INF721

2024/2

UFV

Deep Learning

L17: Transformers

Logistics

Last Lecture

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

UFV

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:

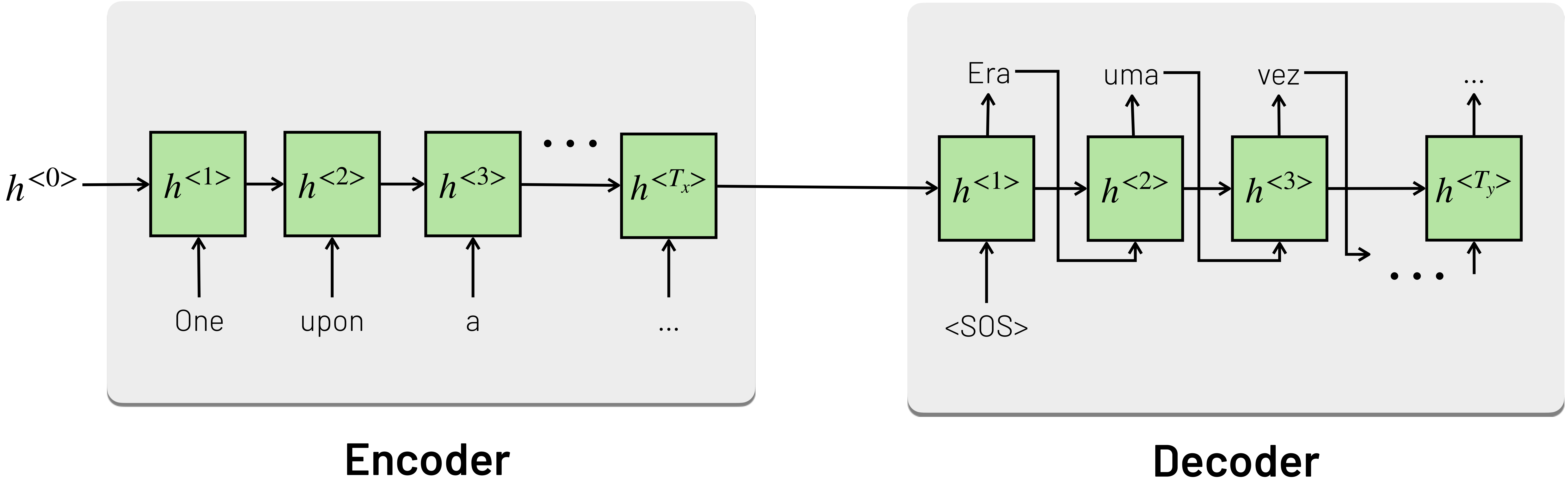
$(x = \{x^{<1>}, x^{<2>}, \ldots, x^{<T_x>}\}, y = \{y^{<1>}, y^{<2>}, \ldots, y^{<T_y>}\})$,

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



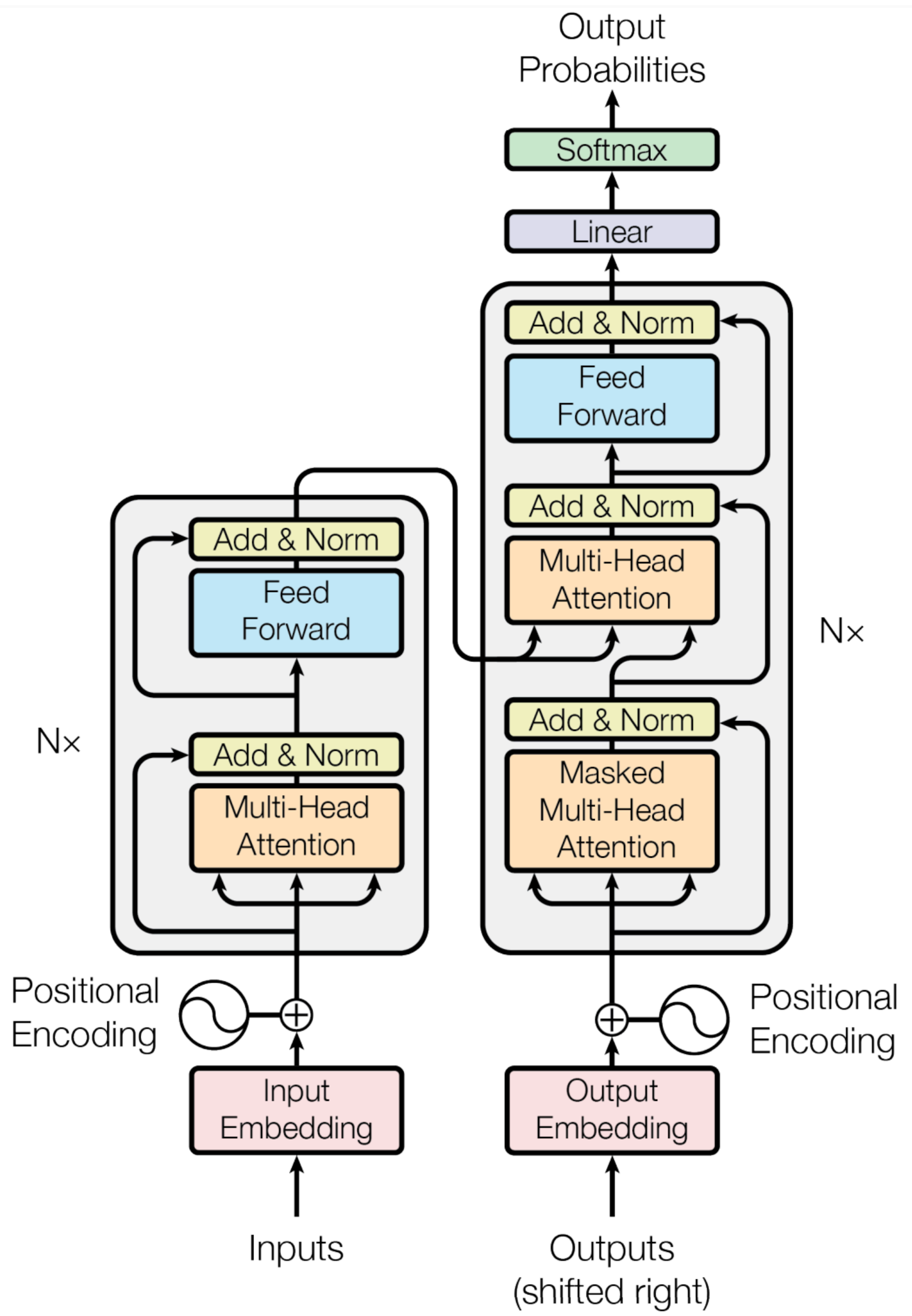
UFV

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:

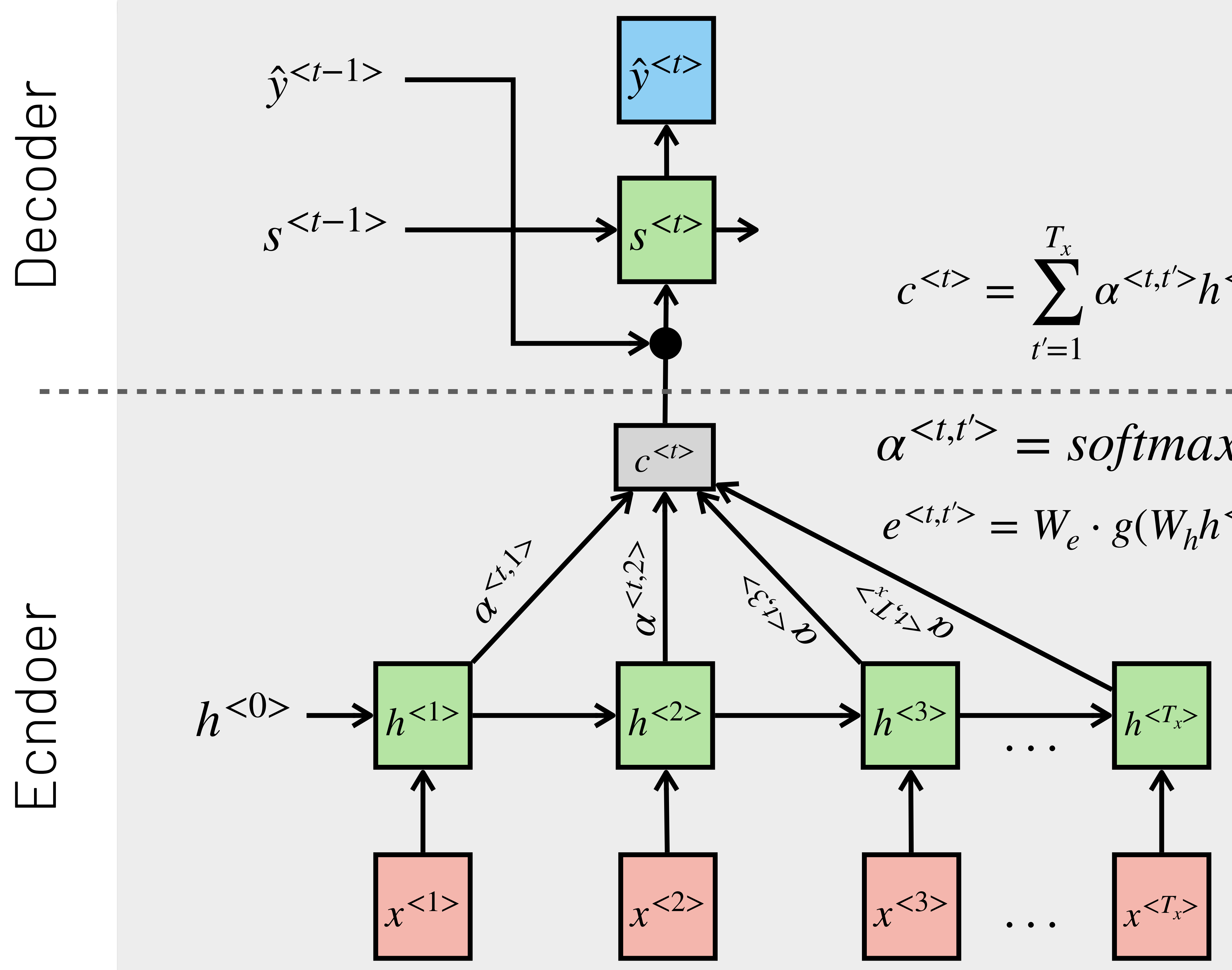
- Natural Language Processing
- Computer Vision
- Reinforcement Learning
- ...



