

# Attention Is (not) All You Need

Элбакян Мовсес

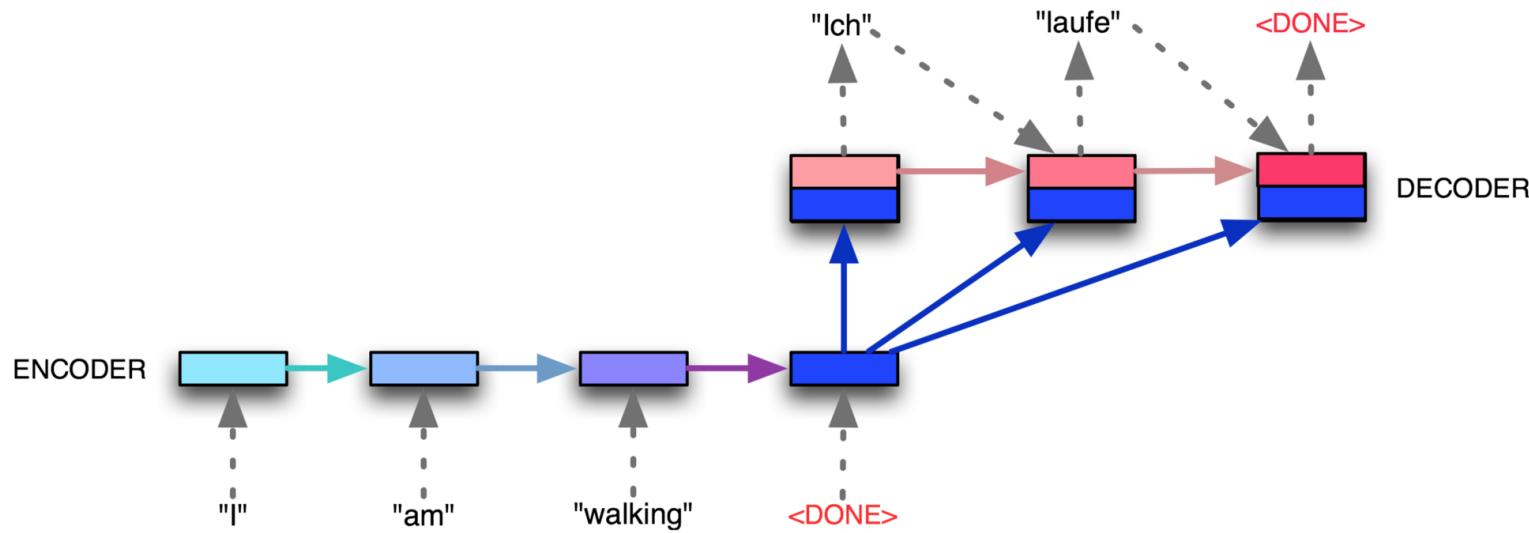
Национальный Исследовательский Университет Высшая Школа Экономики

2019

# План

- Краткая сводка по Seq2Seq
- Transformer
- Universal Transformer
- Пример использования Transformer'а во временных рядах
- Выводы

# Краткая сводка по Seq2Seq



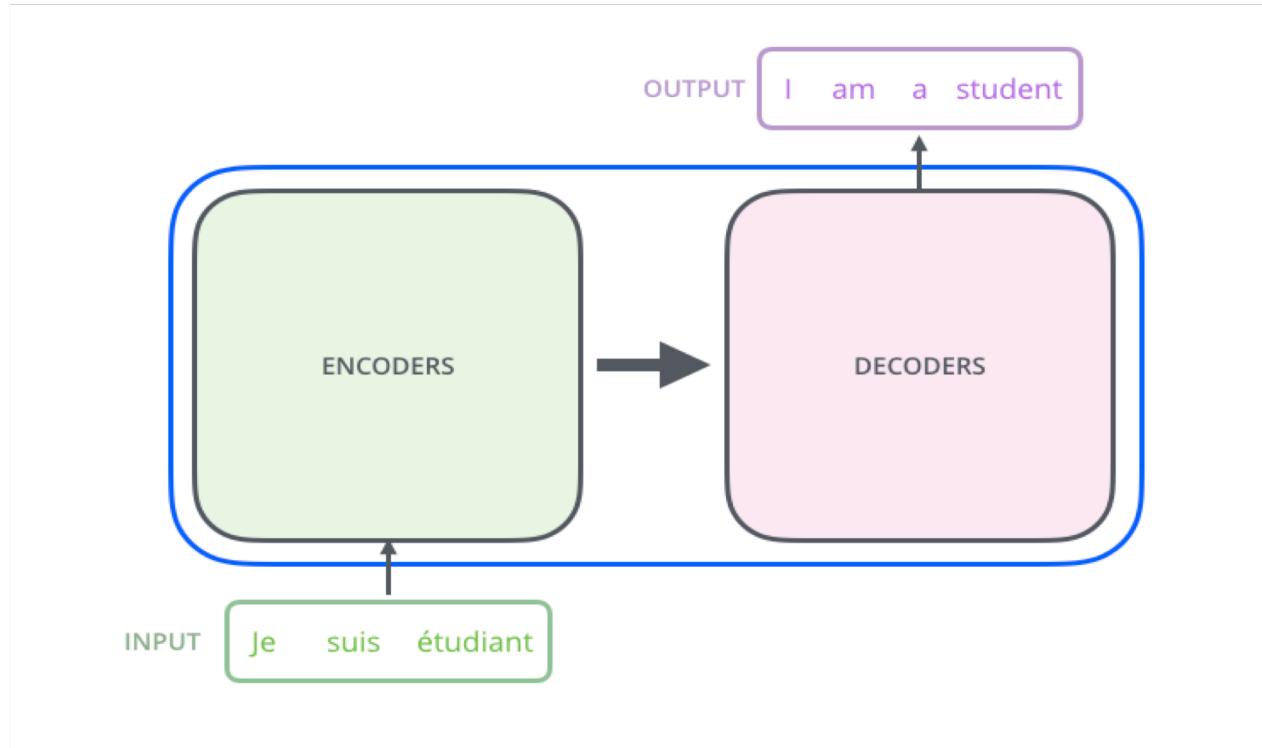
Основная структура:

Encoder(LSTM) + Decoder(LSTM) + Attention

Минусы:

RNN плохо учатся

# Transformer, 2017



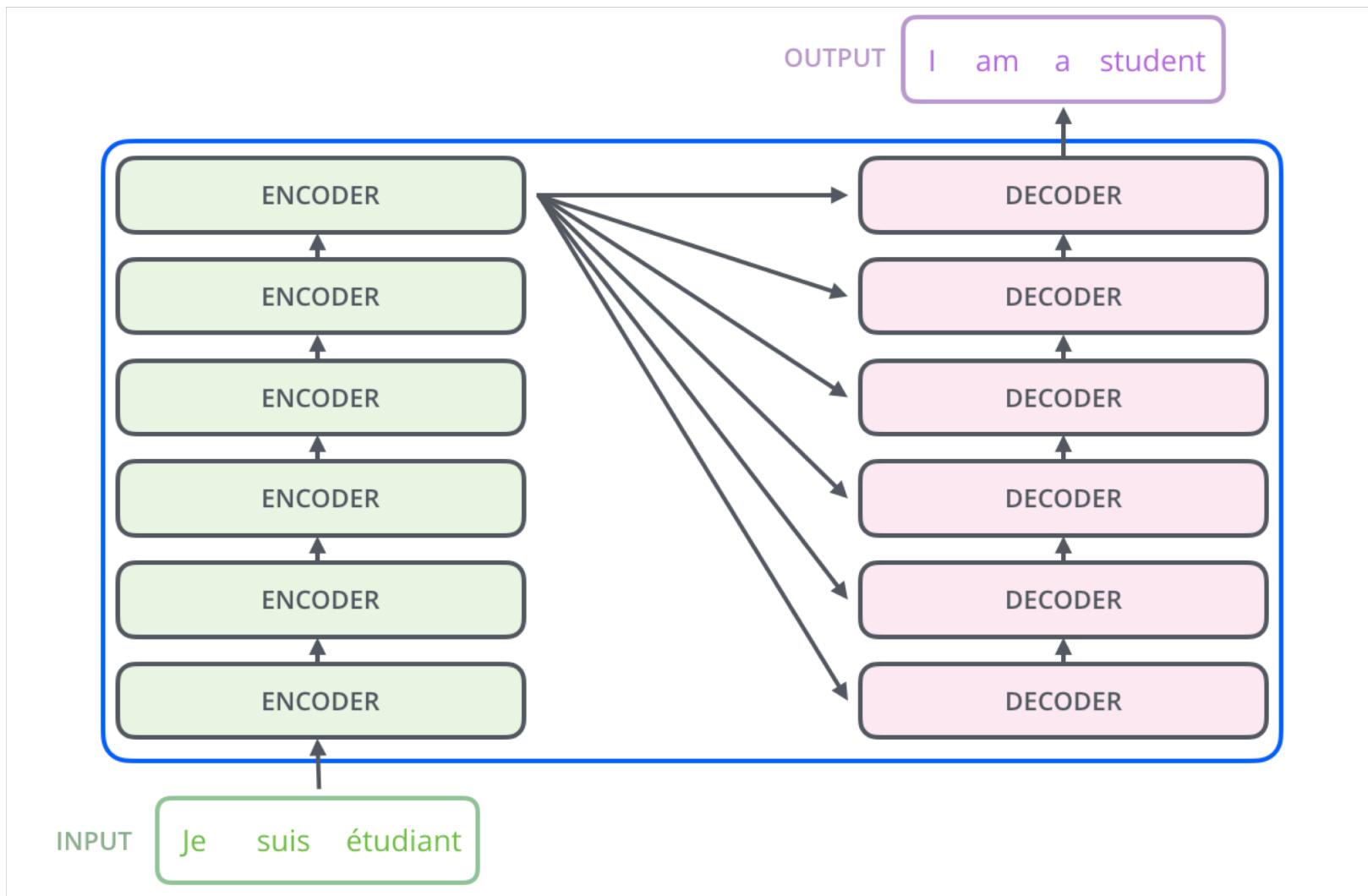
Основная структура:

Encoder(?) + Decoder(?) + Attention(?)

