

INF721

2023/2



Aprendizado em Redes Neurais Profundas

A21: Transformers

Logística

Avisos

- ▶ Entrega da PF: Proposta de Problema nesta quarta-feira (18/10)!

Última aula

- ▶ Mecanismo de Atenção em RNNs

Plano de Aula

- ▶ Transformers
- ▶ Auto-Atenção (Self-Attention)
- ▶ Atenção com Múltiplas Cabeças (Multi-head Attention)
- ▶ Codificação de Posição
- ▶ Treinamento (Masked Multi-head Attention)

Tradução Automática

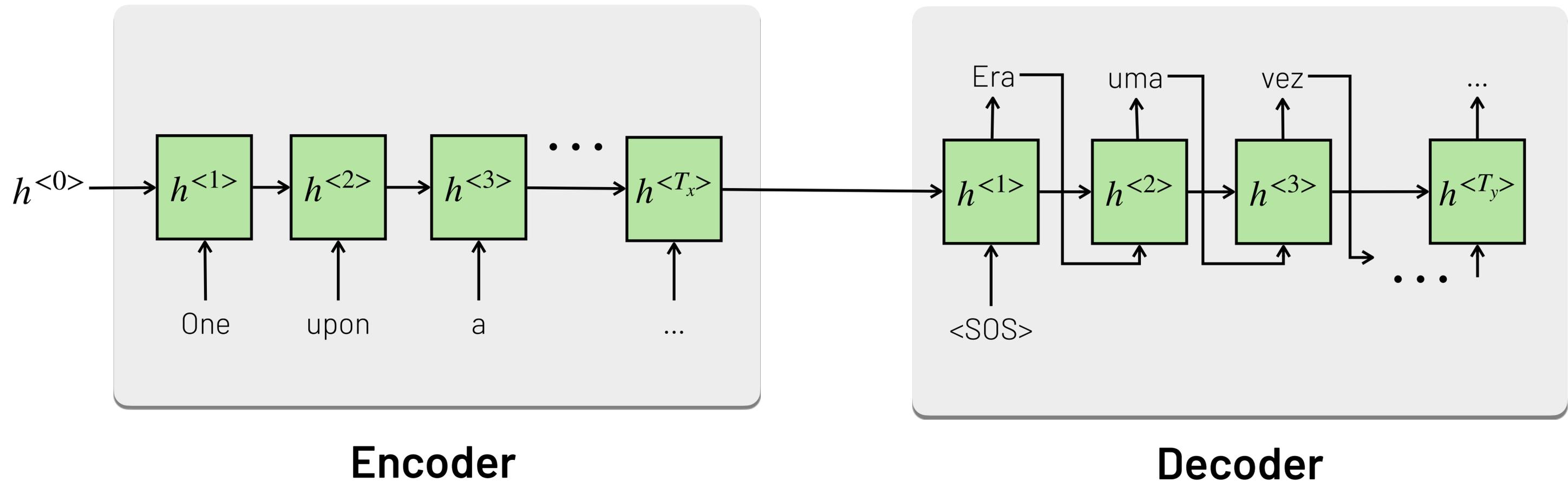
Conjunto de dados

Pares de sentenças de um idioma origem (x) para um idioma destino (y)

Inglês	Português
See you!	Nos vemos!
The book is on the table.	O livro está em cima da mesa.
Lucas is visiting Chile in January.	Lucas irá visitar o Chile em Janeiro.
Lucas is visiting Chile in January.	Em Janeiro, Lucas irá visitar o Chile.
....

Problemas com RNNs

- ▶ São pouco paralelizaveis
- ▶ Não conseguem capturar dependências longas

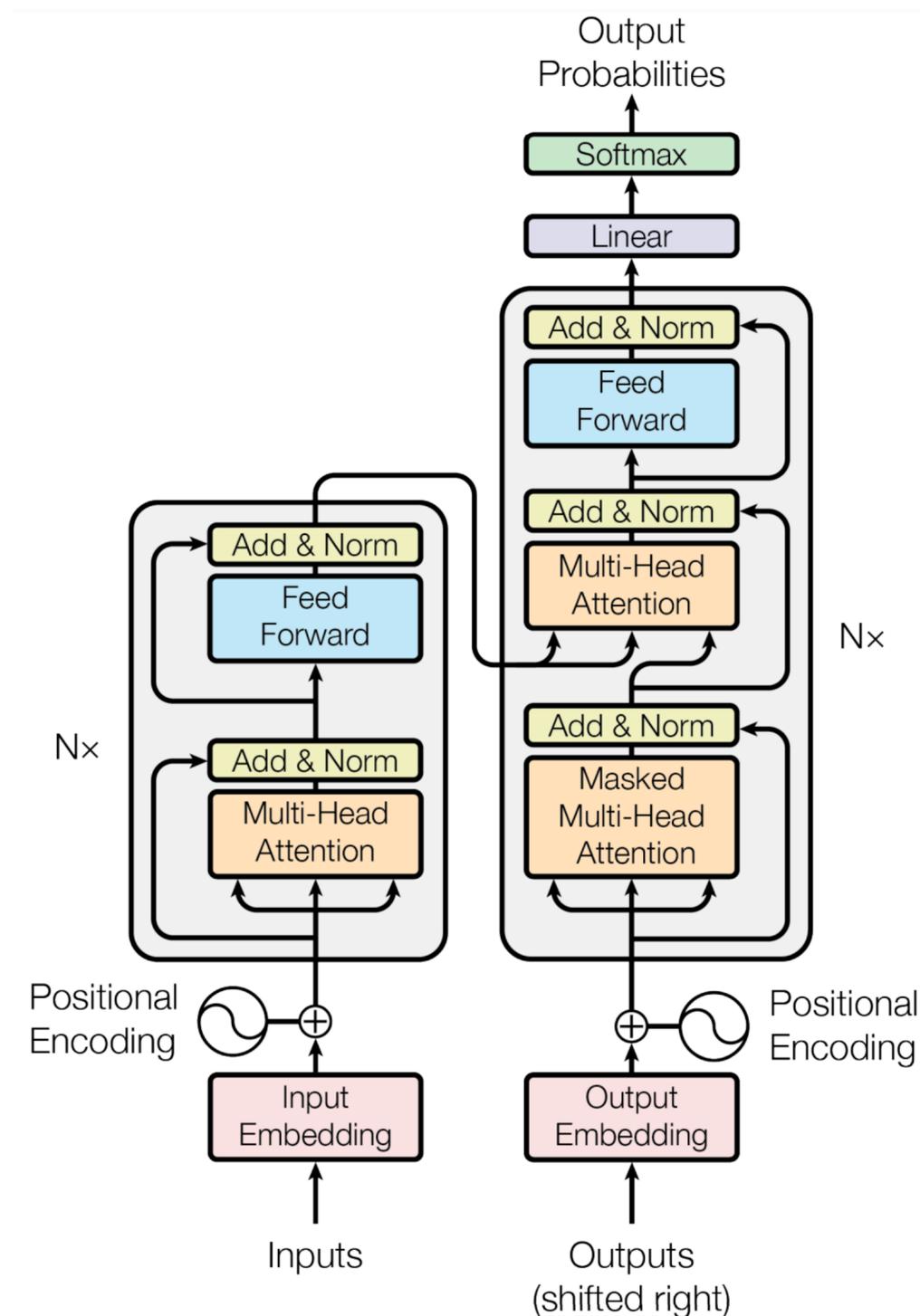


Attention is all you need

Transformers são uma arquitetura encoder-decoder para processamento de sequências utilizando apenas mecanismos atenção (eliminando recorrências).

Inicialmente proposta para o problema de tradução automática, mas se mostrou muito geral, resolvendo problemas em:

- ▶ Processamento de Linguagem Natural
- ▶ Visão Computacional
- ▶ Aprendizado por Reforço
- ▶ ...



