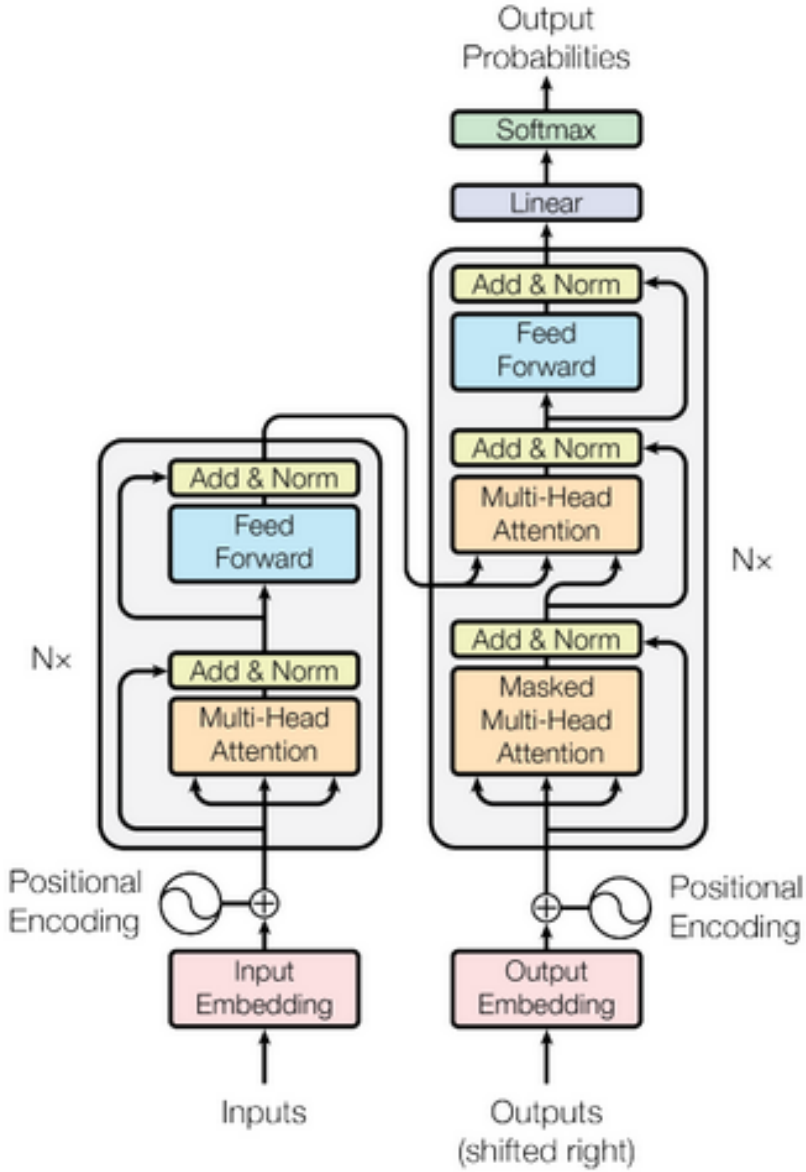
MAC5921 – Deep Learning

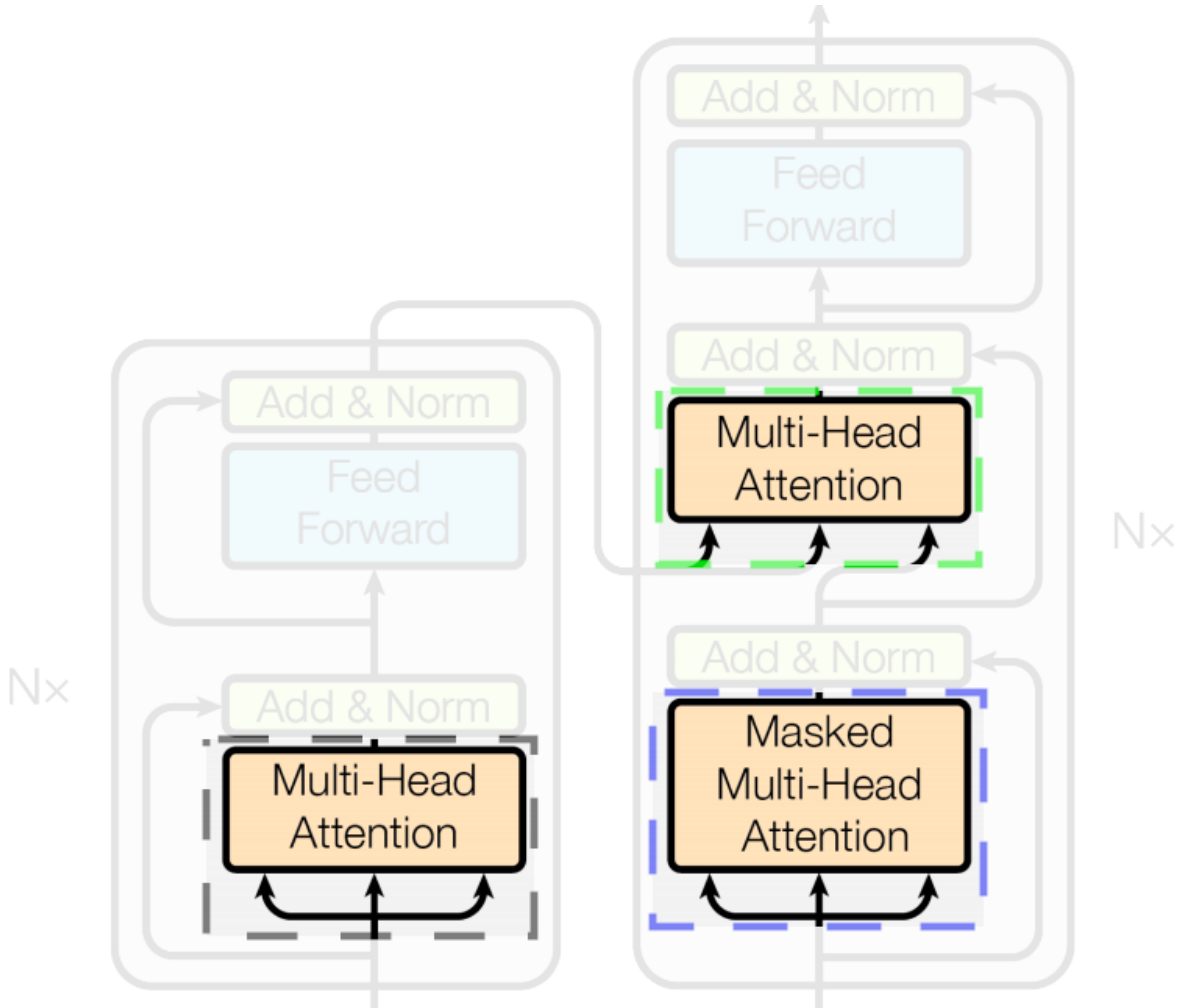
Aula 13 – 05/10/2023

