21. Transformers & LLMs

EEC454 Introduction to
Machine Learning Systems

2023 Fall, Jaeho Lee

For two classes...

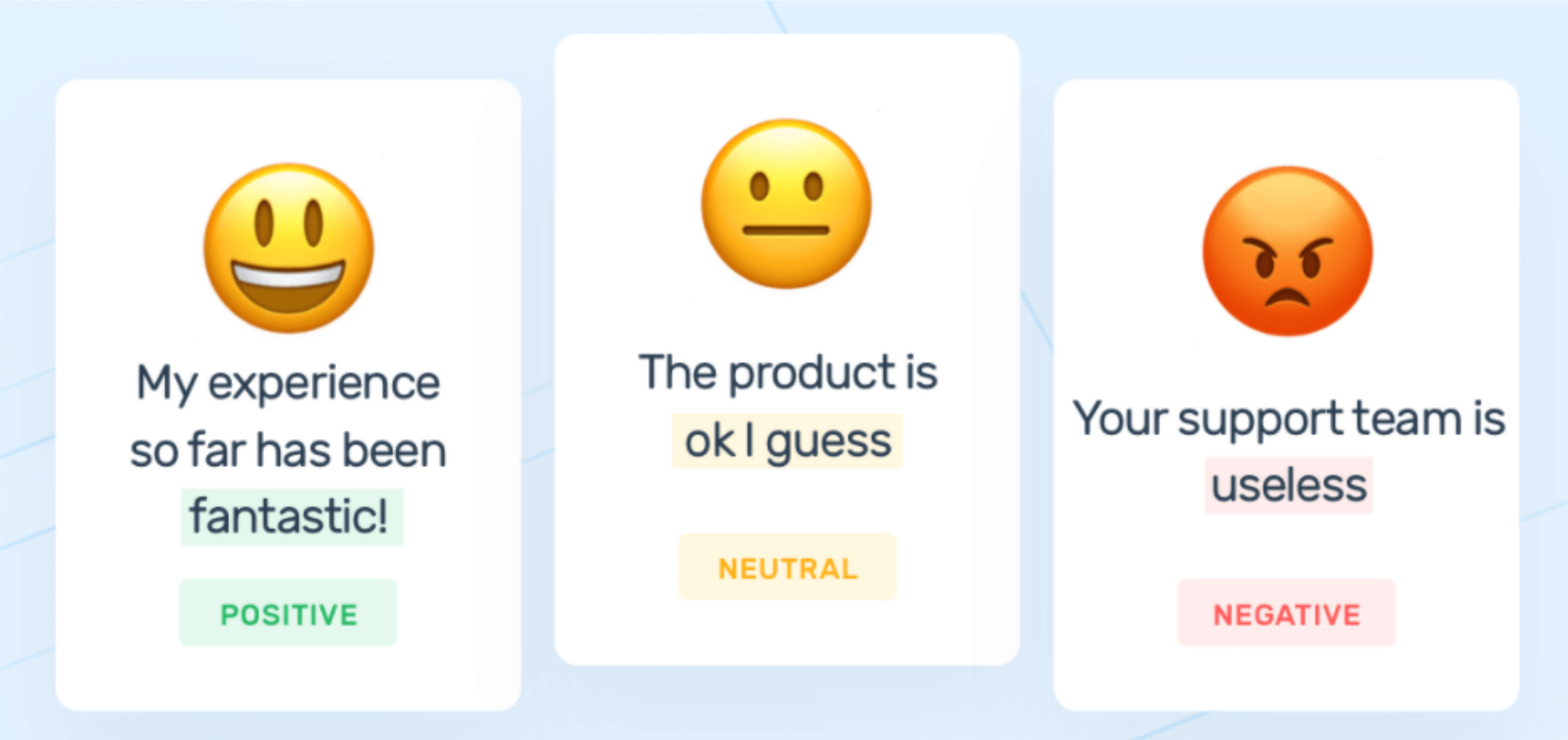
- We are going to learn basic stuffs about ***modern* *** ***language modeling***
  - **Architectures.** Transformers
  - **Training.** Autoregressive training

How can I help you today?

* There are many classic gems, that take strikingly different structures

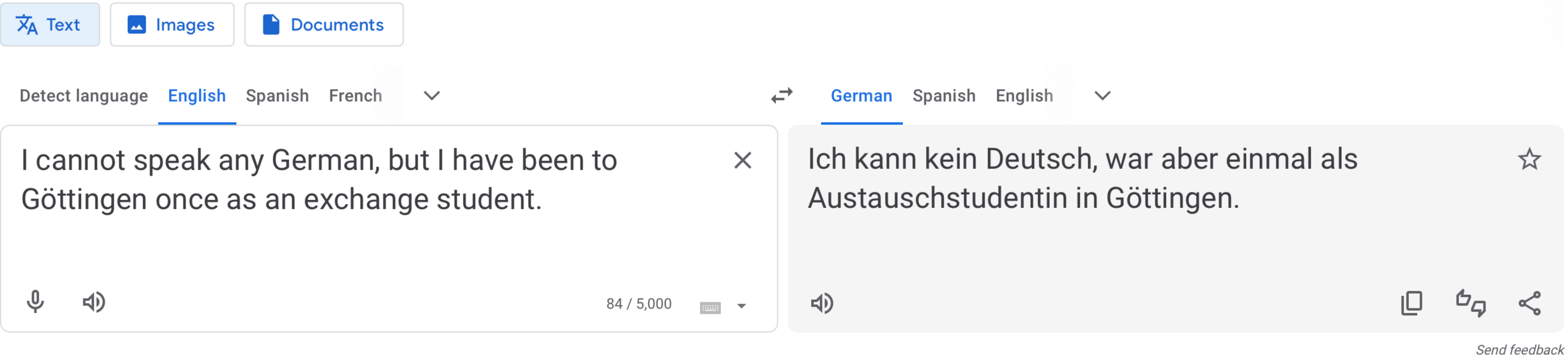
What can language models do?

- **Sentiment Classification**
  - From a customer review, infer the sentiment of the customer.



What can language models do?

- **Machine Translation**
  - Text sentences are translated into another language.



What can language models do?

- **Chatbot**
  - Generate human-like responses to given text prompts

JA **You**

What is the best way to explain how you work in two classes, to undergraduate students?

ChatGPT

Explaining how I work to undergraduate students in two classes can be broken down into a concise and clear explanation. Here's a simplified way to do it:

Class 1: Introduction to Artificial Intelligence

1. Start with the Basics:

- Begin by explaining that I am an AI language model, which means I'm a computer program designed to understand and generate human-like text.

What can language models do?

- **Retrieval-Augmented Generation**
  - Use external tools to provide a more trustworthy and up-to-date response.

JA **You**

Can you look up the Wikipedia and find me the entry for the country: Republic of Korea?

ChatGPT

Visiting en.wikipedia.org

What can language models do?

- **Text-Prompted Image Generation**
  - Generate an image that corresponds to the given query

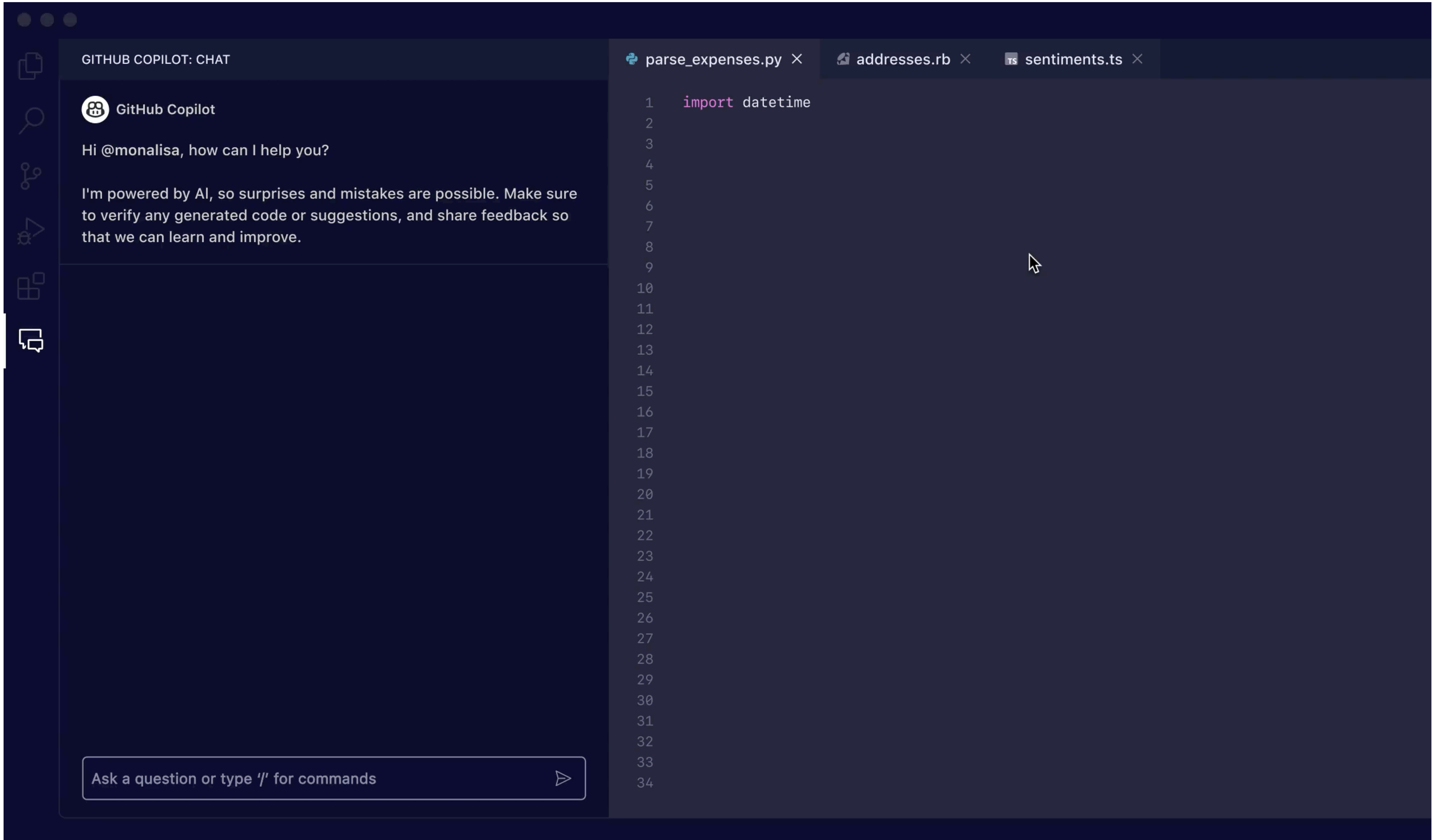
JA **You**

Can you draw me a picture of a bowl of a ramen noodle?

ChatGPT

Here is the illustration of a bowl of ramen noodles that you requested.