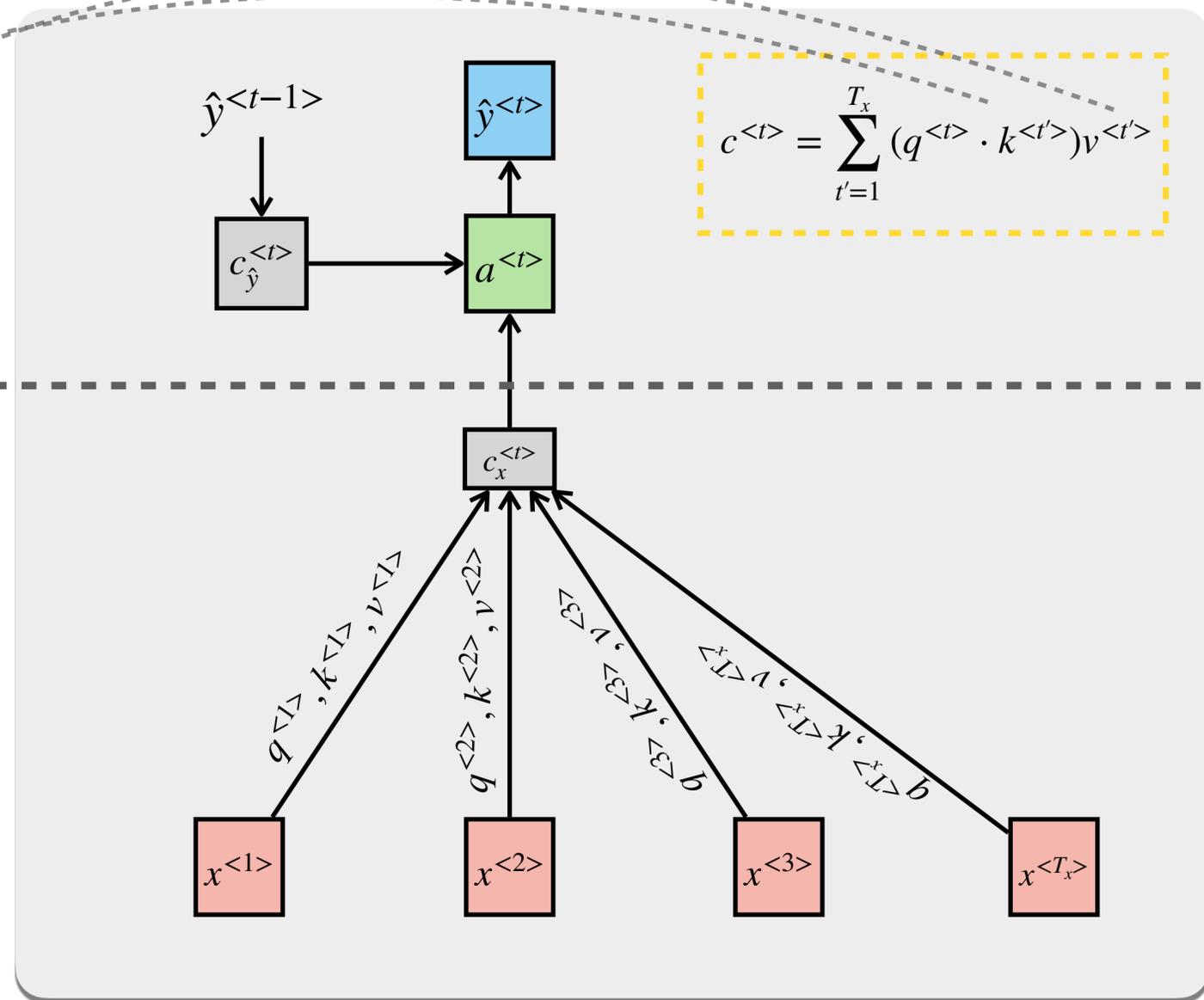
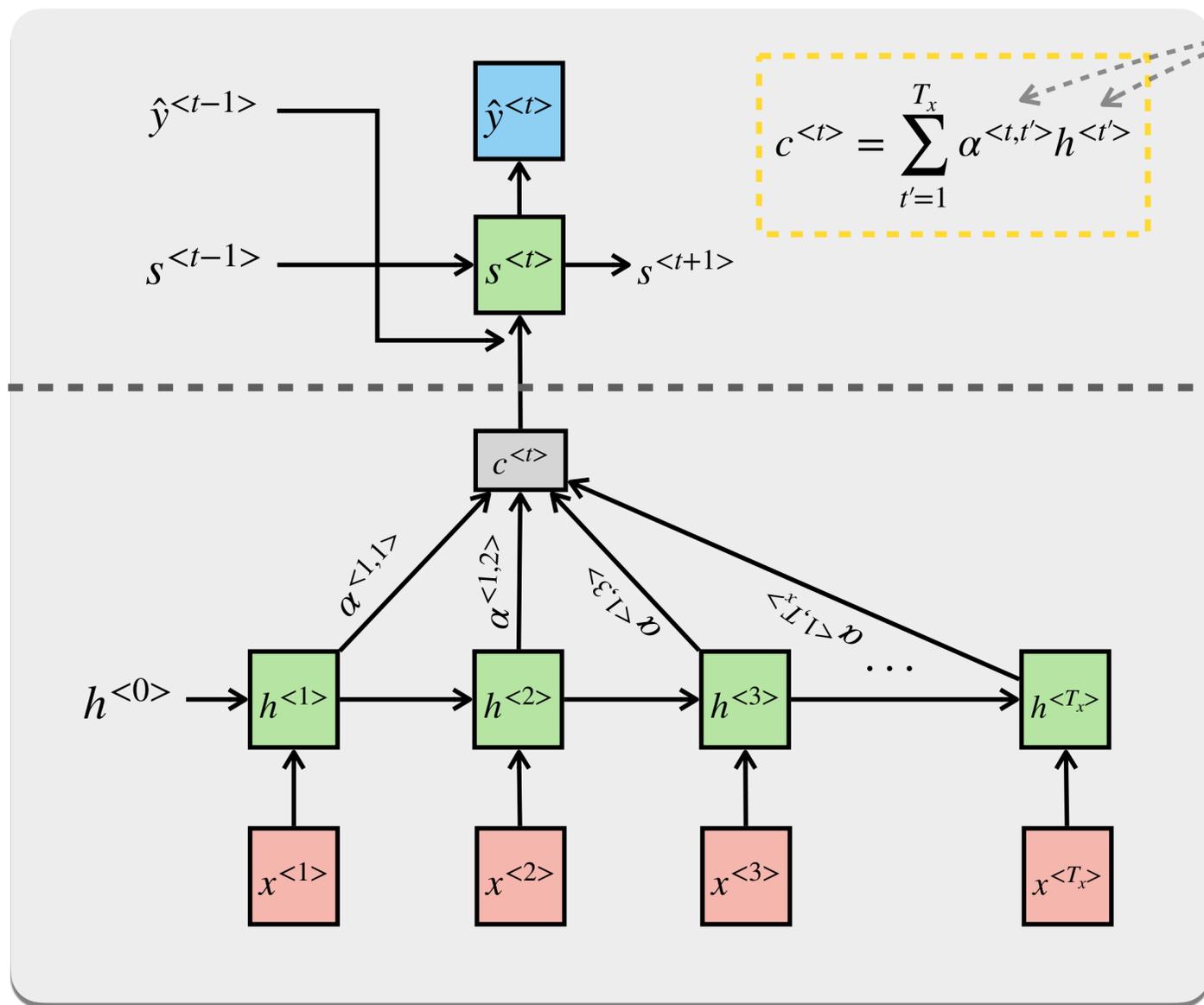
Removendo Recorrência

RNNs Atenção Aditiva

Transformers Auto-Atenção (Self Attention)

