

ТОКЕНИЗАЦИЯ В ЯЗЫКОВЫХ МОДЕЛЯХ

Андреева Дарья
Data Scientist, X5 Tech



Ai
Run

ТОКЕНИЗАЦИЯ

сейчас мы делим текст на слова — как
быть с различными формами одного и
того же слова? и как обрабатывать
незнакомые слова?

деление по пробелам

мама мыла раму -> ['мама', 'мыла', 'раму']

деление по пробелам

мама мыла раму -> ['мама', 'мыла', 'раму']

минусы?

огромный словарь

на редкие слова плохие эмбеддинги

ЧТО ХОТИМ?

словарь ограниченного размера
содержит в себе токены, которые
достаточно часто встречаются

BPE – BYTE PAIR ENCODING

'nog', 'сказать'
└───┘
└─┘
"nog сказать"

BPE – BYTE PAIR ENCODING

“мама мыла раму,
маму мыла рама”

1. vocab = [“м”, “а”, “р”, “у”...]
2. vocab += [“ма”, “му”, ...]
3. vocab += [“мам”, “мыл”...]
+ рама
4. vocab.size == n: stop

BPPE — BYTE PAIR ENCODING

подсказать; "скаж", "ска"...
vocab = [..., "под", "но"...] "мама мыла раму,
маму мыла рама"
подсказать →
["под", "скаж"...]

1. vocab = ["м", "а", "р", "у"...]
2. vocab += ["ма", "му", ...]
3. vocab += ["мам", "мыл"...]
4. vocab.size == n: stop

при токенизации берем
самую большую подстроку

WORD PIECE –

WORD PIECE —
$$\text{score} = \frac{\text{freq_of_pair}[w_1, w_2]}{\text{freq}[w_1] * \text{freq}[w_2]}$$

“мама мыла раму, маму мыла рама”

1. vocab = [“м”, “а”, “р”, “у”...]
2. vocab += [“ма”, “му”, ...] пары с наивысшим скором
3. vocab.size == n: stop

WORD PIECE — $\text{score} = \frac{\text{freq_of_pair}[w_1, w_2]}{\text{freq}[w_1] * \text{freq}[w_2]}$

“мама мыла раму, маму мыла рама”

$$\begin{aligned} [\text{“рама”}, \text{“а”}] &= \frac{1}{2 \cdot 8} = \frac{1}{16} \\ [\text{“ра”}, \text{“ма”}] &= \frac{1}{2 \cdot 4} = \frac{1}{8} \end{aligned}$$

WORD PIECE —
$$\text{score} = \frac{\text{freq_of_pair}[w_1, w_2]}{\text{freq}[w_1] * \text{freq}[w_2]}$$

“мама мыла раму, маму мыла рама”

$$[\text{“рама”}, \text{“а”}] = 1/16$$

$$[\text{“ра”}, \text{“ма”}] = 1/8$$

хотим добавлять токены,
которые отдельно
встречаются редко, но
вместе часто

UNIGRAM –

["дам", "дум", "дах"]
- корпус

$$\begin{aligned} & \text{"a", "д", "м", "у", "да", "ау", "д\textsubscript{1}у", "у\textsubscript{1}а", "ах"} \\ & P(\text{"да", "м"}) = \frac{2}{S} \cdot \frac{2}{S} = \frac{4}{S^2} \\ & P(\text{"д", "а", "м"}) = \frac{3}{S} \cdot \frac{2}{S} \cdot \frac{2}{S} = \frac{12}{S^3} \\ & P(\text{"д", "а\textsubscript{1}м"}) = \frac{3}{S} \cdot \frac{1}{S} = \frac{3}{S^2} \end{aligned}$$

UNIGRAM —

словарь — ВРЕ с максимальным словарём

для каждого токена считаем, насколько
уменьшится вероятность при его удалении

ЯЗЫКОВЫЕ МОДЕЛИ

ЯЗЫКОВЫЕ МОДЕЛИ

$$P(w_1, \dots, w_T) = P(w_1) \prod_{i=2}^T P(w_i | w_1, \dots, w_{i-1})$$

n-gram language model

$$P(w_1, \dots, w_T) = P(w_1) \prod_{i=2}^T P(w_i | w_1, \dots, w_{i-1})$$

предположим, что вероятность слова
не очень зависит от слов, которые были
далеко в начале

$$P(w_i | w_{i-1}, \dots, w_1) \sim P(w_i | w_{i-1}, \dots, w_{i-n})$$

n-gram language model

$$P(w_i | w_{i-1}, \dots, w_{i-n}) = \frac{P(w_{i-n}, \dots, w_i)}{P(w_{i-n}, \dots, w_{i-1})}$$

n-gram language model

$$P(w_i | w_{i-1}, \dots, w_{i-n}) = \frac{P(w_{i-n}, \dots, w_i)}{P(w_{i-n}, \dots, w_{i-1})}$$

