

# INF721

2024/2



# Deep Learning

## L17: Transformers

# Logistics

## Last Lecture

- ▶ Machine Translation
- ▶ Decoding
  - ▶ Greedy Search
  - ▶ Beam Search
- ▶ Attention in RNNs

# Lecture Outline

- ▶ Machine Translation
- ▶ Problems with RNNs
- ▶ Transformers
  - ▶ Self-Attention
  - ▶ Multi-head Attention
  - ▶ Encoder & Decoder
  - ▶ Positional Encoding
  - ▶ Masked Multi-head Attention

# Machine Translation

Given a dataset of sentence pairs:

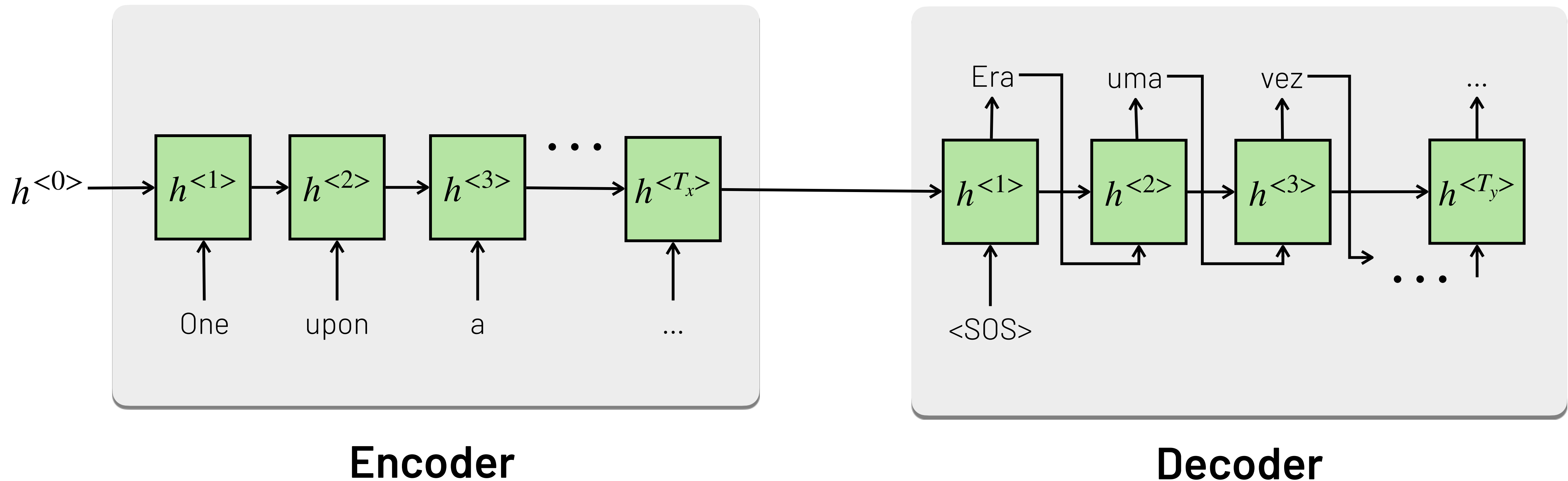
$$(x = \{x^{<1>}, x^{<2>}, \dots, x^{<T_x>}\}, y = \{y^{<1>}, y^{<2>}, \dots, y^{<T_y>}\}),$$

we want to learn a model that maps  $x$  into  $y$ .

Portuguese	English
Olá, como vai você?	Hello, how are you?
O livro está em cima da mesa.	The book is on the table.
Lucas irá viajar ao Rio em Dezembro.	Lucas is travelling to Rio in December.
Em Dezembro, Lucas irá viajar ao Rio.	Lucas is travelling to Rio in December.
....	....

