INF721

2023/2

UFV

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 Múltiplcas Cabeças (Multi-head Attention)
- Codificação de Posição
- Treinamento (Masked Multi-head Attention)

Tradução Automática

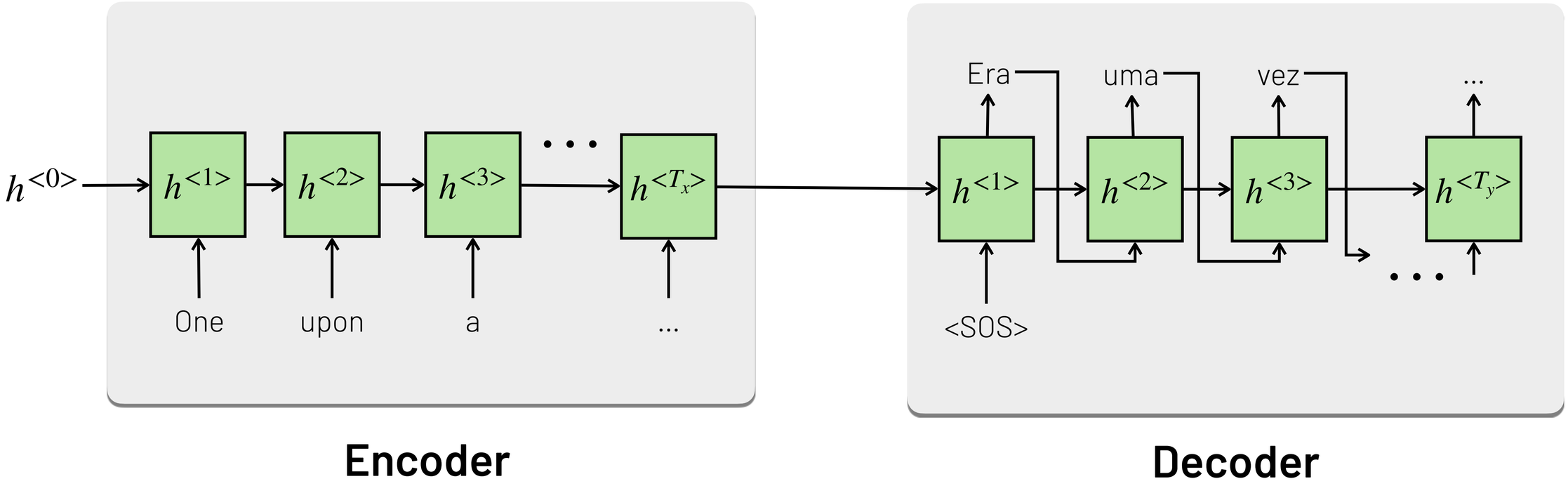
Conjunto de dados

Pares de sentenças de um idioma origem (x) para um idoma 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

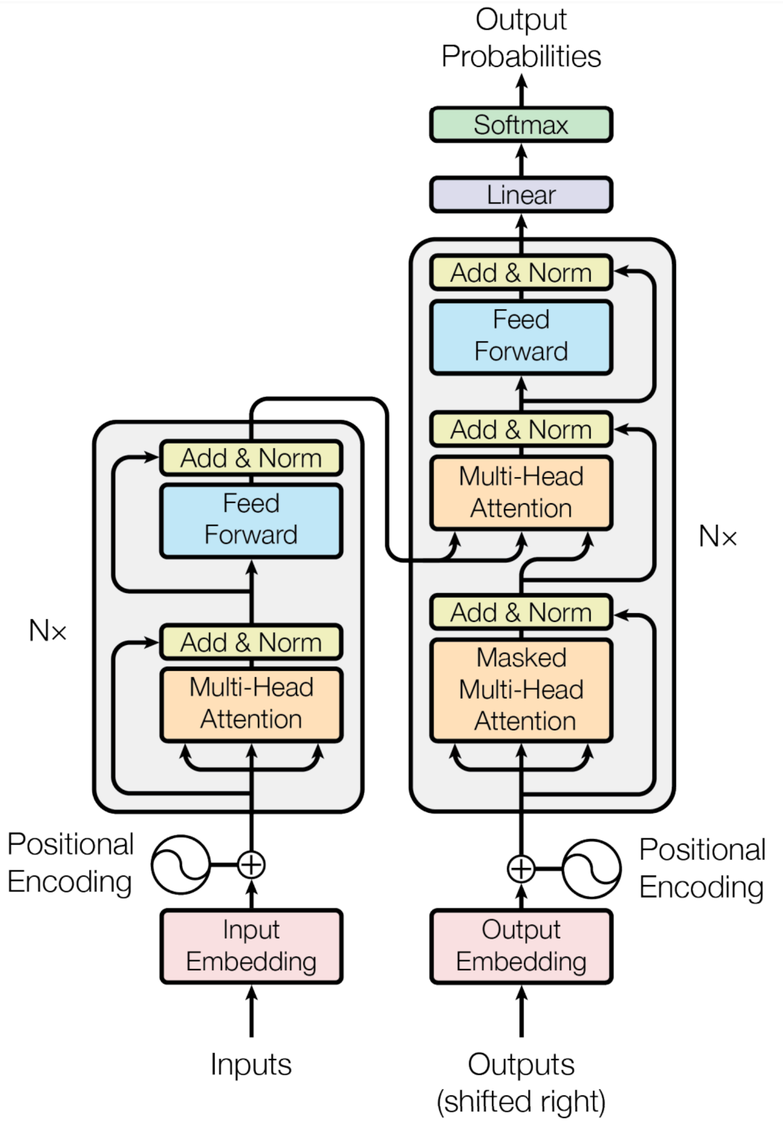


Attention is all you need

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:

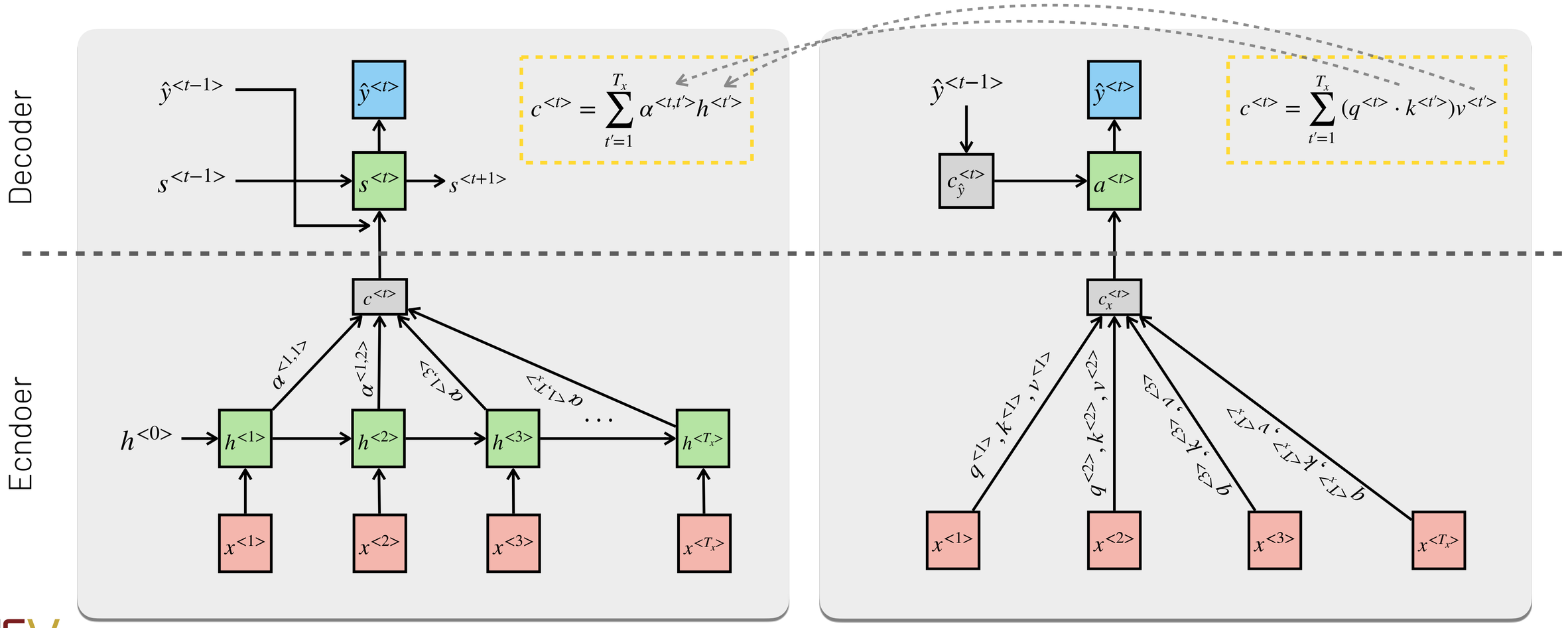
- Processamento de Linguagem Natural
- Visão Computacional
- Aprendizado por Reforço
- ...



