



# Лекция 7. Обработка текстовых данных

Господинов Георгий



# Вступайте в ODS



- 45k+ участников
- 6M+ сообщений
- Обсуждение соревнований, статей, фреймворков и моделей, помочь в составлении резюме, совместное прохождение курсов, публикации вакансий и многое другое

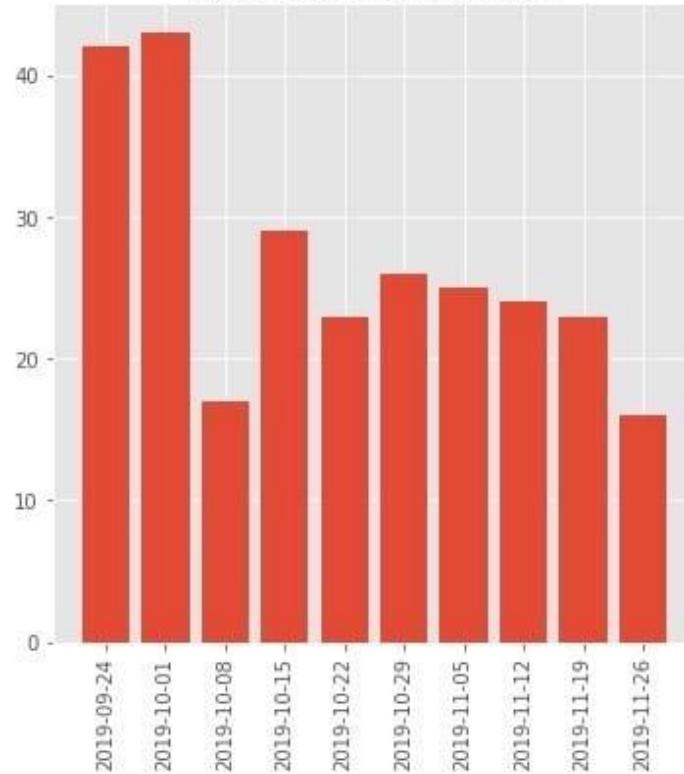


## Open Data Science

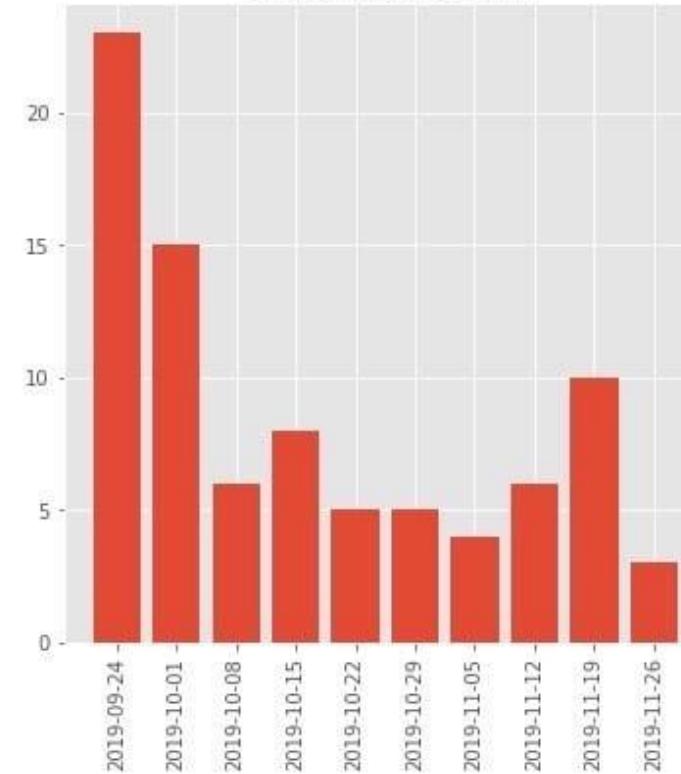
Заходите на [ods.ai](https://ods.ai) и оставляйте заявку!

# Статистика обратной связи по курсу (2019 год)

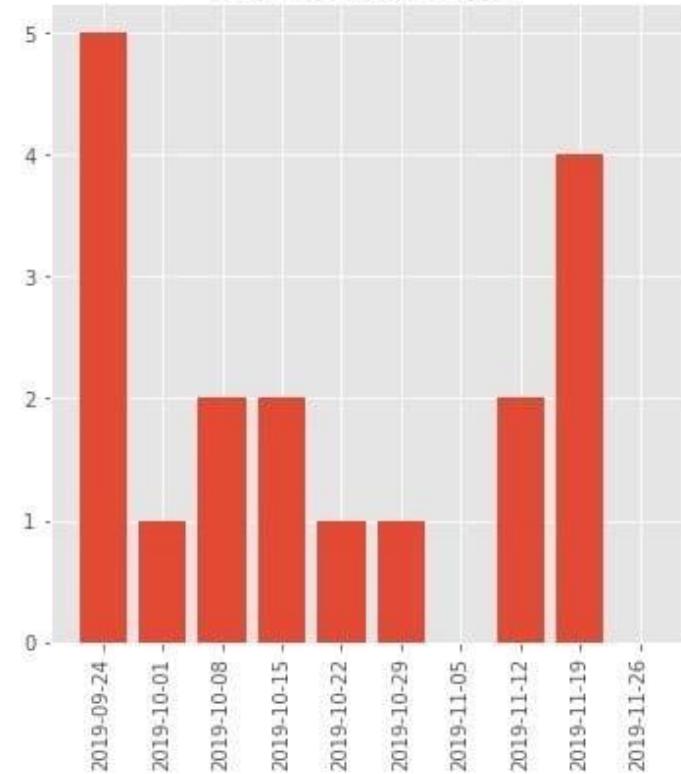
Количество слушателей



Количество оценок



Количество отзывов



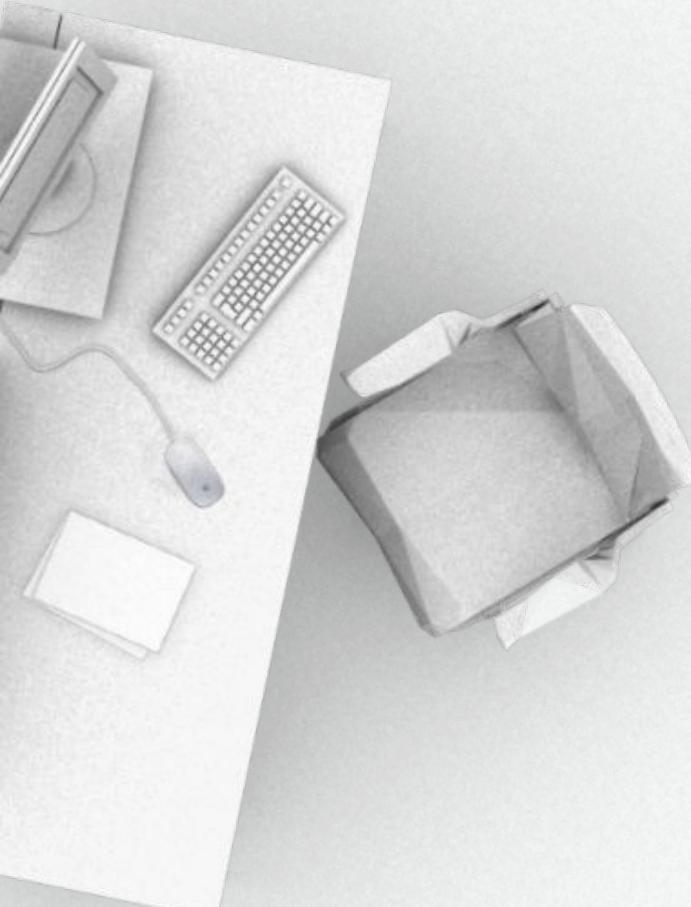
Что произошло 2019-11-19?

Не забудьте отметиться на портале!

---

# План занятия

- 1.** Рассказать план
- 2.** Примеры задач
- 3.** Ручное извлечение признаков
- 4.** Специфика текстовых данных
- 5.** Методы предобработки текстов
  - пунктуация
  - морфология
  - регистр
  - стоп-слова
- 6.** Выделение признаков
  - word / character n-gram, collocations
  - one-hot / counts / tf-idf
- 7.** Эмбеддинги слов
- 8.** Тематическое моделирование
- 9.** Домашнее задание



## Примеры задач

# Задачи

---

## Вспомогательные лингвистические задачи

- Морфологический разбор (приведение слов к нормальной форме или корню)
- Частеречная разметка (POS-tagging, part-of-speech tagging)
- ...

## Базовые ML задачи

- Классификация текста
- Извлечение именованных сущностей
- Синтез текста
- Поиск похожих текстов
- Кластеризация текста
- ....

## Бизнес-задачи / Сервисы

- информационный поиск
- машинный перевод
- чат-боты
- ...

# Классификация текста

- фильтрация враждебных высказываний (hate speech detection)
  - [press release vk](#)
- фильтрация спама
  - иногда для таких задач нужен OCR (optical character recognition)
- фильтрация порнографии
  - Kaspersky Safe Kids
  - модерация контента перед его рекламой
- категоризация контента
  - проставление тегов к статьям (актуально, например, для yandex.zen)
  - создание каталога товаров (например, [каталог](#) yandex.market)
- анализ тональности текста (sentiment analysis)
  - какие отзывы оставляют пользователи?
  - в каком настроении пользователь?



# Извлечение именованных сущностей (Named Entity Recognition, NER)

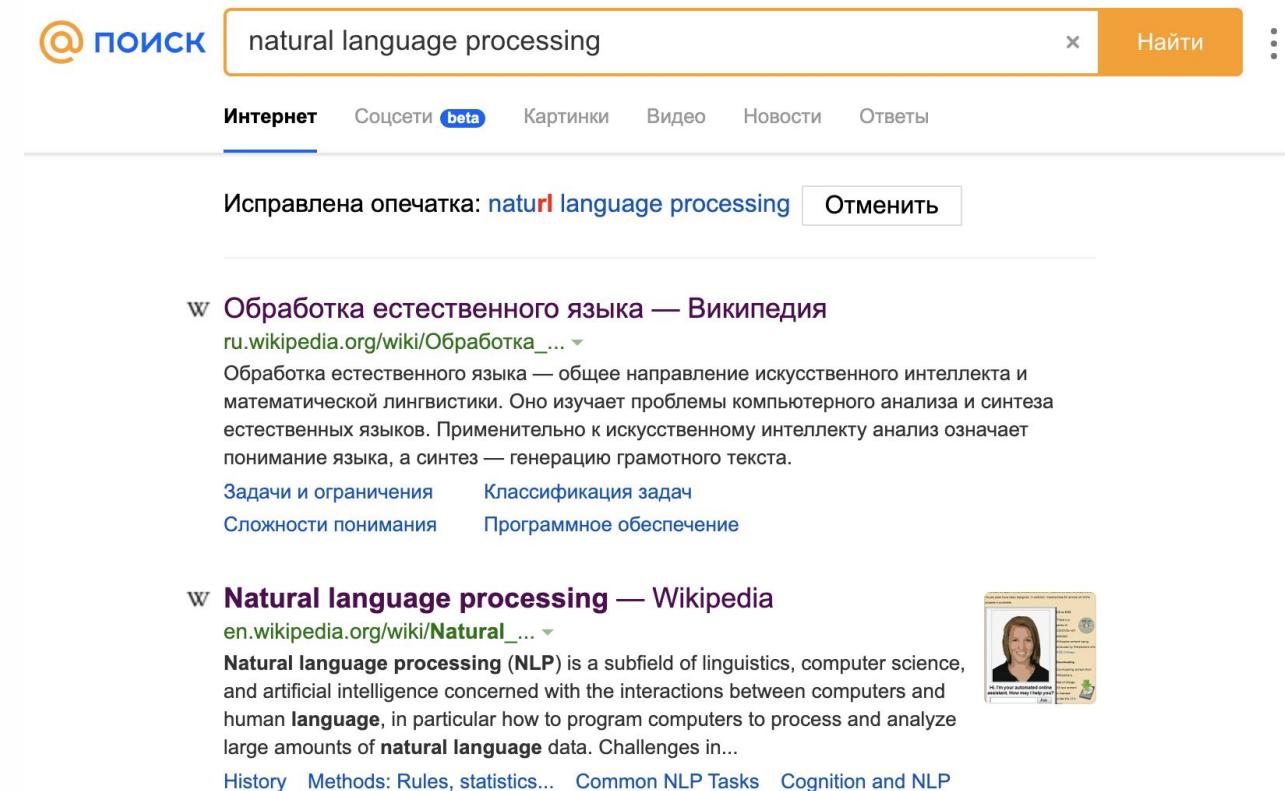
- выделение в тексте дат, адресов, организаций
- может использоваться для структуризации неструктурированных данных (выделение сущностей и их запись в базу)
- вспомогательный этап для многих задач (например, вопросно-ответные системы)

Так говорила в июле 1805 года известная Анна Павловна Шерер, фрейлина и приближенная императрицы Марии Феодоровны, встречая важного и чиновного князя Василия, первого приехавшего на ее вечер. Анна Павловна кашляла несколько дней, у нее был грипп, как она говорила (грипп был тогда новое слово, употреблявшееся только редкими).

- Физ. лица
- Организации
- События
- Гео-объекты
- Объекты времени
- Денежные единицы
- Бренды

# Информационный поиск (Information Retrieval)

- построение поискового индекса
  - по каким документам искать?
- поисковые подсказки
- исправление опечаток
- многоуровневое ранжирование документов по короткому запросу
- генерация снippetов
- ...



The screenshot shows a search engine interface with the following details:

- Search Bar:** natural language processing
- Navigation Tabs:** Интернет (selected), Соцсети (beta), Картинки, Видео, Новости, Ответы
- Text Overlay:** Исправлена опечатка: naturl language processing [Отменить](#)
- Result 1:** w Обработка естественного языка — Википедия  
[ru.wikipedia.org/wiki/Обработка\\_естественного\\_языка](http://ru.wikipedia.org/wiki/Обработка_естественного_языка)  
Обработка естественного языка — общее направление искусственного интеллекта и математической лингвистики. Оно изучает проблемы компьютерного анализа и синтеза естественных языков. Применительно к искусственноому интеллекту анализ означает понимание языка, а синтез — генерацию грамотного текста.  
Задачи и ограничения    Классификация задач  
Сложности понимания    Программное обеспечение
- Result 2:** w Natural language processing — Wikipedia  
[en.wikipedia.org/wiki/Natural\\_language\\_processing](http://en.wikipedia.org/wiki/Natural_language_processing)  
Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data. Challenges in...  
History   Methods: Rules, statistics...   Common NLP Tasks   Cognition and NLP

# Машинный перевод (Machine Translation)

- определение языка
- перевод
- ранжирование кандидатов
- text-to-speech (TTS)
- ...