Transformers – parte 2

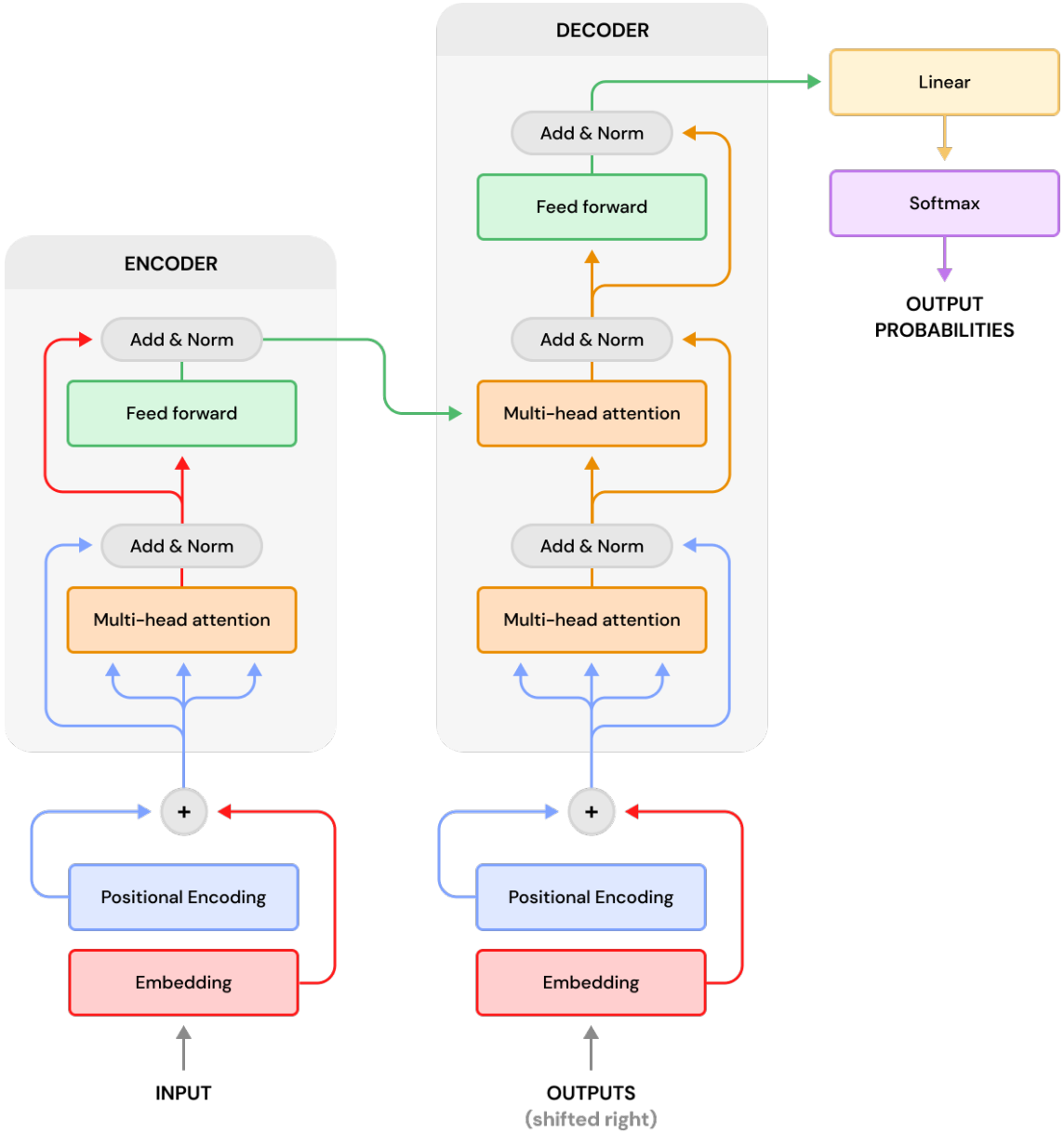
Nina S. T. Hirata

Módulos Encoder e Decoder no transformer