# Problems with RNNs

- ▶ Struggle to capture long dependencies in sequences
- ▶ Hard to parallelize

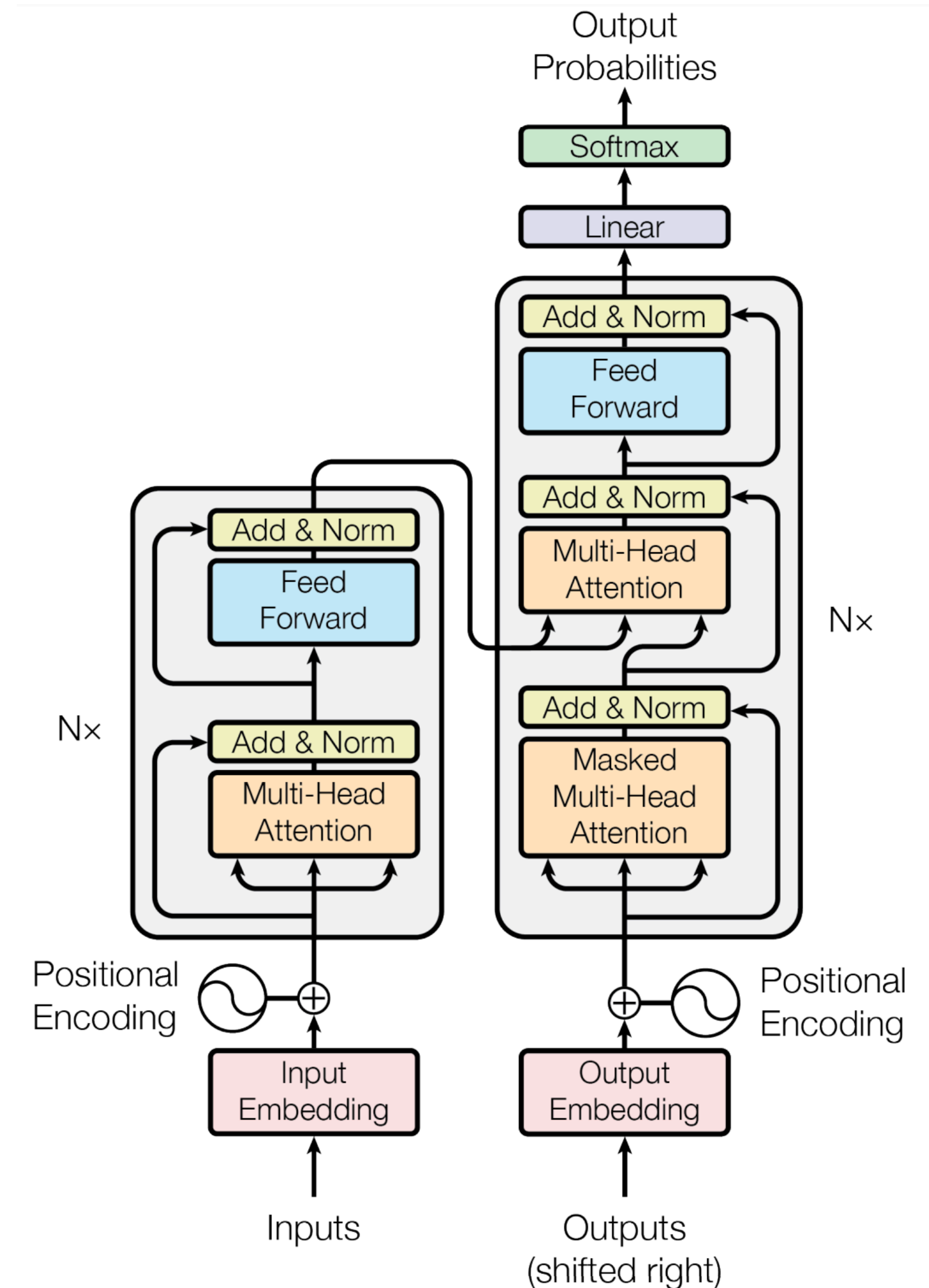


# Transformers

**Transformers** are an encoder-decoder architecture to process sequences using only attention (eliminating recurrence).

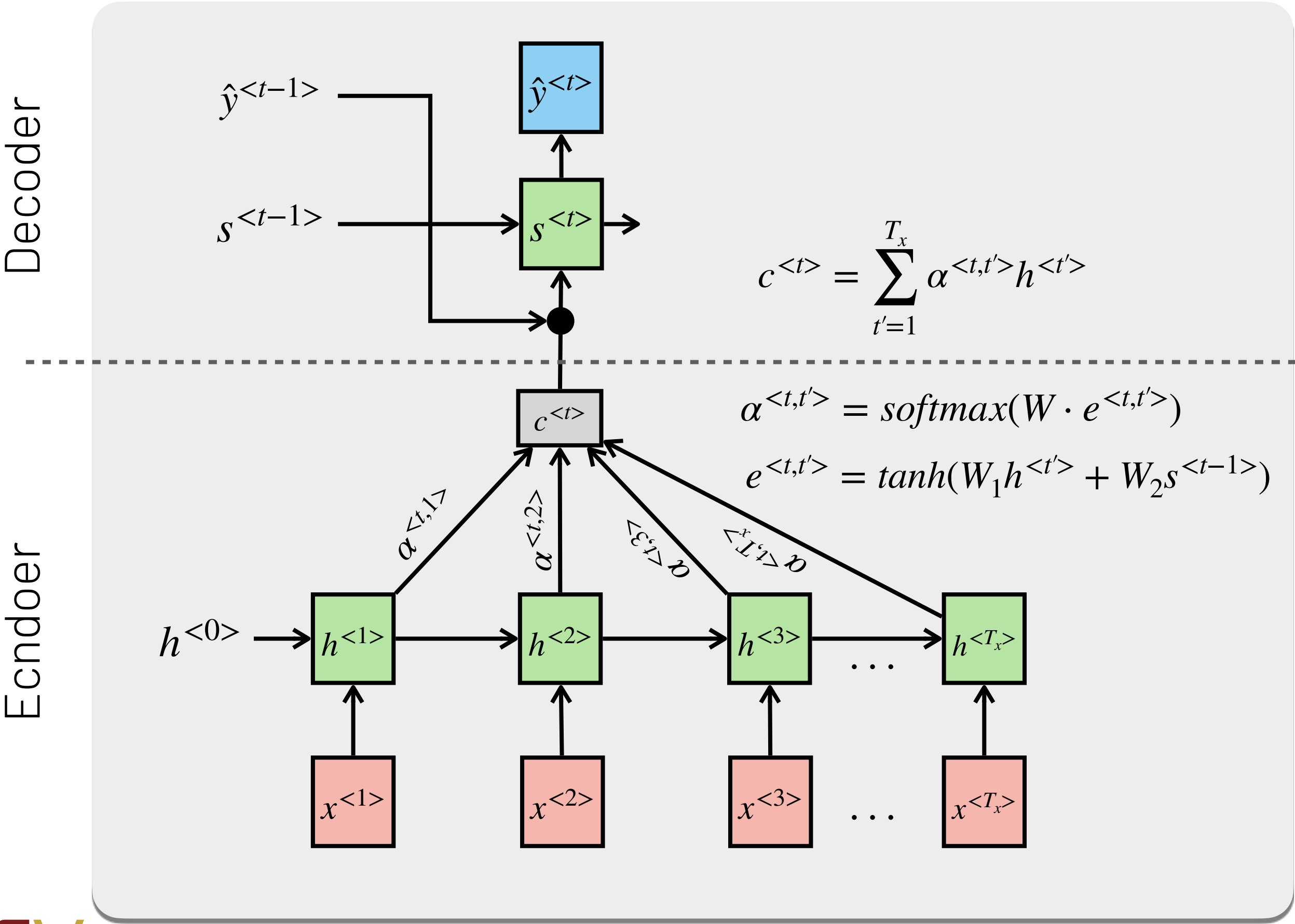
Initially proposed for machine translation, but proved to be very effective in many other problems in:

- ▶ Natural Language Processing
- ▶ Computer Vision
- ▶ Reinforcement Learning
- ▶ ...

