Natural Language Processing: Transformers

HSE Faculty of Computer Science

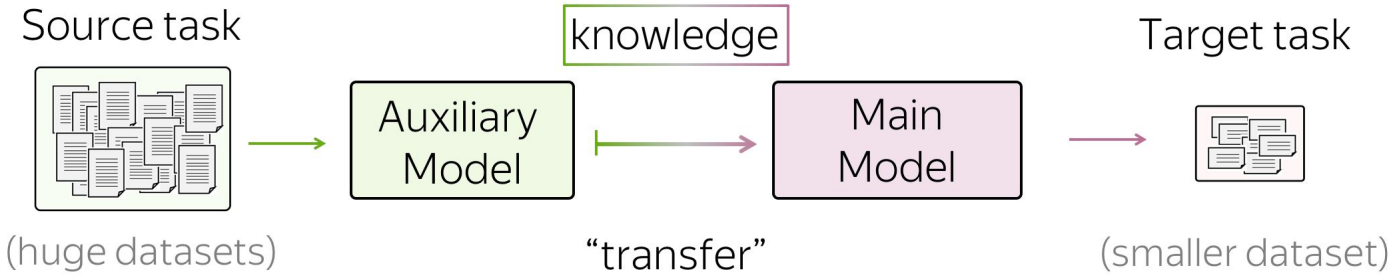
Machine Learning and Data-Intensive Systems

Murat Khazhgeriev

Table of Content

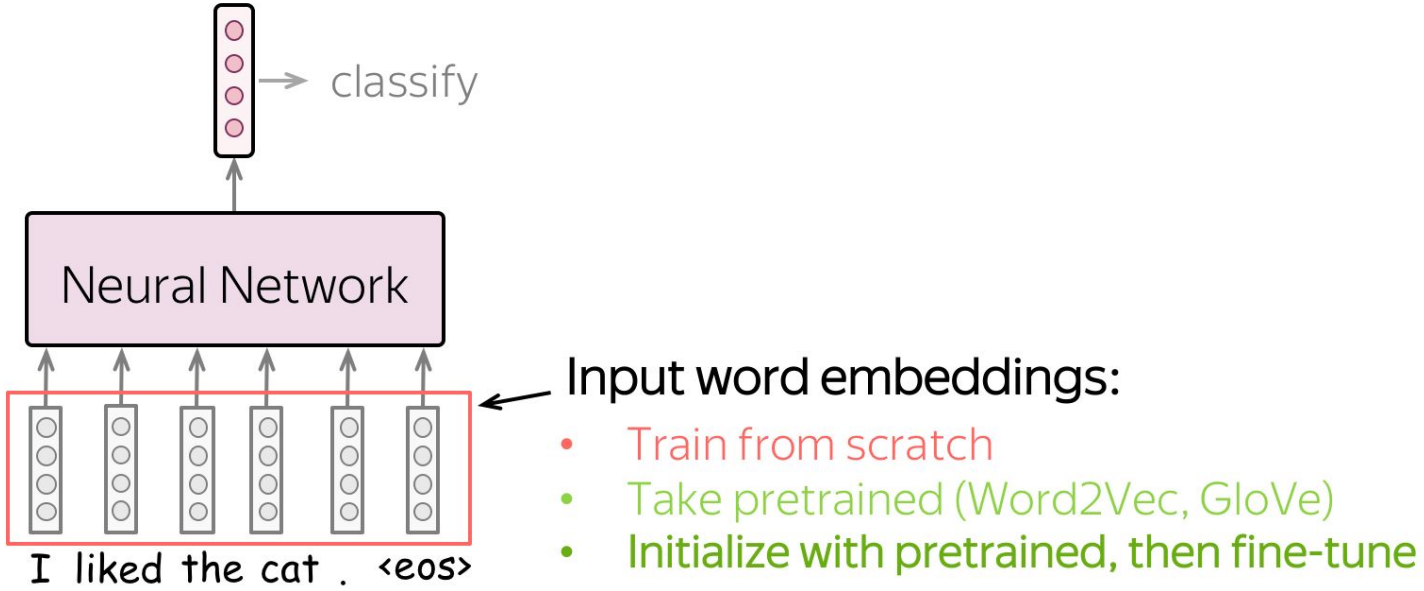
- **The power of transfer learning**
- From word-specific to contextual embeddings
- Transformer architecture overview
- BERT
- GPT

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



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

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



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

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

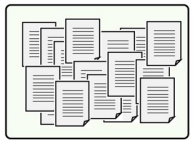
- Train from scratch

What they will know:

May be not enough to learn relationships between words

- Take pretrained (Word2Vec, GloVe)

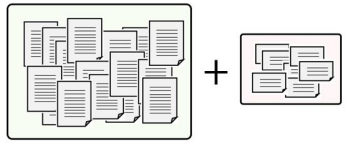
What they will know:



Know relationships between words, but are **not** specific to the task

- **Initialize with pretrained, then fine-tune**

What they will know:



Know relationships between words and adapted for the task

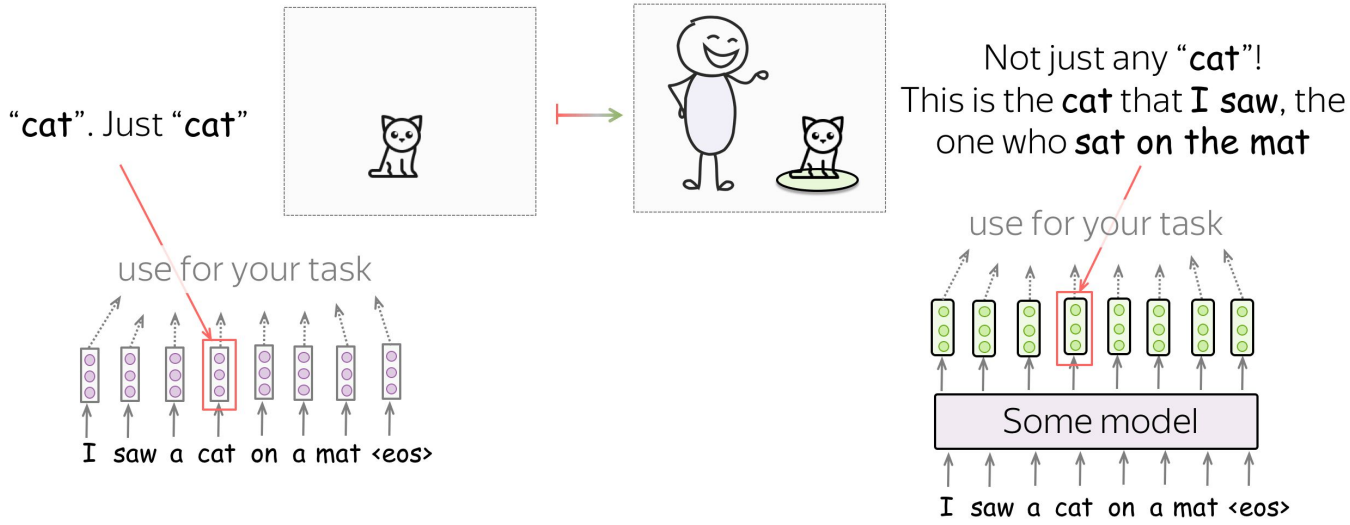
"**Transfer**" knowledge from a huge unlabeled corpus to your task-specific model

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

Table of Content

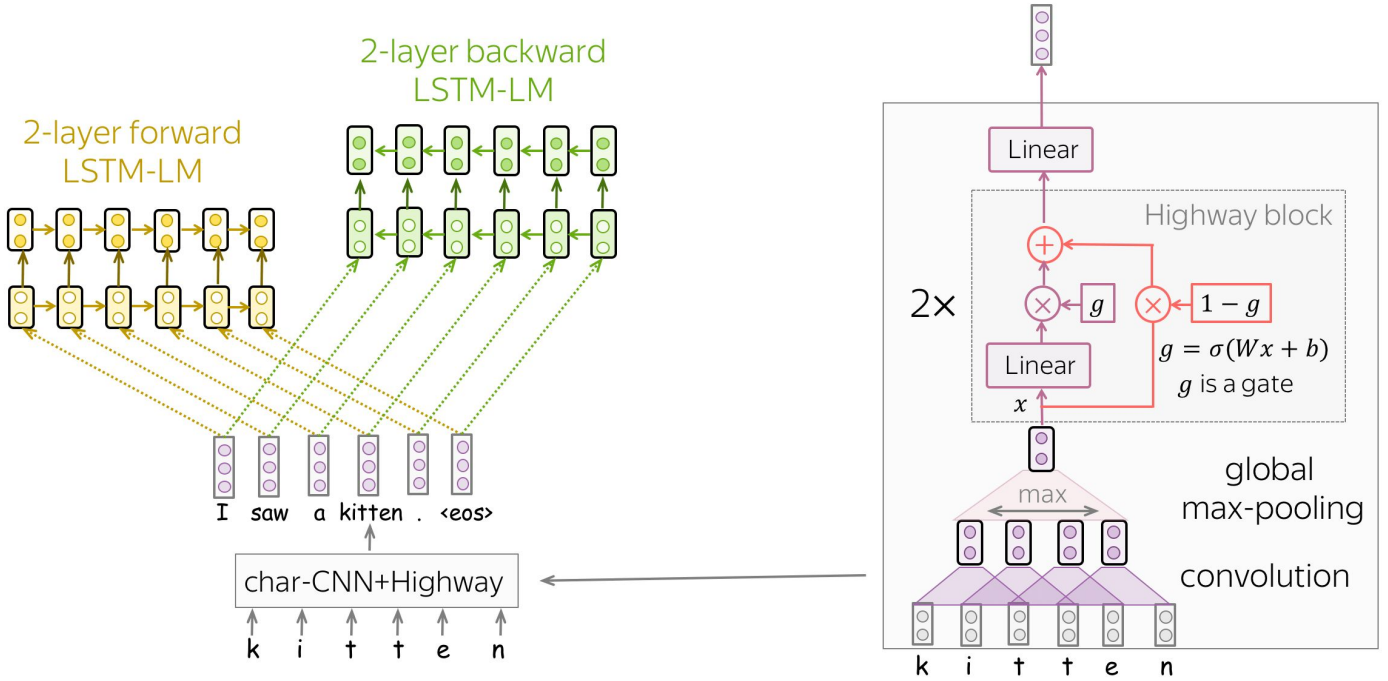
- The power of transfer learning
- **From word-specific to contextual embeddings**
- Transformer architecture overview
- BERT
- GPT

Not just a cat, but the cat!



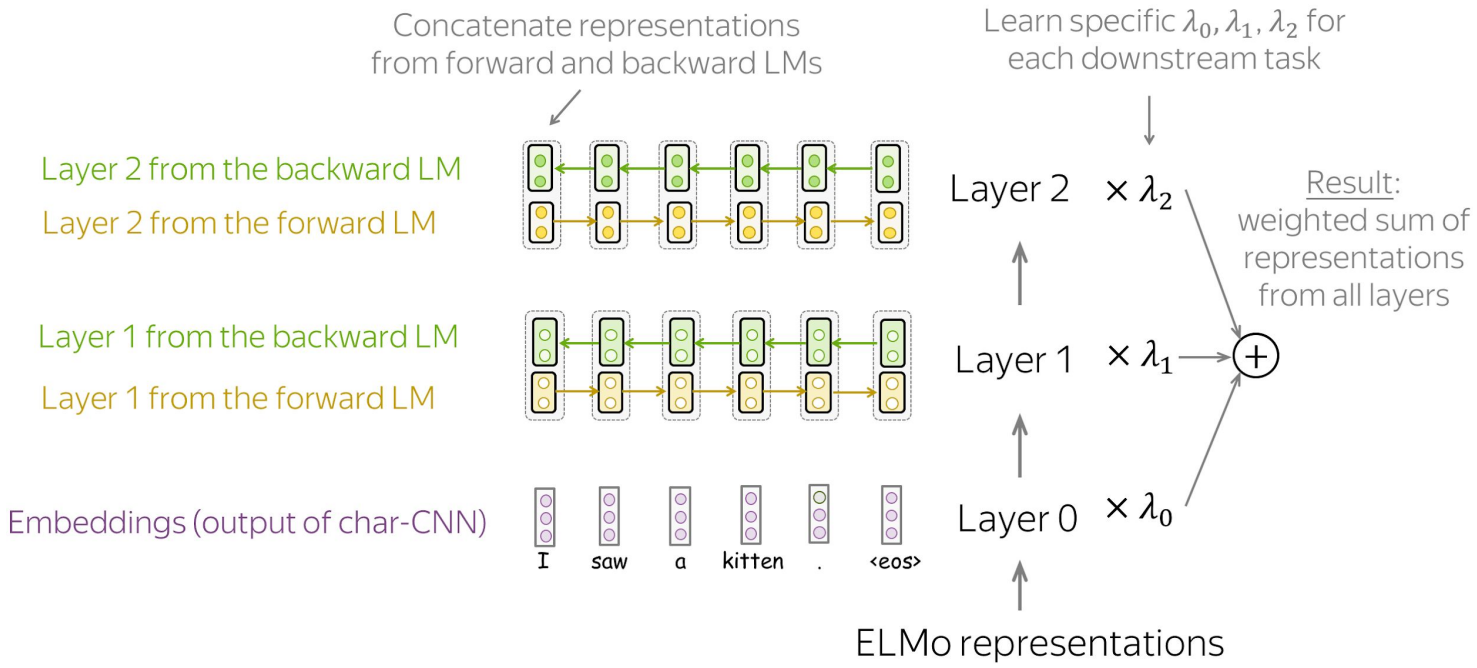
Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

