Transformer

NLP

lecturer: Mollaev D. E.
Sber AI Lab

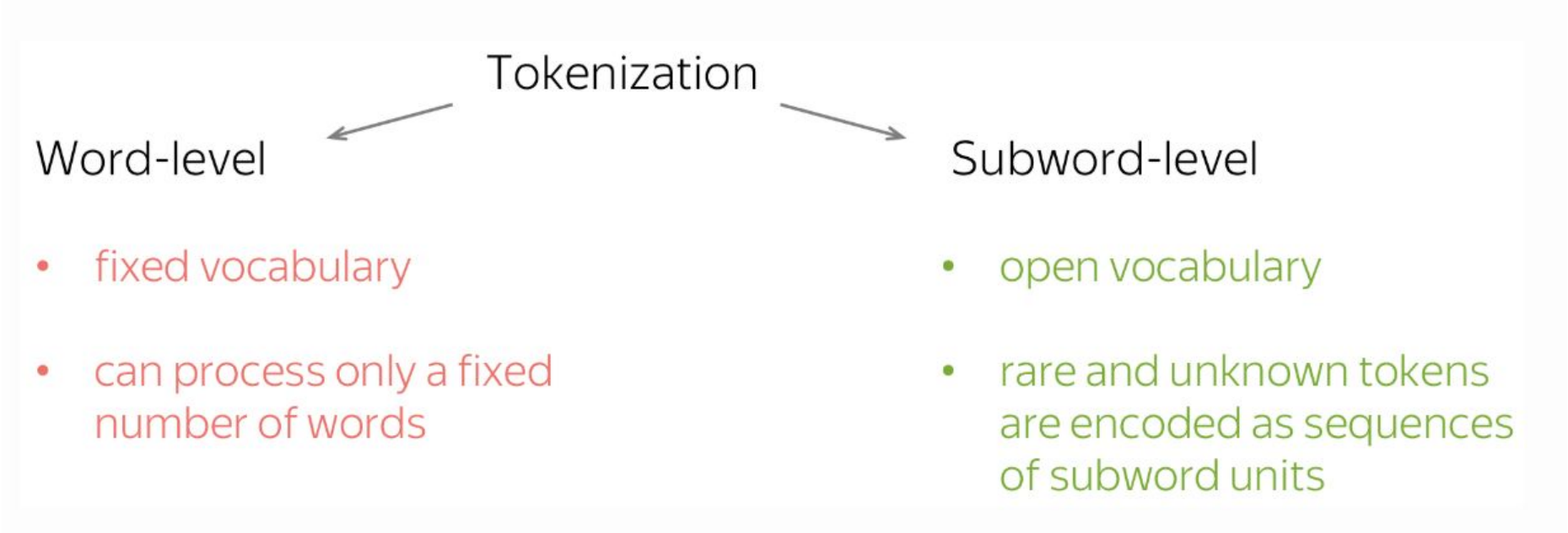
Lecture plan

- Tokenization
- General view:
  - Seq2Seq
  - Ablation: transformer vs rnn with attention
- Transformer: architectures
  - Self-Attention module
  - Feed forward module
  - Positional encoding
  - Other important parts
- Training Details
- BERT-like
- GPT-like

Lecture plan

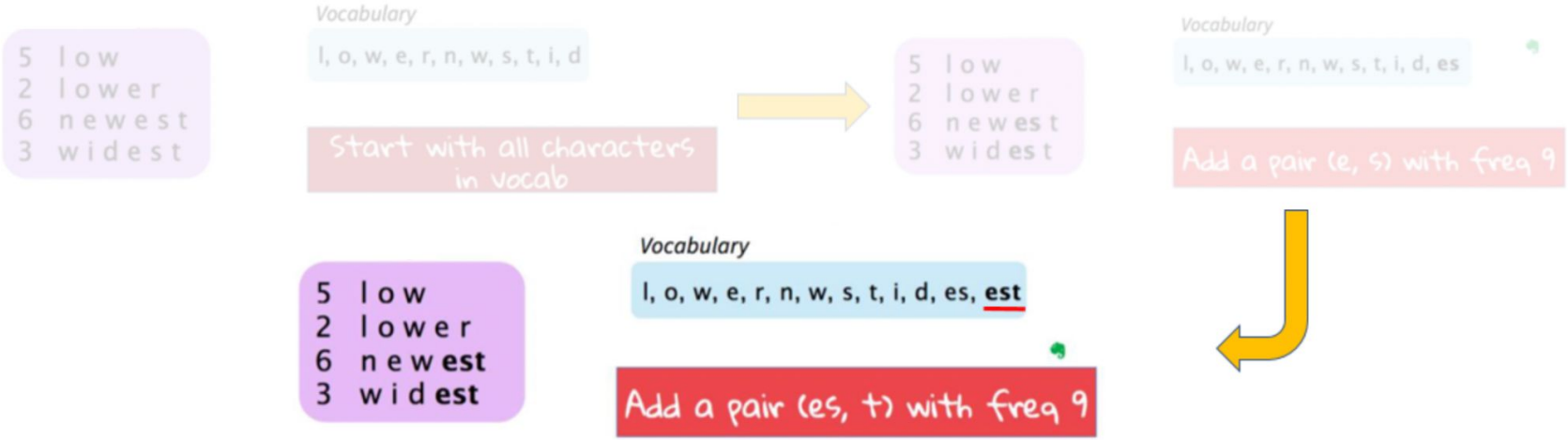
- Tokenization
- General view:
  - Seq2Seq
  - Ablation: transformer vs rnn with attention
- Transformer: architectures
  - Self-Attention module
  - Feed forward module
  - Positional encoding
  - Other important parts
- Training Details
- BERT-like
- GPT-like

Tokenization



Tokenization: BPE - training

From individual symbols, whole words and N-grams– to most common co-occurrences



To build a tokenizer vocabulary:

- Initialize a vocabulary with all unique symbols from a dataset
- Build a character co-occurrences model
- Pick the most common pair
- Repeat...

Tokenization: BPE - inference

The algorithm starts with segmenting a word into a sequence of characters. After that, it iteratively makes the following two steps until no merge it possible:

- among all possible merges at this step, find the highest merge in the table;
- apply this merge.

Lecture plan

- Tokenization
- General view:
  - Seq2Seq
  - Ablation: transformer vs rnn with attention
- Transformer: architectures
  - Self-Attention module
  - Feed forward module
  - Positional encoding
  - Other important parts
- Training Details
- BERT-like
- GPT-like

