Neural Nets (NN), глубокое обучение, рекуррентные сети, LSTM и пр. Работа с временными рядами (прогноз) и с текстами. Трансформеры, BERT

Оленев Роман, Самарин Игорь

Санкт-Петербургский государственный университет
Кафедра статистического моделирования

Санкт-Петербург, 2024

NLP

Обработка текстов на естественном языке (Natural Language Processing, NLP) — общее направление искусственного интеллекта и математической лингвистики. Оно изучает проблемы компьютерного анализа и синтеза текстов на естественных языках.
Подходы к представлению текста:

1. Bag of words
2. Bag of Ngramms
3. TF-IDF
4. Word Embeddings

Word2Vec

Thou shalt not make a machine in the likeness of a human mind

thou	shalt	not	make	a	machine	in	the	…

input word	target word
not	thou
not	shalt
not	make
not	a

Figure: SkipGram

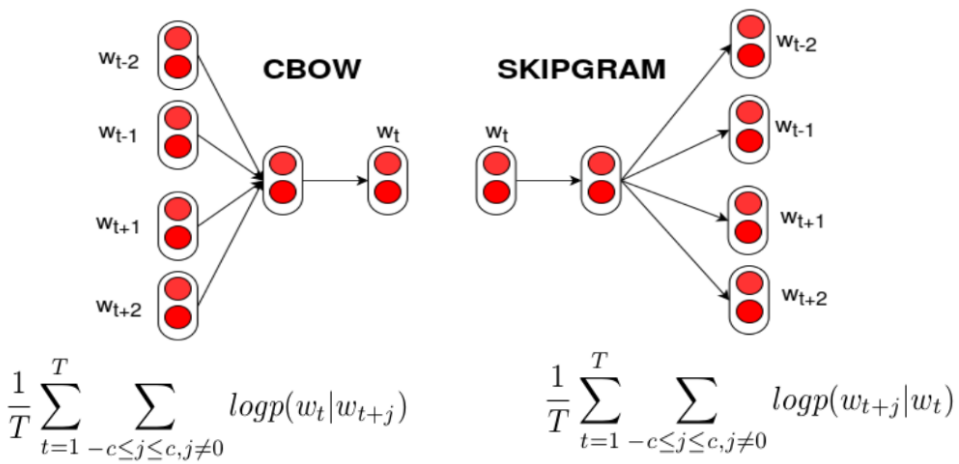
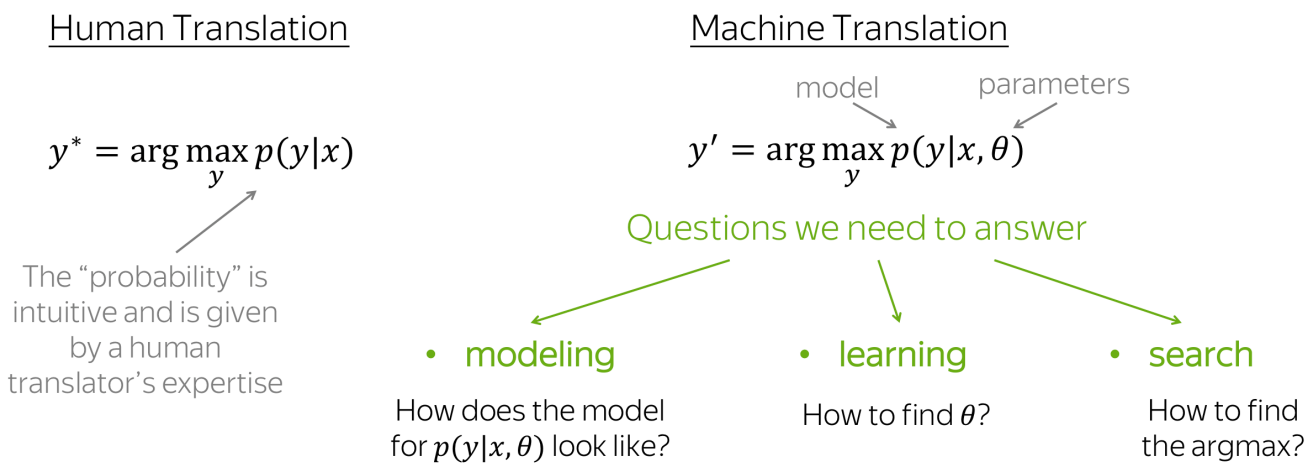