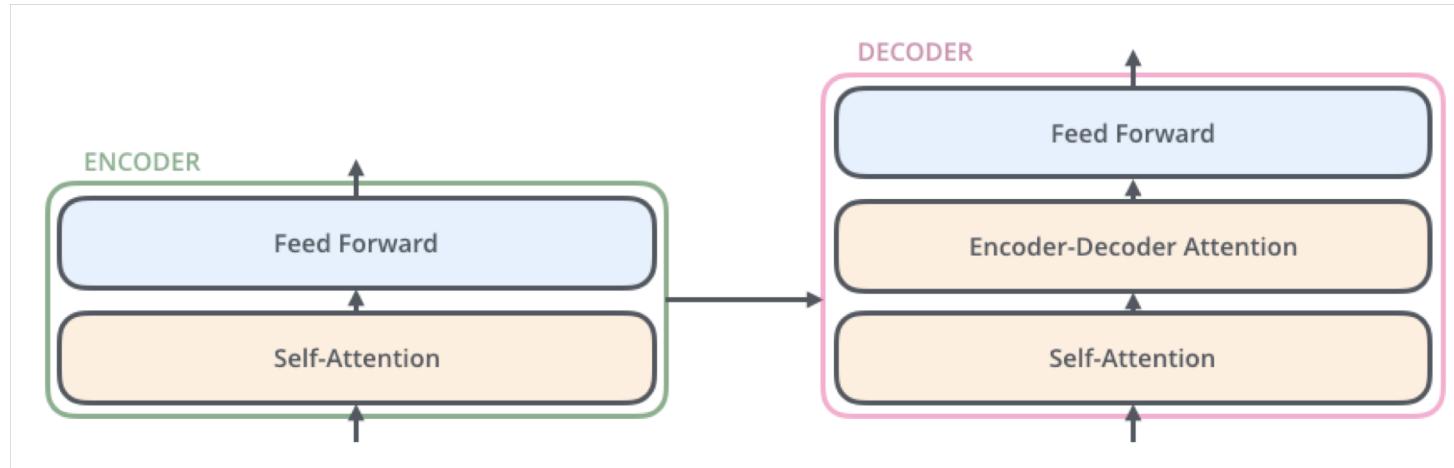
Минусы:

???

# Transformer, Vaswani et al, 2017



# Transformer, Vaswani et al, 2017



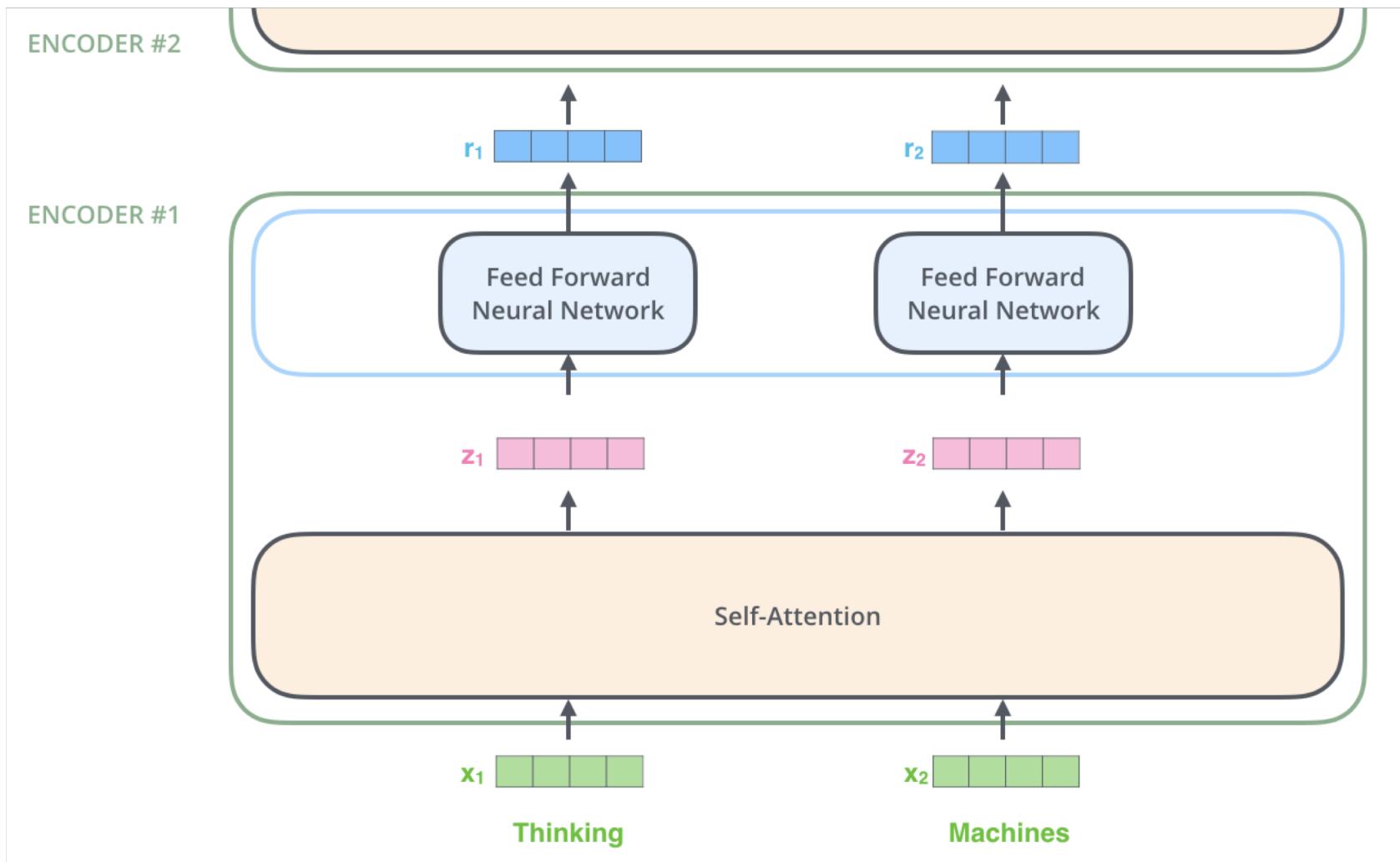
Основная структура:

Encoder(Conv/FCNN) + Decoder(Conv/FCNN) + Attention

Минусы:

???

