MAC5921 – Deep Learning

Aula 12 – 03/10/2023

Transformers

Nina S. T. Hirata

1. **Redes neurais convencionais** (*fully connected*)

2. **Redes neurais convolucionais** (CNN)

3. **Redes neurais recorrentes** (RNN)

Transformers – usa mecanismos de attention

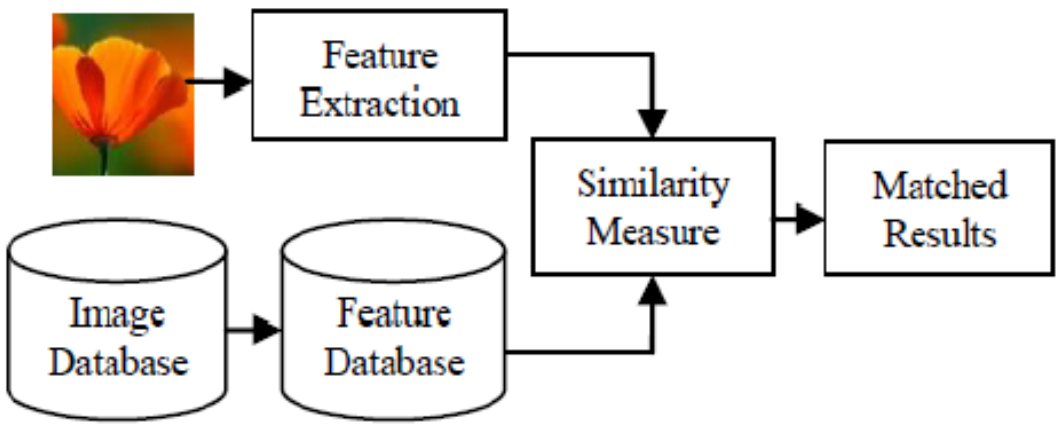
Antes, porém, uma digressão

Como funcionam buscas de objetos complexos?

- Imagens
- Documentos
- Vídeos
- Áudio
- etc

Information retrieval systems

Exemplo: Busca de imagens em um banco de dados



Métricas de similaridade (dados multidimensionais)

https://en.wikipedia.org/wiki/Similarity_measure

Indexação multidimensional

Christian Böhm, Stefan Berchtold, and Daniel A. Keim. 2001. **Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases**. ACM Comput. Surv. 33, 3 (September 2001), 322–373. https://doi.org/10.1145/502807.502809

Similaridade coseno – fim da digressão

Comumente usado para comparar dois vetores de *features*

The cosine of two non-zero vectors can be derived by using the Euclidean dot product formula:

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \, \|\mathbf{B}\| \cos\theta$$

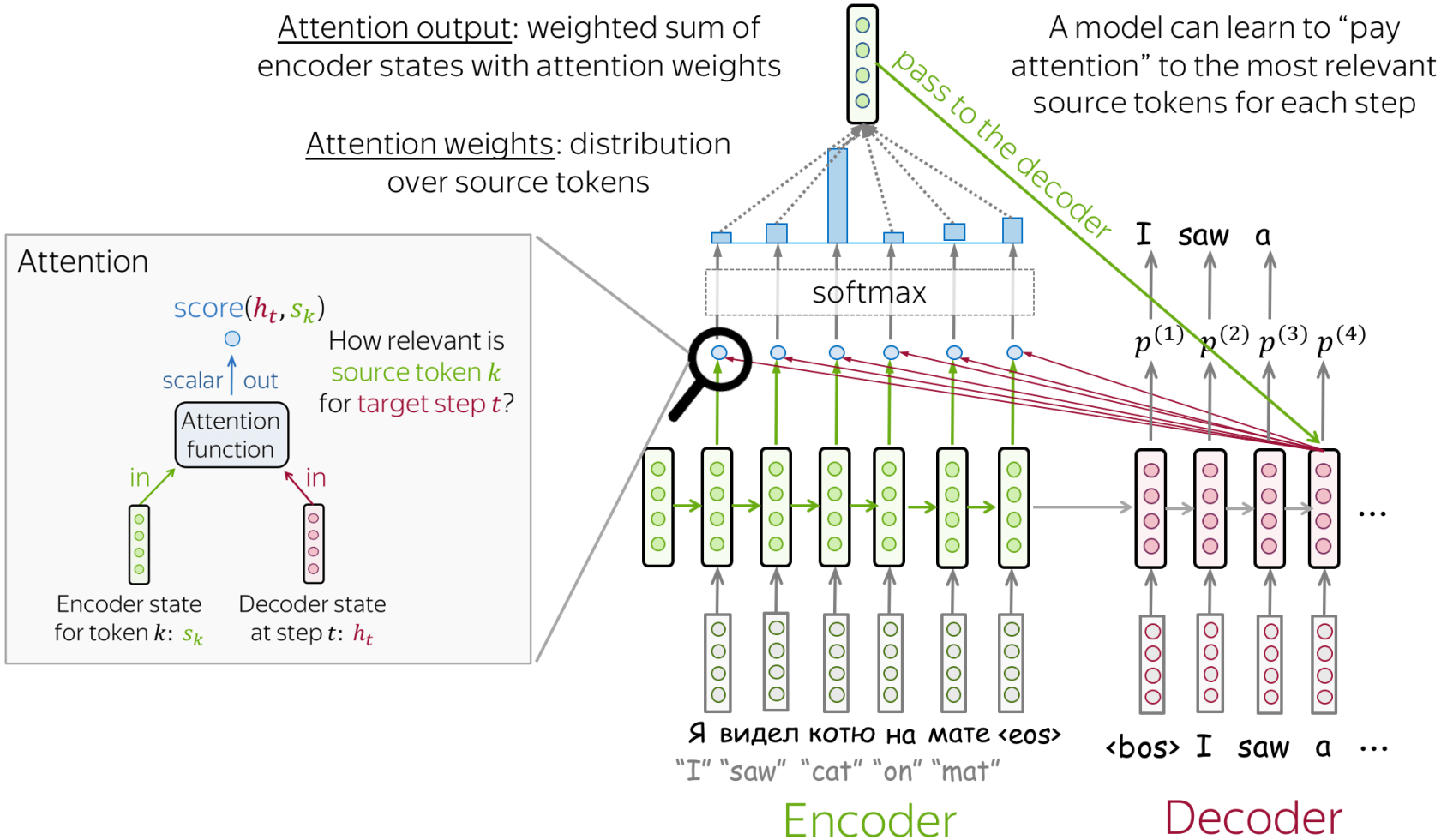
Given two n-dimensional vectors of attributes, **A** and **B**, the cosine similarity, $\cos(\theta)$, is represented using a dot product and magnitude as

$$\text{cosine similarity} = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}},$$

where A_i and B_i are the ith components of vectors **A** and **B**, respectively.

Fonte: Wikipedia

Attention em redes recorrentes para tarefas seq2seq



https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Estados do econder: s_k

Estados do decoder: $h_t = f(h_{t-1}, \hat{y}_{t-1}, c_t)$

Saída do decoder: $\hat{y}_t = g(\hat{y}_{t-1}, h_t, c_t)$

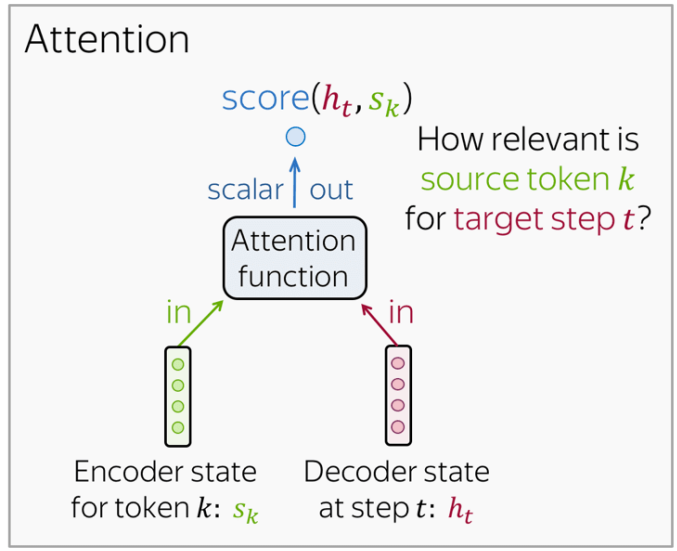
Vetor de contexto:

$$c_t = \sum_{k=1}^{T} \alpha_{tk} s_k$$

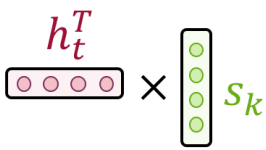
Escore de alinhamento e pesos:

$$e_{tk} = a(h_{t-1}, s_k) \qquad \alpha_{tk} = \frac{exp(e_{tk})}{\sum_{j=1}^{T} exp(e_{tj})}$$

Exemplos de função de alinhamento

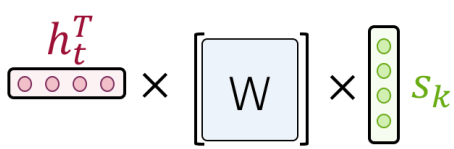


Dot-product



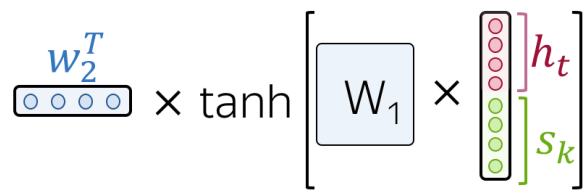
$$\text{score}(h_t, s_k) = h_t^T \; s_k$$

Bilinear



$$\text{score}(h_t, s_k) = h_t^T \; W s_k$$

Multi-Layer Perceptron



$$\text{score}(h_t, s_k) = w_2^T \cdot \tanh(W_1[h_t, s_k])$$

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Attention seria uma convolução com kernel igual ao tamanho da sequência de entrada??