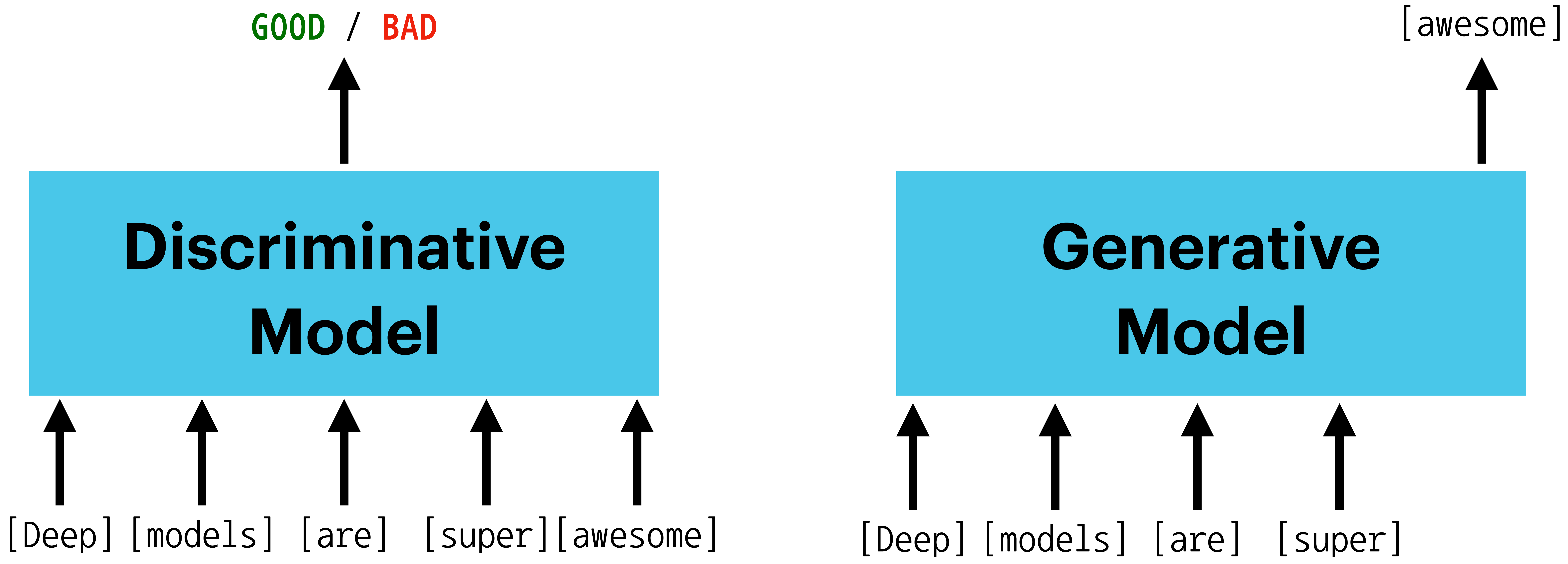
GitHub Copilot



Transformer Basics

Natural Language Processing

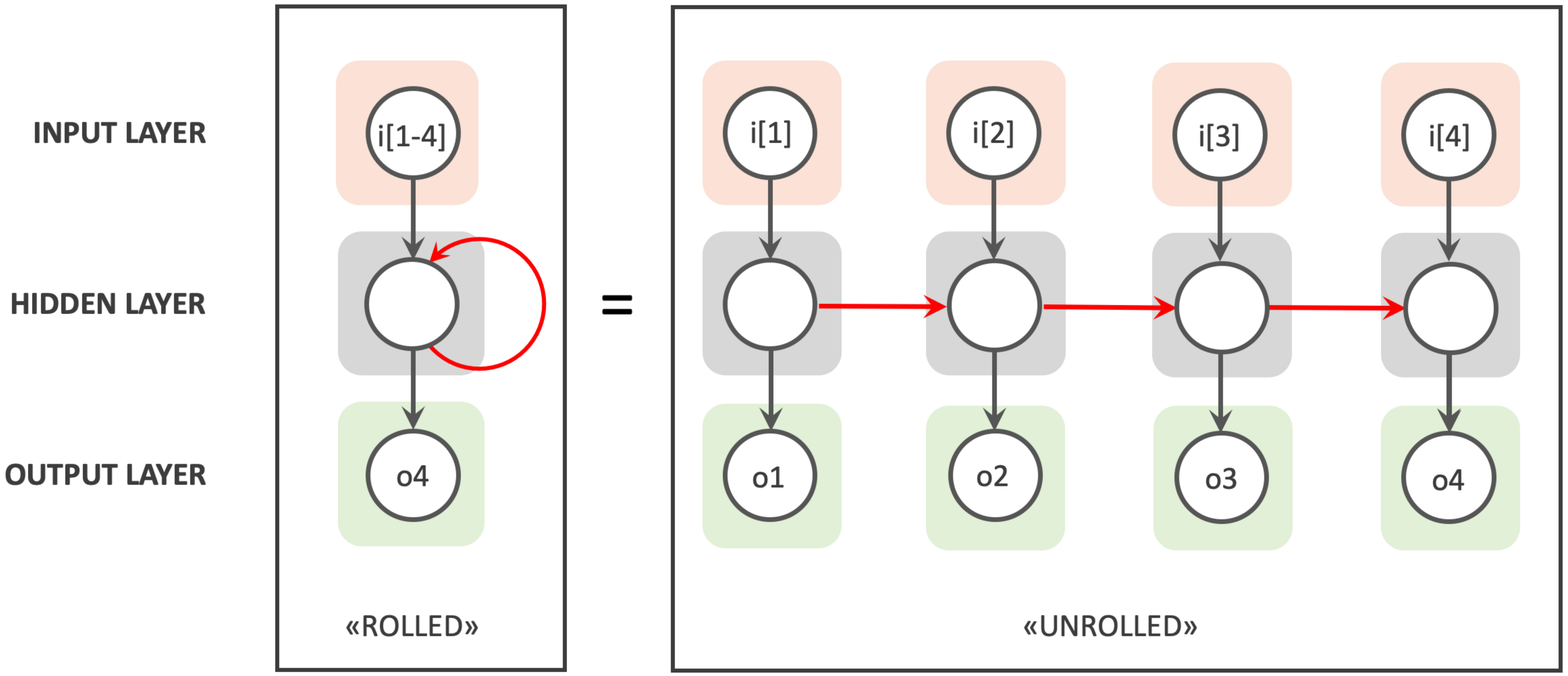
- **Discriminative.** Given a sequence of words, predict the output.
- **Generative.** Given a sequence of words, predict the next word.

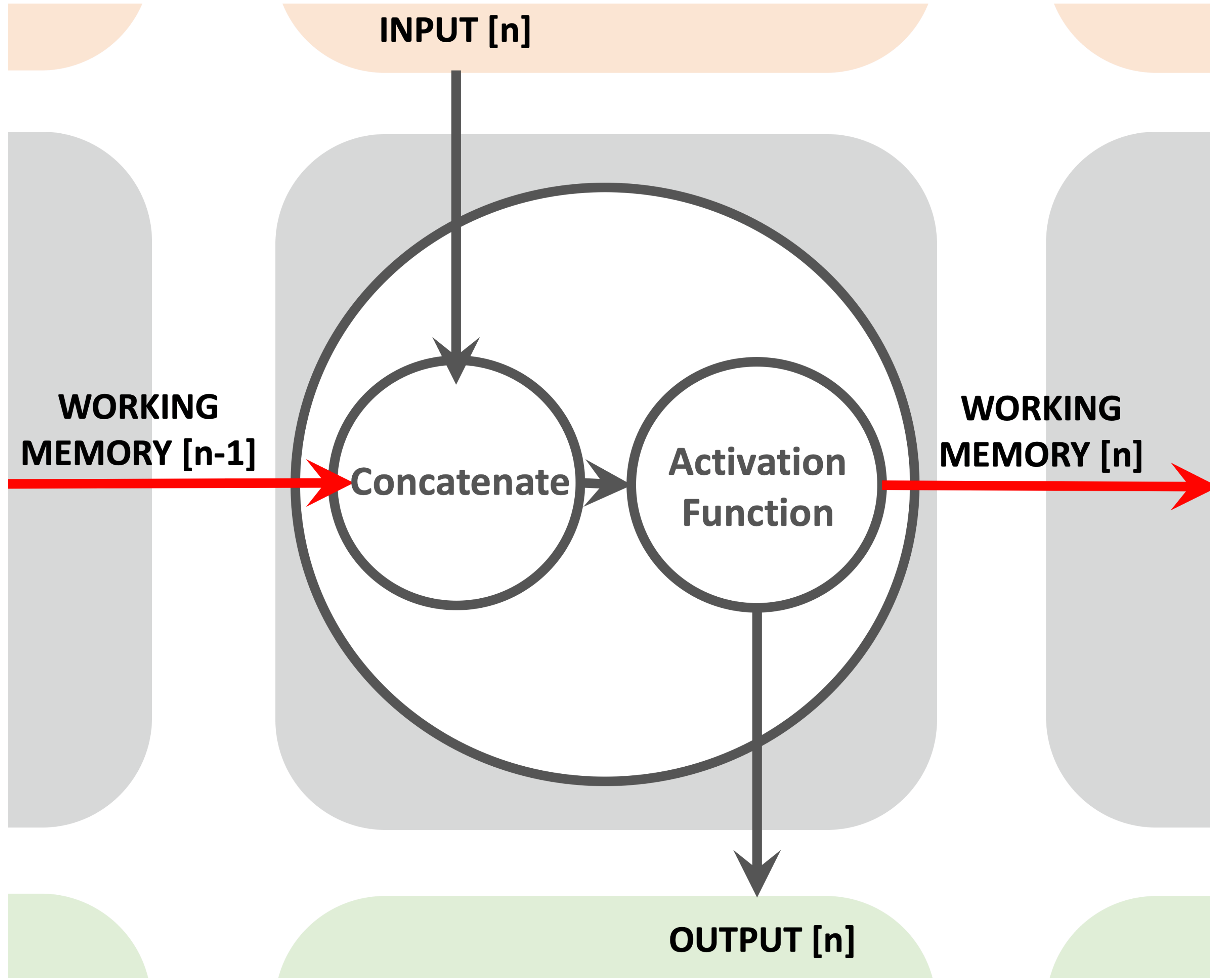


https://www.bouvet.no/bouvet-deler/explaining-recurrent-neural-networks

Past: Recurrent Neural Networks

- **Input.** The ***current input*** and ***past state***.
- **Output.** The ***current output*** and the ***current state***.



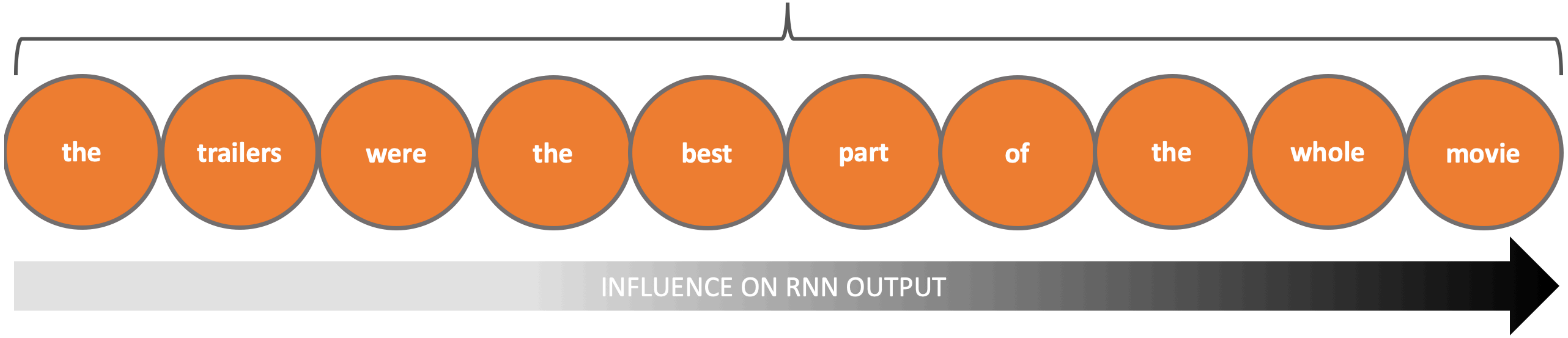


https://www.bouvet.no/bouvet-deler/explaining-recurrent-neural-networks

Past: Recurrent Neural Networks

- **Limitations.**
  - Struggles to capture long-term dependencies.
    - Vanishing / Exploding Gradient
      (LSTMs have explicit modules for "long-term memory")
  - Difficult to scale up—sequential computation is forced.