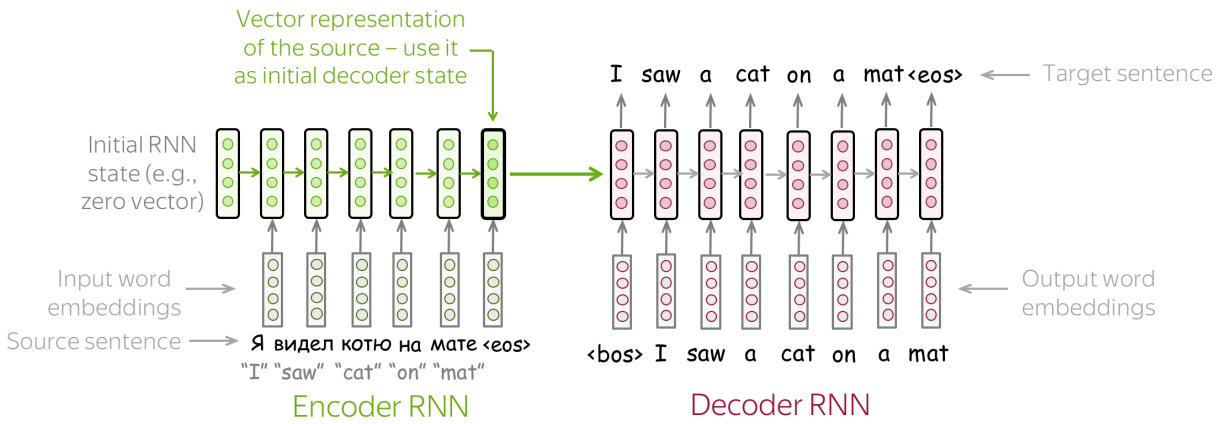
Figure: Word2Vec

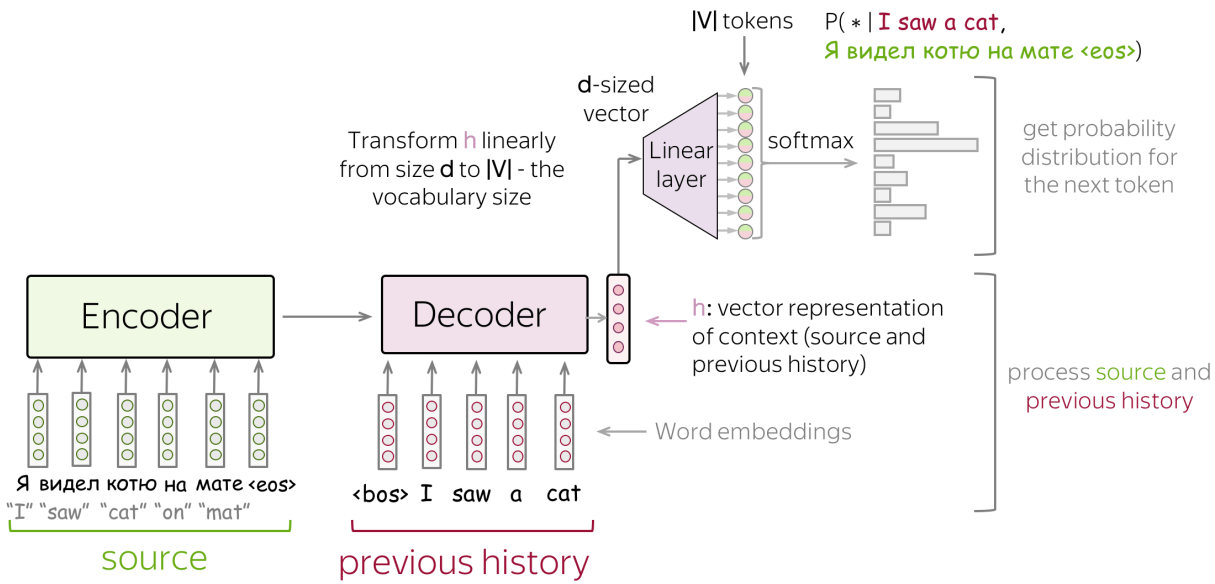
Машинный перевод. Постановка задачи

Пусть есть входная последовательность $x_1, x_2, \ldots, x_m$ и выходная последовательность $y_1, y_2, \ldots, y_n$. Тогда хотим максимизировать $p(y|x)$: $y^* = \arg\max_y p(y|x)$.