Vaswani et al. 2017, Attention Is All You Need

Attention in RNNs vs. Transformers

RNNs Badahnau Attention

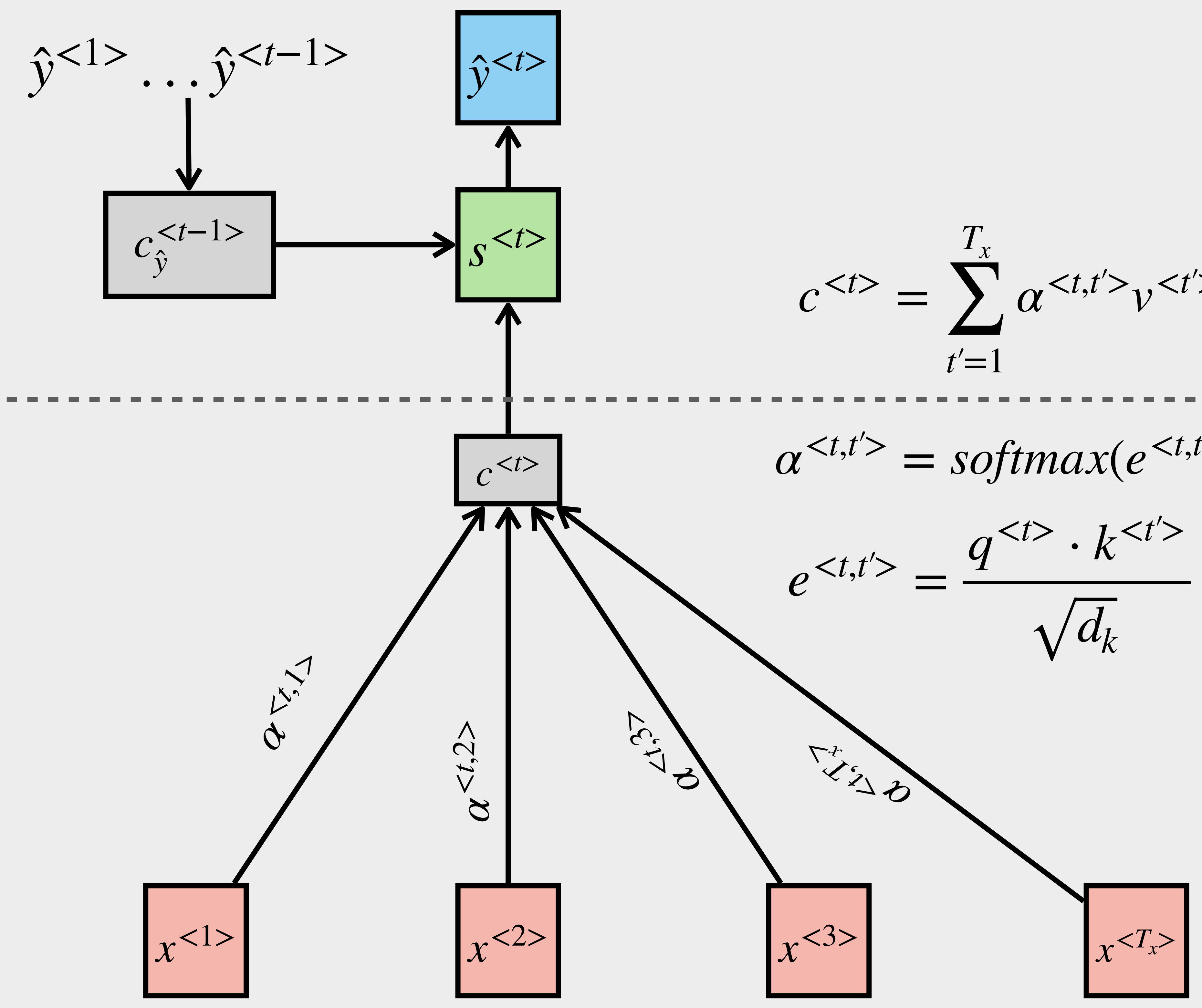


$$c^{<t>} = \sum_{t'=1}^{T_x} \alpha^{<t,t'>} h^{<t'>}$$

$$\alpha^{<t,t'>} = softmax(e^{<t,t'>})$$

$$e^{<t,t'>} = W_e \cdot g(W_h h^{<t'>} + W_s s^{<t-1>})$$

Transformers Scaled Dot-Product Attention



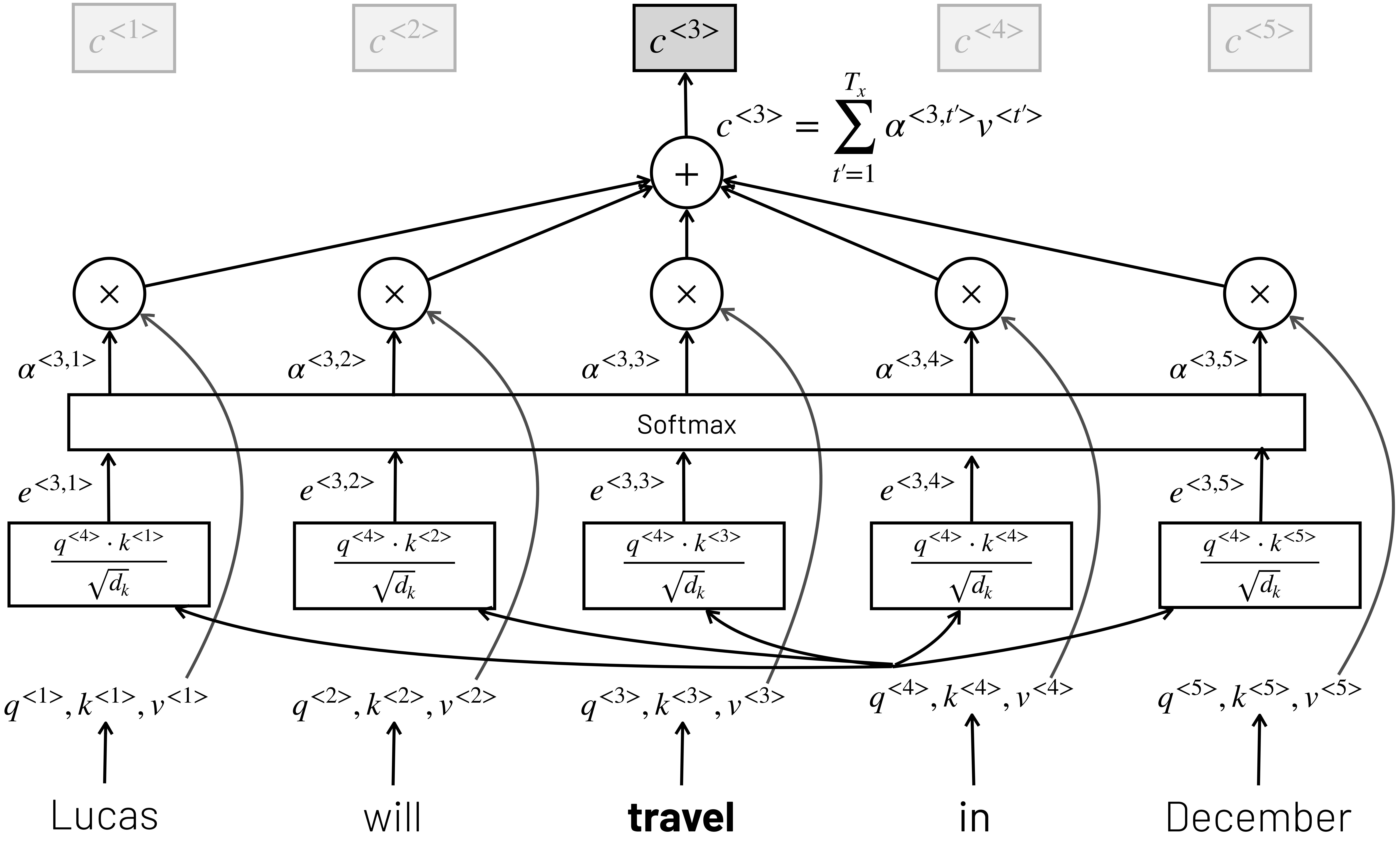
$$c^{<t>} = \sum_{t'=1}^{T_x} \alpha^{<t,t'>} v^{<t'>}$$

$$\alpha^{<t,t'>} = softmax(e^{<t,t'>})$$

$$e^{<t,t'>} = \frac{q^{<t>} \cdot k^{<t'>}}{\sqrt{d_k}}$$

Self-Attention

The key idea behing 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.