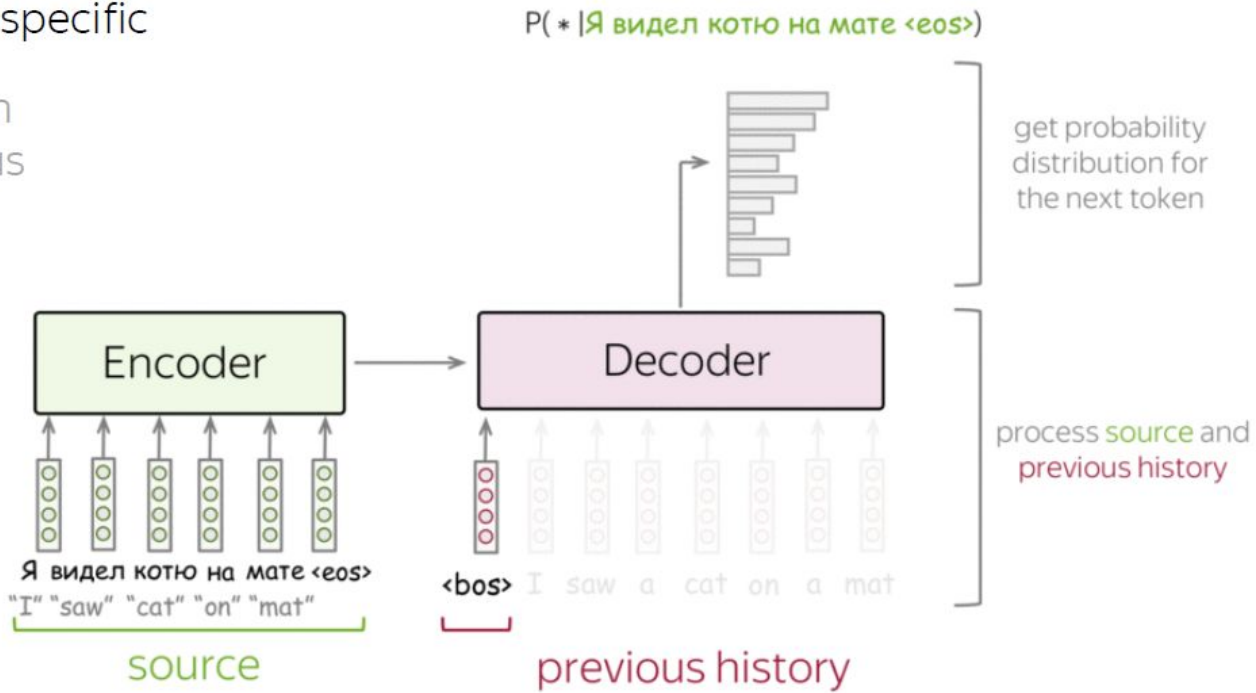
General View

- process context – model-specific

  Get vector representation of the source and previous target tokens

- evaluate probabilities – model-agnostic

  Predict probability distribution for the next target token



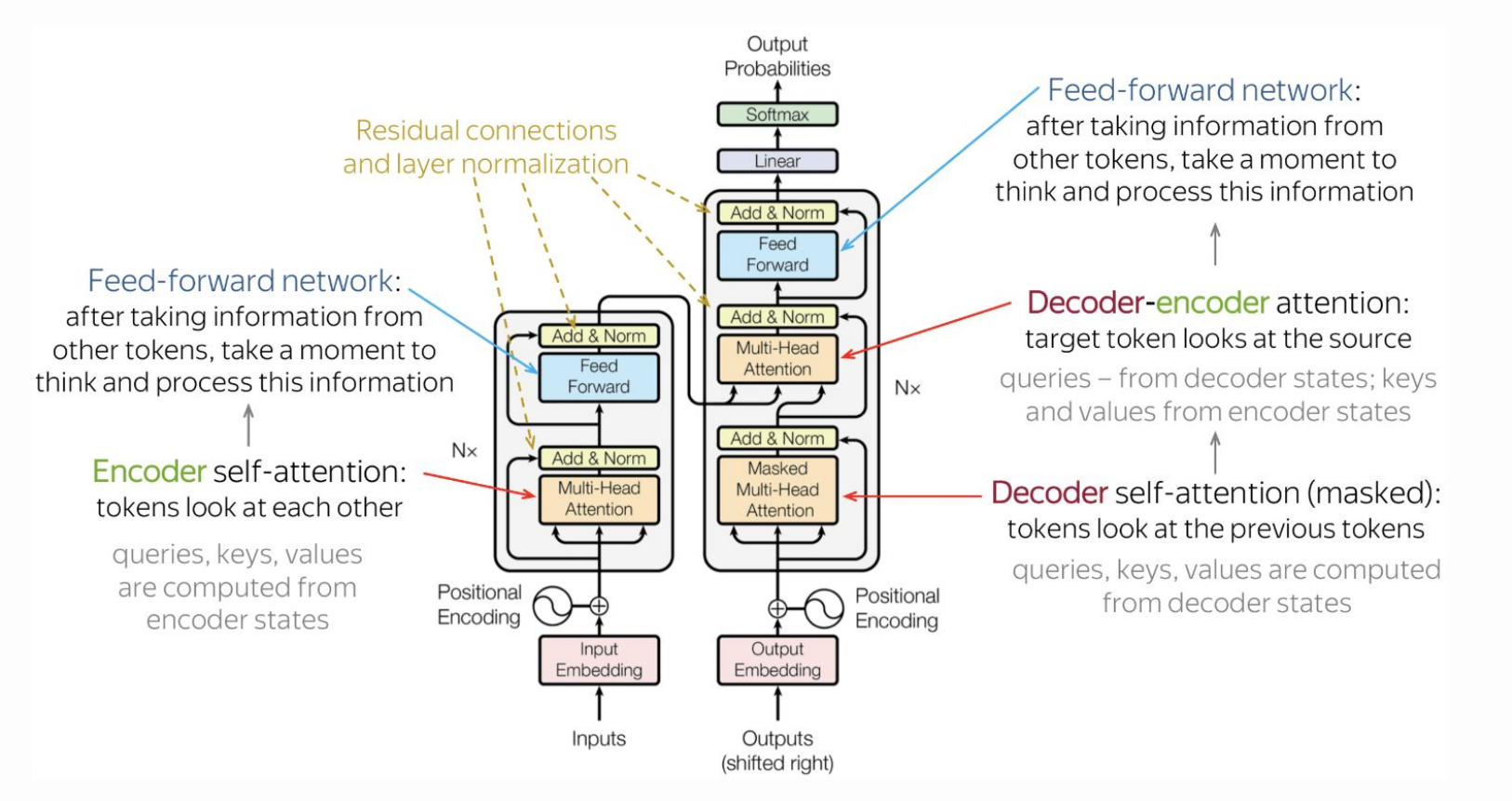
Transformer: Attention is All You Need

	Seq2seq without attention	Seq2seq with attention	Transformer
processing within encoder	RNN/CNN	RNN/CNN	attention
processing within decoder	RNN/CNN	RNN/CNN	attention
decoder-encoder interaction	static fixed-sized vector	attention	attention

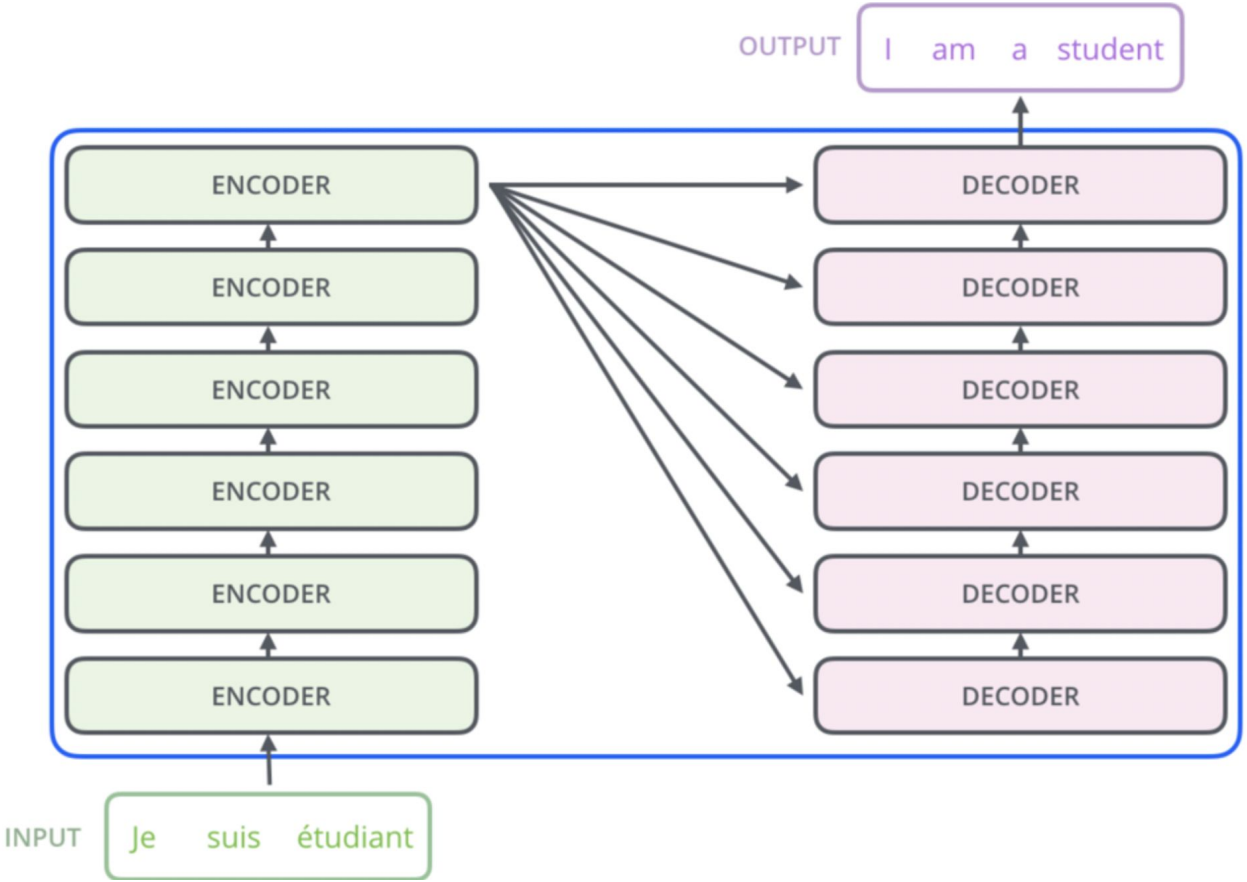
Lecture plan

- Tokenization
- General view:
  - Seq2Seq
  - Ablation: transformer vs rnn with attention
- Transformer: architectures
  - Self-Attention module
  - Feed forward module
  - Positional encoding
  - Other important parts
- Training Details
- BERT-like
- GPT-like

