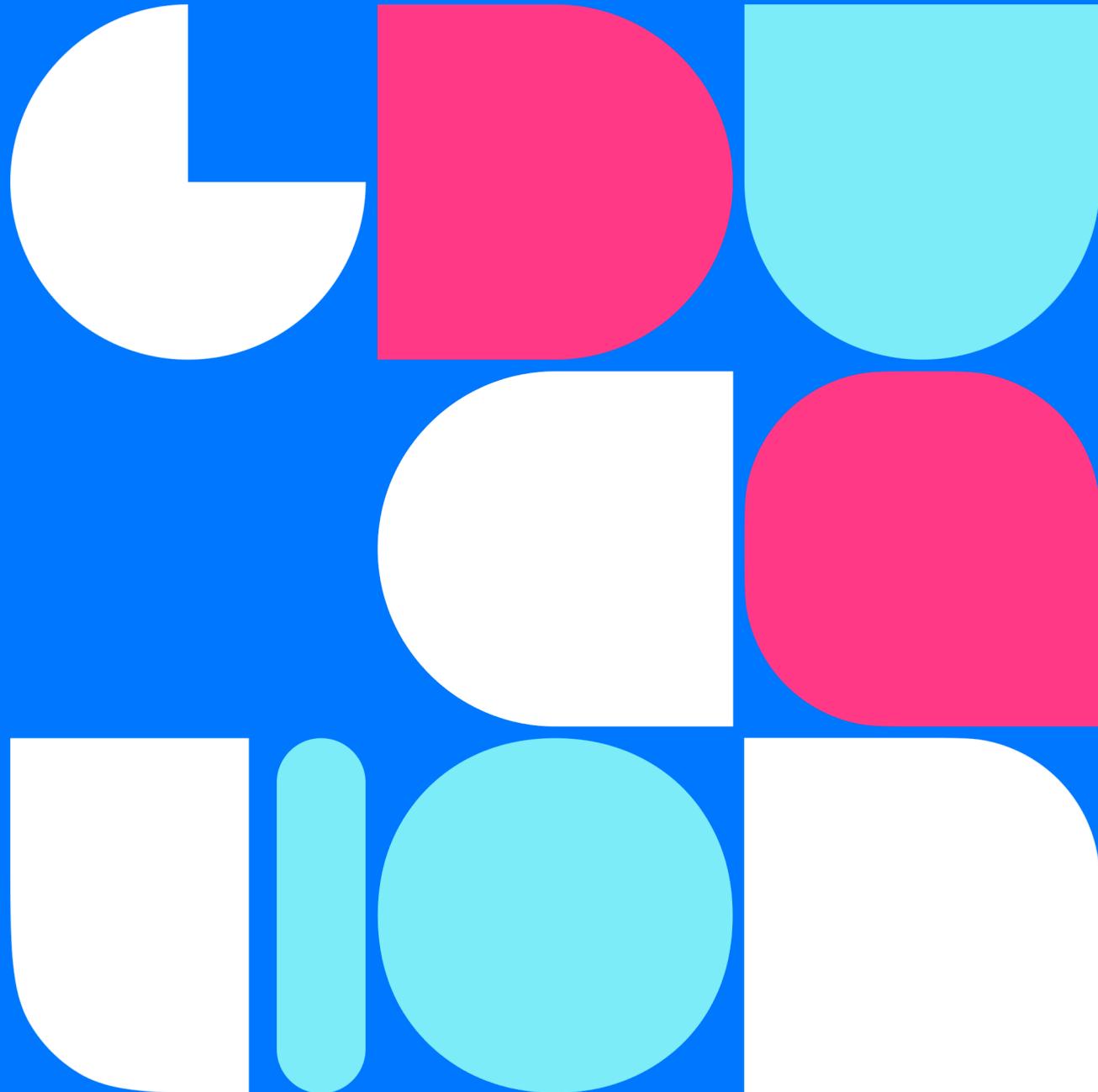




Embeddings



Sparse Vector Representations

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

corpus = [
    'машинное обучение',
    'глубокое обучение',
    'метод максимального правдоподобия'
]

vectorizer = TfidfVectorizer().fit(corpus)

pd.DataFrame(
    data=vectorizer.transform(corpus).todense(),
    columns=vectorizer.get_feature_names_out()
)
```

- какова размерность вектора?
- как сравнить два вектора?
- есть ли проблемы у таких представлений?

	глубокое	максимального	машинное	метод	обучение	правдоподобия
0	0.000000	0.000000	0.795961	0.000000	0.605349	0.000000
1	0.795961	0.000000	0.000000	0.000000	0.605349	0.000000
2	0.000000	0.577350	0.000000	0.577350	0.000000	0.577350

Word Embeddings

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

```
cosine_similarity( "купить смартфон", "приобрести smartphone" )  
cosine_similarity( "карбюратор для машины", "запчасти к авто" )  
cosine_similarity( "туфли с каблуком", "обувь на танкетке" )
```

Word Embeddings

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