Attention seria uma convolução com kernel igual ao tamanho da sequência de entrada??

O kernel de convolução, após o treinamento, tem pesos fixos. Assim, independentemente do passo t na decodificação, sempre olharia os estados do *encoder* da mesma forma

O *attention* calcula dinamicamente o peso a ser dado para cada um dos estados do *encoder*

Ponto fraco das RNNs do tipo encoder-decoder

Recorrência – impossibilita paralelização

Transformers (2017)

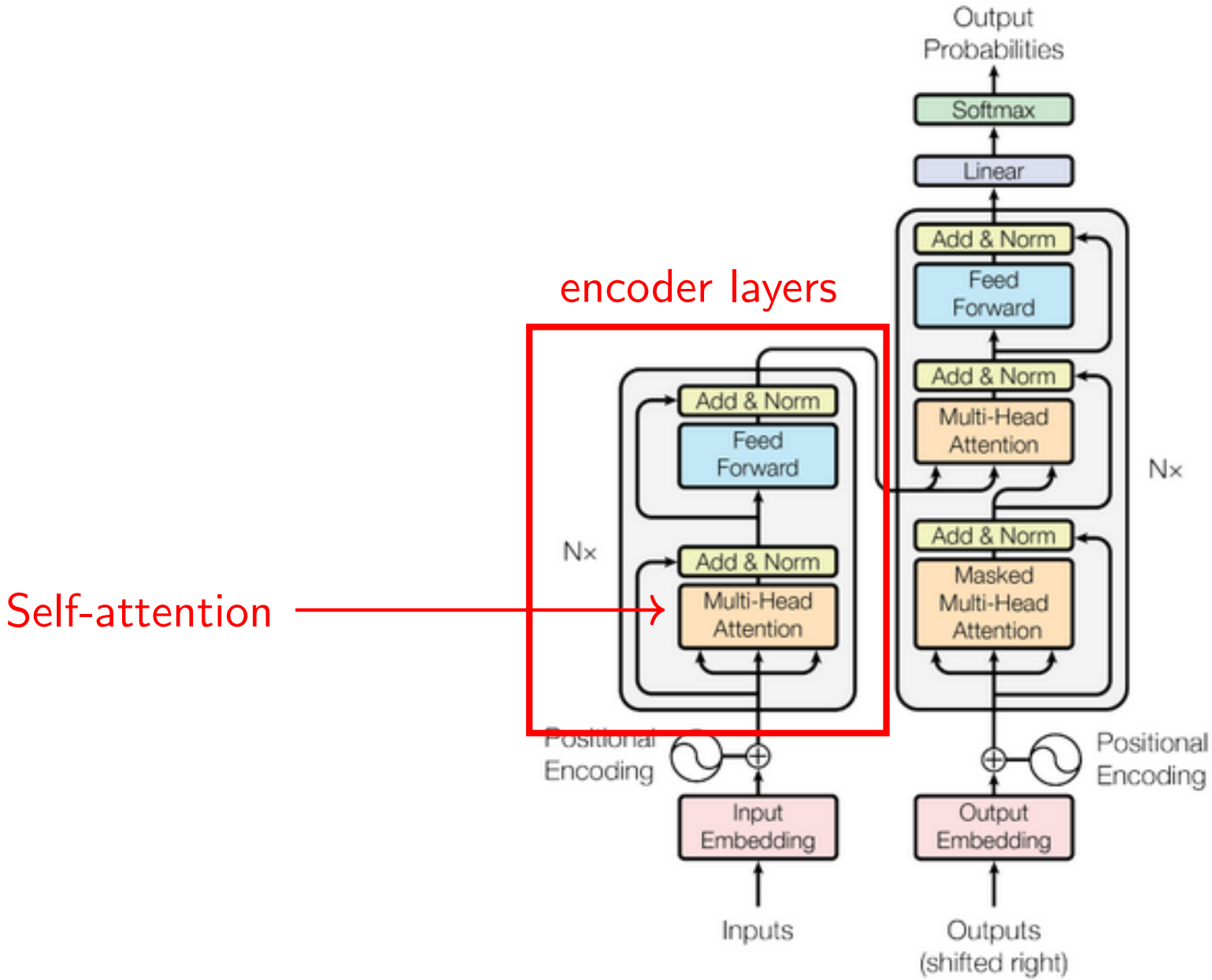
Attention Is All You Need

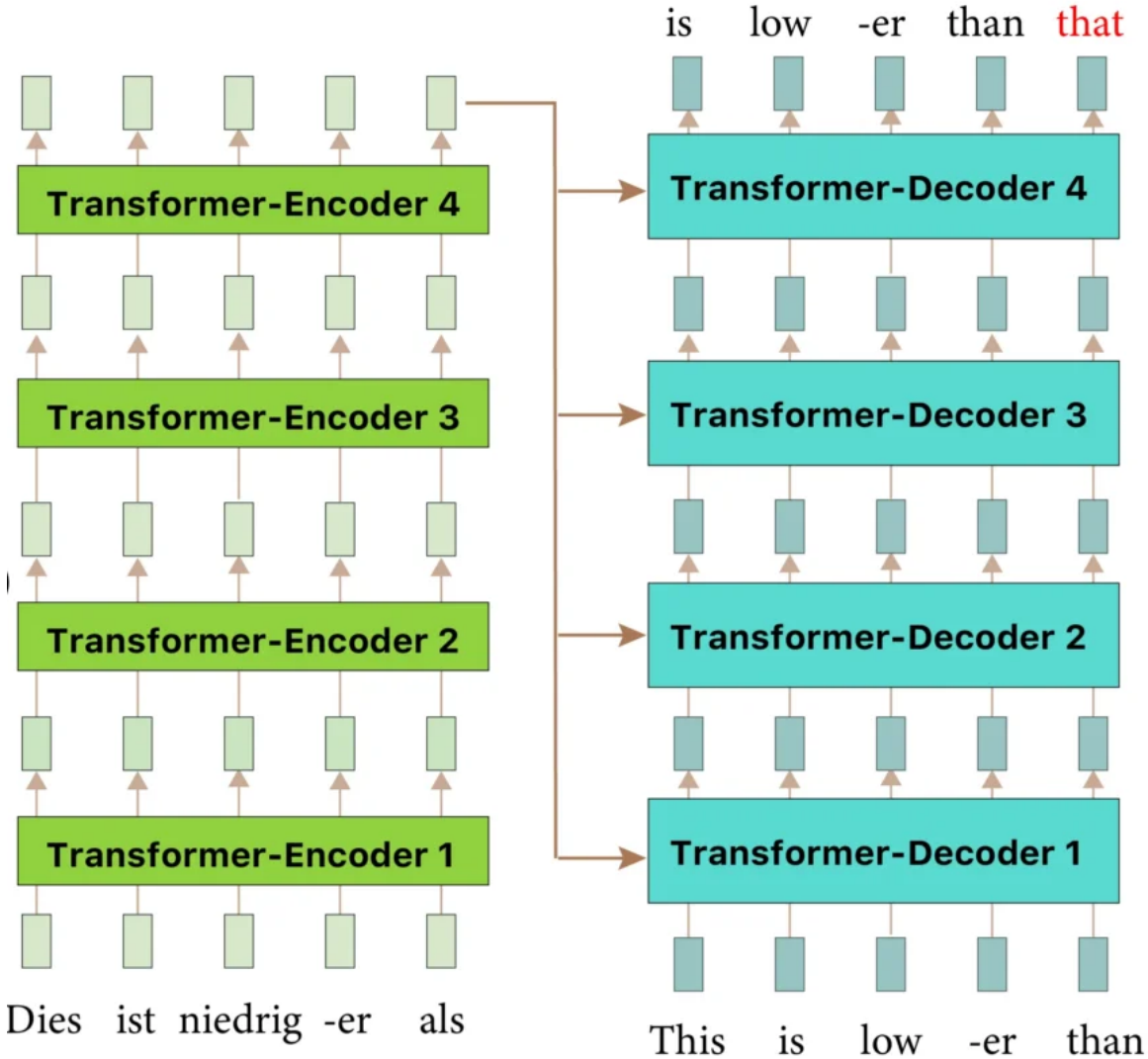
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones

Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin

https://arxiv.org/abs/1706.03762

– elimina recorrência + introduz self-attention


encoder layers
Self-attention
Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)