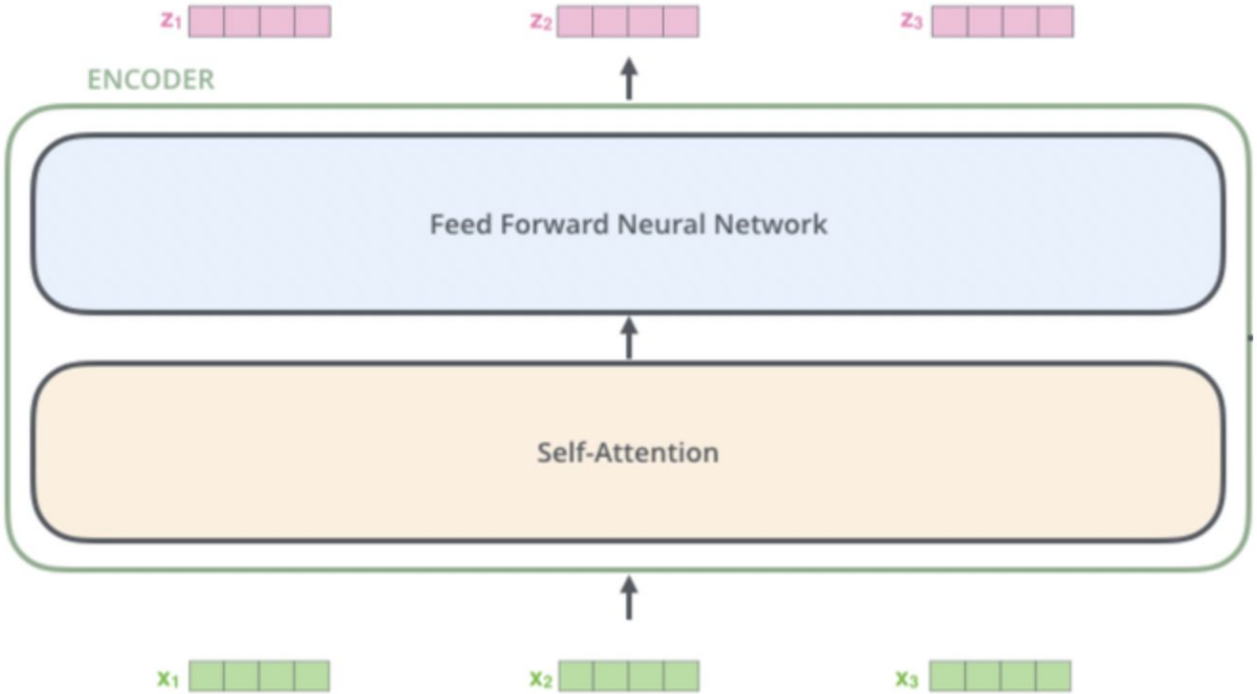
Transformer: overview



Transformer: encoder, decoder



Transformer: encoder blocks



Lecture plan

- Tokenization
- General view:
  - Seq2Seq
  - Ablation: transformer vs rnn with attention
- Transformer: architectures
  - Self-Attention module ⇦
  - Feed forward module
  - Positional encoding
  - Other important parts
- Training Details
- BERT-like
- GPT-like

Transformer: self-attention

Each vector receives three representations (“roles”)

$W_Q \times = =$ Query: vector **from** which the attention is looking

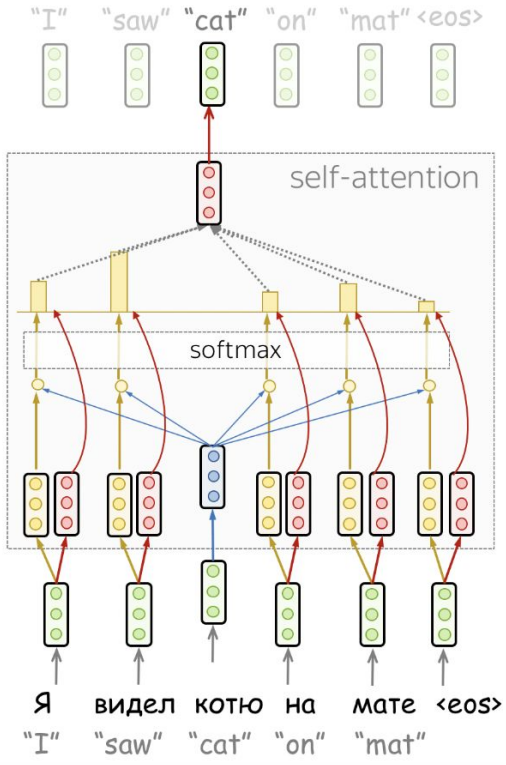
“Hey there, do you have this information?”

$W_K \times = =$ Key: vector **at** which the query looks to compute weights

“Hi, I have this information – give me a large weight!”

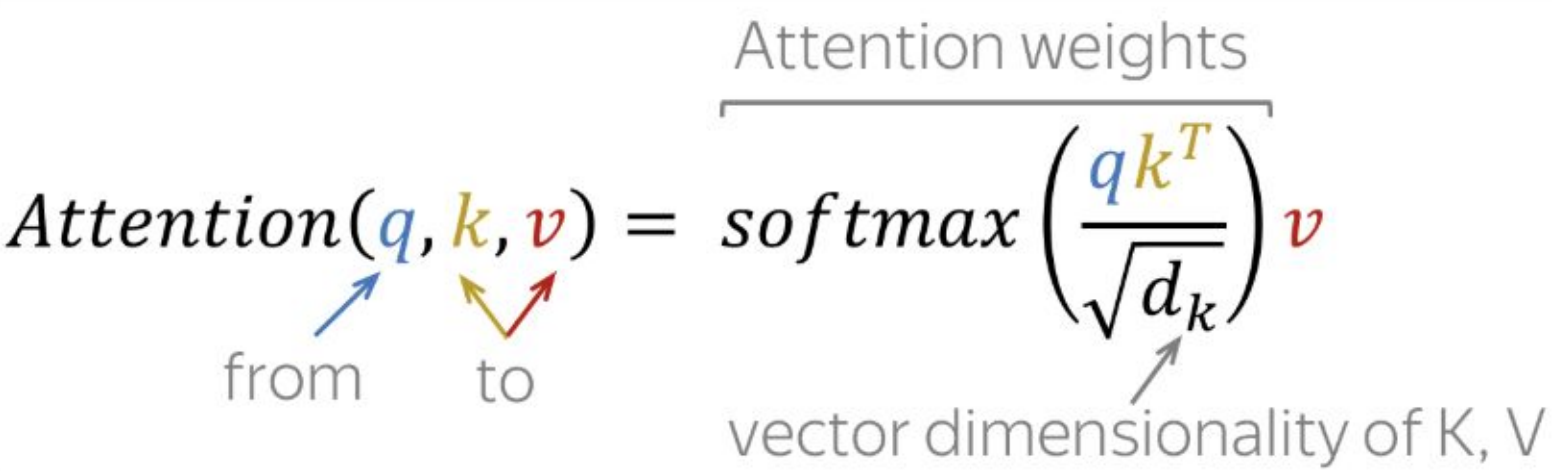
$W_V \times = =$ Value: their weighted sum is attention output

“Here’s the information I have!”

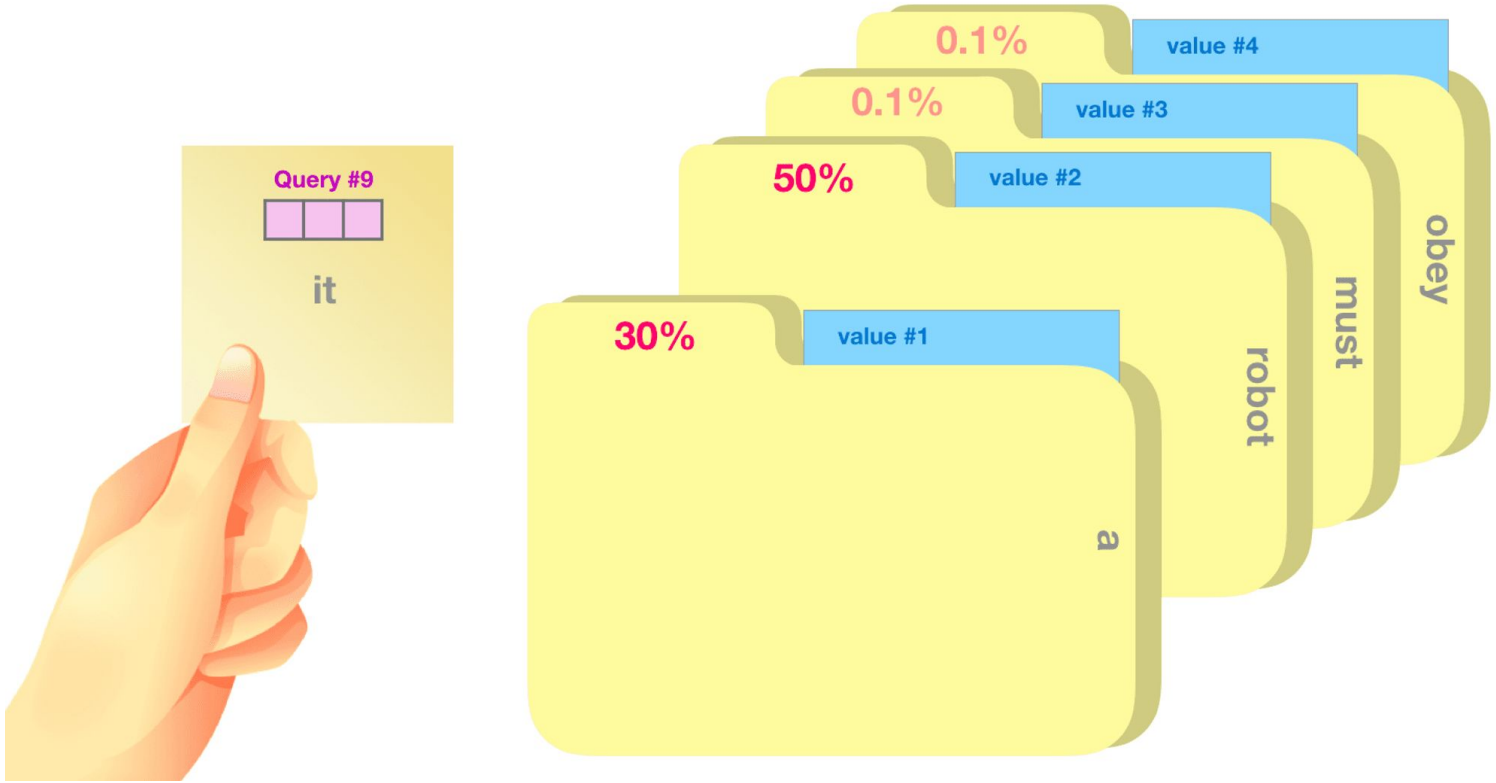


- **query** - asking for information
- **key** - saying that it has some information
- **value -** giving the information