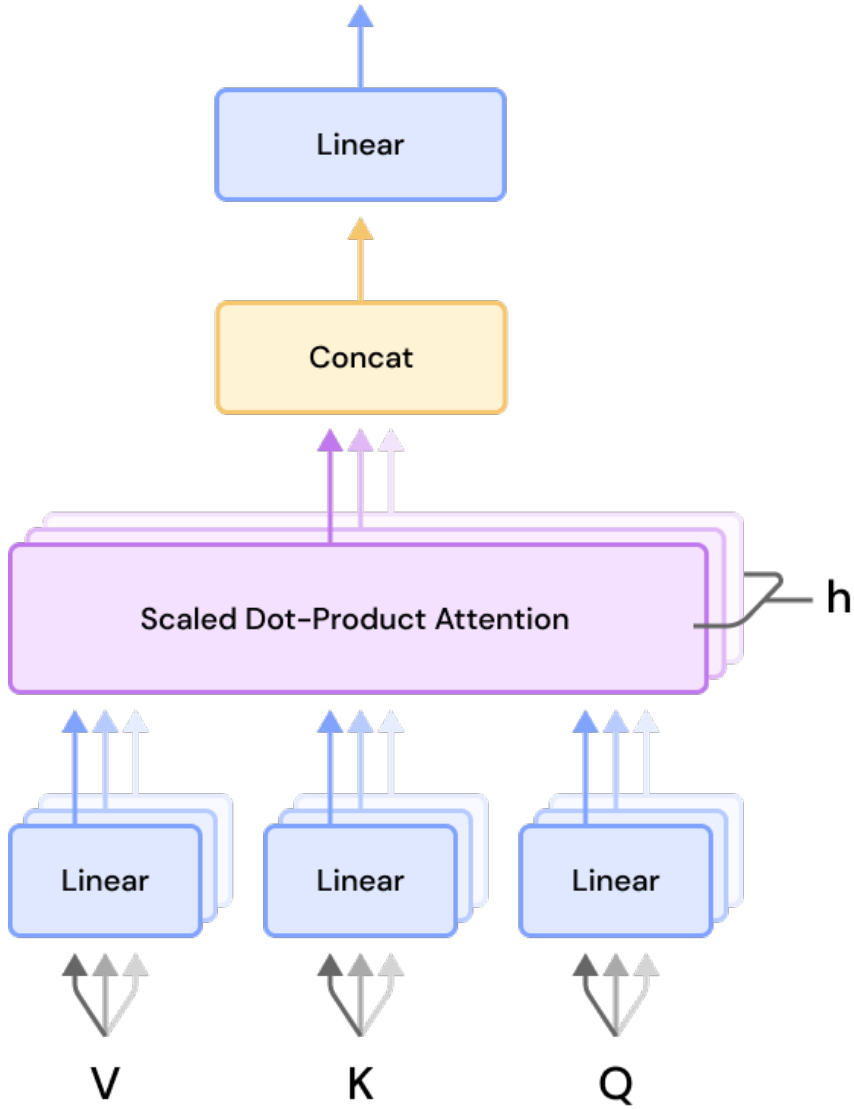


https://tungmphung.com/the-transformer-neural-network-architecture/

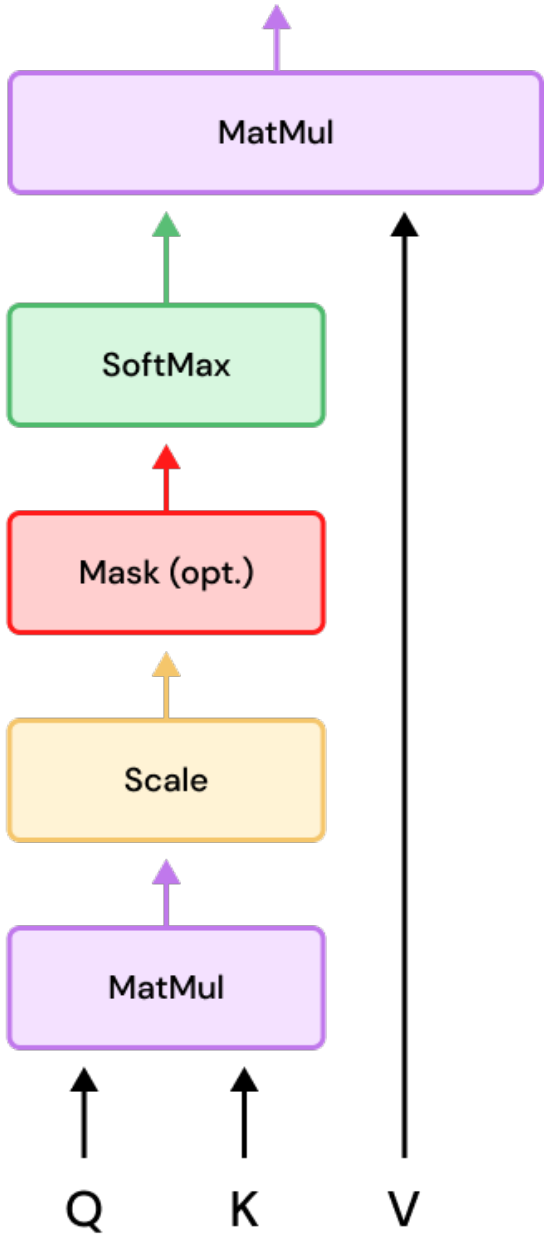


