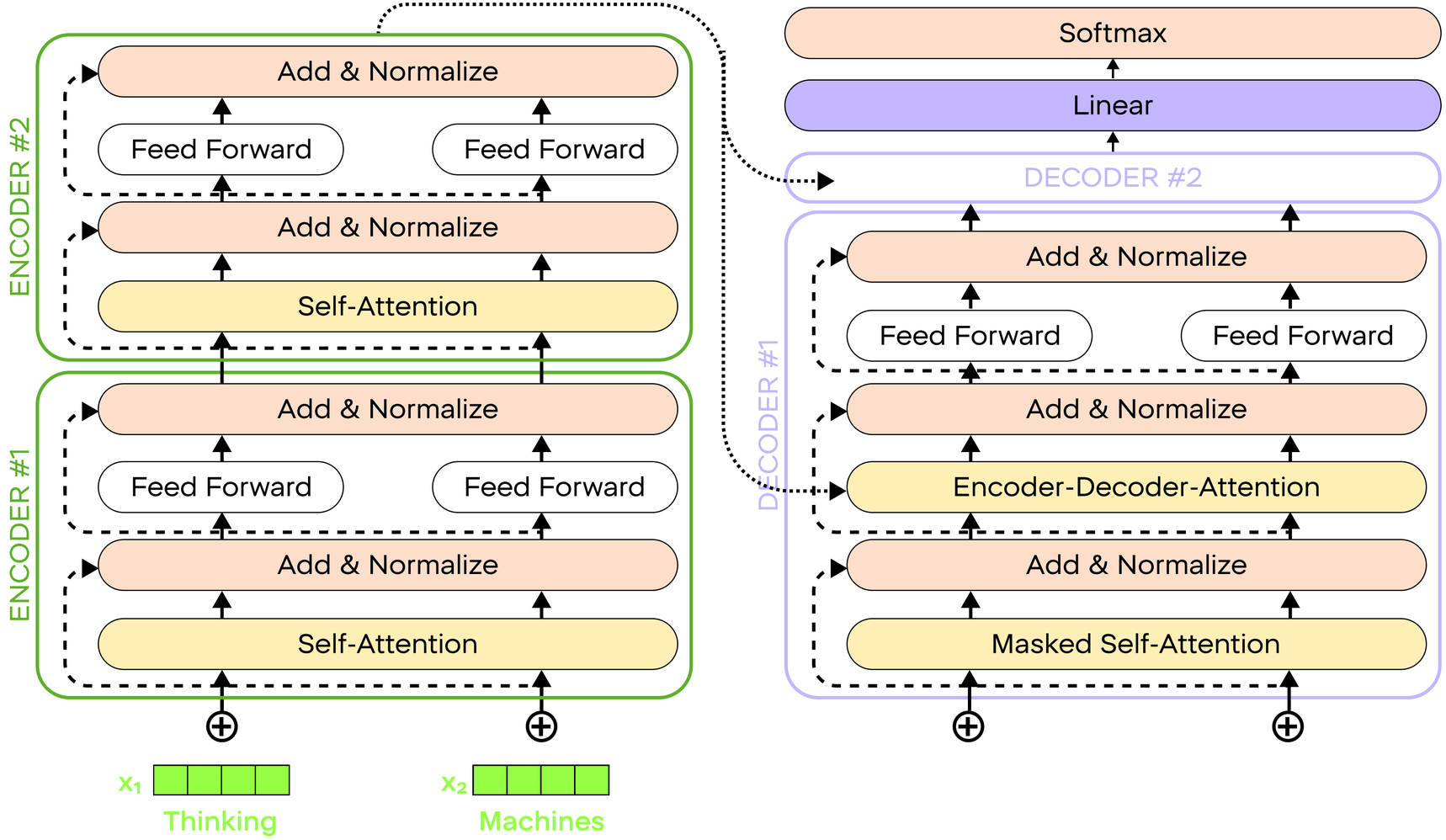
X5 Tech

GPT, LLAMA & THE OTHERS

Андреева Дарья
Data Scientist, X5 Tech

attention is...



X5 Tech

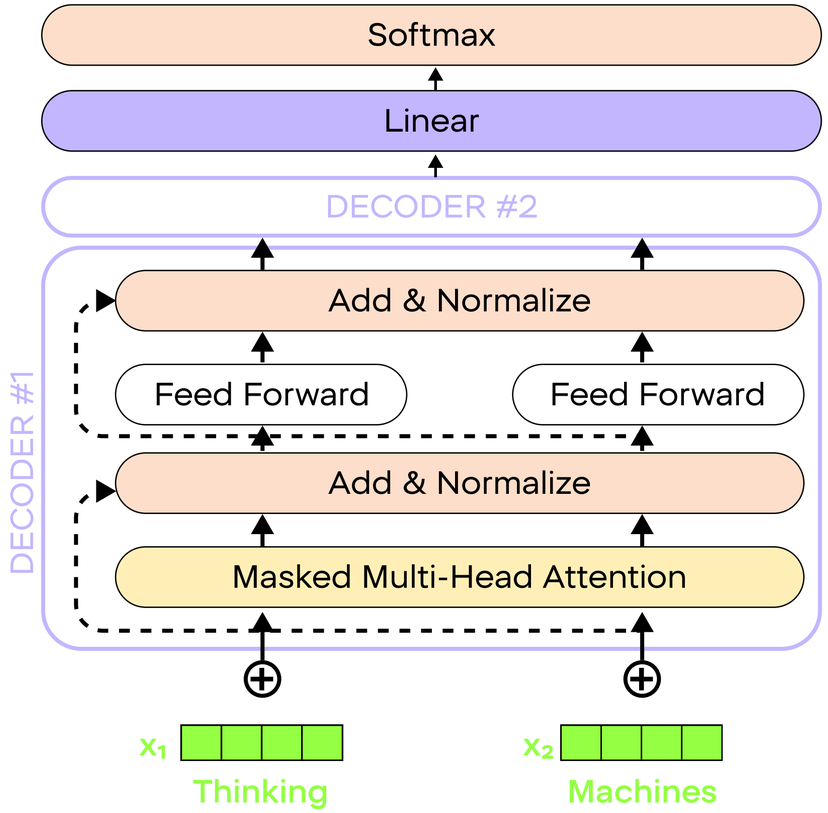
GPT

GENERATIVE
PRE-TRAINED
TRANSFORMER

X5 Tech

Improving Language Understanding by Generative Pre-Training:
https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

gpt



Ключевые отличия:

- Одно направление (использует контекст только слева)
- Состоит только из декодеров

X5 Tech

gpt

	Параметры	Данные
GPT	117M	5гб
GPT-2	1.5B	45гб
GPT-3	175B	45тб

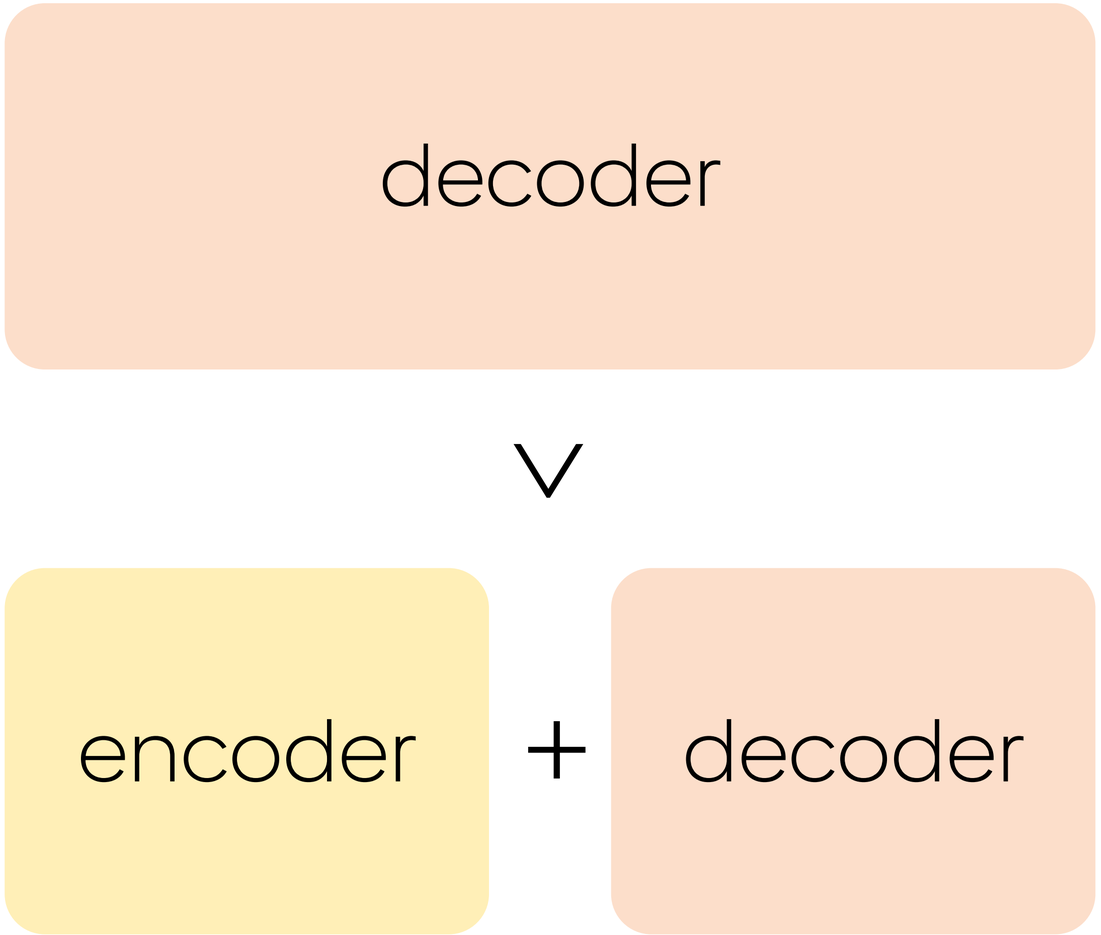
Идеи:

- Учимся предсказанию следующего токена
- Видим и запоминаем очень много текста
- Практически любую задачу NLP можно свести к генерации текста

gpt

предсказываем следующий токен:

где трансформеры?



X5 Tech

HOW TO:
DO FASTER
DO BETTER

X5 Tech

ускорение

существуют **три** вида памяти:

- gpu sram (static random-access memory) – самая быстрая, но небольшая память, вшита в процессор (19 tb/s, 20 mb)
- gpu hbm (high bandwidth memory) – основная память gpu (1.5 tb/s, 40 gb)
- cpu dram (dynamic random-access memory) – оперативная память cpu, самая медленная (12.8 gb/s, >1 tb)