```
cosine_similarity( "купить смартфон", "приобрести smartphone" )  
cosine_similarity( "карбюратор для машины", "запчасти к авто" )  
cosine_similarity( "туфли с каблуком", "обувь на танкетке" )
```

косинусная близость равна нулю в каждом из этих примеров, так как разные слова отвечают разным координатам вектора для TF-IDF представлений

Для вычисления близости между предложениями не только по общим словам, но и по смыслу можно построить плотные низкоразмерные представления для слов с помощью **self-supervised learning**, а затем агрегировать представления отдельных слов в вектор всего предложения

Напоминание: логистическая регрессия для нескольких классов

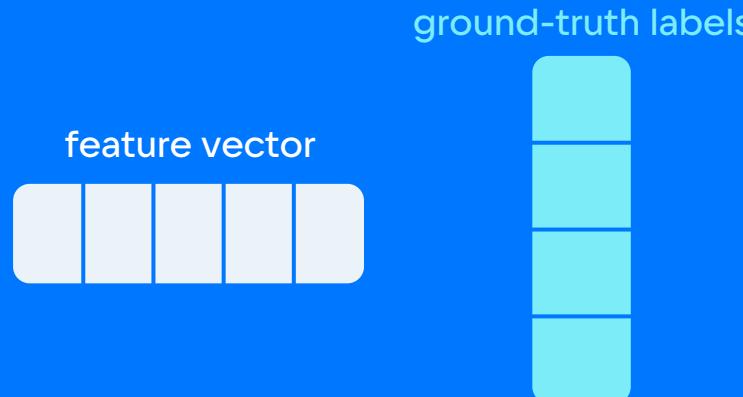
Решаем задачу классификации на **4** класса. Используем логистическую регрессию.
Для каждого объекта **5** признаков

Сколько параметров модели?

Напоминание: логистическая регрессия для нескольких классов

Решаем задачу классификации на **4** класса. Используем логистическую регрессию.
Для каждого объекта **5** признаков

Сколько параметров модели? — 24 (weights: 5×4 + bias: 4)



Напоминание: логистическая регрессия для нескольких классов

Решаем задачу классификации на **4** класса. Используем логистическую регрессию.
Для каждого объекта **5** признаков

Сколько параметров модели? — 24 (weights: 5×4 + bias: 4)



ground-truth labels



Напоминание: логистическая регрессия для нескольких классов

Решаем задачу классификации на **4** класса. Используем логистическую регрессию.
Для каждого объекта **5** признаков

Сколько параметров модели? — 24 (weights: 5×4 + bias: 4)

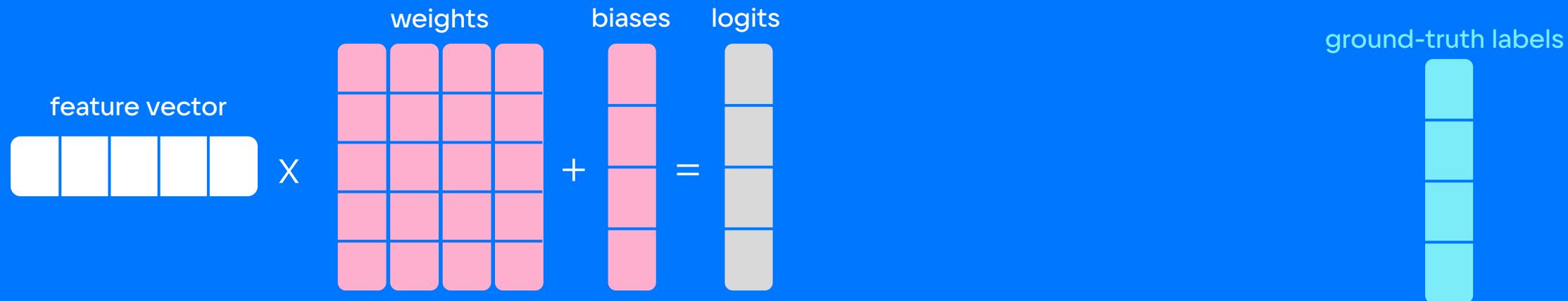


Напоминание: логистическая регрессия для нескольких классов

Решаем задачу классификации на **4** класса. Используем логистическую регрессию.
Для каждого объекта **5** признаков

Сколько параметров модели? — 24 (weights: 5×4 + bias: 4)

На выходе модели будут вещественные числа со знаком (logits).



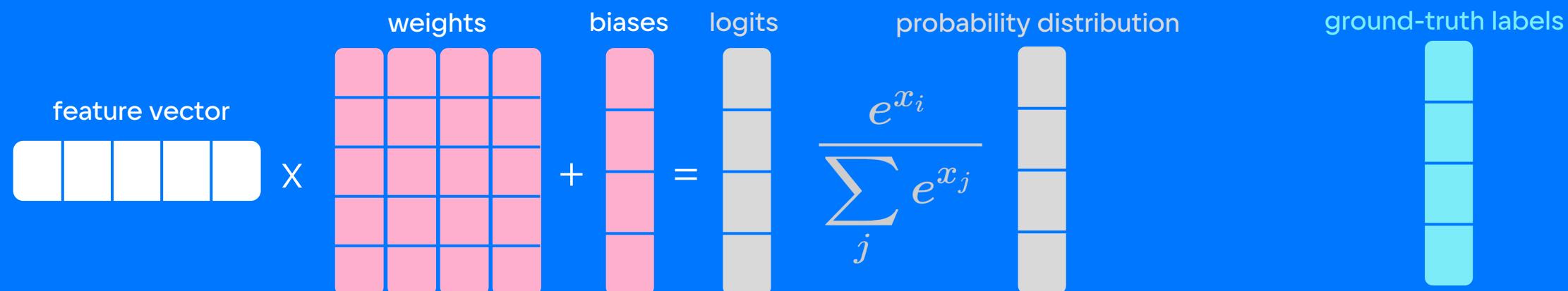
Напоминание: логистическая регрессия для нескольких классов

Решаем задачу классификации на **4** класса. Используем логистическую регрессию.

Для каждого объекта **5** признаков

Сколько параметров модели? — 24 (weights: 5×4 + bias: 4)

На выходе модели будут вещественные числа со знаком (logits). Хотим перевести их в оценки вероятности => softmax



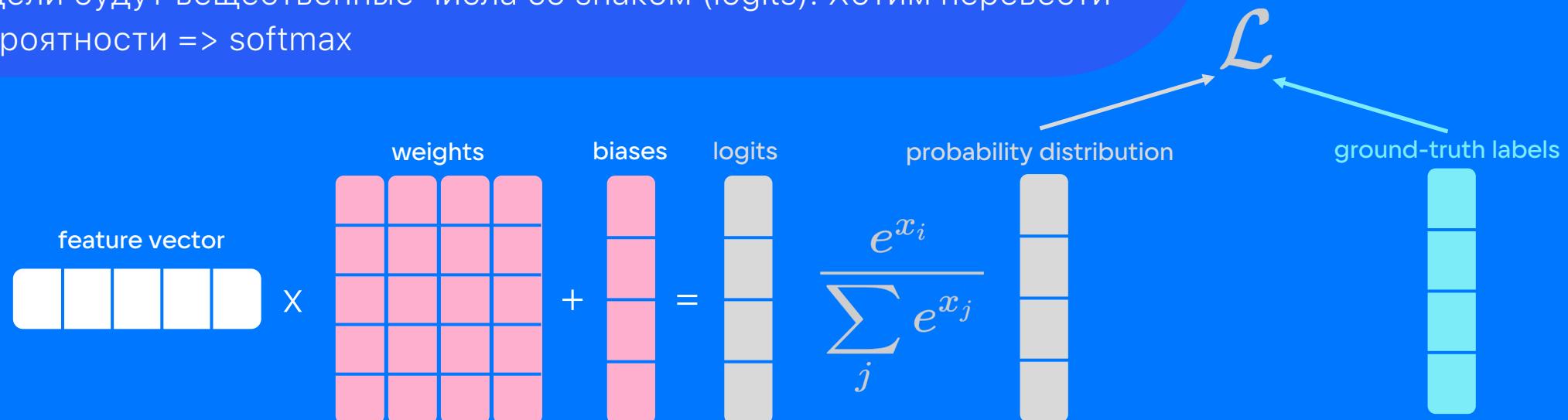
Напоминание: логистическая регрессия для нескольких классов

Решаем задачу классификации на **4** класса. Используем логистическую регрессию.

Для каждого объекта **5** признаков

Сколько параметров модели? — 24 (weights: 5×4 + bias: 4)

На выходе модели будут вещественные числа со знаком (logits). Хотим перевести их в оценки вероятности => softmax

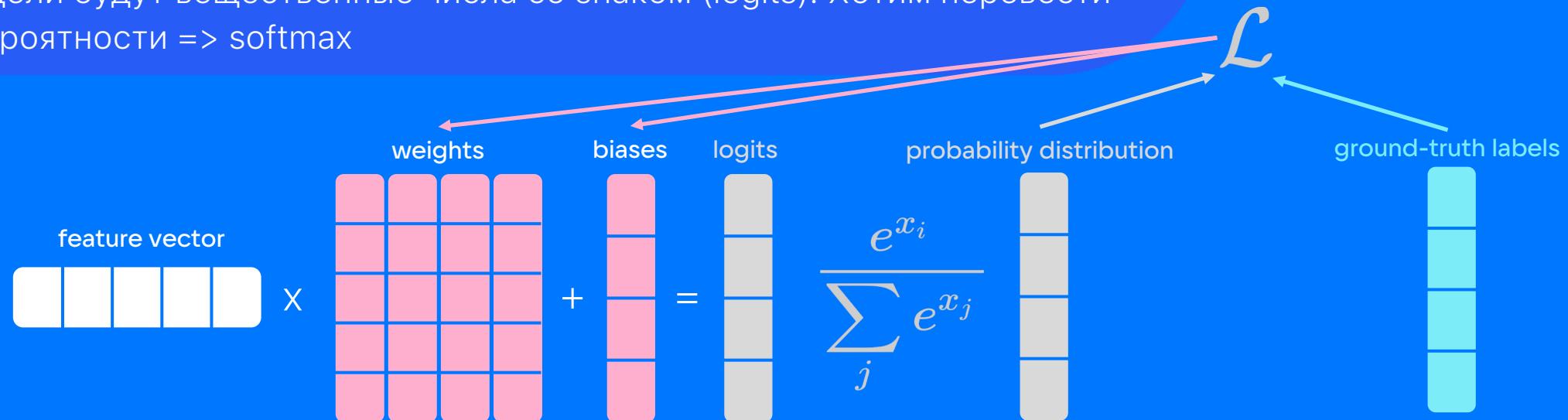


Напоминание: логистическая регрессия для нескольких классов

Решаем задачу классификации на **4** класса. Используем логистическую регрессию.
Для каждого объекта **5** признаков

Сколько параметров модели? — 24 (weights: 5×4 + bias: 4)

На выходе модели будут вещественные числа со знаком (logits). Хотим перевести их в оценки вероятности => softmax



Word2Vec: постановка задачи

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 **к ф** с молоком Как перевести **к ф** ?

Попробуем поставить задачу машинного обучения:
предсказать слово по его окружению

какая это задача?

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

Word2Vec: обучающая выборка

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”

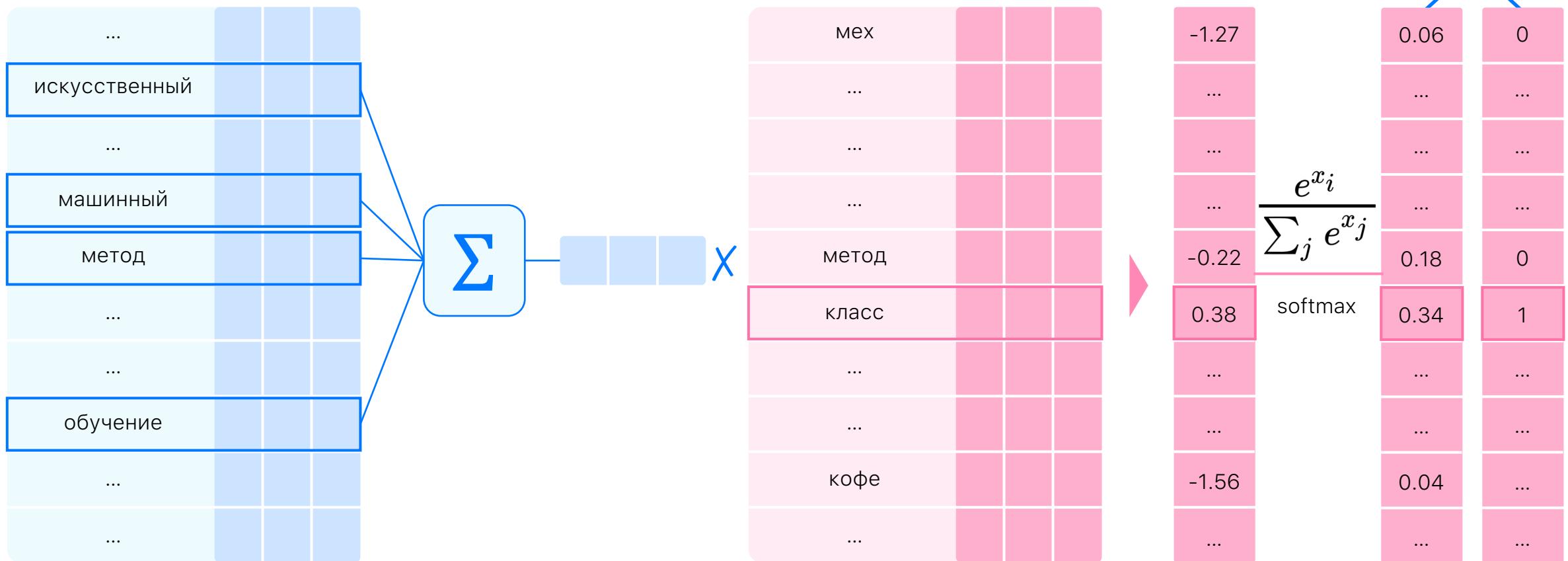
машинный	обучение	класс	метод	искусственный
обучение	класс	метод	искусственный	интеллект
класс	метод	искусственный	интеллект	характерный
метод	искусственный	интеллект	характерный	черта
...

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. 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

Моделирование языка

"All models are wrong, but some are useful" George Box



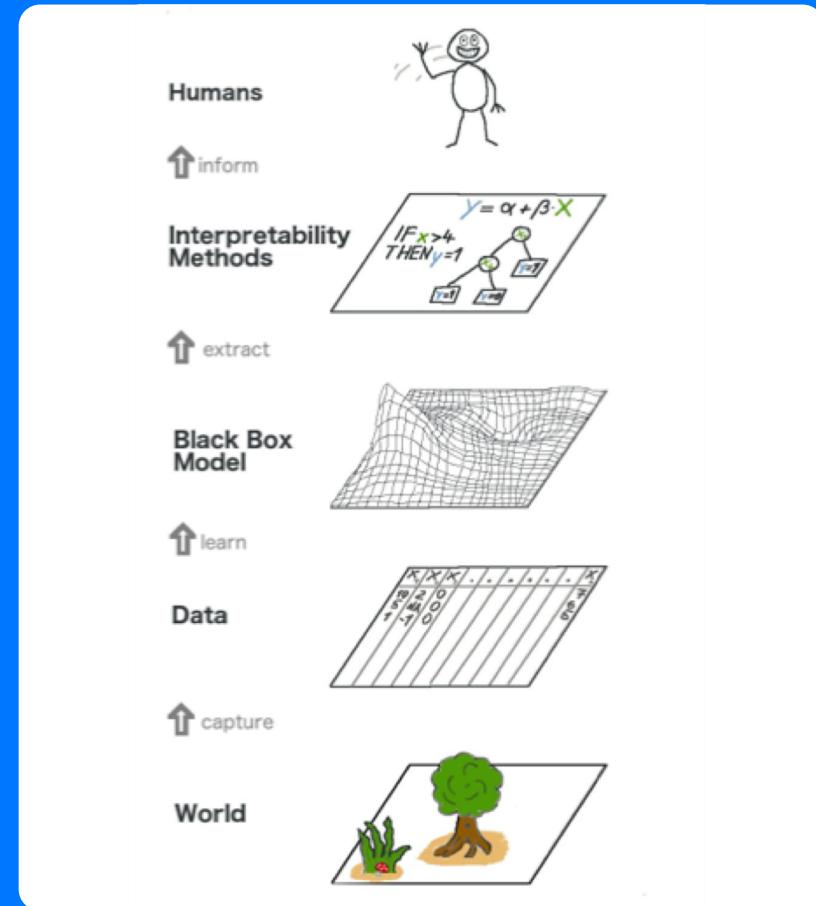
Что такое модель?

Объекты реального мира слишком сложные, для их изучения используются абстрактные представления, описывающие определенные аспекты

атом:

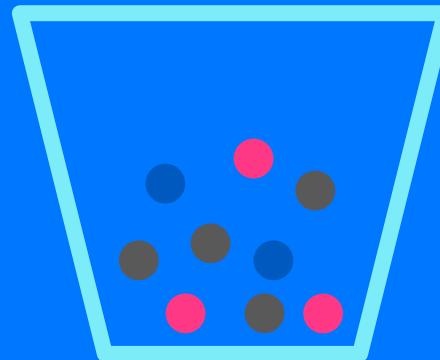
- булочка с изюмом
- планетарная модель
- квант.-мех.

заемщик в банке => вектор признаков



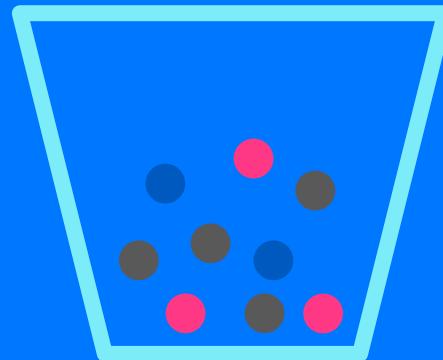
Пример модели: шарики в урне

есть урна с разноцветными шариками, хотим научиться
оценивать вероятность вытащить красный шарик



Пример модели: шарики в урне

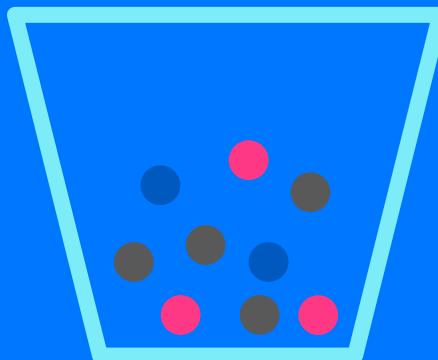
есть урна с разноцветными шариками, хотим научиться оценивать вероятность вытащить красный шарик



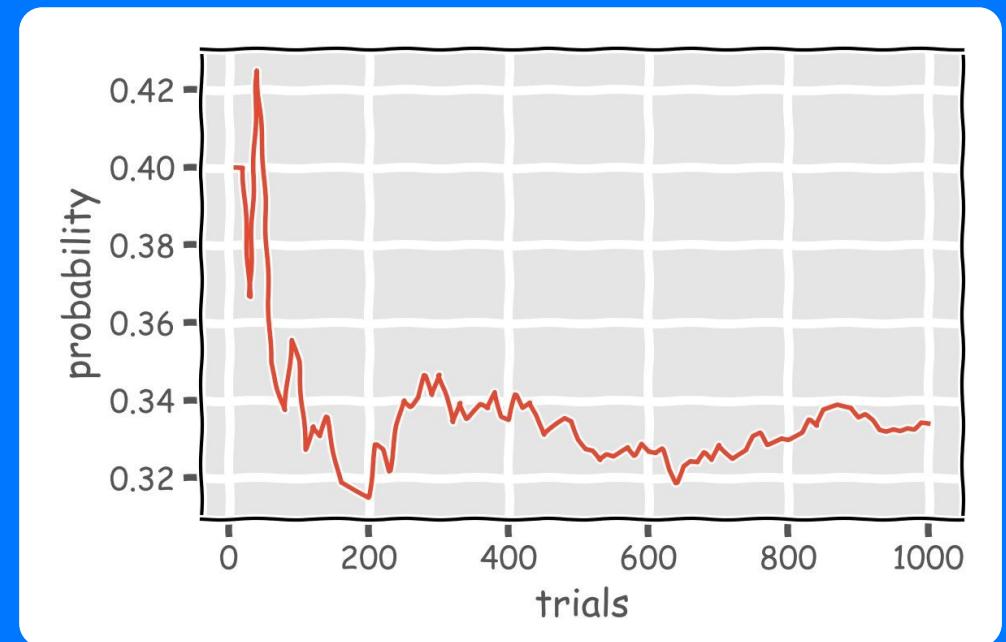
$$P(\text{красный}) = \frac{3}{3+2+4} = \frac{1}{3}$$

Пример модели: шарики в урне

есть урна с разноцветными шариками, хотим научиться оценивать вероятность вытащить красный шарик