https://www.revistek.com/posts/transformer-architecture

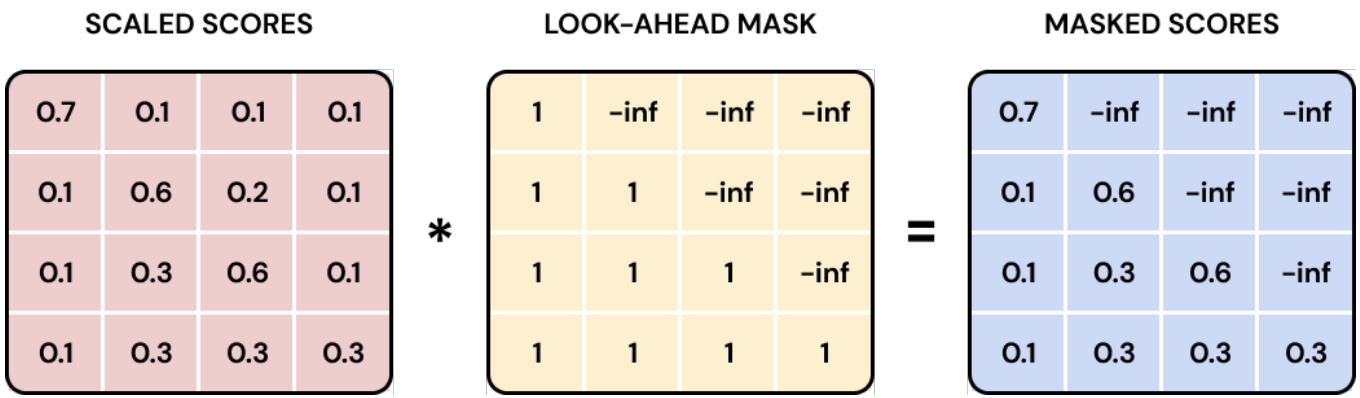
Multi-head attention block



