Natural Language Processing: Transformers

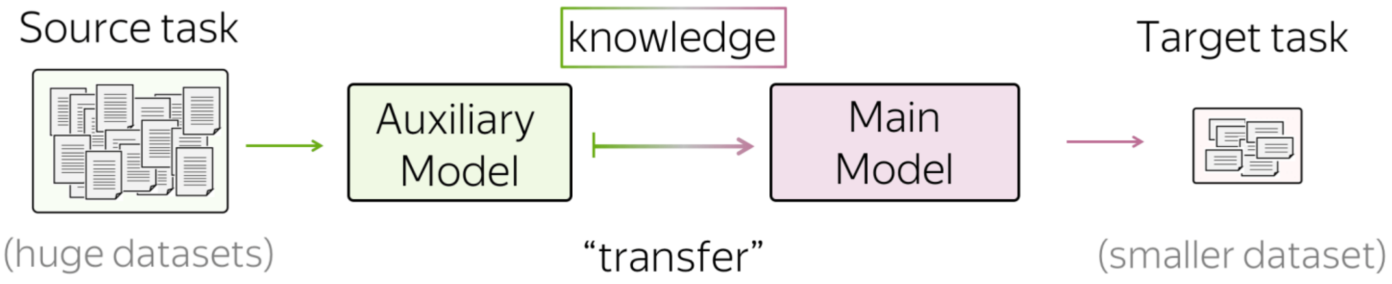
HSE Faculty of Computer Science

Machine Learning and Data-Intensive Systems

Murat Khazhgeriev

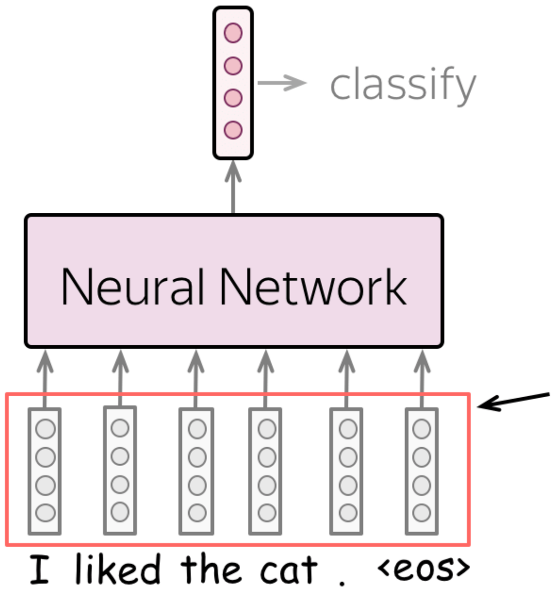
Table of Content

Training a model on a simple task can benefit a downstream one



Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

Training a model on a downstream task can be useful for another

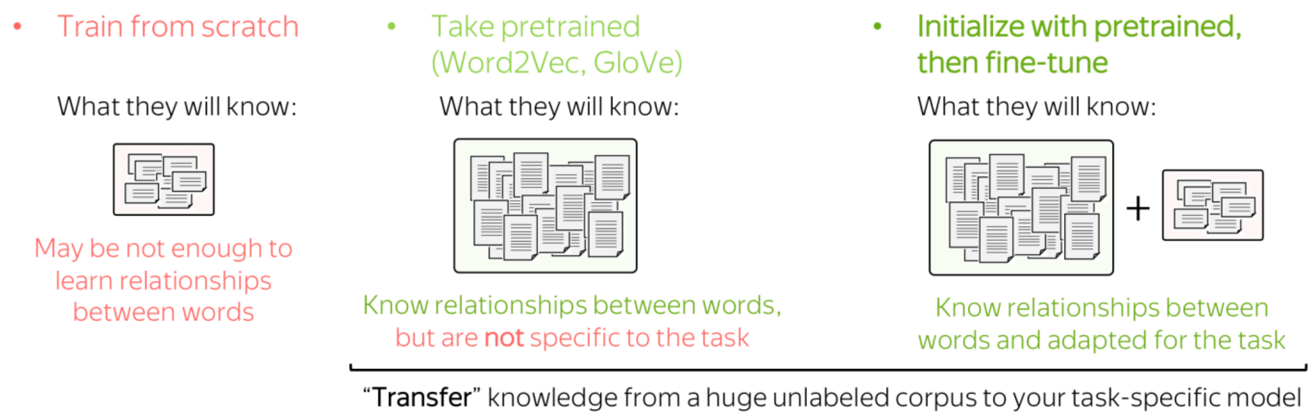


Input word embeddings:

- Train from scratch
- Take pretrained (Word2Vec, GloVe)
- **Initialize with pretrained, then fine-tune**

Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

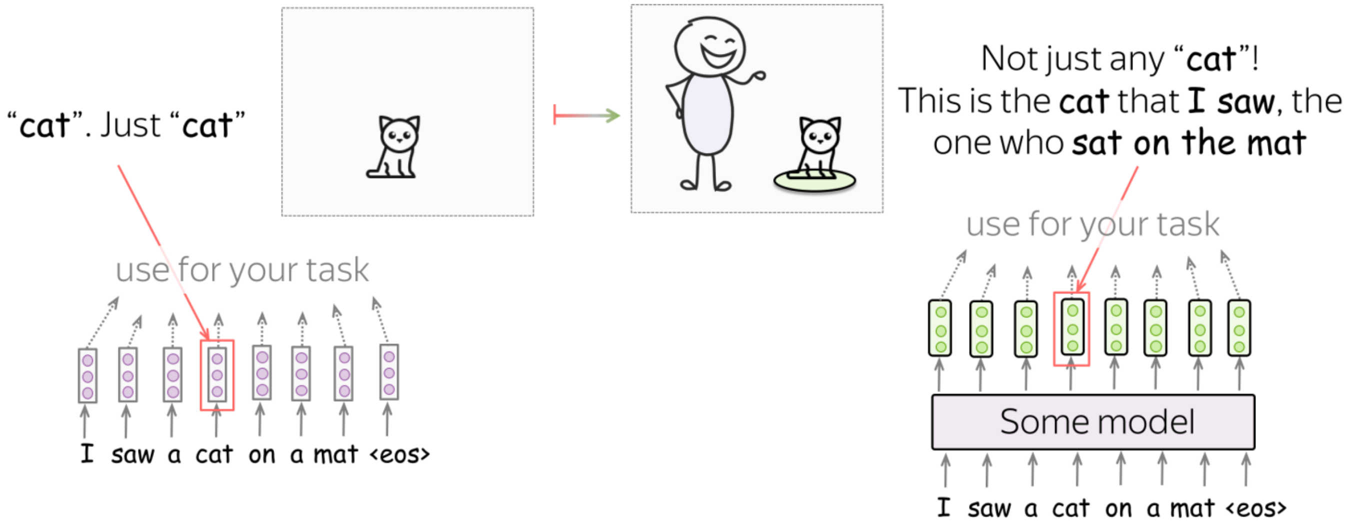
Training a model on a downstream task can be useful for another



Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

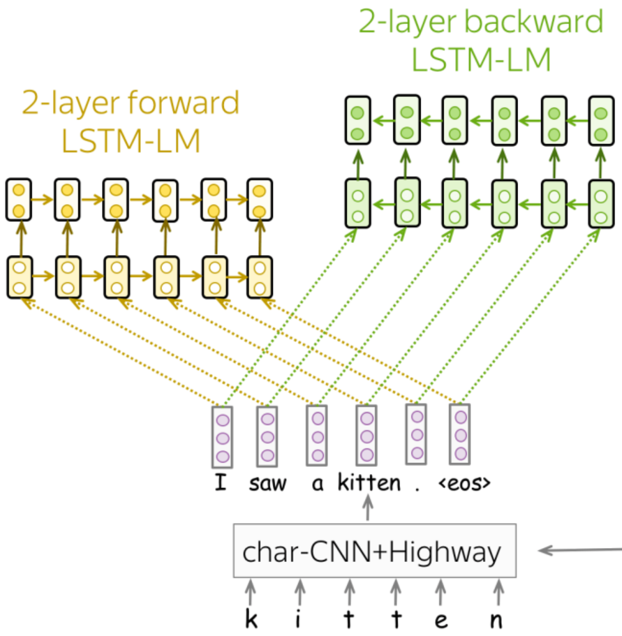
Table of Content

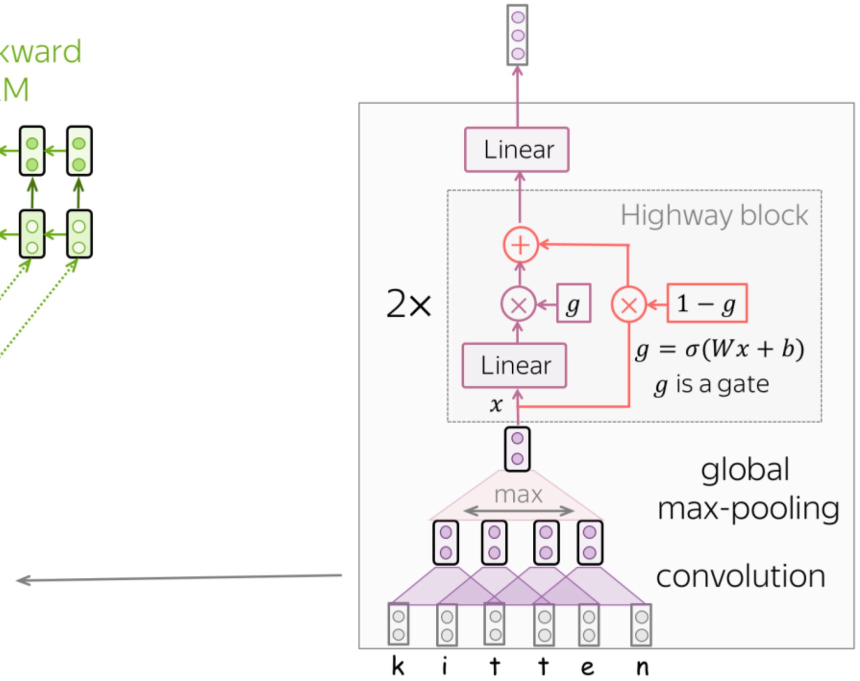
Not just a cat, but the cat!



Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

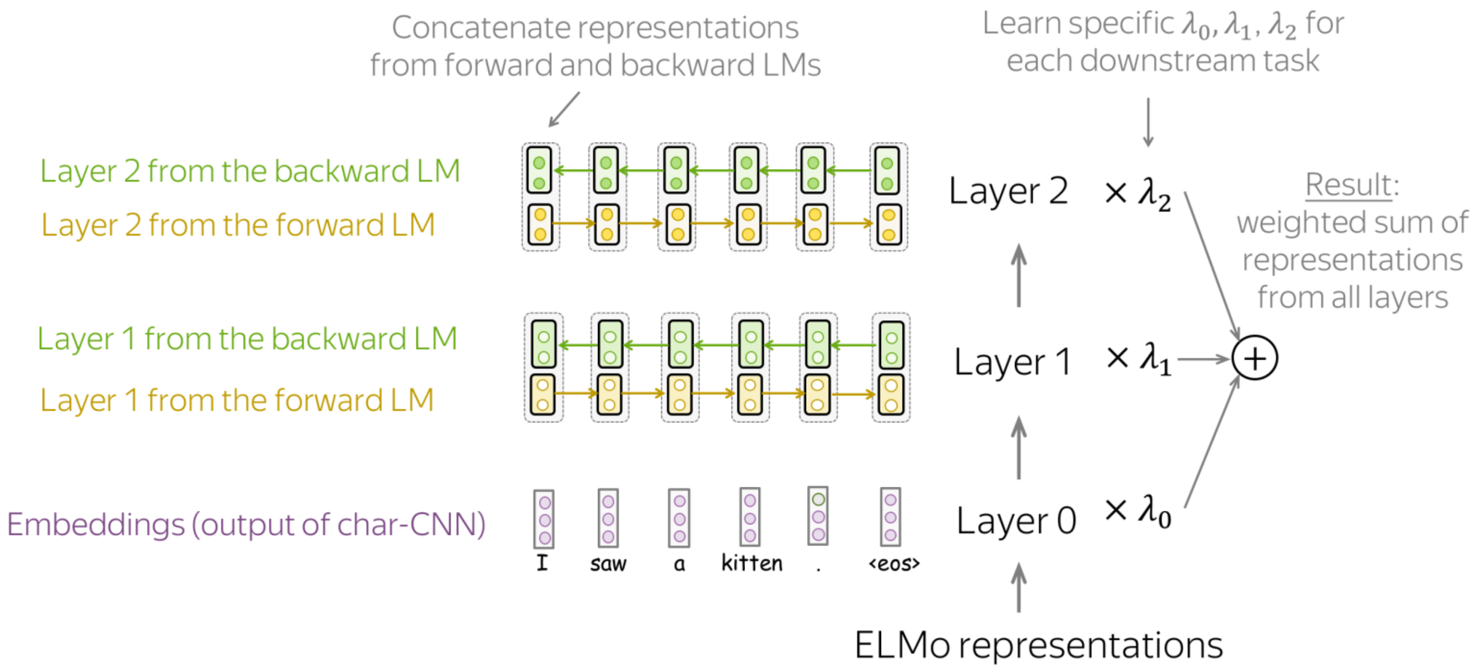
Train a “translator” from word-specific to “contextual” space