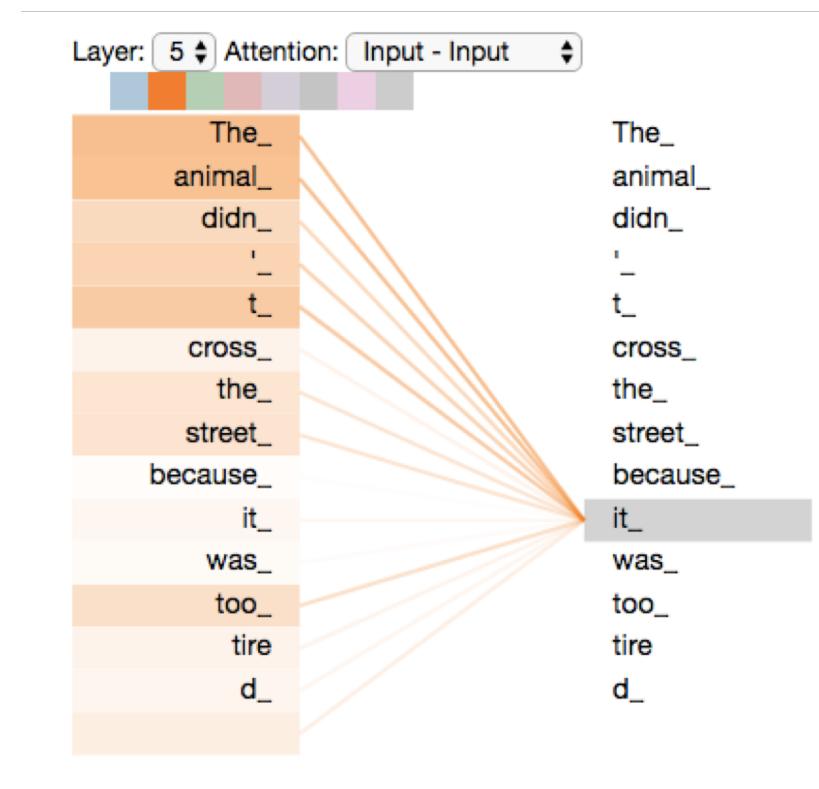
# Transformer. Encoder



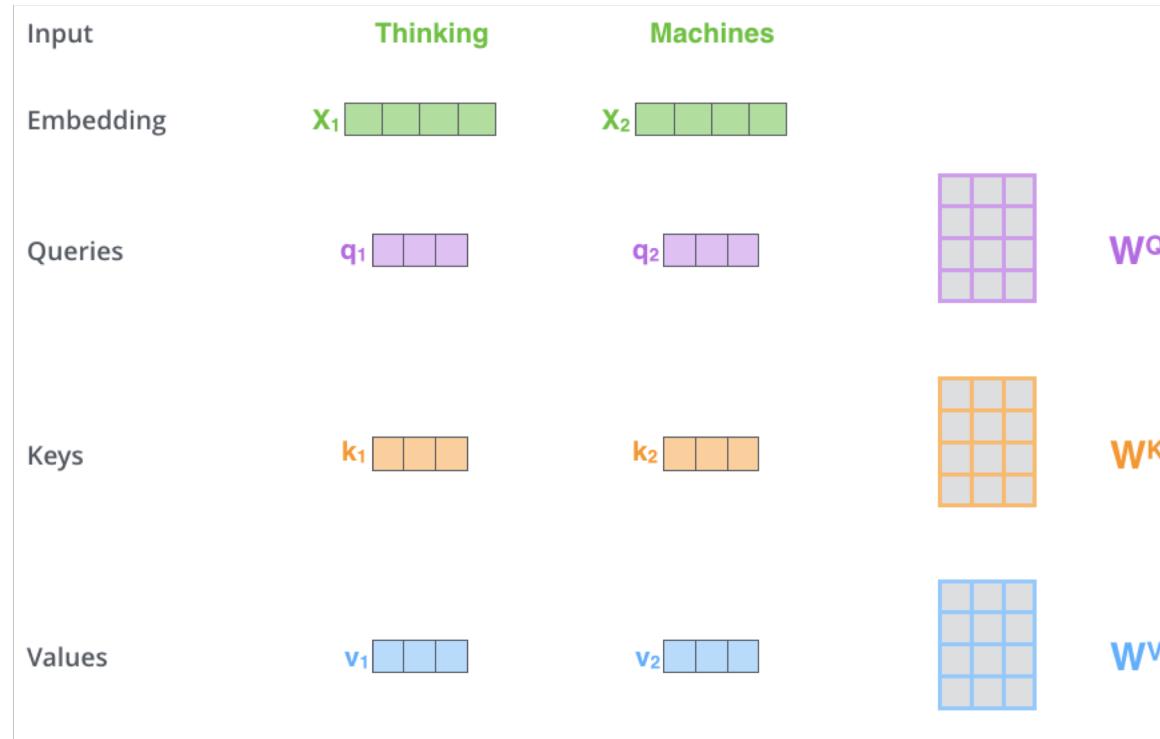
# Transformer. Self-Attention at High Level

"The animal didn't cross the street because it was too tired"  
(к чему относится "it")

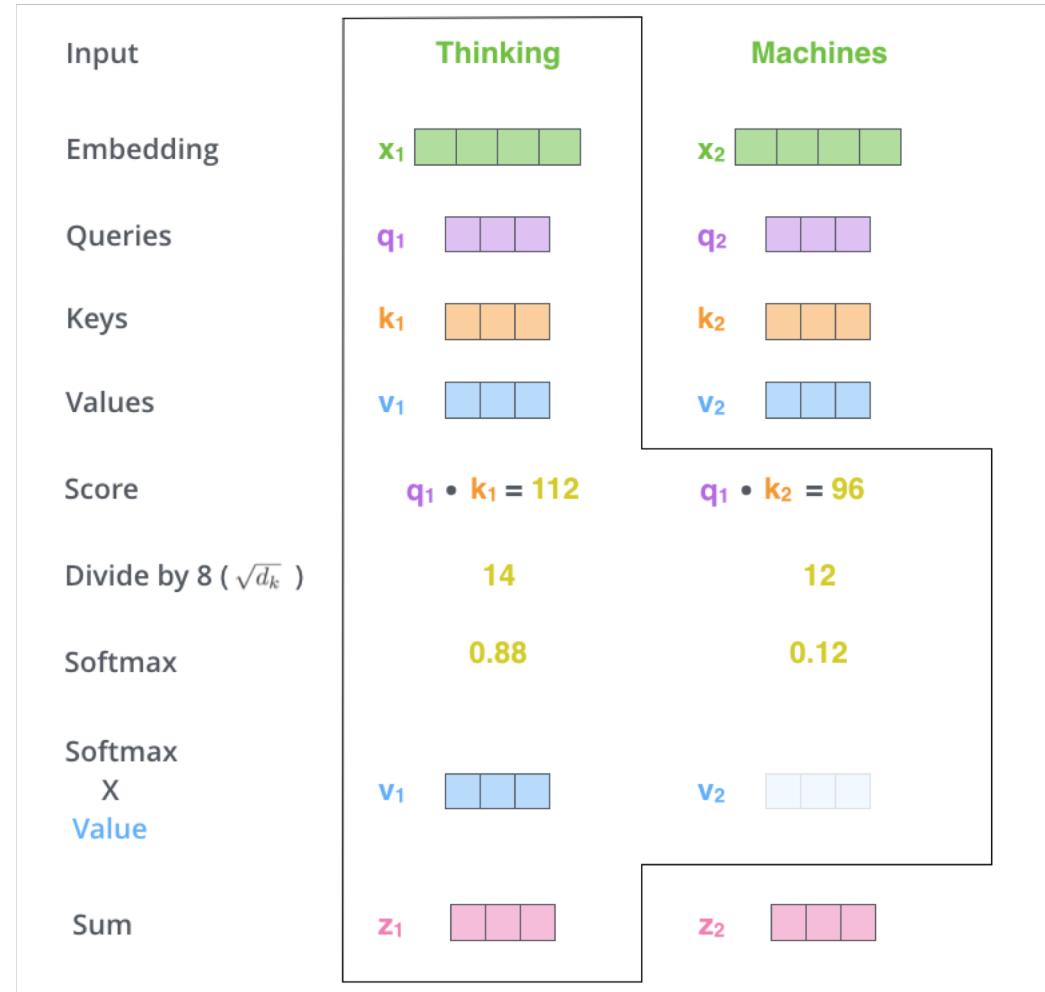
Self-Attention позволяет взглянуть на остальные части входа для того, чтобы найти «подсказки» для лучшего кодирования данного слова. Иначе говоря, Self-Attention помогает Трансформеру оценить релевантные слова к данному.



# Transformer. Self-Attention in details. Scaled-dot Attention



# Transformer. Self-Attention in details. Scaled-dot Attention



# Transformer. Self-Attention in details

## Scaled-dot Attention (Matrix representation)

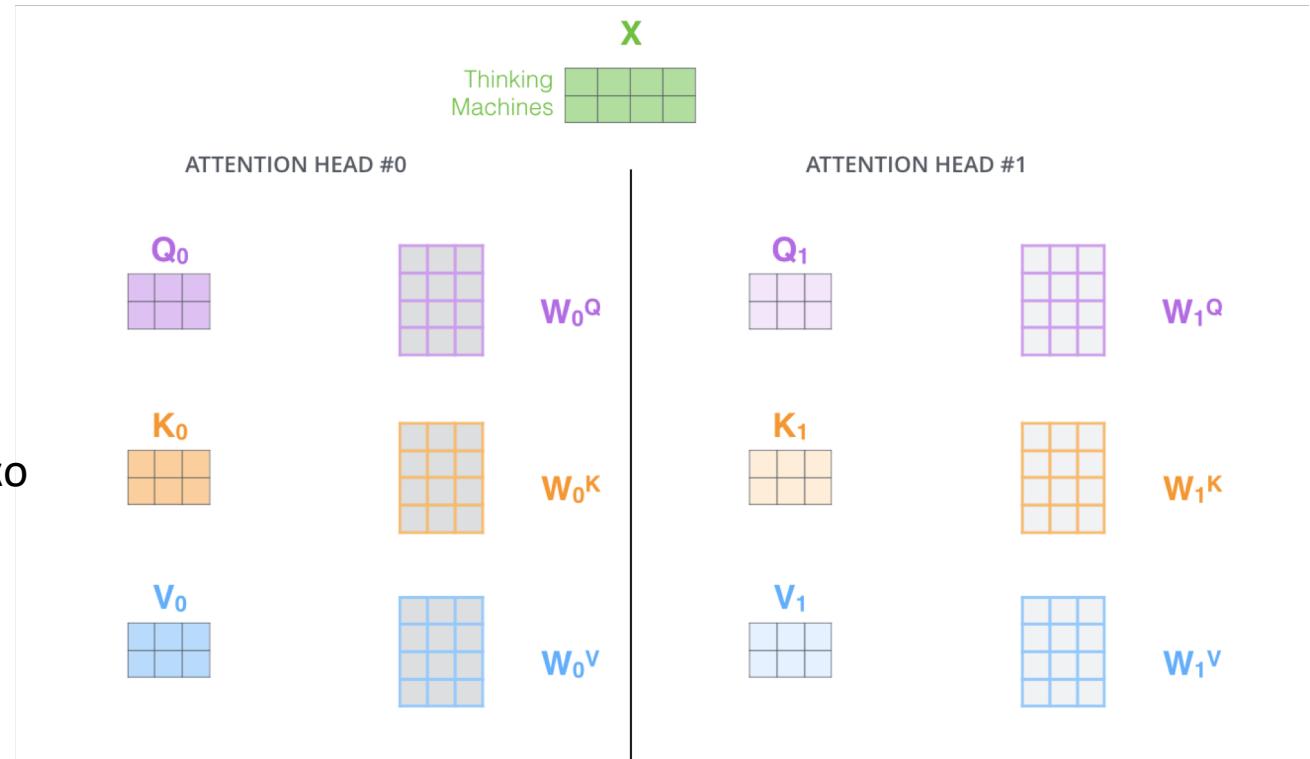
$$\begin{array}{ccc} \mathbf{x} & \mathbf{W^Q} & \mathbf{Q} \\ \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} \times \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} & = & \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} \\ \\ \mathbf{x} & \mathbf{W^K} & \mathbf{K} \\ \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} \times \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} & = & \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} \\ \\ \mathbf{x} & \mathbf{W^V} & \mathbf{V} \\ \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} \times \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} & = & \begin{matrix} \text{---} \\ \text{---} \\ \text{---} \end{matrix} \end{array}$$

$$\begin{aligned} & \text{softmax} \left( \frac{\mathbf{Q} \times \mathbf{K}^T}{\sqrt{d_k}} \right) \mathbf{V} \\ & = \mathbf{Z} \end{aligned}$$

# Transformer. Self-Attention in details

## Multi-head Attention

Несколько голов(heads) в Multi-head Attention используются для того, чтобы модель могла сфокусироваться на нескольких различных позициях, а также этот механизм предоставляется несколько подпространств векторных представлений. Матрицы  $Q_i, Ki, Vi$  инициализируются случайно.



