cos(\text{абрикос}, \text{персик}) > \cos(\text{абрикос}, \text{маска})$

# Внутренние [Intrinsic] меры качества

## Поиск аналогий

- Между словами  $a$  и  $b$  есть некоторое отношение
- Требуется найти слово  $b^*$ , которое будет находиться в таком же отношении со словом  $a^*$

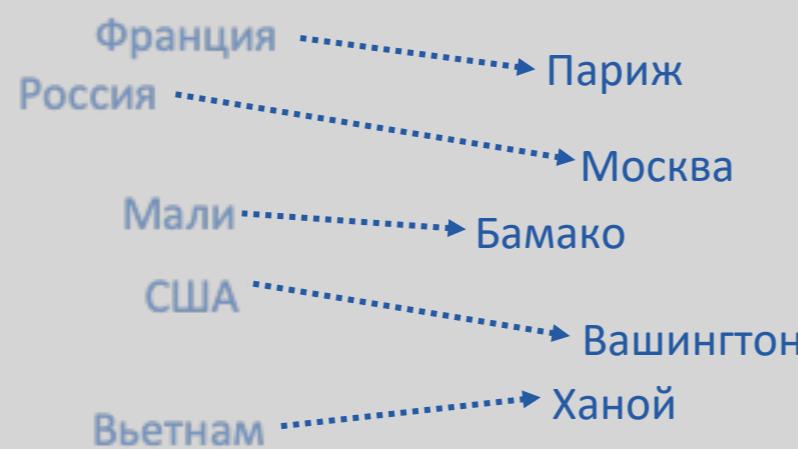
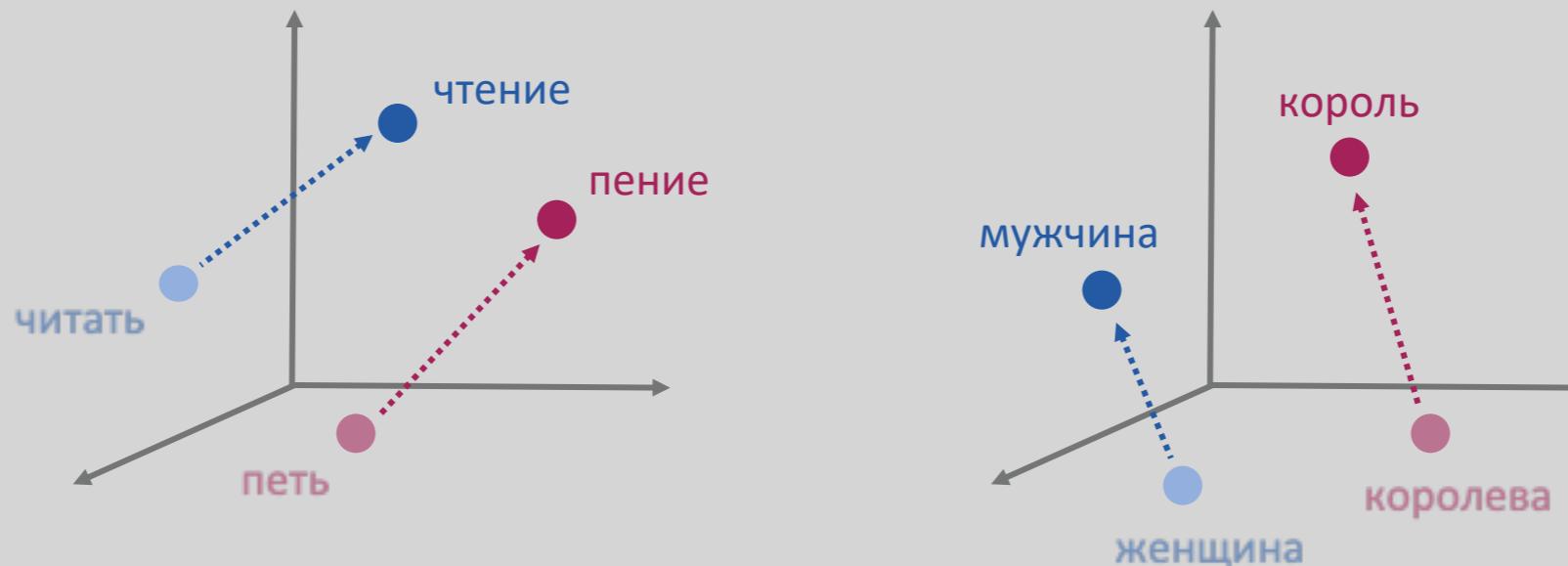
$a = \text{читать}, a^* = \text{чтение}$   
 $b = \text{петь}, b^* = \text{пение}$