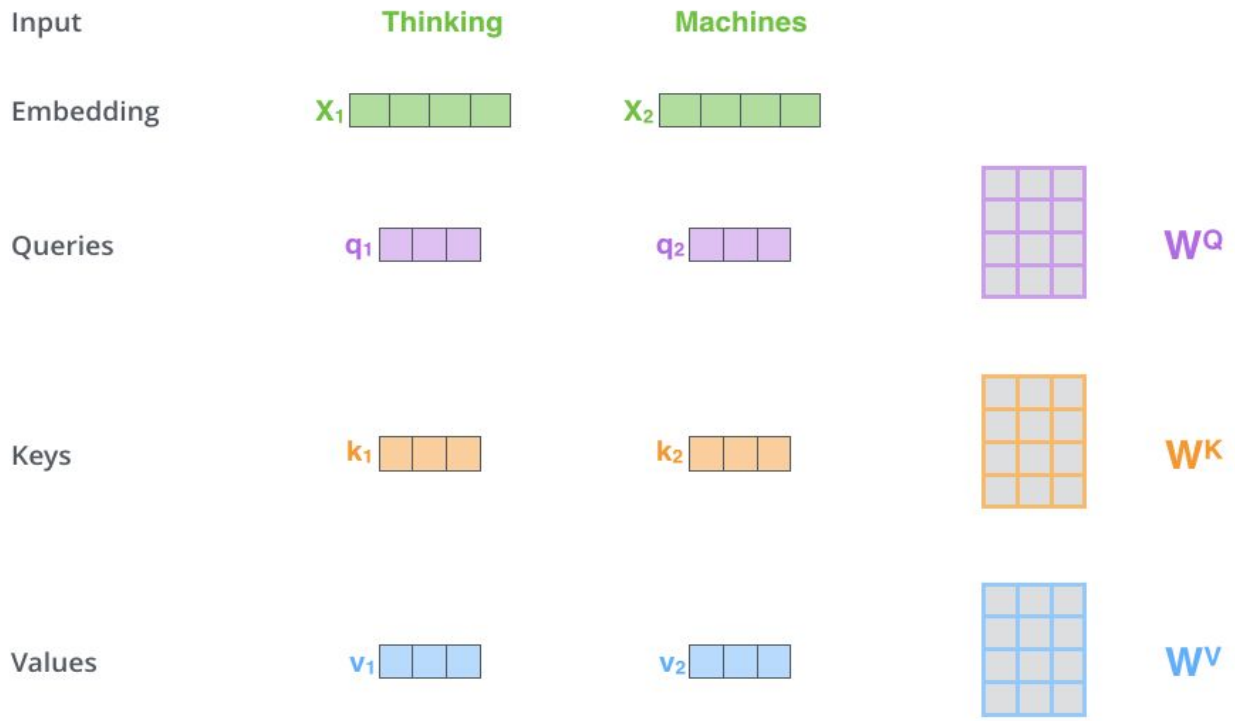
Transformer: self-attention



Transformer: self-attention

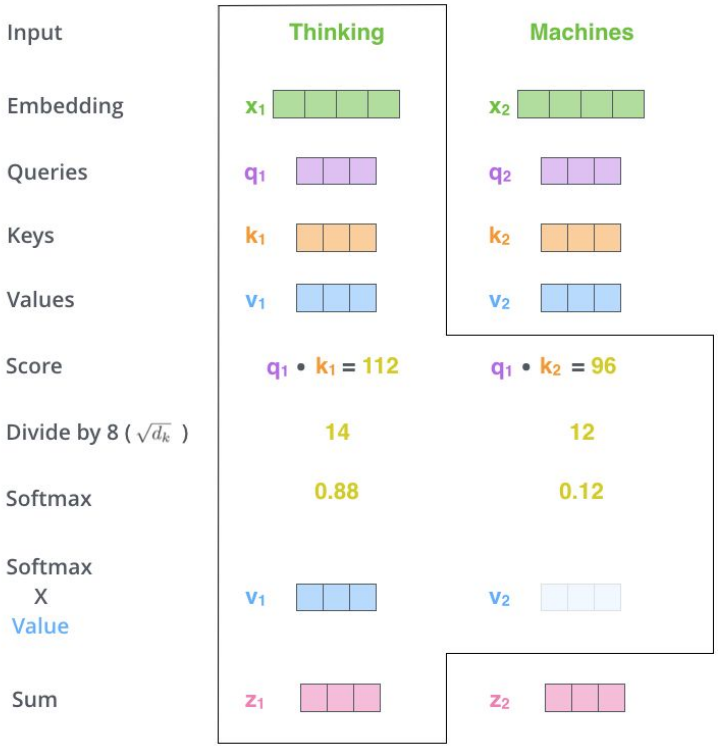


Transformer: self-attention



Multiplying x1 by the WQ weight matrix produces q1, the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

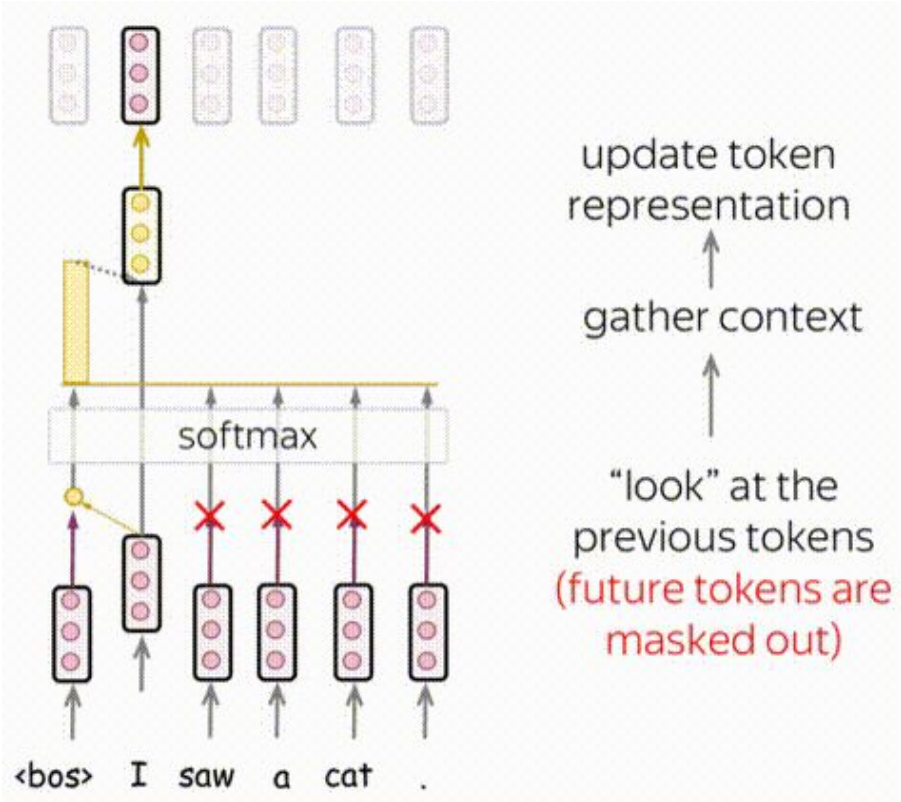
Transformer: self-attention



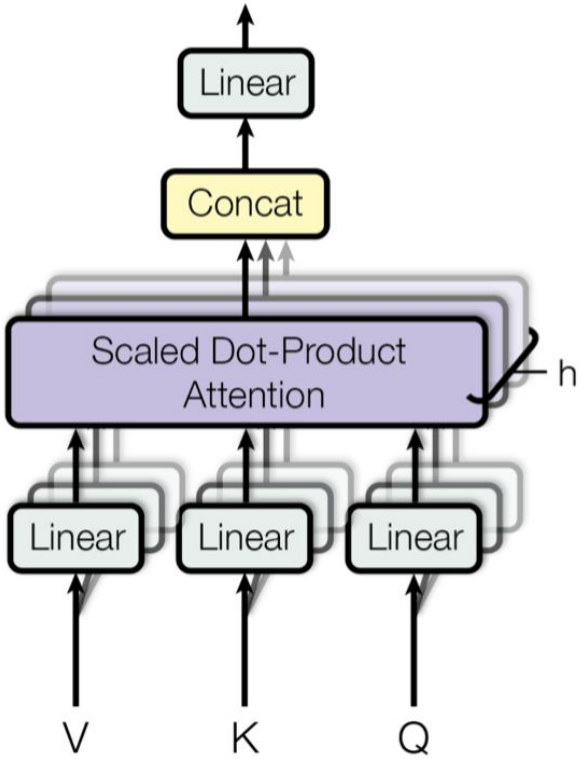
- For a vector **X**, compute **Q, K,** and **V** by multiplying learnable matrices $\mathbf{W_{Q/K/V}}$
- For a fixed $\boldsymbol{q_i}$, compute attention **score** by matching it against **K**
- Assemble result: retrieve a **value** with respect to its **score**