$\delta = 1, 3, 5, 7$ et

$w_i =$ ogra სება

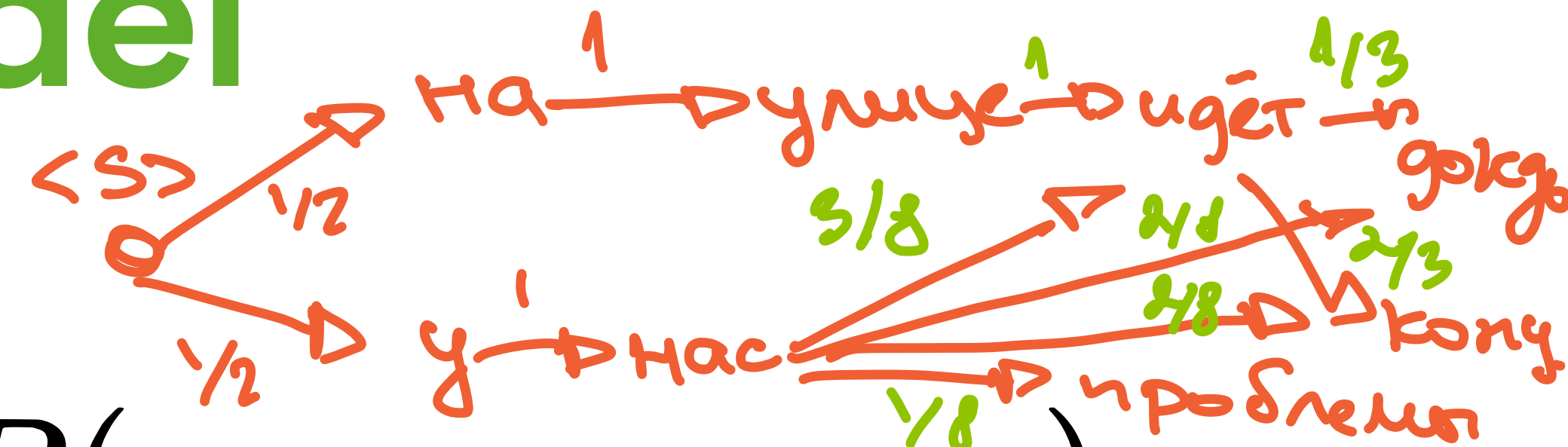
$\#$ მამა

$\#$ მამ

$\delta =$ count(w_{i-n}, \dots, w_i)

$2|V| \cdot$ count(w_{i-n}, \dots, w_{i-1})

n-gram language model



$$P(w_i | w_{i-1}, \dots, w_{i-n}) = \frac{P(w_{i-n}, \dots, w_i)}{P(w_{i-n}, \dots, w_{i-1})}$$

$$= \frac{\text{count}(w_{i-n}, \dots, w_i)}{\text{count}(w_{i-n}, \dots, w_{i-1})}$$

n-gram language model

$$\frac{\text{count}(w_{i-n}, \dots, w_i)}{\text{count}(w_{i-n}, \dots, w_{i-1})}$$

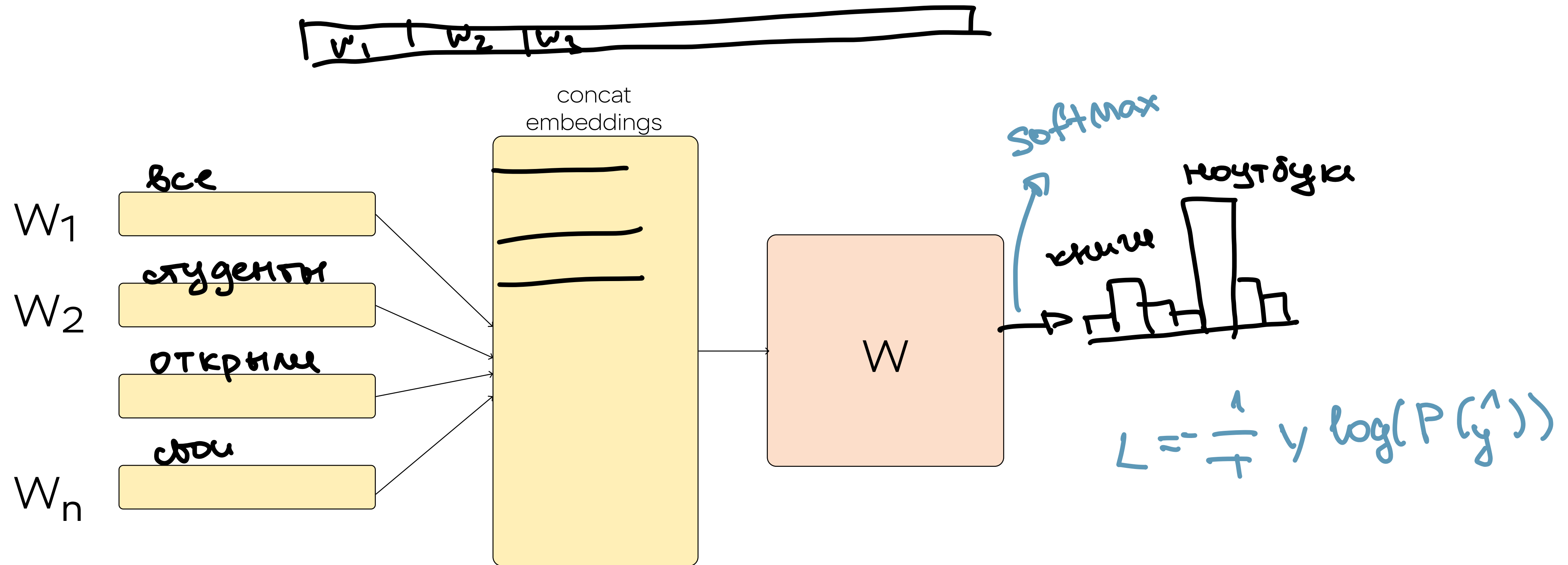
МИНУСЫ?