Машинный перевод. 2 RNNs





Машинный перевод. Обучение и предсказание

В качестве функции потерь используется кросс–энтропия. При обучении на вход случайным образом можно подавать не только предсказанное на предыдущем шаге слово, но и верное слово.

$$Loss(p^*, p) = -p^* \log(p) = -\sum_{i=1}^{|V|} p_i^* \log(p_i).$$

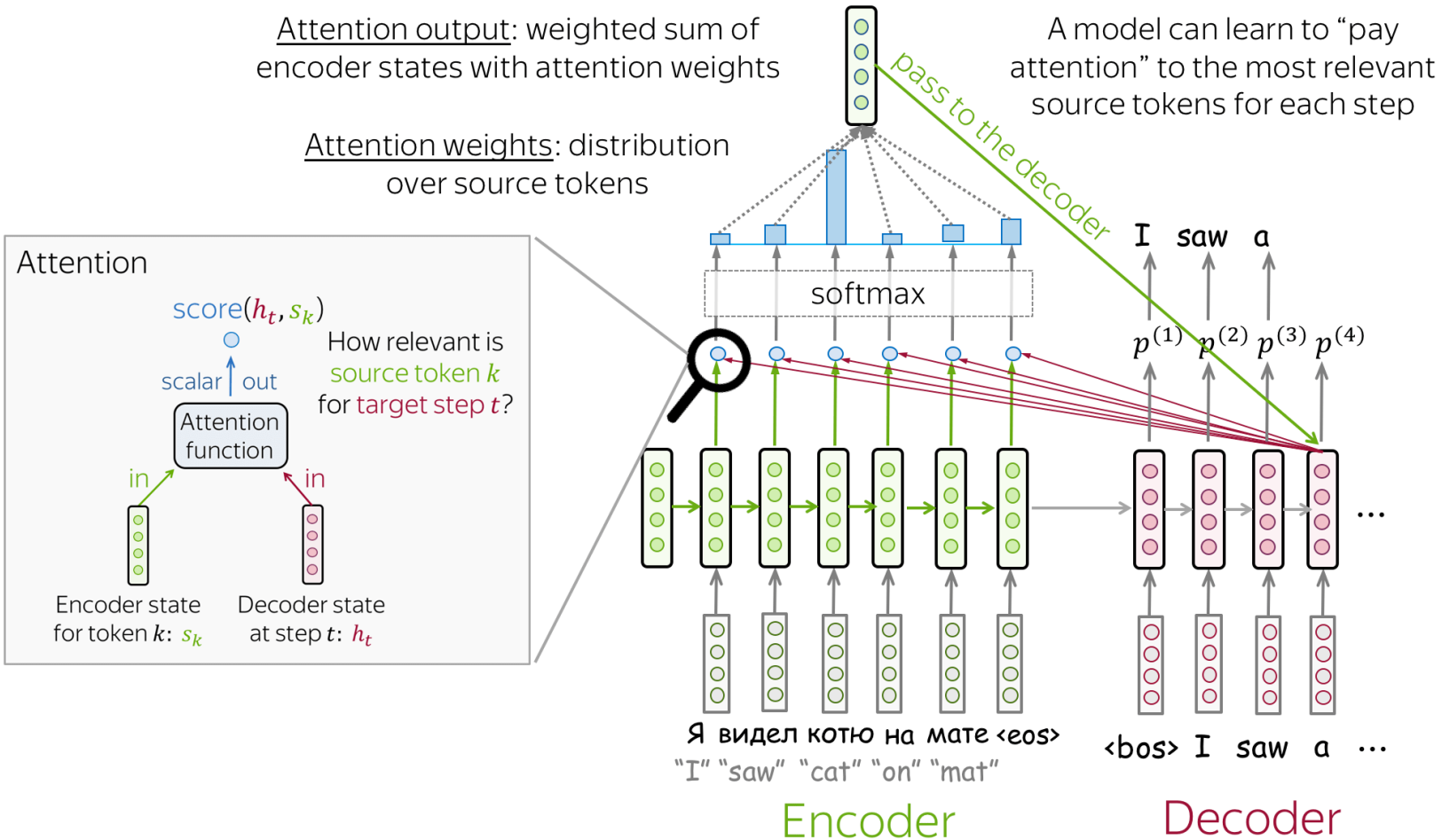
$$Loss(p^*, p) = -\log(p_{y_t}) = -\log(p(y_t | y_{<t}, x)).$$

Прогноз:

1. Greedy Decoding (предсказываем самое вероятное)
2. Beam Search (на каждом шаге храним несколько вероятных последовательностей)

Механизм Attention

Проблема: невозможно сохранить весь контекст в одном векторе. Решением проблемы является механизм внимания.