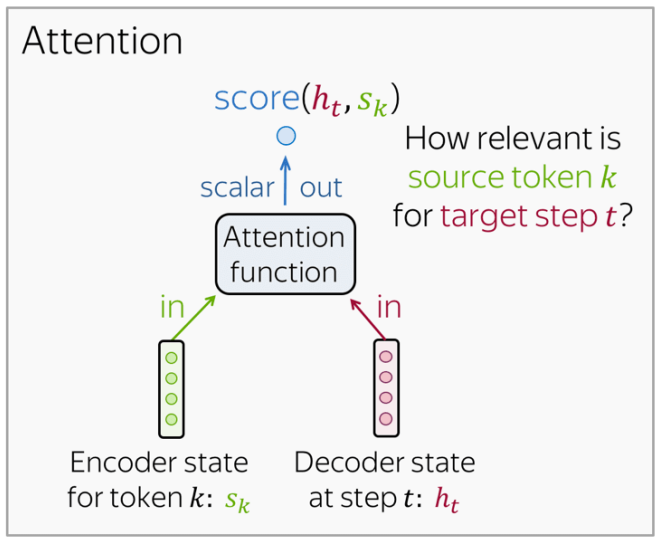
https://cvml-expertguide.net/terms/dl/seq2seq-translation/transformer/ (adaptado)

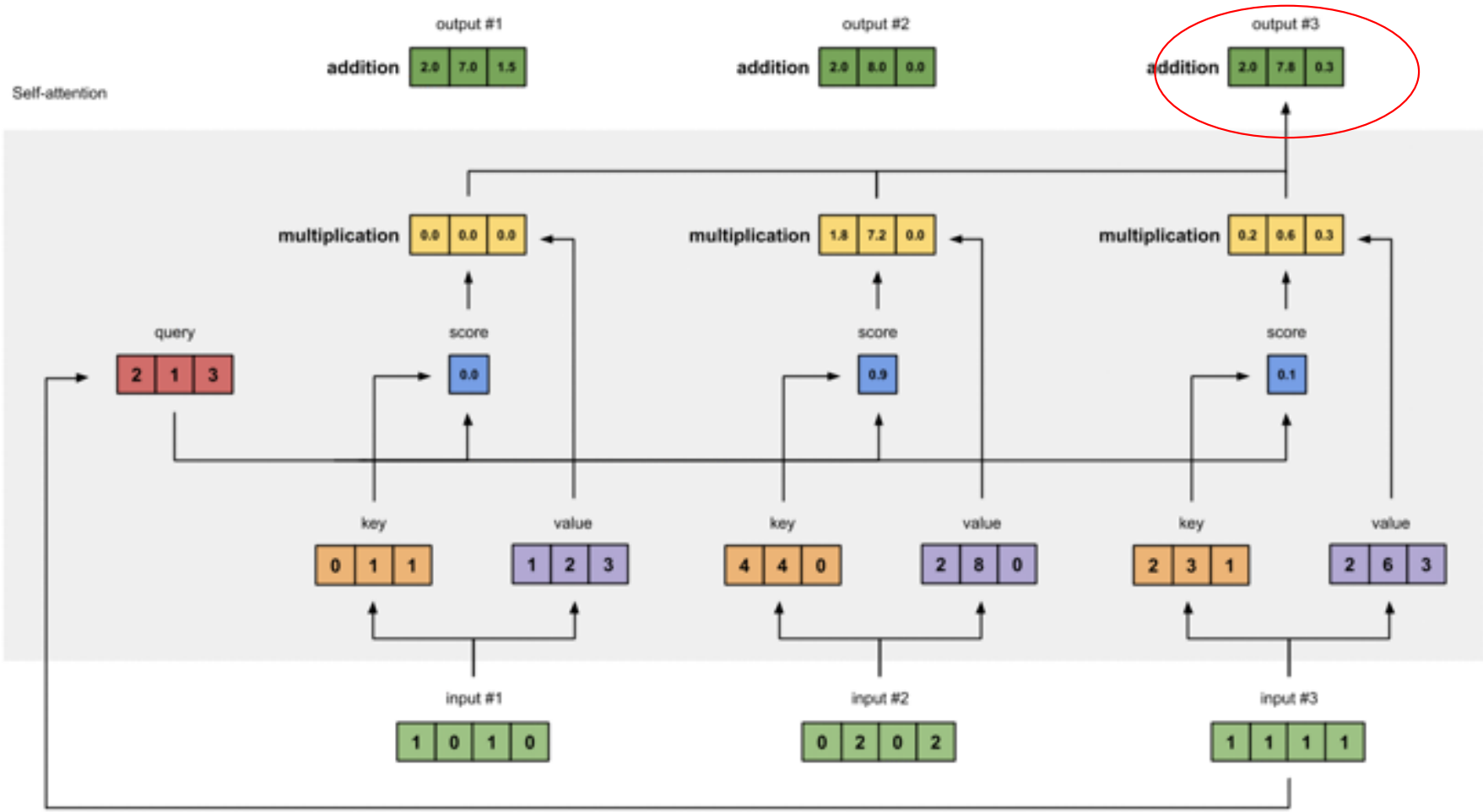
Self-attention

Calculada entre uma camada e outra no encoder (ou no decoder)
Relaciona um elemento a outros elementos na própria sequência

