INF721

2024/2

UFV

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:

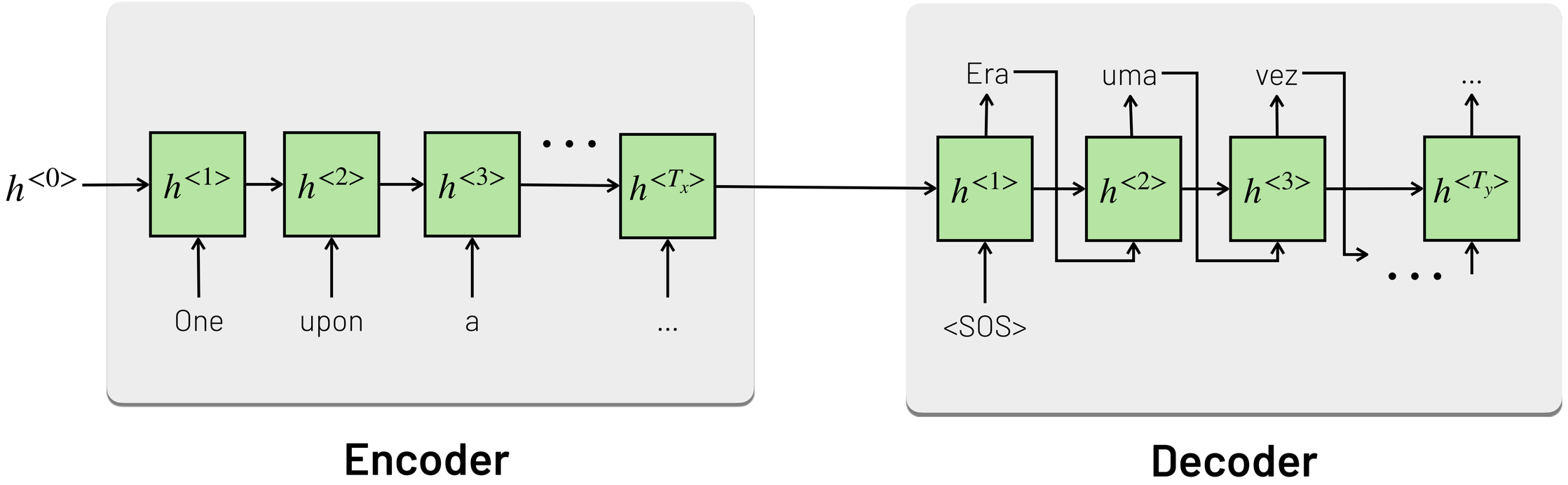
$$(x = \{x^{<1>}, x^{<2>}, \ldots, x^{<T_x>}\}, y = \{y^{<1>}, y^{<2>}, \ldots, y^{<T_y>}\}),$$

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

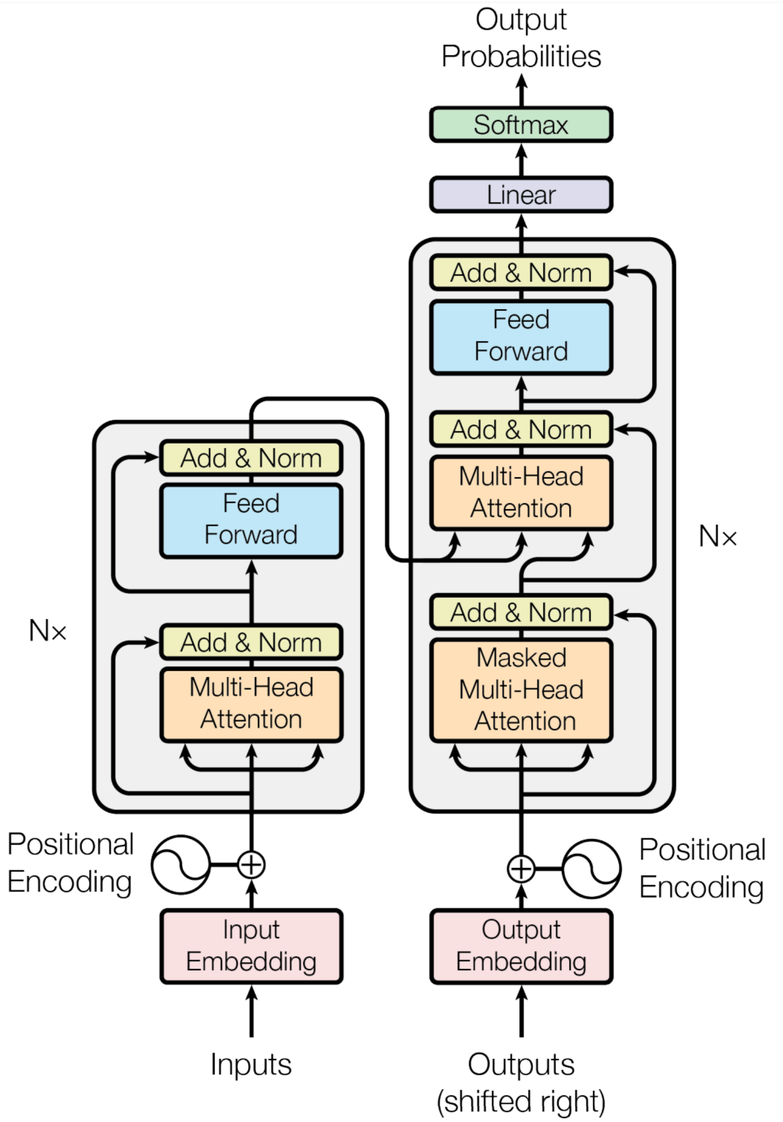


Transformers

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:

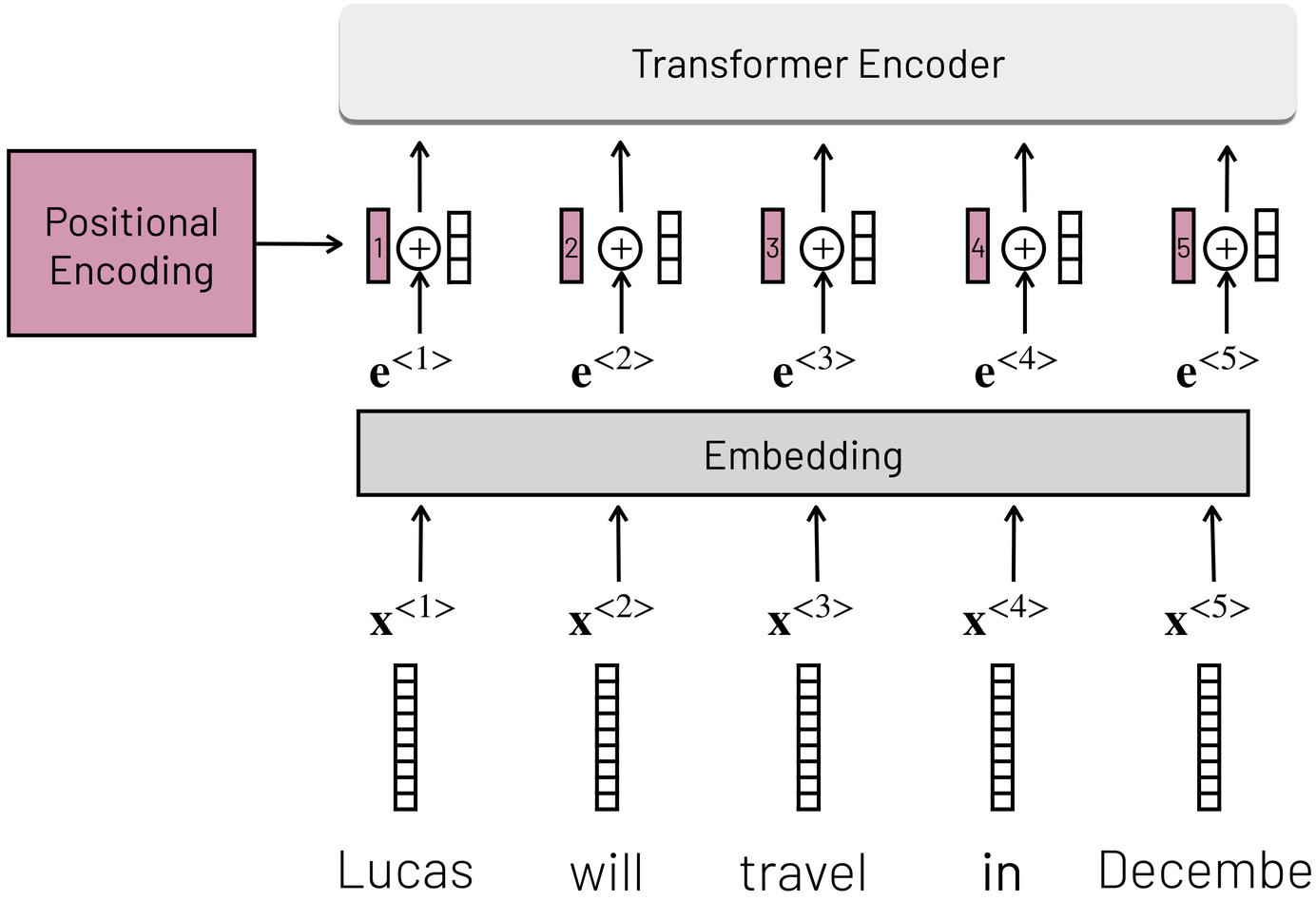
- Natural Language Processing
- Computer Vision
- Reinforcement Learning
- ...



