

Transformers

План

1. NLP, работа с последовательностями
2. Обзор более ранних архитектур
3. Attention is all you need
4. Особенности тренировки
5. Vision Transformer (ViT)

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

Обработка естественного языка (Natural Language Processing, NLP)

Основными тремя направлениями являются:

1. Распознавание речи (Speech Recognition)
2. Понимание естественного языка (Natural Language Understanding)
3. Генерация естественного языка (Natural Language Generation)

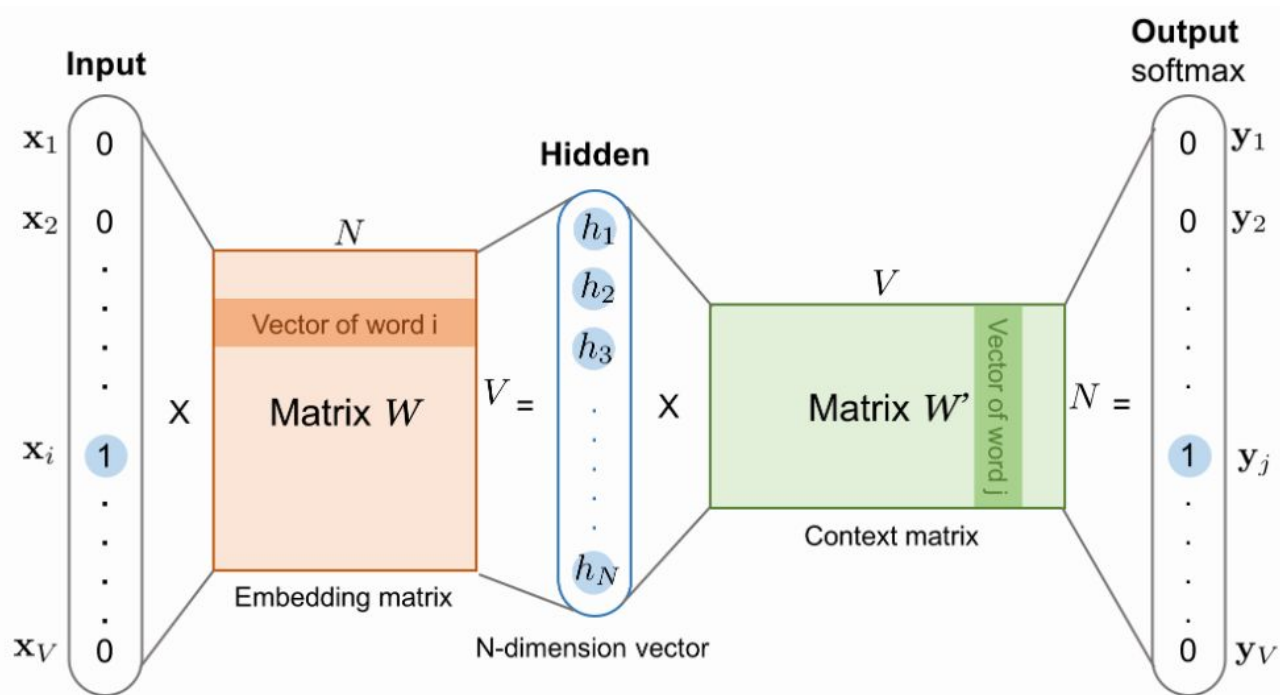
Основные задачи при работе с текстом

1. Классификация текстов (Выставление оценок, Эмоциональная окраска)
2. Перевод текстов
3. Распознавание именованных сущностей (классификация слов на части речи или выделение имен людей, компаний, даты и тд)
4. Продолжение последовательностей (Дать ответ продолжить предложение)

Текст в вектора (one-hot encoding)

Feature (Color)		One Hot Encoded Vector	Red	Green	Yellow
Red	One Hot Encoding →	[1,0,0]	1	0	0
Green		[0,1,0]	0	1	0
Yellow		[0,0,1]	0	0	1
Green		[0,1,0]	0	1	0
Red		[1,0,0]	1	0	0

Word2Vec



Матрица W ($N \times V$)
 N - размер вектора
 V - длина словаря

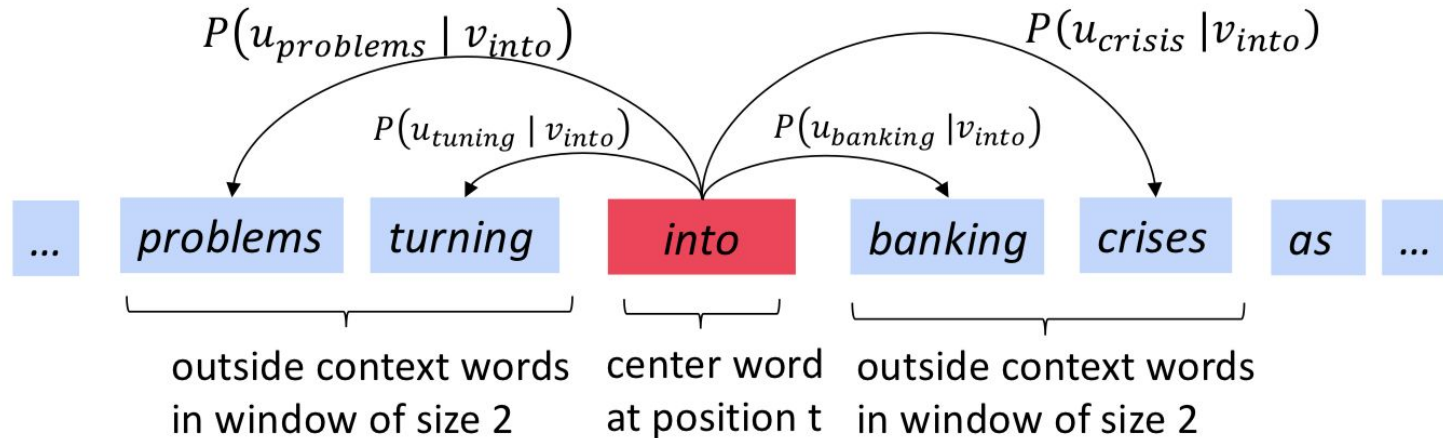
вектор i - строка матрицы W
вектор j - колонка матрицы W'

на выходе получаем скалярное
произведение векторов
 $\text{softmax}(i \cdot j)$

Задача чтобы скалярное
произведение было как можно
больше

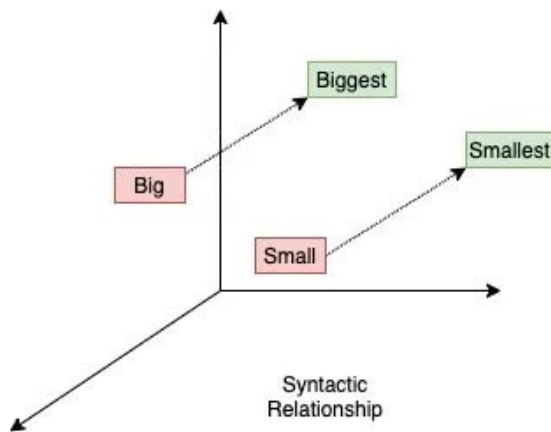
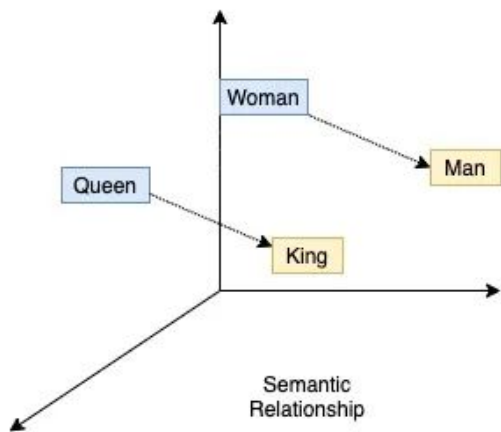
loss - Cross Entropy

Word2Vec



Word2Vec

king - man + woman \approx queen



Word2Vec

Недостатки:

- фиксированный набор словаря
- для редких слов эмбединги получаются не оптимальные
- нужны серьезные предобработки т.к слова с разными окончаниями будут разными (Лемматизация, стемминг)

Стемминг – это грубый эвристический процесс, который отрезает «лишнее» от корня слов, часто это приводит к потере словообразовательных суффиксов.

Лемматизация – это более тонкий процесс, который использует словарь и морфологический анализ, чтобы в итоге привести слово к его канонической форме – лемме.

Материалы

<https://jramkiss.github.io/2019/08/21/word-embeddings/>

<https://arxiv.org/pdf/1301.3781.pdf>

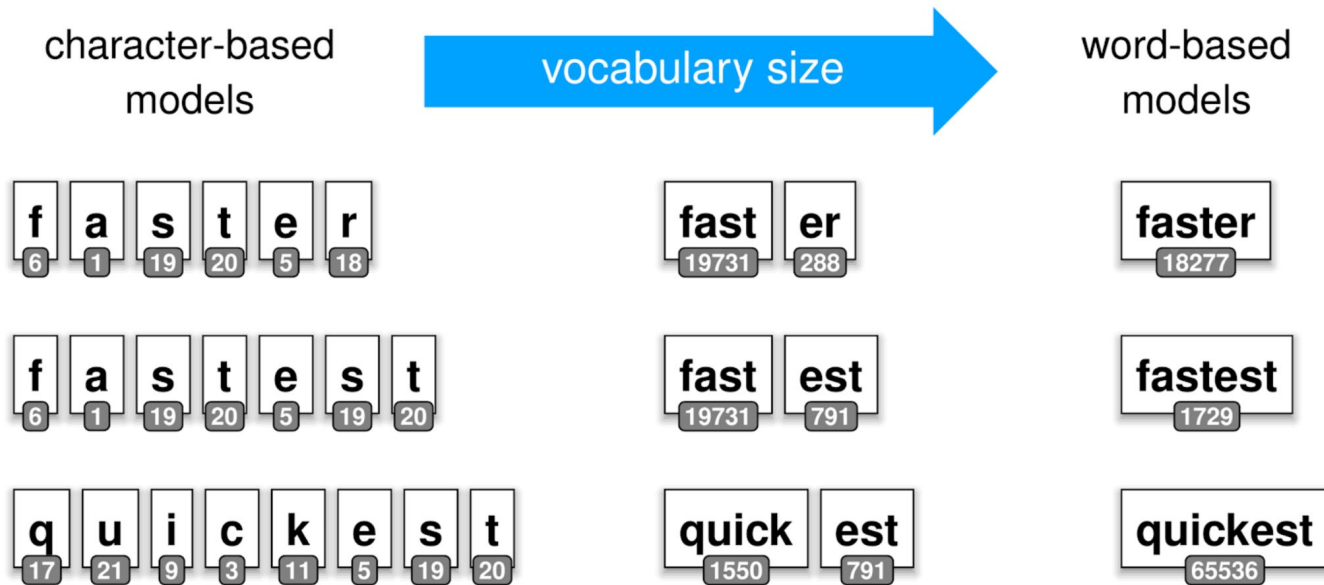
<https://habr.com/ru/articles/446530/>

<https://towardsdatascience.com/word2vec-research-paper-explained-205cb7eccc30>

<https://www.youtube.com/watch?v=WbtQzAvhnRI>

https://mohitmayank.com/a_lazy_data_science_guide/natural_language_processing/word2vec/