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



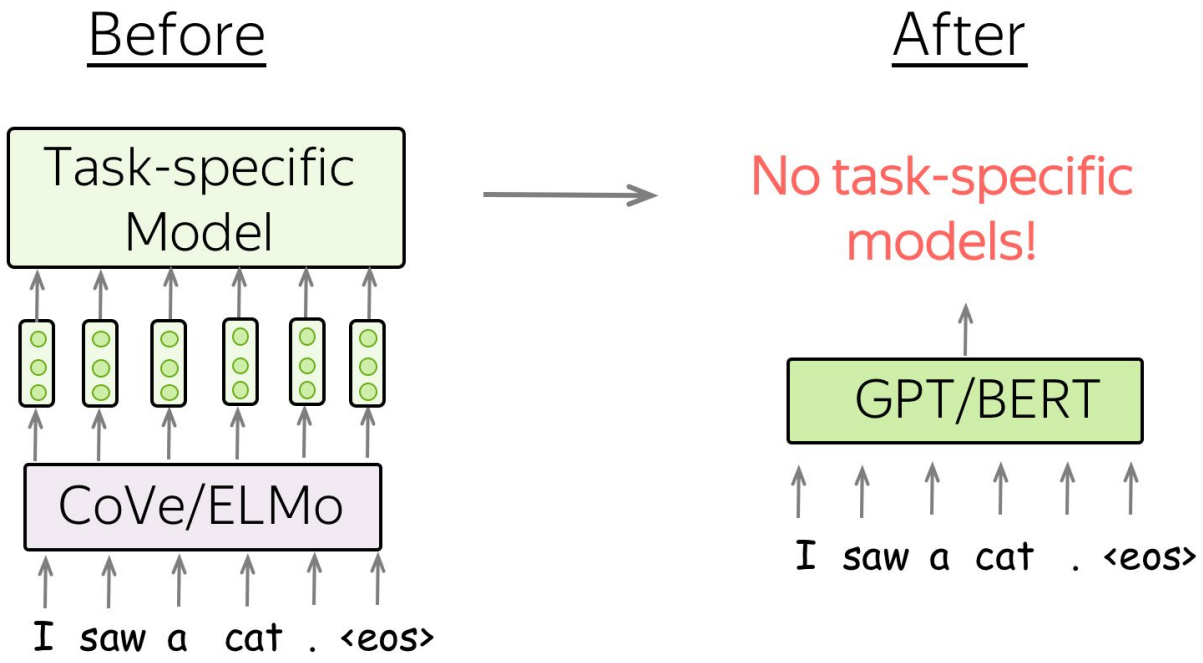
Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

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



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

From an embedding generator to a universal model



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

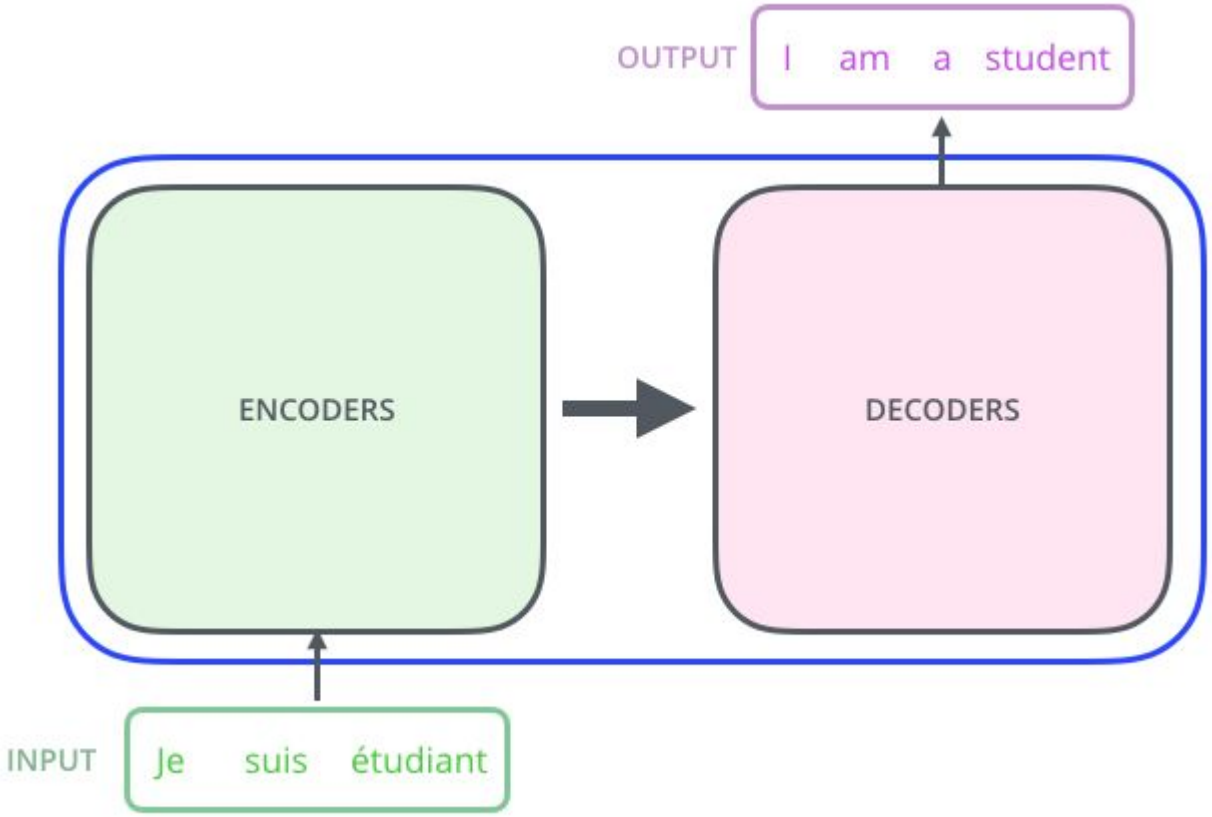
Table of Content

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

Transformer is an example of Encoder-Decoder architecture

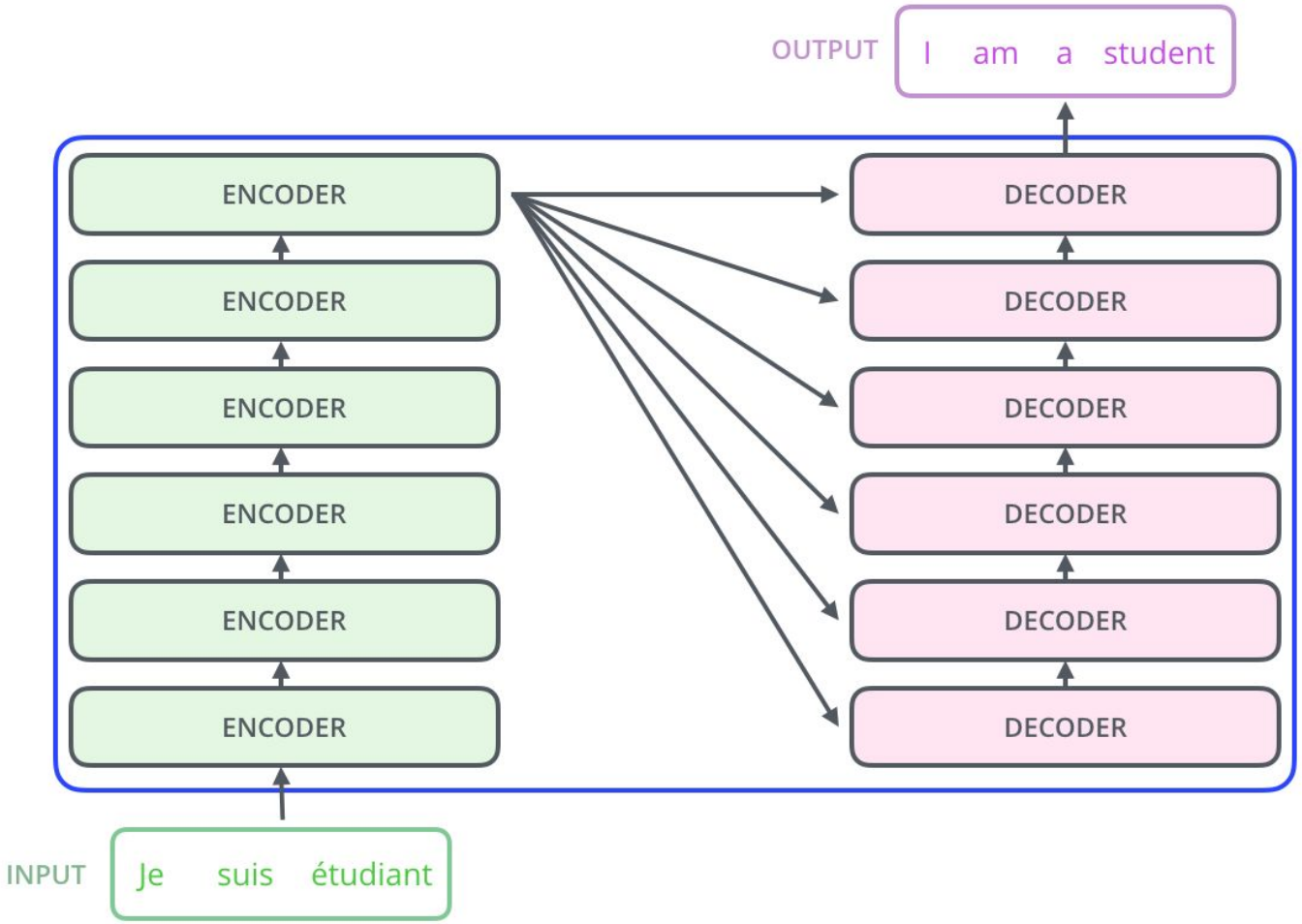
Source: https://jalammar.github.io/illustrated-transformer/

Transformer is an example of Encoder-Decoder architecture



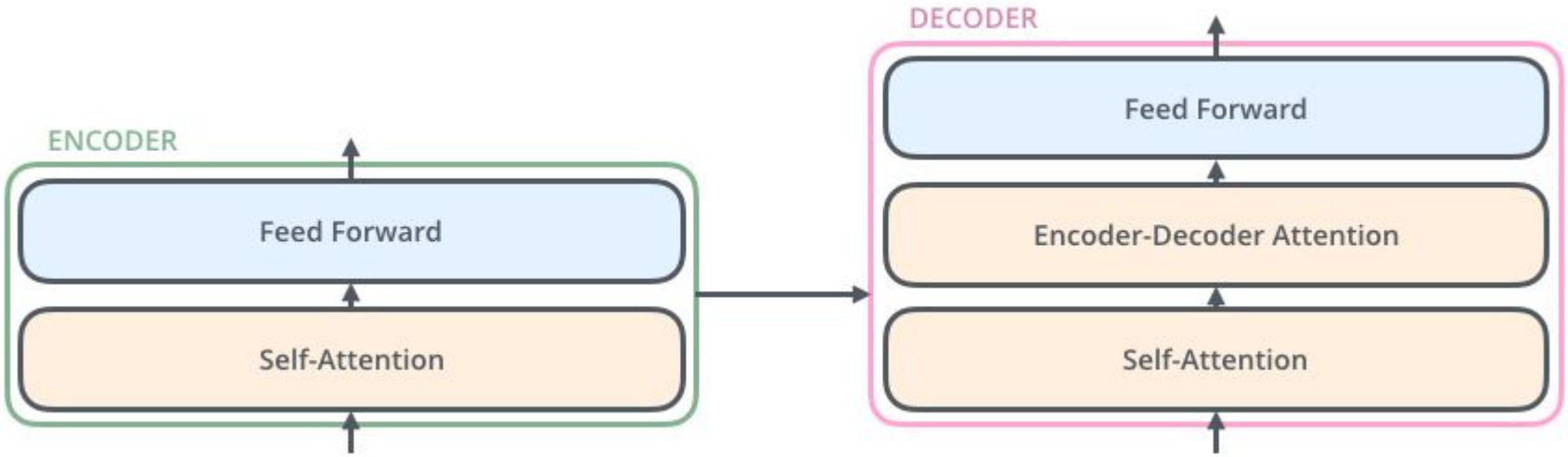
Source: https://jalammar.github.io/illustrated-transformer/