$$a^* - a + b \approx b^*$$

чтение — читать + петь ≈ пение

# Внутренние [Intrinsic] меры качества

## Поиск аналогий



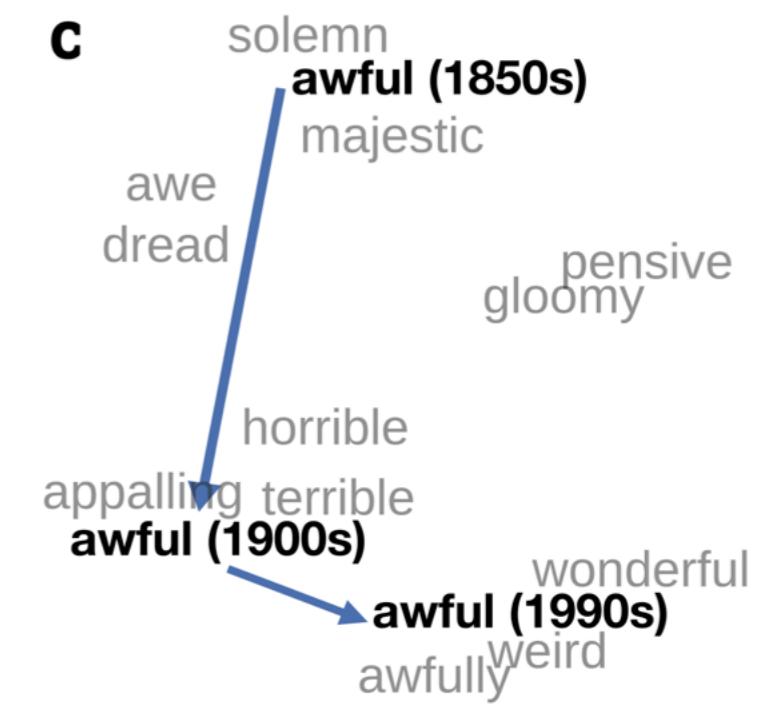
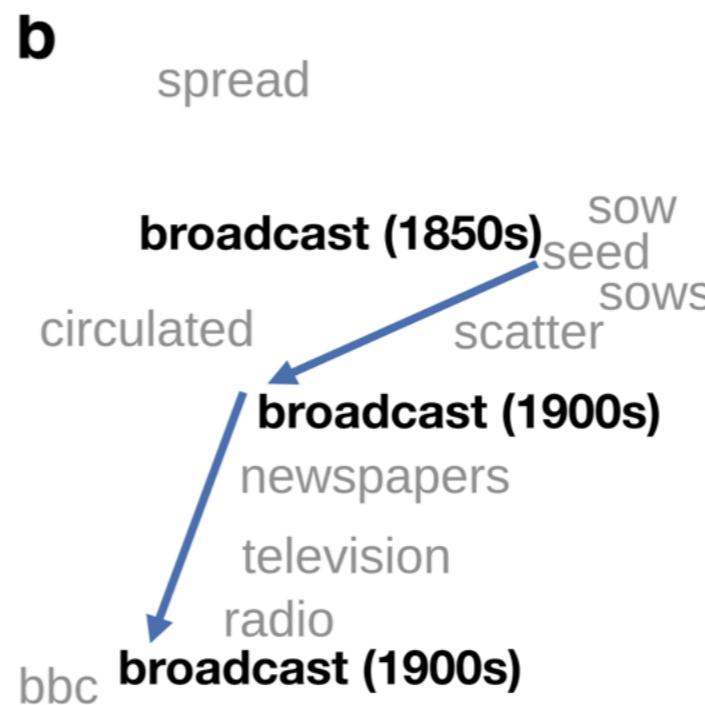
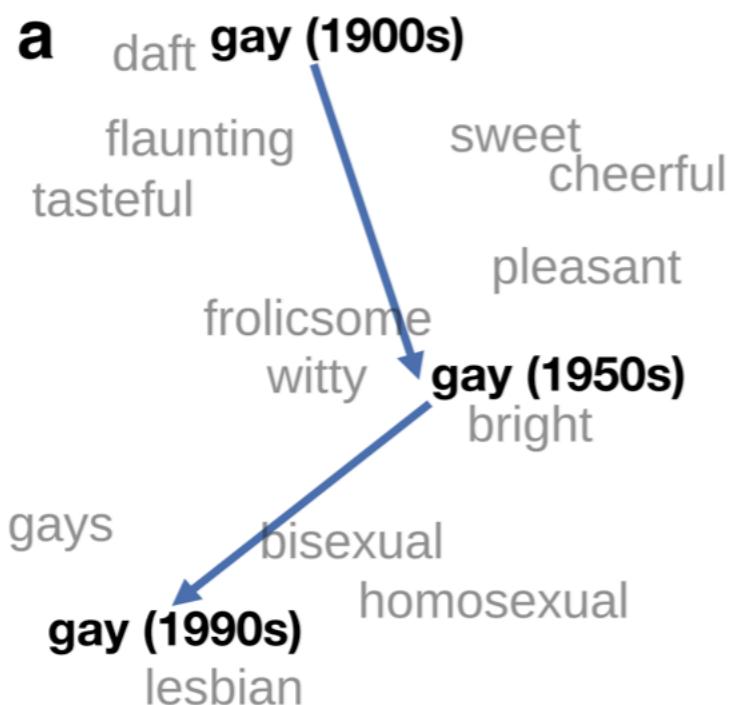
# Внешние [extrinsic] меры качества