Vaswani et al. 2017, Attention Is All You Need

Encoder Input

The transformer encoder takes as input a sequence of word embeddings summed with positional encodings. This sequence has the constant size (T_x, d_{model}) throughout the entire model



The contextual embeddings size is typically called d_{model}

$$e_{x+p} = \{e_x^{<1>} + pe^{<1>}, \ldots, e_x^{<T_x>} + pe^{<T_x>}\}$$

$(T_x, d_{model}) \rightarrow$ word + positional embedding

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

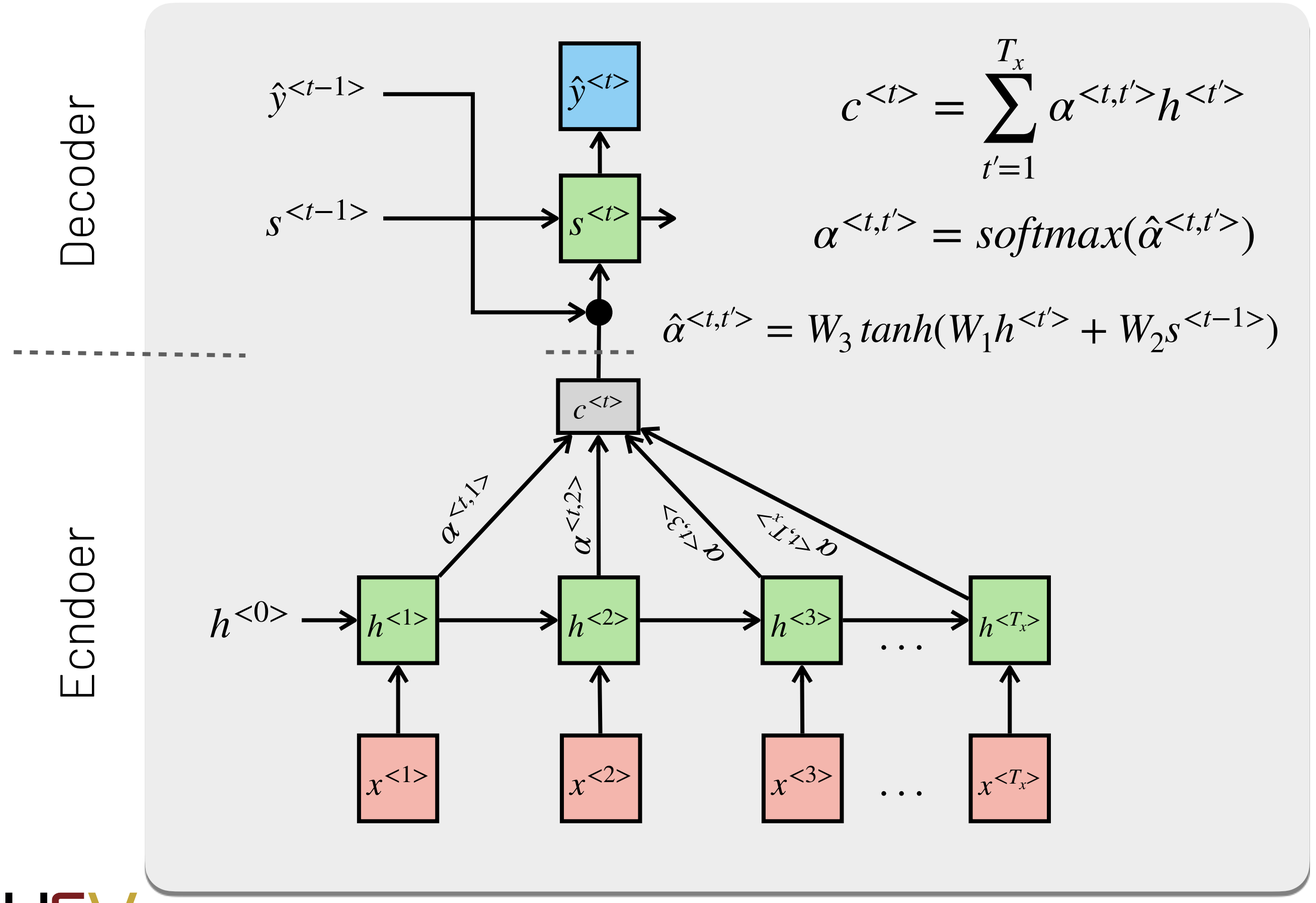
$(T_x, d_{model}) \rightarrow$ word embedding

