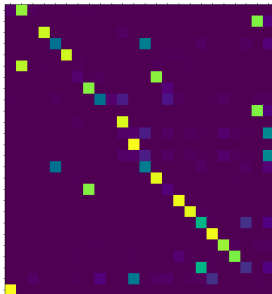


Механизм внимания и трансформер

Виктор Китов

v.v.kitov@yandex.ru

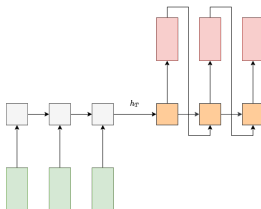


Содержание

1 Модель seq2seq с вниманием

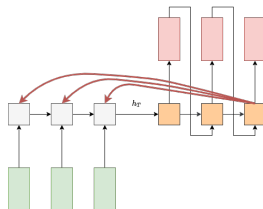
2 Трансформер

Модель seq2seq с вниманием



- Проблемы с кодировкой длинной последовательности вектором фикс. размерности.

Модель seq2seq с вниманием¹

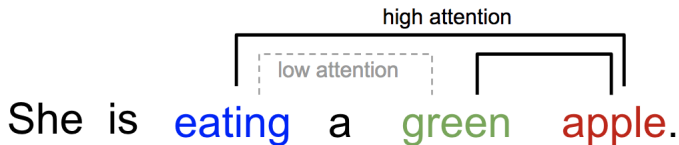


- seq2seq со вниманием (attention): в каждый момент состояние учитывает всю входную последовательность.
 - в переводе опр. слова: важно соответствующее слово входа и контекст, а не эмбеddинг предложения.

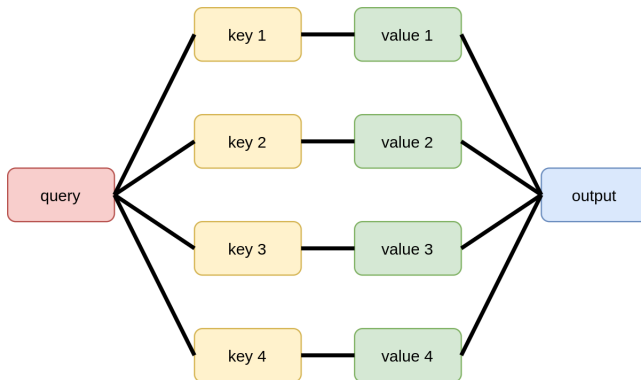
¹<https://arxiv.org/pdf/1409.0473.pdf>

Внимание между словами

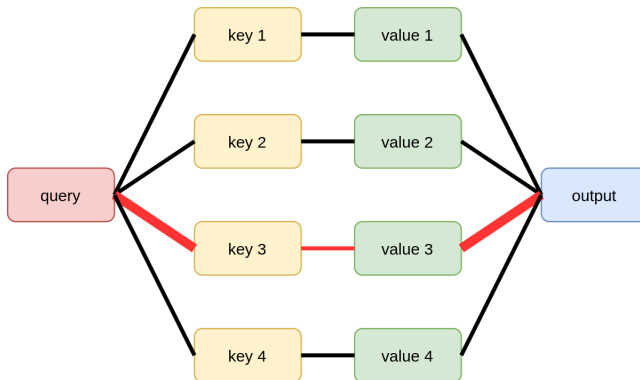
- Внимание важно для корректного перевода предложений
 - "она ела зеленое яблоко"
 - "она ела зеленые яблоки"
 - "она ела зеленую капусту"



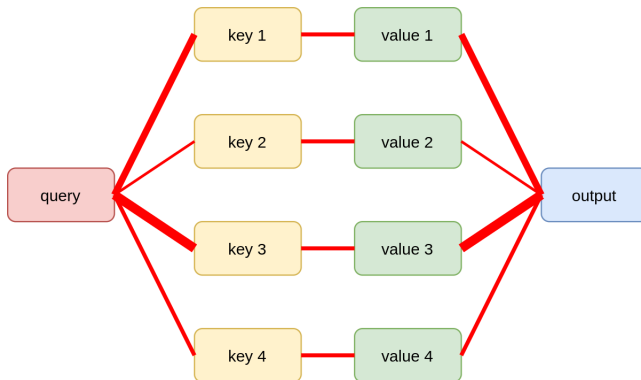
Внимание (attention)