Removendo Recorrência

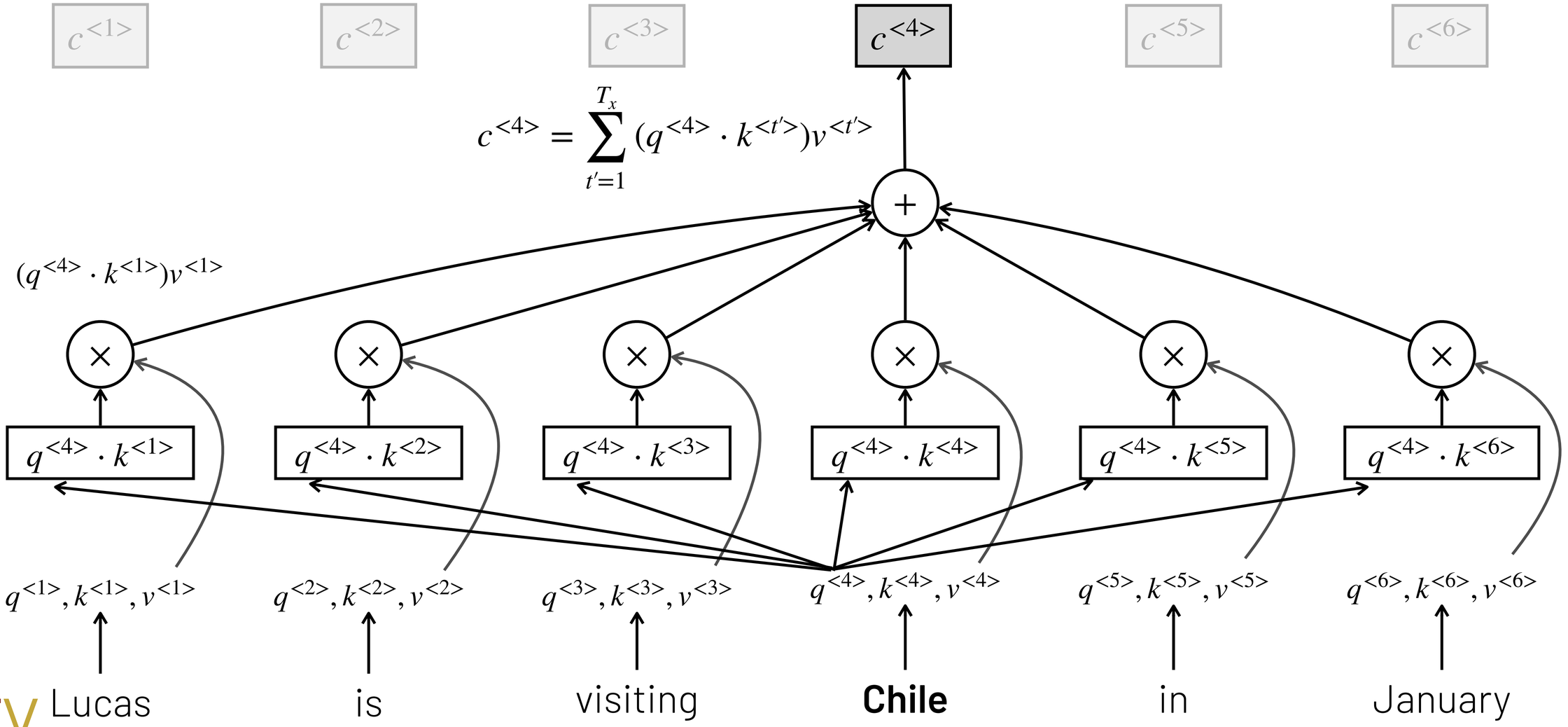
RNNs Atenção Aditiva

Transformers Auto-Atenção (Self Attention)



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

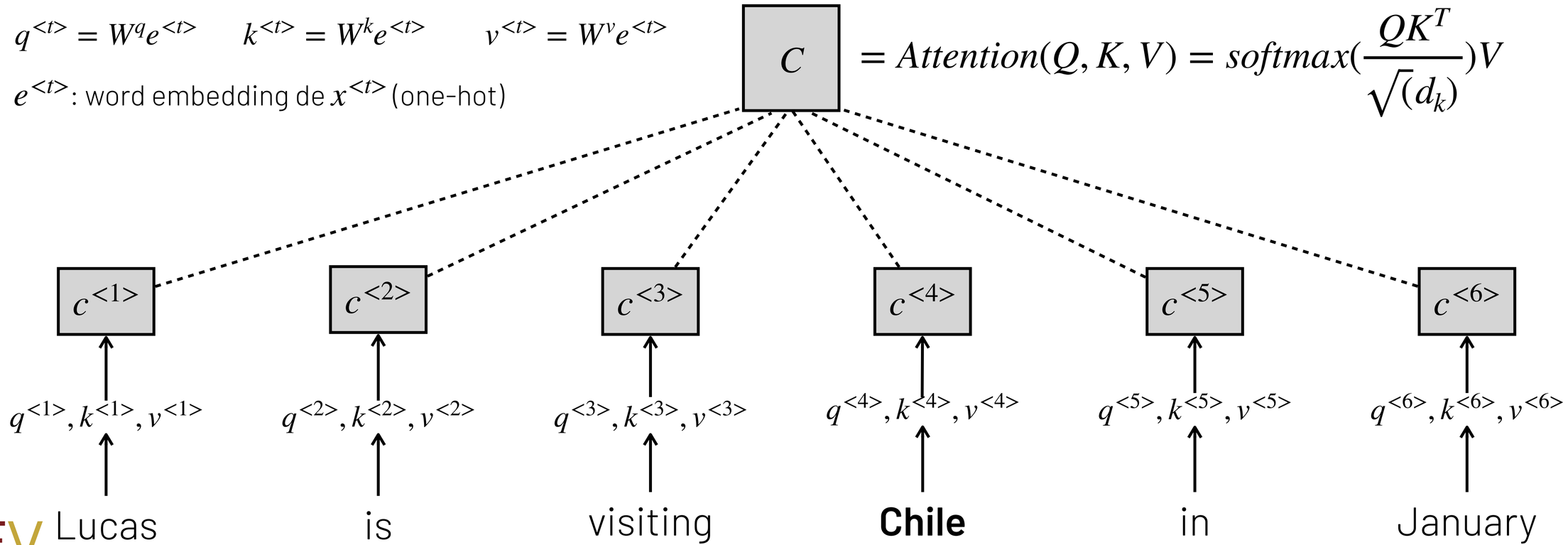
Query (Q)	Key (K)	Value (V)
$q^{<1>}$	$k^{<1>}$	$v^{<1>}$
$q^{<2>}$	$k^{<2>}$	$v^{<2>}$
...	...	...
$q^{<Tx>}$	$k^{<Tx>}$	$v^{<Tx>}$