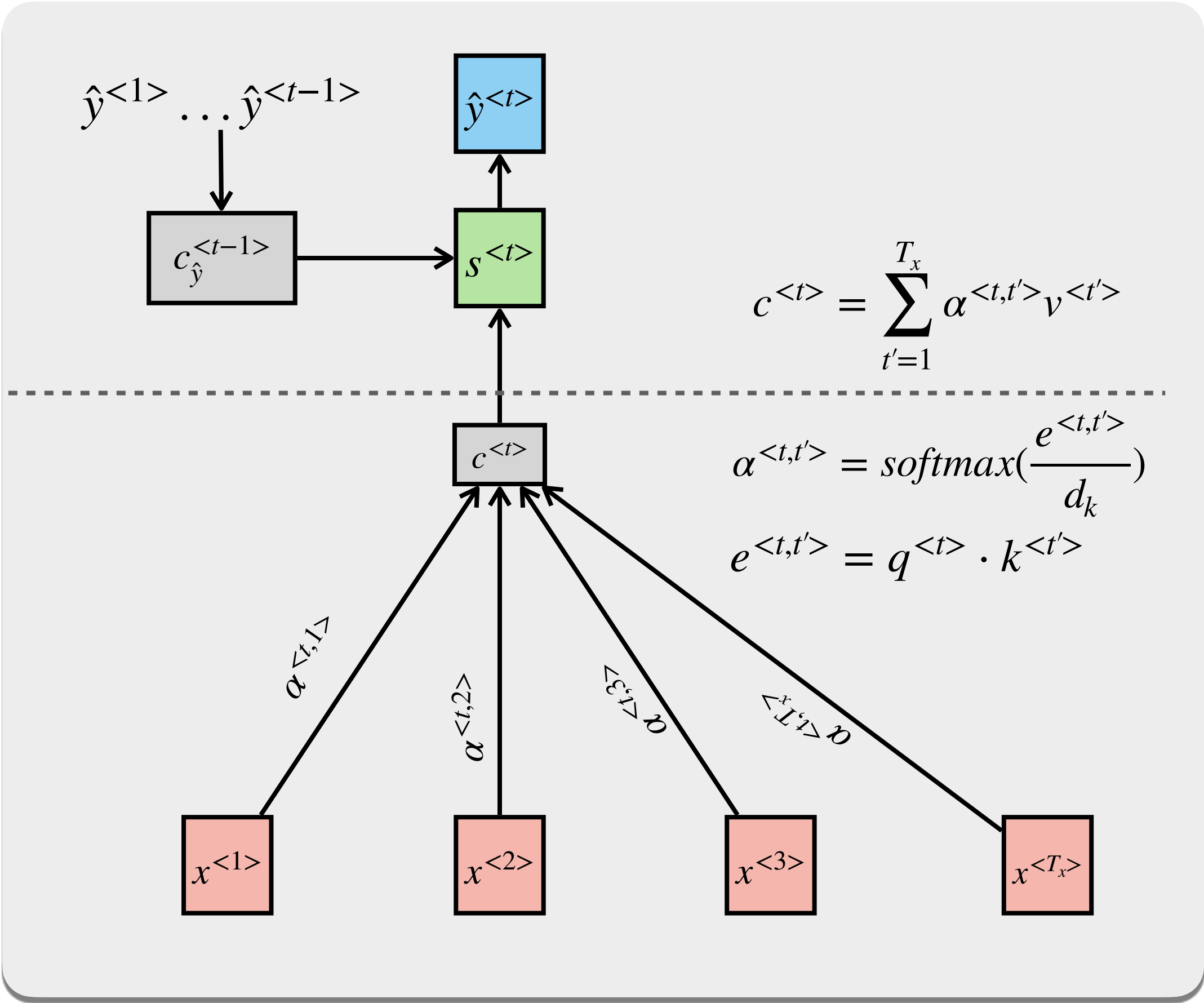


# Attention in RNNs vs. Transformers

## RNNs    Badahnau Attention

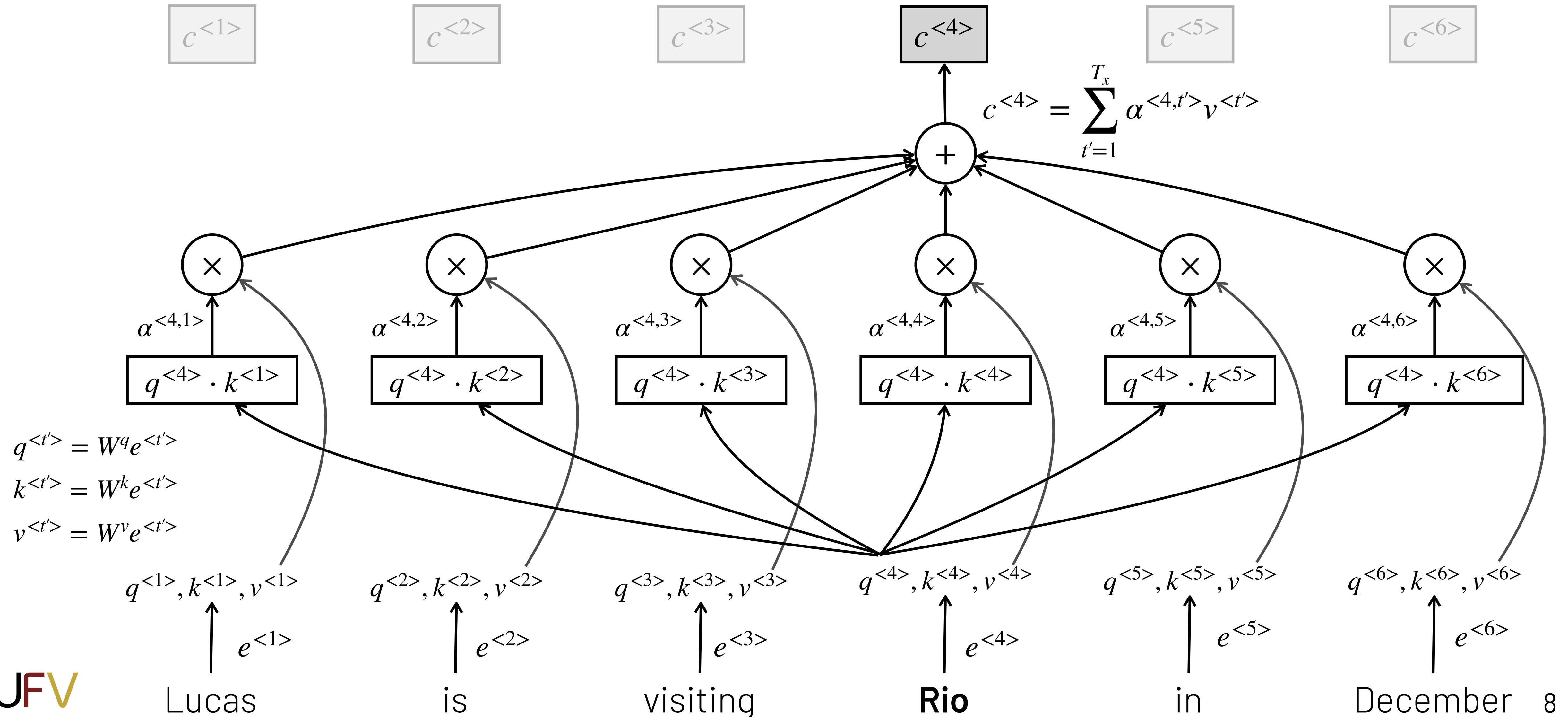


## Transformers    Scaled Dot-Product Attention



# Self-Attention

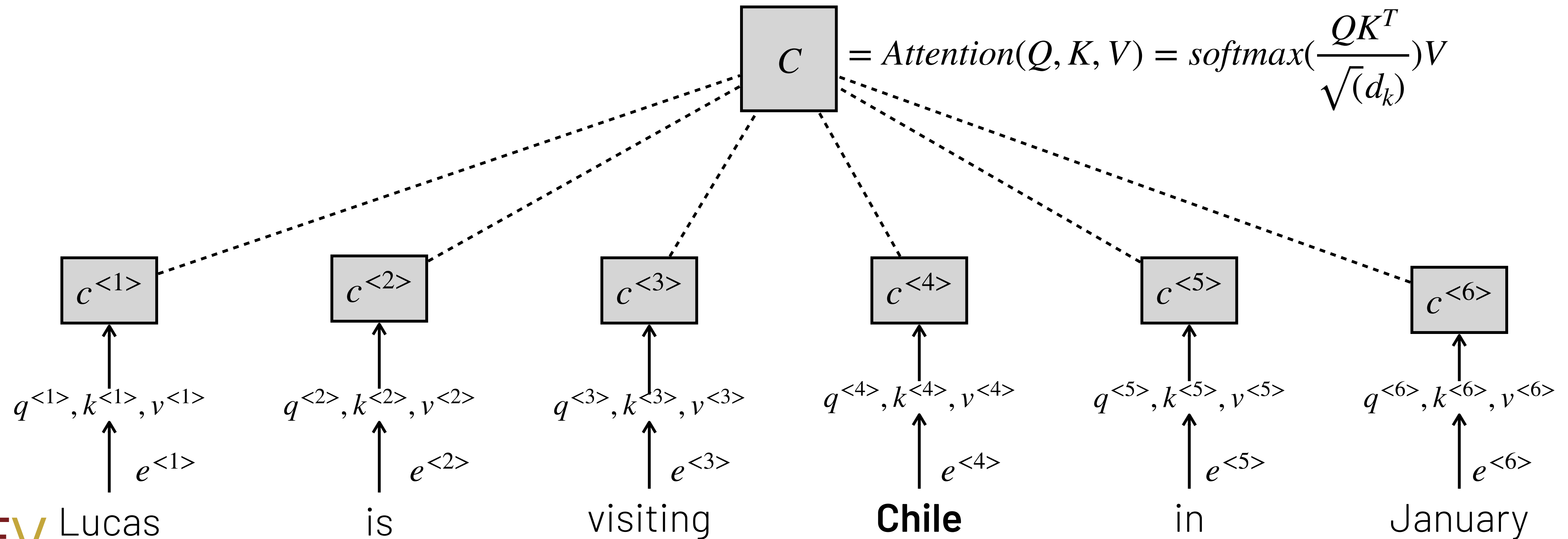
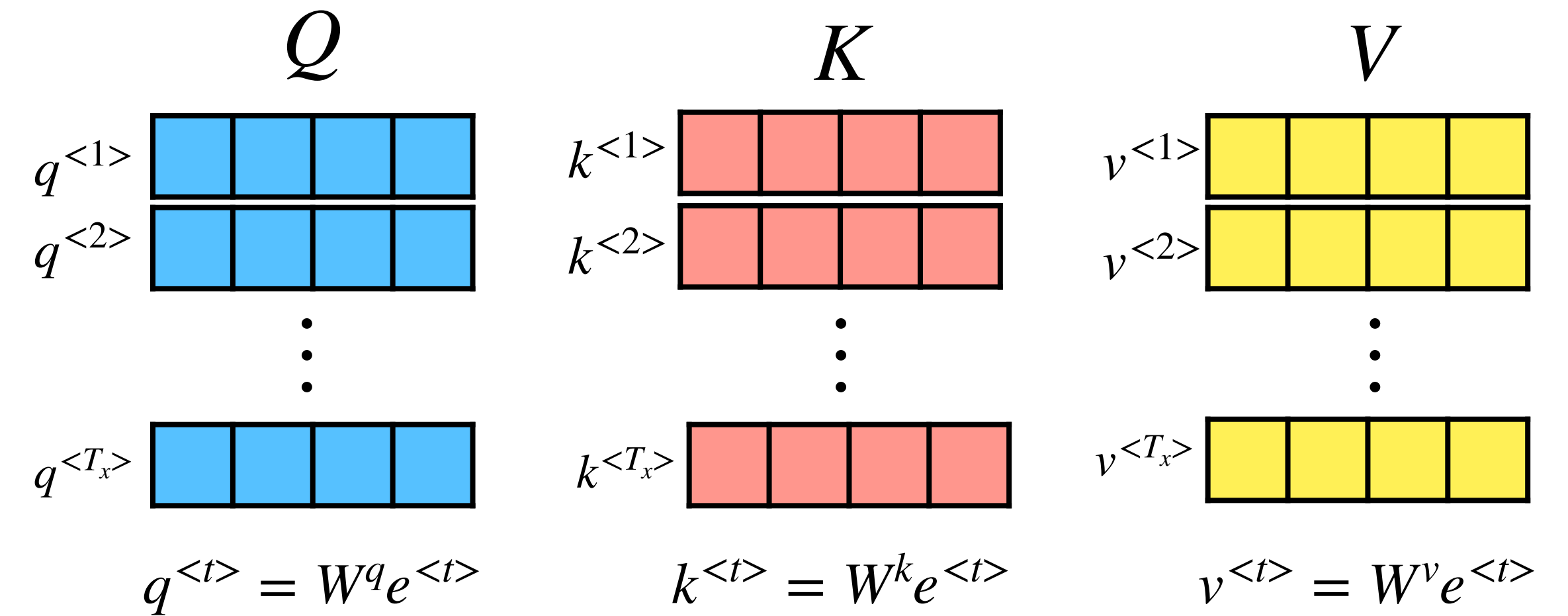
The key idea behind the Transformer is self-attention mechanism, which learns a context vector  $c^{<t>}$  for each input element  $x^{<t>}$  based on the input sequence  $x$  itself.