≡ Google Переводчик

⋮ ⓰

РУССКИЙ (ОПРЕДЕЛЕН АВТОМАТИЧЕСКИ) ↔ АНГЛИЙСКИЙ РУССКИЙ

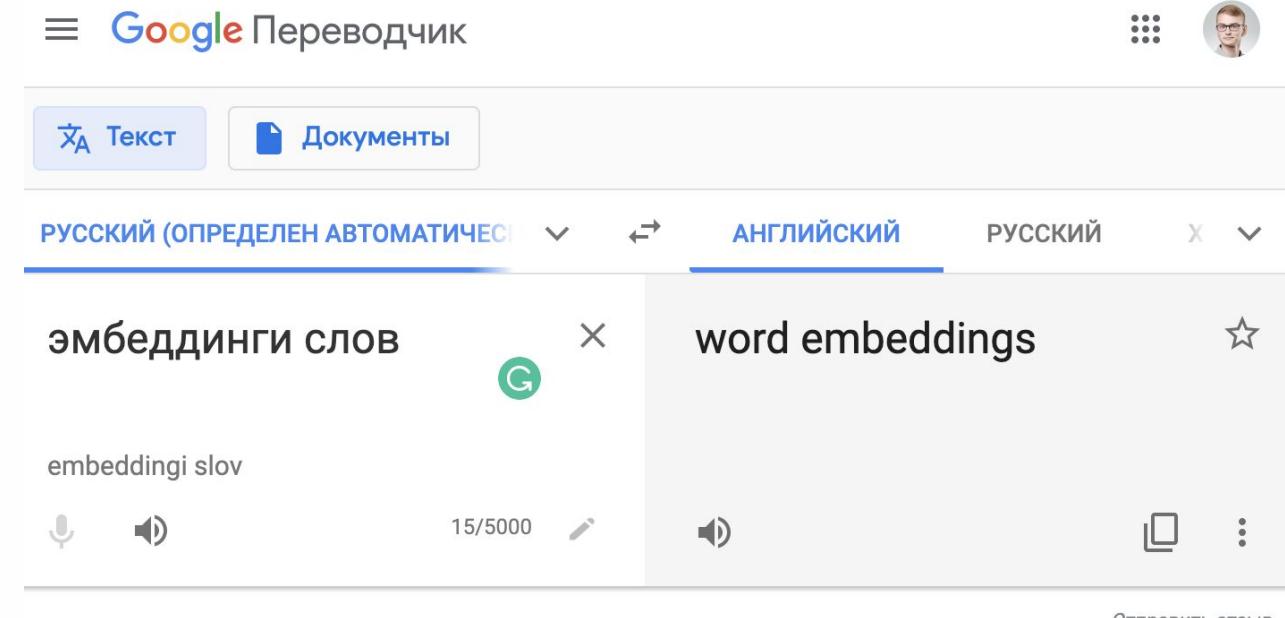
текст документы

эмбеддинги слов word embeddings

embeddingi slov

15/5000

Отправить отзыв



# Диалоговые системы (Conversation AI)

- speech-to-text
- анализ текста
  - определение настроения
  - поддержка диалога
  - ответы на вопросы (QA)
    - Какая погода завтра?
    - Первый президент США?
- text-to-speech

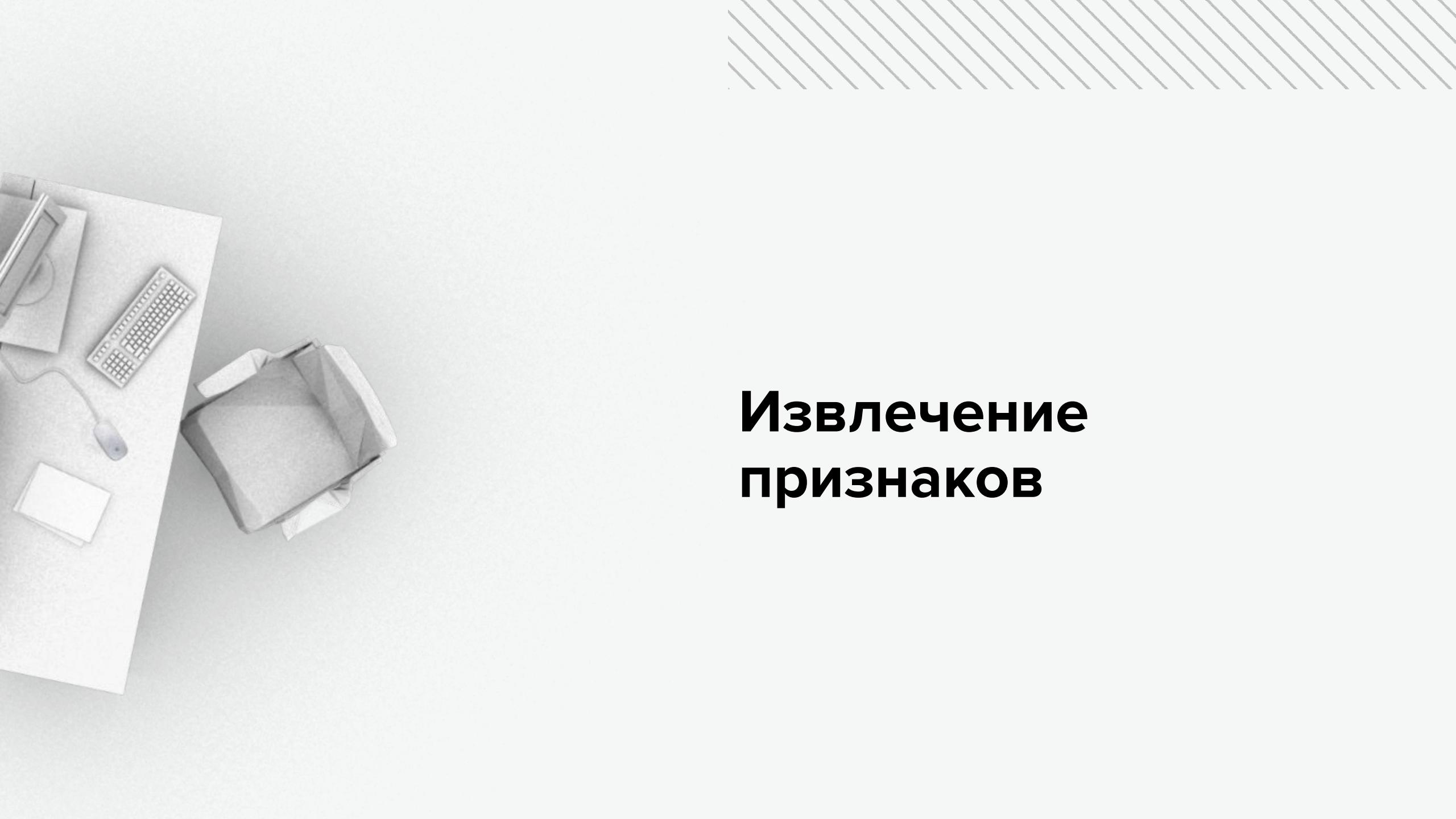


[Her](#) (2013)



Replika

рассказ про современную диалоговую систему [replika.ai](http://replika.ai): [Artem Rodichev, Replika](#)



# **Извлечение признаков**

# Составление признаков (Handcrafted features)

текст	метка класса
Отвратительный перевод. Такое ощущение, будто весь текст анг. оригинала без разбора прогнали через гугл транслейт. В начале думал приложить скрины мест со смешным (в плохом смысле) переводом, но перехотелось. Легче смело отсканировать и вставить всю книжку целиком.  Оригинал - рекомендую. На "это" даже не тратьте свои деньги.	0
Отличное чтение доступным языком. Самое то для начинающих осваивать машинное обучение	1

# Составление признаков (Handcrafted features)

текст	метка класса
Отвратительный перевод. Такое ощущение, будто весь текст анг. оригинала без разбора прогнали через гугл транслейт. В начале думал приложить скрины мест со смешным (в плохом смысле) переводом, но перехотелось. Легче смело отсканировать и вставить всю книжку целиком.  Оригинал - рекомендую. На "это" даже не тратьте свои деньги.	0
Отличное чтение доступным языком. Самое то для начинающих осваивать машинное обучение	1

- длина (возможно, длинные отзывы в среднем хуже)
- посчитать смайлики :), :(
- посчитать количество восклицательных знаков
- можно составить словарь хороших слов: {люблю, обожаю, нравится, идеально, ...}
- признак: количество хороших слов
- словарь плохих слов: {бесит, невыносимо, дрянь, плохой, ужасный, ...}

# Составление признаков

Составление словарей требует ручного труда и не гарантирует хорошего результата

Другой подход: в качестве признаков брать отдельные слова и обучать на них модели

	перевод	такое	...	осваивать	машинное	обучение	метка класса
text_1	1	1	...	0	0	0	0
text_2	0	0	...	1	1	1	1

# Составление признаков

Составление словарей требует ручного труда и не гарантирует хорошего результата

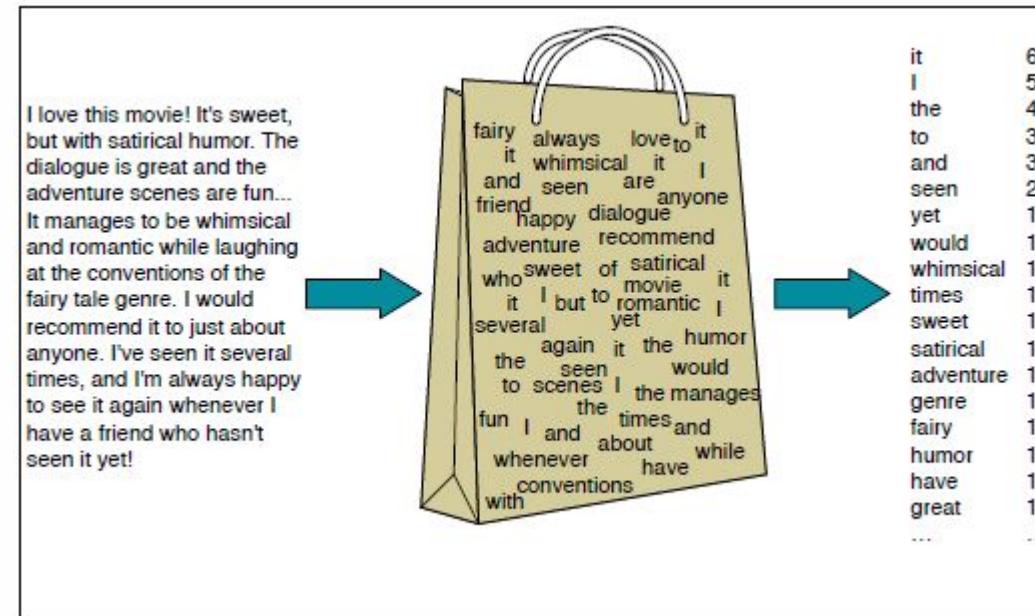
Другой подход: в качестве признаков брать отдельные слова и обучать на них модели

	перевод	такое	...	осваивать	машинное	обучение	метка класса
text_1	1	1	...	0	0	0	0
text_2	0	0	...	1	1	1	1

Например, если строим логистическую регрессию:

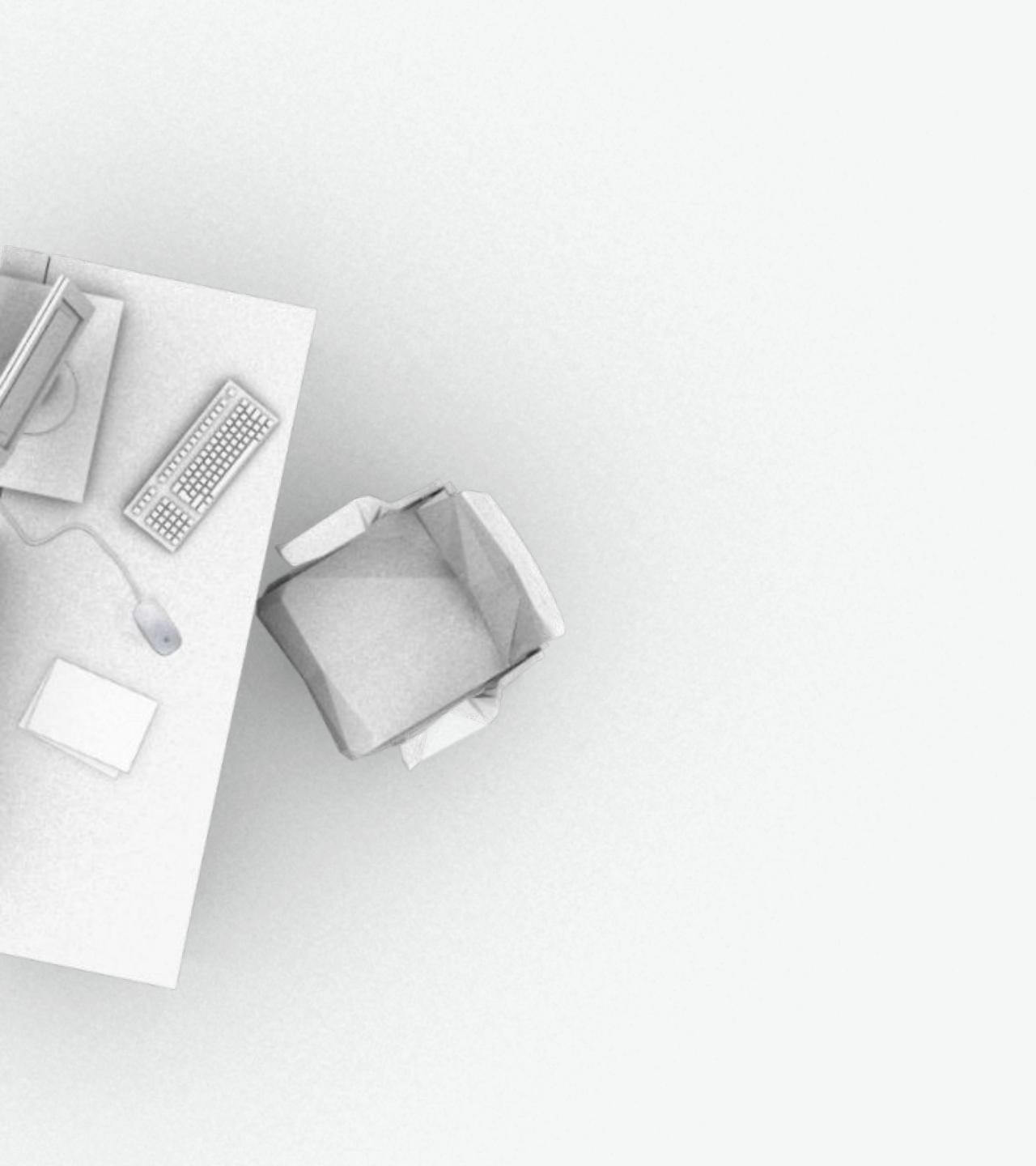
$$p(y = 1|x_i) = \sigma\left(b + \sum_j w_j x_{ij}\right) = \sigma(b + w_1 \cdot \text{count}_i(\text{"перевод"}) + w_2 \cdot \text{count}_i(\text{"такое"}) + \dots + w_n \cdot \text{count}_i(\text{"обучение"}))$$

# Модель мешка слов (bag of words, BoW)



	перевод	такое	...	осваивать	машинное	обучение	метка класса
text_1	1	1	...	0	0	0	0
text_2	0	0	...	1	1	1	1

какую информацию мы потеряли при переходе от исходного текста к такой матрице?