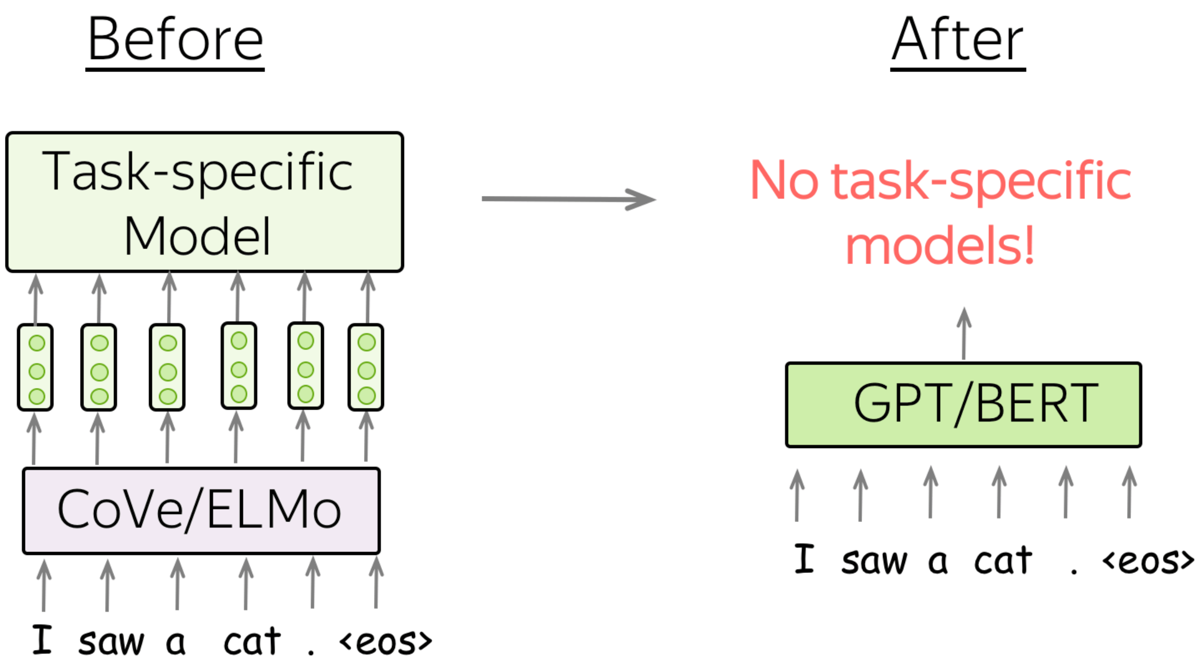
Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

Multiple layers to capture low-level and high-level context



Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

From embedding generator to a universal model



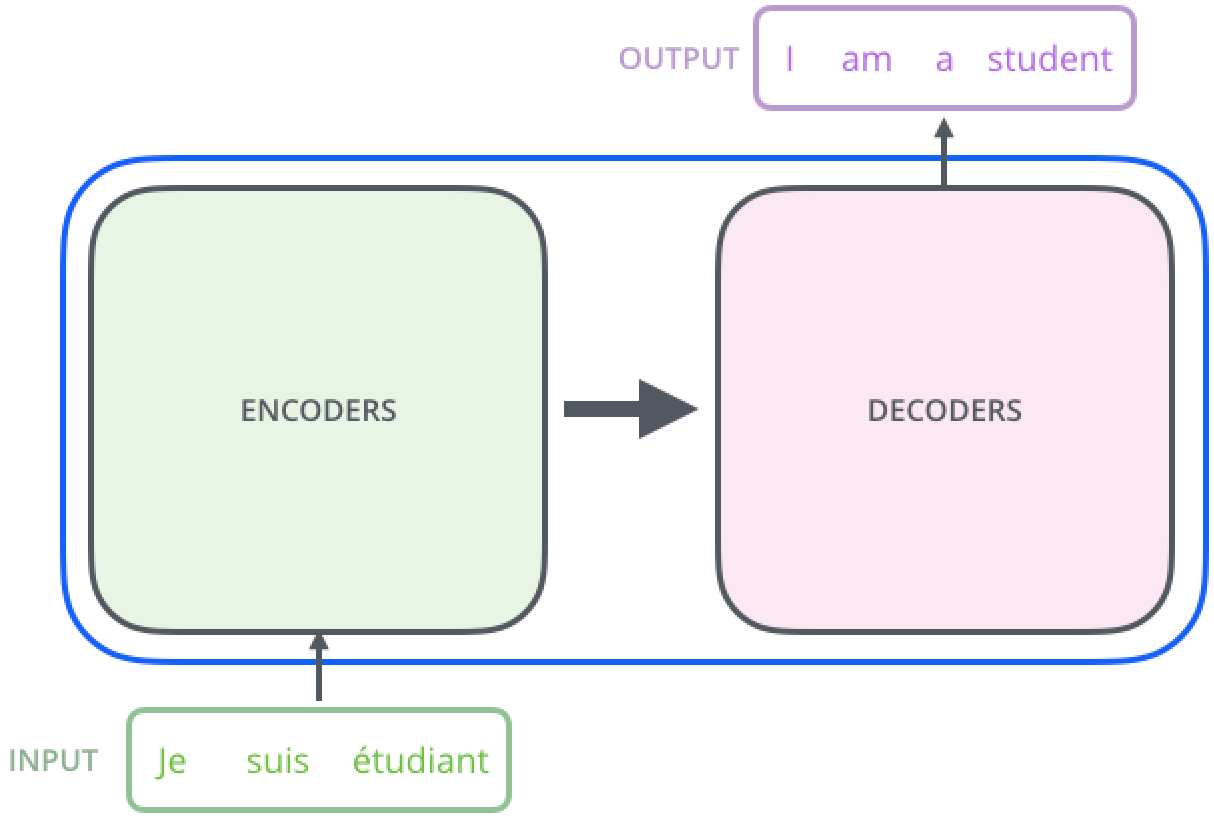
Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

Table of Content

Transformer is an example of Encoder-Decoder architecture

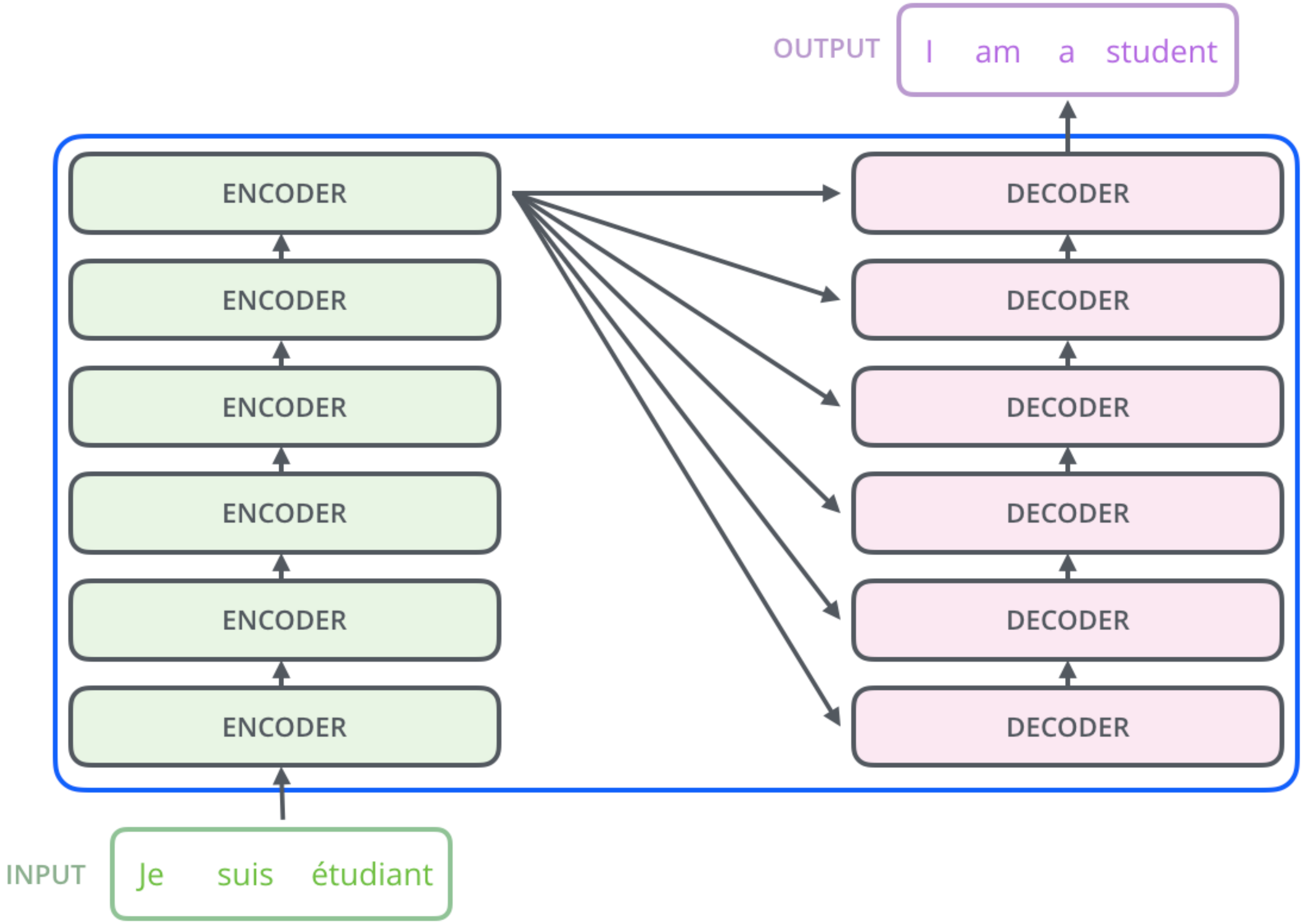
Source: https://jalammar.github.io/illustrated-transformer/

Transformer is an example of Encoder-Decoder architecture



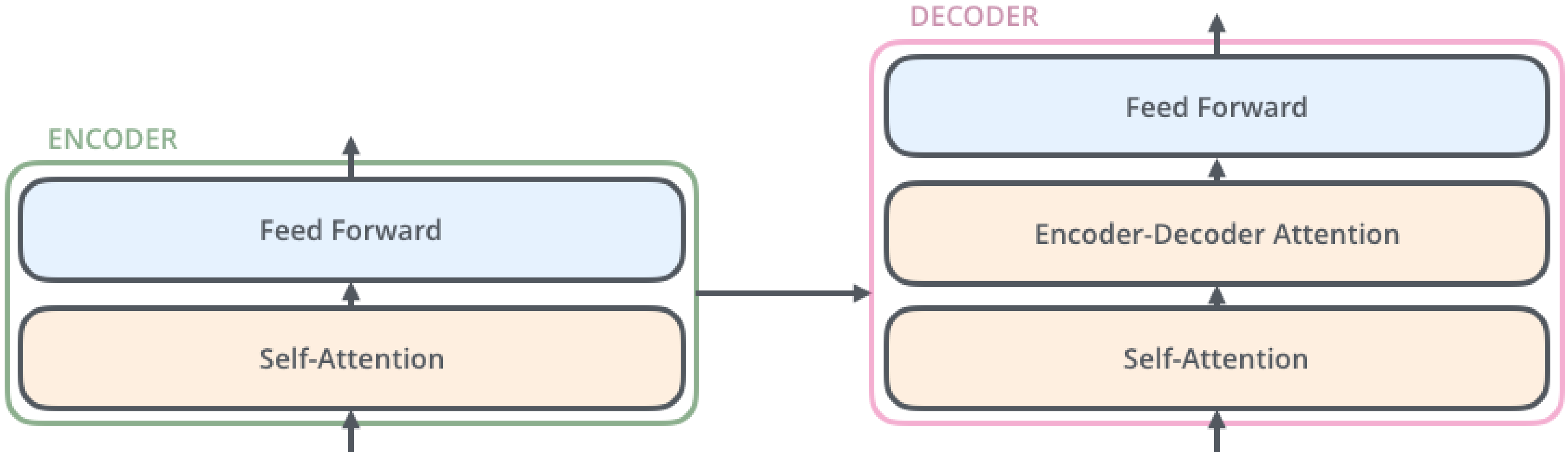
Source: https://jalammar.github.io/illustrated-transformer/