$$\text{биграммы} = |V|^2$$

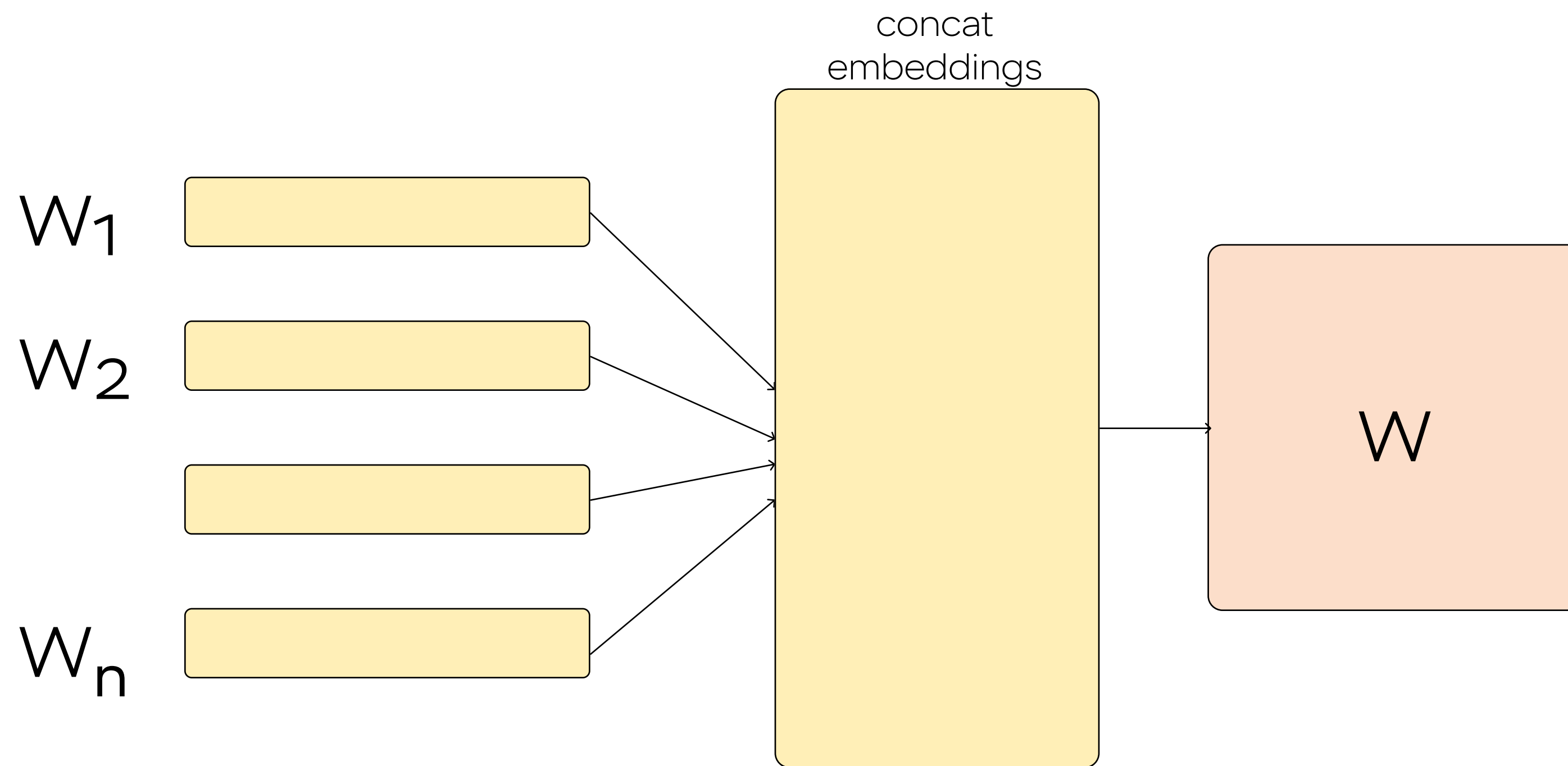
тяжело считать и хранить при больших n

при больших n будут низкие вероятности и
большая дисперсия

нейросетевые модели



нейросетевые модели



ПЛЮСЫ:

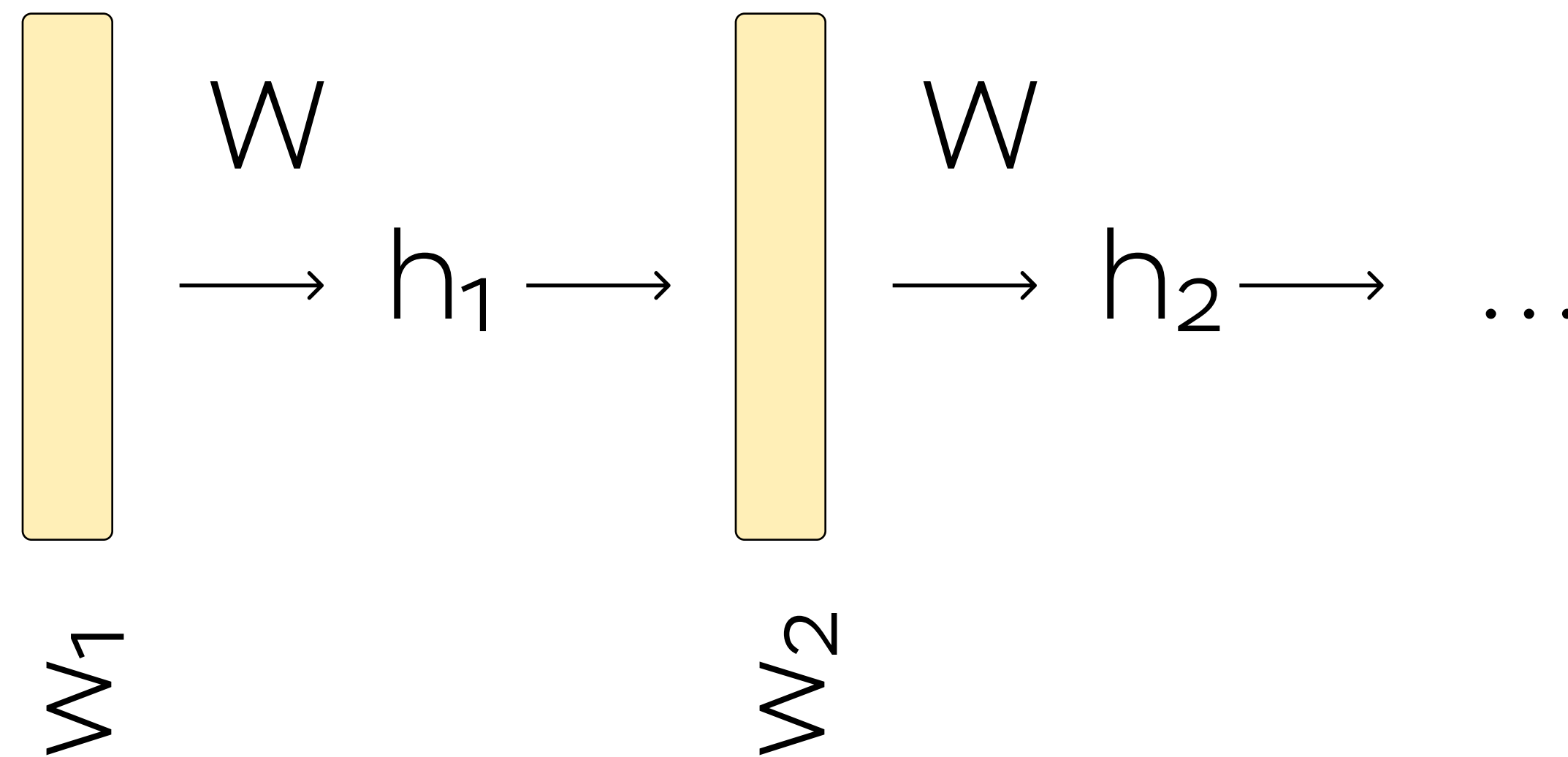
не нужно хранить n-граммы

МИНУСЫ:

увеличение окна (n)
увеличивает w

рекуррентная языковая модель

$$\frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial h_n} \cdot \frac{\partial h_n}{\partial h_{n-1}} \dots \frac{\partial h_2}{\partial h_1}$$



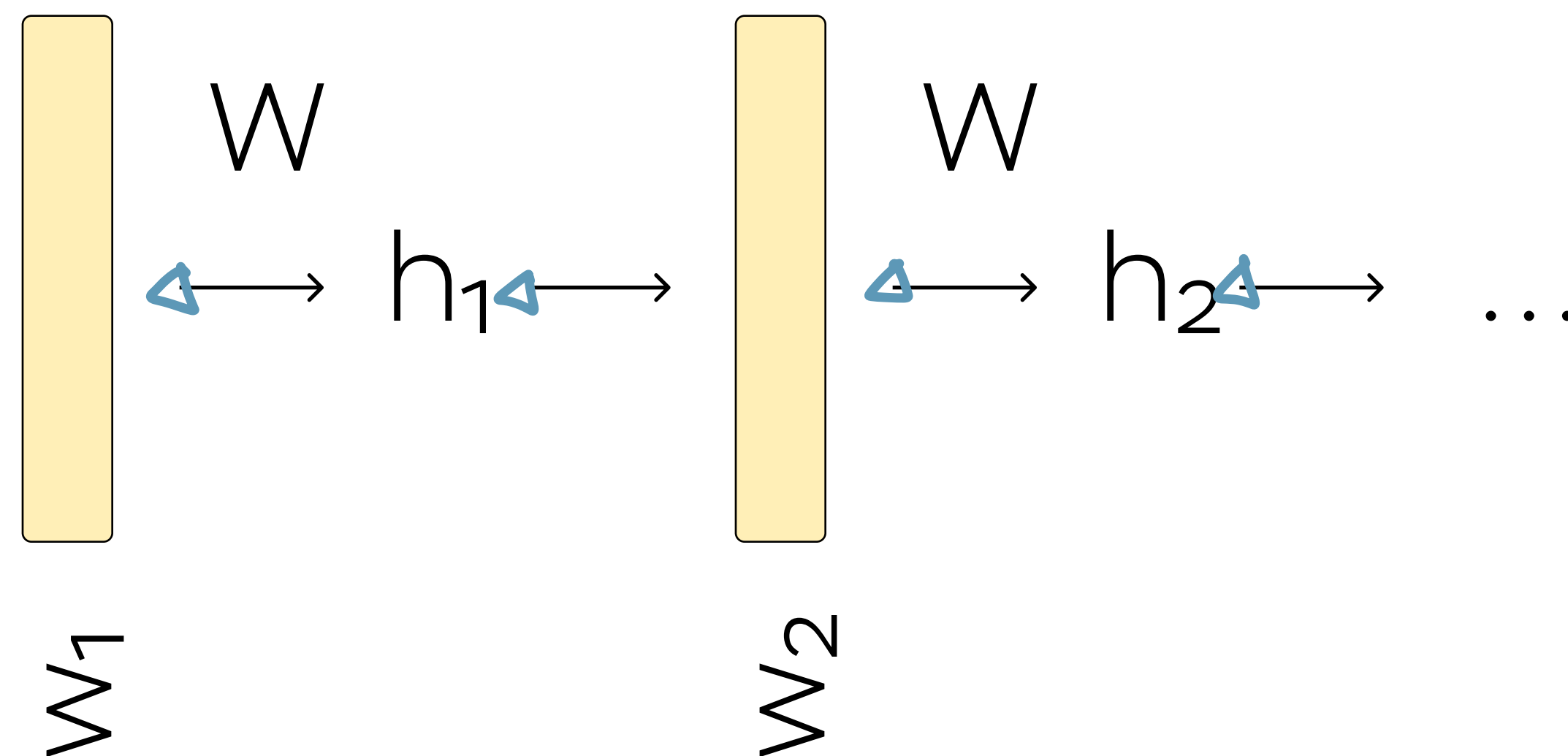
ПЛЮСЫ:

не нужно хранить n-граммы
учитываем порядок

МИНУСЫ:

постепенно забываем, что
было в начале
нельзя параллелить
затухание градиента

рекуррентная языковая модель



плюсы:

не нужно хранить n-граммы
учитываем порядок

минусы:

постепенно забываем, что
было в начале
нельзя параллелить
затухание градиента

**КАК
СЭМПЛИРОВАТЬ?**

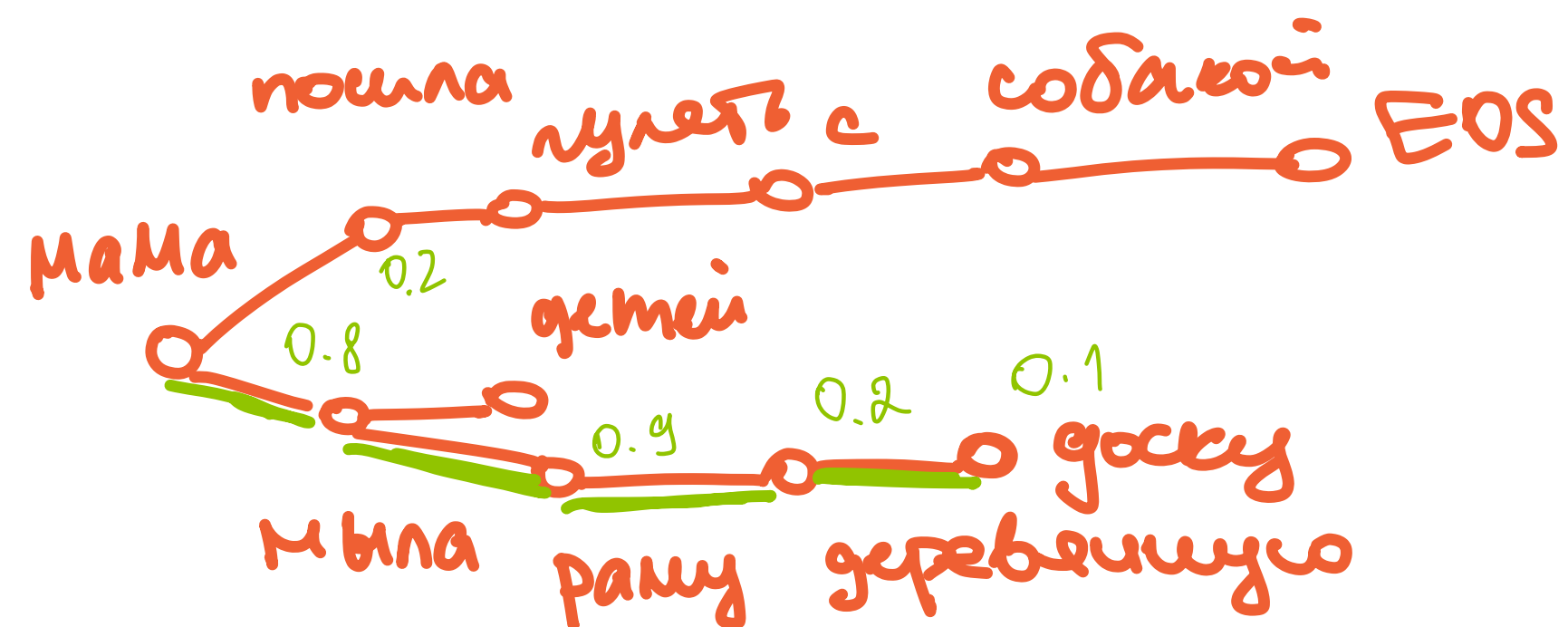
как сэмплировать?

хотим сгенерировать текст, похожий на человека

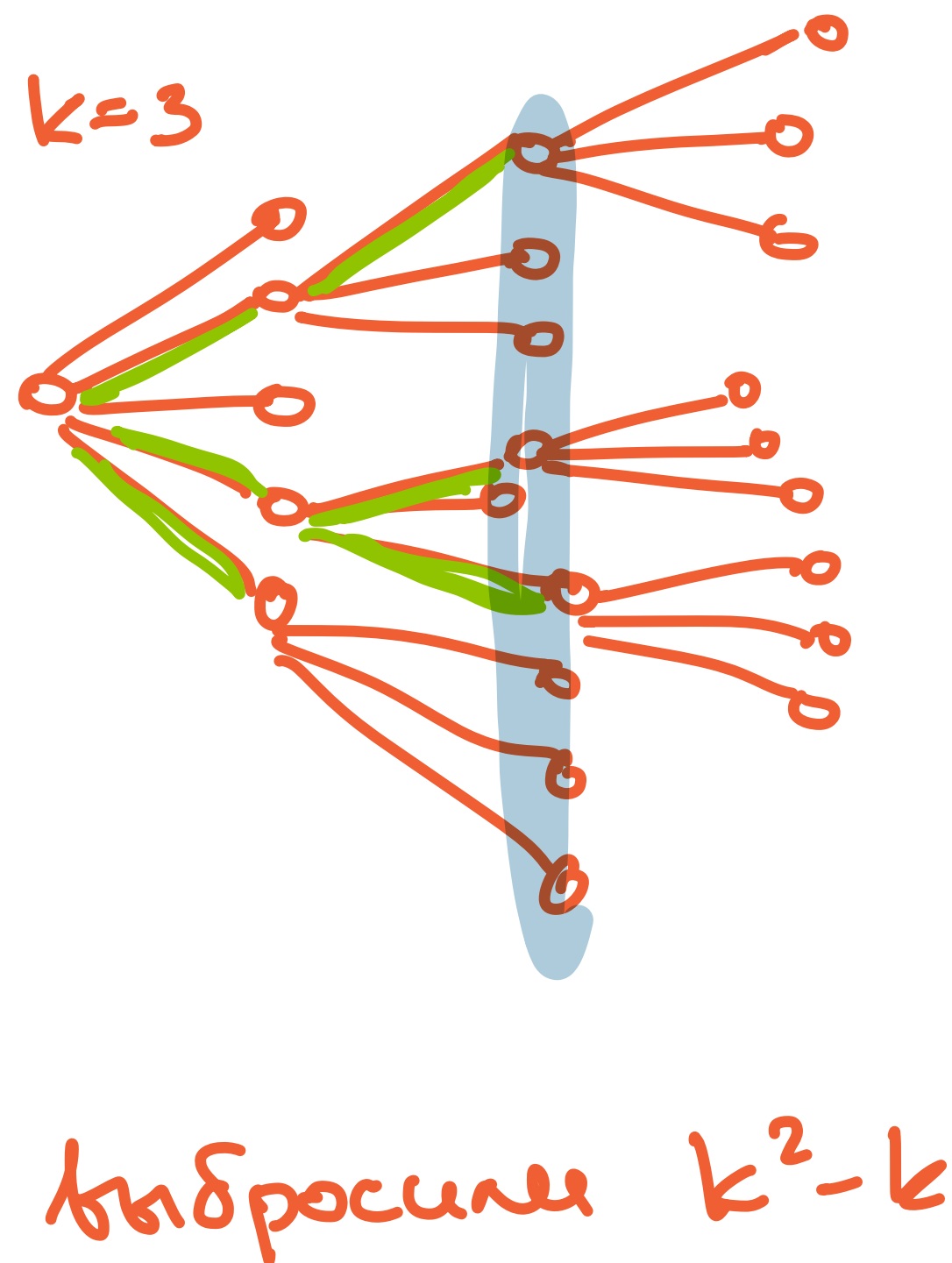
1. текст осознанный
2. текст разнообразный

жадный метод [greedy]