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

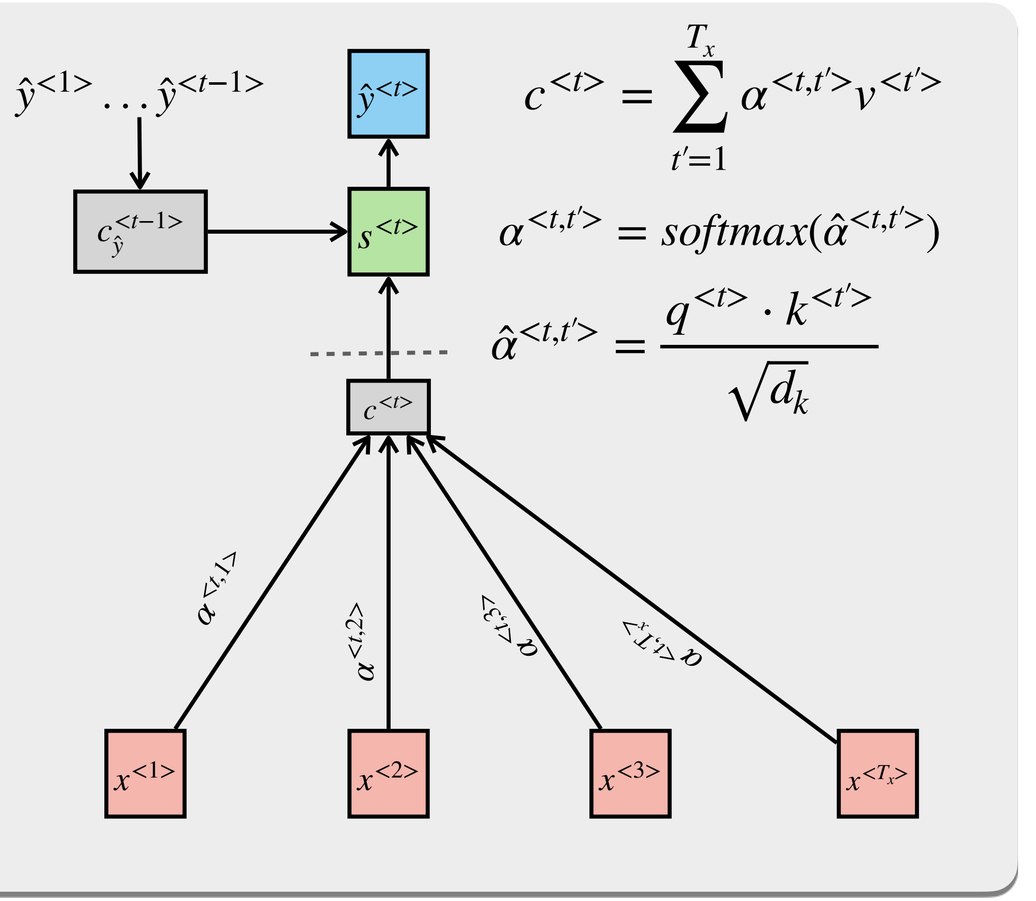
$(T_x, |V|) \rightarrow$ 1-hot

Attention in RNNs vs. Transformers

RNNs Badahnau (Additive) Attention



Transformers Scaled Dot-Product Attention



UFV

Self-Attention

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

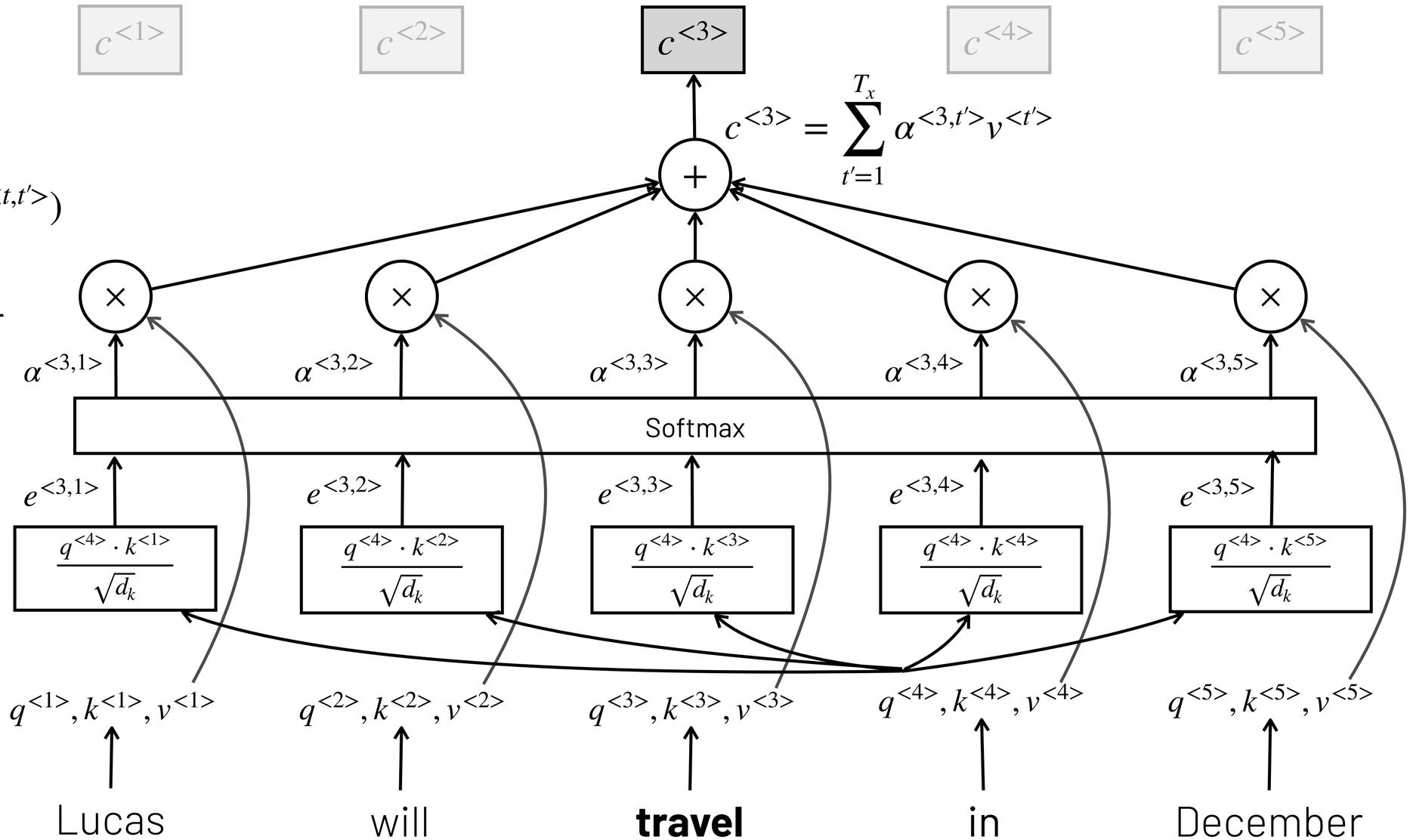
$$\alpha^{<t,t'>} = softmax(\hat{\alpha}^{<t,t'>})$$

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

$$q^{<t>} = W^q e_{x+p}^{<t>}$$

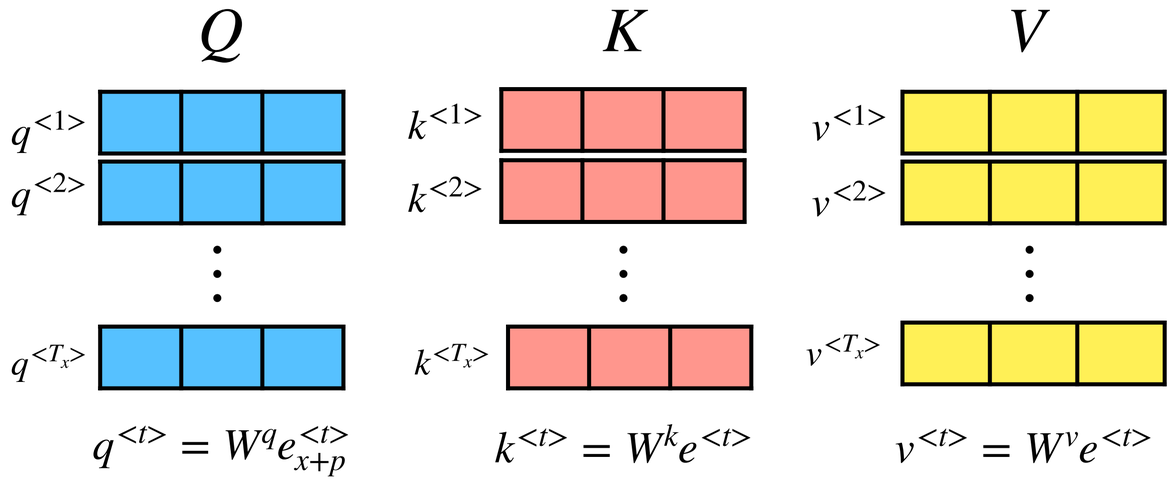
$$k^{<t>} = W^k e_{x+p}^{<t>}$$

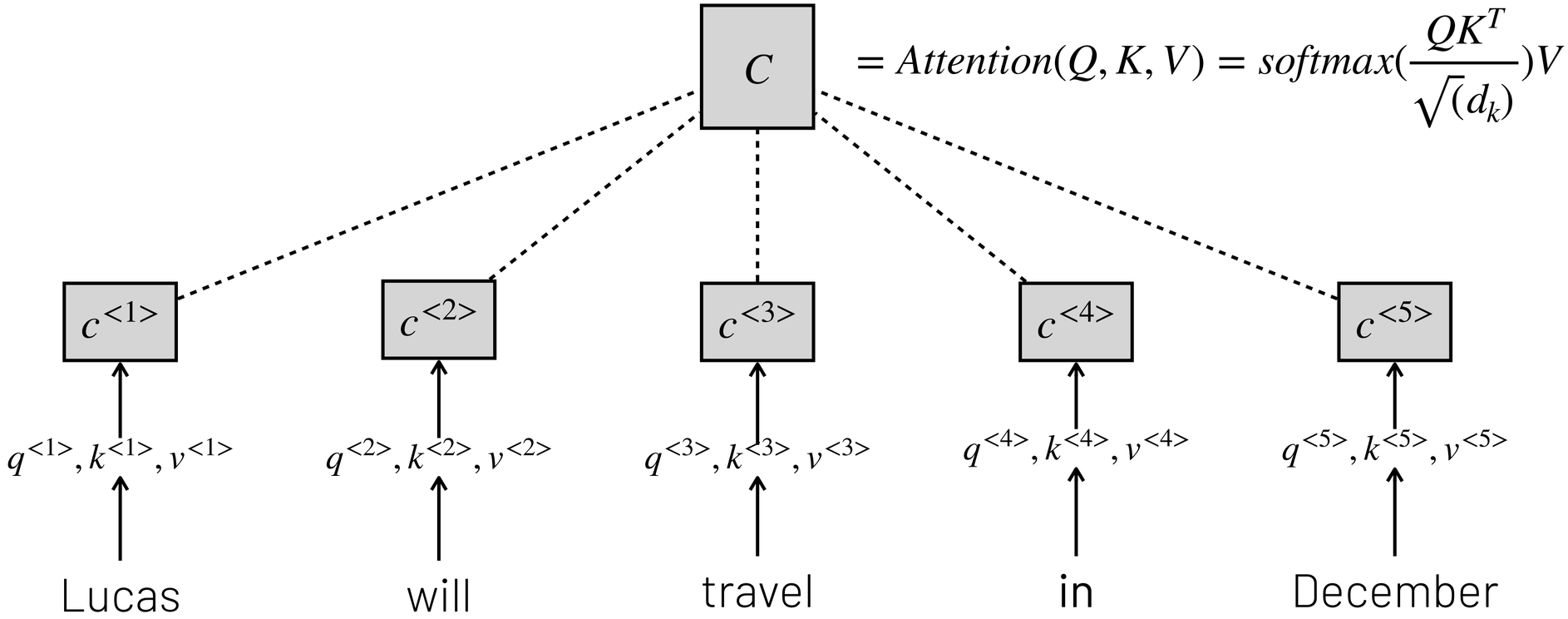
$$v^{<t>} = W^v e_{x+p}^{<t>}$$



Self-Attention

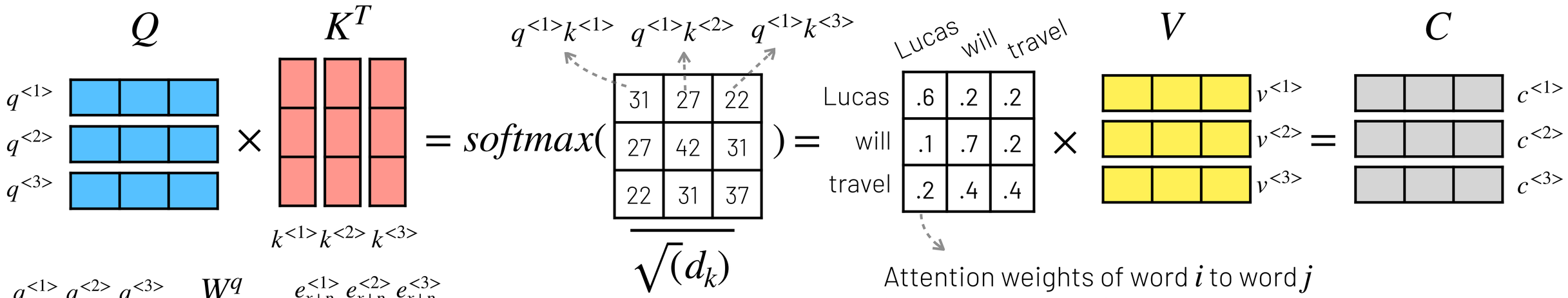
The contextal represention $C = \{c^{<1>}, \ldots, c^{<T_x>}\}$ of the entire input sequence $x = \{x^{<1>}, \ldots, x^{<T_x>}\}$ can be computed in a vectorized way combining vectors $q^{<t>}$, $k^{<t>}$, $v^{<t>}$ in matrices Q, K e V