$$X \in \mathbb{R}^{l \times 512}$$
$$Q_i, Ki, Vi \in \mathbb{R}^{l \times 64}$$

8 heads

# Transformer. Self-Attention in details

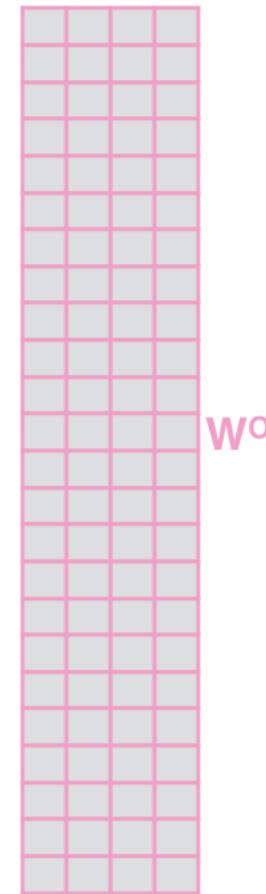
## Multi-head Attention

1) Concatenate all the attention heads



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

$\times$

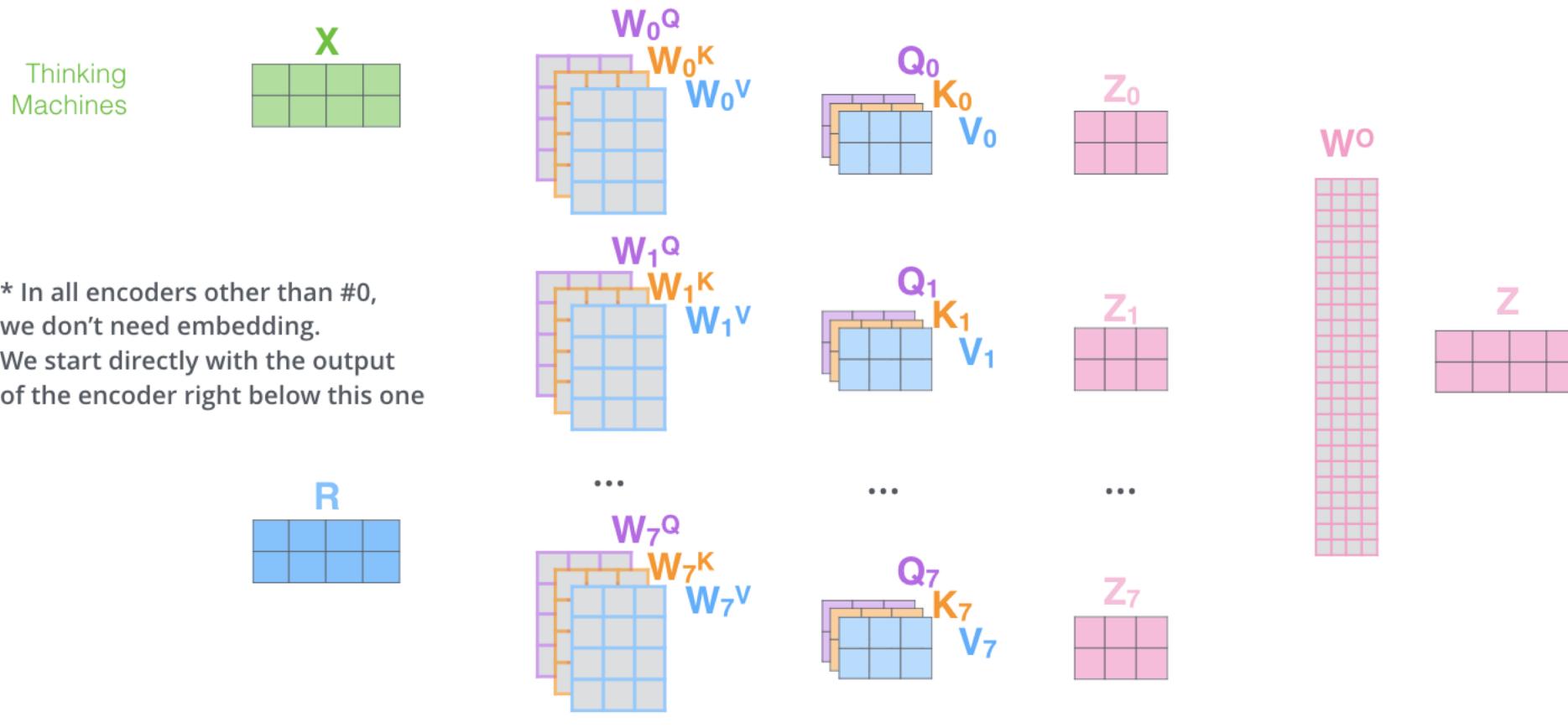


3) The result would be the  $Z$  matrix that captures information from all the attention heads. We can send this forward to the FFNN

$$= \begin{matrix} Z \\ \begin{matrix} \text{---} & \text{---} & \text{---} & \text{---} \end{matrix} \end{matrix}$$

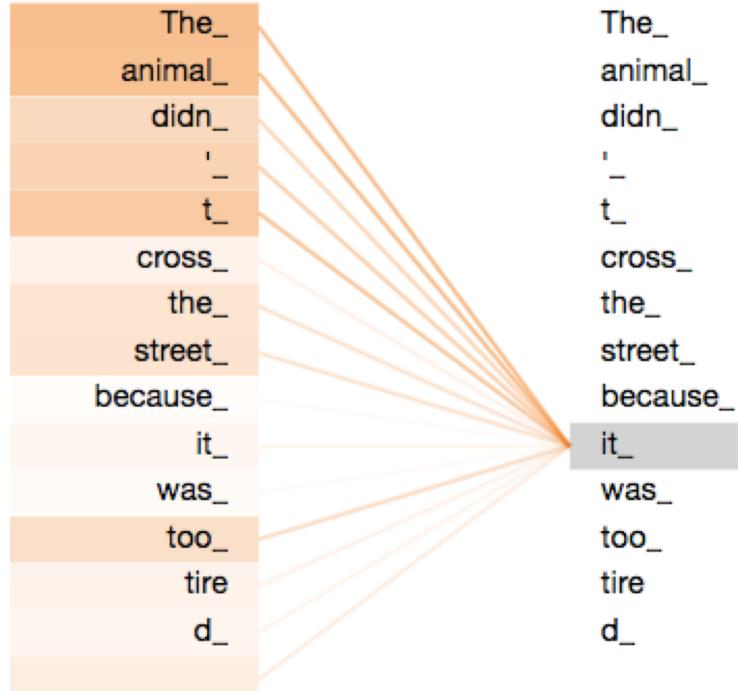
# Transformer. Self-Attention summary

- 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



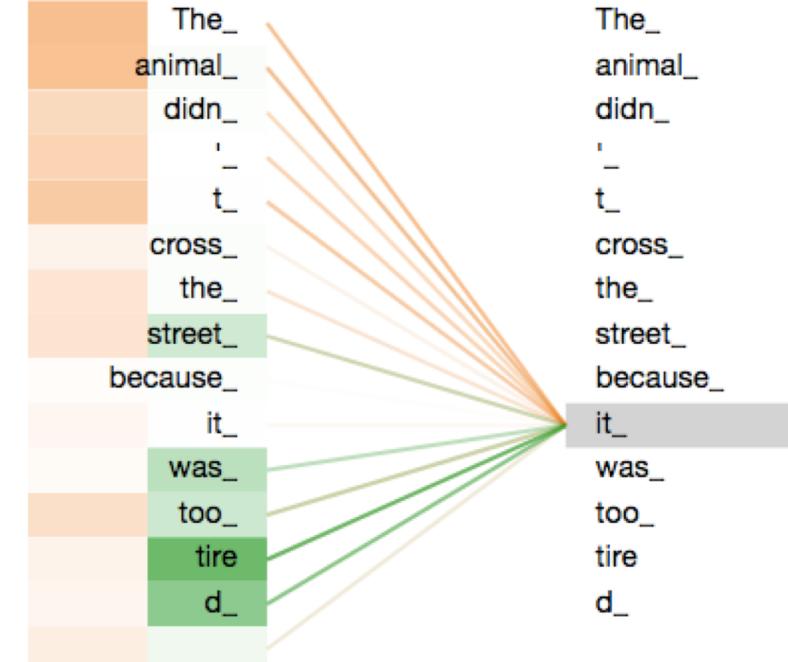
# Transformer. Self-Attention heads interpretation

Layer: 5 Attention: Input - Input



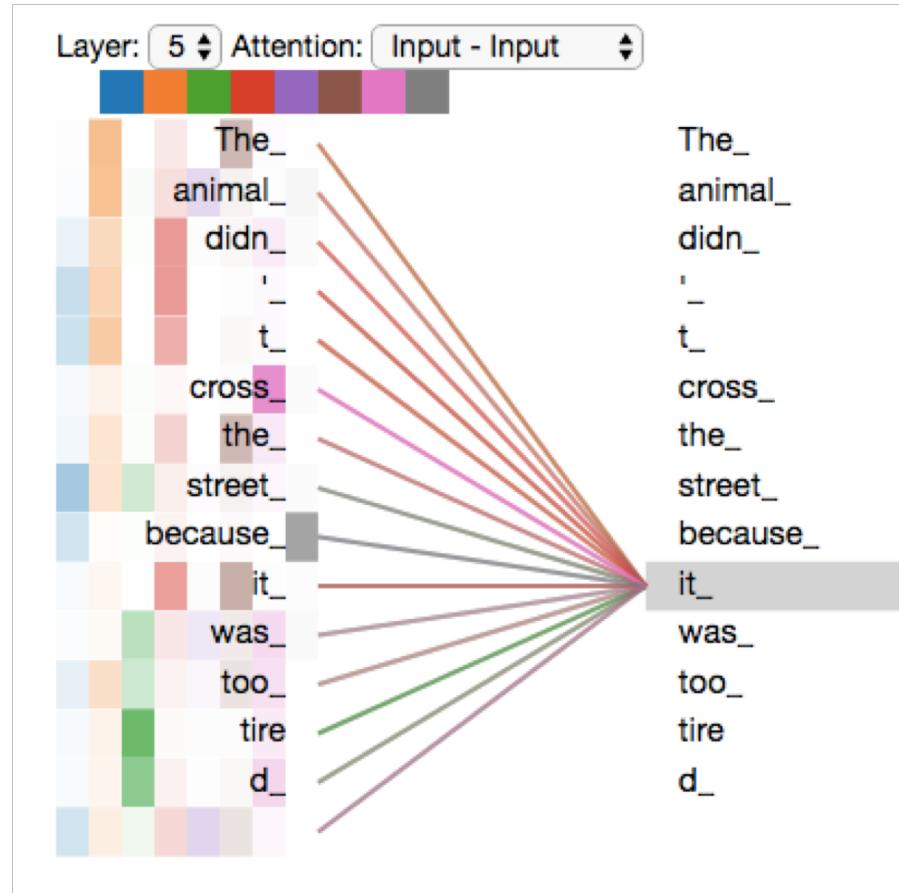
Самое релевантное: «The animal»