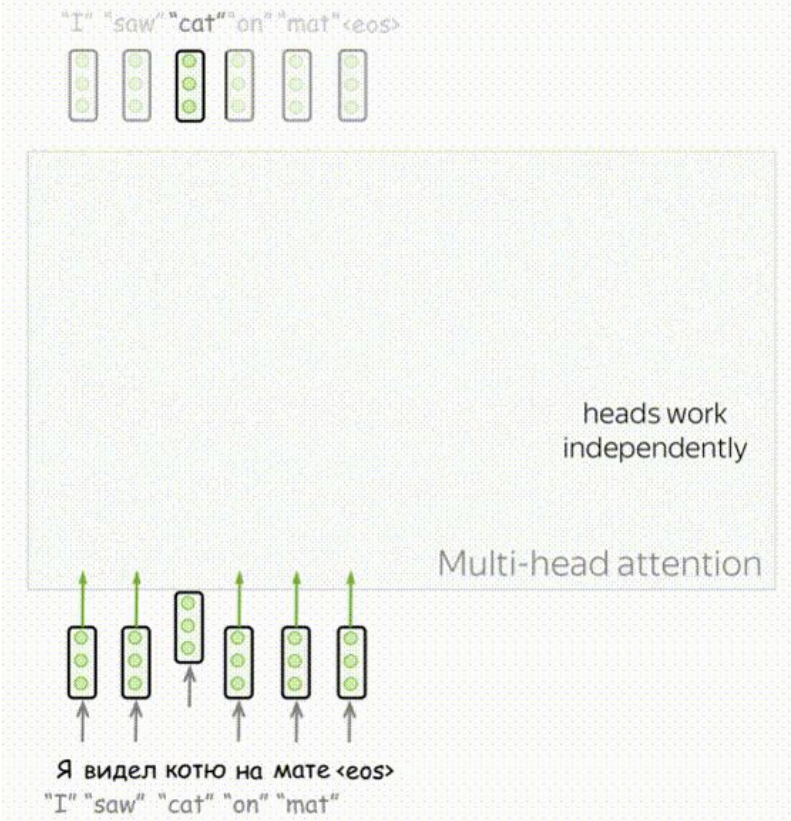
Transformer: masked self-attention

To forbid the decoder to look ahead, the model uses masked self-attention: future tokens are masked out.



Transformer: multi-head self-attention





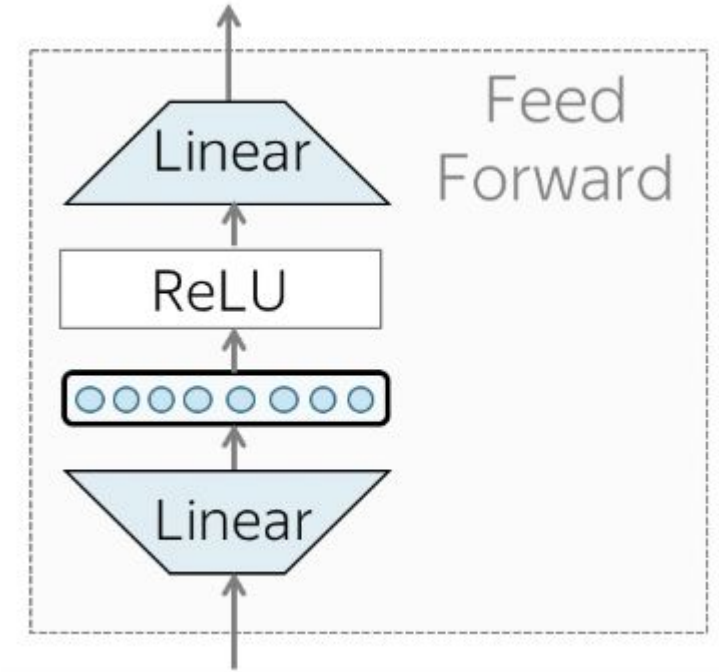
Lecture plan

- Tokenization
- General view:
  - Seq2Seq
  - Ablation: transformer vs rnn with attention
- Transformer: architectures
  - Self-Attention module
  - Feed forward module ⟸
  - Positional encoding
  - Other important parts
- Training Details
- BERT-like
- GPT-like

Transformer: Feedforward block

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2.$$

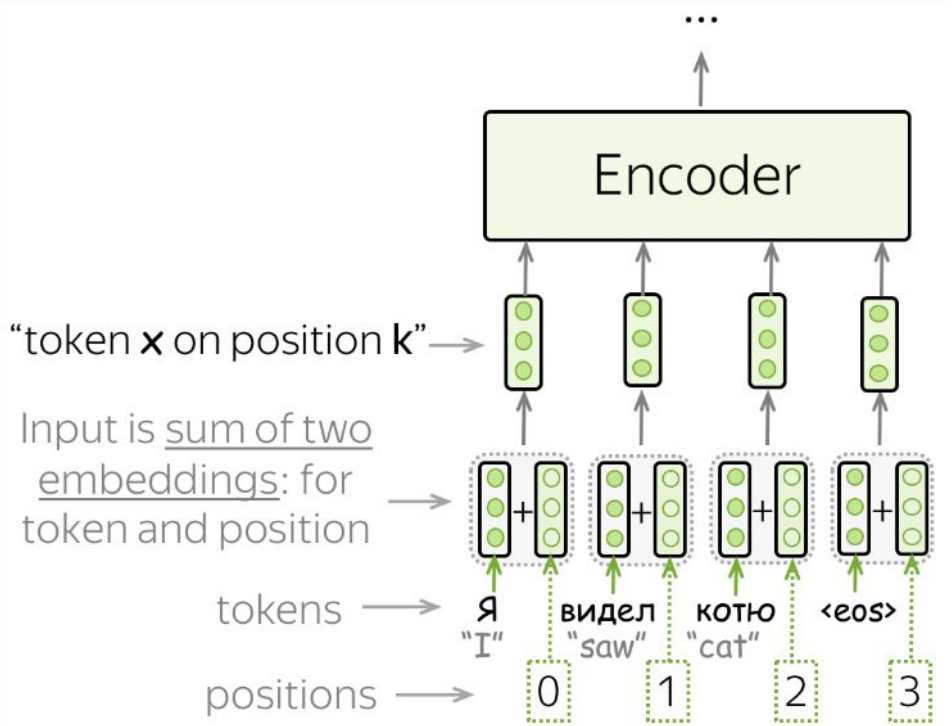
- **Attention** - "look at other tokens and gather information"
- **FFN** - "take a moment to think and process this information"



Lecture plan

- Tokenization
- General view:
  - Seq2Seq
  - Ablation: transformer vs rnn with attention
- Transformer: architectures
  - Self-Attention module
  - Feed forward module
  - Positional encoding ⟸
  - Other important parts
- Training Details
- BERT-like
- GPT-like

Transformer: positional encoding



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

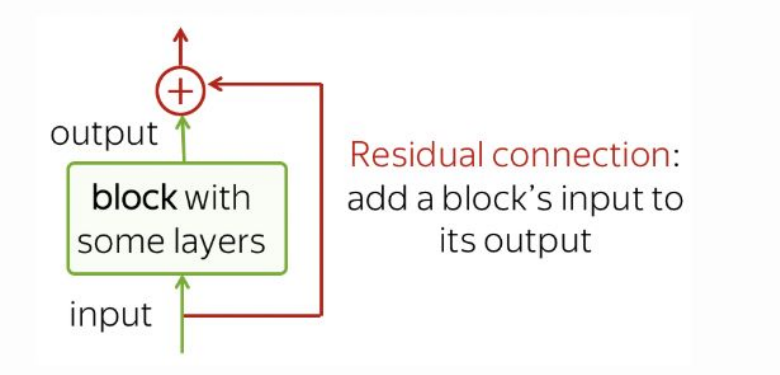
$$\mathrm{PE}_{pos,2i+1} = \cos(pos/10000^{2i/d_{model}}),$$