Transformer is an example of Encoder-Decoder architecture



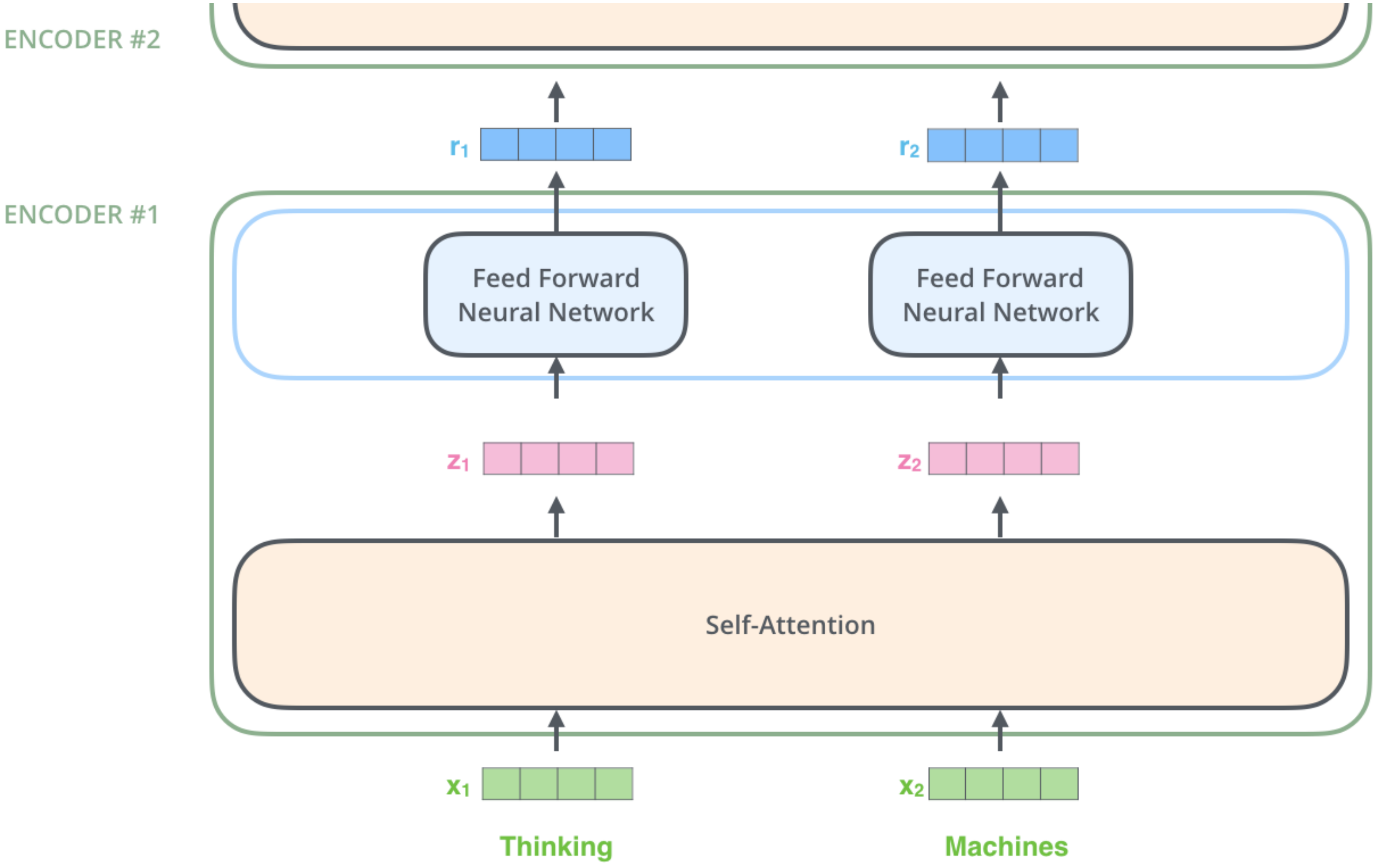
Source: https://jalammar.github.io/illustrated-transformer/

There is always two of them: the Attention and the FFN



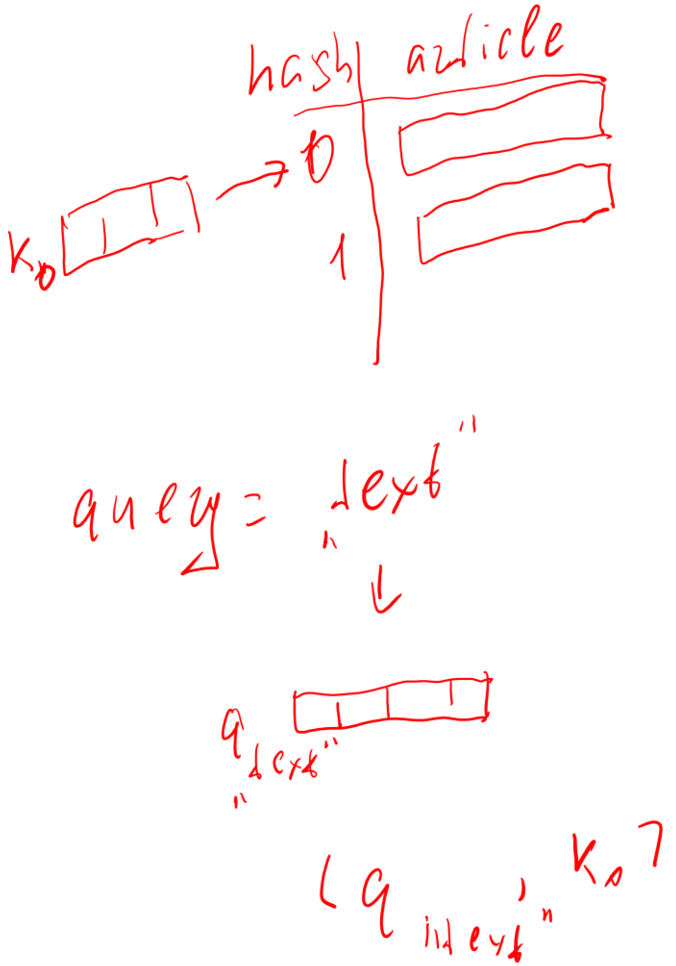
Source: https://jalammar.github.io/illustrated-transformer/

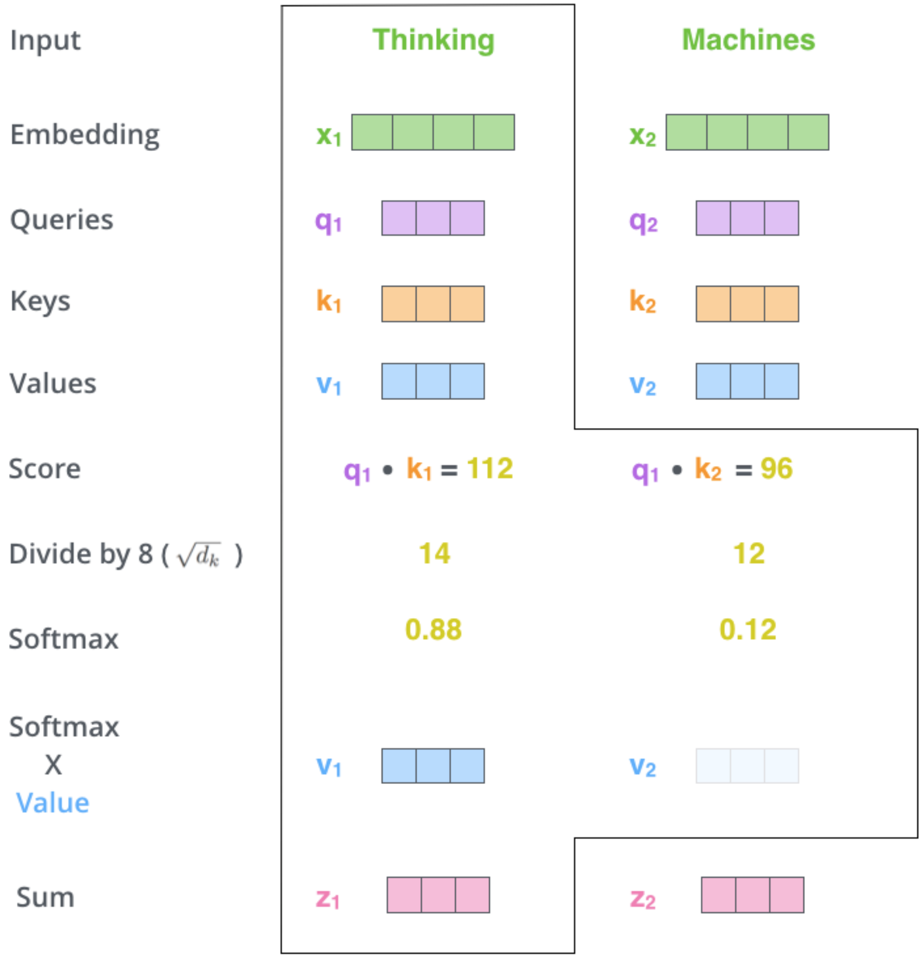
There is always two of them: the Attention and the FFN



Source: https://jalammar.github.io/illustrated-transformer/

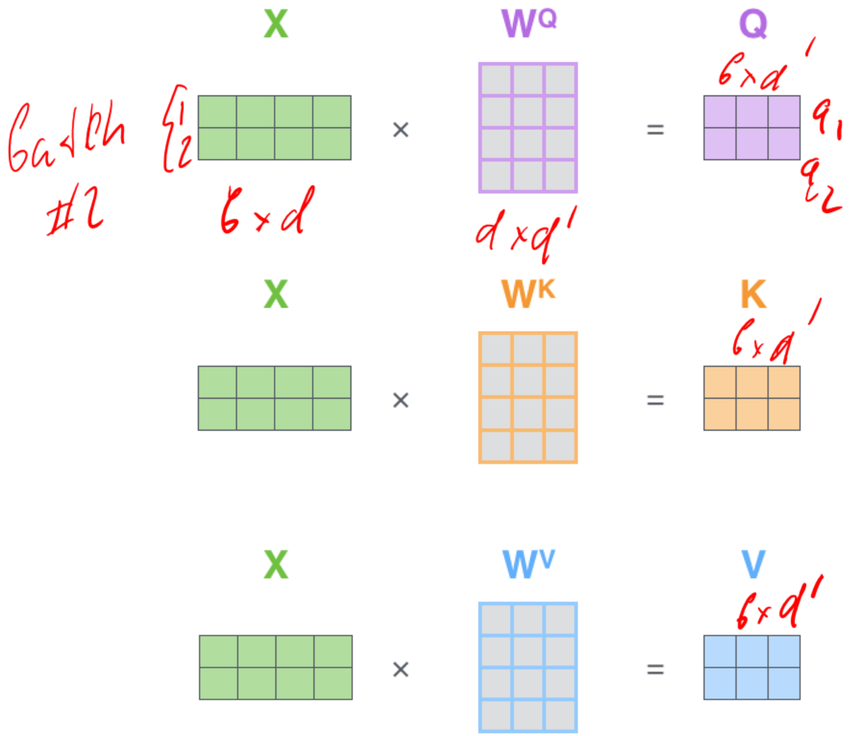
Inside the self-Attention





Source: https://jalammar.github.io/illustrated-transformer/

Inside the self-Attention: Matrix View



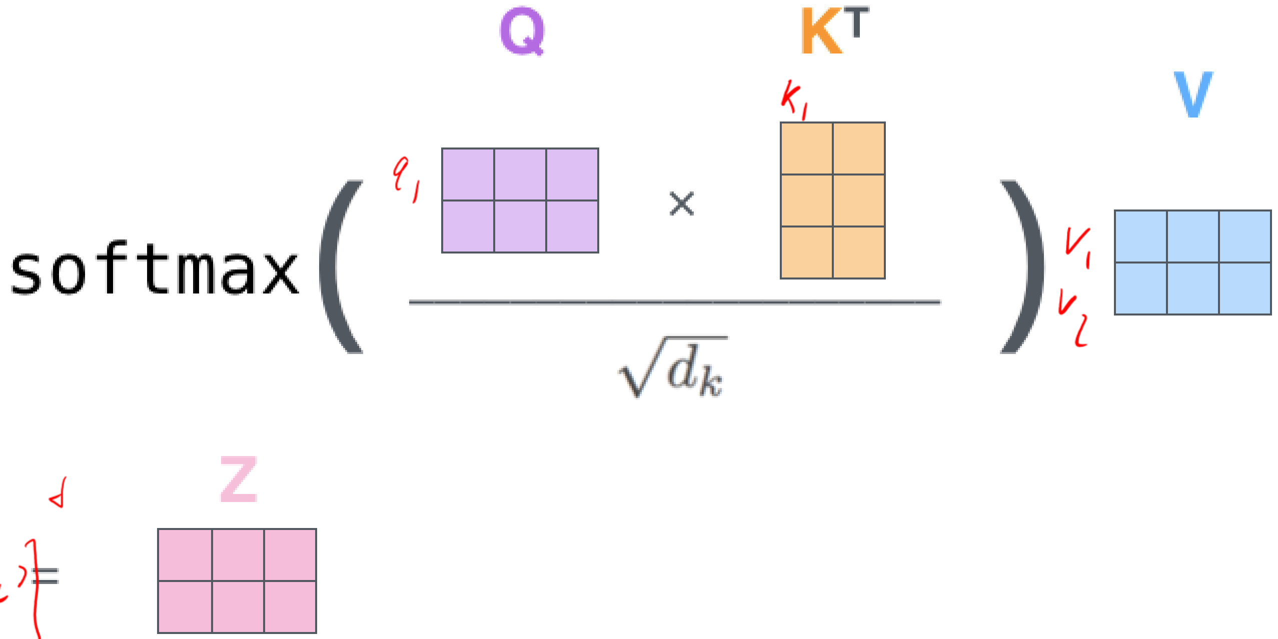
Source: https://jalammar.github.io/illustrated-transformer/