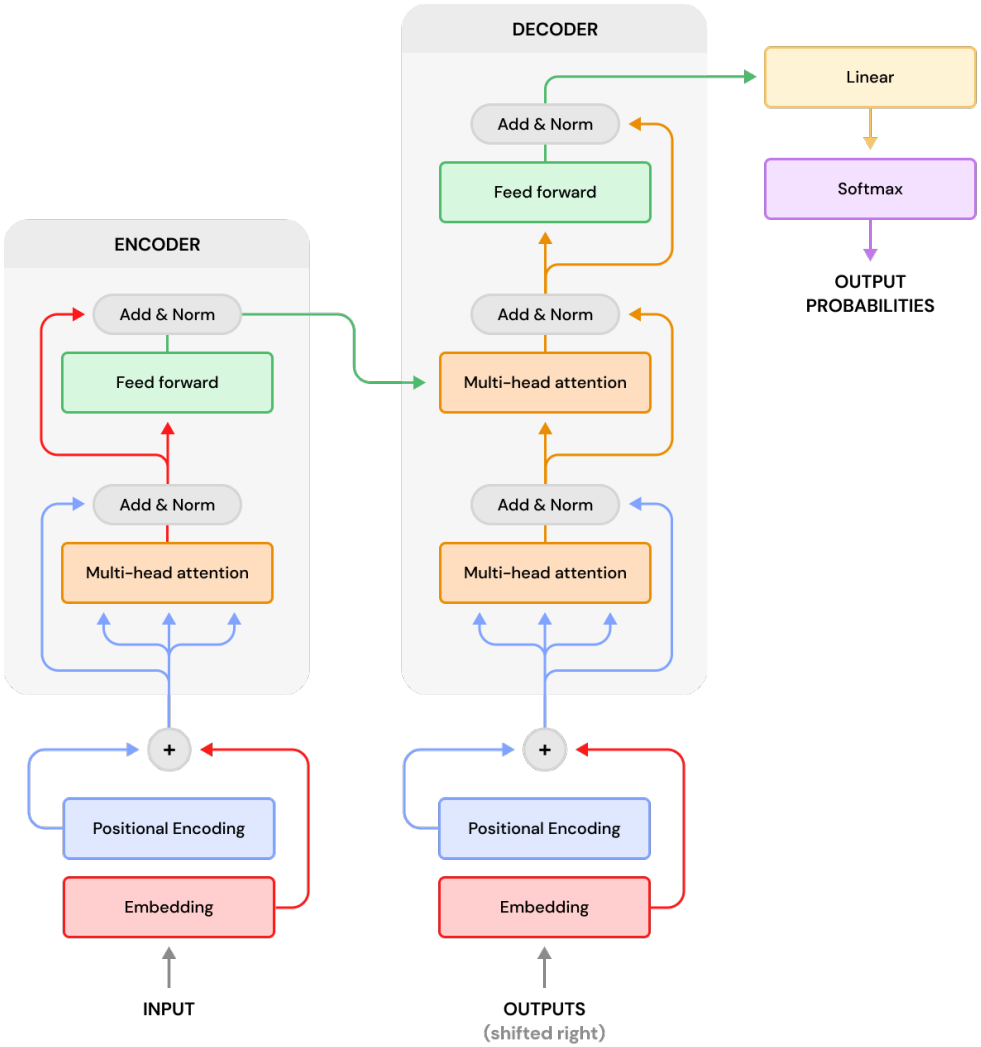
Scale dot-product attention



Masked multi-head attention (o segundo do bloco no decoder)



