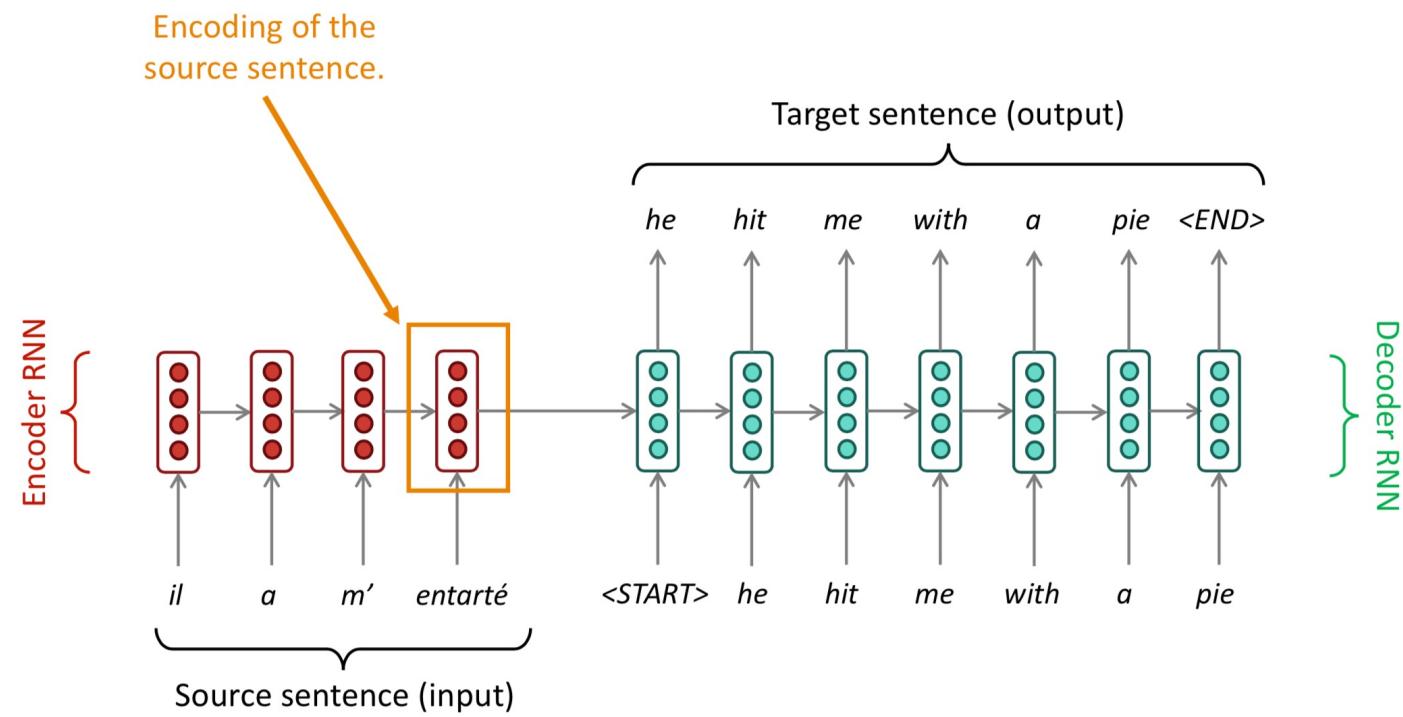


Обработка естественного языка. Трансформеры.

МФТИ, Сбера

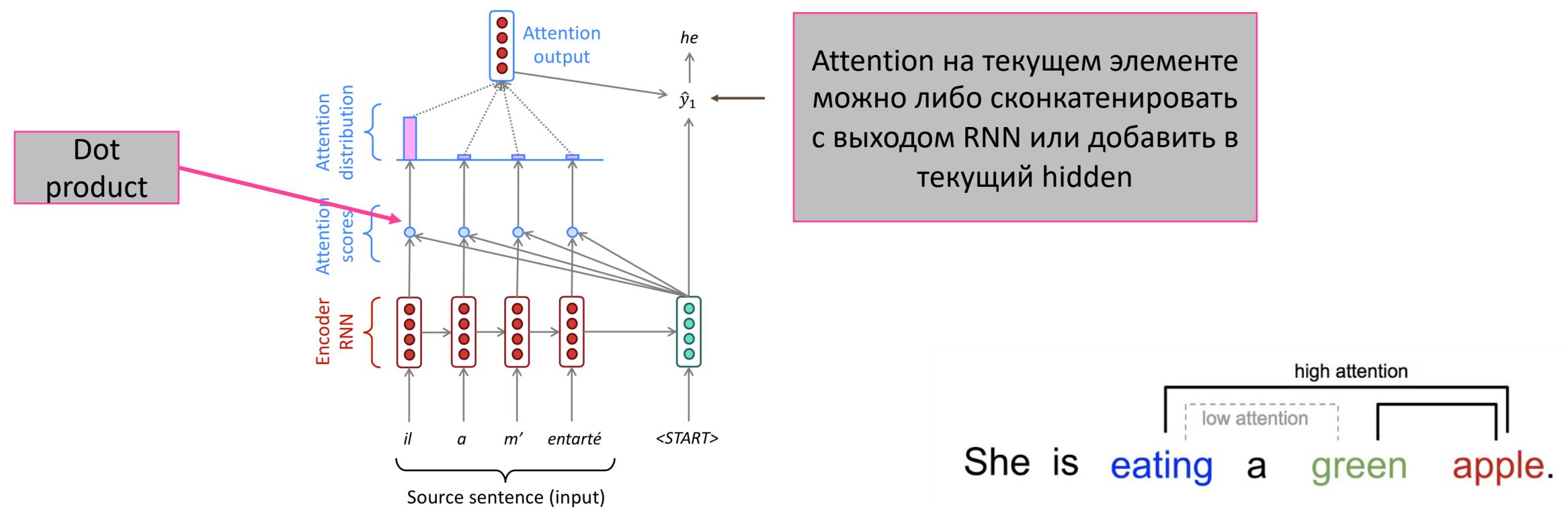
Проблемы Seq2seq



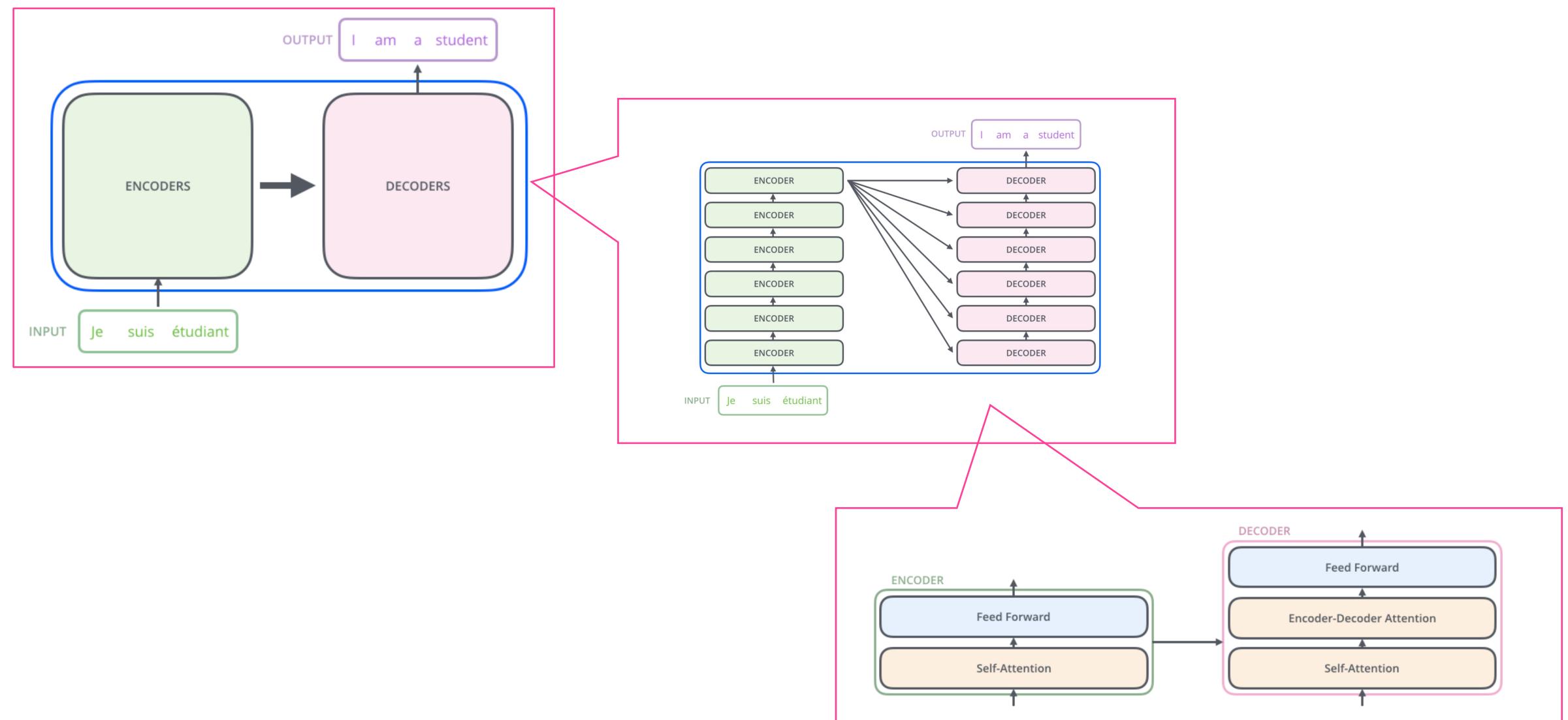
- Входная последовательность символов кодируется в один вектор
- Долго обучаются – обучение нельзя распараллелить

Механизм внимания(attention)

- Attention позволяет решить проблему раскодирования целого предложения из одного вектора
- Помогает определить вклад каждого входного слова в текущее переводимое
- На каждом этапе модель “смотрит” на разные входные элементы, что позволяет учитывать информацию с этапа энкодера.



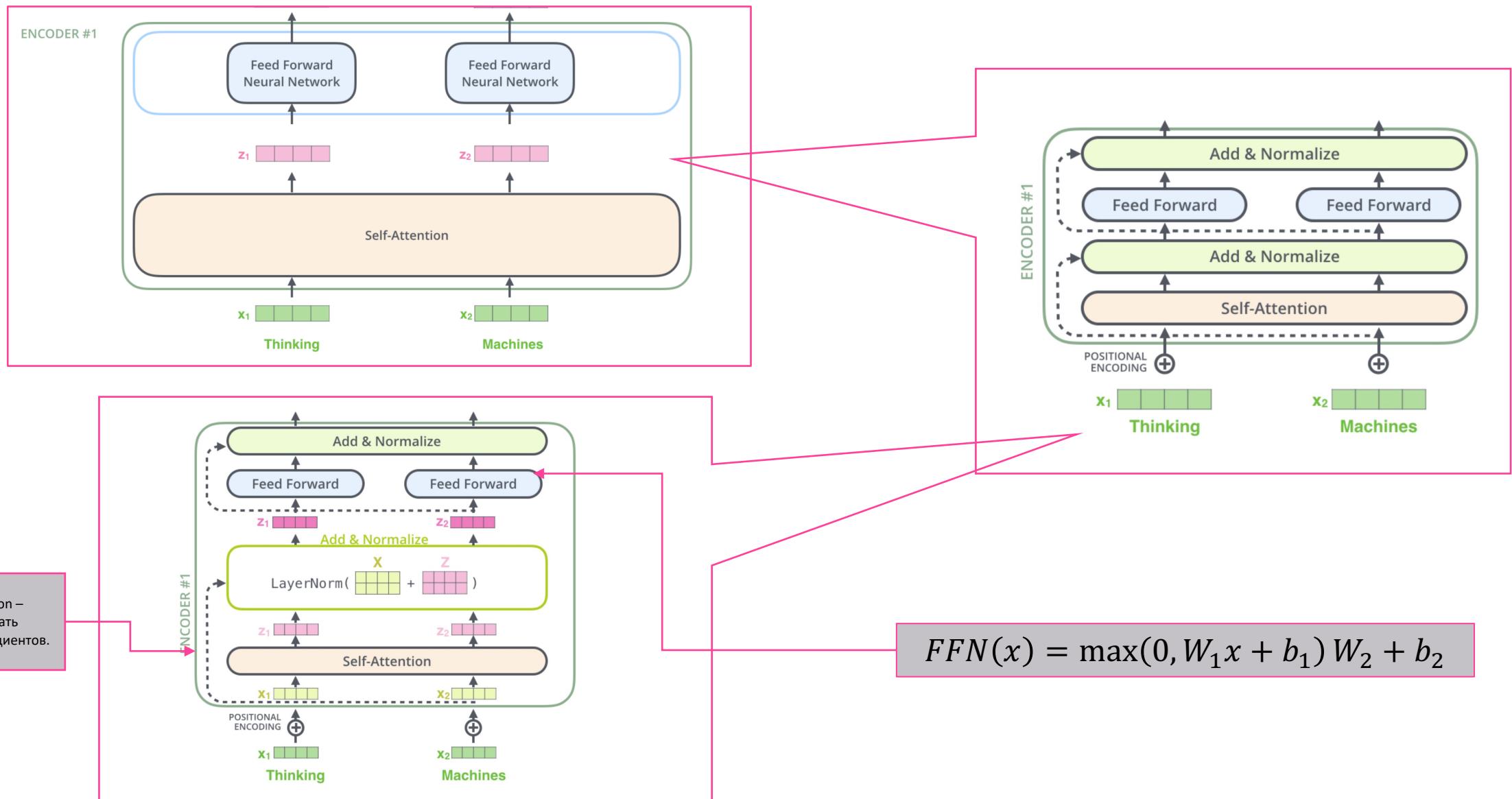
Архитектура трансформера



<https://arxiv.org/pdf/1706.03762.pdf>

<https://jalammar.github.io/illustrated-transformer/>

Архитектура трансформера

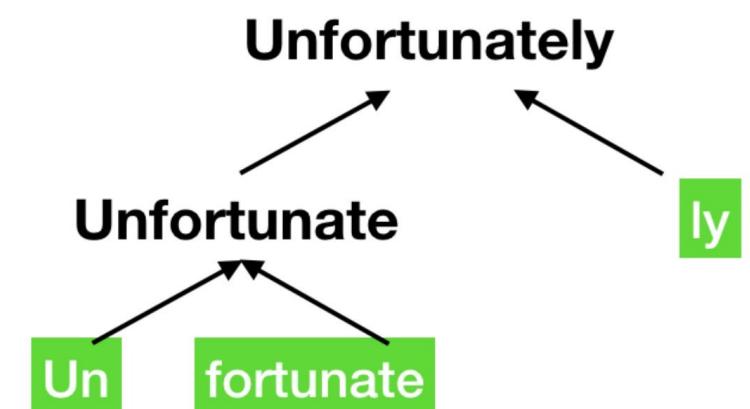


Токенизация

- В процессе обучения получаются огромные словари:
 - В русском языке очень сложная морфология -> огромное количество словоформ
 - Высокий показатель perplexity
- Для того чтобы уменьшить размер словаря в языковом моделировании есть множество приемов на уровне слов, н-грамм, символов
 - Современный подход – уменьшение размера словаря на разбиение входного слова на сабворды
 - На этапе тренировки и тестирования входная последовательность разбивается на сабворды
- BPE-алгоритм(byte-pair encoding)
 - Начинаем с множества символов(алфавит, цифры и тд) и символа конца слова(#)
 - Используя большой корпус текста находим самые часто встречающиеся вместе буквы – (а, м). Добавляем в словарь сочетание ам
 - Создаем словарь до нужного размера(В BERT-base – 30k)
 - Входное предложение жадно токенизуем по словарю