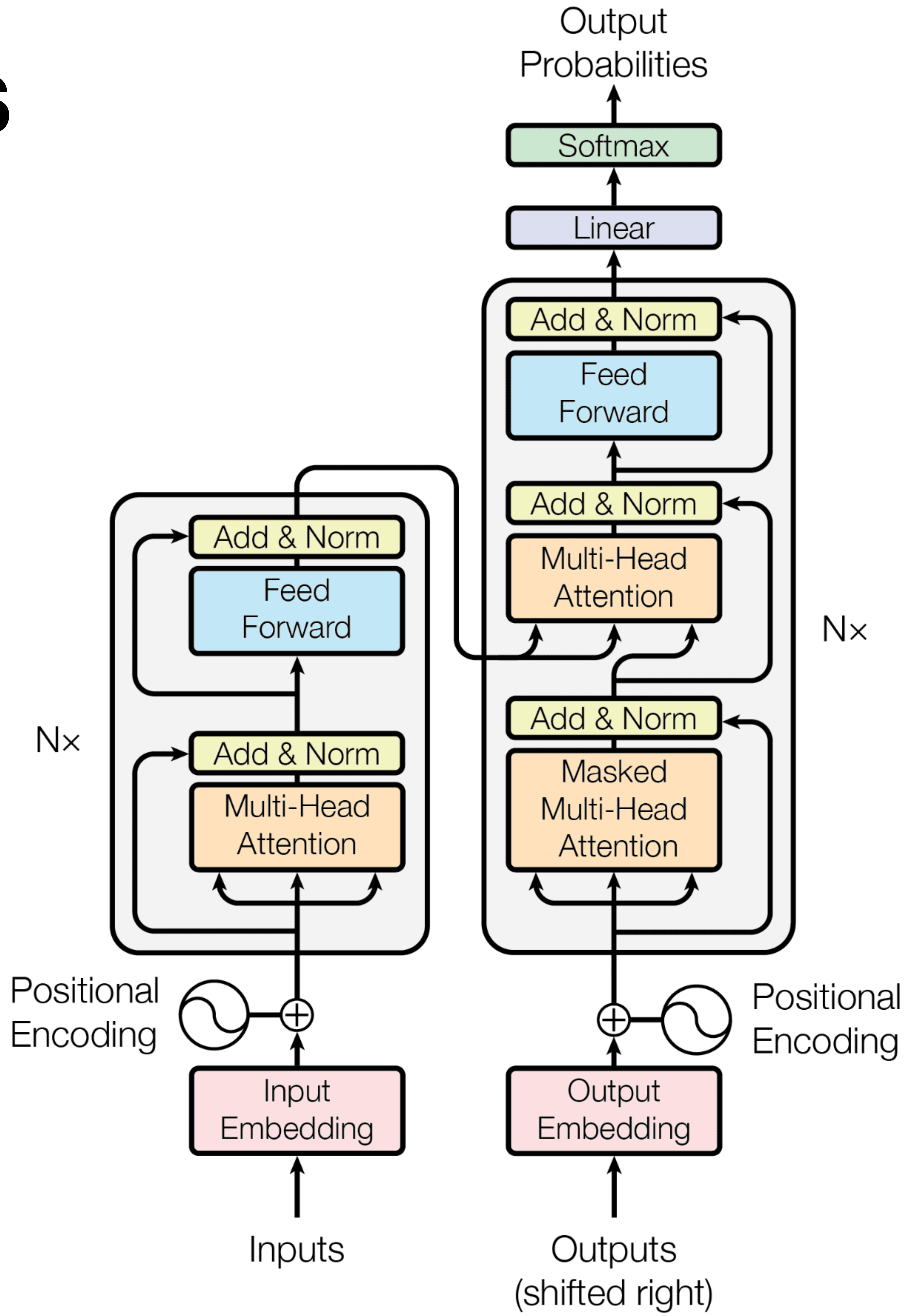
"the trailers were the best part of the whole movie."



Transformers

Key concepts

- Tokenize words
- Map tokens into embeddings
- Transformer blocks
- Positional Encoding
- Linear Prediction Head



https://platform.openai.com/tokenizer

(1) Tokenization: Words —> Tokens

- Maps a word to one or more tokens.

In the fascinating world of large language models (LLMs), much attention is given to model architectures, data processing, and optimization. However, decoding strategies like beam search, which play a crucial role in text generation, are often overlooked. In this article, we will explore how LLMs generate text by delving into the mechanics of greedy search and beam search, as well as sampling techniques with top-k and nucleus sampling.

TEXT TOKEN IDS

[644, 279, 27387, 1917, 315, 3544, 4221, 4211, 320, 4178, 22365, 705, 1790, 6666, 374, 2728, 311, 1646, 78335, 11, 828, 8863, 11, 323, 26329, 13, 4452, 11, 48216, 15174, 1093, 24310, 2778, 11, 902, 1514, 264, 16996, 3560, 304, 1495, 9659, 11, 527, 3629, 45536, 13, 763, 420, 4652, 11, 584, 690, 13488, 1268, 445, 11237, 82, 7068, 1495, 555, 1624, 4504, 1139, 279, 30126, 315, 57080, 2778, 323, 24310, 2778, 11, 439, 1664, 439, 25936, 12823, 449, 1948, 12934, 323, 62607, 25936, 13]

TEXT **TOKEN IDS**

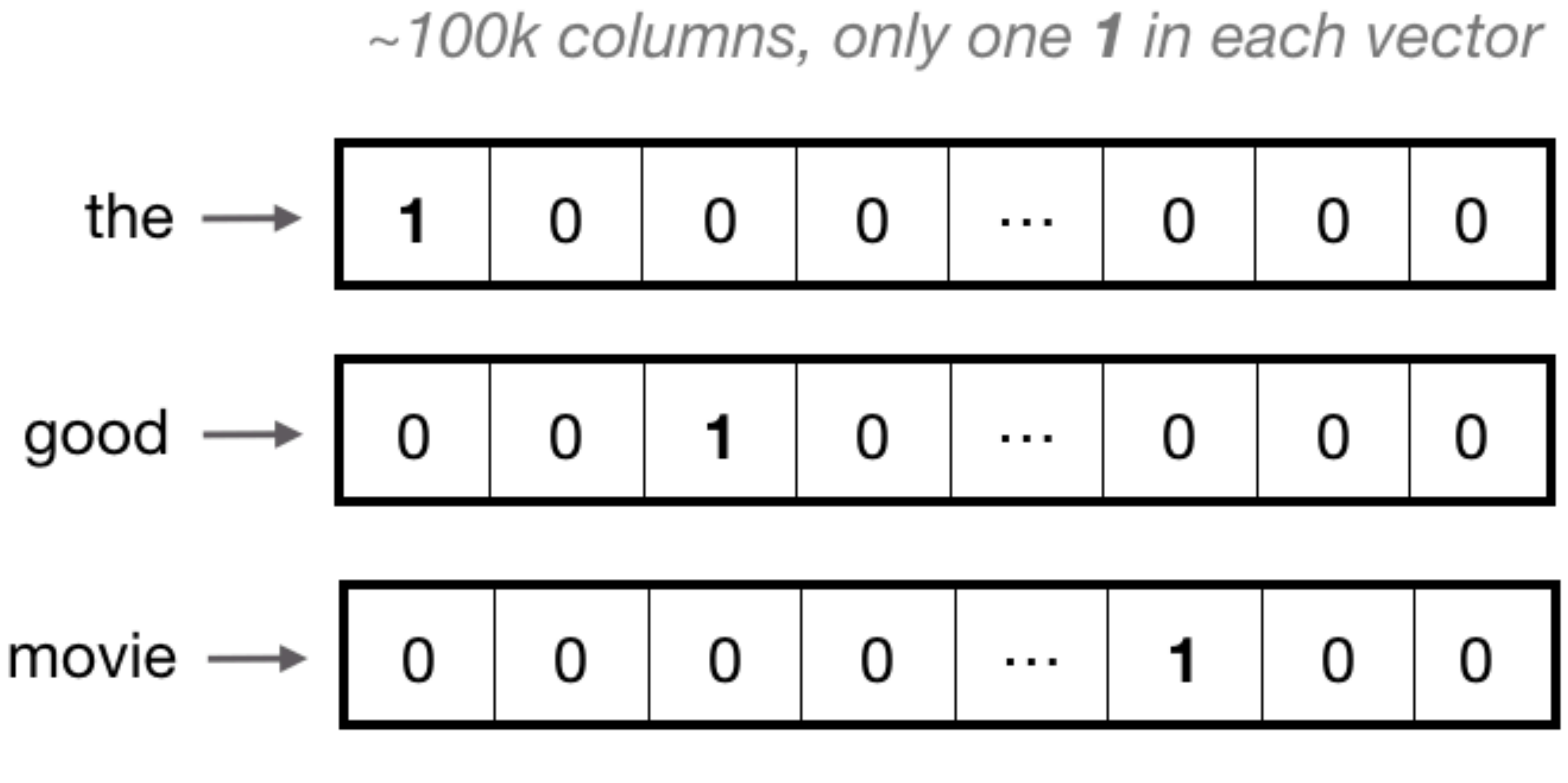
https://cezannec.github.io/CNN_Text_Classification/

(2) Embedding: Tokens —> Embeddings

Maps each token to a high-dimensional vector.

Example. One-hot encoding

- Easy to build
- Very long, if vocab size is large.
- Very sparse—dimensions wasted?
- No semantics



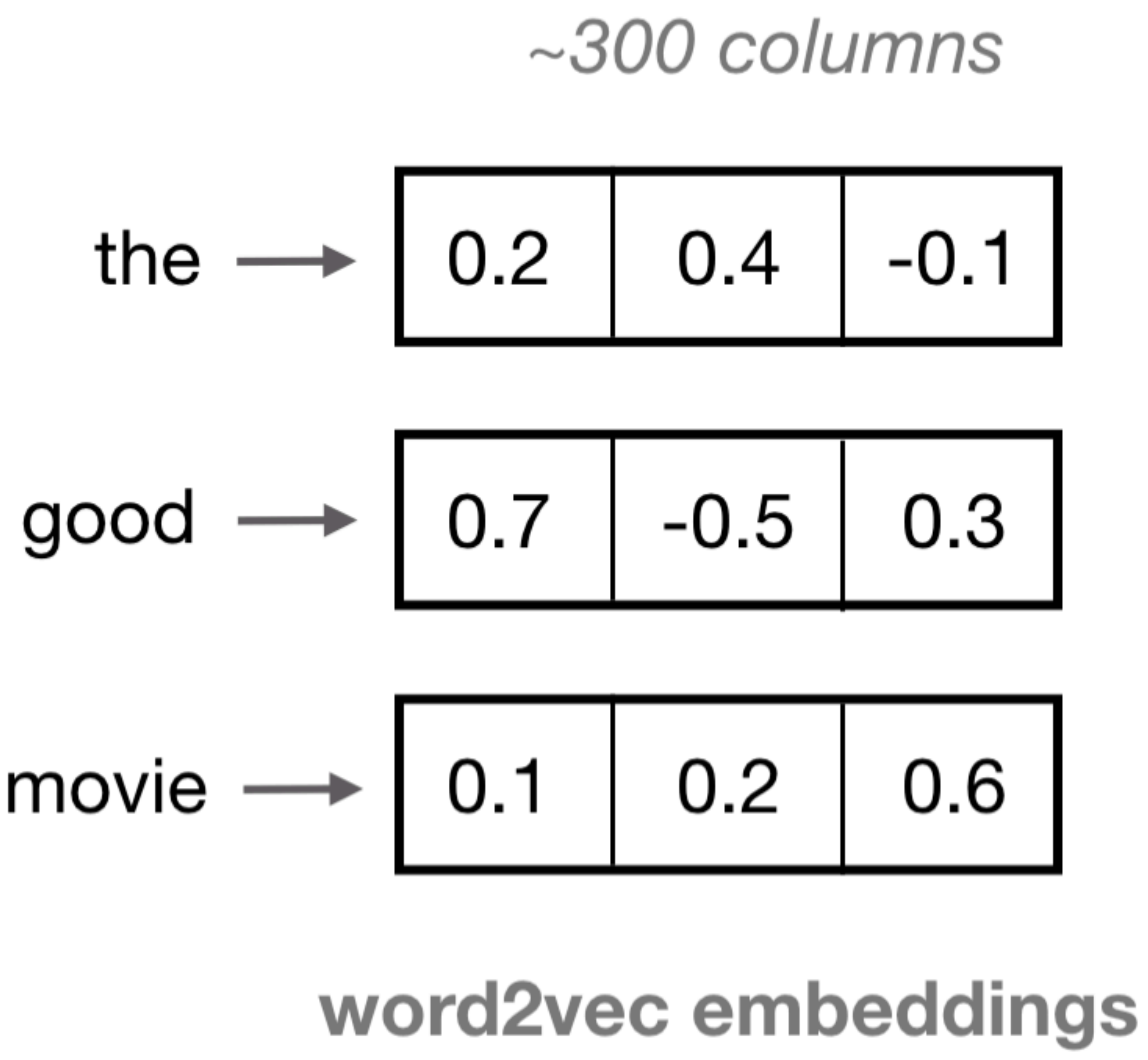
one-hot encoding

(2) Embedding: Tokens —> Embeddings

Typical Choice. Word embedding

(e.g., Word2Vec, GLoVe)

- Low-dimension
- Values take continuous values
- Learned jointly / separately
    - Rich in semantics
    - Can represent "similarity" by inner prod.