$$P(\text{красный}) = \frac{3}{3+2+4} = \frac{1}{3}$$



Пример модели: анализ тональности текста

Sentiment Analysis (определение тональности текстов)
Как мы ее решали?

Пример модели: анализ тональности текста

Sentiment Analysis (определение тональности текстов)

Как мы ее решали?

- 1 разбили текст на слова
- 2 нормализовали (регистр, лемматизация, ...) — предположение
- 3 составили матрицу признаков с помощью tf-idf
(модель BagOfWords) потеряли грамматику языка — предположение
- 4 использовали логистическую регрессию
- 5 получили хорошее качество предсказания => предложенная модель адекватно описывает тональность текстов в изучаемом корпусе

Модель языка: оценка вероятности текста

Задача: построить оценку вероятности предложения:

$$P(\text{“мама мыла раму”}) > P(\text{“vfvf vskf hfve”})$$

Зачем такое может быть нужно?

Модель языка: оценка вероятности текста

Задача: построить оценку вероятности предложения:

$$P(\text{"мама мыла раму"}) > P(\text{"vfvf vskf hfve"})$$

Зачем такое может быть нужно?

- ❖ распознавание речи
 - "бедность не порок" vs "бедность не порог"
 - "услышал скрипка лиса" vs "услышал скрип колеса"

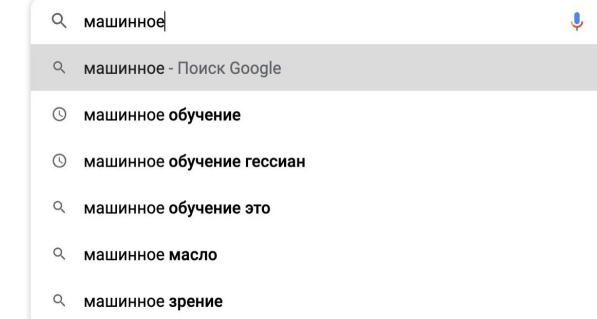
Модель языка: оценка вероятности текста

Задача: построить оценку вероятности предложения:

$$P(\text{"мама мыла раму"}) > P(\text{"vfvf vskf hfve"})$$

Зачем такое может быть нужно?

- ❖ распознавание речи
 - "бедность не порок" vs "бедность не порог"
 - "услышал скрипка лиса" vs "услышал скрип колеса"
- ❖ распознавание речи



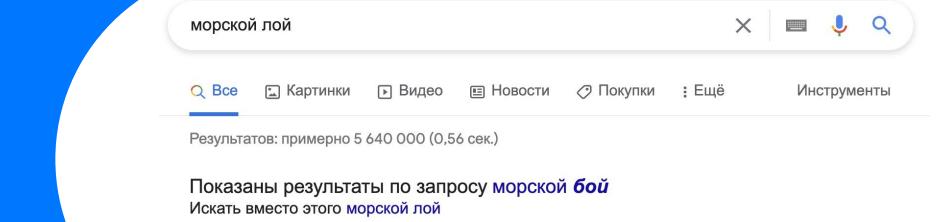
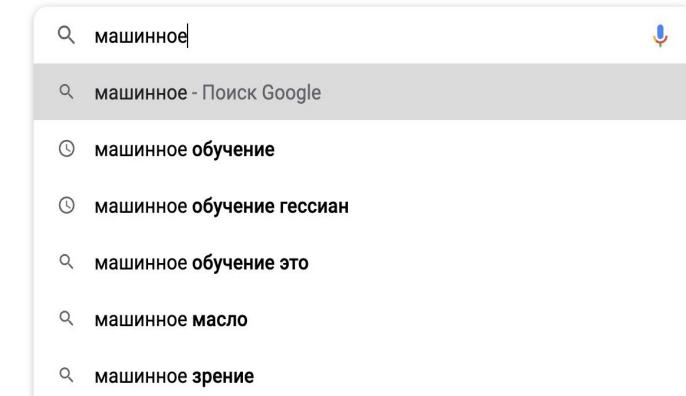
Модель языка: оценка вероятности текста

Задача: построить оценку вероятности предложения:

$$P(\text{"мама мыла раму"}) > P(\text{"vfvf vskf hfve"})$$

Зачем такое может быть нужно?

- ❖ распознавание речи
 - "бедность не порок" vs "бедность не порог"
 - "услышал скрипка лиса" vs "услышал скрип колеса"
- ❖ распознавание речи
- ❖ исправление опечаток
- ❖ ...



Модель языка: шарики в корзинке

Скачиваем интернет, разбиваем тексты на предложения и считаем вероятности аналогично урне с разноцветными шариками

$$P(\text{sentence}) = \frac{1}{|\text{corpus}|} \text{count}(\text{sentence})$$



Здравствуйте!

Даже карманы вшиты таким образом, чтобы не создавать лишних утолщений

Одежда для кукол

Не забывайте ставить лайк подписываться на группу и написать комментарии

Стоит брать?

Еще одно из вероятных последствий - остеоартрит

Здравствуйте будущие арендаторы!

Модель языка: шарики в корзинке проблемы

предложения, которых не было в обучающем корпусе, получат нулевую вероятность

- закон Ципфа => существует длинный хвост редко встречающихся слов => в нашем корпусе статистики либо вообще не будет, либо она будет плохой
- некоторые предложения вполне осмысленные, их вероятность не должна быть нулевой
- гипотезы разной ценности становятся равновероятными

$$P(\text{"глубокое обучение с тензорфлоу"}) = \frac{\text{count}(\text{"глубокое обучение с тензорфлоу"})}{|\text{corpus}|} = 0$$

$$P(\text{"глубокое обучение с tensorflow"}) = \frac{\text{count}(\text{"глубокое обучение с tensorflow"})}{|\text{corpus}|} = 0$$

$$P(\text{"глбкее обуч tenзrфlflw"}) = \frac{\text{count}(\text{"глбкее обуч tenзrфlflw"})}{|\text{corpus}|} = 0$$

Как можем побороть?

Модель языка: шарики в корзинке проблемы

предложения, которых не было в обучающем корпусе, получат нулевую вероятность

- закон Ципфа => существует длинный хвост редко встречающихся слов => в нашем корпусе статистики либо вообще не будет, либо она будет плохой
- некоторые предложения вполне осмысленные, их вероятность не должна быть нулевой
- гипотезы разной ценности становятся равновероятными

$$P(\text{"глубокое обучение с тензорфлоу"}) = \frac{\text{count}(\text{"глубокое обучение с тензорфлоу"})}{|\text{corpus}|} = 0$$

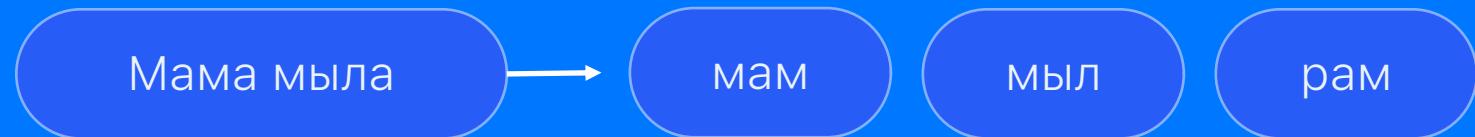
$$P(\text{"глубокое обучение с tensorflow"}) = \frac{\text{count}(\text{"глубокое обучение с tensorflow"})}{|\text{corpus}|} = 0$$

$$P(\text{"глбкее обуч tenзшщорфлв"}) = \frac{\text{count}(\text{"глбкее обуч tenзшщорфлв"})}{|\text{corpus}|} = 0$$

Как можем побороть? нужно переходить к более мелким лингвистическим единицам: токенам

Модель языка: независимые токины

умеем сегментировать предложение на отдельные слова:



$$P(\text{sentence}) = P(w_1, w_2, \dots, w_n) \approx P(w_1) \cdot P(w_2) \cdot \dots \cdot P(w_n)$$

какие проблемы у такого подхода?

Модель языка: независимые токины

умеем сегментировать предложение на отдельные слова:



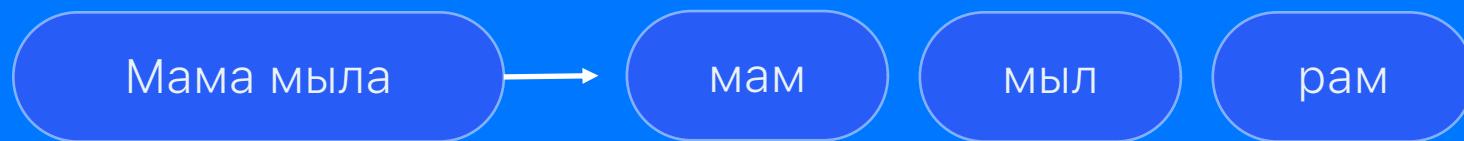
$$P(\text{sentence}) = P(w_1, w_2, \dots, w_n) \approx P(w_1) \cdot P(w_2) \cdot \dots \cdot P(w_n)$$

какие проблемы у такого подхода?

- ❖ потеряю порядок слов (bag of words)
- ❖ какие предложения получат самую высокую вероятность?

Модель языка: независимые токины

