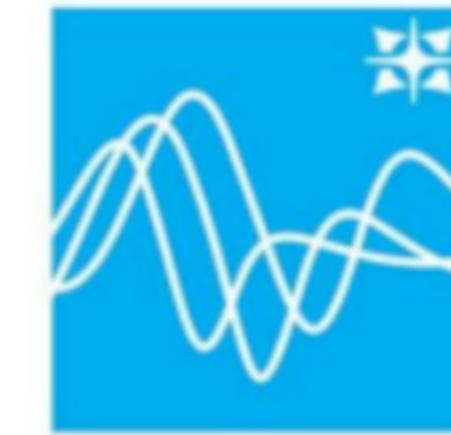
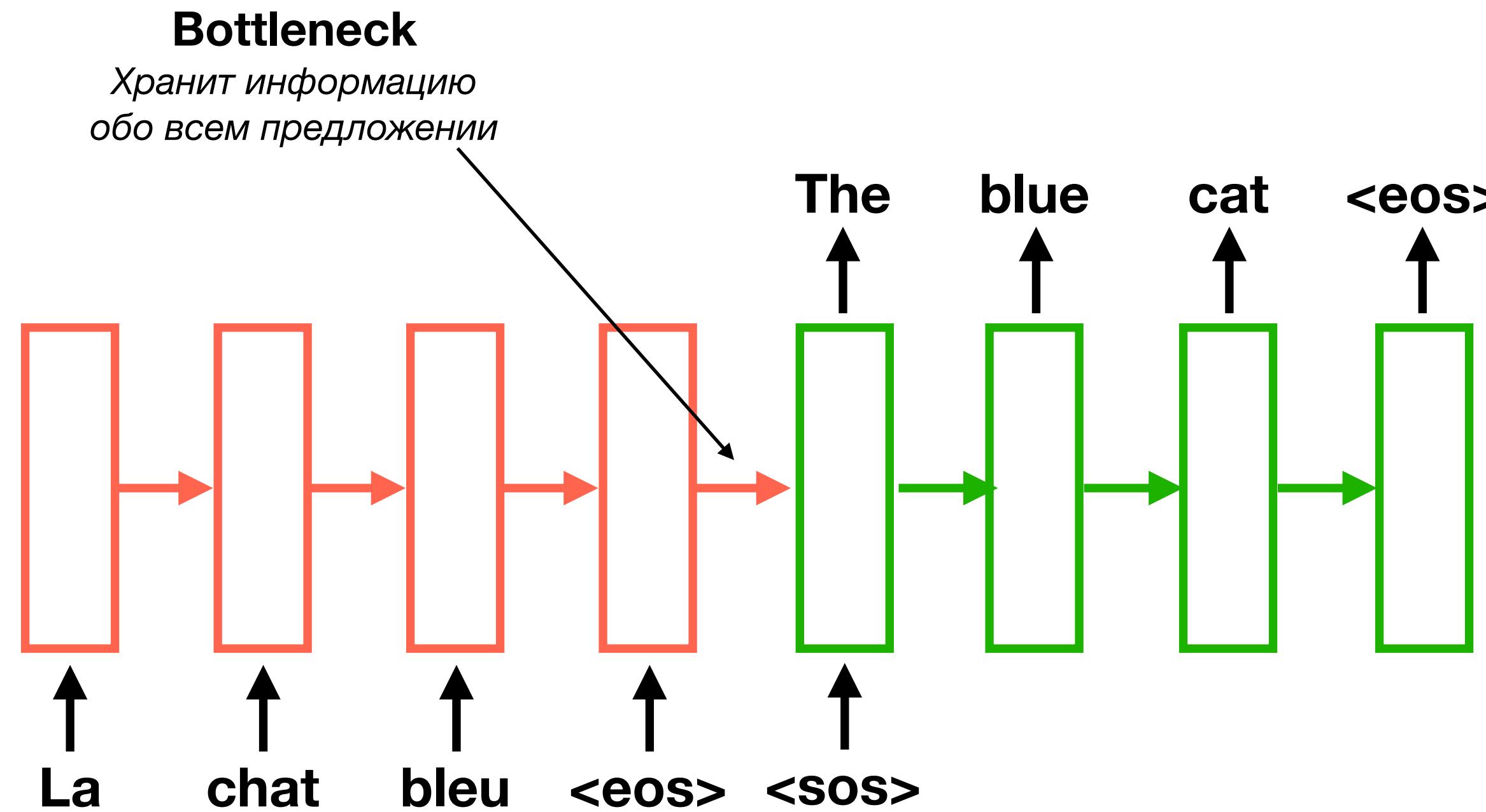


Attention is all you need

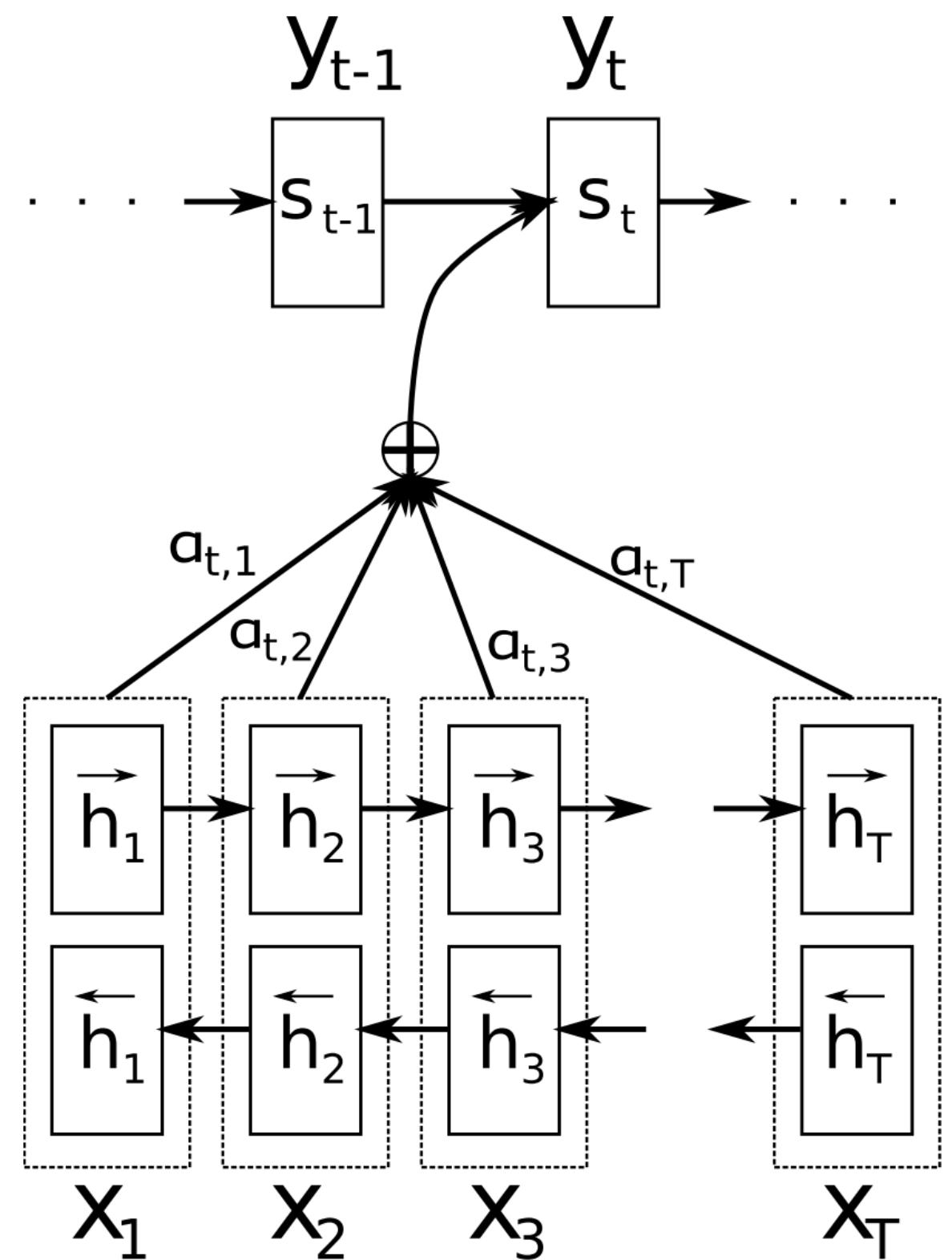
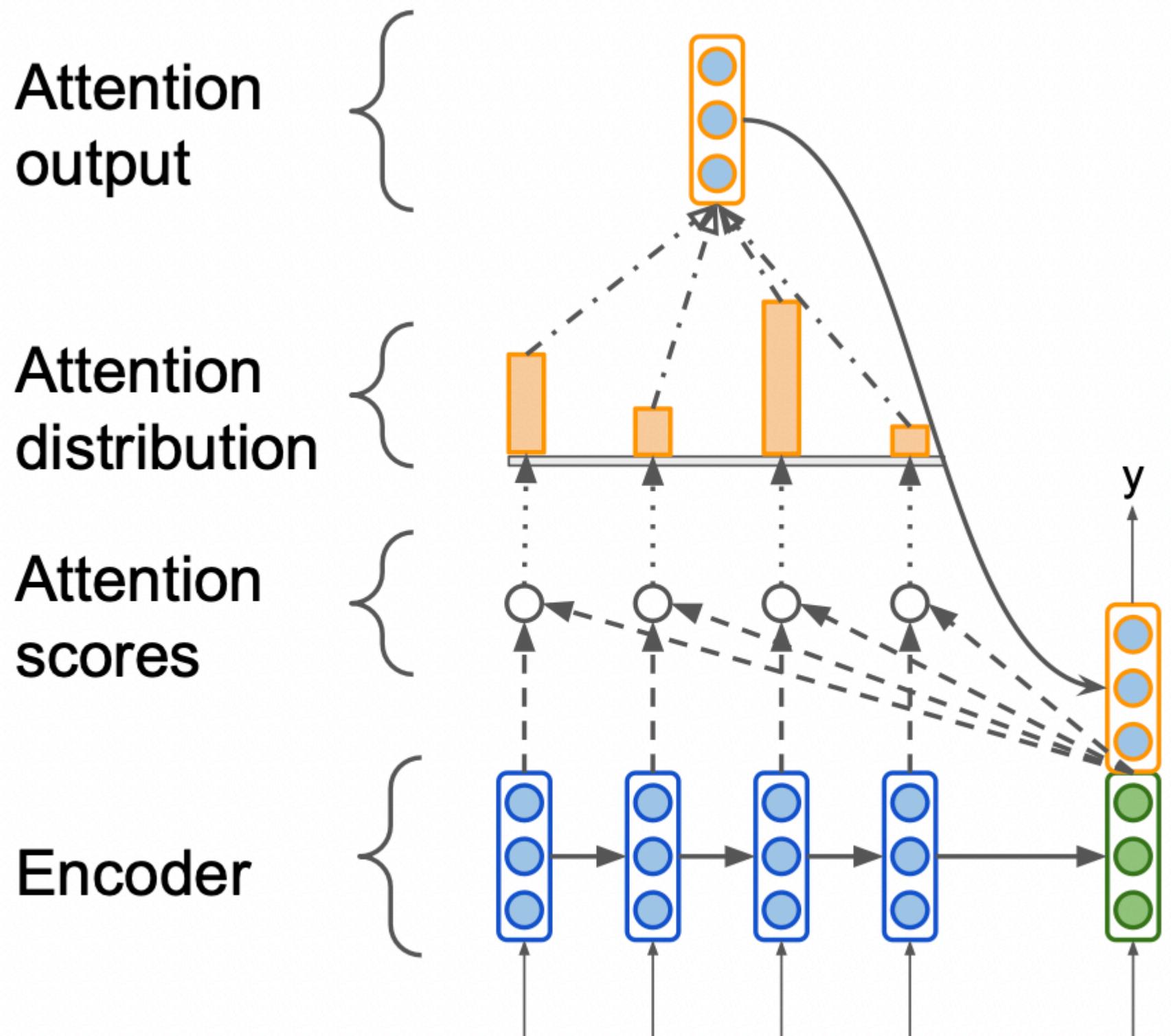


Кафедра
технологий
проектирования
сложных
технических
систем

Recap: Sequence to sequence



Recap: Attention



Скрытый слой RNN

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

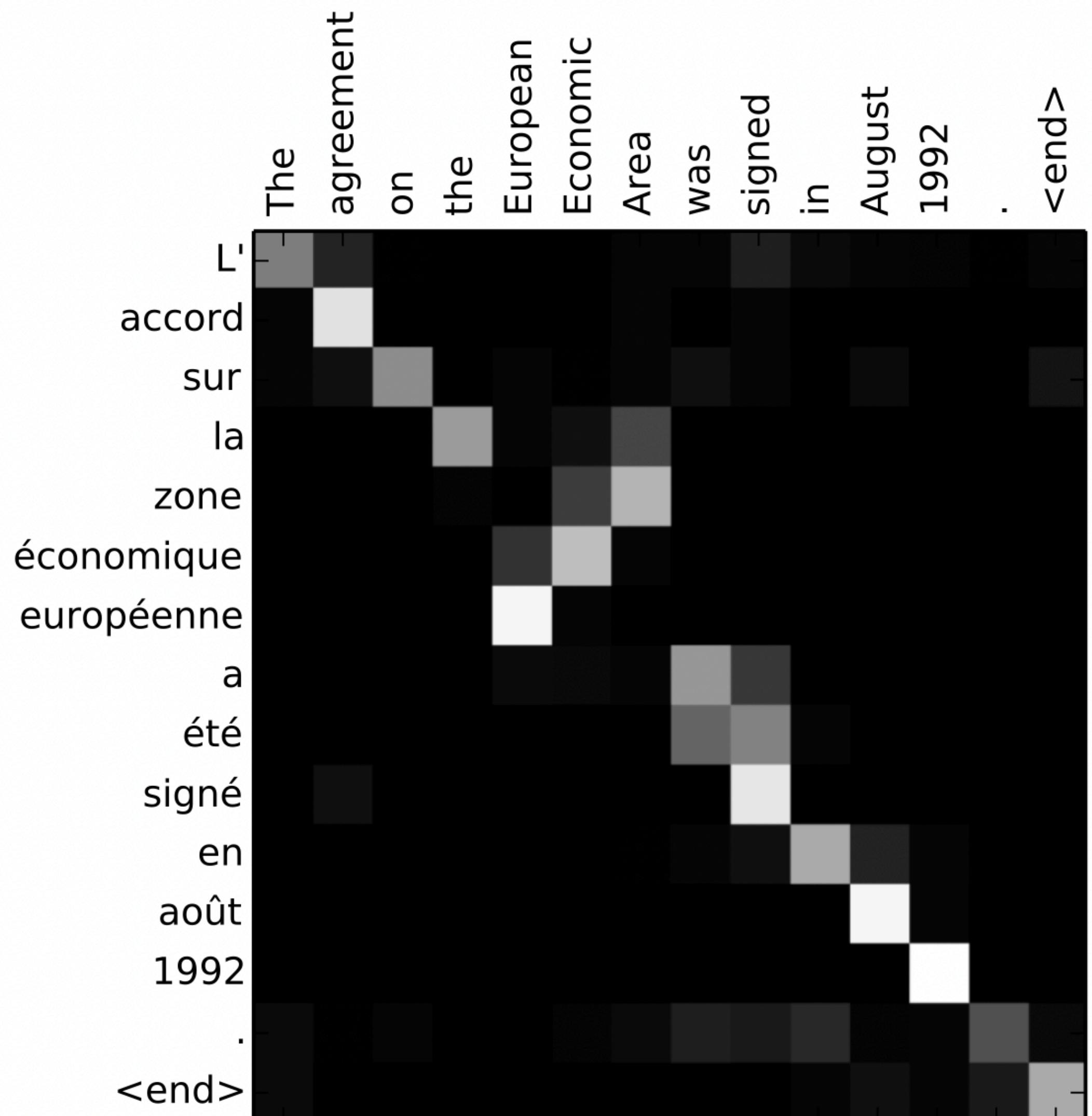
Взвешенная сумма

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

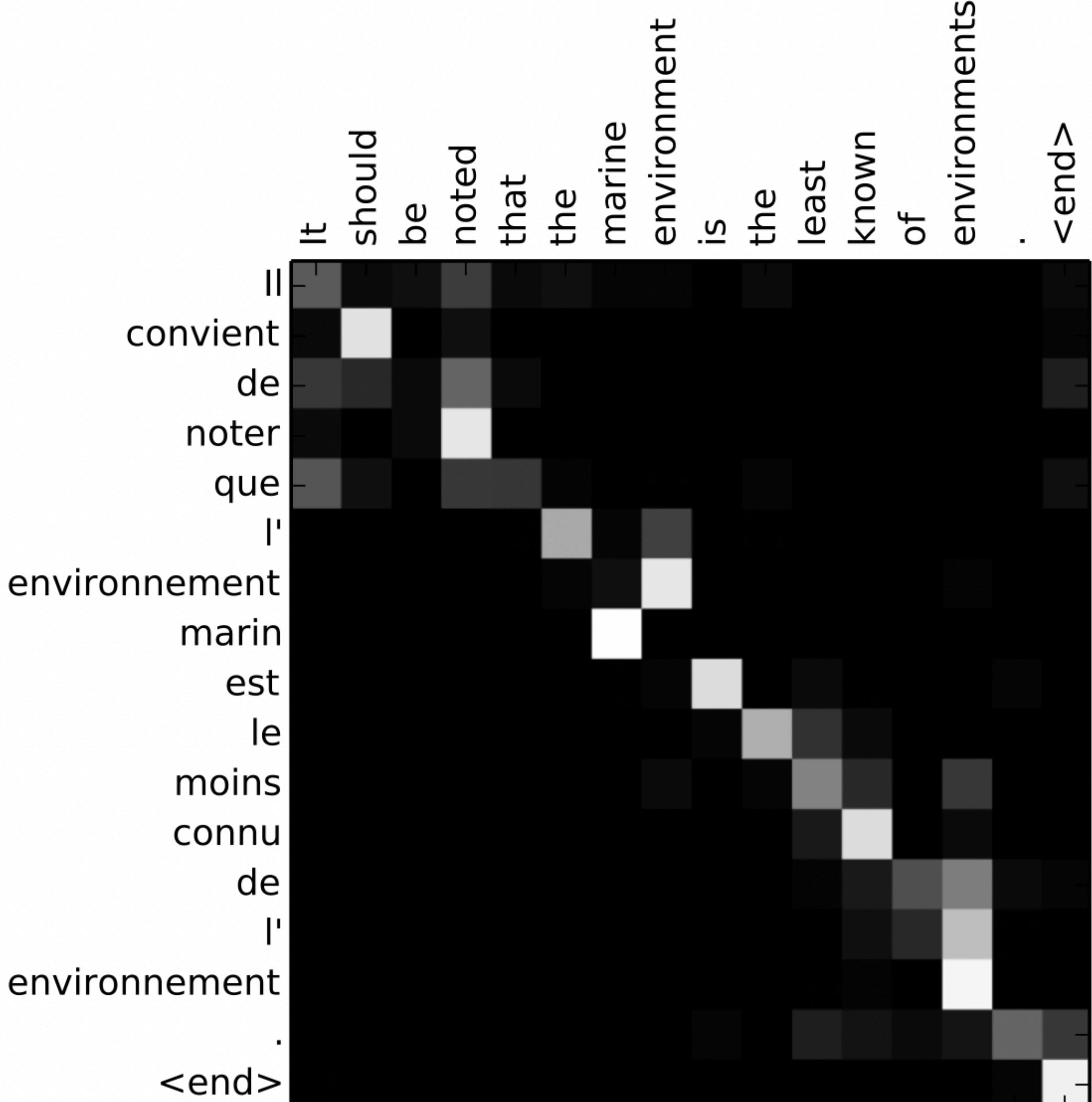
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

Recap: Attention



(a)

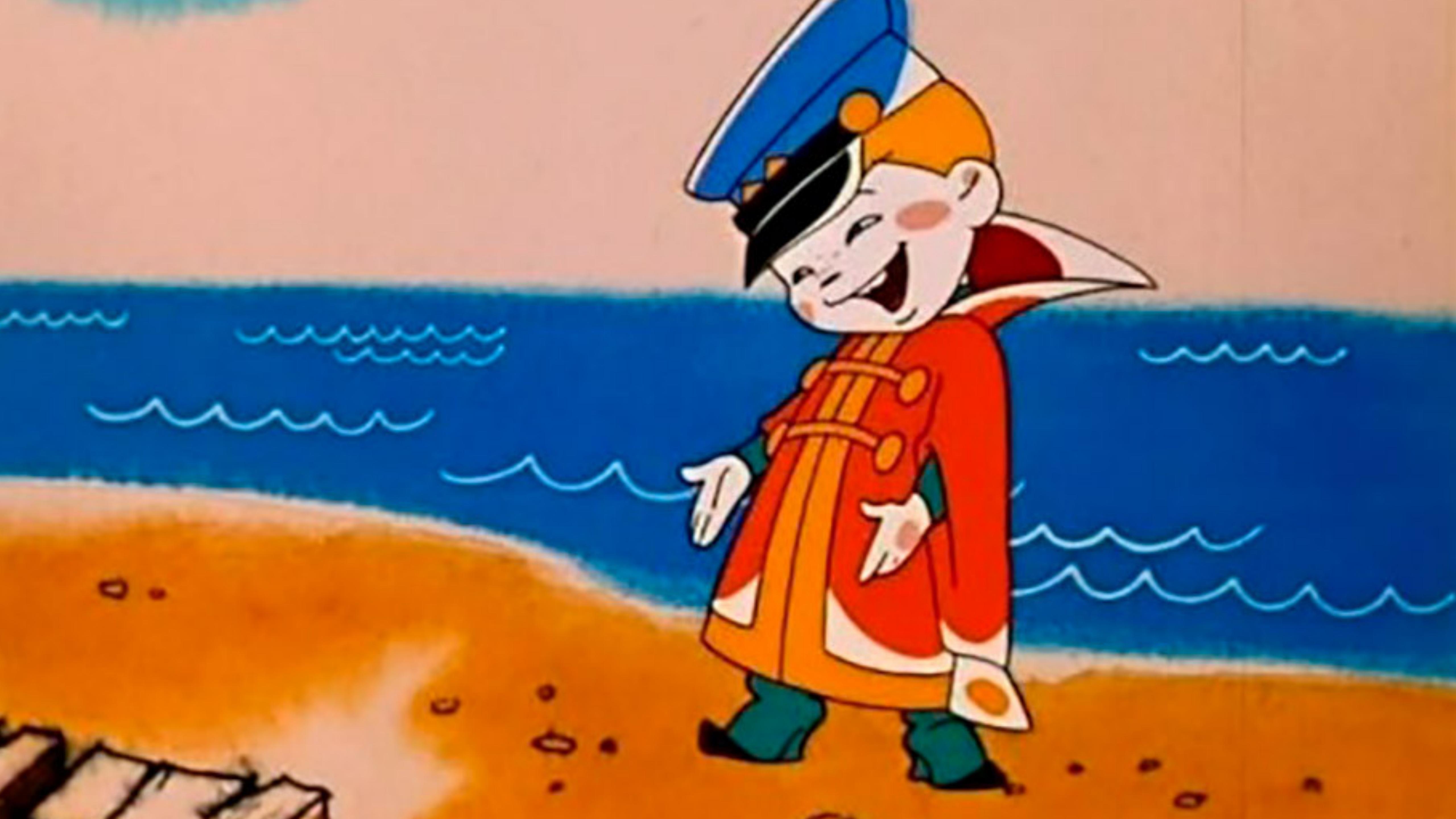


(b)

А что если без RNN?

А что если без RNN?

Но ведь нам нужно обрабатывать последовательности...