Layer: 5 Attention: Input - Input



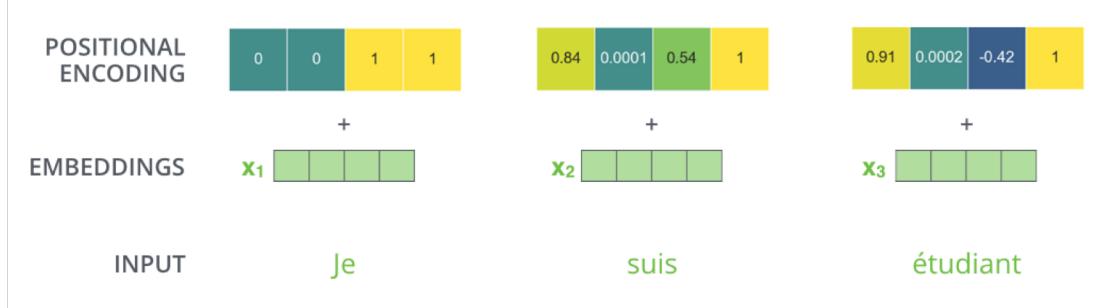
Самое релевантное: «street»

# Transformer. Self-Attention heads interpretation



Самое релевантное: ???

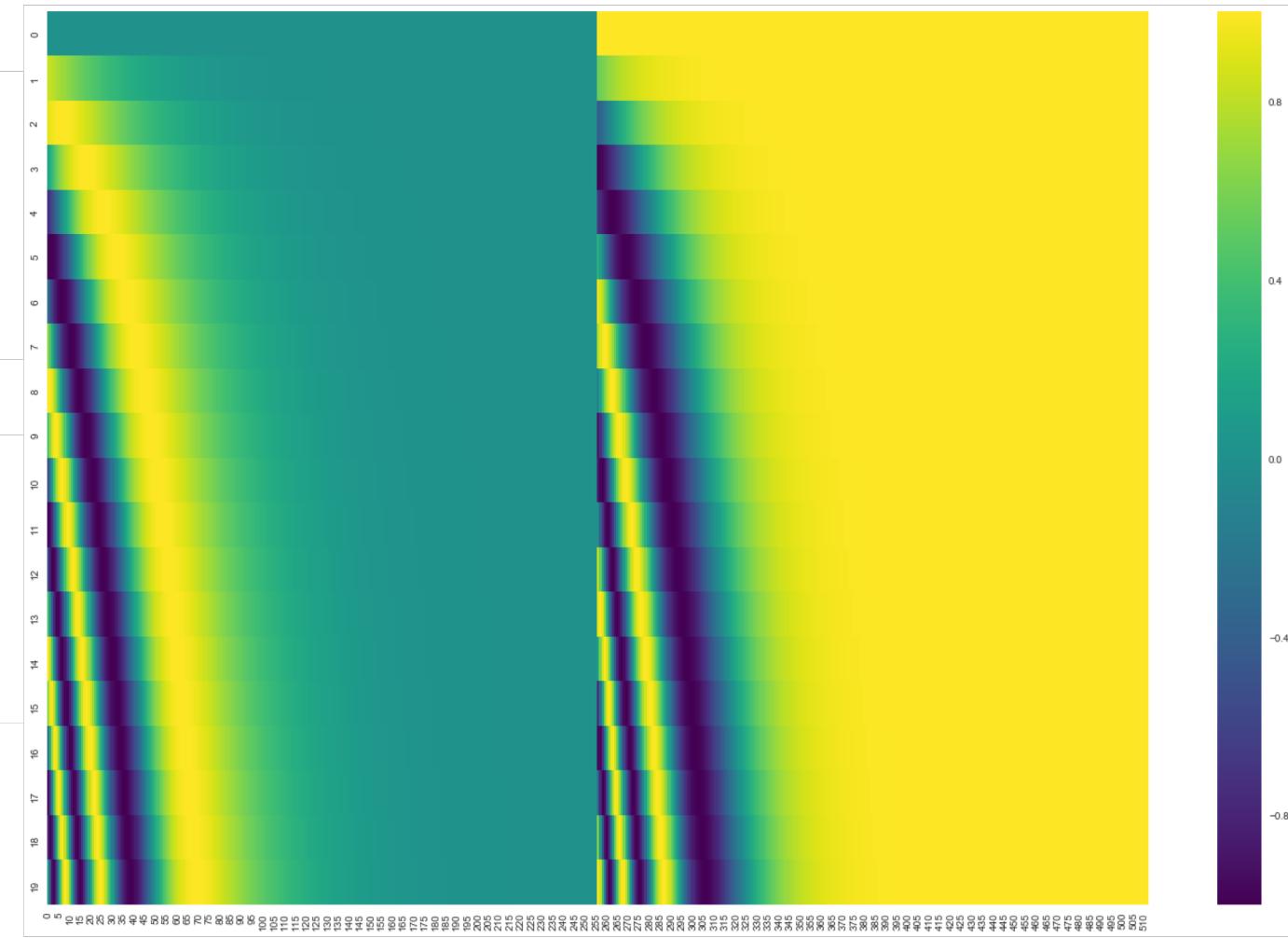
# Transformer. Input encodings



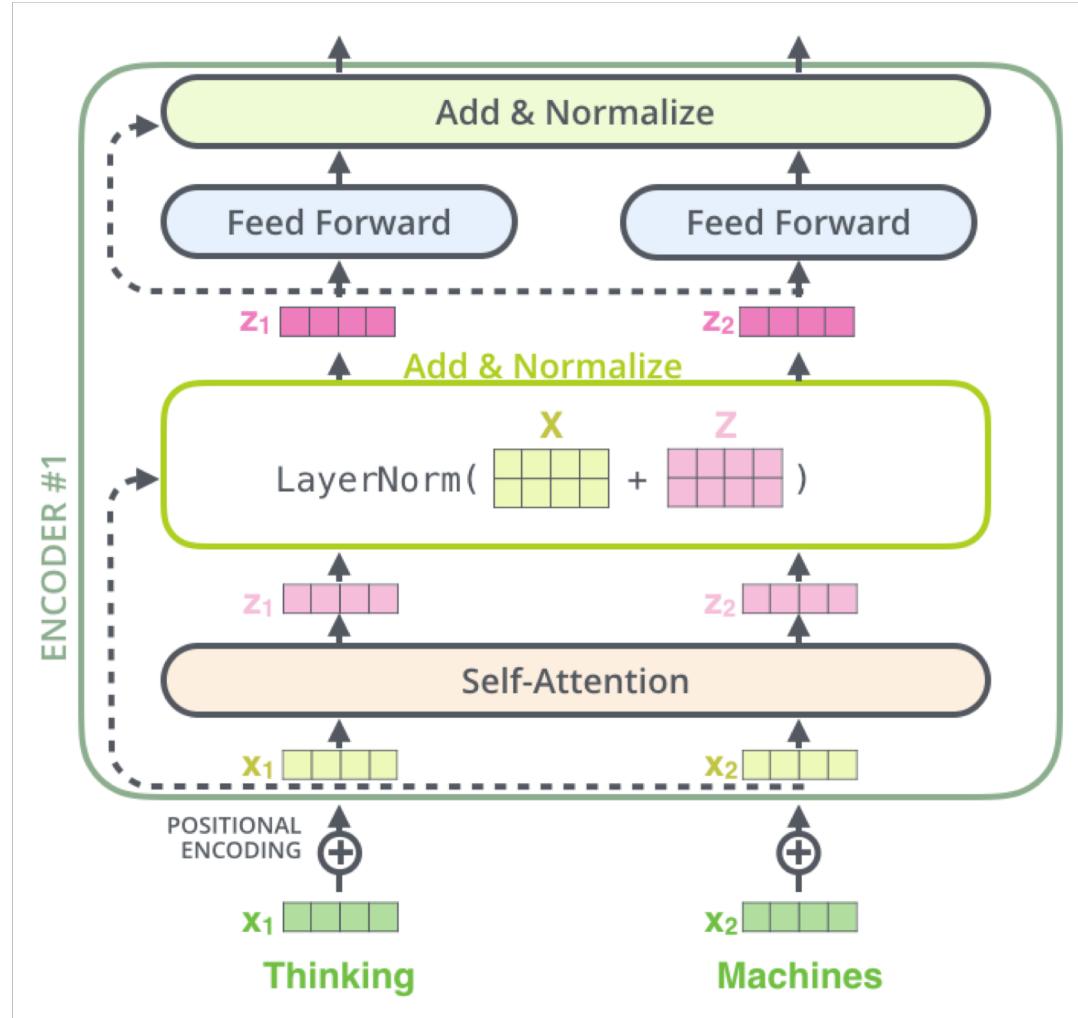
$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$