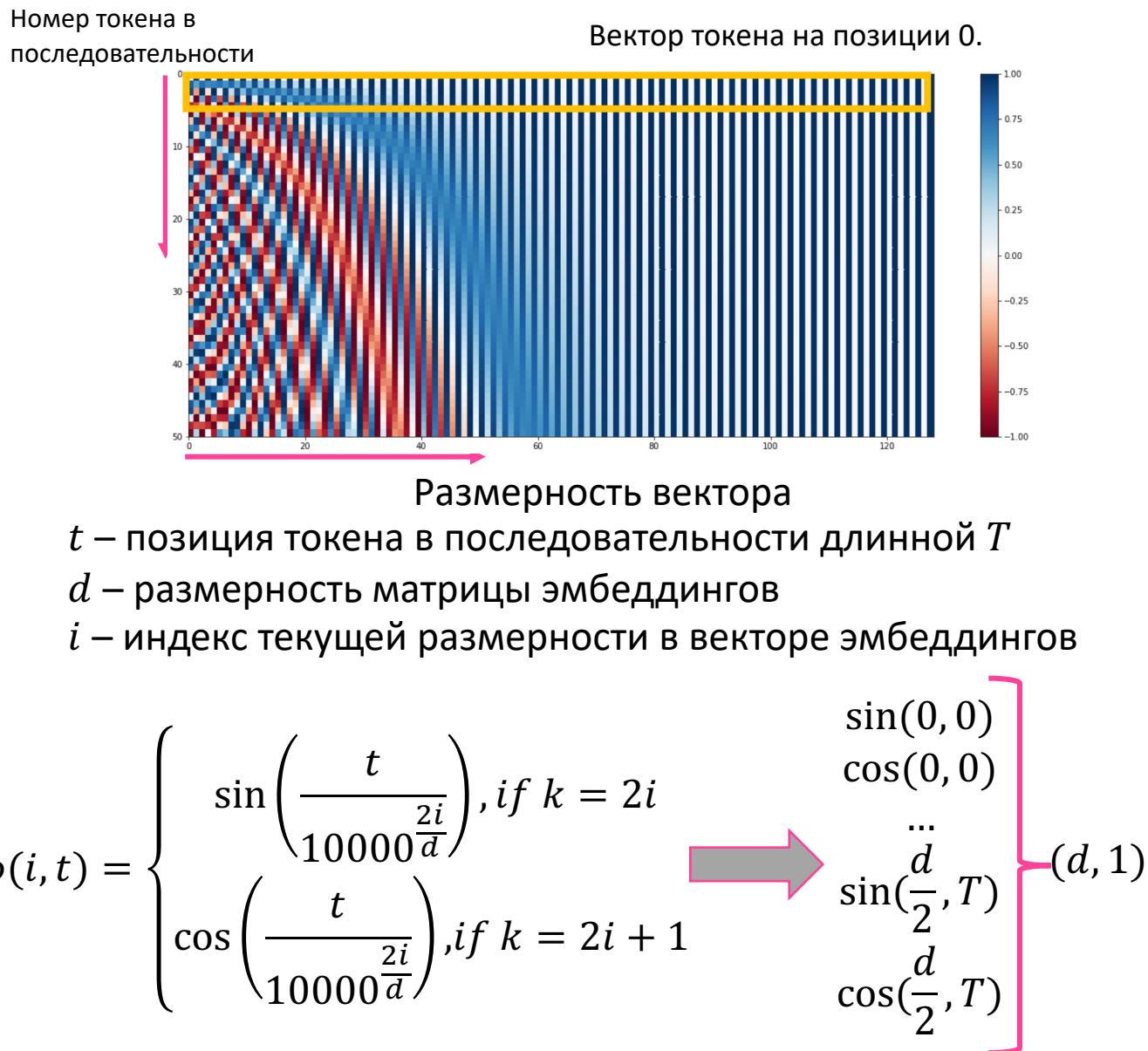
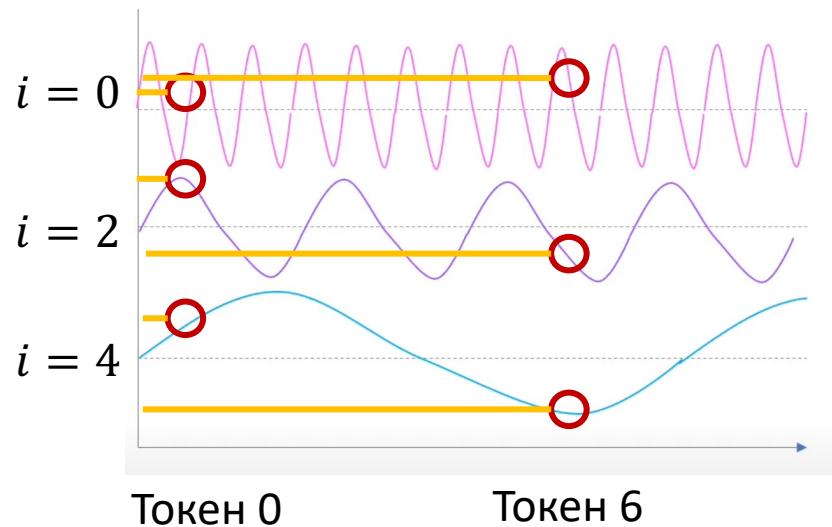
```
#tokenizer example
string_example = 'i have a funny story about dinosours'
print(TOKENIZER.tokenize(string_example))
print(TOKENIZER.encode(string_example))

['i', 'have', 'a', 'funny', 'story', 'about', 'dino', '##so', '##urs']
[101, 1045, 2031, 1037, 6057, 2466, 2055, 22412, 6499, 9236, 102]
```



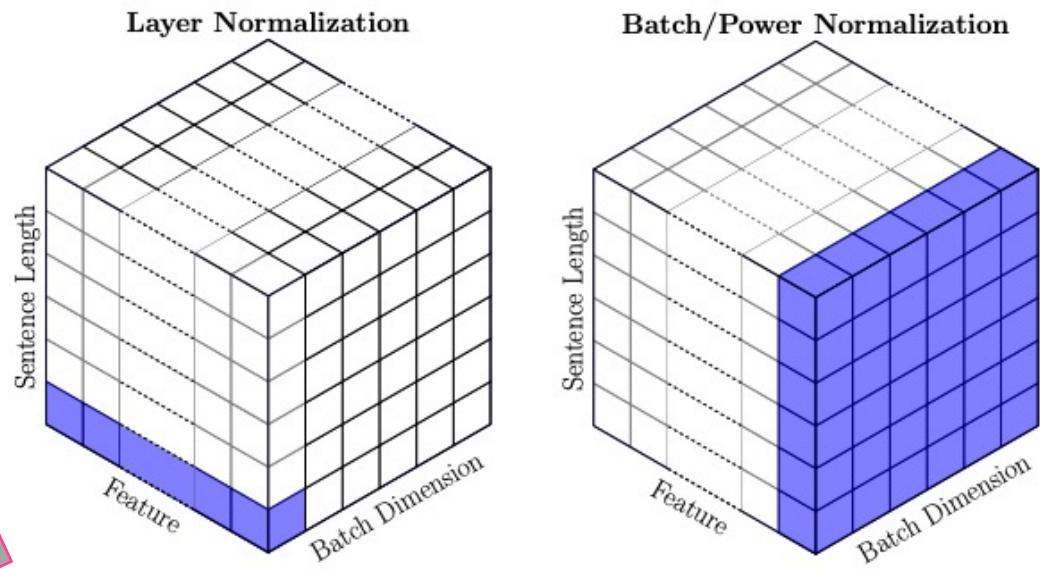
Позиционные эмбеддинги

- Self-attention и сам трансформер не обладают рекуррентной архитектурой и не учитывают порядок слов в предложении. Как учитывать порядок слов в предложении? **Positional encoding**
- Представим каждый элемент последовательности вектором характеризующим позицию - $p_t \in R^d$, где $t \in \{0, 1, \dots, T\}$
- Информацию легко внести в модель – просто добавим к эмбеддингам слов вектор позиции – сложением $\hat{e}_t = e_t + p_t$ или конкатенацией $\hat{e}_t = [e_t; p_t]$
- Требования к функции - значения вектора позиции не должно меняться в зависимости от длины последовательности.



Batch Normalization vs Layer Normalization

- + Не зависит от batch-size
- + Можно применить в RNN, каждое последующее состояние зависит от предыдущего
- + Можно использовать в распределенном обучении



```
gamma = torch.Tensor(requires_grad=True)
beta = torch.Tensor(requires_grad=True)
```

```
mean = x.mean(dim=1, keepdim=True)
var = ((x - mean) ** 2).mean(dim=1, keepdim=True)
std = (var + eps).sqrt()
y = (x - mean) / std
y = gamma * y + beta
```

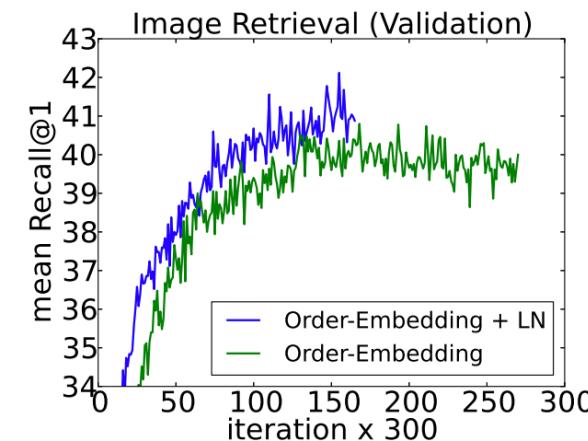
```
gamma = torch.Tensor(requires_grad=True)
beta = torch.Tensor(requires_grad=True)
```

```
mean = x.mean(dim=0, keepdim=True)
var = ((x - mean) ** 2).mean(dim=0, keepdim=True)
std = (var + eps).sqrt()
y = (x - mean) / std
y = gamma * y + beta
```

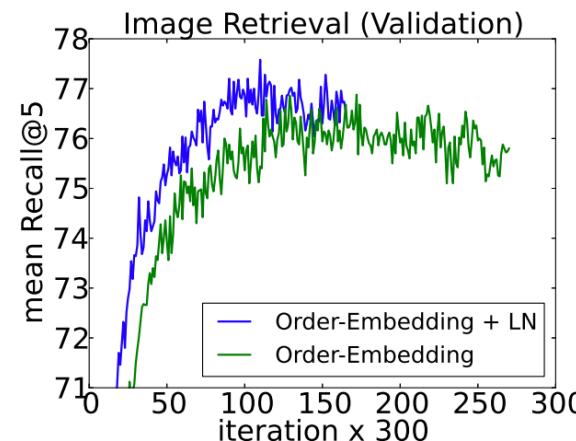
- Зависит от batch-size
- Сложно применить в RNN, каждое последующее состояние зависит от предыдущего
- Сложно использовать в распределенном обучении