Transformer is an example of Encoder-Decoder architecture



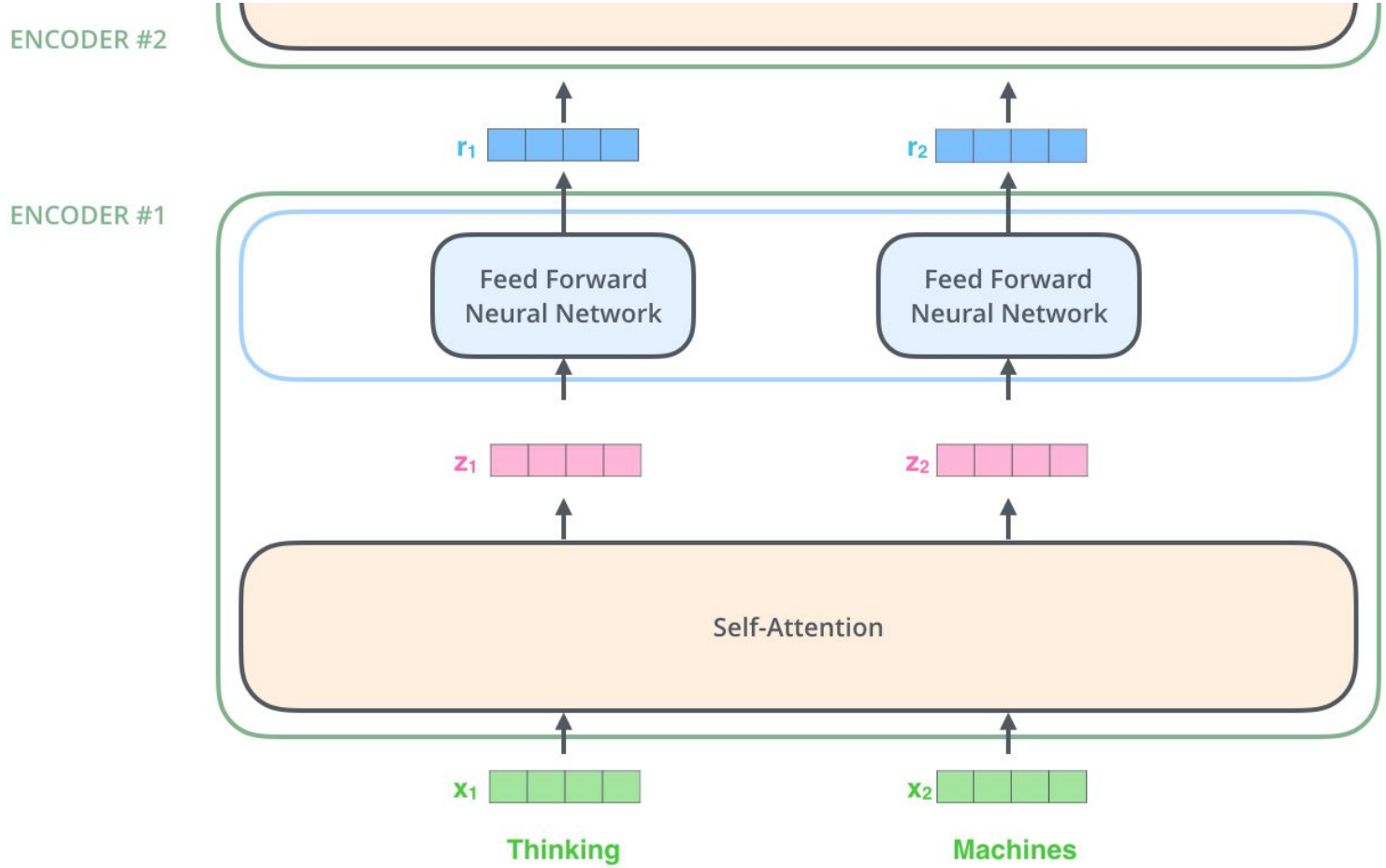
Source: https://jalammar.github.io/illustrated-transformer/

Always two there are. No more, no less. An Attention and an FFN. © Yoda



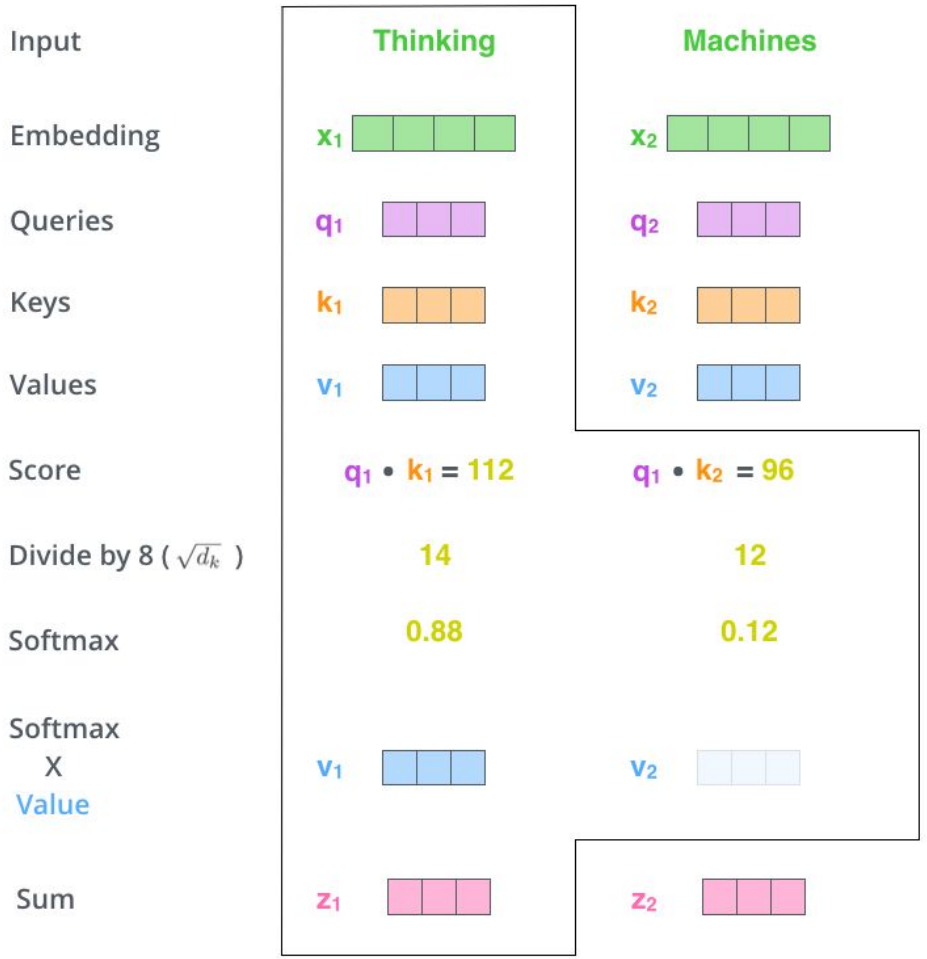
Source: https://jalammar.github.io/illustrated-transformer/

Always two there are. No more, no less. An Attention and an FFN. © Yoda



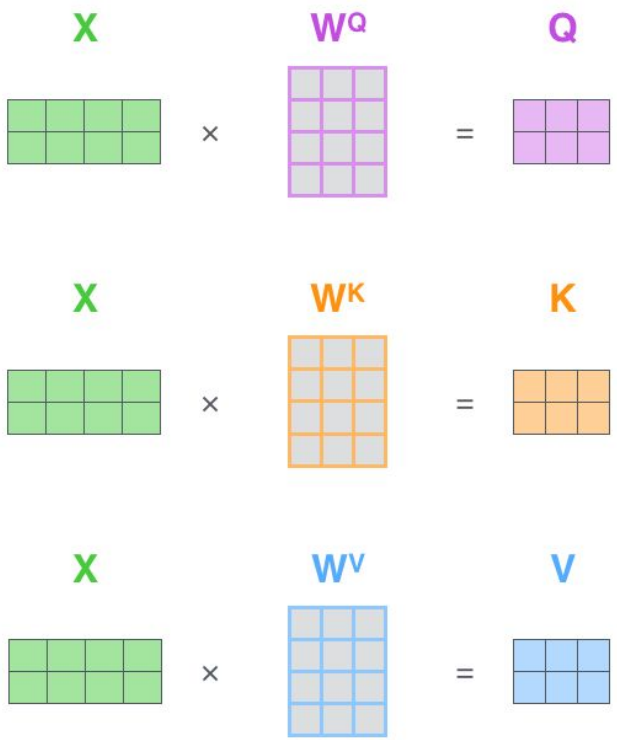
Source: https://jalammar.github.io/illustrated-transformer/

Inside a Self-Attention



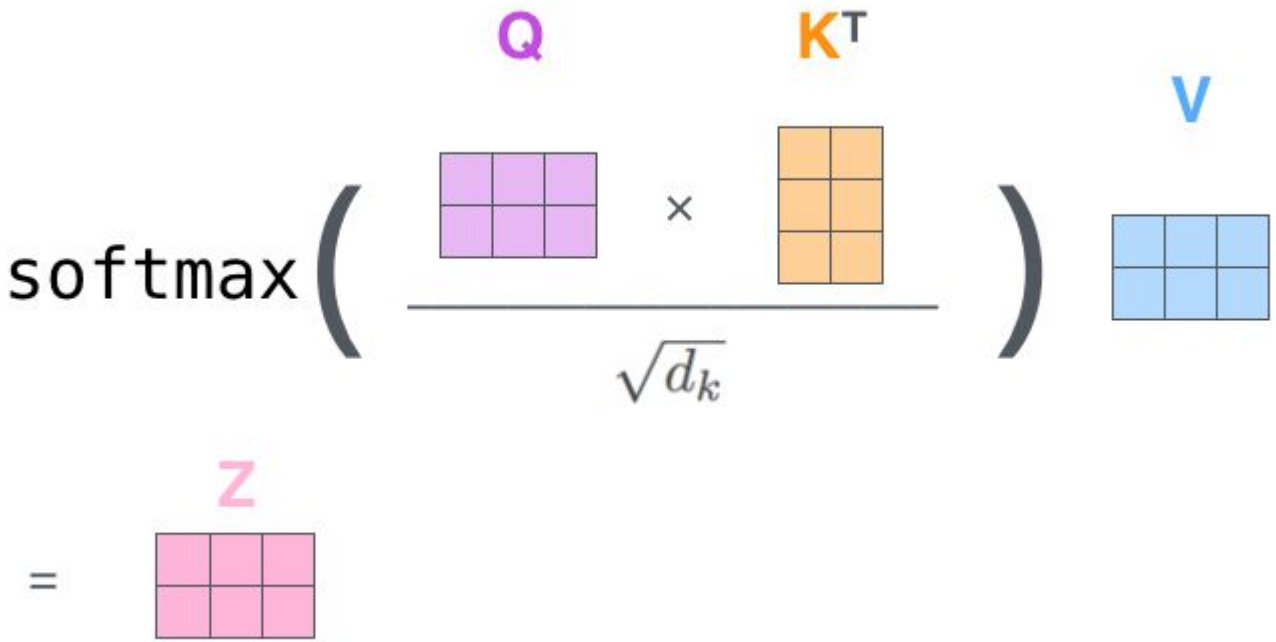
Source: https://jalammar.github.io/illustrated-transformer/