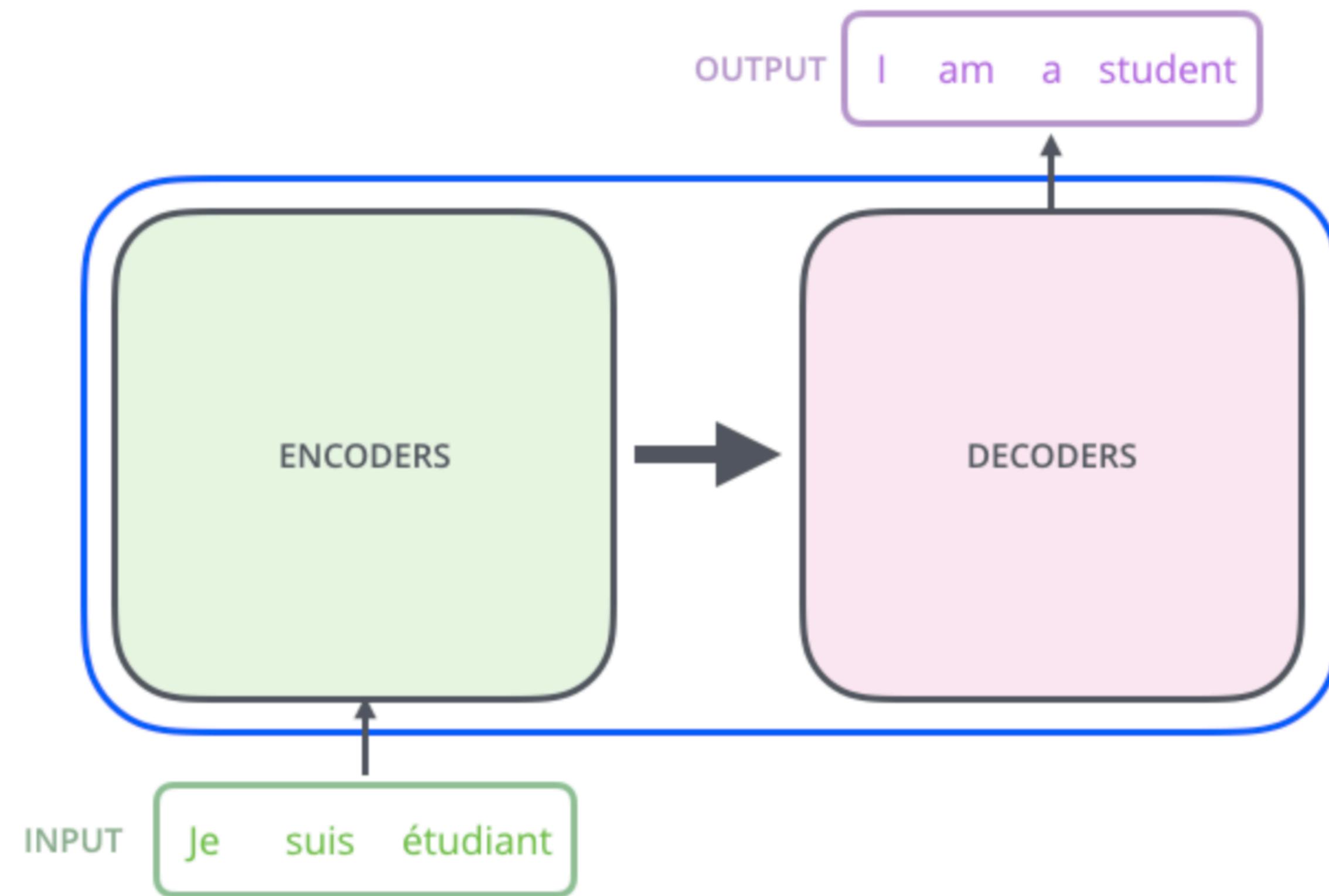
Self-Attention

The Illustrated Transformer

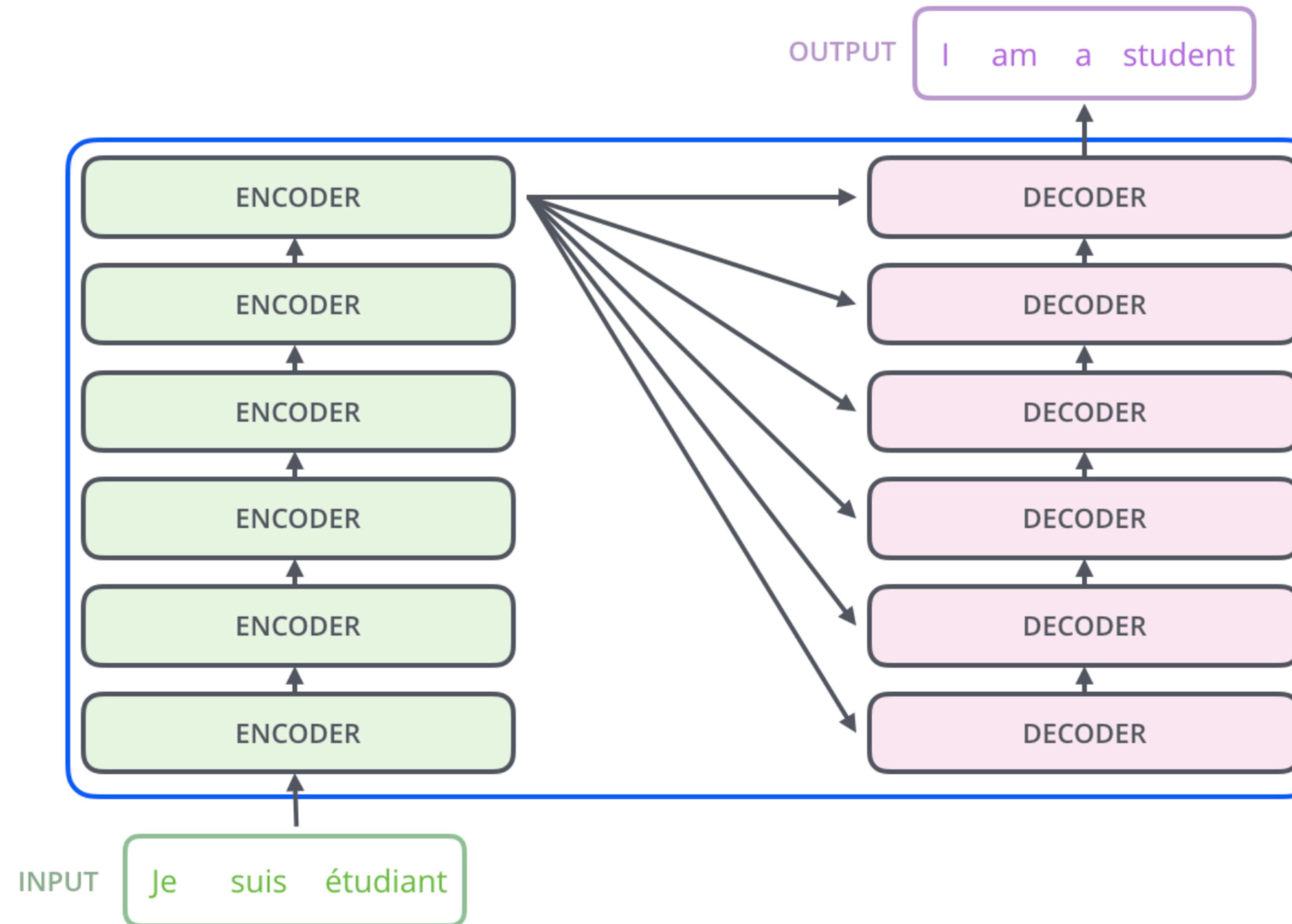
Self-Attention



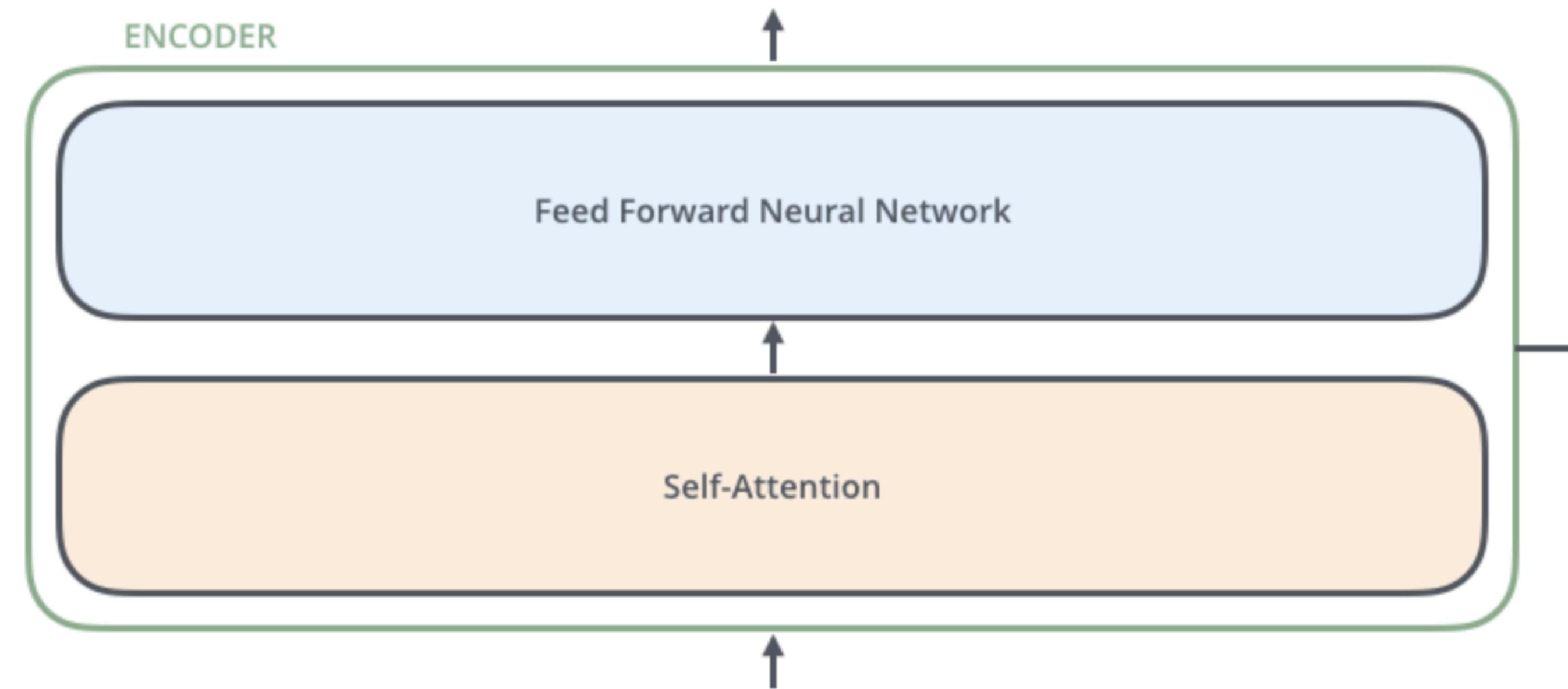
Self-Attention



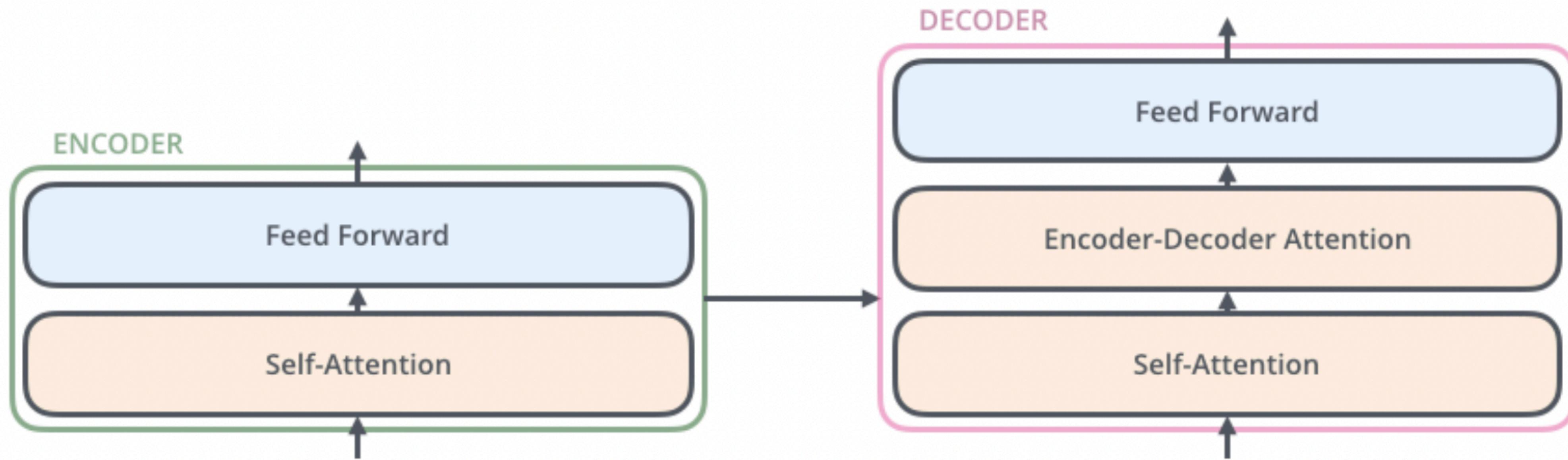
Self-Attention



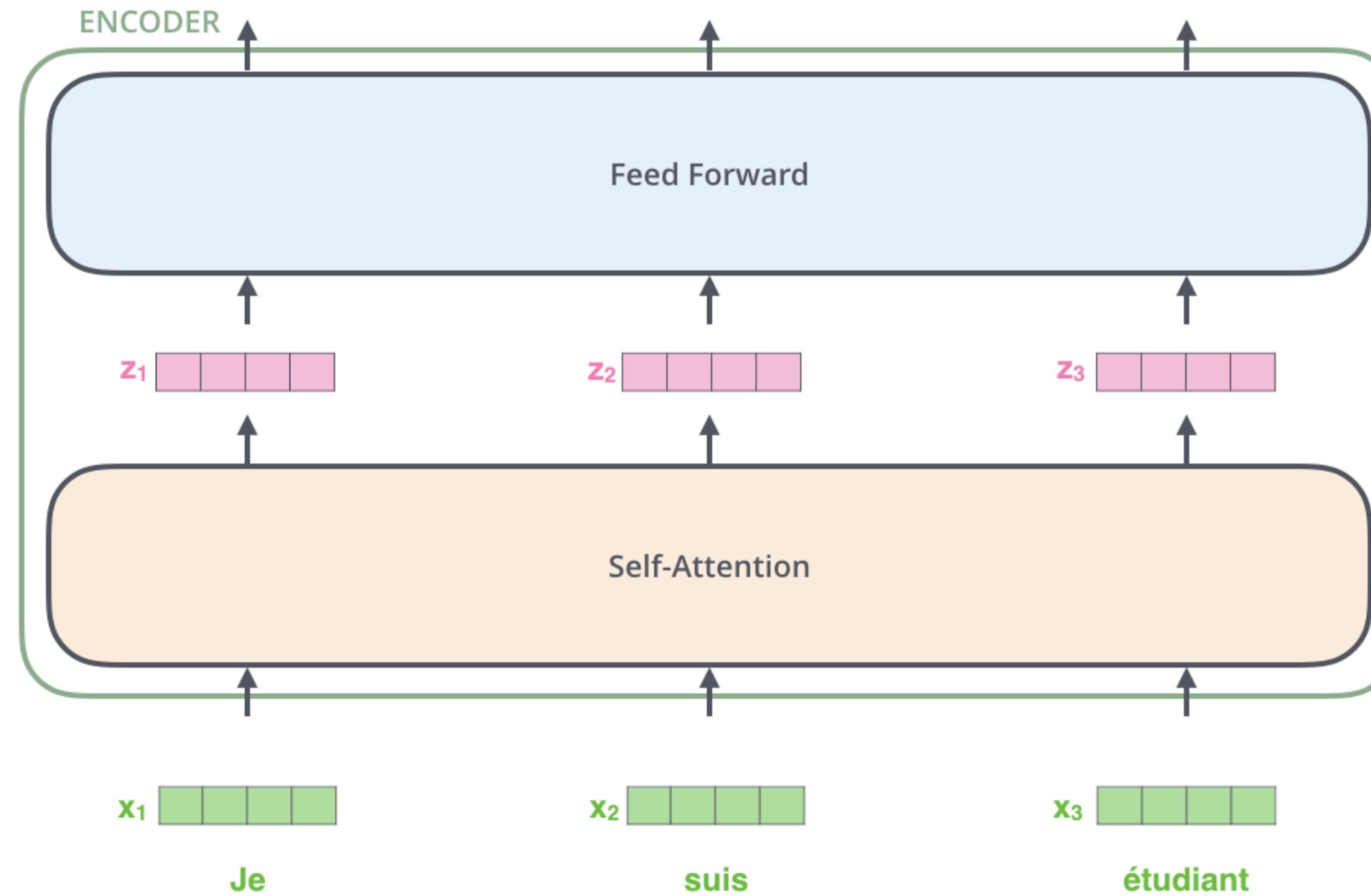
Self-Attention



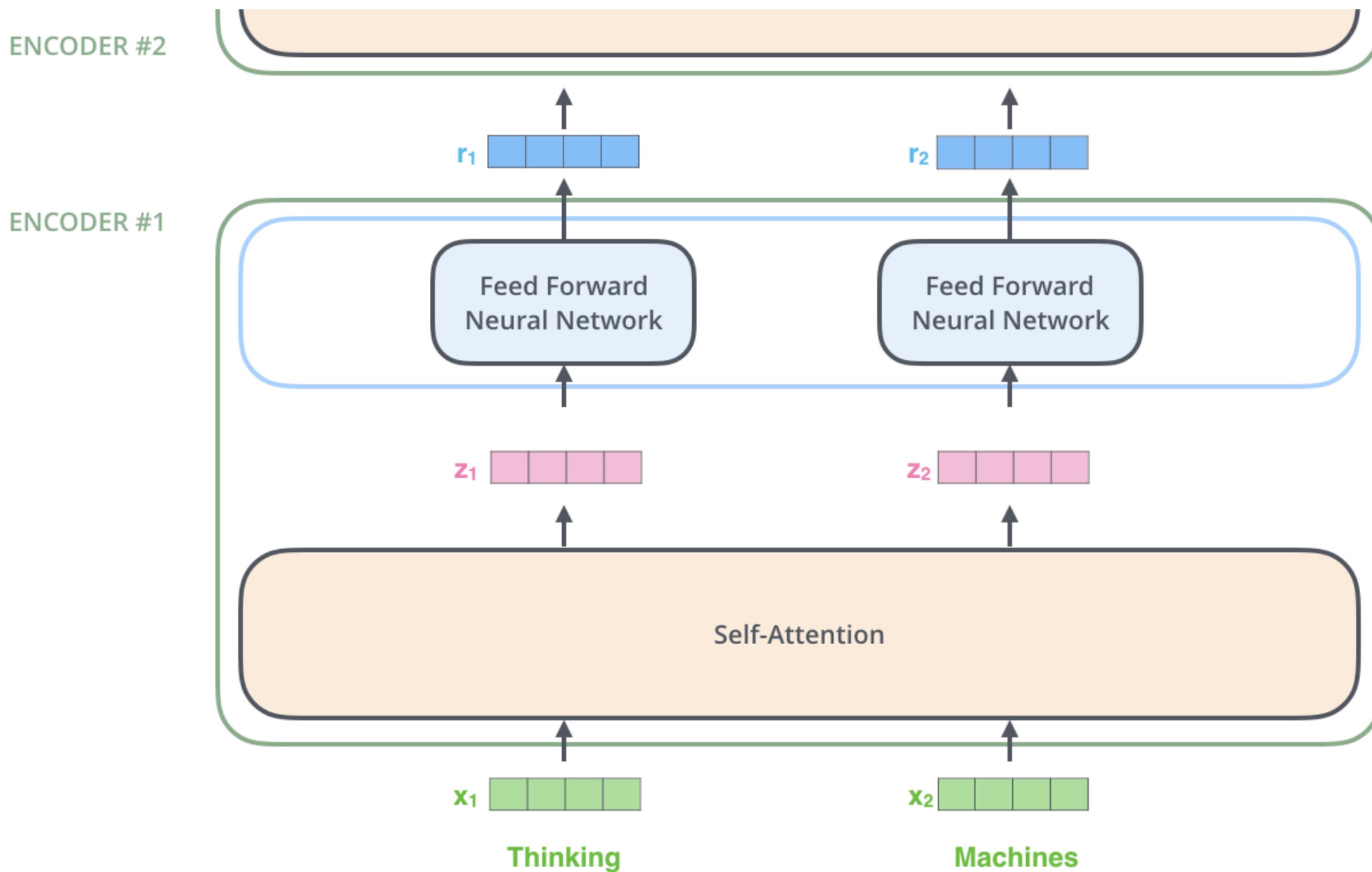
Self-Attention



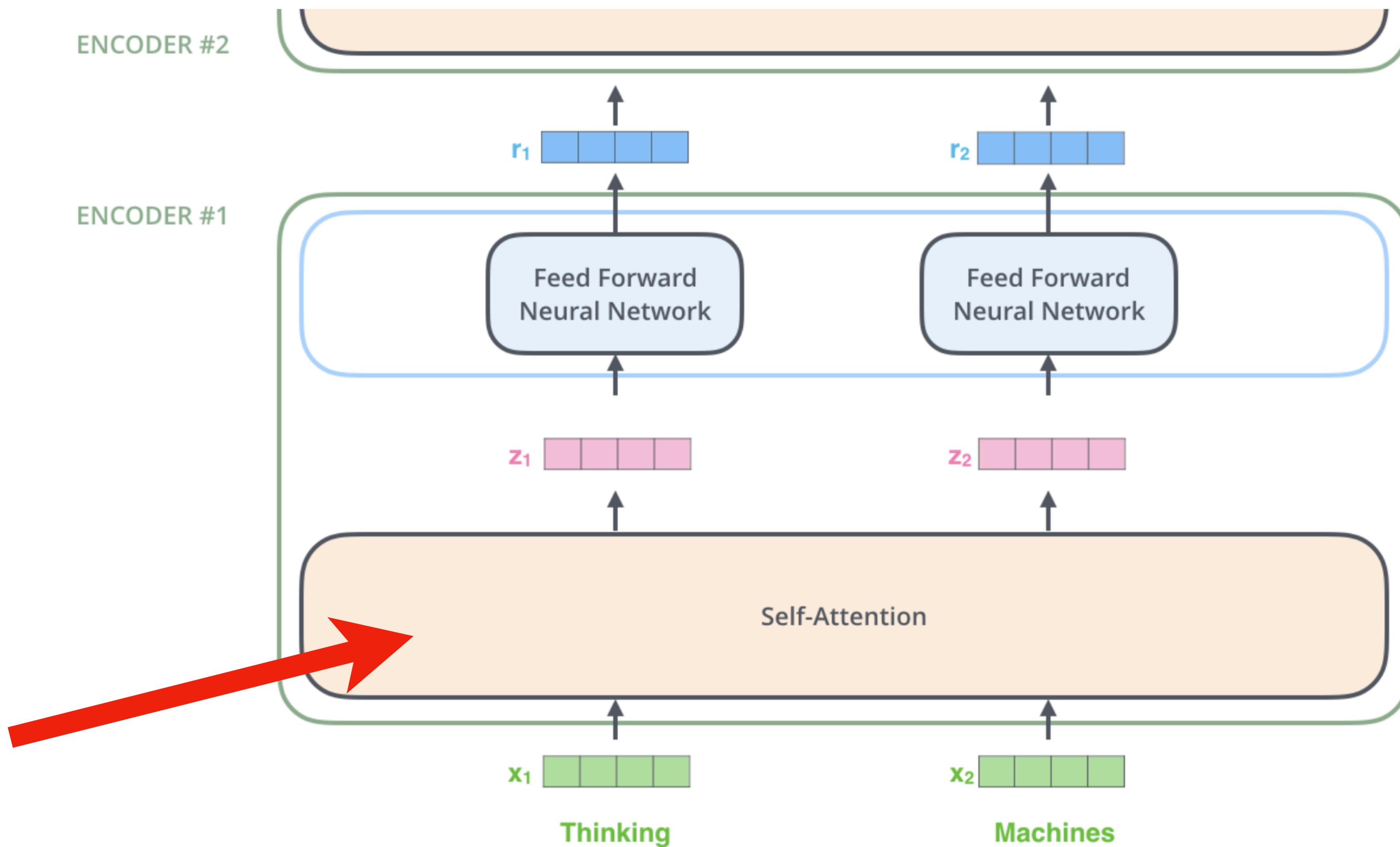
Self-Attention



Self-Attention

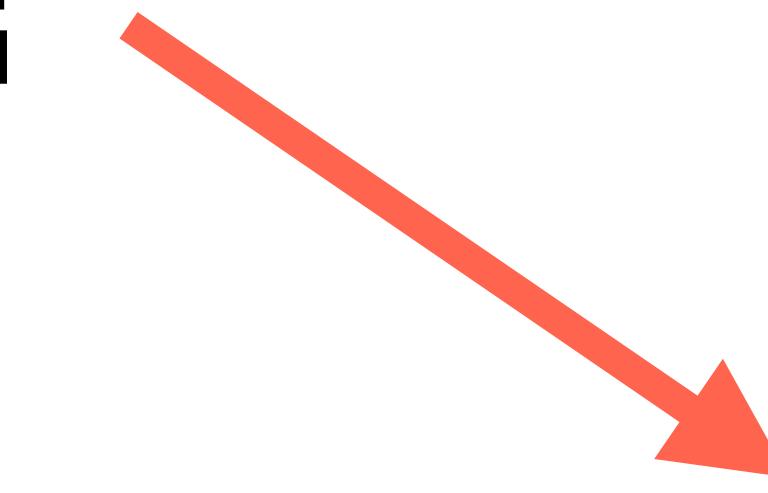


Self-Attention



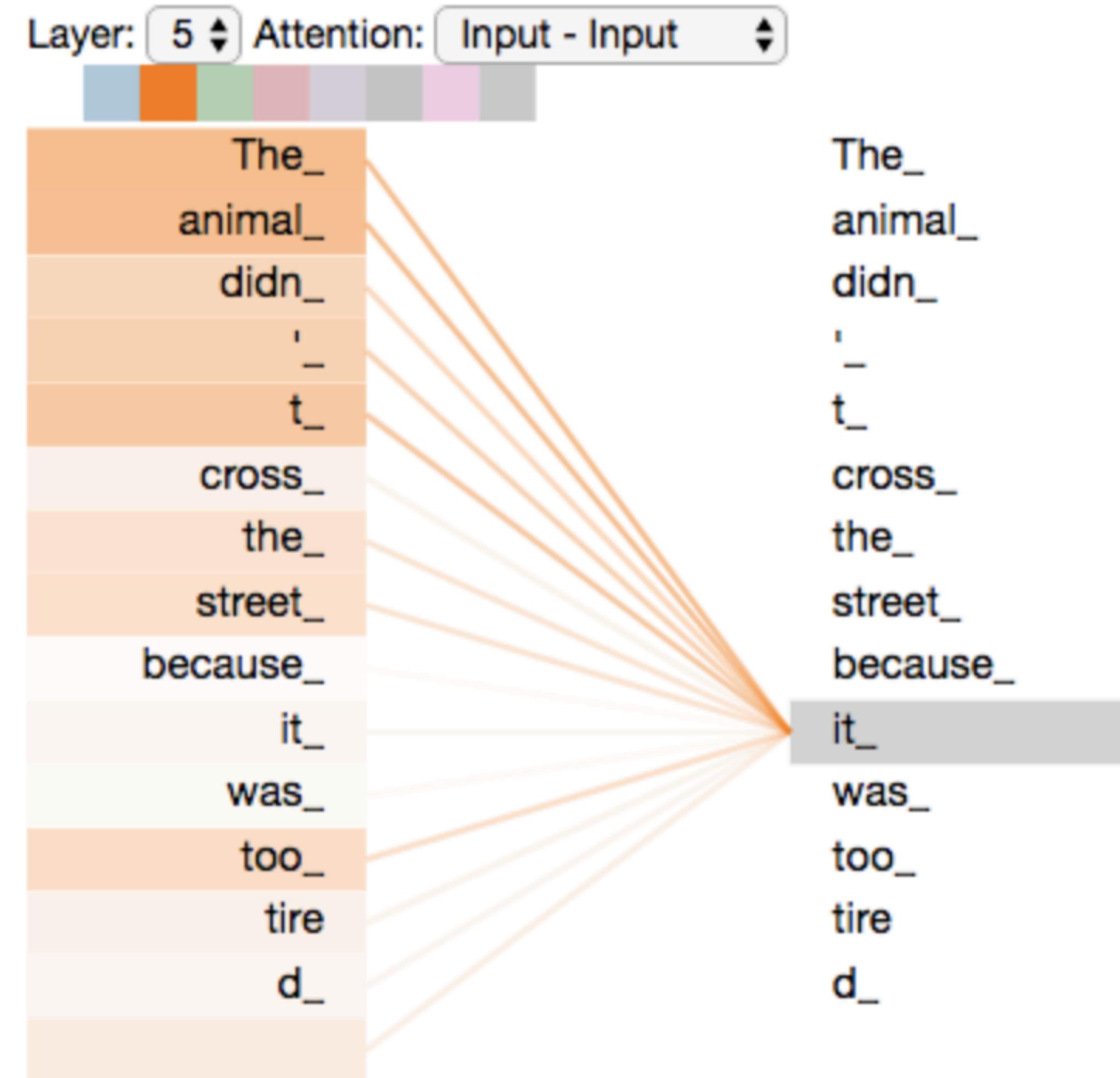
The animal didn't cross the street because it was too tired

?

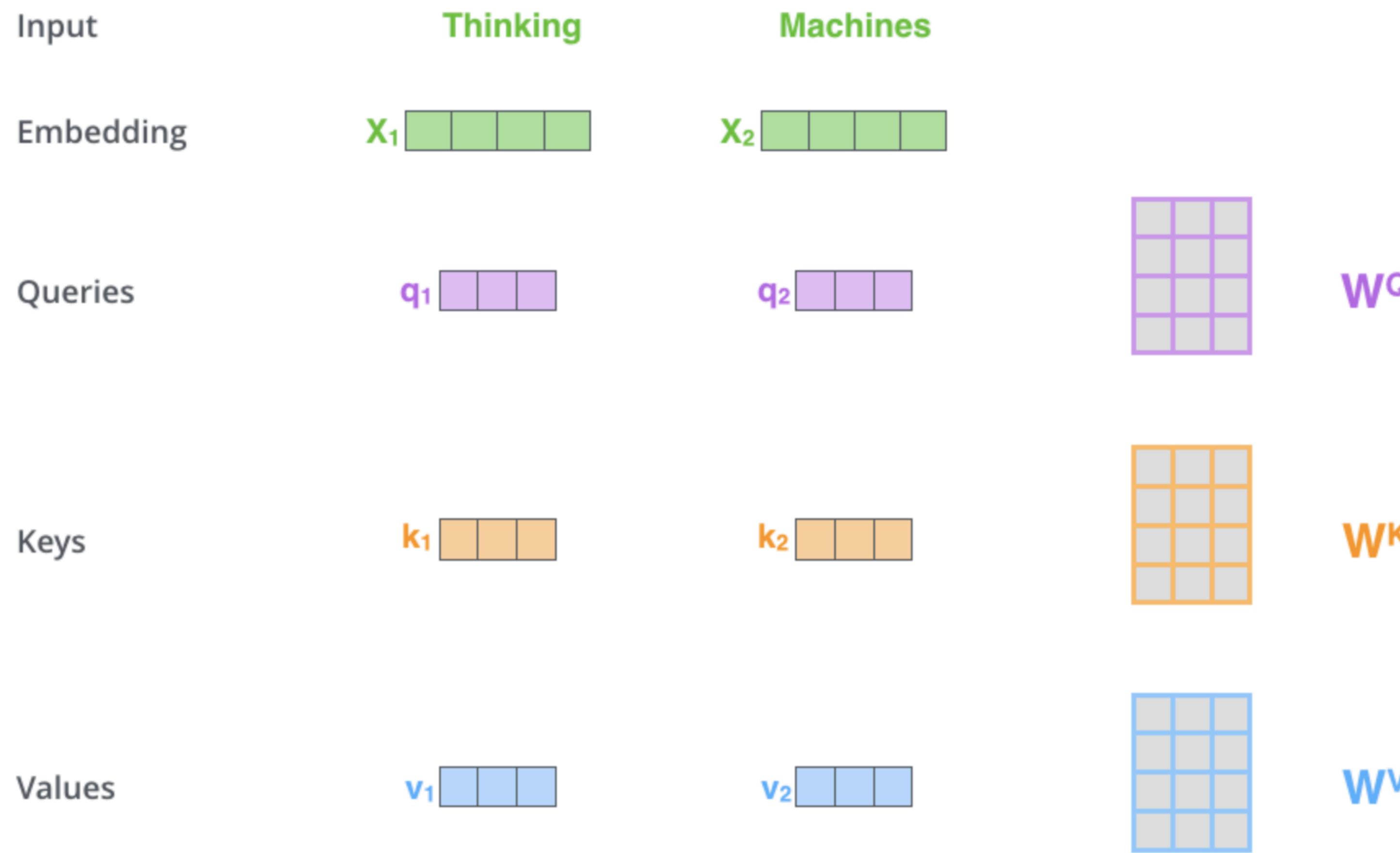


*The animal didn't cross the street because **it** was too tired*

Self-Attention



Шаг 1 - Векторы Queries, Keys, Values



Шаг 1 - Векторы Queries, Keys, Values

$$\mathbf{X} \times \mathbf{W}^Q = \mathbf{Q}$$

A diagram illustrating the first step of generating Query, Key, and Value vectors. It shows a green input matrix \mathbf{X} (3 rows, 4 columns) multiplied by a purple weight matrix \mathbf{W}^Q (4 rows, 4 columns) to produce a purple output matrix \mathbf{Q} (3 rows, 4 columns). The matrices are represented as grids of colored squares.

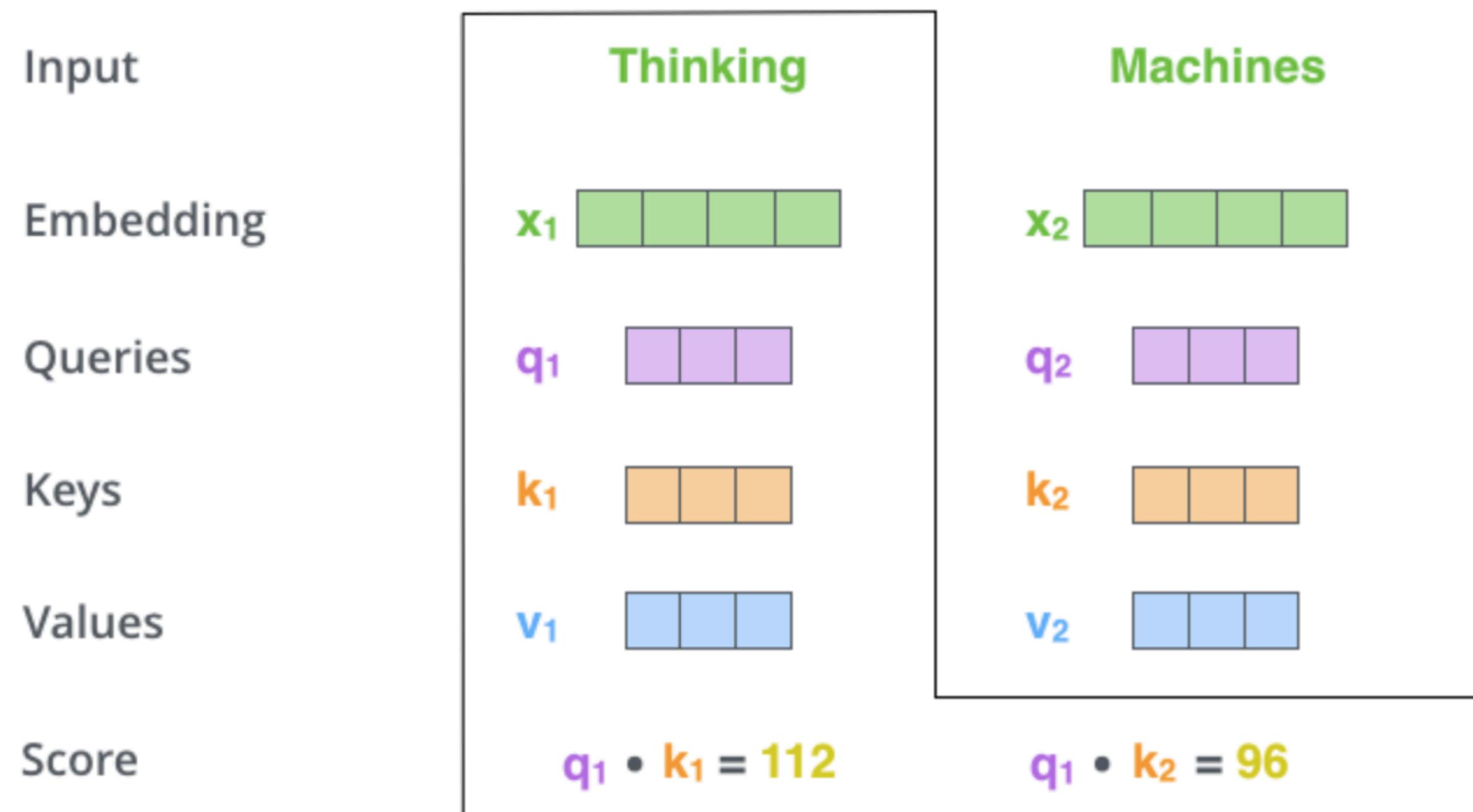
$$\mathbf{X} \times \mathbf{W}^K = \mathbf{K}$$

A diagram illustrating the second step of generating Query, Key, and Value vectors. It shows a green input matrix \mathbf{X} (3 rows, 4 columns) multiplied by an orange weight matrix \mathbf{W}^K (4 rows, 4 columns) to produce an orange output matrix \mathbf{K} (3 rows, 4 columns). The matrices are represented as grids of colored squares.

$$\mathbf{X} \times \mathbf{W}^V = \mathbf{V}$$

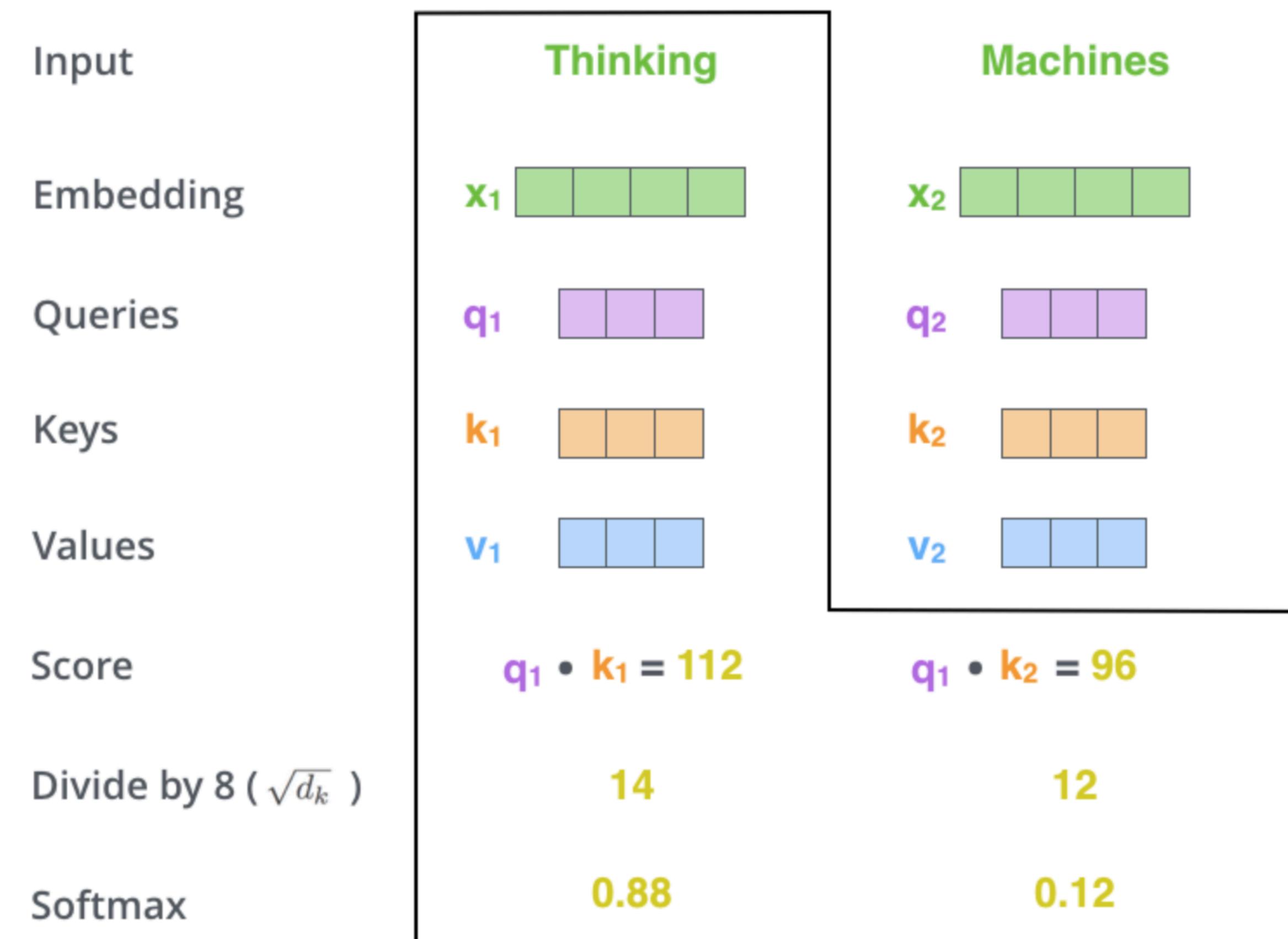
A diagram illustrating the third step of generating Query, Key, and Value vectors. It shows a green input matrix \mathbf{X} (3 rows, 4 columns) multiplied by a blue weight matrix \mathbf{W}^V (4 rows, 4 columns) to produce a blue output matrix \mathbf{V} (3 rows, 4 columns). The matrices are represented as grids of colored squares.