# **Извлечение слов из текста**

# Токенизация

Если хотим в качестве признаков использовать отдельные слова, то нужно как-то разбивать текст на слова

Процесс разбиения текста на слова (токены / термы / униграммы) называется токенизацией

```
import re
import razdel
import nltk

sentence = "Как же так?! Олег... Мы же в 18.00 договаривались встретиться:("

sentence.split()
# ['Как', 'же', 'так?!', 'Олег...', 'Мы', 'же', 'в', '18.00', 'договаривались', 'встретиться:(']

re.findall("[а-яА-ЯёЁ]+", sentence)
# ['Как', 'же', 'так', 'Олег', 'Мы', 'же', 'в', 'договаривались', 'встретиться']

re.split(r"[-\s.,;!?]+", sentence) # \s – все пробельные символы
# ['Как', 'же', 'так', 'Олег', 'Мы', 'же', 'в', '18', '00', 'договаривались', 'встретиться:(']

nltk.tokenize.casual_tokenize(sentence)
# ['Как', 'же', 'так', '?', '!', 'Олег', '...', 'Мы', 'же', 'в', '18.00', 'договаривались', 'встретиться', ':(']

[token.text for token in razdel.tokenize(sentence)]
# ['Как', 'же', 'так', '?!', 'Олег', '...', 'Мы', 'же', 'в', '18.00', 'договаривались', 'встретиться', ':(']
```

Сравнение токенизаторов для русского языка: [razdel#evaluation](#)

# Выравнивание регистра, нормализация по регистру (case folding, case normalization)

```
sentence = "Как же так?! Олег... Мы же в 18.00 договаривались встретиться:( "
```

Термы “как” и “мы” встретились в начале предложения => написаны с большой буквы.  
Имеет смысл приводить к нижнему регистру, поскольку их функция не изменилась

# Выравнивание регистра, нормализация по регистру (case folding, case normalization)

sentence = "Как же так?! Олег... Мы же в 18.00 договаривались встретиться:( "

Термы “как” и “мы” встретились в начале предложения => написаны с большой буквы.  
Имеет смысл приводить к нижнему регистру, поскольку их функция не изменилась



!=



!=



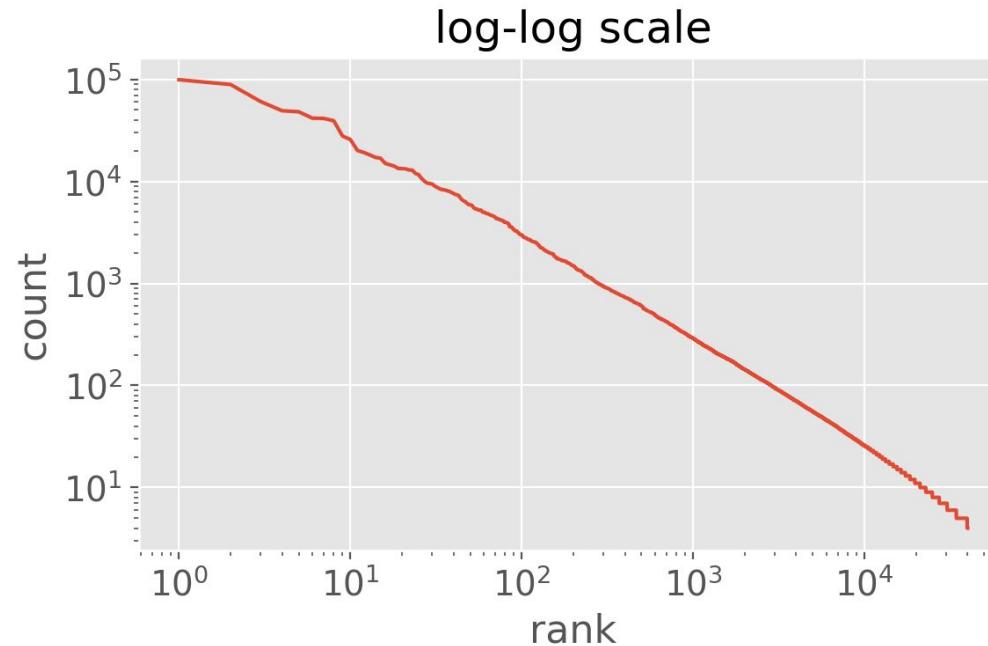
Зависит от задачи. Если решаете NER, то просто .lower() везде вставлять не стоит

# Частотные закономерности: закон Ципфа

rank	token	count
1	и	99066
50	был	5896
100	со	2982
150	вопрос	1951
200	использовать	1491
250	наш	1140
300	никто	938

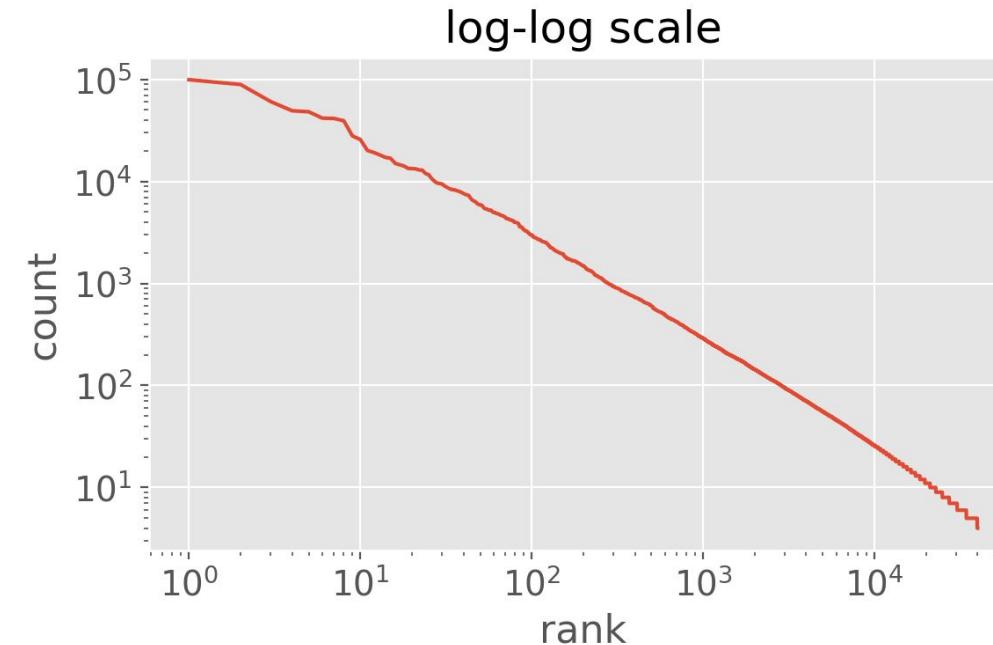
# Частотные закономерности: закон Ципфа

rank	token	count
1	и	99066
50	был	5896
100	со	2982
150	вопрос	1951
200	использовать	1491
250	наш	1140
300	никто	938



# Частотные закономерности: закон Ципфа

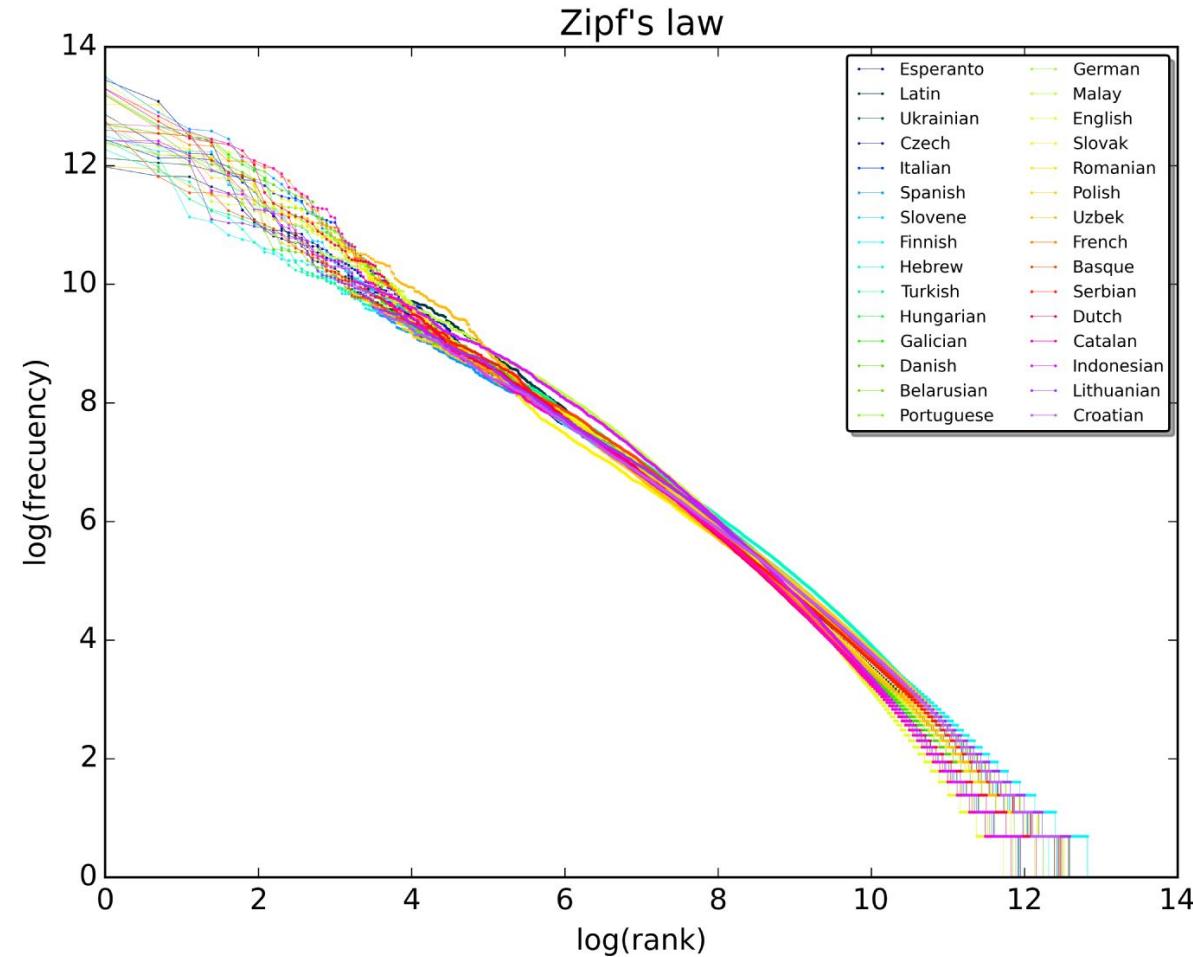
rank	token	count
1	и	99066
50	был	5896
100	со	2982
150	вопрос	1951
200	использовать	1491
250	наш	1140
300	никто	938



Математически описывается распределением Парето. Распределение с тяжелым хвостом: если хотите выучить иностранный язык, первым делом учите самые распространенные слова

# Закон Ципфа

Справедлив не только для разных языков, но и за пределами текстов.  
Например, [популяции городов США](#)



wikipedia: [Zipf's law](#)

---

# **Какие слова самые частотные?**

# Какие слова самые частотные?

word	count
и	99066
в	88906
что	60461
я	49177
это	48070
на	41620
не	41373
мы	39265
с	27817

Самые частые слова в корпусе — стоп-слова: служебные части речи (предлоги, союзы, частицы). Также местоимения

Они есть почти везде и не добавляют полезной информации в модель, их имеет смысл пробовать отбрасывать

Если анализируете тональность, нужно быть аккуратным с отрицательными частицами

```
import nltk  
  
nltk.corpus.stopwords.words('russian')  
# ['и', 'в', 'во', 'не', 'что', 'он', 'на', 'я', 'с', 'ко', ...]
```

# Нормализация слов

За словоизменение и словообразование отвечает морфология. У разных языков она разная. Например, русский — флексивный (лат. *flectivus* — гибкий): слова существенно изменяются для согласования друг с другом

Если разбивать текст на слова и применять только те преобразования, которые мы рассмотрели до этого, то слова “студент”, “студента”, “студенту”, будут разными признаками в модели

# Стемминг

Стемминг — поиск общей основы ([стема](#)) различных форм слова путем отбрасывания суффиксов, окончаний

стемминг — сложная задача:

- ending => end
- running => run (нужно не просто удалить ing !)
- sing => sing

отбрасывание множественного числа:

- words => word
- но надо учесть bus, lens

[стеммер Портера](#) и [Snowball Stemmer](#)

<https://github.com/jedijulia/porter-stemmer/>

# Лемматизация

Лемматизация — приведение слов к словарной форме (лемме): для существительного - ед. число, им. падеж; для глагола - инфинитив; ...

- бегущий => бежать
- люди => человек
- рой => рыть
- рой => рой

[pymorphy2](#), [mystem](#)

# Лемматизация

Лемматизация — приведение слов к словарной форме (лемме): для существительного - ед. число, им. падеж; для глагола - инфинитив; ...

- бегущий => бежать
- люди => человек
- рой => рыть
- рой => рой

## pymorphy2, mystem

```
import pymorphy2

lemmatizer = pymorphy2.MorphAnalyzer()

lemmatizer.parse('стекла')

[Parse(word='стекло', tag=OpencorporaTag('NOUN,inan,neut sing,nomn'), normal_form='стекло', score=0.75,
methods_stack=((<DictionaryAnalyzer>, 'стекло', 545, 0),)),
 Parse(word='стекло', tag=OpencorporaTag('NOUN,inan,neut sing,accs'), normal_form='стекло', score=0.1875,
methods_stack=((<DictionaryAnalyzer>, 'стекло', 545, 3),)),
 Parse(word='стекло', tag=OpencorporaTag('VERB,perf,intr neut,sing,past,indc'), normal_form='стечь', score=0.0625,
methods_stack=((<DictionaryAnalyzer>, 'стекло', 968, 3),))]
```

# Лемматизация

Лемматизация — приведение слов к словарной форме (лемме): для существительного - ед. число, им. падеж; для глагола - инфинитив; ...

- бегущий => бежать
- люди => человек
- рой => рыть
- рой => рой

## pymorphy2, mystem

```
import pymorphy2

lemmatizer = pymorphy2.MorphAnalyzer()

lemmatizer.parse('хендай')

[Parse(word='хендай', tag=OpencorporaTag('VERB,perf,tran sing,impr,excl'), normal_form='хендать',
score=0.2548476454293629, methods_stack=((<DictionaryAnalyzer>, 'дай', 843, 13), (<UnknownPrefixAnalyzer>, 'хен'))),
...]
```

# Стемминг vs Лемматизация

Лемматизация — более вычислительно сложный процесс

Стемминг неплохо работает для английского языка, но не идеально:

- university => univers
- universe => univers

Для русского ситуация хуже

- буря => бур
- бурить => бур

---

## **TF-IDF: term frequency, inverse document frequency**

До этого мы рассматривали разбиение текста на слова и подсчет их количества

Какие недостатки есть у такого подхода?

## TF-IDF: term frequency, inverse document frequency

До этого мы рассматривали разбиение текста на слова и подсчет их количества

Какие недостатки есть у такого подхода?