Inside a Self-Attention: Matrix View



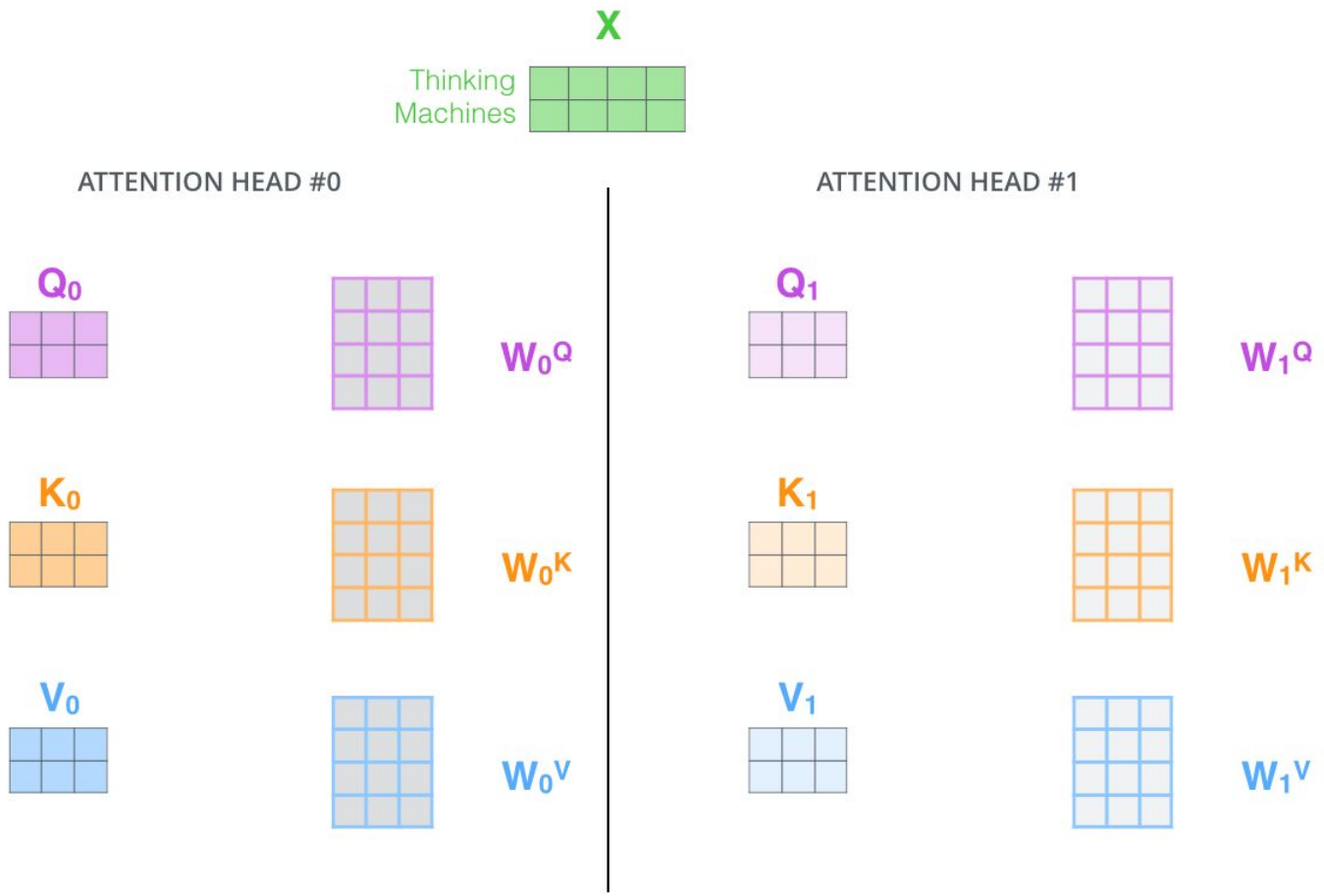
Source: https://jalammar.github.io/illustrated-transformer/

Inside a Self-Attention: Matrix View



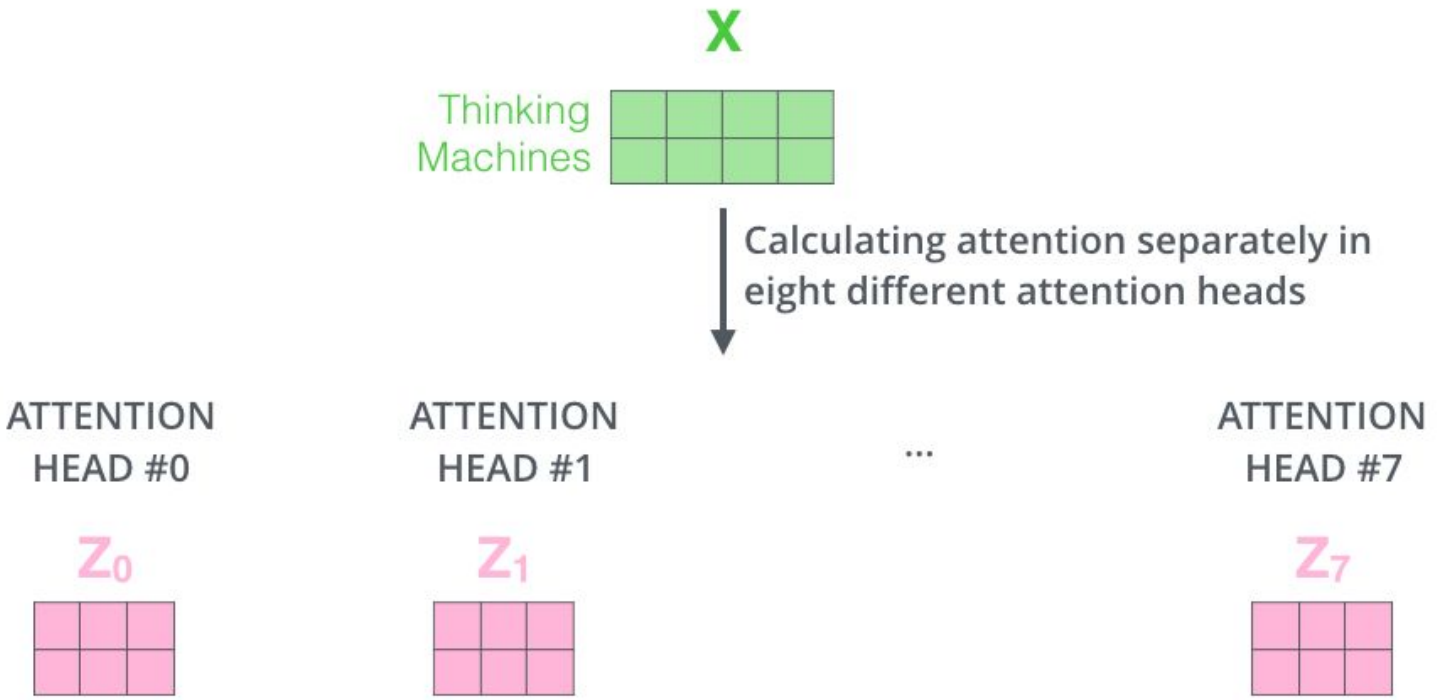
Source: https://jalammar.github.io/illustrated-transformer/

A beast with many heads



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

A beast with many heads



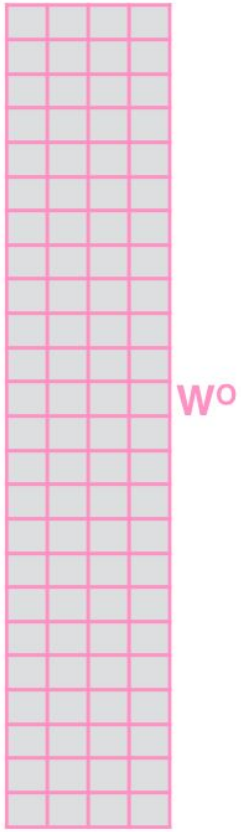
Source: https://jalammar.github.io/illustrated-transformer/

A beast with many heads

1) Concatenate all the attention heads

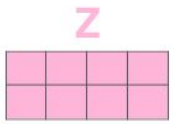
2) Multiply with a weight matrix W^O that was trained jointly with the model

X



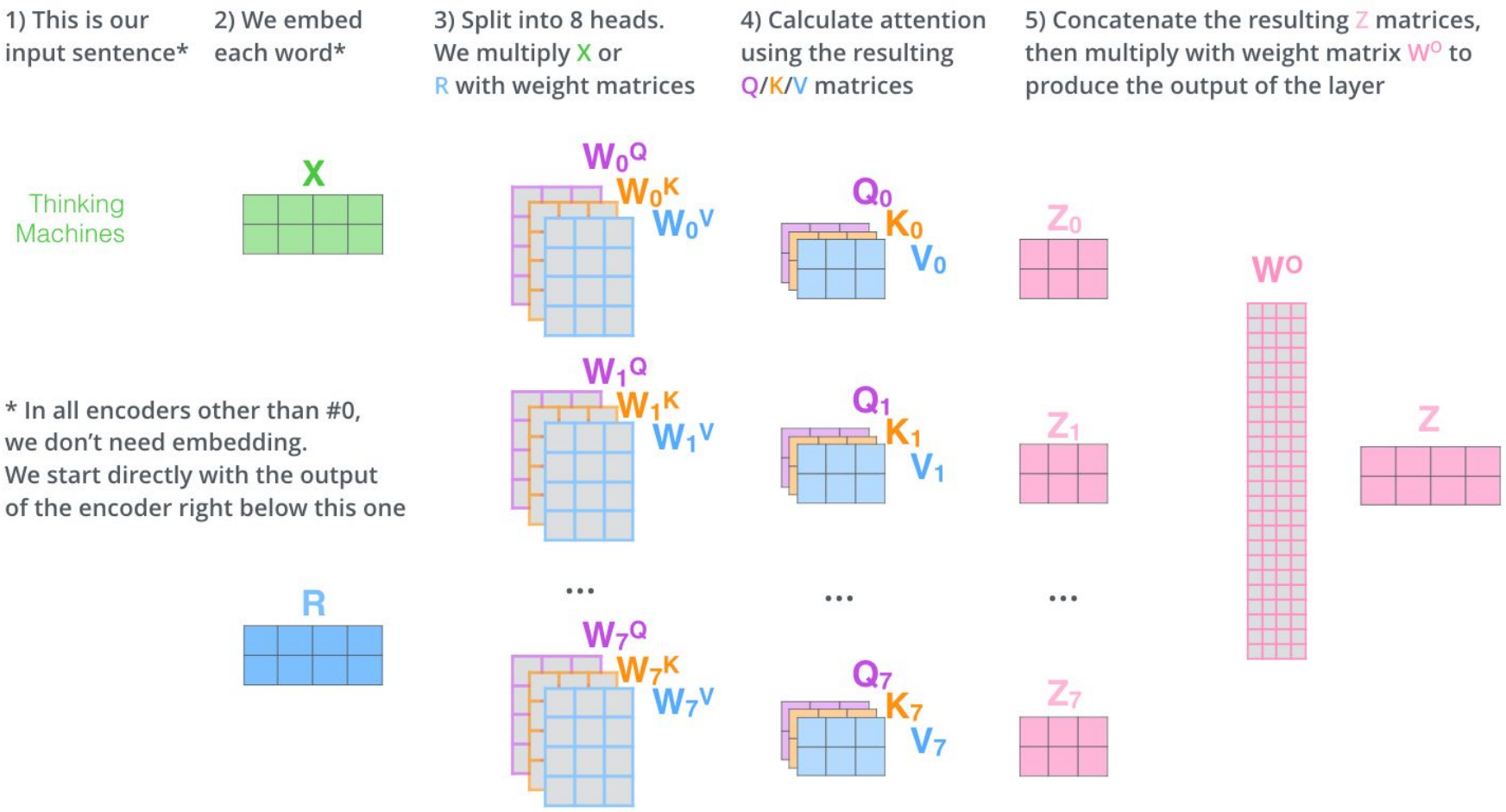
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

=



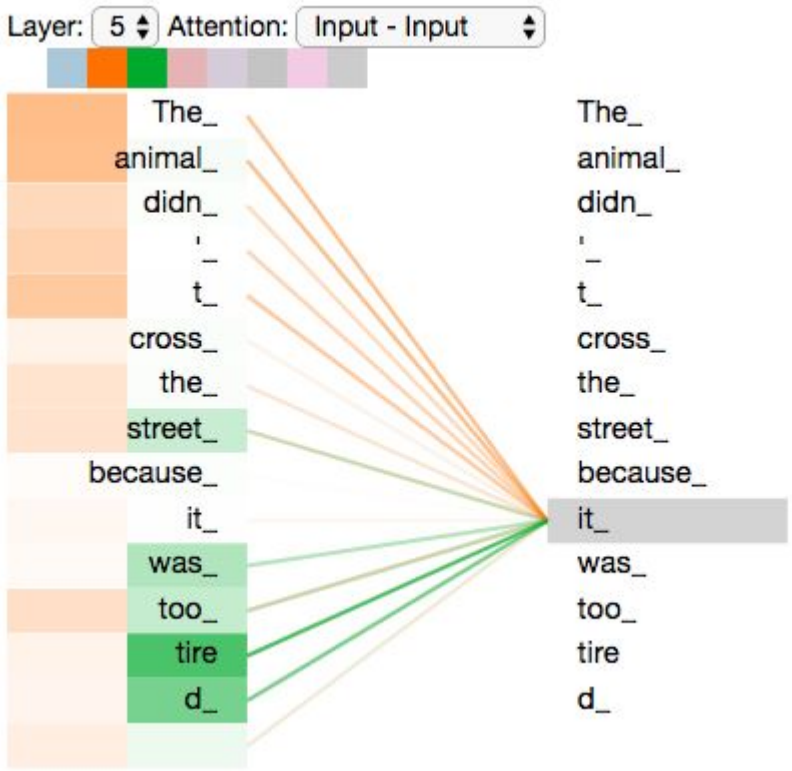
Source: https://jalammar.github.io/illustrated-transformer/