<https://arxiv.org/pdf/1607.06450.pdf>

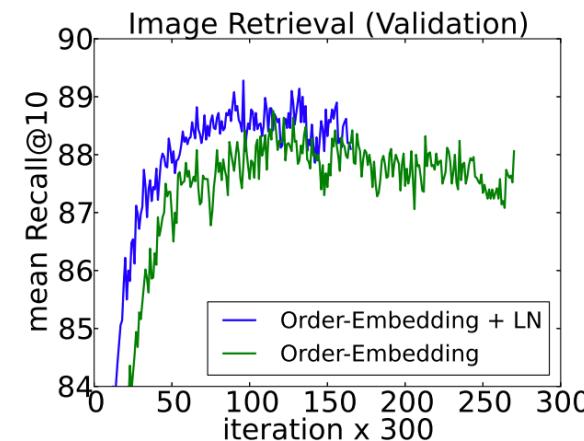
Layer Normalization



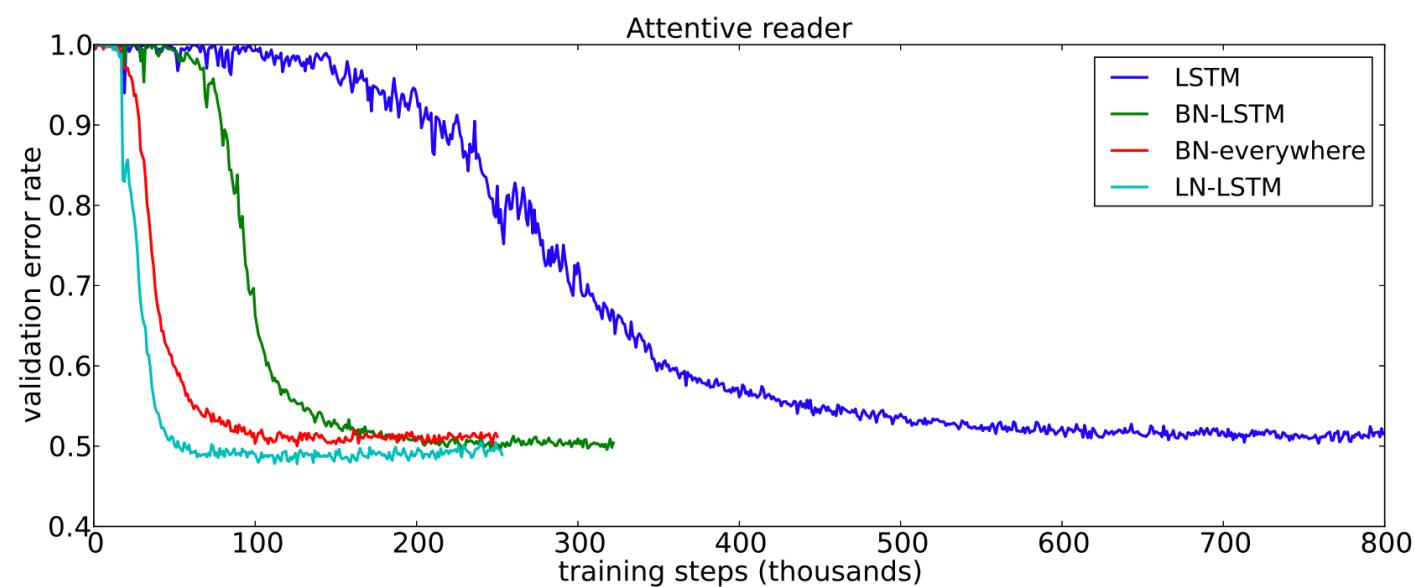
(a) Recall@1



(b) Recall@5



(c) Recall@10



Self-attention

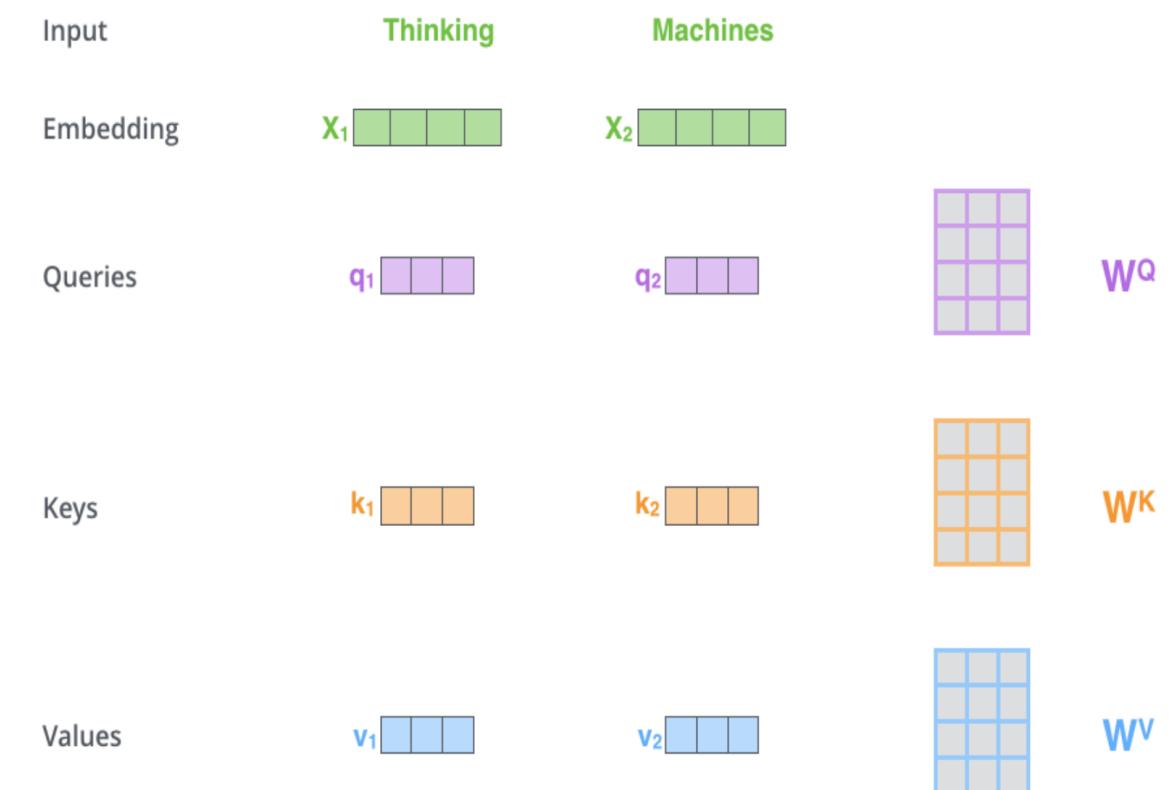
Self-attention - кодирует информацию о влиянии других слов на текущее слово.

Есть 3 вектора

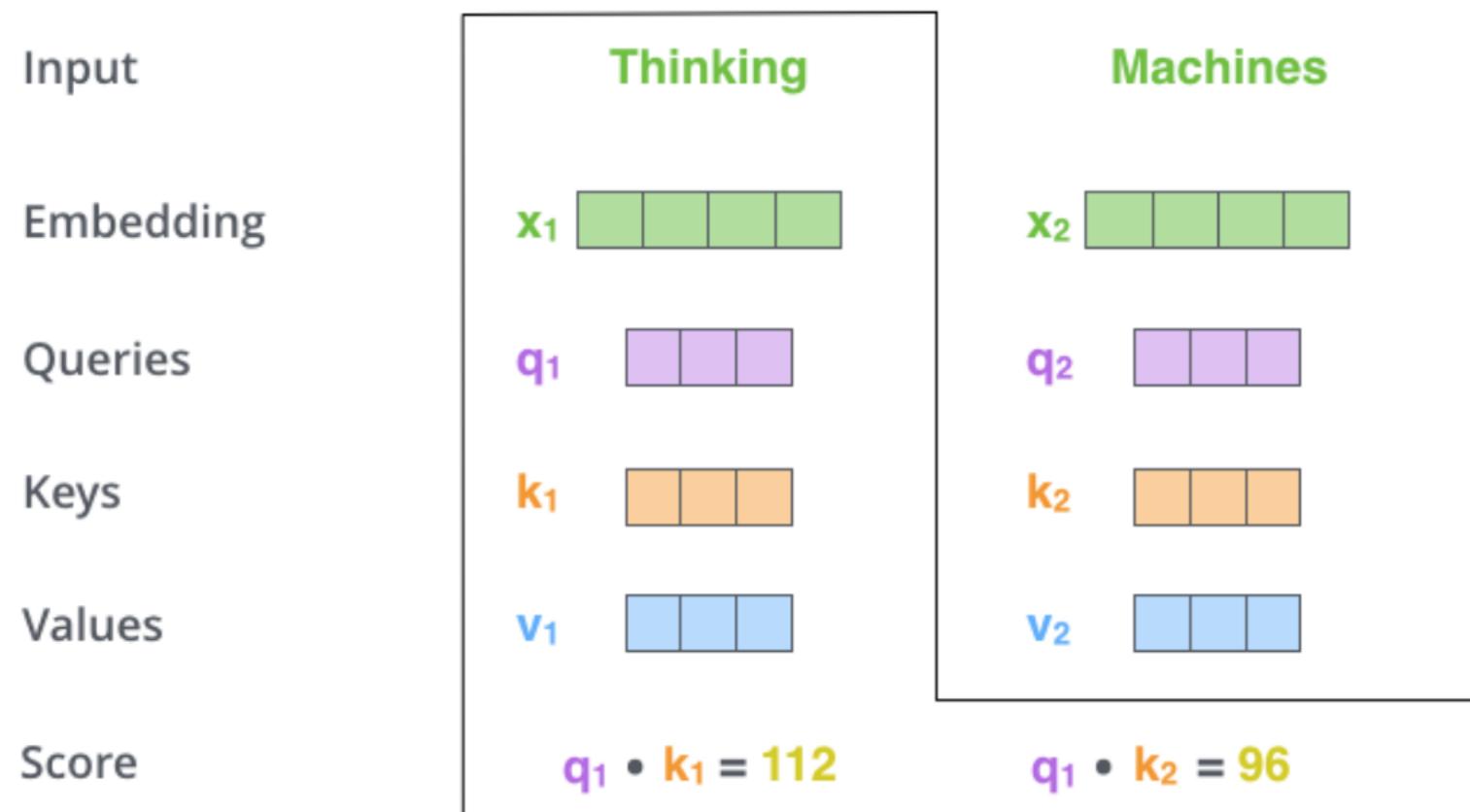
- Query – вектор текущего слова в последовательности
- Key – вектор характеризующий остальные слова
- Value – вектор, учитывающий влияние всех keys на query

$$\text{Attention vector} = \text{softmax} \left(\frac{\text{Query} * \text{Key}}{\sqrt{d_k}} \right) * \text{Value}$$

W_q, W_k, W_v – обучающиеся матрицы для преобразования query, key, value в новое пространство

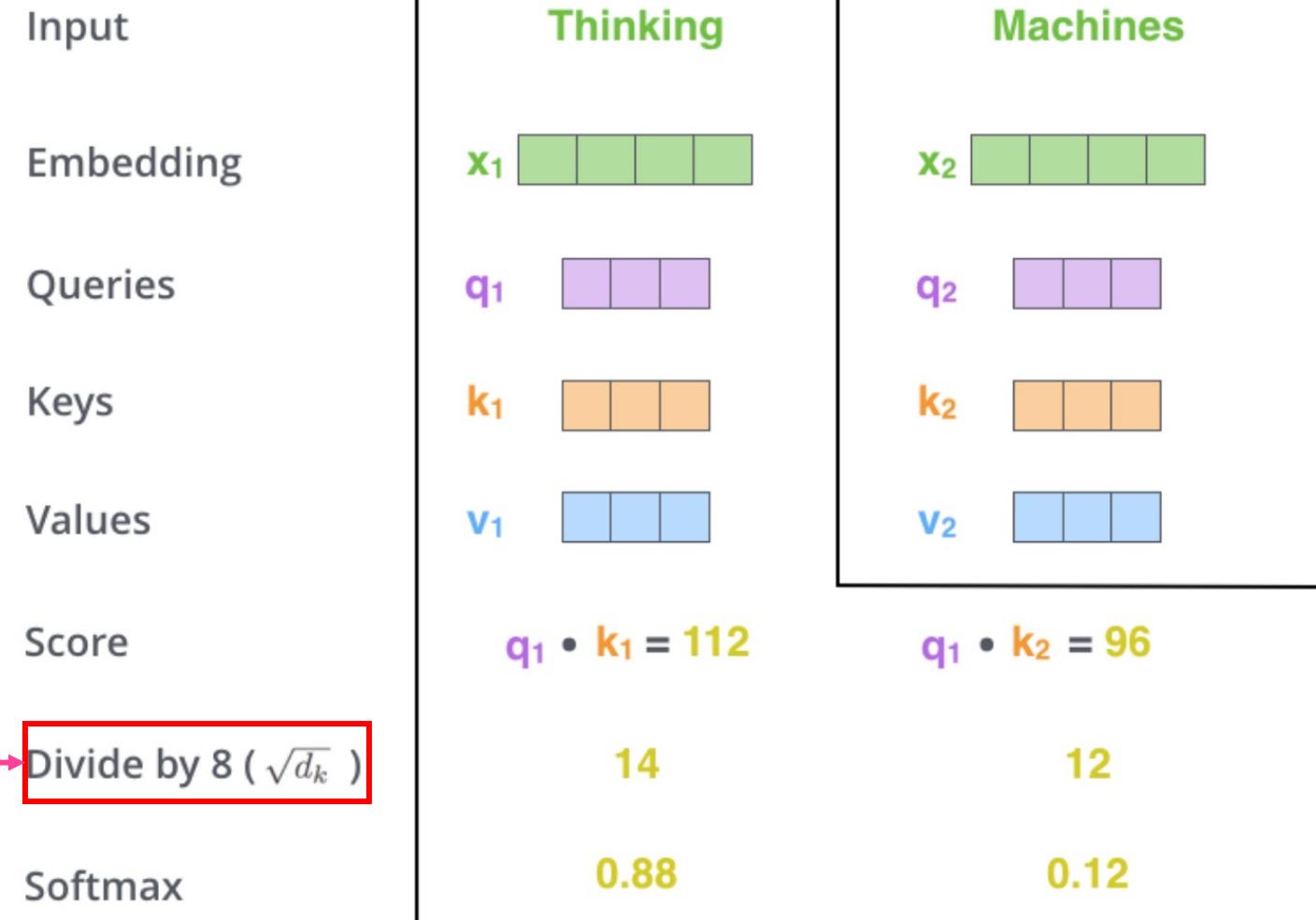


Self-attention

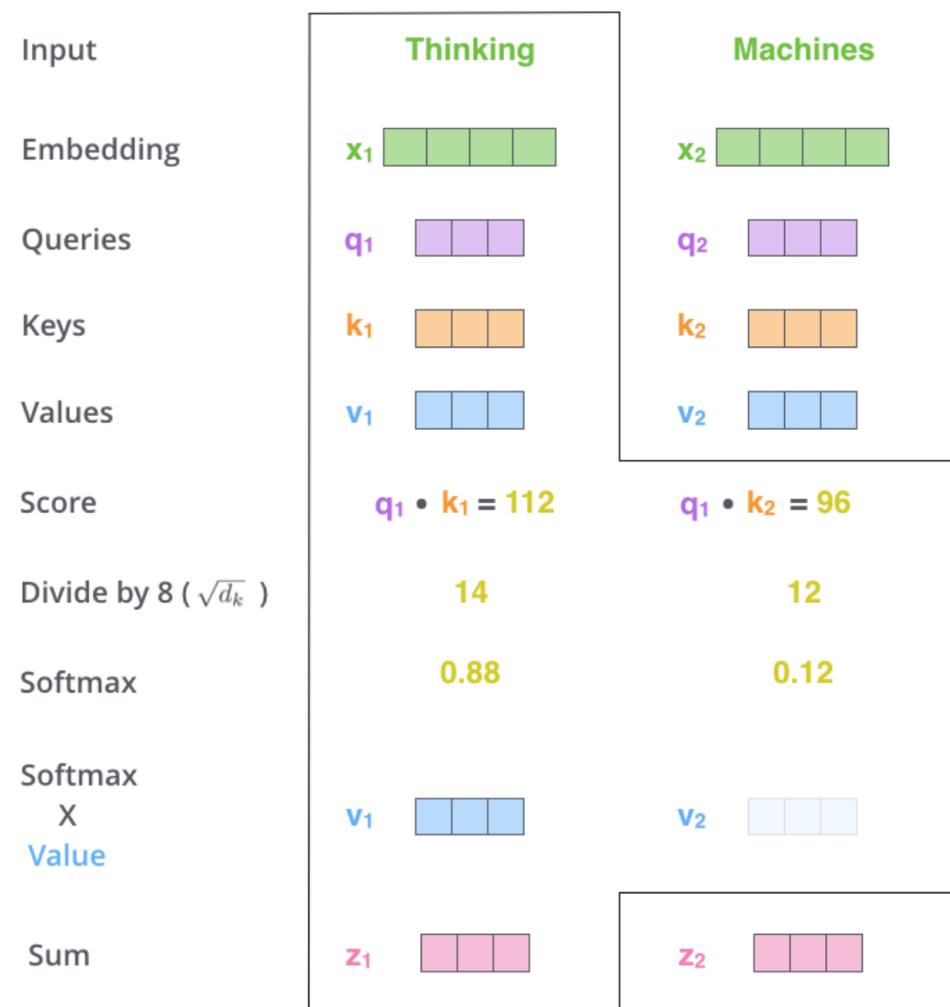


Self-attention

Скалярное перемножение q и k может дать большие значения, тем самым сделав градиенты маленькими. d – размерность матрицы attention.
Поэтому делим, чтобы уменьшить результат dot product key и query



Self-attention



Self-attention

$$X \times W^Q = Q$$

Diagram illustrating the computation of Query (Q) from input X. Input X is represented as a green 2x4 matrix. It is multiplied by weight matrix W^Q (purple 4x4 matrix) to produce Query matrix Q (purple 2x2 matrix).

$$X \times W^K = K \rightarrow \text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

Diagram illustrating the computation of Key (K) and Value (V) from input X. Input X is represented as a green 2x4 matrix. It is multiplied by weight matrix W^K (orange 4x4 matrix) to produce Key matrix K (orange 2x2 matrix). A pink arrow points to the softmax operation.