умеем сегментировать предложение на отдельные слова:



$$P(\text{sentence}) = P(w_1, w_2, \dots, w_n) \approx P(w_1) \cdot P(w_2) \cdot \dots \cdot P(w_n)$$

какие проблемы у такого подхода?

- ❖ потеряем порядок слов (bag of words)
- ❖ какие предложения получат самую высокую вероятность?
 - "и", "в", "и и", "в и", "в в", "и и и", ...

Модель языка: probability chain rule

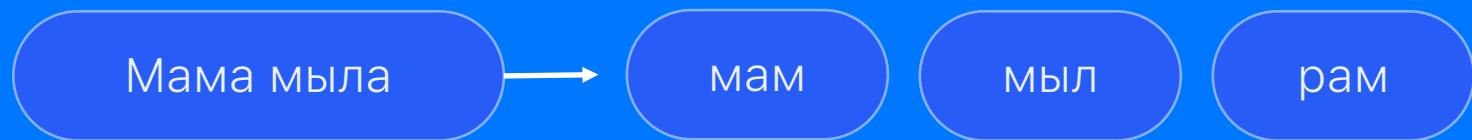
умеем сегментировать предложение на отдельные слова:



$$P(w_1, w_2, w_3) = P(w_3 | w_1, w_2)P(w_1, w_2)$$

Модель языка: probability chain rule

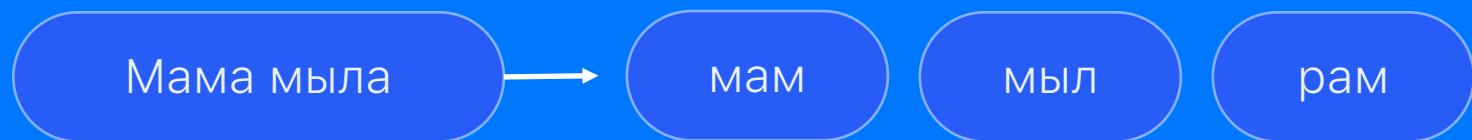
умеем сегментировать предложение на отдельные слова:



$$P(w_1, w_2, w_3) = P(w_3|w_1, w_2)P(w_1, w_2) = P(w_3|w_1, w_2)P(w_2|w_1)P(w_1)$$

Модель языка: probability chain rule

умеем сегментировать предложение на отдельные слова:



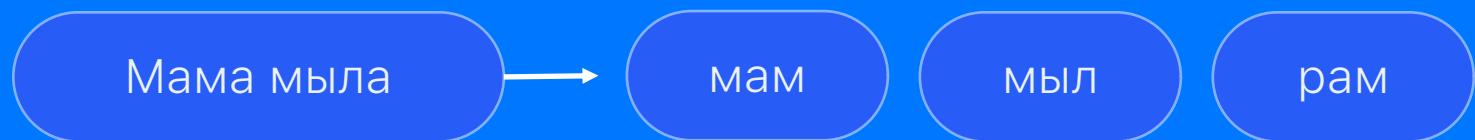
$$P(w_1, w_2, w_3) = P(w_3|w_1, w_2)P(w_1, w_2) = P(w_3|w_1, w_2)P(w_2|w_1)P(w_1)$$

нужно по корпусу текста вычислить статистики:

$$\frac{\text{count(мама, мыла, раму)}}{\text{count(мама, мыла, *)}} \quad \frac{\text{count(мама, мыла)}}{\text{count(мама, *)}}$$

Модель языка: probability chain rule

умеем сегментировать предложение на отдельные слова:



$$P(w_1, w_2, w_3) = P(w_3|w_1, w_2)P(w_1, w_2) = P(w_3|w_1, w_2)P(w_2|w_1)P(w_1)$$

нужно по корпусу текста вычислить статистики:

$$\frac{\text{count(мама, мыла, раму)}}{\text{count(мама, мыла, *)}}$$

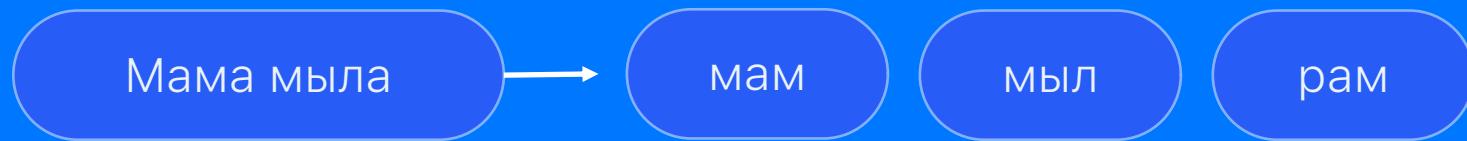
$$\frac{\text{count(мама, мыла)}}{\text{count(мама, *)}}$$

$$P(w_1, w_2, \dots, w_n) = \prod_k P(w_k|w_1, \dots, w_{(k-1)})$$

какие проблемы у такого подхода?

Модель языка: probability chain rule

умеем сегментировать предложение на отдельные слова:



$$P(w_1, w_2, w_3) = P(w_3|w_1, w_2)P(w_1, w_2) = P(w_3|w_1, w_2)P(w_2|w_1)P(w_1)$$

нужно по корпусу текста вычислить статистики:

$$\frac{\text{count(мама, мыла, раму)}}{\text{count(мама, мыла, *)}} \quad \frac{\text{count(мама, мыла)}}{\text{count(мама, *)}}$$

$$P(w_1, w_2, \dots, w_n) = \prod_k P(w_k|w_1, \dots, w_{(k-1)})$$

какие проблемы у такого подхода?

- придется считать вероятность всего предложения как в модели шариков в корзинке :)

Введение в машинное обучение с помощью sklearn

$$\frac{\text{count(Введение, в, машинное, обучение, с помощью, sklearn)}}{\text{count(Введение, в, машинное, с помощью, *)}}$$

Модель языка: Markov assumption

гипотеза: слово в предложении зависит только от предыдущих N слов

Модель языка: Markov assumption

гипотеза: слово в предложении зависит только от предыдущих N слов

$$P(w_1, w_2, \dots, w_n) = \prod_k P(w_k | w_1, \dots, w_{(k-1)}) \approx \prod_k P(w_k | w_{(k-N+1)}, \dots, w_{(k-1)})$$

Модель языка: Markov assumption

гипотеза: слово в предложении зависит только от предыдущих N слов

$$P(w_1, w_2, \dots, w_n) = \prod_k P(w_k | w_1, \dots, w_{(k-1)}) \approx \prod_k P(w_k | w_{(k-N+1)}, \dots, w_{(k-1)})$$

$P(\text{введение в машинное обучение с помощью sklearn}) = P(\text{sklearn} | \text{введение в машинное обучение с помощью}) \times P(\text{помощью} | \text{введение в машинное обучение с })$
 $\times P(\text{с} | \text{введение в машинное обучение}) \times P(\text{обучение} | \text{введение в машинное}) \times$
 $P(\text{машинное} | \text{введение в})$
 $\times P(\text{в} | \text{введение}) \times$
 $P(\text{введение})$

Модель языка: Markov assumption

гипотеза: слово в предложении зависит только от предыдущих N слов

$$P(w_1, w_2, \dots, w_n) = \prod_k P(w_k | w_1, \dots, w_{(k-1)}) \approx \prod_k P(w_k | w_{(k-N+1)}, \dots, w_{(k-1)})$$

$P(\text{введение в машинное обучение с помощью sklearn}) = P(\text{sklearn} | \text{введение в машинное обучение с помощью}) \times P(\text{помощь} | \text{введение в машинное обучение с })$
 $\times P(\text{с} | \text{введение в машинное обучение}) \times P(\text{обучение} | \text{введение в машинное}) \times P(\text{машинное} | \text{введение в})$
 $\times P(\text{в} | \text{введение}) \times P(\text{введение})$

Модель языка: Markov assumption

гипотеза: слово в предложении зависит только от предыдущих N слов

$$P(w_1, w_2, \dots, w_n) = \prod_k P(w_k | w_1, \dots, w_{(k-1)}) \approx \prod_k P(w_k | w_{(k-N+1)}, \dots, w_{(k-1)})$$

$P(\text{введение в машинное обучение с помощью sklearn}) = P(\text{sklearn} | \text{введение в машинное обучение с помощью}) \times P(\text{помощь} | \text{введение в машинное обучение с })$
 $\times P(\text{с} | \text{введение в машинное обучение}) \times P(\text{обучение} | \text{введение в машинное}) \times P(\text{машинное} | \text{введение в })$
 $\times P(\text{в} | \text{введение}) \times P(\text{введение}) \times P(\text{введение})$

придется считать только статистику по тройкам слов!

Count (с помощью sklearn)

Count (обучение с помощью)

Модель языка: N-gram Language Model (Statistical Language Model)