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

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

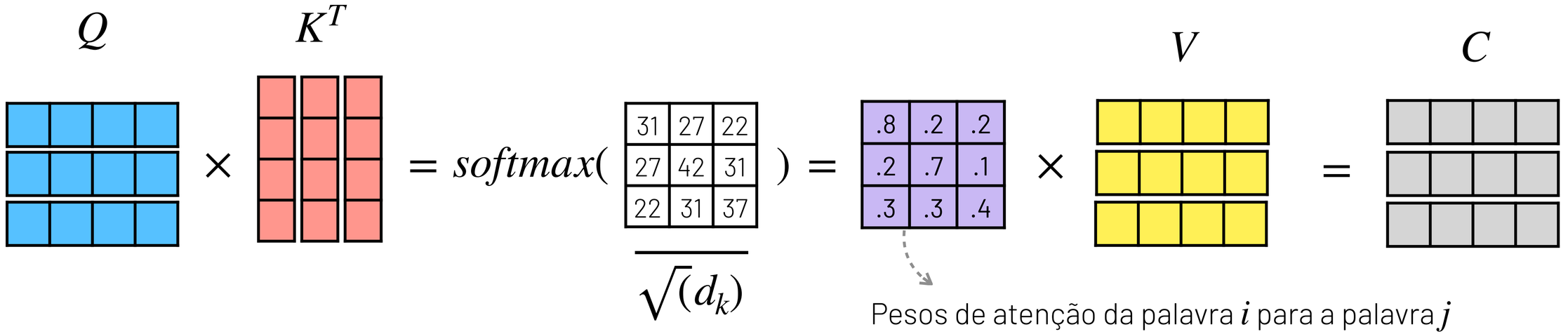
$e^{<t>}$: word embedding de $x^{<t>}$ (one-hot)

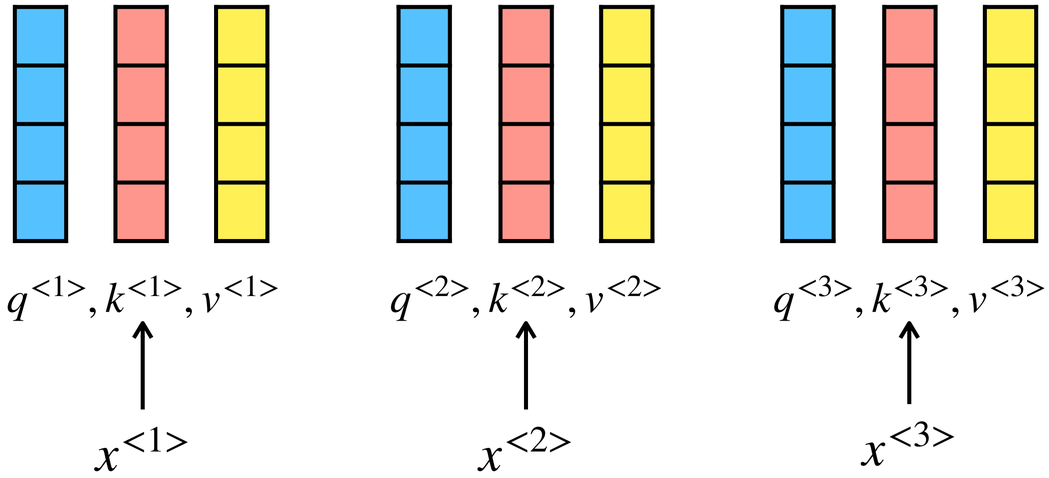
$$C = Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{(d_k)}})V$$



Auto-Atenção

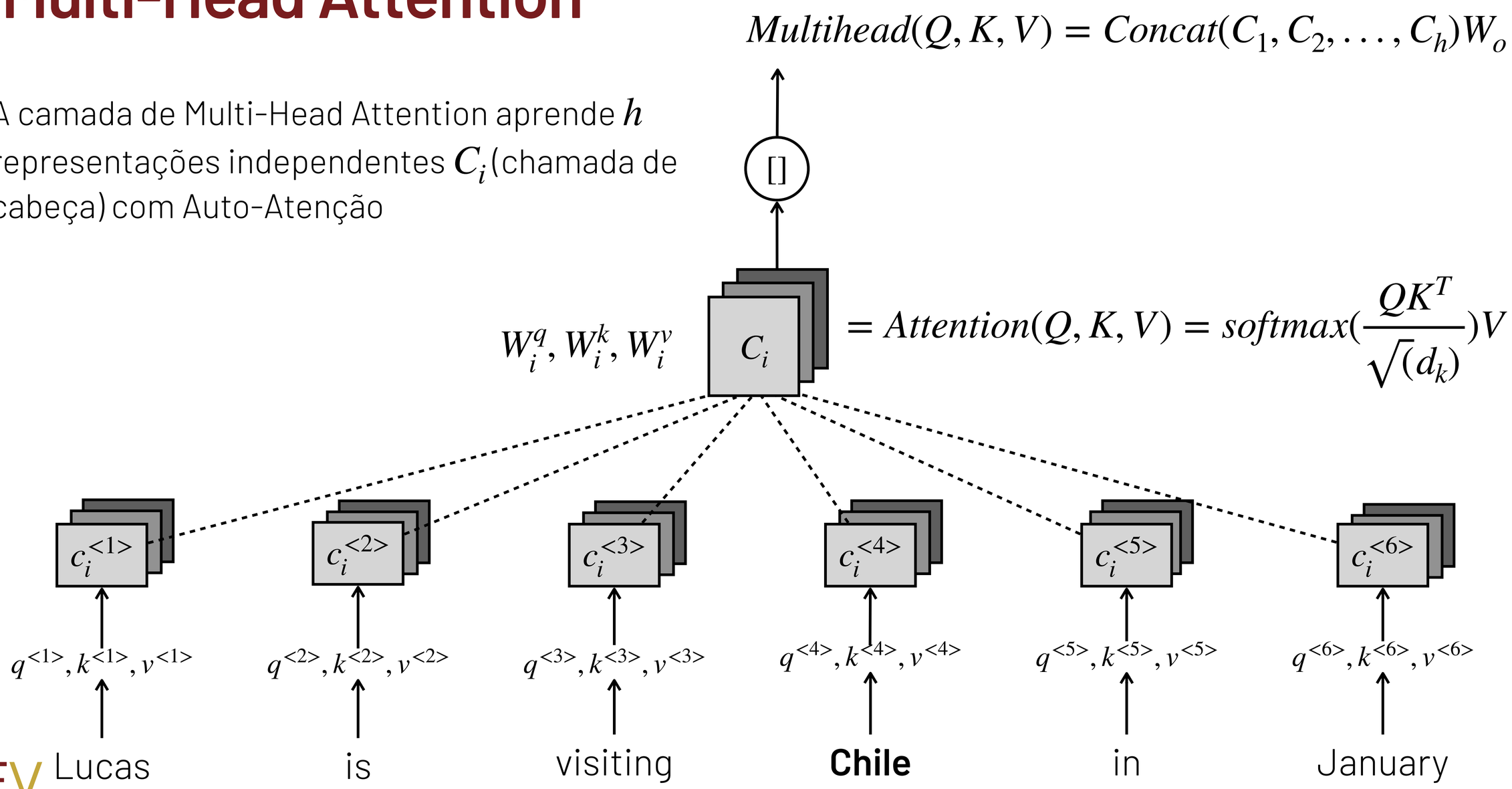
$$C = Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{(d_k)}})V$$