The softmax operation is defined as:

$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right)$$

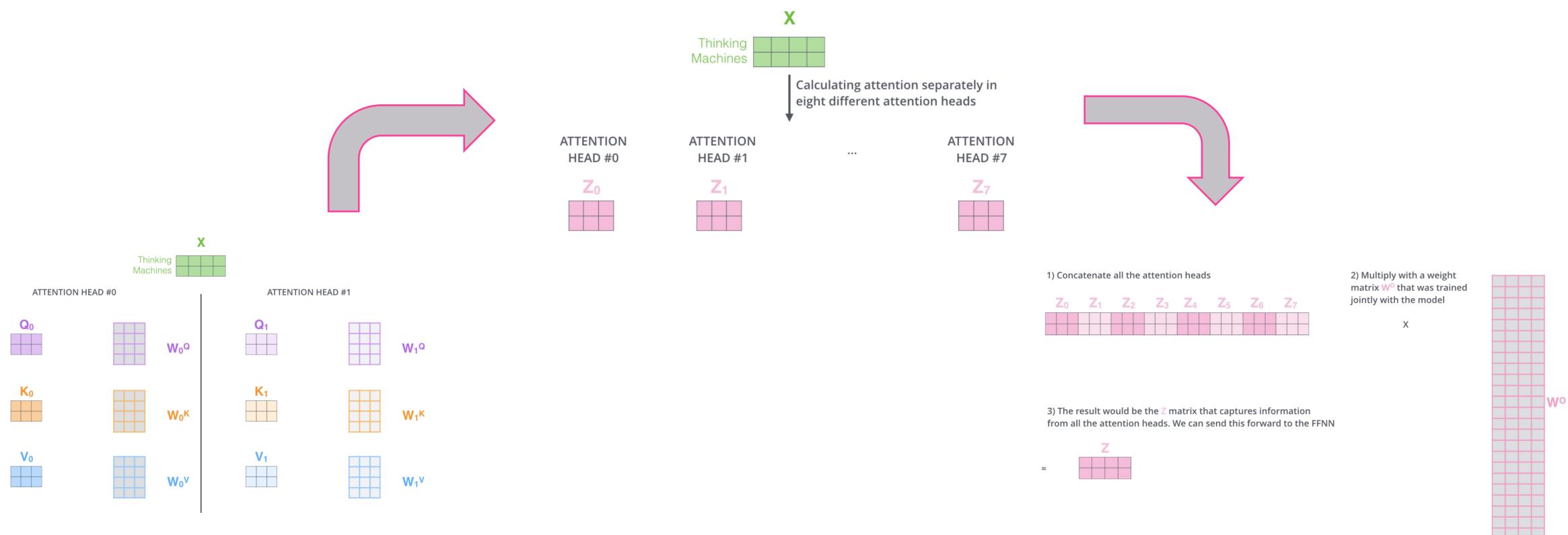
where Q is the Query matrix (purple 2x2), K^T is the transpose of the Key matrix (orange 2x2), and $\sqrt{d_k}$ is the square root of the dimension of the key space.

$$X \times W^V = V$$

Diagram illustrating the computation of Value (V) from input X. Input X is represented as a green 2x4 matrix. It is multiplied by weight matrix W^V (blue 4x4 matrix) to produce Value matrix V (blue 2x2 matrix).

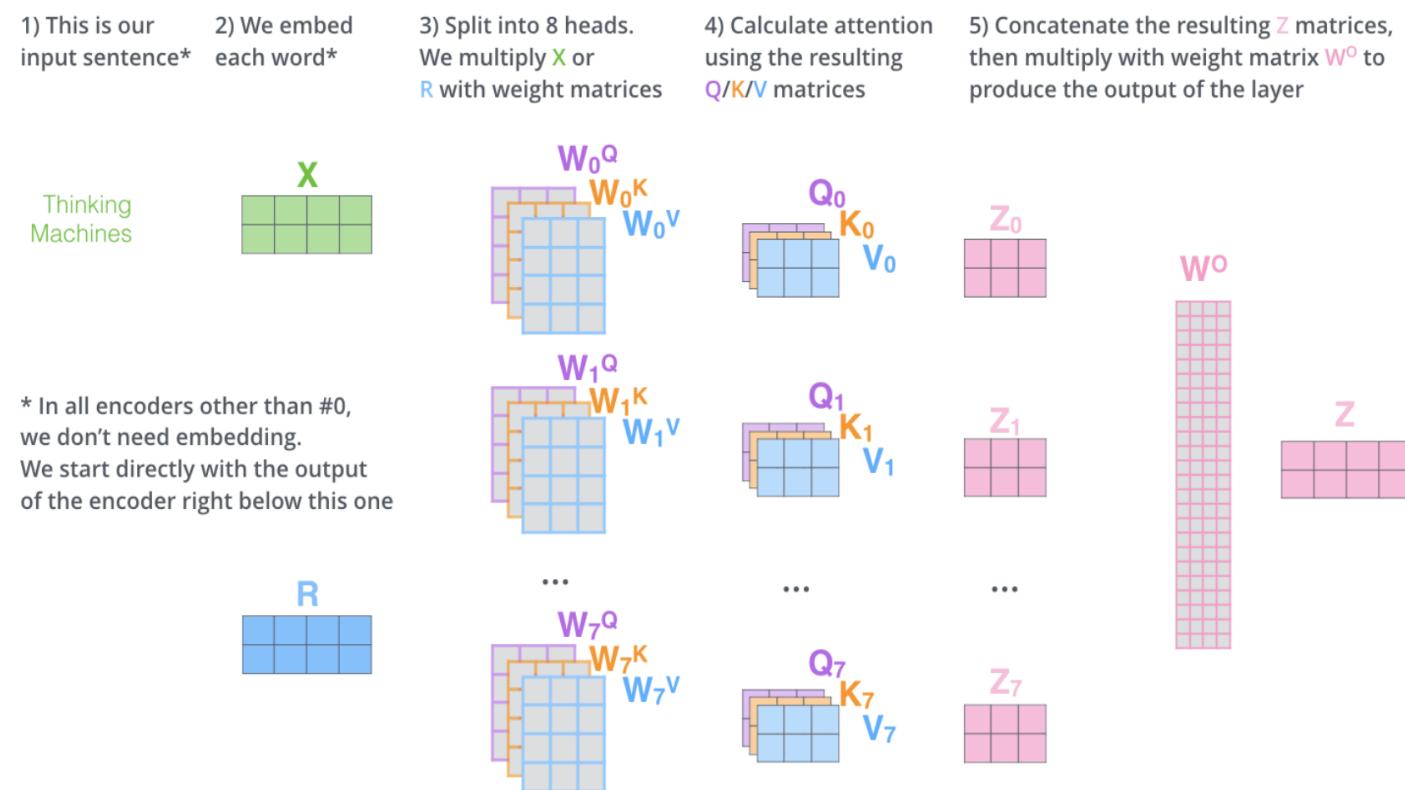
Multi-head attention

- W_k, W_q, W_v - случайно инициализированные матрицы, которые позволяют оценивать влияние остальных слов на текущее
- Множество матриц W_k^i, W_q^i, W_v^i позволяют оценить это влияние с разных сторон
- Давайте увеличим количество self-attention матриц

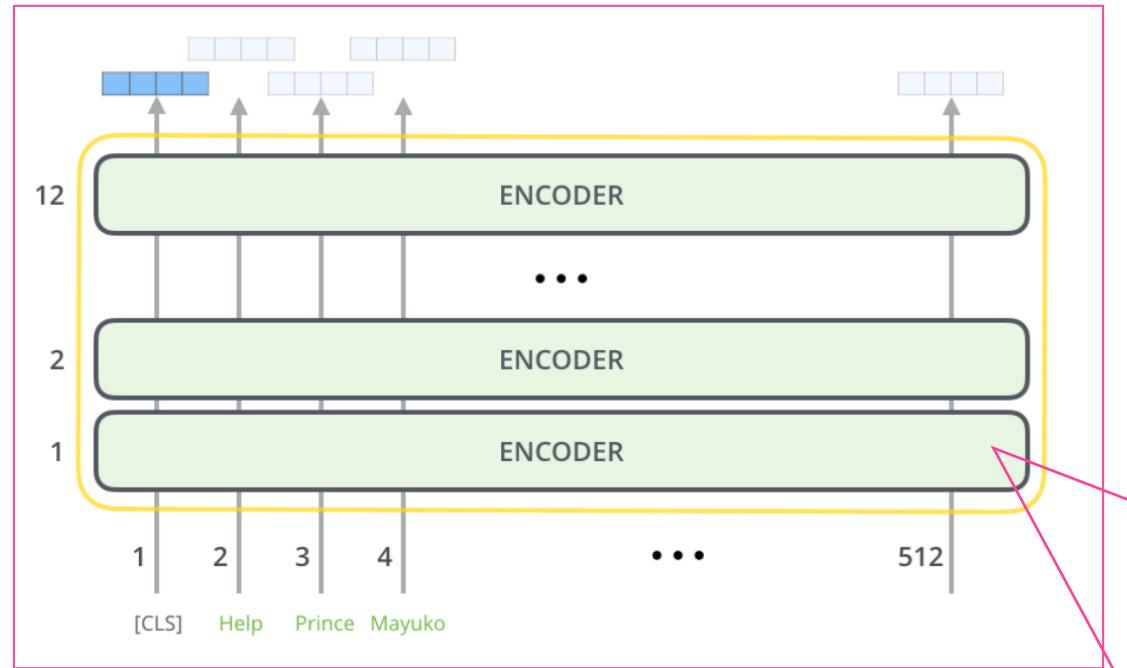


Multi-head attention

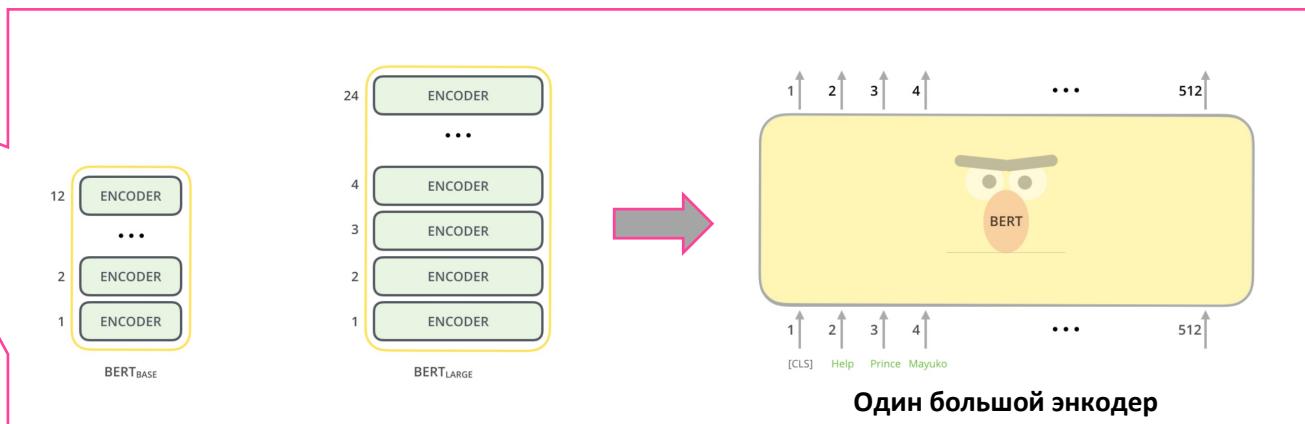
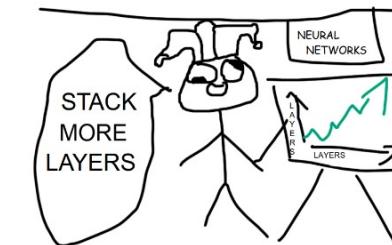
- Multi-head attention – позволяет увеличить кол-во информации о словах, которые влияют на текущее слово за счет увеличения кол-ва матриц.
- Каждое слово получает столько векторных взвешенных представлений, сколько есть “голов” у механизма внимания.