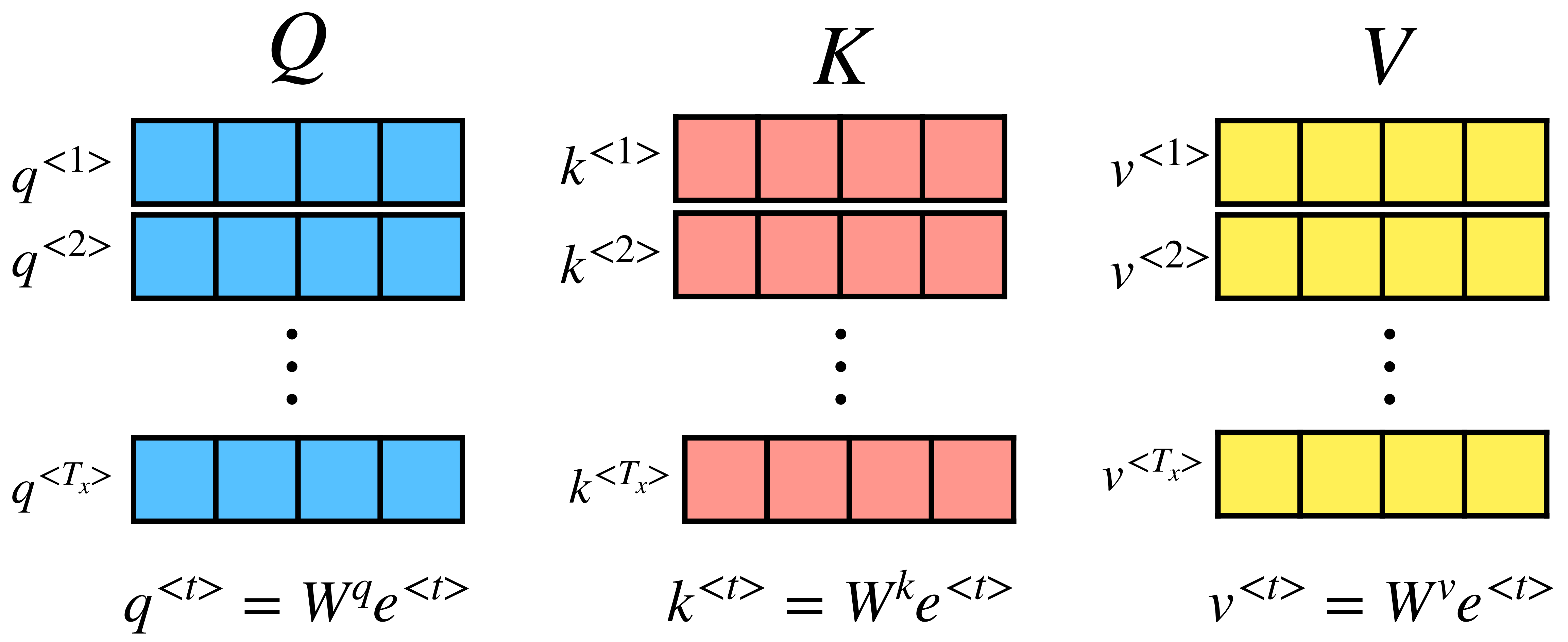


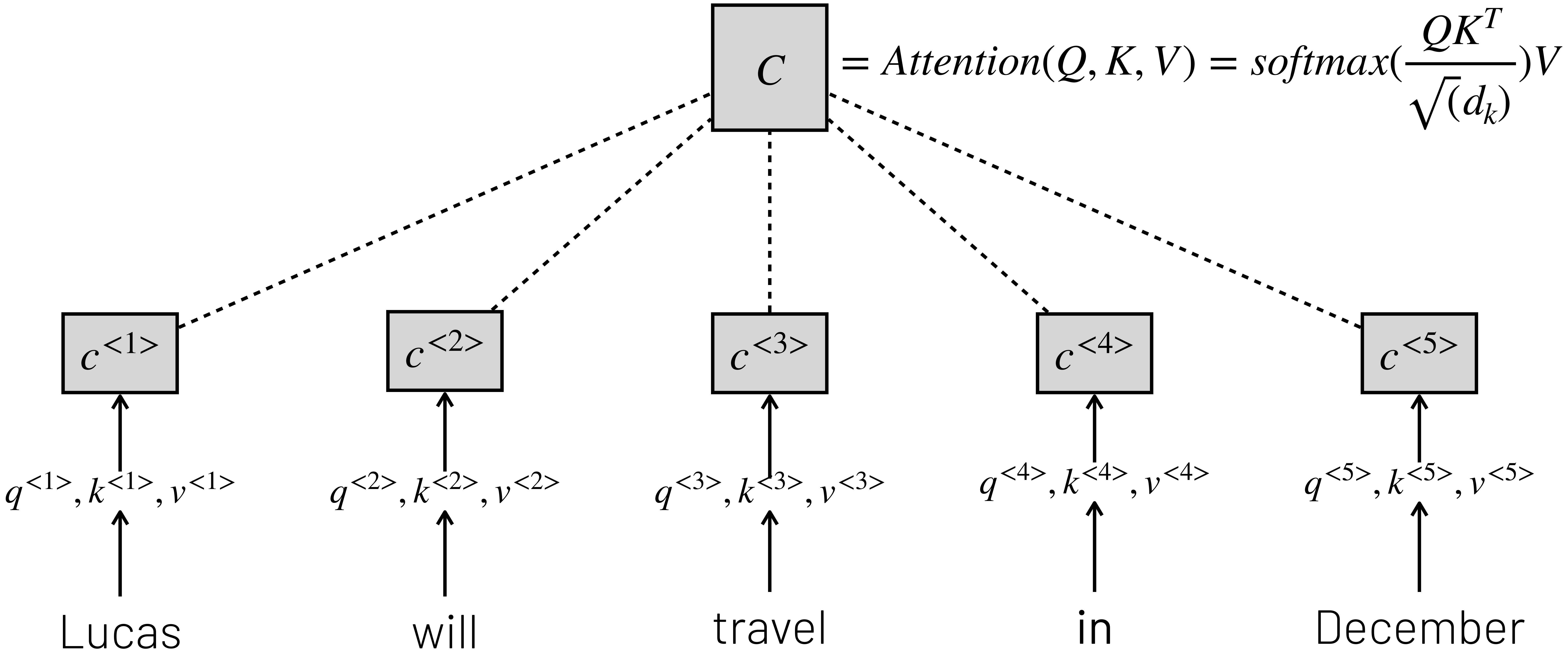
$$q^{<t'>} = W^q e^{<t'>}$$
$$k^{<t'>} = W^k e^{<t'>}$$
$$v^{<t'>} = W^v e^{<t'>}$$

UFV

Self-Attention

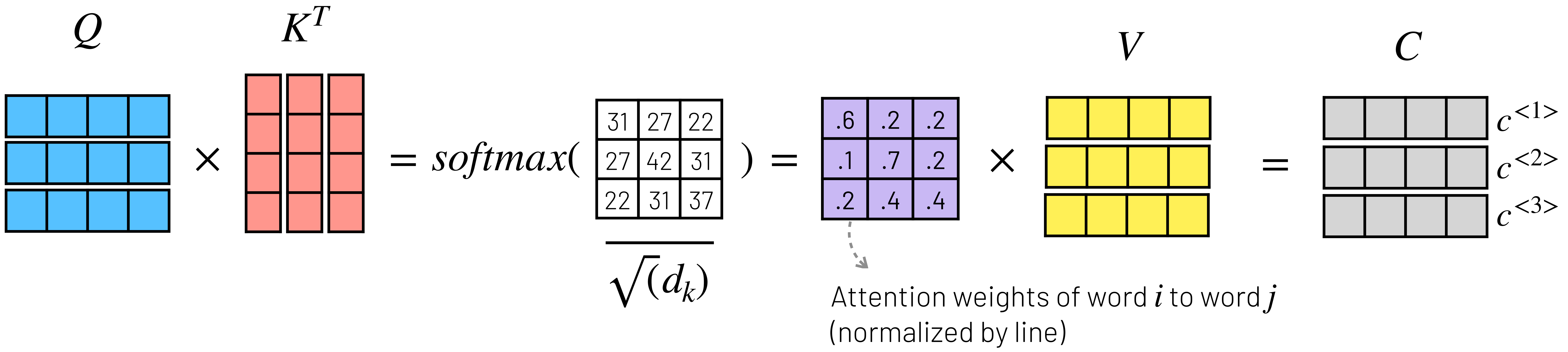
The contextal represention $C = \{c^{<1>}, \ldots, c^{<T_x>}\}$ of the entire input sequence $x = \{x^{<1>}, \ldots, x^{<T_x>}\}$ can be computed in a vectorized way combining vectors $q^{<t>}$, $k^{<t>}$, $v^{<t>}$ in matrices Q, K e V