Механизм Attention

Attention output

$$c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \cdots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$$

"source context for decoder step t"

(weighted sum)

Attention weights

$$a_k^{(t)} = \frac{\exp(\text{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\text{score}(h_t, s_i))}, \text{k} = 1..\text{m}$$

"attention weight for source token k at decoder step t"

(softmax)

Attention scores

$$\text{score}(h_t, s_k), \text{k} = 1..\text{m}$$

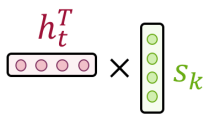
"How relevant is source token k for target step t?"

Attention input

$s_1, s_2, \ldots, s_m$ all encoder states

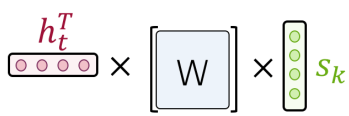
h_t one decoder state

Dot-product



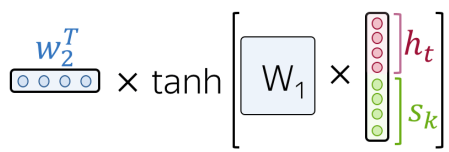
$$\text{score}(h_t, s_k) = h_t^T \, s_k$$

Bilinear



$$\text{score}(h_t, s_k) = h_t^T \, W s_k$$

Multi-Layer Perceptron



$$\text{score}(h_t, s_k) = w_2^T \cdot \tanh(W_1 [h_t, s_k])$$

Трансформеры

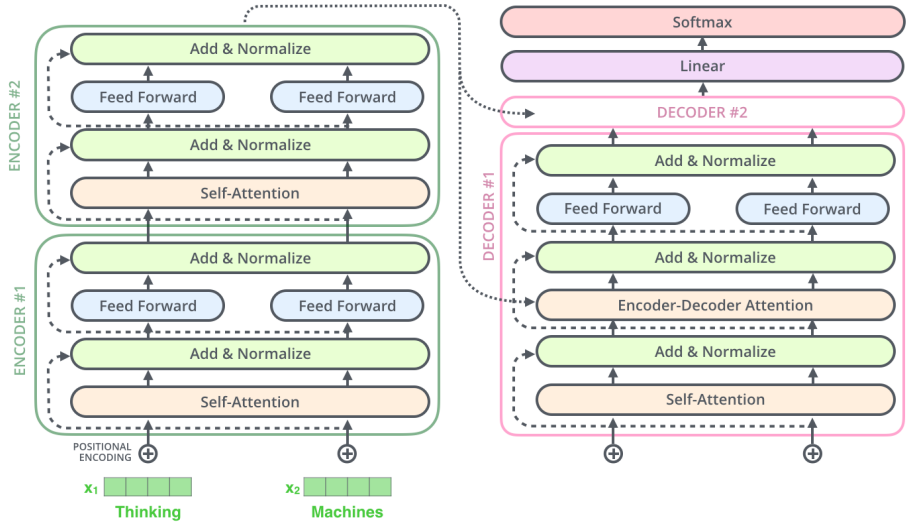
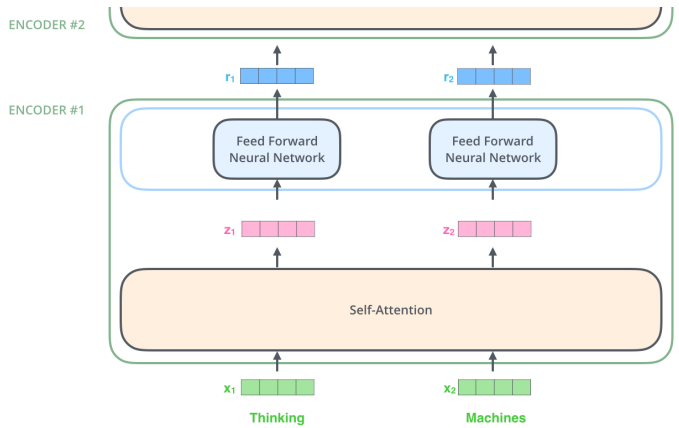
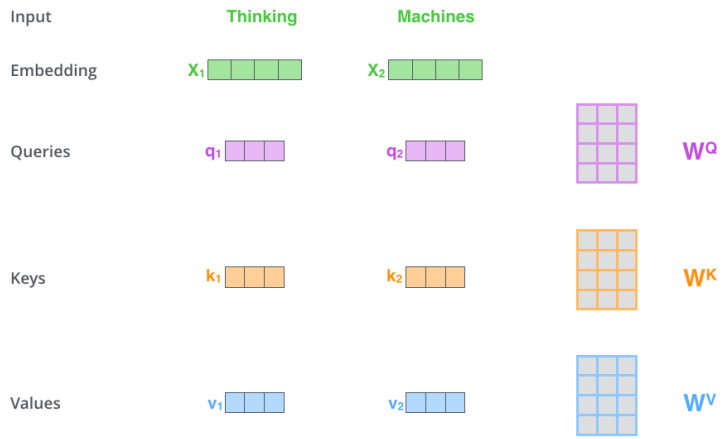
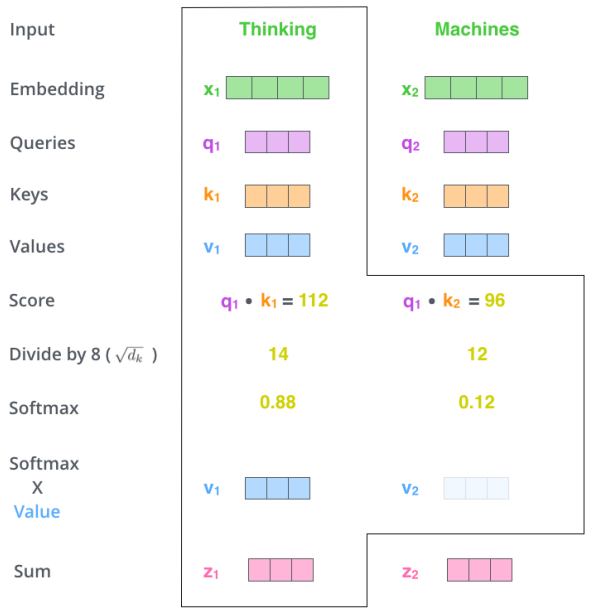