Шаг 2 - Вычисление Score



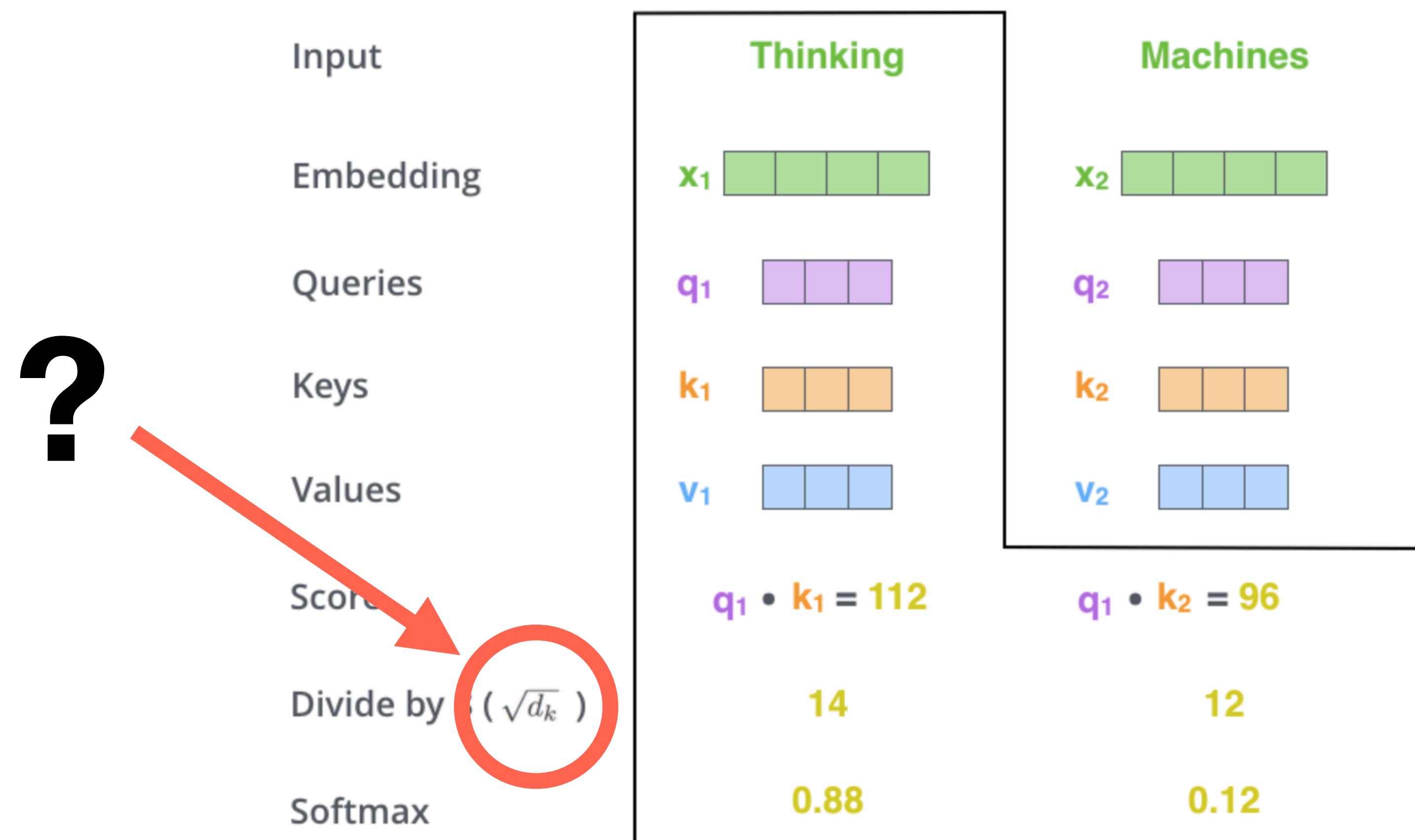
Шаг 3: Делим Скор на $\text{sqrt}(d)$

Шаг 4: Softmax



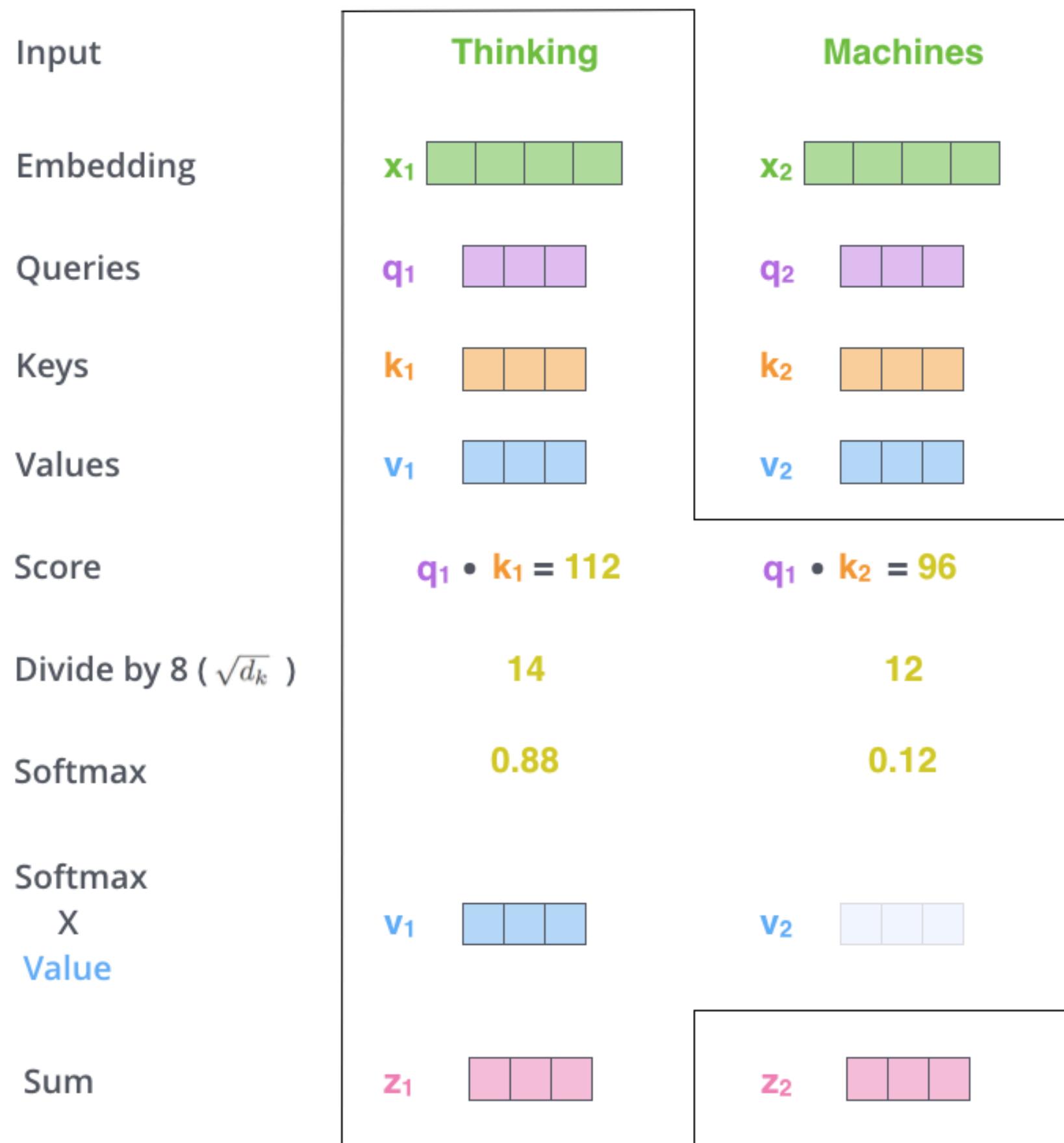
Шаг 3: Делим Скор на $\text{sqrt}(d)$

Шаг 4: Softmax

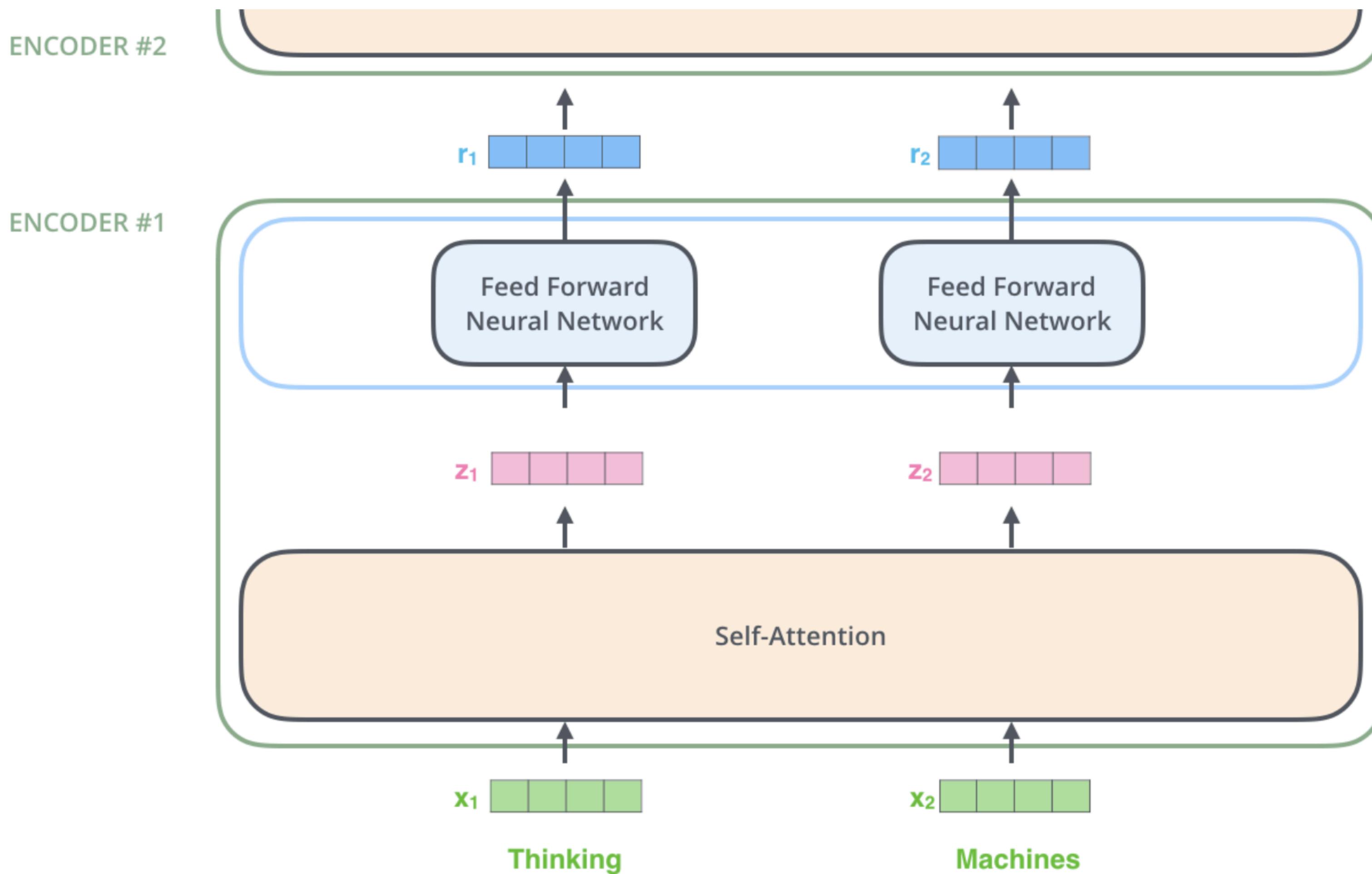


Шаг 5: Умножение каждого Values вектора на значение softmax

Шаг 6: Суммирование векторов



Self-Attention



Шаг 1 - Векторы Queries, Keys, Values

$$\mathbf{X} \times \mathbf{W}^Q = \mathbf{Q}$$

$$\mathbf{X} \times \mathbf{W}^K = \mathbf{K}$$

$$\mathbf{X} \times \mathbf{W}^V = \mathbf{V}$$

Self-Attention

$$\text{softmax} \left(\frac{\begin{matrix} \mathbf{Q} & \mathbf{K}^T \\ \times & \end{matrix}}{\sqrt{d_k}} \right) \mathbf{V}$$

\mathbf{Q} \mathbf{K}^T

$=$

\mathbf{z}

The diagram illustrates the Self-Attention mechanism. It shows the computation of attention weights from query matrix \mathbf{Q} and key matrix \mathbf{K}^T , scaled by $\sqrt{d_k}$, and multiplied by value matrix \mathbf{V} to produce output matrix \mathbf{z} . The matrices are represented as 3x3 grids of colored squares.

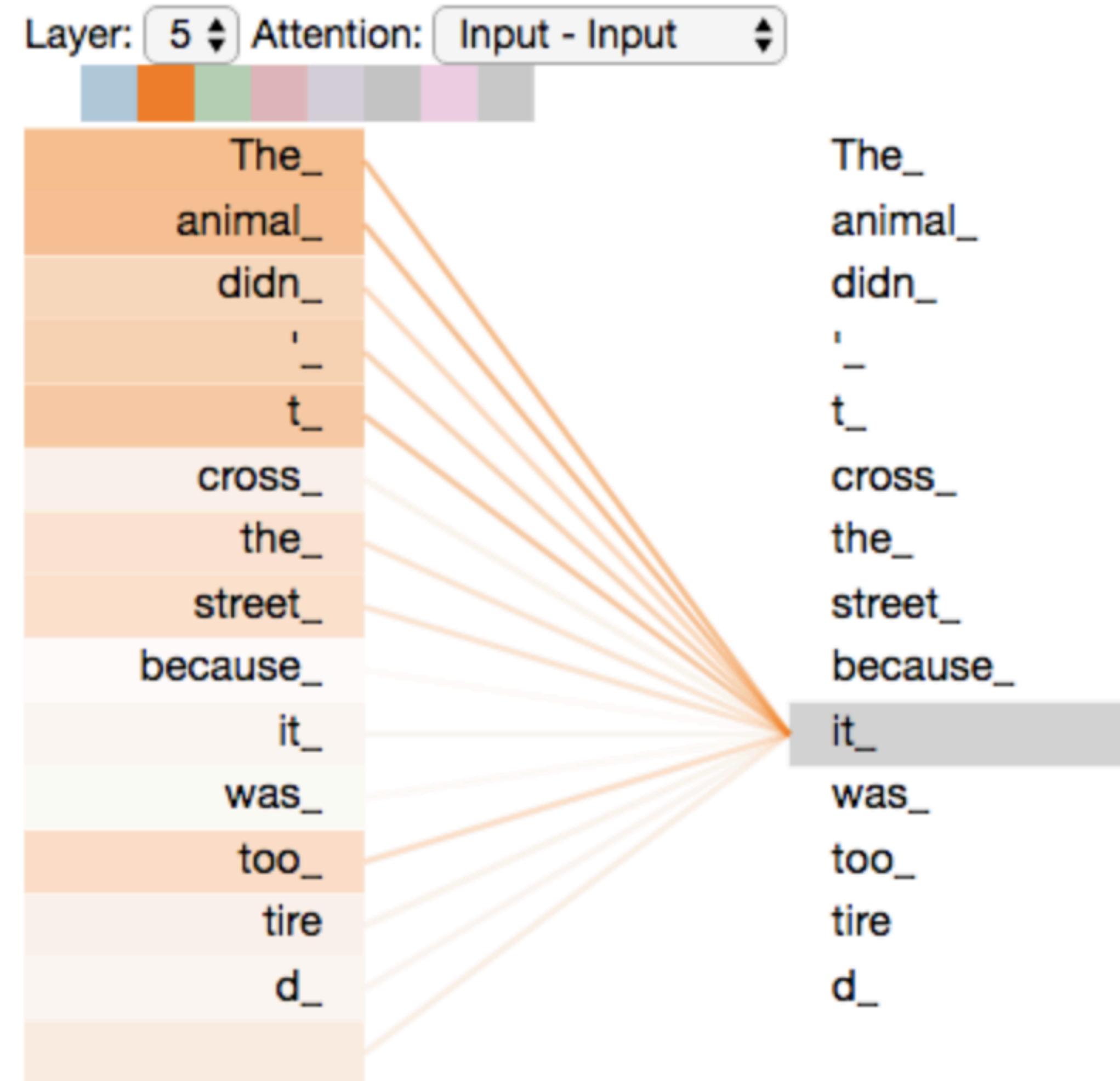
Purple	Purple	Purple
Purple	Purple	Purple
Purple	Purple	Purple

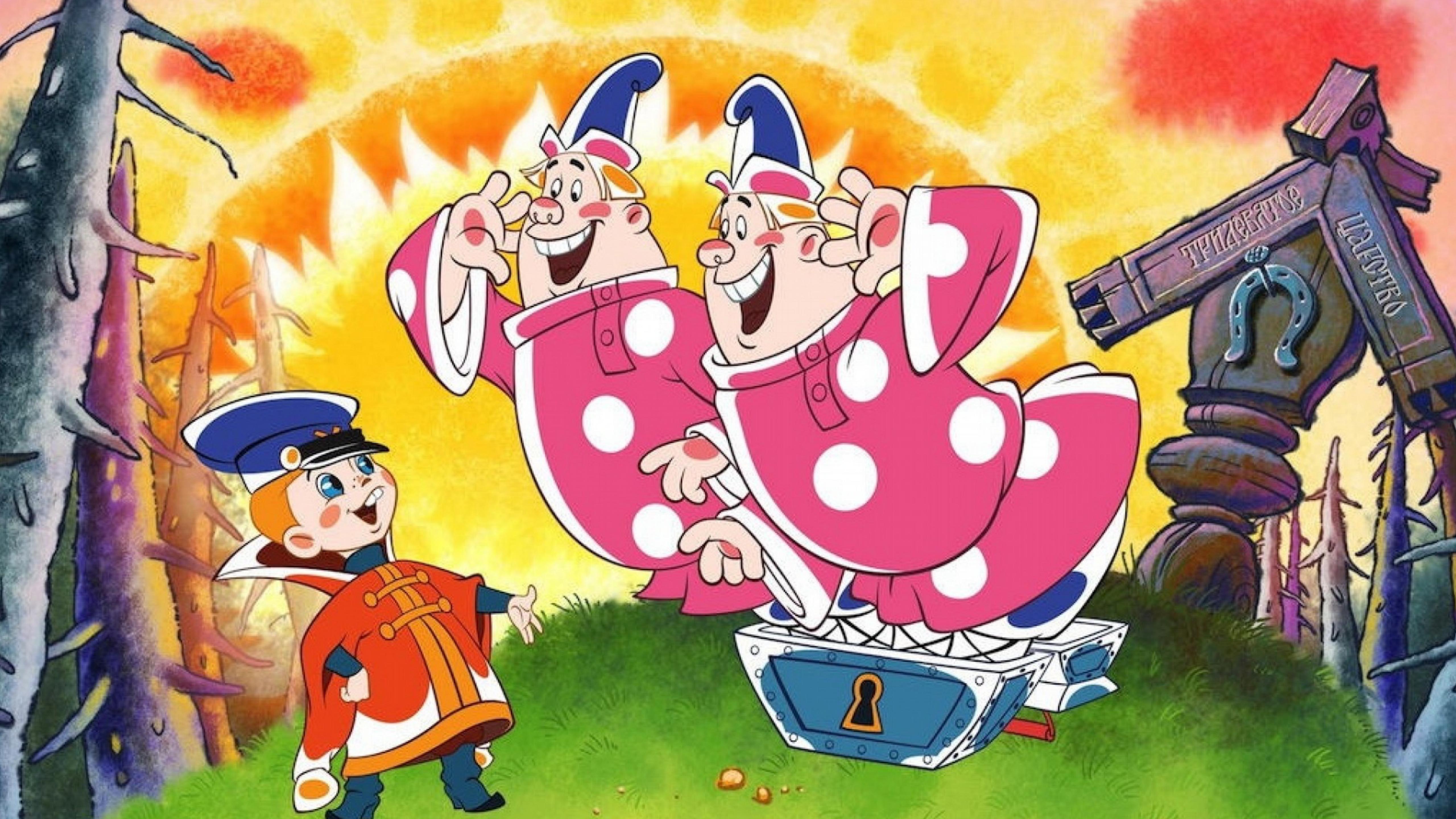
Orange	Orange	Orange
Orange	Orange	Orange
Orange	Orange	Orange

Blue	Blue	Blue
Blue	Blue	Blue
Blue	Blue	Blue

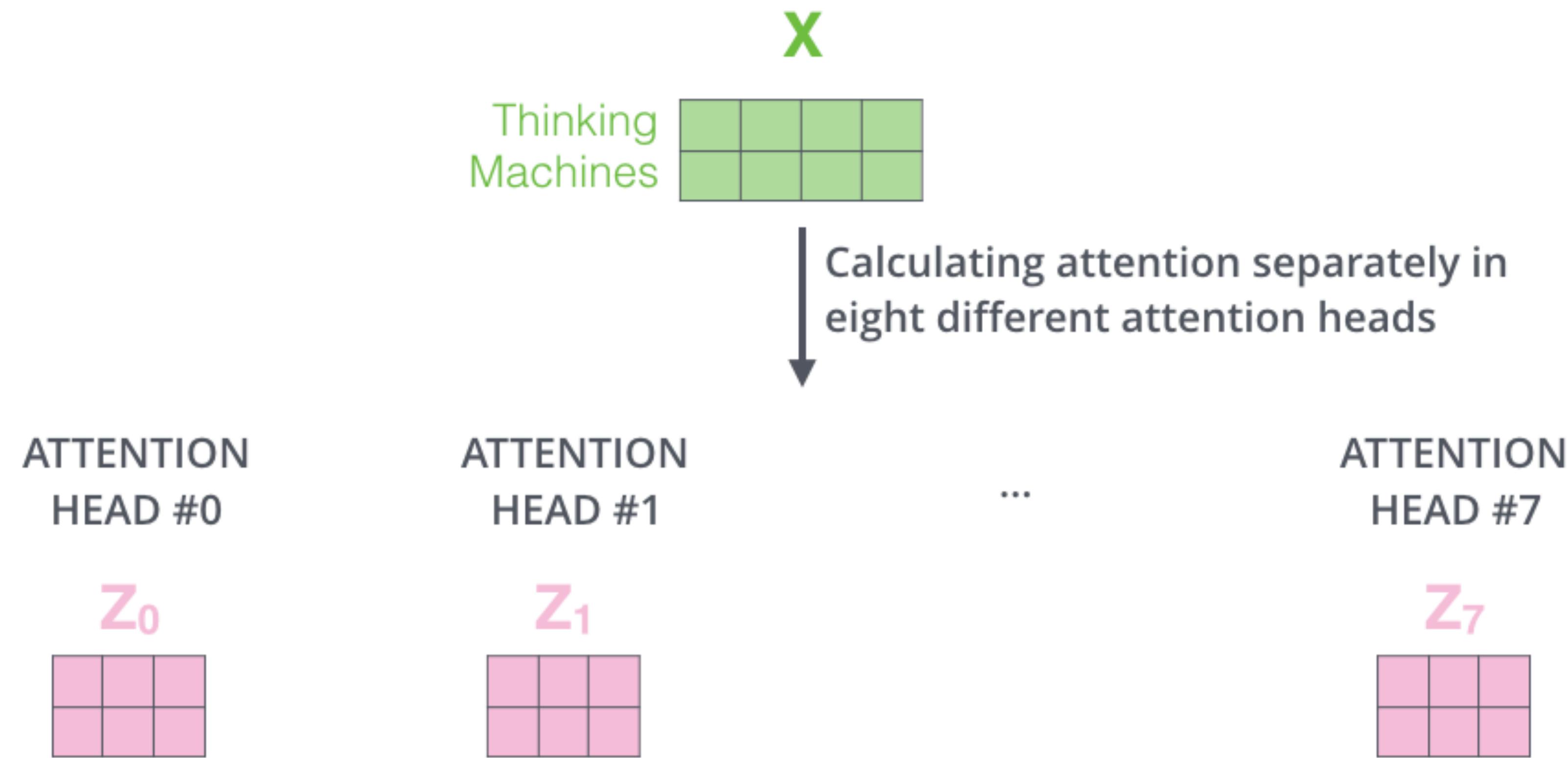
Pink	Pink	Pink
Pink	Pink	Pink
Pink	Pink	Pink