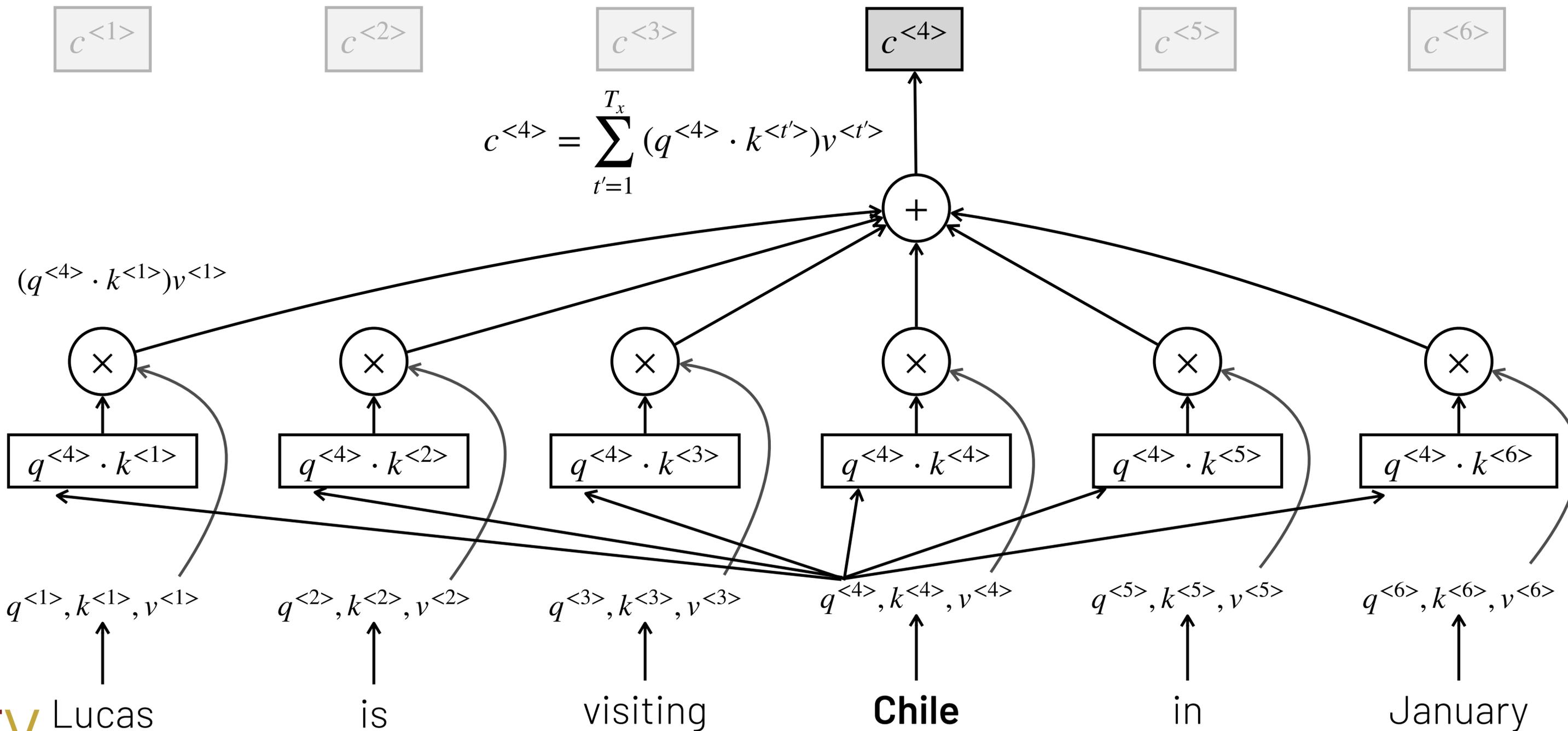
Decoder

Encoder



Auto-Atenção

O mecanismo de auto-atenção aprende um vetor de contexto $c^{<t>}$ para cada elemento $x^{<t>}$ da sequência x com base nela mesma.



Auto-Atenção

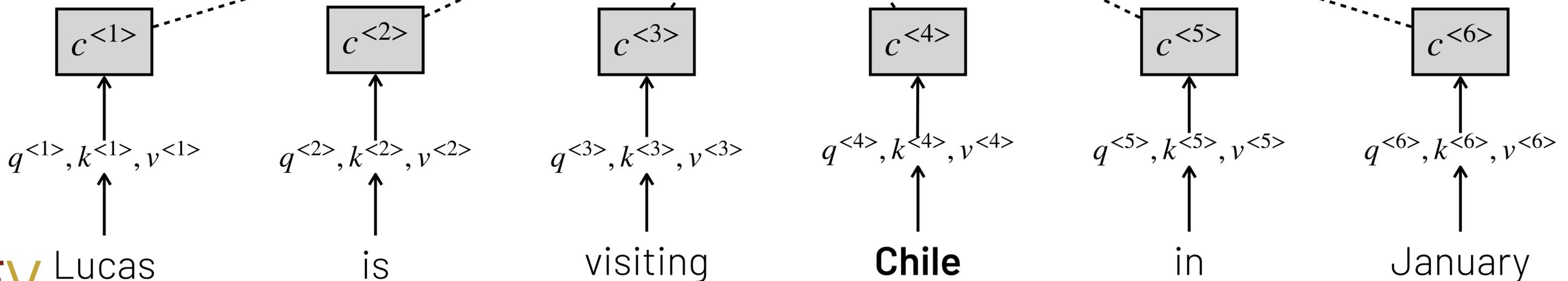
Query (Q)	Key (K)	Value (V)
$q^{<1>}$	$k^{<1>}$	$v^{<1>}$
$q^{<2>}$	$k^{<2>}$	$v^{<2>}$
...
$q^{<T_x>}$	$k^{<T_x>}$	$v^{<T_x>}$

A representação $C = \{c^{<1>}, \dots, c^{<T_x>}\}$ da entrada x é calculada de forma vetorizada empacotando os vetores $q^{<t>}, k^{<t>}, v^{<t>}$ em matrizes Q, K e V

$$q^{<t>} = W^q e^{<t>} \quad k^{<t>} = W^k e^{<t>} \quad v^{<t>} = W^v e^{<t>}$$

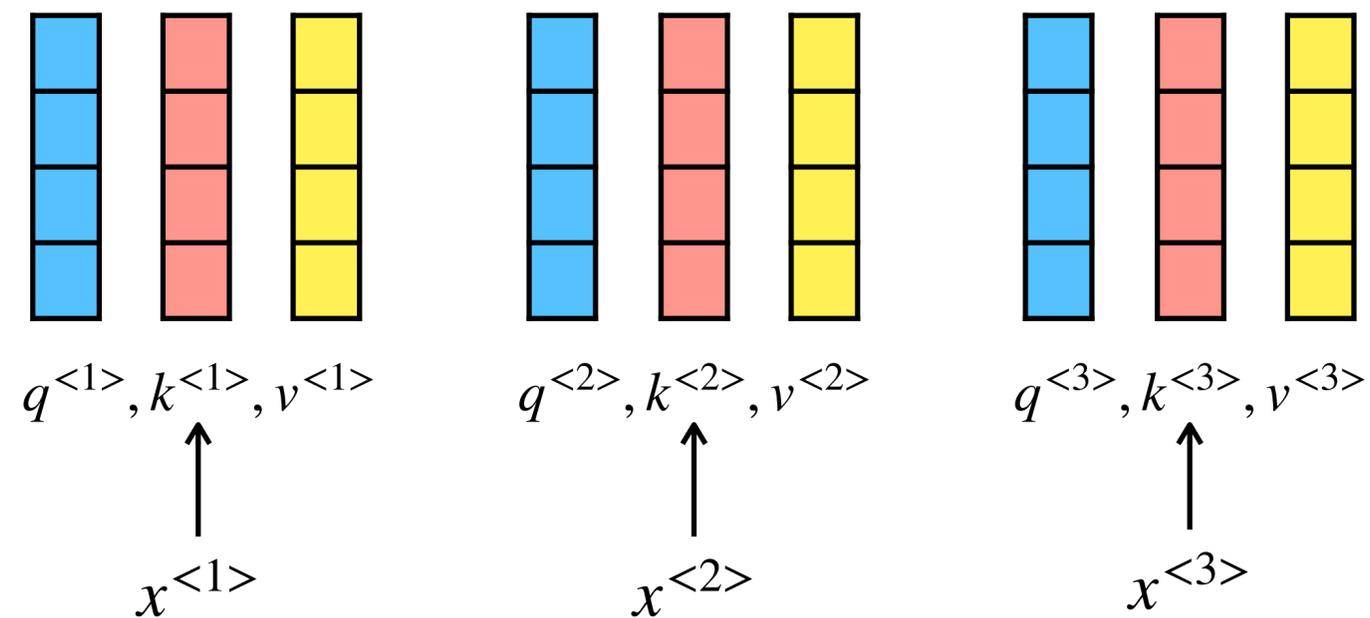
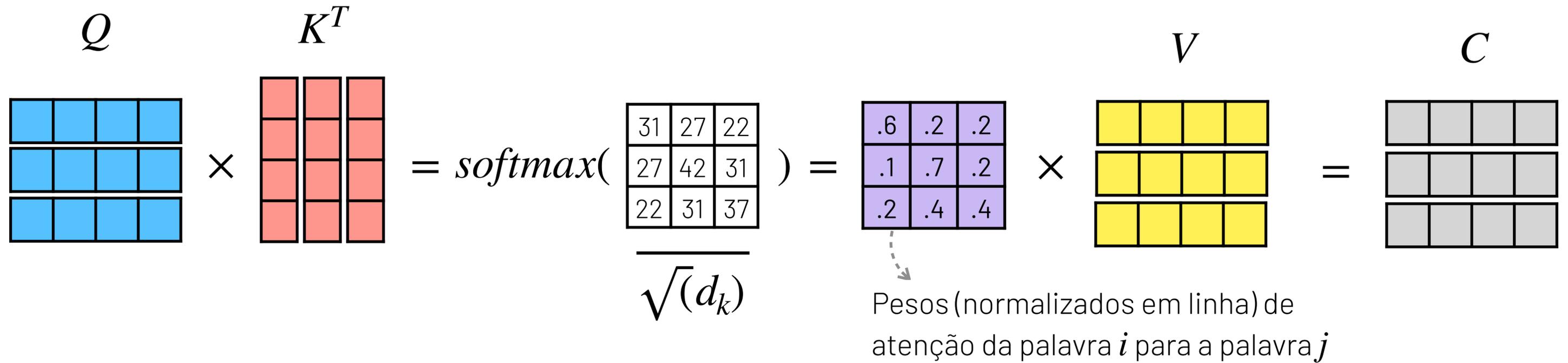
$e^{<t>}$: word embedding de $x^{<t>}$ (one-hot)