Figure: Архитектура Трансформер

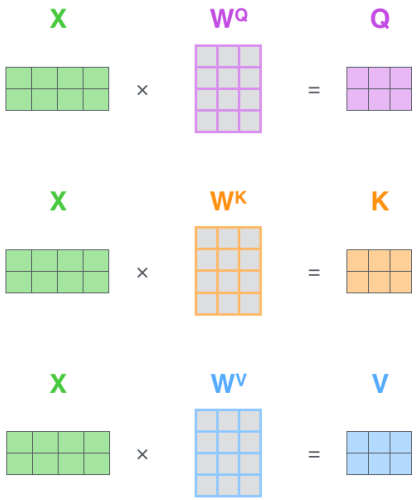


Трансформеры. Self Attention

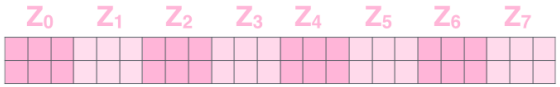




Трансформеры. Self Attention



1) Concatenate all the attention heads



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

X

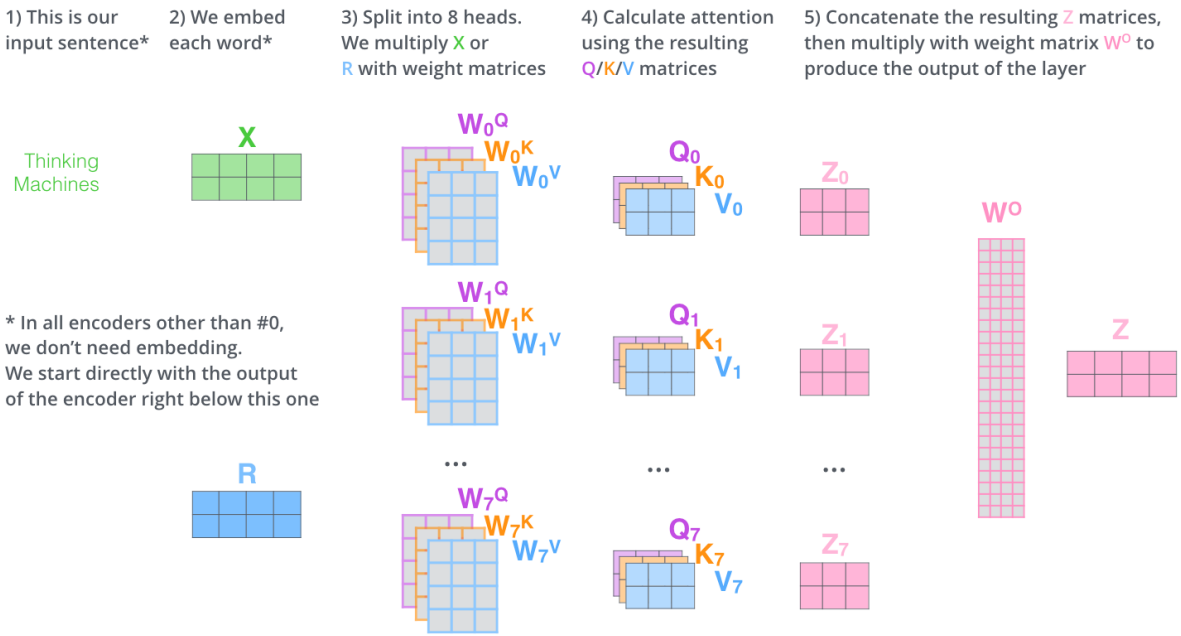
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