Multi-Head Attention

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

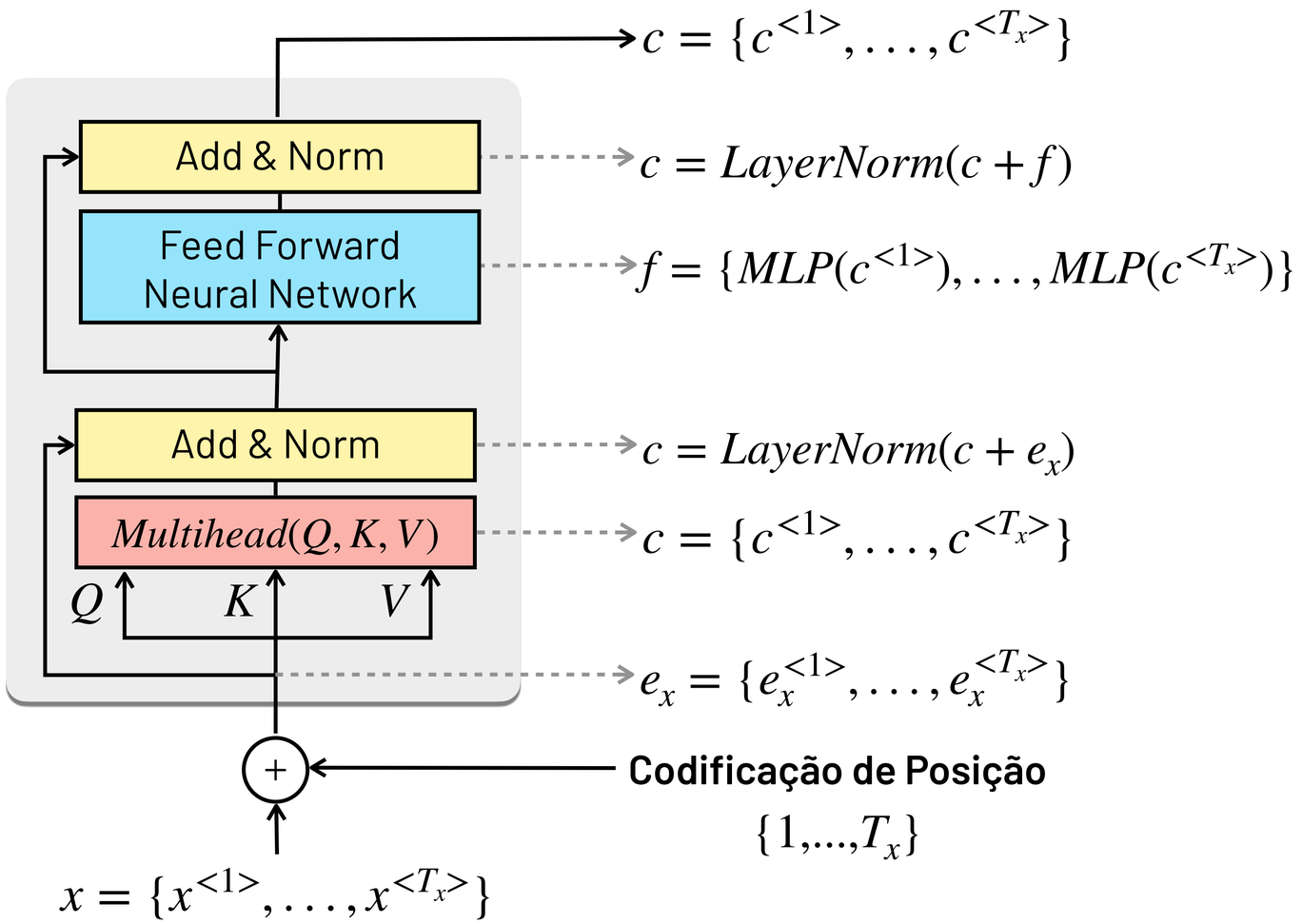


Encoder

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

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

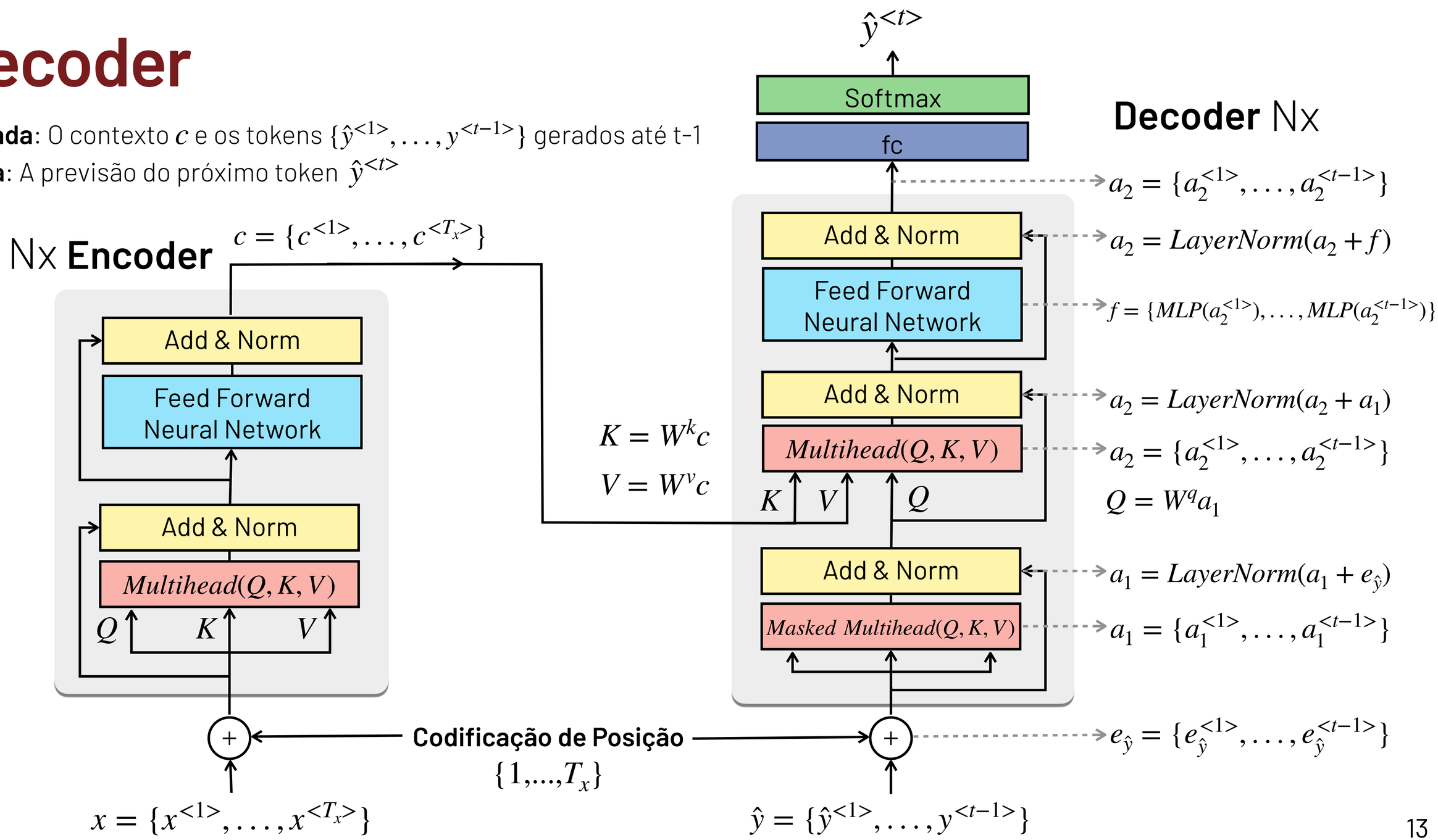
Para isso, ele aplica uma camada ***Multihead*** seguida de uma ***Feed Forward Neural Network*** (MLP). Ambas são normalizadas (***Norm***) e conectadas com as camadas anteriores residualmente (***Add***)