N = 1 (unigram LM)

$$P(w_1, \dots, w_n) = \prod_{k=1}^n P(w_k)$$

N = 2 (bigram LM)

$$P(w_1, \dots, w_n) = \prod_{k=1}^n P(w_k | w_{k-1})$$

N = 3 (trigram LM)

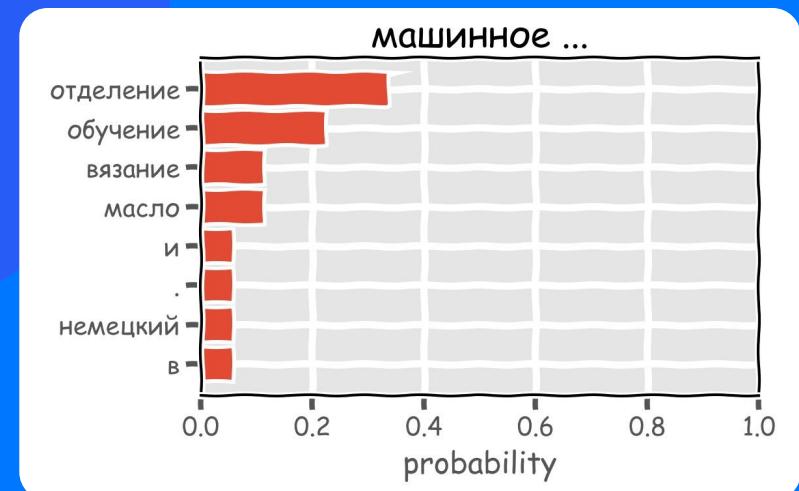
$$P(w_1, \dots, w_n) = \prod_{k=1}^n P(w_k | w_{k-2}, w_{k-1})$$

$$P(w_t | w_{t-n+1}, \dots, w_{t-1}) \approx \frac{\text{count}(w_{t-n+1} \dots, w_{t-1}, w_t)}{\sum_w \text{count}(w_{t-n+1} \dots, w_{t-1}, w)} = \frac{\text{count}(w_{t-n+1} \dots, w_{t-1}, w_t)}{\text{count}(w_{t-n+1} \dots, w_{t-1})}$$

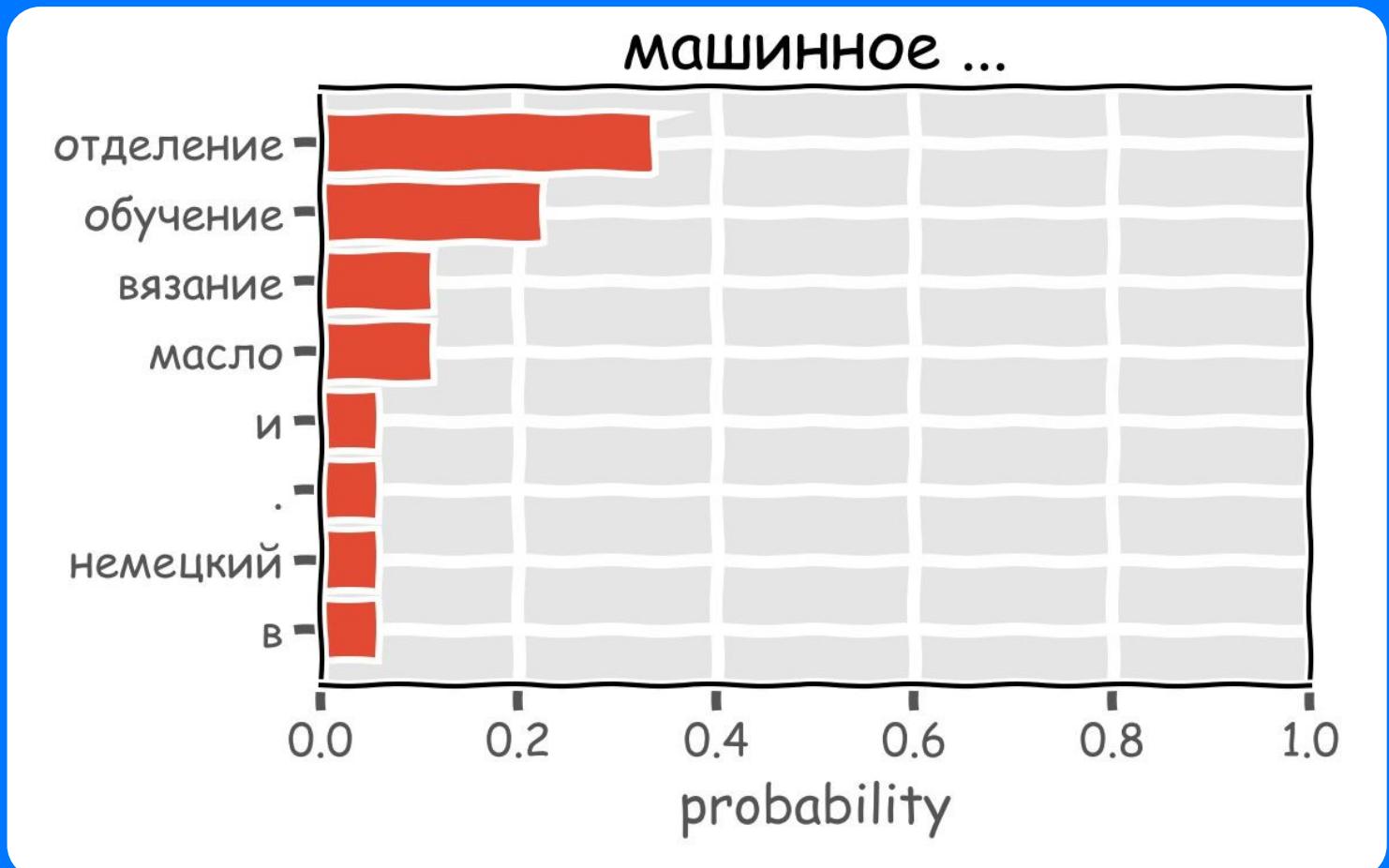
Модель языка: N-gram Language Model (Statistical Language Model)

Итак, обучение:

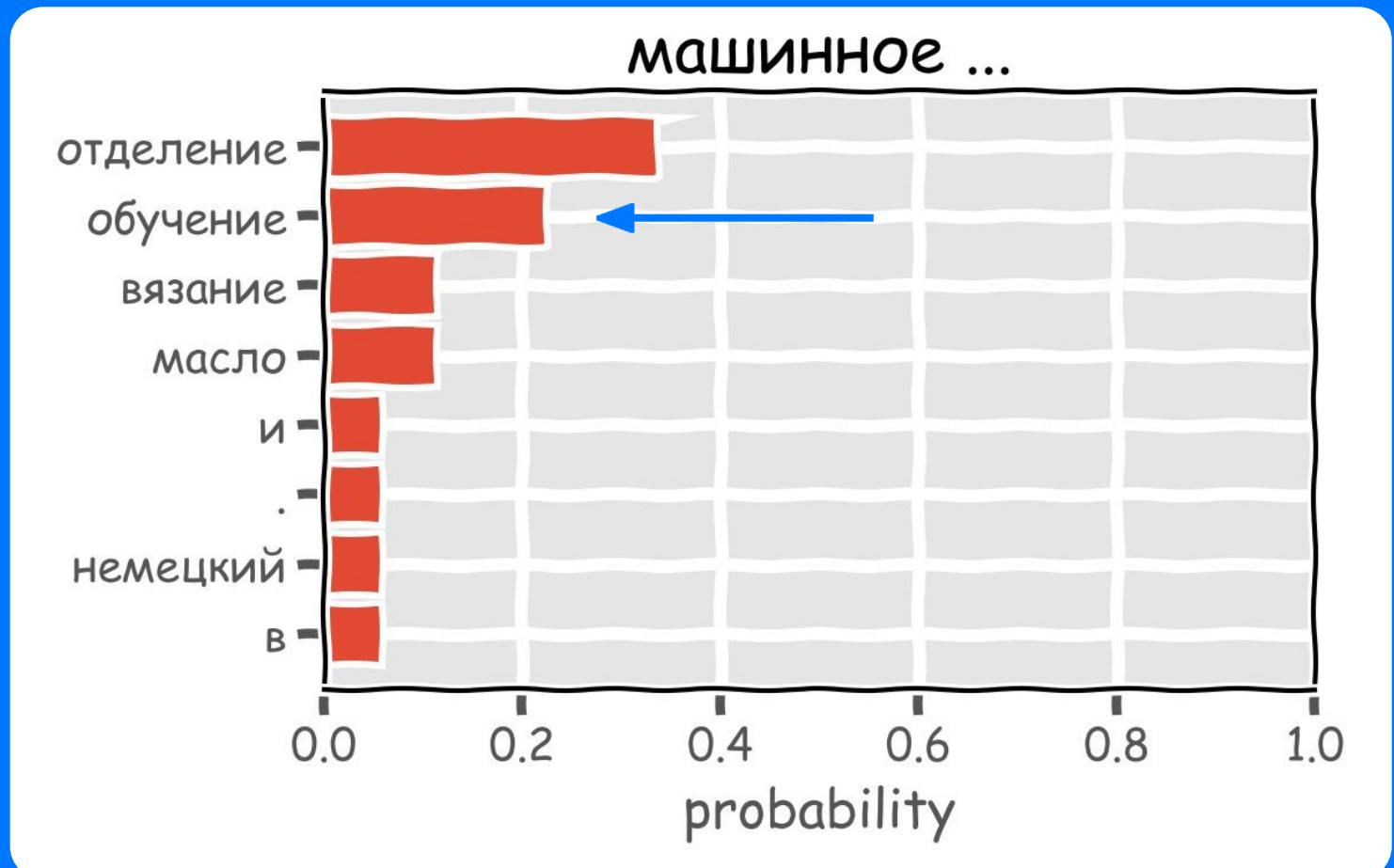
- фиксируем порядок (N) языковой модели
- токенизуем коллекцию текстов, добавляем спец. символы (BOS & EOS | <s> &</s>)
- считаем, сколько раз встречались последовательности длины N и (N - 1)
- оцениваем вероятность следующего слова при условии контекста длины (N - 1)



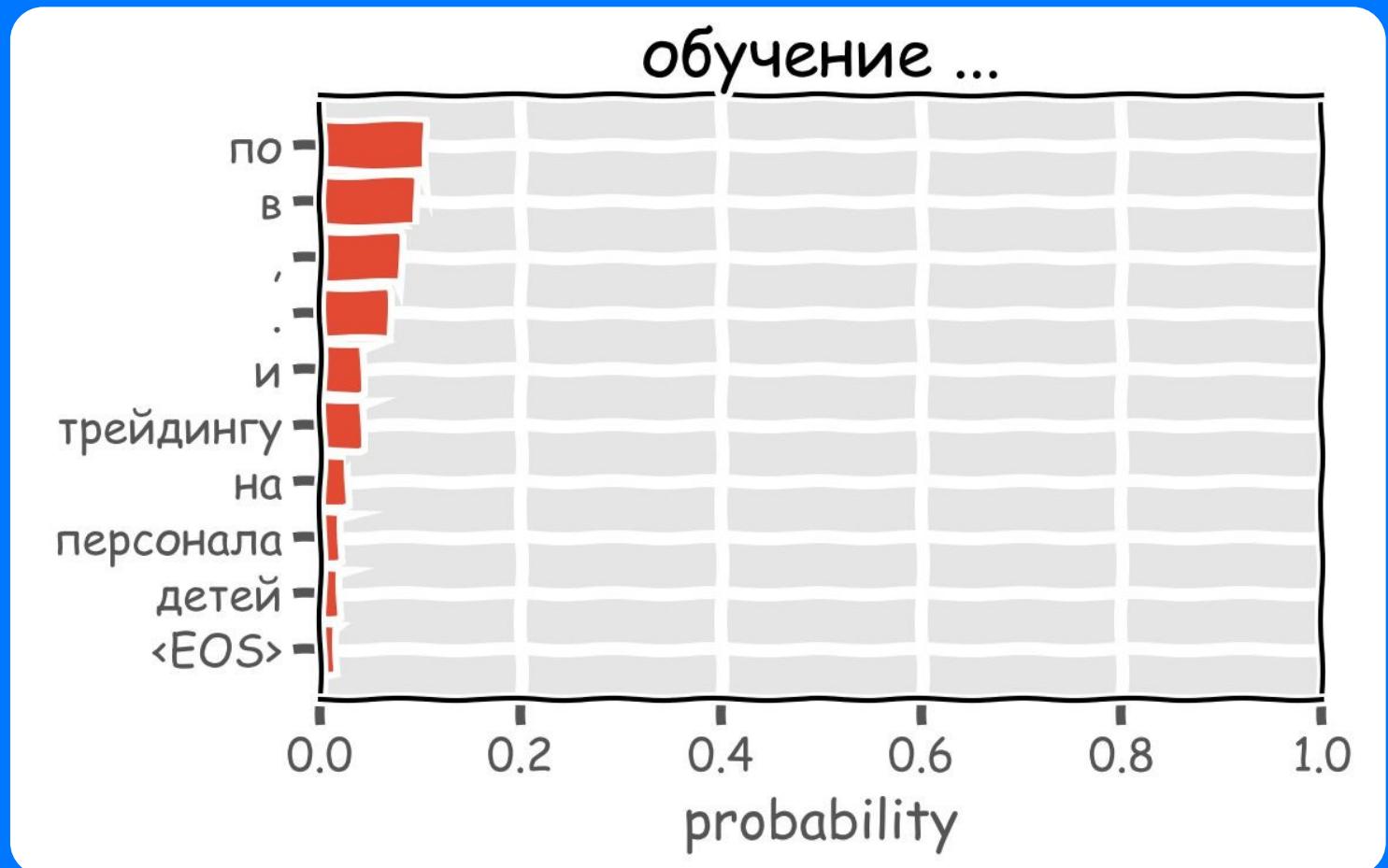
Генерация текста: машинаное обучение



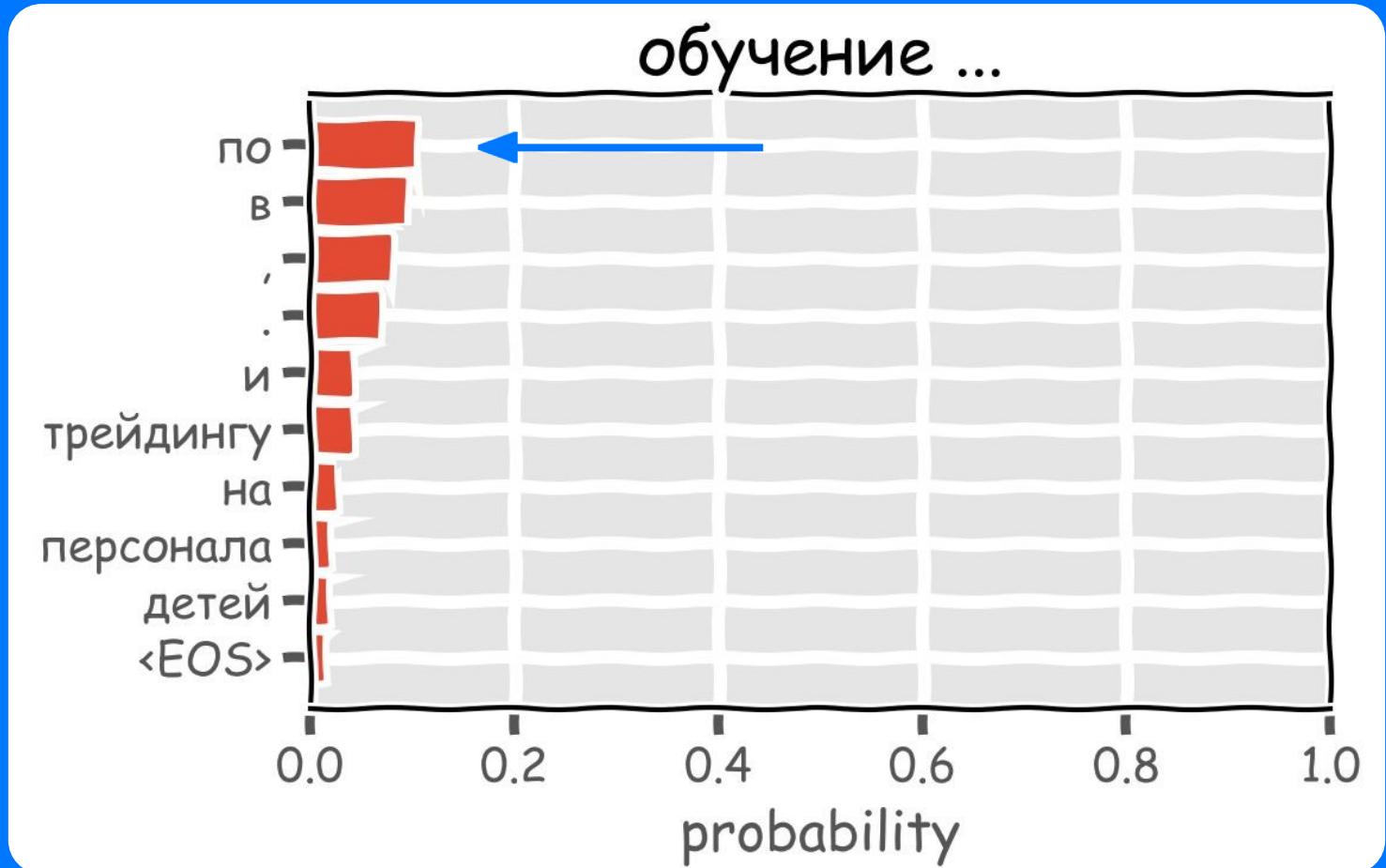
Генерация текста: машинаное обучение



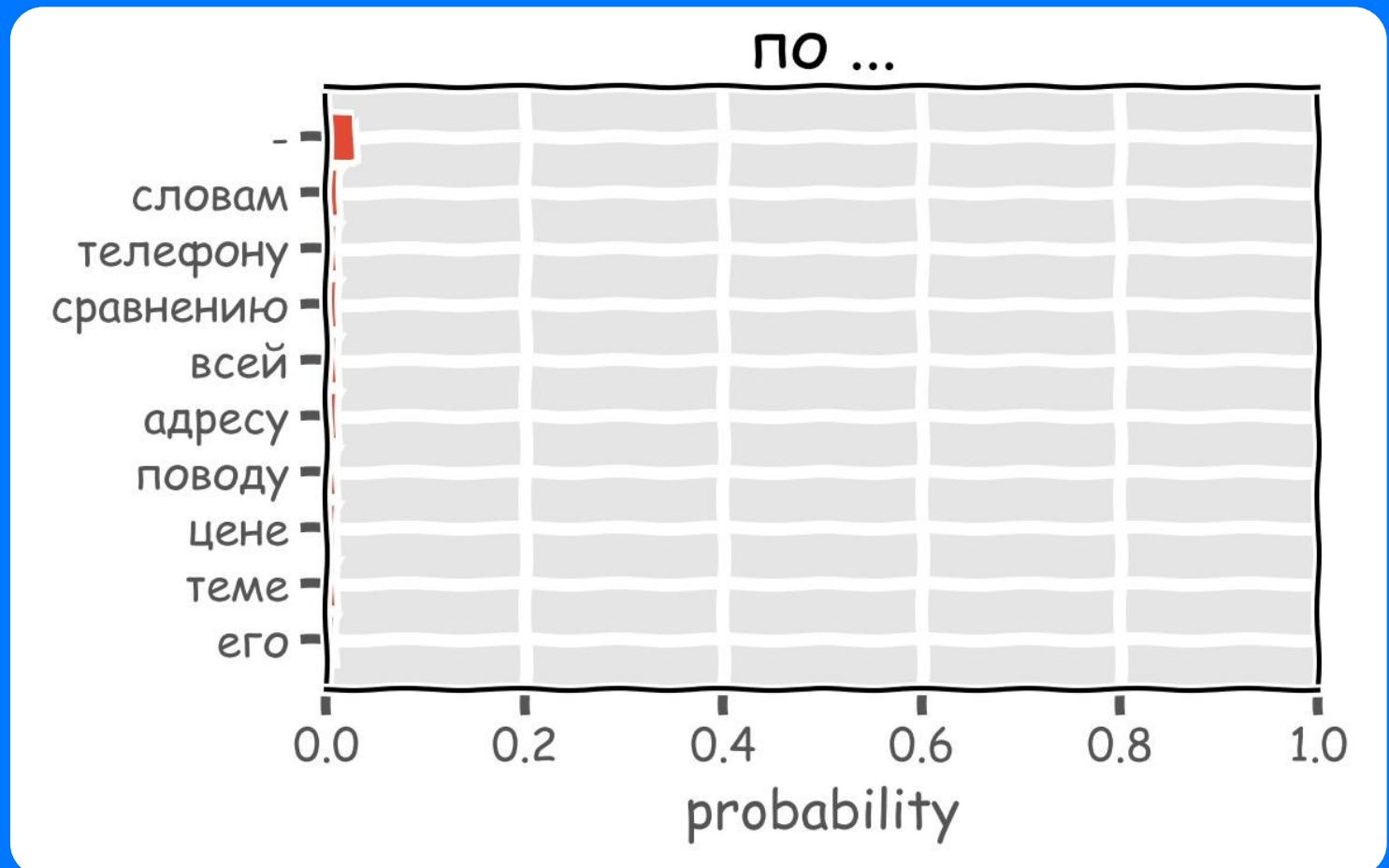
Генерация текста: машинаное обучение



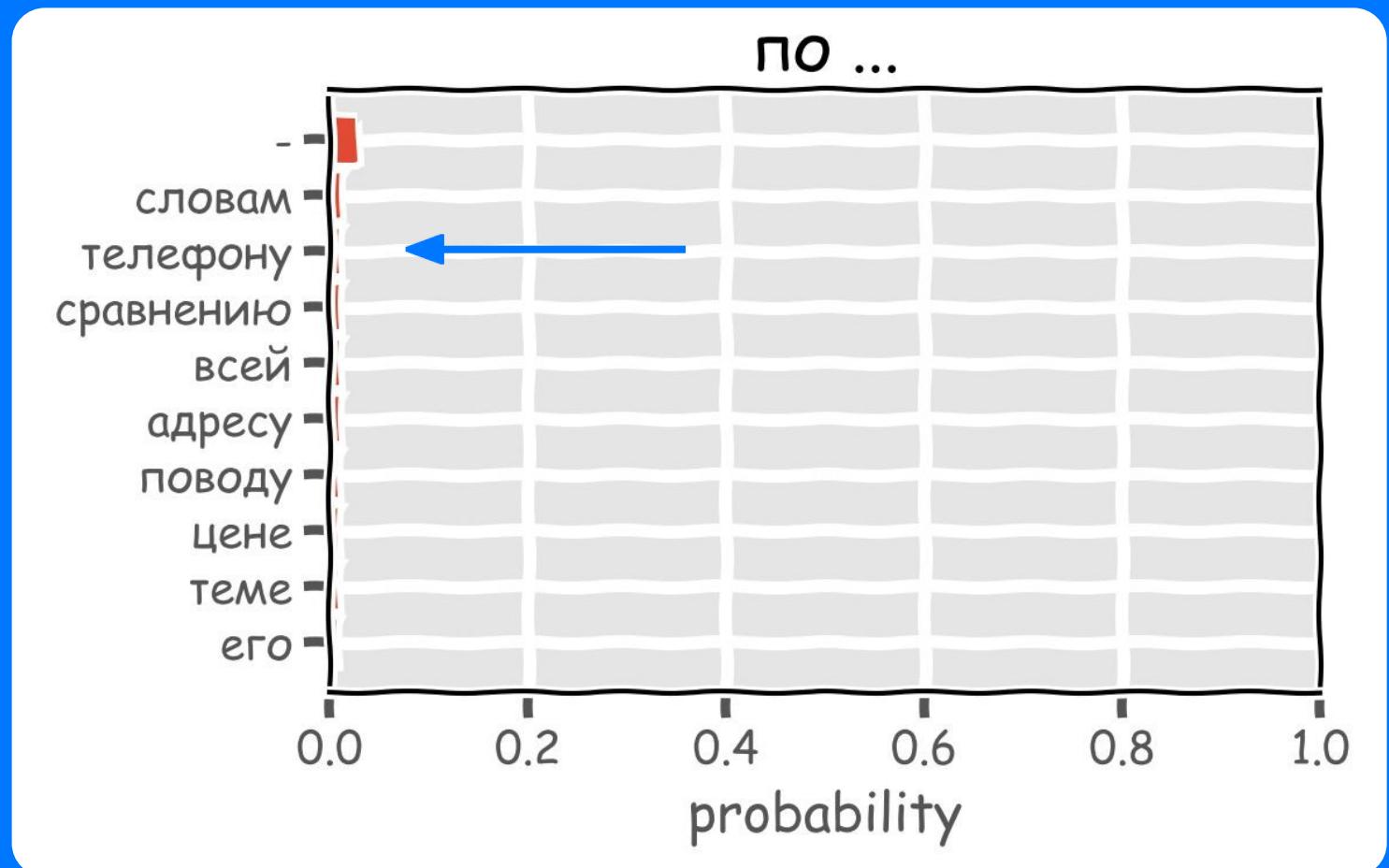
Генерация текста: машинаное обучение



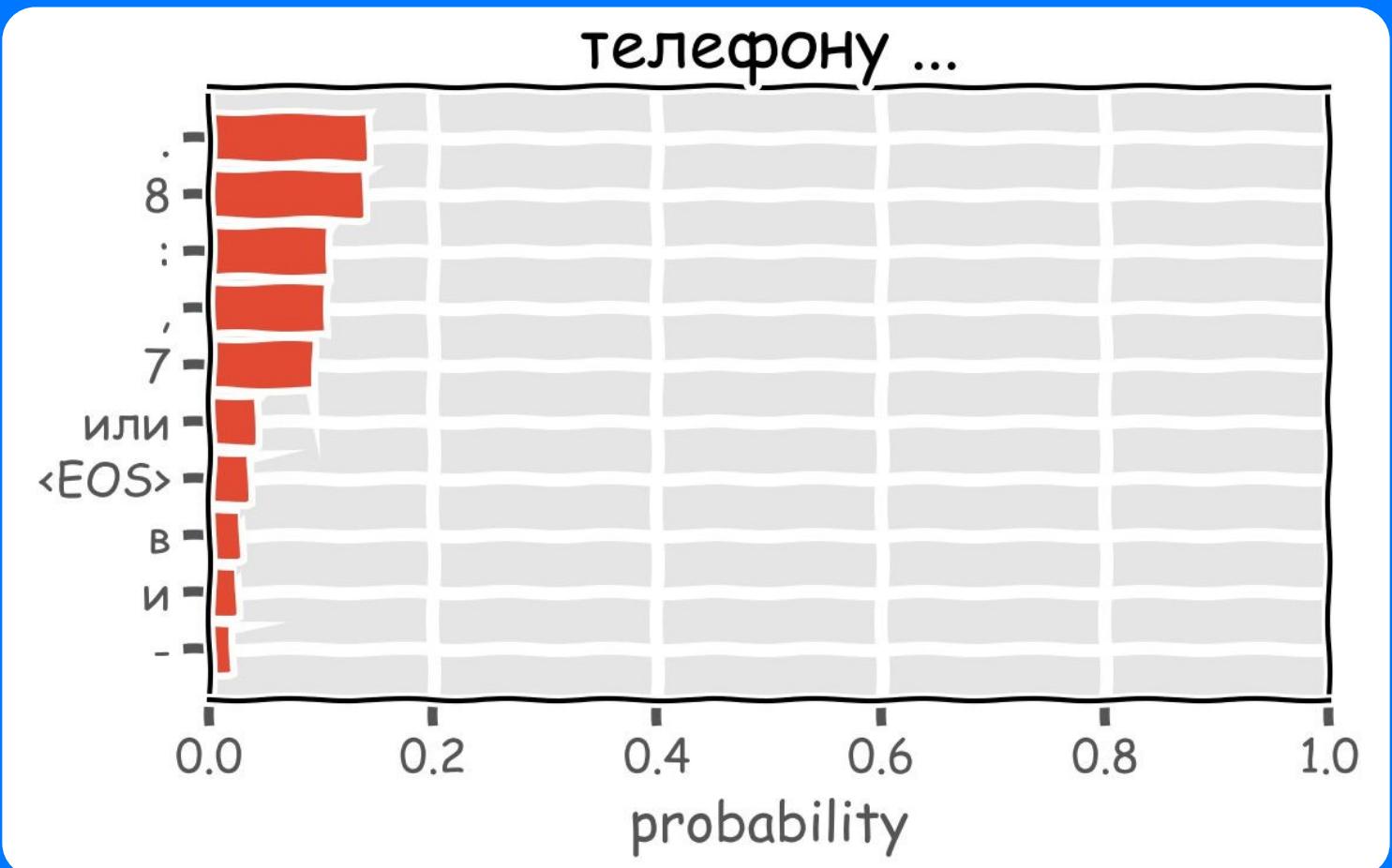
Генерация текста: машинаное обучение по



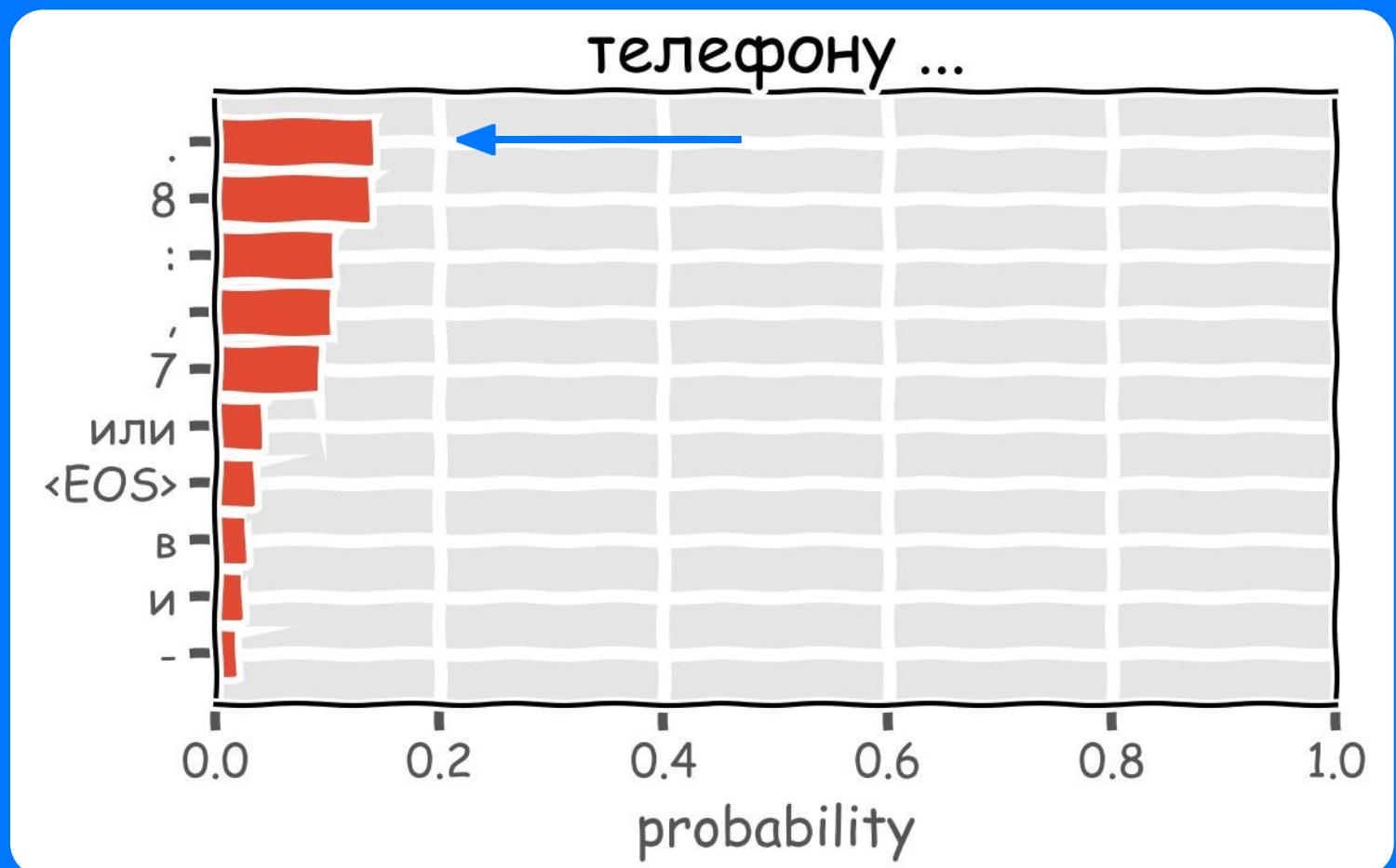
Генерация текста: машинное обучение по



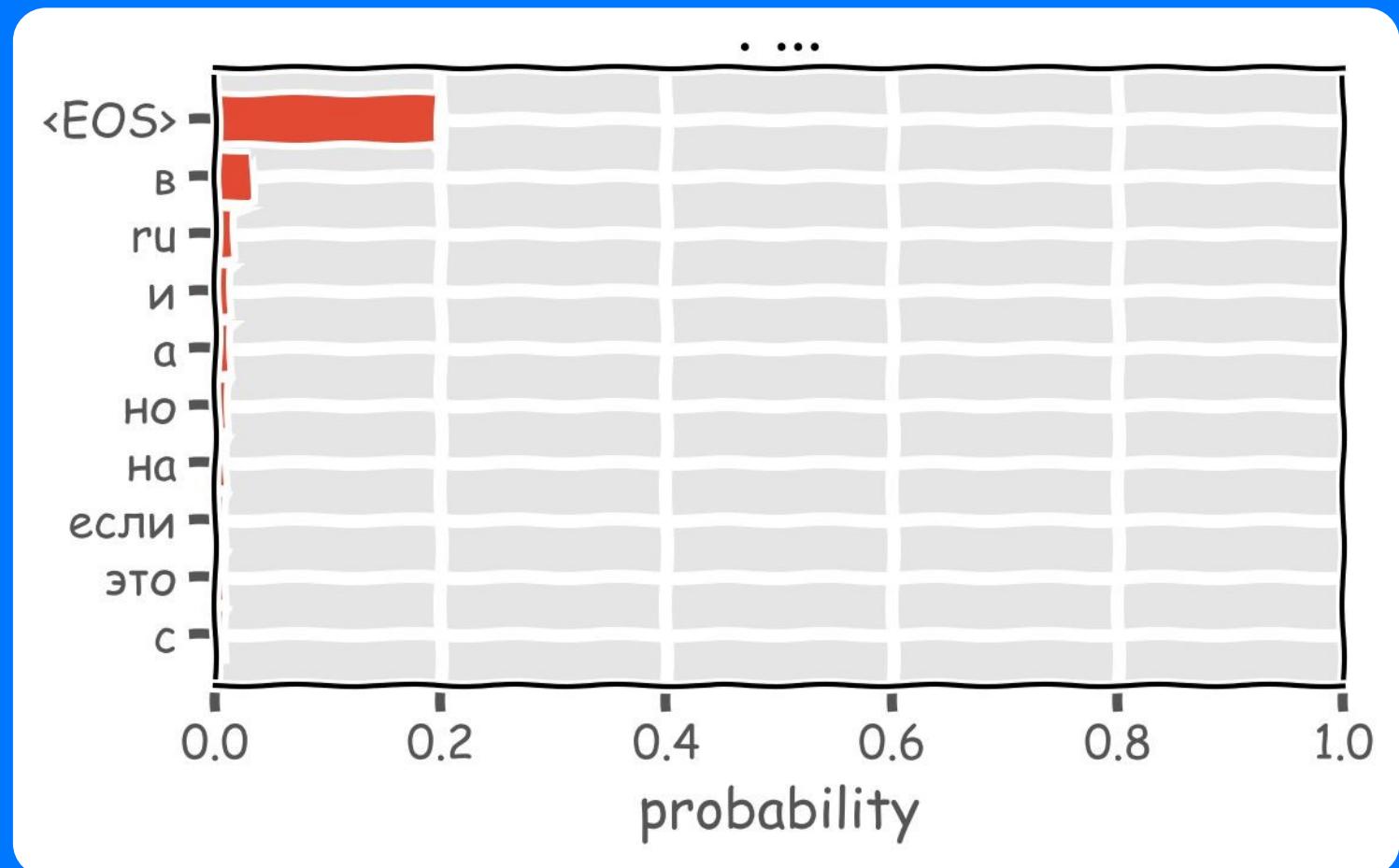
Генерация текста: машинное обучение по телефону



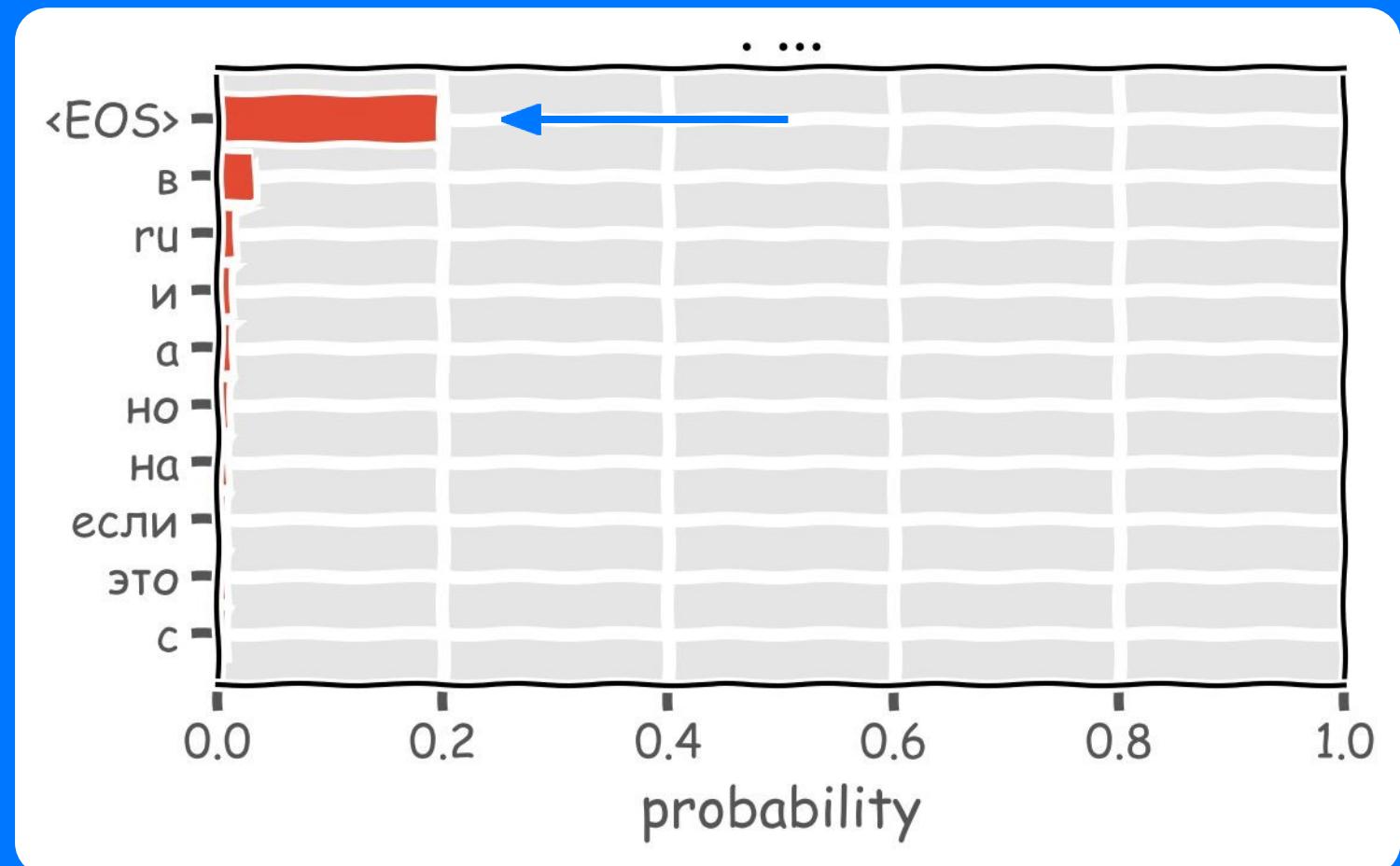
Генерация текста: машинное обучение по телефону



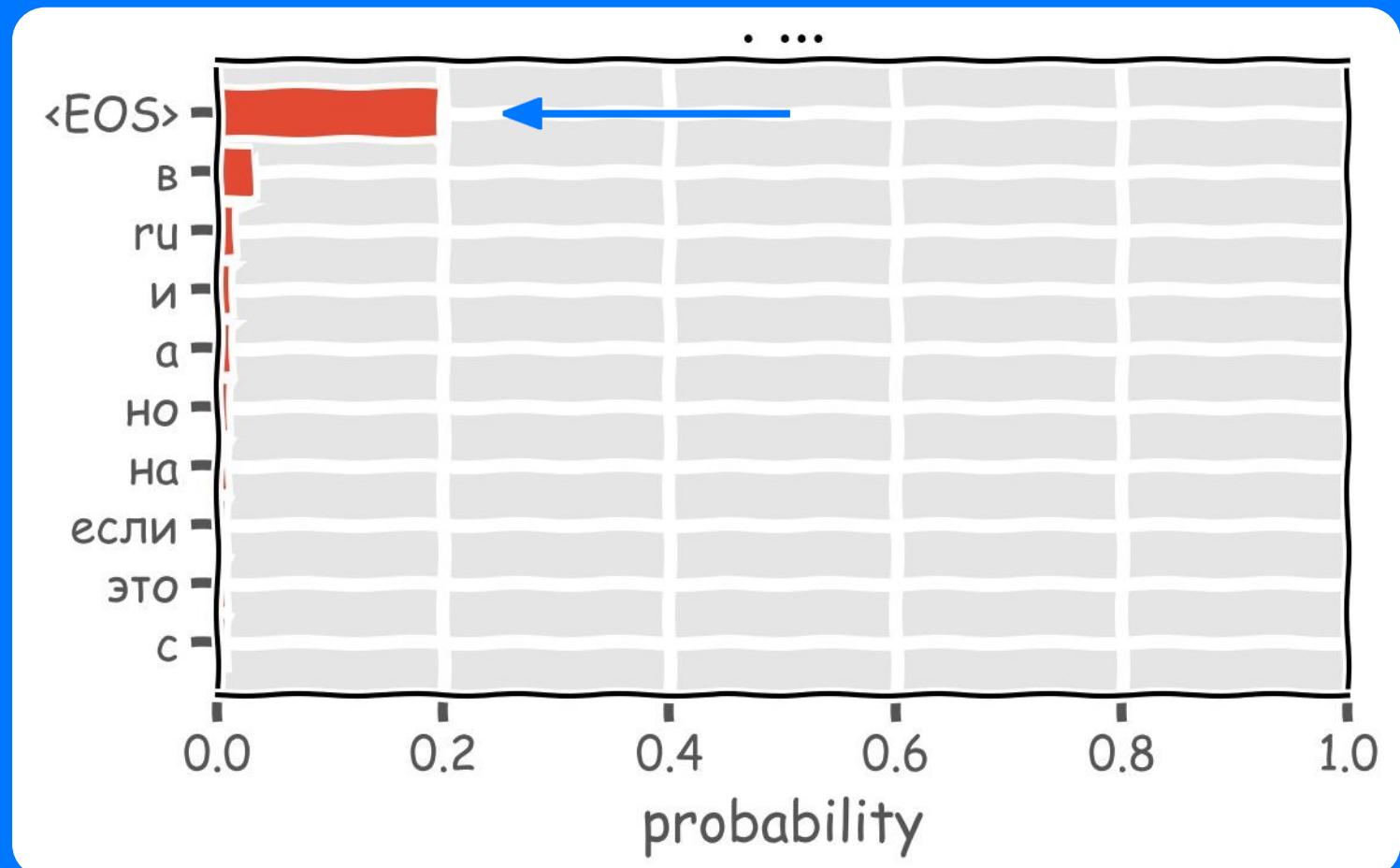
Генерация текста: машинное обучение по телефону



Генерация текста: машинное обучение по телефону



Генерация текста: машинное обучение по телефону. <EOS>



Генерация текста: чего хотим от языковых моделей?

Генерация текста: чего хотим от языковых моделей?

- Связности (coherency)
- Разнообразия (diversity)

Генерация текста: чего хотим от языковых моделей?

- Связности (coherency)
- Разнообразия (diversity)

Для увеличения связности можно брать более высокие порядки, увеличивать обучающий корпус, переходить от статистических к нейросетевым языковым моделям

Для увеличения разнообразия можно применить сэмплирование с температурой

Генерация текста: чего хотим от языковых моделей?

- Связности (coherency)
- Разнообразия (diversity)

Для увеличения связности можно брать более высокие порядки, увеличивать обучающий корпус, переходить от статистических к нейросетевым языковым моделям

Для увеличения разнообразия можно применить сэмплирование с температурой

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

$$\text{softmax}(x, T)_i = \frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x_j}{T}}}$$

Генерация текста: чего хотим от языковых моделей?

- Связности (coherency)
- Разнообразия (diversity)

Для увеличения связности можно брать более высокие порядки, увеличивать обучающий корпус, переходить от статистических к нейросетевым языковым моделям

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

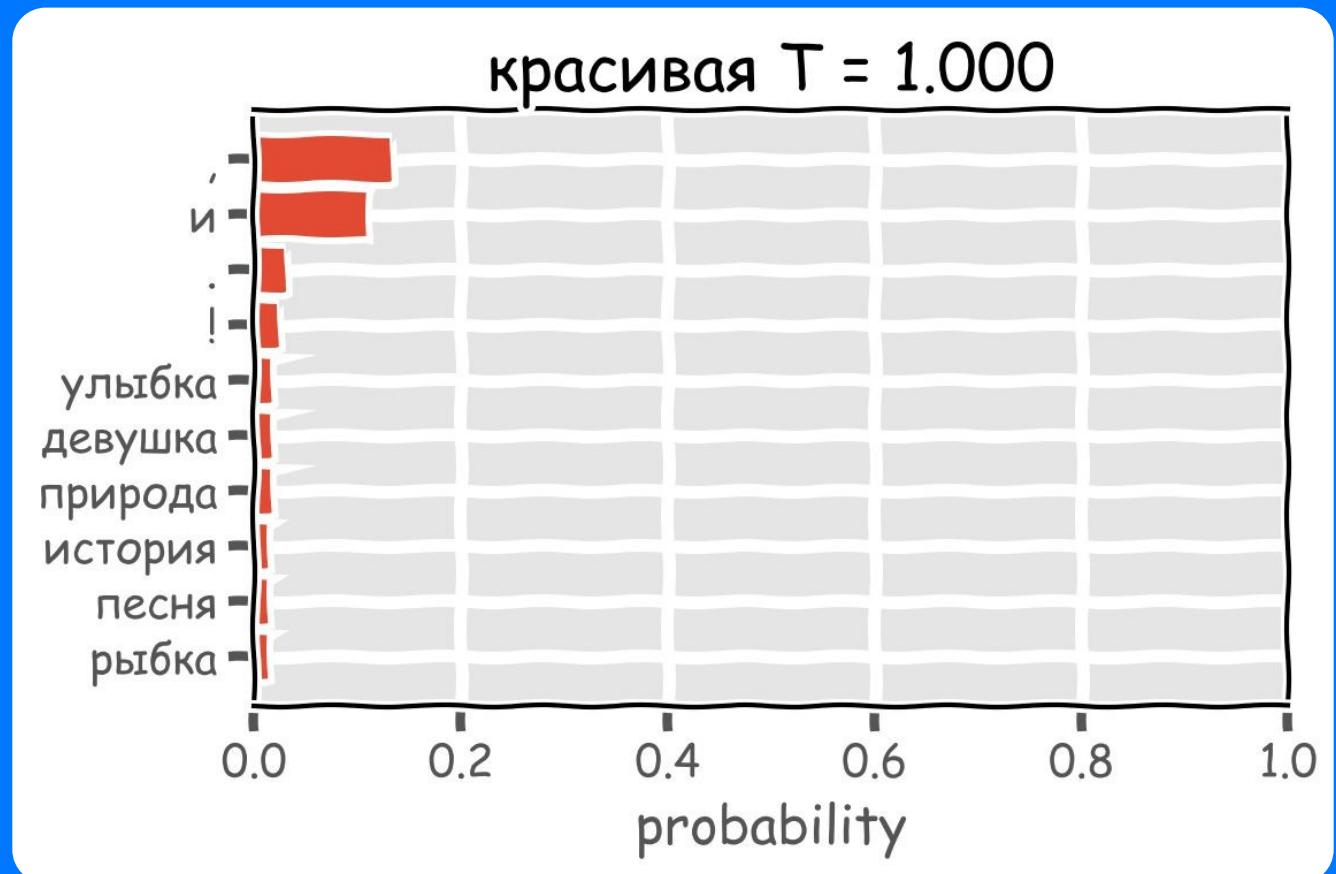
$$\text{softmax}(x, T)_i = \frac{e^{\frac{x_i}{T}}}{\sum_j e^{\frac{x_j}{T}}}$$

Для увеличения разнообразия можно применить сэмплирование с температурой

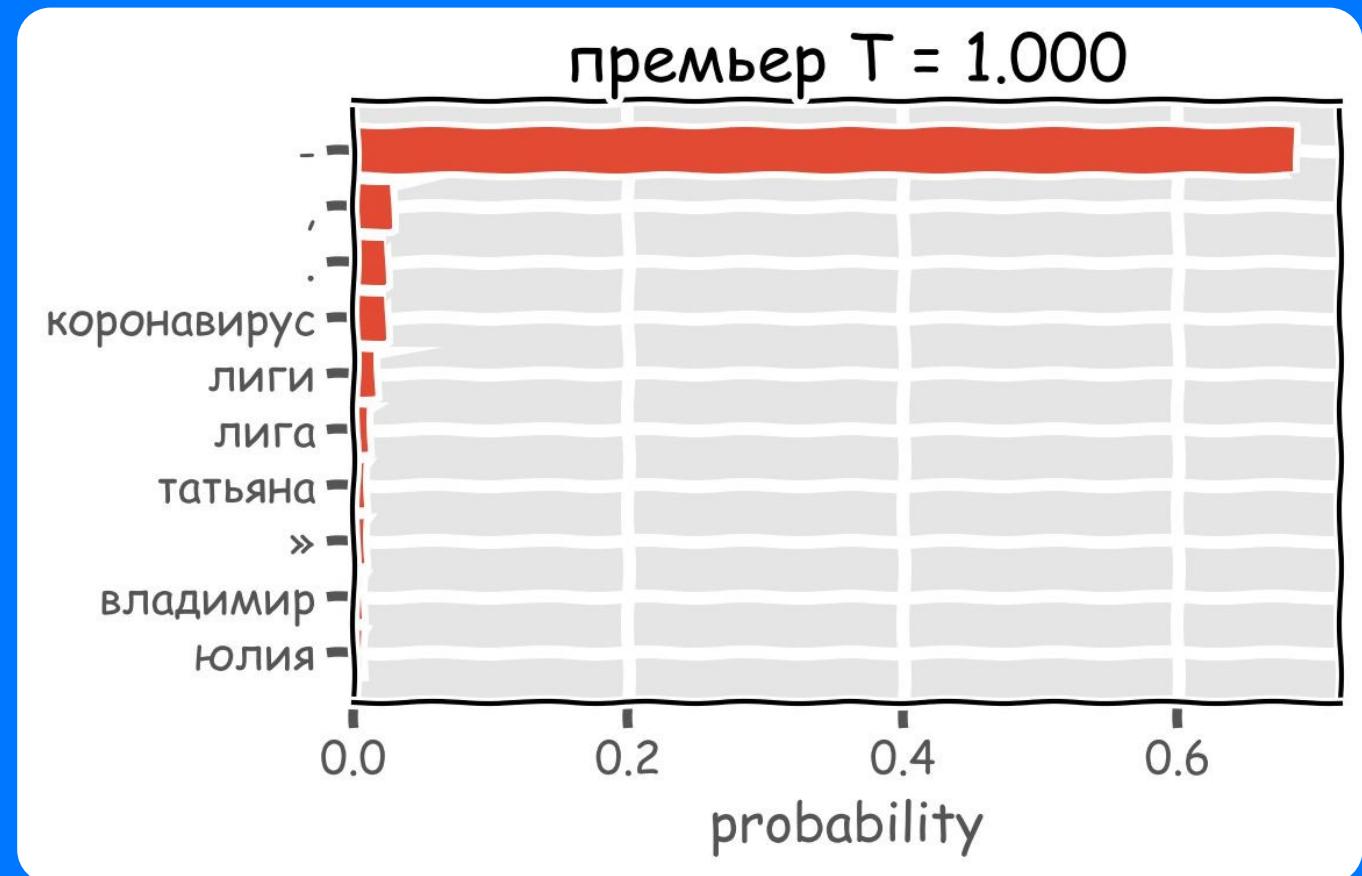
$$p = e^{\log p}$$

$$p_T \sim e^{\frac{\log p}{T}} = e^{\frac{1}{T} \log p} = p^{\frac{1}{T}}$$

Генерация текста: понижение температура



Генерация текста: понижение температуры



Примеры сгенерированных текстов

машинальное обучение с помощью которого вы и ваши друзья примеряете шляпу индианы джонса и отправляетесь на встречу приключениям в самое сердце женщины

машинальное обучение с помощью scikit learn и tensorflow концепции , инструменты и садовая техника , товары для спорта и охоты , устройство конструкции и последовательность сборки своими руками , для настольной лампы своими руками из природных материалов .

машинальное обучение с помощью и командлетов можно довольно просто скопировать членство в группах одного пользователя и добавить в бульон пока варится картофель займемся всеми грибами

машинальное обучение с помощью scikit learn и tensorflow концепции , инструменты и технологии поддержки инновационного предпринимательства на базе технопарков , а также показали золото и бронзу паралимпиады

Интерпретация языковых моделей



bigram language model

говорить ...

$$\begin{aligned} P("о" | "говорить") &\sim 0.30 \\ P("с" | "говорить") &\sim 0.02 \\ P("над" | "говорить") &= 0 \end{aligned}$$

красивая ...

$$\begin{aligned} P("жизнь" | "красивая") &\sim 0.03 \\ P("песня" | "красивая") &\sim 0.01 \\ P("улыбка" | "красивая") &\sim 0.01 \\ P("умная" | "красивая") &= 0 \end{aligned}$$

bigram language model

говорить ...