Альтернативный вариант: для каждого слова  $w$  документа  $d$  в коллекции документов  $D$  рассчитывать tf-idf:

$$\text{tf}(w, d) = \frac{\text{count}(w)}{|d|}$$

$$\text{idf}(w, D) = \log \frac{|D|}{\{d \in D | w \in d\}}$$

$$\text{tf-idf}(w, d, D) = \text{tf}(w, d) \times \text{idf}(w, D)$$

---

## TF-IDF: нормировка term frequency

В каком из двух текстов слово “собака” встретится чаще?

1. записка ветеринару
2. Война и Мир

В каком из этих текстов это слово важнее?

# TF-IDF: нормировка term frequency

В каком из двух текстов слово “собака” встретится чаще?

1. записка ветеринару
2. Война и Мир

В каком из этих текстов это слово важнее?

Другой пример:

статья на википедии	count(“ислам”)
Ислам	~ 200
Ислам в СССР	~ 400

# TF-IDF: нормировка term frequency

В каком из двух текстов слово “собака” встретится чаще?

1. записка ветеринару
2. Война и Мир

В каком из этих текстов это слово важнее?

Другой пример:

статья на википедии	count(“ислам”)	длина в символах
Ислам	~ 200	594434
Ислам в СССР	~ 400	1679405

по этой причине счетчики слов нормируют на количество слов в документе

# TF-IDF: логарифмирование inverse document frequency

token	document_count	inv_doc_freq	log_inv_doc_freq
конвенция	20	104.650000	4.650621
витать	19	110.157895	4.701915
одержимость	18	116.277778	4.755982
ливия	17	123.117647	4.813140
мина	16	130.812500	4.873765
нелёгкая	15	139.533333	4.938304
напряжённость	14	149.500000	5.007296
примирение	13	161.000000	5.081404
дипломатия	12	174.416667	5.161447
перерости	11	190.272727	5.248458
цивилизовать	10	209.300000	5.343769
перетекать	9	232.555556	5.449129
усугубить	8	261.625000	5.566912
афганец	7	299.000000	5.700444
подписание	6	348.833333	5.854594
проинструктировать	5	418.600000	6.036916
командующий	4	523.250000	6.260059
идейный	3	697.666667	6.547741
рабин	2	1046.500000	6.953207
легитимировать	1	2093.000000	7.646354

Вряд ли “подписание” в 2 раза важнее, чем “дипломатия”

при сравнении частотностей двух слов, даже если они встречаются примерно одинаковое количество раз, частотность более употребляемого слова будет экспоненциально выше, чем менее частого

$$\text{idf}(w, D) = \log \frac{|D|}{\{d \in D | w \in d\}}$$

---

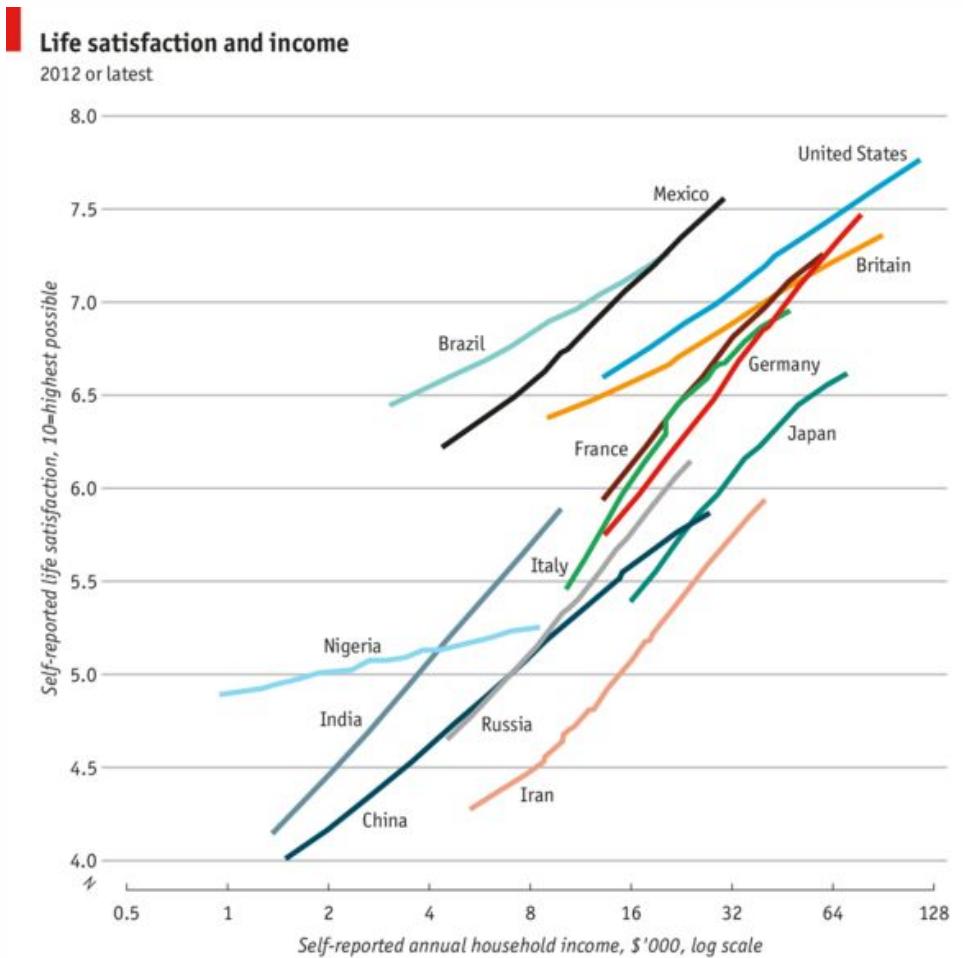
# **Логарифмирование: для каких еще признаков полезно?**



# Логарифмирование: для каких еще признаков полезно?

Например, зарплата имеет логарифмическую ценность: при повышении зарплаты с 25,000 до 50,000 вы обрадуетесь примерно также, как и при повышении с 50,000 до 100,000

=> деньги имеет смысл пробовать логарифмировать



Source: "Subjective Well-Being and Income: Is There Any Evidence of Satiation?",  
by Betsey Stevenson and Justin Wolfers. NBER Working Paper 18992. April 2013

Economist.com/graphicdetail

Everything you need to know about whether money makes you happy

---

## Как использовать tf-idf?

пусть мы сопоставили каждому тексту в корпусе вектор значений tf-idf. Что дальше?

# Как использовать tf-idf?

пусть мы сопоставили каждому тексту в корпусе вектор значений tf-idf. Что дальше?

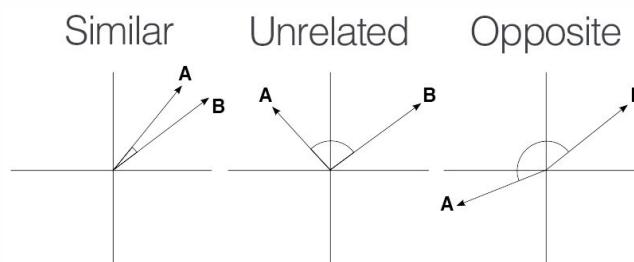
- обучать на таких признаках модели
  - например, все тот же sentiment analysis
- выделять ключевые слова
  - например, для анализа кластеров в случае кластеризации текстов
- искать похожие тексты
  - с помощью вычисления расстояния между tf-idf векторами

# Меры близости между векторами

Для вычисления меры близости между векторами используют разные функции скалярного произведения. Одна из самых популярных — косинусная близость (cosine similarity)

$$\cos(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Такая мера легко интерпретируема: лежит в диапазоне от -1.0 до +1.0



Иногда также говорят о косинусном “расстоянии”:

$$D(\mathbf{A}, \mathbf{B}) = 1 - \cos(\mathbf{A}, \mathbf{B})$$

# Cosine Similarity

$$\cos(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

- Если векторы нормированы (единичной длины), косинусная похожесть совпадает со скалярным произведением
- Если векторы нормированы, косинусная похожесть монотонно связана с евклидовым расстоянием между векторами

$$\|\mathbf{A} - \mathbf{B}\|^2 = \sum_{i=1}^n (A_i - B_i)^2 = \sum_{i=1}^n A_i^2 + \sum_{i=1}^n B_i^2 - 2 \sum_{i=1}^n A_i B_i$$

$$\|\mathbf{A}\| = 1, \|\mathbf{B}\| = 1$$

$$\|\mathbf{A} - \mathbf{B}\|^2 = \sum_{i=1}^n (A_i - B_i)^2 = 2(1 - \cos(\mathbf{A}, \mathbf{B}))$$

# Коллокации (collocations)

В языках существуют устойчивые словосочетания:

- хмели сунели
- ноу хау
- углекислый газ
- ice cream
- Mr. Muscle
- San Francisco

Но есть и такие:

- а я
- и это
- but the

Если строить словарь биграмм слов, то самыми частотными будут как раз вторые

# Pointwise Mutual Information

Для выделения коллокации можно воспользоваться взаимной информацией:

$$PMI(x, y) = \log \frac{p(x,y)}{p(x)p(y)}$$

$$p(w_i) \sim \frac{\text{count}(w_i)}{\sum_{j=1}^n \text{count}(w_j)} = \frac{\text{count}(w_i)}{N}$$

$$PMI(w_1, w_2) \sim \log \left( \frac{N \cdot \text{count}(w_1, w_2)}{\text{count}(w_1)\text{count}(w_2)} \right)$$

полученные коллокации можно использовать в качестве признаков или для интерпретации групп текстов

---

# Оптимально ли разбивать текст на слова?

Русский язык:

машына, расказ, телек, привеет



# **Оптимально ли разбивать текст на слова?**

Русский язык:

машына, расказ, телек, привеет

Немецкий язык:

Sehenswürdigkeiten, Sommerschlussverkauf

# **Оптимально ли разбивать текст на слова?**

Русский язык:

машына, расказ, телек, привеет

Немецкий язык:

Sehenswürdigkeiten, Sommerschlussverkauf

Японский язык:

今日は自然言語処理についての講義があります

# Оптимально ли разбивать текст на слова?

Русский язык:

машына, расказ, телек, привеет

Немецкий язык:

Sehenswürdigkeiten, Sommerschlussverkauf

Японский язык:

今日は自然言語処理についての講義があります

можно разбивать текст на n-gram'ы символов

```
import nltk

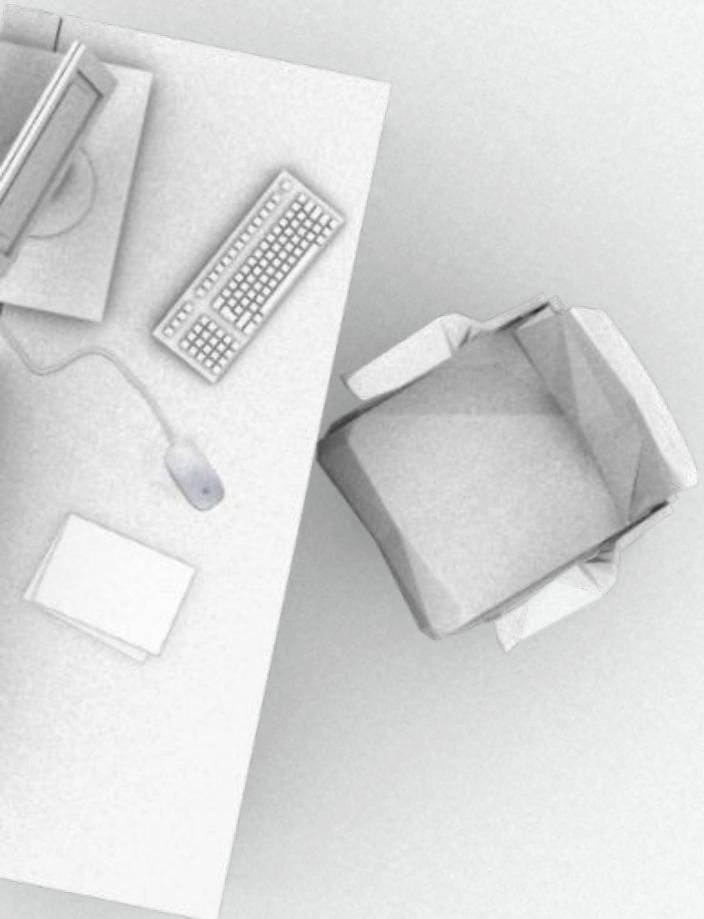
def get_ngrams(text, n):
    n_grams = nltk.ngrams(text, n)
    return [''.join(grams) for grams in n_grams]

sentence = 'лекция про тексты'

get_ngrams(sentence, 2)
# ['ле', 'ек', 'кц', 'ци', 'ия', 'я ', 'п', 'пр', 'ро', 'о ', 'т', 'те', 'ек', 'кс', 'ст', 'ты']
get_ngrams(sentence, 3)
# ['лек', 'екц', 'кци', 'ция', 'ия ', 'я п', 'пр', 'ро', 'о т', 'те', 'тек', 'екс', 'кст', 'сты']
get_ngrams(sentence, 4)
# ['лекц', 'екци', 'кция', 'ция ', 'ия п', 'я пр', 'пр', 'ро', 'ро ', 'о т', 'о те', 'тек', 'текс', 'ект', 'ксты']
```

# Этапы подготовки признаков

1. Чистка текста (html-тэги, спец. символы, ...)
2. Приведение к нижнему регистру
3. Токенизация
  - на отдельные слова (word unigram)
  - на последовательности слов (word n-gram)
    - можно отбирать с помощью взаимной информации (PMI)
  - на последовательности символов (char n-gram)
4. Морфологический разбор (стемминг / лемматизация)
5. Фильтрация токенов
  - слишком редких (на них можем переобучиться)
  - с большой документной частотой (стоп-слова, вносят шум в модель)