A multi-head attention overview



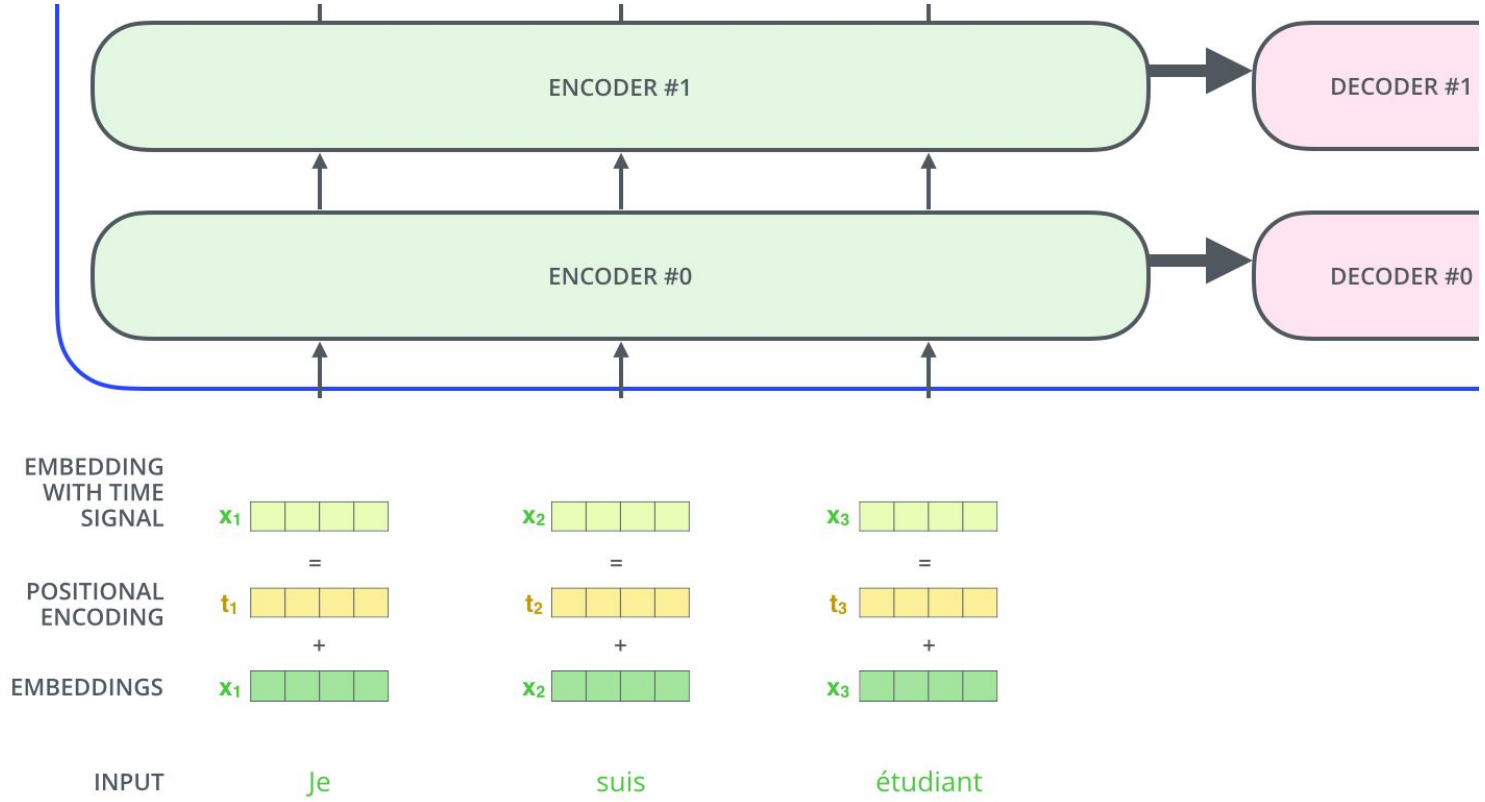
Source: https://jalammar.github.io/illustrated-transformer/

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/

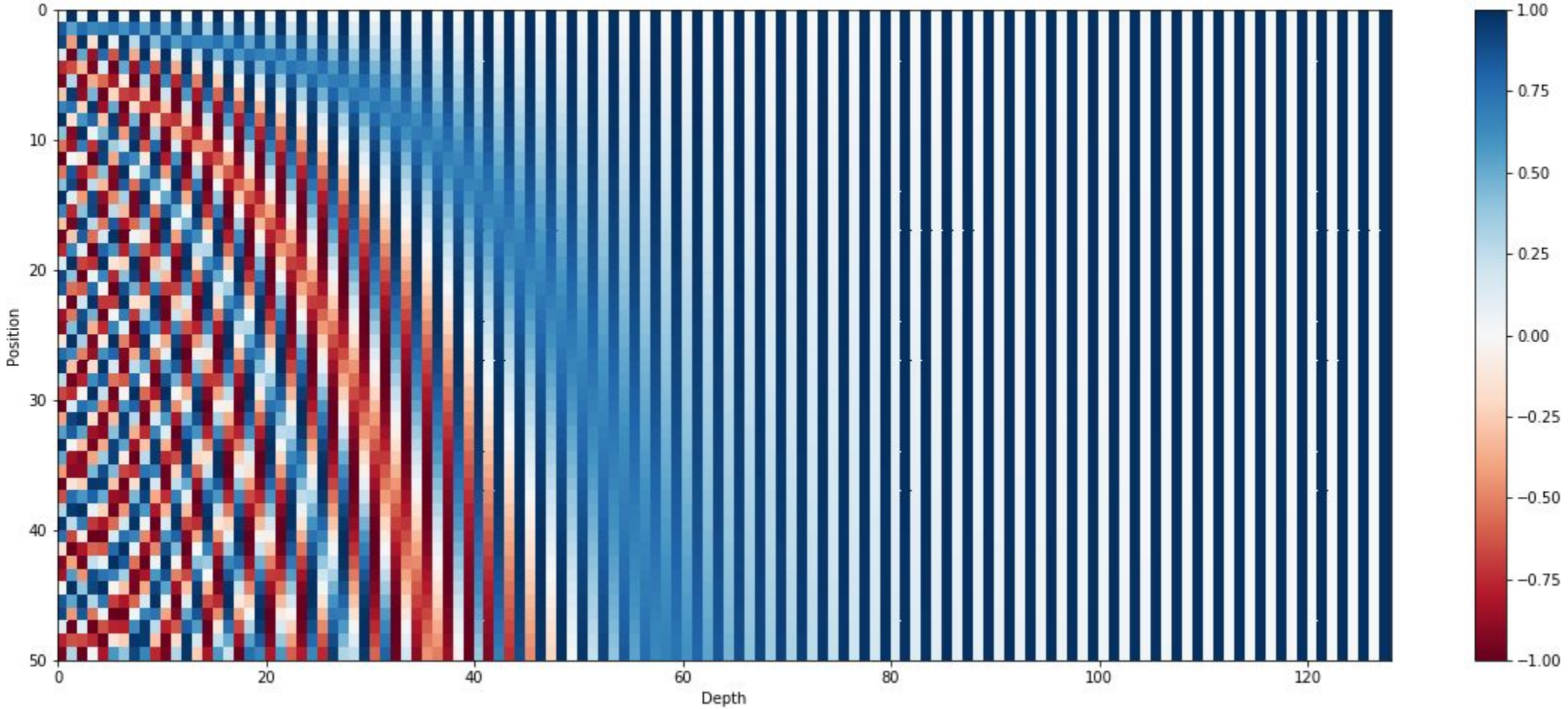
Absolute Positional Encoding

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

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

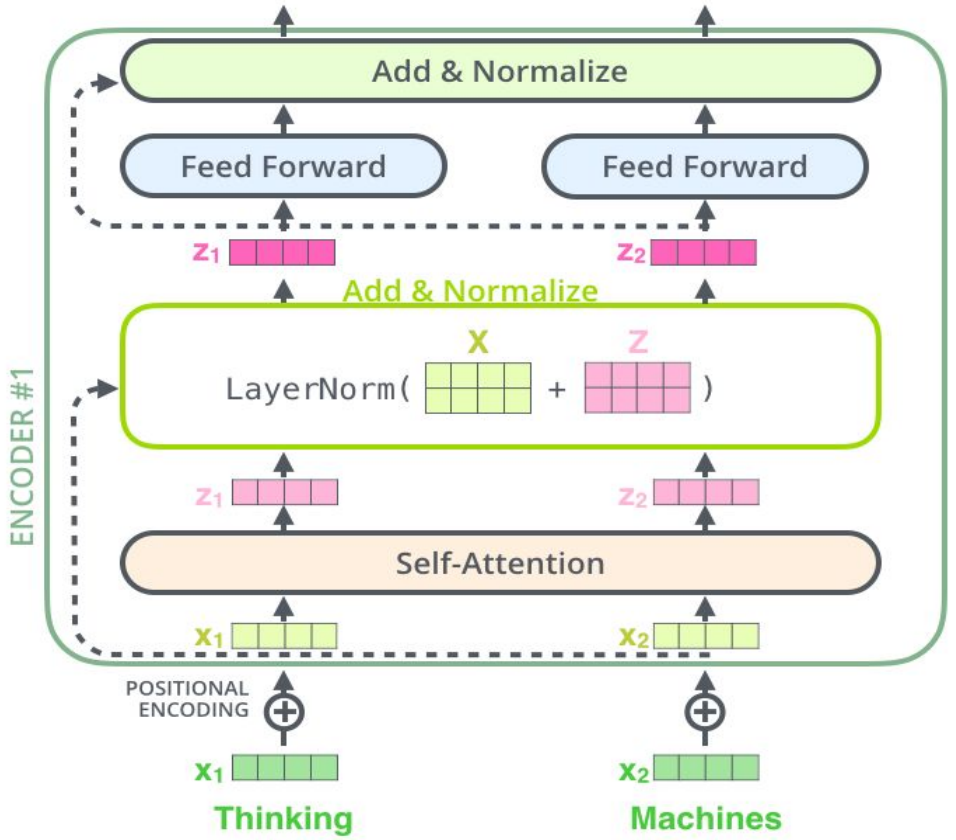
Source: https://arxiv.org/abs/1706.03762v7

What it looks like



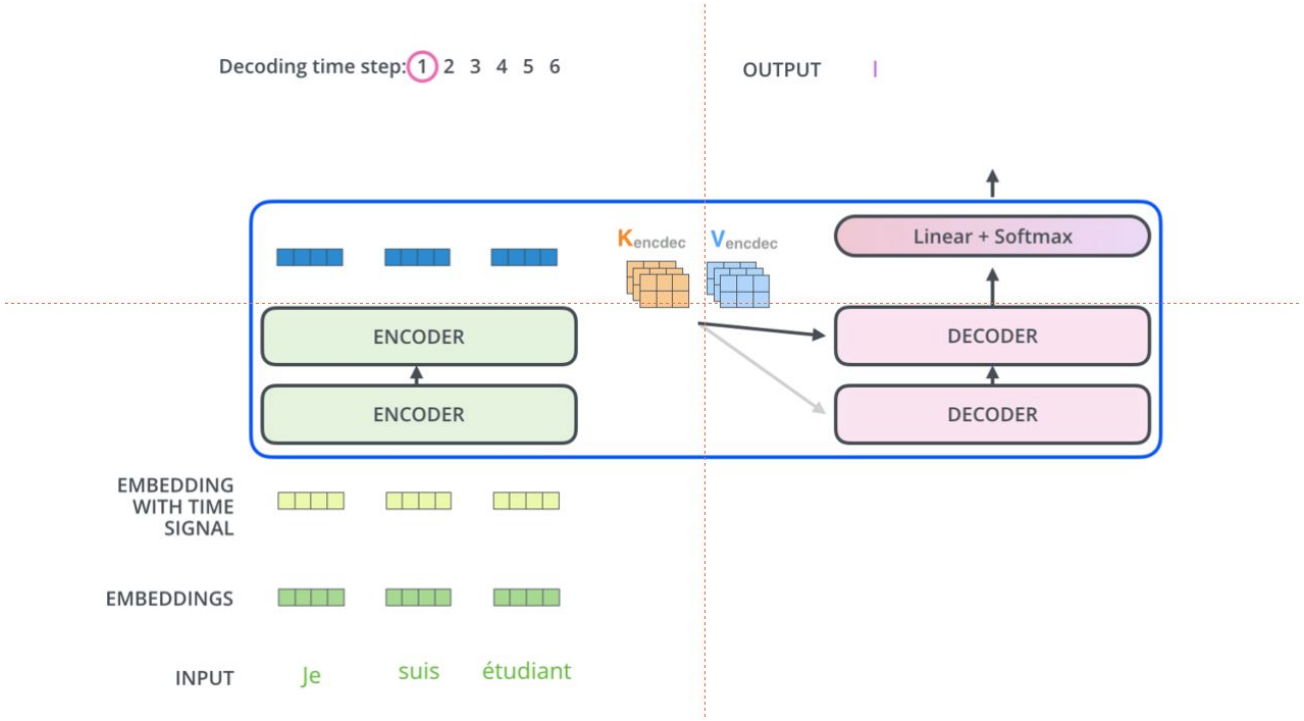
Source: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Skip Connection and Layer Normalization for a robust training