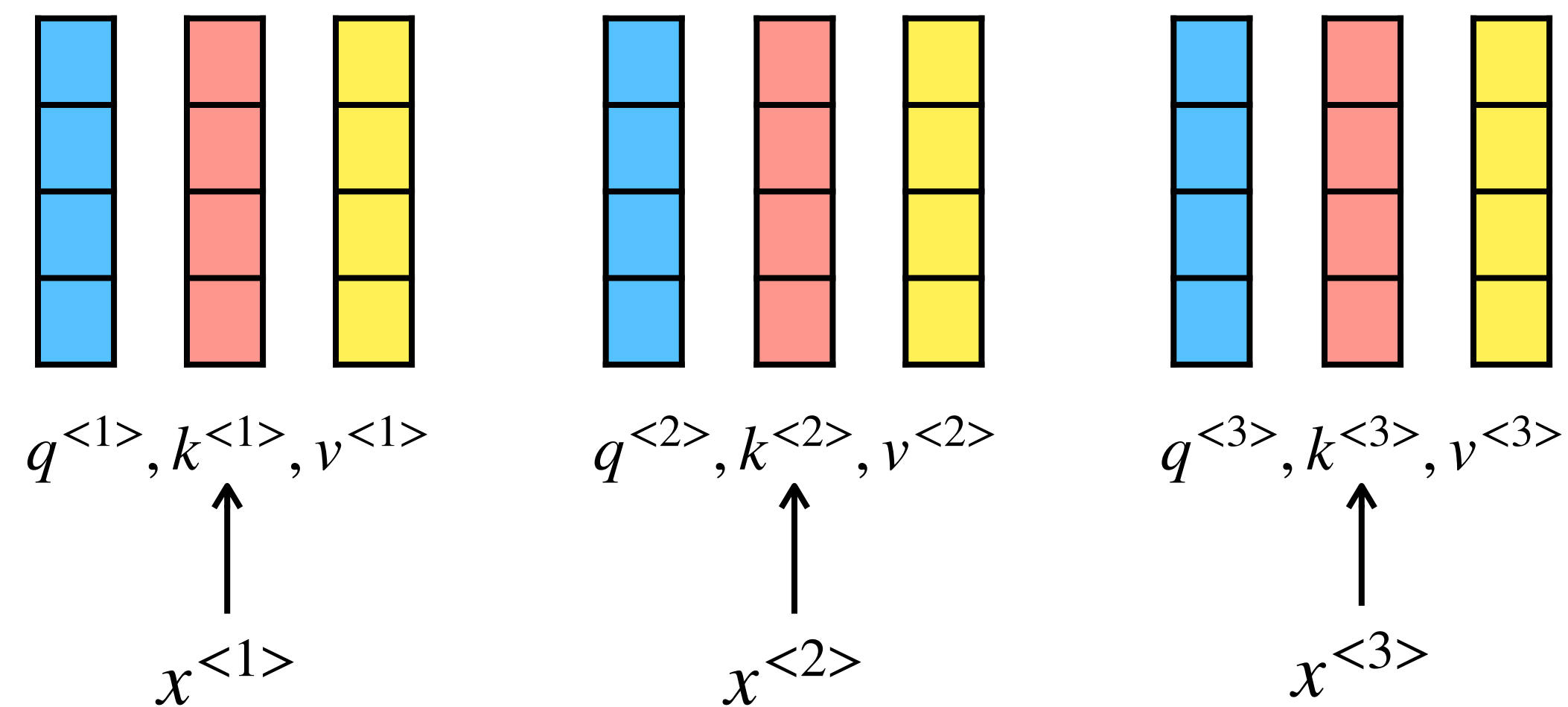
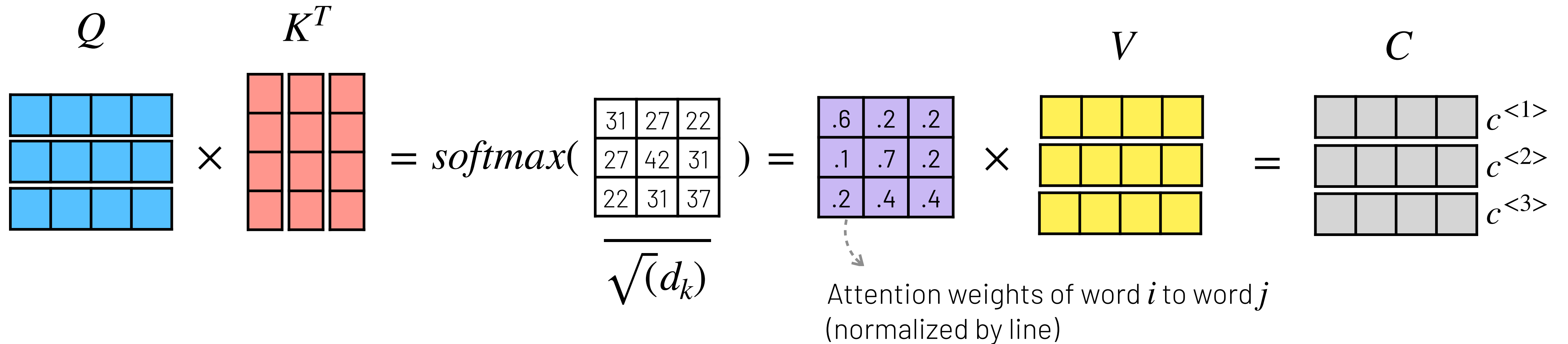
# Self-Attention