Decoder multi-head attention (o segundo do bloco no decoder)



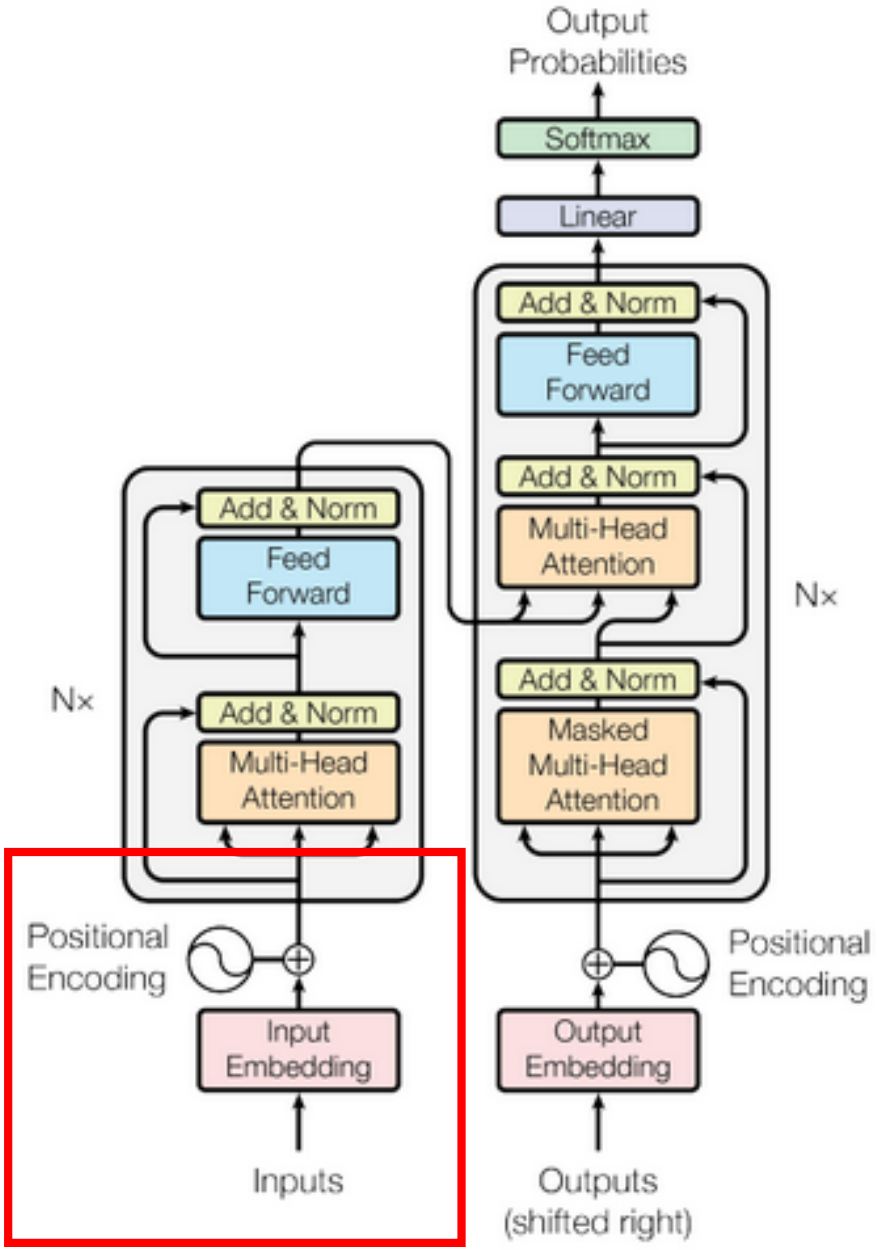
K e V são computados a partir da saída do encoder

Q é computado a partir da saída do módulo de Masked multi-head attention

Código em que é possível ver a estrutura discutida nos slides anteriores

https://www.datacamp.com/tutorial/building-a-transformer-with-py-torch

Positional encoding no transformer



Por que positional encoding?