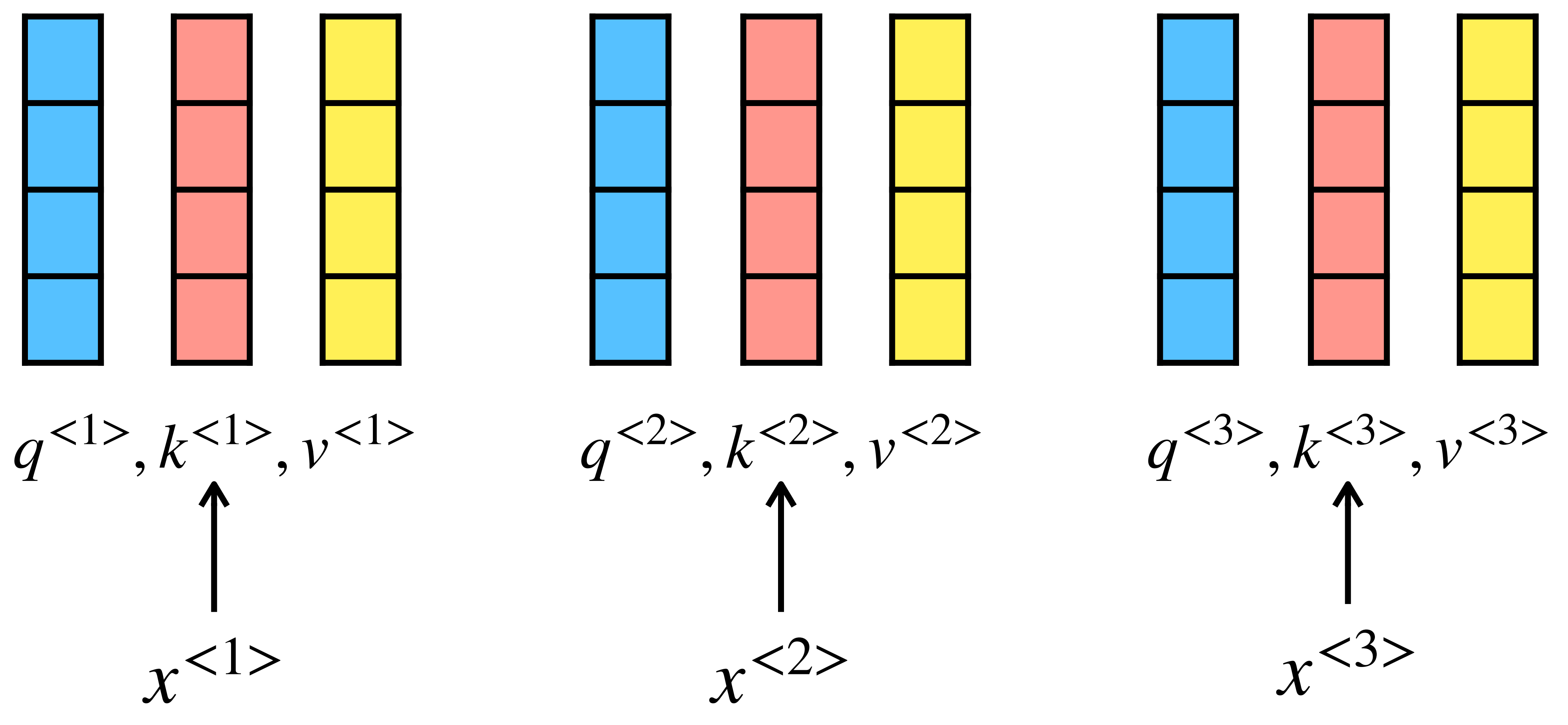




Self-Attention

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

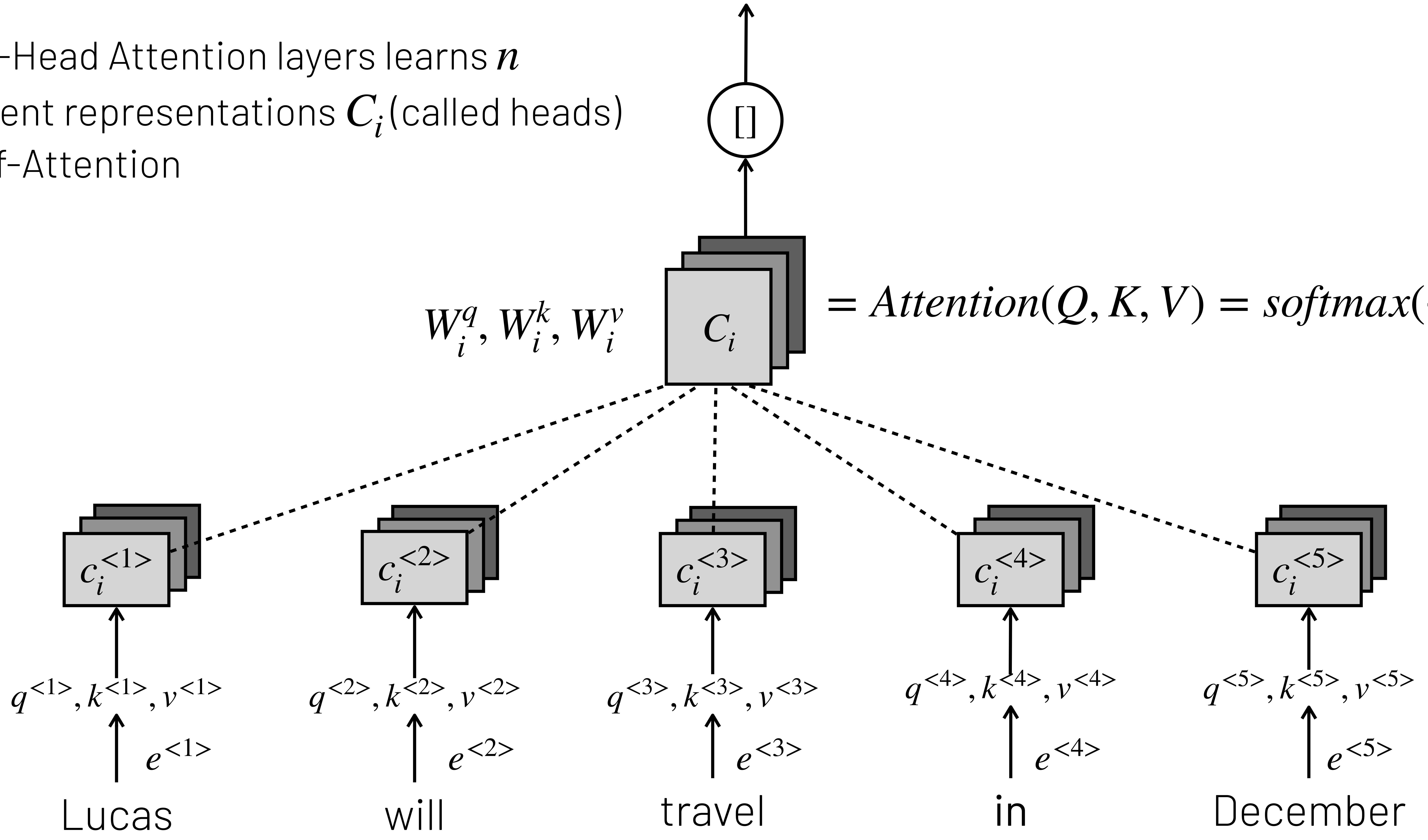




Multi-Head Attention

$$Multihead(Q, K, V) = Concat(C_1, C_2, \ldots, C_h)W_o$$

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



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

UFV

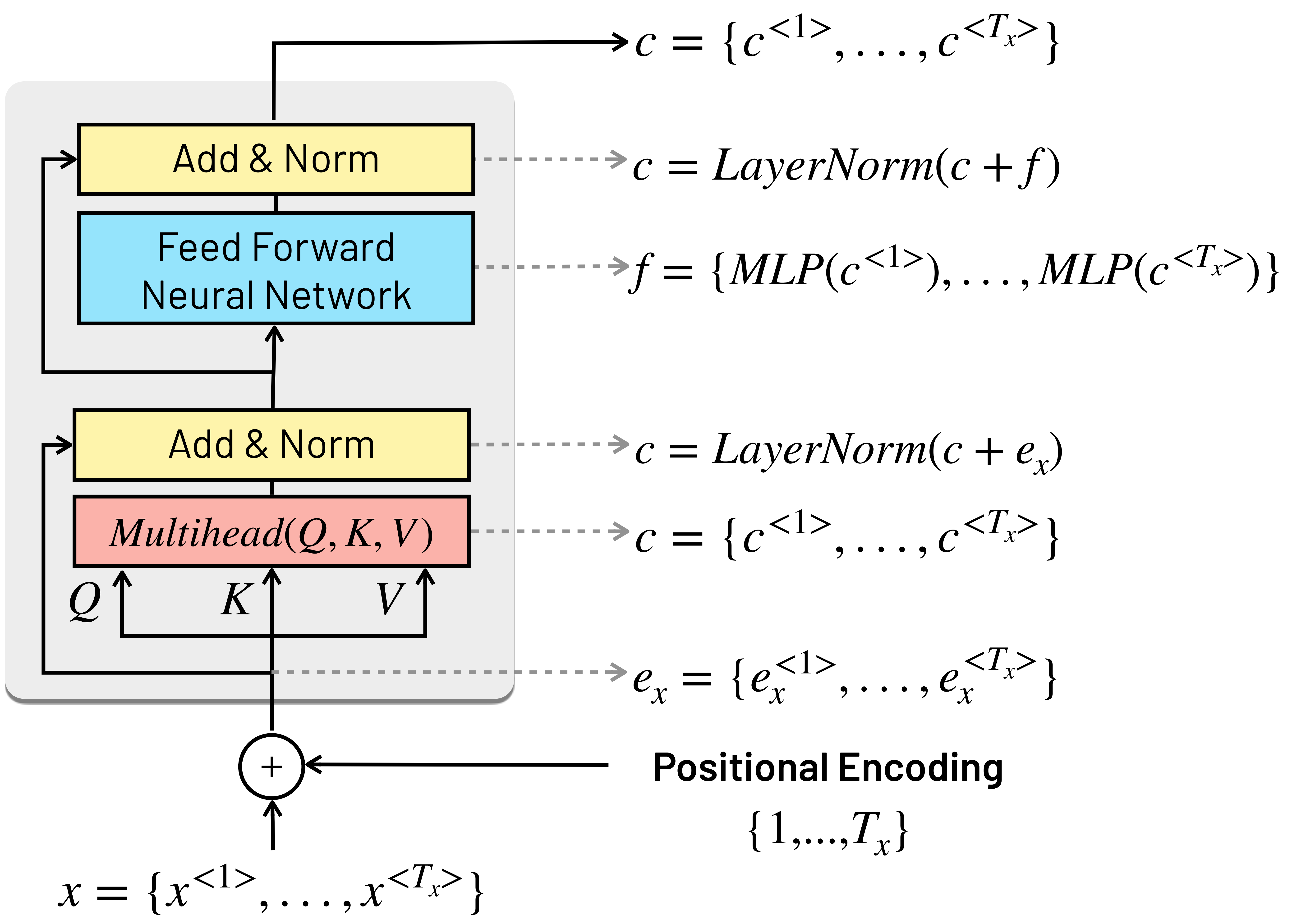
Encoder

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

Ouput: 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 conneced with a residual connection (***Add***)