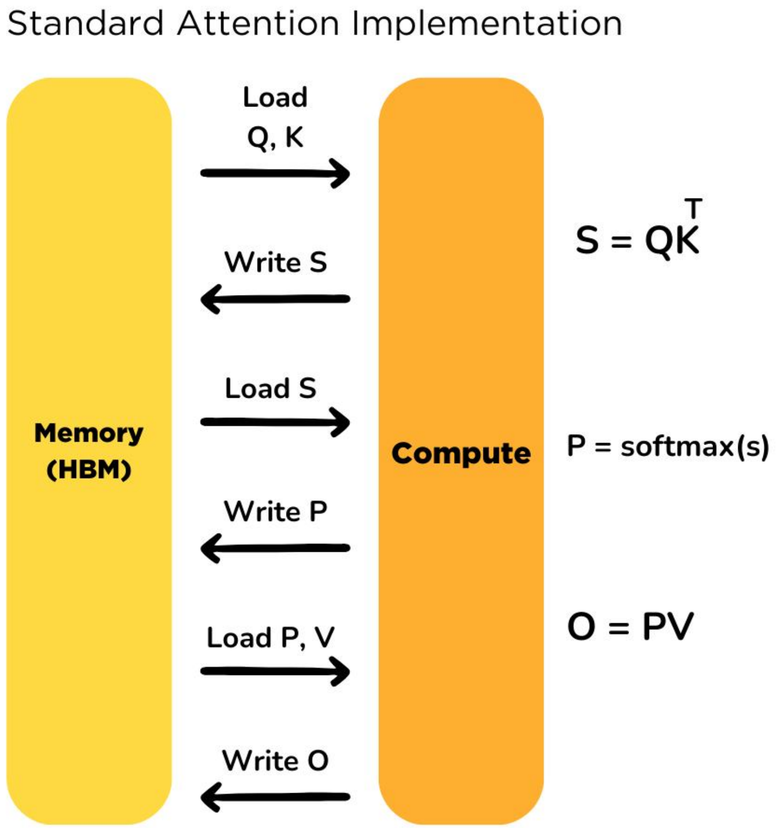
Algorithm 0 Standard Attention Implementation

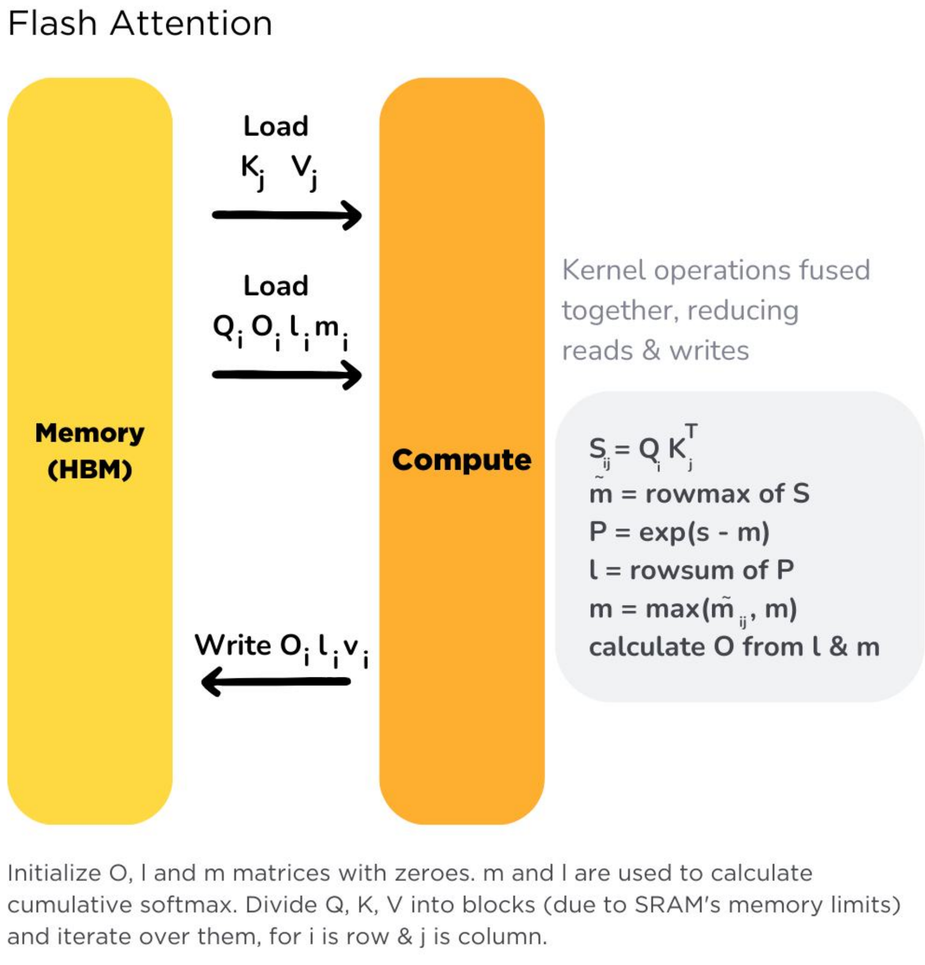
Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ in HBM.

1: Load $\mathbf{Q}, \mathbf{K}$ by blocks from HBM, compute $\mathbf{S} = \mathbf{Q}\mathbf{K}^\top$, write $\mathbf{S}$ to HBM.
2: Read $\mathbf{S}$ from HBM, compute $\mathbf{P} = \mathrm{softmax}(\mathbf{S})$, write $\mathbf{P}$ to HBM.
3: Load $\mathbf{P}$ and $\mathbf{V}$ by blocks from HBM, compute $\mathbf{O} = \mathbf{P}\mathbf{V}$, write $\mathbf{O}$ to HBM.
4: Return $\mathbf{O}$.

flash attention

- attention считается по блокам, которые влезают в sram
- o – матрица выходов, l – нормировочная константа softmax, m – максимальное значение скора внимания





X5 Tech

flash attention

- attention считается по блокам, которые влезают в sram
- o – матрица выходов, l – нормировочная константа softmax, m – максимальное значение скора внимания

Attention	Standard	Flash
GFLOPs	66.6	75.2
HBM R/W (GB)	40.3	4.4
Runtime (ms)	41.7	7.3