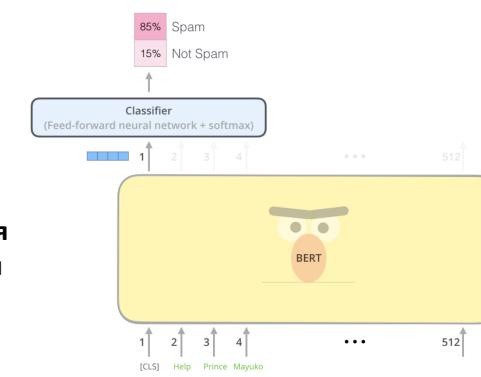
Bidirectional Encoder Representations from Transformers (BERT)



Стек энкодеров



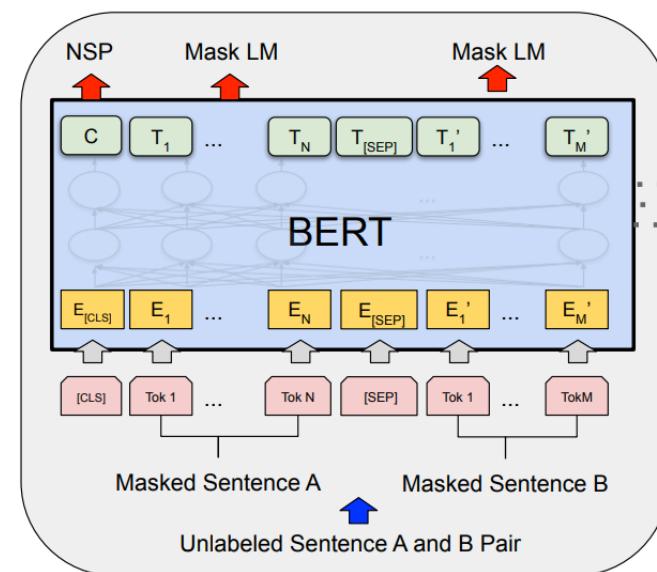
Инструмент для
решения задач
NLP



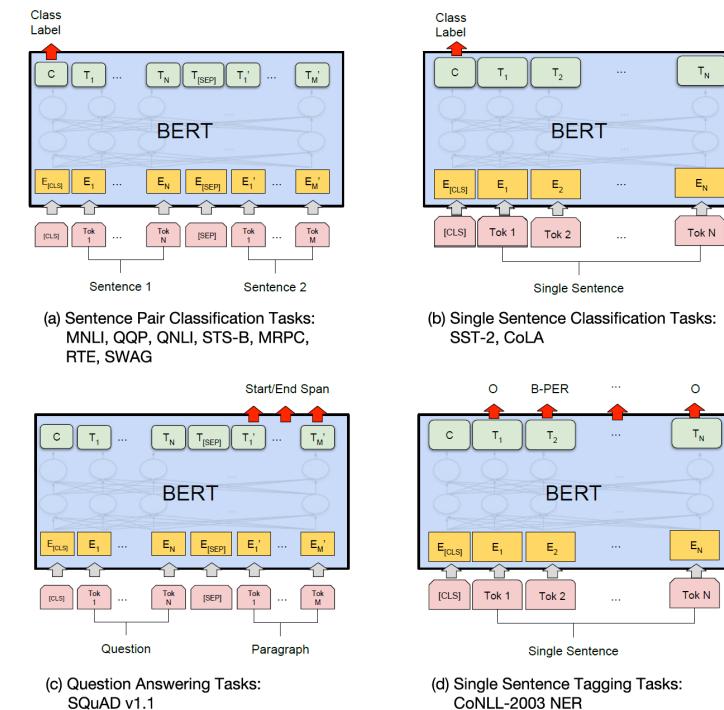
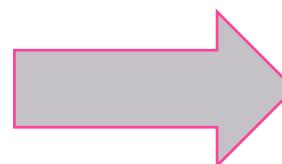
<https://arxiv.org/abs/1810.04805>

Фазы обучения BERT

- Основная идея – использовать большую предобученную модель языкового моделирования для решения небольшой конкретной задачи.
- Способы использования
 - Векторизация и использование, как эмбеддингов
 - Fine-tuning
- Дешевый способ значительно улучшить качество предсказания.



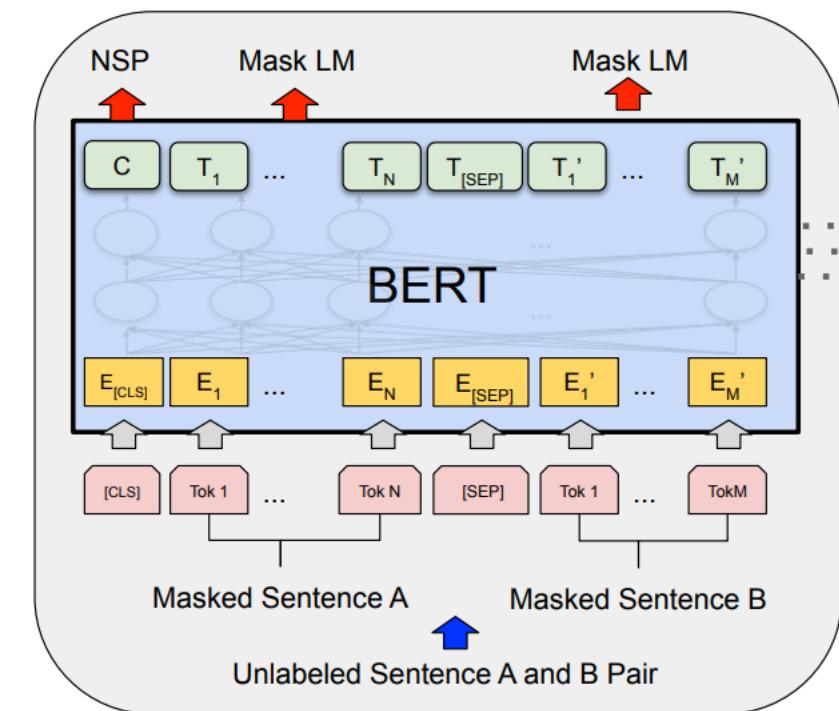
Pretraining



Finetuning

BERT pretraining

- $BERT \text{ pretrain loss} = L_{NSP} + L_{MLM}$
- **Лосс в обоих случаях – сумма cross-entropy loss на двух задачах**
 - Предсказание некоторого слова в предложении – оценка распределения по всему словарю(masked language modeling)
 - Я пошел в MASK и купил MASK
 - Определение логичности следующего предложения – классификация(next sentence prediction)
 - Я пошел в магазин. Там я купил молоко. – **True**
 - Я купил собаку. В Москве холодаает. – **False**



Ключевые пункты обучения BERT

- Обучение с учителем
 - Маскируется 15% токенов в предложении – решается задача MLM
 - Берется либо последующее предложение из текста, либо случайное – решается задача NSP
- Модели BERT
 - BERT-base – 12 трансформеров, hidden dim 768, 12 attention heads, 110 миллионов параметров
 - BERT-large – 24 трансформера, hidden dim 1024, 16 attention heads, 340 миллионов параметров
- Датасеты
 - Wikipedia
 - Корпуса книг
- Обучение
 - BERT pretraining делают на TPU
 - BERT finetuning делают на GPU

Метрики на SuperGlue

| Rank Name | Model | URL | Score | CoLA | SST-2 | MRPC | STS-B | QQP | MNLI-m | MNLI-mm | QNLI | RTE | WNLI | AX |
|---|---|-----|-------|------|-------|-----------|-----------|-----------|--------|---------|------|------|------|------|
| 1 ERNIE Team - Baidu | ERNIE | | 90.9 | 74.4 | 97.8 | 93.9/91.8 | 93.0/92.6 | 75.2/90.9 | 91.9 | 91.4 | 97.3 | 92.0 | 95.9 | 51.7 |
| 2 DeBERTa Team - Microsoft | DeBERTa / TuringNLVRv4 | | 90.8 | 71.5 | 97.5 | 94.0/92.0 | 92.9/92.6 | 76.2/90.8 | 91.9 | 91.6 | 99.2 | 93.2 | 93.2 | 52.2 |
| 3 HFL iFLYTEK | MacALBERT + DKM | | 90.7 | 74.8 | 97.0 | 94.5/92.6 | 92.8/92.6 | 74.7/90.6 | 91.3 | 91.1 | 97.8 | 92.0 | 94.5 | 52.6 |
| 4 Alibaba DAMO NLP | StructBERT + TAPT | | 90.6 | 75.3 | 97.3 | 93.9/91.9 | 93.2/92.7 | 74.8/91.0 | 90.9 | 90.7 | 97.4 | 91.2 | 94.5 | 49.1 |
| 5 PING-AN Omni-Sinicic | ALBERT + DAAF + NAS | | 90.6 | 73.5 | 97.2 | 94.0/92.0 | 93.0/92.4 | 76.1/91.0 | 91.6 | 91.3 | 97.5 | 91.7 | 94.5 | 51.2 |
| 6 T5 Team - Google | T5 | | 90.3 | 71.6 | 97.5 | 92.8/90.4 | 93.1/92.8 | 75.1/90.6 | 92.2 | 91.9 | 96.9 | 92.8 | 94.5 | 53.1 |
| 7 Microsoft D365 AI & MSR AI & GATECHMT-DNN-SMART | | | 89.9 | 69.5 | 97.5 | 93.7/91.6 | 92.9/92.5 | 73.9/90.2 | 91.0 | 90.8 | 99.2 | 89.7 | 94.5 | 50.2 |
| 8 Huawei Noah's Ark Lab | NEZHA-Large | | 89.8 | 71.7 | 97.3 | 93.3/91.0 | 92.4/91.9 | 75.2/90.7 | 91.5 | 91.3 | 96.2 | 90.3 | 94.5 | 47.9 |
| 9 Zihang Dai | Funnel-Transformer (Ensemble B10-10H1024) | | 89.7 | 70.5 | 97.5 | 93.4/91.2 | 92.6/92.3 | 75.4/90.7 | 91.4 | 91.1 | 95.8 | 90.0 | 94.5 | 51.6 |
| 10 ELECTRA Team | ELECTRA-Large + Standard Tricks | | 89.4 | 71.7 | 97.1 | 93.1/90.7 | 92.9/92.5 | 75.6/90.8 | 91.3 | 90.8 | 95.8 | 89.8 | 91.8 | 50.7 |
| 11 Microsoft D365 AI & UMD | FreeLB-RoBERTa (ensemble) | | 88.4 | 68.0 | 96.8 | 93.1/90.8 | 92.3/92.1 | 74.8/90.3 | 91.1 | 90.7 | 95.6 | 88.7 | 89.0 | 50.1 |

Весна 2021

| Rank Name | Model | URL | Score | CoLA | SST-2 | MRPC | STS-B | QQP | MNLI-m | MNLI-mm | QNLI | RTE | WNLI | AX |
|---|------------------------|-----|-------|------|-------|-----------|-----------|-----------|--------|---------|------|------|------|------|
| 1 JDExplore d-team | Vega v1 | | 91.3 | 73.8 | 97.9 | 94.5/92.6 | 93.5/93.1 | 76.7/91.1 | 92.1 | 91.9 | 96.7 | 92.4 | 97.9 | 51.4 |
| 2 Microsoft Alexander v-team | Turing NLR v5 | | 91.2 | 72.6 | 97.6 | 93.8/91.7 | 93.7/93.3 | 76.4/91.1 | 92.6 | 92.4 | 97.9 | 94.1 | 95.9 | 57.0 |
| 3 DIRL Team | DeBERTa + CLEVER | | 91.1 | 74.7 | 97.6 | 93.3/91.1 | 93.4/93.1 | 76.5/91.0 | 92.1 | 91.8 | 96.7 | 93.2 | 96.6 | 35.2 |
| 4 ERNIE Team - Baidu | ERNIE | | 91.1 | 75.5 | 97.8 | 93.9/91.8 | 93.0/92.6 | 75.2/90.9 | 92.3 | 91.7 | 97.3 | 92.6 | 95.9 | 51.7 |
| 5 AliceMind & DIRL | StructBERT + CLEVER | | 91.0 | 75.3 | 97.7 | 93.9/91.9 | 93.5/93.1 | 75.6/90.8 | 91.7 | 91.5 | 97.4 | 92.5 | 95.2 | 49.1 |
| 6 DeBERTa Team - Microsoft | DeBERTa / TuringNLVRv4 | | 90.8 | 71.5 | 97.5 | 94.0/92.0 | 92.9/92.6 | 76.2/90.8 | 91.9 | 91.6 | 99.2 | 93.2 | 94.5 | 53.2 |
| 7 HFL iFLYTEK | MacALBERT + DKM | | 90.7 | 74.8 | 97.0 | 94.5/92.6 | 92.8/92.6 | 74.7/90.6 | 91.3 | 91.1 | 97.8 | 92.0 | 94.5 | 52.6 |
| 8 PING-AN Omni-Sinicic | ALBERT + DAAF + NAS | | 90.6 | 73.5 | 97.2 | 94.0/92.0 | 93.0/92.4 | 76.1/91.0 | 91.6 | 91.3 | 97.5 | 91.7 | 94.5 | 51.2 |
| 9 T5 Team - Google | T5 | | 90.3 | 71.6 | 97.5 | 92.8/90.4 | 93.1/92.8 | 75.1/90.6 | 92.2 | 91.9 | 96.9 | 92.8 | 94.5 | 53.1 |
| 10 Microsoft D365 AI & MSR AI & GATECH MT-DNN-SMART | | | 89.9 | 69.5 | 97.5 | 93.7/91.6 | 92.9/92.5 | 73.9/90.2 | 91.0 | 90.8 | 99.2 | 89.7 | 94.5 | 50.2 |
| 11 Huawei Noah's Ark Lab | NEZHA-Large | | 89.8 | 71.7 | 97.3 | 93.3/91.0 | 92.4/91.9 | 75.2/90.7 | 91.5 | 91.3 | 96.2 | 90.3 | 94.5 | 47.9 |

<https://super.gluebenchmark.com/leaderboard>

Использование BERT e2e

Процесс обучения

1. Токенизируем BPE алгоритмом текст - T
2. Считаем позиционные эмбеддинги - E_{pos}
3. Считаем с помощью матрицы эмбеддингов эмбеддинги для токенов – E_t
4. Считаем с помощью матрицы эмбеддингов эмбеддинги для сегмента – E_s
5. $\hat{E} = E_{pos} + E_t + E_s$
6. Forward-pass через стек энкодеров
7. 2 выхода
 1. Pooler output – mean/max hidden states pooling
 2. Hidden states –hidden представление для токенов последовательности