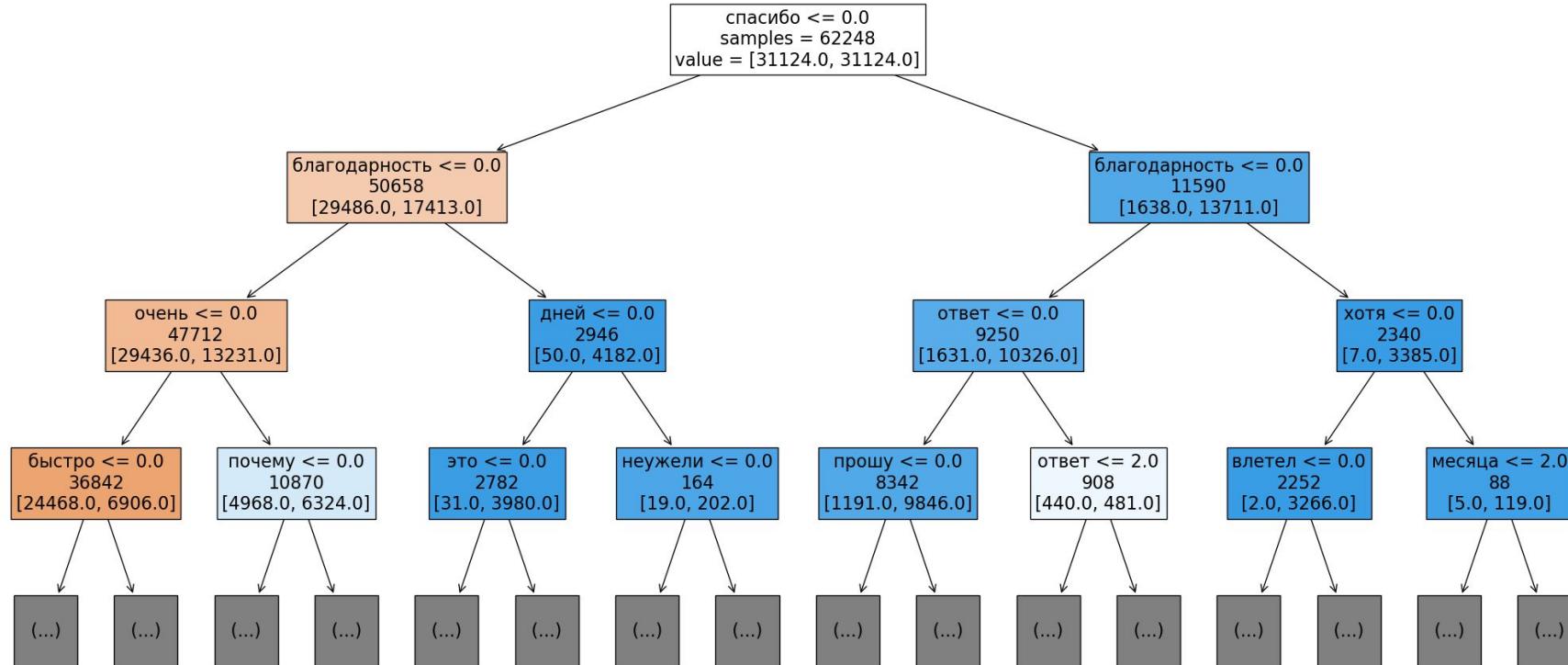


# Выбор модели для классификации текста

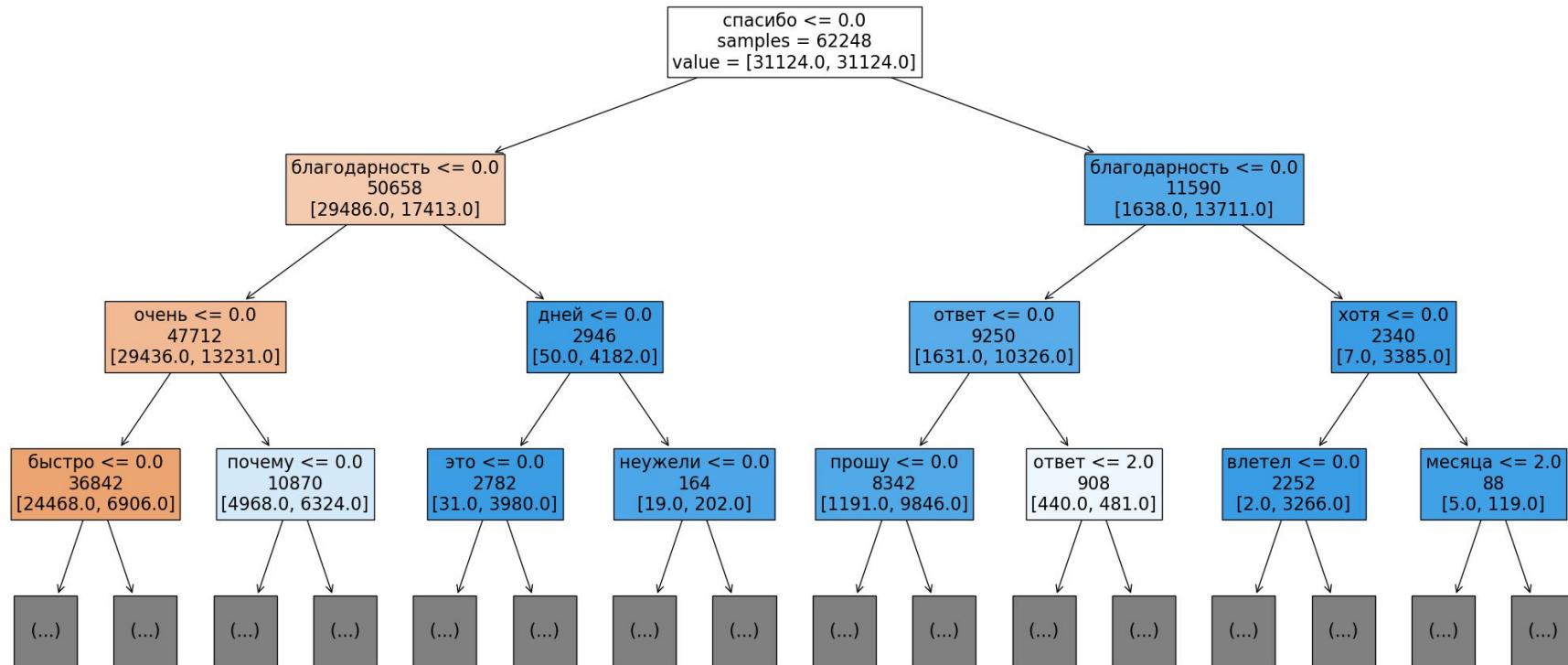
“[All models are wrong, but some are useful](#)”

[George Box](#)

# Дерево решений



# Дерево решений



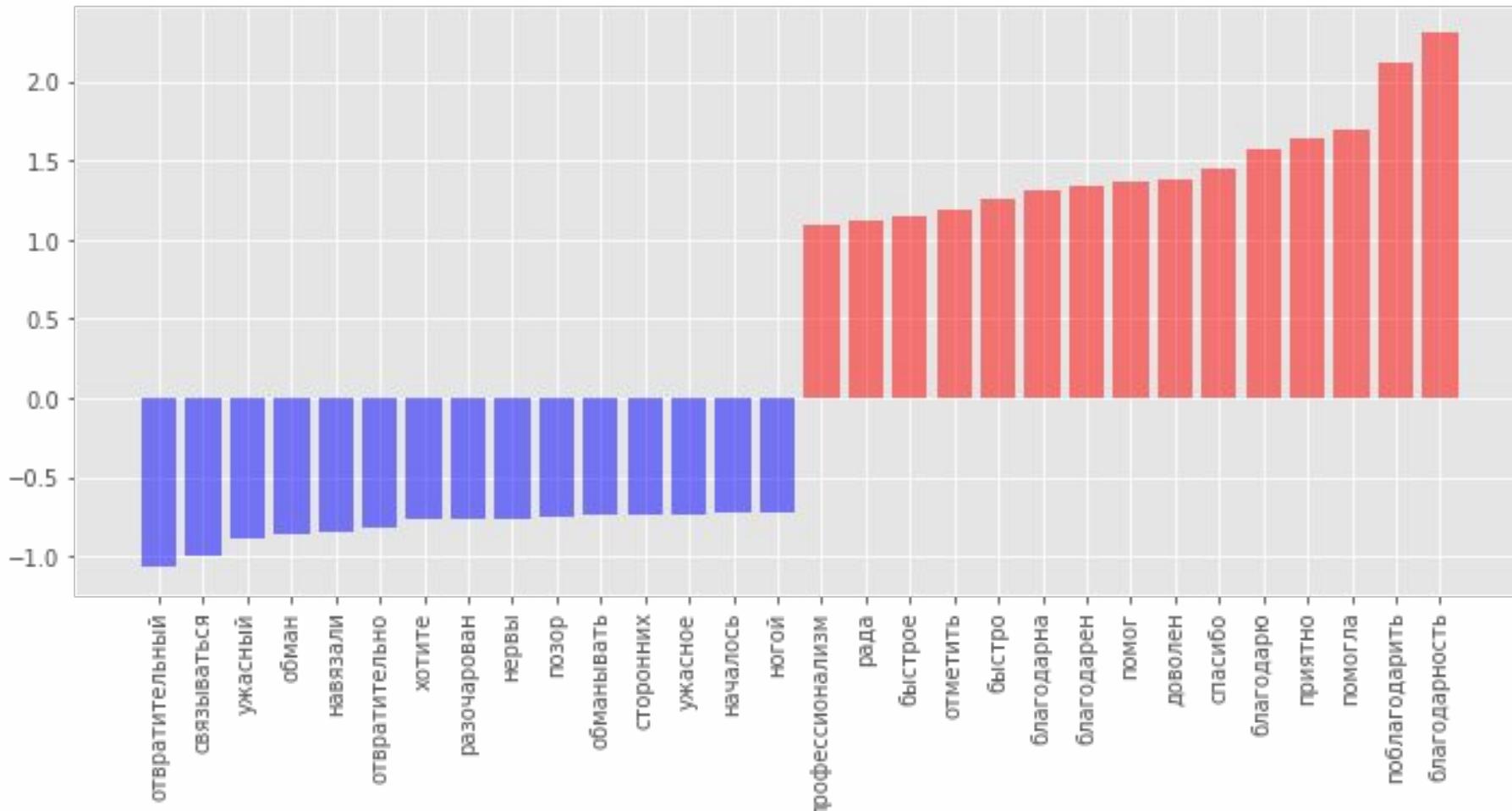
Текстовые данные существенно разреженные

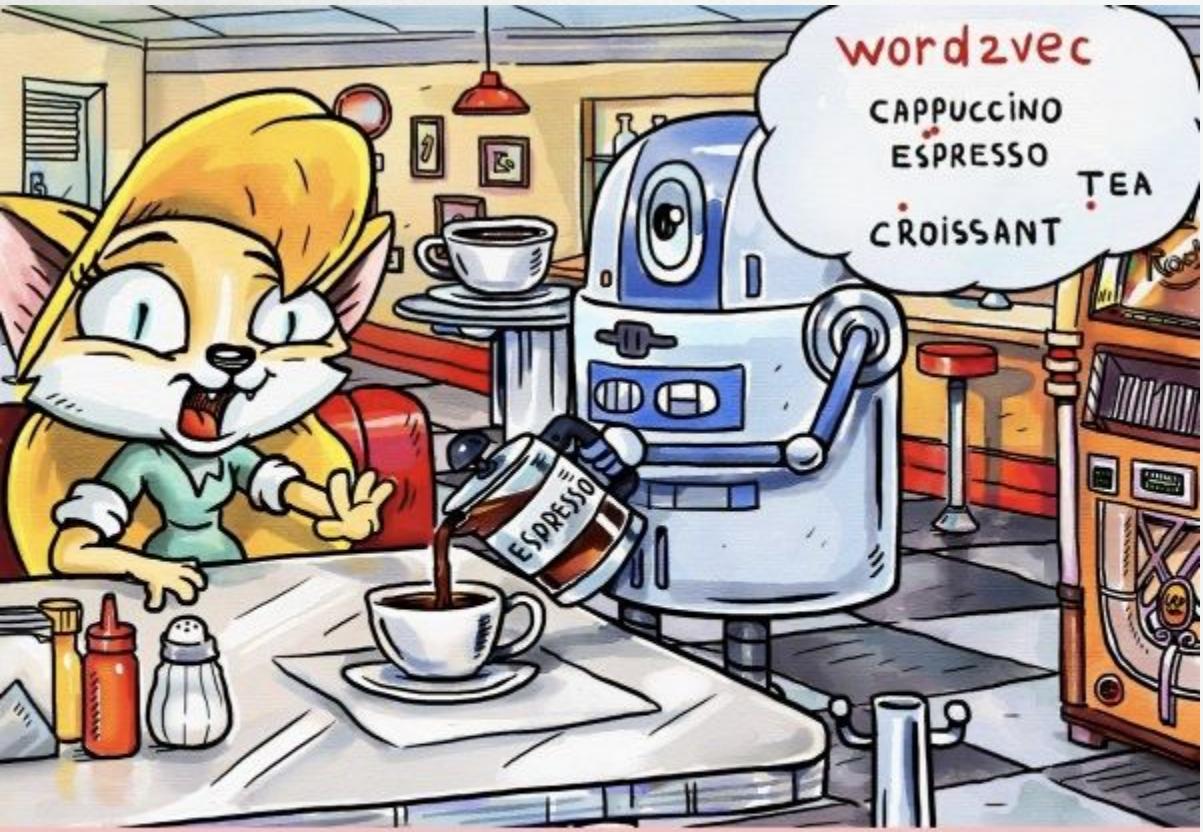
Если дерево мелкое, то ему сложно выделить закономерности в текстах.

Если дерево глубокое, то легко переобучится

# Логистическая регрессия

Логистическая регрессия на tf-idf факторах простой и сильный baseline для текстовой классификации. Начинать лучше всего определенно с нее





- Espresso? But I ordered a cappuccino!
- Don't worry, the cosine distance between them is so small that they are almost the same thing.

## Word Embeddings

# В чем проблема

Какая похожесть между следующими фразами, если мы используем в качестве векторов TF-IDF ?

`cosine_similarity( “купить айфон”, “приобрести iphone” )`

`cosine_similarity( “карбюратор для машины”, “запчасти к авто” )`

`cosine_similarity( “туфли с каблуком”, “обувь на танкетке” )`

# Embedding

Отображение какой-либо сущности (узел графа, слово/предложение/документ, изображение, аудиосигнал, ... ) в вектор вещественных чисел фиксированной размерности

Синонимы:

векторные представления (вложения, погружения, сопоставления, ...)

# Embedding

Отображение какой-либо сущности (узел графа, слово/предложение/документ, изображение, аудиосигнал, ... ) в вектор вещественных чисел фиксированной размерности

Синонимы:

векторные представления (вложения, погружения, сопоставления, ...)

Отображение сущности в вектор случайных чисел не очень полезно на практике. Хотим получить такое отображение слов в векторы, что у близких по смыслу слов будут близкие векторы:

# Embedding

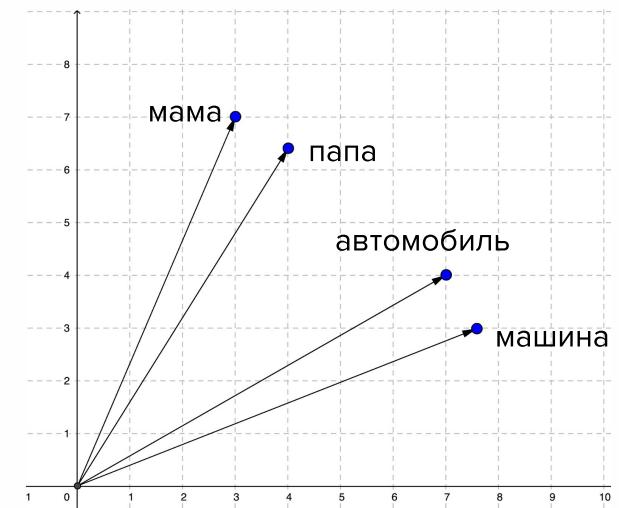
Отображение какой-либо сущности (узел графа, слово/предложение/документ, изображение, аудиосигнал, ... ) в вектор вещественных чисел фиксированной размерности

Синонимы:

векторные представления (вложения, погружения, сопоставления, ...)

Отображение сущности в вектор случайных чисел не очень полезно на практике. Хотим получить такое отображение слов в векторы, что у близких по смыслу слов будут близкие векторы:

мама	[-2.4117122 , 1.458589, ... , 0.275895, -0.5202267]
папа	[-2.6366158, 0.7762581, ... , -1.0616287, -0.08835255]
...	...
автомобиль	[0.41592726, 1.7632927, ... , -1.1502137, -1.818282]
машина	[-2.4796112, 1.7327983, ... , -1.0928246, -2.8900113]



# Дистрибутивная семантика

Согласно дистрибутивной гипотезе **смысл** слова можно понять по его контексту:

“You shall know a word by the company it keeps“  
([Firth, J. R. 1957:11](#))

Пример: Мама пьет **каффи** с молоком  
Как перевести **каффи** ?