## Задача классификации

- Рассмотрим задачу классификации по тональности
- Каждый текст может быть представлен:
  - Усреднением векторов слов
  - Мешком векторов слов
  - Конкатенацией векторов
- Вектора слов могут быть разные
- Если мы зафиксируем классификатор, то оценка качества классификатора будет косвенно показывать и качество модели векторов слов

# Определение семантического сдвига

# Определение семантического сдвига



# Определение семантического сдвига

## Ортогональное преобразование Прокруста

- Пусть дано два корпуса текстов, принадлежащих к периодам времени  $t$  и  $t + 1$
- Обучим две векторные модели,  $W^{(t)} \in \mathbb{R}^{d \times |V|}$  и  $W^{(t+1)} \in \mathbb{R}^{d \times |V|}$
- Выравнивание пространств слов:

$$R^{(t)} = \arg \min_{Q^T Q = I} \|W^{(t)} Q - W^{(t+1)}\|_F$$

- $R^{(t)} \in \mathbb{R}^{d \times d}$  – матрица поворота и растяжения – сохранит попарные косинусов расстояние между словами

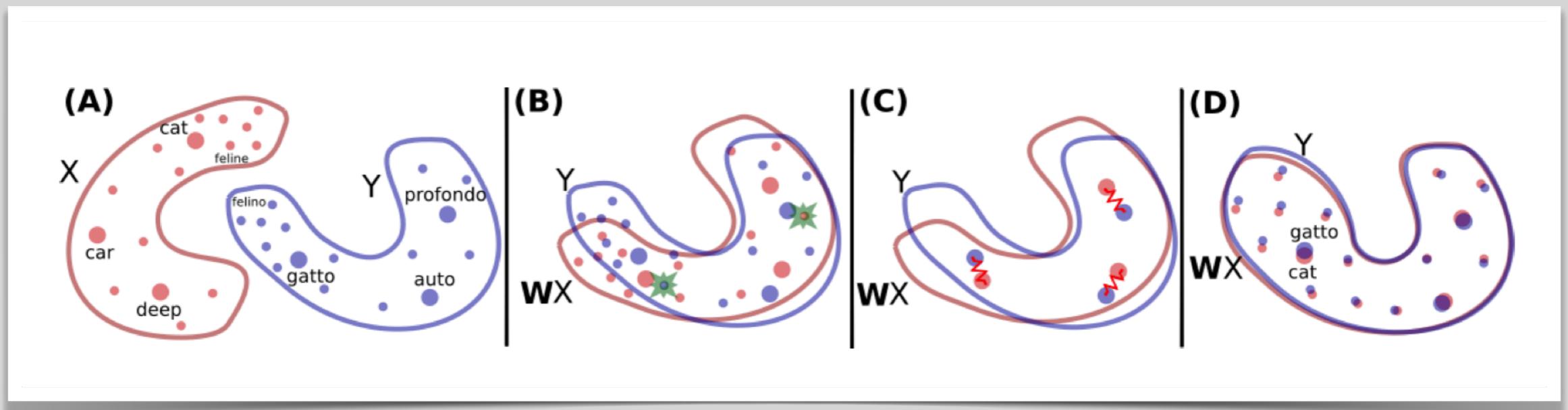
# Определение семантического сдвига

- Как изменился смысл слова?
  - cosine distance( $w_t, w_{t+1}$ )
- Как происходят изменения в языке?
  - $s^{(t)}(w_i, w_j) = \text{cosine similarity}(w_i^{(t)}, w_j^{(t)})$
  - $\rho(s^{(t)}, s^{(t+1)})$  – корреляция между значениями попарной меры близости

# двуязычные векторные представления слов

Hamilton, W.L., Leskovec, J. and Jurafsky, D., 2016, August. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1489-1501).