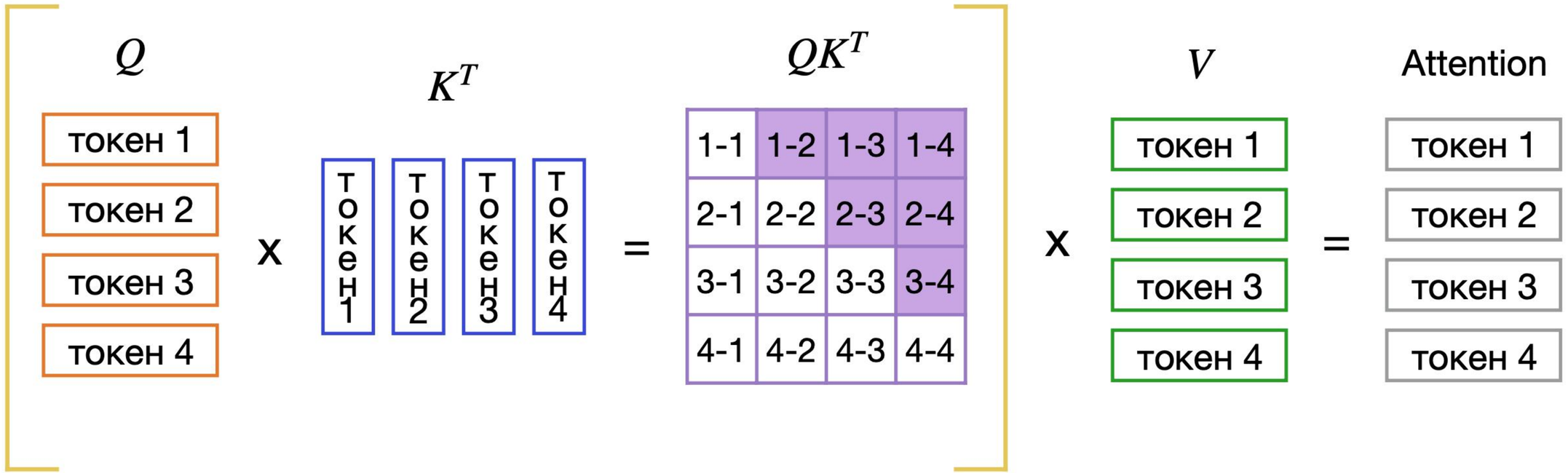
ГЕНЕРАЦИЯ ТЕКСТА

$$Attention(Q, V, K) = Softmax(\frac{QK^T}{\sqrt{d}})V$$

1. Один подсчет внимания занимает $O(l^2hd + lh^2d^2)$ операций
2. Пропускаем всю последовательность через модель на каждой итерации