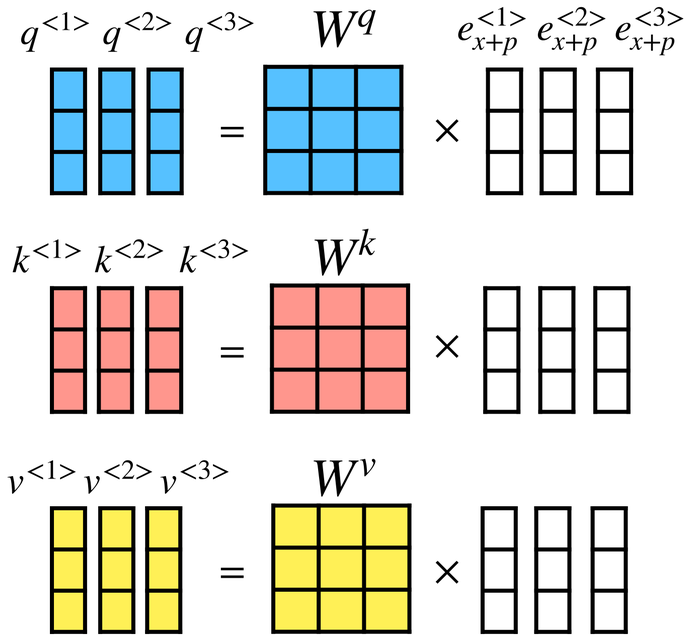


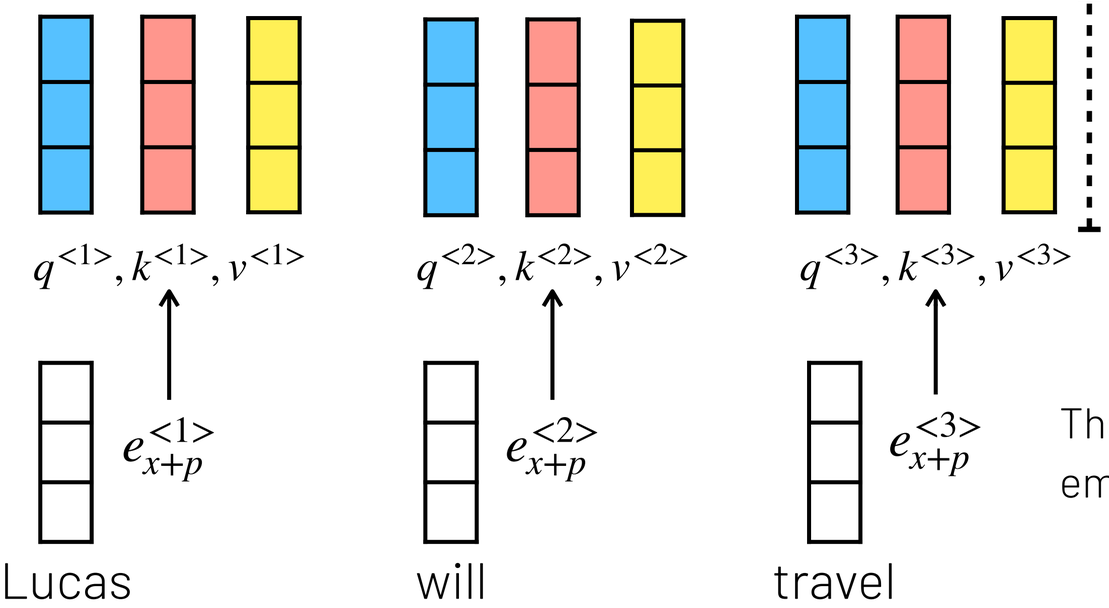


Self-Attention

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







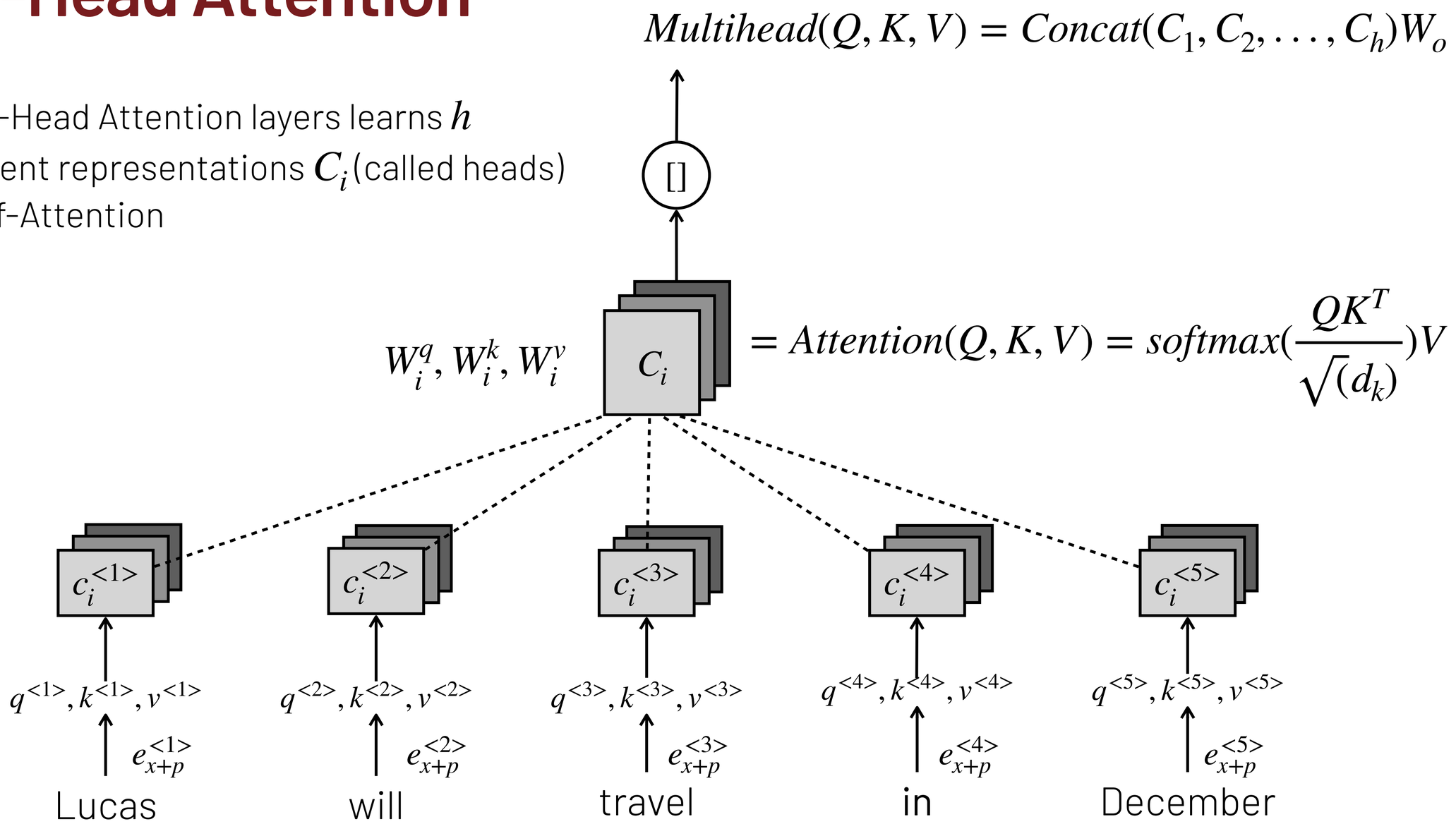
$$d_e = d_q = d_k = d_v = 4$$

The sizes of key and query have to be the same. But embeddings and value typically also have same sizes.