берём токены с наибольшей вероятностью



beam search



выбираем k наиболее вероятных токенов

для каждого из них выбираем свои k наиболее вероятных токенов

чтобы не порваться от экспоненциальности, отбрасываем $k^2 - k$ вариантов, на каждом шаге оставляем только топ- k путей

greedy vs beam search

greedy

детерминированный

быстрый

можем выбрать лучшее на
каждом шаге

не гарантирует разнообразия
реплик

beam search

детерминированный

$k = 1$ -- жадный

вычислительно сложный

можно генерировать более
разнообразные реплики

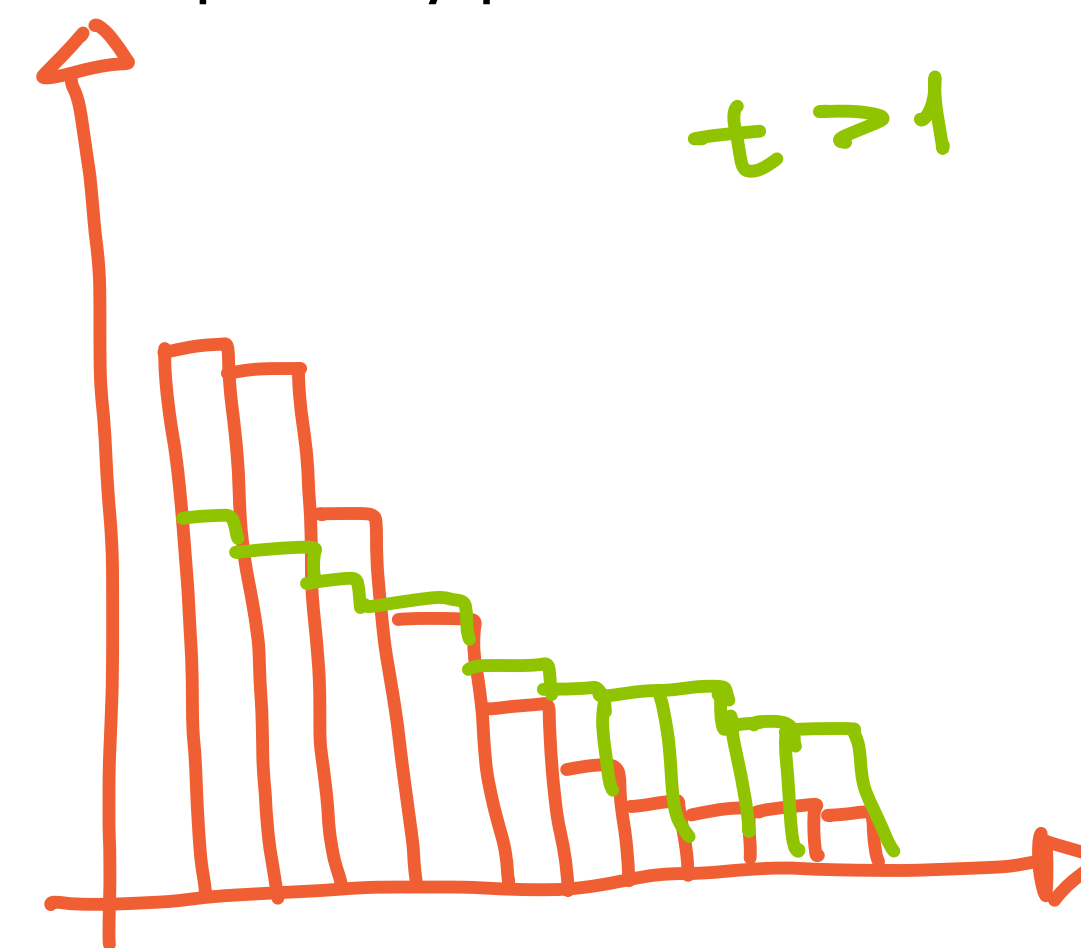
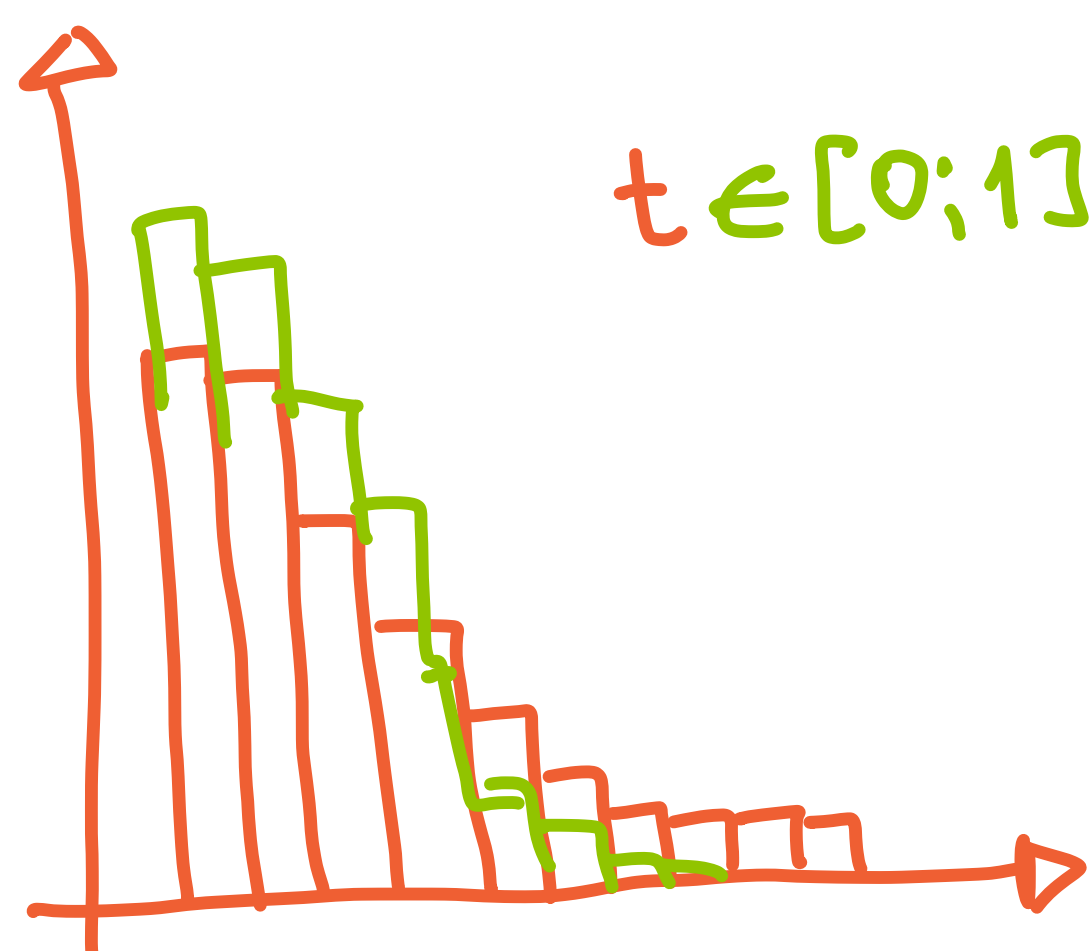
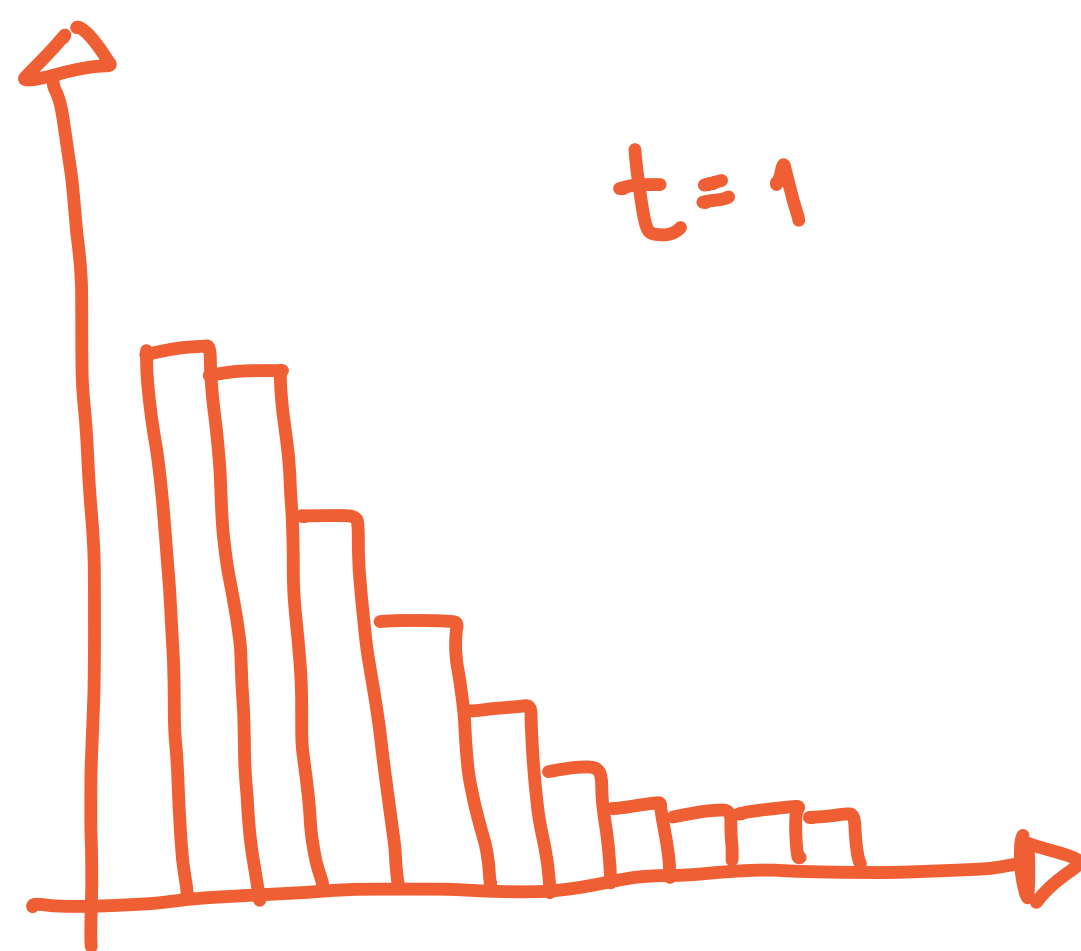
реплики очень общие