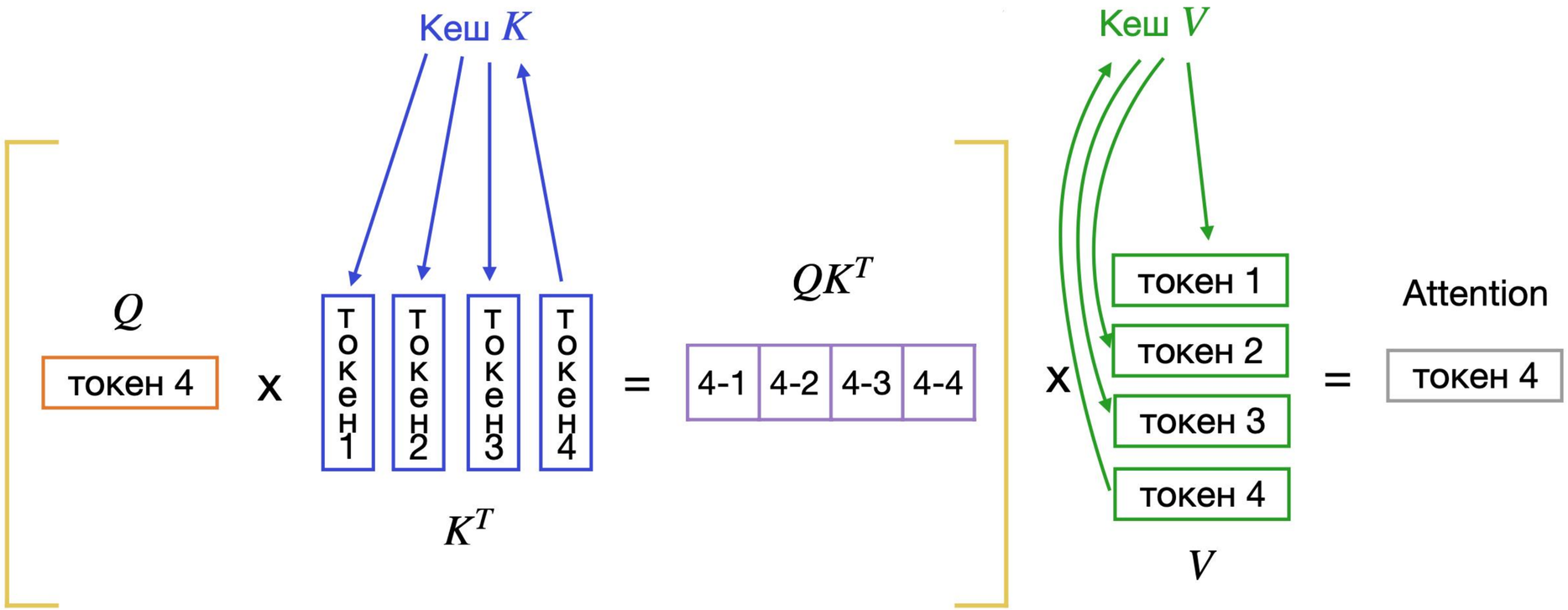
X5 Tech

генерация текста



X5 Tech

kv-кэширование

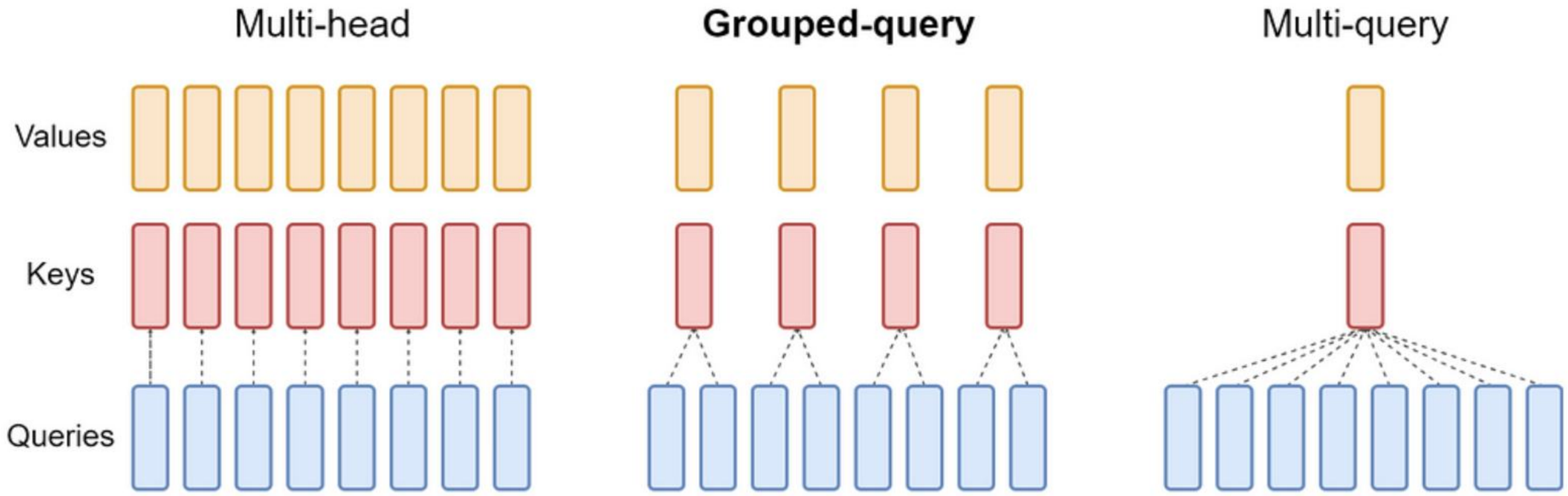


$$O(l^2hd + lh^2d^2) \longrightarrow O(lhd + h^2d^2)$$

X5 Tech

multi-query attention

- храним только одну или несколько матриц k и v (на этапе инференса)
- на декодерах ускоряемся почти в 10 раз



X5 Tech

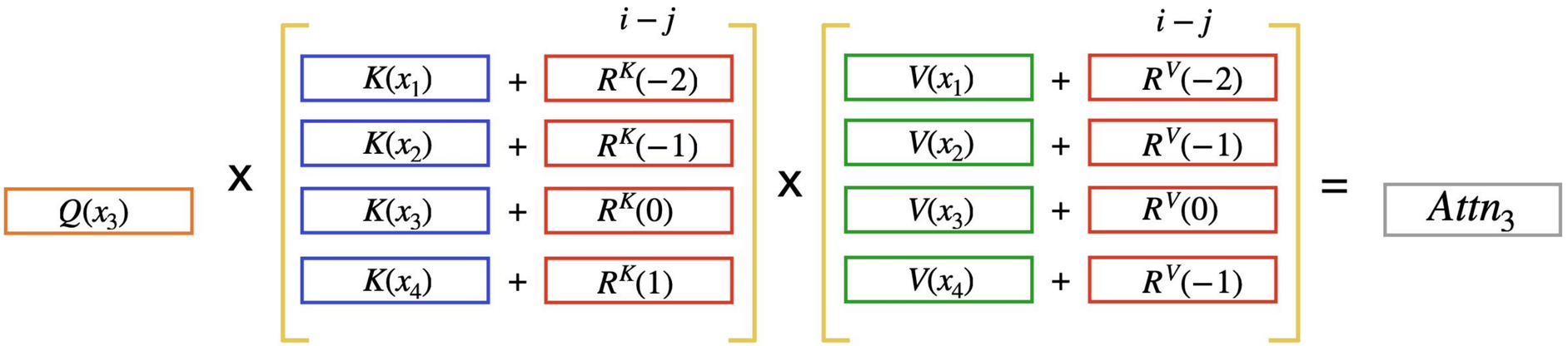
ПОЗИЦИОННЫЕ ЭМБЕДДИНГИ

Сейчас:

1. Не учитываем относительные позиции
2. Усложняем работу с длинными последовательностями

relative position encodings (rpe)

$$Attn_i = softmax\left(\frac{Q_i^T(K^T + R_i^K)}{\sqrt{d}}\right)(V + R_i^V)$$



X5 Tech

relative position encodings (rpe)

- для генерализации можем договориться о максимальном расстоянии, которое используем
- новая проблема: храним больше матриц

$$Attn_i = softmax\left(\frac{Q_i^T(K^T + R_i^K)}{\sqrt{d}}\right)(V + R_i^V)$$

$$P = \begin{pmatrix} 0 & 1 & 2 & 2 & 2 \\ -1 & 0 & 1 & 2 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -2 & -1 & 0 & 1 \\ -2 & -2 & -2 & -1 & 0 \end{pmatrix}$$

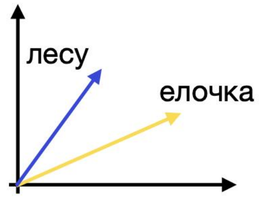
$$R^K = Emb_K(P) \in \mathbb{R}^{[n \times n \times d]}$$

$$R^V = Emb_V(P) \in \mathbb{R}^{[n \times n \times d]}$$

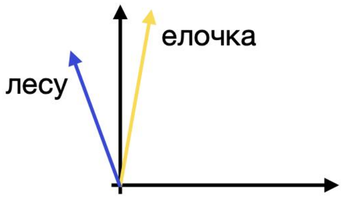
rotary position embeddings (rope)