- Self-attention não tem percepção da posição dos elementos na sequência
- Transformer não tem noção da ordem dos elementos
  Porém, a ordem dos elementos é relevante (por exemplo, a ordem das palavras é importante para o significado de uma frase)
- *Positional encoding* busca suprir essa deficiência em relação à ordem dos elementos

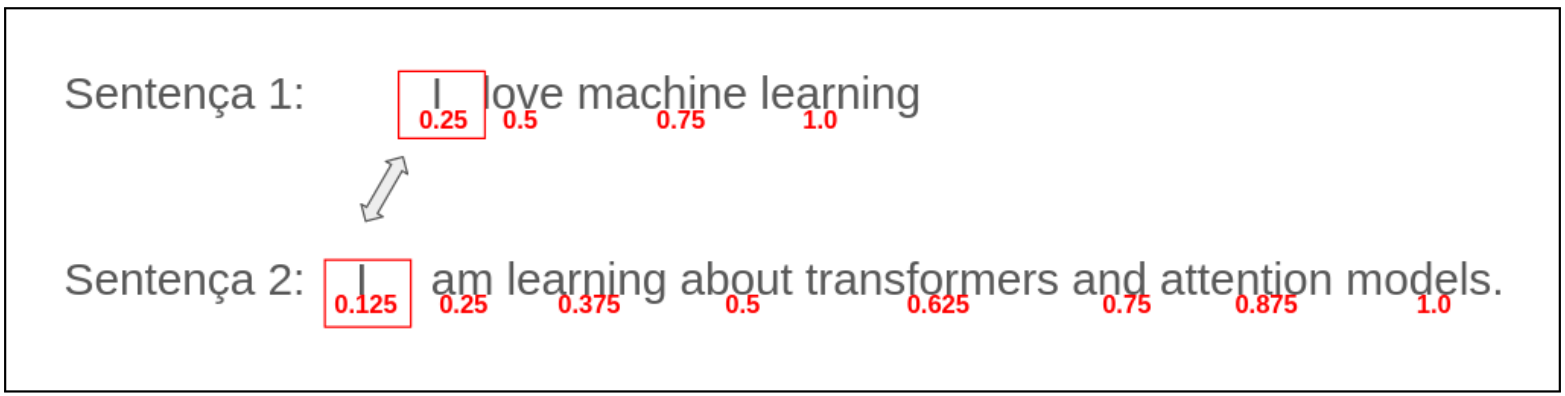
Primeira tentativa: acrescentar, na codificação da palavra, um número inteiro correspondente à posição ocupada pela palavra na sequência

Sentença 1: I (1) love (2) machine (3) learning (4)

Sentença 2: I (1) am (2) learning (3) about (4) transformers (5) and (6) attention (7) models. (8)

- Os números podem ficar grandes (feature com valor gigante)
- A sequência de teste pode ter tamanho maior (posição nunca vista antes)
- Treinamento pode não ter visto sequências de um determinado tamanho

Segunda tentativa: acrescentar, na codificação da palavra, um número real equiespaçadamente espalhado em $]0, 1]$



Codificação de posição não é consistente entre sequências de tamanhos distintos

Características desejáveis em positional encoding

https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

- It should output a unique encoding for each time-step (word's position in a sentence)
- Distance between any two time-steps should be consistent across sentences with different lengths.
- Our model should generalize to longer sentences without any efforts. Its values should be bounded.
- It must be deterministic.

Solução proposta no paper do Waswani

- Usa-se vetor de tamanho d para codificar a posição
- Essa codificação é incorporada ao input (não ao modelo)

  Especificamente, a codificação é somada ao *embedding* da palavra

Inspiração

Código binário

Bit menos significativo (LSB)
  0 e 1 se alternam

Próximo LSB
  00 11 00 11 ...

Próximo LSB
  0000 1111 0000 1111 ...