Lecture plan

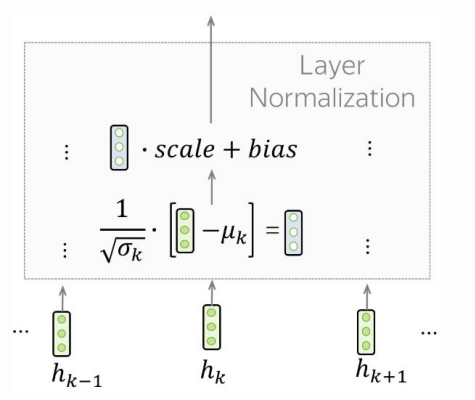
- Tokenization
- General view:
  - Seq2Seq
  - Ablation: transformer vs rnn with attention
- Transformer: architectures
  - Self-Attention module
  - Feed forward module
  - Positional encoding
  - Other important parts ⟸
- Training Details
- BERT-like
- GPT-like

Transformer: Other important parts

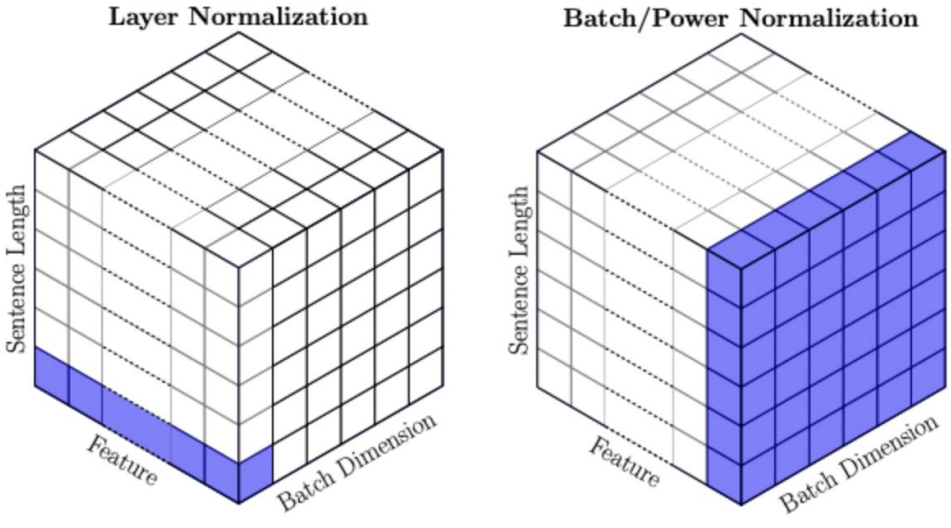
Residual connection



Layer norm

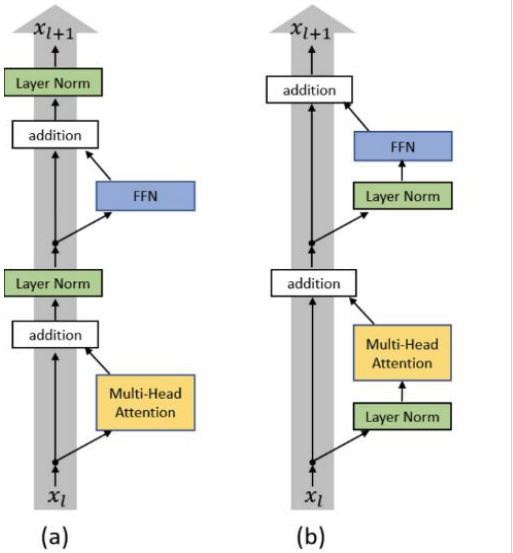


Transformer: layer norm



- **Layer norm**: По каждому отдельному примеру: нормализует значения внутри одного слоя.
- **Batch norm**: По мини-батчу: для каждого признака вычисляет среднее и стандартное отклонение по всему батчу.

Transformer: layer norm

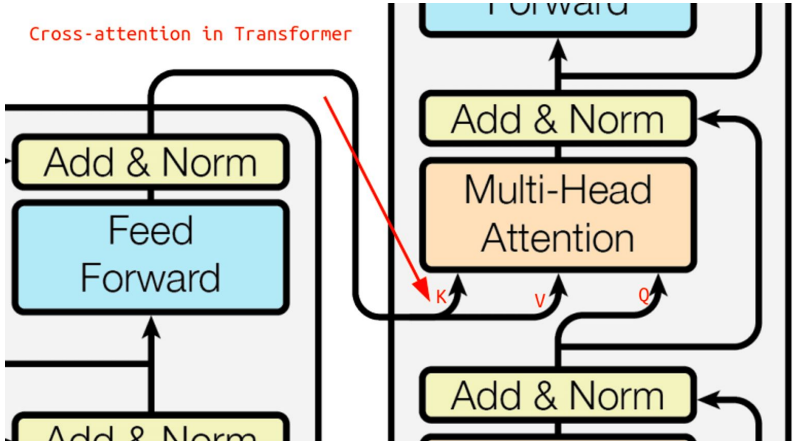


Оригинальная версия из *Attention Is All You Need* использует **Post-LN**, однако современные LLM без исключения строятся на **Pre-LN**. Разница в их поведении заключается в следующем:

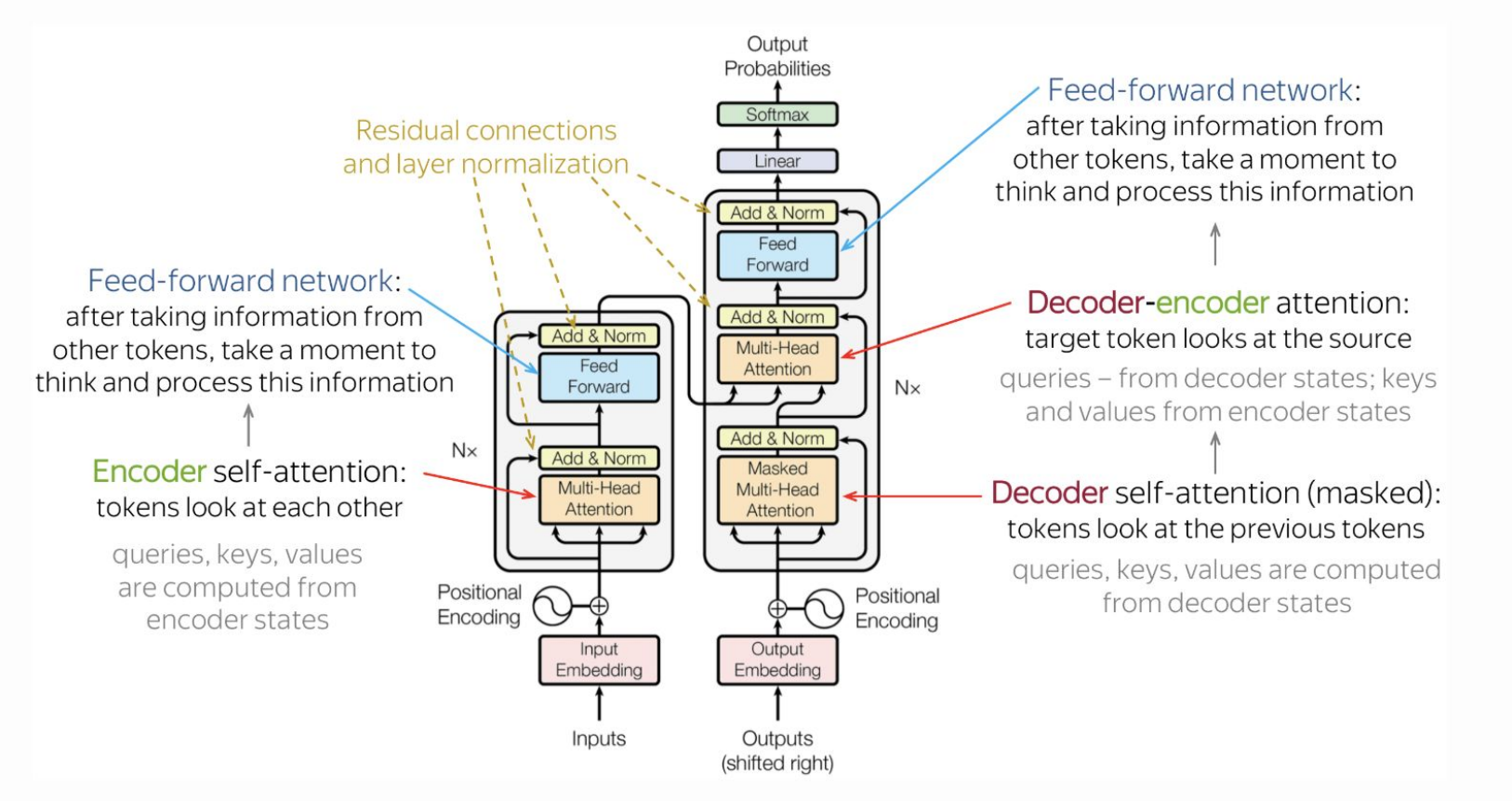
- **Post-LN** показывает хорошие результаты на небольших трансформерах (до 6 слоёв), но при увеличении глубины обучения становится нестабильным и может расходиться.
- **Pre-LN** обеспечивает более стабильное обучение, что делает его предпочтительным для глубоких моделей.