Tokenization



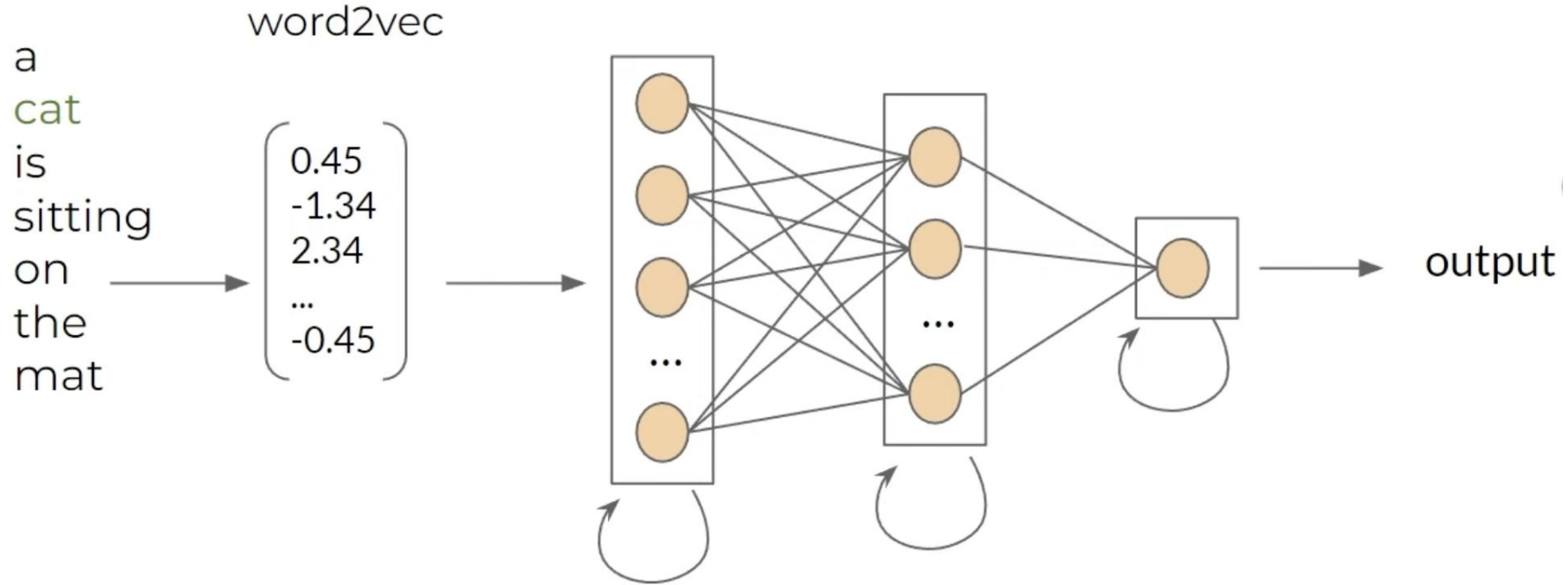
<https://platform.openai.com/tokenizer> tokenizer gpt

https://huggingface.co/docs/transformers/tokenizer_summary tokenizer_summary

https://youtu.be/iOrNbK2T92M?si=j7L2c_EiCqtxEuho Igor Kotenkov

<https://youtu.be/zduSFxRajkE?si=i-63AmBDdb1aZ-mhH> Andrej Karpathy

Recurrent neural network (RNN)

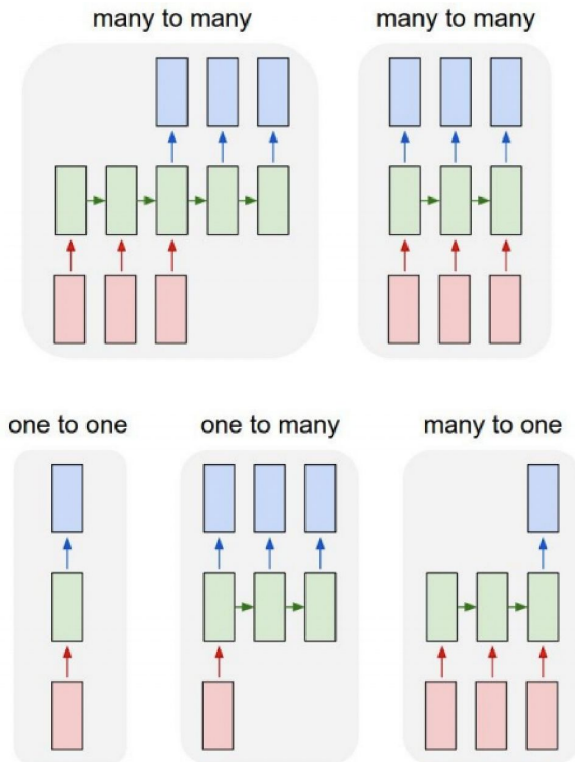


Рекуррентный слой принимает текущее состояние + вектор состояния с предыдущего шага

Типы рекуррентных сетей

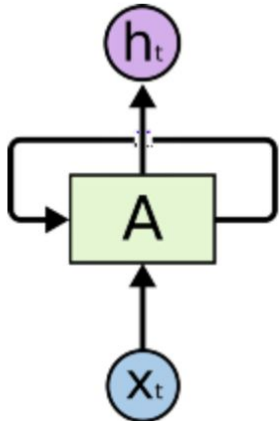
- Many to Many - для перевода текстов
- Many to One - для анализа окраски текста (на входе текст на выходе категории положительный нейтральный отрицательный)
- One to Many - для генерации текста для изображений или автодополнение
- One to One - относительно редкая архитектура для вычисление нелинейных рекуррентных вычислений

Основным недостатком простых рекуррентных сетей является забывание начала предложения



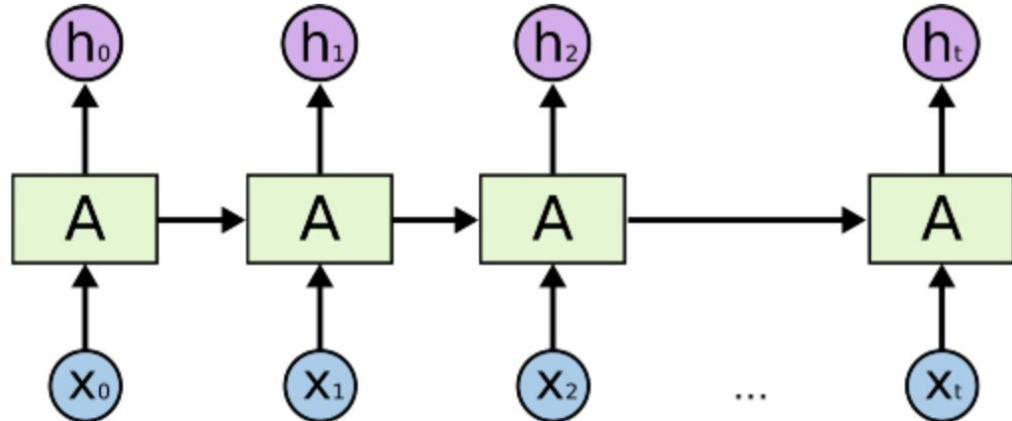
Recurrent neural network (RNN)