Фазы обучения

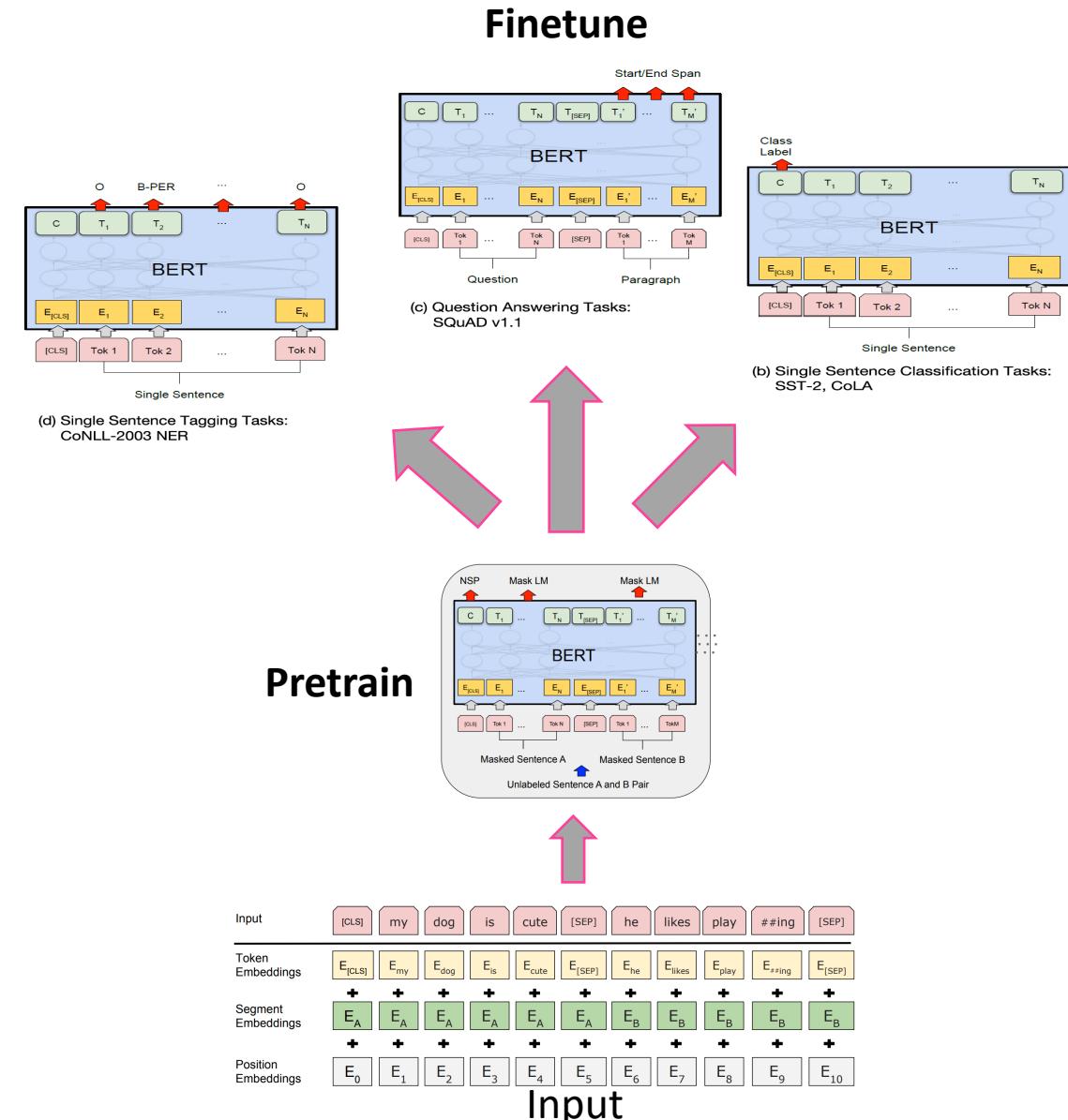
- Pre-train фаза – обучается на сумме двух ошибок NSP и MLM
- Fine-tune фаза – дообучается на специальной задаче(unfreeze layers)

Датасет

- Большой набор данных – википедия + художественные произведения

Результат

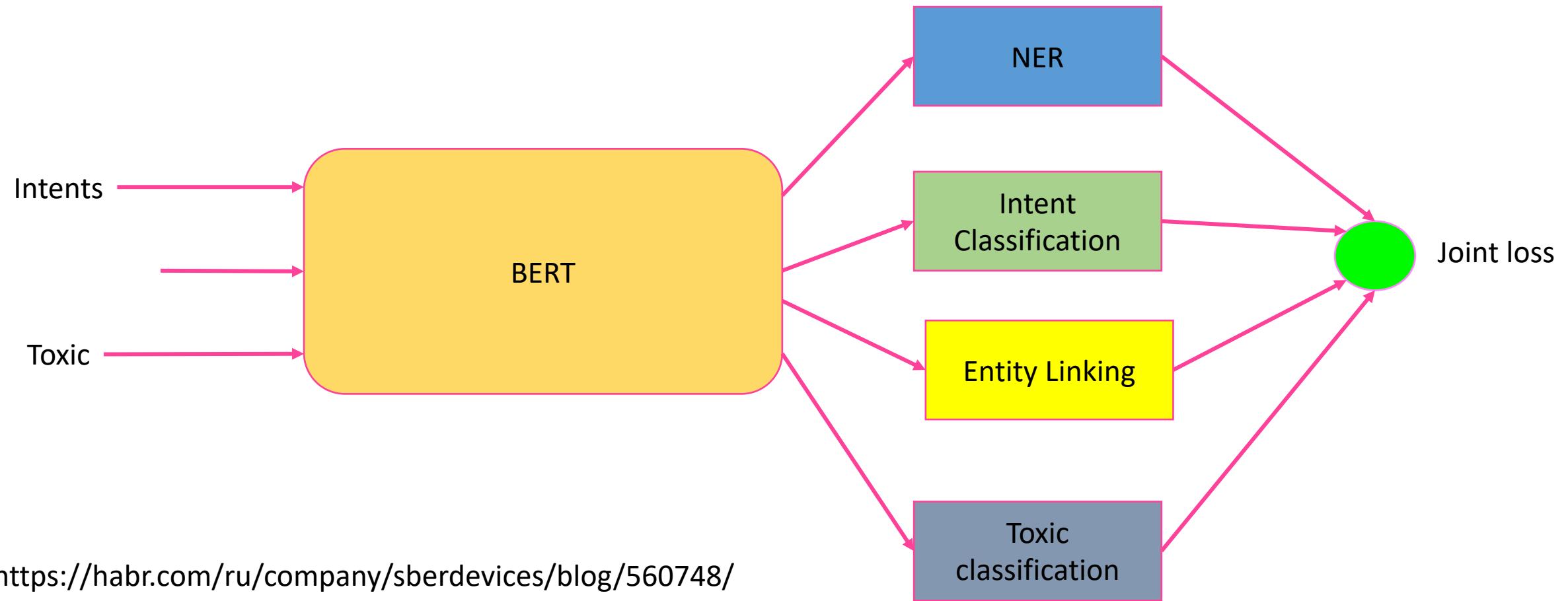
- Большая языковая модель для задач NLP
- Контекстно зависимые словарные эмбеддинги



Multi-task learning c BERT

$$\text{Joint loss} = \sum_{i=0} \text{task loss}_i * \text{alpha}_i$$

$$\sum \text{alpha}_i = 1$$

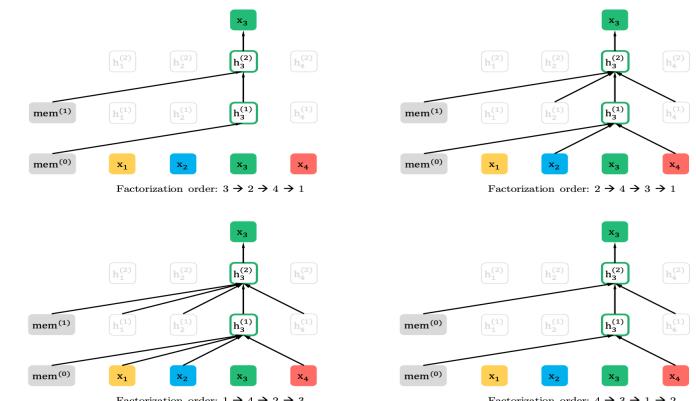
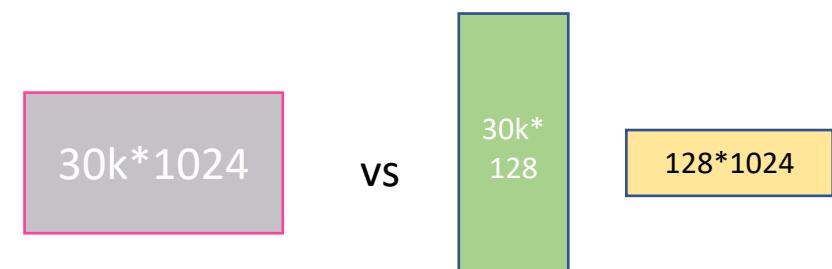


<https://habr.com/ru/company/sberdevices/blog/560748/>

BERT family

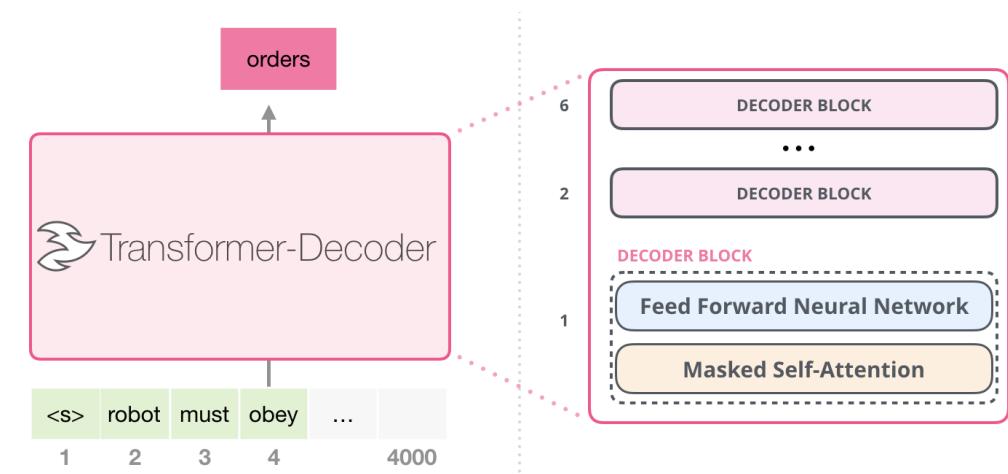
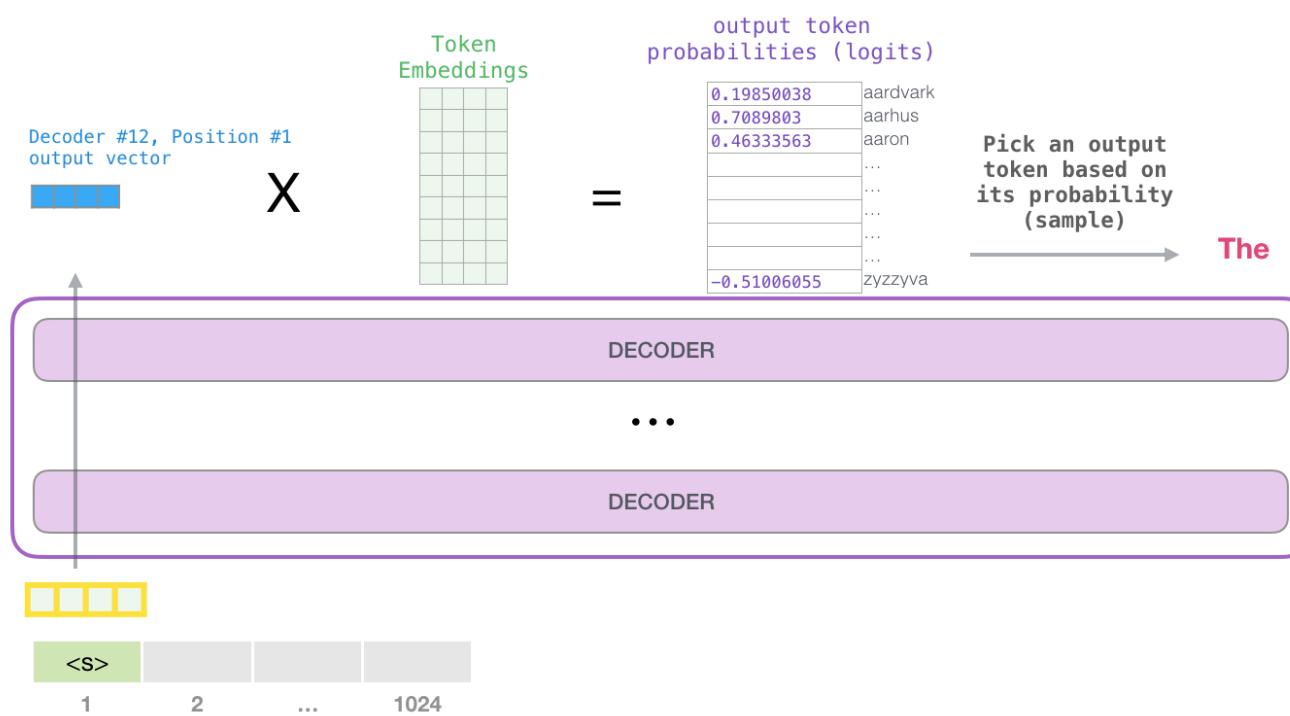
- RoBERTa
 - Увеличение количества эпох и увеличение количества данных влияет на качество BERT
 - Использование только MLM задачи
- ALBERT
 - Уменьшенная матрица эмбеддингов за счет факторизации
 - Веса всех трансформеров общие
- XLNET
 - Изменение positional embedding
 - Обучение авторегрессионной модели на предложениях, где слова в предложении случайным образом переставлены
 - Модель учится по возможным сочетаниям других токенов предсказать целевой токен

| Model | data | bsz | steps | SQuAD (v1.1/2.0) | MNLI-m | SST-2 |
|--------------------------|-------|-----|-------|---------------------|-------------|-------------|
| RoBERTa | | | | | | |
| with BOOKS + WIKI | 16GB | 8K | 100K | 93.6/87.3 | 89.0 | 95.3 |
| + additional data (§3.2) | 160GB | 8K | 100K | 94.0/87.7 | 89.3 | 95.6 |
| + pretrain longer | 160GB | 8K | 300K | 94.4/88.7 | 90.0 | 96.1 |
| + pretrain even longer | 160GB | 8K | 500K | 94.6/89.4 | 90.2 | 96.4 |
| BERT _{LARGE} | | | | | | |
| with BOOKS + WIKI | 13GB | 256 | 1M | 90.9/81.8 | 86.6 | 93.7 |
| XLNet _{LARGE} | | | | | | |
| with BOOKS + WIKI | 13GB | 256 | 1M | 94.0/87.8 | 88.4 | 94.4 |
| + additional data | 126GB | 2K | 500K | 94.5/88.8 | 89.8 | 95.6 |