- векторы q и k поворачиваются на угол i*theta, где i – позиция в тексте
- таким образом, относительное расстояние не меняется при изменении позиции
- векторы для похожих позиций будут поворачиваться на похожий угол, далекие векторы – на разные углы

в **лесу** родилась **елочка**



в нашем зимнем **лесу** родилась **елочка**



X5 Tech

rotary position embeddings (rope)

$$R_j^d = \begin{pmatrix} \cos j\theta_1 & -\sin j\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin j\theta_1 & \cos j\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos j\theta_2 & -\sin j\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin j\theta_2 & \cos j\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos j\theta_{d/2} & -\sin j\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin j\theta_{d/2} & \cos j\theta_{d/2} \end{pmatrix},$$

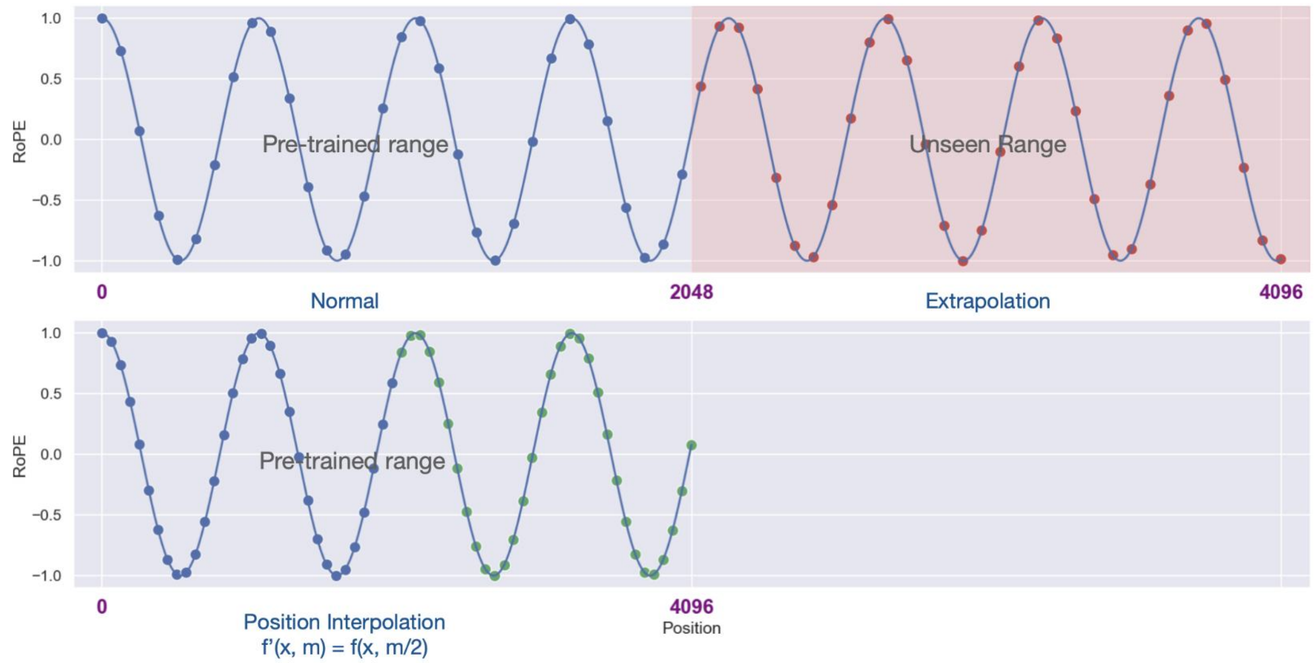
$$Q_i^{rT} K_j^r = (R_i^d Q_i)^T (R_j^d K_j) = Q_i^T R_i^{dT} R_j^d K_j = Q_i^T R_{j-i}^d K_j$$

X5 Tech

rotary position embeddings (rope)

выводы:

- не добавляем обучаемых параметров
- можем эффективно посчитать
- учитываем относительное расположение токенов
- плохо генерализуемся при увеличении длины контекста (на инференсе)



X5 Tech

attention with linear biases (alibi)

идея:

- добавляем отрицательный сдвиг к каждому значению внимания
- чем больше разница между позициями, тем меньше внимания остается

$$Attention(Q, V, K) = Softmax(\frac{QK^T + M}{\sqrt{d}})V$$

обычный attention

$$M_{ij} = \begin{cases} 0, & i \le j \\ -\infty, & i > j \end{cases}$$

alibi

$$M_{ij}^h = \begin{cases} m_h(i-j), & i \le j \\ -\infty, & i > j \end{cases}$$

для каждой головы свой $m_h = 2^{-h/2}$

X5 Tech

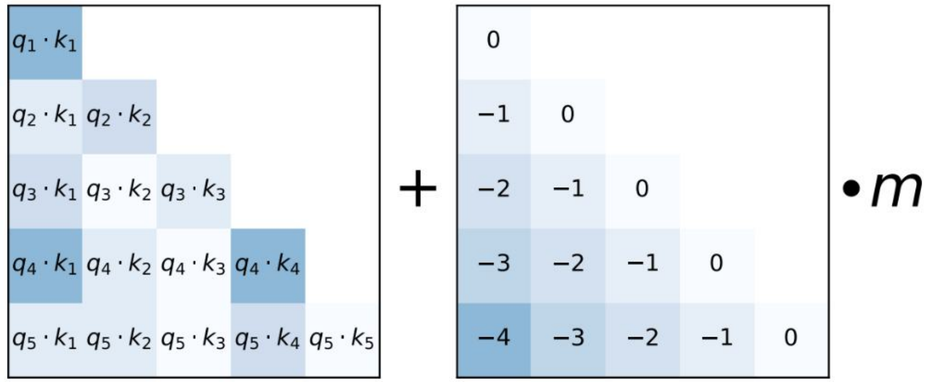
attention with linear biases (alibi)