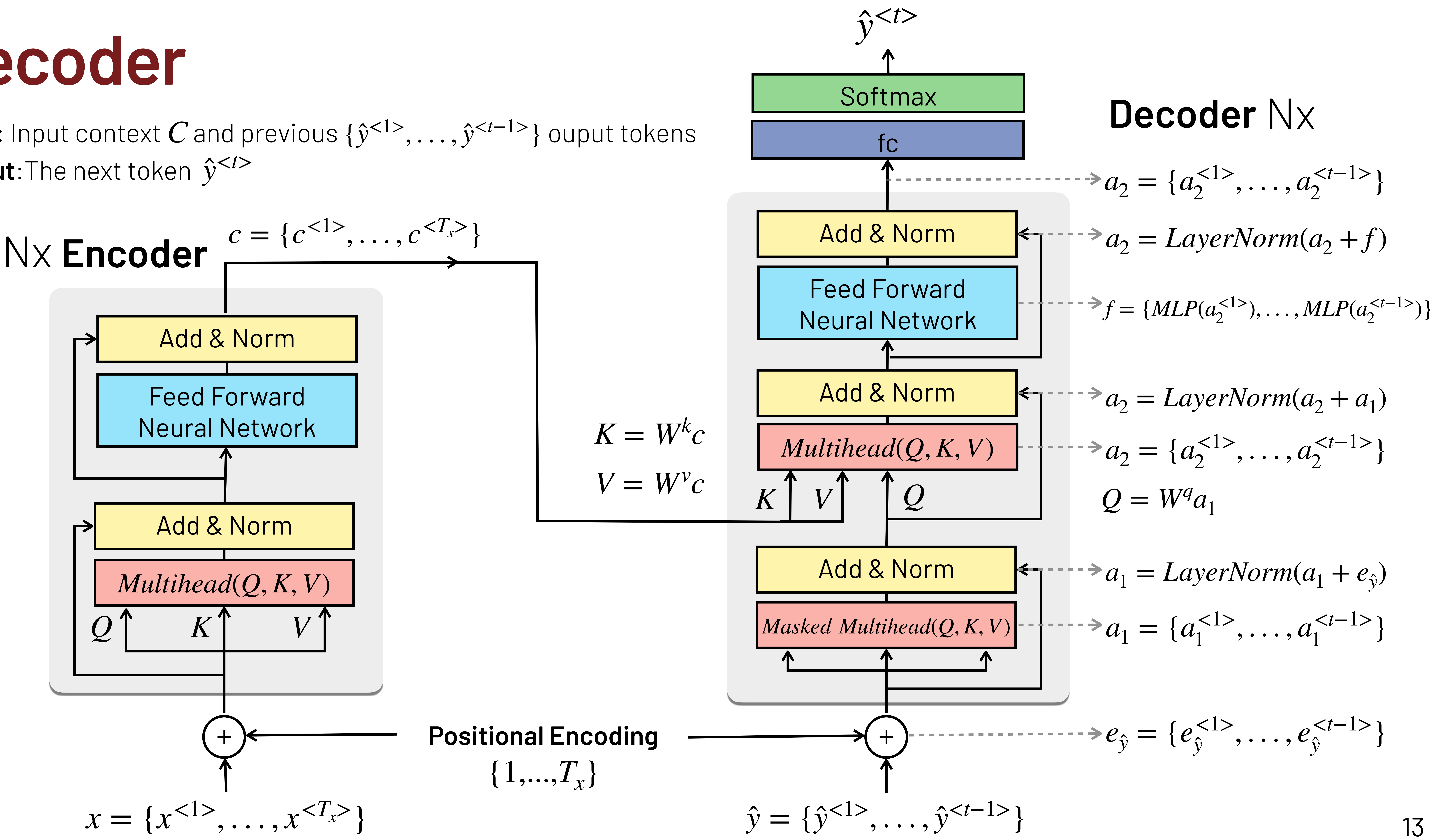


UFV

Decoder

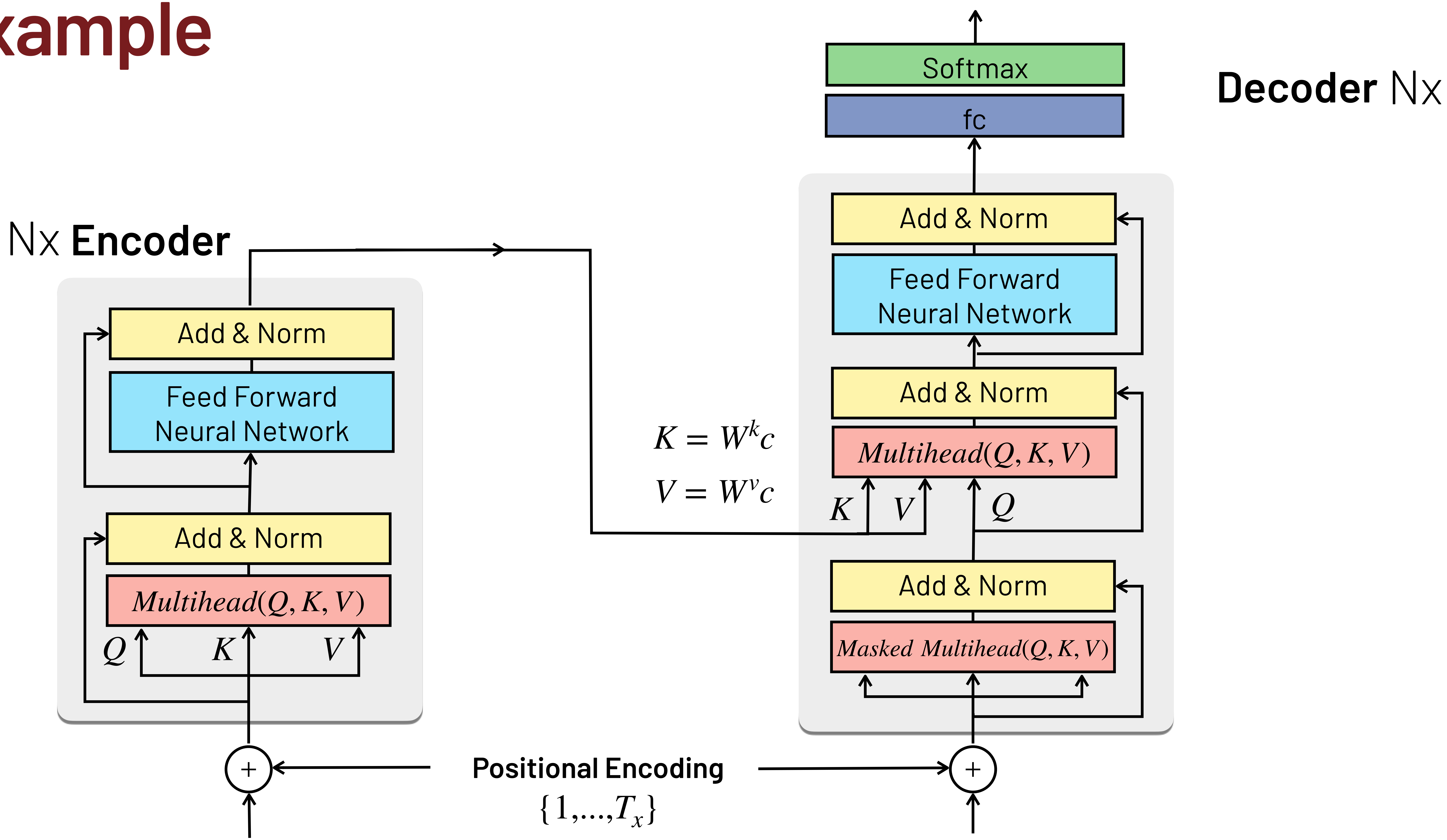
Input: Input context C and previous $\{\hat{y}^{<1>}, \ldots, \hat{y}^{<t-1>}\}$ ouput tokens