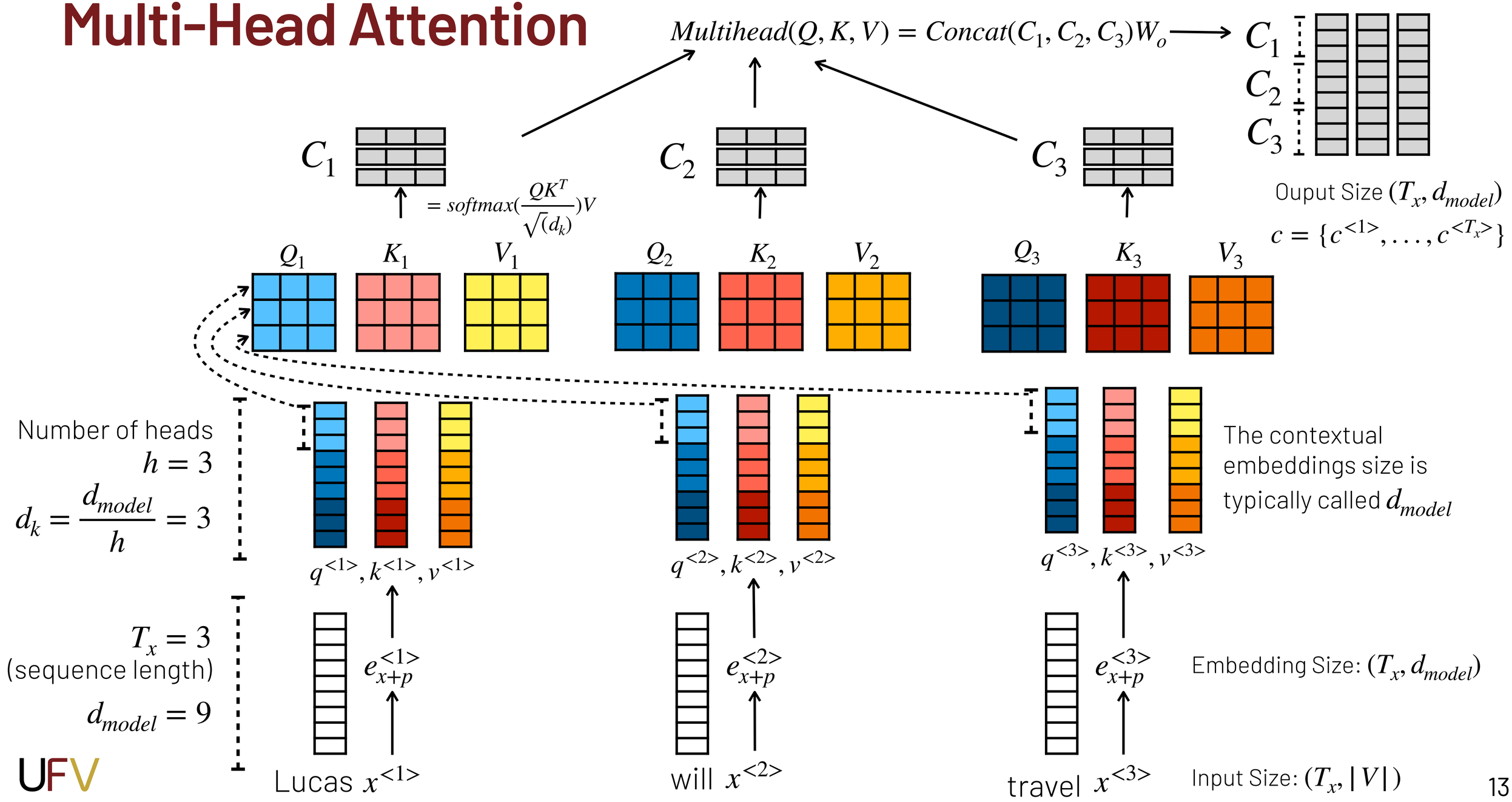
The attention layer receives the embedded vectors $e^{<t'>}$ as input

Multi-Head Attention

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



Multi-Head Attention



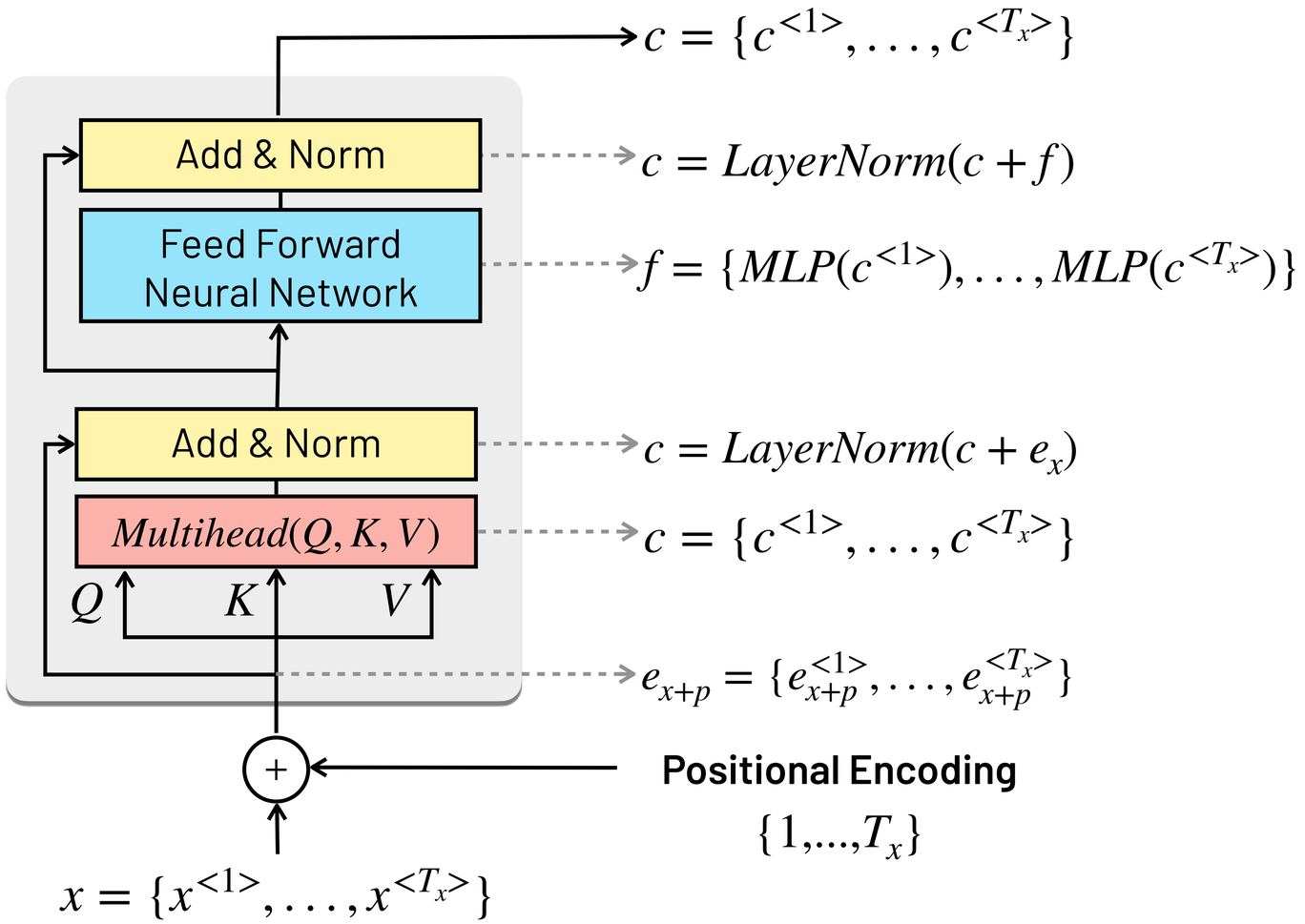
Encoder

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

Ouput: a contextual representation $C = \{c^{<1>}, \ldots, c^{<T_x>}\}$ 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***)