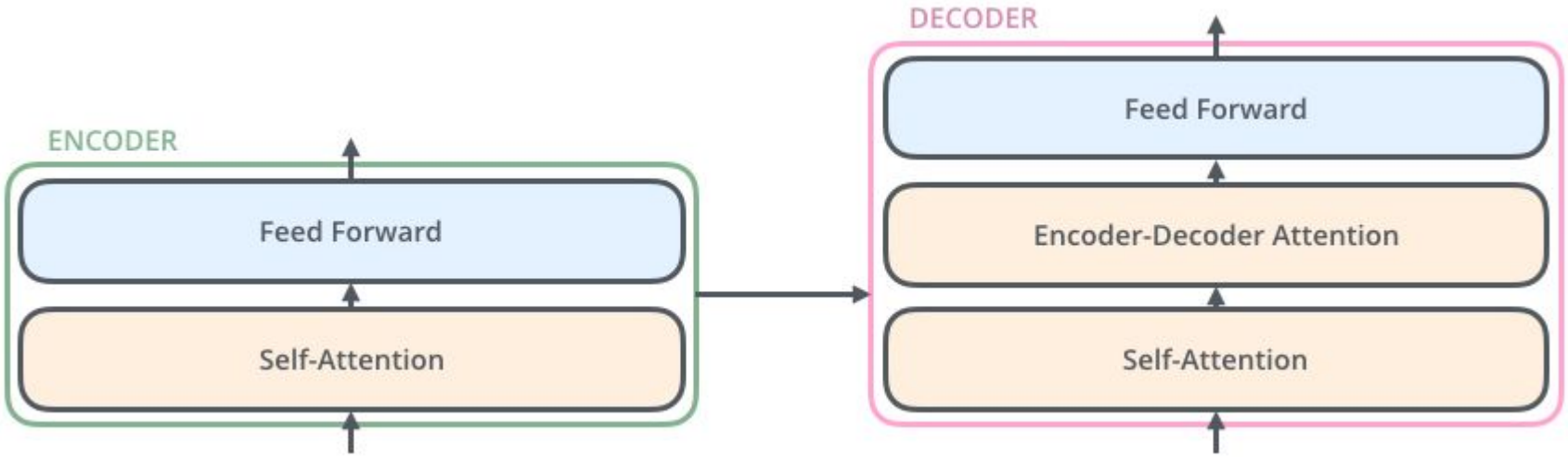
Source: https://jalammar.github.io/illustrated-transformer/

Bringing it all together



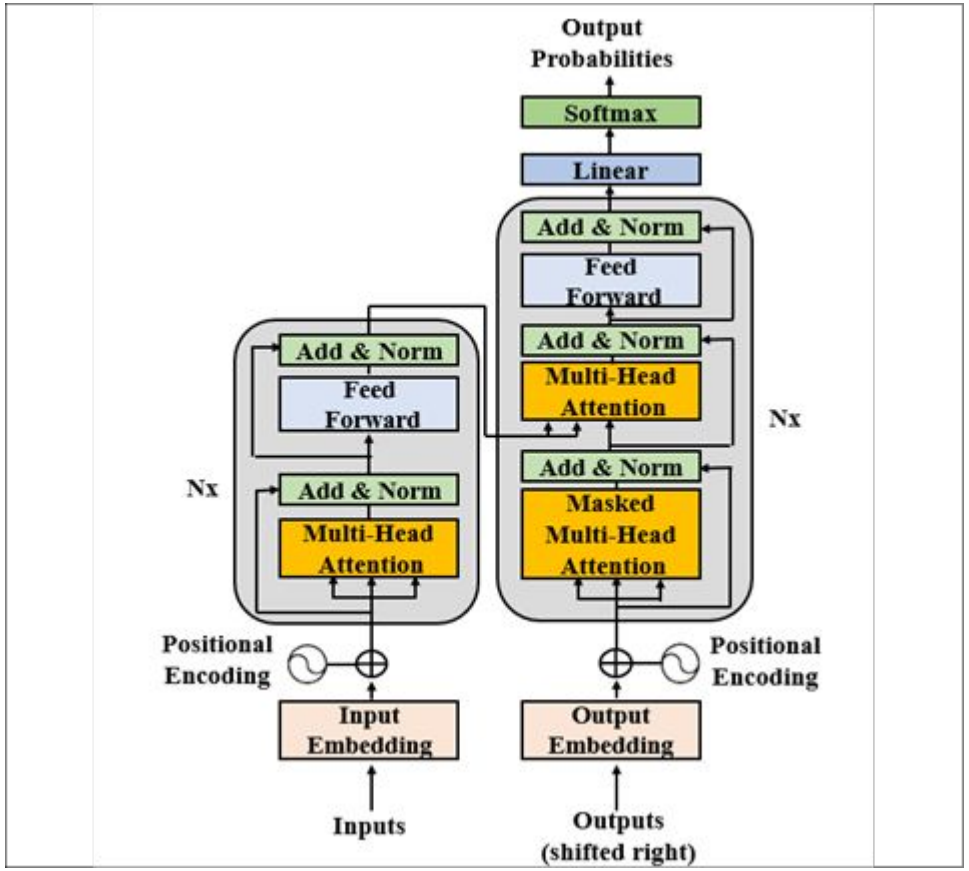
Source: https://jalammar.github.io/illustrated-transformer/

Always two there are. No more, no less. An Attention and an FFN. © Yoda



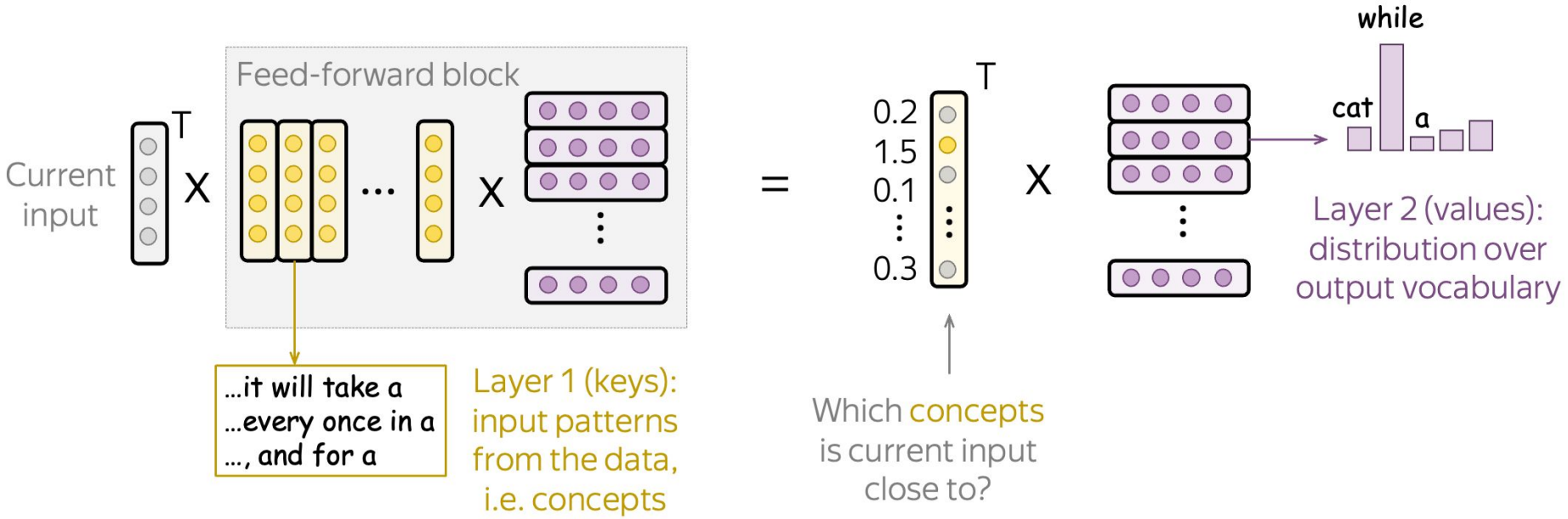
Source: https://jalammar.github.io/illustrated-transformer/

Attention is all you need



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

Attention is all you need (but not really)



Source: https://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

BERT is just an Encoder part

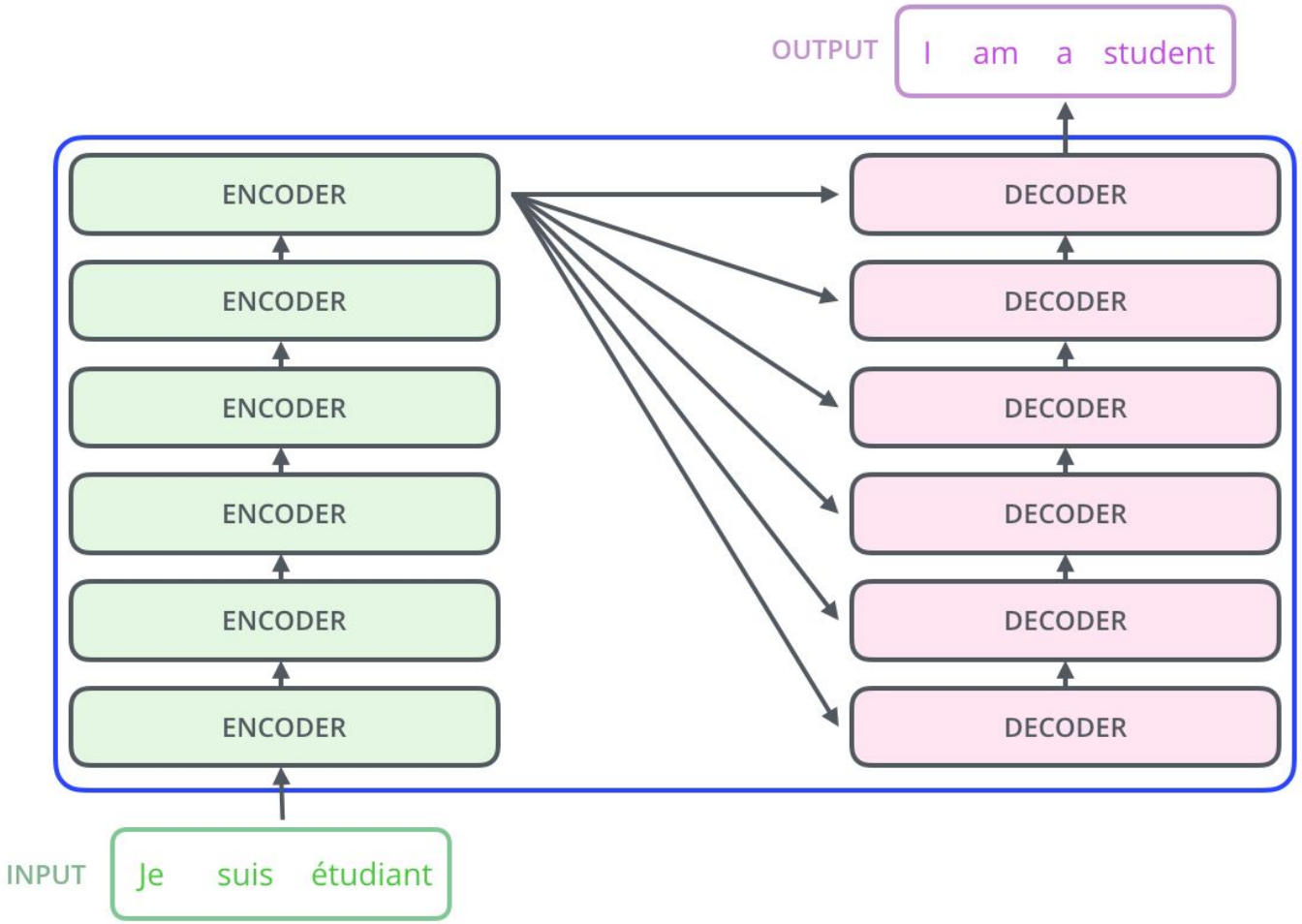
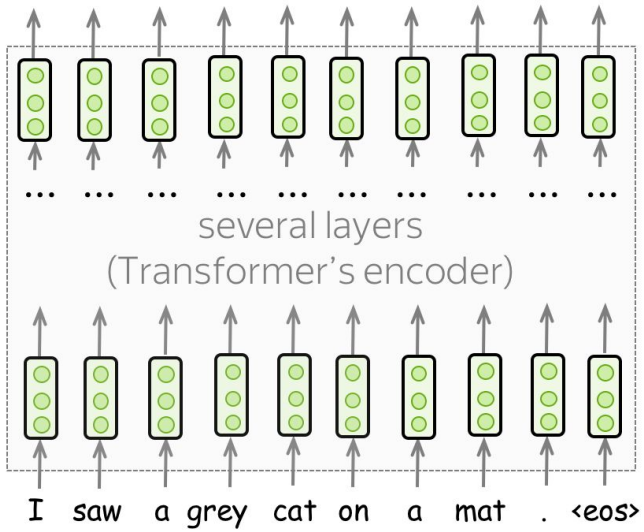