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



Auto-Atenção

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

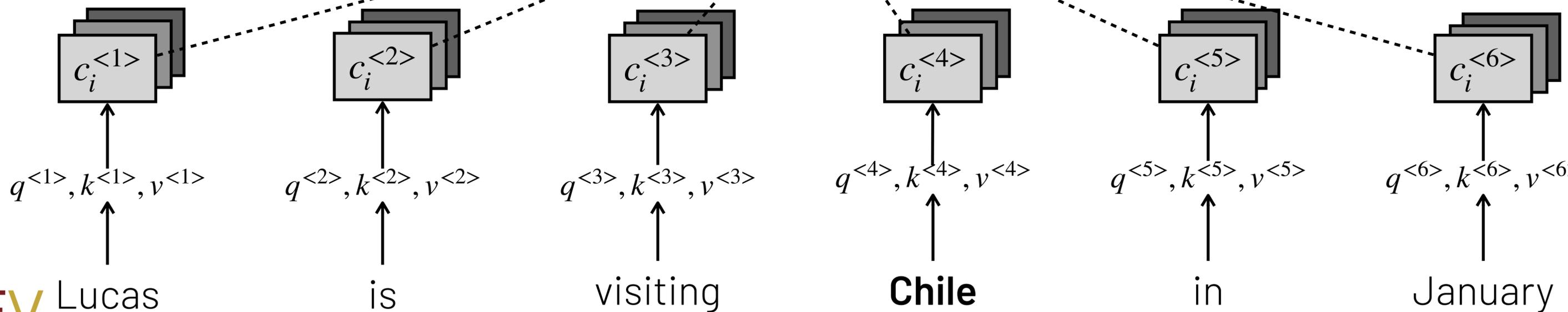


Multi-Head Attention

A camada de Multi-Head Attention aprende h representações independentes C_i (chamada de cabeça) com Auto-Atenção

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

$$W_i^q, W_i^k, W_i^v \quad C_i \quad = Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