# Перевод по словам



# Перевод по словам

**Поворачиваем пространство векторов слов на одном языке  
в пространство слов на другом языке в несколько шагов**

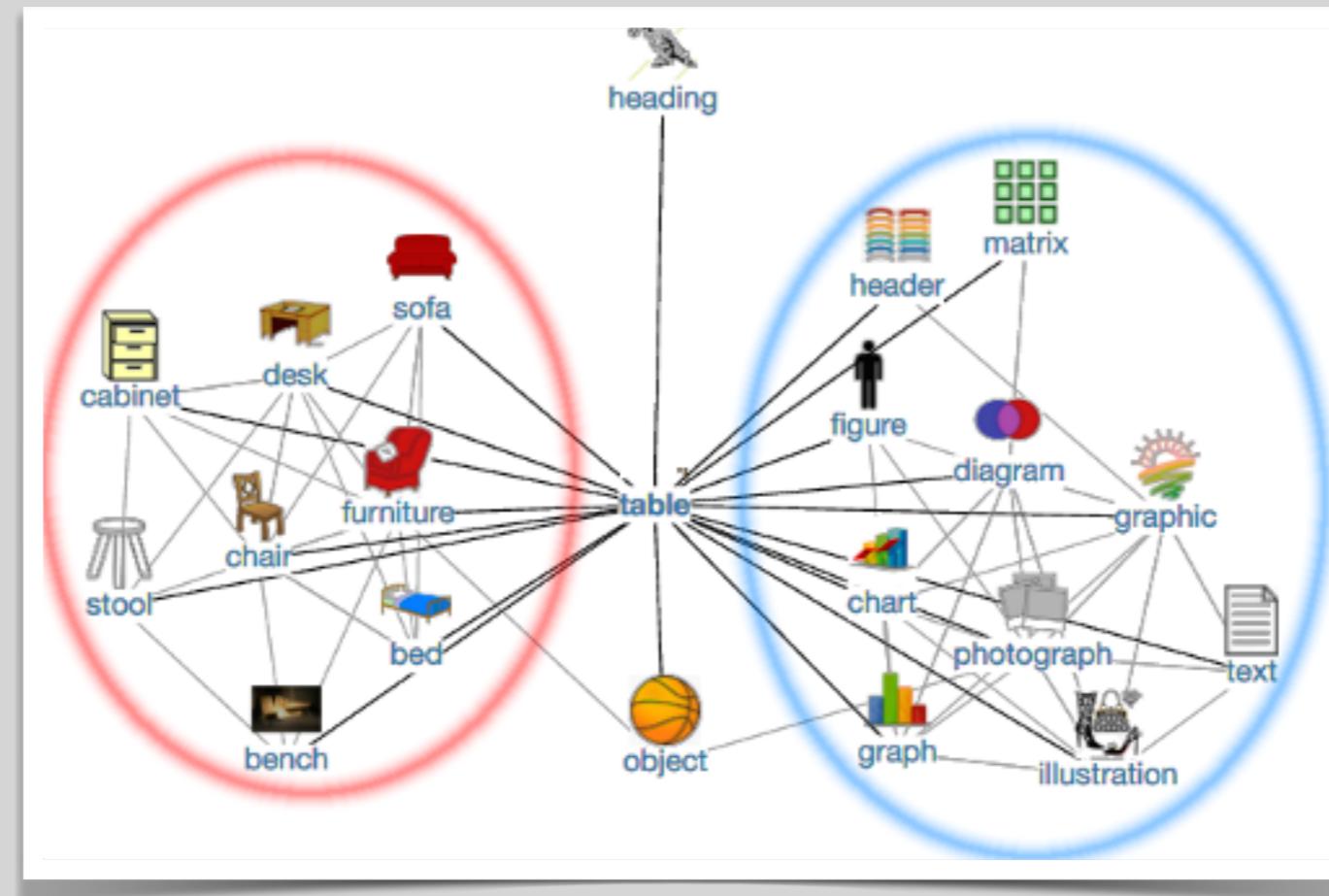
- Шаг А. Задаем независимые пространства векторов слов на каждом языке,  $X$  и  $Y$
- Шаг В. Выбираем опорные точки – несколько пар слов, перевод которых известен
- Шаг С. Используем преобразование Прокруста и получаем приблизительное выравнивание для остальных слов
  - Преобразование Прокруста растягивает и поворачивает пространство  $X$  так, чтобы опорные точки в  $X$  совпали с опорными точками  $Y$
- Шаг D. С помощью метода ближайших соседей доуточняем выравнивание вокруг частых слов.

# Векторное представление многозначных слов

Hamilton, W.L., Leskovec, J. and Jurafsky, D., 2016, August. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1489-1501).

# Разрешение многозначности

## Кластеризация ego-networks слов-соседей



# Нужны ли вектора слов в 2021?...

## GLOVE

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

January  
2, 2014

January  
16, 2013

## WORD2VEC

Word2Vec Paper by Tomas Mikolov et al

July 15,  
2016

## FASTTEXT

Enriching Word Vectors with Subword Information by Piotr Bojanowski et al

## TRANSFORMER

Attention Is All You Need by Ashish Vaswani et al

June 12,  
2017

## ELMO

Deep contextualized word representations by Matthew E. Peters et al

## BERT

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

October  
11, 2018