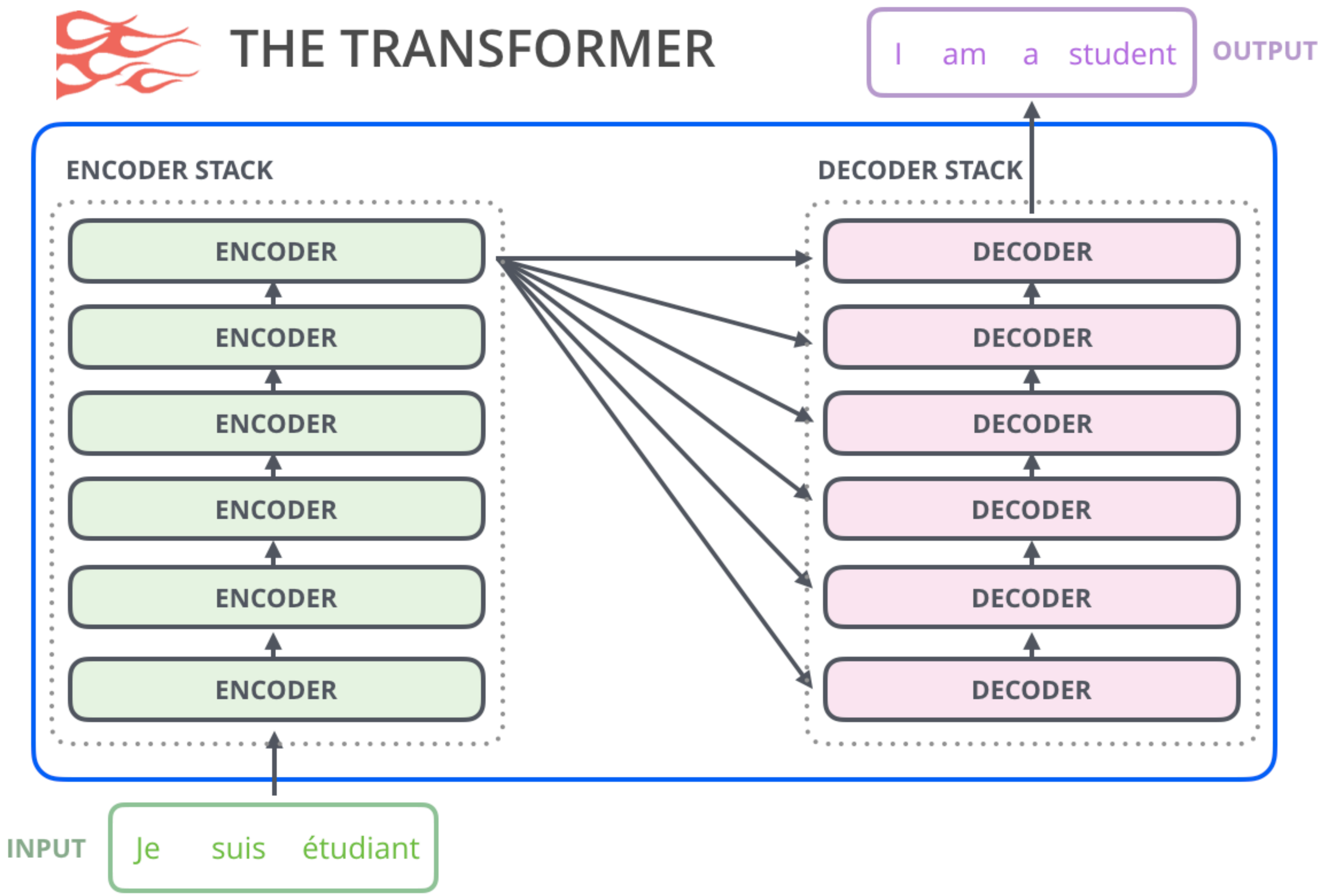
https://jalammar.github.io/illustrated-gpt2/

(3) Transformer Blocks

Transformers consists of a stack of encoders & a stack of decoders

- **Encoder-only:** BERT
- **Decoder-only:** GPT

(our focus)

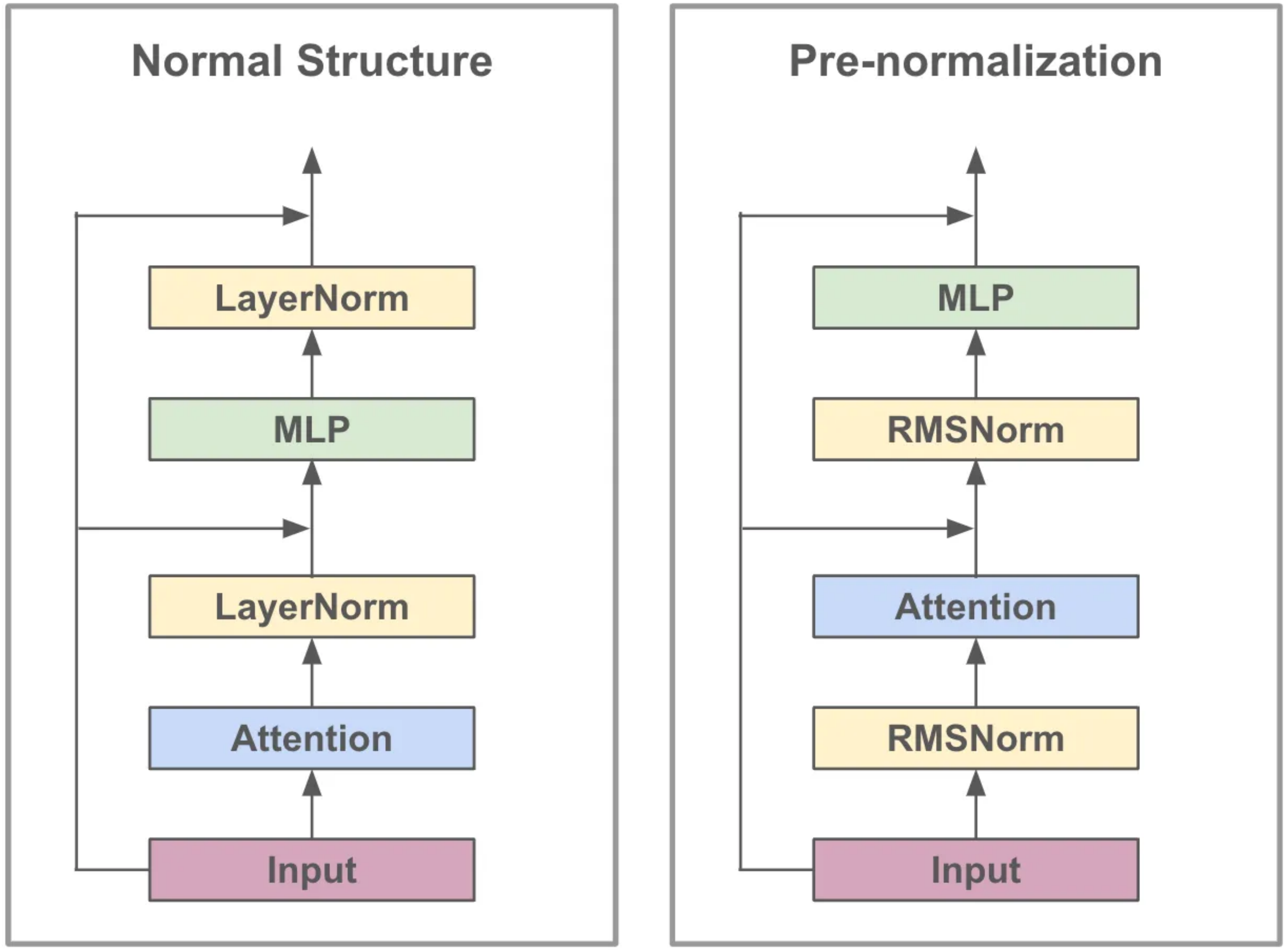


https://cameronrwolfe.substack.com/p/llama-2-from-the-ground-up

(3) Transformer Blocks

Each encoder/decoder block consists of ***four elements***

- **Multi-Head Attention** (MHA)
- **Feed-Forward Network** (FFN)
- **LayerNorm / RMSNorm**
- **Residual Connections**

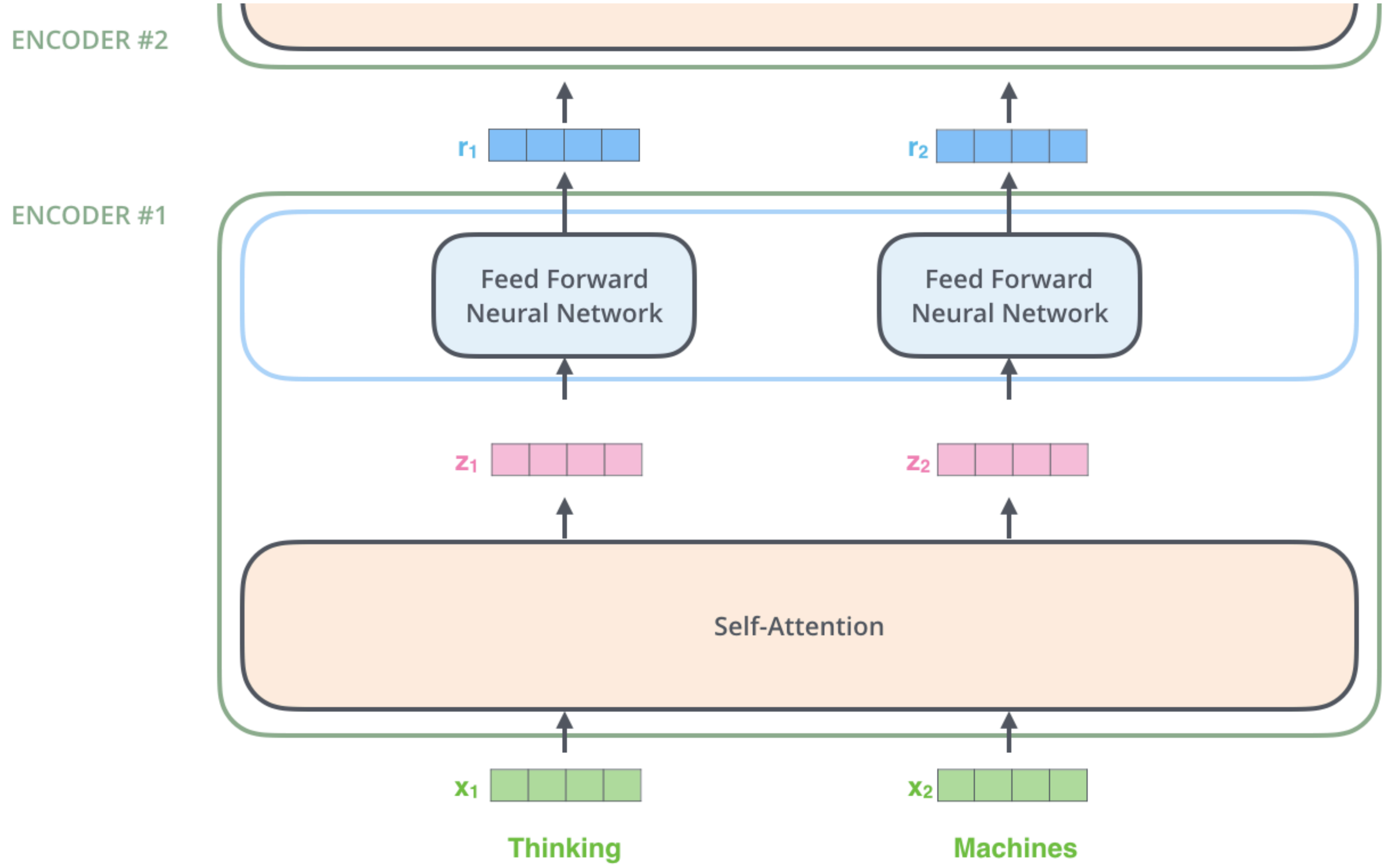


(3) Transformer Blocks

MHA. Generates a vector for each tokens.