Рекуррентный слой



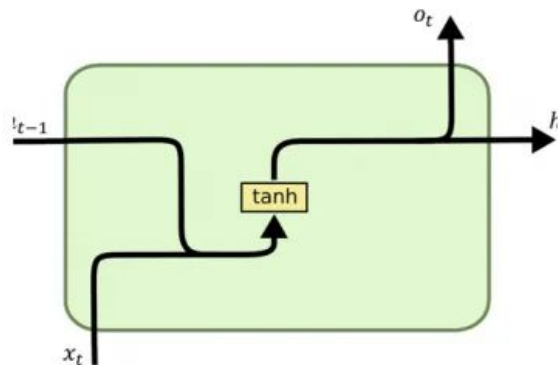
=

Рекуррентный слой развернутый во времени

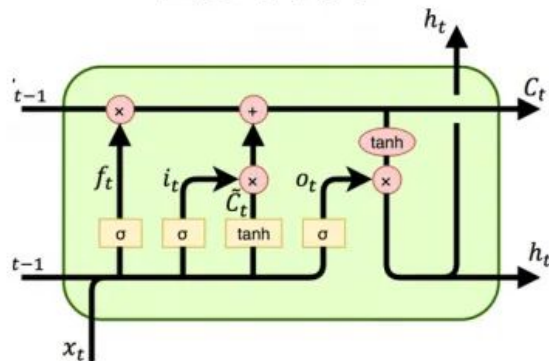


Вариации слоев

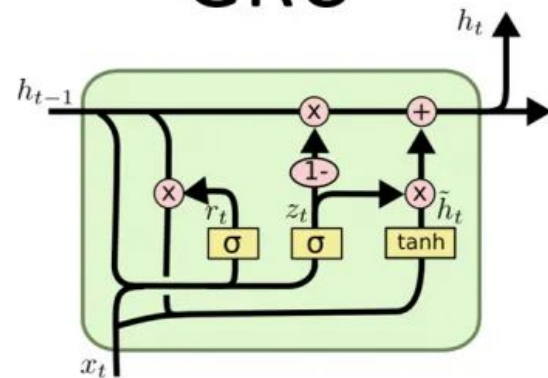
RNN



LSTM



GRU



http://vbystricky.ru/2021/05/rnn_lstm_gru_etc.html

https://youtu.be/3OlkWQ2Uc0?si=K_JzNqXz0ToSFSfb

Attention is all you need

1. Feed Forward
2. Multi-Head Attention
3. Masked Multi-Head Attention
4. Add & Norm
5. Positional Encoding

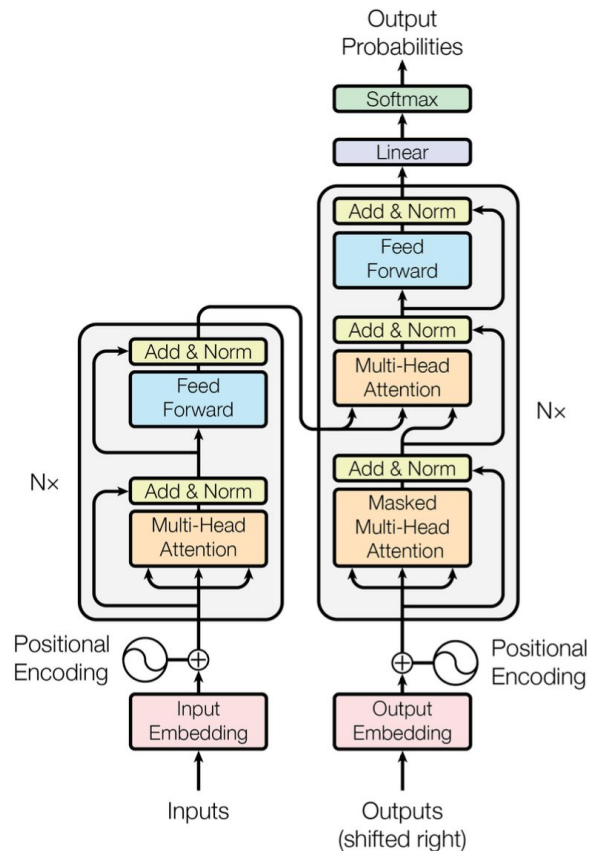


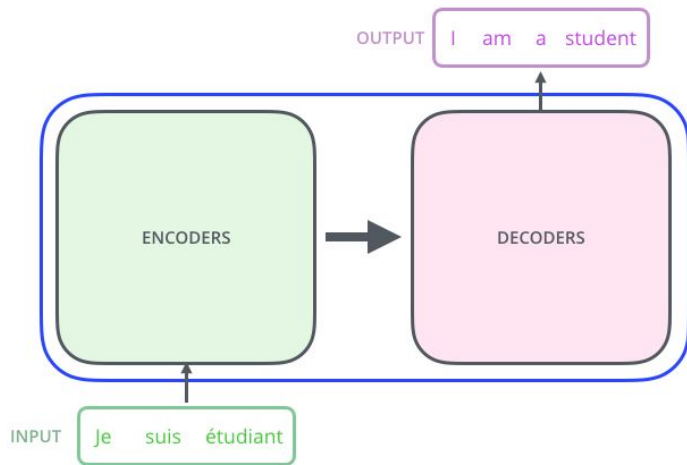
Figure 1: The Transformer - model architecture.

<https://jalammar.github.io/illustrated-transformer/>
<https://arxiv.org/abs/1706.03762>

Transformer

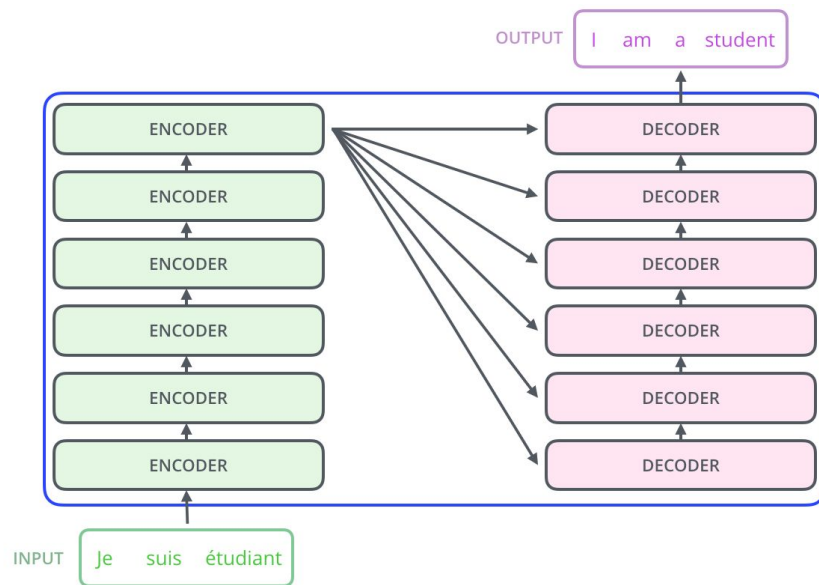


Архитектура взялась из
задачи перевода текста



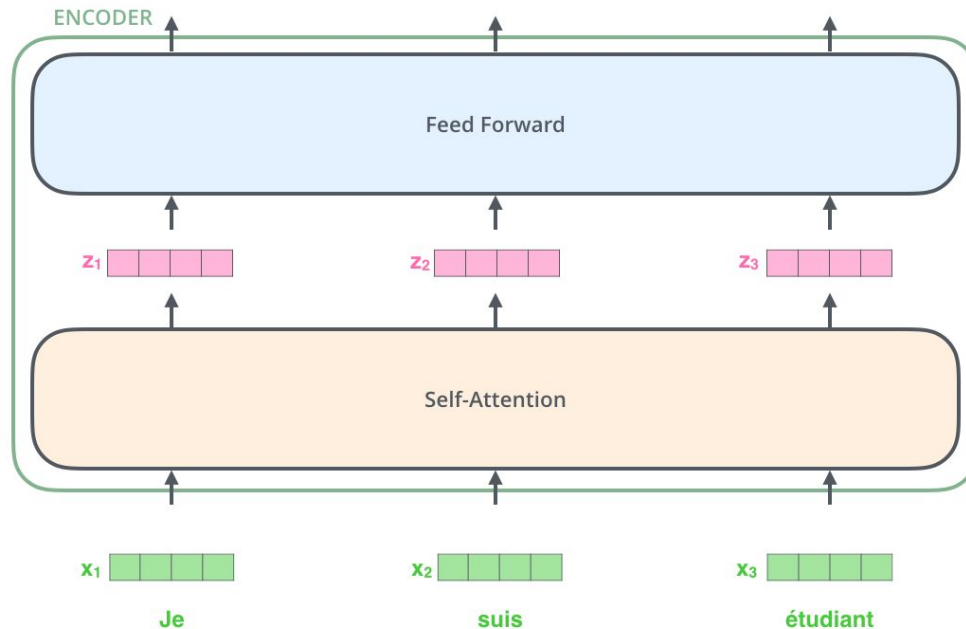
Transformer

- Так как у нас 6 блоков энкодеров это значит что размерность выхода каждого каждого энкодера должна соответствовать входу
- Выход и энкодеров далее подается в декодеры
- На вход подается тензор размера $[b, l, E_{dim}]$
- b batch size
- l длина предложения
- E_{dim} размерность эмбединга слова



Encoder block

В каждом слое – слой self-attention и feedforward layer, который независимо применяется к каждой позиции входа

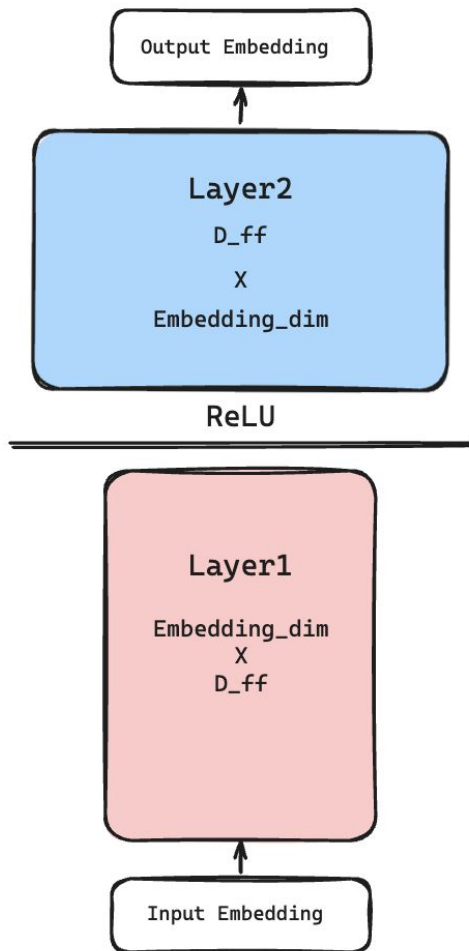


Feed forward layer

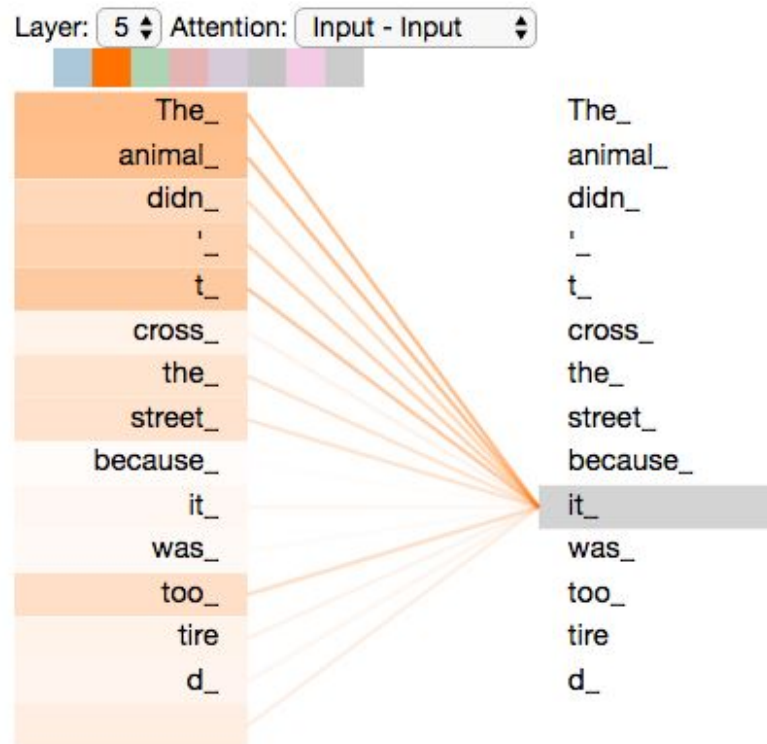
Feed-forward layer **eats 2/3 of the model params!**