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



UFV

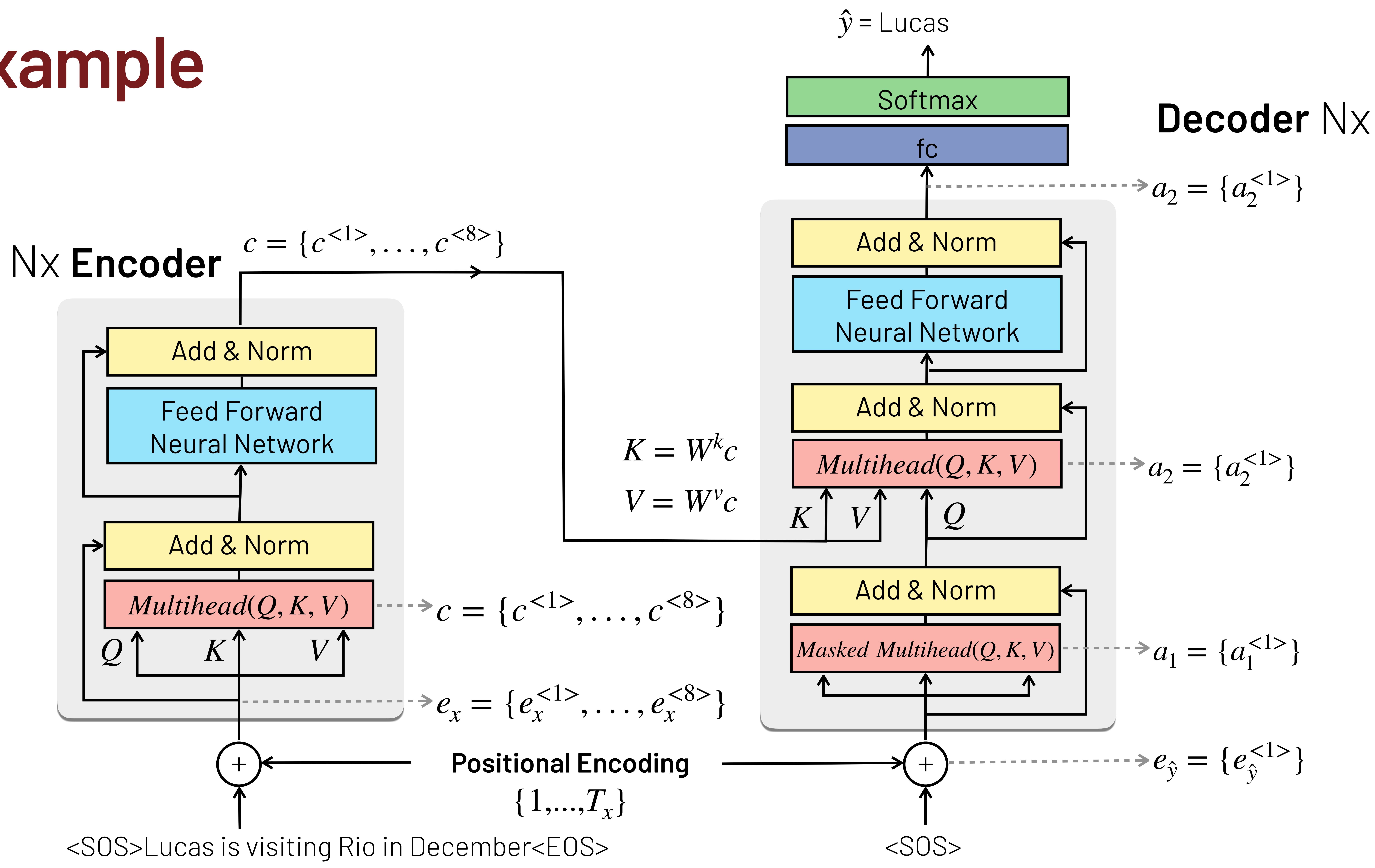
Example



<SOS>Lucas is visiting Rio in December<EOS>

UFV

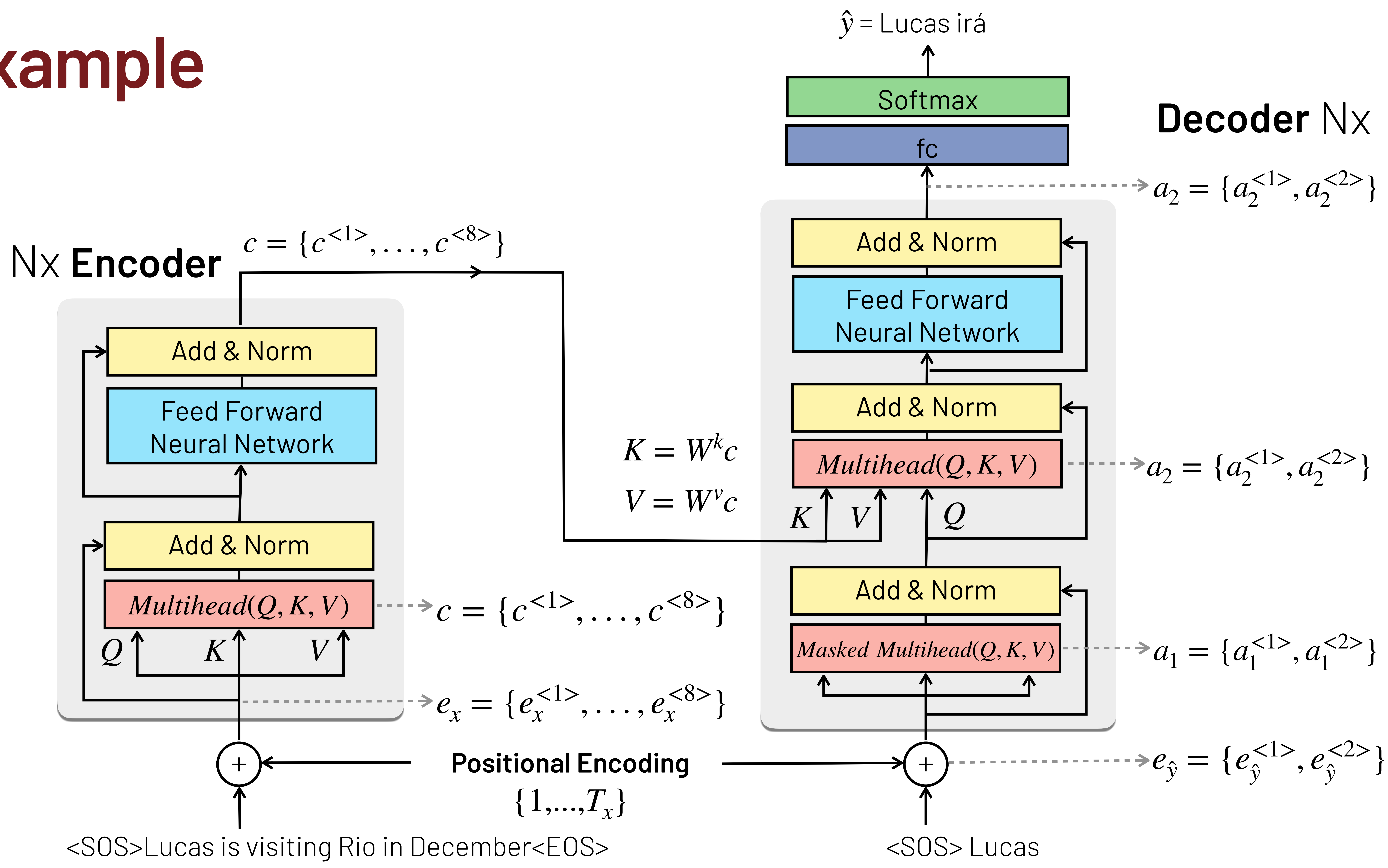
Example



$\hat{y}$ = Lucas
Softmax
fc
Decoder Nx
$a_2 = \{a_2^{<1>}\}$
Nx Encoder
$c = \{c^{<1>}, \ldots, c^{<8>}\}$
Add & Norm
Feed Forward Neural Network
Add & Norm
$Multihead(Q, K, V)$
Q K V
$c = \{c^{<1>}, \ldots, c^{<8>}\}$
$e_x = \{e_x^{<1>}, \ldots, e_x^{<8>}\}$
$K = W^k c$
$V = W^v c$
Add & Norm
Feed Forward Neural Network
Add & Norm
$Multihead(Q, K, V)$
K V Q
$a_2 = \{a_2^{<1>}\}$
Add & Norm
$Masked\ Multihead(Q, K, V)$
$a_1 = \{a_1^{<1>}\}$
$e_{\hat{y}} = \{e_{\hat{y}}^{<1>}\}$
Positional Encoding
$\{1, \ldots, T_x\}$
<SOS>Lucas is visiting Rio in December<EOS>
<SOS>

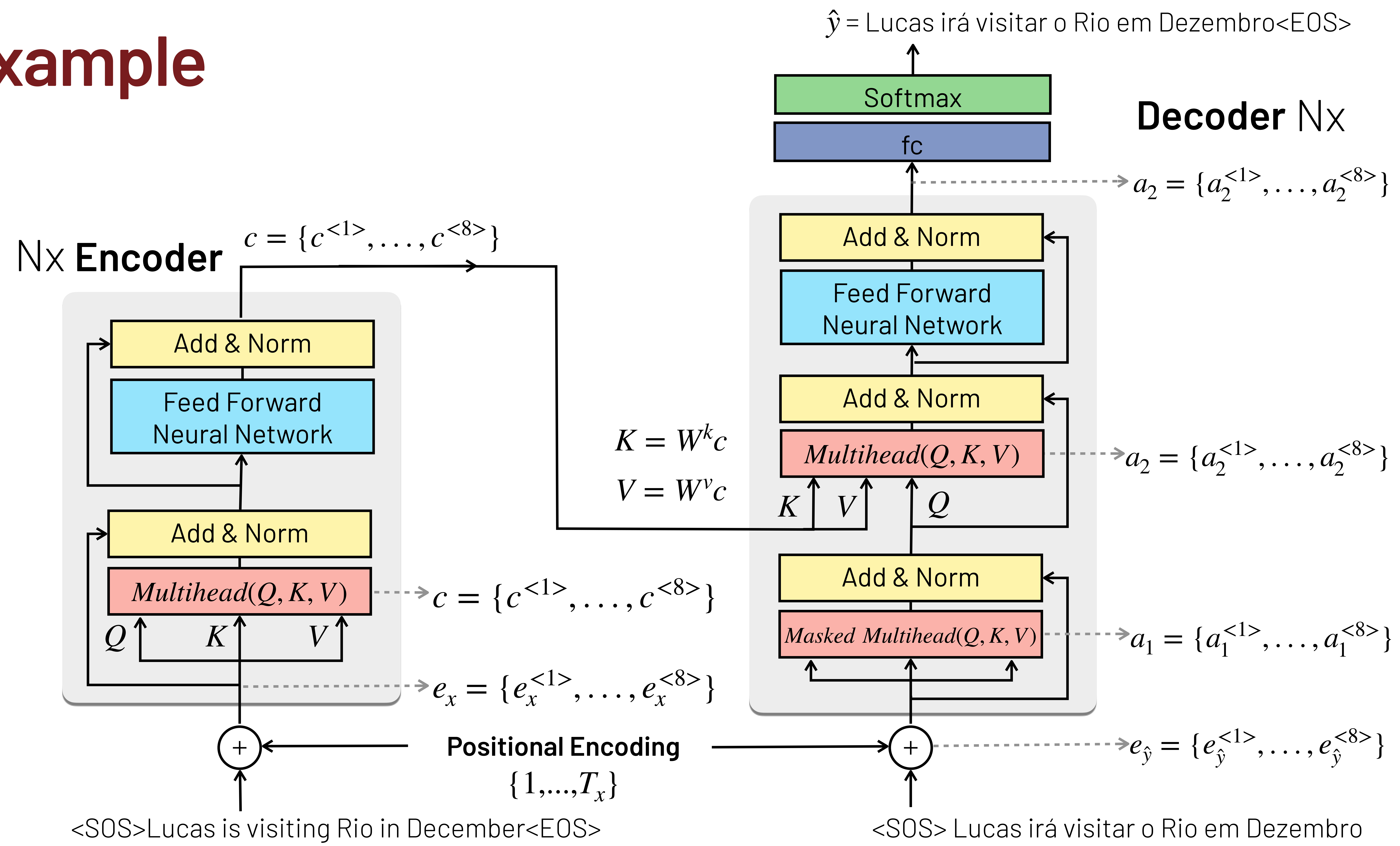
UFV

Example



$\hat{y}$ = Lucas irá
Softmax
fc
Decoder Nx
$a_2 = \{a_2^{<1>}, a_2^{<2>}\}$
Nx Encoder
$c = \{c^{<1>}, \ldots, c^{<8>}\}$
Add & Norm
Feed Forward Neural Network
Add & Norm
$Multihead(Q, K, V)$
Q
K
V
$c = \{c^{<1>}, \ldots, c^{<8>}\}$
$e_x = \{e_x^{<1>}, \ldots, e_x^{<8>}\}$
$K = W^k c$
$V = W^v c$
Add & Norm
Feed Forward Neural Network
Add & Norm
$Multihead(Q, K, V)$
$a_2 = \{a_2^{<1>}, a_2^{<2>}\}$
K
V
Q
Add & Norm
$Masked\ Multihead(Q, K, V)$
$a_1 = \{a_1^{<1>}, a_1^{<2>}\}$
Positional Encoding
$\{1, \ldots, T_x\}$
$e_{\hat{y}} = \{e_{\hat{y}}^{<1>}, e_{\hat{y}}^{<2>}\}$
<SOS>Lucas is visiting Rio in December<EOS>
<SOS> Lucas

Example



$\hat{y}$ = Lucas irá visitar o Rio em Dezembro<EOS>
Softmax
fc
Decoder Nx
$a_2 = \{a_2^{<1>}, \ldots, a_2^{<8>}\}$
Nx Encoder
$c = \{c^{<1>}, \ldots, c^{<8>}\}$
Add & Norm
Feed Forward Neural Network
Add & Norm
Feed Forward Neural Network
$K = W^k c$
$V = W^v c$
Add & Norm
$Multihead(Q, K, V)$
$a_2 = \{a_2^{<1>}, \ldots, a_2^{<8>}\}$
K V Q
Add & Norm
$Multihead(Q, K, V)$
$c = \{c^{<1>}, \ldots, c^{<8>}\}$
Add & Norm
$Masked\ Multihead(Q, K, V)$
$a_1 = \{a_1^{<1>}, \ldots, a_1^{<8>}\}$
Q K V
$e_x = \{e_x^{<1>}, \ldots, e_x^{<8>}\}$
Positional Encoding
$\{1, \ldots, T_x\}$
$e_{\hat{y}} = \{e_{\hat{y}}^{<1>}, \ldots, e_{\hat{y}}^{<8>}\}$
<SOS>Lucas is visiting Rio in December<EOS>
<SOS> Lucas irá visitar o Rio em Dezembro