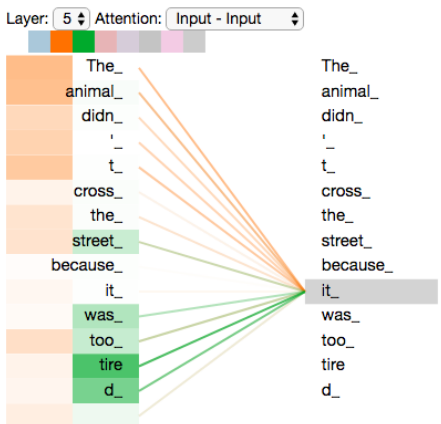
=

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

=

Трансформеры. Self Attention





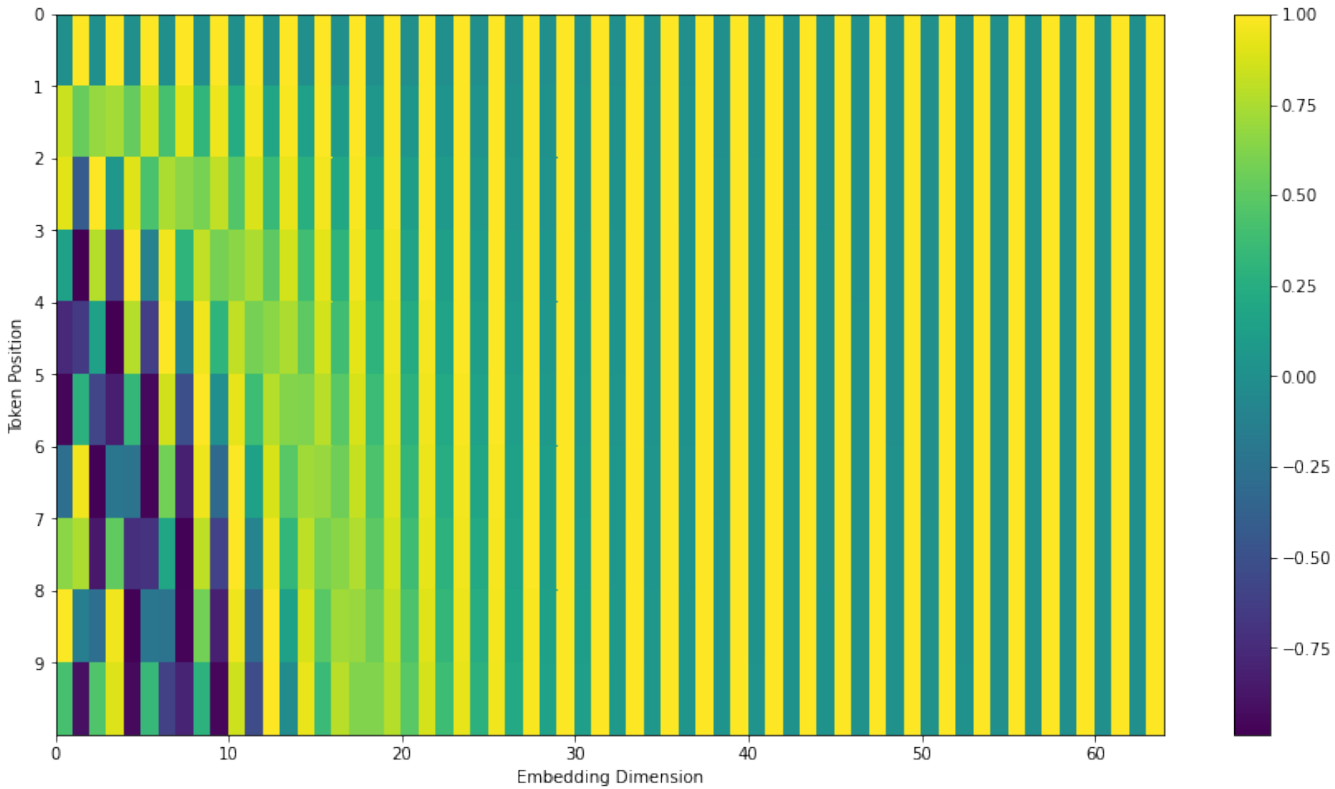
Трансформеры. Positional encoding