Inside the self-Attention: Matrix View

$$Q = \begin{bmatrix} - & q_1 & - \\ - & q_2 & - \end{bmatrix}$$

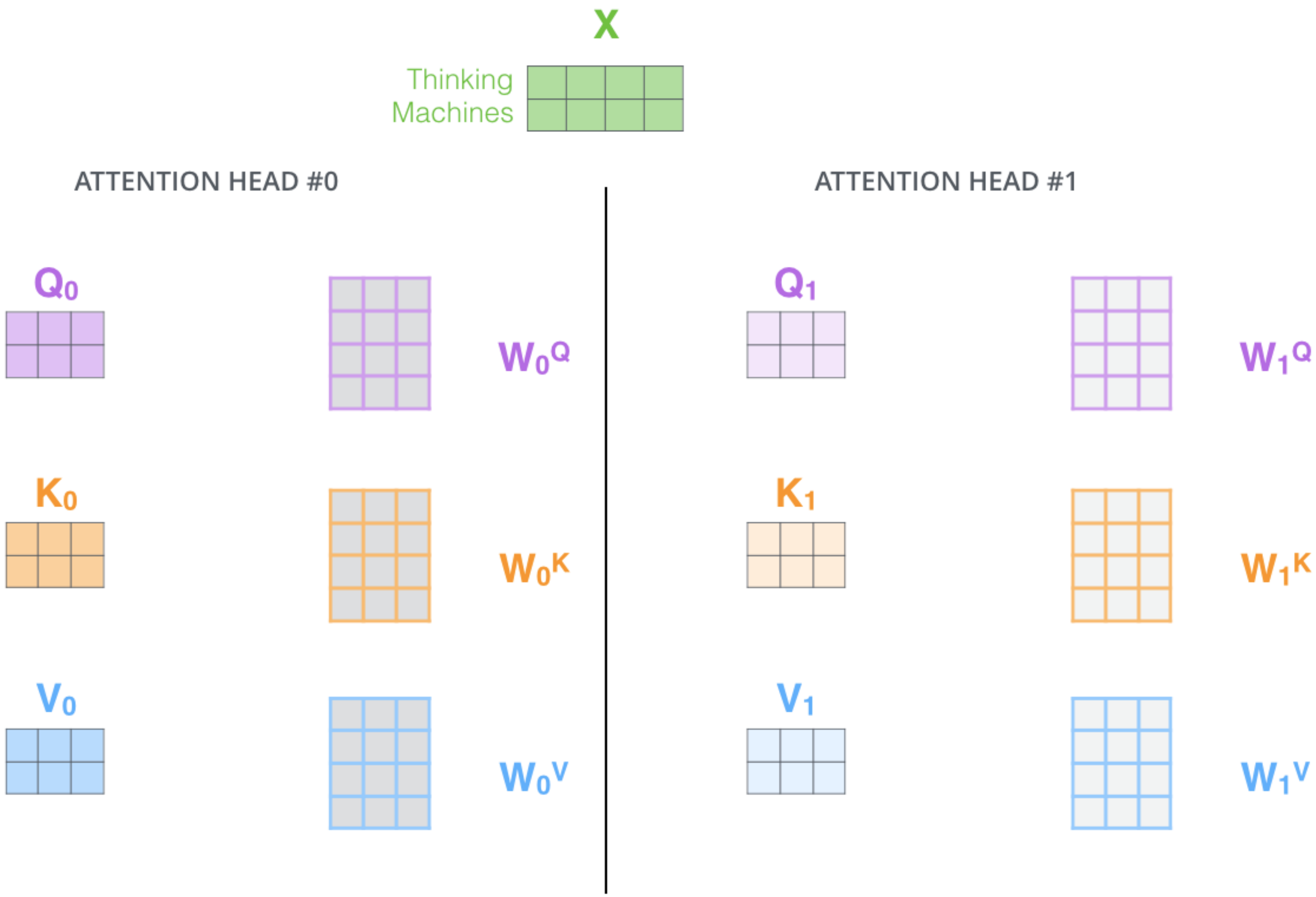
$$K^T = \begin{bmatrix} | & | \\ k_1 & k_2 \\ | & | \end{bmatrix}$$

$$QK^T = \begin{bmatrix} \langle q_1, k_1 \rangle & \langle q_1, k_2 \rangle \\ \langle q_2, k_1 \rangle & \langle q_2, k_2 \rangle \end{bmatrix}$$



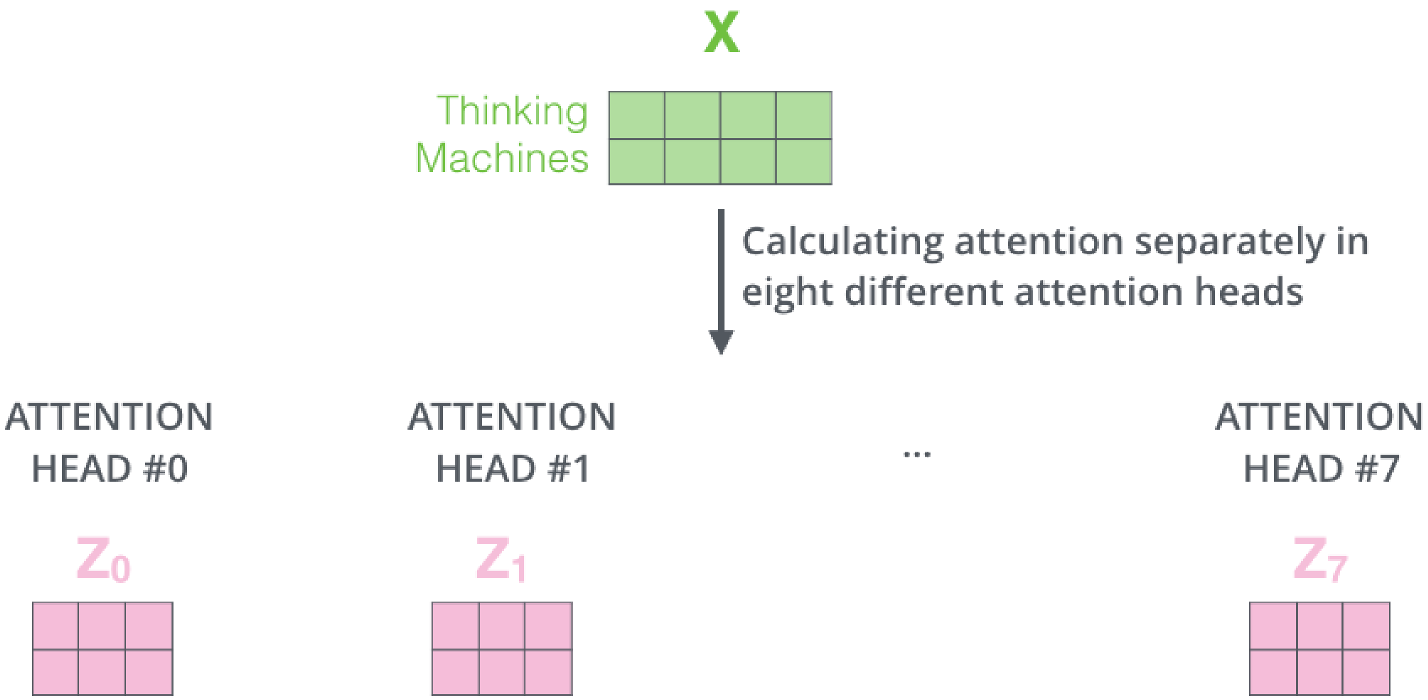
Source: https://jalammar.github.io/illustrated-transformer/

A beast with many heads



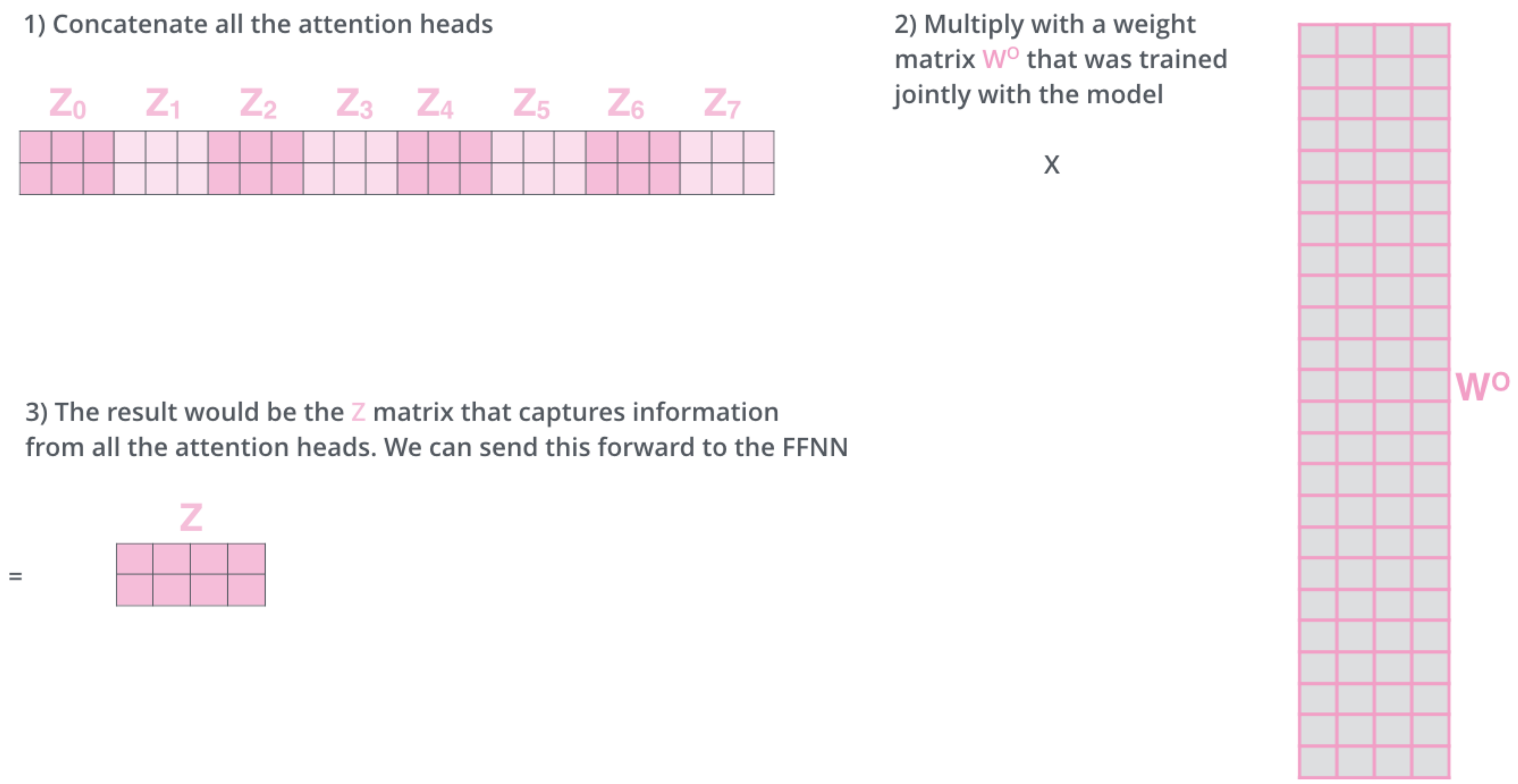
Source: https://jalammar.github.io/illustrated-transformer/

A beast with many heads



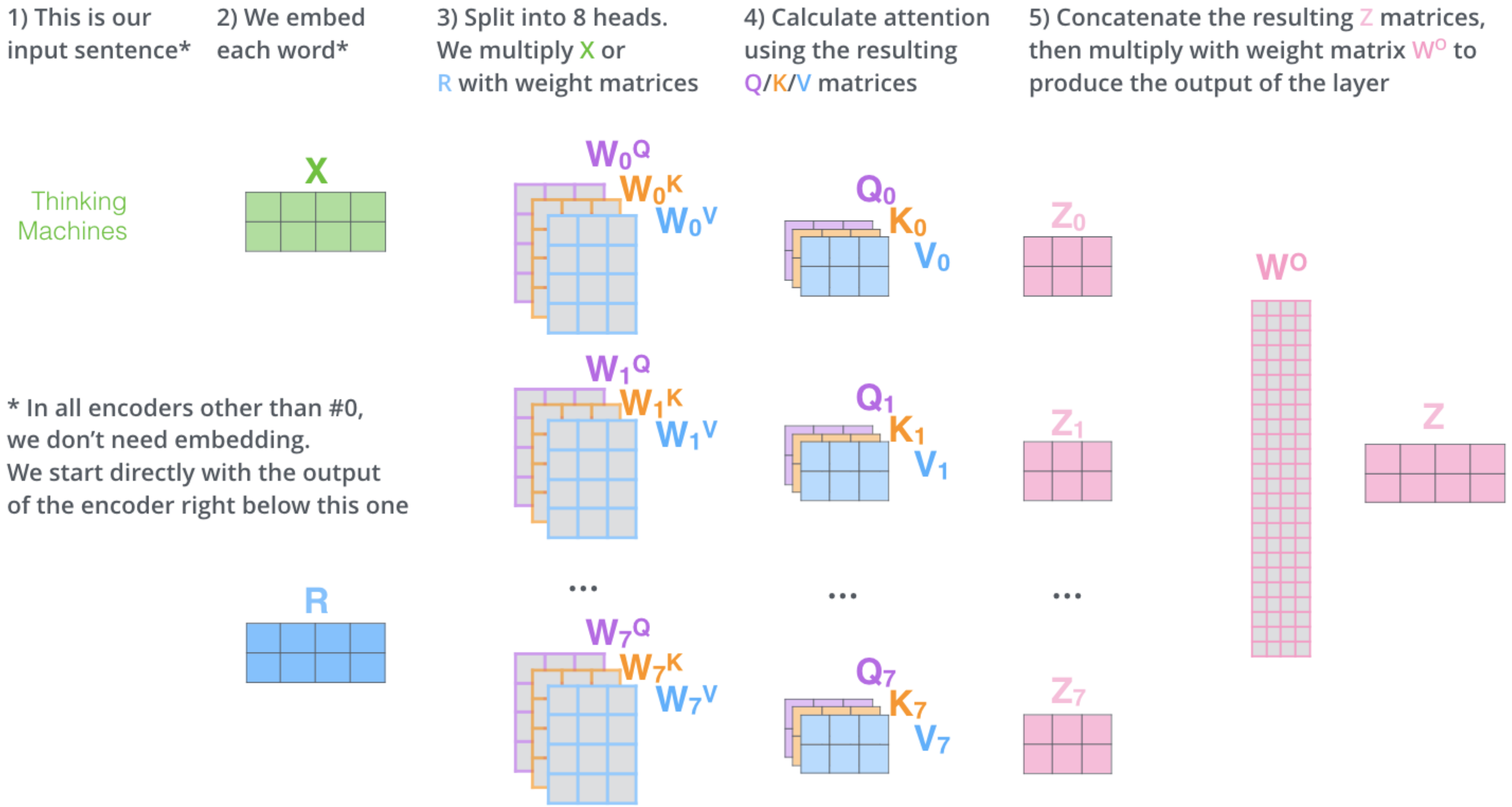
Source: https://jalammar.github.io/illustrated-transformer/