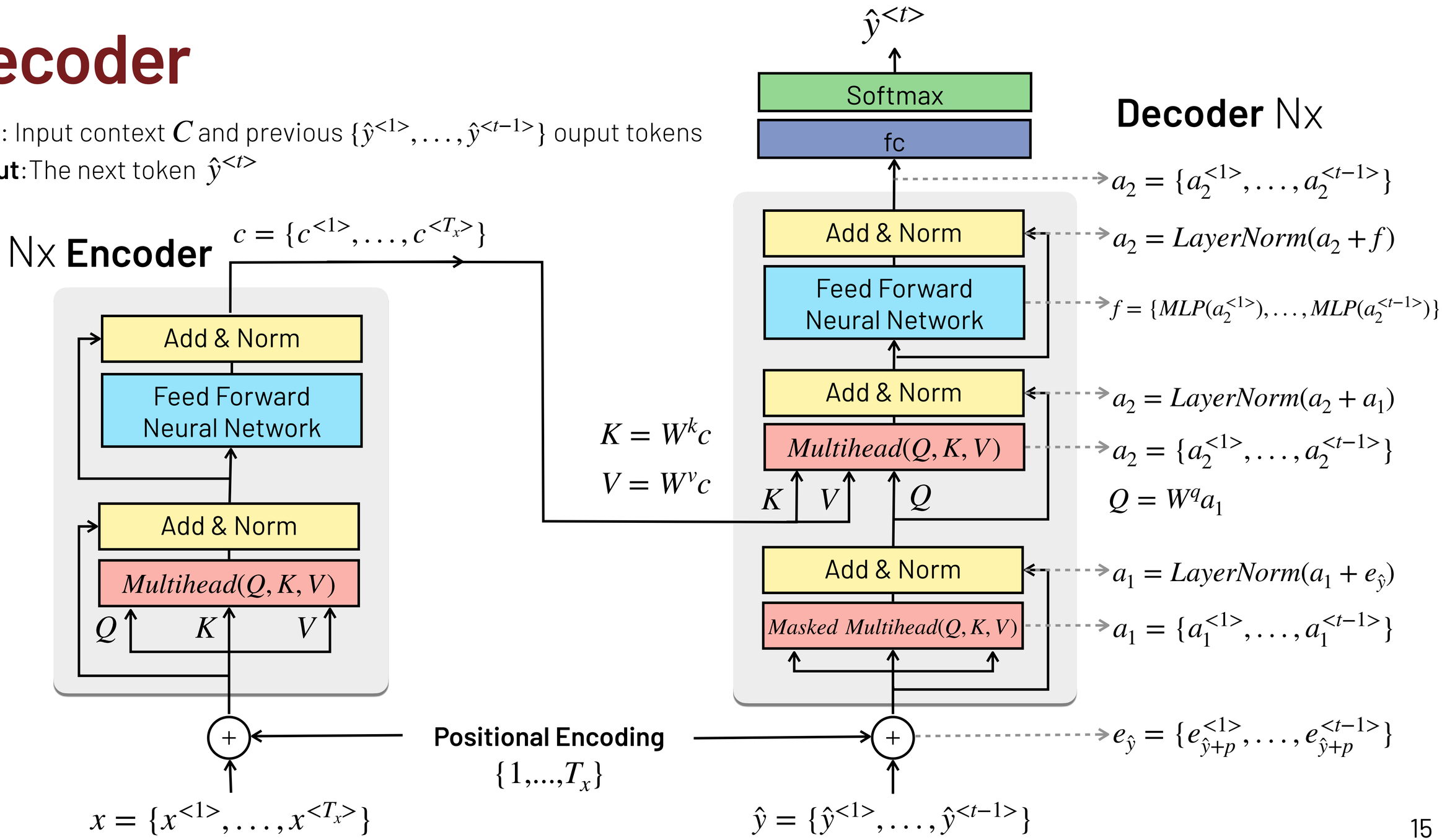


UFV

Decoder

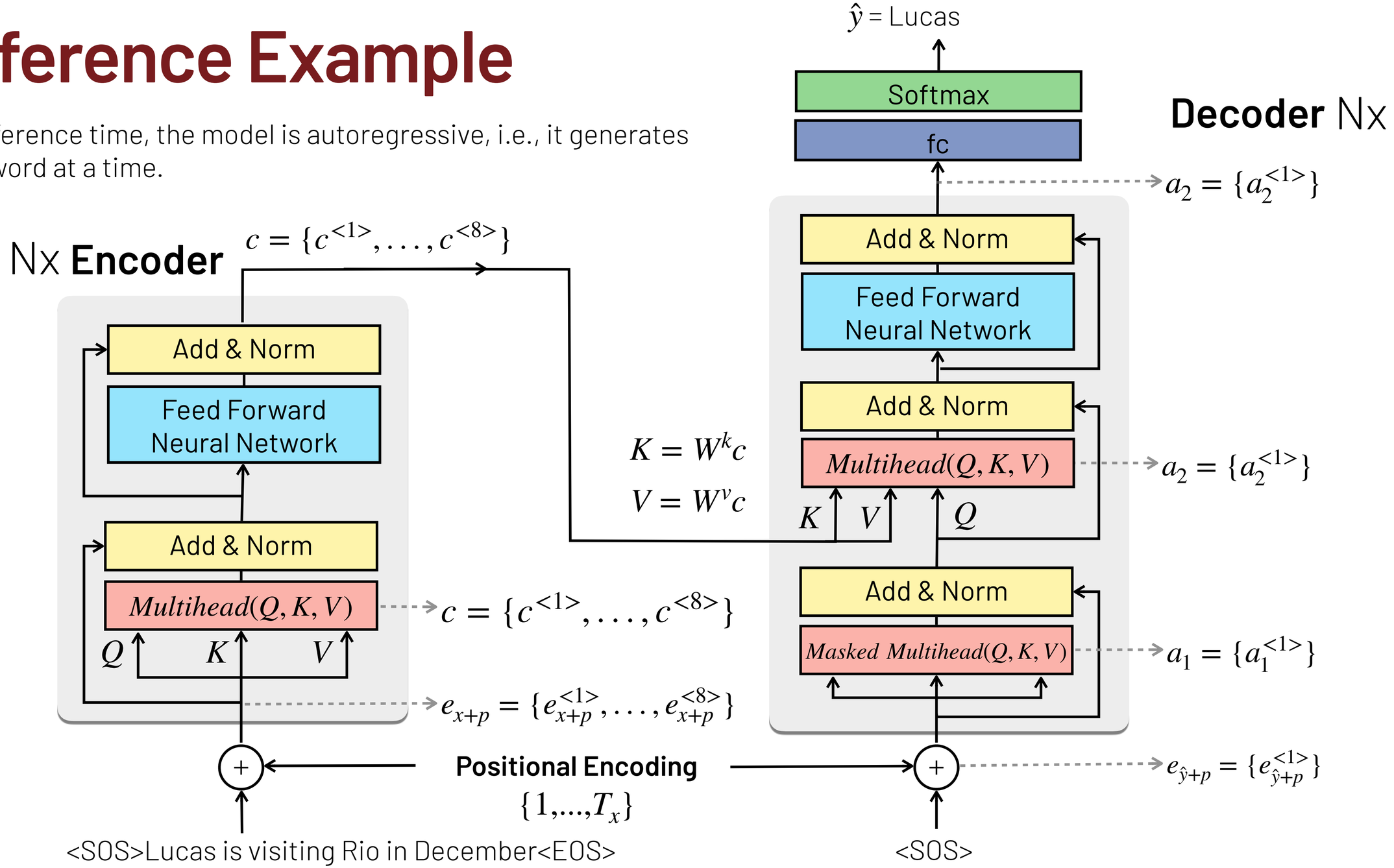
Input: Input context C and previous $\{\hat{y}^{<1>}, \ldots, \hat{y}^{<t-1>}\}$ ouput tokens

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



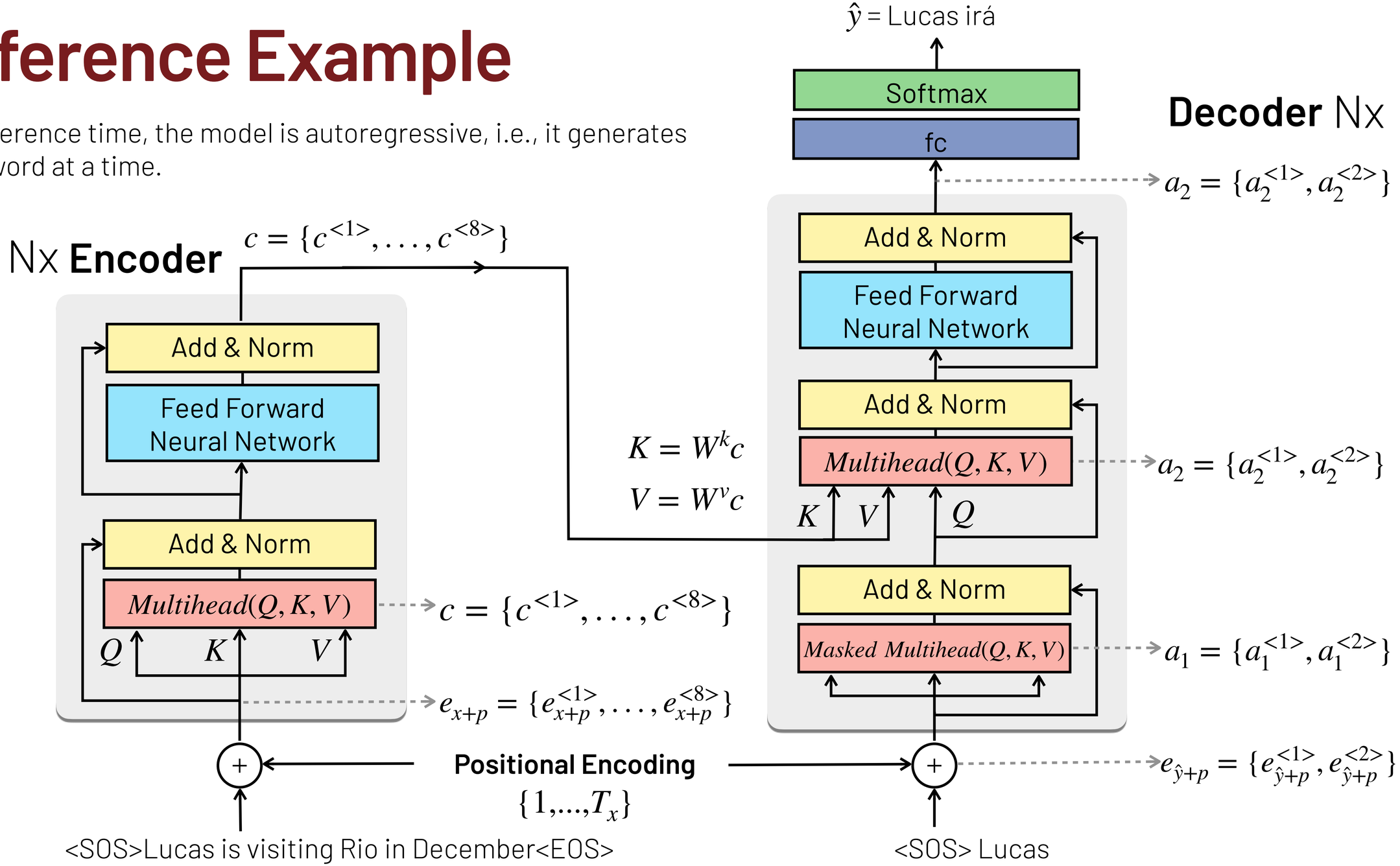
Inference Example

At inference time, the model is autoregressive, i.e., it generates one word at a time.