# Дистрибутивная семантика

Согласно дистрибутивной гипотезе **смысл** слова можно понять по его контексту:

“You shall know a word by the company it keeps“  
([Firth, J. R. 1957:11](#))

Пример: Мама пьет **каффи** с молоком  
Как перевести **каффи** ?

Попробуем поставить задачу машинного обучения:  
предсказать слово по его окружению

— Какая это задача?

# Дистрибутивная семантика

Согласно дистрибутивной гипотезе **смысл** слова можно понять по его контексту:

“You shall know a word by the company it keeps“  
([Firth, J. R. 1957:11](#))

Пример: Мама пьет **каффи** с молоком  
Как перевести **каффи** ?

Попробуем поставить задачу машинного обучения:  
предсказать слово по его окружению

- Какая это задача?
- Классификация

# Дистрибутивная семантика

Согласно дистрибутивной гипотезе **смысл** слова можно понять по его контексту:

“You shall know a word by the company it keeps“  
[\(Firth, J. R. 1957:11\)](#)

Пример: Мама пьет **каффи** с молоком  
Как перевести **каффи** ?

Попробуем поставить задачу машинного обучения:  
предсказать слово по его окружению

- Какая это задача?
- Классификация

Будем моделировать распределение (**distribution**) вероятности встретить слово  
в контексте других слов

# Как собрать обучающую выборку?

подойдут тексты без разметки! (например, dump википедии)

ПРИМЕР: “Машинное обучение — класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач”

машинный	обучение	класс	метод	искусственный
обучение	класс	метод	искусственный	интеллект
класс	метод	искусственный	интеллект	характерный
метод	искусственный	интеллект	характерный	черта
...	...	...	...	...

по окружению (синим словам) будем тренировать модель предсказывать центральное слово (выделено красным)

# Алгоритм обучения

машины	обучение	класс	метод	искусственный
--------	----------	-------	-------	---------------

Случайно инициализируем две матрицы: матрица контекстных слов (синяя) и матрица центральных слов (красная). Эти матрицы — параметры модели, которые настраиваются с помощью градиентного спуска.

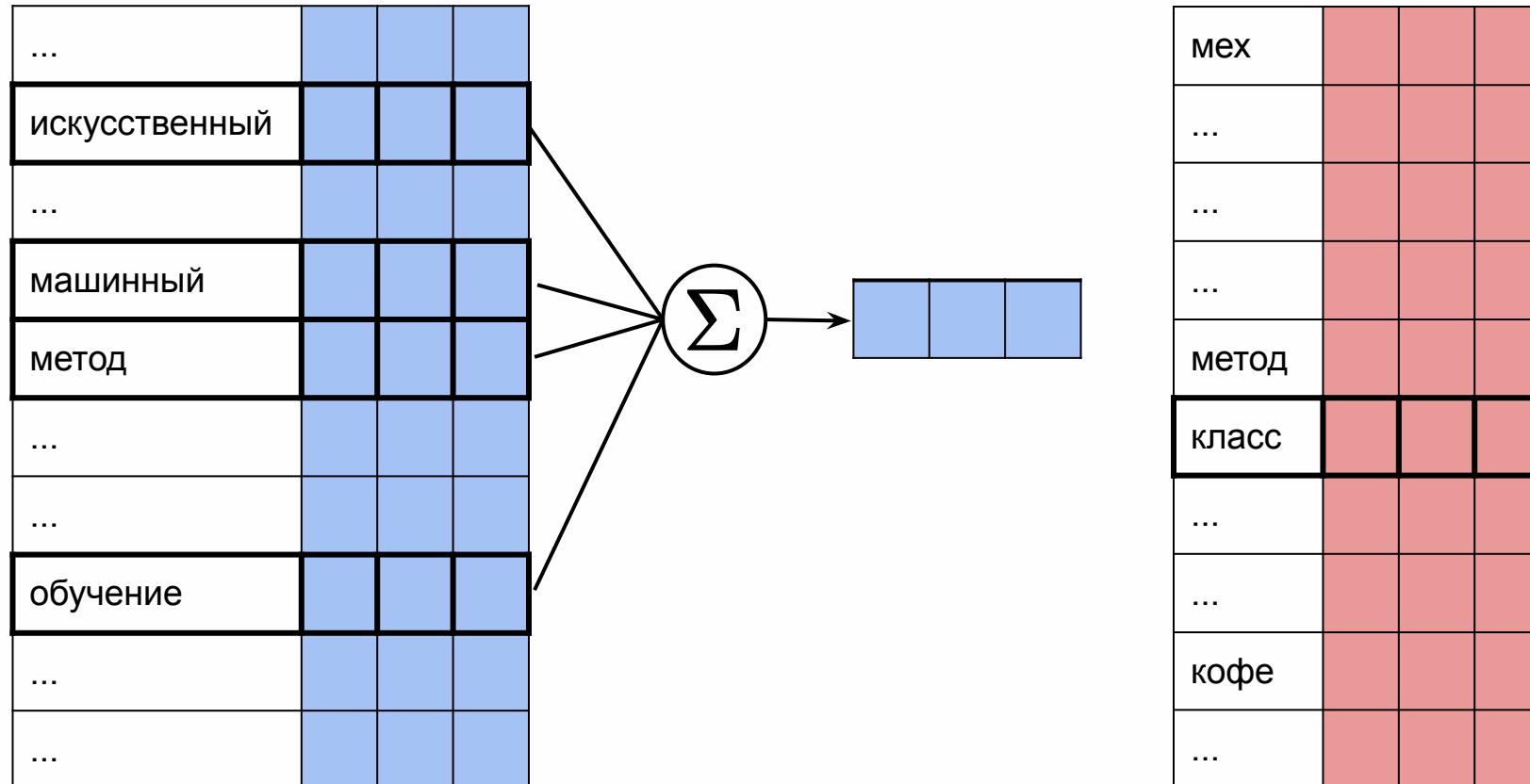
...				
искусственный				
...				
машины				
метод				
...				
...				
обучение				
...				
...				

мех				
...				
...				
...				
метод				
...				
...				
класс				
...				
...				
кофе				
...				

# Алгоритм обучения



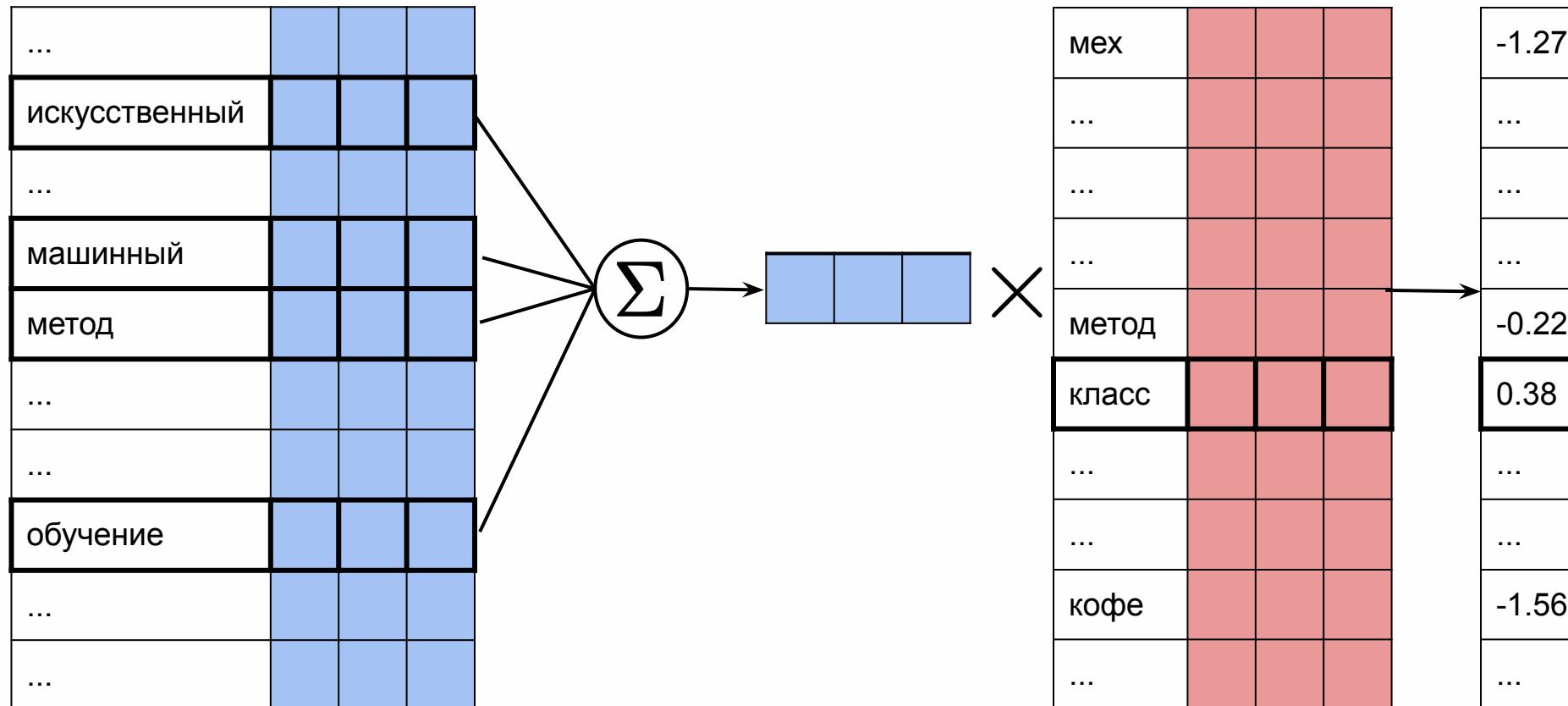
Случайно инициализируем две матрицы: матрица контекстных слов (синяя) и матрица центральных слов (красная). Эти матрицы — параметры модели, которые настраиваются с помощью градиентного спуска. Суммируем векторы контекстных слов,



# Алгоритм обучения



Случайно инициализируем две матрицы: матрица контекстных слов (синяя) и матрица центральных слов (красная). Эти матрицы — параметры модели, которые настраиваются с помощью градиентного спуска. Суммируем векторы контекстных слов, результат умножаем с матрицей центральных слов.

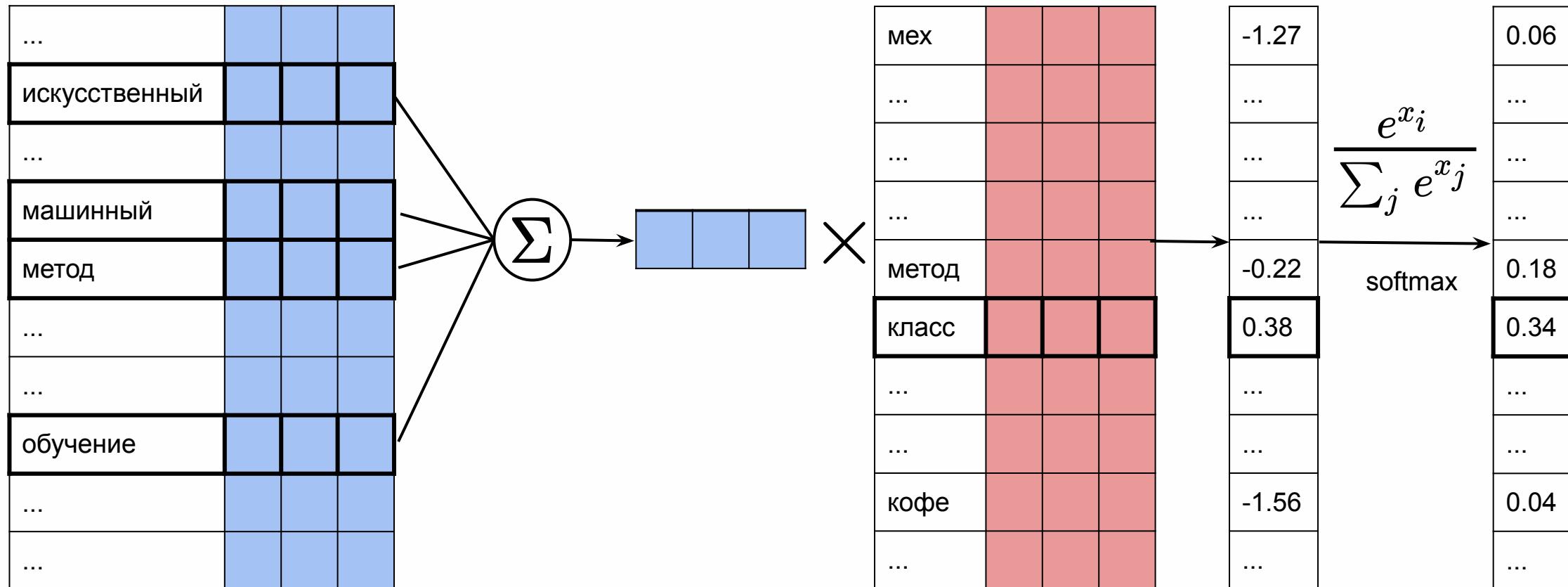


\*На самом деле softmax в word2vec считается несколько иначе для ускорения обучения

# Алгоритм обучения



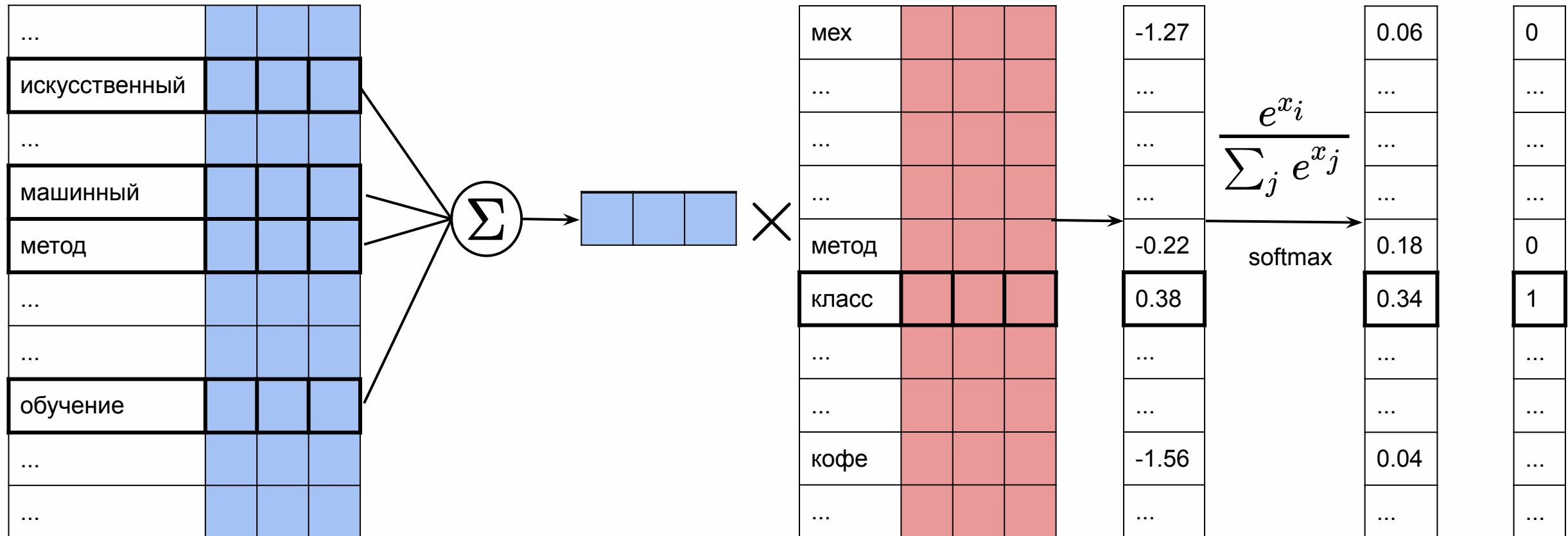
Случайно инициализируем две матрицы: матрица контекстных слов (синяя) и матрица центральных слов (красная). Эти матрицы — параметры модели, которые настраиваются с помощью градиентного спуска. Суммируем векторы контекстных слов, результат умножаем с матрицей центральных слов. Для оценок вероятности применяем к результату softmax.