Self-Attention



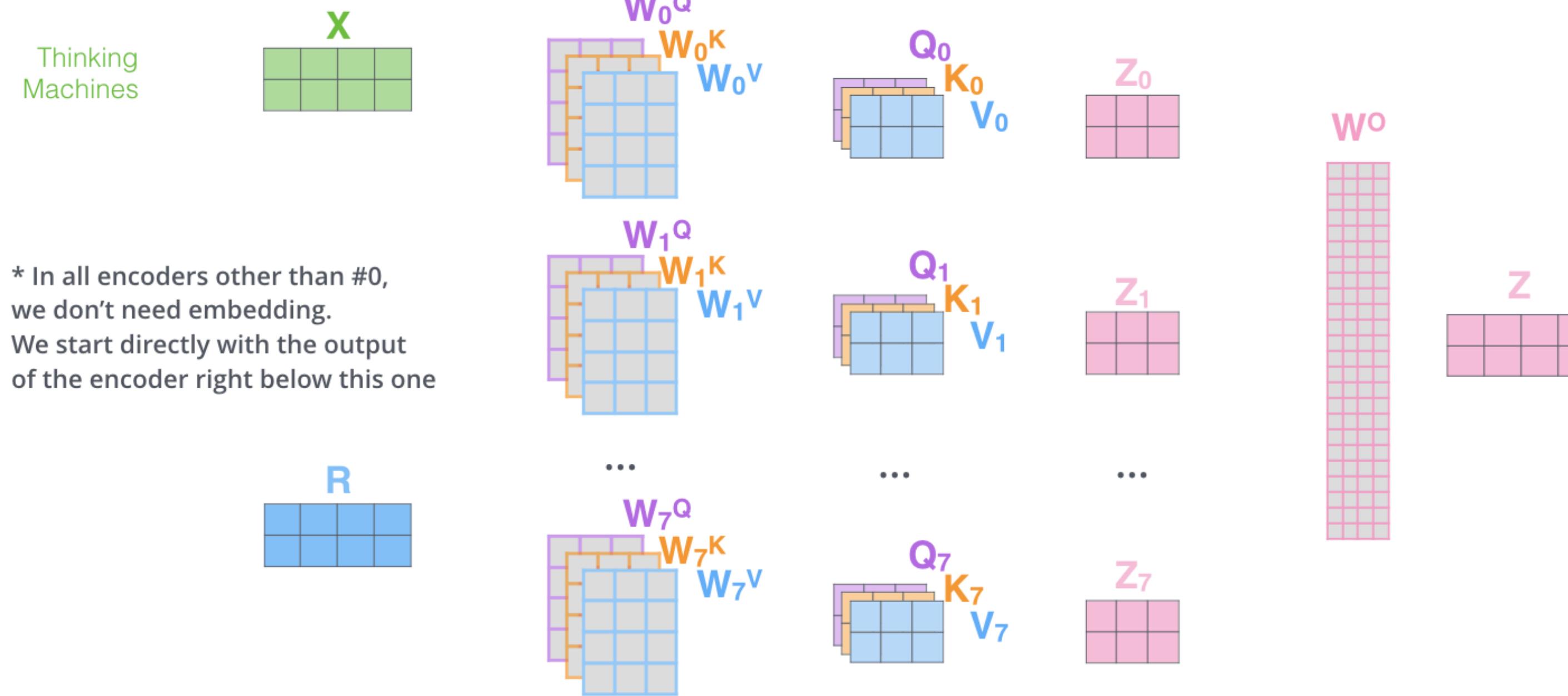


Multihead Self-Attention

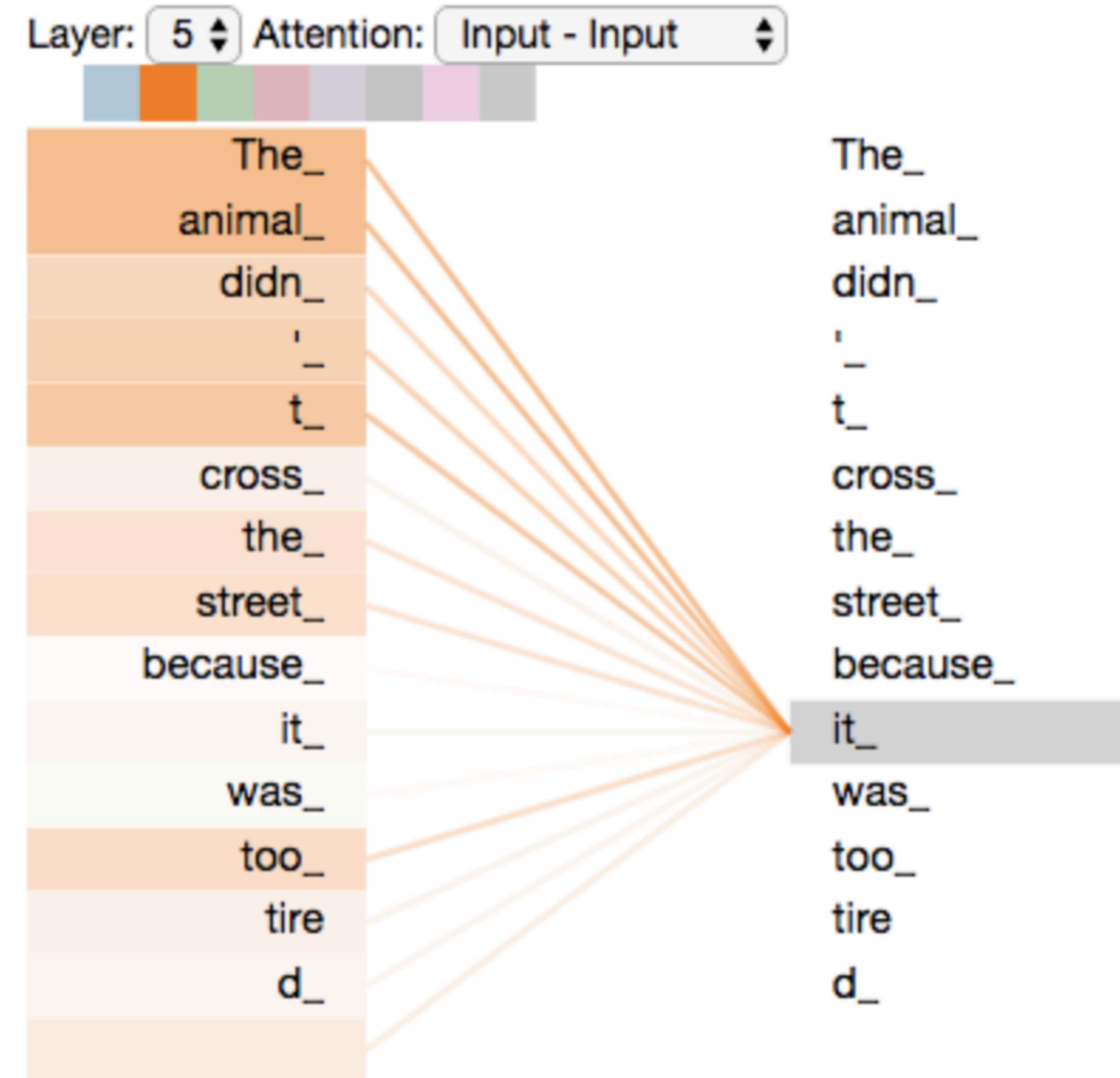


Multihead Self-Attention

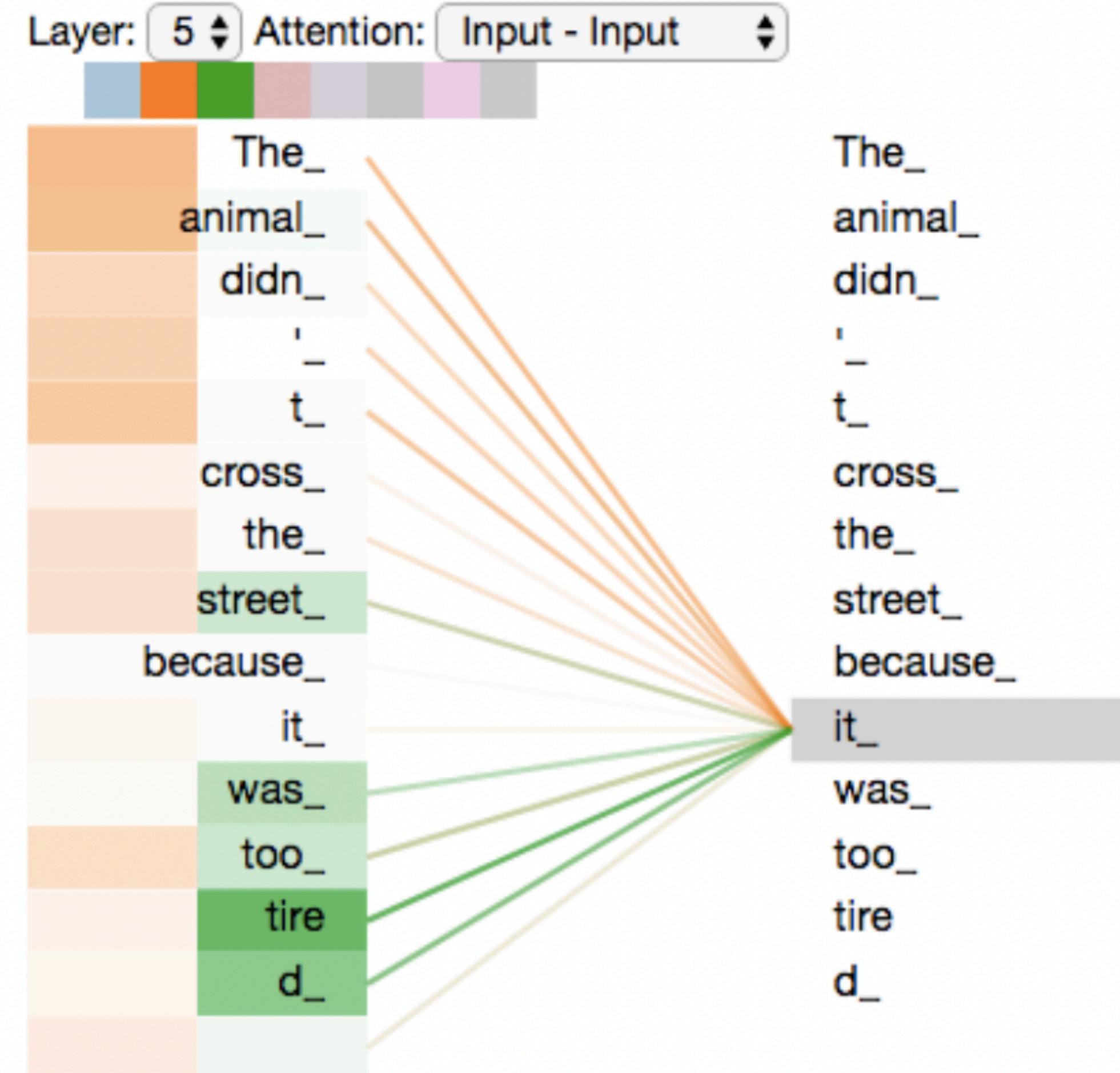
- 1) This is our input sentence*
- 2) We embed each word*
- 3) Split into 8 heads. We multiply X or R with weight matrices
- 4) Calculate attention using the resulting $Q/K/V$ matrices
- 5) Concatenate the resulting Z matrices, then multiply with weight matrix W^o to produce the output of the layer



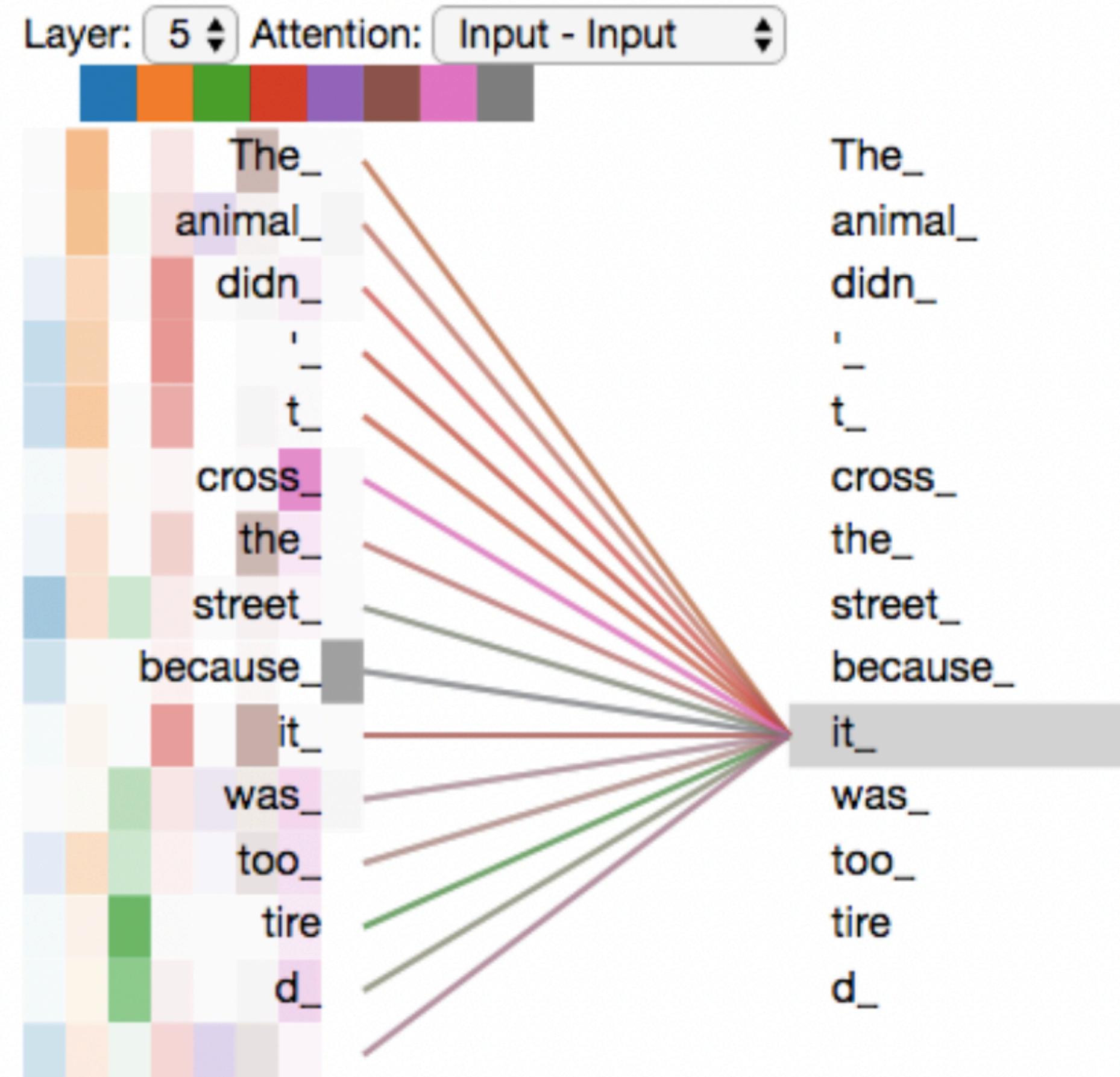
Self-Attention



Self-Attention

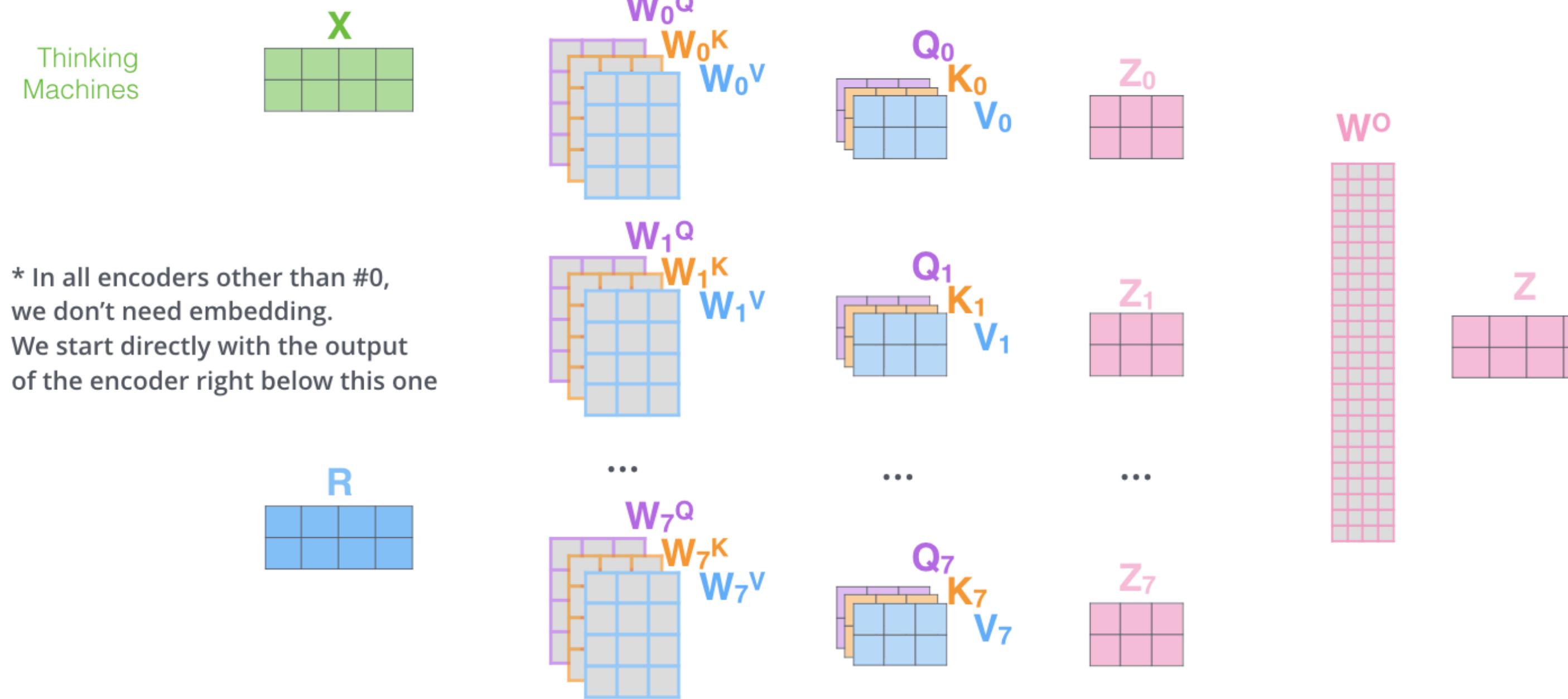


Self-Attention



Multihead Self-Attention

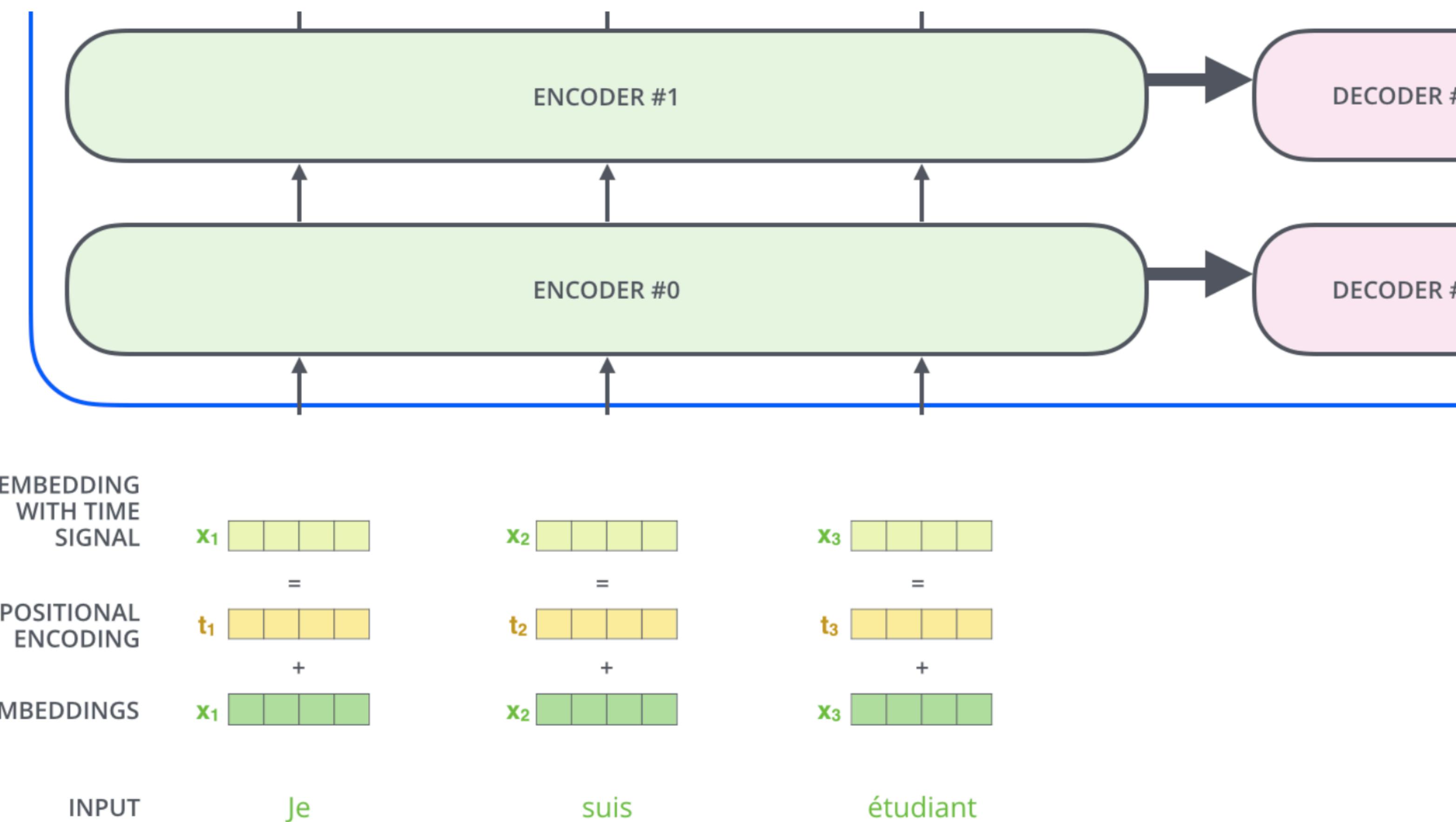
- 1) This is our input sentence*
- 2) We embed each word*
- 3) Split into 8 heads. We multiply X or R with weight matrices
- 4) Calculate attention using the resulting $Q/K/V$ matrices
- 5) Concatenate the resulting Z matrices, then multiply with weight matrix W^o to produce the output of the layer



Но ведь нам нужно обрабатывать последовательности...



Positional Encoding



Positional Encoding

$$\vec{p_t}^{(i)} = f(t)^{(i)} = \begin{cases} \sin(\omega_k t), & \text{if } i = 2k \\ \cos(\omega_k t), & \text{if } i = 2k + 1 \end{cases}$$

$$\omega_k = \frac{1}{10000^{2k/d}}$$

Positional Encoding

$$\vec{p}_t^{(i)} = f(t)^{(i)} = \begin{cases} \sin(\omega_k t), & \text{if } i = 2k \\ \cos(\omega_k t), & \text{if } i = 2k + 1 \end{cases}$$

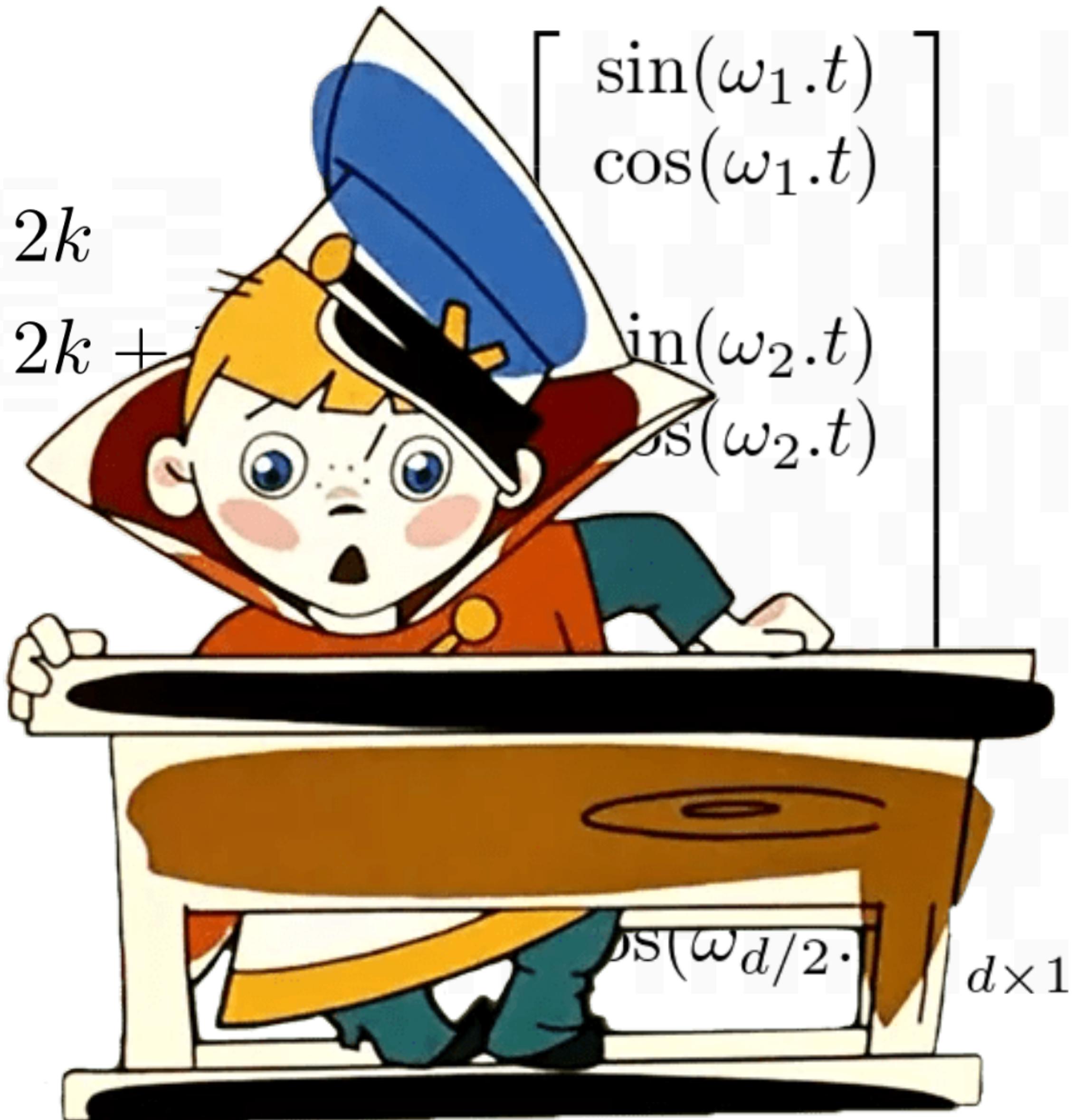
$$\omega_k = \frac{1}{10000^{2k/d}}$$

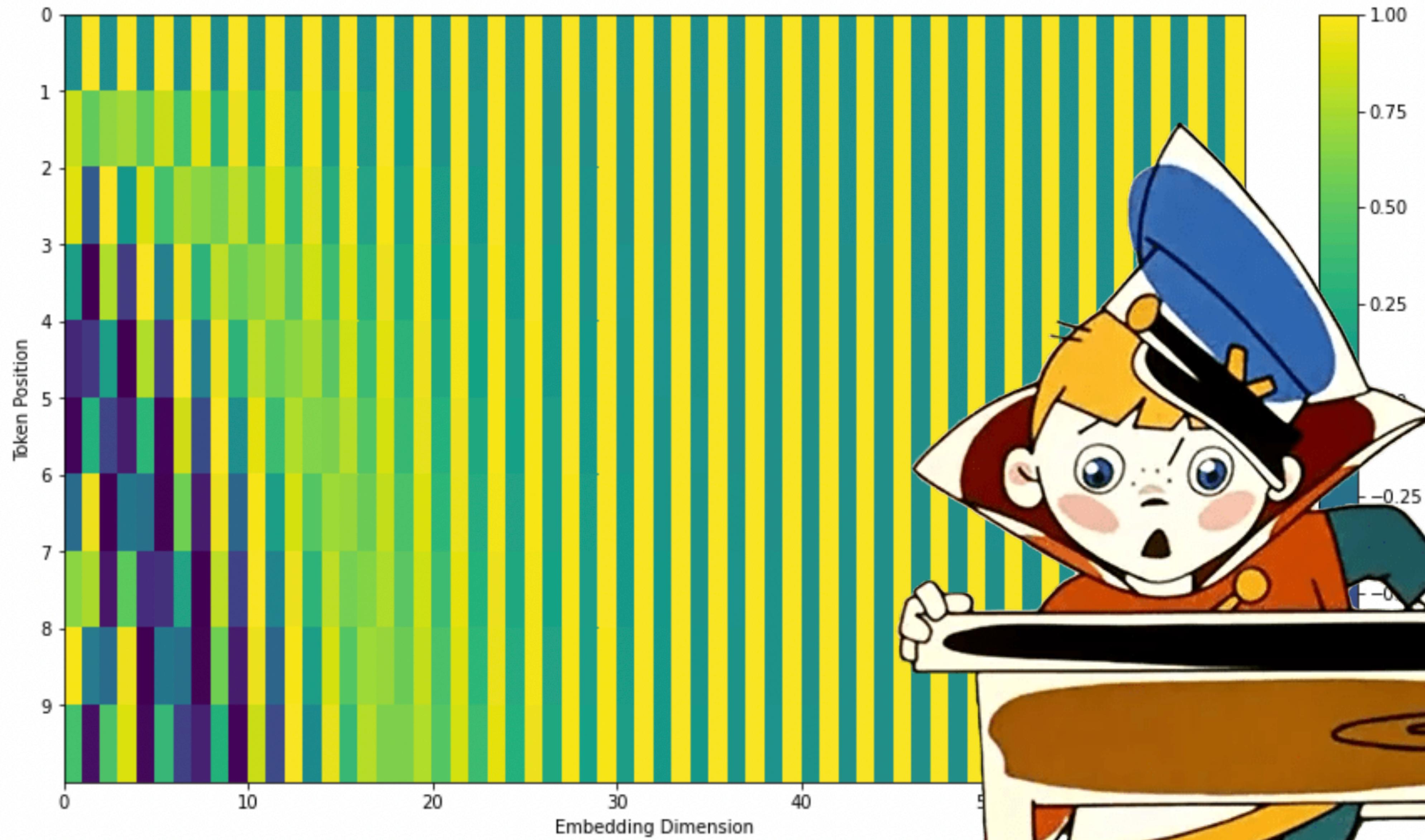
$$\vec{p}_t = \begin{bmatrix} \sin(\omega_1 \cdot t) \\ \cos(\omega_1 \cdot t) \\ \vdots \\ \sin(\omega_{d/2} \cdot t) \\ \cos(\omega_{d/2} \cdot t) \end{bmatrix}_{d \times 1}$$

Positional Encoding

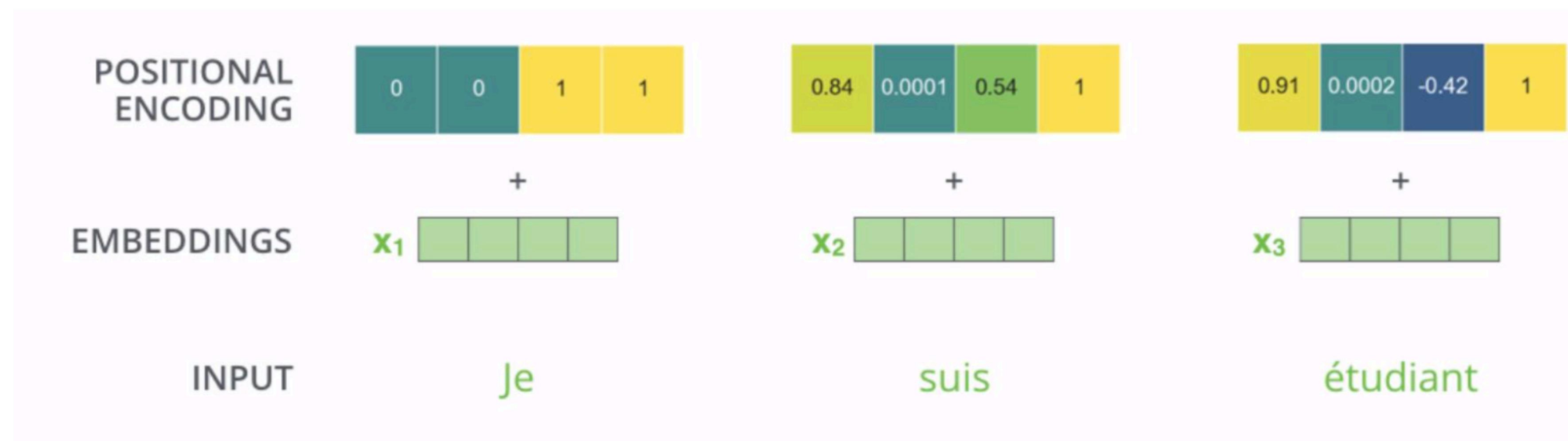
$$\vec{p}_t^{(i)} = f(t)^{(i)} = \begin{cases} \sin(\omega_k t), & \text{if } i = 2k \\ \cos(\omega_k t), & \text{if } i = 2k + 1 \end{cases}$$

$$\omega_k = \frac{1}{10000^{2k/d}}$$





Positional Encoding



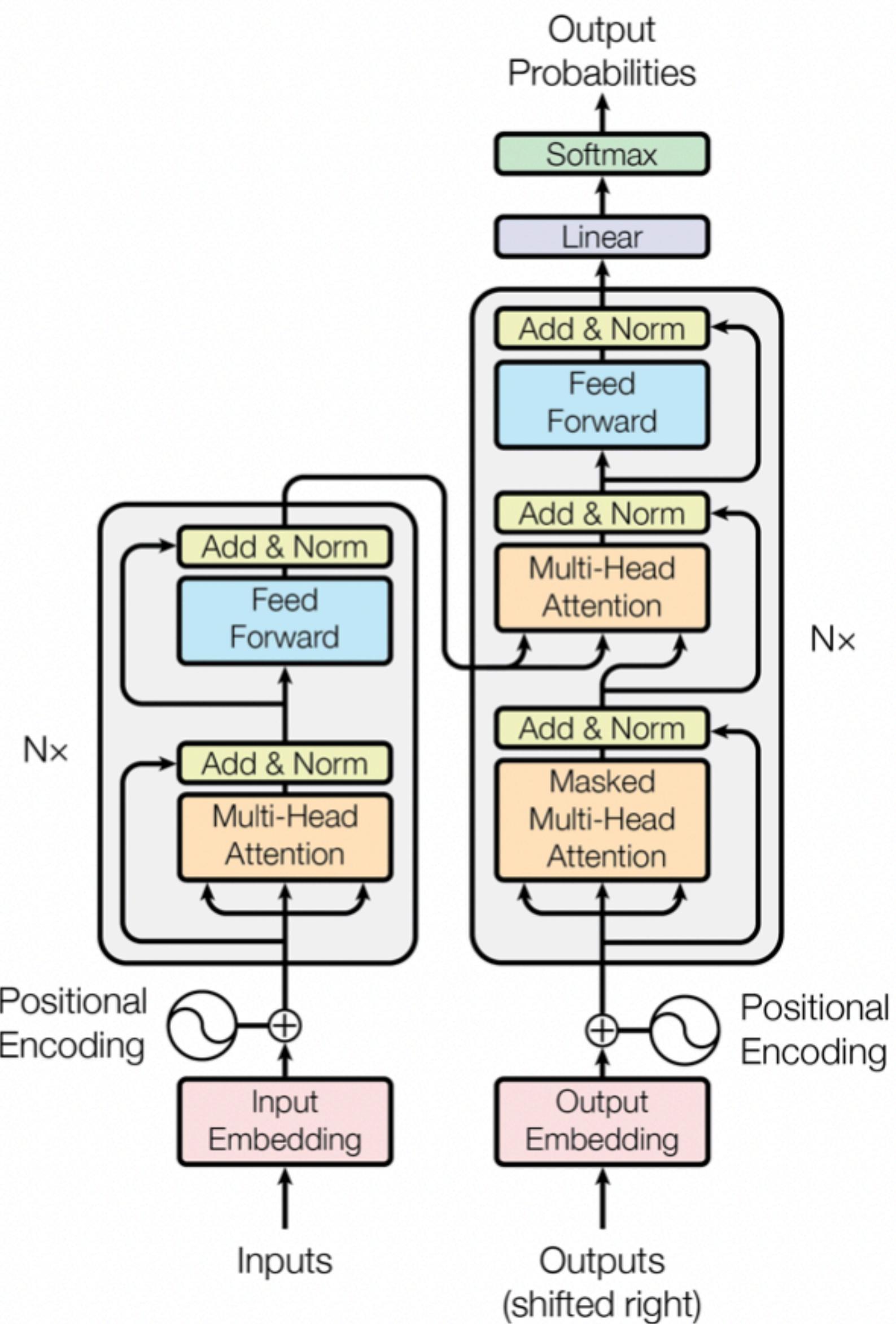
Positional Encoding

We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k , PE_{pos+k} can be represented as a linear function of PE_{pos} .

$$M \begin{bmatrix} \sin(\omega_k t) \\ \cos(\omega_k t) \end{bmatrix} = \begin{bmatrix} \sin(\omega_k(t + \phi)) \\ \cos(\omega_k(t + \phi)) \end{bmatrix}$$