Generative Pretrained Transformer(GPT)

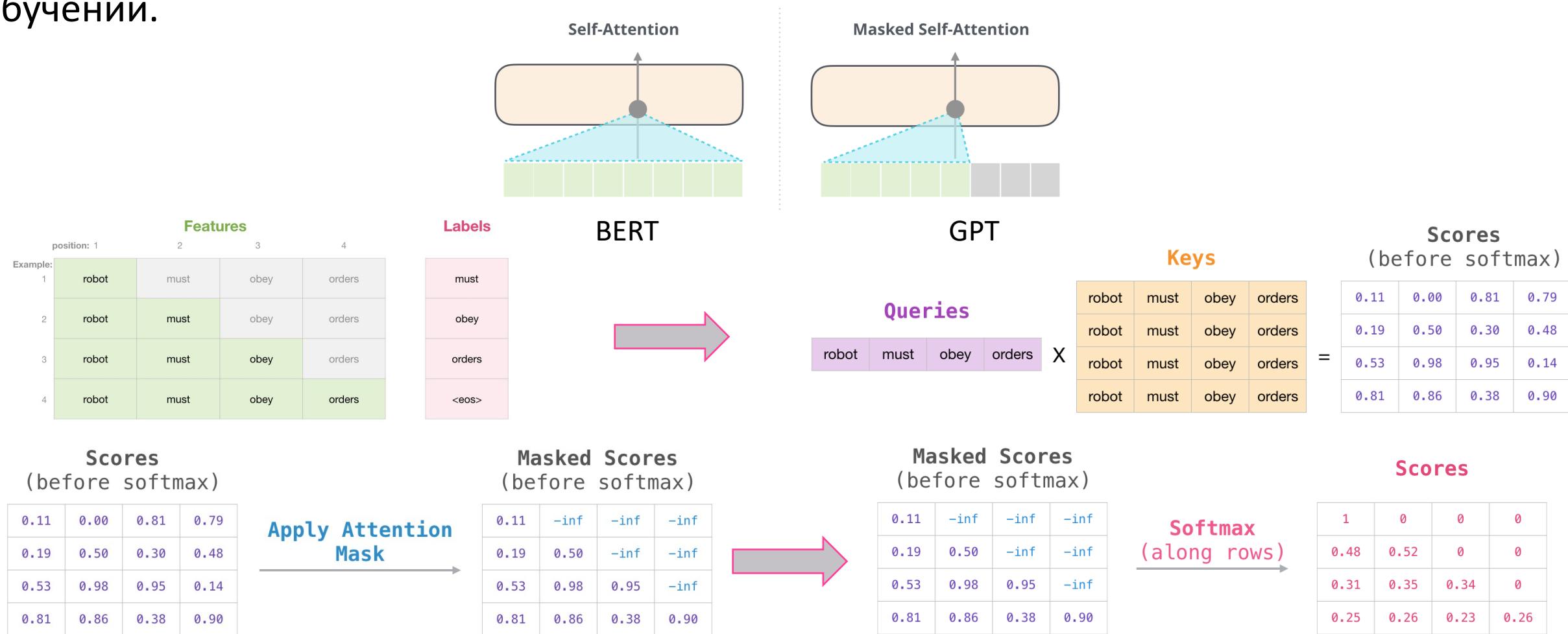
- GPT – большая предобученная языковая модель, состоящая из декодер блоков трансформера.
- Самая большая модель GPT-3 состоит из 175 млрд параметров, обученная на датасете 570 Гб.
- Увеличение модели состоит в увеличении количества блоков декодера, увеличении размерности эмбеддингов и увеличения датасета.
- Основное достижение в обучении таких моделей – создание инфраструктуры, позволяющей обучать такие модели.



<https://jalammar.github.io/illustrated-gpt2/>

Generative Pretrained Transformer(GPT)

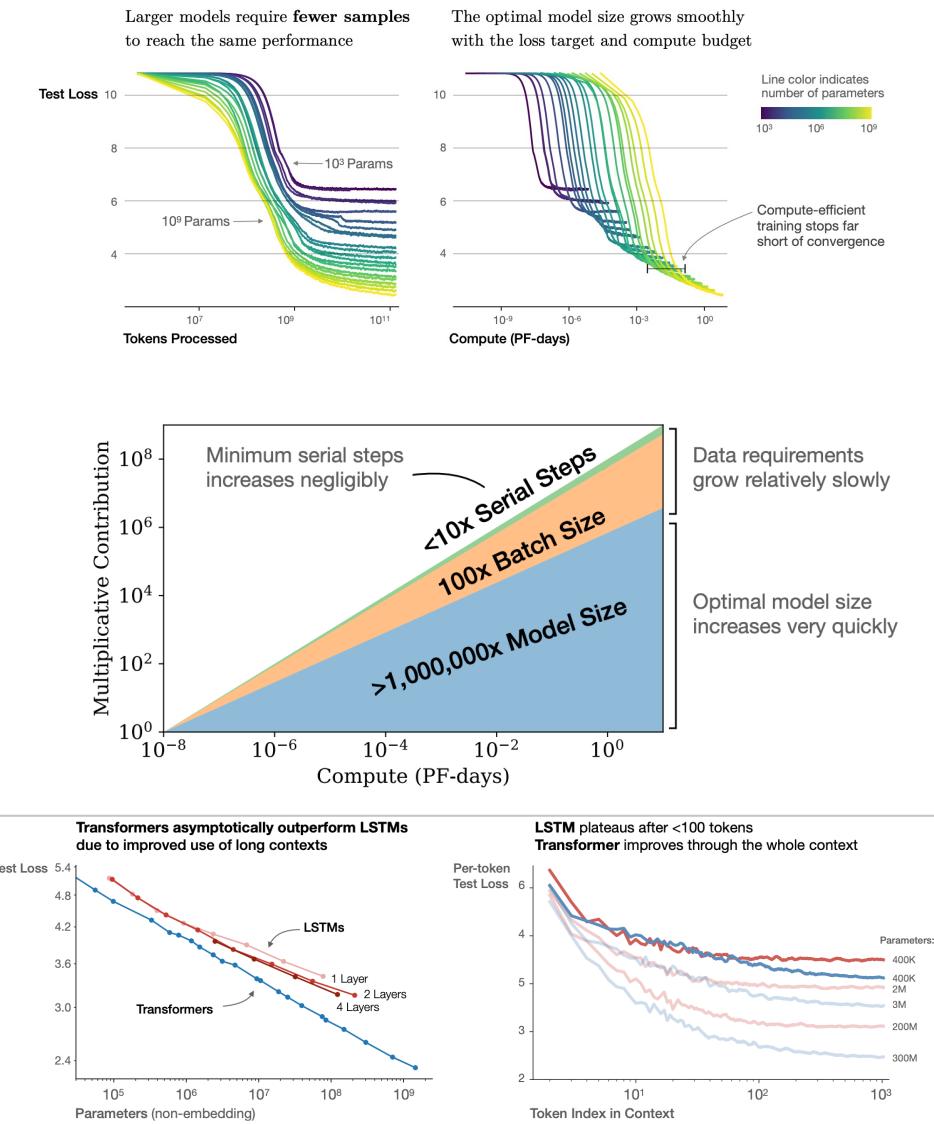
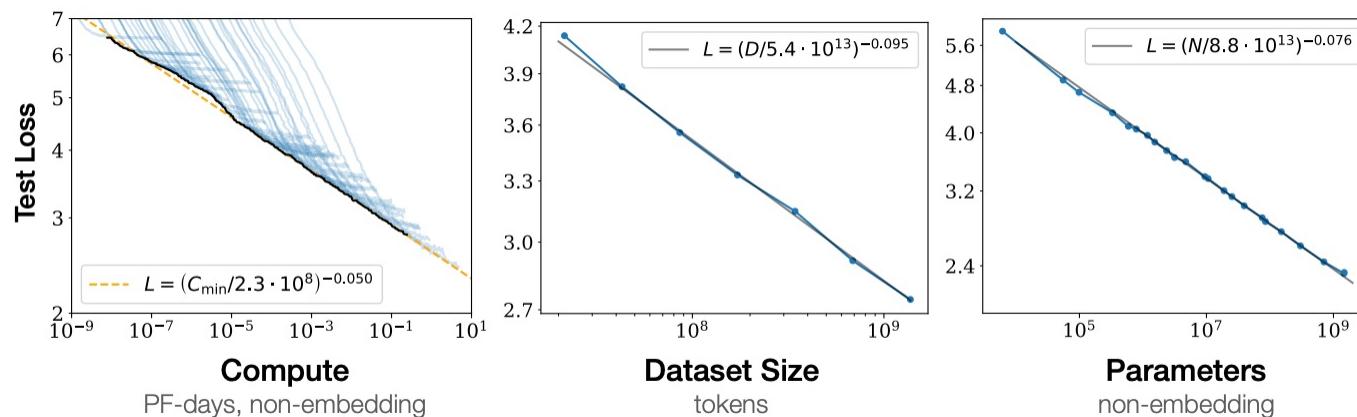
Отличие GPT от BERT – GPT авто-рекурсионная модель, BERT – учитывает весь контекст при обучении.



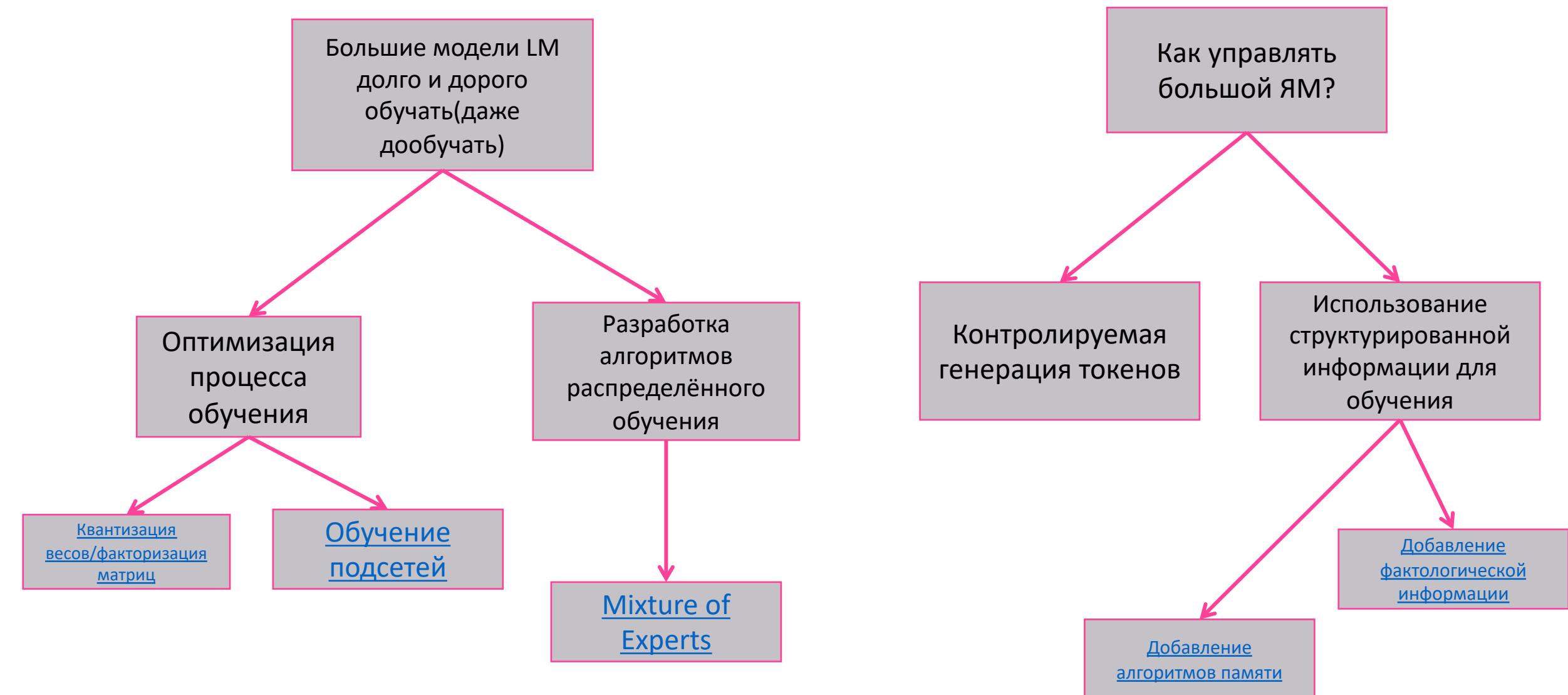
<https://jalammar.github.io/illustrated-gpt2/>

Законы масштабирования больших моделей

- 1) Чем больше параметров у модели(N), количество токенов в датасете(D) и время обучения (C) – тем лучше качество модели.
- 2) Существует степенная зависимость между качеством модели и каждым из параметров обучения.
- 3) Чем дальше мы увеличиваем N и D – тем дальше увеличивается качество. Но выходит на плато, если N и D не меняются.
- 4) Transfer learning на новом распределении данных улучшается пропорционально качеству на обучающей выборке.
- 5) Если есть ресурсы, лучше недообучить большую модель, чем обучить маленькую.