The contextual representation  $C = \{c^{<1>}, \dots, c^{<T_x>}\}$  of the entire input sequence  $x = \{x^{<1>}, \dots, x^{<T_x>}\}$  can be computed in a vectorized way combining vectors  $q^{<t>}, k^{<t>}, v^{<t>}$  in matrices  $Q, K$  e  $V$



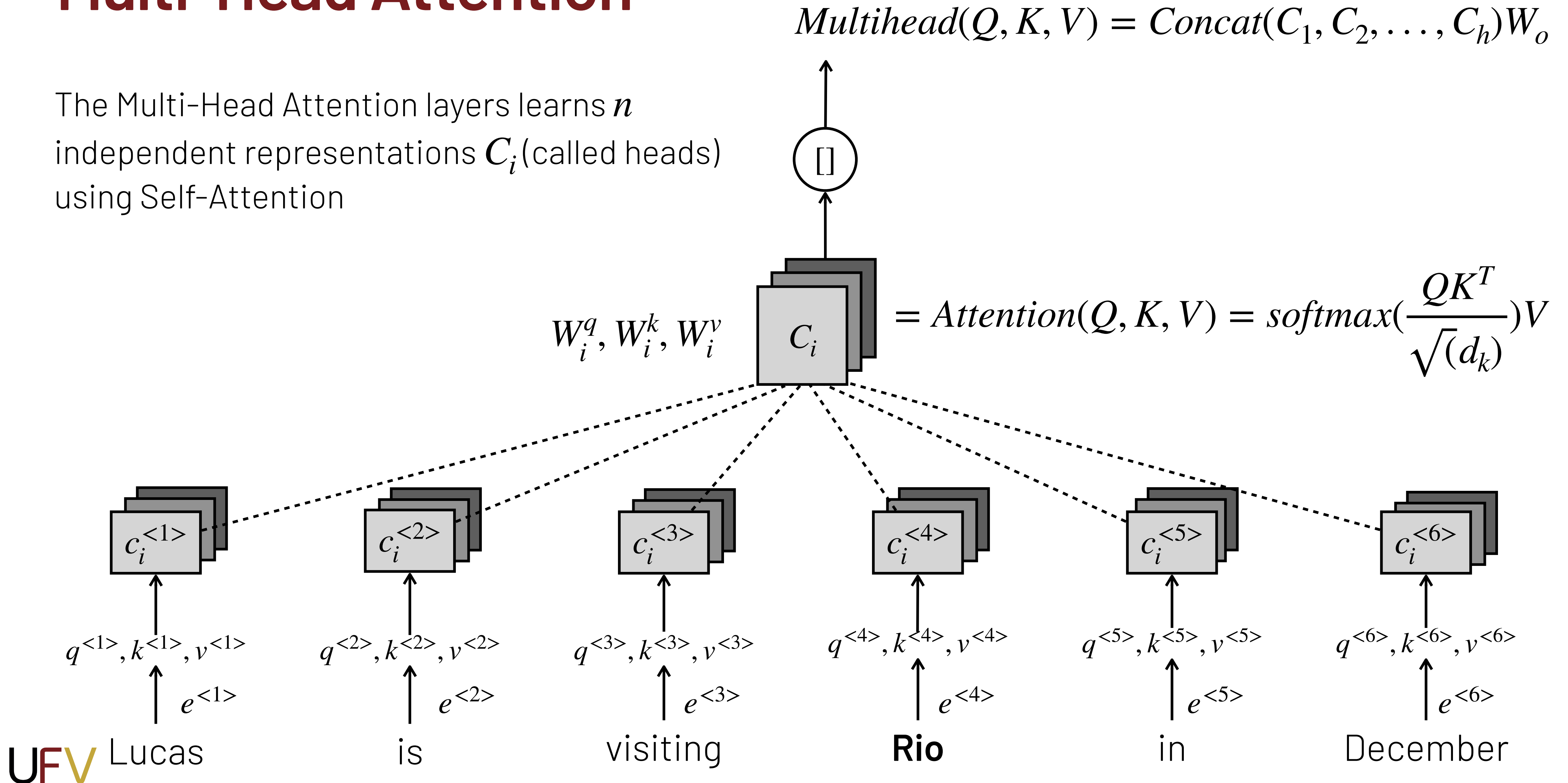
# Self-Attention

$$C = \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



# Multi-Head Attention

The Multi-Head Attention layers learns  $n$  independent representations  $C_i$  (called heads) using Self-Attention