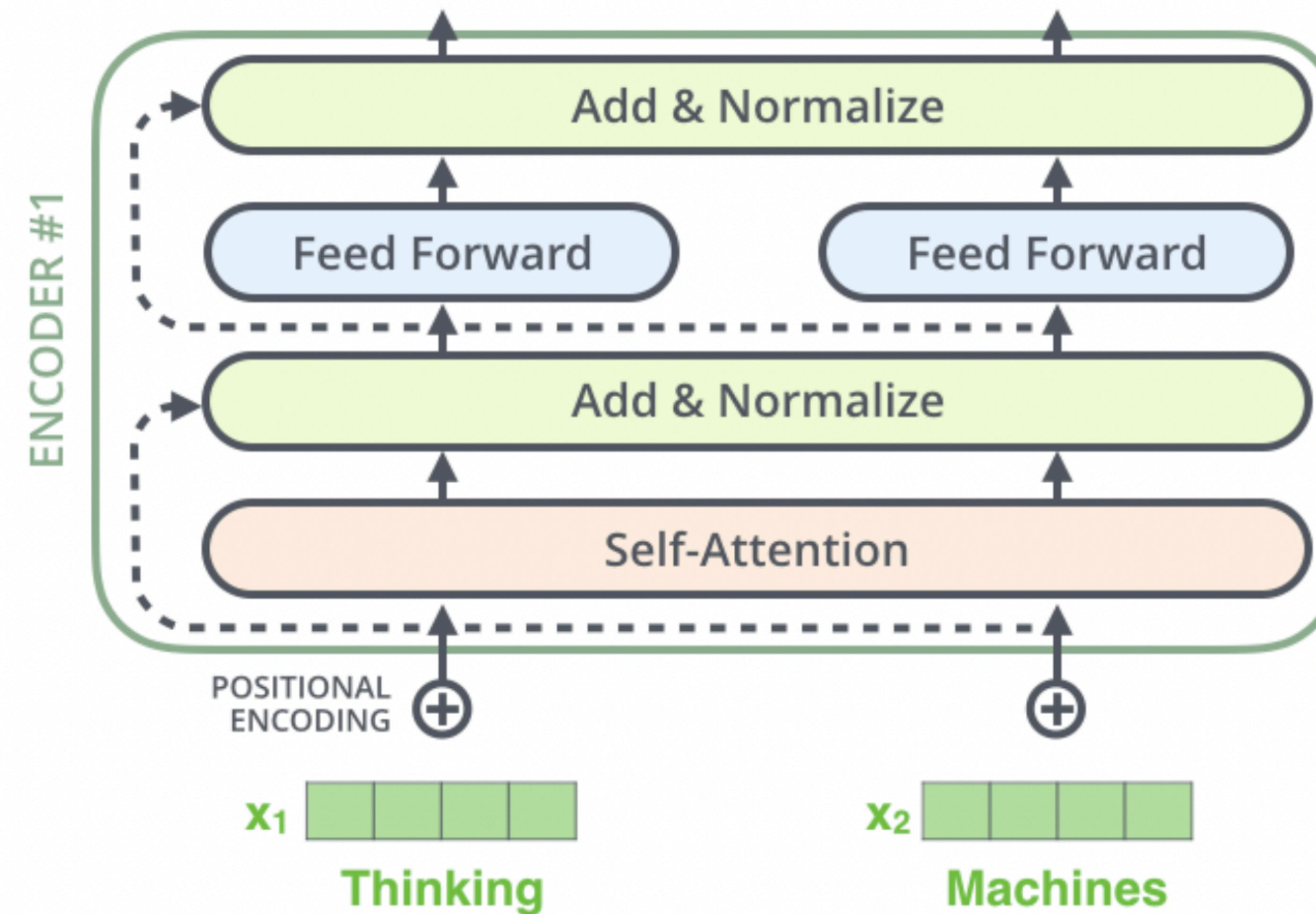
The Residuals & layer-normalization

The Residuals & layer-normalization

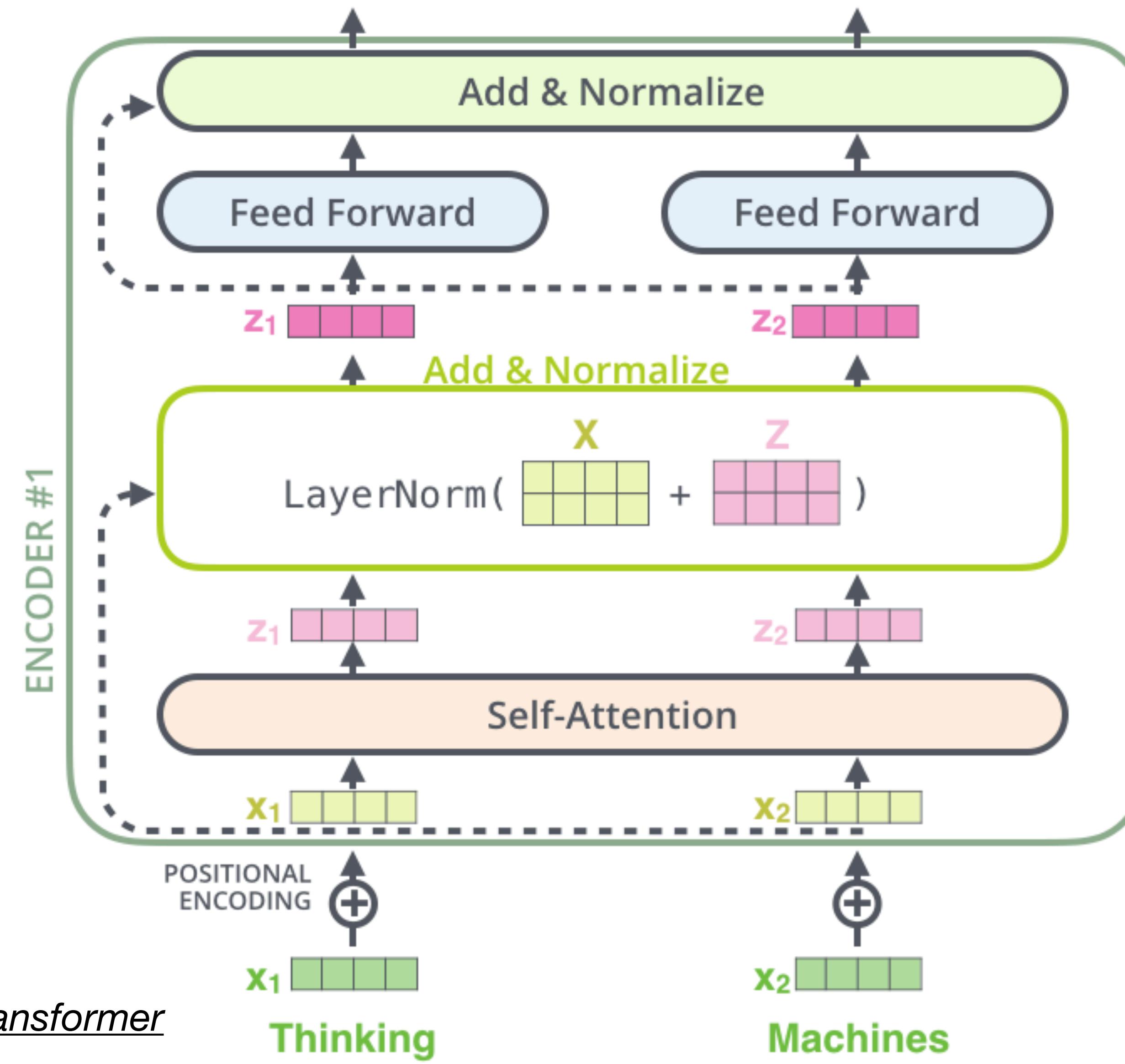


Attention Is All You Need

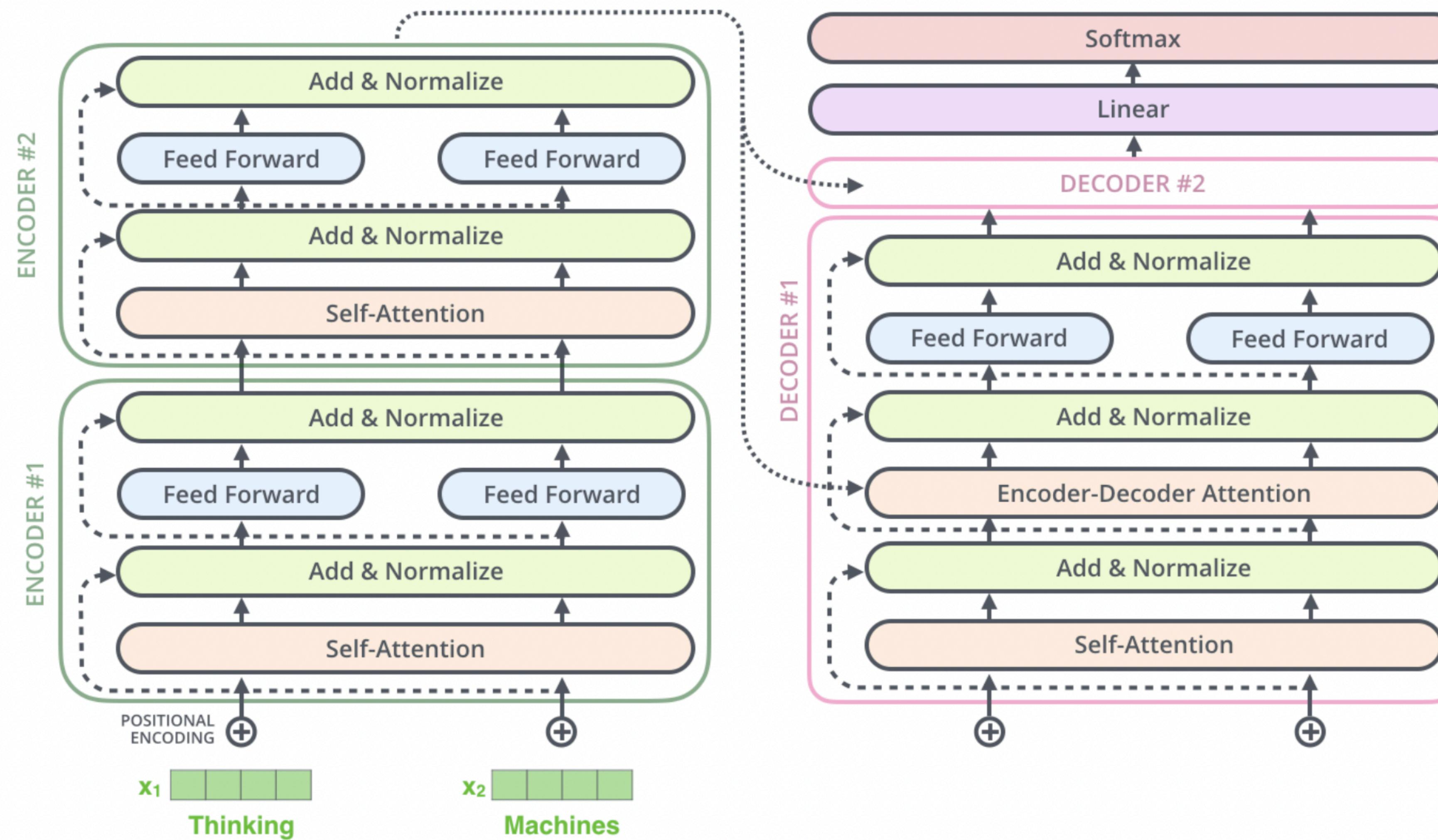
The Residuals & layer-normalization



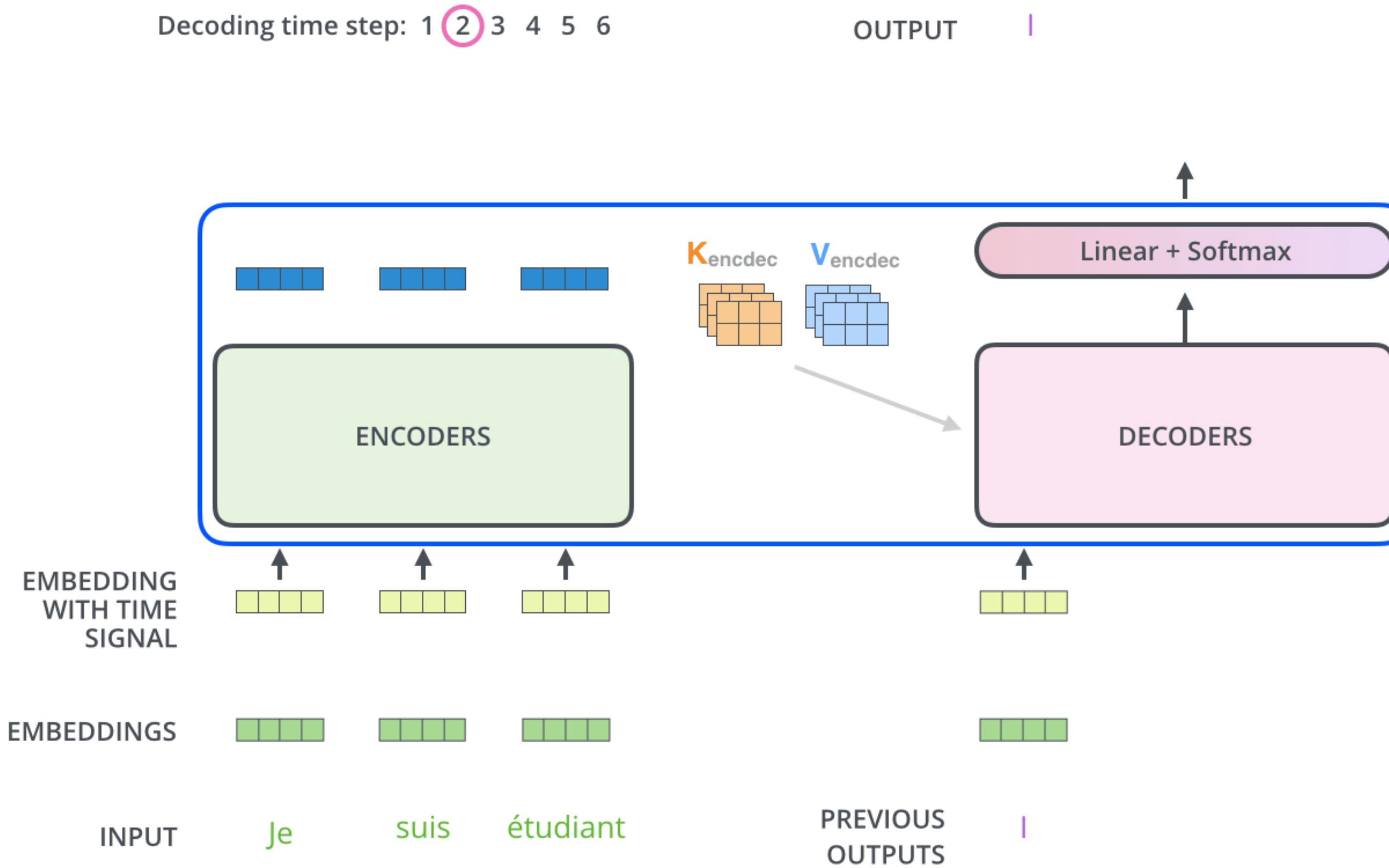
The Residuals & layer-normalization

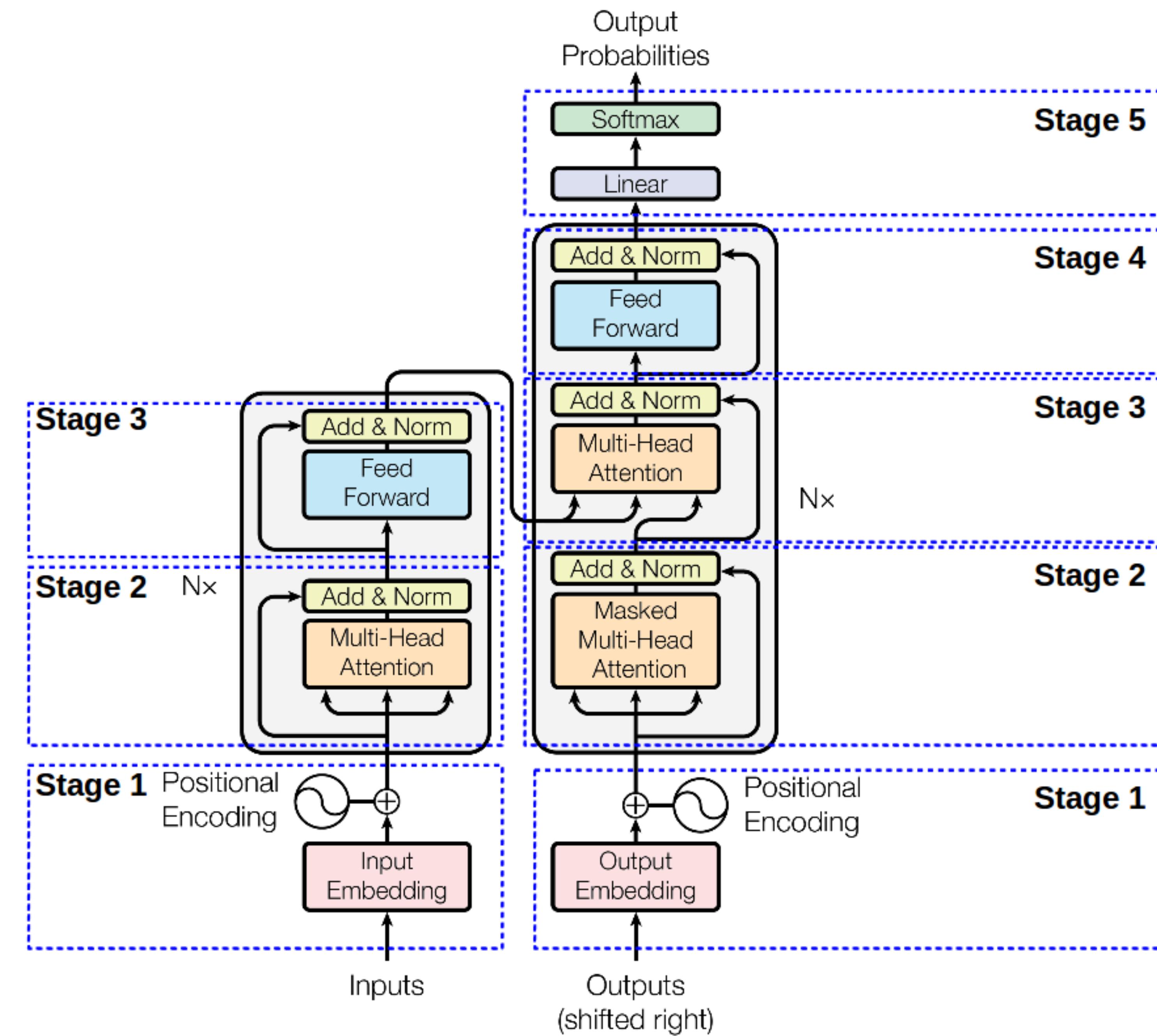


Transformer: Decoder Side

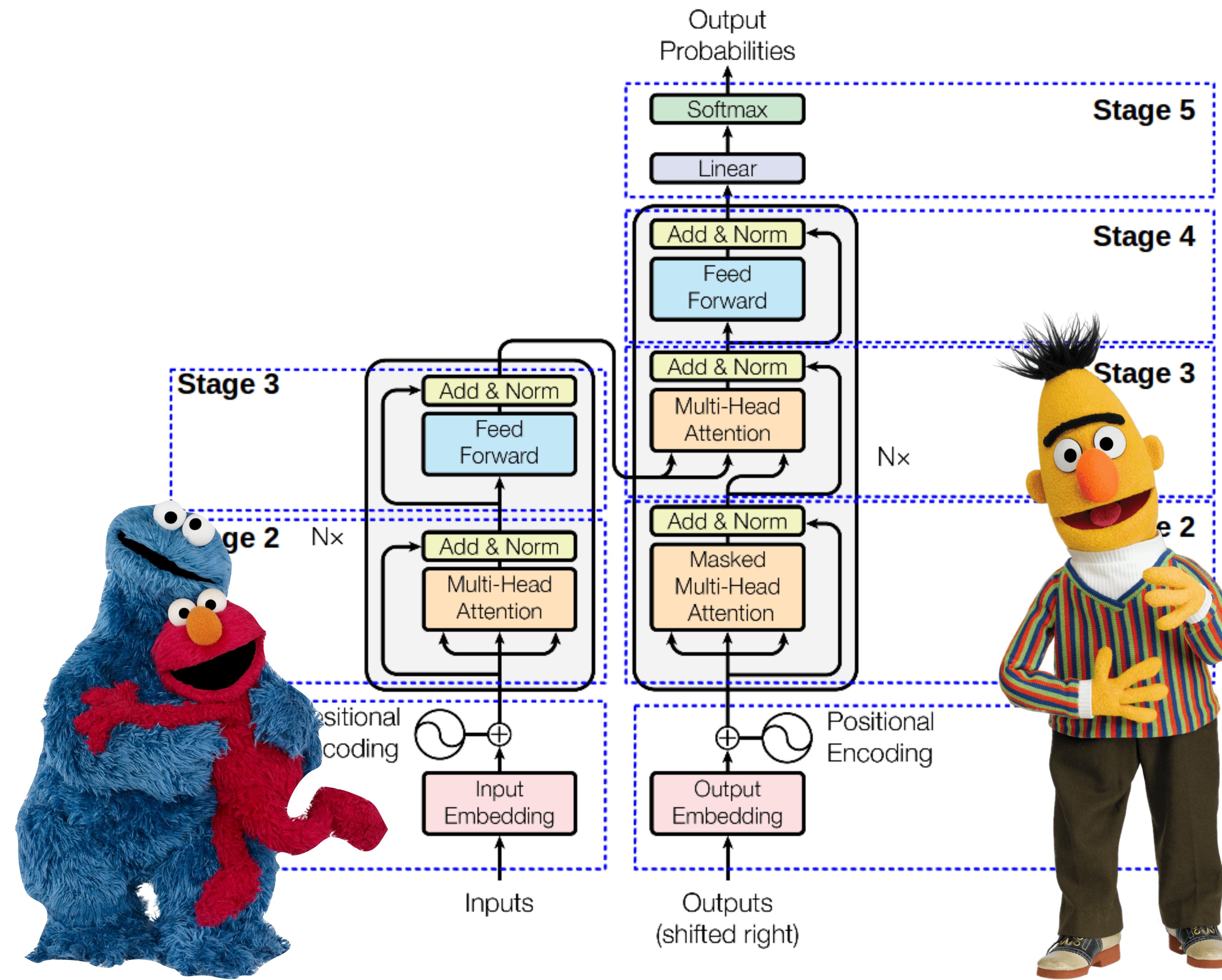


Transformer: Decoder Side





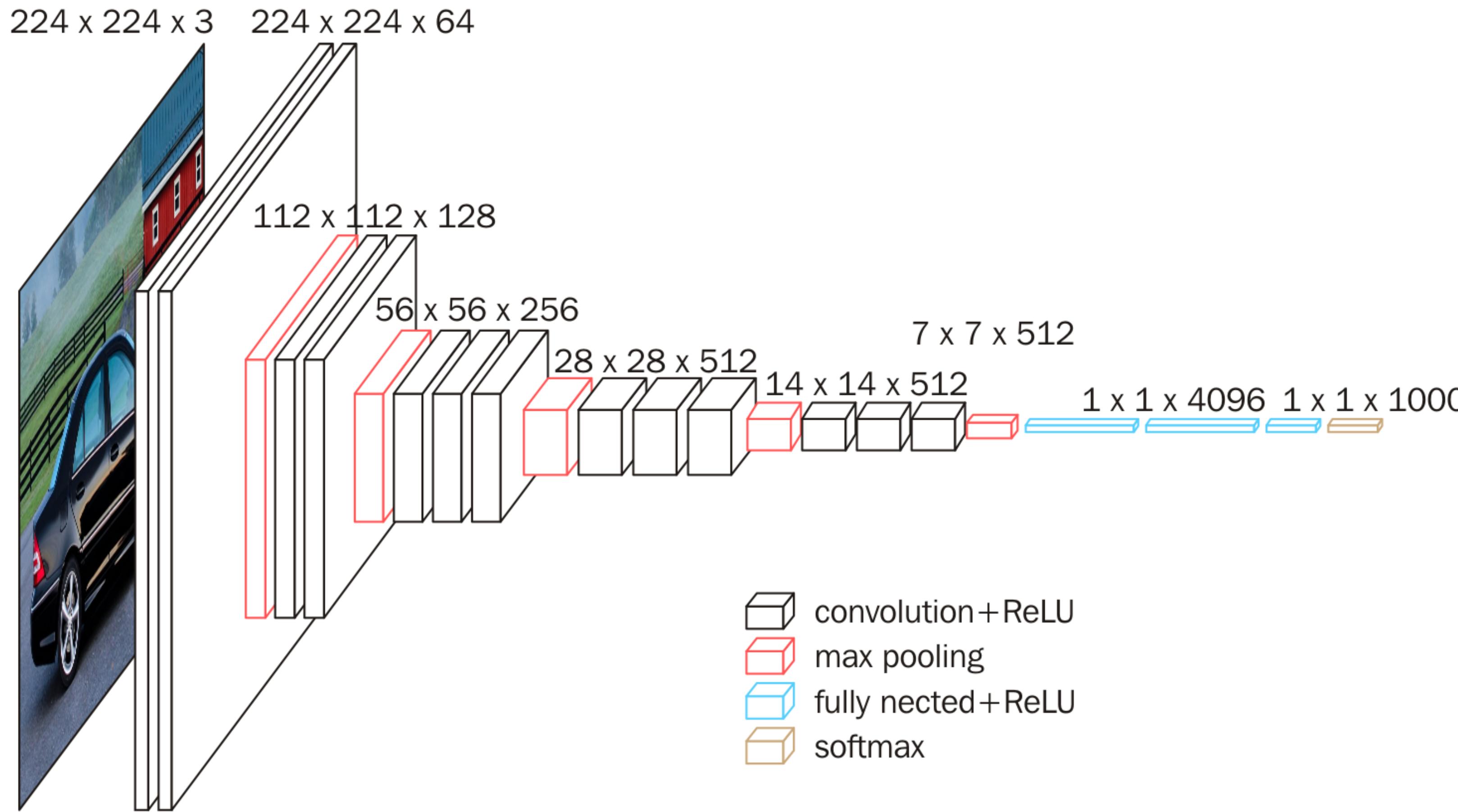




Другие модели

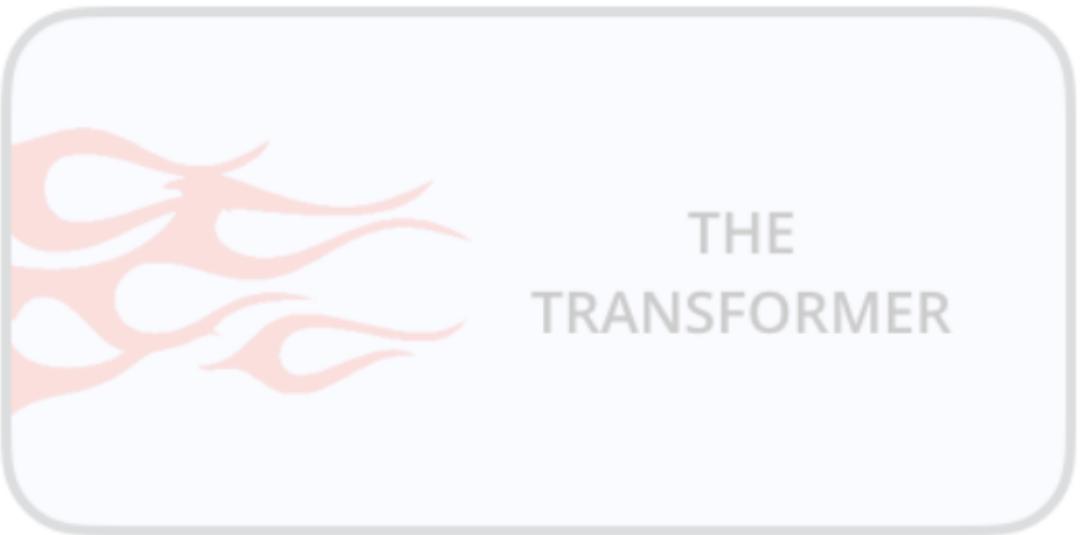


Recap: Applying transfer learning in CV

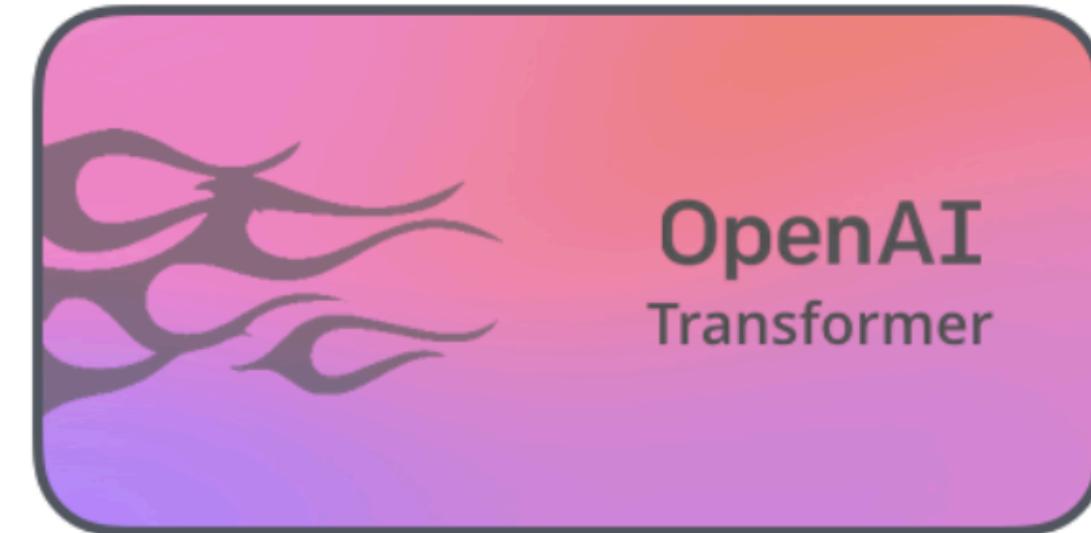


Transfer learning in NLP?