A beast with many heads



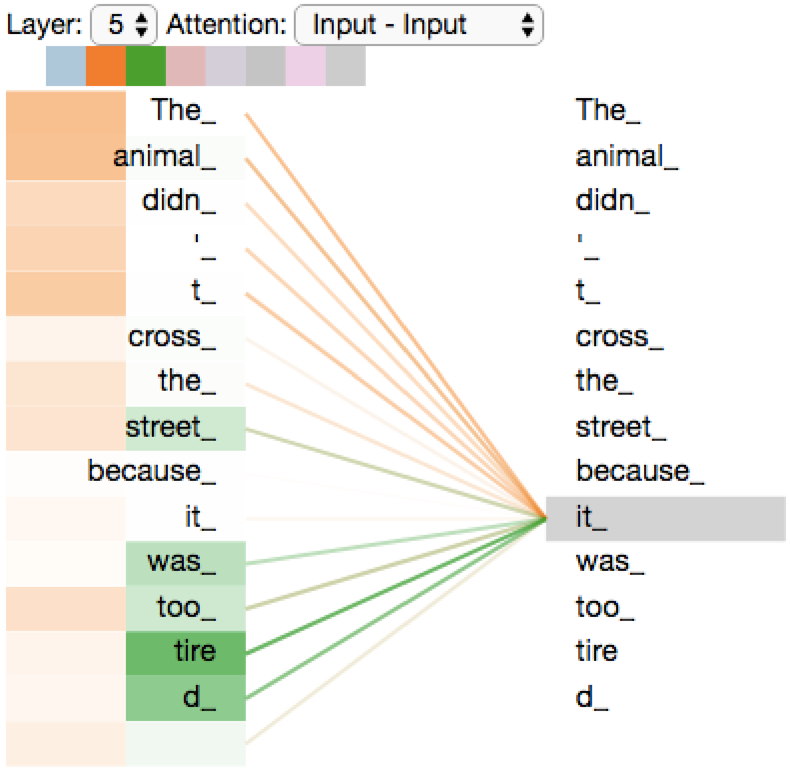
Source: https://jalammar.github.io/illustrated-transformer/

A multi-head attention overview



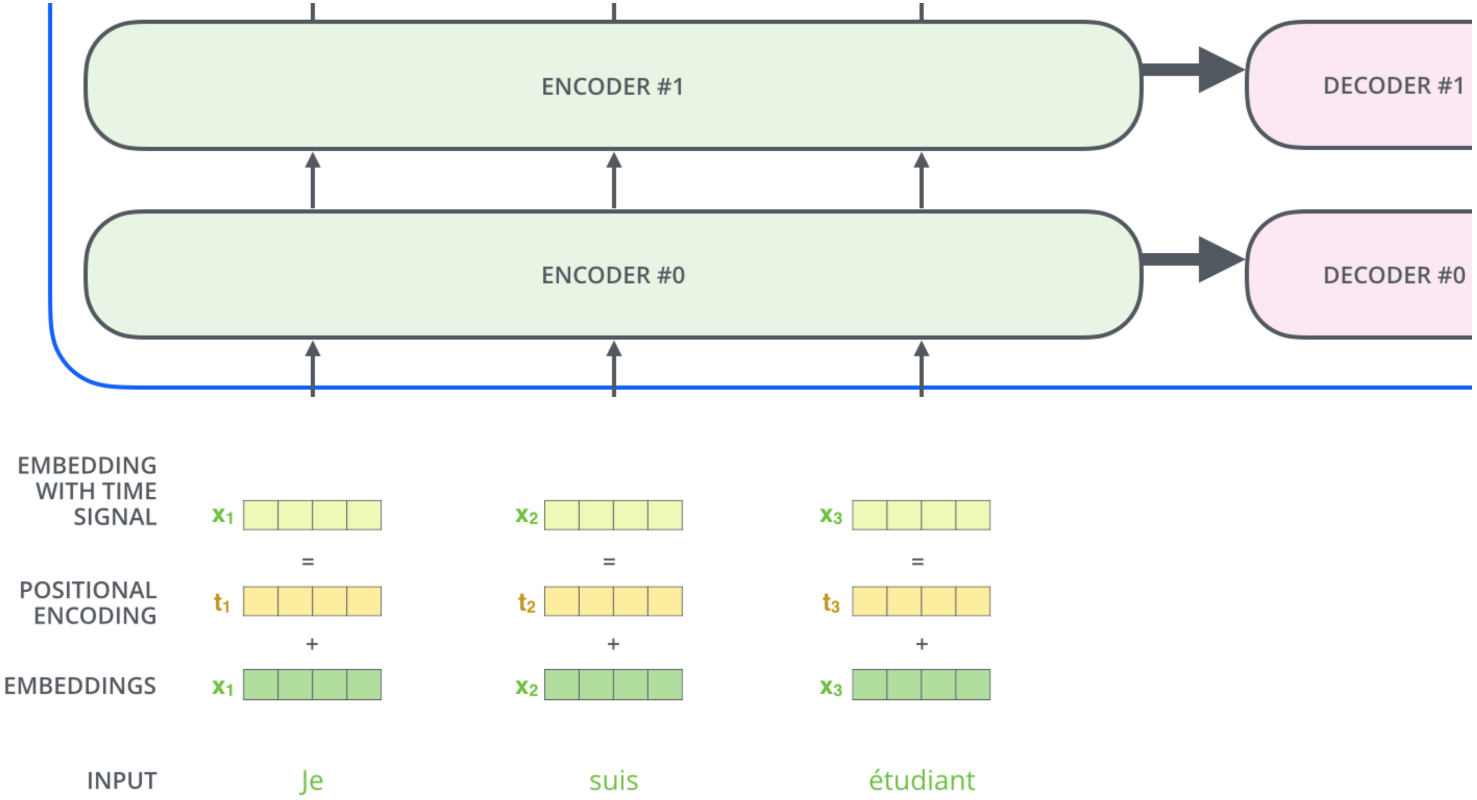
Source: https://jalammar.github.io/illustrated-transformer/

Each head focuses on a specific representation



Source: https://jalammar.github.io/illustrated-transformer/

As opposed to RNNs, Transformers do not track the position implicitly



Source: https://jalammar.github.io/illustrated-transformer/

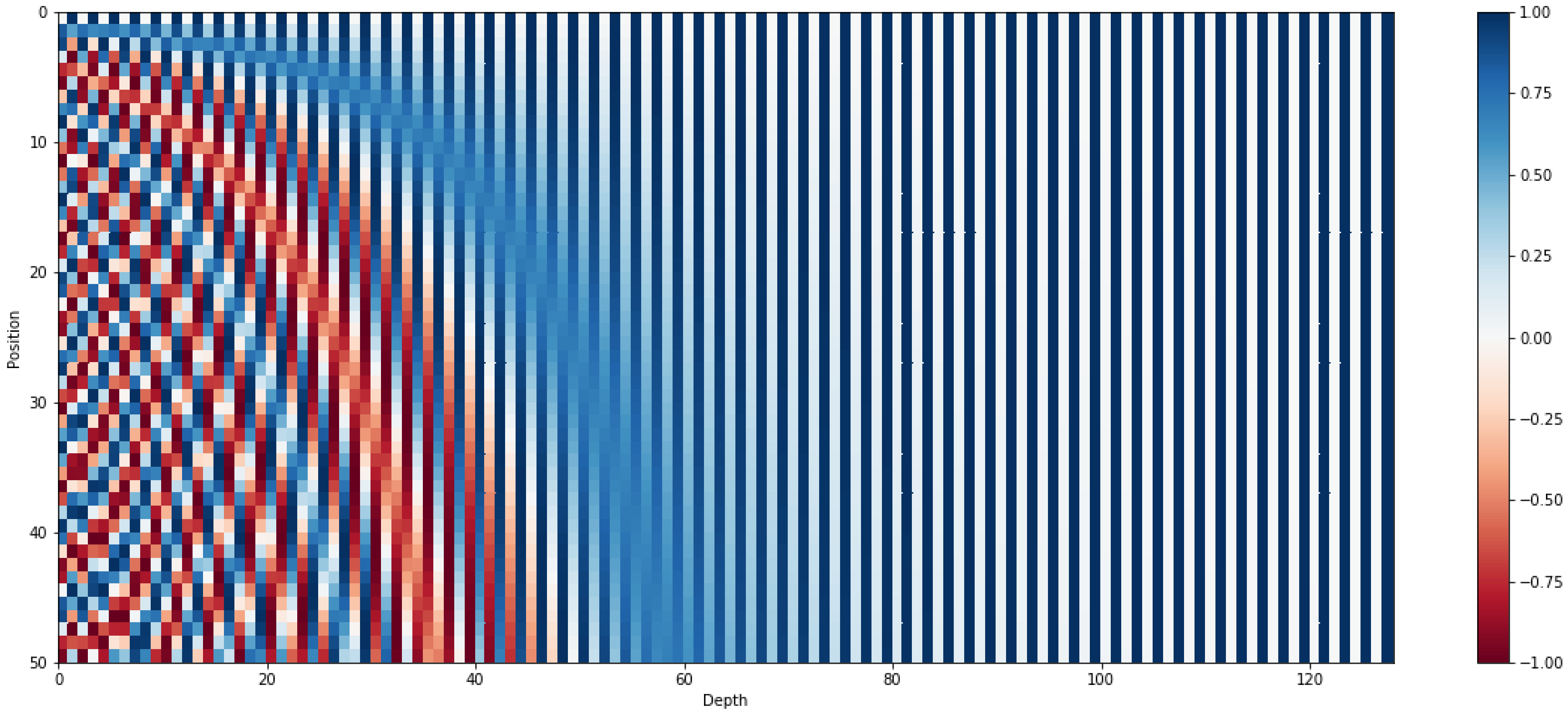
Absolute Positional Encoding

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

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

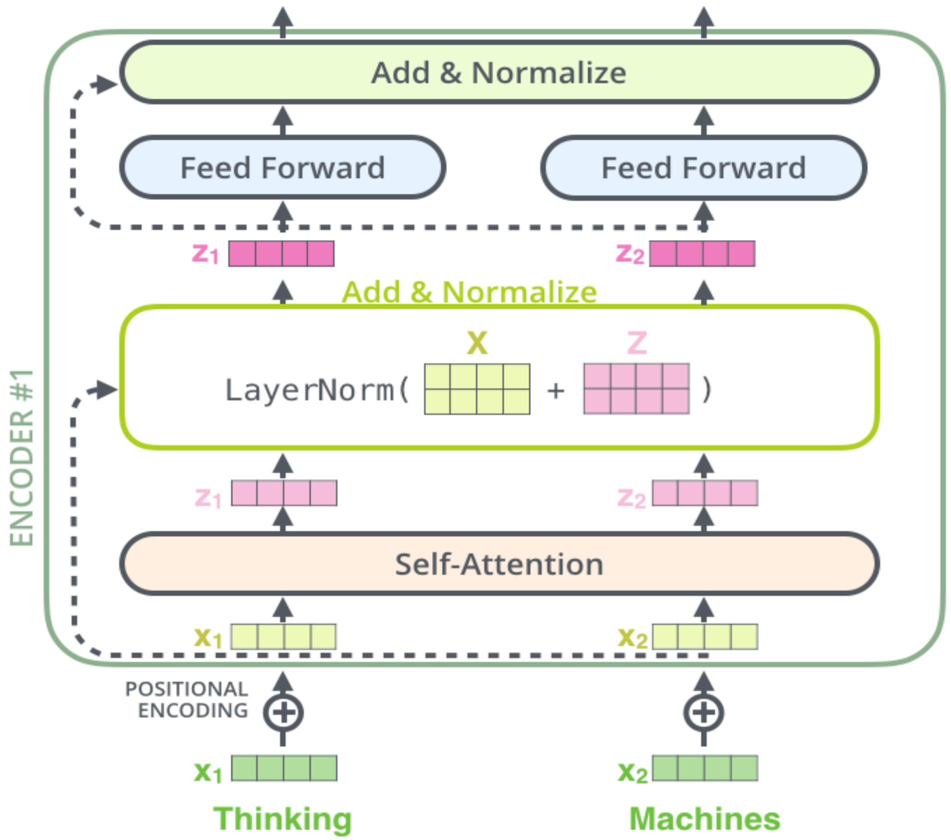
Source: https://arxiv.org/abs/1706.03762v7