Проблемы обучения больших языковых моделей



Ложное языковое моделирование

Sentence
Kiev is a capital of

Run Model

Model Output

| Prediction | Score |
|--|--------|
| Kiev is a capital of the Russian Federation. | 57,8 % |
| Kiev is a capital of the Soviet Union, and ... | 40,3 % |
| Kiev is a capital of Donetsk, and has been ... | 1,3 % |
| Kiev is a capital of Donetsk, the region's ... | 0,7 % |
| Kiev is a capital of Baltic states and is situated ... | 0 % |

Sentence
iphone is produced by

Run Model

Model Output

| Prediction | Score |
|---|--------|
| iphone is produced by Huawei. The company has ... | 82,9 % |
| iphone is produced by its maker. | 15,7 % |
| iphone is produced by one of Nokia's largest ... | 1,3 % |
| iphone is produced by Facebook and other big tech ... | 0,2 % |
| iphone is produced by Topaz, a new ... | 0 % |

Sentence
Alexander Pushkin is a famous

Run Model

Model Output

| Prediction | Score |
|--|--------|
| Alexander Pushkin is a famous French novelist, who was ... | 96,7 % |
| Alexander Pushkin is a famous Marxist thinker, writers and ... | 2,2 % |
| Alexander Pushkin is a famous mathematician who authored many scientific ... | 0,6 % |
| Alexander Pushkin is a famous Turk in Turkey and a ... | 0,5 % |

Sentence
Dollar is currency in [MASK]

Run Model

Model Output

Mask 1

| Prediction | Score |
|------------------------------|--------|
| Dollar is currency in . | 55,8 % |
| Dollar is currency in India | 3,2 % |
| Dollar is currency in ; | 2,6 % |
| Dollar is currency in China | 2,3 % |
| Dollar is currency in Brazil | 1,5 % |

Sentence
Everest is the highest

Run Model

Model Output

| Prediction | Score |
|--|--------|
| Everest is the highest building in the city's ... | 99,1 % |
| Everest is the highest form of advertising. It ... | 0,5 % |
| Everest is the highest marker of cognitive impairment in ... | 0,3 % |
| Everest is the highest tension, high-speed ... | 0,1 % |
| Everest is the highest form of competitive Play in ... | 0 % |

SearchCreativeWork
eminem eminem show

PlayMusic
eminem 8 mile

SearchScreeningEvent
what song was played in final titanic scene

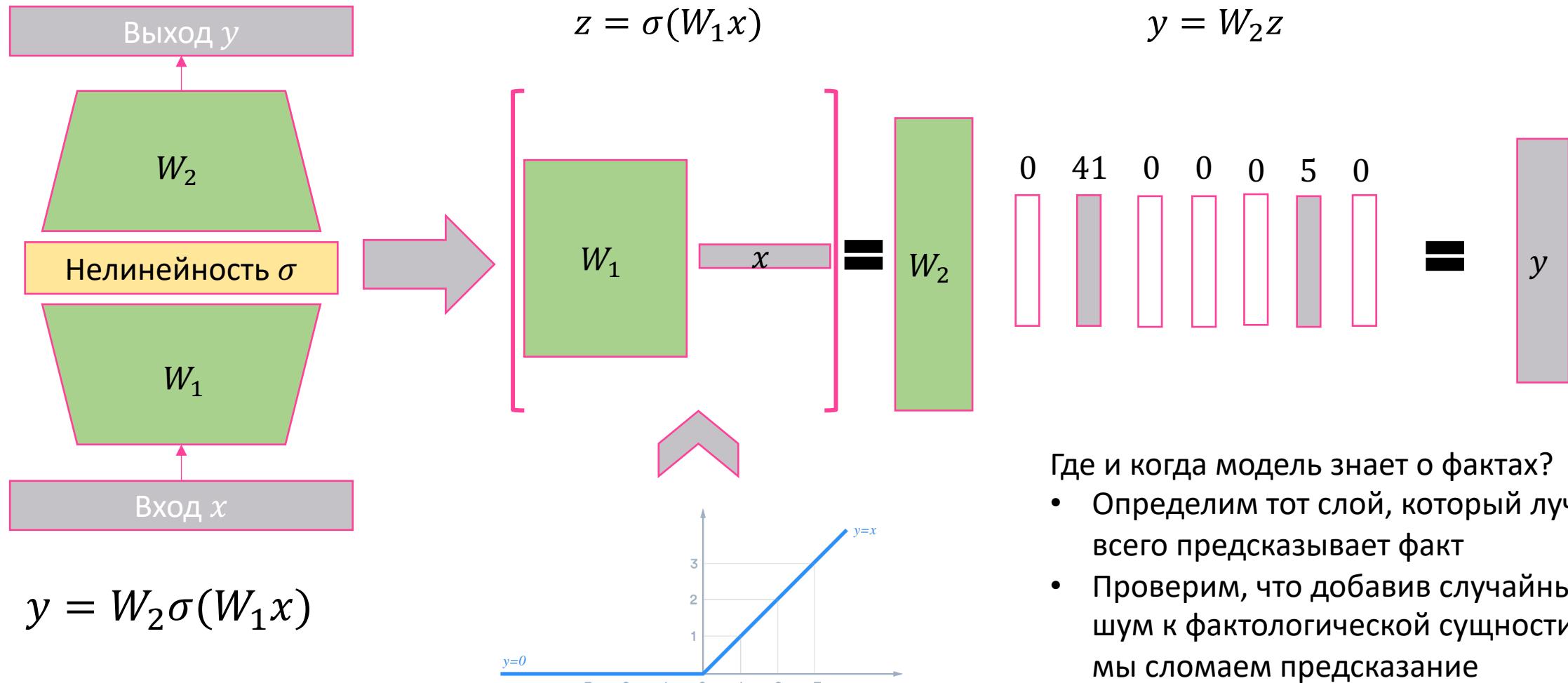
<https://demo.allennlp.org/next-token-lm>

<https://demo.allennlp.org/masked-lm>

<https://demo.deeppavlov.ai/#/en/intent>

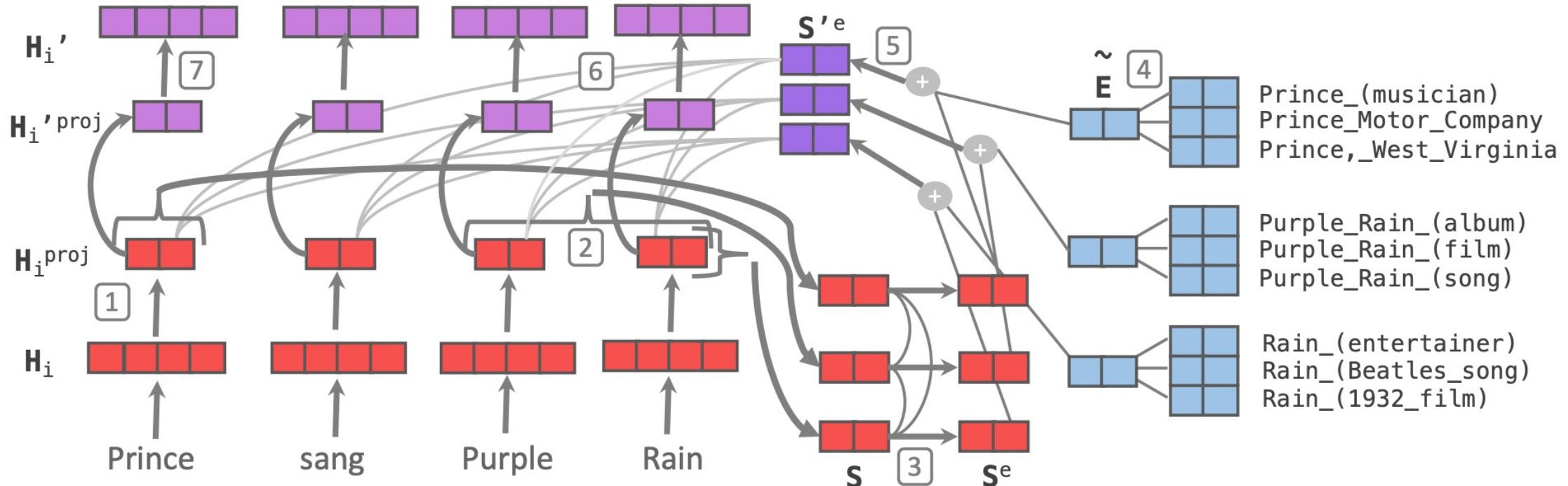
Как большие языковые модели хранят информацию?

Полносвязные слои трансформера - память в формате ключ-значение. ([Geva, 2021](#))



Добавление фактологической информации в большие LM

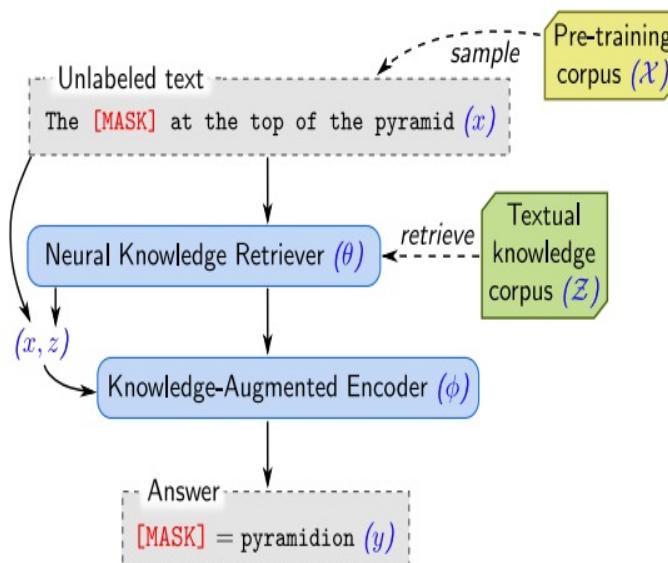
Knowledge attention and Recontextualization component



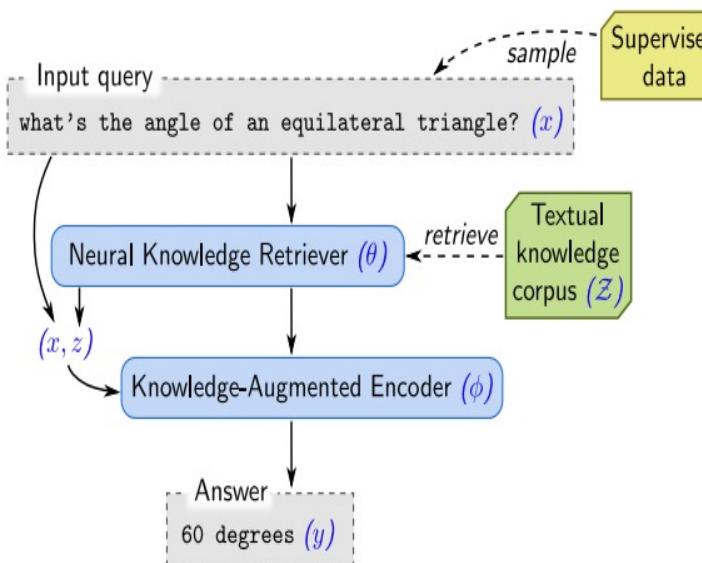
1. Эмбеддинги токенов BERT H_i проецируются в размерность меньшего пространства H_i^{proj}
2. Усредняем эмбеддинги по упоминаниям используя словарный подход по именованным сущностям – получаем S
3. Считаем attention между векторами упоминаний, получаем S^e
4. По словарю именованных сущностей считаем средний вектор для каждого упоминания \tilde{E} и складываем с S^e - получаем S'^e
5. Пересчитываем attention между исходными упоминаниями H_i^{proj} и с дополненными информацией S'^e
6. Проецируем обратно в исходную размерность BERT H_i'
7. Модуль добавляется в середину модели BERT

<https://arxiv.org/pdf/1909.04164.pdf>

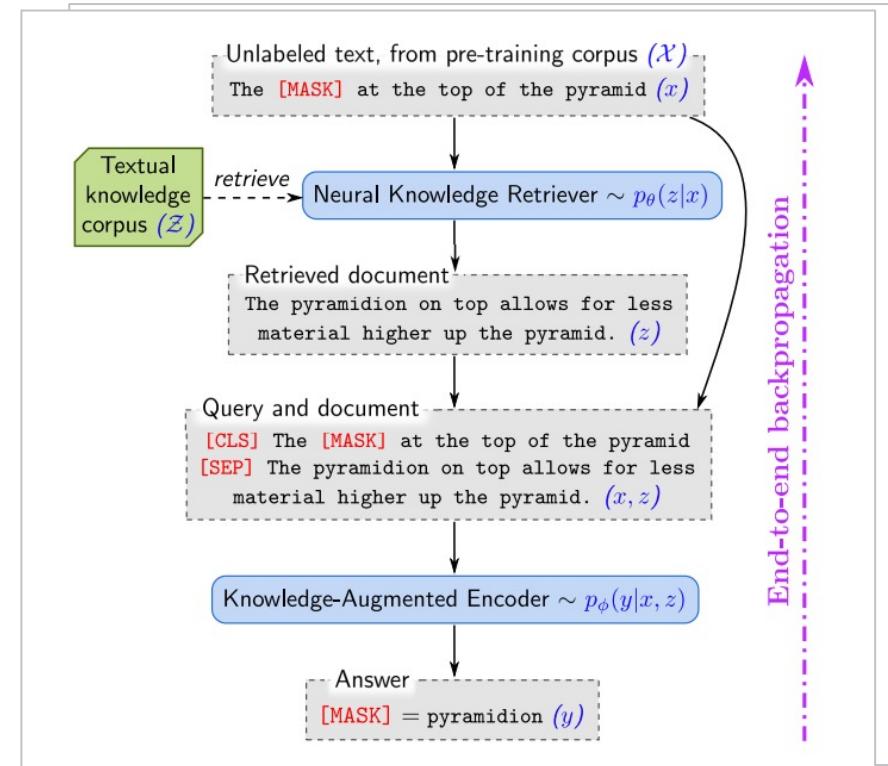
Retrieval-Augmented Language Model Pre-training



Pretraining - MLM



Finetuning - QA



$$p(y | x) = \sum_{z \in \mathcal{Z}} p(y | z, x) p(z | x).$$

Заключение

- 1) Появление модели трансформера и реализация первой большой языковой модели BERT на его основе – дало сильный толчок NLP.
- 2) Если ориентироваться на текущее качество моделей – чем больше модель, чем больше количество данных и чем дольше обучаться модель – тем лучше качество. Итого – главное развитие больших языковых моделей в решении задач распределённого обучения.
- 3) При обучении больших языковых моделей возникают новые проблемы – проблемы контроля генерации, проблемы истинности знания и проблемы предвзятости моделей.
- 4) Большие языковые модели улучшают свое качество за счет обучения на множестве задач, на больших объемах данных, при большем количестве параметров и при увеличении времени обучения.
- 5) В большие языковые модели можно добавлять фактологическую информацию на основании структурированных баз знаний и индексов.