Embedding_dim \rightarrow D_ff \rightarrow Embedding_dim

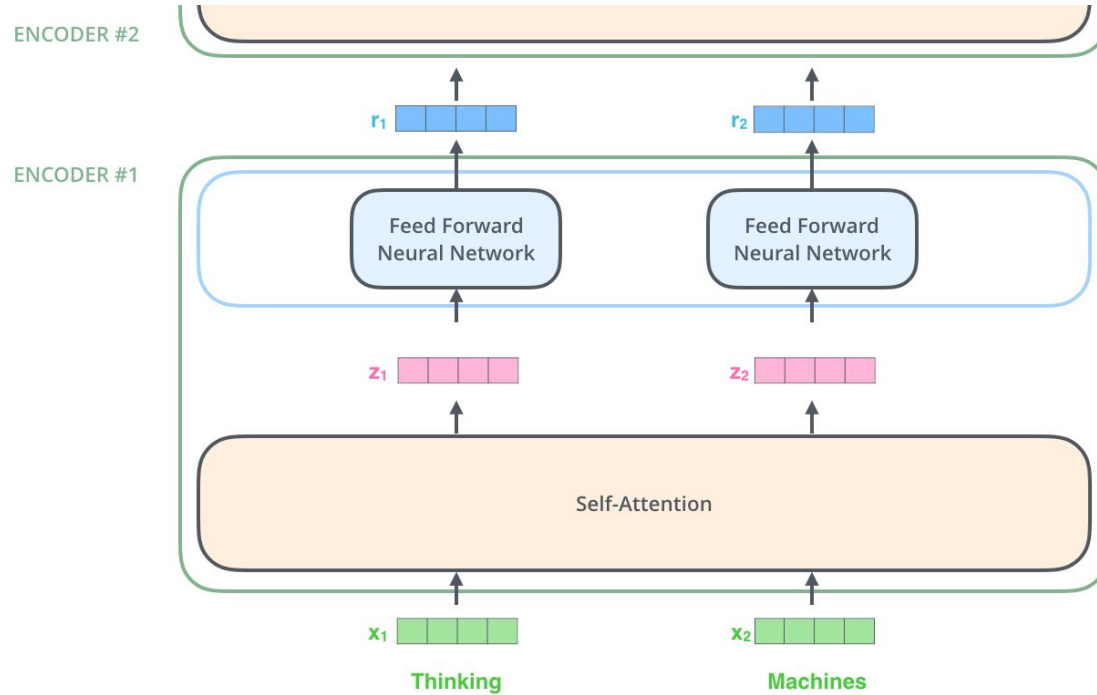
D_ff (x4) or Embedding_dim



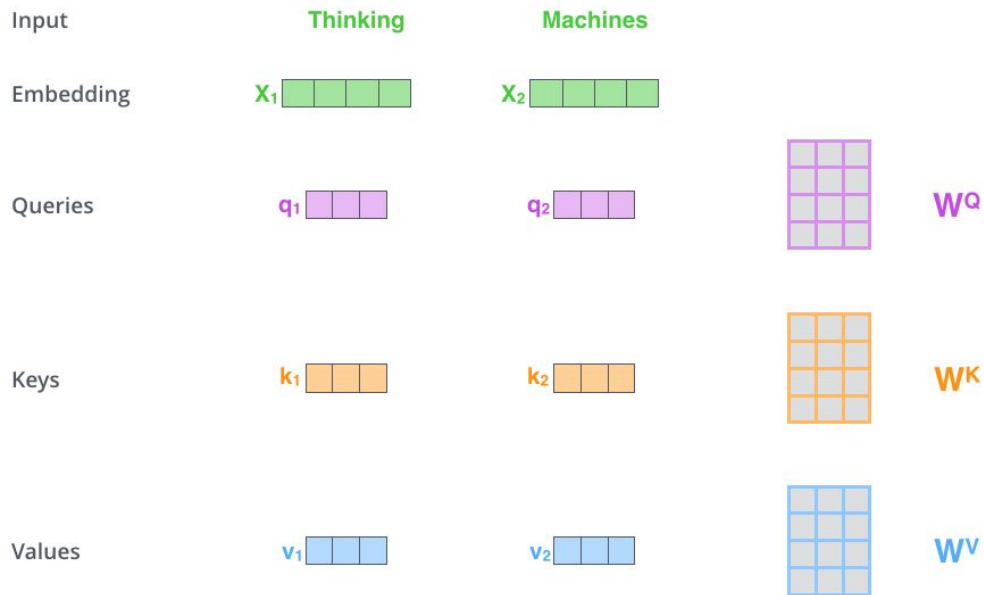
Self-Attention



Self-Attention



Self-Attention



$$x = (1, n)$$

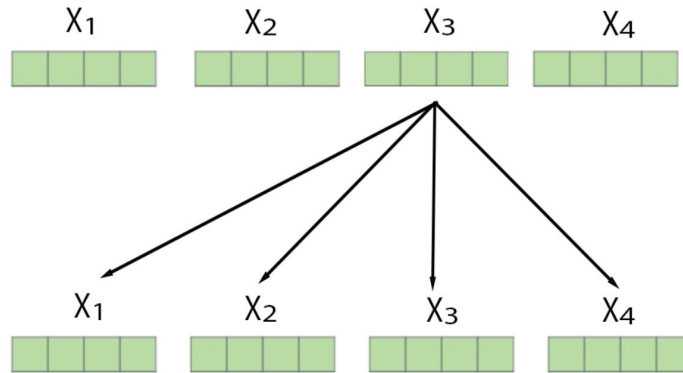
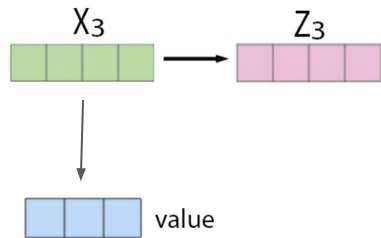
$$W^Q, W^K, W^V = (n, m)$$

$$q = x * W^Q$$

$$k = x * W^K$$

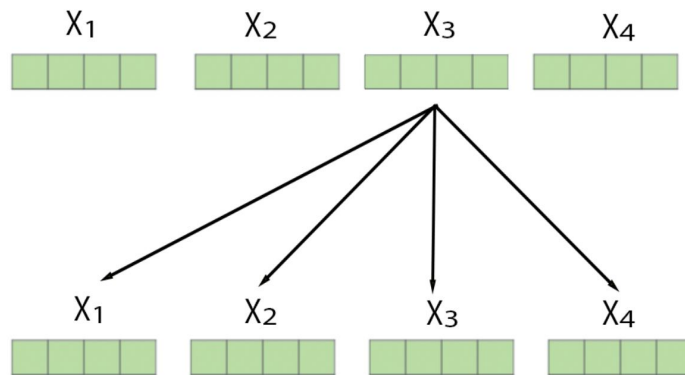
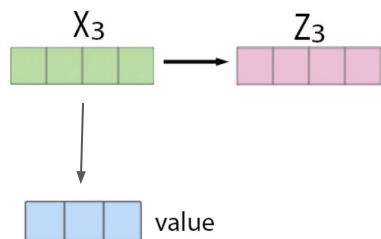
$$v = x * W^V$$

Self-Attention



$$Z_3 = \sum_{i=1}^4 a_i * v_i \quad ; \quad \sum_{i=1}^4 a_i = 1$$

Self-Attention

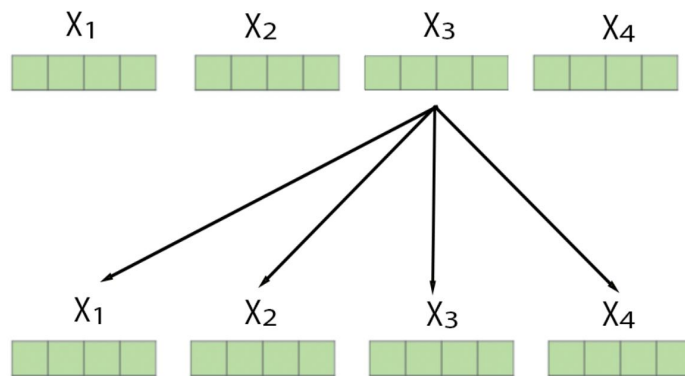
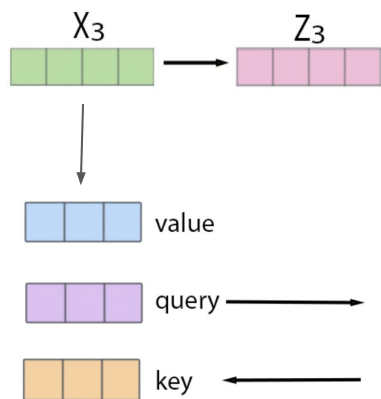


$$Z_3 = \sum_{i=1}^4 a_i * v_i \quad ; \quad \sum_{i=1}^4 a_i = 1$$

$$\left\{ \hat{a}_i \right\}_{i=1}^4 \xrightarrow{\text{Softmax}} \left\{ a_i \right\}_{i=1}^4$$

$$\hat{a}_i = f(x_3, x_i)$$

Self-Attention

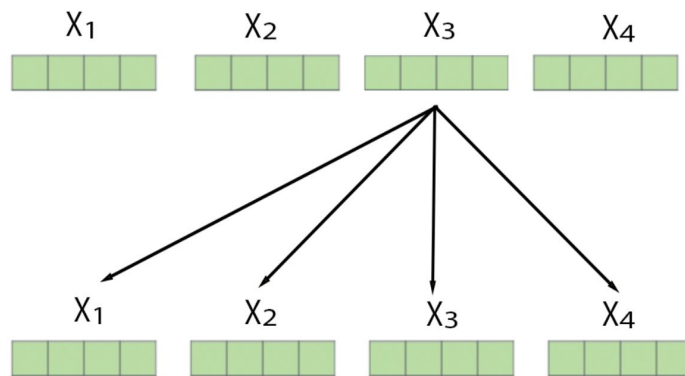
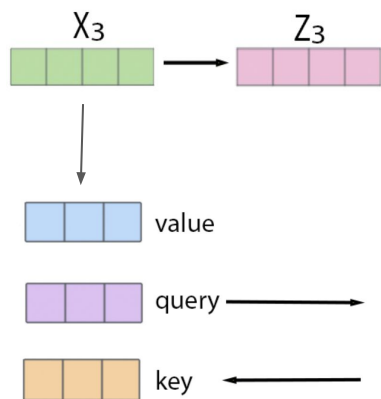


$$Z_3 = \sum_{i=1}^4 a_i * v_i \quad ; \quad \sum_{i=1}^4 a_i = 1$$

$$\left\{ \hat{a}_i \right\}_{i=1}^4 \xrightarrow{\text{Softmax}} \left\{ a_i \right\}_{i=1}^4$$

$$\hat{a}_i = f(x_3, x_i)$$

Self-Attention

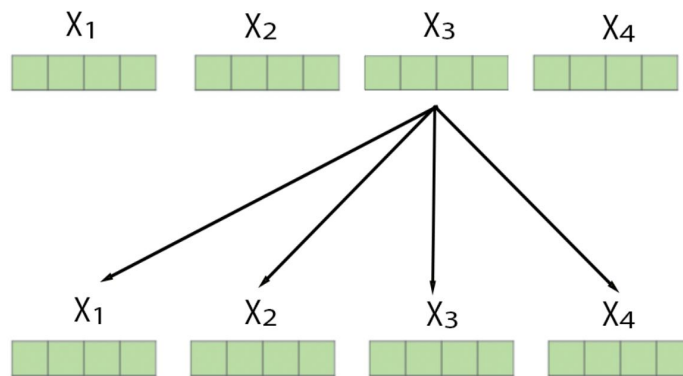
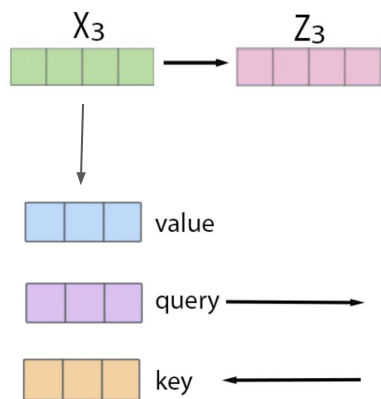