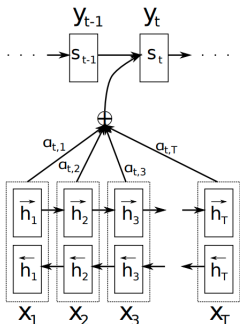


Внимание (attention)



Внимание (attention)



seq2seq с вниманием²

Кодировщик - двунапр. RNN

- состояния конкатенируются

- Степень соответствия:

$$e_{tj} = \text{score}(s_{t-1}, h_j)$$

- Веса учета состояний:

$$\begin{aligned} \alpha_{ti} &= \text{align}(s_t, h_s) \\ &= \exp(e_{ti}) / \sum_j \exp(e_{tj}) \end{aligned}$$

- Контекстный вектор:

$$c_t = \sum_j \alpha_{tj} h_j$$

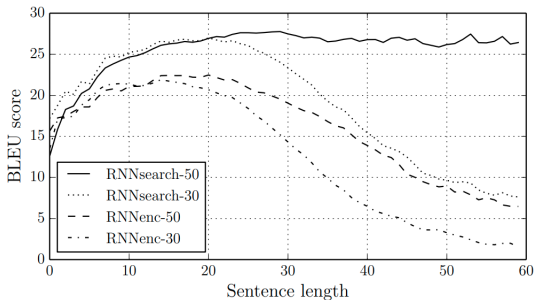
- Пересчёт состояний:

$$s_t = f(s_{t-1}, y_{t-1}, c_t)$$

²<https://arxiv.org/pdf/1409.0473.pdf>

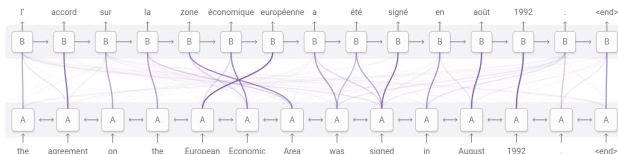
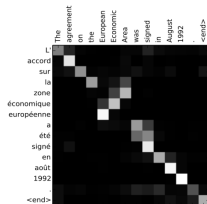
Результаты работы

- RNNenc-X - seq2seq
- RNNsearch-X - seq2seq со вниманием
- Обучение: машинный перевод предложений длины до X слов.



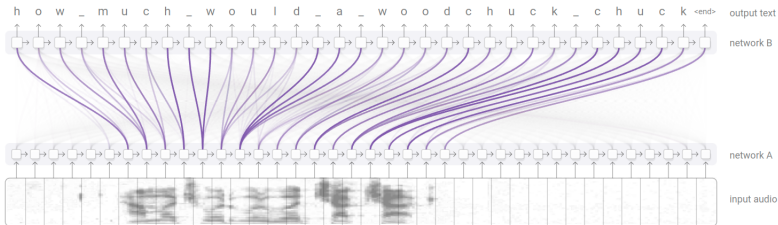
Визуализация весов α_{tj}

Визуализация весов α_{tj} (перевод английский->французский):

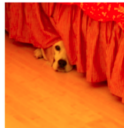


- Матрица близка к диагональной => сеть выучилась.
 - можно явно добавить диагонализующий регуляризатор

Визуализация внимания в других задачах³



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



A stop sign is on a road with a mountain in the background.



³Источник иллюстраций.

Варианты функции соответствия

Варианты соответствия $e_{tj} = \text{score}(s_{t-1}, h_j)$, $s_{t-1}, h_j \in \mathbb{R}^d$

basic dot-product	$s^T h$
scaled dot-product	$s^T h / \sqrt{d}$
content-based attention	$s^T h / (\ s\ \ h\)$
additive attention	$w^T \tanh(W_1 s + W_2 h)$
multiplicative attention	$s^T W h$

Локальное внимание⁴

- Обычное внимание усредняет по всем состояниям входа.
 - долго работает (например-перевод целого абзаца)

⁴<https://arxiv.org/pdf/1508.04025.pdf>

Локальное внимание⁴

- Обычное внимание усредняет по всем состояниям входа.
 - долго работает (например-перевод целого абзаца)
- Локальное внимание (local attention):
 - контекст c_t зависит только от $[p_t - D, p_t + D]$, веса $\alpha_t \in \mathbb{R}^{2D+1}$
 - $\alpha_{tj} = \text{align}(s_t, h_s) \exp\left(-\frac{(s-p_t)^2}{2\sigma^2}\right)$, $\sigma = \frac{D}{2}$.