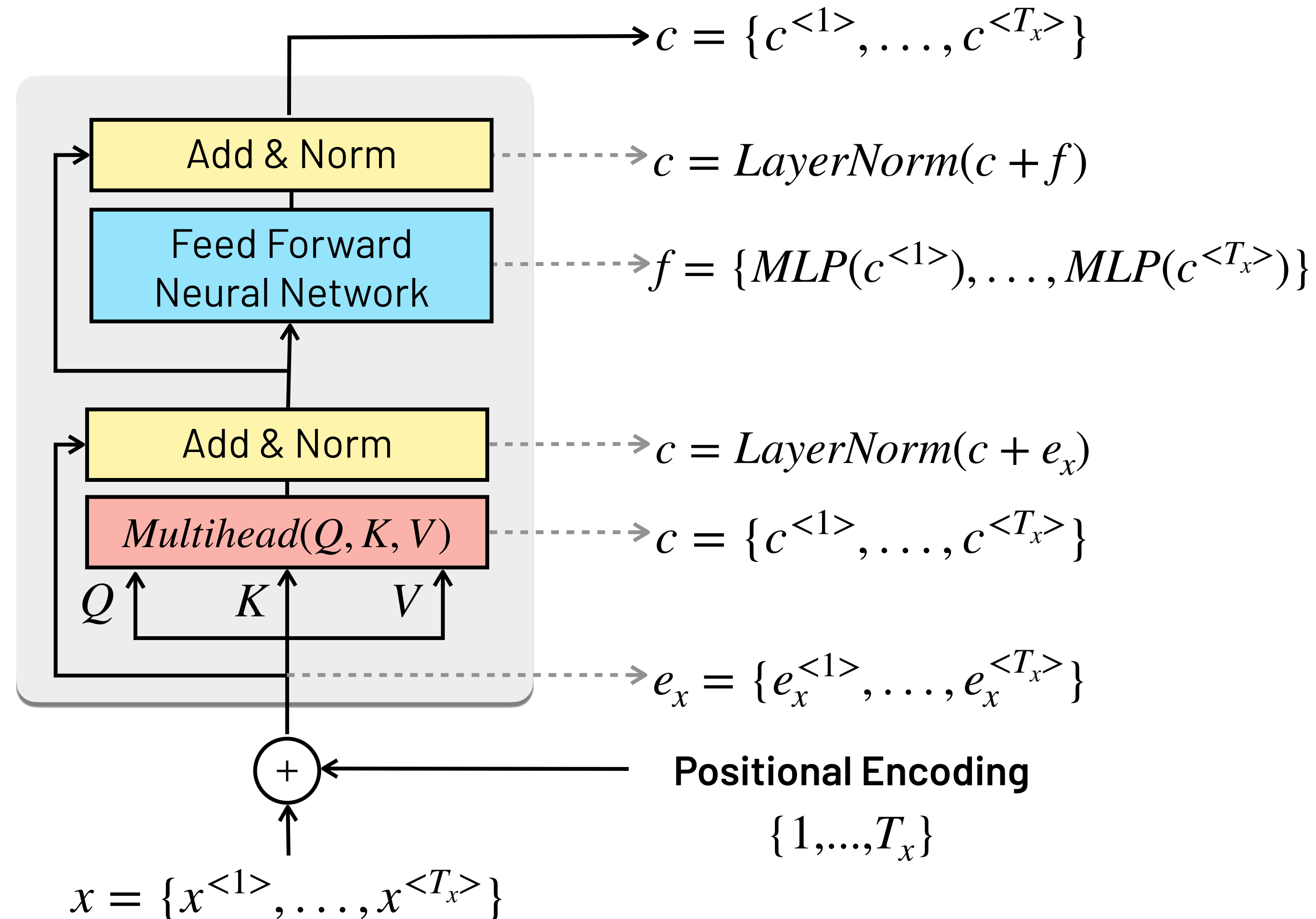
# Encoder

**Input:** a sequence  $x = \{x^{<1>}, \dots, x^{<T_x>}\}$

**Output:** a contextual representation  $C = \{c^{<1>}, \dots, c^{<T_x>}\}$  of  $x$

The encoder applies a **Multihead Layer** followed by a **Feed Forward Neural Network** (MLP).

Both are normalized with Layer Norm (**Norm**) and connected with a residual connection (**Add**)

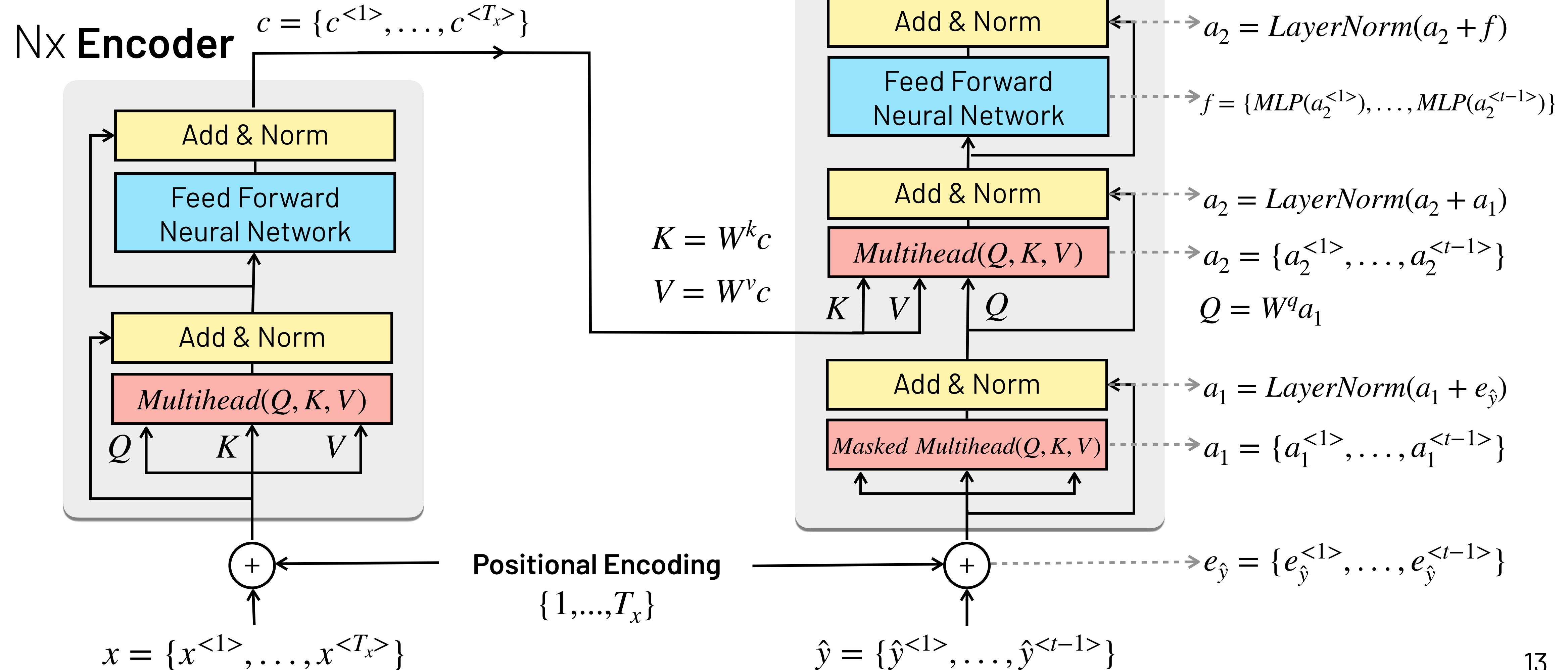


# Decoder

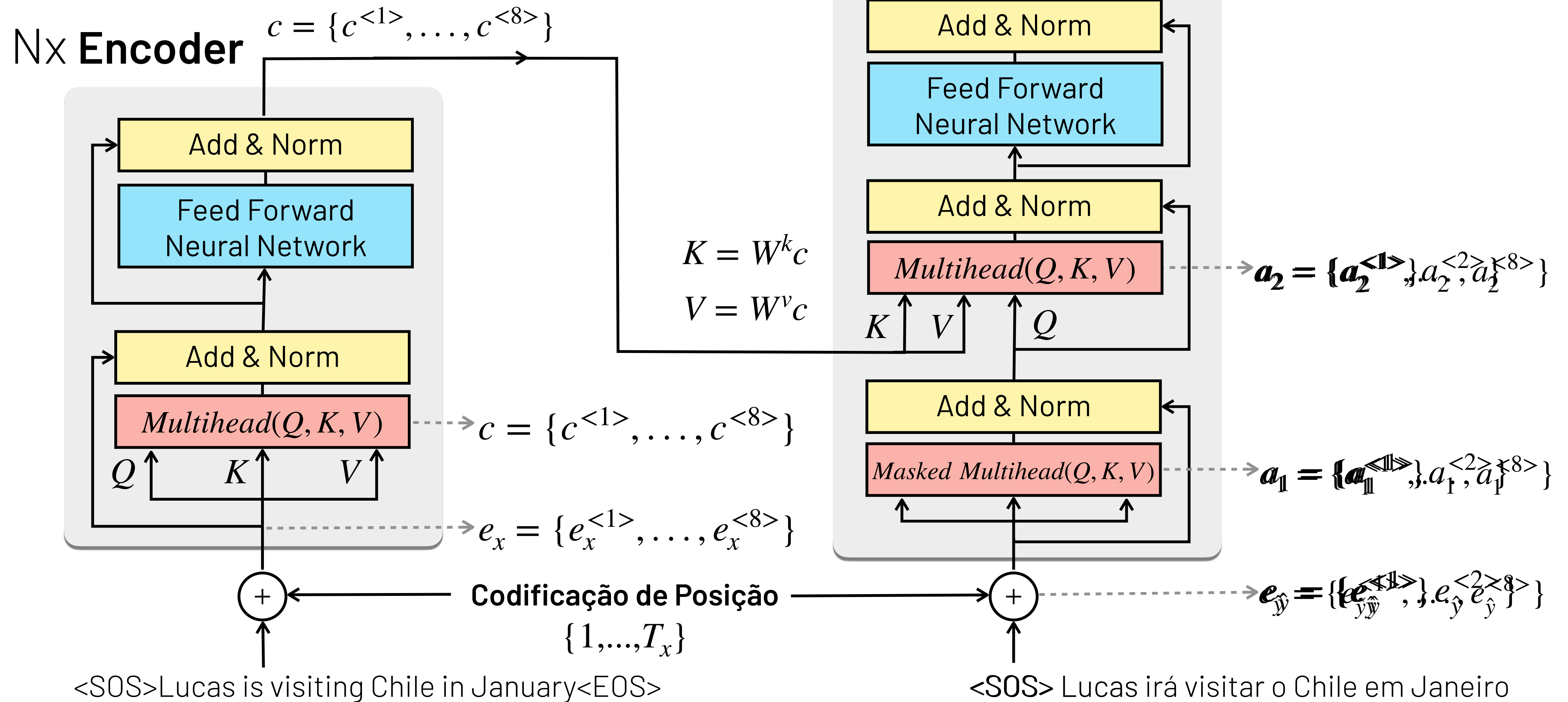
**Input:** Input context  $C$  and previous  $\{\hat{y}^{<1>}, \dots, \hat{y}^{<t-1>}\}$  output tokens