$$Z_3 = \sum_{i=1}^4 a_i * v_i ; \quad \sum_{i=1}^4 a_i = 1$$

$$\left\{ \hat{a}_i \right\}_{i=1}^4 \xrightarrow{\text{Softmax}} \left\{ a_i \right\}_{i=1}^4$$

$$\hat{a}_i = f(x_3, x_i) \quad \hat{a}_i = \langle q_3, k_i \rangle$$

Self-Attention



$$Z_3 = \sum_{i=1}^4 a_i * v_i \quad ; \quad \sum_{i=1}^4 a_i = 1$$

$$\left\{ \hat{a}_i \right\}_{i=1}^4 \xrightarrow{\text{Softmax}} \left\{ a_i \right\}_{i=1}^4$$

$$\hat{a}_i = f(x_3, x_i) \quad \hat{a}_i = \langle q_3, k_i \rangle$$

$$\left\{ \hat{a}_i \right\}_{i=1}^4 = \{ \langle q_3, k_1 \rangle, \langle q_3, k_2 \rangle, \langle q_3, k_3 \rangle, \langle q_3, k_4 \rangle \}$$

$$Z_1 = a_1 * v_1 + a_2 * v_2$$

$$Z_1 = < q_1 * k_1 > * v_1 + < q_1 * k_2 > * v_2$$

Делим на квадратный корень из
размерности чтобы не увеличивать
дисперсию

Input

Embedding

Queries

Keys

Values

Score

Divide by 8 ($\sqrt{d_k}$)

Softmax

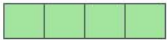
Softmax

X

Value

Sum

Thinking

x_1 

q_1 

k_1 

v_1 

$q_1 \bullet k_1 = 112$

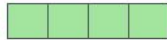
14

0.88

v_1 

z_1 

Machines

x_2 

q_2 

k_2 

v_2 

$q_1 \bullet k_2 = 96$

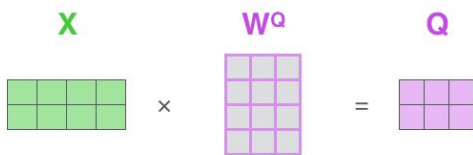
12

0.12

v_2 

z_2 

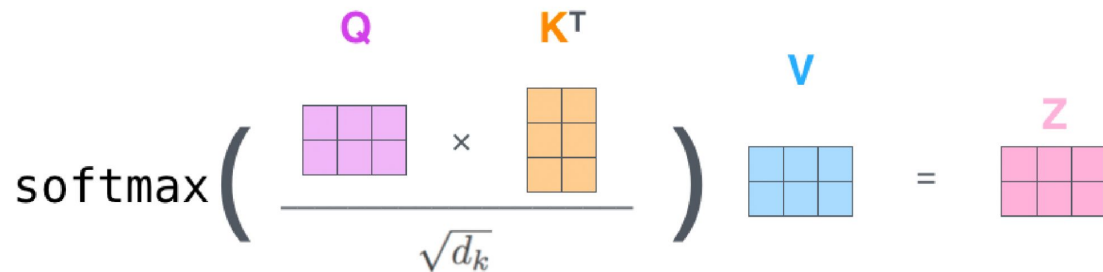
Self-Attention

$$\mathbf{X} \times \mathbf{W}^Q = \mathbf{Q}$$


$$\mathbf{X} \times \mathbf{W}^K = \mathbf{K}$$

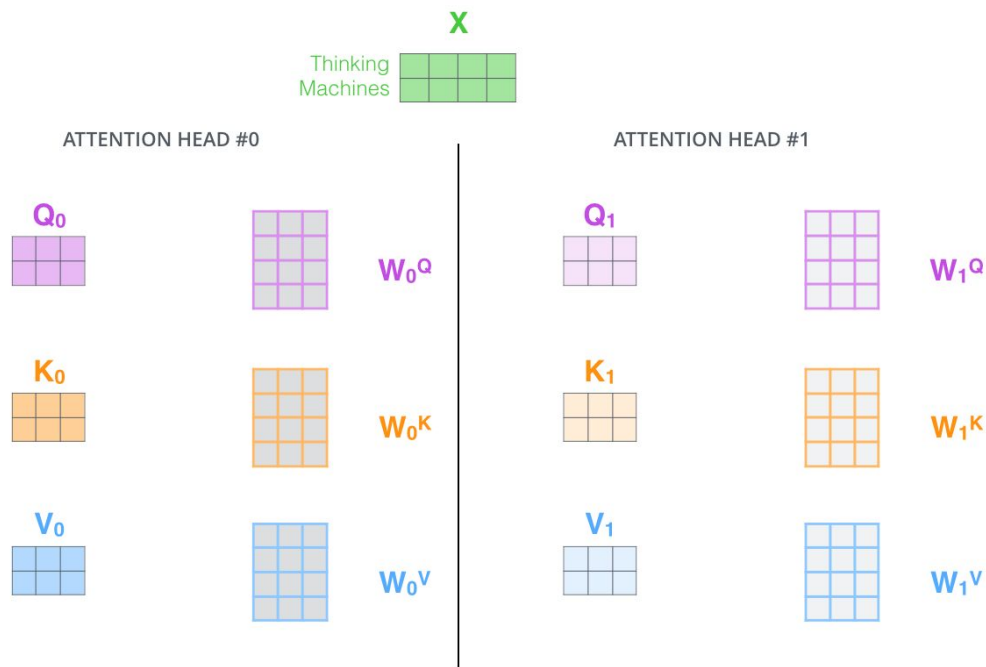

$$\mathbf{X} \times \mathbf{W}^V = \mathbf{V}$$


Финальная формула

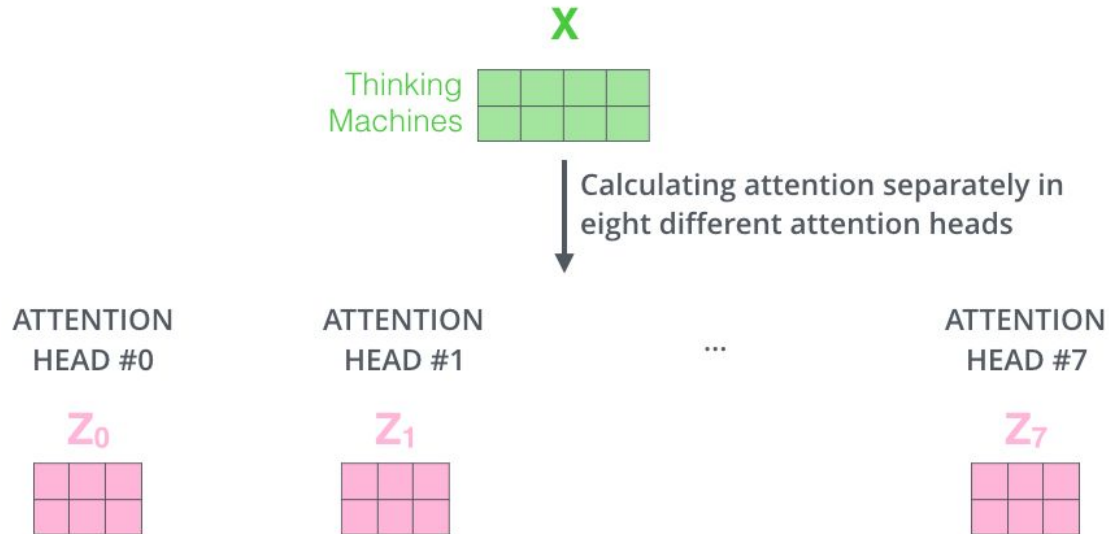
$$\text{softmax}\left(\frac{\mathbf{Q} \times \mathbf{K}^T}{\sqrt{d_k}}\right) \mathbf{V} = \mathbf{Z}$$


$$Z_3 = \sum_{i=1}^4 a_i * v_i$$

Multi-Head Attention



Multi-Head Attention



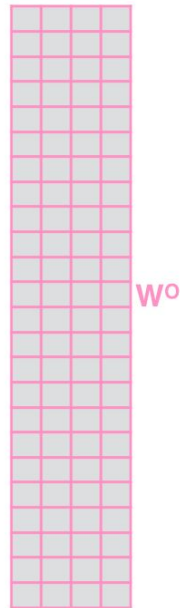
Multi-Head Attention

1) Concatenate all the attention heads



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

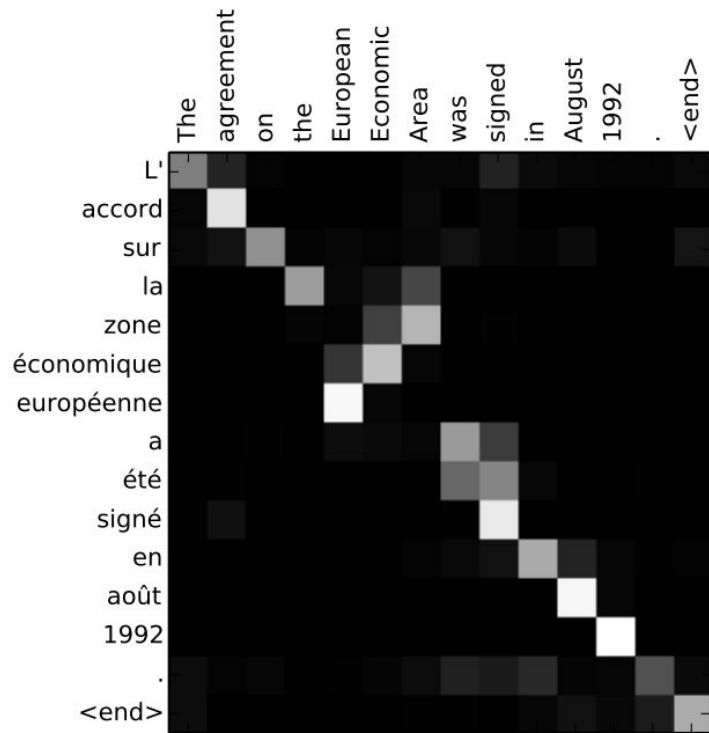
x



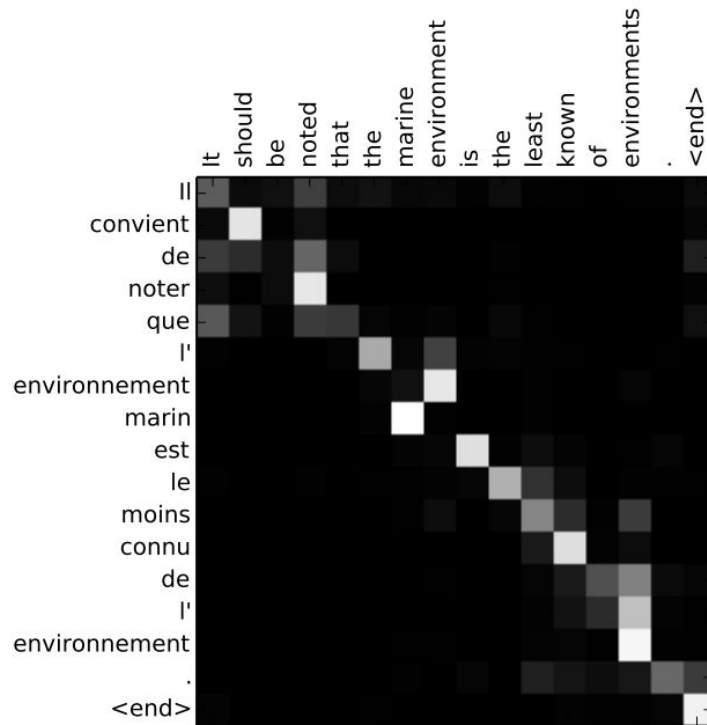
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN



Attention visualization

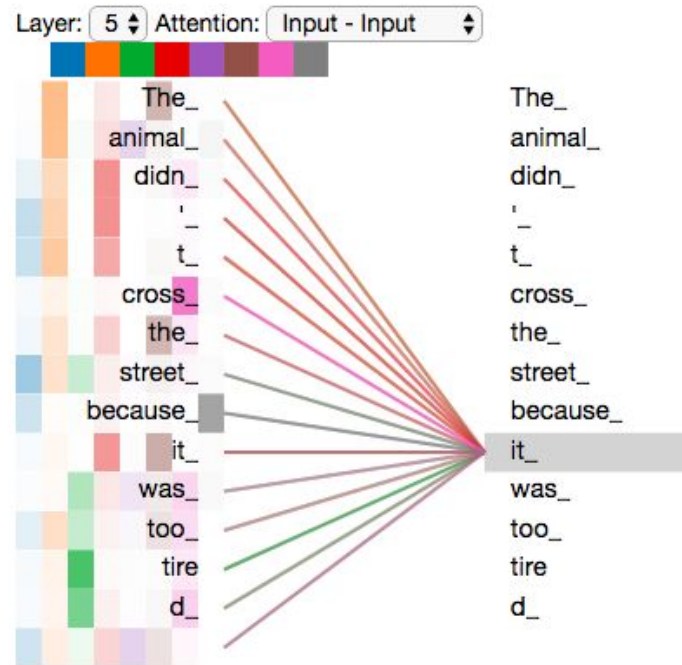
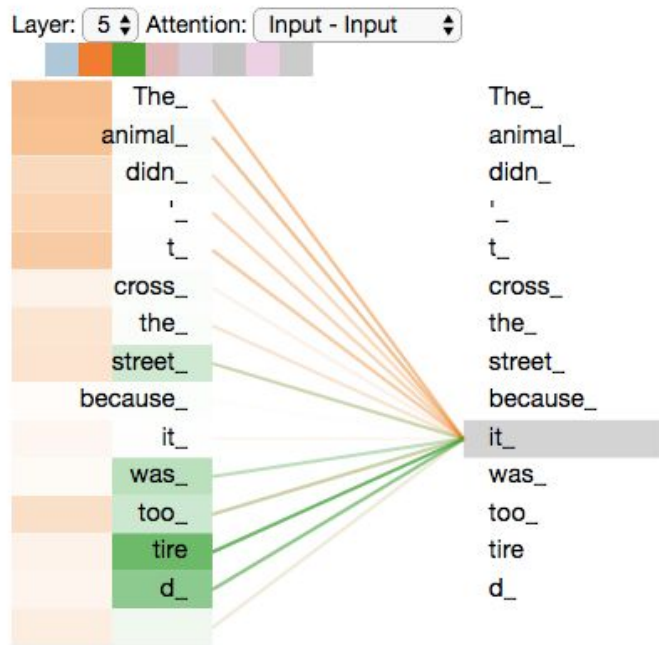


(a)



(b)

Multi-Head Attention



Визуализация разных голов

1) This is our
input sentence*

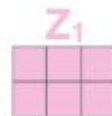
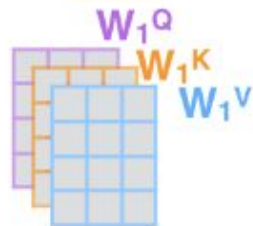
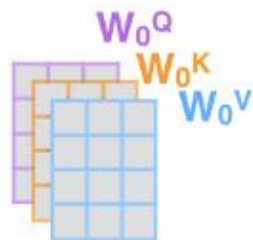
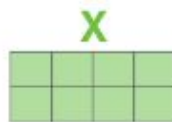
2) We embed
each word*

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

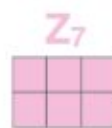
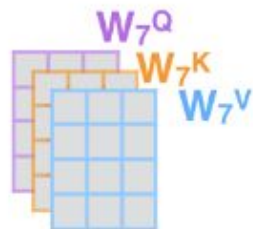
Thinking
Machines



...

...

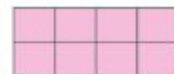
...



W^O



Z

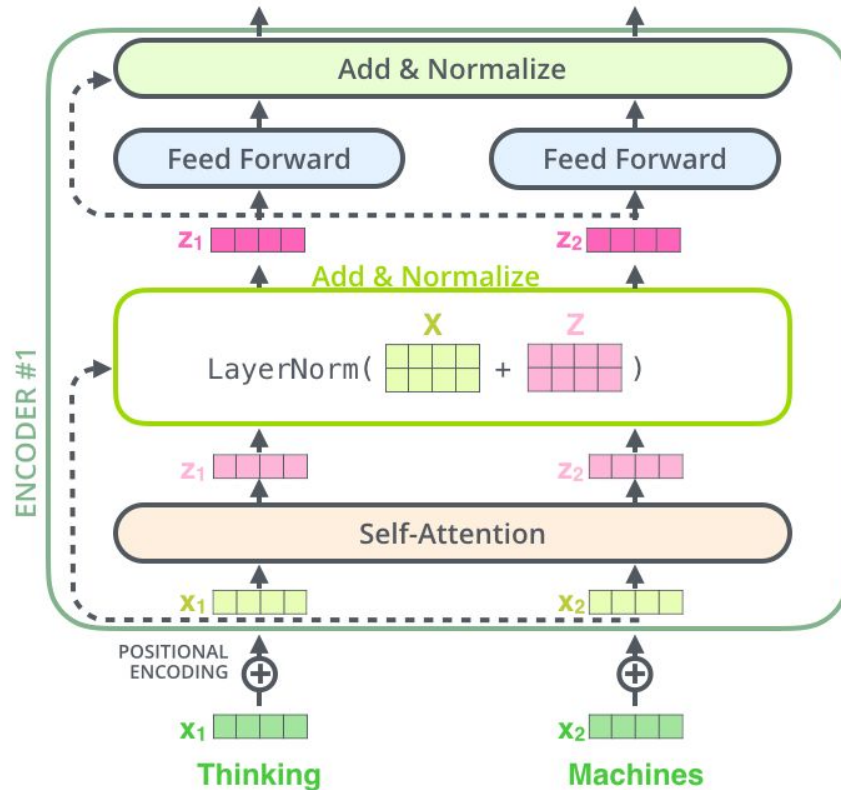
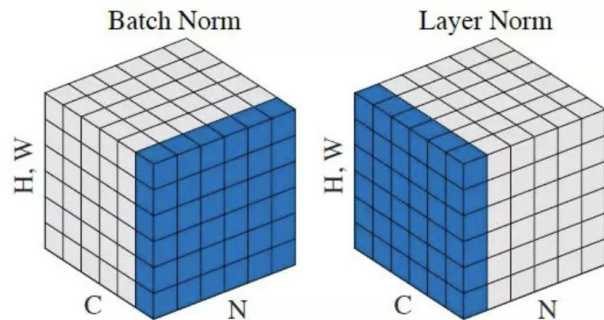


* In all encoders other than #0,
we don't need embedding.
We start directly with the output
of the encoder right below this one

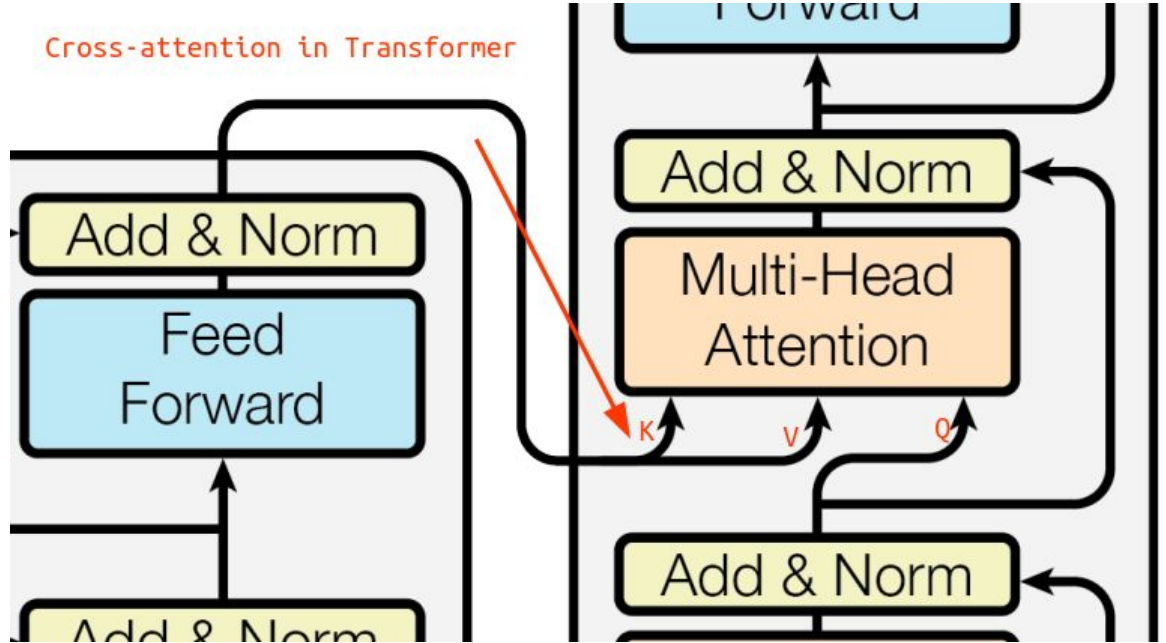
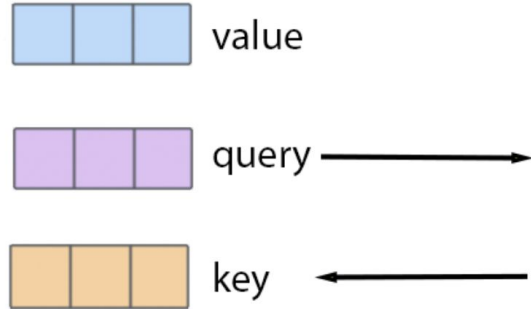