- Quantifies the relationship between tokens

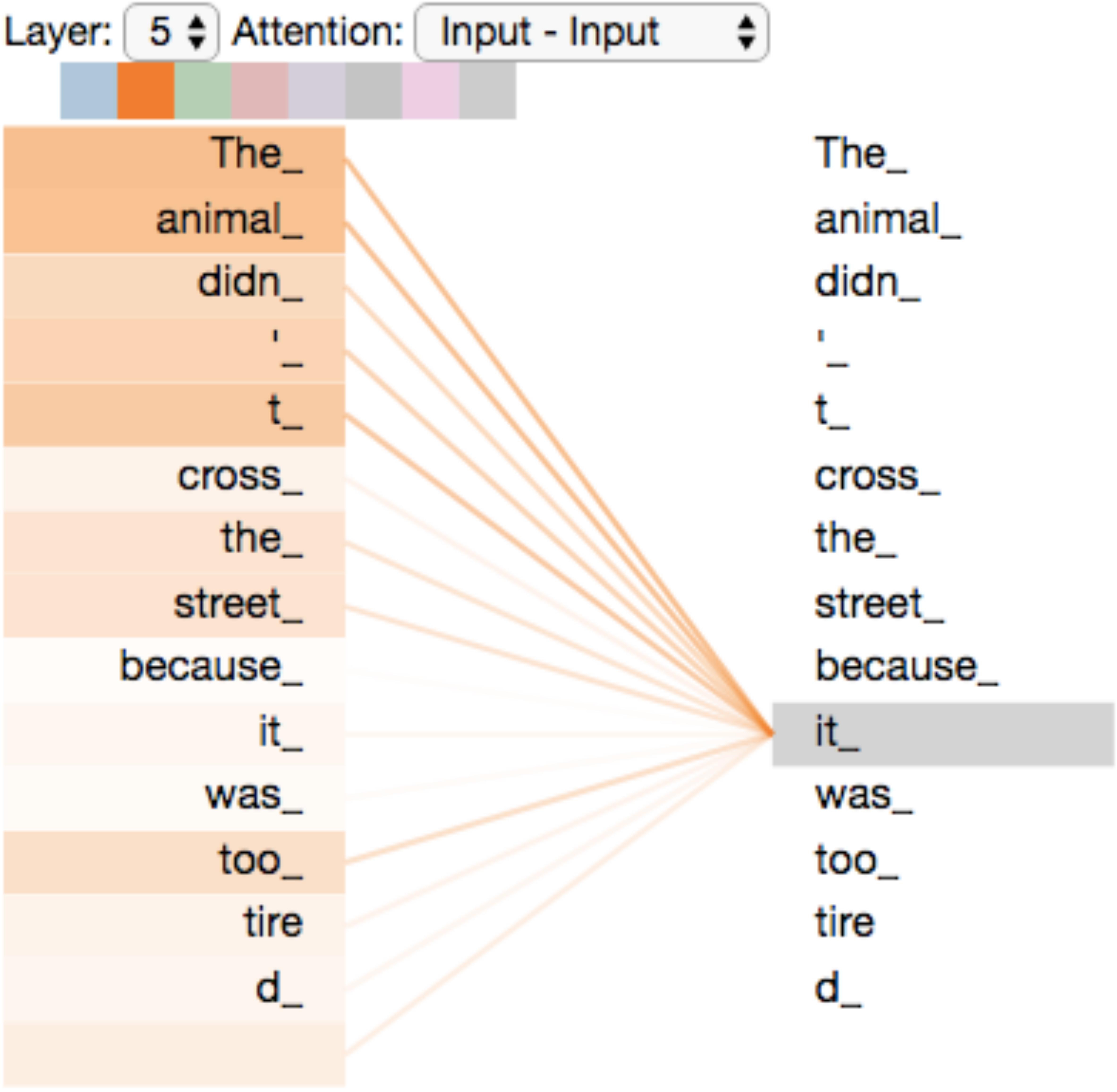
FFN. Concatenates and process each vector separately.



(3-1) Multi-Head Attention

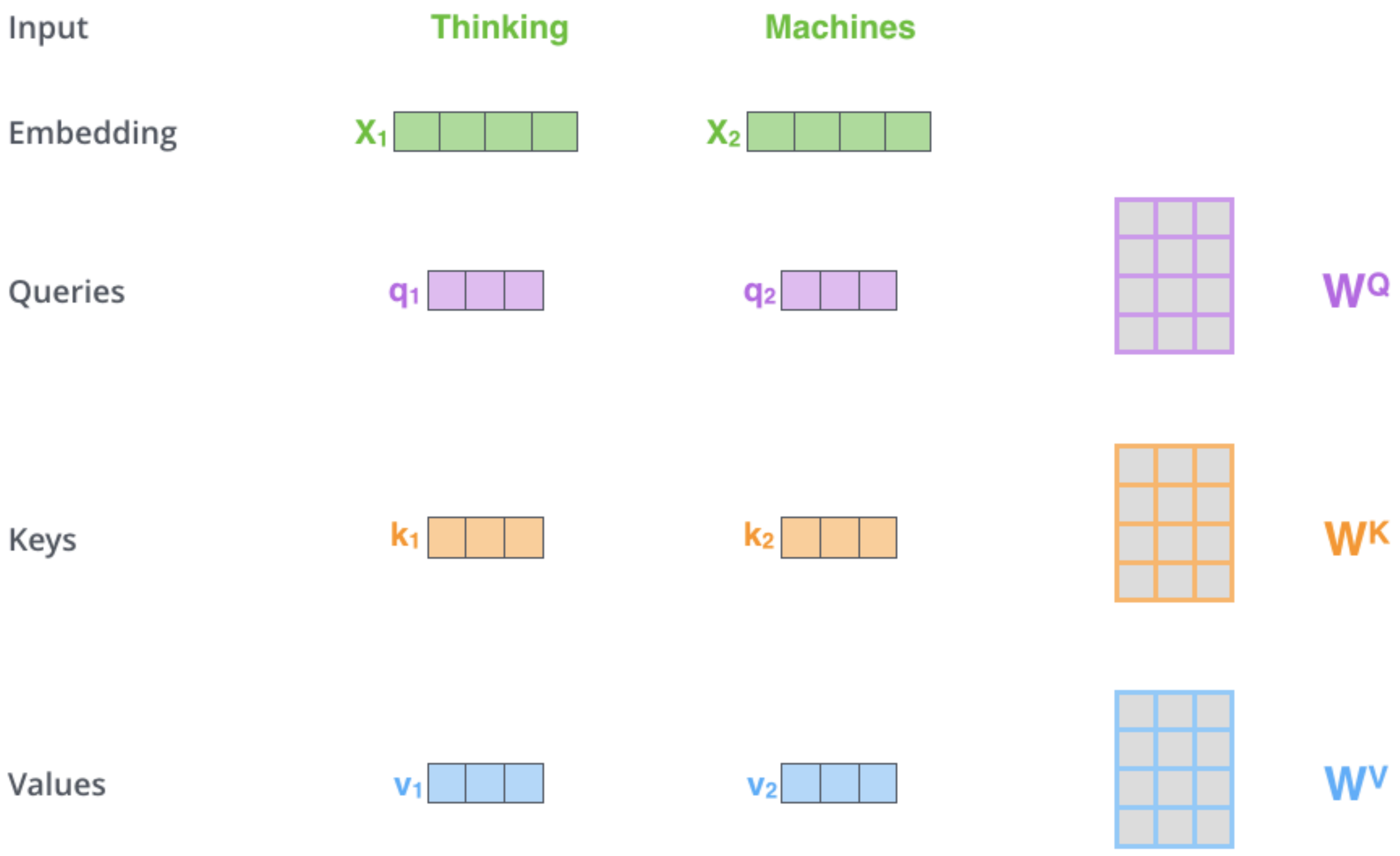
High-Level Idea

- Quantifies how much the information in a token is related to another token.
    1. Q,K,V
    2. Q,K —> Attention score
    3. Attention, V —> Output



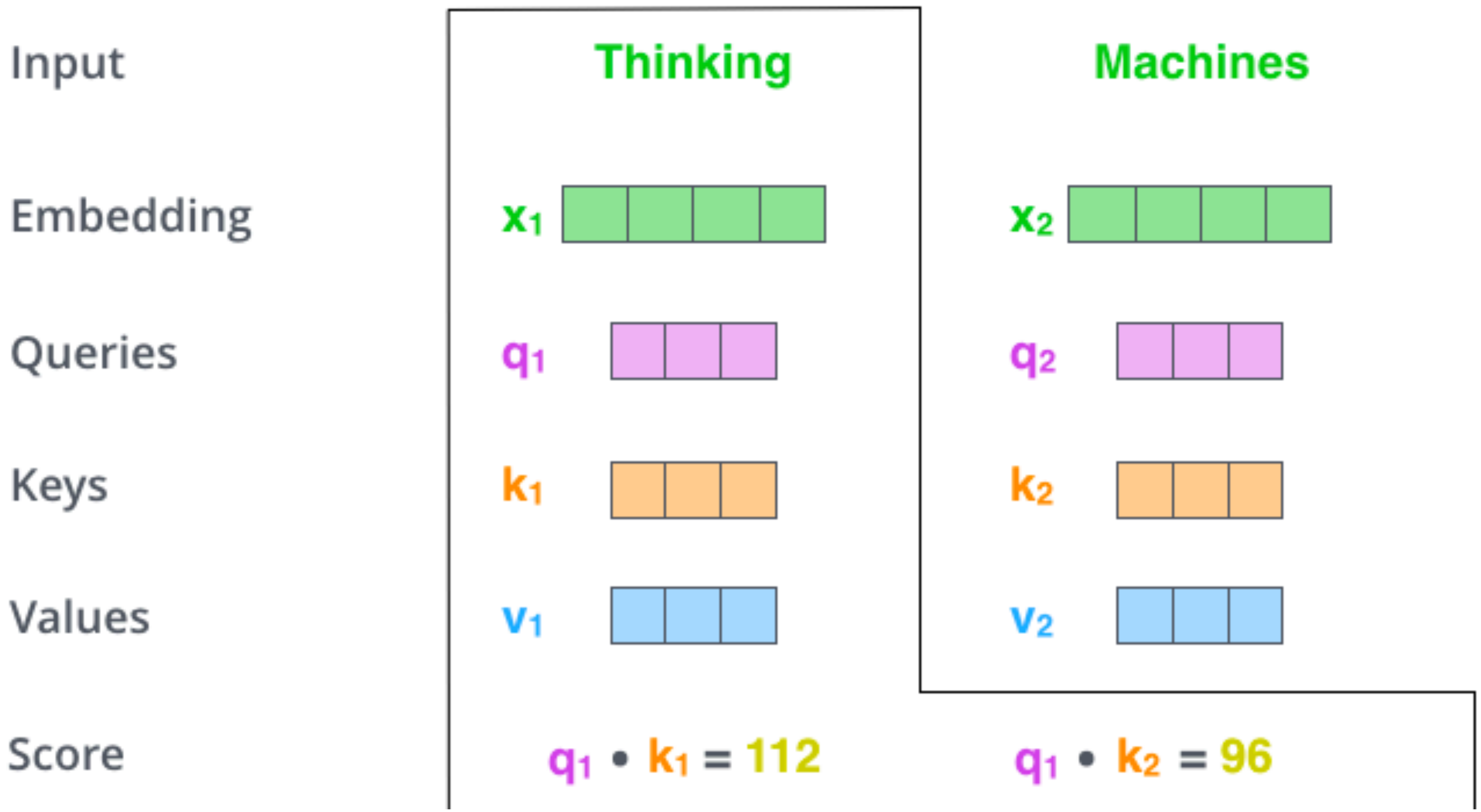
(3-1) Multi-Head Attention

- **Step 1.** For each **token**, we compute **query**, **key**, and **value**.