**Output:** The next token  $\hat{y}^{<t>}$

## Decoder Nx



# Example





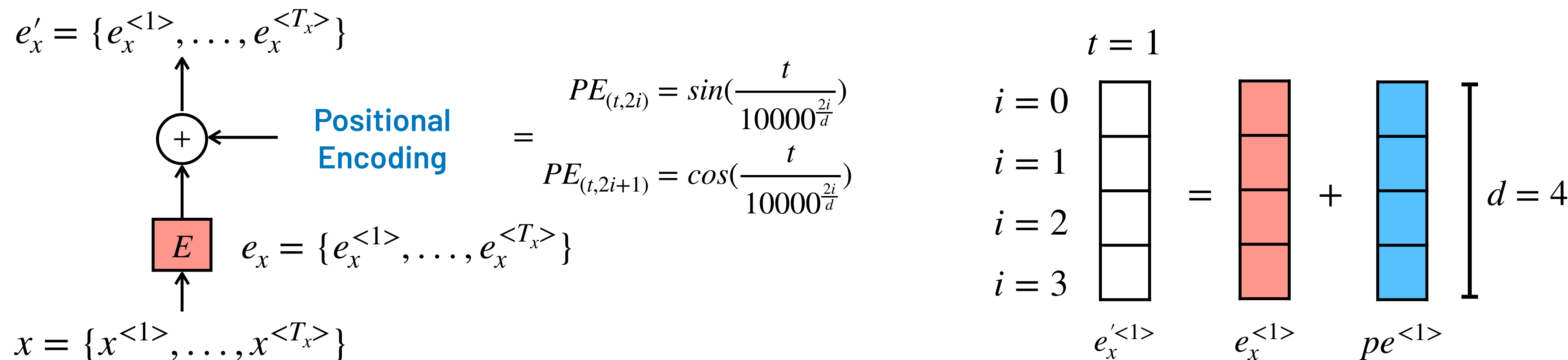
# Positional Encoding

The self-attention mechanism does not consider the position of the words.

C

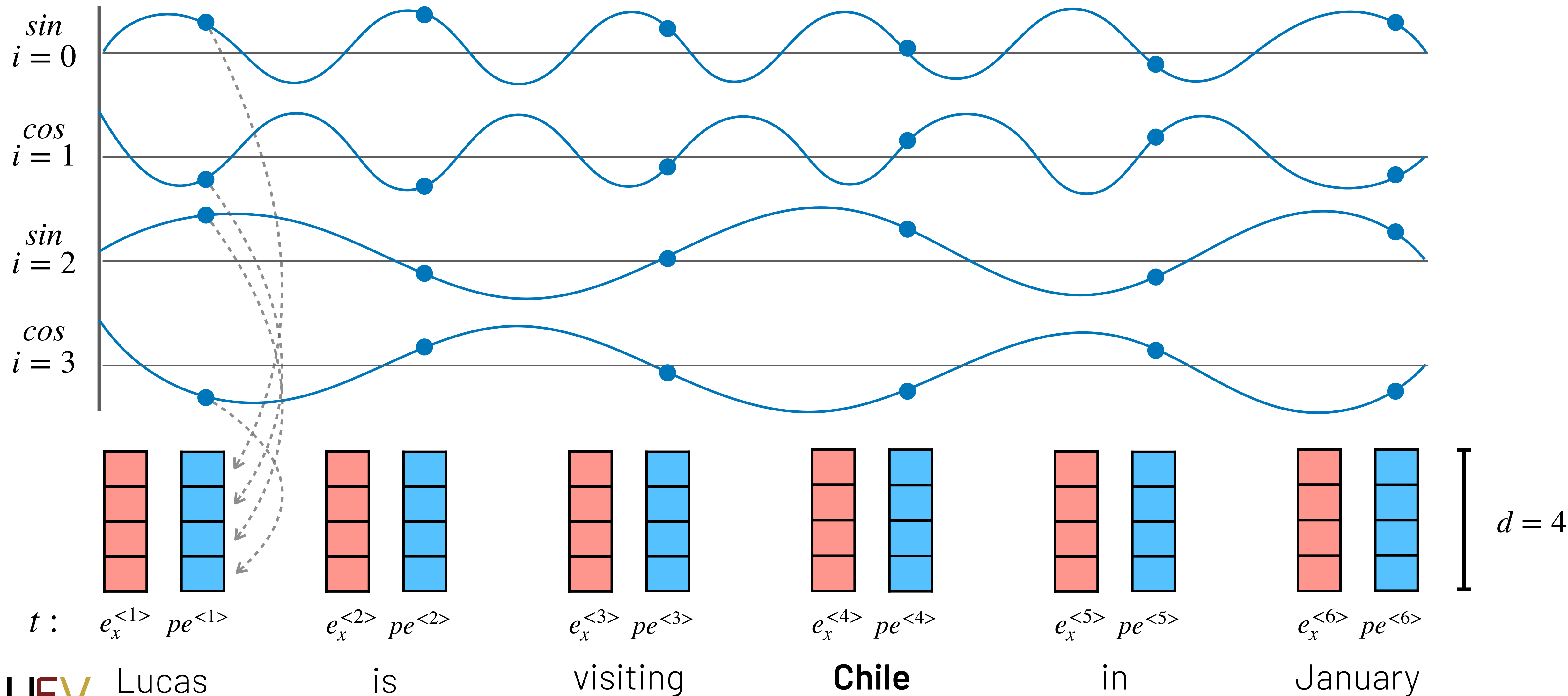
= Attention(Q, K, V) = softmax( $\frac{QK^T}{\sqrt{(d_k)}}$ )V

To add this information to the learned contextual representation **C**, both encoder and decoder add an positional information to each element  $x^{<t>}$  of the input  $x = \{x^{<1>}, \dots, x^{<T_x>}\}$