- различные m позволяют смотреть как на локальную, так и на глобальную информацию
- не добавляем новых операций и ничего не обучаем

обычный attention **alibi**

$$M_{ij} = \begin{cases} 0, & i \le j \\ -\infty, & i > j \end{cases} \qquad M^h_{ij} = \begin{cases} m_h(i-j), & i \le j \\ -\infty, & i > j \end{cases}$$

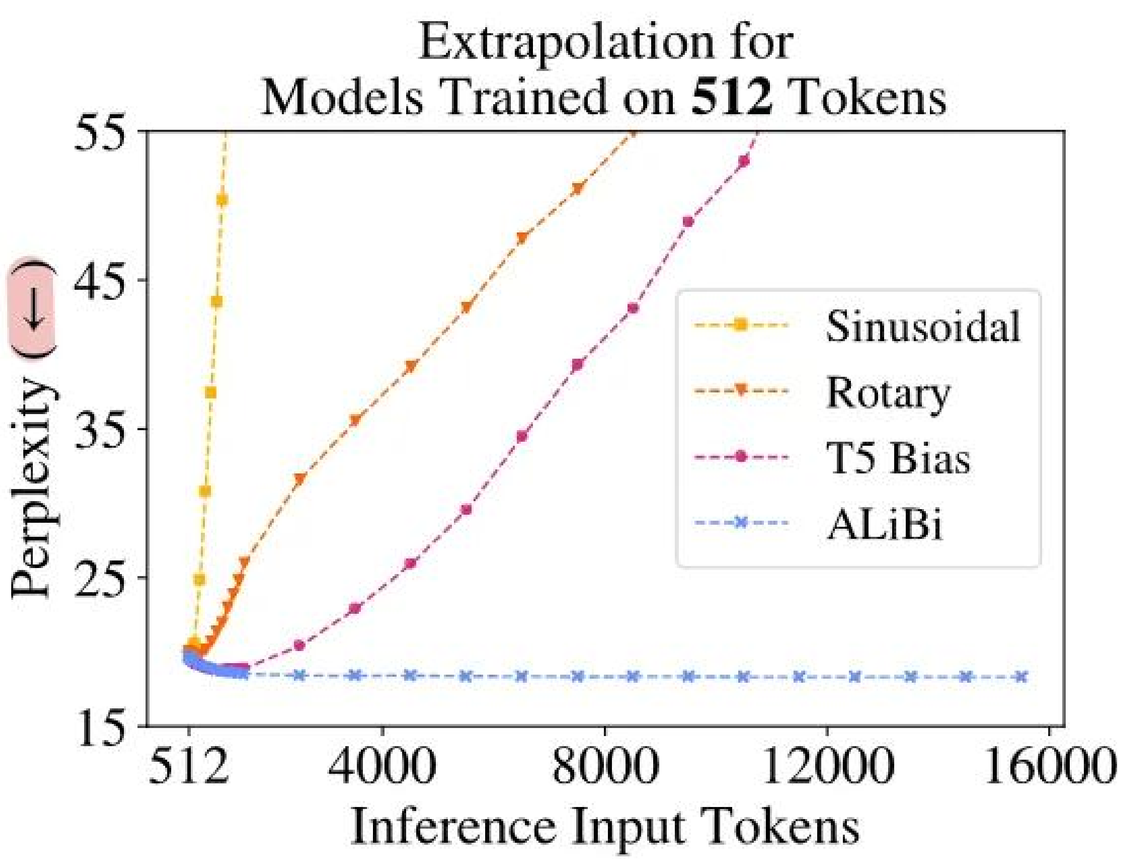
для каждой головы свой $m_h = 2^{-h/2}$

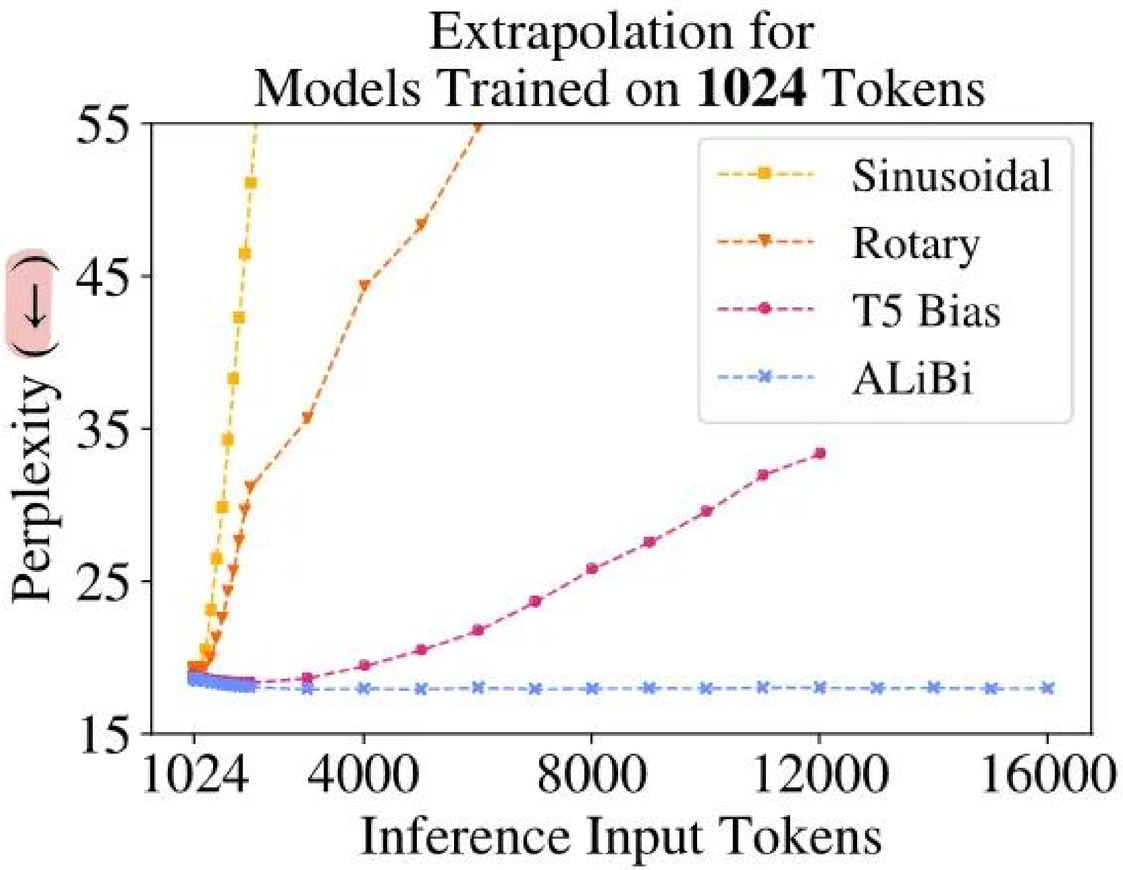


X5 Tech

attention with linear biases (alibi)

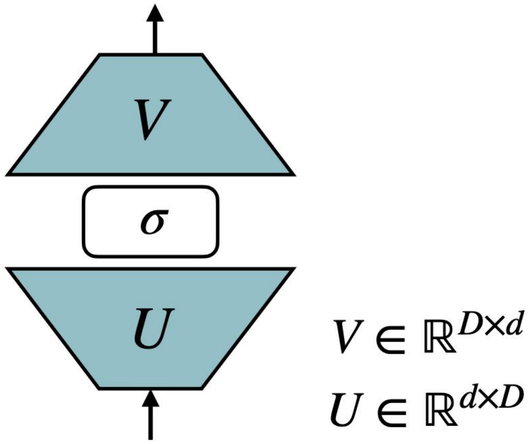
- хорошо приспосабливаемся к изменению длины текста





X5 Tech

FEED FORWARD NETWORK



что это было:

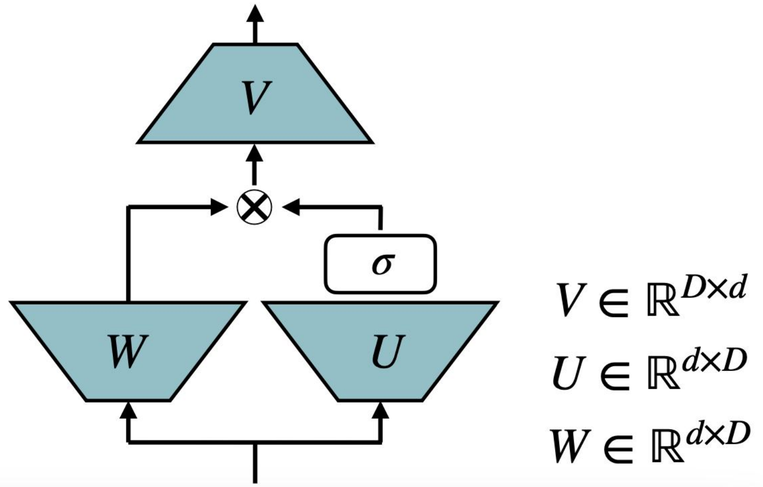
для каждого токена по отдельности обрабатывает информацию

замедляет обработку

X5 Tech

gated linear unit (glu)

- glu добавляет дополнительный линейный слой w, выходы которого умножаются на выходы u после активации
- играет роль фильтра, отсеивающего ненужные компоненты



$$V \in \mathbb{R}^{D \times d}$$
$$U \in \mathbb{R}^{d \times D}$$
$$W \in \mathbb{R}^{d \times D}$$