# Алгоритм обучения



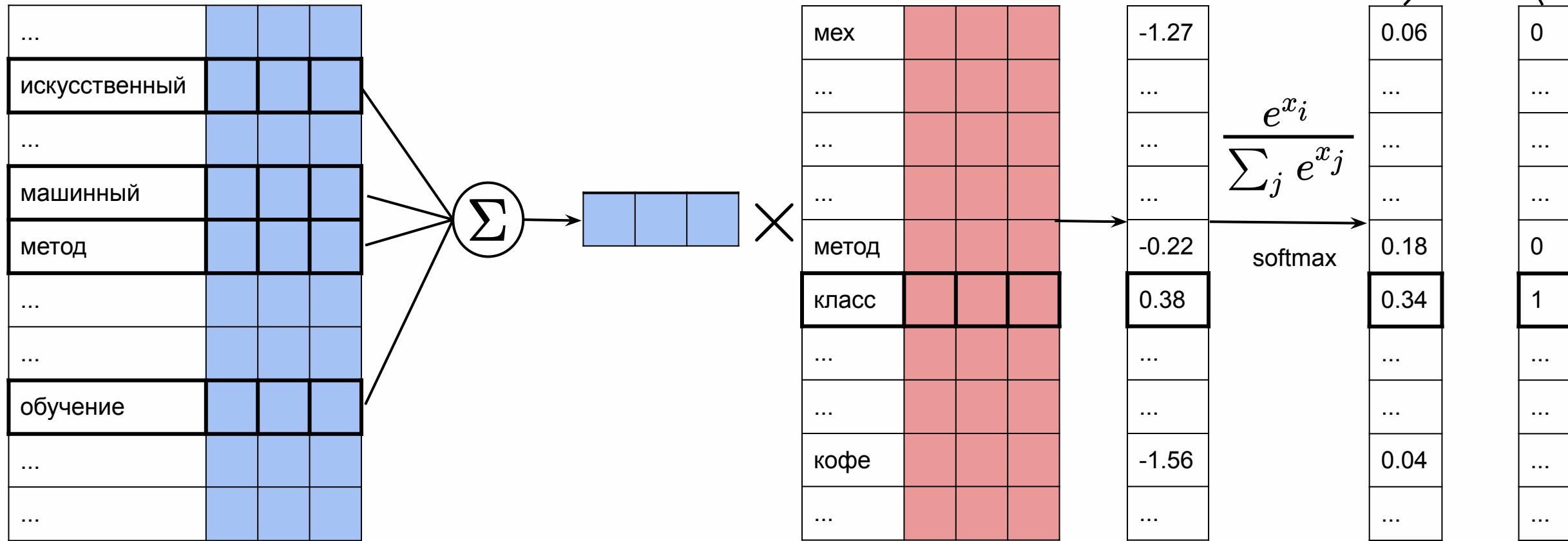
Случайно инициализируем две матрицы: матрица контекстных слов (синяя) и матрица центральных слов (красная). Эти матрицы — параметры модели, которые настраиваются с помощью градиентного спуска. Суммируем векторы контекстных слов, результат умножаем с матрицей центральных слов. Для оценок вероятности применяем к результату softmax.



# Алгоритм обучения



Случайно инициализируем две матрицы: матрица контекстных слов (синяя) и матрица центральных слов (красная). Эти матрицы — параметры модели, которые настраиваются с помощью градиентного спуска. Суммируем векторы контекстных слов, результат умножаем с матрицей центральных слов. Для оценок вероятности применяем к результату softmax. Варьируем значения в красной и синей матрицах для достижения минимума функции потерь



\*На самом деле softmax в word2vec считается несколько иначе для ускорения обучения

# Применение модели

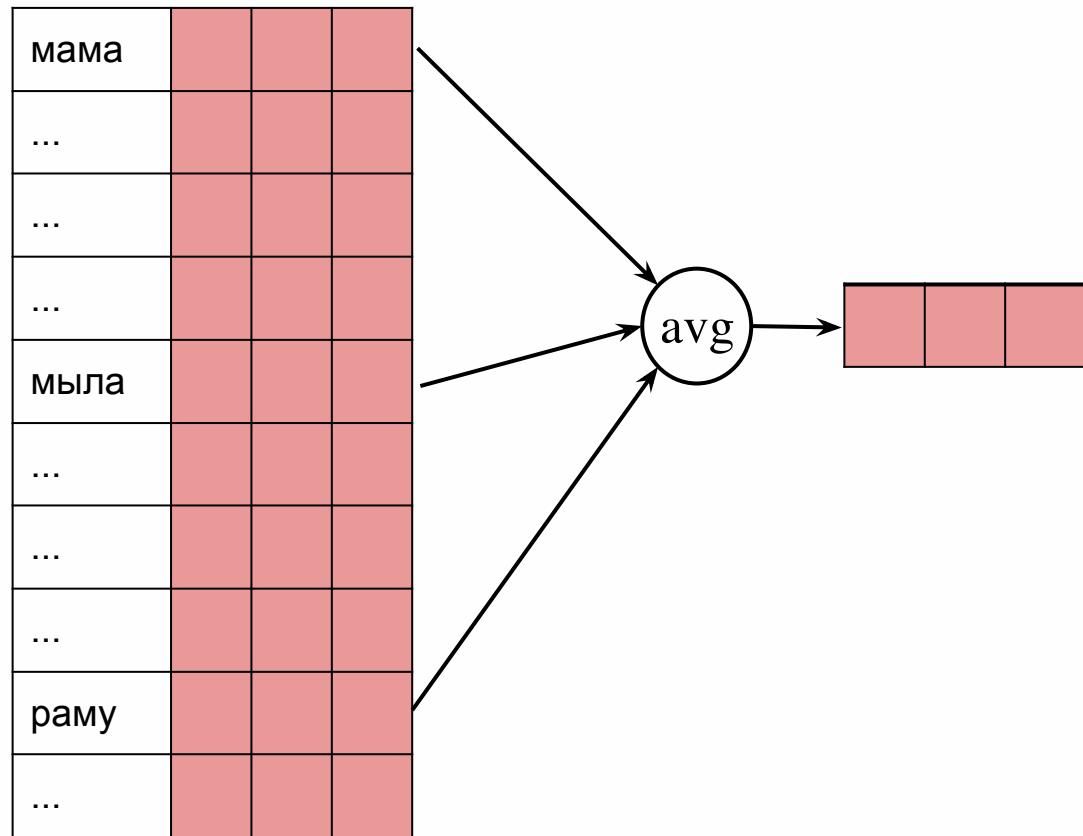
Оставляем только одну из матриц и используем в качестве признаков слов

мама			
...			
...			
...			
мыла			
...			
...			
...			
раму			
...			

# Применение модели

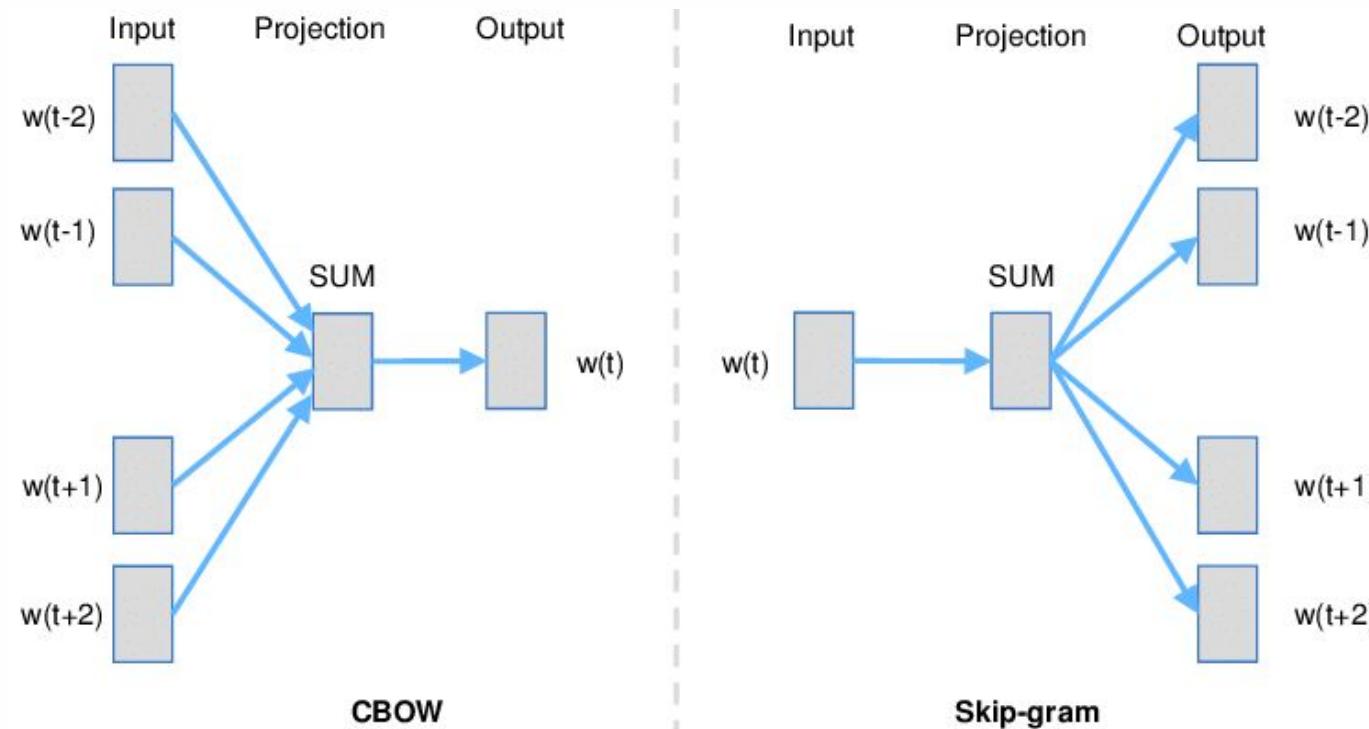
Оставляем только одну из матриц и используем в качестве признаков слов

Для получения вектора предложения можно усреднить векторы входящих в него слов



# Архитектуры word2vec

Мы рассмотрели CBOW (Continuous Bag of Words), существует также skip-gram:



# Аналогии и векторная арифметика

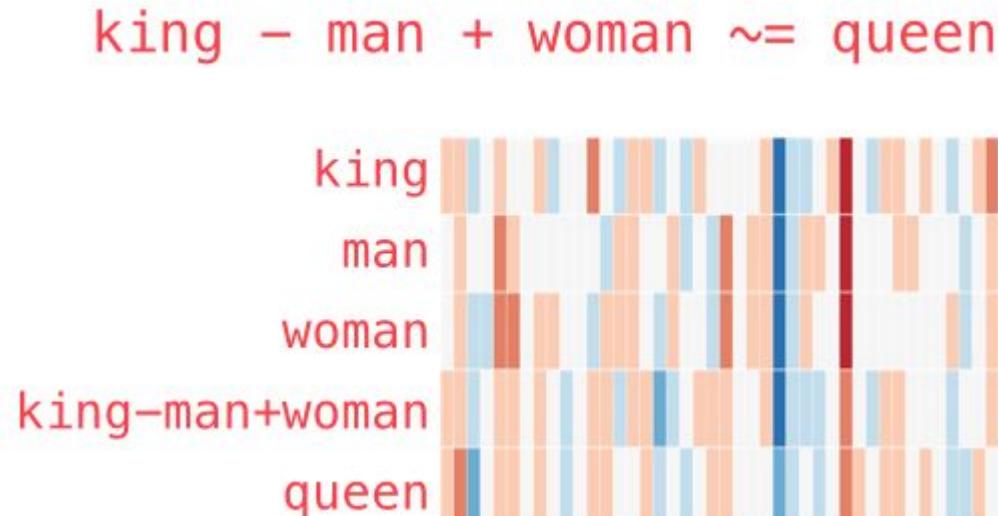
<http://jalammar.github.io/illustrated-word2vec/>