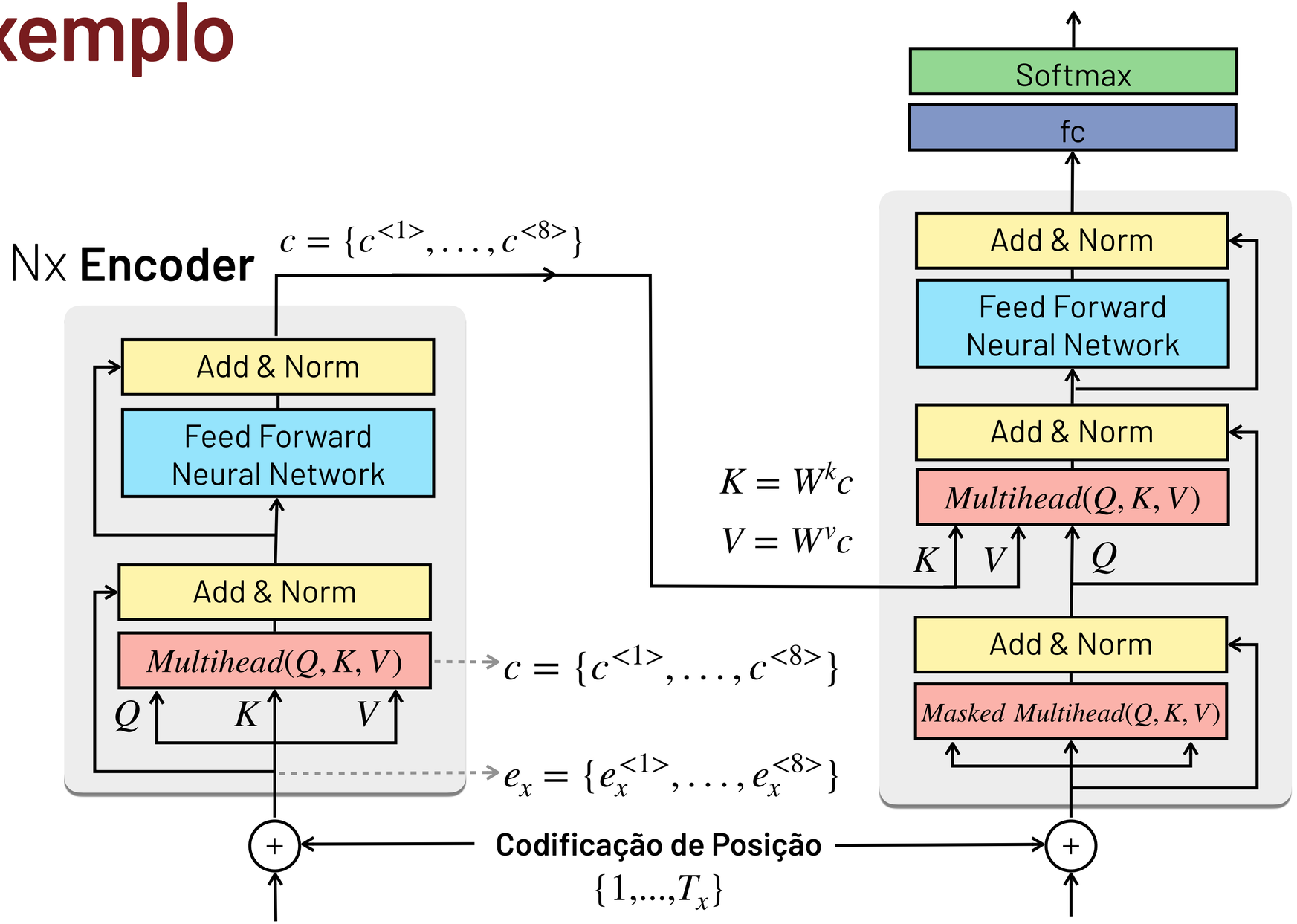
Decoder

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

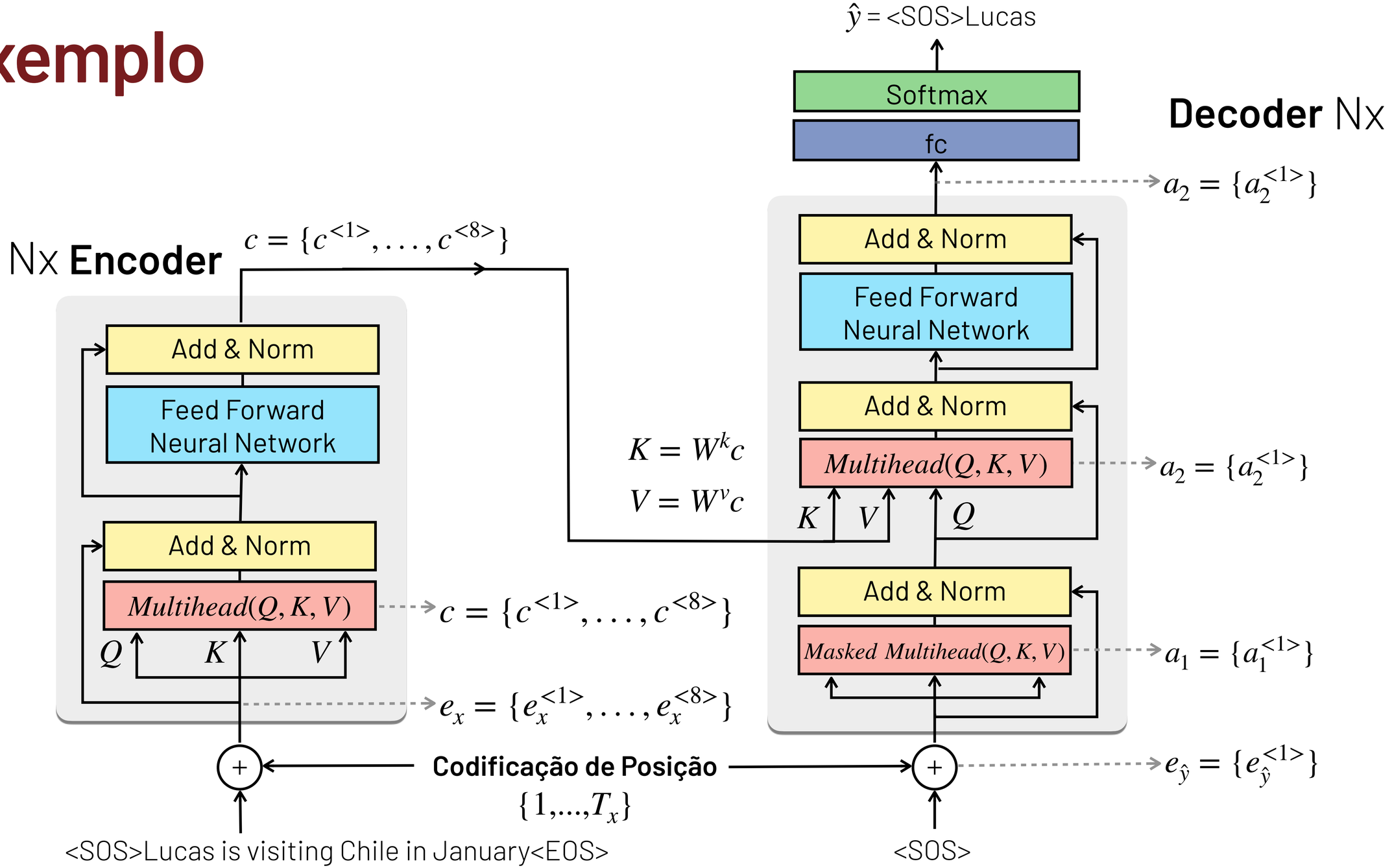
Saída: A previsão do próximo token $\hat{y}^{<t>}$