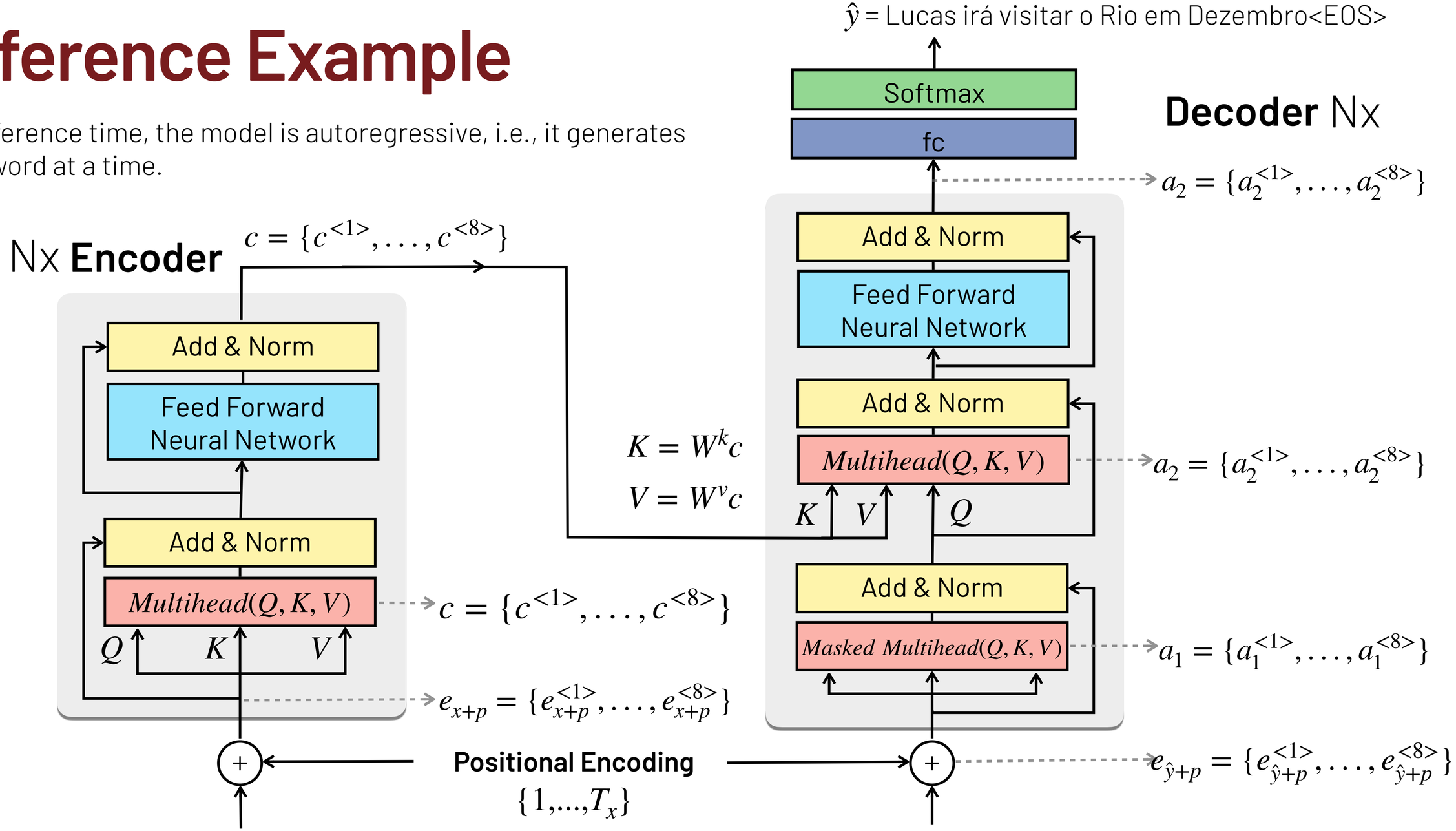
Inference Example

At inference time, the model is autoregressive, i.e., it generates one word at a time.



Inference Example

At inference time, the model is autoregressive, i.e., it generates one word at a time.

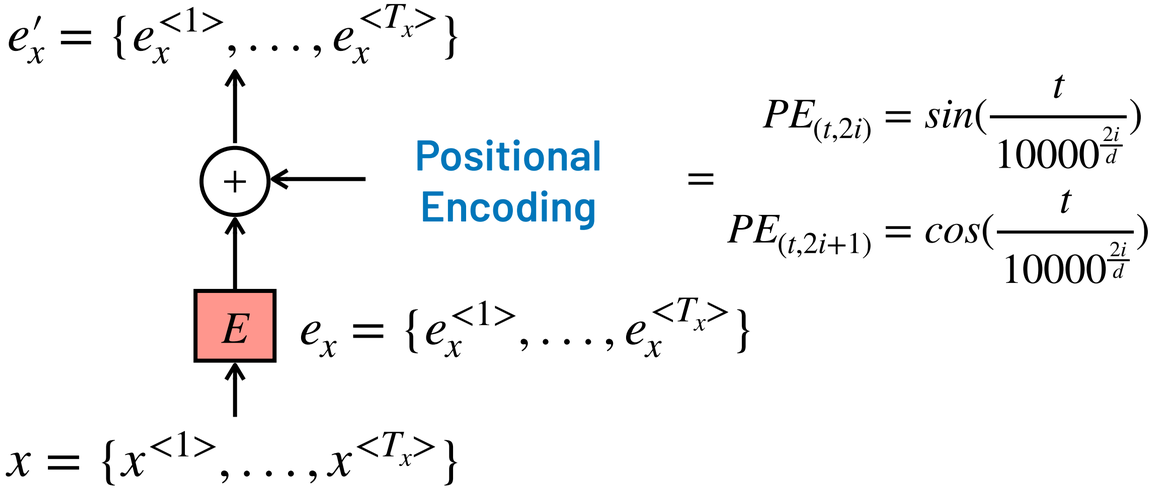


Positional Encoding

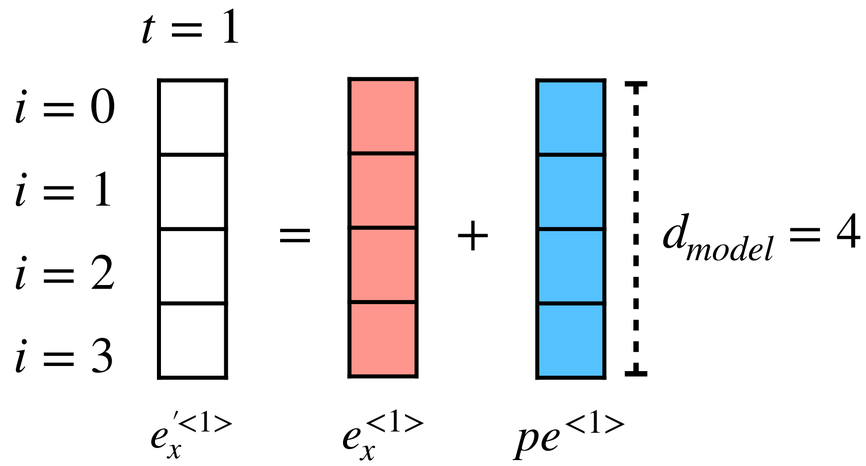
The self-attention mechanism does not consider the position of the words.