⁴<https://arxiv.org/pdf/1508.04025.pdf>

Локальное внимание⁴

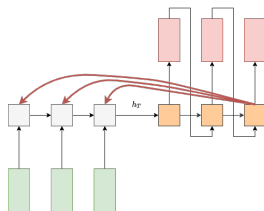
- Обычное внимание усредняет по всем состояниям входа.
 - долго работает (например-перевод целого абзаца)
- Локальное внимание (local attention):
 - контекст c_t зависит только от $[p_t - D, p_t + D]$, веса $\alpha_t \in \mathbb{R}^{2D+1}$
 - $\alpha_{tj} = \text{align}(s_t, h_s) \exp\left(-\frac{(s-p_t)^2}{2\sigma^2}\right)$, $\sigma = \frac{D}{2}$.
- Варианты генерации p_t :
 - $p_t = t$ (предполагаем входная и выходная посл-ти выровнены)
 - $p_t = S\sigma(v^T \tanh(Ws_t))$

⁴<https://arxiv.org/pdf/1508.04025.pdf>

Содержание

- 1 Модель seq2seq с вниманием
- 2 Трансформер

Модель seq2seq с вниманием

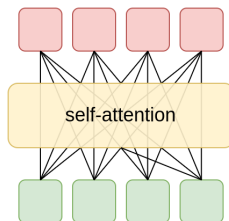


- seq2seq+attention:
 - не нужно представлять вектором входную последовательность
 - но всё еще нужно - информацию об уже сгенерированной последовательности (state)
- Хотим помнить всю входную и выходную последовательность (к текущему моменту).
- Используем трансформер (Attention Is All You Need⁵)

⁵<https://arxiv.org/pdf/1706.03762.pdf>

Модуль самовнимания (self-attention)

- Transformer - SOTA для машинного перевода и др. задач на последовательностях.
 - вход: эмбединги слов, выход: распределения слов.
- Проблема: слова в контексте приобретают другой смысл.
- Решение: модуль self-attention-преобразует s входов в s выходов
 - размерность входов и выходов: D
 - зависимость: каждый от каждого



Модуль самовнимания (self-attention)⁶

- $X_{s \times D}$ - s входов размерности D .
- Генерируем для каждого входа соответствующие
 - ключи (keys): $K_{s \times d} = X_{s \times D} W_{D \times d}^K$
 - значения (values): $V_{s \times \bar{d}} = X_{s \times D} W_{D \times \bar{d}}^V$
 - запросы (queries): $Q_{s \times d} = X_{s \times D} W_{D \times d}^Q$



⁶ Иллюстрации работы трансформера.

Модуль самовнимания (self-attention)

Выход для одного входа:

$$y_{1 \times \bar{d}} = \text{softmax} \left(\frac{1}{\sqrt{d}} q_{1 \times d} (K^T)_{d \times s} \right)_{1 \times s} V_{s \times \bar{d}}$$

В матричной форме:

$$Y_{s \times \bar{d}} = \text{softmax} \left(\frac{1}{\sqrt{d}} Q_{s \times d} (K^T)_{d \times s} \right)_{s \times s} V_{s \times \bar{d}}$$

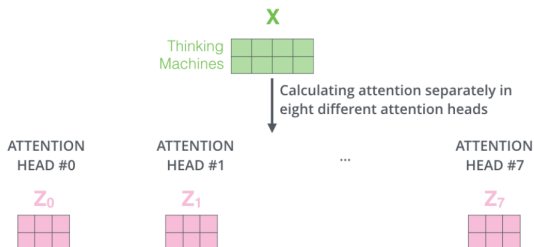
$$\text{softmax} \left(\frac{\begin{matrix} \text{Q} \\ \begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \square & \square & \square \end{array} \end{matrix} \times \begin{matrix} \text{K}^T \\ \begin{array}{|c|c|} \hline \square & \square \\ \hline \square & \square \end{array} \end{matrix}}{\sqrt{d_k}} \right) \begin{matrix} \text{V} \\ \begin{array}{|c|c|} \hline \square & \square \\ \hline \square & \square \end{array} \end{matrix} = \begin{matrix} \text{Z} \\ \begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \square & \square & \square \end{array} \end{matrix}$$