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

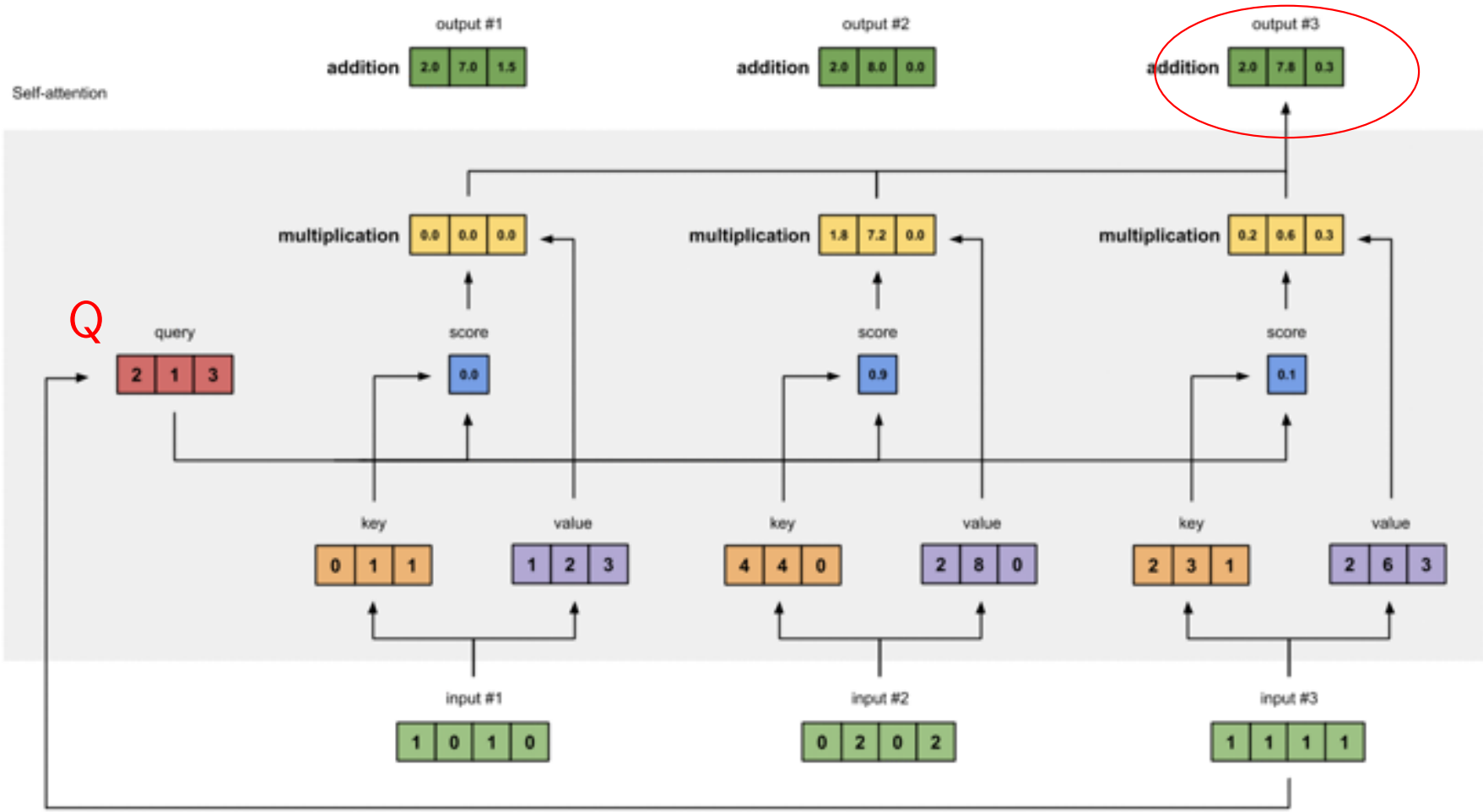
Q: query
K: key
V: value





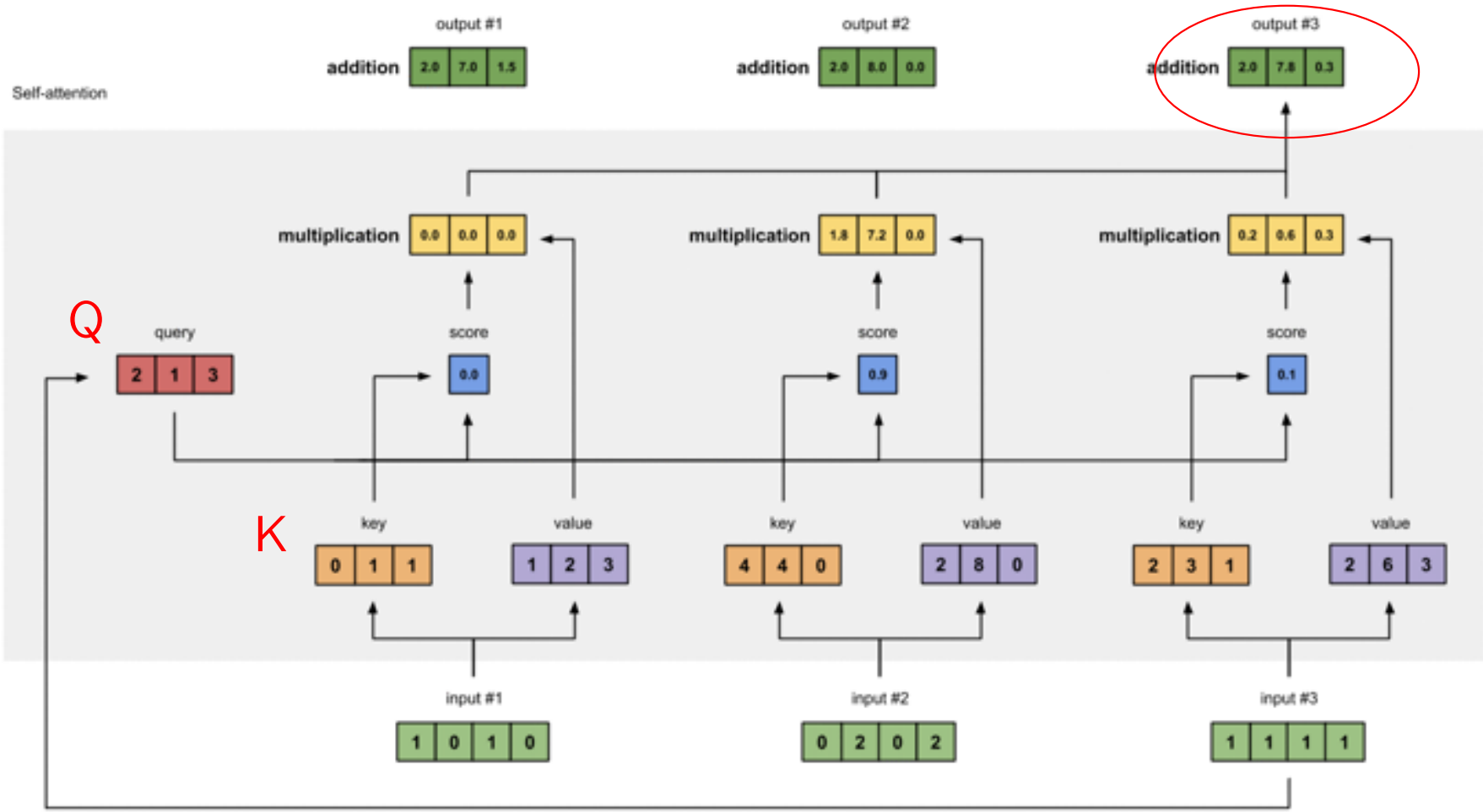
https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

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



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

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

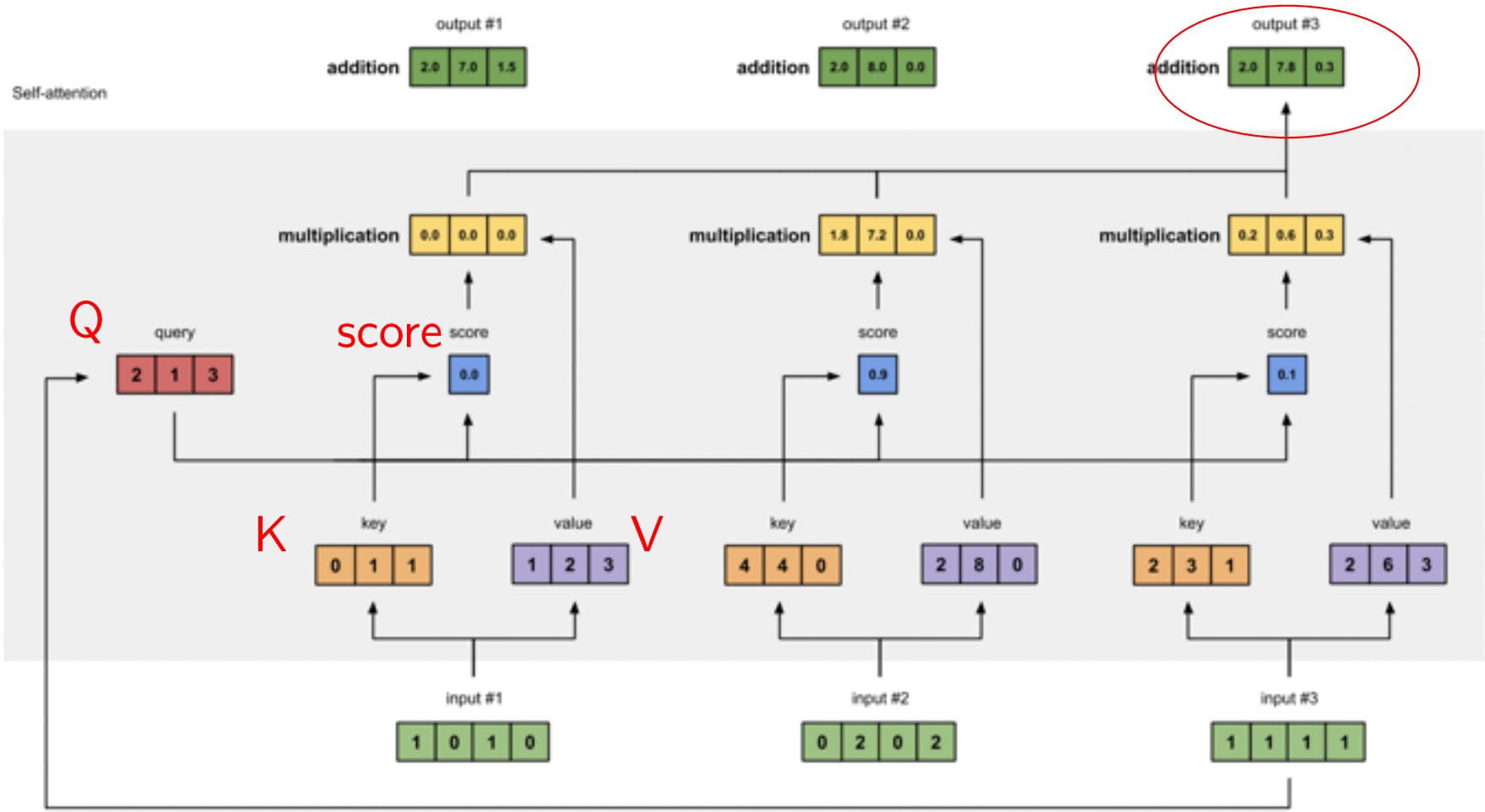


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

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

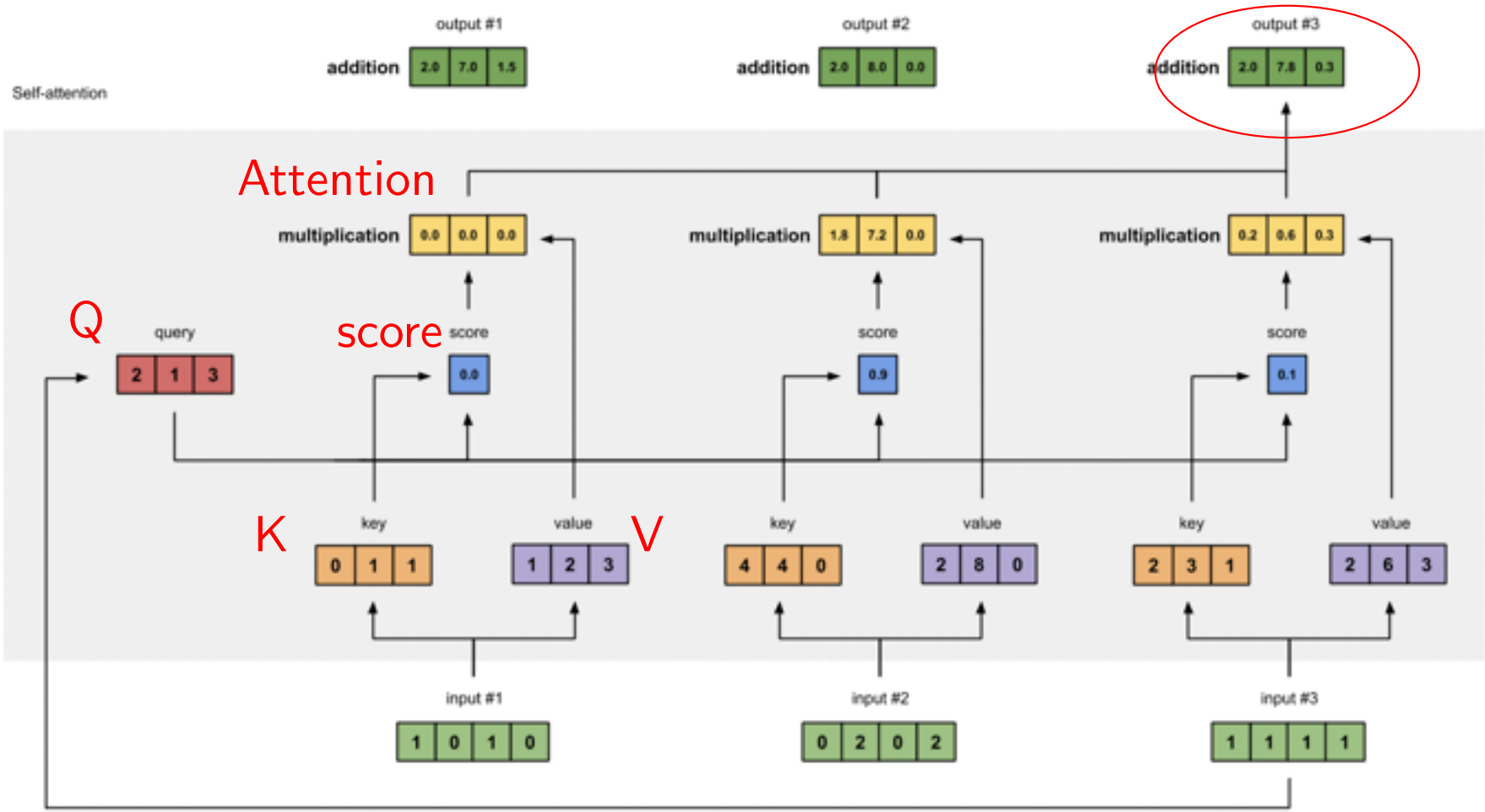
https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