Add & Norm

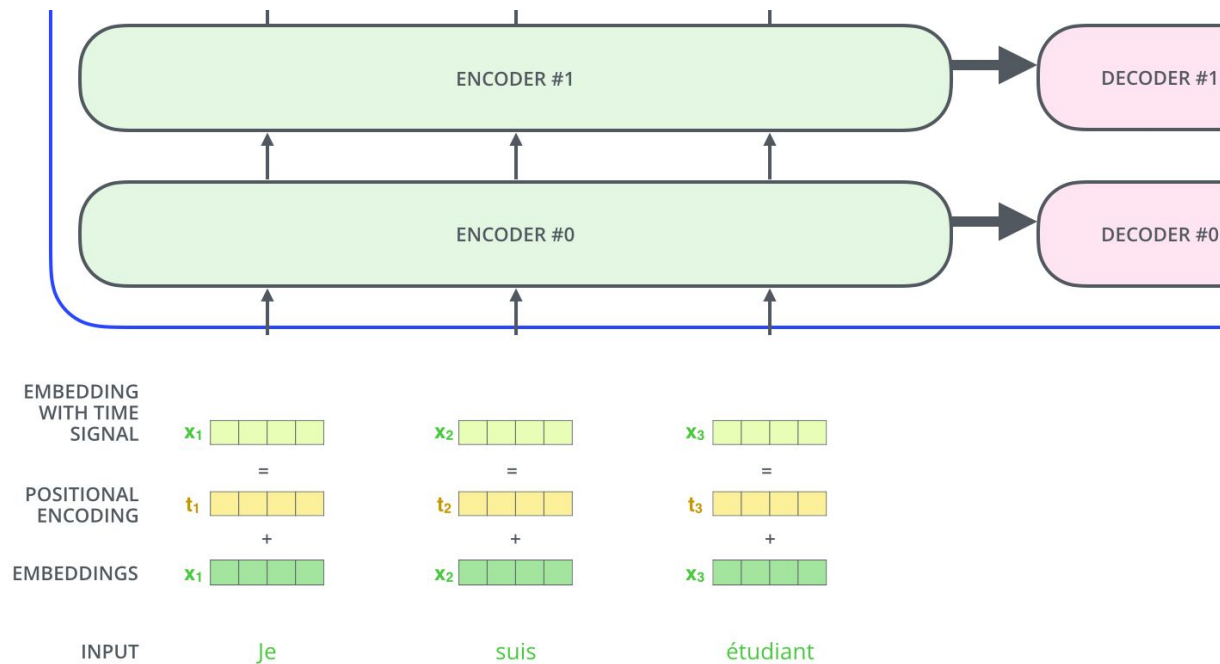
1. Residual connections
2. Normalize to $N(0,1)$ к нулевой мат ожиданием и дисперсией
3. В Layer norm нормализация по фичам



Cross Attention



Positional encoding



Positional encoding

Если в качестве примера размер эмбединга будет равен 4 то то positional вектор будет выглядеть следующим образом



Positional encoding

Простой пример с двоичным кодированием

Мы можем определить скорость изменения между различными битами.

0 : 0 0 0 0

1 : 0 0 0 1

2 : 0 0 1 0

3 : 0 0 1 1

4 : 0 1 0 0

5 : 0 1 0 1

6 : 0 1 1 0

7 : 0 1 1 1

8 : 1 0 0 0

9 : 1 0 0 1

10 : 1 0 1 0

11 : 1 0 1 1

12 : 1 1 0 0

13 : 1 1 0 1

14 : 1 1 1 0

15 : 1 1 1 1

Positional encoding

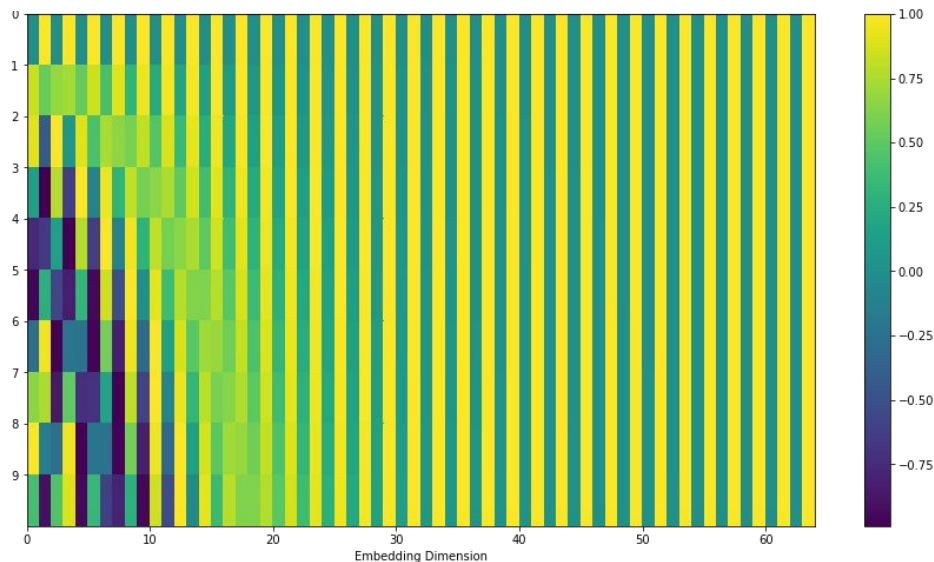
t - номер слова

ω_k частота

Главный плюс
этого кодирования
что норма этого
вектора всегда
постоянная
независимо от t

$$\vec{p}_t = \begin{bmatrix} \sin(\omega_1 \cdot t) \\ \cos(\omega_1 \cdot t) \\ \\ \sin(\omega_2 \cdot t) \\ \cos(\omega_2 \cdot t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} \cdot t) \\ \cos(\omega_{d/2} \cdot t) \end{bmatrix}_{d \times 1}$$

$$\vec{p}_t^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k \cdot t), & \text{if } i = 2k \\ \cos(\omega_k \cdot t), & \text{if } i = 2k + 1 \end{cases} \quad \omega_k = \frac{1}{10000^{2k/d}}$$



Masked Multi-Head Attention

Decoding time step: 1 2 3 4 5 6

OUTPUT |

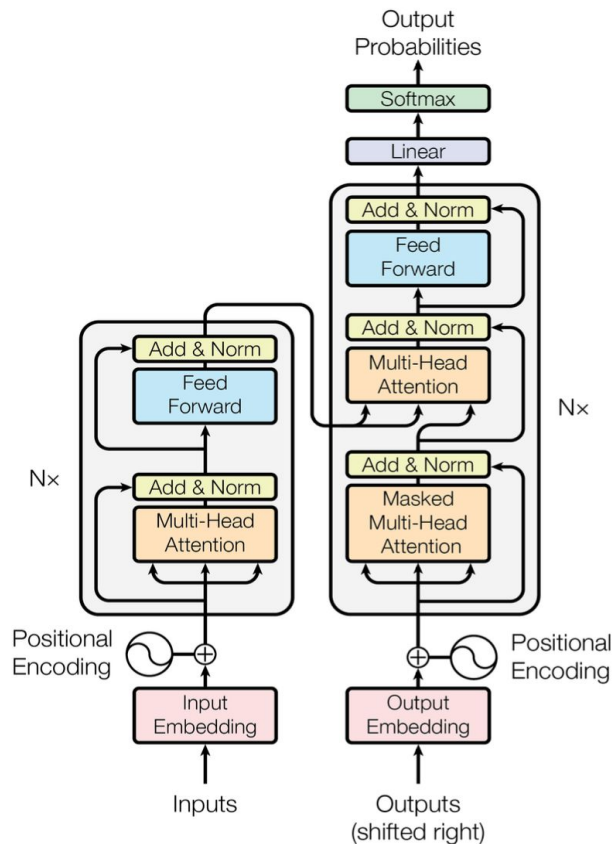
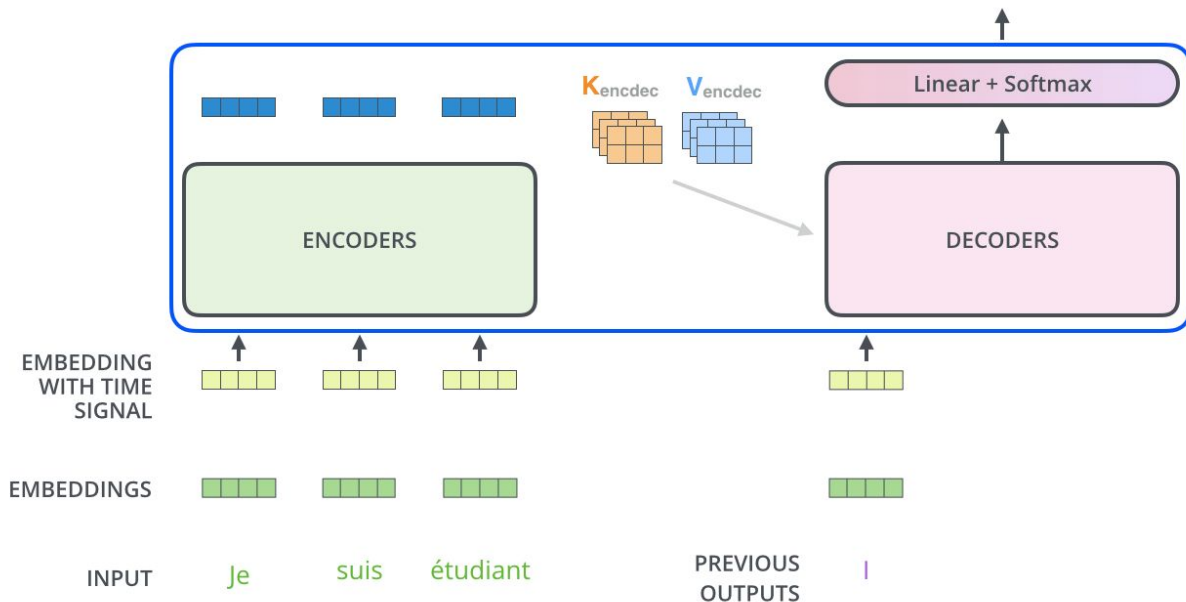


Figure 1: The Transformer - model architecture.