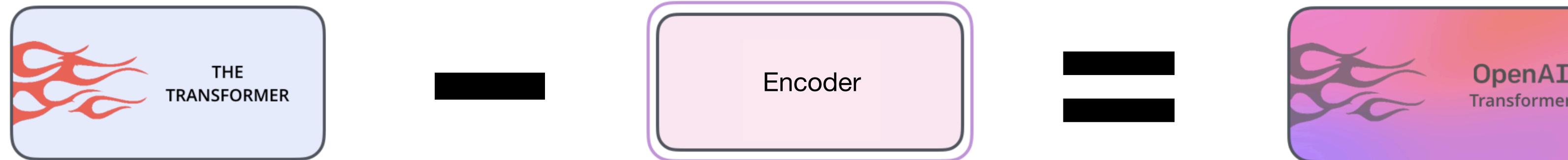
Другие модели



OpenAI Transformer

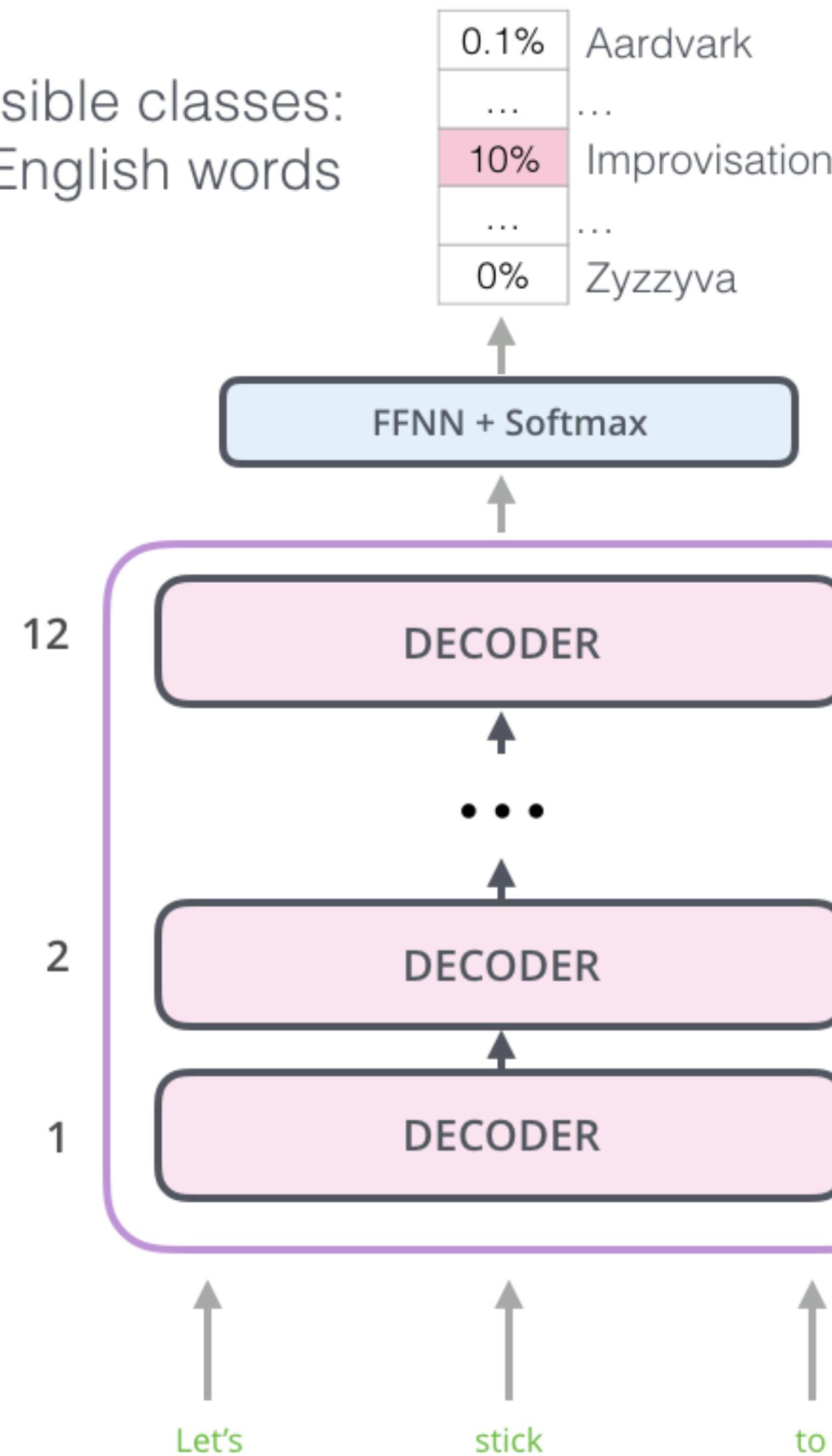


Pre-training a Transformer Decoder for
Language Modeling

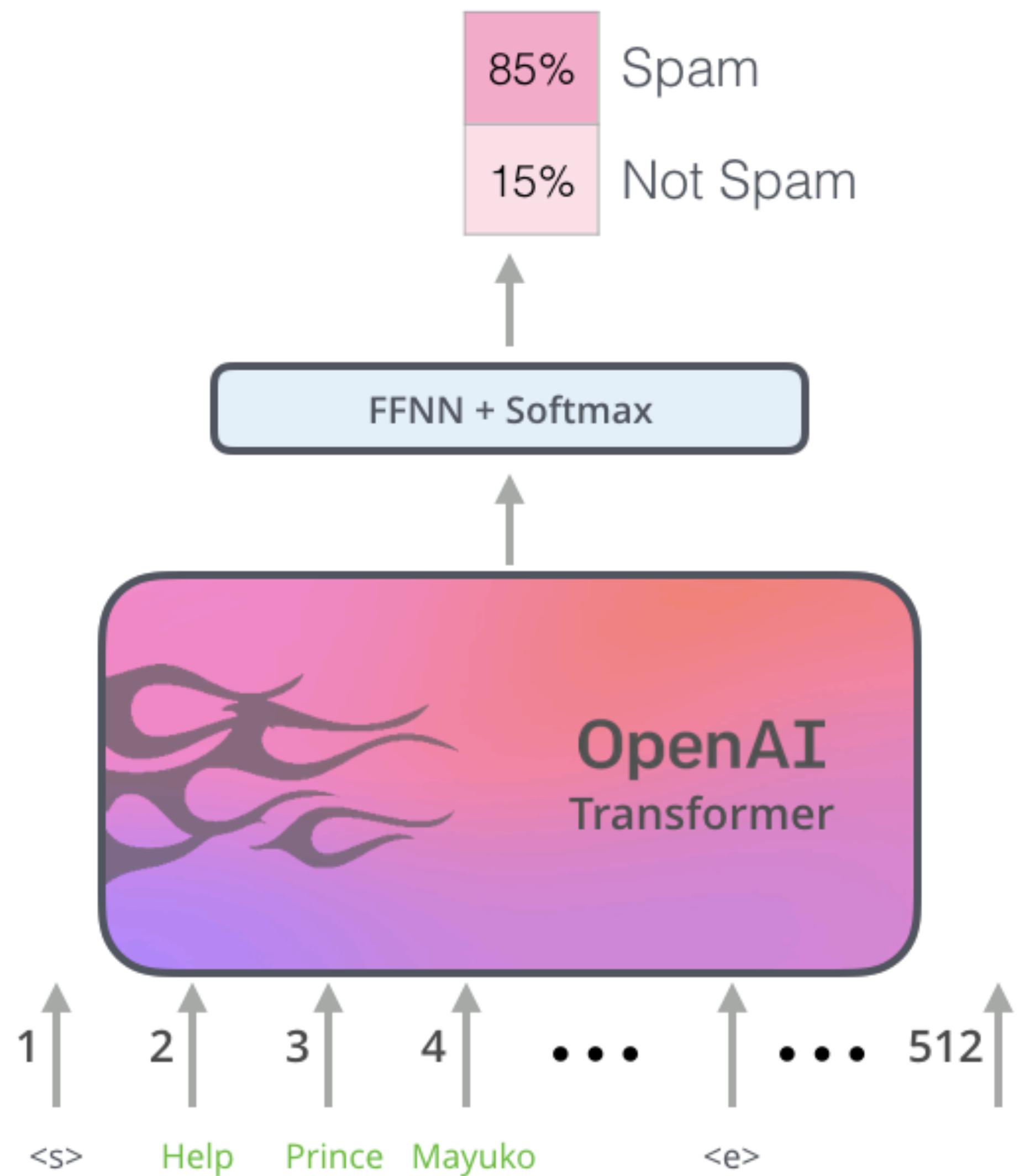


OpenAI Transformer

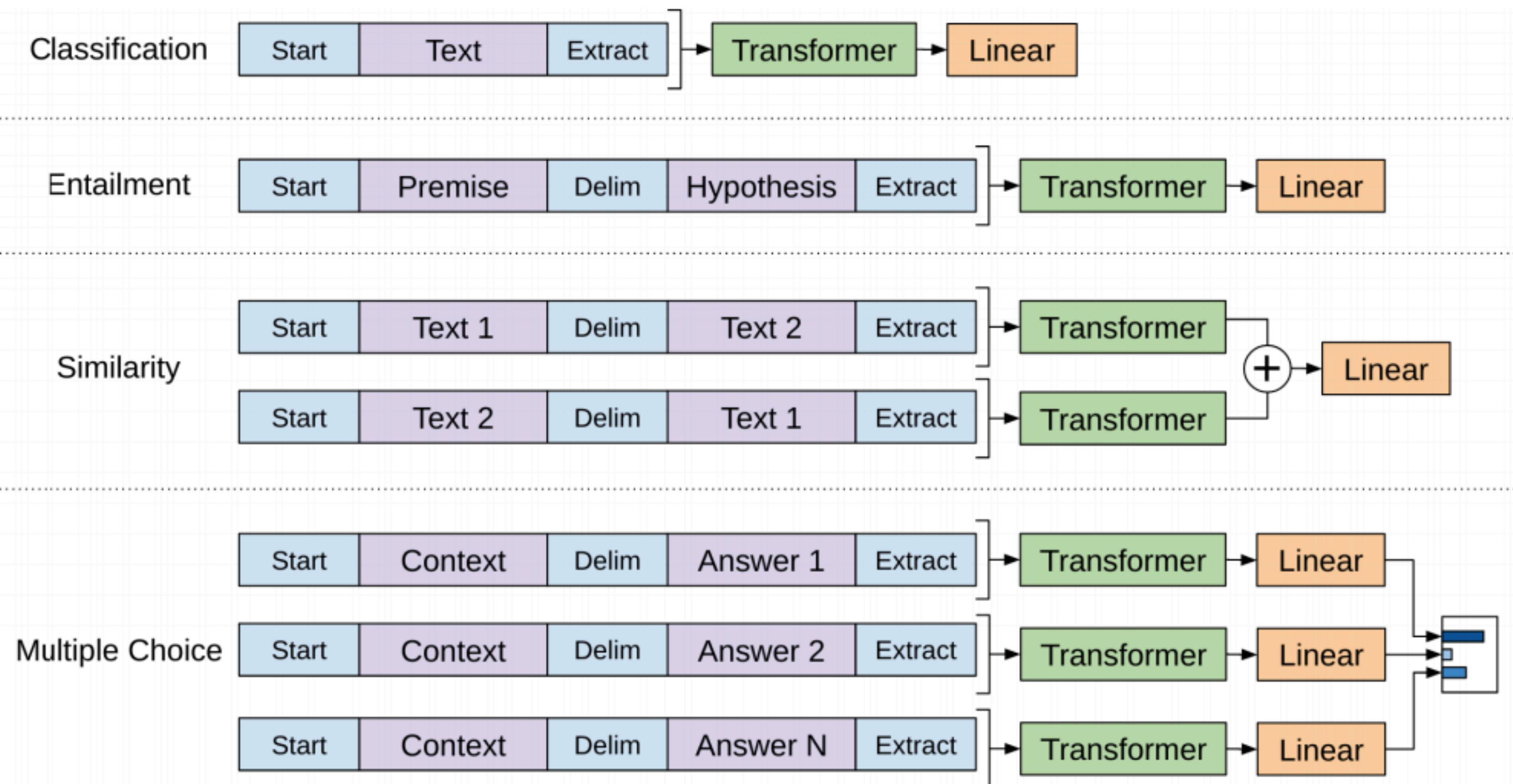
Possible classes:
All English words



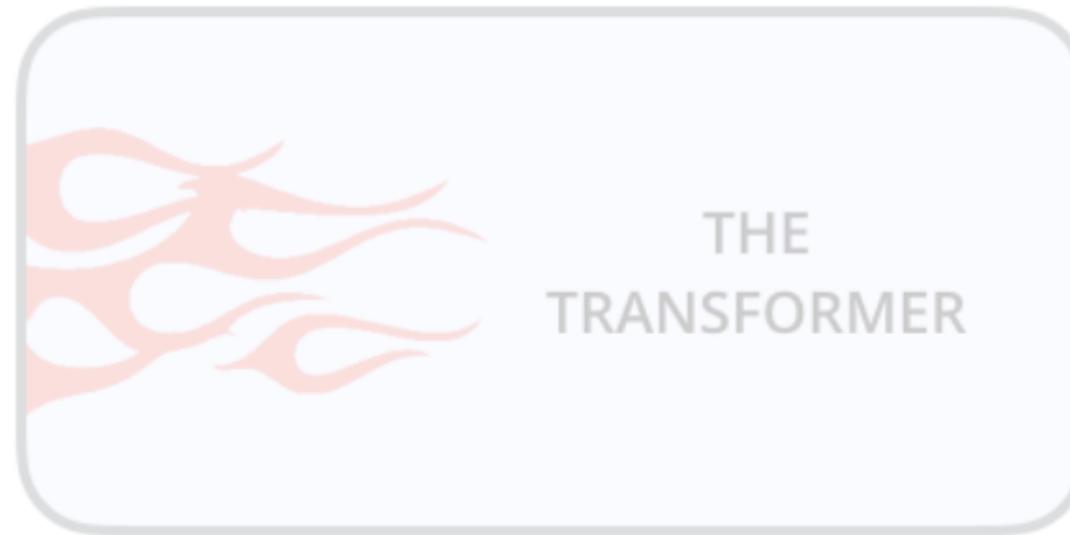
OpenAI Transformer



OpenAI Transformer



План лекции



ELMo



1. Expedited Labour Market Opinion
2. Electric Light Machine Organization
3. Enough Let's Move On

Special thx for @Anastasia Yanina:

https://github.com/ml-mipt/ml-mipt/blob/advanced/week05_BERT_and_LDA/Lecture_BERT_DIHT.pdf

ELMo



1. Expedited Labour Market Opinion
2. Electric Light Machine Organization
3. Enough Let's Move On

4. Embeddings from Language Models

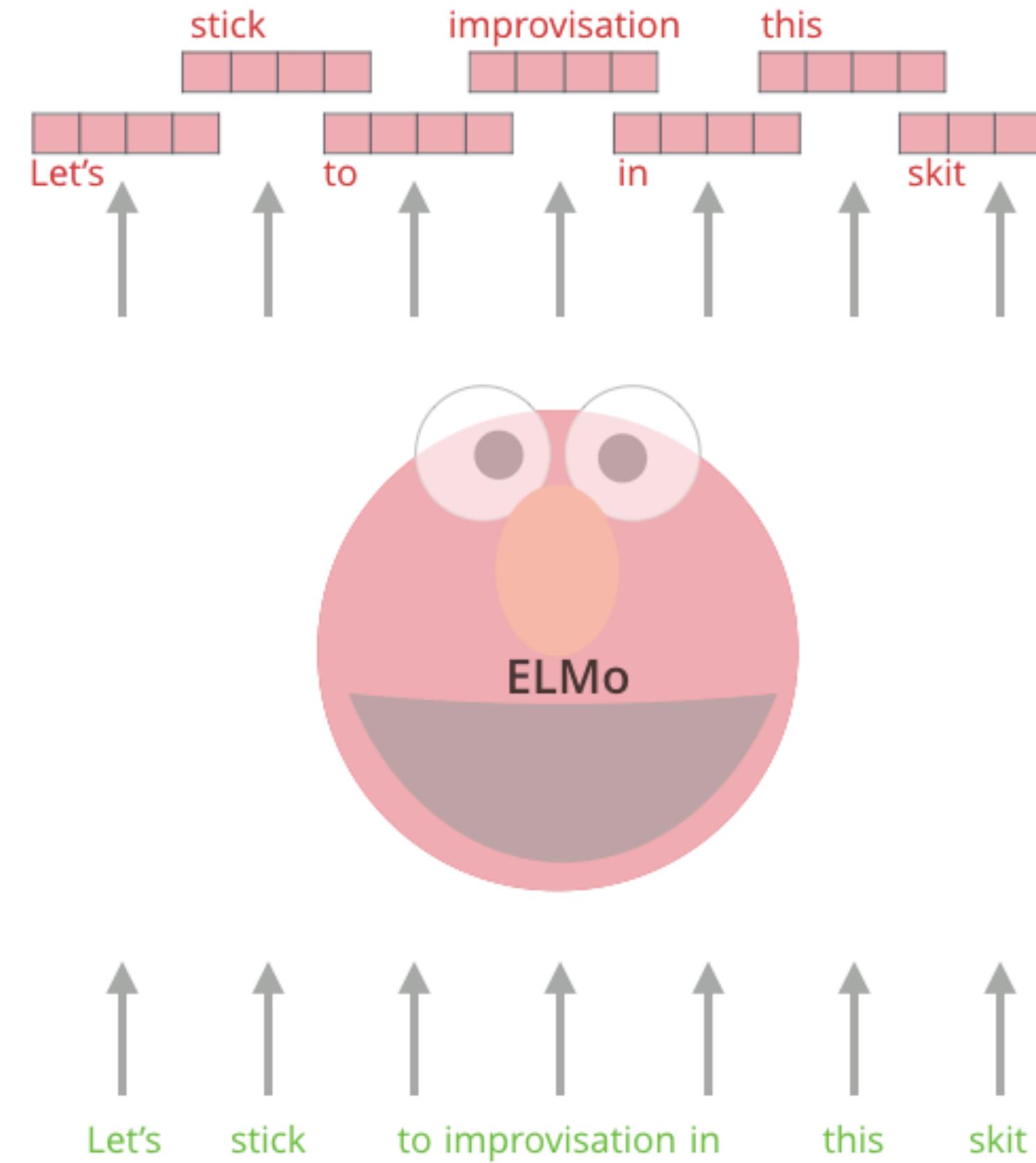
Special thx for @Anastasia Yanina:

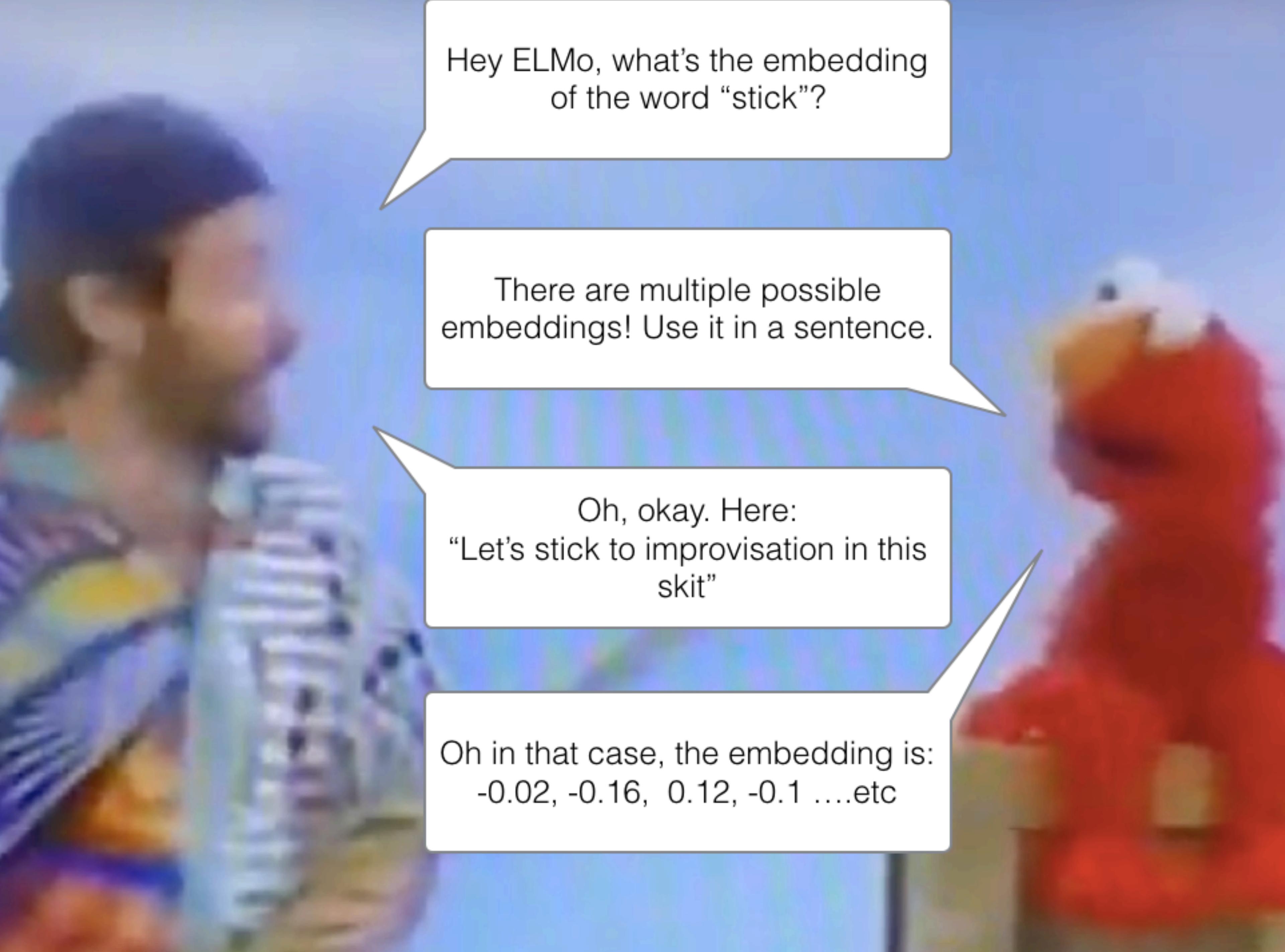
https://github.com/ml-mipt/ml-mipt/blob/advanced/week05_BERT_and_LDA/Lecture_BERT_DIHT.pdf

ELMo

ELMo
Embeddings

Words to embed





Hey ELMo, what's the embedding
of the word "stick"?

There are multiple possible
embeddings! Use it in a sentence.

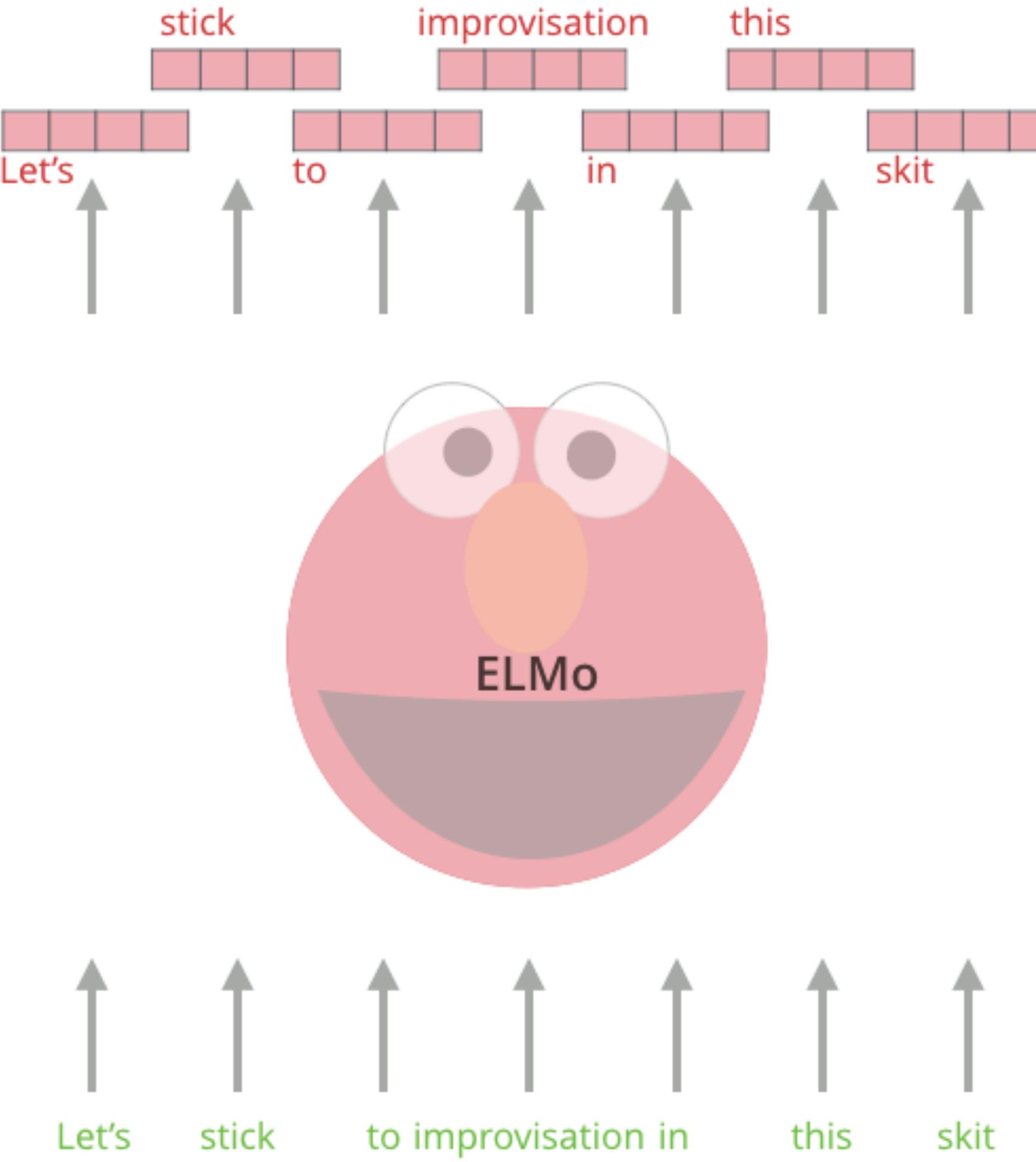
Oh, okay. Here:
"Let's stick to improvisation in this
skit"

Oh in that case, the embedding is:
-0.02, -0.16, 0.12, -0.1etc

ELMo Embeddings

ELMo

Words to embed



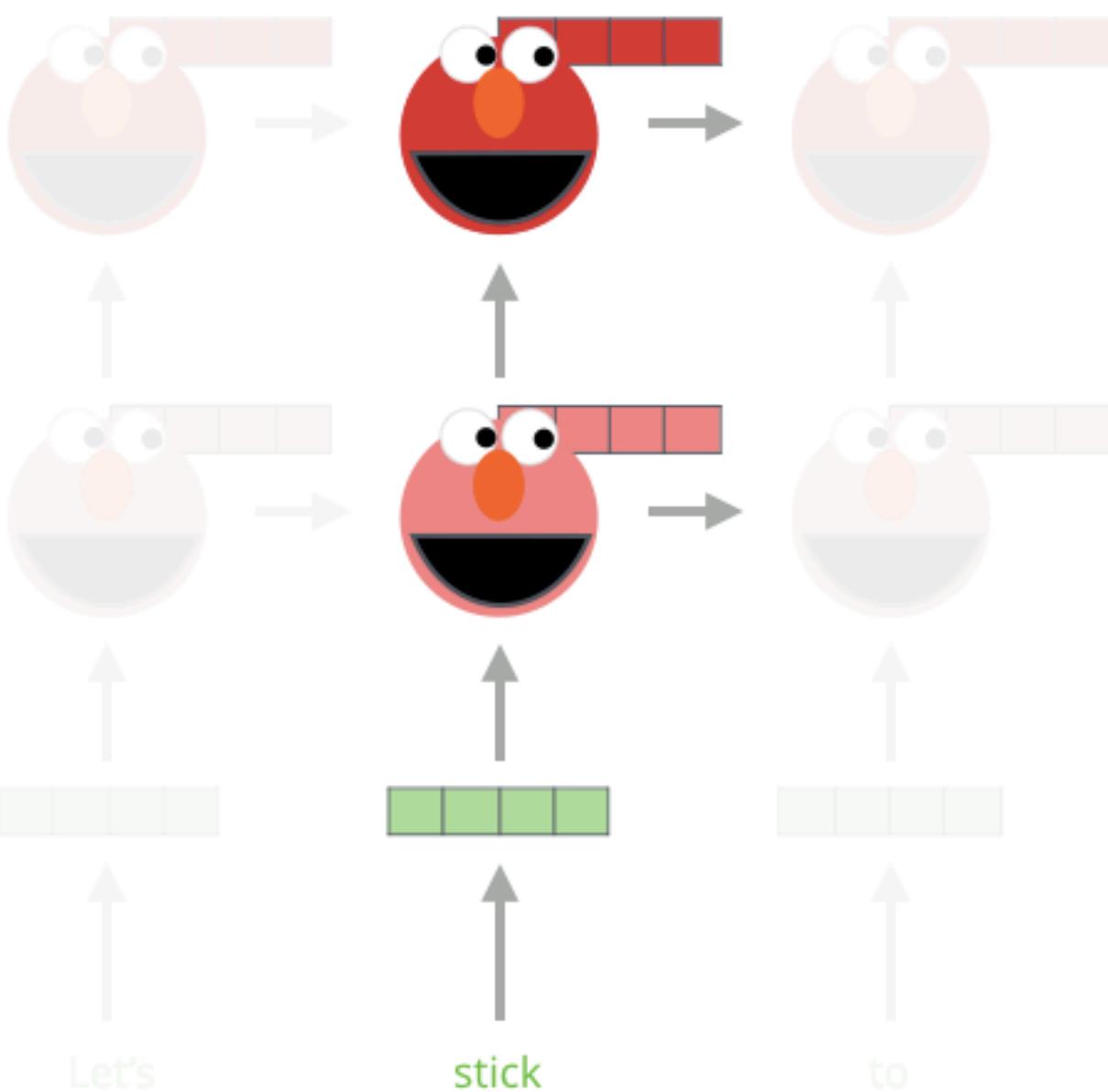
ELMo

Embedding of “stick” in “Let’s stick to” - Step #2

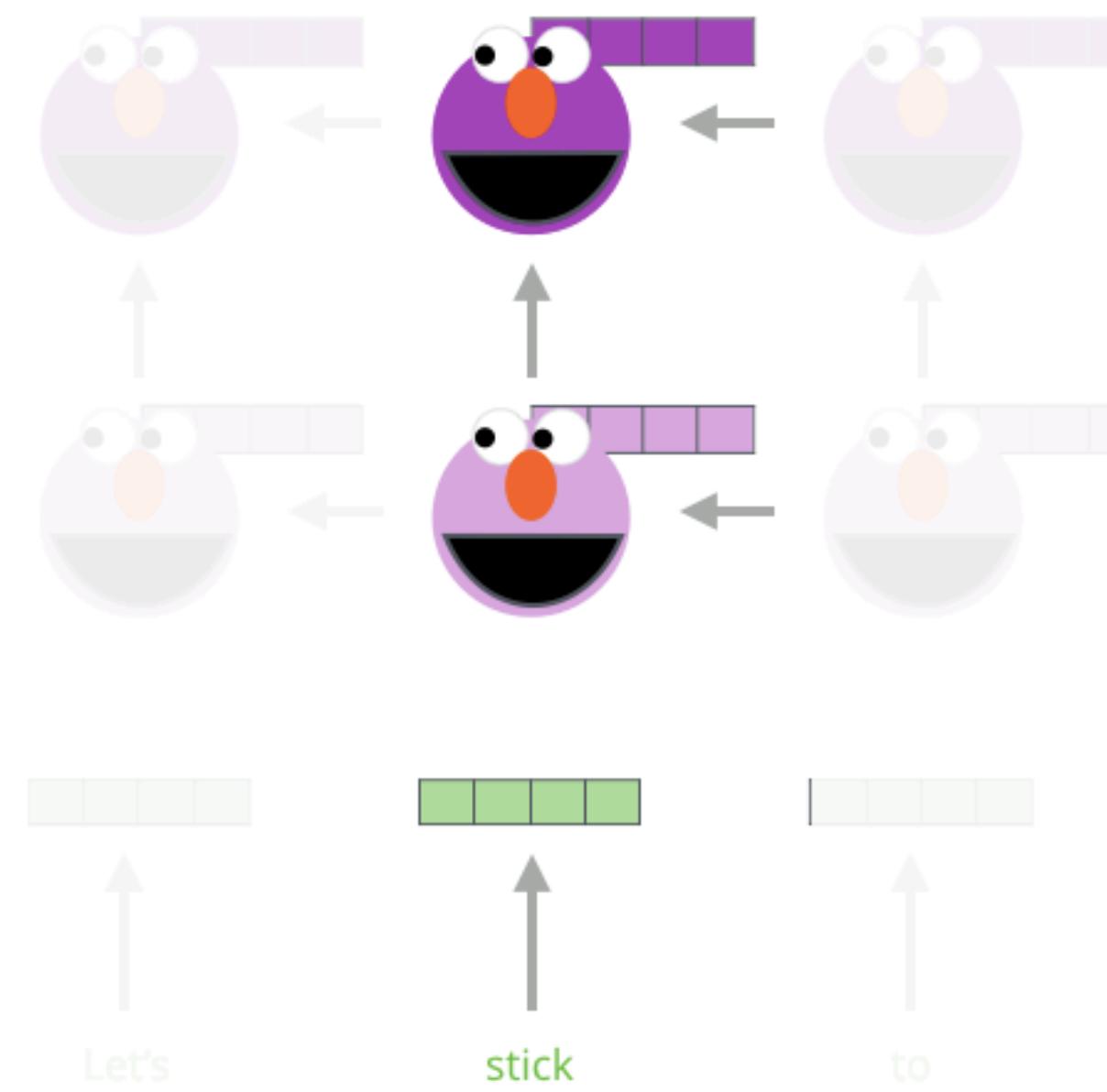
1- Concatenate hidden layers



Forward Language Model



Backward Language Model



2- Multiply each vector by a weight based on the task

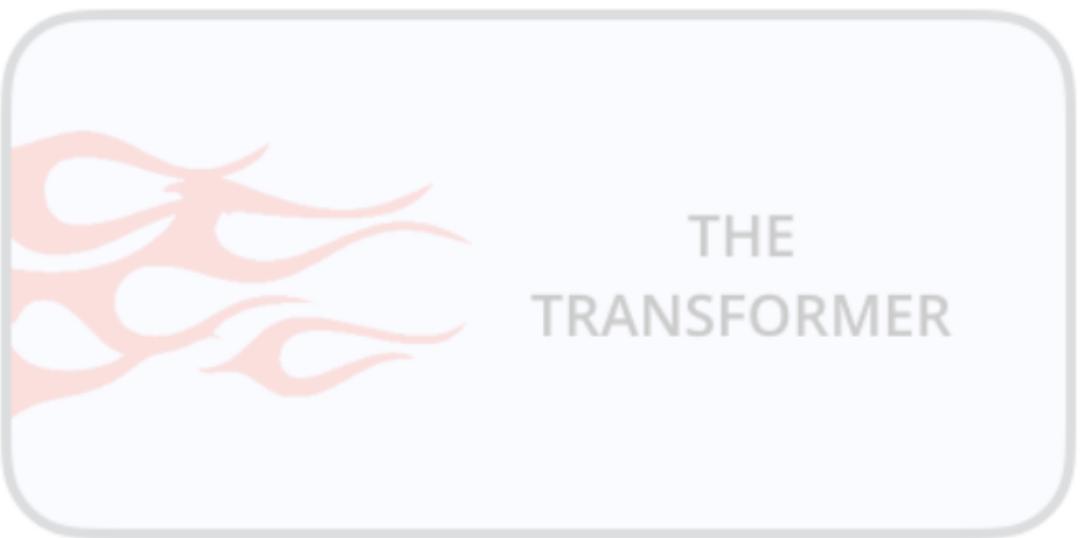
$$\begin{array}{l} \text{red and purple} \times s_2 \\ \text{red and light purple} \times s_1 \\ \text{green} \times s_0 \end{array}$$