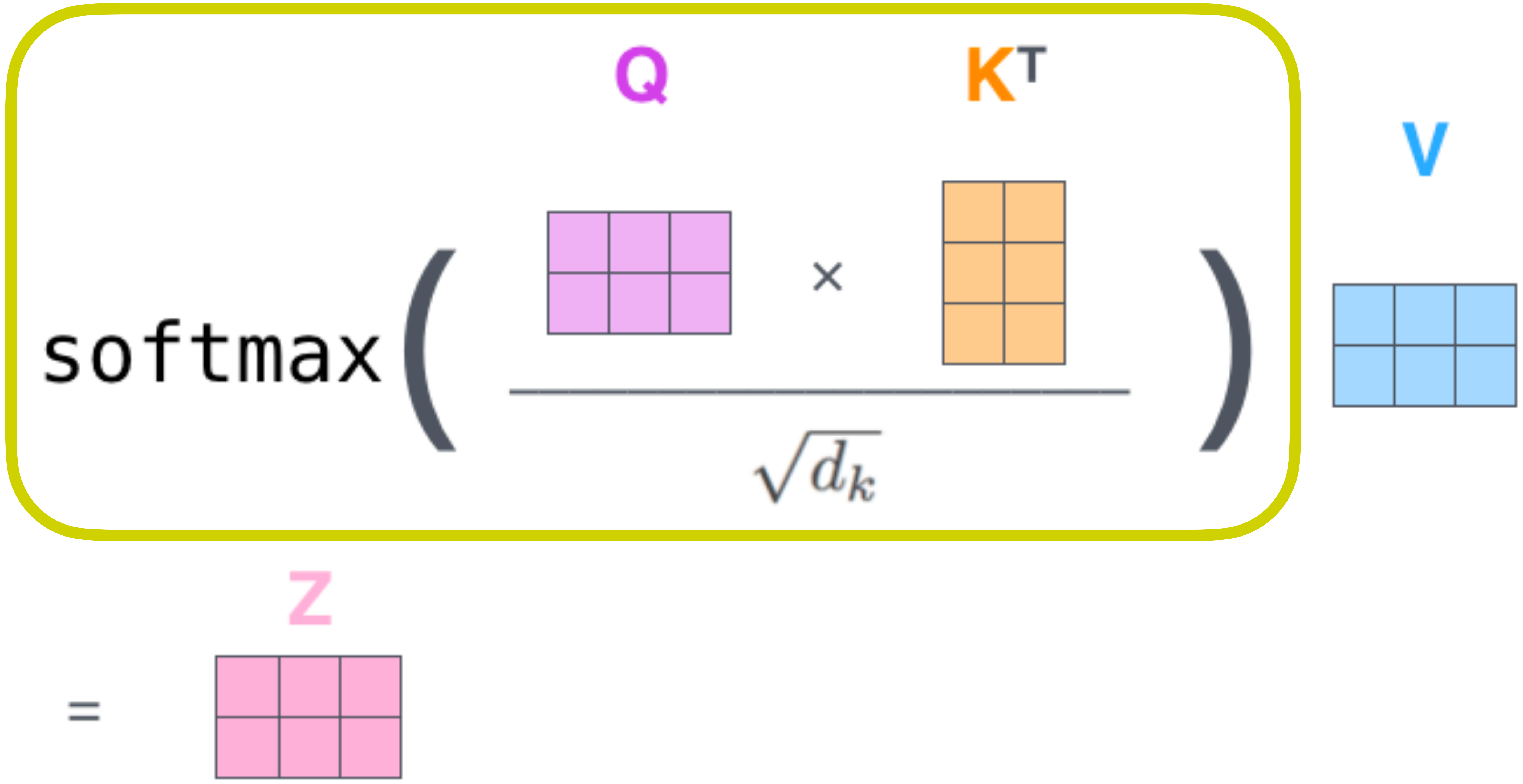
(3-1) Multi-Head Attention

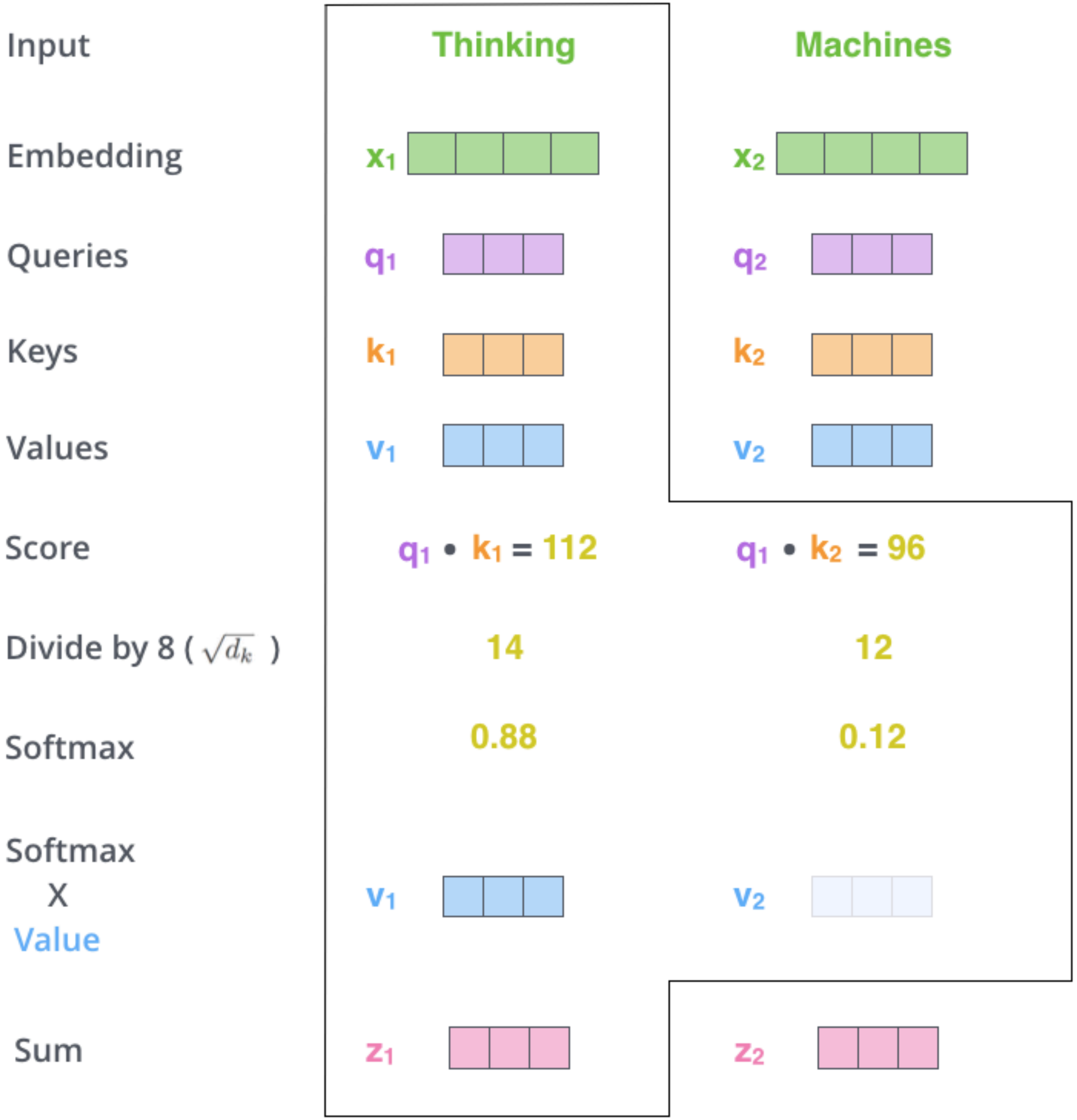
- **Step 2.** Compute attn scores from query (self) and key (self,others)



(3-1) Multi-Head Attention

- **Step 3.** Compute output as a weighted sum of **values**, weighted by the **softmax of attn scores**.





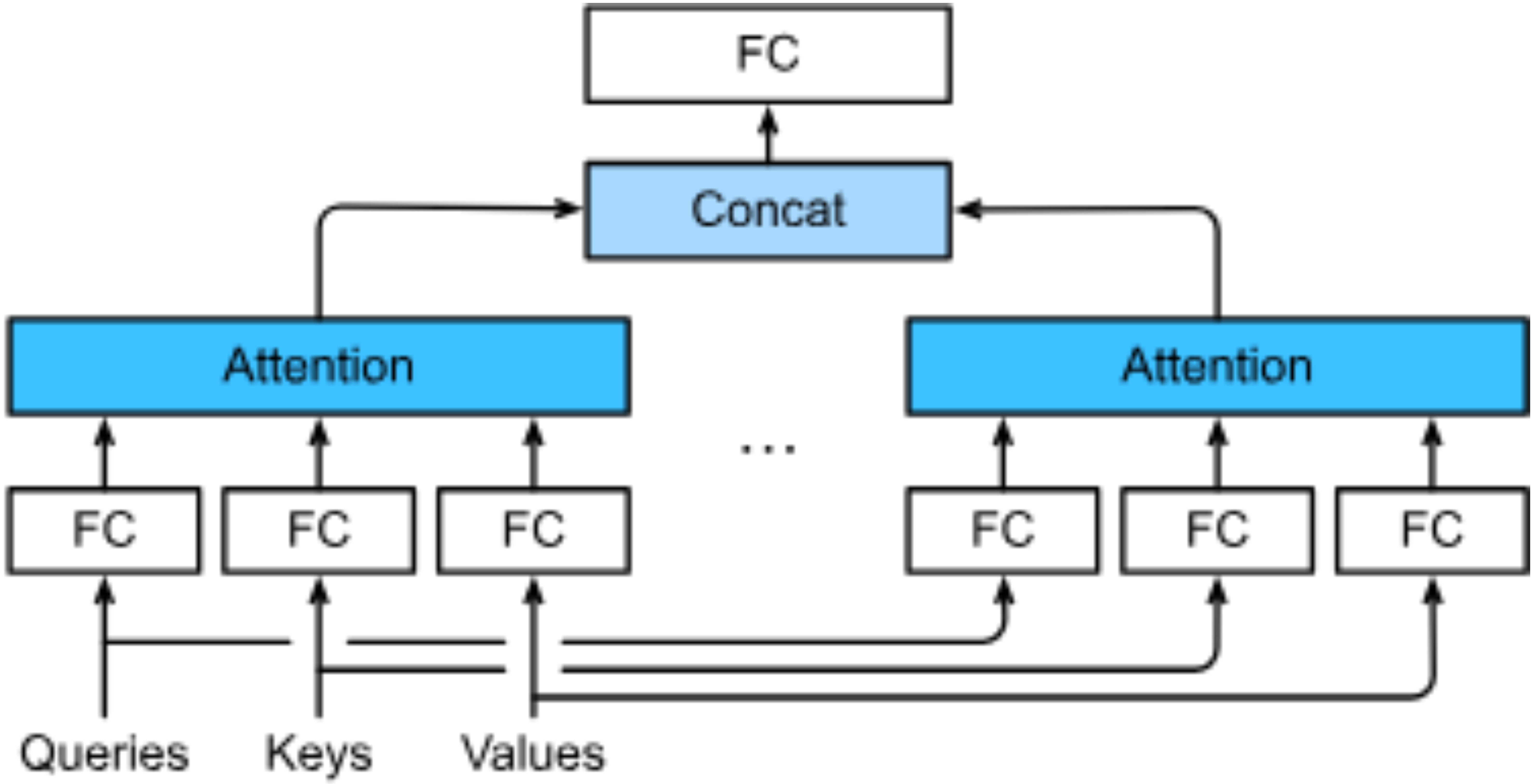
Computation & Memory

Suppose that we have n tokens.
We compute...

- Q/K/V for each tokens,
  - $O(n)$
- Attention for each Q-K pairs
  - $O(n^2)$
- Weighted sum
  - $O(n^2)$

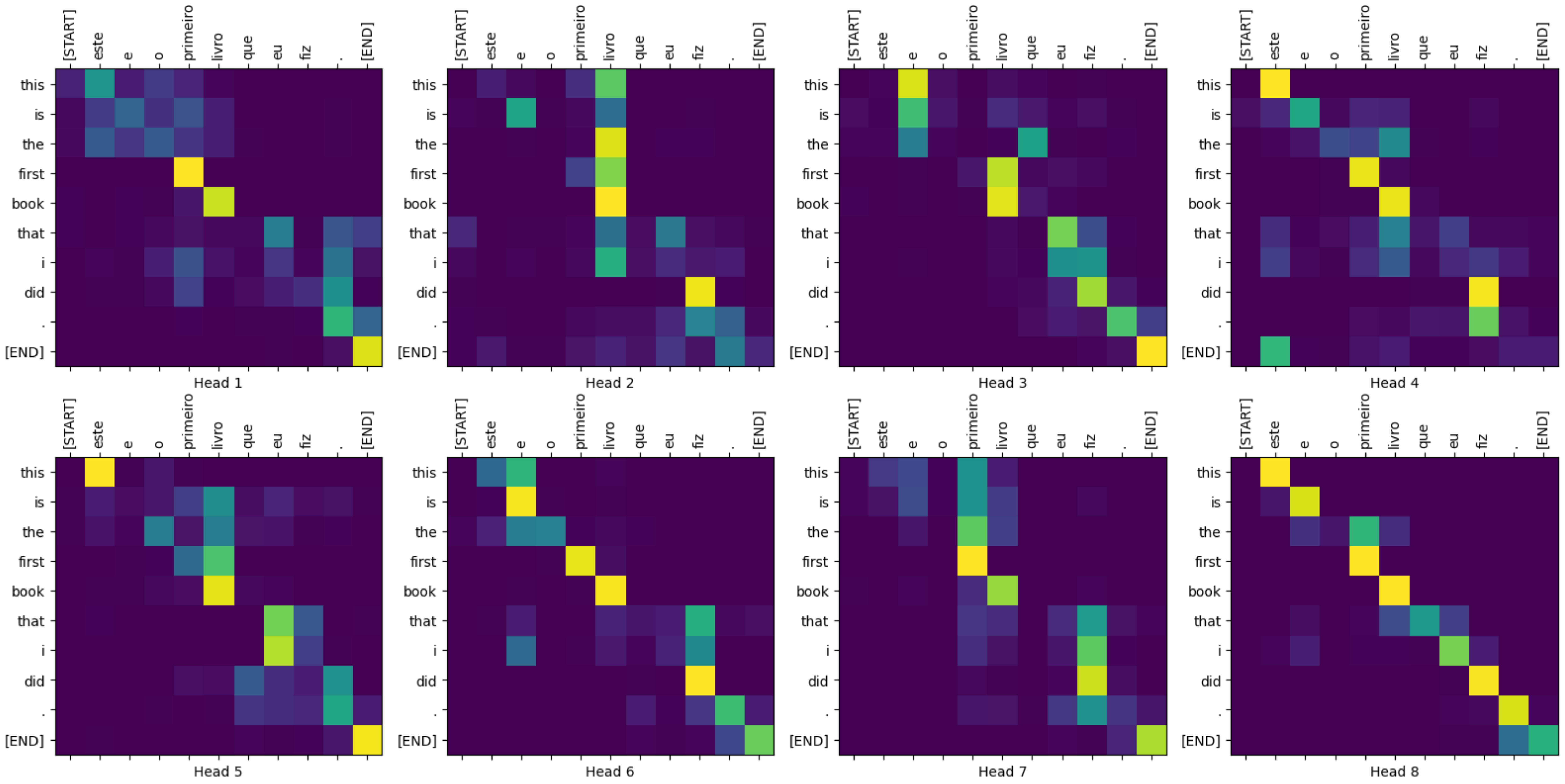
(3-1) Multi-Head Attention

- **Multi-Head.** We have multiple parallel attention layers.
  —> Concatenate the outputs, and do linear projection.