What it looks like



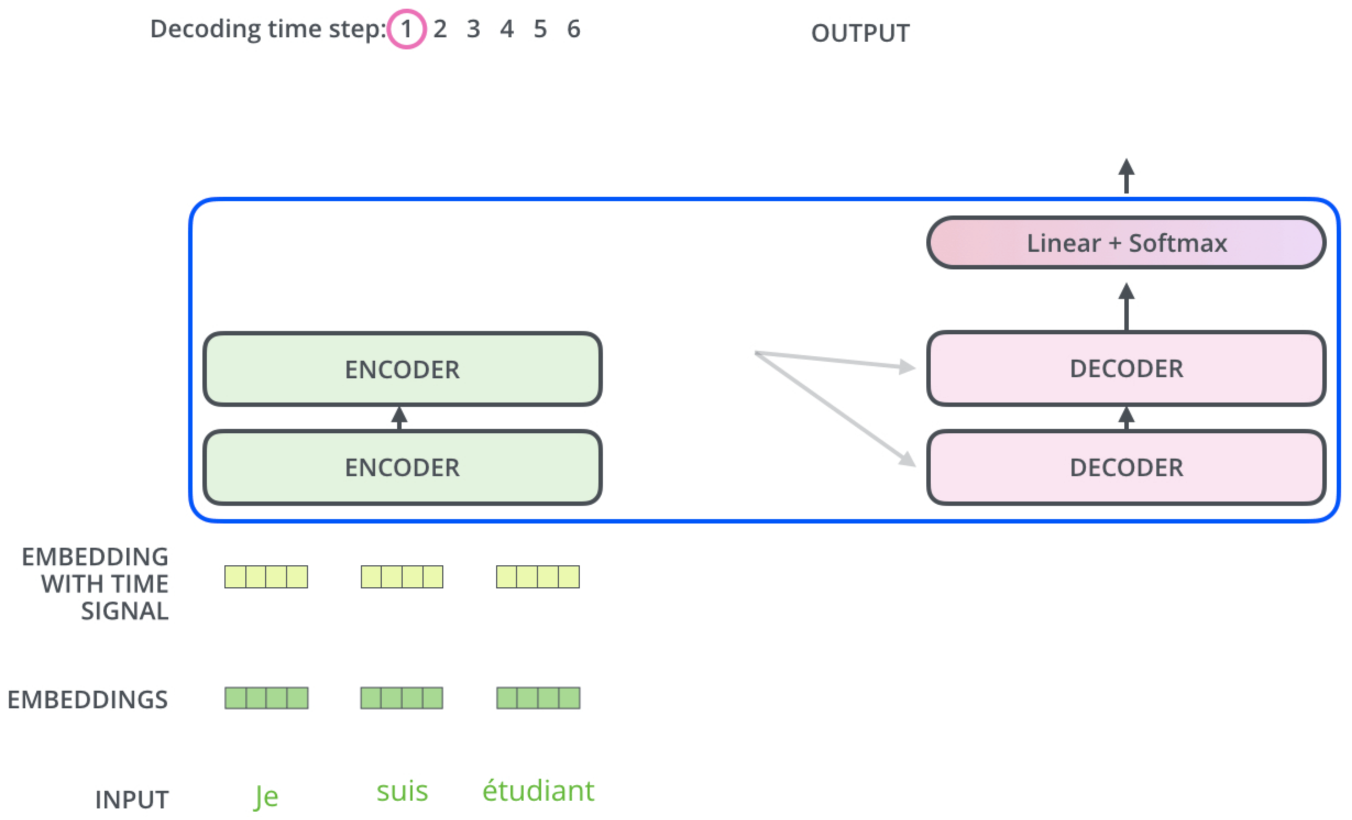
Source: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Skip Connection and Layer Normalization for a robust training



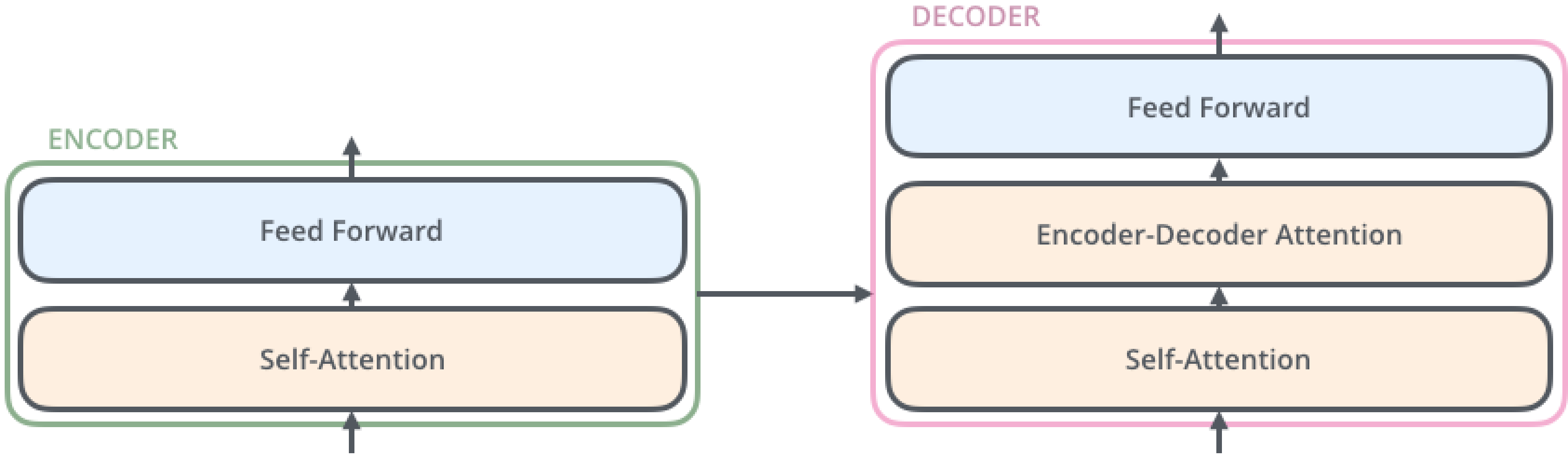
Source: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Bringing it all together



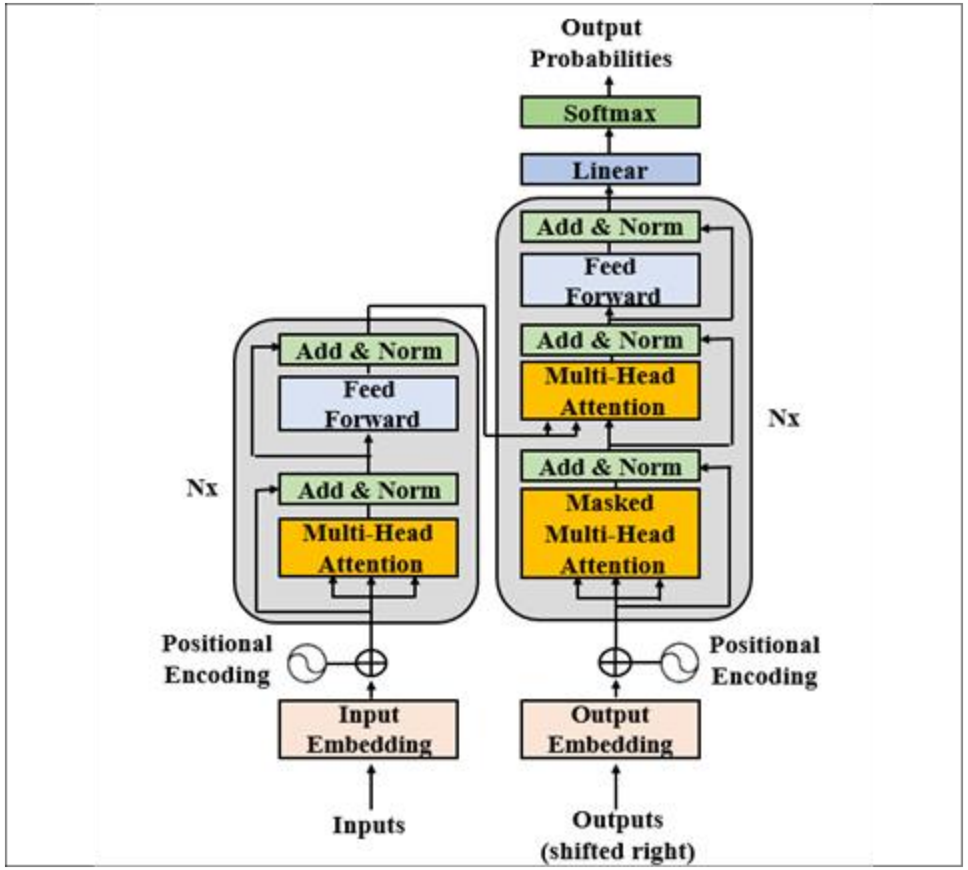
Source: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

There is always two of them: the Attention and the FFN



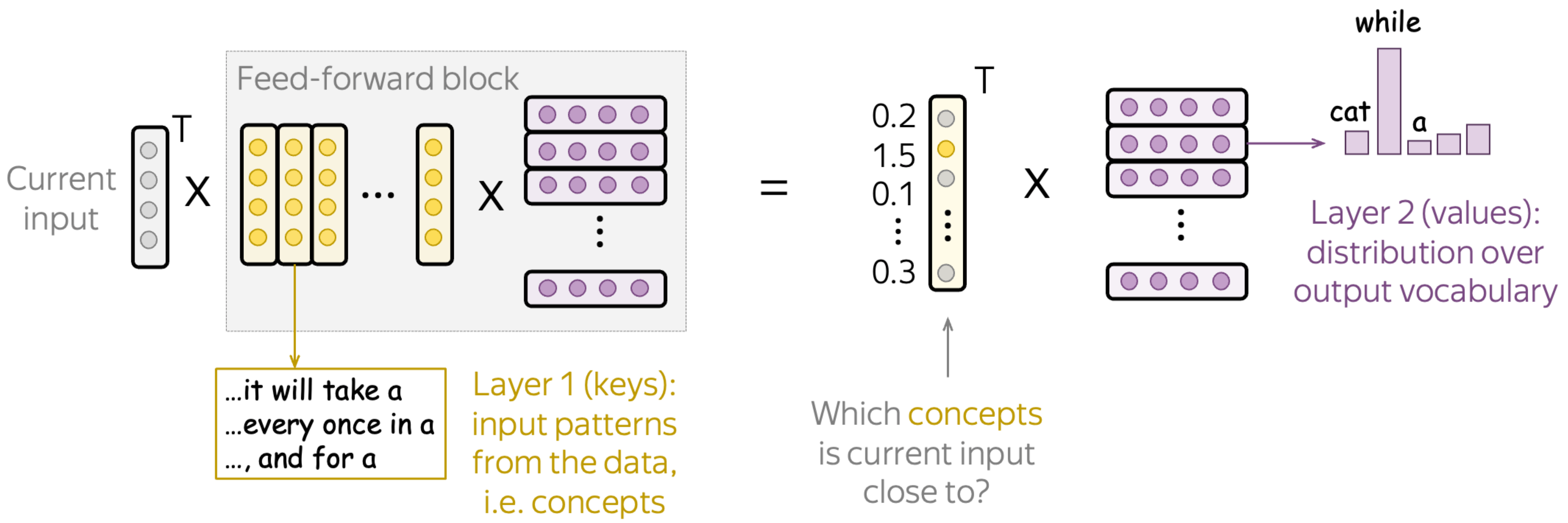
Source: https://jalammar.github.io/illustrated-transformer/

Attention is all you need



Source: https://arxiv.org/pdf/1706.03762

Attention is all you need (but not really)



Source: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Table of Content

BERT is just an Encoder part

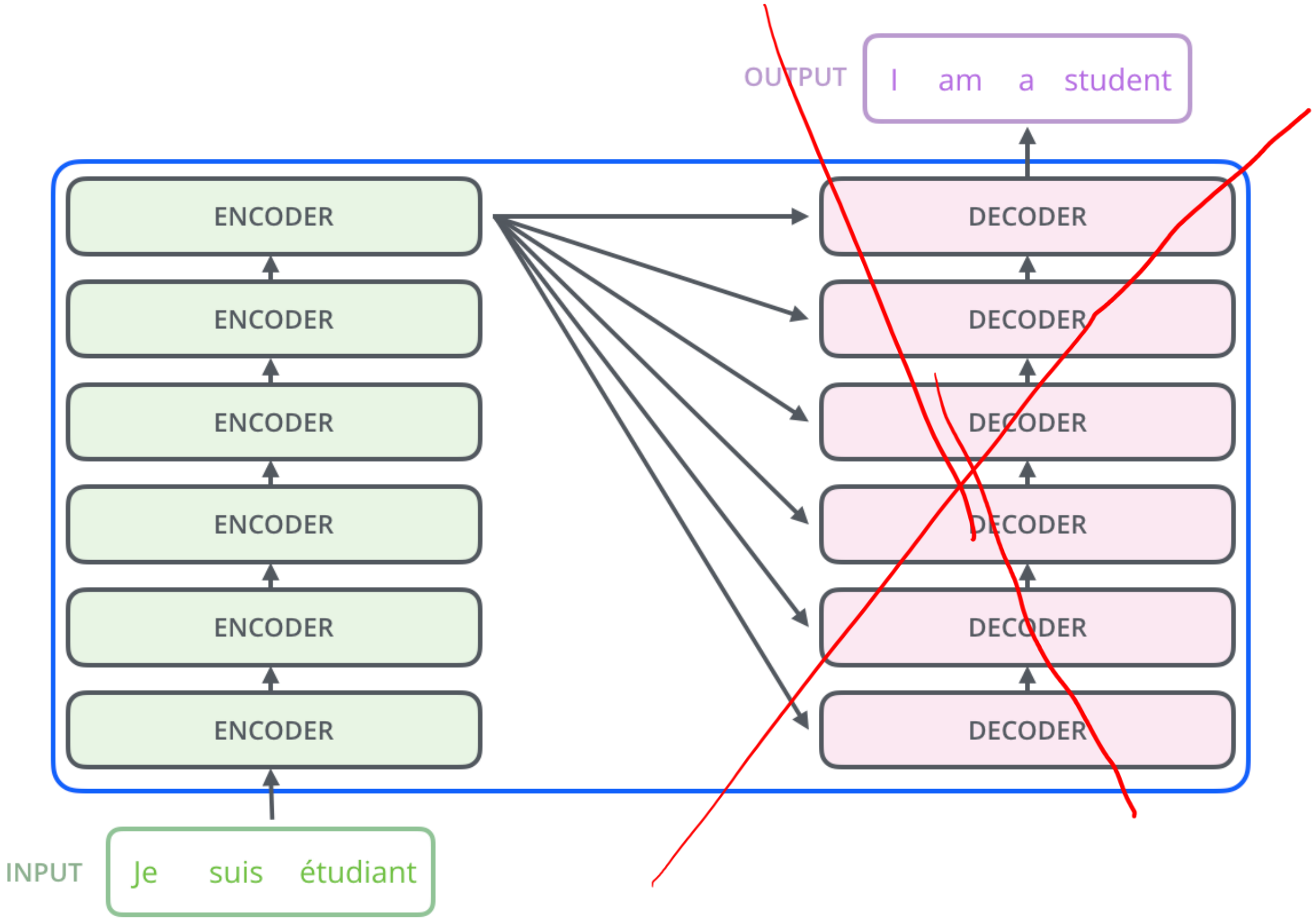
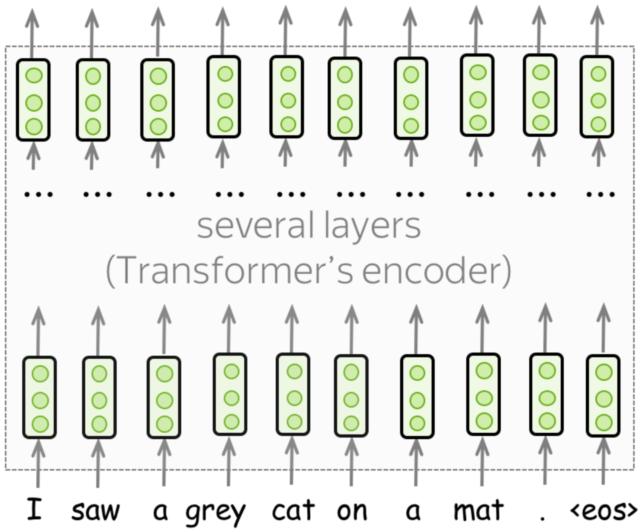