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



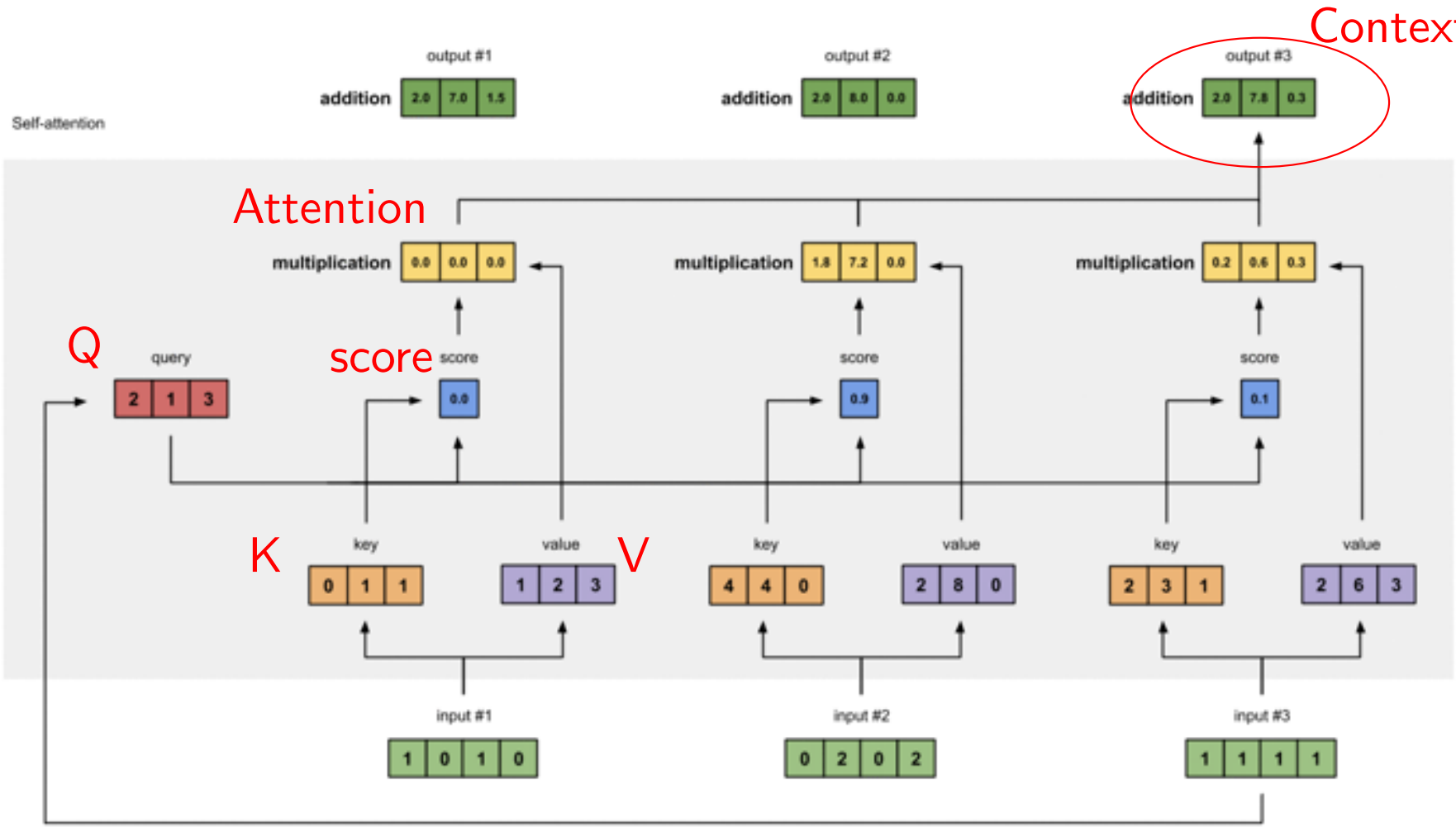
https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

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



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

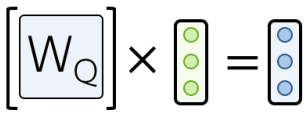
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q\,K^T}{\sqrt{d_k}}\right)V$$



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

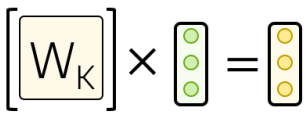
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q\,K^T}{\sqrt{d_k}}\right)V$$

Each vector receives three representations ("roles")



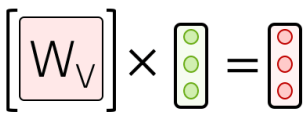
Query: vector **from** which the attention is looking

"Hey there, do you have this information?"



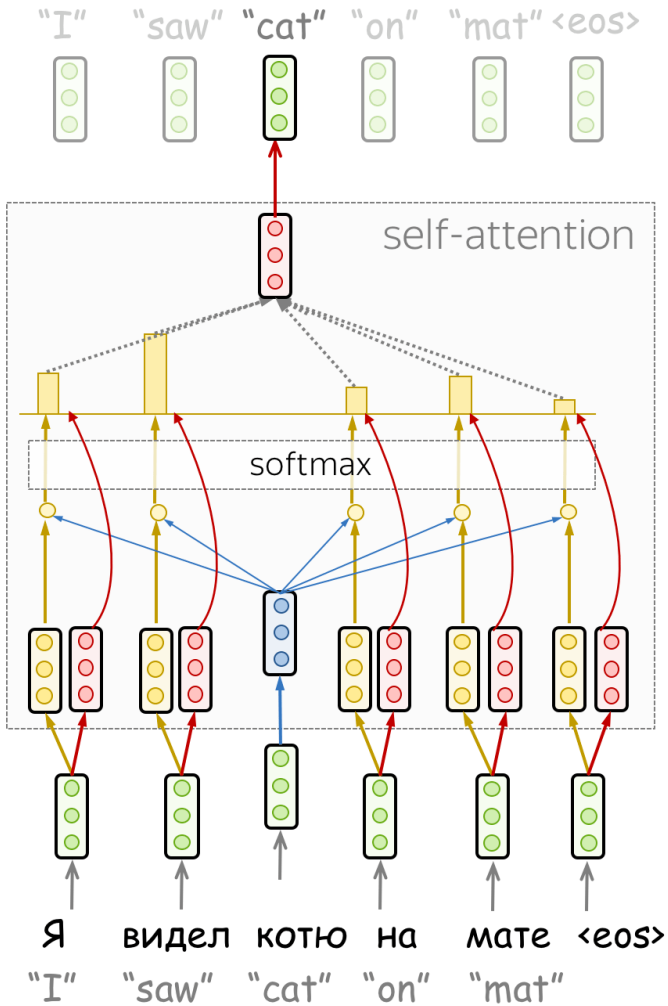
Key: vector **at** which the query looks to compute weights

"Hi, I have this information – give me a large weight!"