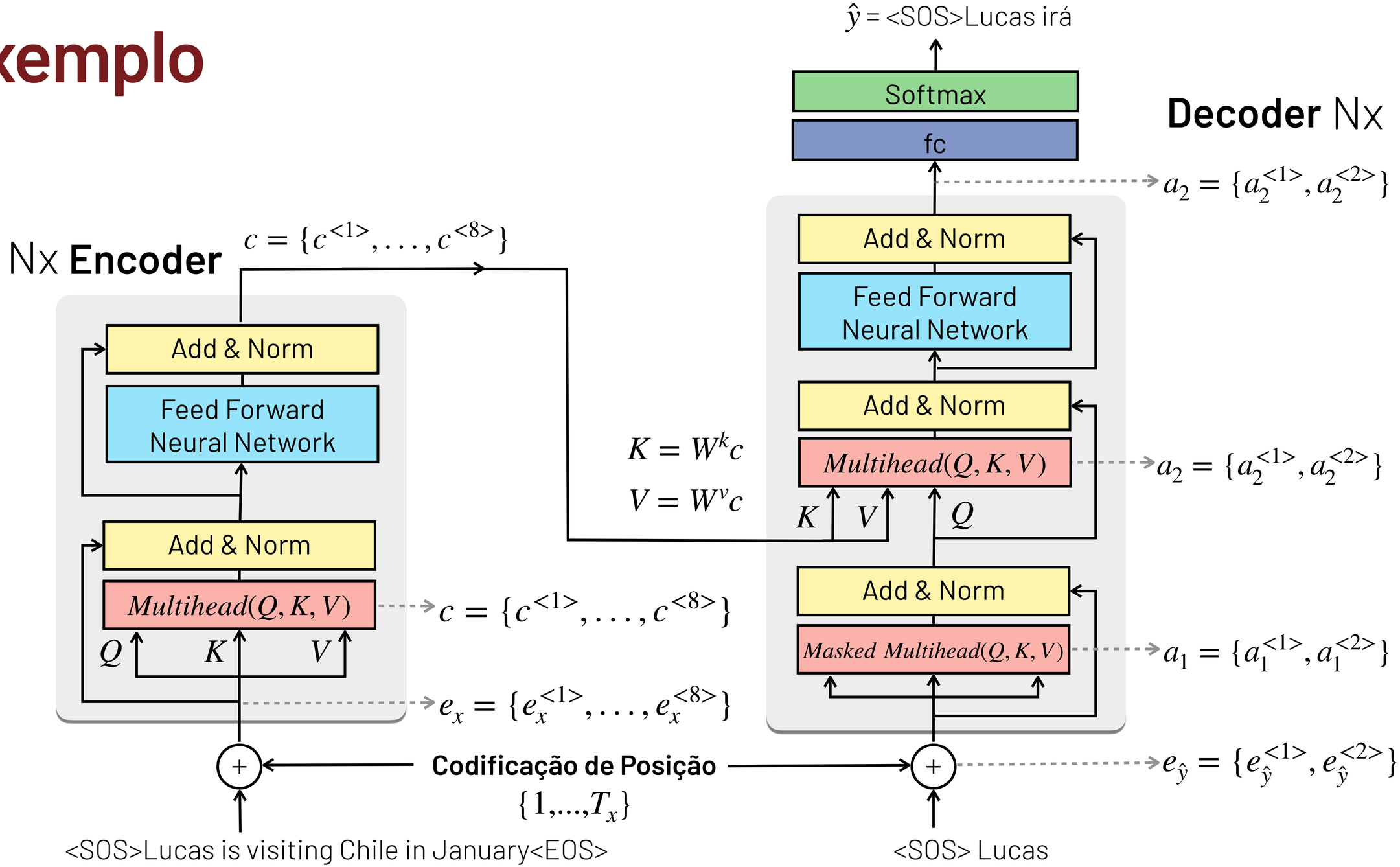
Exemplo


Softmax
fc
Add & Norm
Feed Forward Neural Network
Add & Norm
Multihead(Q, K, V)
Q K V
Add & Norm
Masked Multihead(Q, K, V)
Decoder Nx
Nx Encoder
c = {c<1>, . . . , c<8>}
Add & Norm
Feed Forward Neural Network
Add & Norm
Multihead(Q, K, V)
Q K V
K = W^k c
V = W^v c
c = {c<1>, . . . , c<8>}
e_x = {e_x<1>, . . . , e_x<8>}
Codificação de Posição
{1,...,T_x}
<SOS>Lucas is visiting Chile in January<EOS>

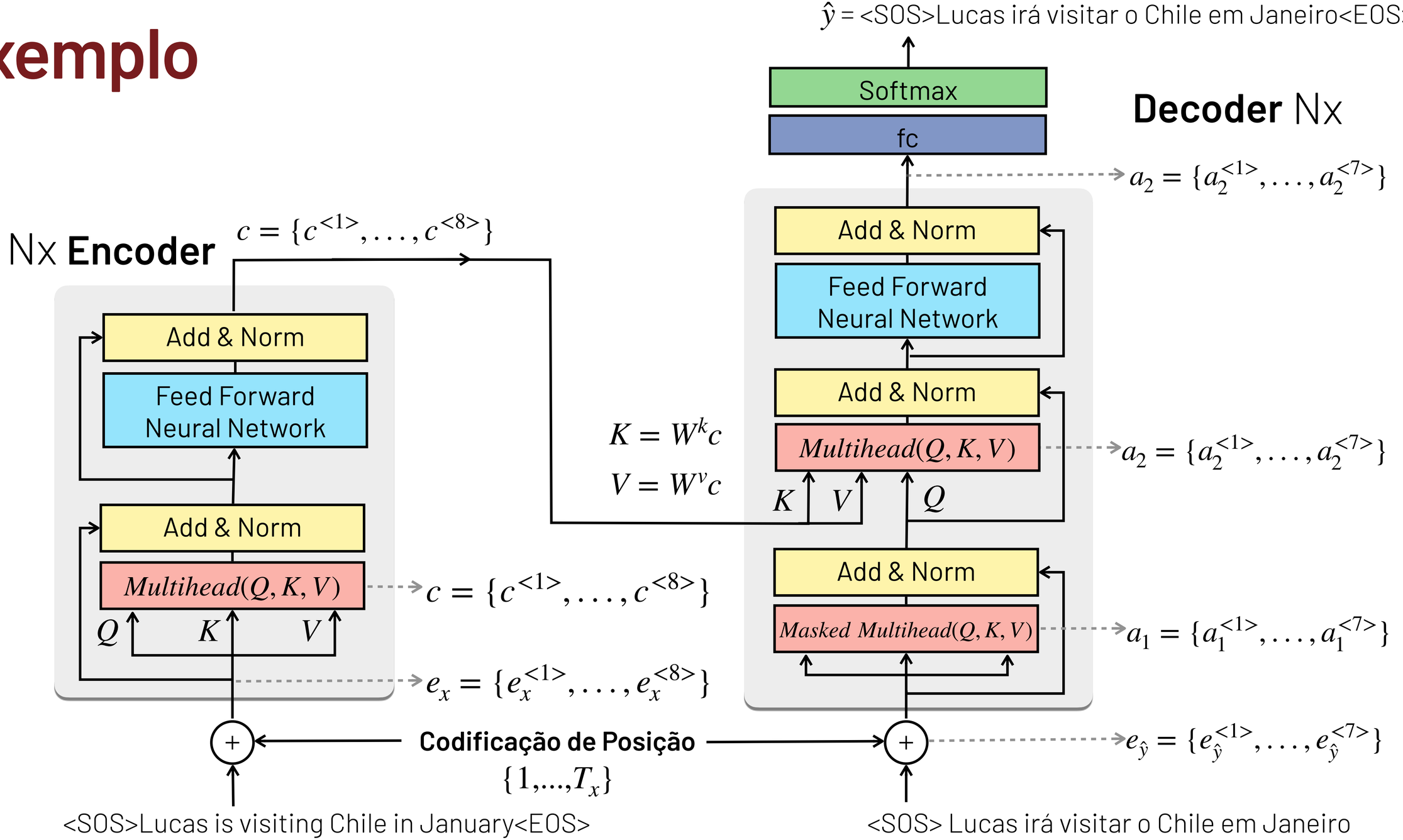
Exemplo


ŷ = <SOS>Lucas
Softmax
fc
Decoder Nx
a2 = {a2<1>}
Add & Norm
Feed Forward Neural Network
Add & Norm
Multihead(Q, K, V)
a2 = {a2<1>}
K
V
Q
Add & Norm
Masked Multihead(Q, K, V)
a1 = {a1<1>}
eŷ = {eŷ<1>}
Nx Encoder
c = {c<1>, . . . , c<8>}
Add & Norm
Feed Forward Neural Network
Add & Norm
Multihead(Q, K, V)
c = {c<1>, . . . , c<8>}
Q
K
V
ex = {ex<1>, . . . , ex<8>}
K = W^k c
V = W^v c
Codificação de Posição
{1,...,Tx}
<SOS>Lucas is visiting Chile in January<EOS>
<SOS>