Masked Multi-Head Attention

Softmax $\left(QK^T + \text{attention mask} \right) \times V$

0.2	0.3	0.3	0.1

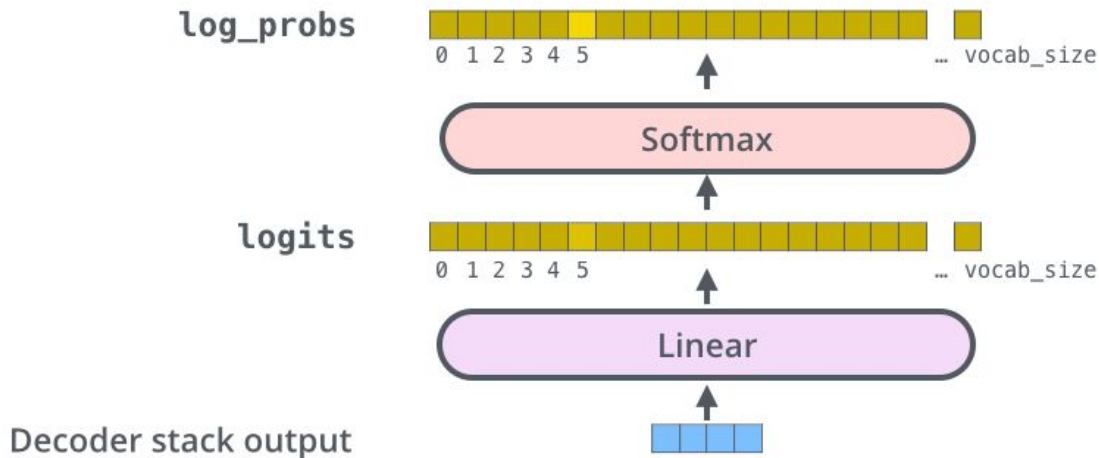
0	-inf	-inf	-inf
0	0	-inf	-inf
0	0	0	-inf
0	0	0	0

<s>	
who	
am	
i	

Loss function

Which word in our vocabulary
is associated with this index?

Get the index of the cell
with the highest value
(**argmax**)



Loss function

Target Model Outputs

Output Vocabulary: a am I thanks student <eos>

position #1 0.0 0.0 1.0 0.0 0.0 0.0

position #2 0.0 1.0 0.0 0.0 0.0 0.0

position #3 1.0 0.0 0.0 0.0 0.0 0.0

position #4 0.0 0.0 0.0 0.0 1.0 0.0

position #5 0.0 0.0 0.0 0.0 0.0 1.0

a am I thanks student <eos>

Trained Model Outputs

Output Vocabulary: a am I thanks student <eos>

position #1 0.01 0.02 0.93 0.01 0.03 0.01

position #2 0.01 0.8 0.1 0.05 0.01 0.03

position #3 0.99 0.001 0.001 0.001 0.002 0.001

position #4 0.001 0.002 0.001 0.02 0.94 0.01

position #5 0.01 0.01 0.001 0.001 0.001 0.98

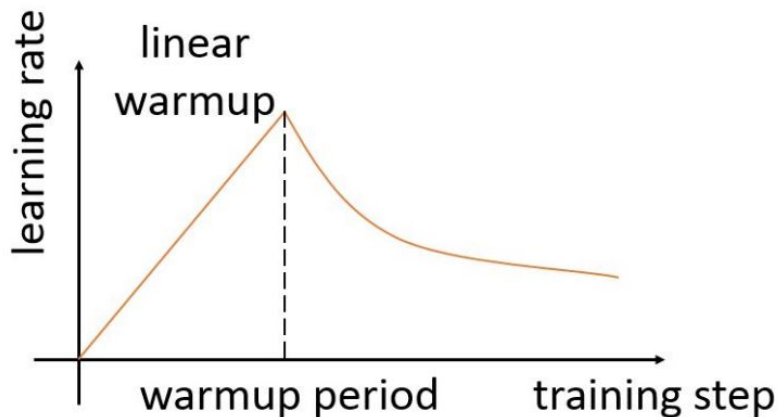
a am I thanks student <eos>

Training

warmup time

Одна из особенностей это
увеличенный learning rate в начале
обучения

и плавное снижение в процессе
обучения



Training

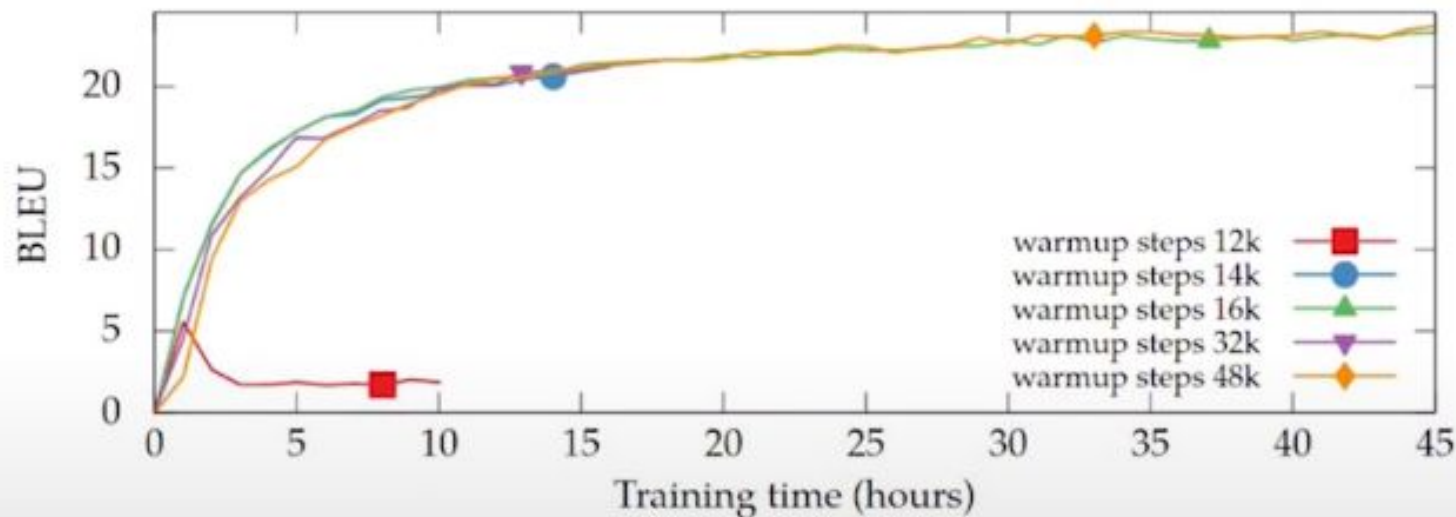
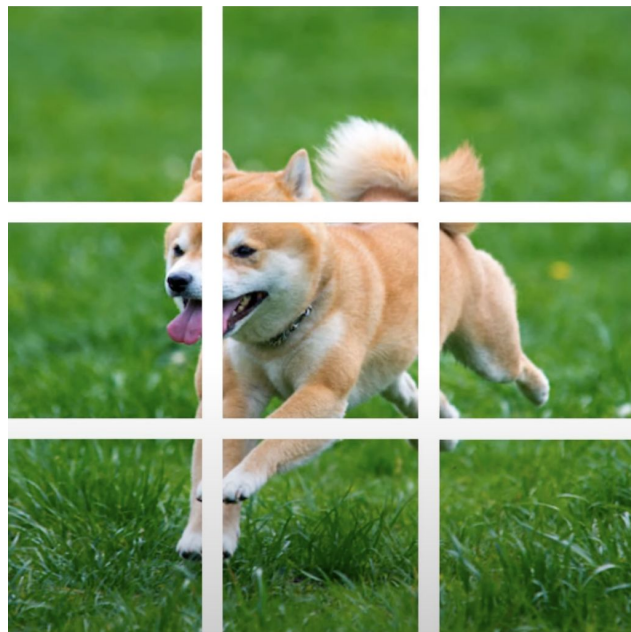
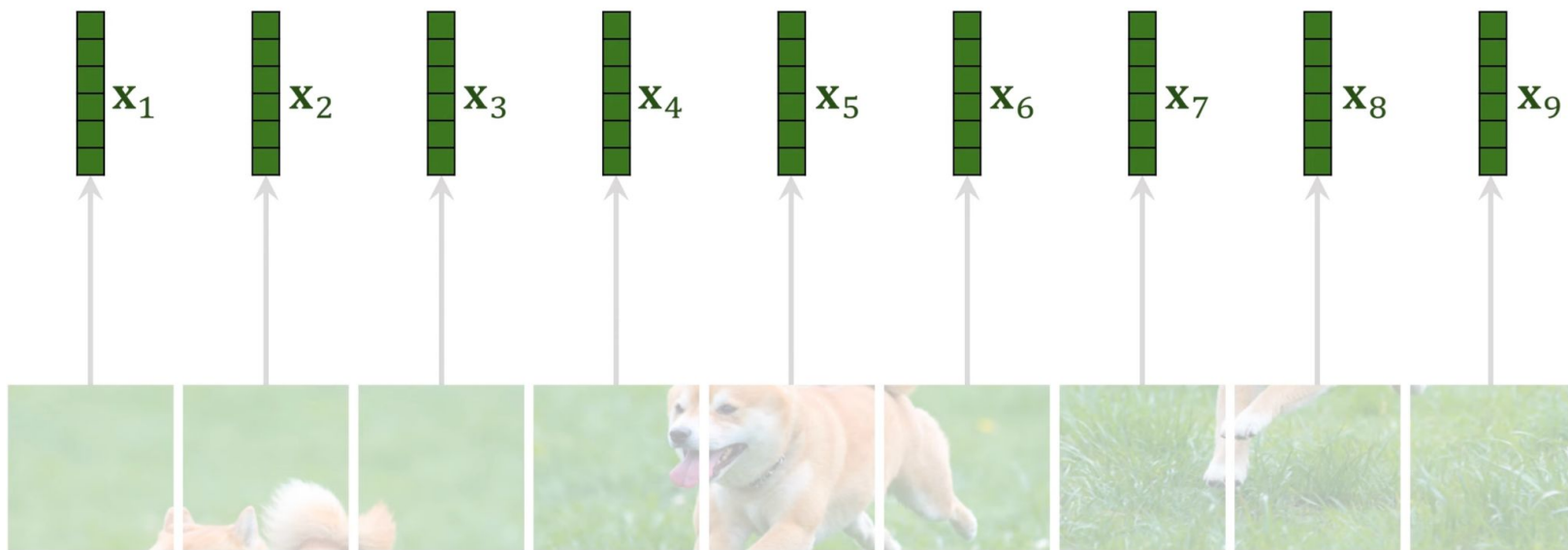


Figure 8: Effect of the warmup steps on a single GPU. All trained on CzEng 1.0 with the default batch size (1500) and learning rate (0.20).

Vision Transformer (ViT)



Vision Transformer (ViT)



Vision Transformer (ViT)