Можем закодировать информацию о позиции слова следующим образом:

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

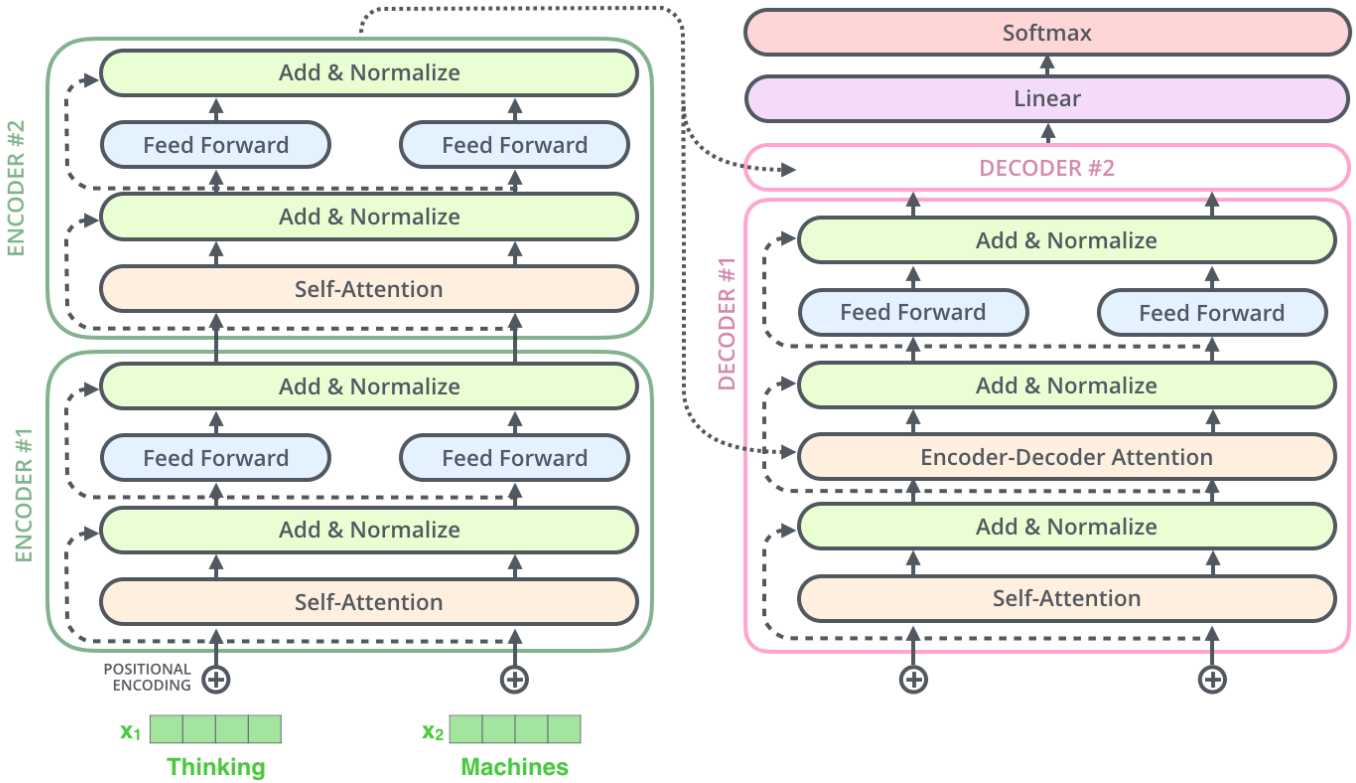
$\text{PE}_{pos,2i+1} = \cos(pos/10000^{2i/d_{model}})$,