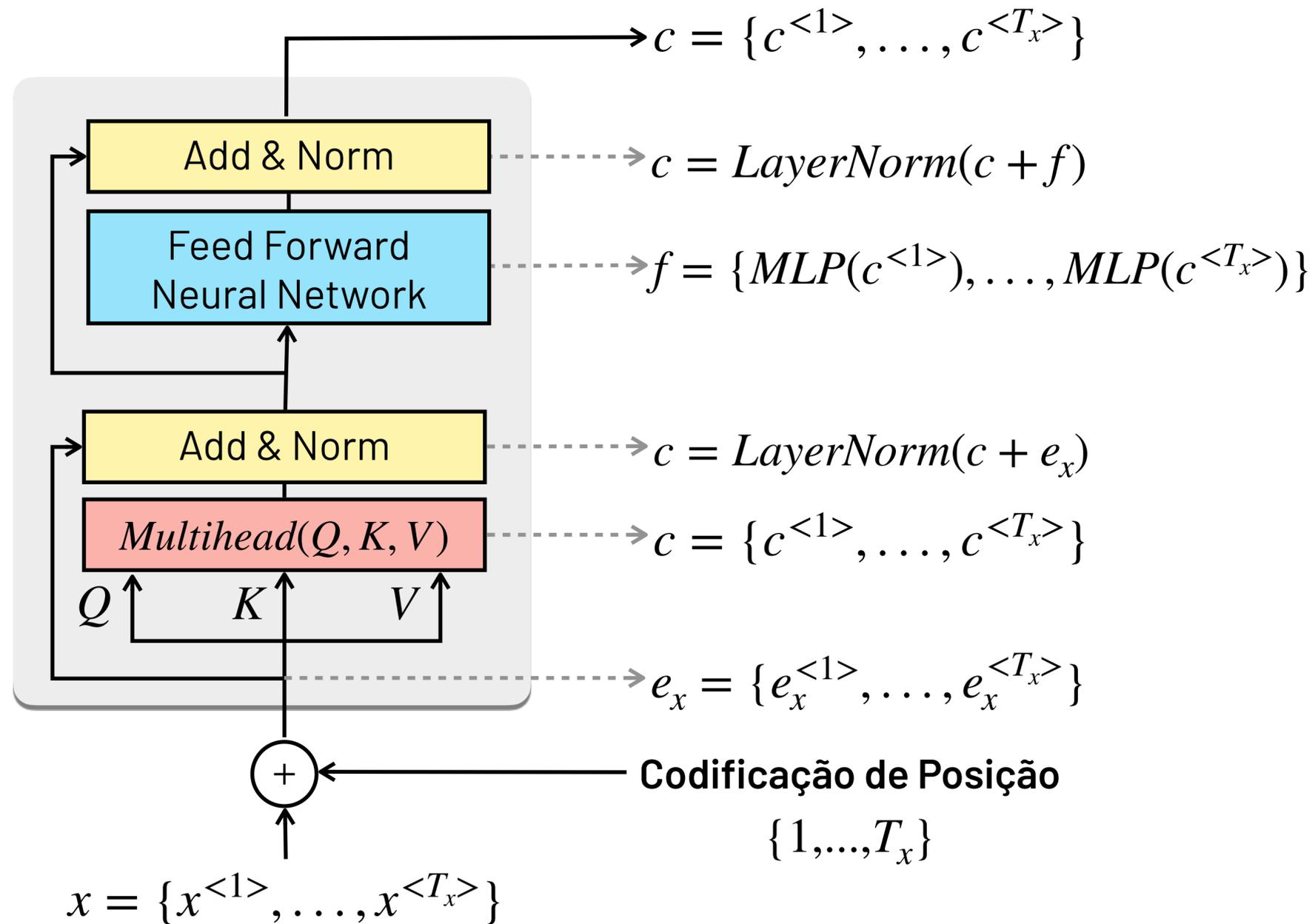


Encoder

Entrada: uma sequência $x = \{x^{<1>}, \dots, x^{<T_x>}\}$

Saída: uma representação contextual $c = \{c^{<1>}, \dots, c^{<T_x>}\}$ para x

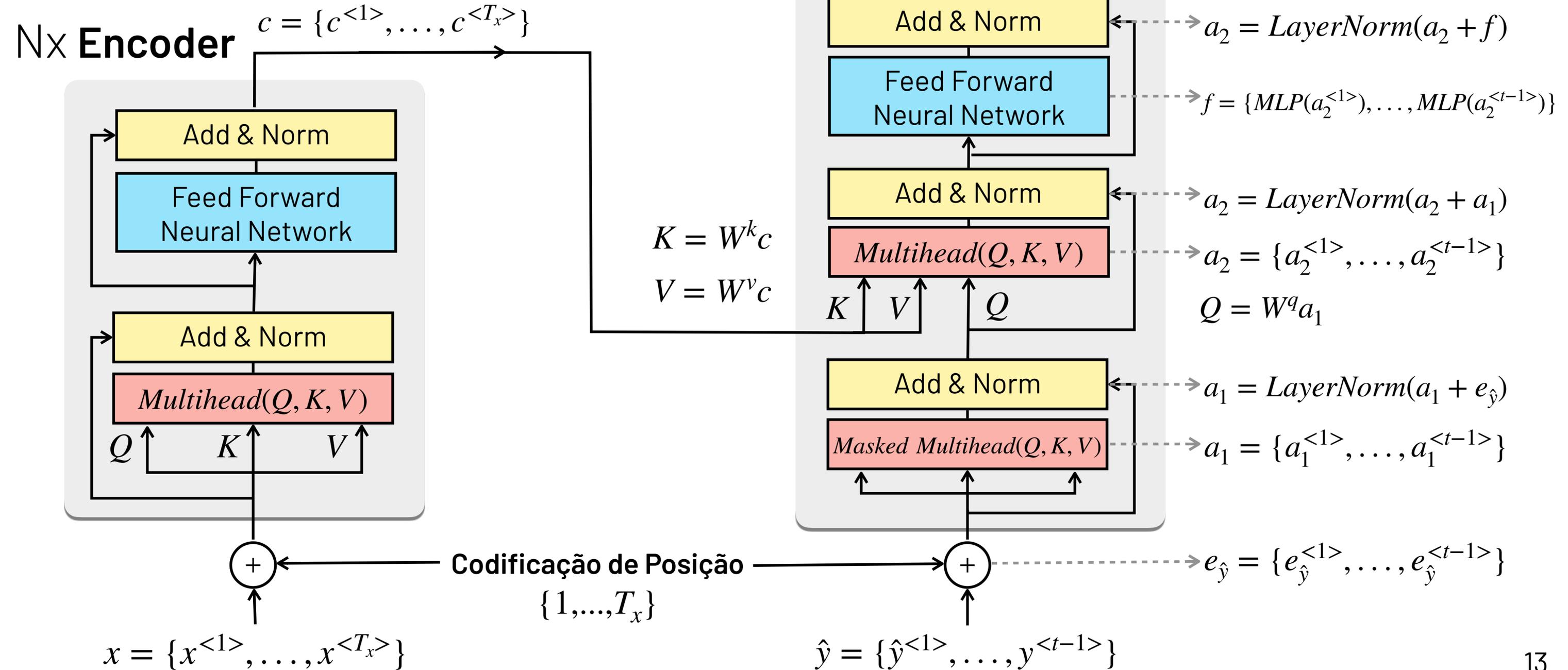
Para isso, ele aplica uma camada **Multihead** seguida de uma **Feed Forward Neural Network** (MLP). Ambas são normalizadas (**Norm**) e conectadas com as camadas anteriores residualmente (**Add**)



Decoder

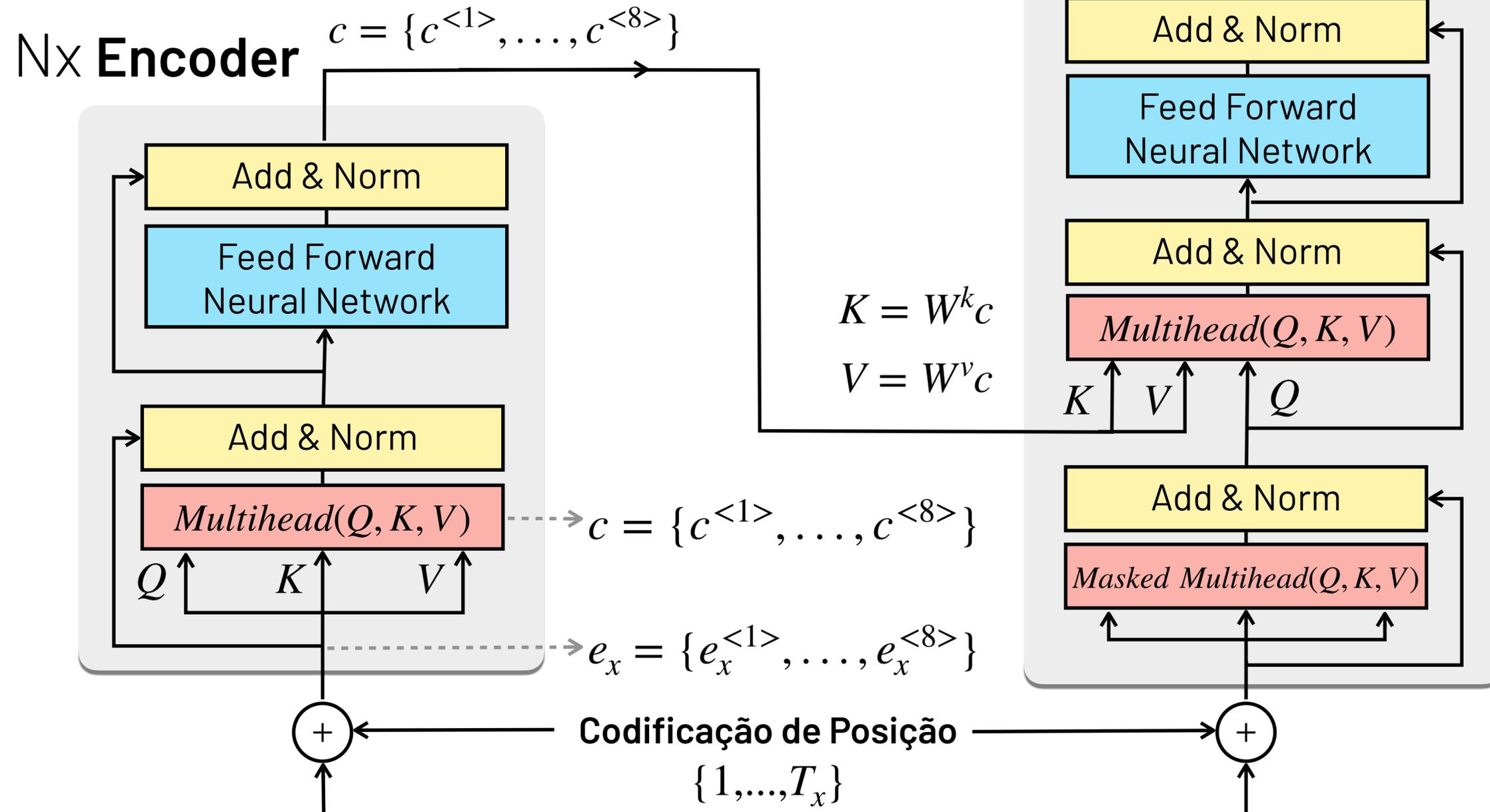
Entrada: O contexto c e os tokens $\{\hat{y}^{<1>}, \dots, y^{<t-1>}\}$ gerados até $t-1$