Image source: https://jalammar.github.io/illustrated-transformer/

Using encoder as an embedding generator



Model architecture:

- Transformer's encoder

What is special about it:

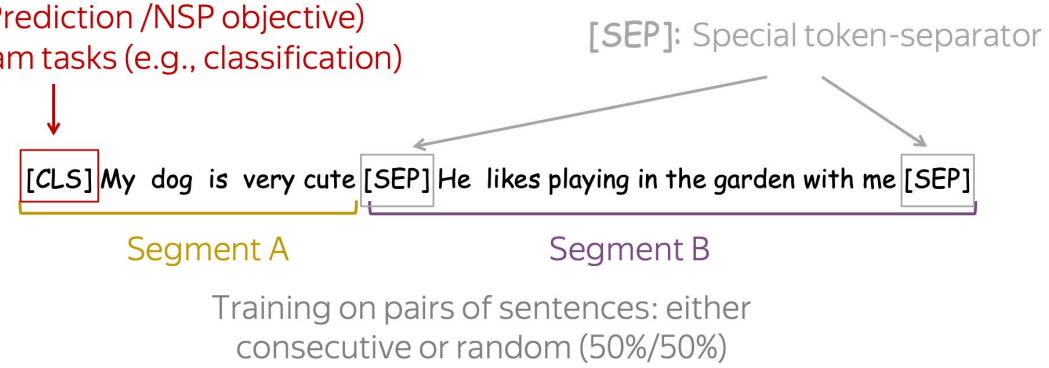
- Training objectives
  - MLM: Masked language modeling
  - NSP: Next sentence prediction
- The way it is used
  - No task-specific models

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

Objective one: tell whether the two sequences are consecutive

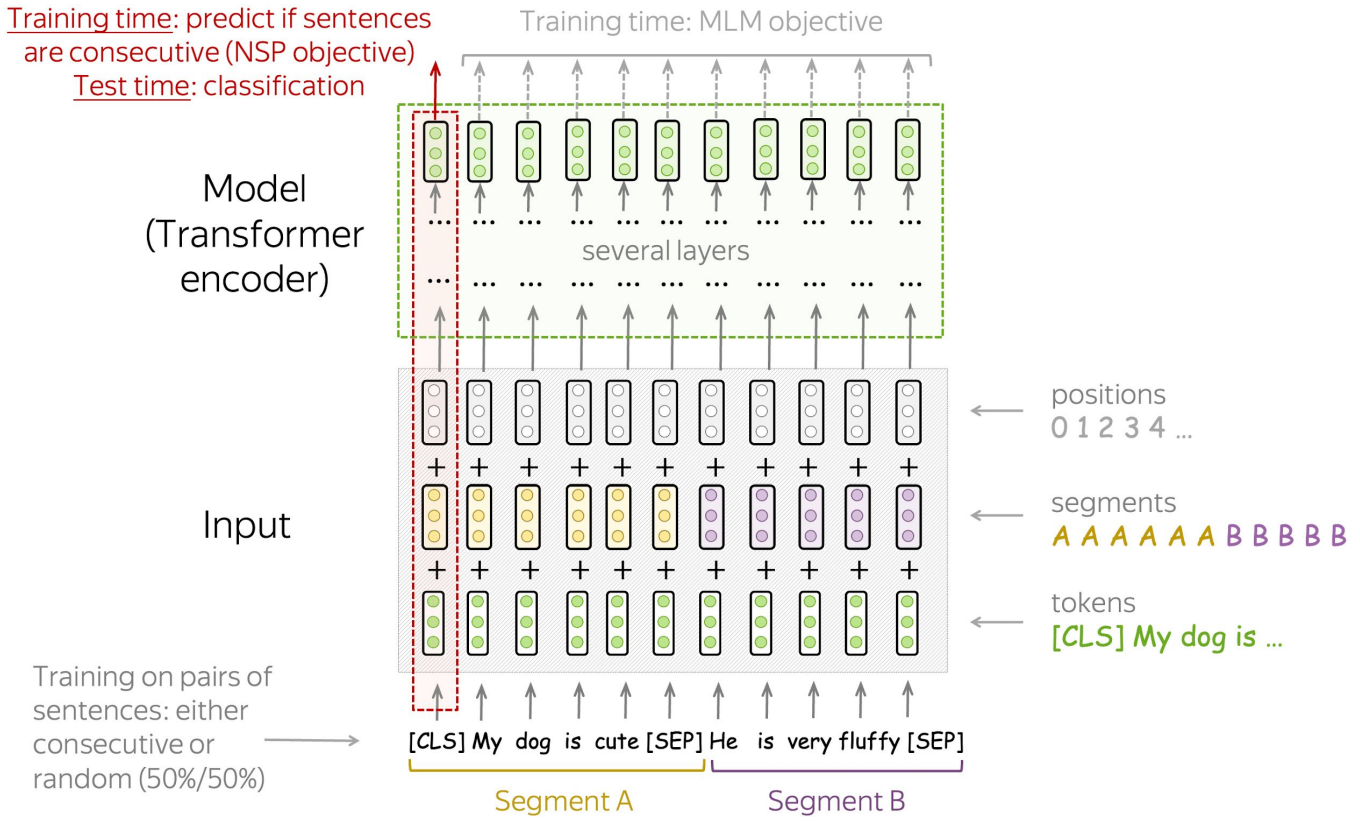
[CLS]: Special token

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



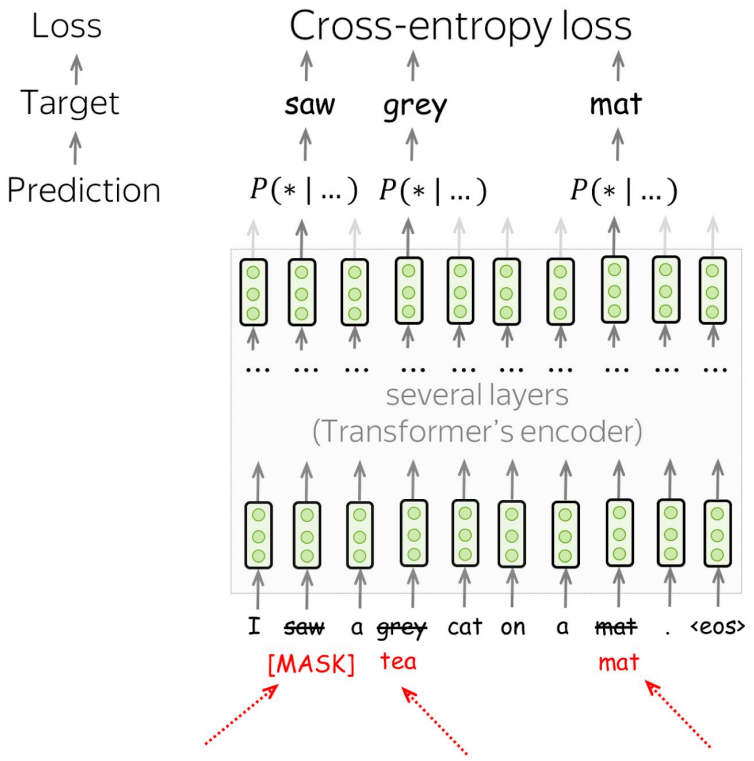
Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

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

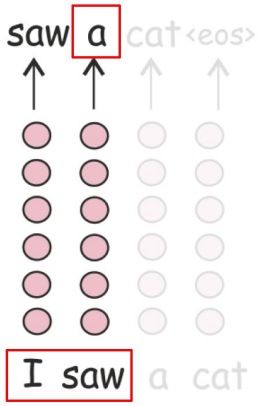
- **[MASK]**, with p = 80%
- Random token, with p = 10%
- Original token, with p = 10%

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

LM vs. MLM

Language Modeling

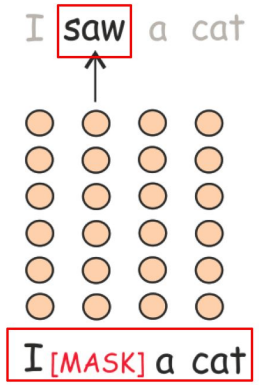
- Target: next token
- Prediction: $P(* \mid \texttt{I saw})$



left-to-right, does not see future

Masked Language Modeling

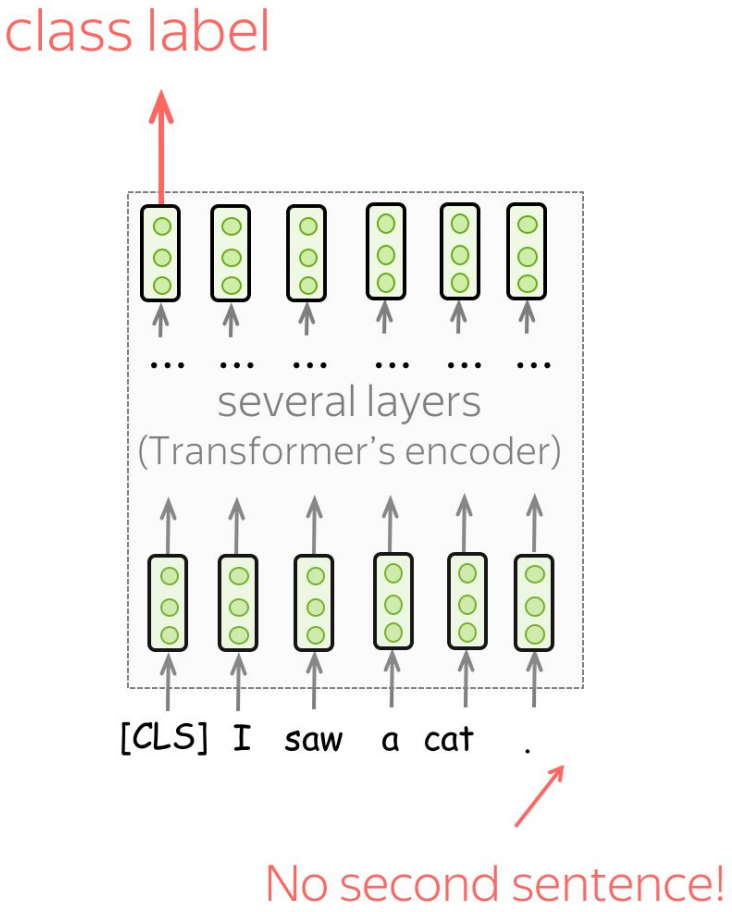
- Target: current token (the true one)
- Prediction: $P(* \mid \texttt{I [MASK] a cat})$



sees the whole text, but something is corrupted

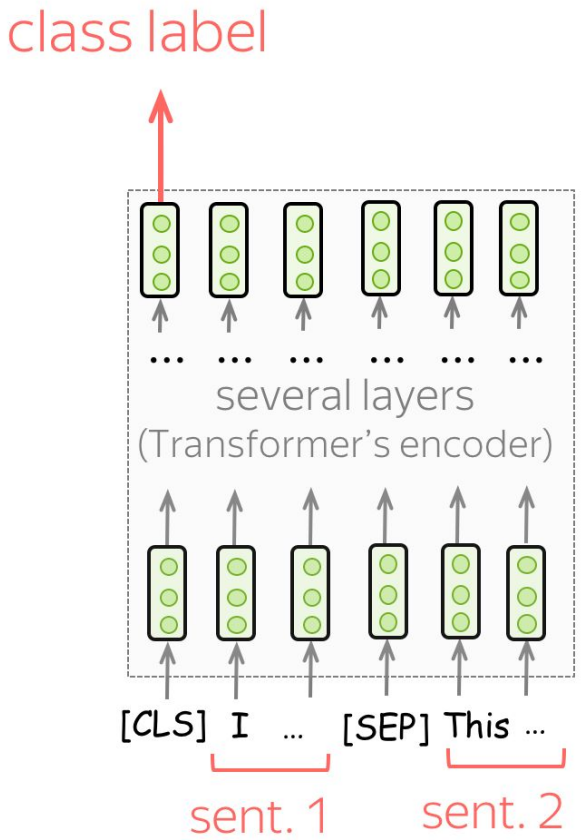
Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