Value: their weighted sum is attention output

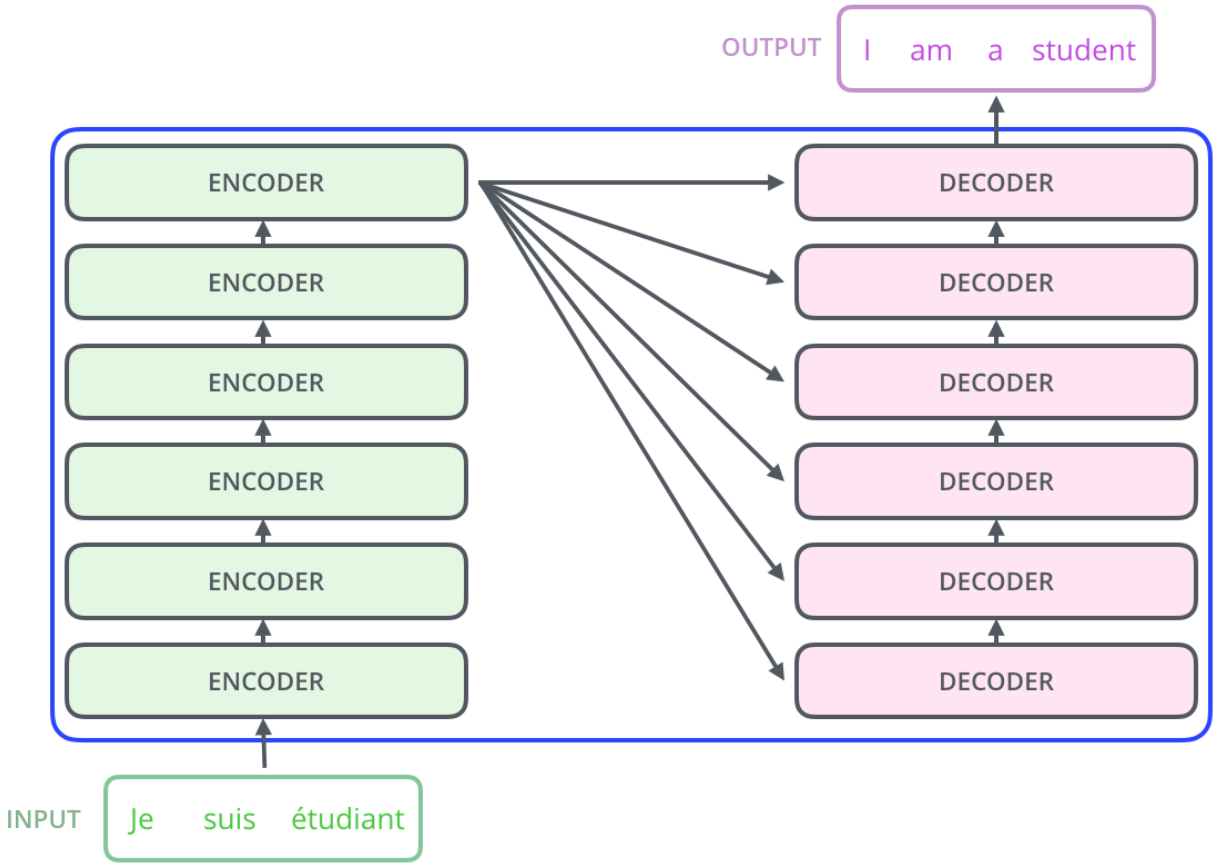
"Here's the information I have!"



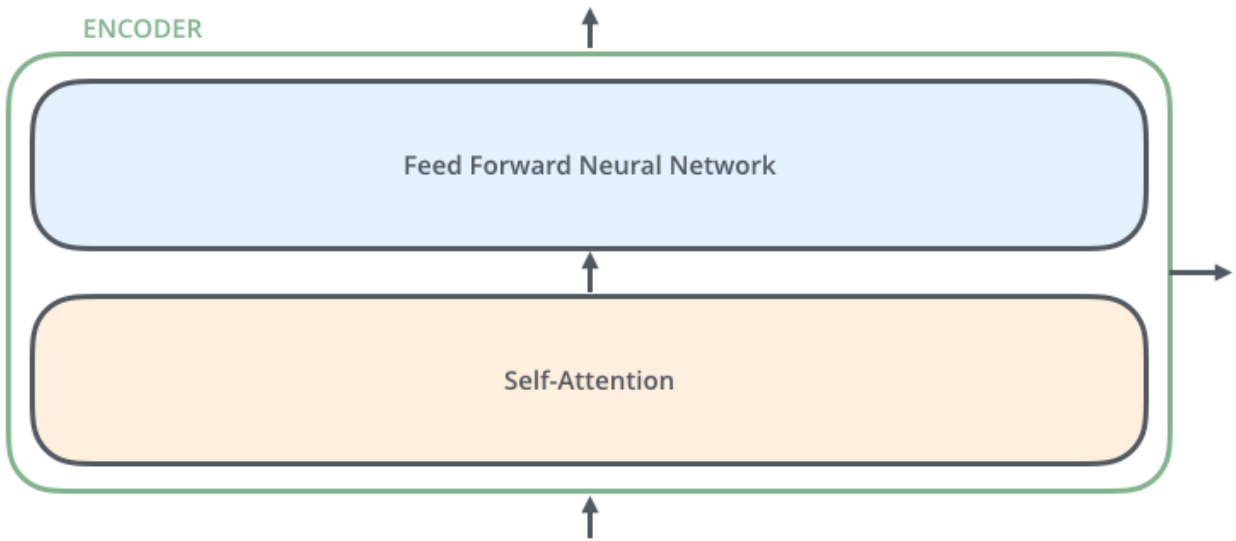
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q\,K^T}{\sqrt{d_k}}\right)V$$

Vários detalhes em

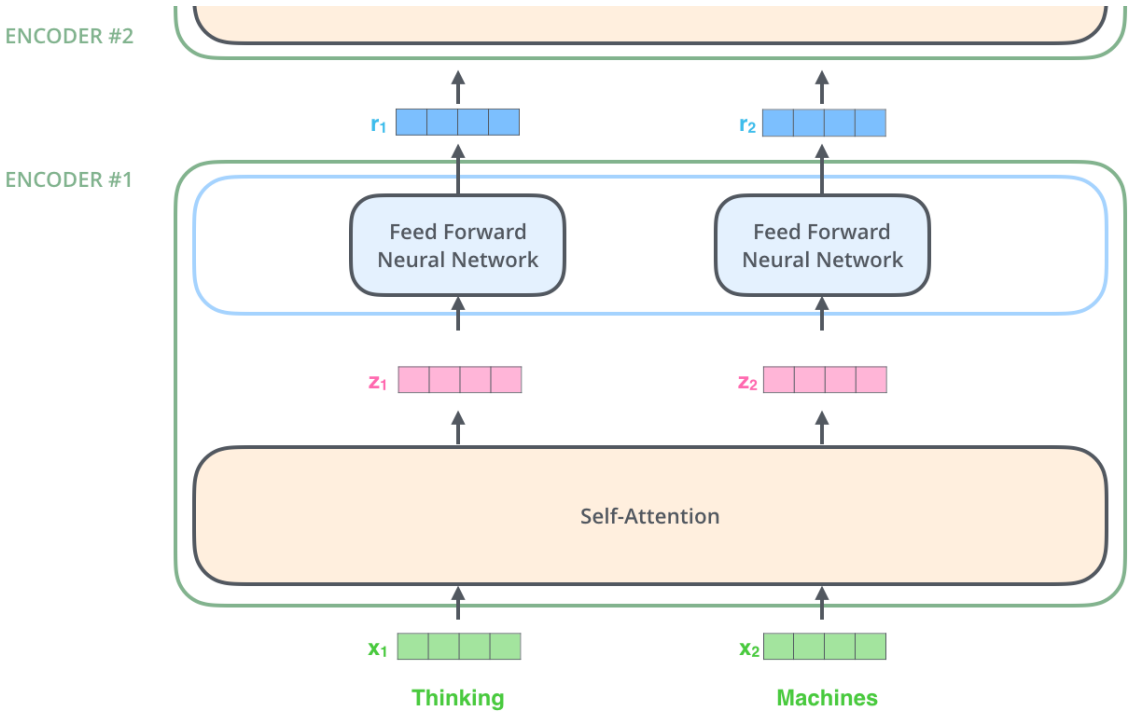
https://jalammar.github.io/illustrated-transformer/