после чего можем добавить получившийся энкодинг к эмбеддингам.



Трансформеры. Masked Attention

Входная последовательность декодера маскируется таким образом, что Attention Score для каждого токена считается только с токенами, которые находятся перед ним. Для encoder–decoder attention матрица Q берется из декодера, а матрицы K и V — из энкодера.



Трансформеры. Train & Inference

Последний линейный слой преобразовывает выход декодера из размерности (batch size, sequence lenght, embedding size) в размерность (batch size, sequence length, vocabulary size), после чего считается

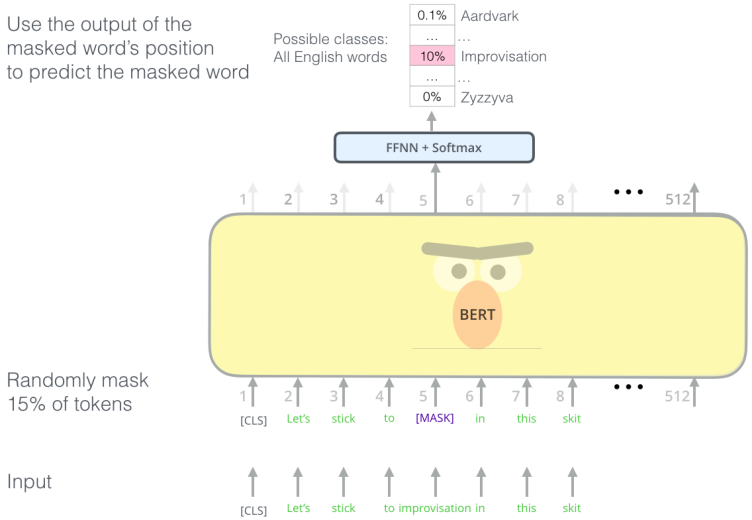
$$Loss = -\sum_{t=1}^{T} \log(p_{y_t}).$$

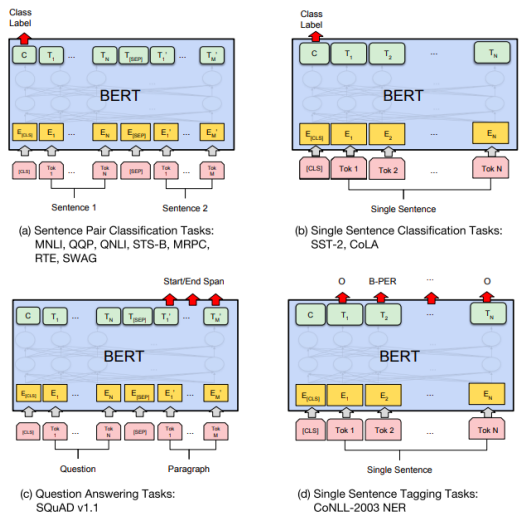
Для прогнозирования подаем декодеру <SOS> (start of sequence) токен, после чего добавляем ко входу декодера предсказанный токен и повторяем процесс до того, как декодер выведет <EOS> (end of sequence) токен.

Преимущества архитектуры:

1. Работает быстрее, так как нет рекуррентных блоков.
2. Много вычислений, которые можно распараллелить.

BERT (Bidirectional Encoder Representations from Transformers)





Transfer Learning