Image source: https://jalammar.github.io/illustrated-transformer/

Using encoder as an embedding generator



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

Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

Objective one: tell whether the two sequences are consecutive

[CLS]: Special token

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

[SEP]: Special token-separator

[CLS] My dog is very cute [SEP] He likes playing in the garden with me [SEP]

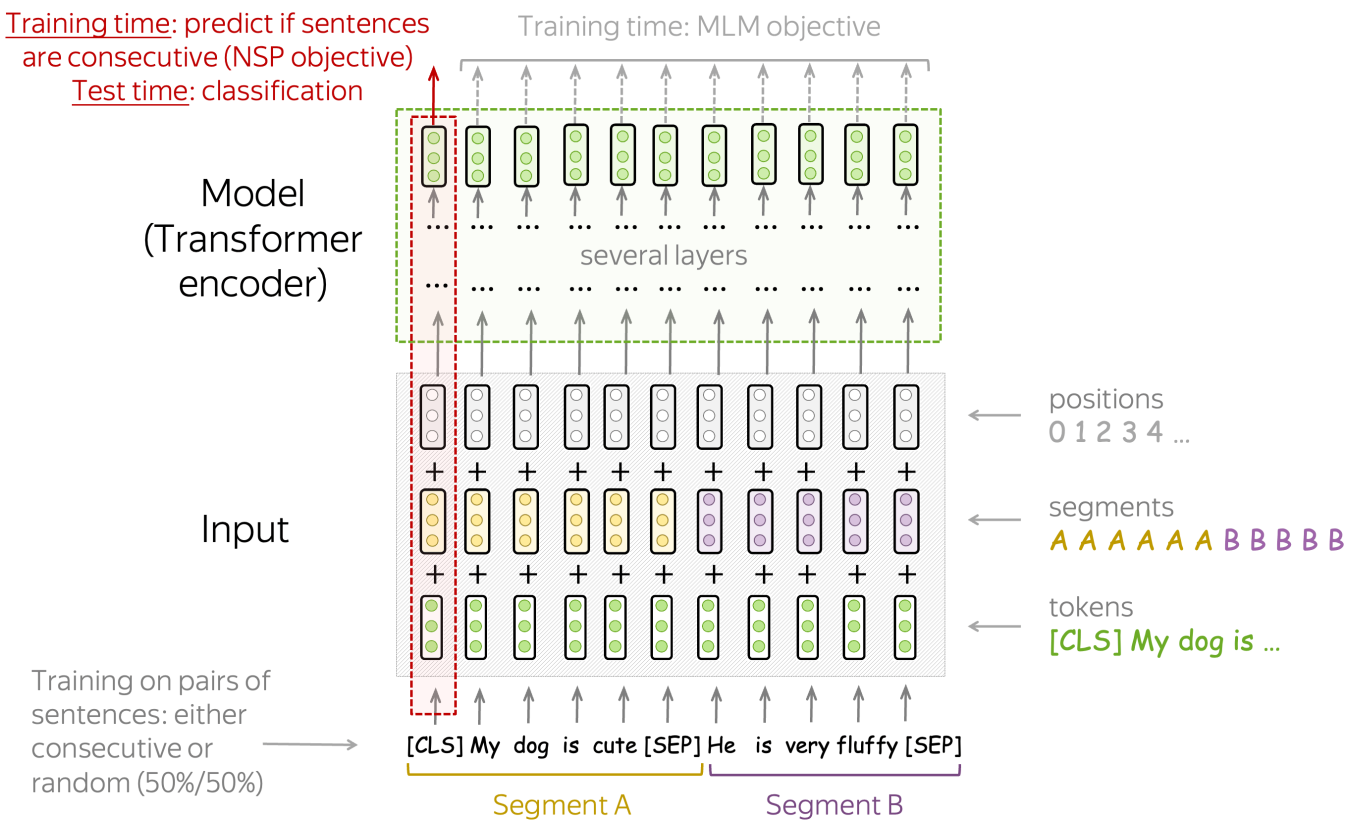
Segment A

Segment B

Training on pairs of sentences: either consecutive or random (50%/50%)

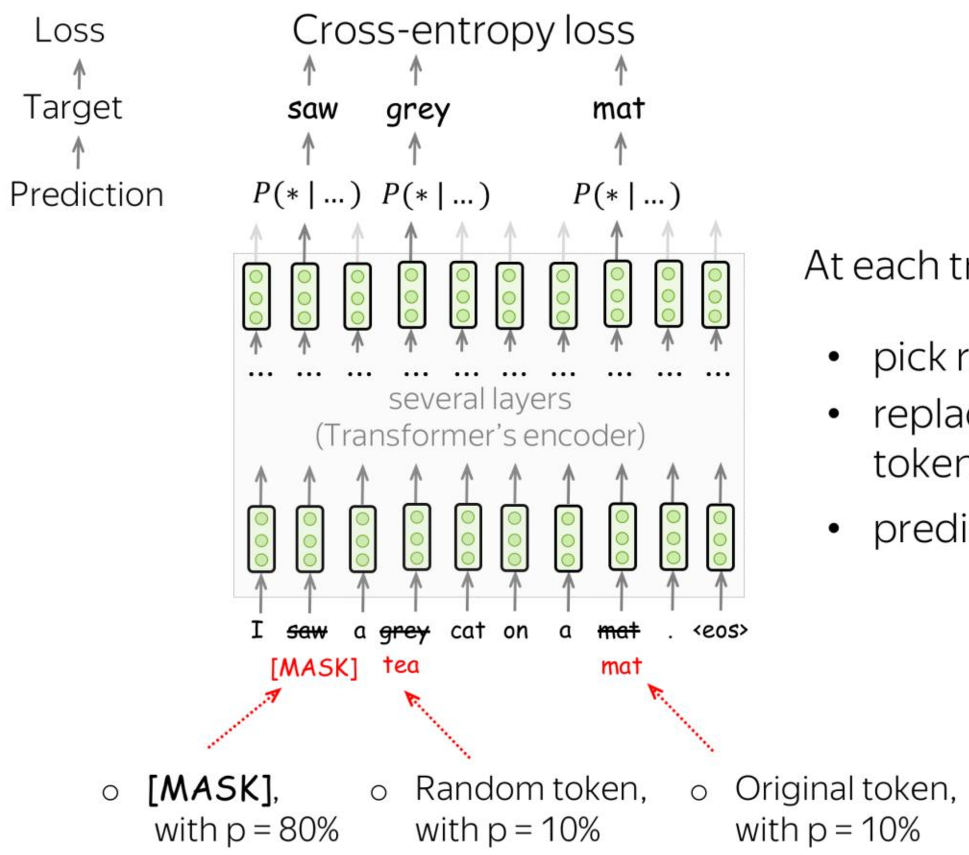
Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

Using encoder as an embedding generator



Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

Objective two: Masked Language Modeling



At each training step:

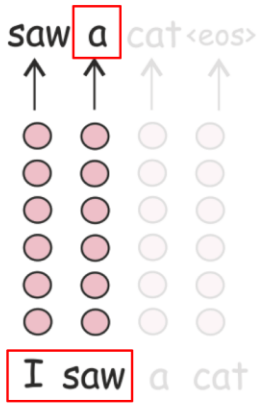
- pick randomly 15% of tokens
- replace each of the chosen tokens with something
- predict original chosen tokens

Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

LM vs. MLM

Language Modeling

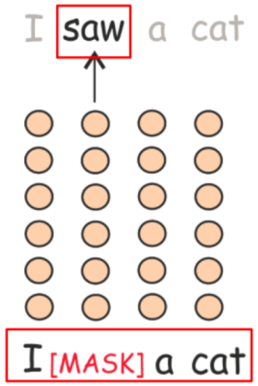
- Target: next token
- Prediction: $P(* \mid \text{I saw})$



left-to-right, does not see future

Masked Language Modeling

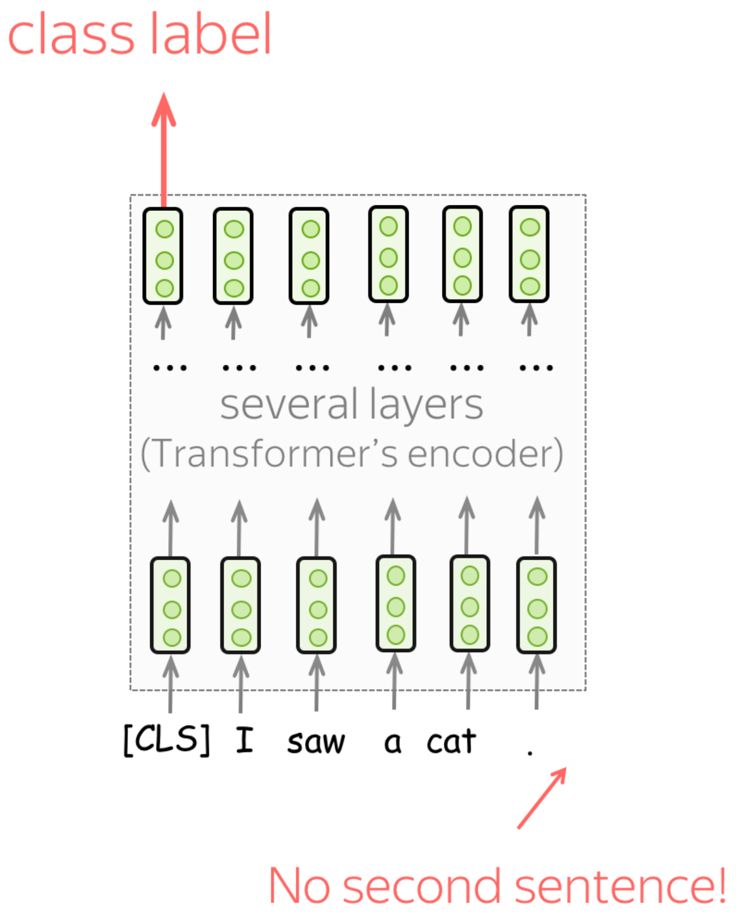
- Target: current token (the true one)
- Prediction: $P(* \mid \text{I [MASK] a cat})$



sees the whole text, but something is corrupted

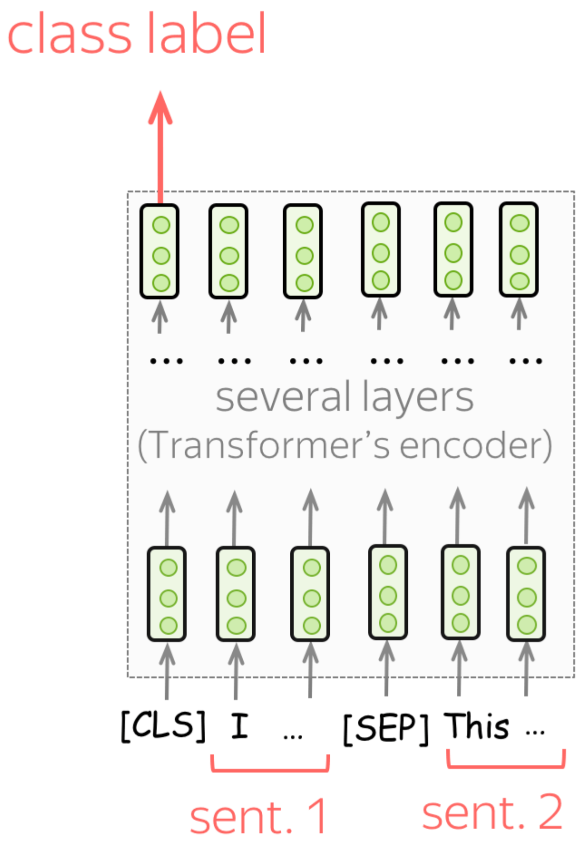
Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