Decoder

- Cross-Attention: **K** and **V** are coming from encoder, **Q** is from previous layer
- Masked MHA: assign large negative number to any token from the future



Transformer: overview



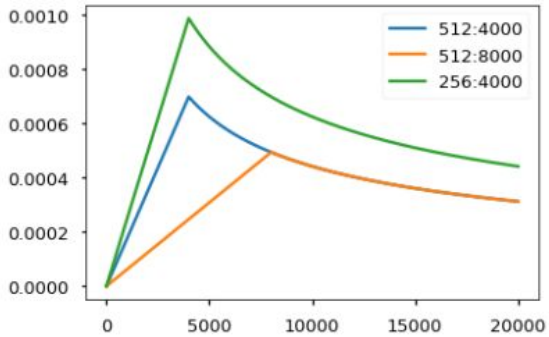
Lecture plan

- Tokenization
- General view:
  - Seq2Seq
  - Ablation: transformer vs rnn with attention
- Transformer: architectures
  - Self-Attention module
  - Feed forward module
  - Positional encoding
  - Other important parts
- Training Details and Hyperparameters:
- BERT-like
- GPT-like

Training details and hyperparameters

- Cross-Entropy Loss $NLL(y_{1:M}) = -\sum_{t=1}^{M} \log p(y_t|t_{t-1})$
- Teacher Forcing for decoder
- Adam Optimizer
- lr with warm-up scheduling $lr = d_{model}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup^{-1.5})$
- BPE

lr schedule



Hypers:

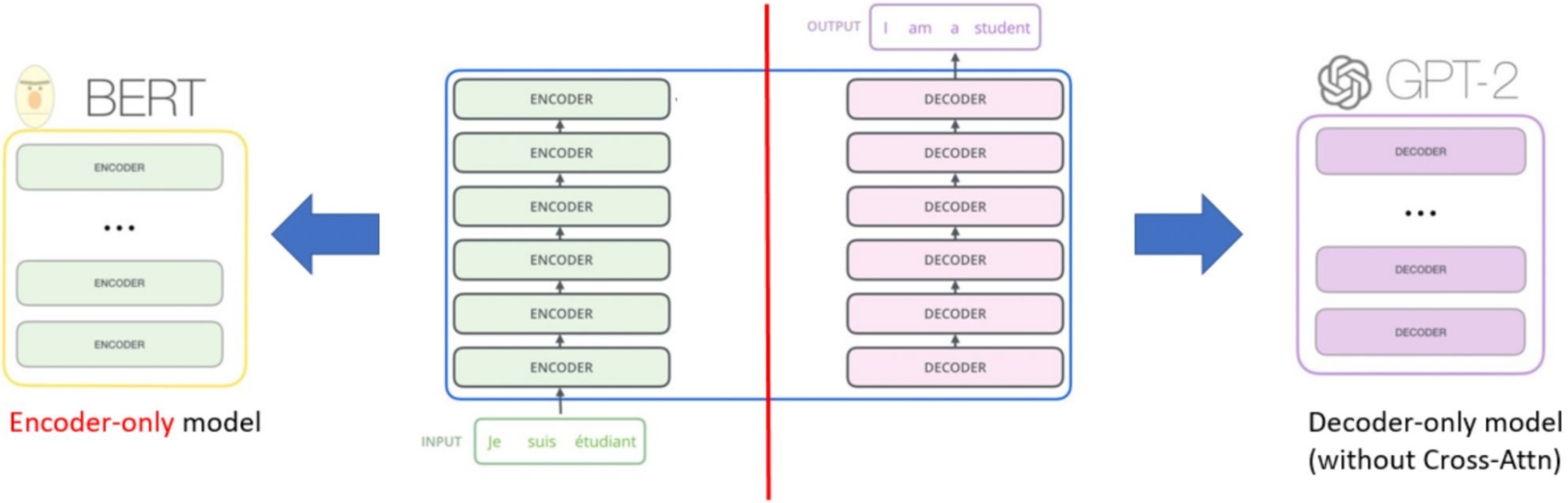
- n_blocks (6 blocks)
- n_head (8 heads)
- n_dim (512)

Lecture plan

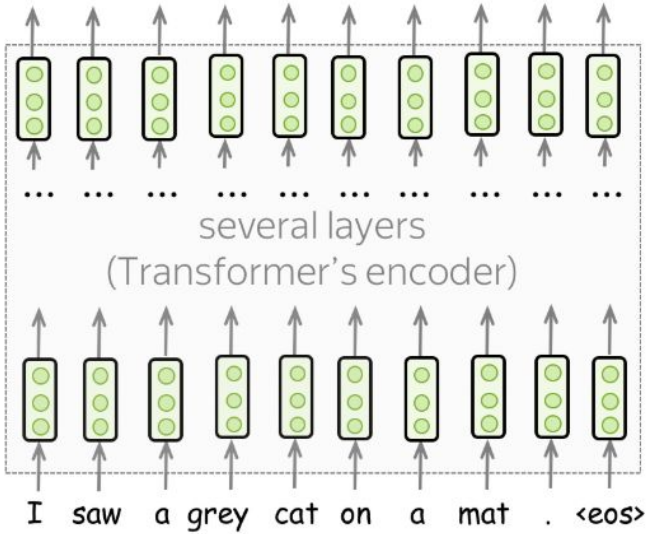
- Tokenization
- General view:
  - Seq2Seq
  - Ablation: transformer vs rnn with attention
- Transformer: architectures
  - Self-Attention module
  - Feed forward module
  - Positional encoding
  - Other important parts
- Training Details
- BERT-like
- GPT-like

BERT vs GPT

- Transformer (invented for translation task):



BERT (Bidirectional Encoder Representations from Transformers): overview



Model architecture:

- Transformer's encoder

What is special about it:

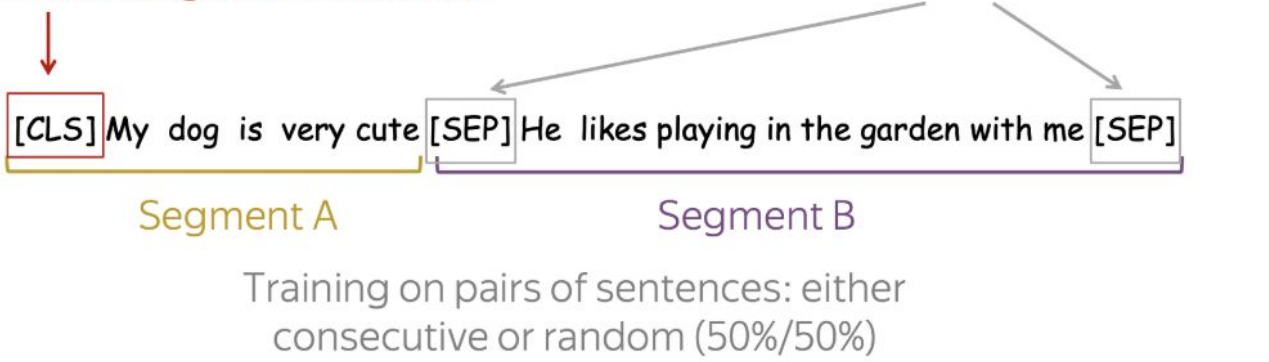
- Training objectives
  - MLM: Masked language modeling
  - NSP: Next sentence prediction
- The way it is used
  - No task-specific models

BERT - Training input

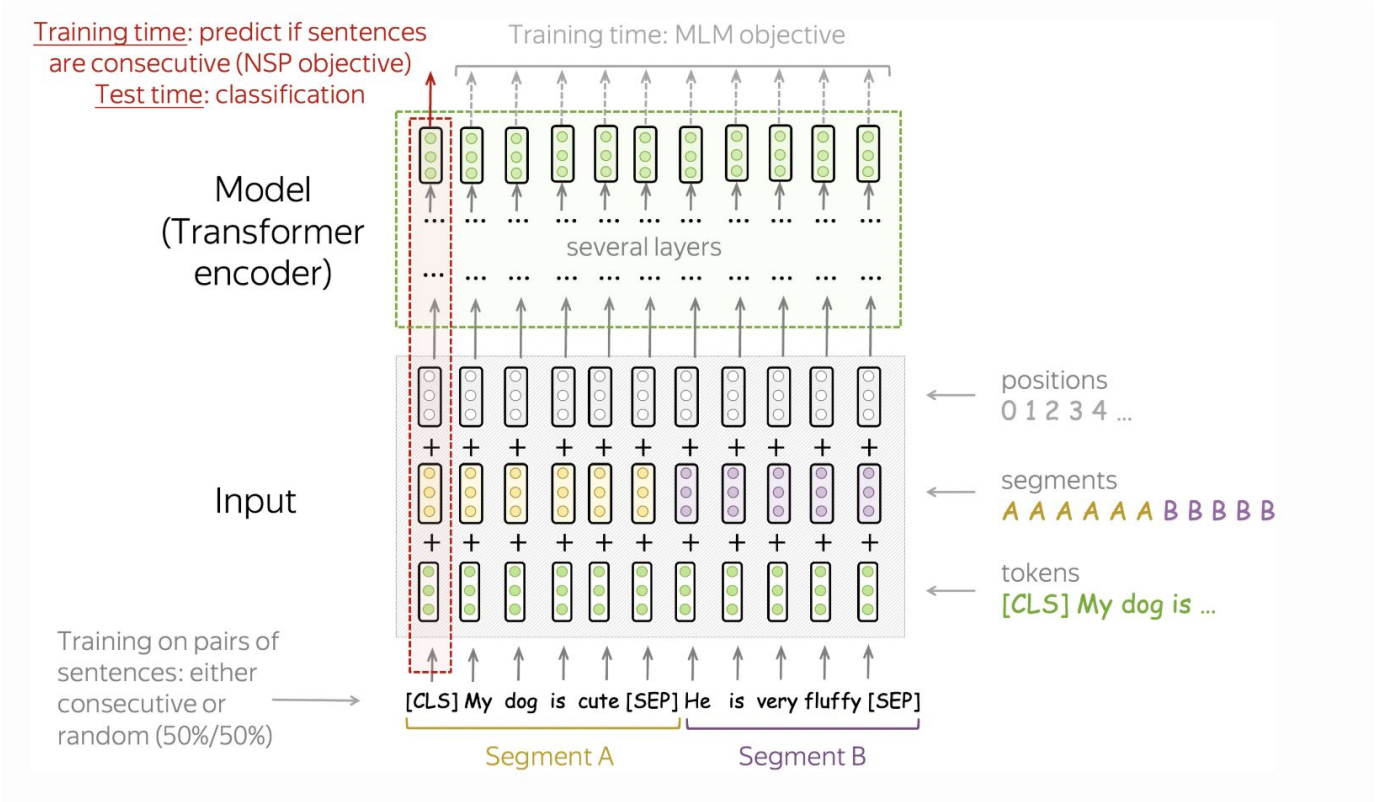
Training Input: Pairs of Sentences with Special Tokens