3- Sum the (now weighted) vectors



ELMo embedding of “stick” for this task in this context

Другие модели

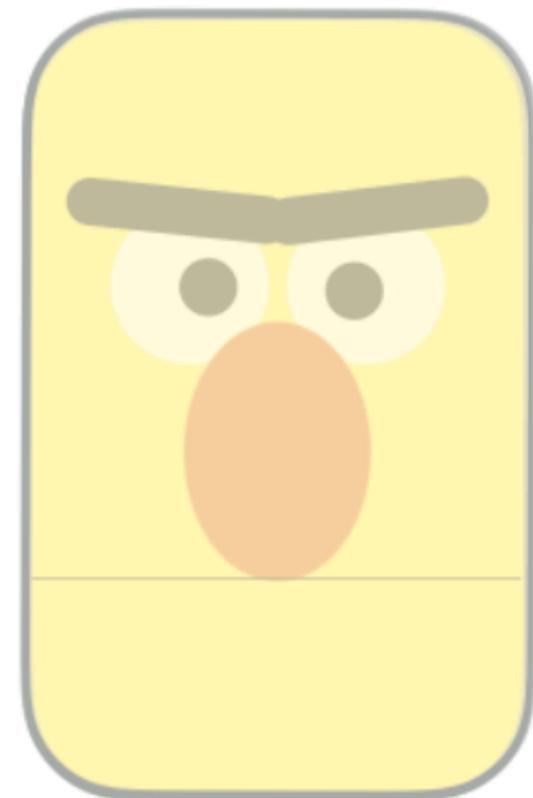


BERT

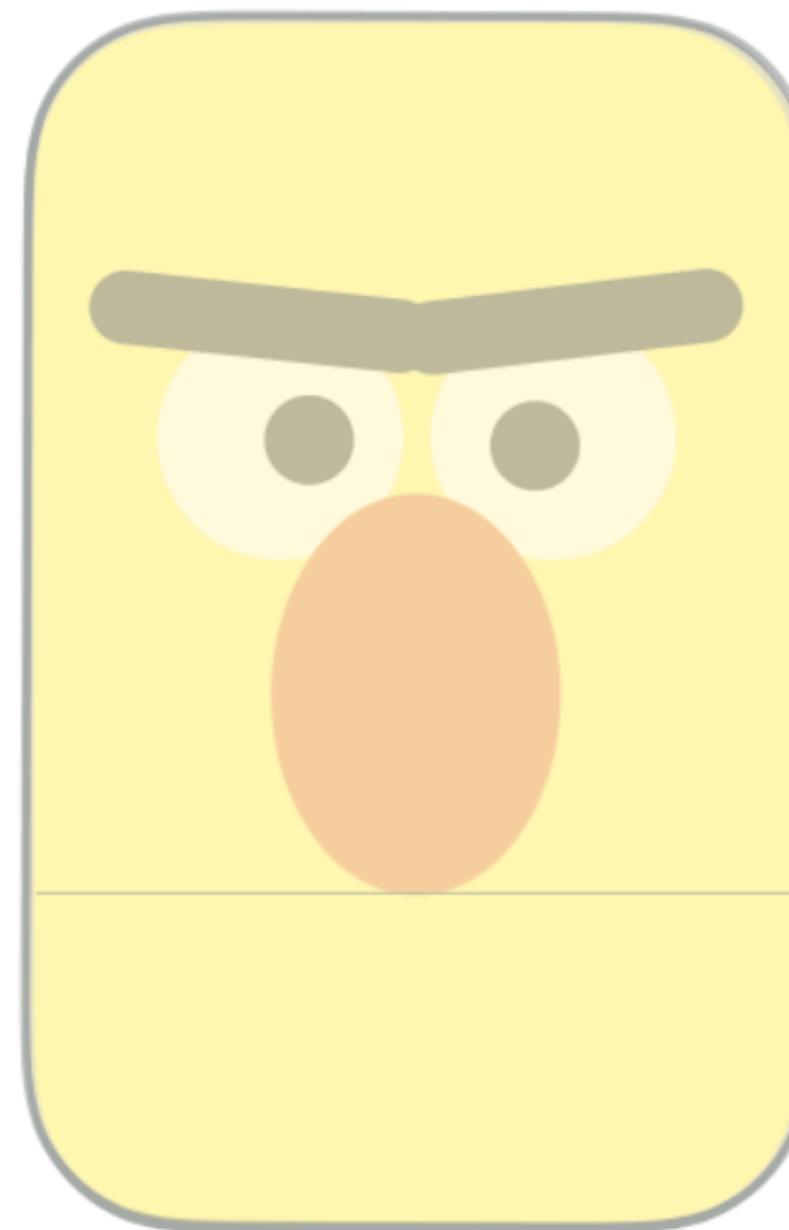


[1810.04805] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT

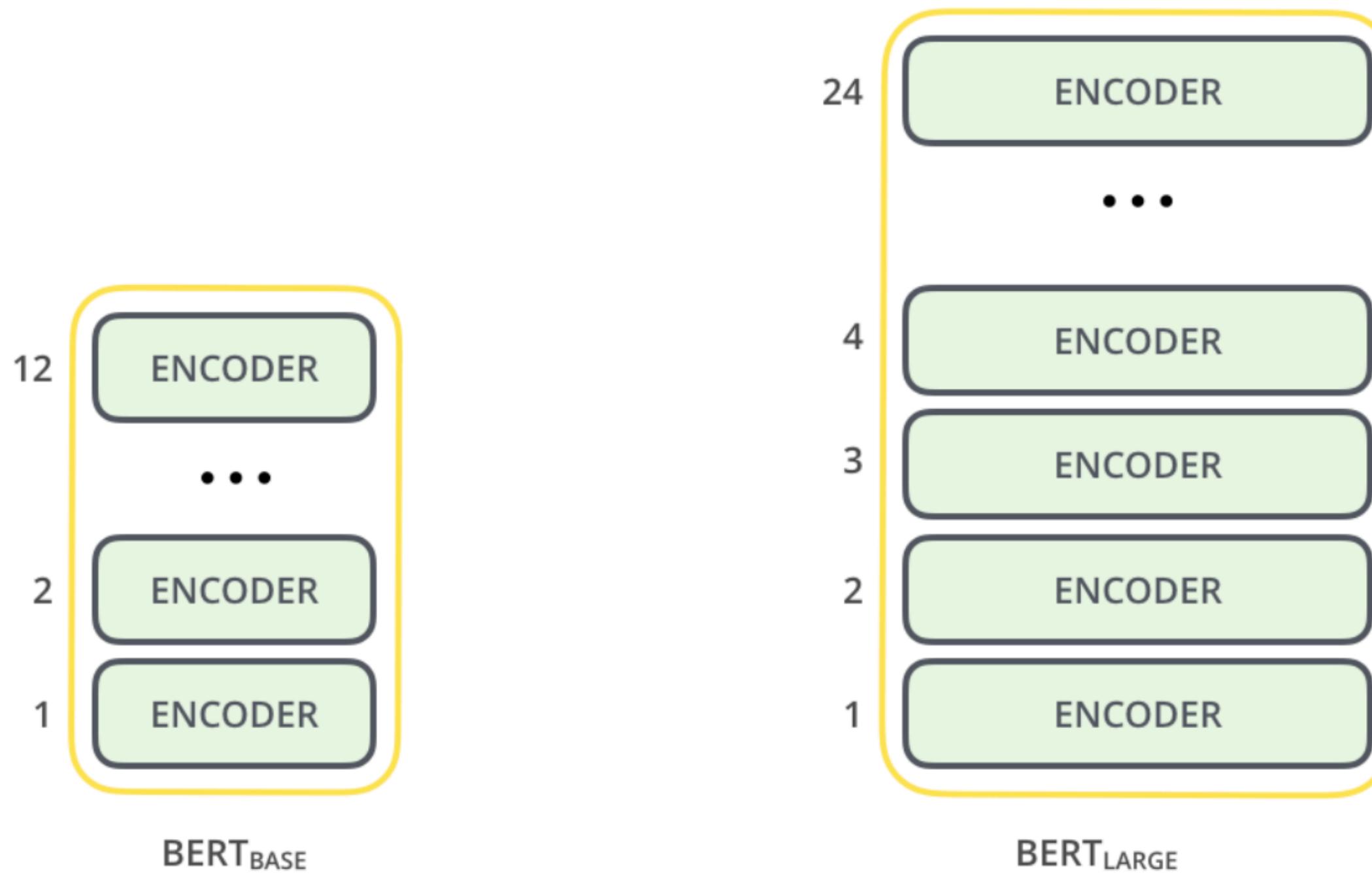


BERT_{BASE}



BERT_{LARGE}

BERT



BERT

Use the output of the masked word's position to predict the masked word

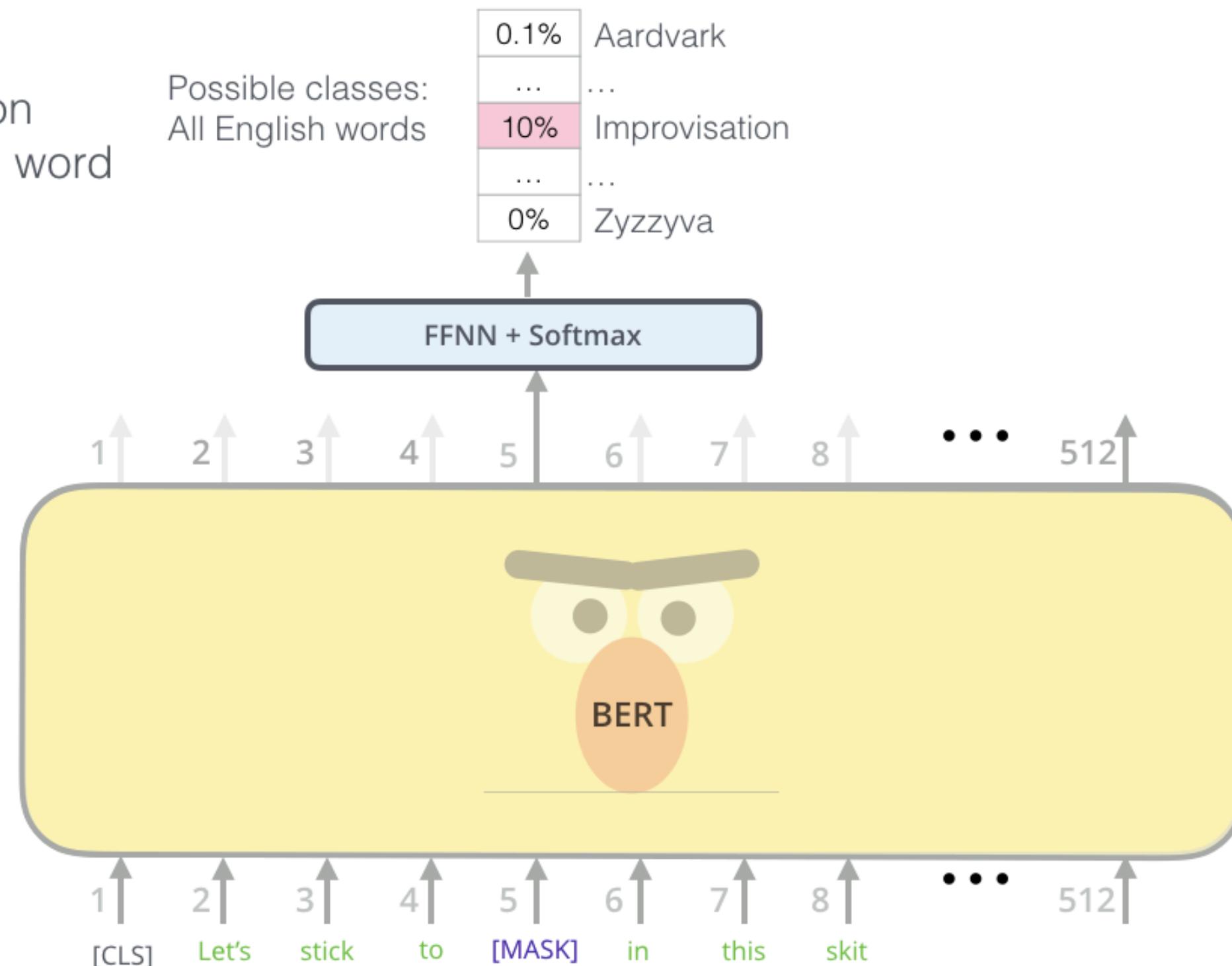
Possible classes:
All English words

0.1%	Aardvark
...	...
10%	Improvisation
...	...
0%	Zyyzyva

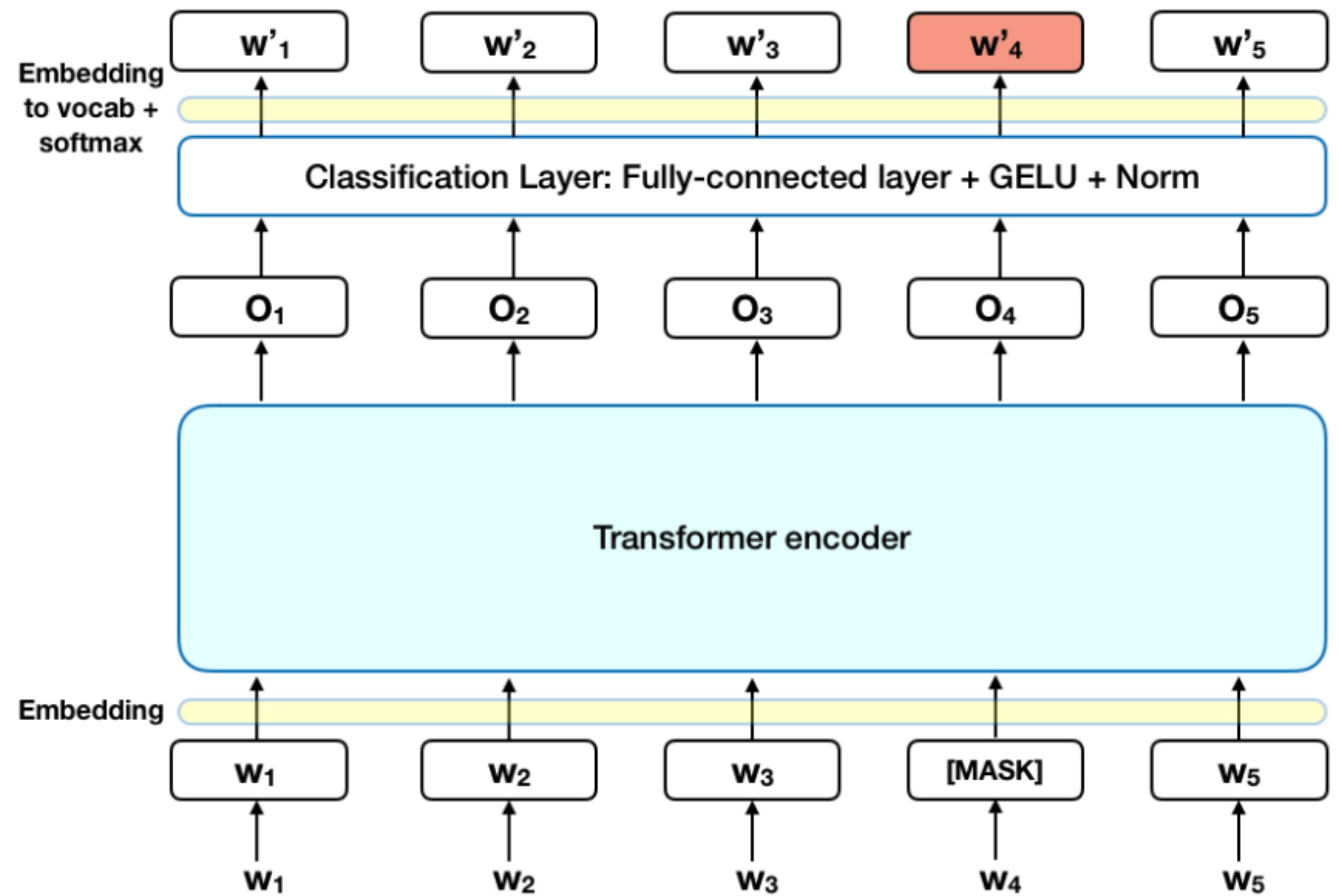
Randomly mask
15% of tokens

Input

[CLS] Let's stick to [MASK] in this skit



BERT

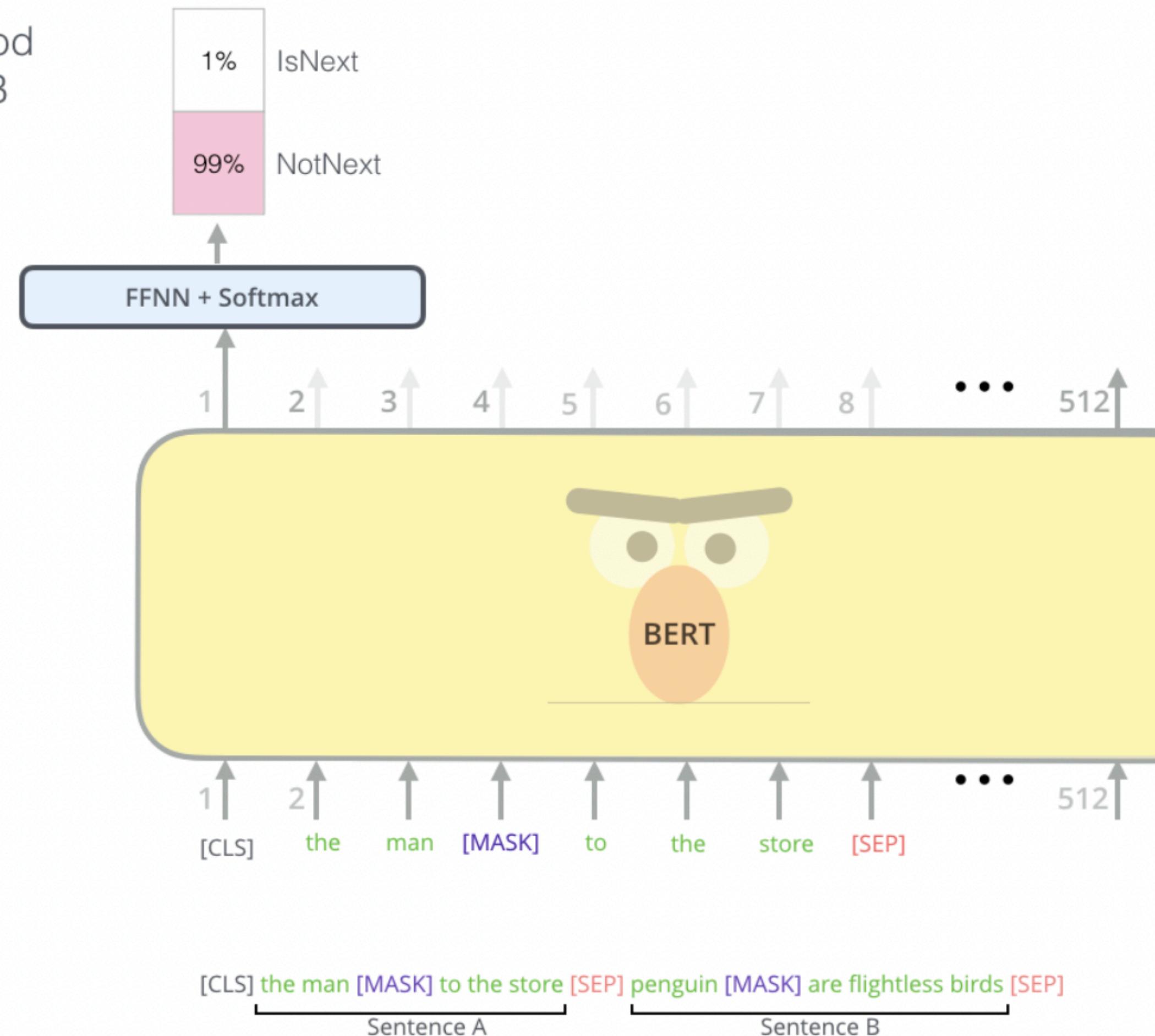


BERT

Predict likelihood
that sentence B
belongs after
sentence A

Tokenized
Input

Input



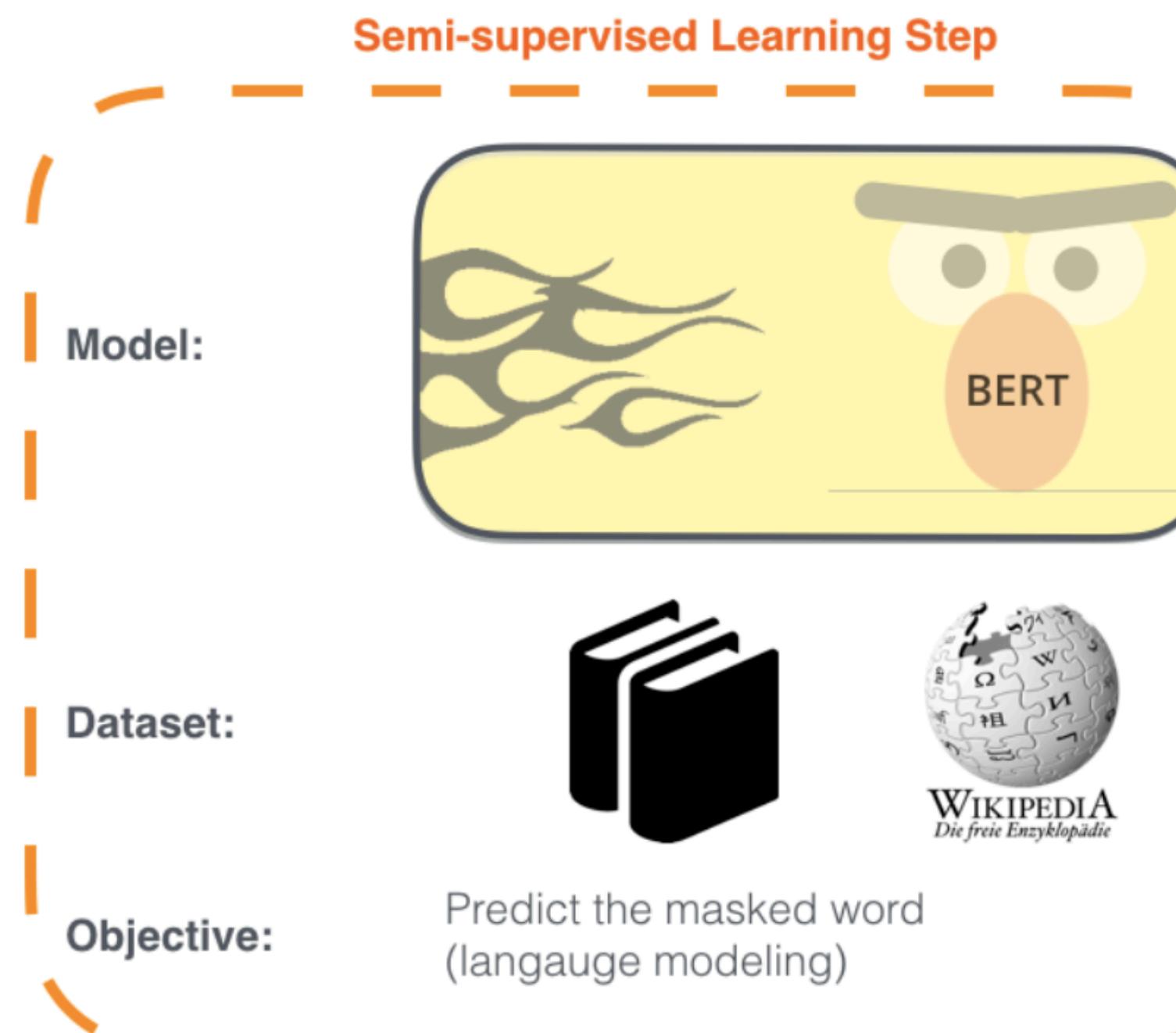
The second task BERT is pre-trained on is a two-sentence classification task. The tokenization is oversimplified in this graphic as BERT actually uses WordPieces as tokens rather than words --- so some words are broken down into smaller chunks.



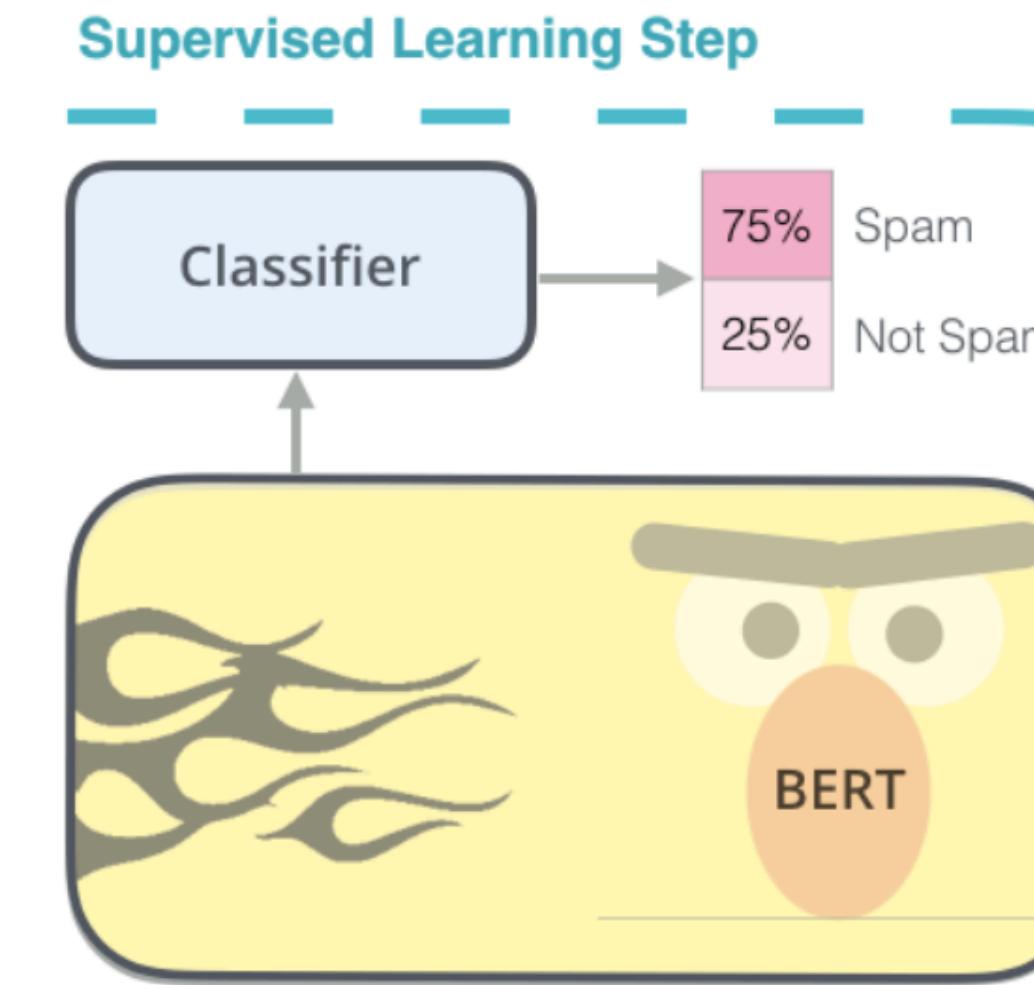
BERT

1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

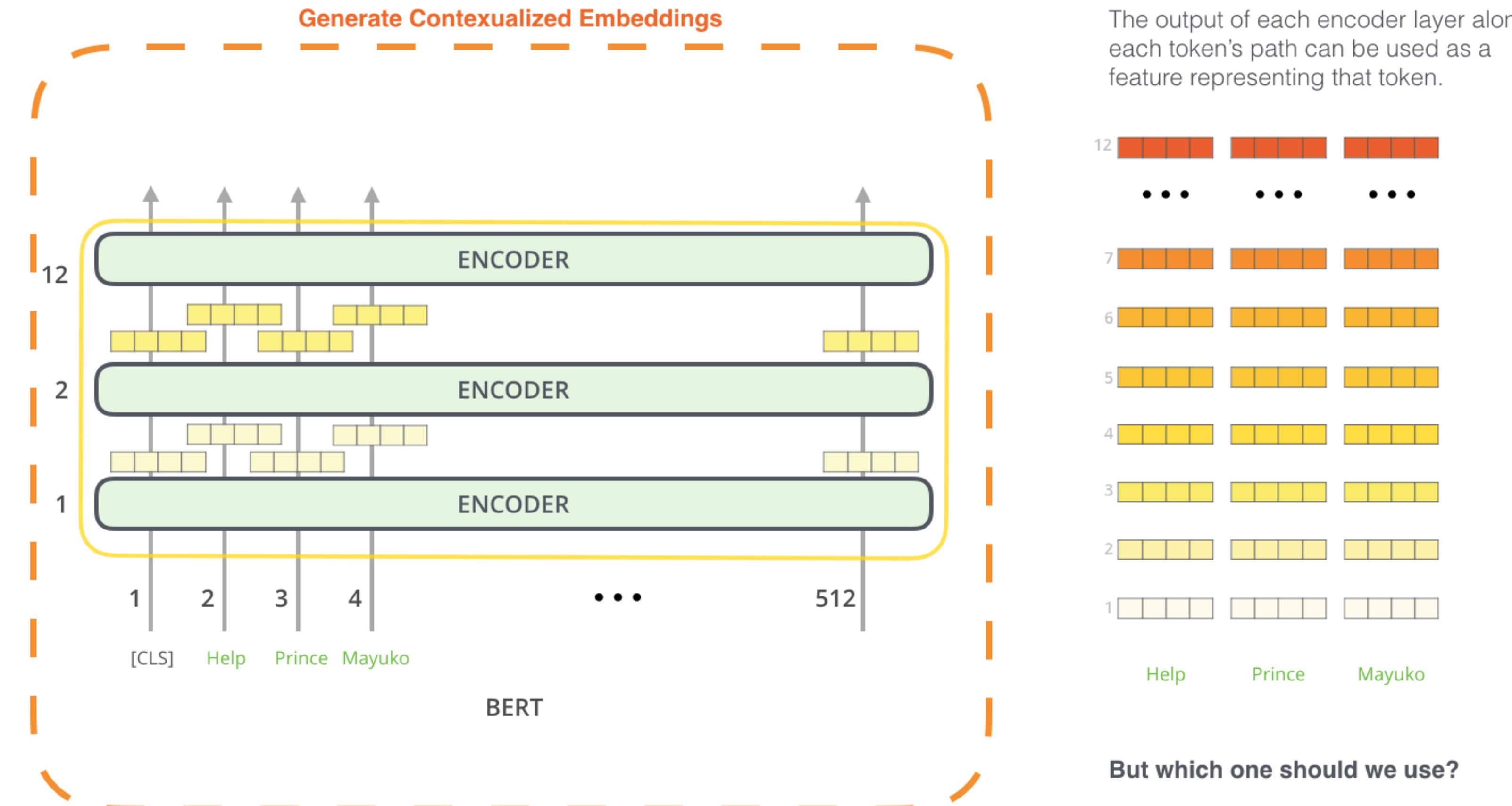


2 - Supervised training on a specific task with a labeled dataset.