king - man + woman ~ queen



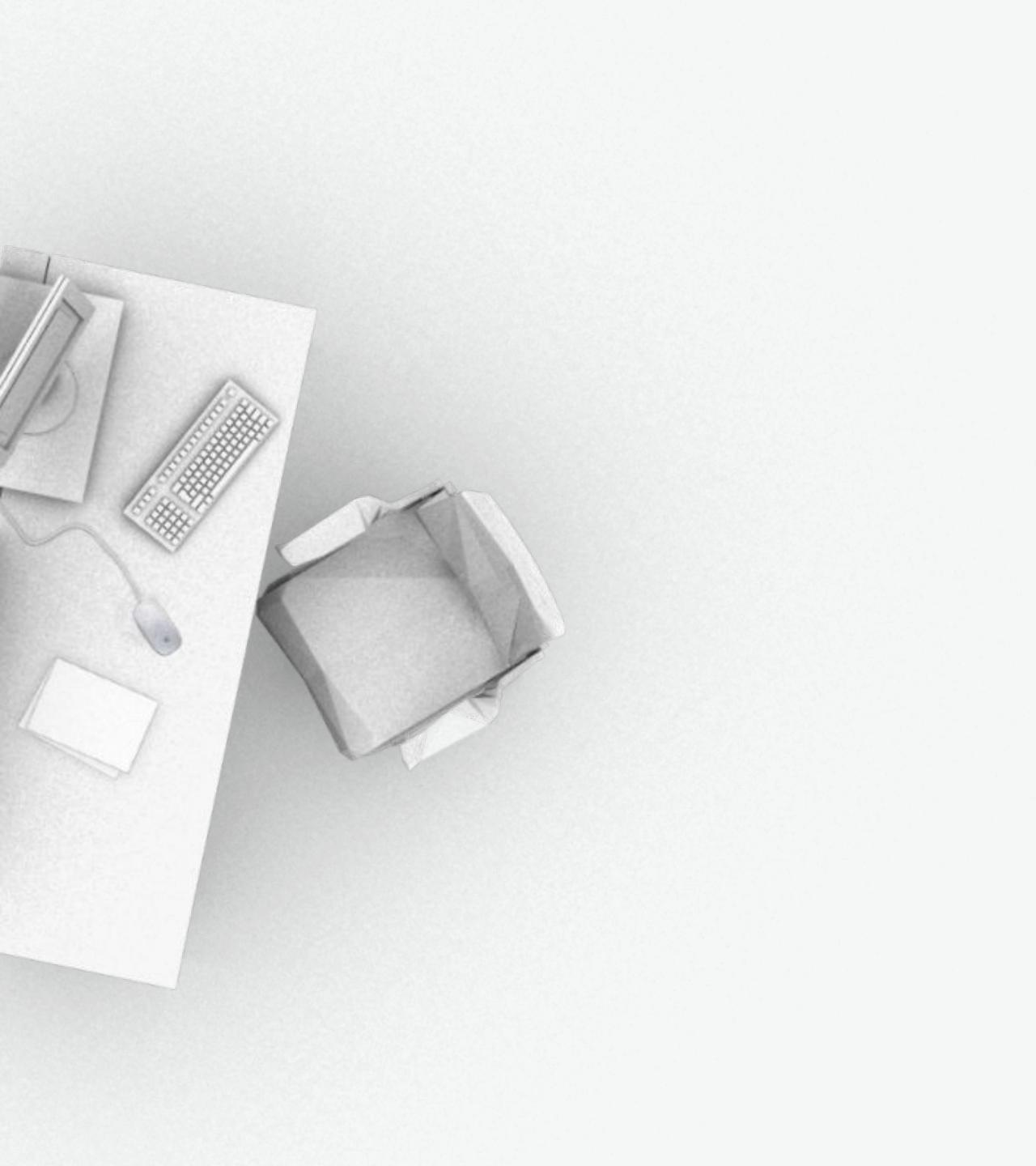
# Аналогии и векторная арифметика

<http://jalammar.github.io/illustrated-word2vec/>



<https://rusvectores.org/ru/#>

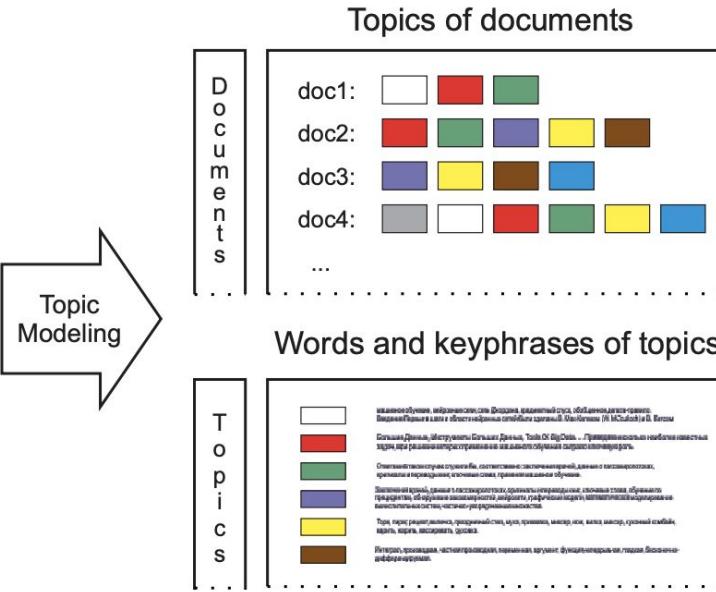
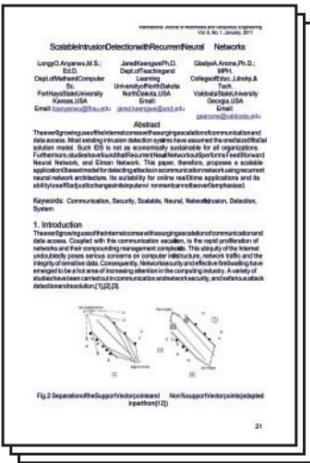




# **Тематическое моделирование**

# ЧТО ХОТИМ ПОЛУЧИТЬ

Text documents



Topic 68		Topic 79	
research	4.56	институт	6.03
technology	3.14	университет	3.35
engineering	2.63	программа	3.17
institute	2.37	учебный	2.75
science	1.97	технический	2.70
program	1.60	технология	2.30
education	1.44	научный	1.76
campus	1.43	исследование	1.67
management	1.38	наука	1.64
programs	1.36	образование	1.47
		goals	4.48
		league	3.99
		club	3.76
		season	3.49
		scored	2.72
		cup	2.57
		goal	2.48
		apps	1.74
		debut	1.69
		match	1.67
		матч	6.02
		игрок	5.56
		сборная	4.51
		фк	3.25
		против	3.20
		клуб	3.14
		футболист	2.67
		гол	2.65
		забивать	2.53
		команда	2.14

Vorontsov, Frei, Apishev, Romov, Suvorova. BigARTM: Open Source Library for Regularized Multimodal Topic Modeling of Large Collections. AIST-2015.

# Обозначения и основная идея

$W$  — словарь терминов

$D$  — коллекция текстовых документов

$n_{dw}$  — количество вхождения термина  $w$  в документ  $d$

$n_d$  — длина документа  $d$  (количество слов)

$p(w|t)$  — условное распределение на множестве терминов (тема)

$p(t|d)$  — условное распределение на множестве тем (тематика документа)

Тематическая модель автоматически выявляет скрытые темы  $t \in T$  в коллекции документов  $D$  по наблюдаемым частотам терминов в документах  $\hat{p}(w|d) = \frac{n_{dw}}{n_d}$

После обучения можно использовать  $p(t|d)$  как сжатое представление документа, а  $p(w|t)$  для интерпретации полученных тем

# Этапы решения задачи

Предварительная обработка и чистка

Формирование словаря

Предположения модели

- порядок терминов в документе не важен (bag of words)
- каждая пара  $(w, d)$  связана с некоторой темой  $t \in T$
- слова в документе порождаются именно темой, а не самим документом (гипотеза условной независимости):

$$p(w|d, t) = p(w|t)$$

---

## Этапы решения задачи

Согласно формуле полной вероятности:

$$p(w|d) = \sum_{t \in T} p(w|d, t) p(t|d)$$

Согласно вероятностному предположению модели:

$$p(w|d, t) = p(w|t)$$

$$p(w|d) = \sum_{t \in T} p(w|t) p(t|d)$$

## Этапы решения задачи

Согласно формуле полной вероятности:

$$p(w|d) = \sum_{t \in T} p(w|d, t) p(t|d)$$

Согласно вероятностному предположению модели:

$$p(w|d, t) = p(w|t)$$

$$p(w|d) = \sum_{t \in T} p(w|t) p(t|d)$$

Напоминает произведение матриц ( $c_{ij} = \sum_k a_{ik} b_{kj}$ )

## Этапы решения задачи

Согласно формуле полной вероятности:

$$p(w|d) = \sum_{t \in T} p(w|d, t) p(t|d)$$

Согласно вероятностному предположению модели:

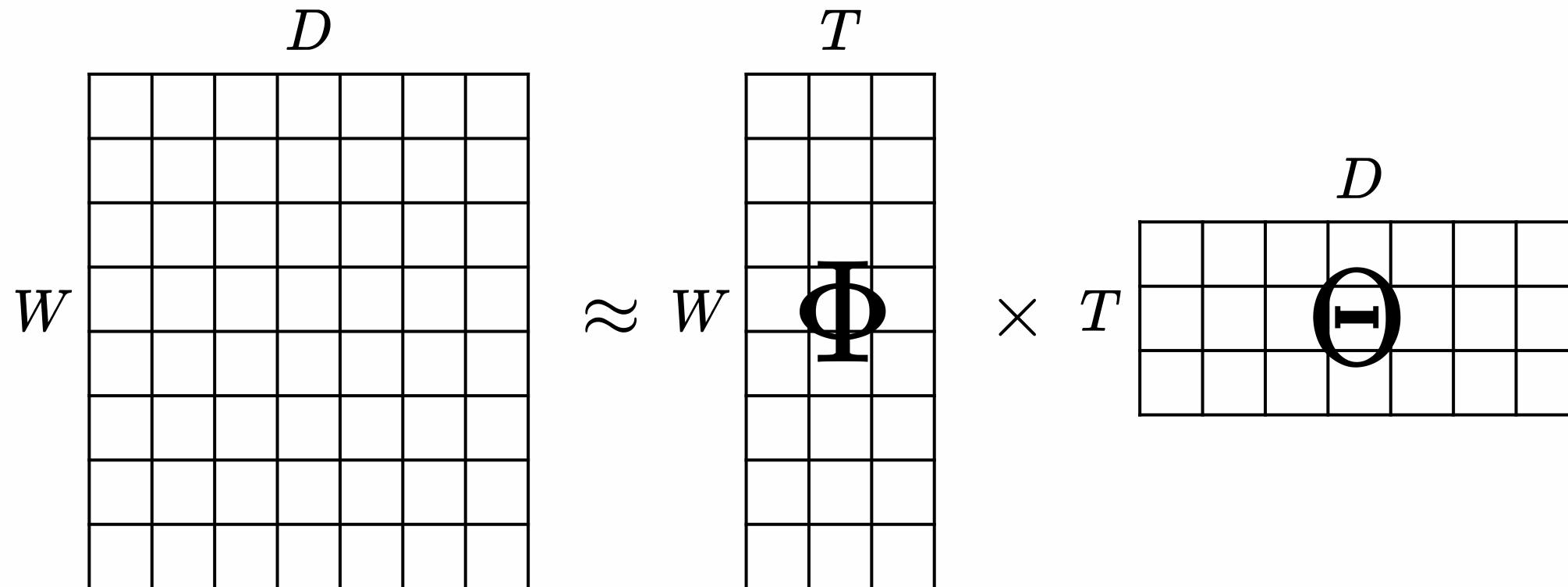
$$p(w|d, t) = p(w|t)$$

$$p(w|d) = \sum_{t \in T} p(w|t) p(t|d)$$

Напоминает произведение матриц ( $c_{ij} = \sum_k a_{ik} b_{kj}$ )

Таким образом, задачу тематического моделирования можно трактовать как задачу матричного разложения!

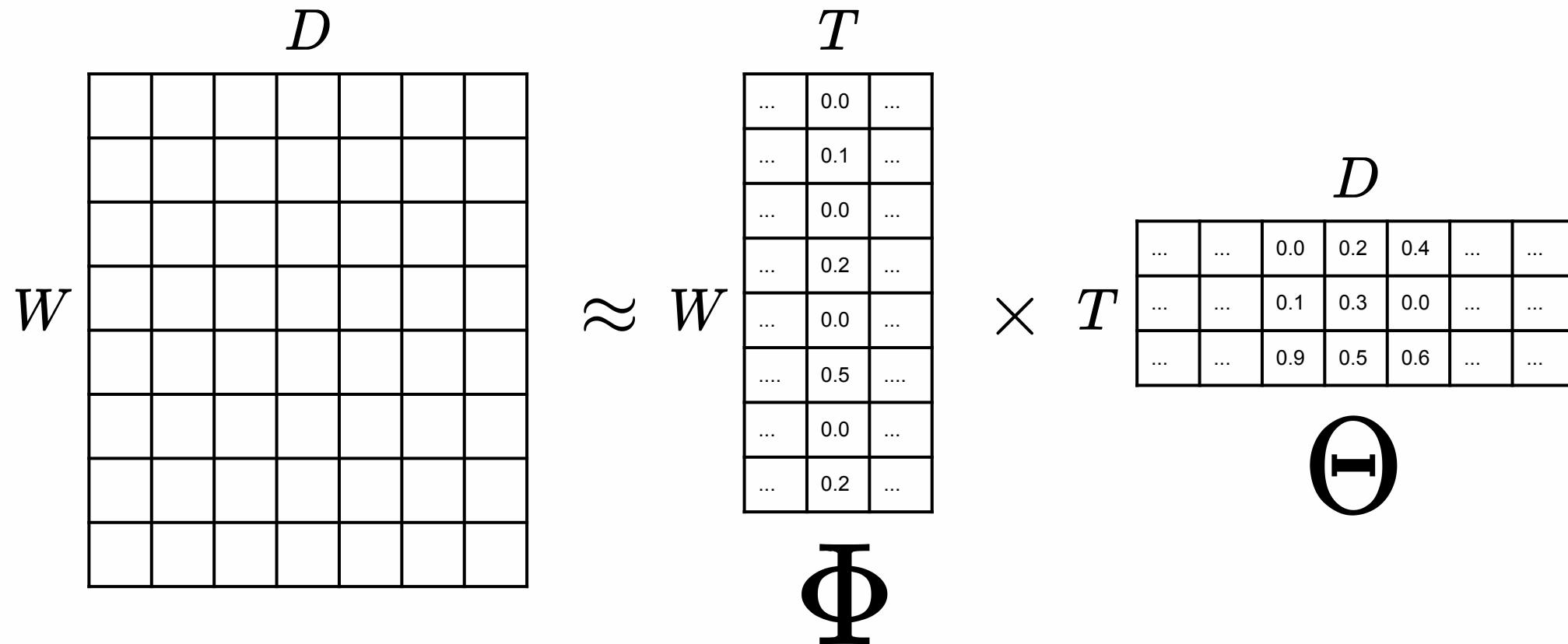
# Тематическое моделирование как матричное разложение



произведение  $\Phi$  и  $\Theta$  должно давать частотные оценки условных вероятностей слов в документах:

$$\hat{p} (w|d) = \frac{n_{dw}}{n_d}$$

# Тематическое моделирование как матричное разложение



Матрица  $\Phi$  задает распределение слов в темах => нормирована по столбцам  
Матрицы  $\Theta$  — распределение тем в документах => нормирована по столбцам

# Тематическое моделирование как матричное разложение

Разложение наблюдаемой матрицы частот слов в документах на произведение двух не является единственным:

$$\Phi\Theta = \Phi \left( SS^{-1} \right) \Theta = (\Phi S) \left( S^{-1}\Theta \right) = \Phi' \Theta'$$

Чтобы из множества решений выбрать наиболее подходящее задается критерий регуляризации. Разные регуляризаторы приводят к разным тематическим моделям:

- pLSA
- LDA
- ARTM



# **Домашнее задание**

# Классификация текстов: определение языка

ДЗ №7

The screenshot shows the Kaggle competition interface for 'IntroML 2020. Language detection'. The top navigation bar includes 'InClass Prediction Competition' and the title 'IntroML 2020. Language detection'. Below the title, it says 'Введение в машинное обучение. Определение языка предложения'. A timer indicates '22 days to go'. The main menu items are Overview, Data, Notebooks, Discussion, Leaderboard, Rules, Team (which is underlined), Host, My Submissions, and Submit Predictions. At the bottom, there's a 'Manage Team' section where the team name '[MSU] Georgy Gospodinov' is entered and a 'Save Team Name' button.

7

Баллов  
за задание

24.11.20

Срок  
сдачи

<https://www.kaggle.com/t/6b23f7b92b4b497e9aead739dd00ae1a>

пожалуйста, указывайте название вуза

# Спасибо за внимание!

Господинов Георгий

✉ g.gospodinov@corp.mail.ru

👉 georgygospodinov