Exemplo


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

Exemplo

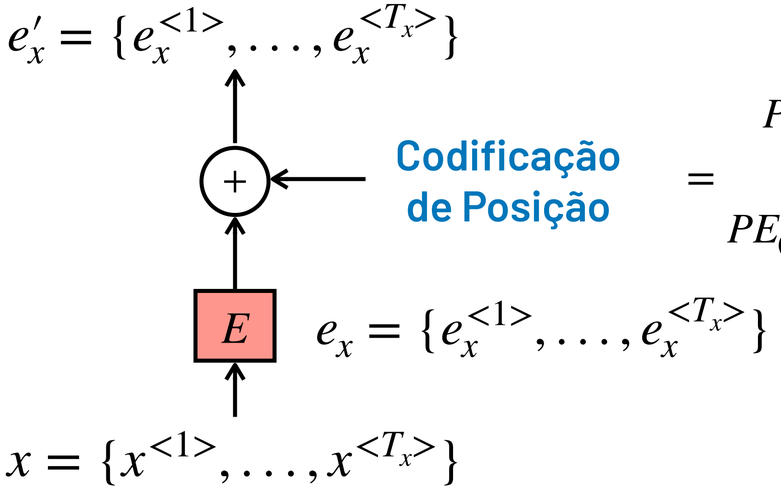

ŷ = <SOS>Lucas irá visitar o Chile em Janeiro<EOS>
Softmax
fc
Decoder Nx
a_2 = {a_2^<1>, ..., a_2^<7>}
Add & Norm
Feed Forward Neural Network
Add & Norm
Multihead(Q, K, V)
a_2 = {a_2^<1>, ..., a_2^<7>}
K V Q
Add & Norm
Masked Multihead(Q, K, V)
a_1 = {a_1^<1>, ..., a_1^<7>}
e_ŷ = {e_ŷ^<1>, ..., e_ŷ^<7>}
Nx Encoder
c = {c^<1>, ..., c^<8>}
Add & Norm
Feed Forward Neural Network
Add & Norm
Multihead(Q, K, V)
Q K V
c = {c^<1>, ..., c^<8>}
e_x = {e_x^<1>, ..., e_x^<8>}
K = W^k c
V = W^v c
Codificação de Posição
{1,...,T_x}
<SOS>Lucas is visiting Chile in January<EOS>
<SOS> Lucas irá visitar o Chile em Janeiro

Codificação de Posição

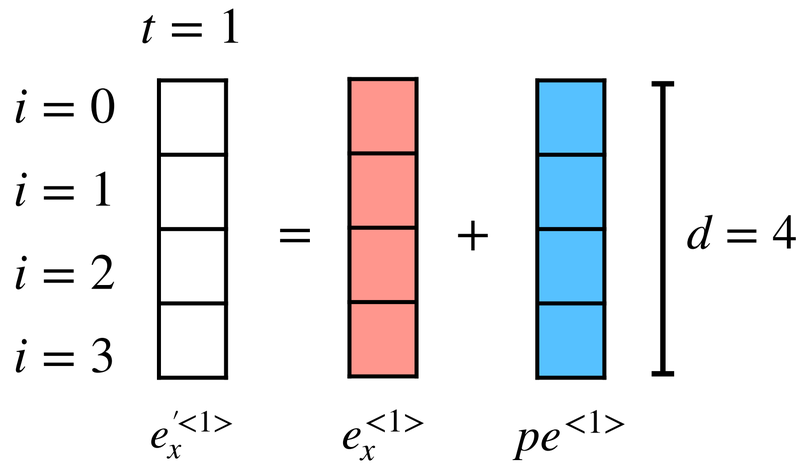
O mecâmismo de Auto-Atenção não considera as informações de posição das palavras.