Saída: A previsão do próximo token $\hat{y}^{<t>}$



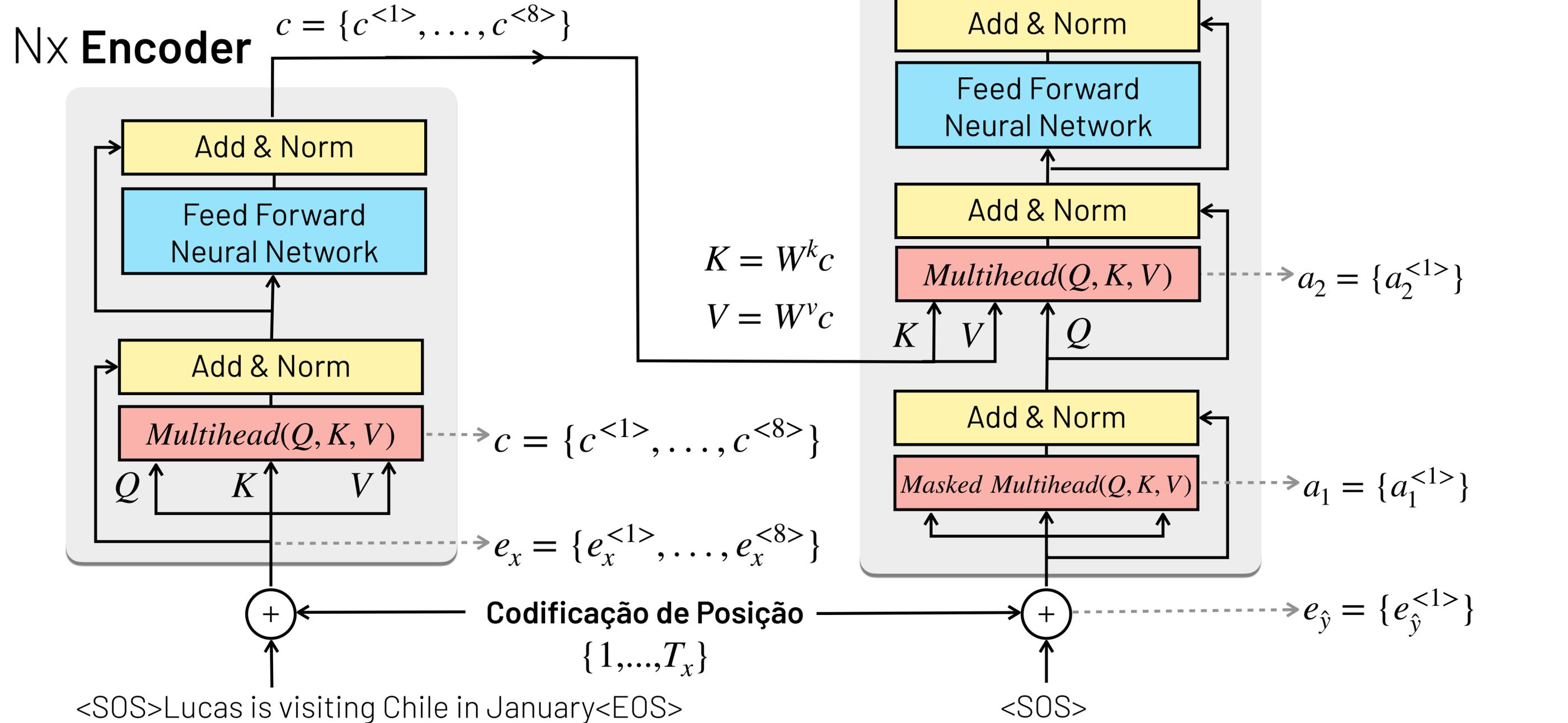
Exemplo

Decoder Nx

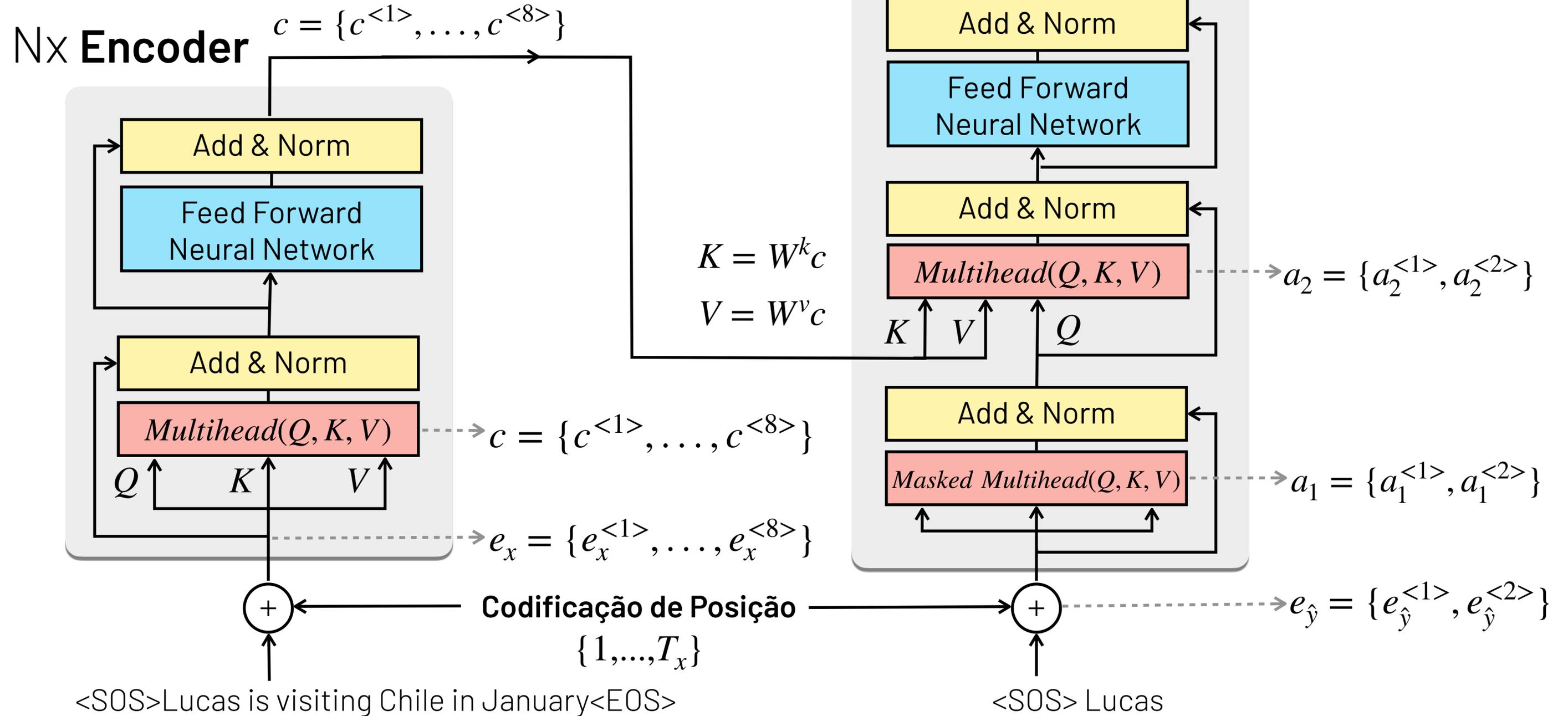


<SOS>Lucas is visiting Chile in January<EOS>

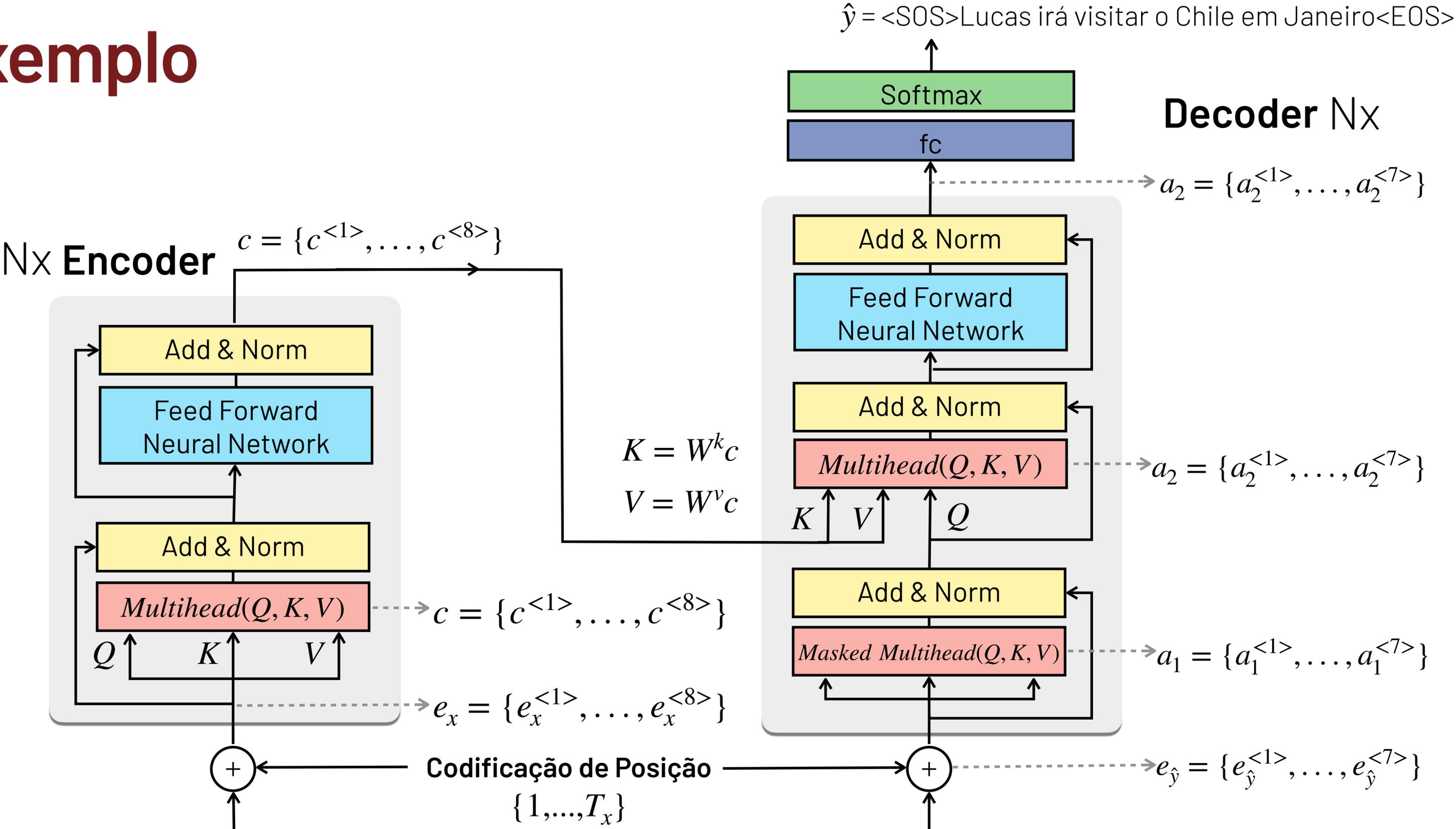
Exemplo



Exemplo



Exemplo

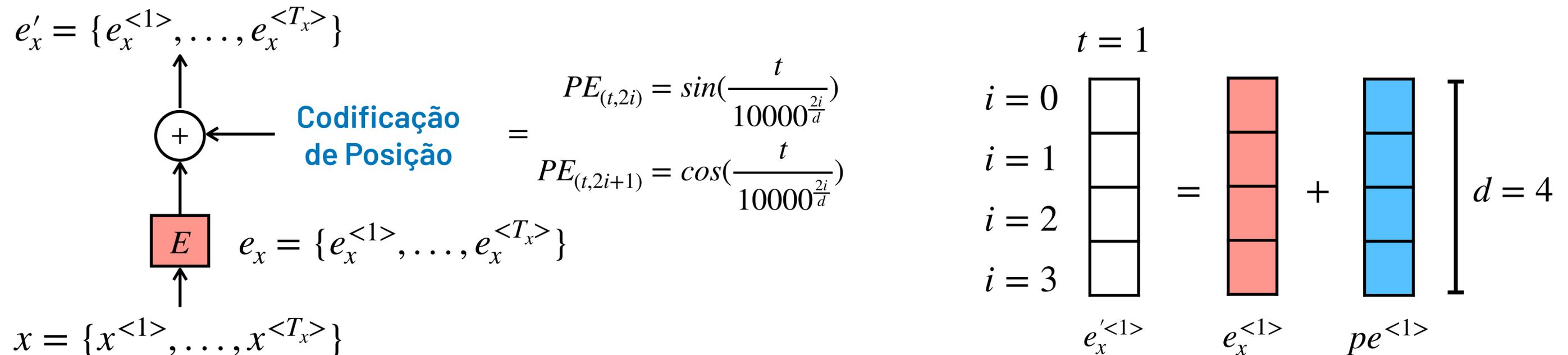


Codificação de Posição