$$\begin{aligned} P("о" | "говорить") &\sim 0.30 \\ P("с" | "говорить") &\sim 0.02 \\ P("над" | "говорить") &= 0 \end{aligned}$$

красивая ...

$$\begin{aligned} P("жизнь" | "красивая") &\sim 0.03 \\ P(" песня" | "красивая") &\sim 0.01 \\ P(" улыбка" | "красивая") &\sim 0.01 \\ P("умная" | "красивая") &= 0 \end{aligned}$$

модель выучивает грамматику:

- после глагола вероятен предлог
- говорят обычно о чем-то и с кем-то (предлог "над" неуместен)
- после прилагательного "красивый" следует существительное (а не еще одно прилагательное)
- красивых вещей в мире много => вероятностная масса для префикса "красивая" размазана по разным словам

bigram language model

говорить ...

$$P("о" | "говорить") \sim 0.30$$

$$P("с" | "говорить") \sim 0.02$$

$$P("над" | "говорить") = 0$$

красивая ...

$$P("жизнь" | "красивая") \sim 0.03$$

$$P("песня" | "красивая") \sim 0.01$$

$$P("улыбка" | "красивая") \sim 0.01$$

$$P("умная" | "красивая") = 0$$

слушать ...

$$P("музыку" | "слушать") \sim 0.05$$

$$P("аудиокнигу" | "слушать") \sim 0.01$$

модель выучивает грамматику:

- после глагола вероятен предлог
- говорят обычно о чем-то и с кем-то (предлог "над" неуместен)
- после прилагательного "красивый" следует существительное (а не еще одно прилагательное)
- красивых вещей в мире много => вероятностная масса для префикса "красивая" размазана по разным словам

bigram language model

говорить ...

$$P("о" | "говорить") \sim 0.30$$

$$P("с" | "говорить") \sim 0.02$$

$$P("над" | "говорить") = 0$$

красивая ...

$$P("жизнь" | "красивая") \sim 0.03$$

$$P("песня" | "красивая") \sim 0.01$$

$$P("улыбка" | "красивая") \sim 0.01$$

$$P("умная" | "красивая") = 0$$

слушать ...

$$P("музыку" | "слушать") \sim 0.05$$

$$P("аудиокнигу" | "слушать") \sim 0.01$$

модель выучивает грамматику:

- после глагола вероятен предлог
- говорят обычно о чем-то и с кем-то (предлог "над" неуместен)
- после прилагательного "красивый" следует существительное (а не еще одно прилагательное)
- красивых вещей в мире много => вероятностная масса для префикса "красивая" размазана по разным словам

модель выучивает знания о мире:

- музыку слушают чаще, чем аудиокниги

bigram language model

президент россии ...

$P(\text{"владимир"} \mid \text{"президент россии"}) \sim 0.62$
 $P(\text{"дмитрий"} \mid \text{"президент россии"}) \sim 0.08$

президент бразилии ...

$P(\text{"жайр"} \mid \text{"президент бразилии"}) \sim 0.28$
 $P(\text{"мишел"} \mid \text{"президент бразилии"}) \sim 0.24$
 $P(\text{"луис"} \mid \text{"президент бразилии"}) \sim 0.02$

модель выучивает факты:

(хоть иногда и устаревшие, определяются
корпусом обучающих текстов)

Сглаживание вероятностей

• • • •



N-gram language model

гипотеза: слово в предложении зависит только от предыдущих N слов

$$P(w_1, w_2, \dots, w_n) = \prod_k P(w_k | w_1, \dots, w_{(k-1)}) \approx \prod_k P(w_k | w_{(k-N+1)}, \dots, w_{(k-1)})$$

$$P(w_t | w_{t-n+1}, \dots, w_{t-1}) \approx \frac{\text{count}(w_{t-n+1} \dots, w_{t-1}, w_t)}{\sum_w \text{count}(w_{t-n+1} \dots w_{t-1}, w)} = \frac{\text{count}(w_{t-n+1} \dots, w_{t-1}, w_t)}{\text{count}(w_{t-n+1} \dots w_{t-1})}$$

N-gram language model

гипотеза: слово в предложении зависит только от предыдущих N слов

$$P(w_1, w_2, \dots, w_n) = \prod_k P(w_k | w_1, \dots, w_{(k-1)}) \approx \prod_k P(w_k | w_{(k-N+1)}, \dots, w_{(k-1)})$$

$$P(w_t | w_{t-n+1}, \dots, w_{t-1}) \approx \frac{\text{count}(w_{t-n+1} \dots, w_{t-1}, w_t)}{\sum_w \text{count}(w_{t-n+1} \dots w_{t-1}, w)} = \frac{\text{count}(w_{t-n+1} \dots, w_{t-1}, w_t)}{\text{count}(w_{t-n+1} \dots w_{t-1})}$$

- встречаемость последовательностей слов оцениваем по корпусу текстов
- после обучения модели к нам поступают предложения, вероятности которых нужно оценить

N-gram language model

гипотеза: слово в предложении зависит только от предыдущих N слов

$$P(w_1, w_2, \dots, w_n) = \prod_k P(w_k | w_1, \dots, w_{(k-1)}) \approx \prod_k P(w_k | w_{(k-N+1)}, \dots, w_{(k-1)})$$

$$P(w_t | w_{t-n+1}, \dots, w_{t-1}) \approx \frac{\text{count}(w_{t-n+1} \dots, w_{t-1}, w_t)}{\sum_w \text{count}(w_{t-n+1} \dots w_{t-1}, w)} = \frac{\text{count}(w_{t-n+1} \dots, w_{t-1}, w_t)}{\text{count}(w_{t-n+1} \dots w_{t-1})}$$

- встречаемость последовательностей слов оцениваем по корпусу текстов
- после обучения модели к нам поступают предложения, вероятности которых нужно оценить
- что произойдет, если в новом предложении встретится пара словосочетаний, которые никогда раньше не встречались?

N-gram language model

гипотеза: слово в предложении зависит только от предыдущих N слов

$$P(w_1, w_2, \dots, w_n) = \prod_k P(w_k | w_1, \dots, w_{(k-1)}) \approx \prod_k P(w_k | w_{(k-N+1)}, \dots, w_{(k-1)})$$

$$P(w_t | w_{t-n+1}, \dots, w_{t-1}) \approx \frac{\text{count}(w_{t-n+1} \dots, w_{t-1}, w_t)}{\sum_w \text{count}(w_{t-n+1} \dots w_{t-1}, w)} = \frac{\text{count}(w_{t-n+1} \dots, w_{t-1}, w_t)}{\text{count}(w_{t-n+1} \dots w_{t-1})}$$