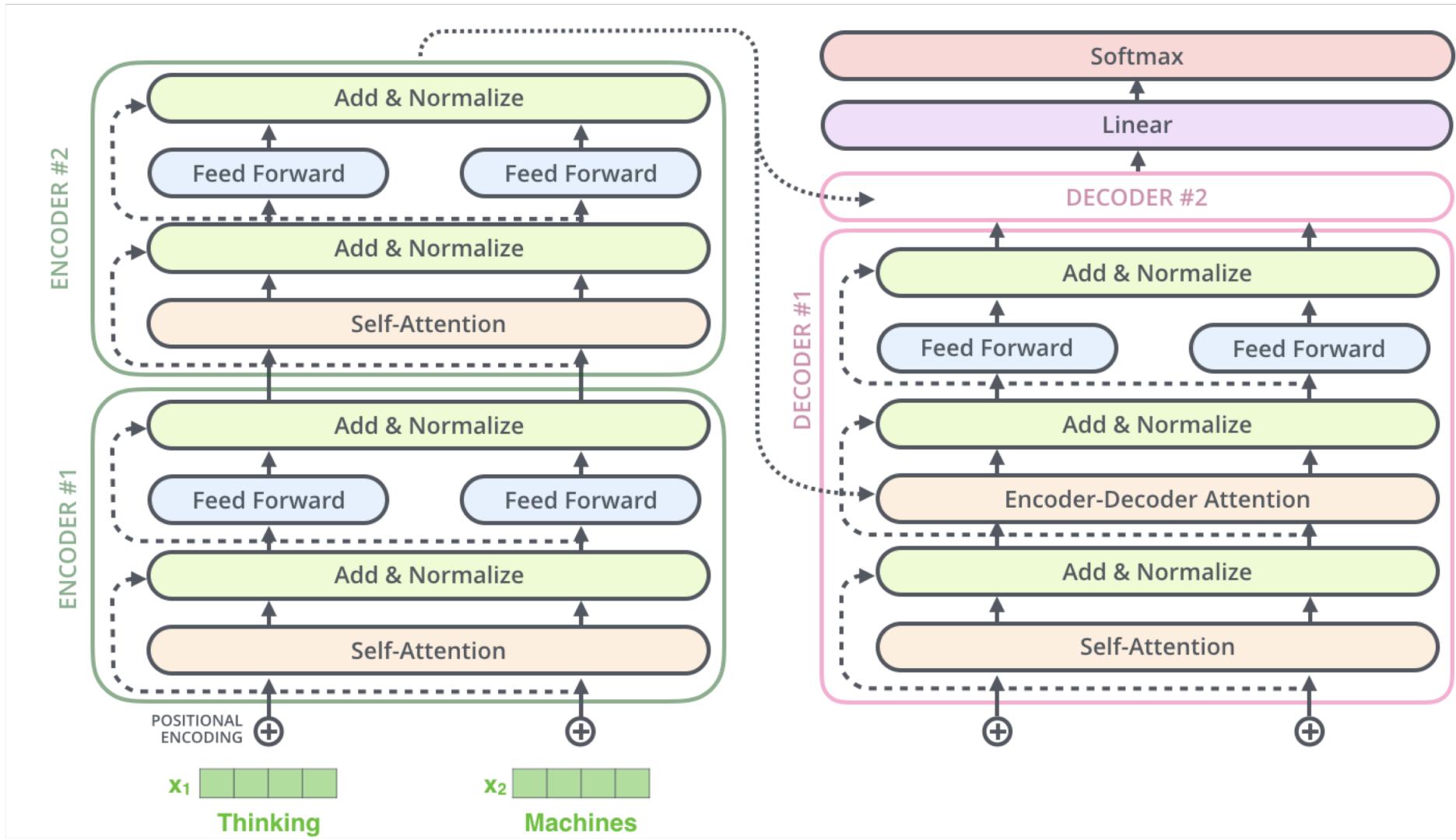
pos – номер слова в предложении  
i – i-ая размерность эмбеддинга



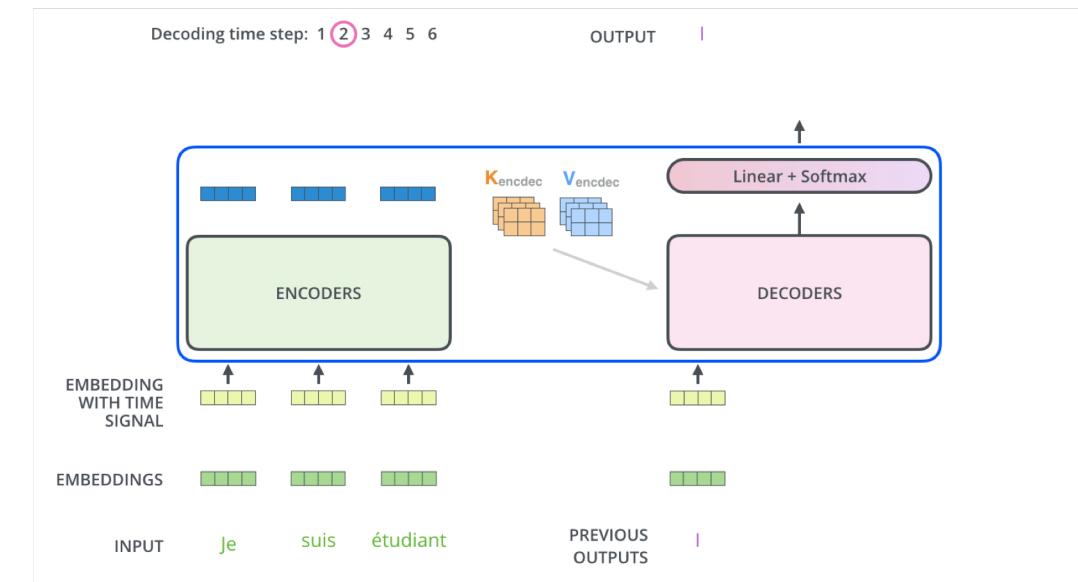
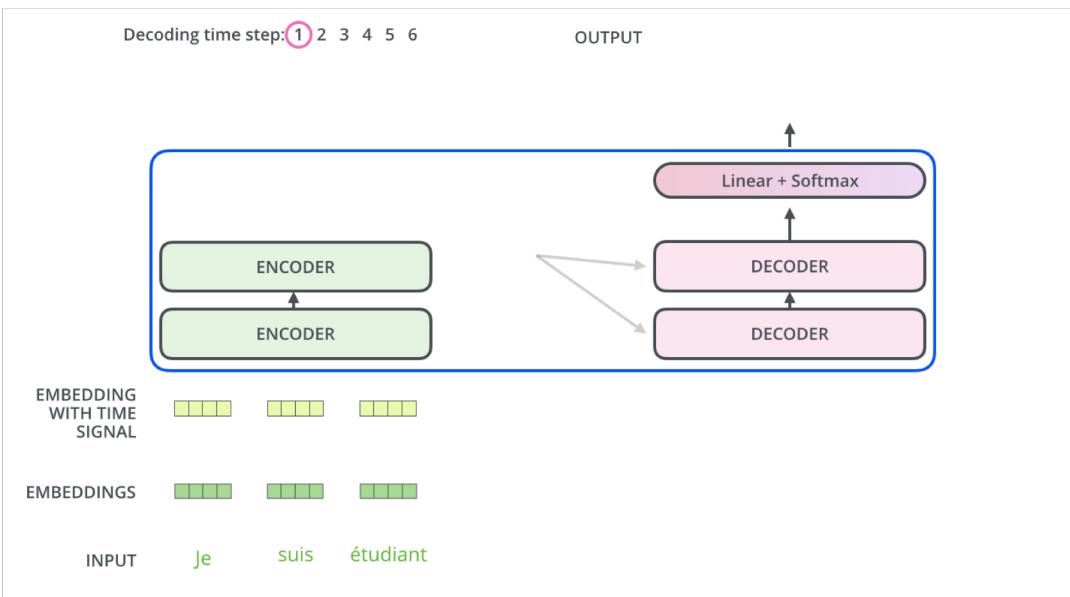
# Transformer. Encoder summary