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

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



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



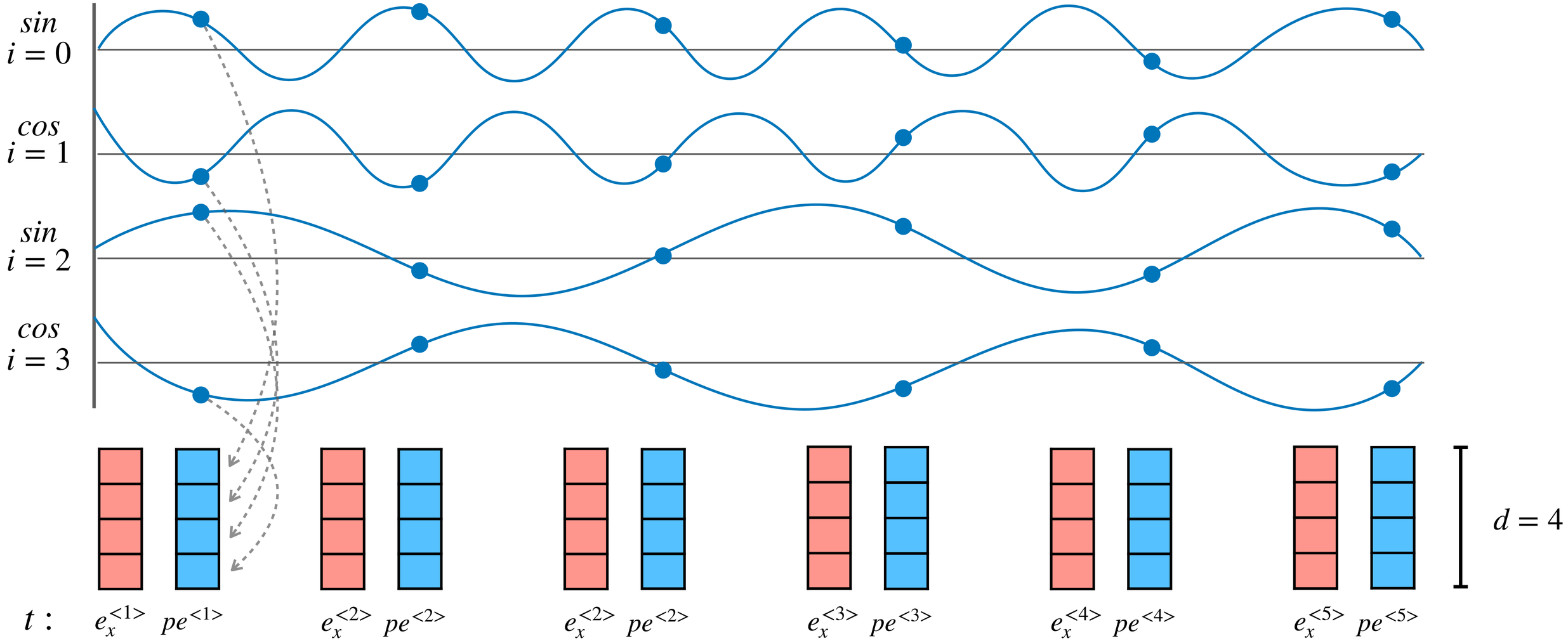
Codificação de Posição

Se i par

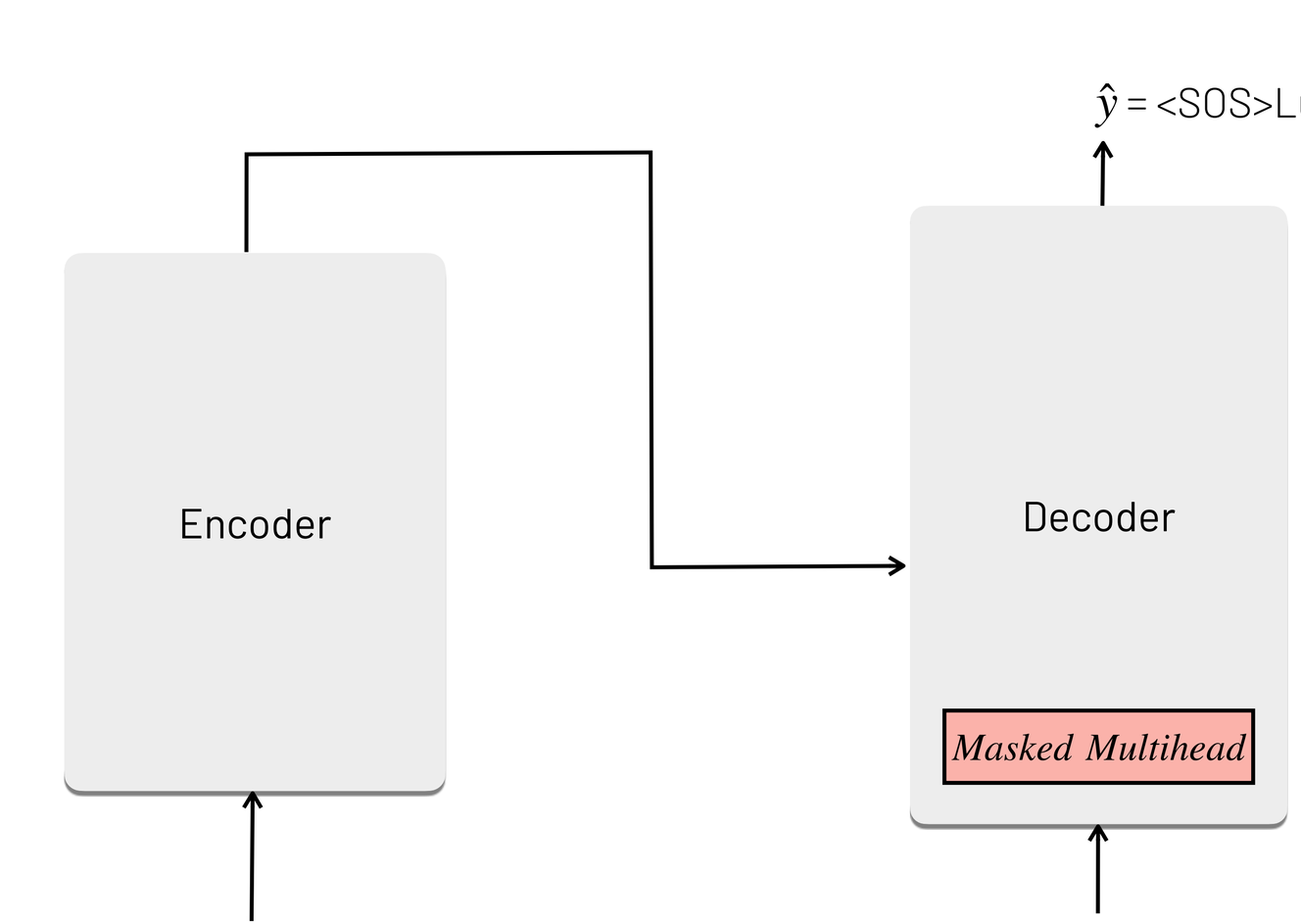
$$PE_{(t,2i)} = sin(\frac{t}{10000^{\frac{2i}{d}}})$$

Se i ímpar

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



Treinamento (Masked Multi-head Attention)

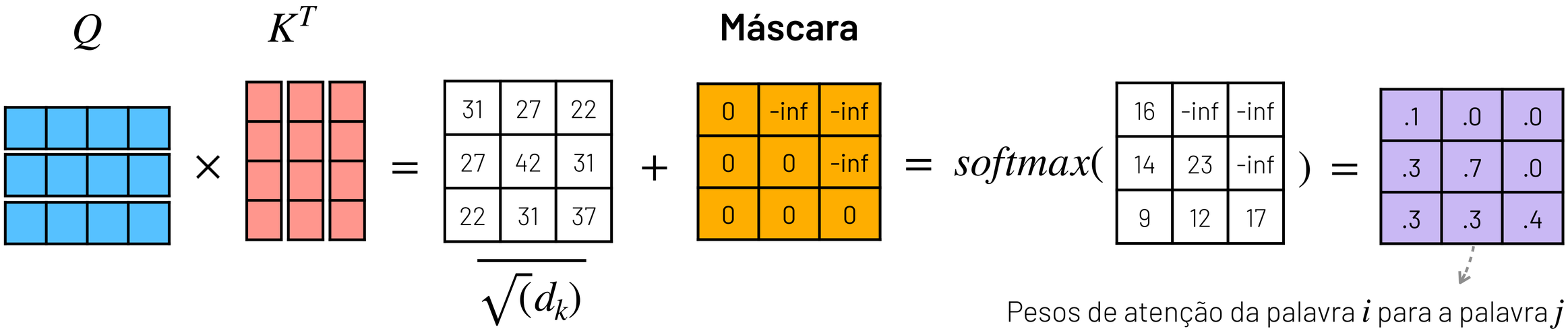


Durante o treinamento, o modelo não produz $\hat{y}$ um token de cada vez.

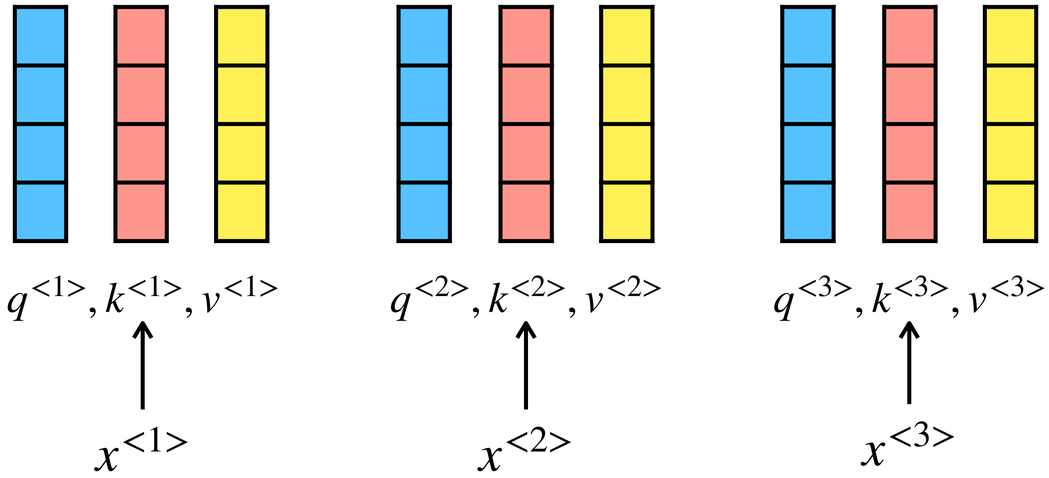
Ao invés disso, a saída $\hat{y}$ é produzida de uma vez, em paralelo, com vetorização.

Para isso, a sentença alvo (completa) y é dada como entrada para o Decoder, que utiliza uma **máscara** para prever $\hat{y}^{<t>}$, fazendo com ele não atenda os tokens $\hat{y}^{<t'>}$ posteriores a $\hat{y}^{<t>}$ $(t' > t)$

Treinamento (Masked Multi-head Attention)



Pesos de atenção da palavra i para a palavra j após aplicacação da máscara



Próxima aula

A22: Estudo de Casos de Transformers

BERT e GPT