Модуль самовнимания (self-attention)

Получили выход одной головки модуля самовнимания:

$$\begin{aligned} \text{head} \left(X | W^K, W^V, W^Q \right)_{s \times \bar{d}} &= \text{softmax} \left(\frac{1}{\sqrt{d}} Q_{s \times d} \left(K^T \right)_{d \times s} \right)_{s \times s} V_{s \times \bar{d}} \\ &= \text{softmax} \left(\frac{1}{\sqrt{d}} \left(\underbrace{XW^Q}_Q \right) \left(\underbrace{XW^K}_K \right)^T \right) \underbrace{XW^V}_V \end{aligned}$$

Используется 8 головок (каждая - со своими W^Q, W^K, W^V).



Модуль самовнимания (self-attention)

Итоговый выход $\in \mathbb{R}^{D \times s}$ - конкатенация выходов + линейное преобразование:

$$\text{concat}_{s \times 8\bar{d}} \left[\text{head} \left(X | W_n^K, W_n^V, W_n^Q \right) \right]_{n=1}^8 W_{8\bar{d} \times D}^O$$

1) Concatenate all the attention heads

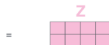


2) Multiply with a weight matrix W^O that was trained jointly with the model

x



3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN



Модуль самовнимания (self-attention)

1) This is our
input sentence*

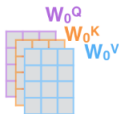
2) We embed
each word*

3) Split into 8 heads.
We multiply X or
 R with weight matrices

4) Calculate attention
using the resulting
 $Q/K/V$ matrices

5) Concatenate the resulting Z matrices,
then multiply with weight matrix W^O to
produce the output of the layer

Thinking
Machines



W^O



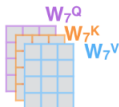
* In all encoders other than #0,
we don't need embedding.
We start directly with the output
of the encoder right below this one



...

...

...



Пример: на что смотрят блоки самовнимания

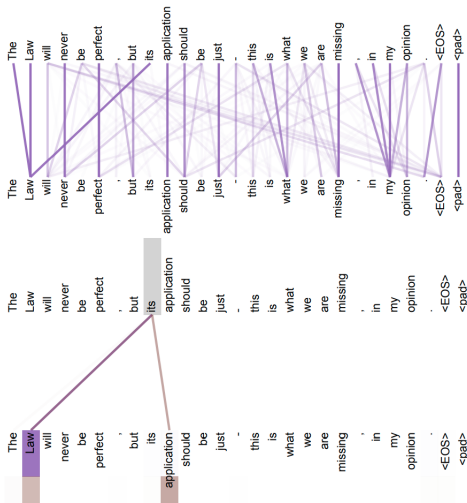


Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top: Full attentions for head 5. Bottom: Isolated attentions from just the word 'its' for attention heads 5 and 6. Note that the attentions are very sharp for this word.

Пример: на что смотрят блоки самовнимания

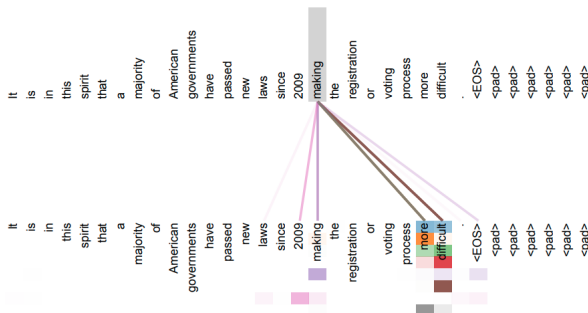
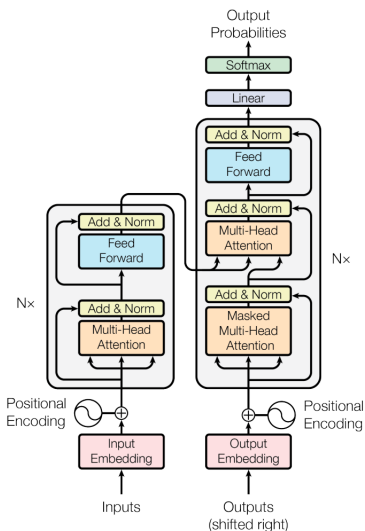


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb 'making', completing the phrase 'making...more difficult'. Attentions here shown only for the word 'making'. Different colors represent different heads. Best viewed in color.