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

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



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



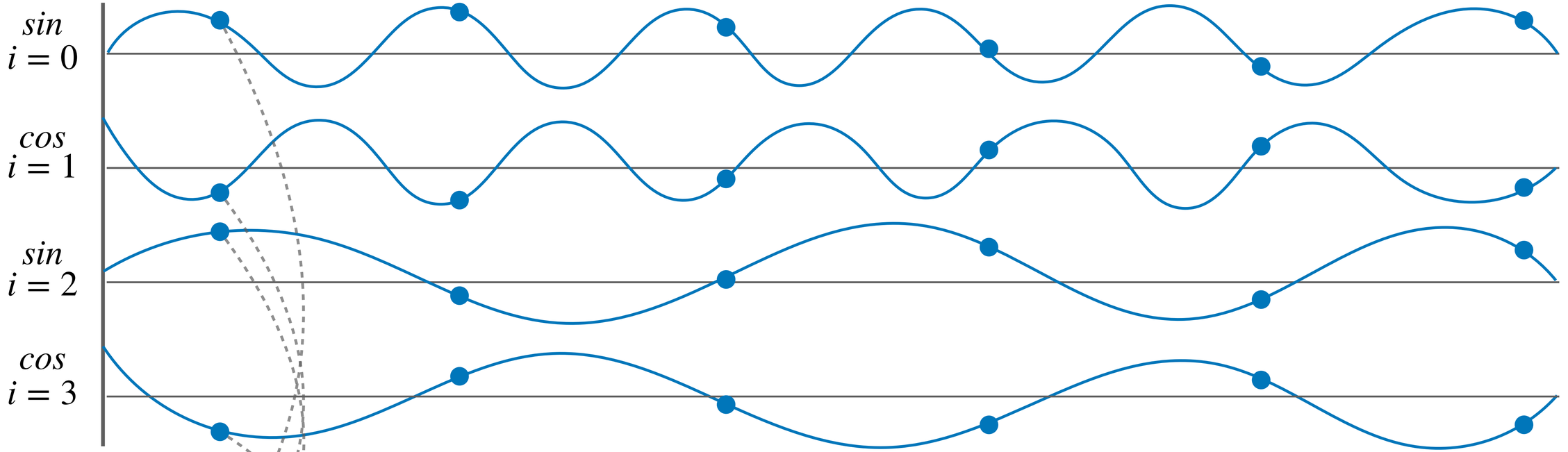
Positional Encoding

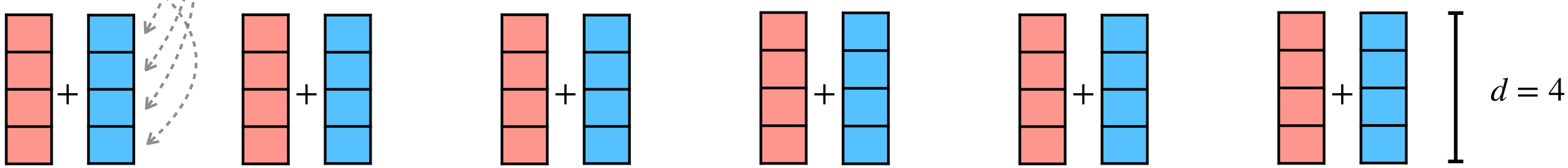
if i is even

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

if i is odd

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

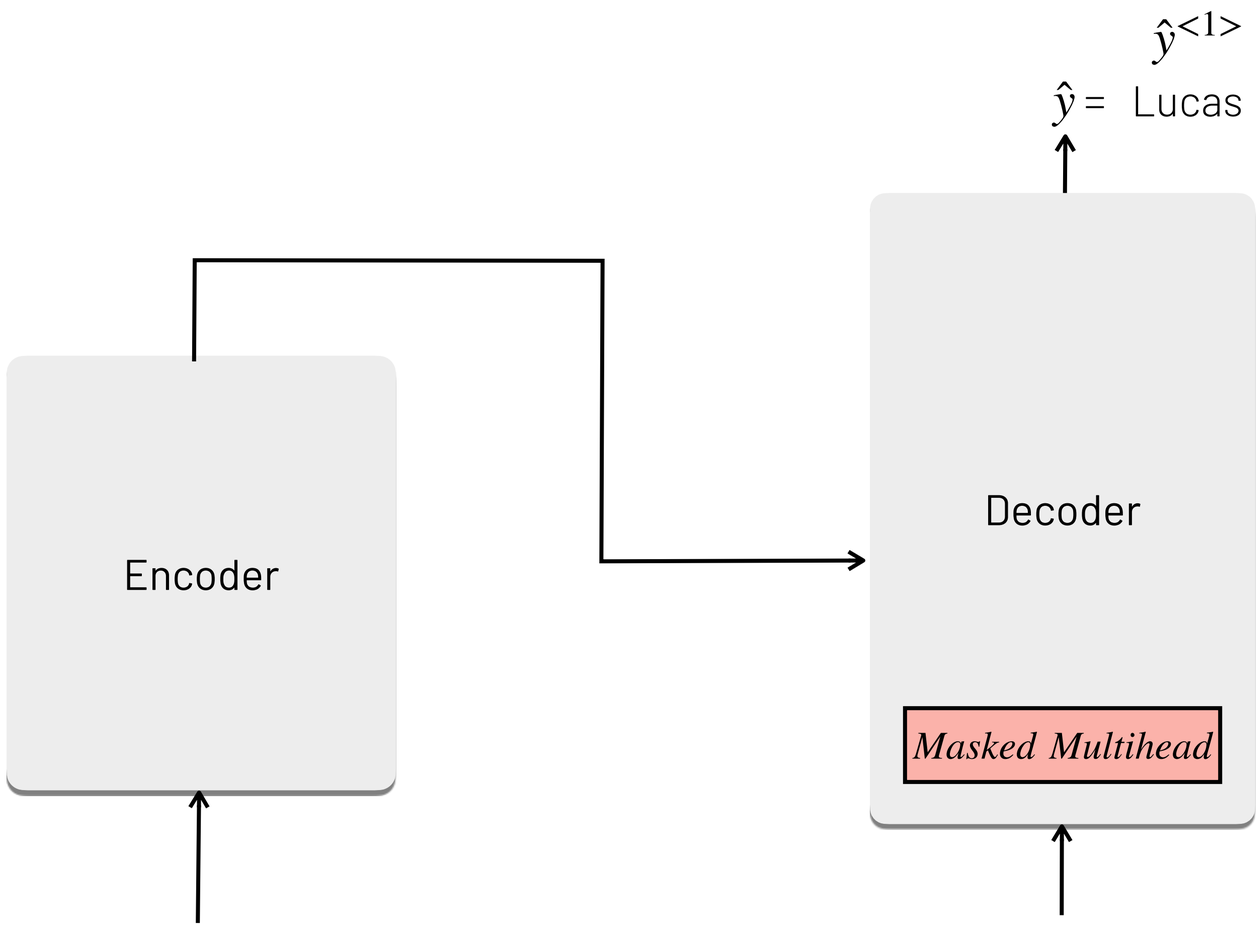




t : $e_x^{<1>}$ $pe^{<1>}$ $e_x^{<2>}$ $pe^{<2>}$ $e_x^{<3>}$ $pe^{<3>}$ $e_x^{<4>}$ $pe^{<4>}$ $e_x^{<5>}$ $pe^{<5>}$ $e_x^{<6>}$ $pe^{<6>}$

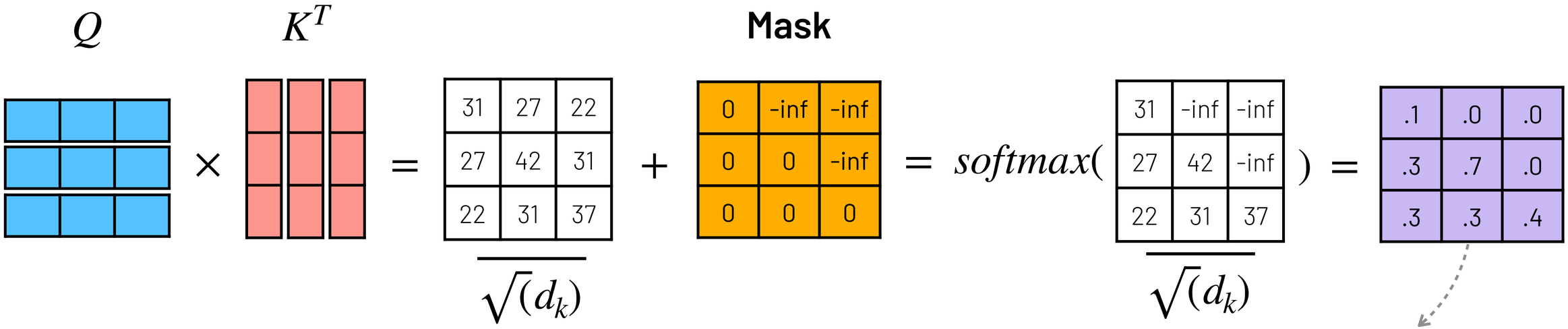
UFV Lucas is visiting **Rio** in December

Training with Masked Multi-head Attention



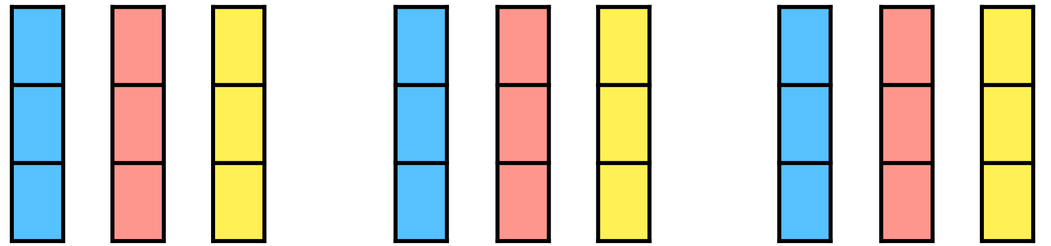
- During training, the model **does not** produce $\hat{y}$ one token at at time;
- Instead, the output $\hat{y}$ is produced once, in parallel with vectorization.
- Thus, the full target sentence $y = \{y^{<1>}, \ldots, y^{<T_y>}\}$ is given as input to the Decoder, which uses a **mask** to predict $\hat{y}^{<t>}$, so it does not attent to the future tokens $(\hat{y}^{<t'>}, t' > t)$. For example, for $t = 5$:

Training with Masked Multi-head Attention



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

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



$q^{<1>}, k^{<1>}, v^{<1>}$ $q^{<2>}, k^{<2>}, v^{<2>}$ $q^{<3>}, k^{<3>}, v^{<3>}$

Lucas $x^{<1>}$ will $x^{<2>}$ travel $x^{<3>}$

Next Lecture

L17: Transformers (Part II)

Case studies of transformers: BERT and GPT