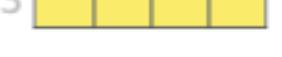
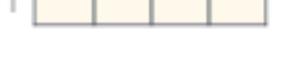
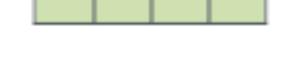
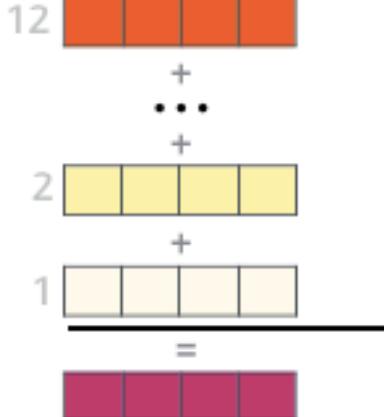
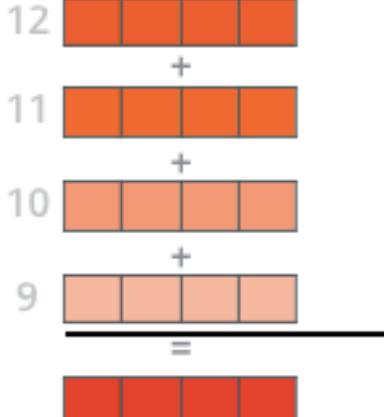
Email message	Class
Buy these pills	Spam
Win cash prizes	Spam
Dear Mr. Atreides, please find attached...	Not Spam

BERT for feature extraction



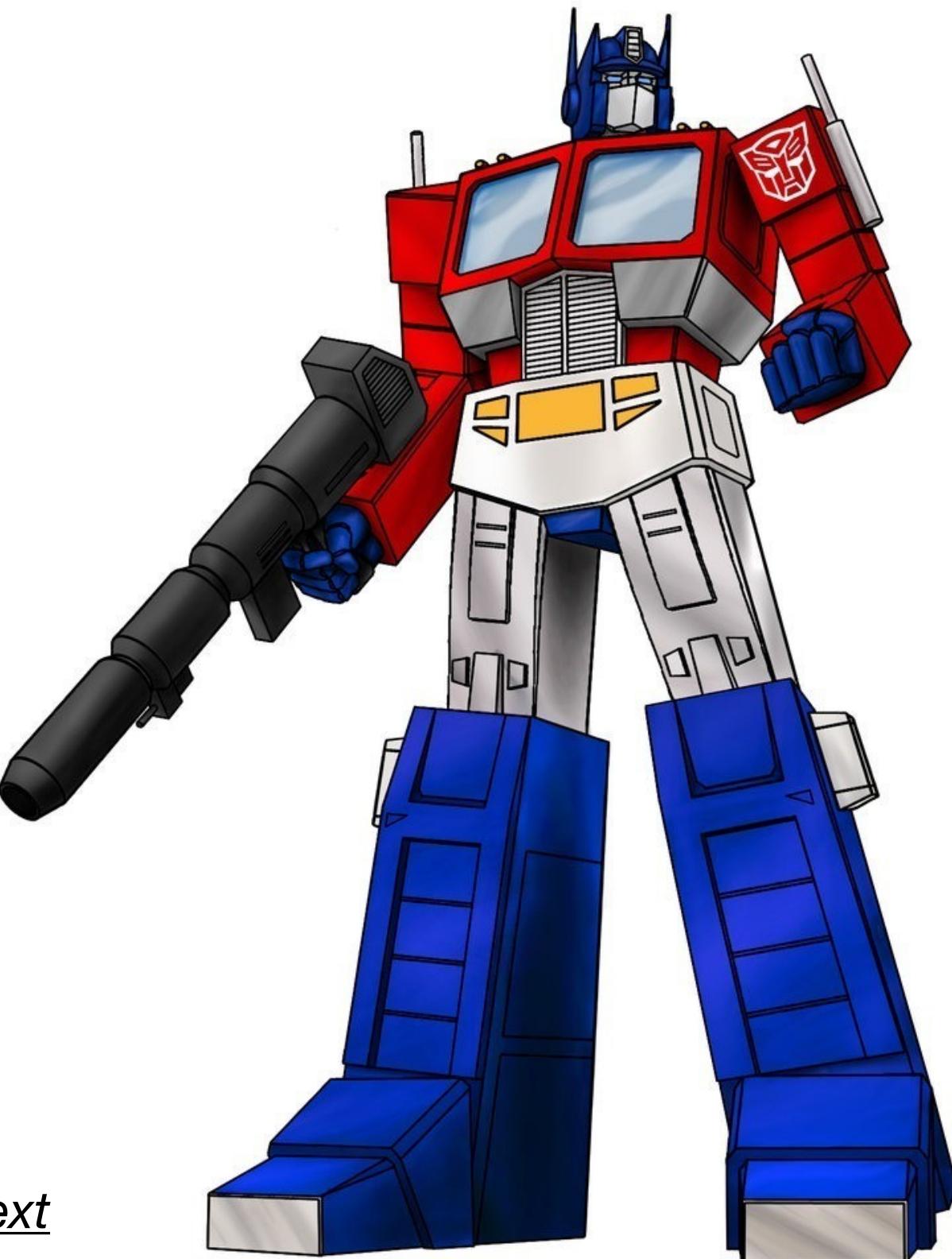
BERT for feature extraction

What is the best contextualized embedding for “Help” in that context?
For named-entity recognition task CoNLL-2003 NER

		Dev F1 Score
12		
• • •		
7		
6		
5		
4		
3		
2		
1		
		
Help		
First Layer		91.0
Last Hidden Layer		94.9
Sum All 12 Layers		95.5
Second-to-Last Hidden Layer		95.6
Sum Last Four Hidden		95.9
Concat Last Four Hidden		96.1

Feature of NLP

Transformer XL



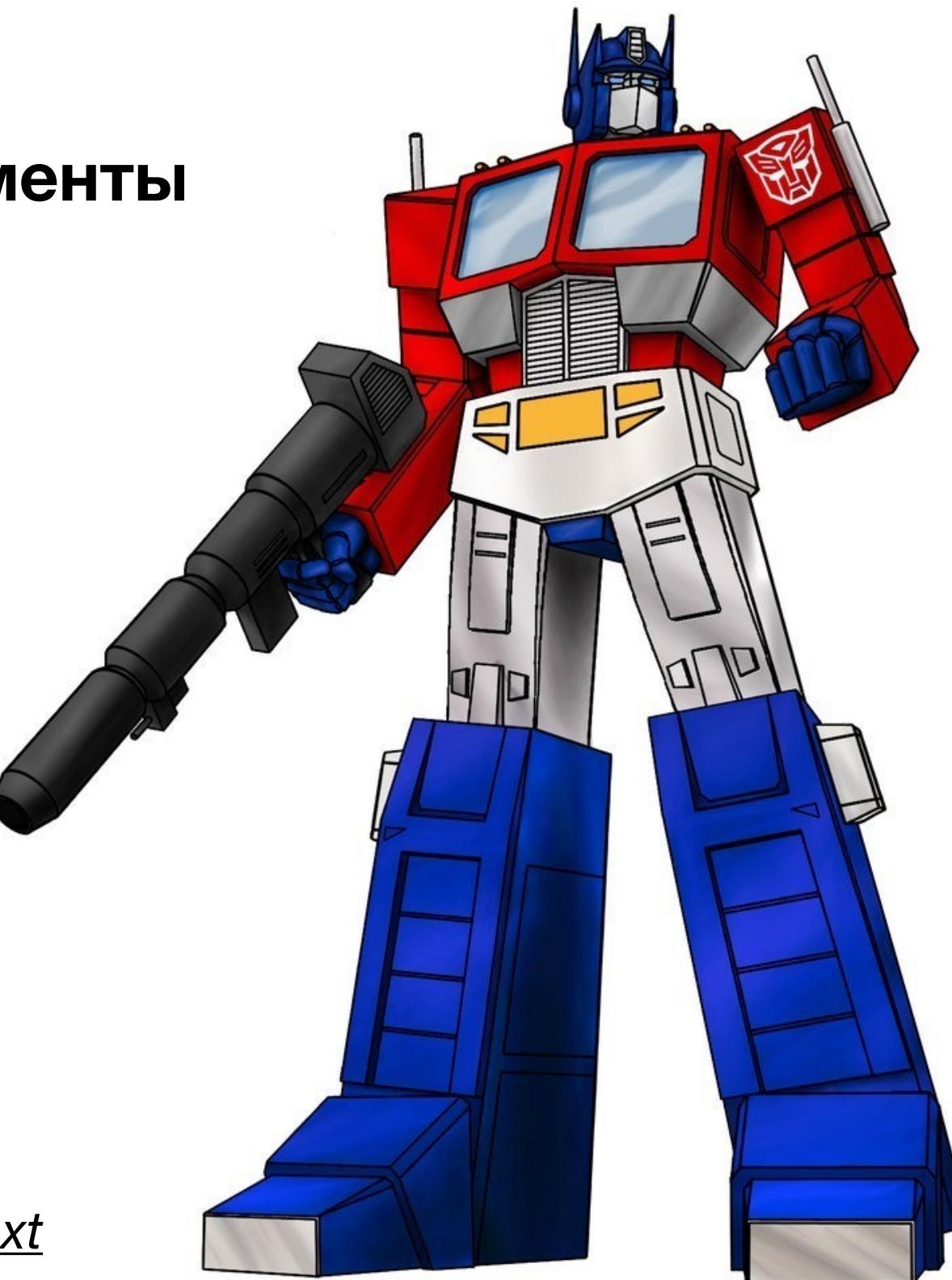
Transformer XL

Чем плох Open AI Transformer?



Transformer XL

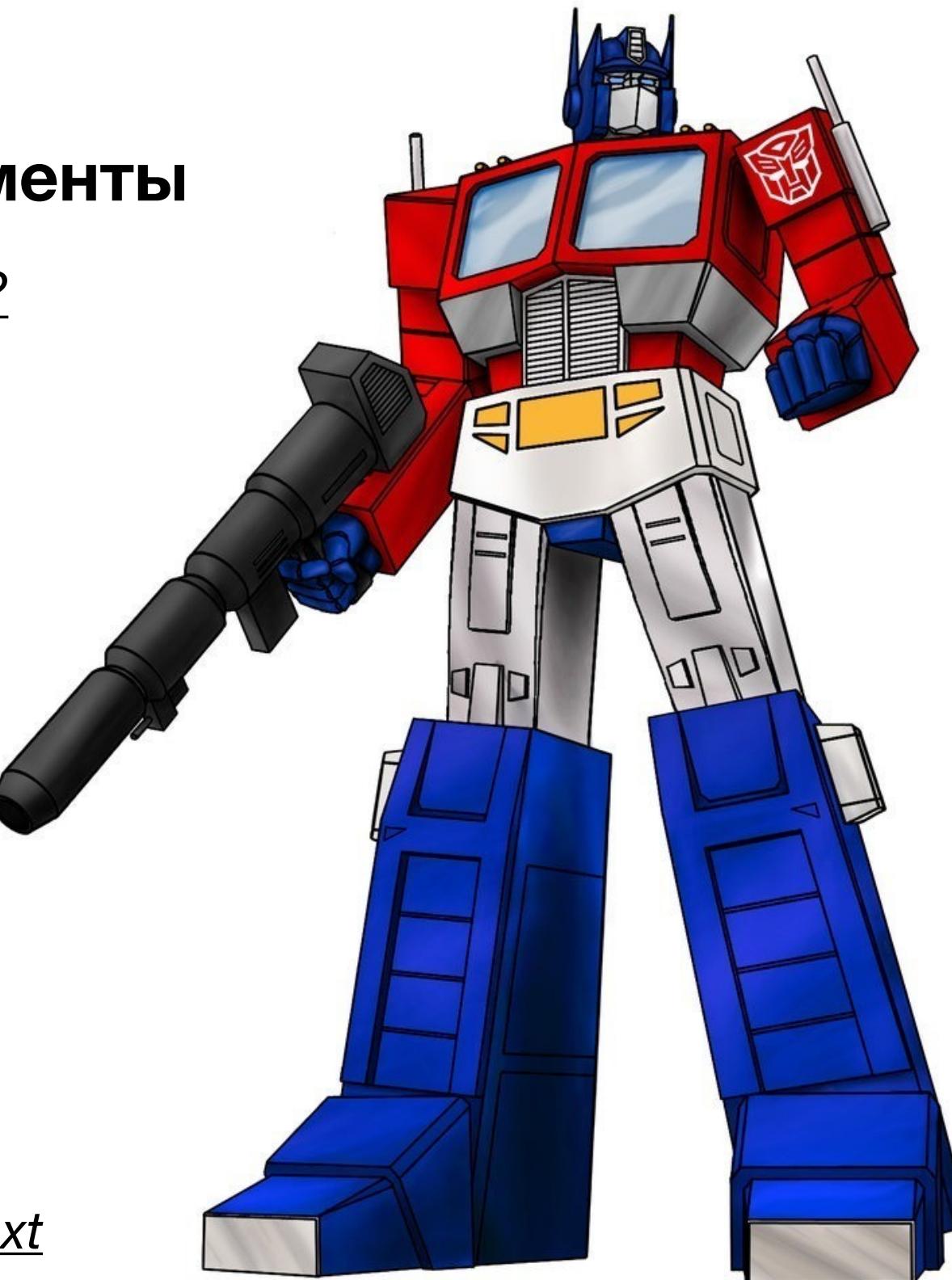
Давайте разобьем наш текст на сегменты



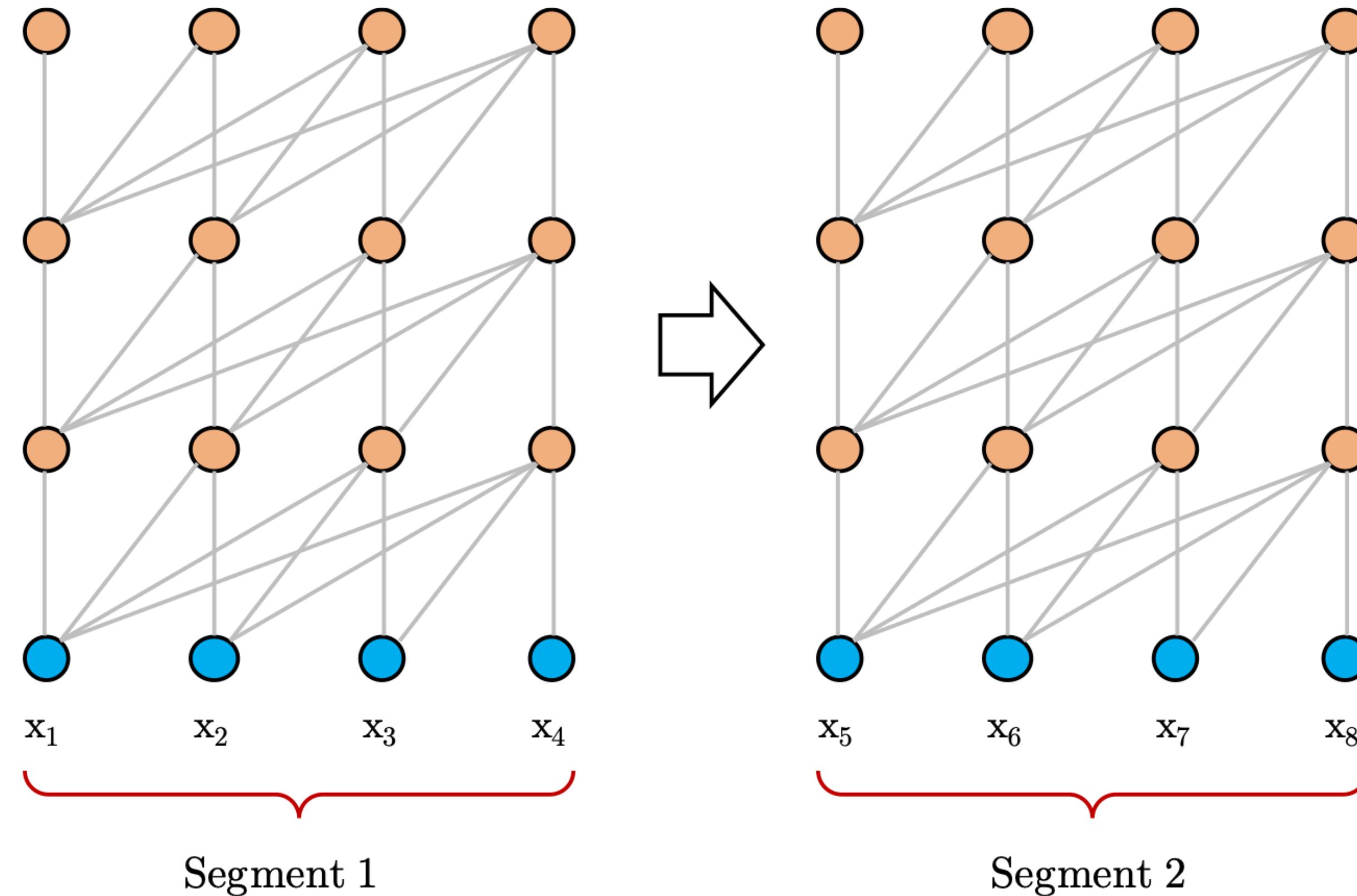
Transformer XL

Давайте разобьем наш текст на сегменты

Как будет работать обычный transformer?

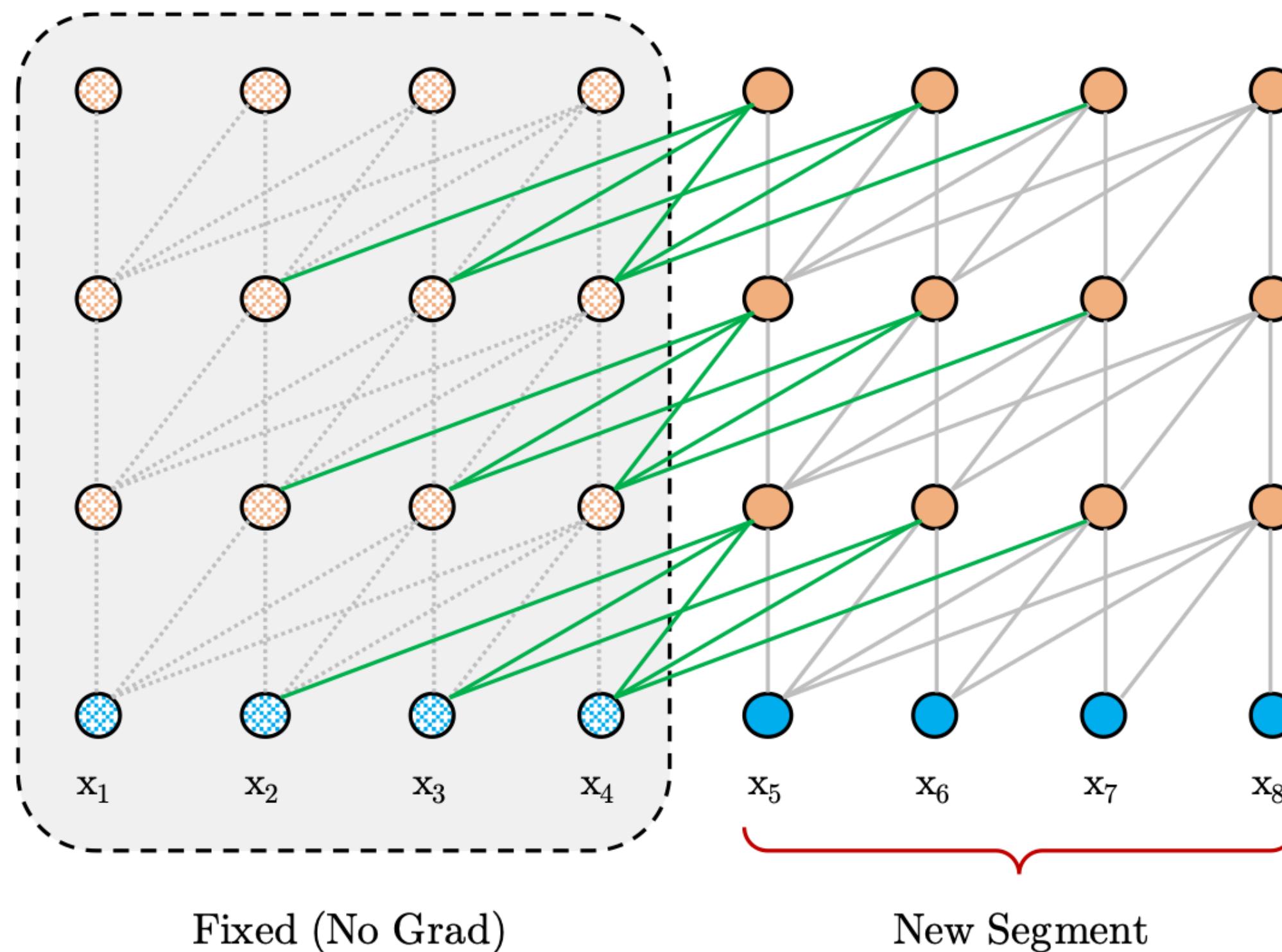


Transformer XL



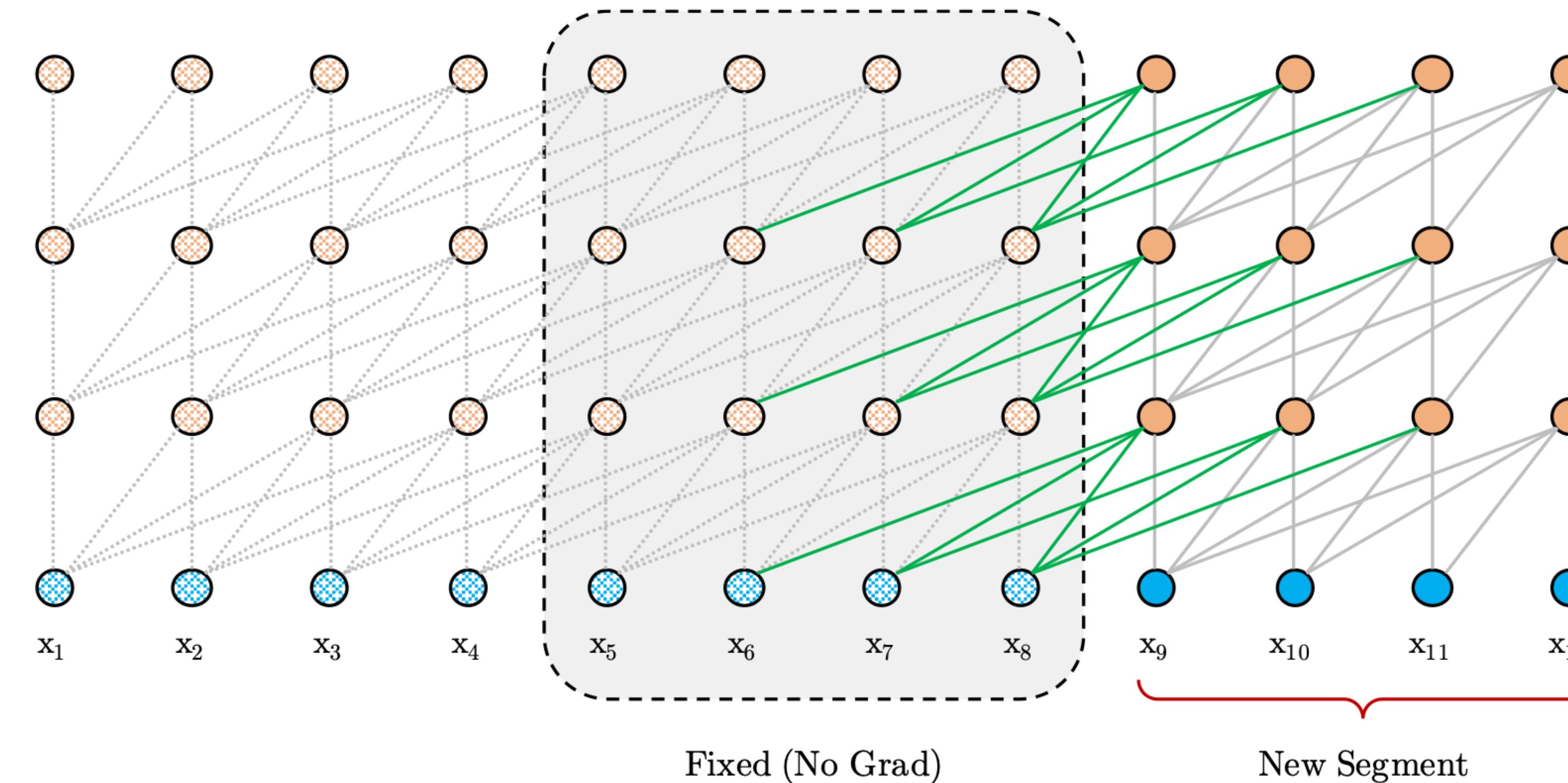
Transformer XL

Segment-level recurrence:



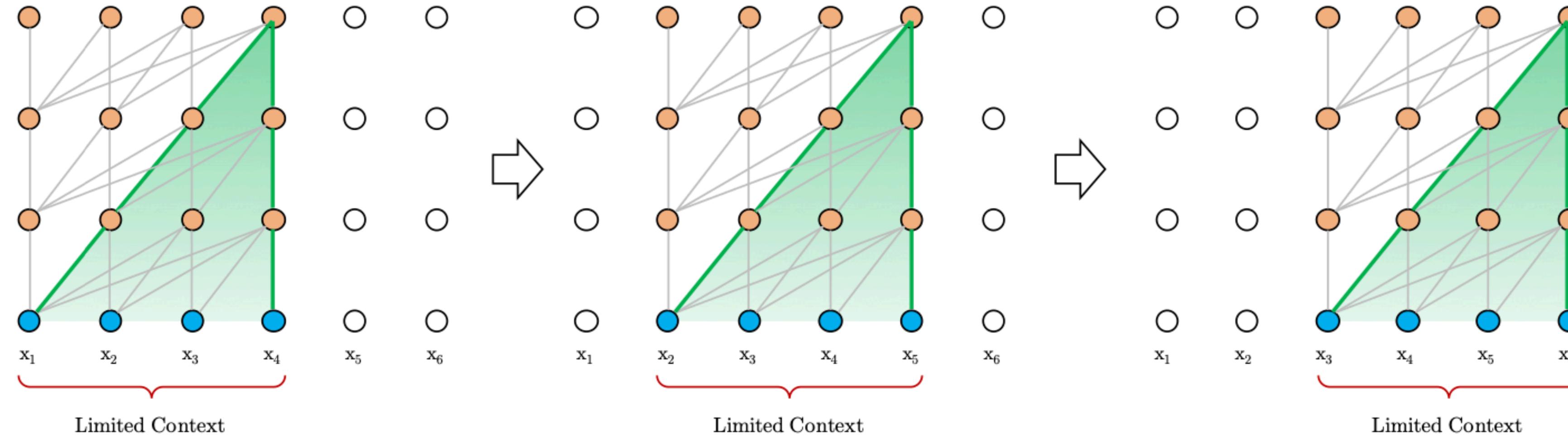
Transformer XL

Segment-level recurrence:



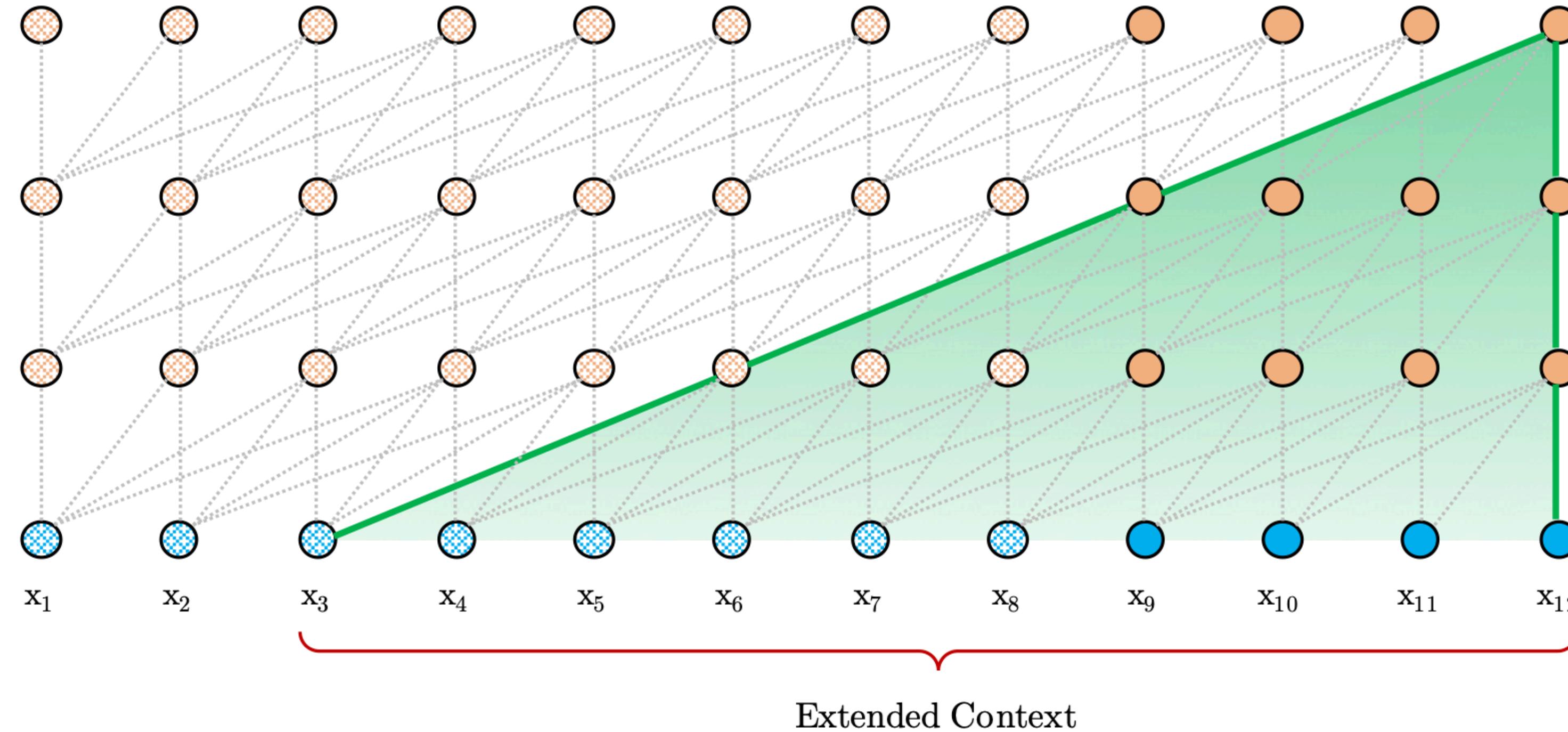
Transformer XL

Evaluation phase of Vanilla transformer:



Transformer XL

Evaluation phase of Transformer XL:



SOTA GPT-2

SOTA GPT-2

- Transformer-based architecture
- trained to predict the **next** word
- 1.5 billion parameters
- Trained on 8 million web-pages



<https://openai.com/blog/gpt-2-6-month-follow-up/>

https://github.com/ml-mipt/ml-mipt/blob/advanced/week05_BERT_and_LDA/Lecture_BERT_DIHT.pdf

GPT-2: fake news and hype

Top stories



[OpenAI built a text generator so good, it's considered too dangerous to release](#)

TechCrunch

11 hours ago



[Elon Musk's AI company created a fake news generator it's too scared to make public](#)

BGR.com

9 hours ago



[The AI That Can Write A Fake News Story From A Handful Of Words](#) >

NDTV.com

2 hours ago