Single Sentence Classification



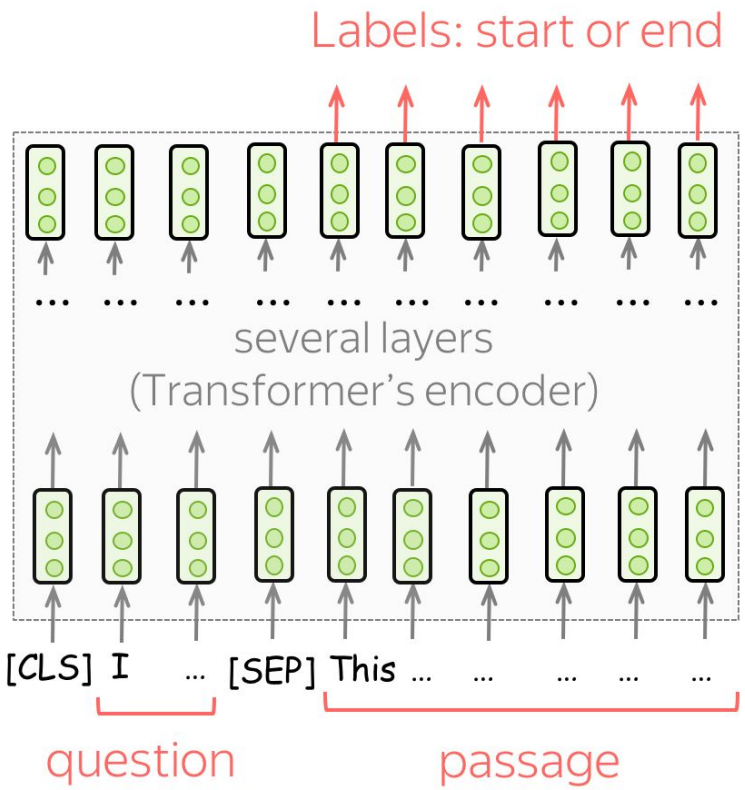
Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

Sentence Pair Classification



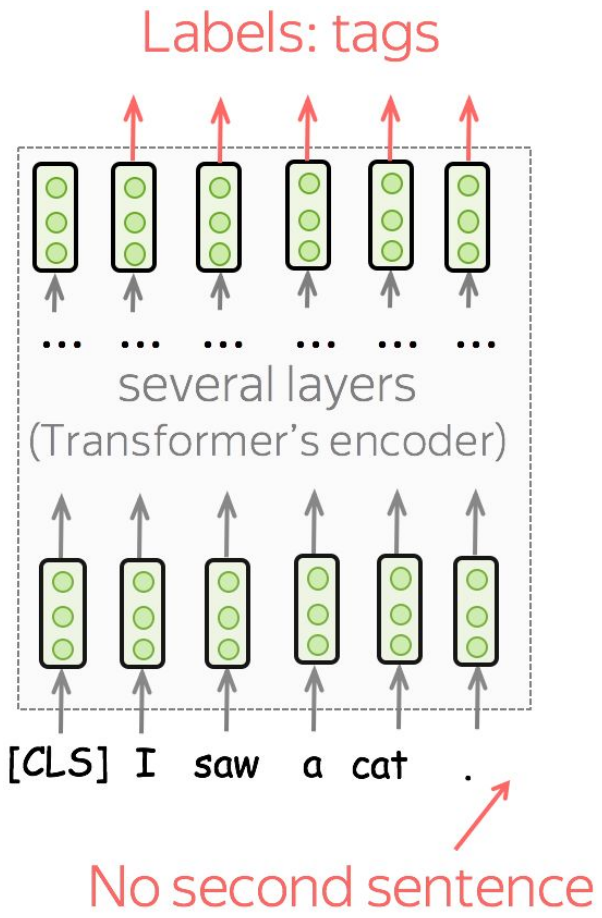
Source: https://lena-voita.github.io/nlp_course/transfer_learning.html

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

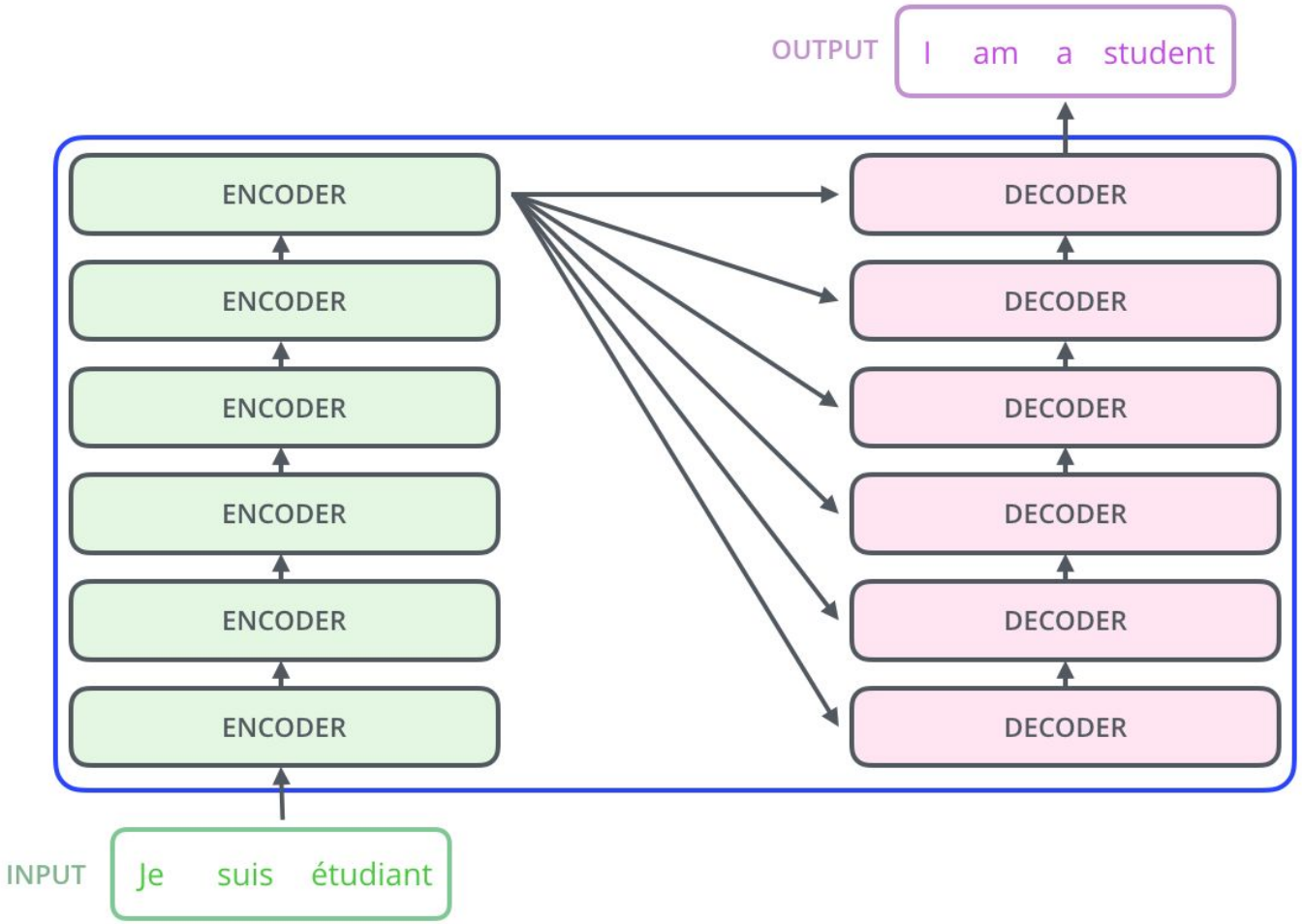
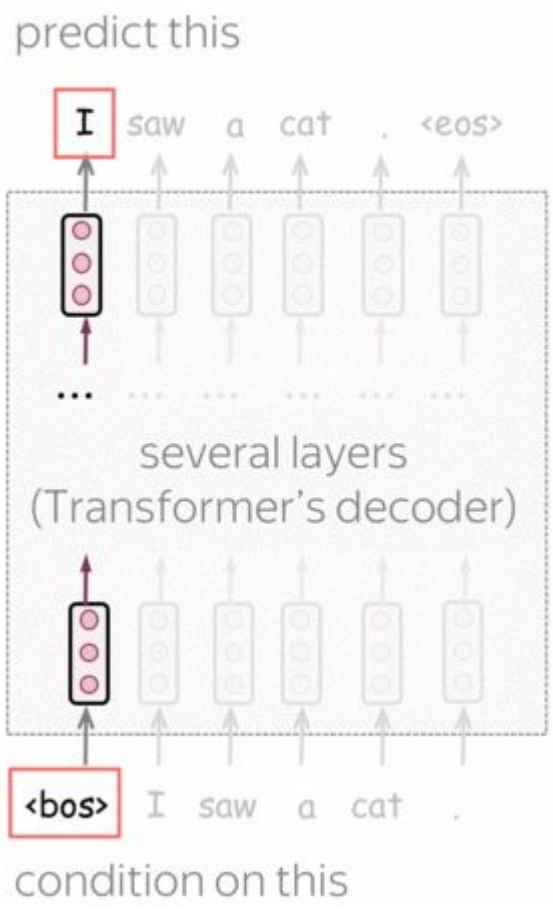


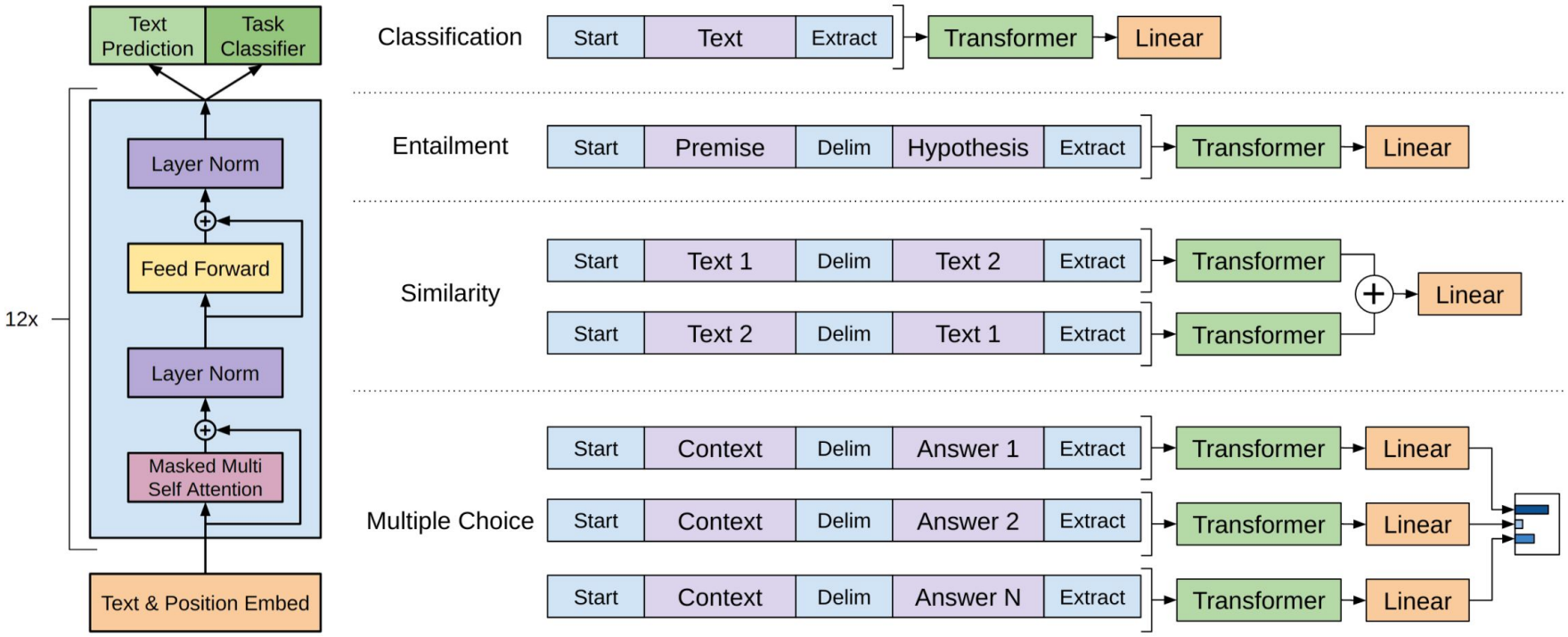
Image source: https://jalammar.github.io/illustrated-transformer/

Decoder as a universal model



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

Decoder as a universal model



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


HSE UNIVERSITY
EST 1992