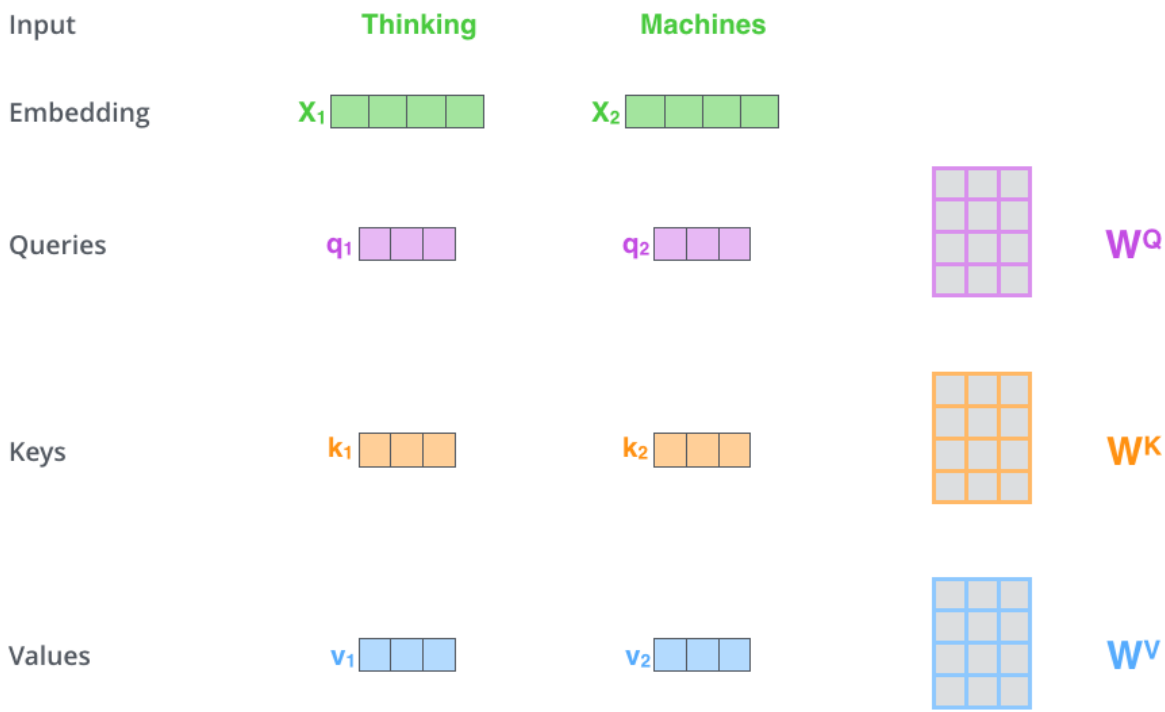
Visão geral de uma camada do encoder



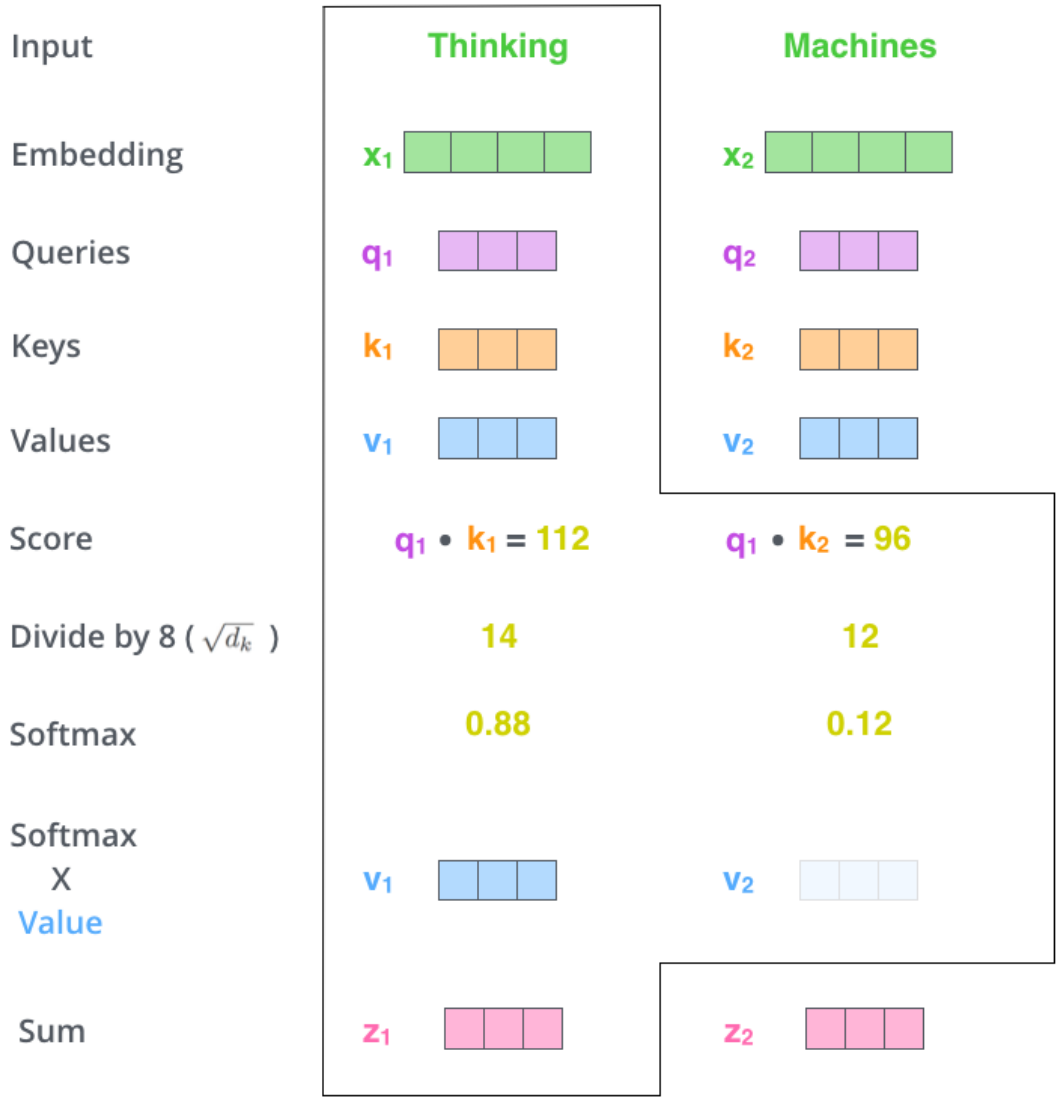
Visão mais detalhada de uma camada do encoder



Geração dos elementos (Q, K, V) usados no cálculo do attention

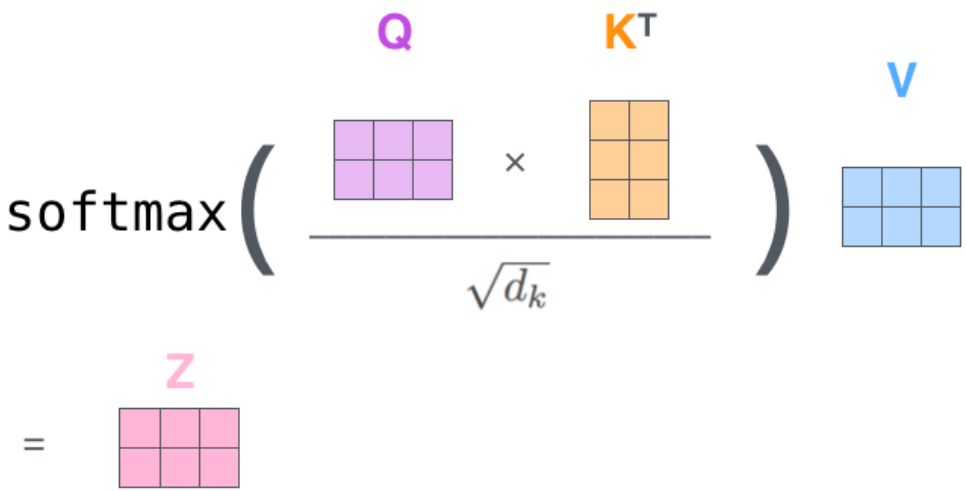


Context – saída da camada

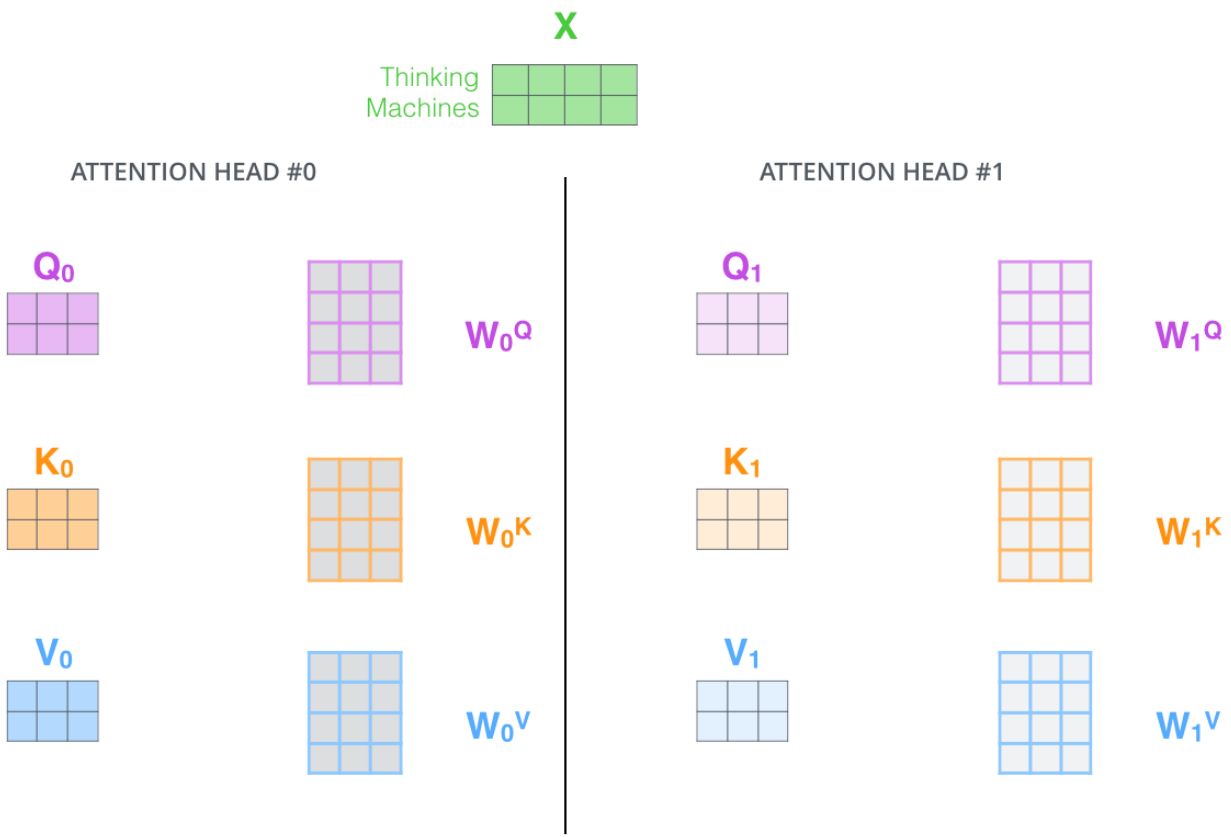


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q\,K^T}{\sqrt{d_k}}\right)V$$