O mecanismo de Auto-Atenção não considera as informações de posição das palavras.

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

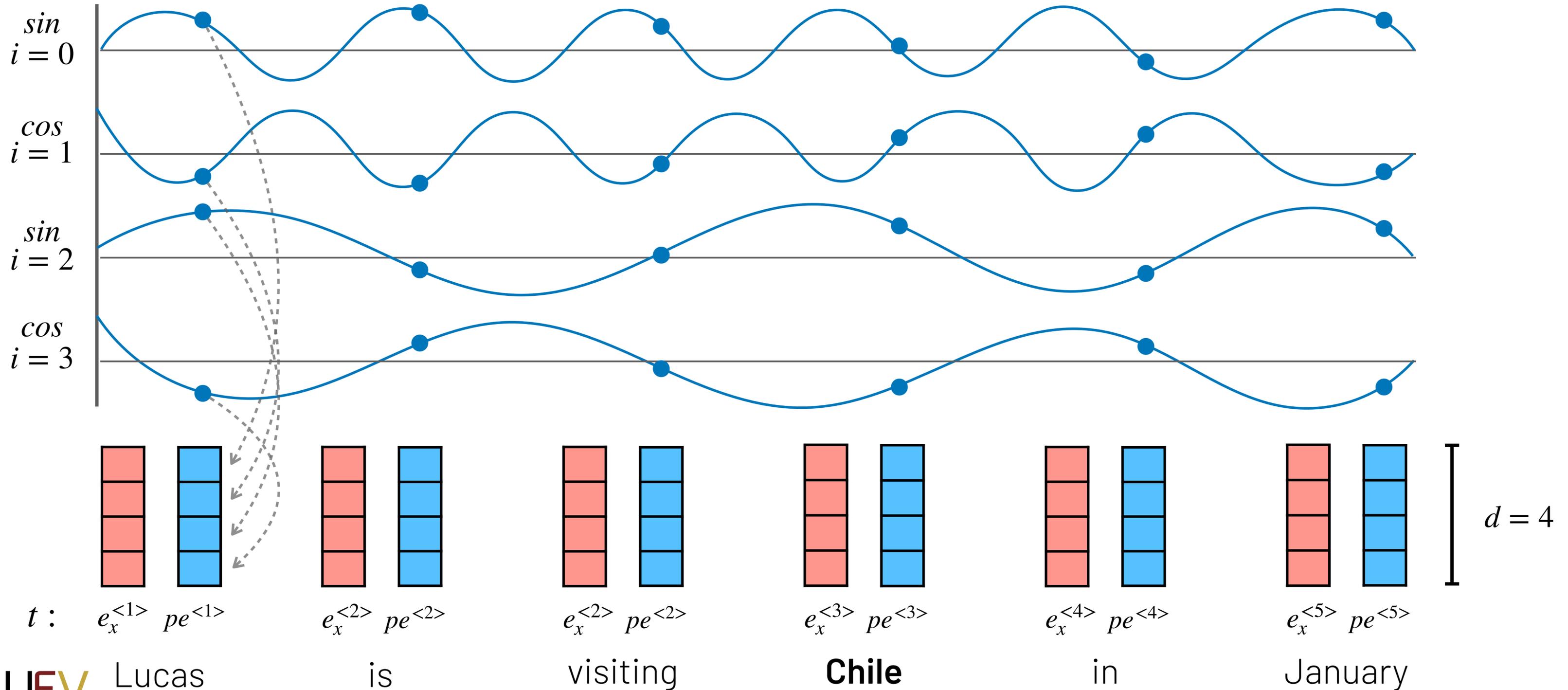
Para adicionar essa informação à representação aprendida c , o Encoder adiciona uma informação de posição à cada elemento $x^{<t>}$ da entrada $x = \{x^{<1>}, \dots, x^{<T_x>}\}$