- встречаемость последовательностей слов оцениваем по корпусу текстов
- после обучения модели к нам поступают предложения, вероятности которых нужно оценить
- что произойдет, если в новом предложении встретится пара словосочетаний, которые никогда раньше не встречались?

В числителе и/или знаменателе формулы для оценки вероятности будет стоять ноль

Laplace Smoothing

$$P(w_1, w_2, \dots, w_n) = \prod_k P(w_k | w_1, \dots, w_{(k-1)}) \approx \prod_k P(w_k | w_{(k-N+1)}, \dots, w_{(k-1)})$$

$$P(w_t | w_{t-n+1}, \dots, w_{t-1}) \approx \frac{\text{count}(w_{t-n+1} \dots, w_{t-1}, w_t) + 1}{\sum_w (\text{count}(w_{t-n+1} \dots w_{t-1}, w) + 1)} = \frac{\text{count}(w_{t-n+1} \dots, w_{t-1}, w_t) + 1}{\text{count}(w_{t-n+1} \dots w_{t-1}) + |V|}$$

Laplace Smoothing

$$P(w_1, w_2, \dots, w_n) = \prod_k P(w_k | w_1, \dots, w_{(k-1)}) \approx \prod_k P(w_k | w_{(k-N+1)}, \dots, w_{(k-1)})$$

$$P(w_t | w_{t-n+1}, \dots, w_{t-1}) \approx \frac{\text{count}(w_{t-n+1} \dots, w_{t-1}, w_t) + 1}{\sum_w (\text{count}(w_{t-n+1} \dots w_{t-1}, w) + 1)} = \frac{\text{count}(w_{t-n+1} \dots, w_{t-1}, w_t) + 1}{\text{count}(w_{t-n+1} \dots w_{t-1}) + |V|}$$

- перераспределяем часть вероятностной массы на события, которые не наблюдали в ходе обучения
- есть ли проблемы?

Delta Smoothing

$$P(w_1, w_2, \dots, w_n) = \prod_k P(w_k | w_1, \dots, w_{(k-1)}) \approx \prod_k P(w_k | w_{(k-N+1)}, \dots, w_{(k-1)})$$

$$P(w_t | w_{t-n+1}, \dots, w_{t-1}) \approx \frac{\text{count}(w_{t-n+1} \dots, w_{t-1}, w_t) + \delta}{\sum_w (\text{count}(w_{t-n+1} \dots, w_{t-1}, w) + \delta)} = \frac{\text{count}(w_{t-n+1} \dots, w_{t-1}, w_t) + \delta}{\text{count}(w_{t-n+1} \dots, w_{t-1}) + \delta|V|}$$

- вносим меньшее искажение в истинные оценки

Другие подходы к сглаживанию

- Stupid Backoff: используем модель порядка N, если нам попадается неизвестная N-грамма, то пробуем оценить вероятность с помощью языковой модели (N - 1) порядка, то есть уменьшаем длину контекста

Другие подходы к сглаживанию

- Stupid Backoff: используем модель порядка N, если нам попадается неизвестная N-грамма, то пробуем оценить вероятность с помощью языковой модели (N - 1) порядка, то есть уменьшаем длину контекста

$$\hat{P}(w_4|w_1, w_2, w_3) = \lambda_4 P(w_4|w_1, w_2, w_3) + \lambda_3 P(w_4|w_2, w_3) + \lambda_2 P(w_4|w_3) + \lambda_1 P(w_4)$$

- Linear Interpolation

$$\sum_i \lambda_i = 1$$

Другие подходы к сглаживанию

- Stupid Backoff: используем модель порядка N, если нам попадается неизвестная N-грамма, то пробуем оценить вероятность с помощью языковой модели (N - 1) порядка, то есть уменьшаем длину контекста

$$\hat{P}(w_4|w_1, w_2, w_3) = \lambda_4 P(w_4|w_1, w_2, w_3) + \lambda_3 P(w_4|w_2, w_3) + \lambda_2 P(w_4|w_3) + \lambda_1 P(w_4)$$

- Linear Interpolation

$$\sum_i \lambda_i = 1$$

- Kneser-Ney / Witten-Bell — наиболее популярные, учитывают коллокации

Оценка качества

• • • •



Подходы к оценке качества

Extrinsic (внешние)

- обучаем модель для решения практической задачи
 - по метрикам качества этой задачи можно понять, хорошо ли работает модель
-
- + позволяет явно оценить, приносит ли компонент пользу
 - долго

Intrinsic (внутренние)

- подготовить тестовые датасеты, не связанные с конечной задачей
 - для эмбеддингов слов [RuSimLex 365](#)
 - для языковых моделей перплексия
-
- + относительно быстро
 - оцениваем качество модели в отрыве от конечной задачи

Перплексия

- perplexity: недоумение, замешательство, растерянность
- считается на отдельном наборе данных (dev / test), который модель не видела (хотим обобщения, а не заучивания)
- вероятность, нормированная на длину текста

$$P(w_1, \dots, w_N)^{-\frac{1}{N}} = \left(\prod_t P(w_t | w_{t-n+1}, \dots, w_{t-1}) \right)^{-\frac{1}{N}} = \frac{1}{\sqrt[N]{\prod_t P(w_t | w_{t-n+1}, \dots, w_{t-1})}}$$

$$\begin{aligned} e^{\log P(w_1, \dots, w_N)^{-\frac{1}{N}}} &= e^{-\frac{1}{N} \log P(w_1, \dots, w_N)} = e^{-\frac{1}{N} \log \left(\prod_t P(w_t | w_{t-n+1}, \dots, w_{t-1}) \right)} \\ &= e^{-\frac{1}{N} \sum_t \log P(w_t | w_{t-n+1}, \dots, w_{t-1})} \end{aligned}$$

- какая для идеальной модели?
- какая для худшей модели?

Перплексия

<https://research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words/>

Model	Perplexity
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	67.6
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + BlackOut sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 (Jozefowicz et al., 2016)	43.7
2-layer LSTM-8192 (Jozefowicz et al., 2016)	30
Ours small (LSTM-2048)	43.9
Ours large (2-layer LSTM-2048)	39.8

<https://paperswithcode.com/task/language-modelling>

Проблемы статистических языковых моделей

• • • • •

Нейросетевые языковые модели

• • • • •