Attention é calculada para todos os elementos da sequência em paralelo

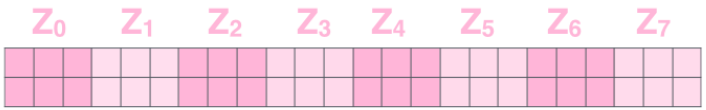


Podemos ter **Multi-head attention**
(análogo a múltiplos filtros na CNN)



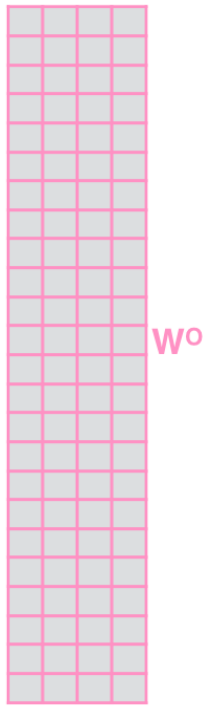
Multi-head attention é calculada em paralelo

1) Concatenate all the attention heads



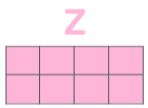
2) Multiply with a weight matrix W^O that was trained jointly with the model

X

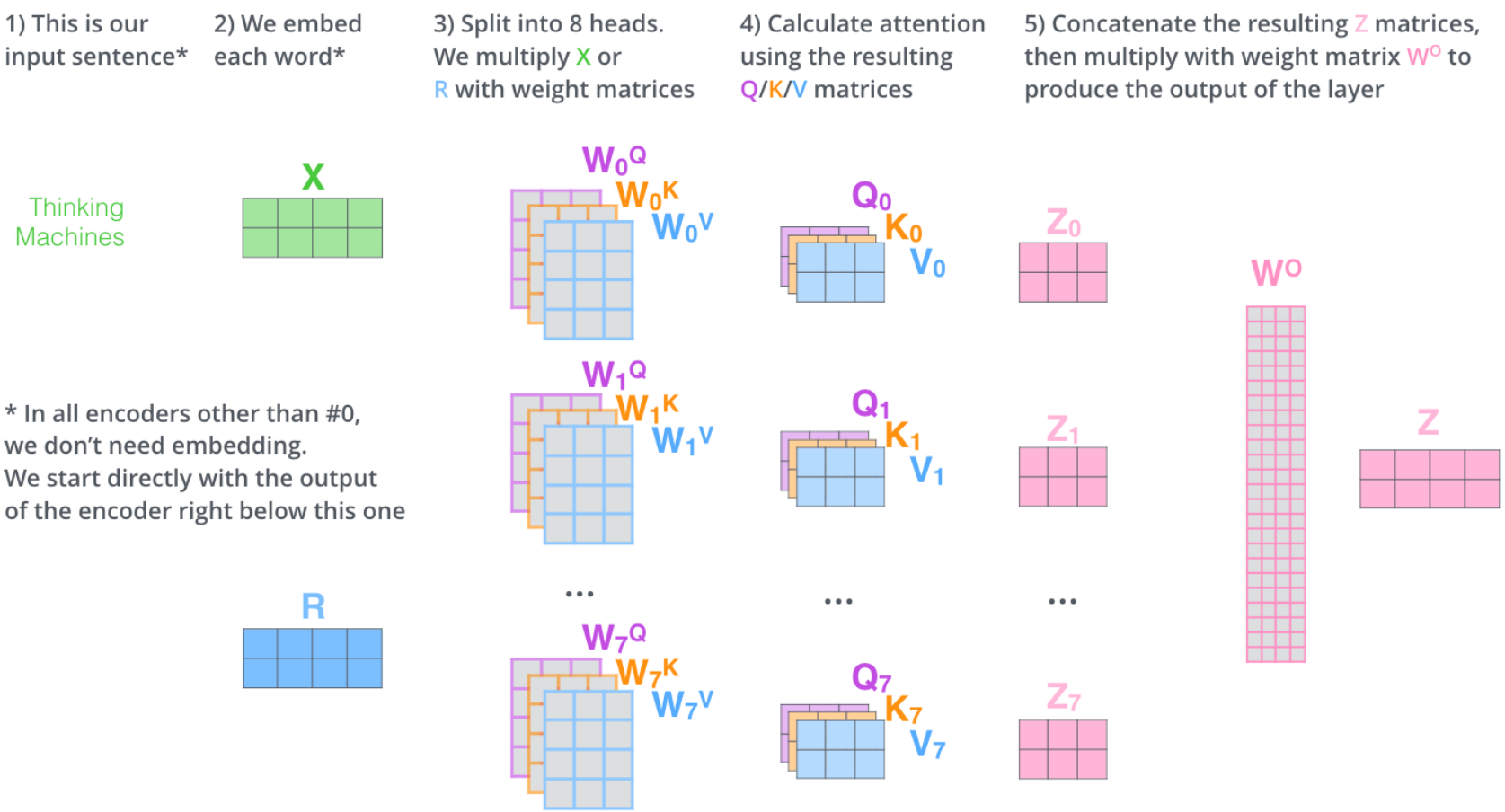


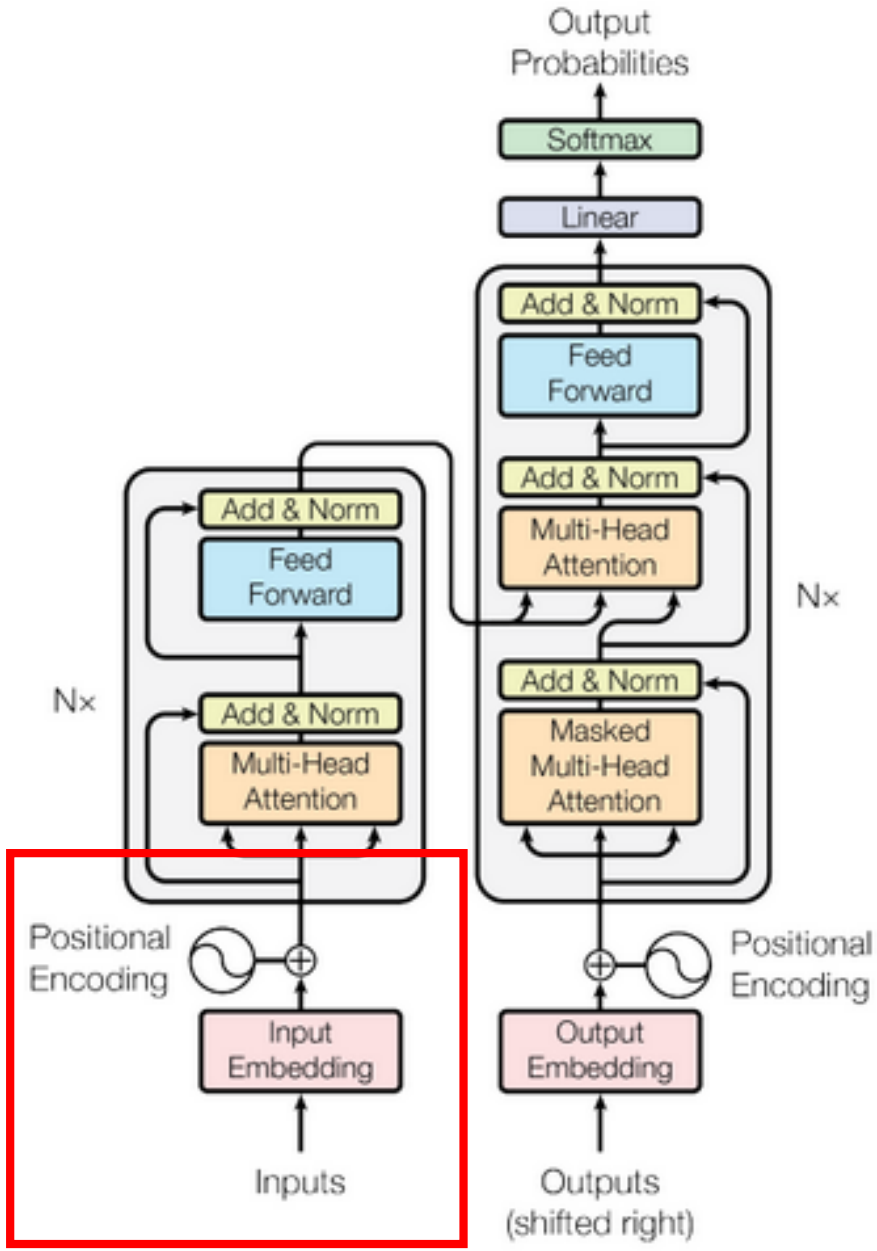
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

=



Visão geral com alguns detalhes




Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
N×
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
N×
Positional Encoding
Input Embedding
Inputs
Positional Encoding
Output Embedding
Outputs (shifted right)

Positional encoding

Tokens and embedding