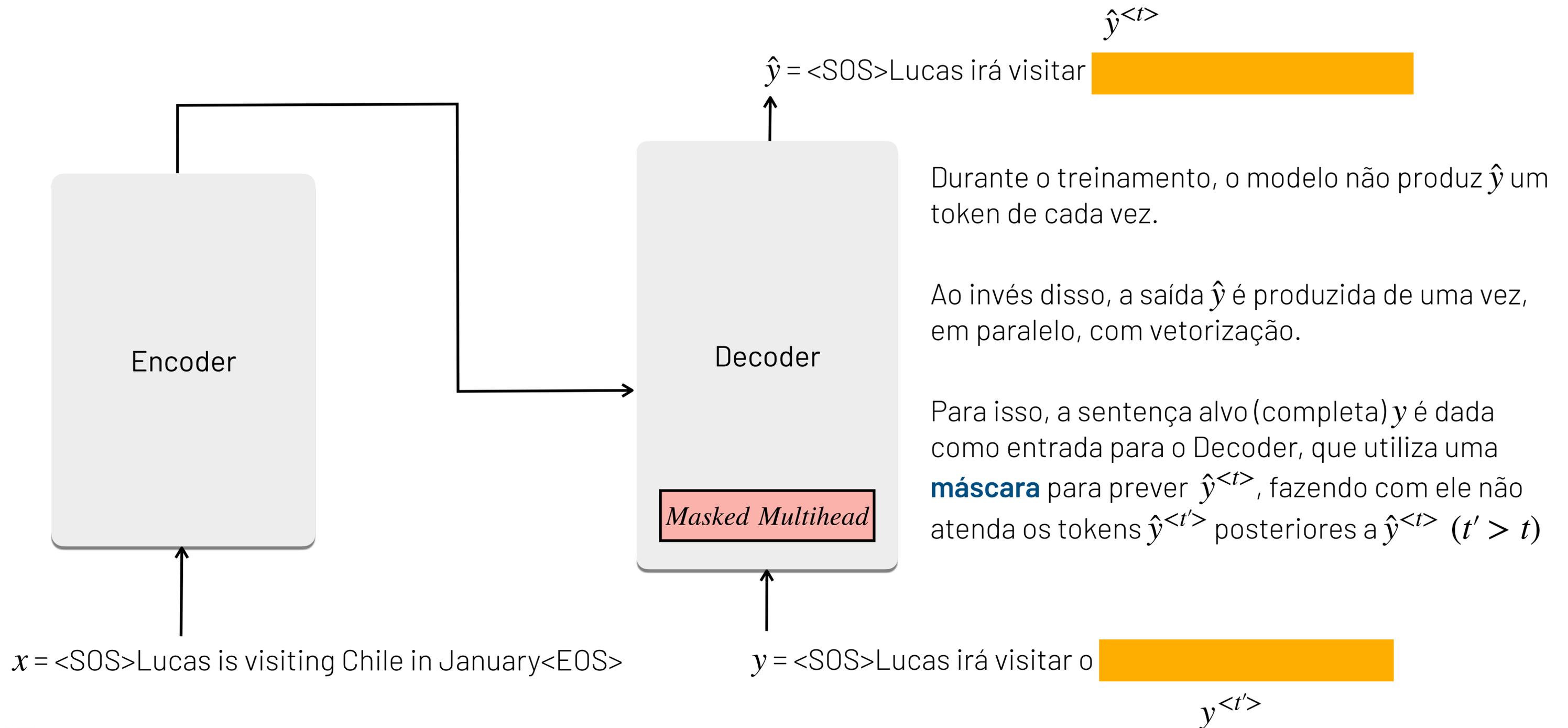
Codificação de Posição

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

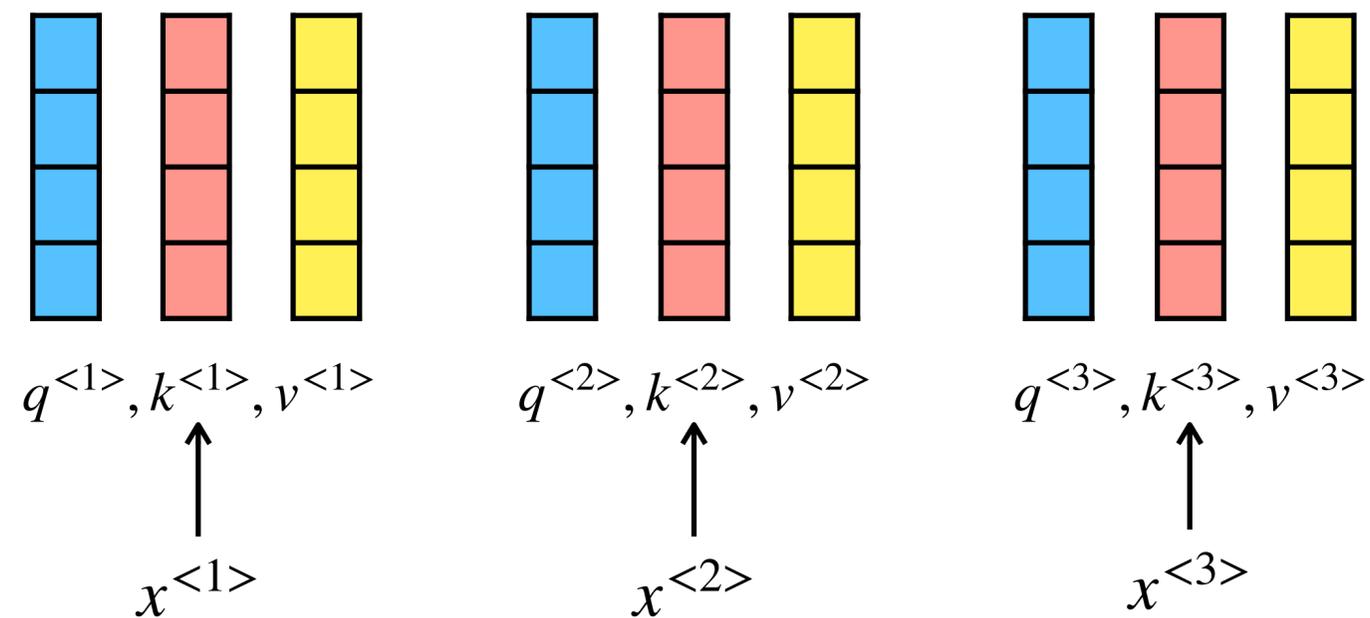
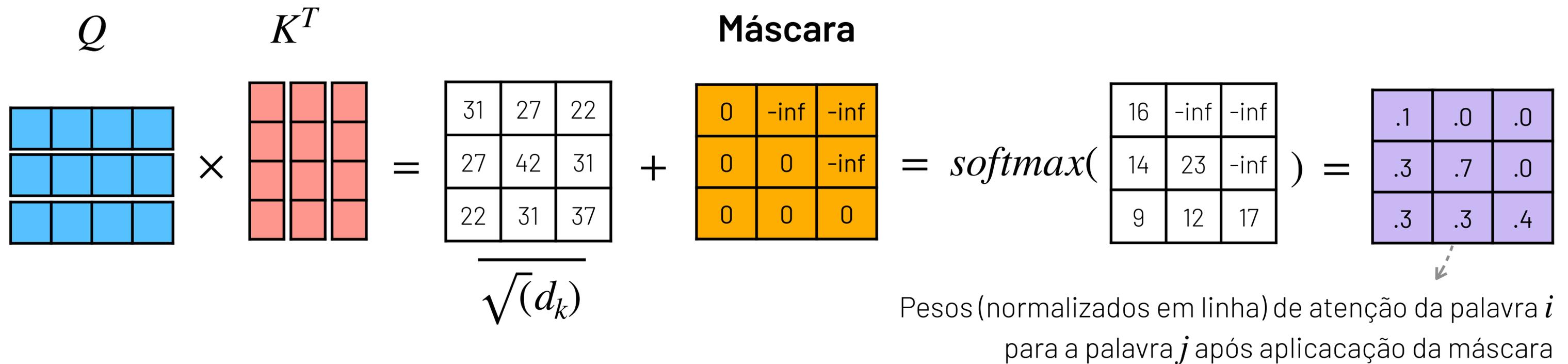
Se i ímpar
 $PE_{(t,2i+1)} = \cos\left(\frac{t}{10000^{\frac{2i}{d}}}\right)$



Treinamento (Masked Multi-head Attention)



Treinamento (Masked Multi-head Attention)



Próxima aula

A22: Estudo de Casos de Transformers

BERT e GPT