[CLS]: Special token

- Training time: predict if sentences are consecutive or not (Next Sentence Prediction /NSP objective)
- Test time: downstream tasks (e.g., classification)



BERT: cls token



BERT: pre-training objectives NSP

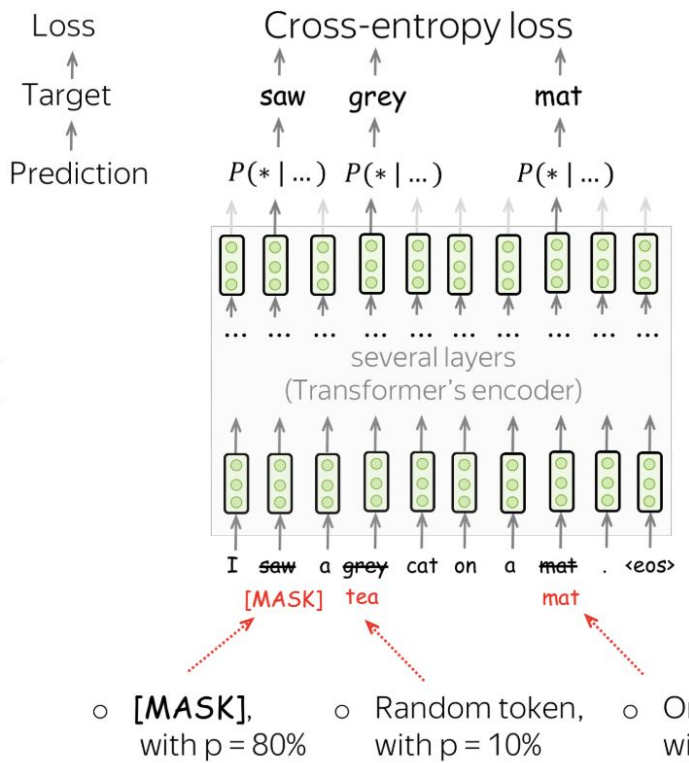
Input: [CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]
Label: isNext

Input: [CLS] the man went to [MASK] store [SEP] penguin [MASK] are flight ##less birds [SEP]
Label: notNext

BERT: pre-training objectives MLM

- **select some tokens**
  (each token is selected with the probability of 15%)
- **replace these selected tokens**
  (with the special token [MASK] - with p=80%, with a random token - with p=10%, with the original token (remain unchanged) - with p=10%)
- **predict original tokens (compute loss).**

BERT: pre-training objectives MLM

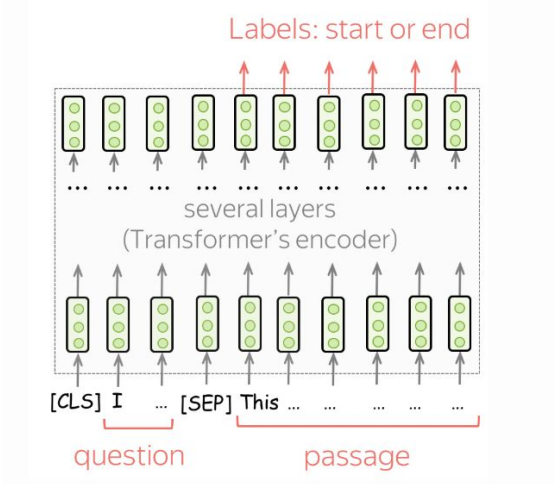


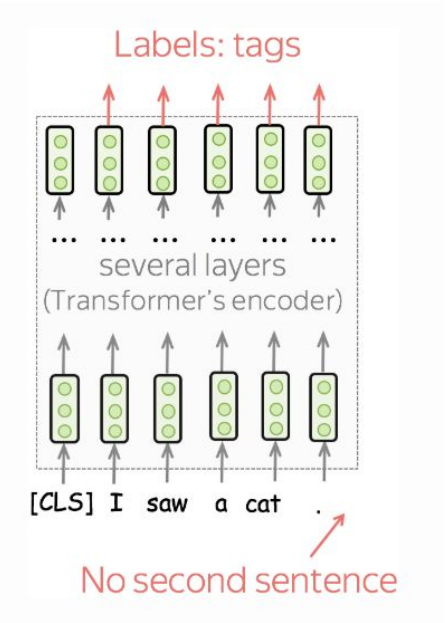
At each training step:

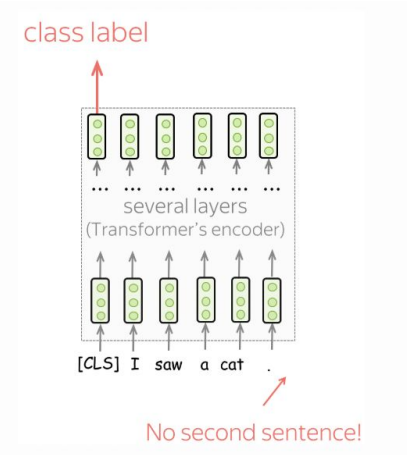
- pick randomly 15% of tokens
- replace each of the chosen tokens with something
- predict original chosen tokens

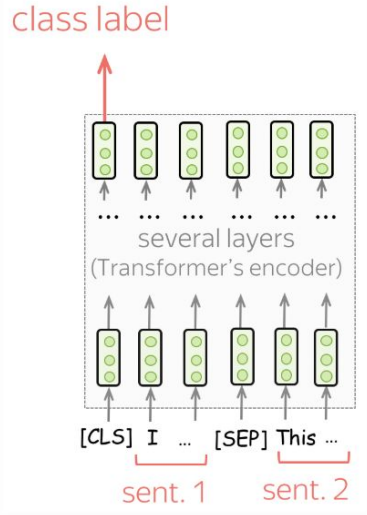
BERT: fine-tuning

- Single sentence classification
- Sentence pair classification
- Question Answering
- Single sentence tagging (NER)





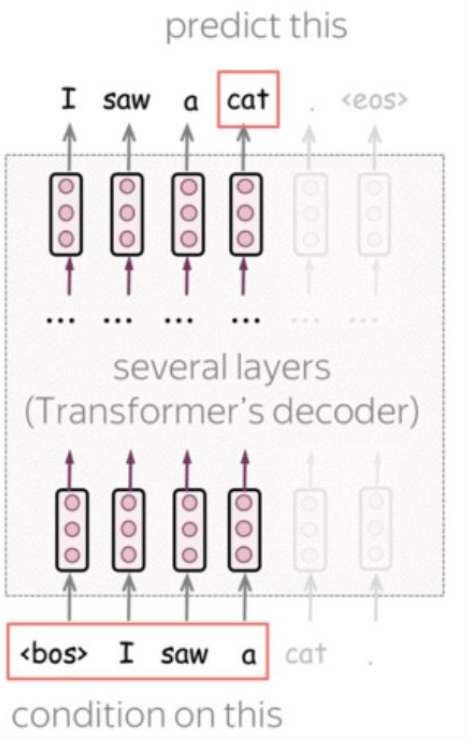




Lecture plan

- Tokenization
- General view:
  - Seq2Seq
  - Ablation: transformer vs rnn with attention
- Transformer: architectures
  - Self-Attention module
  - Feed forward module
  - Positional encoding
  - Other important parts
- Training Details
- BERT-like
- GPT-like

GPT-like



- GPT-1: Improving Language Understanding by Generative Pre-Training
- GPT-2: Language Models are Unsupervised Multitask Learners
- GPT-3: Language Models are Few-Shot Learners