```
from transformers import AutoTokenizer, AutoModelForCausalLM

checkpoint = "ai-forever/rugpt3small_based_on_gpt2"

tokenizer = AutoTokenizer.from_pretrained(checkpoint)
model = AutoModelForCausalLM.from_pretrained(checkpoint)

tokenized_tensors_dict = tokenizer(
    text='самое красивое в людях – это',
    return_tensors='pt'
)

generated = model.generate(
    **tokenized_tensors_dict,
    max_length=300,
    temperature=1.0,
    do_sample=True
)

print(
    tokenizer.decode(generated.numpy()[0])
)
```

Neural LM

<https://huggingface.co/models>

```
from transformers import AutoTokenizer, AutoModelForCausalLM

checkpoint = "ai-forever/rugpt3small_based_on_gpt2"

tokenizer = AutoTokenizer.from_pretrained(checkpoint)
model = AutoModelForCausalLM.from_pretrained(checkpoint)

tokenized_tensors_dict = tokenizer(
    text='самое красивое в людях — это',
    return_tensors='pt'
)

generated = model.generate(
    **tokenized_tensors_dict,
    max_length=300,
    temperature=1.0,
    do_sample=True
)

print(
    tokenizer.decode(generated.numpy()[0])
)
```

Neural LM

самое красивое в людях — это их внутренний мир. Только его надо уметь слушать. Быть самим собой — это самое прекрасное и сложное в жизни. А еще — это время. Это время, в котором ты должен принимать решения, обдумывать и обдумывать свое поведение, чтобы жизнь не казалась такой пресной и бессмысленной, как раньше...

Как начать обучение

Мы начинаем с базовых принципов: любви, уважения, сострадания и милосердия. Однако, это не означает, что нужно говорить со всеми и каждому. Необходимо следовать этим принципам, не ожидая "конструктивной обратной связи". Вы должны быть готовы к тому, что все, что вы хотите получить — это как раз то, что вам нужно!

Самое главное — это не просто получить все, что вы хотите, но и понять, есть ли у вас какие-то ценности или принципы, которые вы хотите реализовать. Не говорите прямо: это не так важно, это все можно будет найти. Ищите ценности других людей, то, что кажется вам интересным. Вы увидите, что вы обязательно найдете что-то общее в своих отношениях и в отношениях с другими людьми. Что именно, вы обязательно можете найти.

Когда вы все узнаете, возможно, вы будете чувствовать себя так, будто все, о чем вы не хотите говорить, есть в вас самом и есть его, если вы захотите услышать от других.

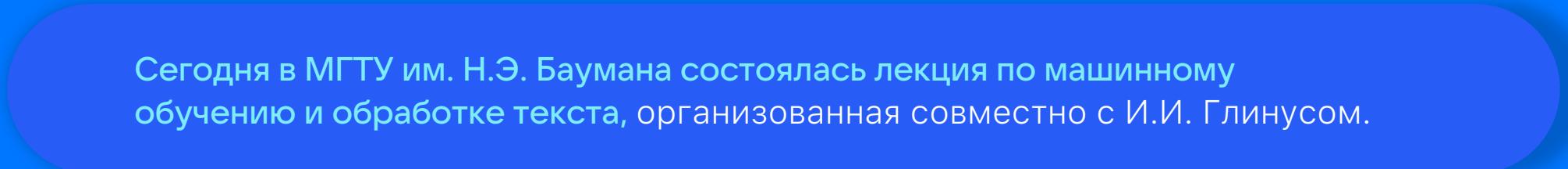
Когда вы научитесь слушать других людей и принимать решения "свой — чужой". Вы поймете

<https://huggingface.co/models>



Сегодня в МГТУ им. Н.Э. Баумана состоялась лекция по машинному обучению и обработке текста

rugpt3small



Сегодня в МГТУ им. Н.Э. Баумана состоялась лекция по машинному обучению и обработке текста, организованная совместно с И.И. Глинусом.

1. Конечная цель обучения - создать условия для развития навыков восприятия текста как средства выразительности речи на основе авторского и языкового программирования.
2. По завершении лекций была выдана брошюра "Творческий подход, обучение и эффективность" (М.: МГАУ, 2013). Данная книга призвана повысить мотивацию учащихся к овладению языком, познакомить их с различными аспектами творческой деятельности.
3. В этом году в МГТУ имени Н.Э. Баумана состоялась творческая встреча с Ю. Гребенщиковым, И. П. Ивановым, Н.Н. Кагановичем, И. А. Тихомировым, Г.Л. Марченко и другими. Для многих из них в рамках этого мероприятия стала возможной публикация ее в журнале "Русский язык и культура: XXI век".
4. В рамках "Открытых городских лекций" состоялась лекция "Космос, история, культура и традиции МГУ имени Н.Э. Баумана (2010 год). Тема лекции: "Влияние на восприятие текста: формирование умения выделять "животные" и находить их "находящие" слова и действия.