- <pad> на входе - для выравнивания посл-тей минибатча.
- Внимание маскировалось, чтобы не смотреть на <pad>.

Трансформер: вся модель



- Feed Forward: сеть с одинаковыми весами, применяемая к каждому элементу

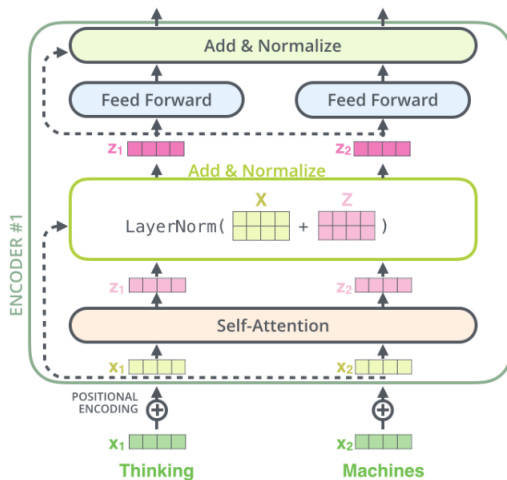
$$out = ReLU(xW_1 + b_1)W_2 + b_2$$

- Эмбединги позиций прибавляются к эмбедингам слов.
- Блоки повторяются $N = 6$ раз.
- Add&Norm:

$$LayerNorm(x + SubLayer(x))$$

- Кодировщик работает сразу.
- Декодировщик-в авторегрессионном режиме много раз до $\langle EOS \rangle$.

Визуализация первого блока кодировщика



Детали

- Нормализация слоя (LayerNorm) $[x_1, \dots, x_D]$:

$$x := \alpha \frac{x - \mu}{\sigma} + \beta, \quad \mu = \frac{1}{D} \sum_{i=1}^D x_i, \quad \sigma = \sqrt{\frac{1}{D} \sum_{i=1}^D (x_i - \mu)^2}$$

- α, β - выучиваемые параметры
- работает независимо для каждого объекта (здесь-токена)
- обучение и применения не различаются

Детали

- Нормализация слоя (LayerNorm) $[x_1, \dots, x_D]$:

$$x := \alpha \frac{x - \mu}{\sigma} + \beta, \quad \mu = \frac{1}{D} \sum_{i=1}^D x_i, \quad \sigma = \sqrt{\frac{1}{D} \sum_{i=1}^D (x_i - \mu)^2}$$

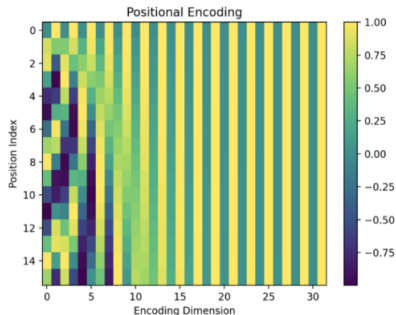
- α, β - выучиваемые параметры
 - работает независимо для каждого объекта (здесь-токена)
 - обучение и применения не различаются
- Positional embedding: кодирует расположение слов.
 - pos - позиция слова, i - индекс D -мерного эмбединга

$$PE_{(pos, 2i)} = \sin \left(pos / 10000^{2i/D} \right)$$

$$PE_{(pos, 2i+1)} = \cos \left(pos / 10000^{2i/D} \right)$$

Позиционное кодирование

Позиционный эмбеддинг $\in \mathbb{R}^{32}$:



Аналогия:

$$0 \rightarrow (0, 0, 0)$$

$$1 \rightarrow (1, 0, 0)$$

$$2 \rightarrow (0, 1, 0)$$

$$3 \rightarrow (1, 1, 0)$$

$$4 \rightarrow (0, 0, 1)$$

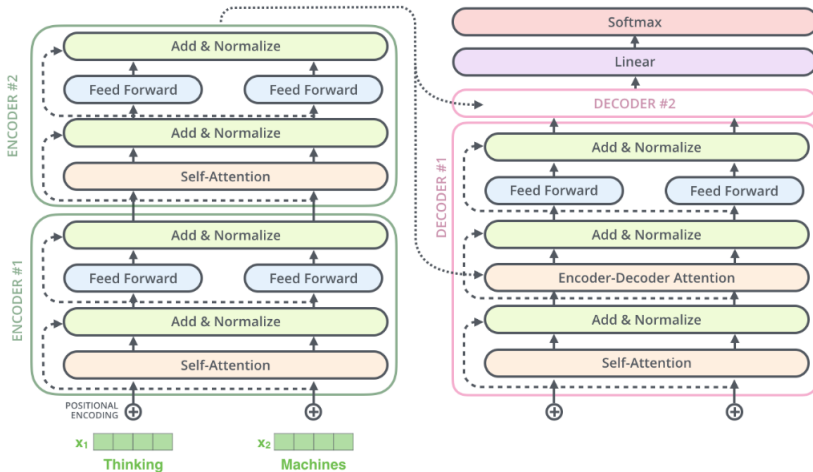
$$5 \rightarrow (1, 0, 1)$$

$$6 \rightarrow (0, 1, 1)$$

$$7 \rightarrow (1, 1, 1)$$

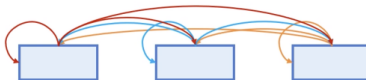
Полосы - \sin/\cos .

Схема двухуровневого трансформера



Виды внимания

Encoder Self-Attention:



Masked Decoder Self-Attention:



Encoder-Decoder Attention:



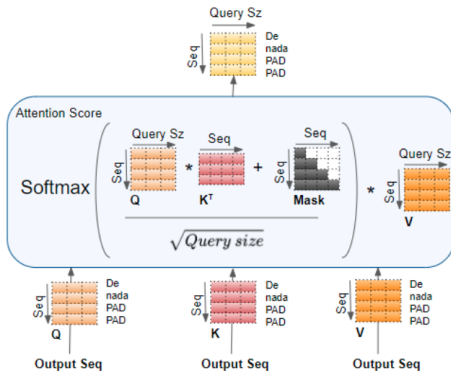
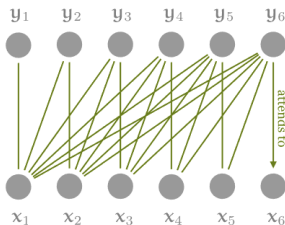
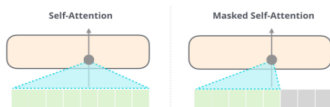
Виды внимания

Блоки multi-head attention:

- в кодировщике Q,K,V считаются:
 - в первом блоке: по эмбедингам слов+позиций
 - в последующих блоках: по выходам кодировщика предыдущего слоя
- в первом блоке декодировщика:
 - masked multi-head attention: по эмбедингам предсказанных слов+позиций (маскированным)
 - multi-head attention: Q-по выходам декодировщика, K,V - по финальным выходам кодировщика
- в последующих блоках декодировщика:
 - masked multi-head attention: по выходам декодировщика (маскированным)
 - multi-head attention: Q-по выходам декодировщика, K,V - по финальным выходам кодировщика

Маскирование⁷

- Masked multi-head attention декодировщика: элемент i не должен смотреть на $i + 1, i + 2, \dots$ (их еще нет).
- Прибавляем $-\infty$ к соотв. аргументам SoftMax:



⁷Источник иллюстрации.

Особенности настройки

- Использовался dropout:
 - в residual-блоках:

$$\text{LayerNorm}(x + \text{DropOut} \odot \text{SubLayer}(x))$$

- Также применялся dropout к начальным входам кодировщика и декодировщика.
- Сглаживались метки классов (слов).
- Пример кода на PyTorch с комментариями.

Заключение

- seq2seq RNN - представляют вектором
 - входную последовательность (исправляет seq2seq RNN+attention)
 - выходную посл-ть (к текущему моменту)
- Трансформер помнит
 - как всю входную последовательность
 - так и всю выходную посл-ть (к текущему моменту)
- Трансформер - SOTA на многих задачах обработки последовательностей
 - машинный перевод, языковое моделирование, ответы на вопросы, выделение именованных сущностей, суммаризация и т.д.