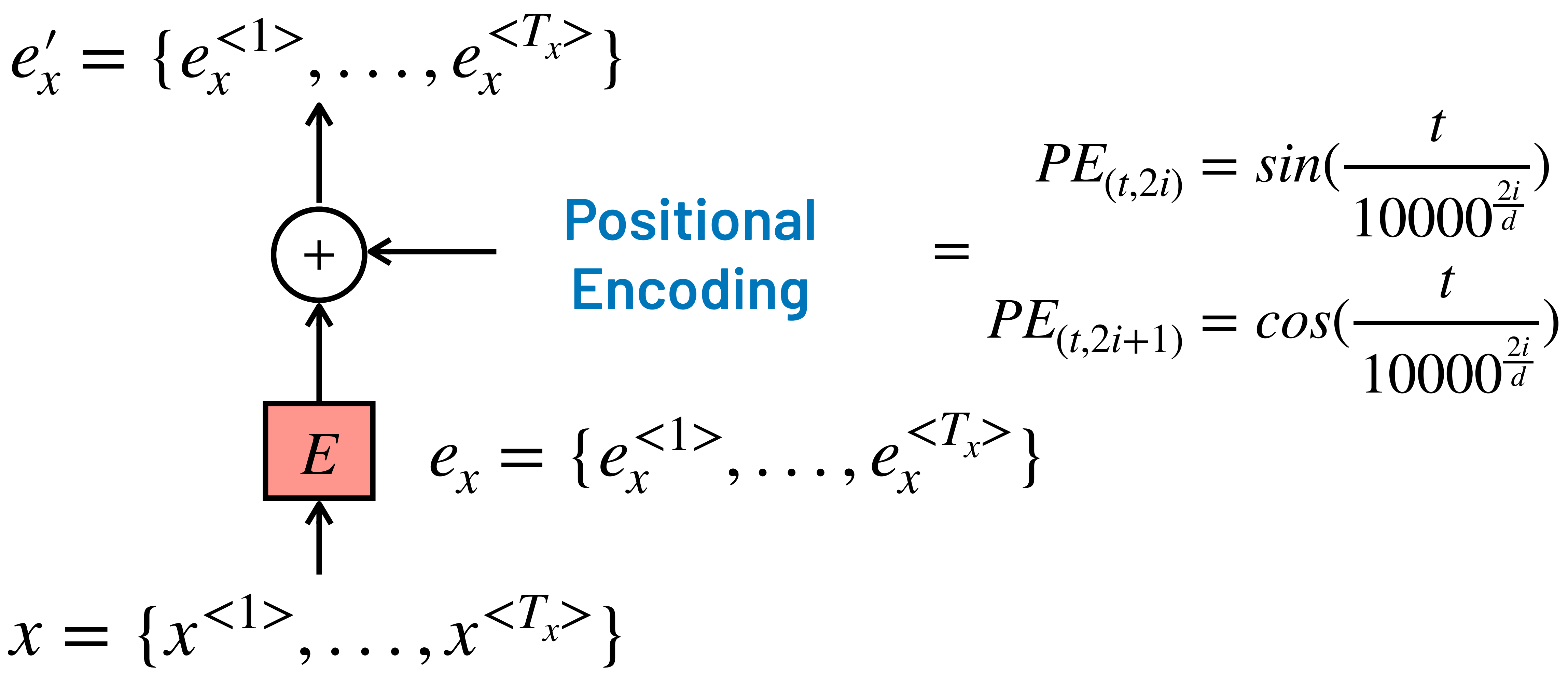
UFV

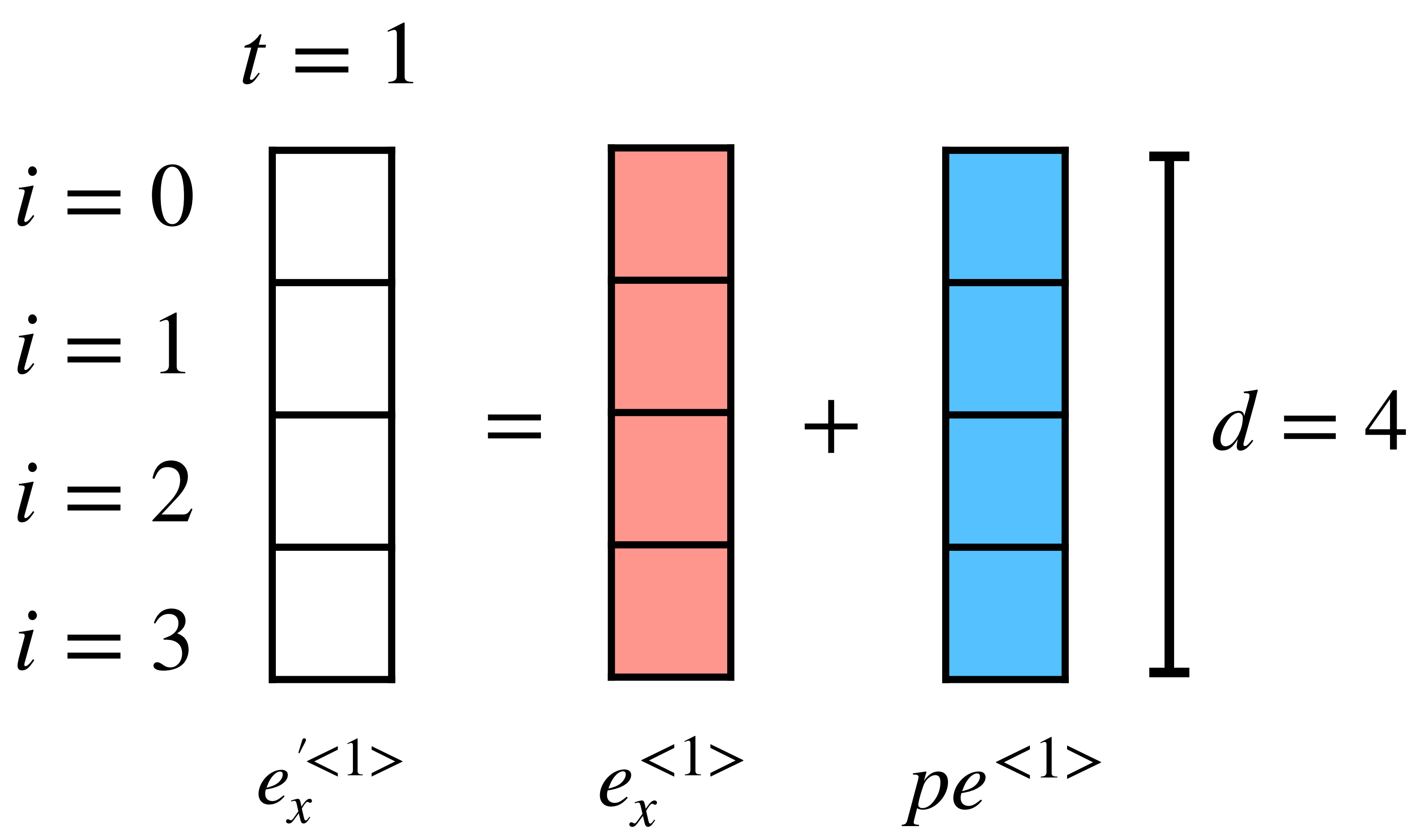
Positional Encoding

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

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

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 $x = \{x^{<1>}, \ldots, x^{<T_x>}\}$