Single Sentence Classification



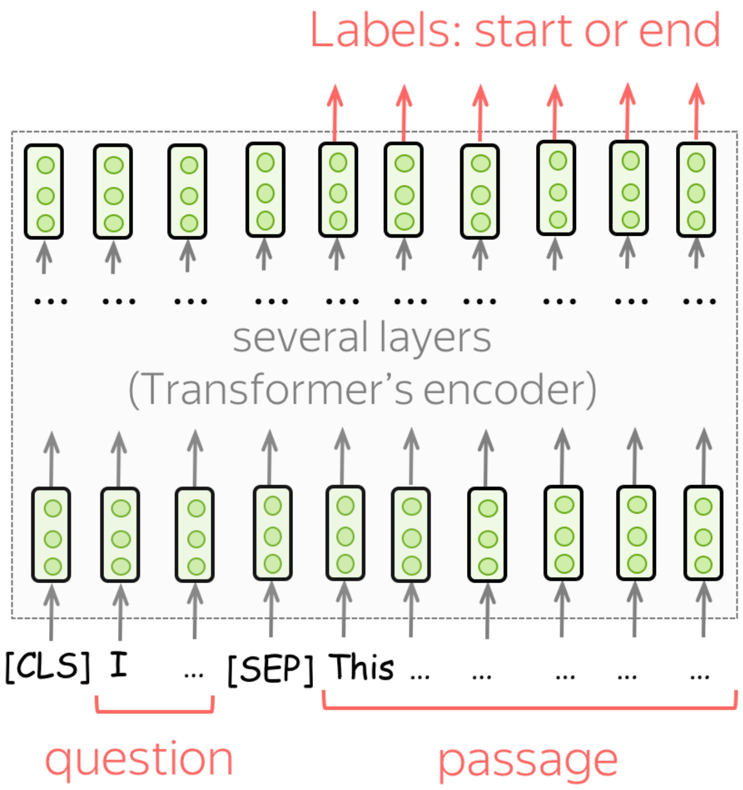
Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

Sentence Pair Classification



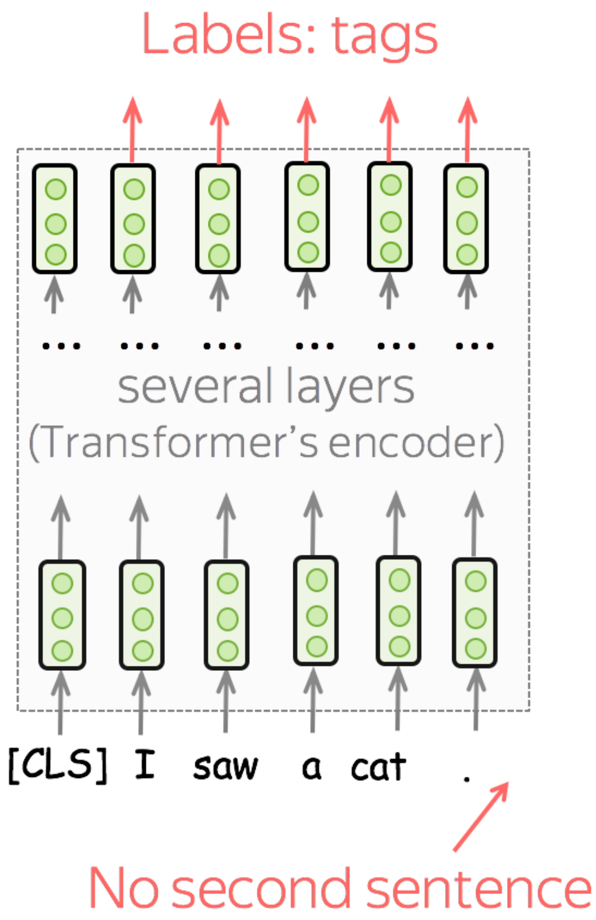
Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

Question Answering



Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

Input tagging



Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

Table of Content

- The power of transfer learning
- From word-specific to contextual embeddings
- Transformer architecture overview
- BERT
- **GPT**

GPT is just a Decoder part

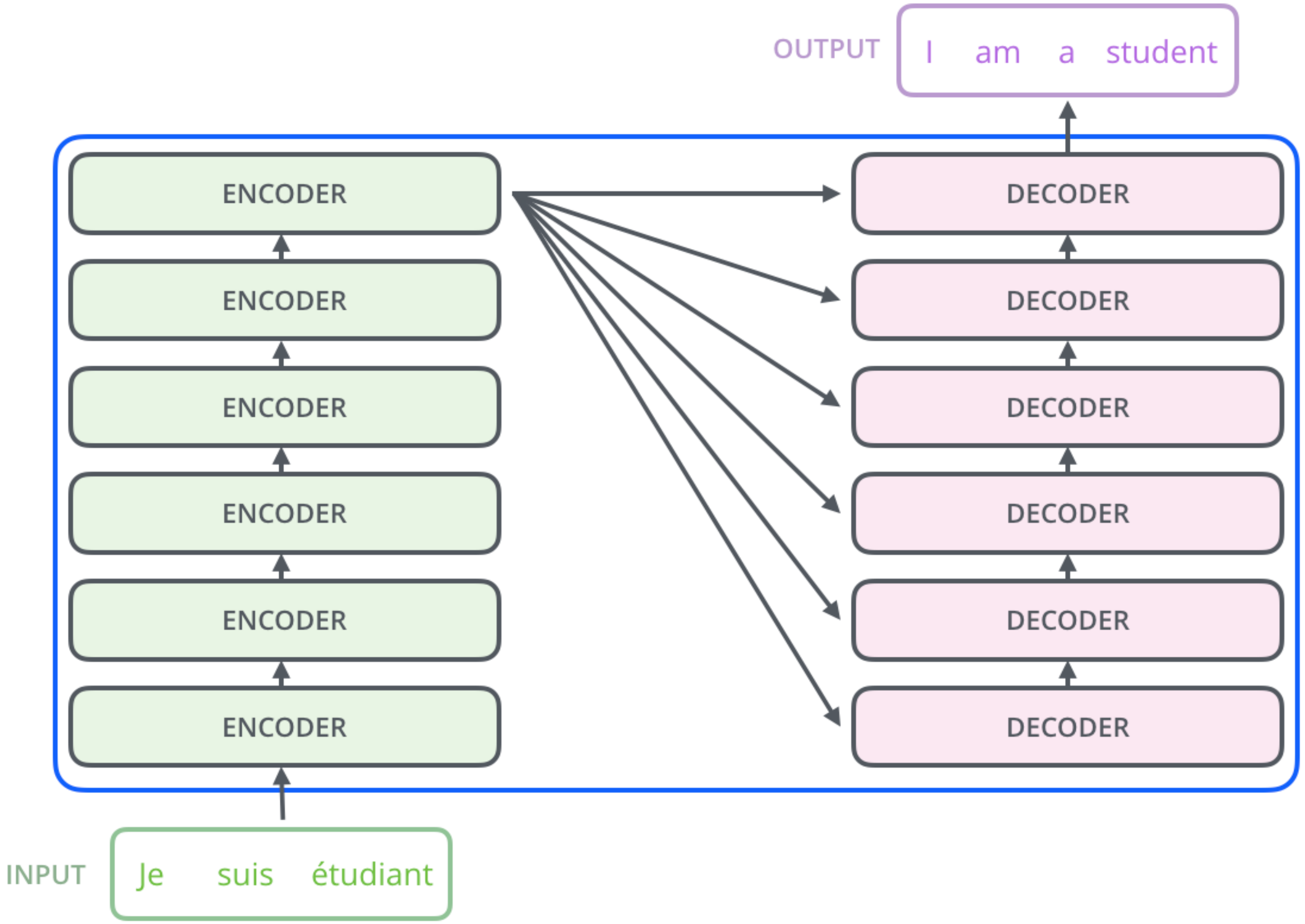


Image source: https://jalammar.github.io/illustrated-transformer/

Decoder as a universal model

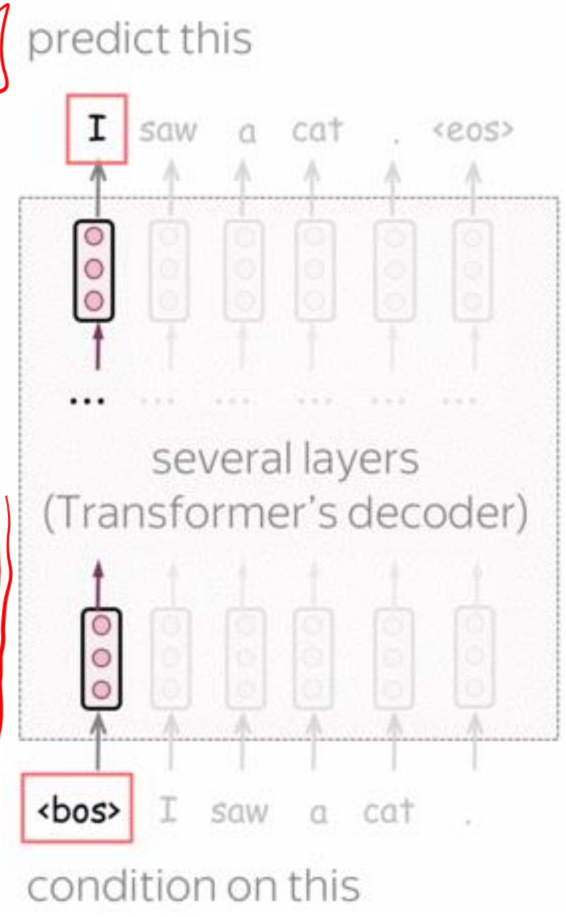
inp=[bos, thinking, machines]

out=[thinking, machines, eos]

$softmax(QK^T + mask)$

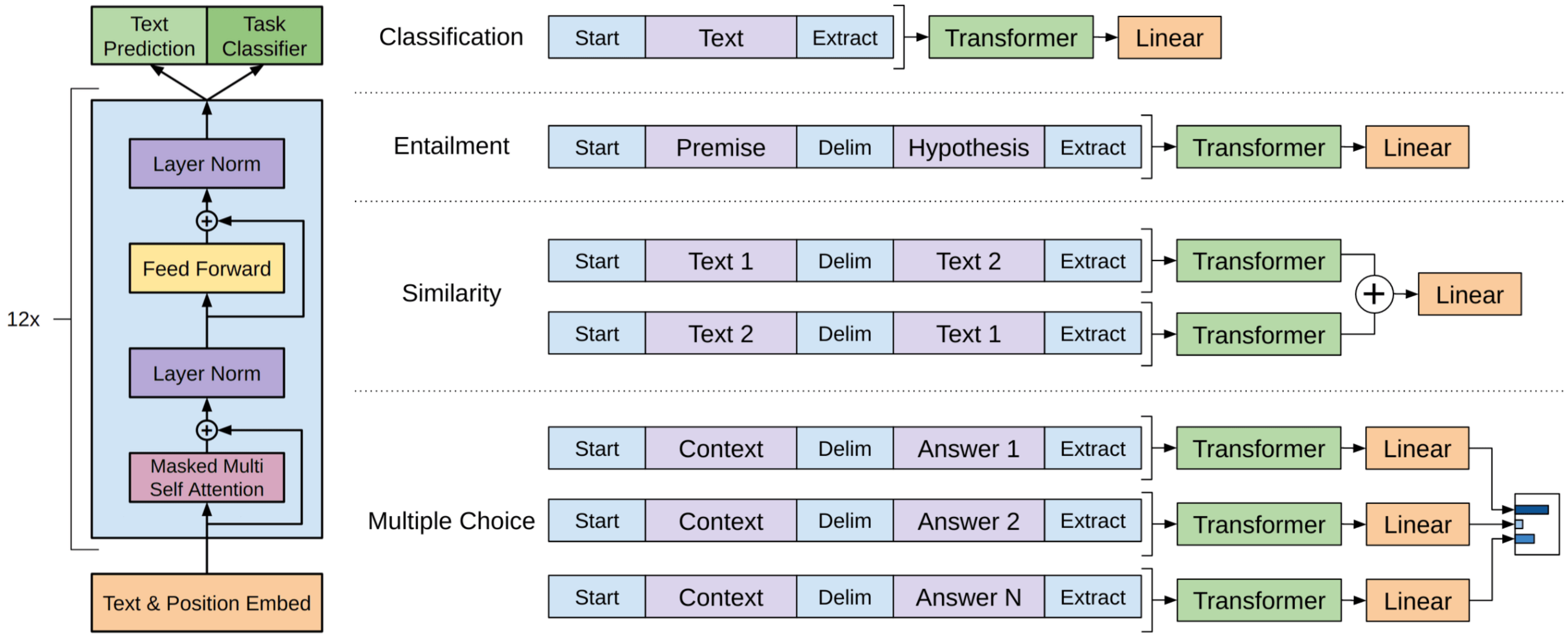
$$\begin{bmatrix} \langle q_1, k_1 \rangle & \langle q_1, k_2 \rangle & \dots \\ & & \end{bmatrix} + \begin{bmatrix} 0 & -inf & \dots \\ 0 & 0 & -inf \\ 0 & 0 & 0 \end{bmatrix}$$

softmax



Source: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

Decoder as a universal model



Source: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

HSE UNIVERSITY • EST 1992 •