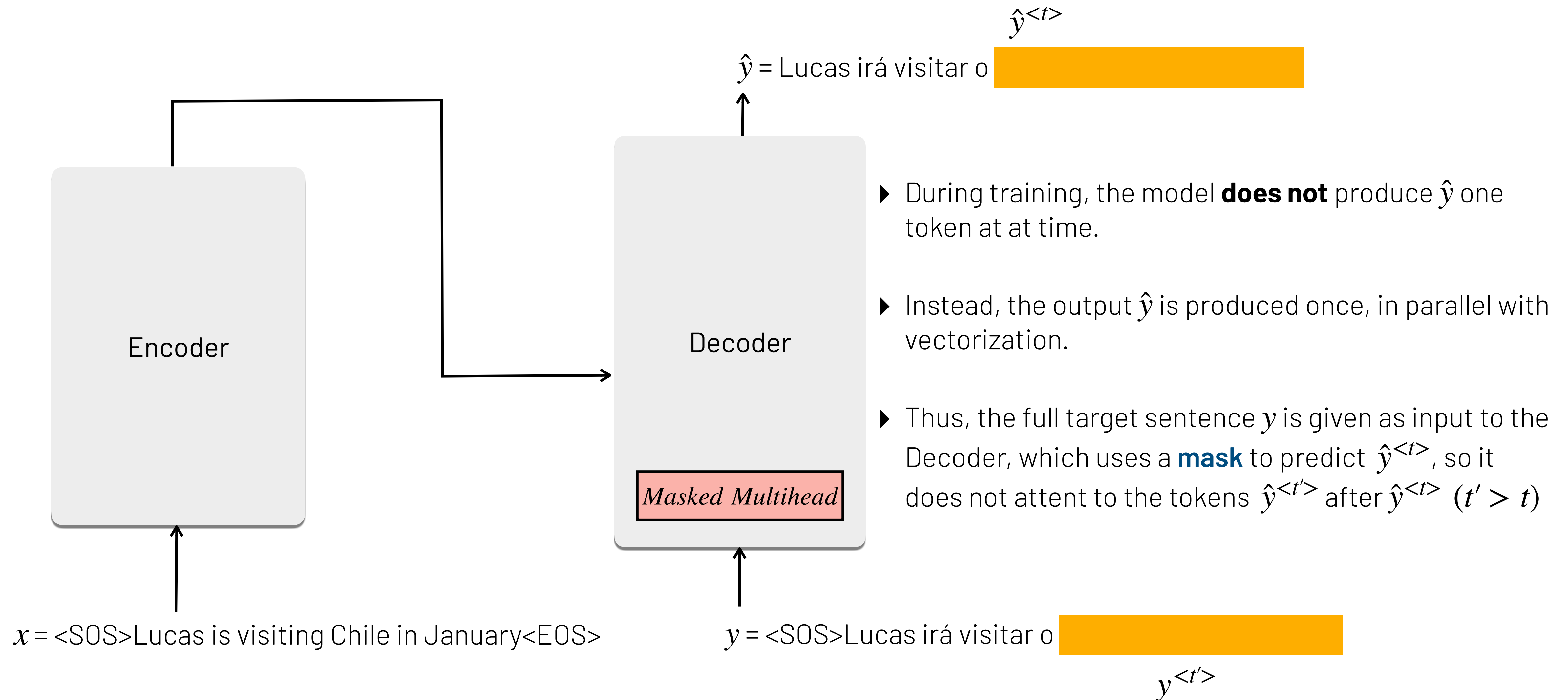
# Positional Encoding

$$\begin{aligned} \text{if } i \text{ is even} \quad & PE_{(t,2i)} = \sin\left(\frac{t}{10000^{\frac{2i}{d}}}\right) \\ \text{if } i \text{ is odd} \quad & PE_{(t,2i+1)} = \cos\left(\frac{t}{10000^{\frac{2i}{d}}}\right) \end{aligned}$$





# Training with Masked Multi-head Attention



# Treinamento (Masked Multi-head Attention)

$Q$   $K^T$  **Mask**

$\times$   $=$   $+$   $= \text{softmax}(\frac{\sqrt{(d_k)}}{}$   $) =$

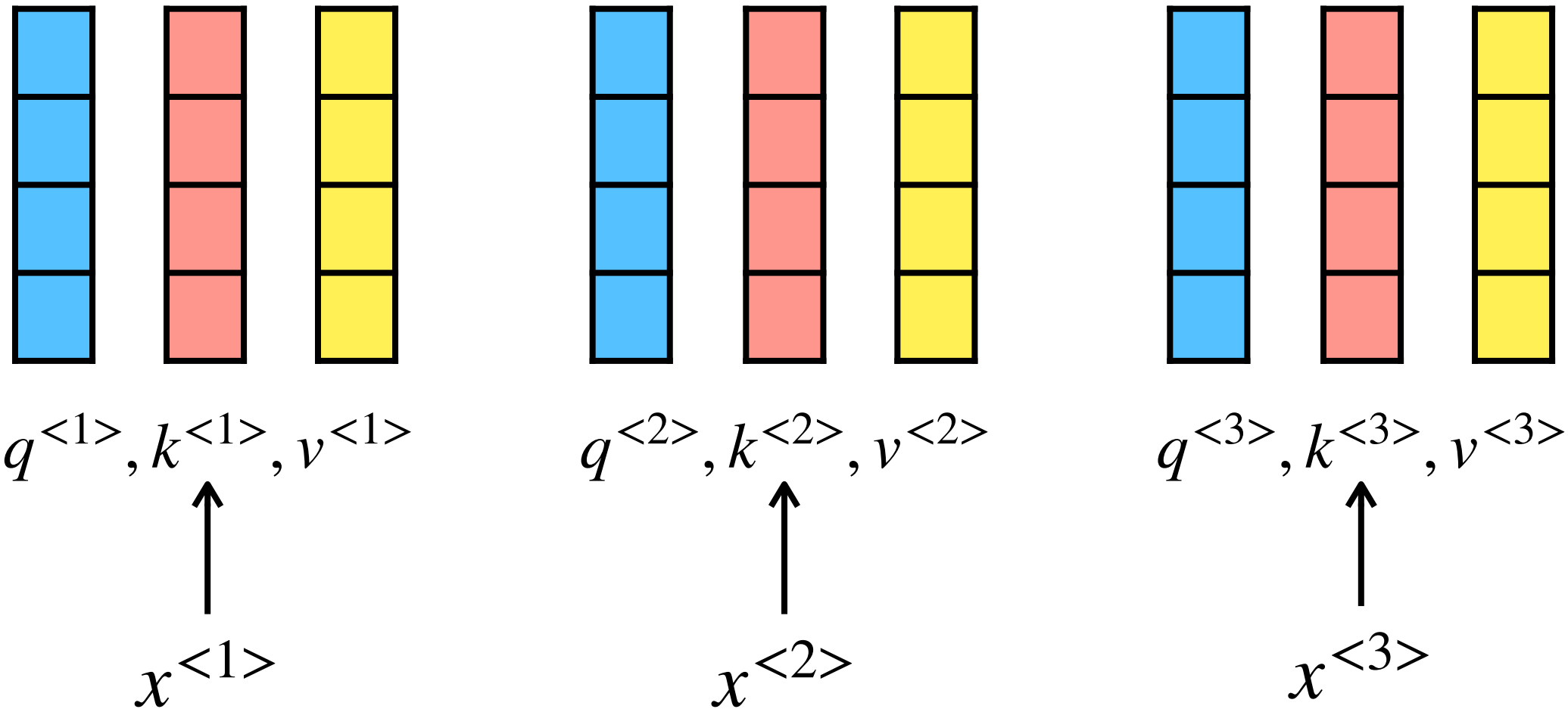
31	27	22
27	42	31
22	31	37

0	-inf	-inf
0	0	-inf
0	0	0

16	-inf	-inf
14	23	-inf
9	12	17

.1	.0	.0
.3	.7	.0
.3	.3	.4

Attention weights (normalized by row) of work  $i$  to word  $j$  after mask application



# Next Lecture

**L17:** Transformers (Part II)

Case studies of transformers: BERT and GPT