$$FFN_{GLU}(x) = \left(\sigma(xU + b_u) \otimes (xW + b_w)\right)V + b_v$$

X5 Tech

gated linear unit (glu)

$$FFN_{GLU}(x) = \left(\sigma(xU + b_u) \otimes (xW + b_w)\right)V + b_v$$

	Score Average	CoLA MCC	SST-2 Acc	MRPC F1	MRPC Acc	STSB PCC	STSB SCC	QQP F1	QQP Acc	MNLIm Acc	MNLImm Acc	QNLI Acc	RTE Acc
FFN_{ReLU}	83.80	51.32	94.04	**93.08**	**90.20**	89.64	89.42	89.01	91.75	85.83	86.42	92.81	80.14
FFN_{GELU}	83.86	53.48	94.04	92.81	**90.20**	89.69	89.49	88.63	91.62	85.89	86.13	92.39	80.51
FFN_{Swish}	83.60	49.79	93.69	92.31	89.46	89.20	88.98	88.84	91.67	85.22	85.02	92.33	81.23
FFN_{GLU}	84.20	49.16	94.27	92.39	89.46	89.46	89.35	88.79	91.62	86.36	86.18	92.92	**84.12**
FFN_{GEGLU}	84.12	53.65	93.92	92.68	89.71	90.26	90.13	89.11	91.85	86.15	86.17	92.81	79.42
$FFN_{Bilinear}$	83.79	51.02	**94.38**	92.28	89.46	90.06	89.84	88.95	91.69	**86.90**	**87.08**	92.92	81.95
FFN_{SwiGLU}	84.36	51.59	93.92	92.23	88.97	**90.32**	**90.13**	**89.14**	**91.87**	86.45	86.47	**92.93**	83.39
FFN_{ReGLU}	**84.67**	**56.16**	**94.38**	92.06	89.22	89.97	89.85	88.86	91.72	86.20	86.40	92.68	81.59