temperature samplings

$$p(X_i) = \frac{e^{x_i/\tau}}{\sum_{j=1}^{|V|} e^{x_j/\tau}}$$

перевзвешиваем веса

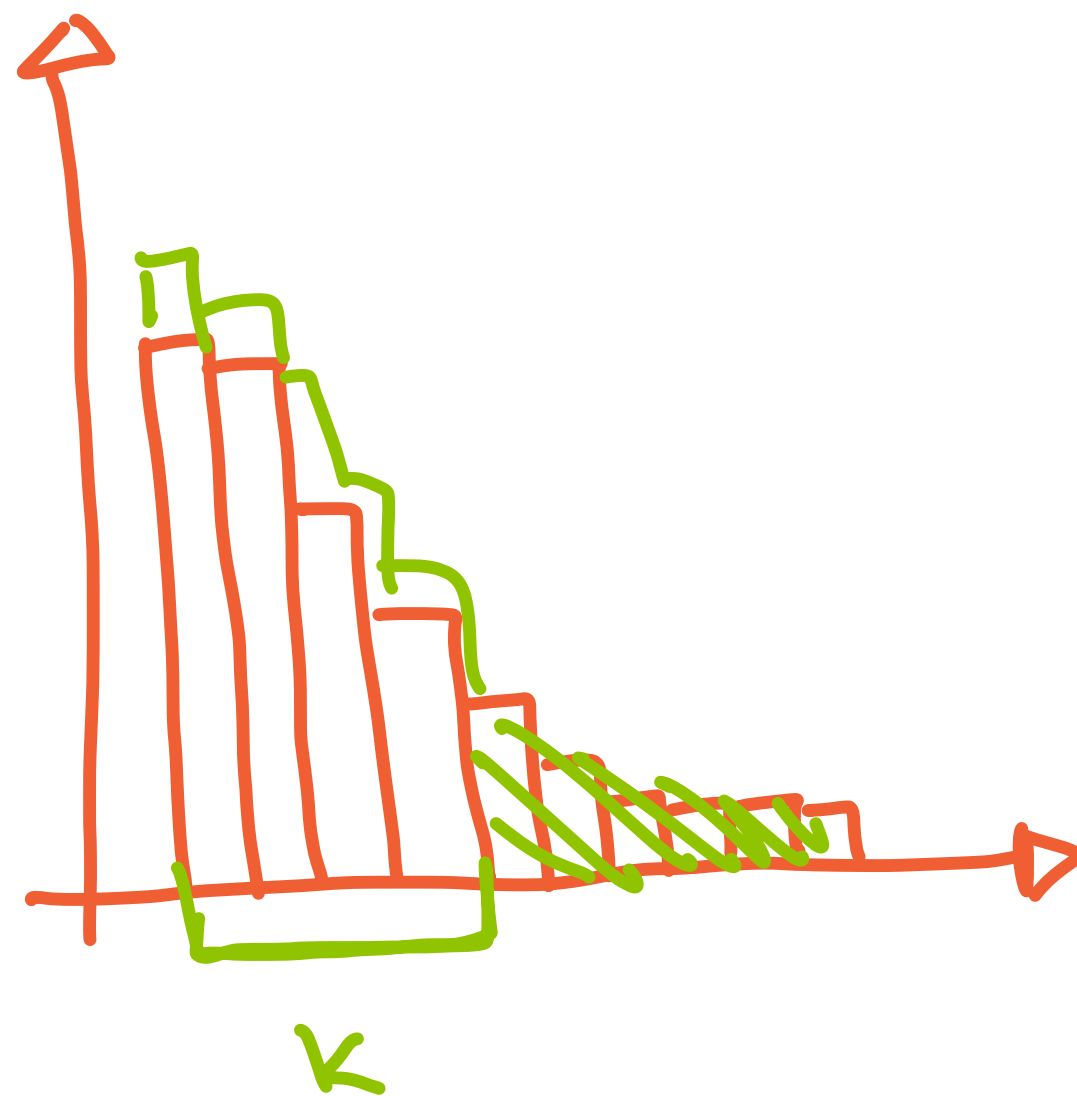
параметр τ назовём температурой



$\langle s \rangle$
[UNK]

top-k sampling

$$p' = \sum_{x \in \text{Vocab}} p(x_i | x_{1:i-1})$$



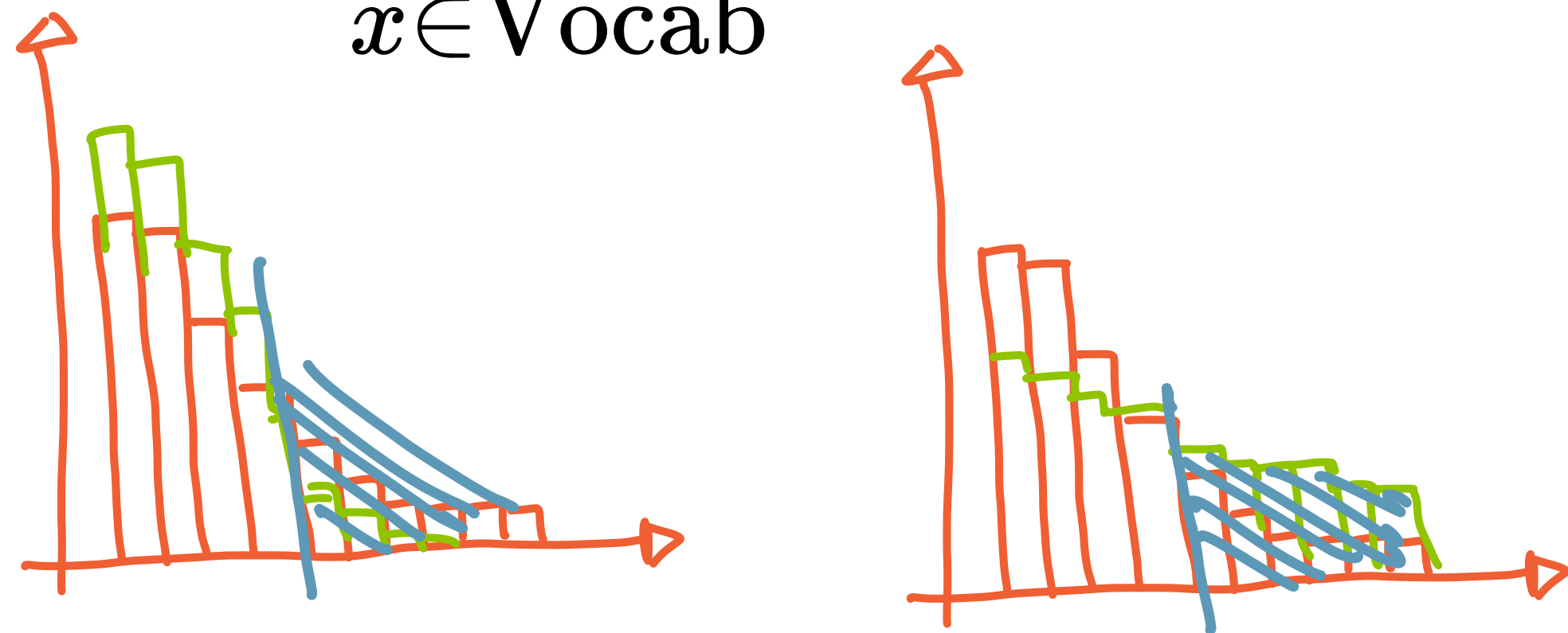
на каждом шаге отбираем k
самых вероятных слов

введём нормировочный
коэффициент

шкалируем выходы софтмакса

top-k sampling

$$p' = \sum_{x \in \text{Vocab}} p(x_i | x_{1:i-1})$$



на каждом шаге отбираем k
самых вероятных слов

введём нормировочный
коэффициент

шкалируем выходы софтмакса

из-за разности распределений хотим иметь разный k на каждом шаге

nucleus [top-p] sampling [2020]

$$\sum_{x \in \text{Vocab}} p(x_i | x_{1:i-1}) \geq p$$

берем минимальное число
токенов, сумма вероятностей
которых $\geq p$