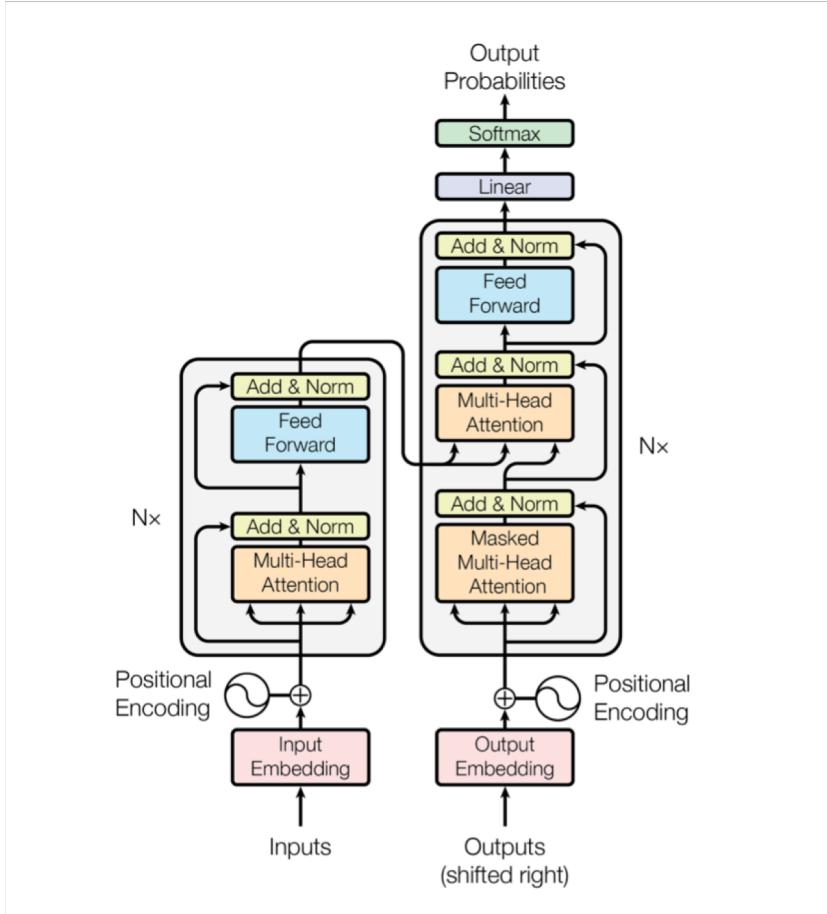
# Transformer. Encoder+Decoder



# Transformer. Encoder+Decoder



# Transformer. Summary

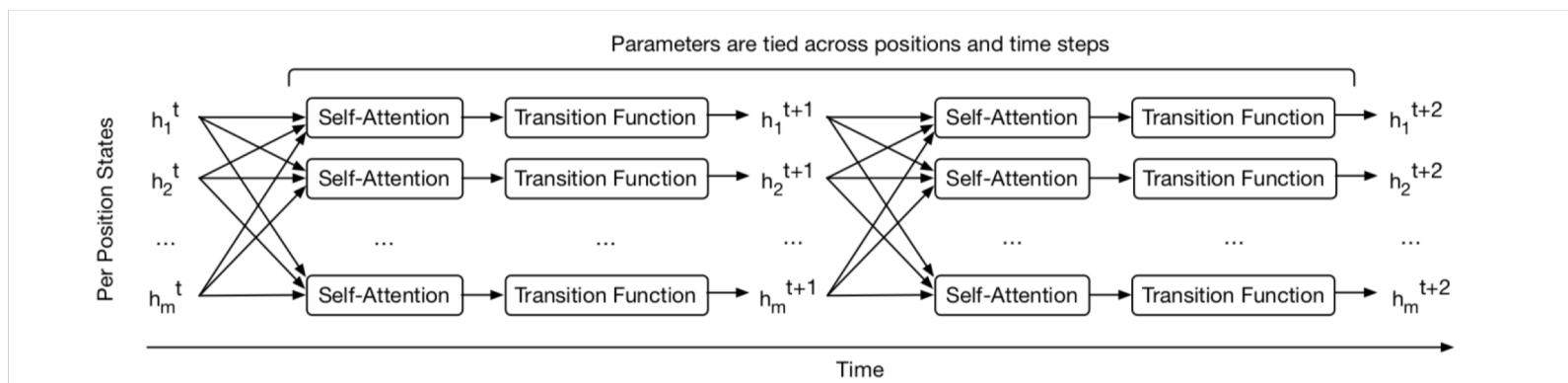
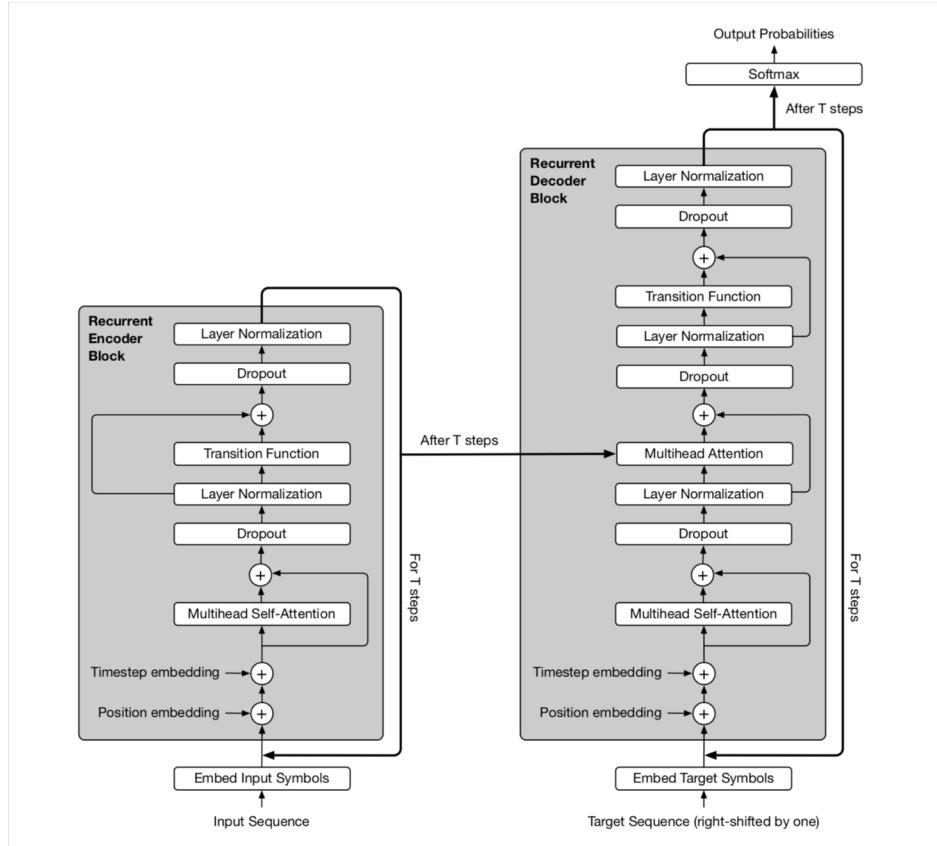


- WMT14: 28.4 BLEU (+2 к топовому результату)
- Решена проблема с LSTM
- Больше простора для параллелизации вычисления

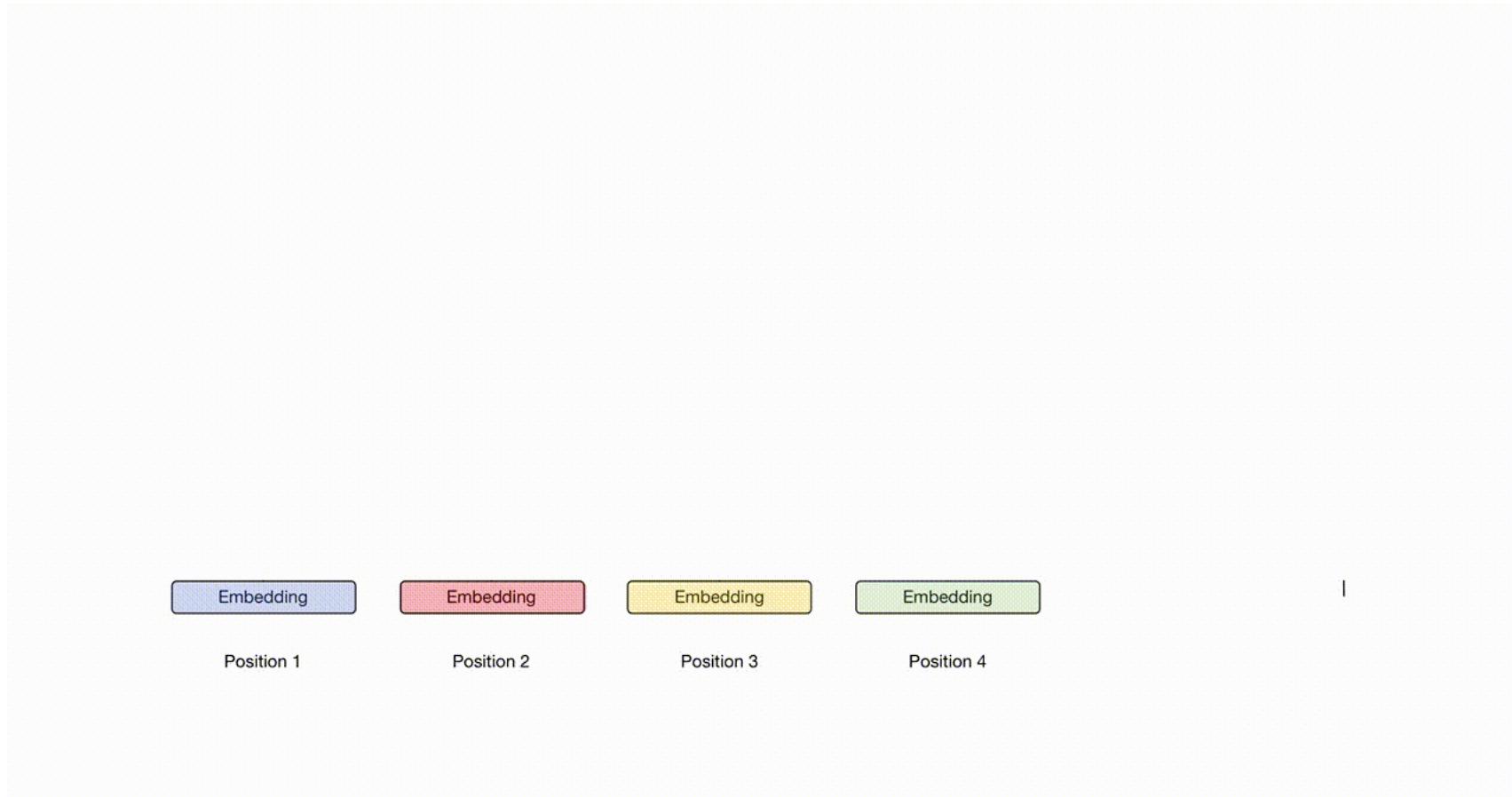
Но есть ли минусы?

Да, Transformer не может справиться с многими задачами о последовательностях, с которыми RNN легко справляются(например, дублирование строки). В целом он является ограниченным по своим возможностям к обобщению.