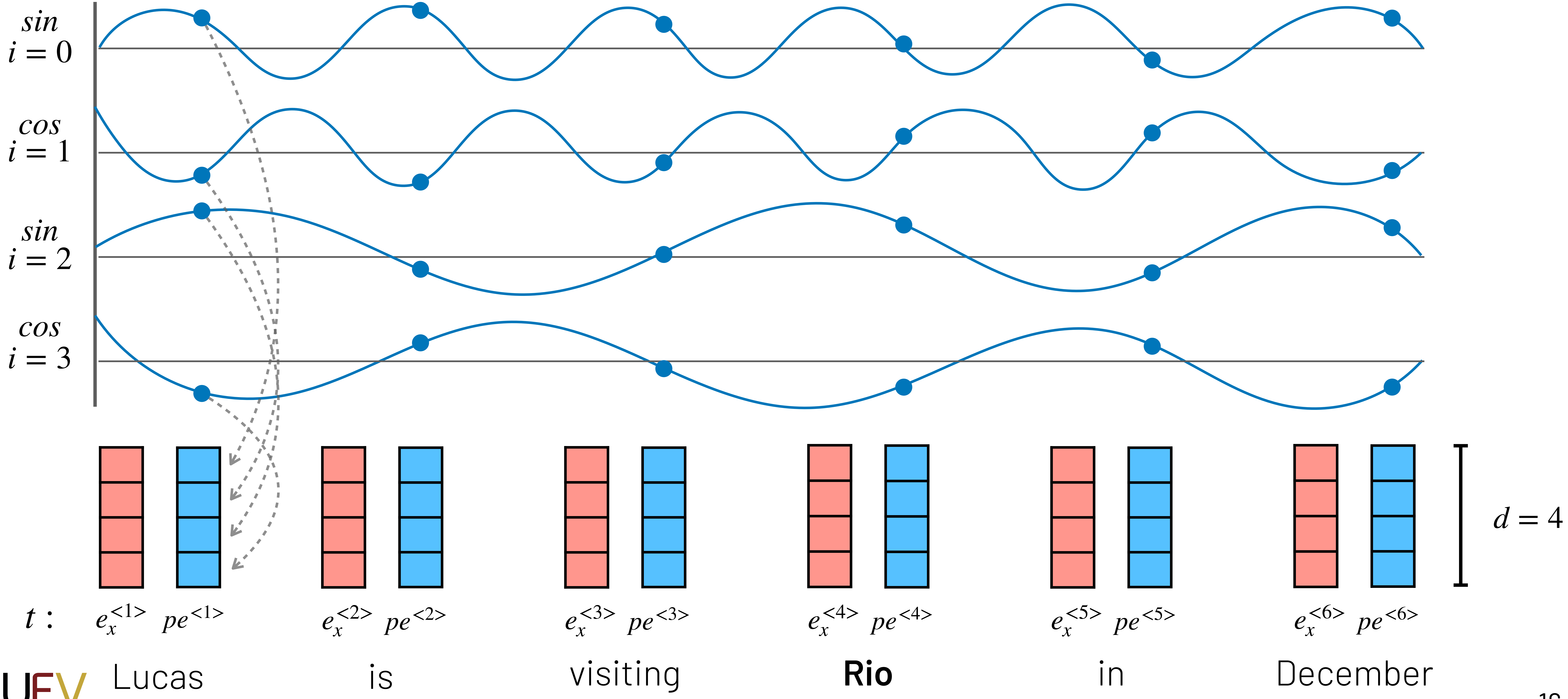
Positional Encoding

if i is even

$$PE_{(t,2i)} = sin(\frac{t}{10000^{\frac{2i}{d}}})$$

if i is odd

$$PE_{(t,2i+1)} = cos(\frac{t}{10000^{\frac{2i}{d}}})$$

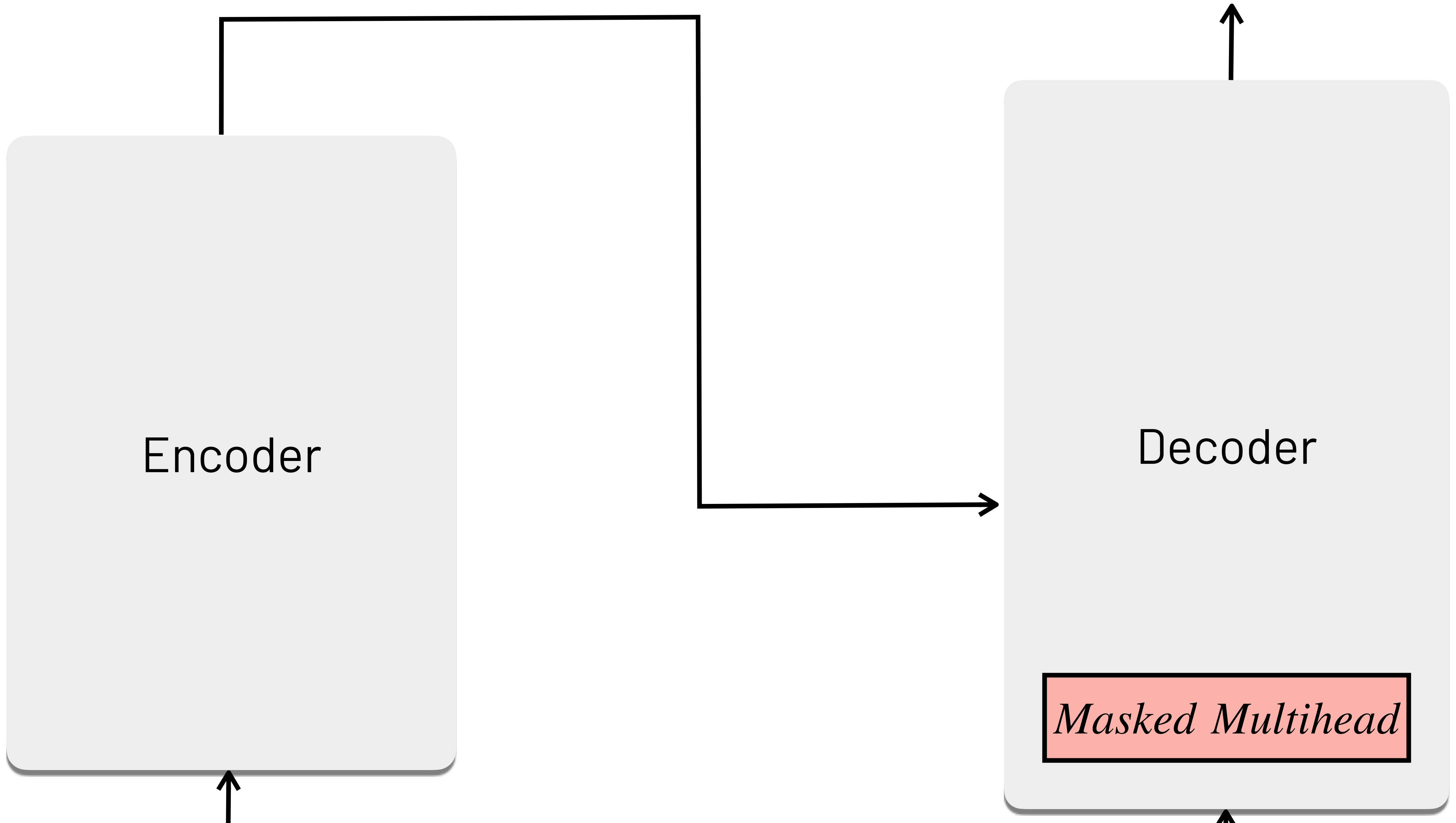


UFV

Training with Masked Multi-head Attention

$\hat{y}^{<t>}$

$\hat{y}$ = Lucas irá visitar o



x = <SOS>Lucas is visiting Rio in December <EOS>

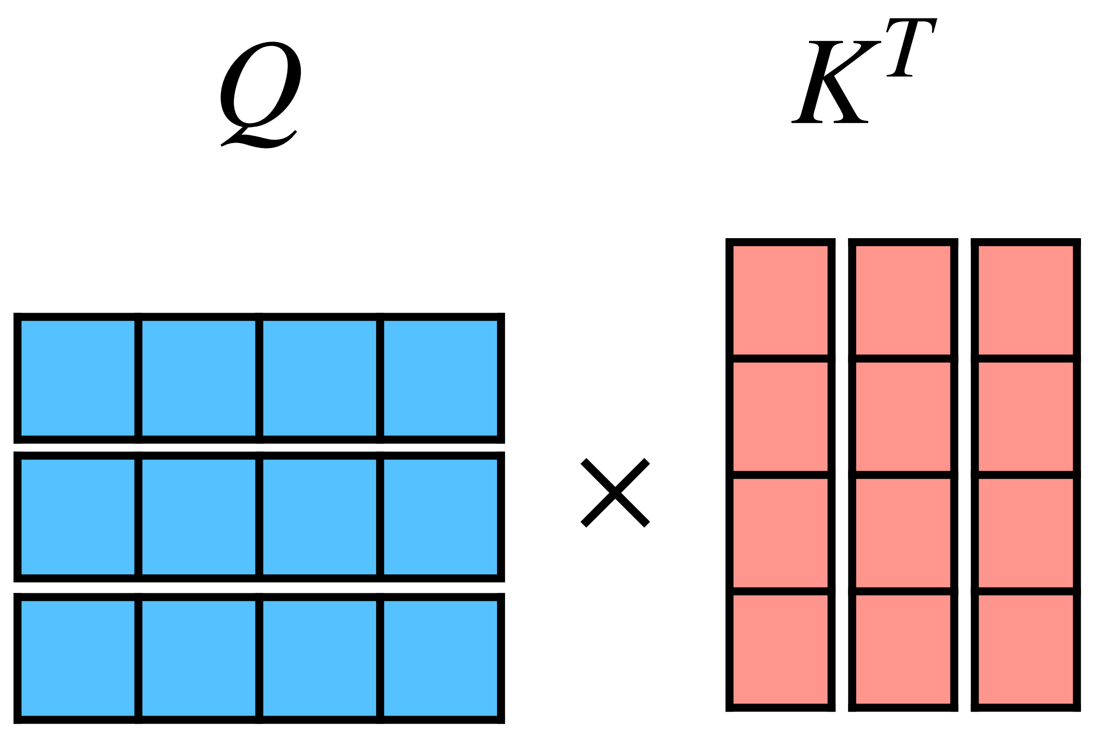
y = <SOS>Lucas irá visitar o

$y^{<t'>}$

- During training, the model **does not** produce $\hat{y}$ one token at at time.
- Instead, the output $\hat{y}$ is produced once, in parallel with vectorization.
- Thus, the full target sentence y is given as input to the Decoder, which uses a **mask** to predict $\hat{y}^{<t>}$, so it does not attent to the tokens $\hat{y}^{<t'>}$ after $\hat{y}^{<t>}$ $(t' > t)$

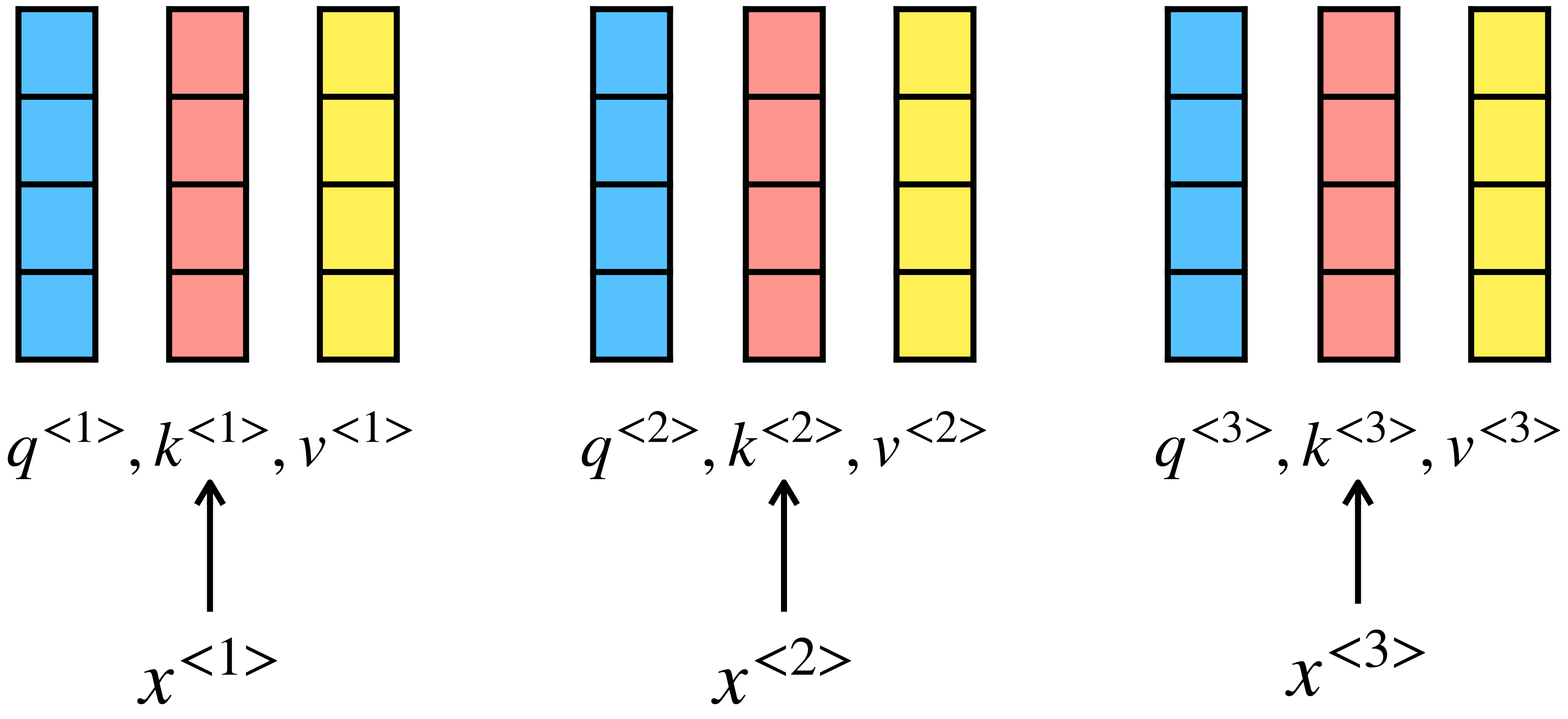
UFV

Treinamento (Masked Multi-head Attention)



$$\times \quad = \frac{\begin{bmatrix} 31 & 27 & 22 \\ 27 & 42 & 31 \\ 22 & 31 & 37 \end{bmatrix}}{\sqrt{(d_k)}} + \overset{\textbf{Mask}}{\begin{bmatrix} 0 & -\text{inf} & -\text{inf} \\ 0 & 0 & -\text{inf} \\ 0 & 0 & 0 \end{bmatrix}} = softmax\left(\begin{bmatrix} 16 & -\text{inf} & -\text{inf} \\ 14 & 23 & -\text{inf} \\ 9 & 12 & 17 \end{bmatrix} \right) = \begin{bmatrix} .1 & .0 & .0 \\ .3 & .7 & .0 \\ .3 & .3 & .4 \end{bmatrix}$$

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