(3-1) Multi-Head Attention

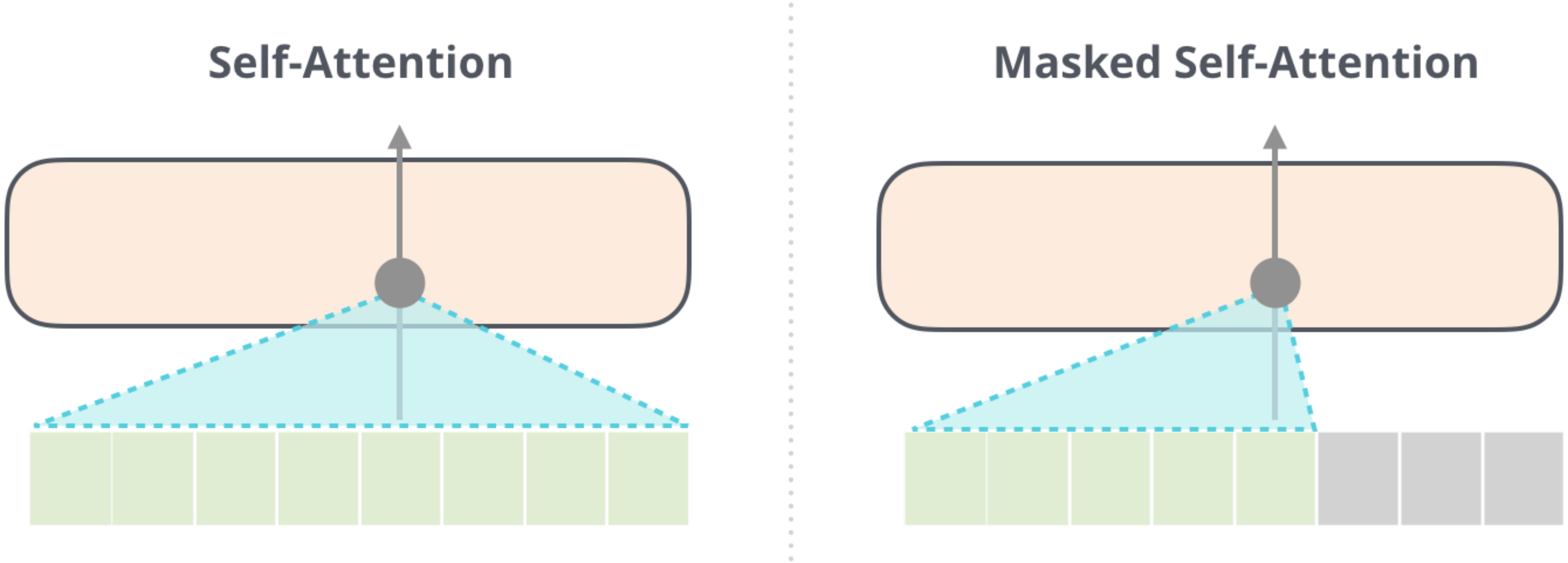
- Heads can capture diverse attention patterns.



https://jalammar.github.io/illustrated-gpt2/

(3-1) Multi-Head Attention

- In decoders, the self-attention layers is ***masked***:
  - Can only see previous inputs to generate current output.

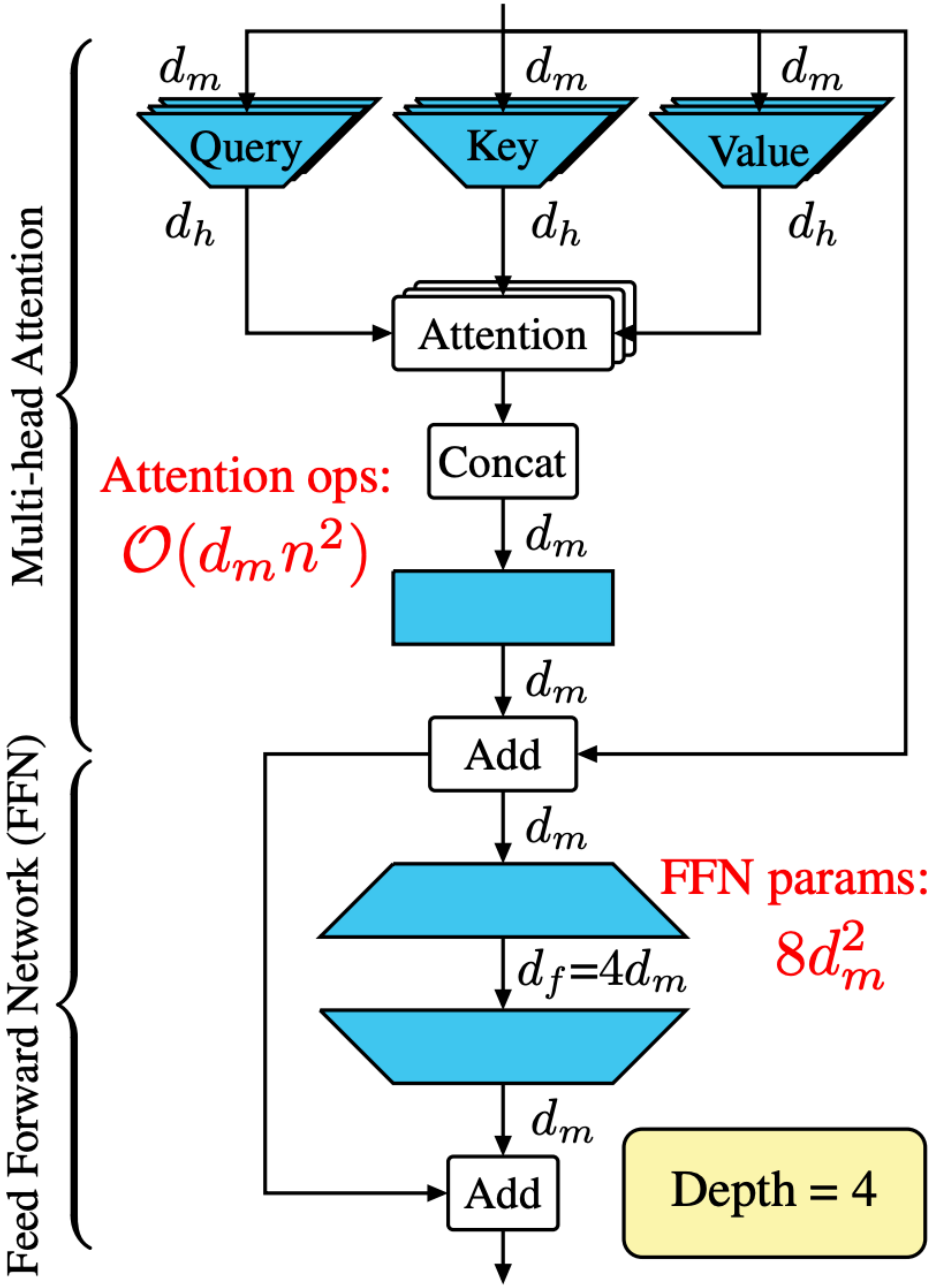


Mehta et al., “DeLight: Deep and lightweight transformer,” ICLR 2021

(3-2) Feed-Forward Network

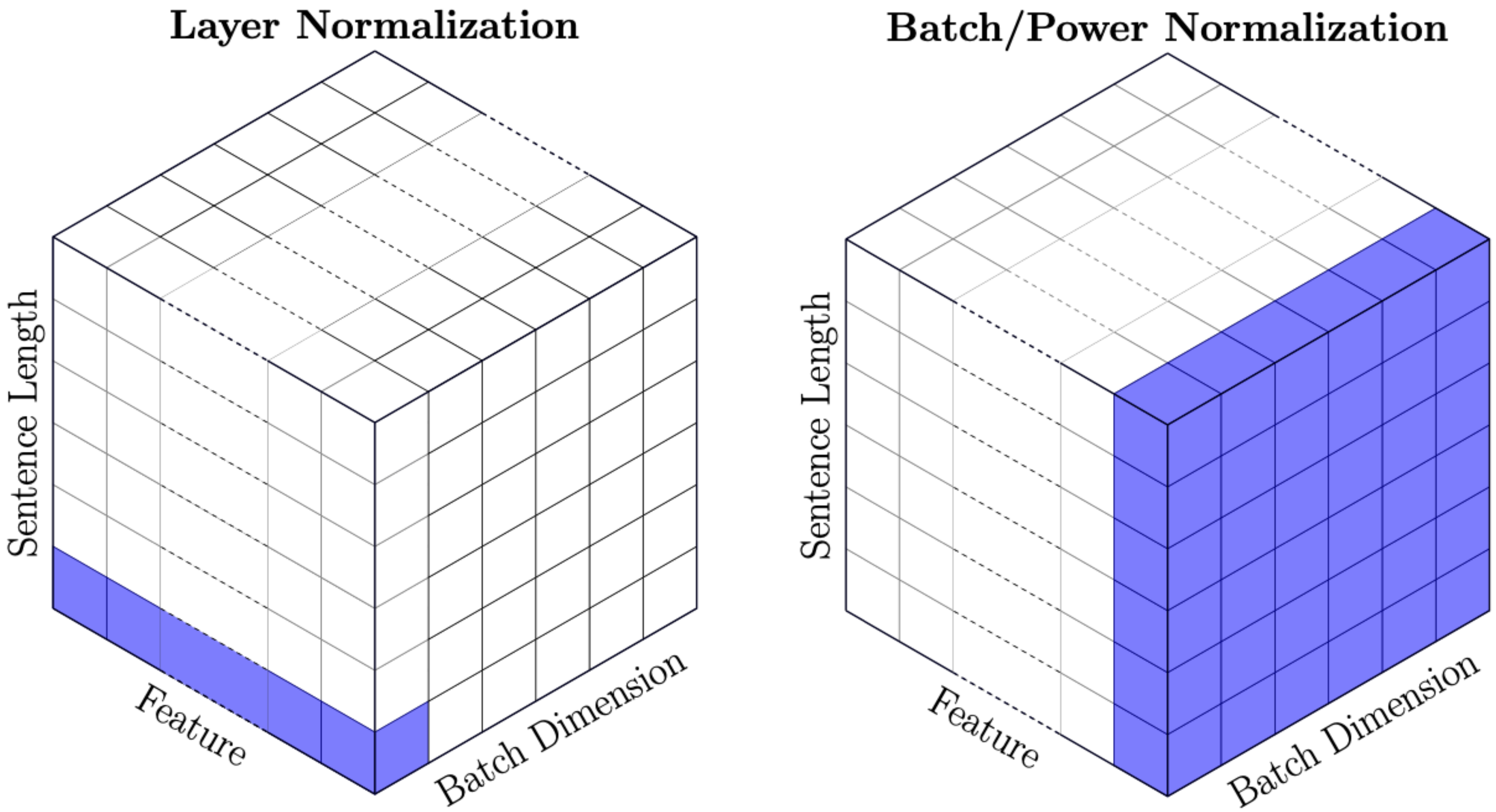
- Fully-connected layers that follow the MHA.
  - Inverted bottleneck.
  - Compute-Heavy