# Universal Transformer, Deghani et al, 2018

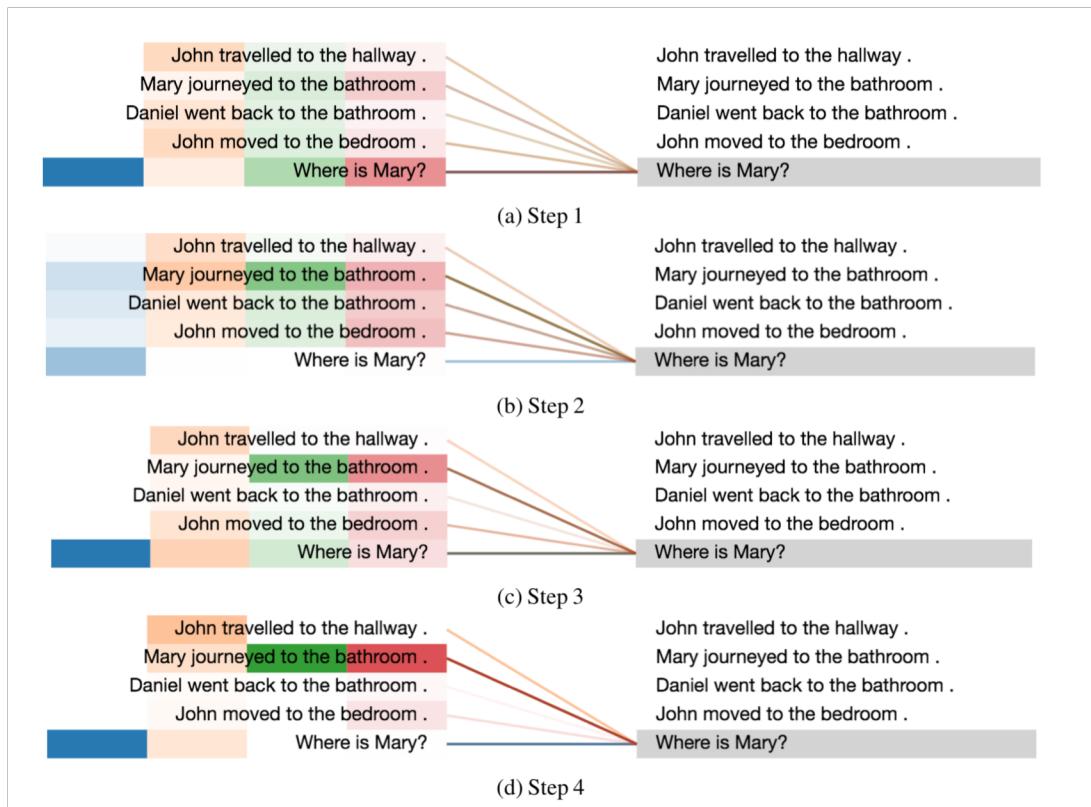


# Universal Transformer, Adaptive computation time



# Universal Transformer, Performance

<b>Story:</b>	John travelled to the hallway. Mary journeyed to the bathroom. Daniel went back to the bathroom. John moved to the bedroom
<b>Query:</b>	Where is Mary?
<b>Model's output:</b>	bathroom



- WMT14: 28.9(+0.5 к результату Трансформера)
- Тьюринг-полный
- Имеет рекуррентный вид, но при этом не обладает недостатками обычных RNN
- Хорошо параллелизуемый

# Пример использования Transformer'а во временных рядах

## Данные:

Electric Reliability Council of Texas (ERCOT),  
(потребление в КВт/Ч за каждый час  
с 2003 по 2016

## Признаки:

КВт/ч, таймстемп, день недели(one-hot), год, час

## Задача:

По предыдущим 24ч предсказывать  
потребление за последующие 12ч

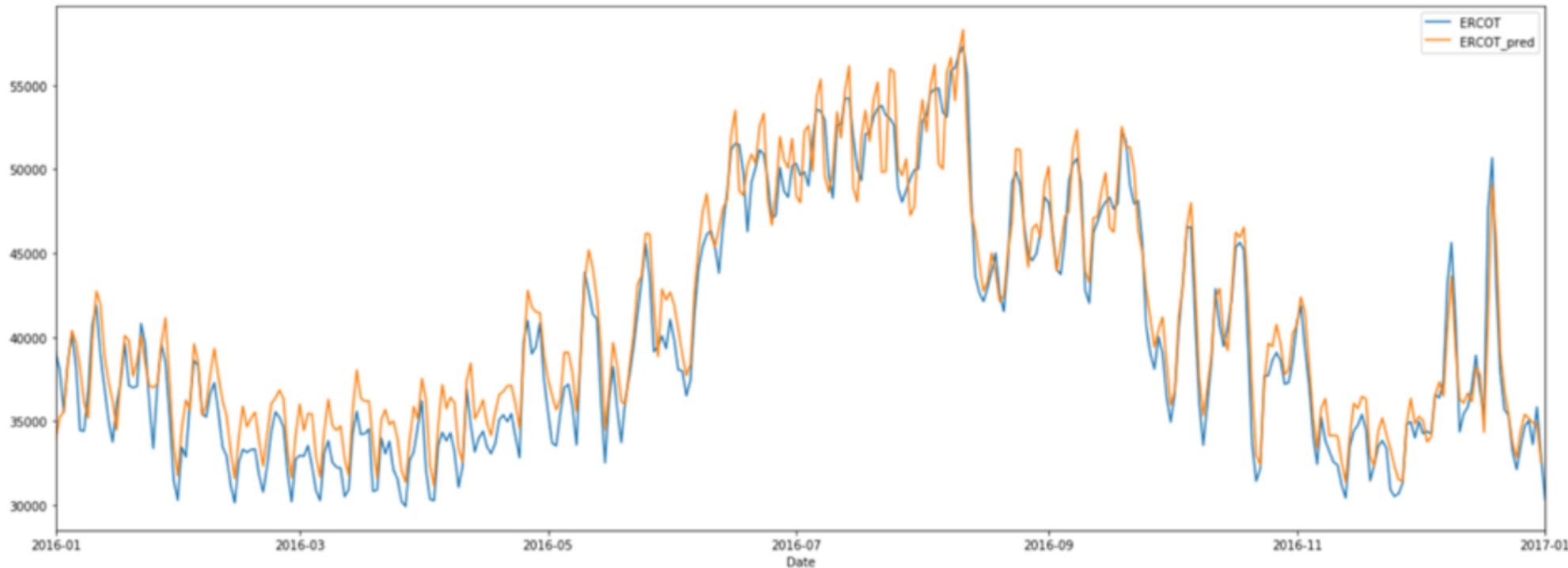
## Изменения в архитектуре:

Убираем эмбеддинги(данные и так числовые)  
Убираем SoftMax(у нас задача регрессии)

## Лосс:

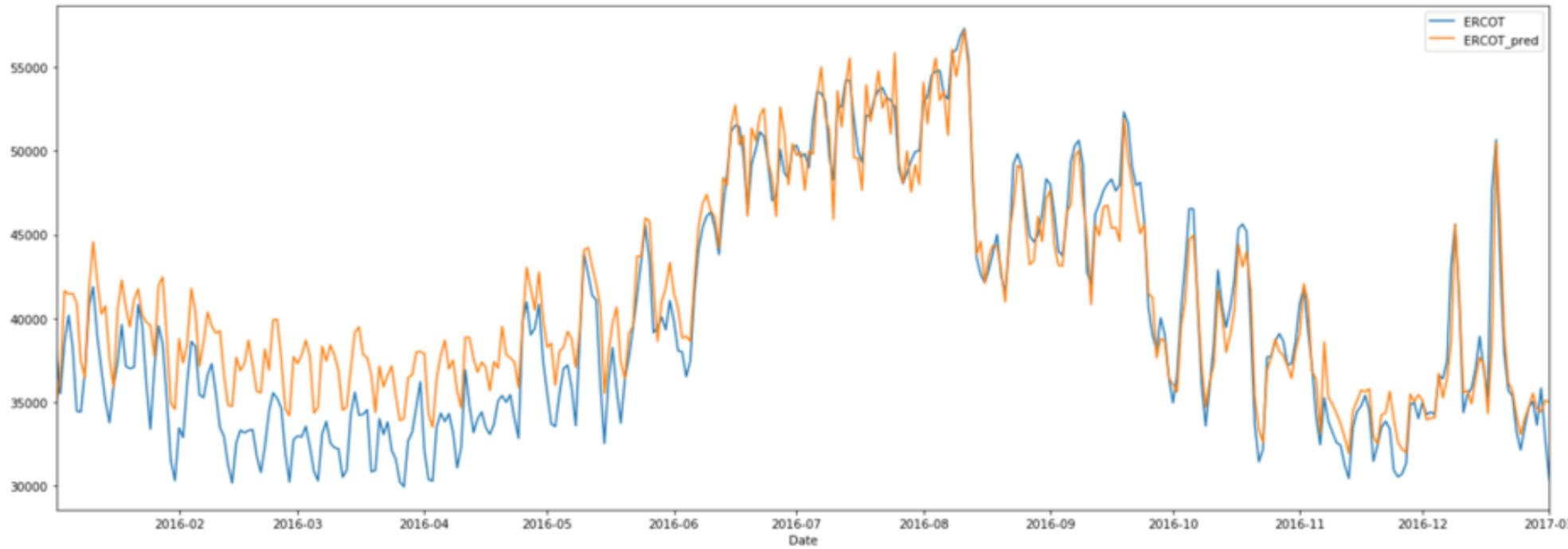
MSE

# Пример использования Transformer'а во временных рядах



одночасовое предсказание, полученное по предыдущим 24-часам, в течение одного года

# Пример использования Transformer'а во временных рядах



12-часовое предсказание, полученное по предыдущим 24-часам, в течение одного года

# Пример использования Transformer'а во временных рядах

Вывод:

Предсказания получаются довольно точными, однако чем больше период, на который мы хотим предсказывать, тем хуже точность.

# Выводы

- Transformer >> Seq2Seq (Качество и эффективность вычислений)
- Universal Transformer > Transformer (Качество + типы решаемых задач)
- Однако Transformer имеет теоретические ограничения в приложениях, в отличие от Тьюринг-полного Universal Transformer

# Источники

- <https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04>
- <https://arxiv.org/abs/1706.03762>
- <https://arxiv.org/abs/1807.03819>
- <http://jalammar.github.io/illustrated-transformer/>