1	description	FLOPs / update	% FLOPS MHA	% FLOPS FFN	% FLOPS attn	% FLOPS logit
8	OPT setups					
9	760M	4.3E+15	35%	44%	14.8%	5.8%
10	1.3B	1.3E+16	32%	51%	12.7%	5.0%
11	2.7B	2.5E+16	29%	56%	11.2%	3.3%
12	6.7B	1.1E+17	24%	65%	8.1%	2.4%
13	13B	4.1E+17	22%	69%	6.9%	1.6%
14	30B	9.0E+17	20%	74%	5.3%	1.0%
15	66B	9.5E+17	18%	77%	4.3%	0.6%
16	175B	2.4E+18	17%	80%	3.3%	0.3%



(3-3) LayerNorm

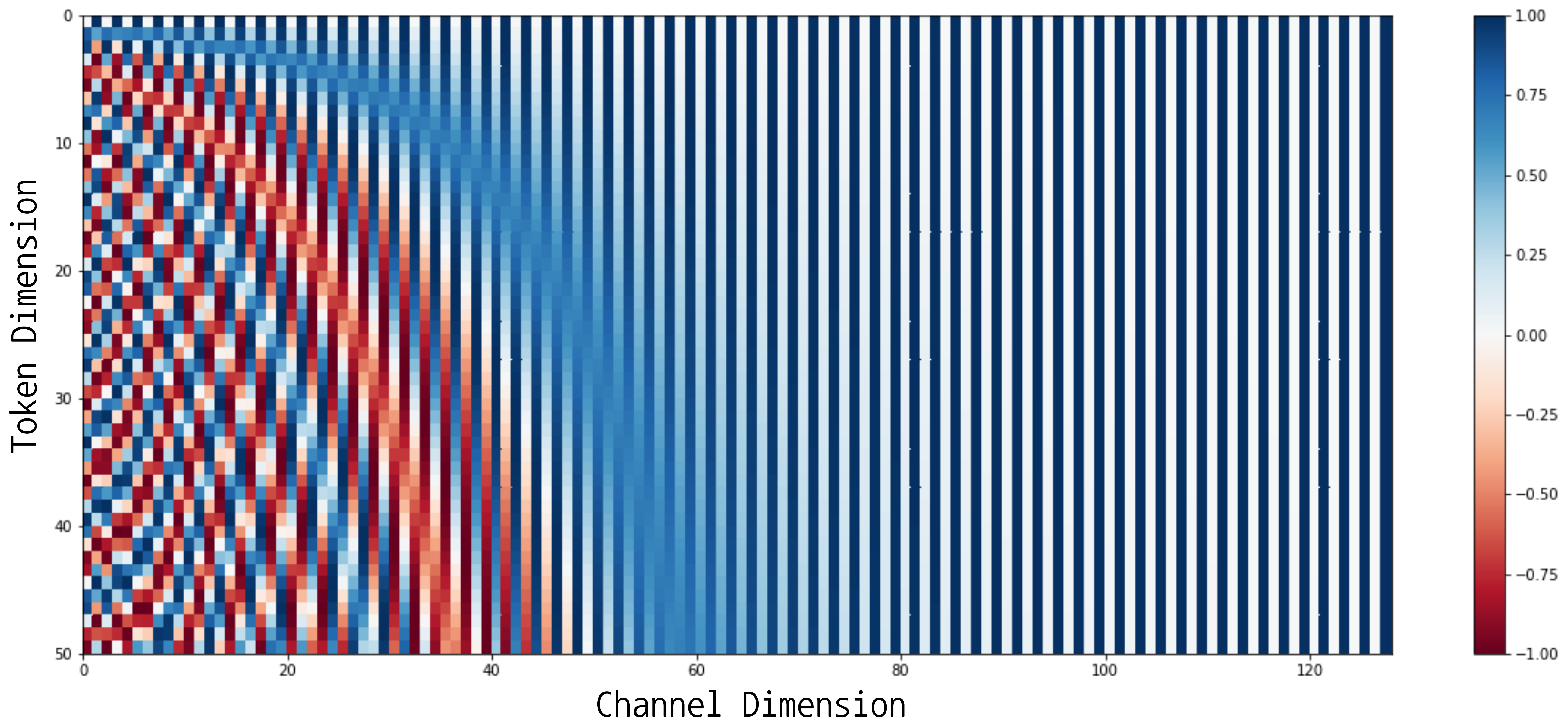
- Same as BatchNorm, but normalizes in ***feature dimension***.
- Applied for each sample, each token.

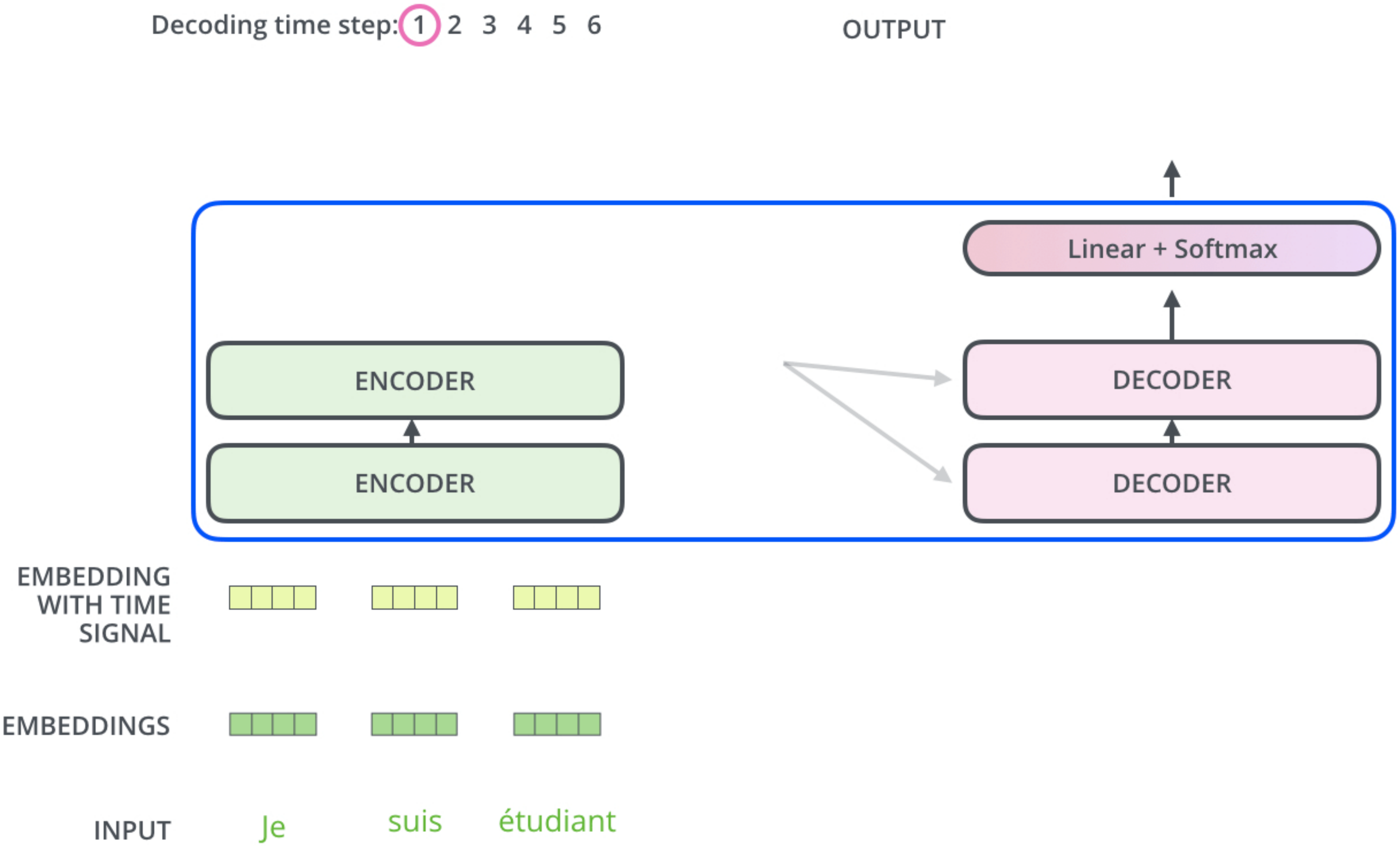


(4) Positional Encodings

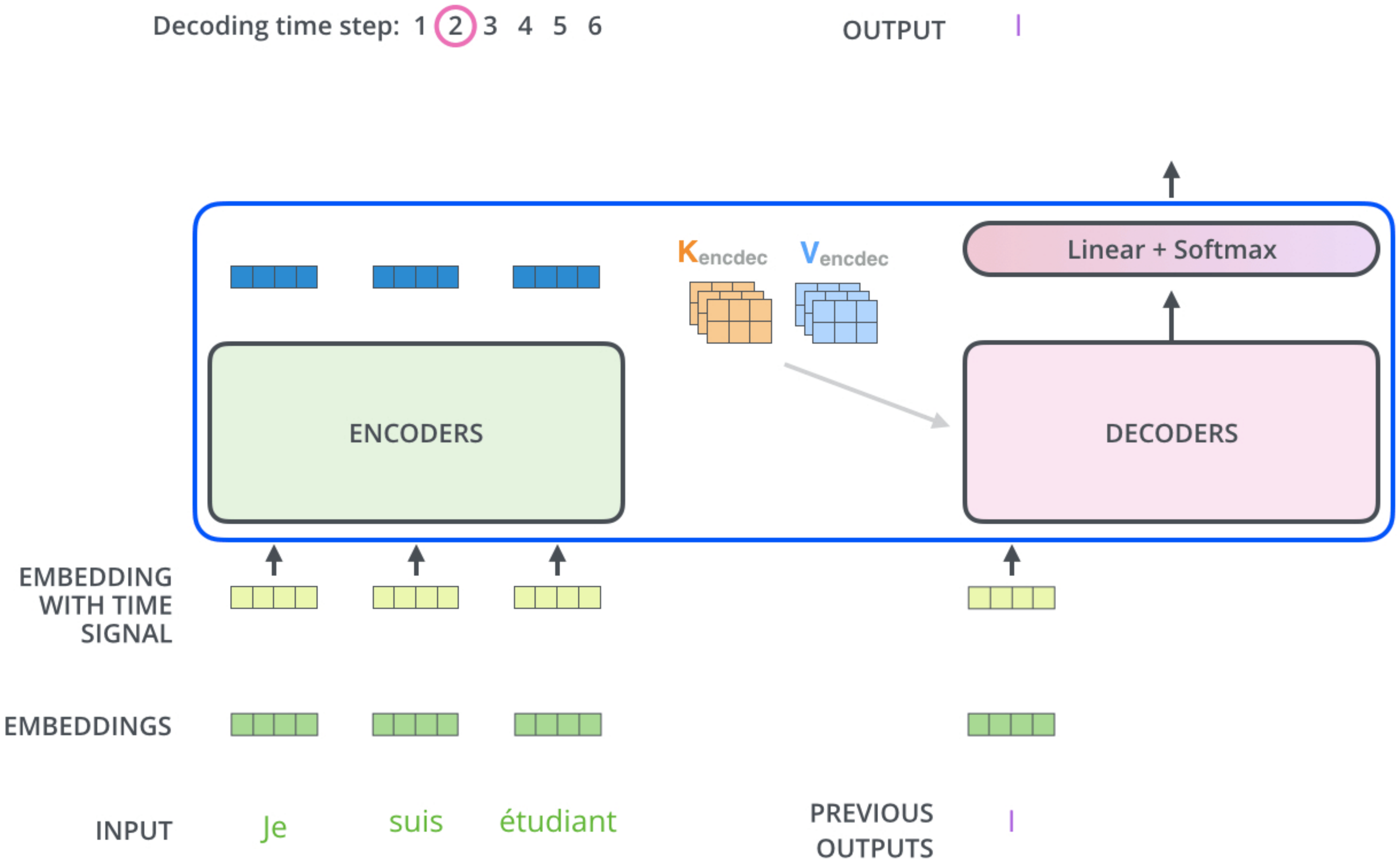
- Transformer architecture disregards the position information!
- To resolve this, we simply ***add*** position-specific info on the data.

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




Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

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



More references

Beginner

- Jay Alammar, "The Illustrated Transformer"
  - https://jalammar.github.io/illustrated-transformer/

Advanced

- Phuong and Hutter, "Formal Algorithms for Transformers," 2022
  - https://arxiv.org/abs/2207.09238
- He and Hoffman, "Simplifying Transformer Blocks," 2023
  - https://arxiv.org/abs/2311.01906

Cheers

- *Next up.* Training Language Models