Frequência vai diminuindo

```
 0 :  0 0 0 0
 1 :  0 0 0 1
 2 :  0 0 1 0
 3 :  0 0 1 1
 4 :  0 1 0 0
 5 :  0 1 0 1
 6 :  0 1 1 0
 7 :  0 1 1 1
 8 :  1 0 0 0
 9 :  1 0 0 1
10 :  1 0 1 0
11 :  1 0 1 1
12 :  1 1 0 0
13 :  1 1 0 1
14 :  1 1 1 0
15 :  1 1 1 1
```

Solução: em vez de binário, usar `float`

t posição
d tamanho do vetor de encoding (d par)
i componente do vetor de encoding

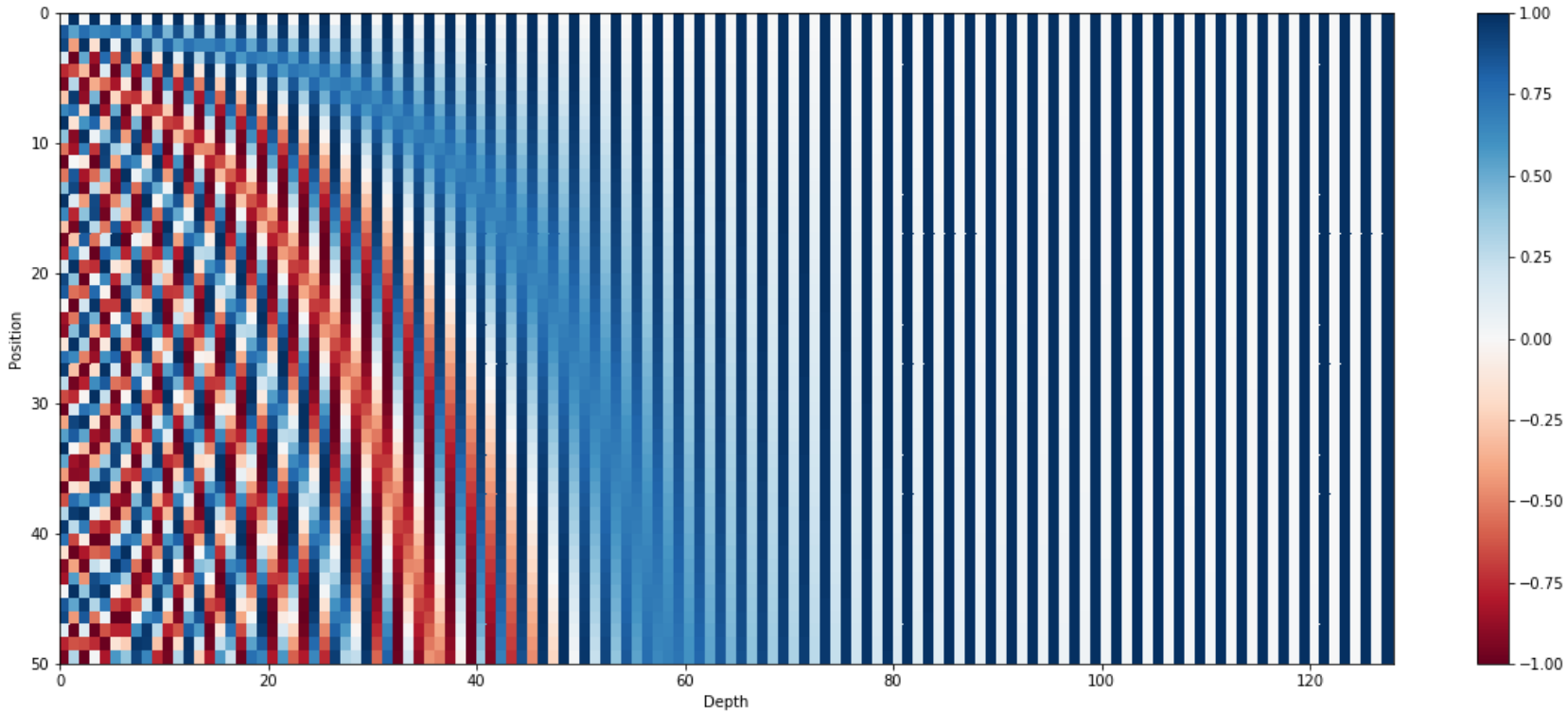
Encoding da posição t, calculado por:

$$\vec{p_t}^{(i)} = \begin{cases} \sin(\omega_k \cdot t), & \text{se } i = 2k \\ \cos(\omega_k \cdot t), & \text{se } i = 2k+1 \end{cases}$$

em que $\omega_k = \frac{1}{10000^{\frac{2k}{d}}}$

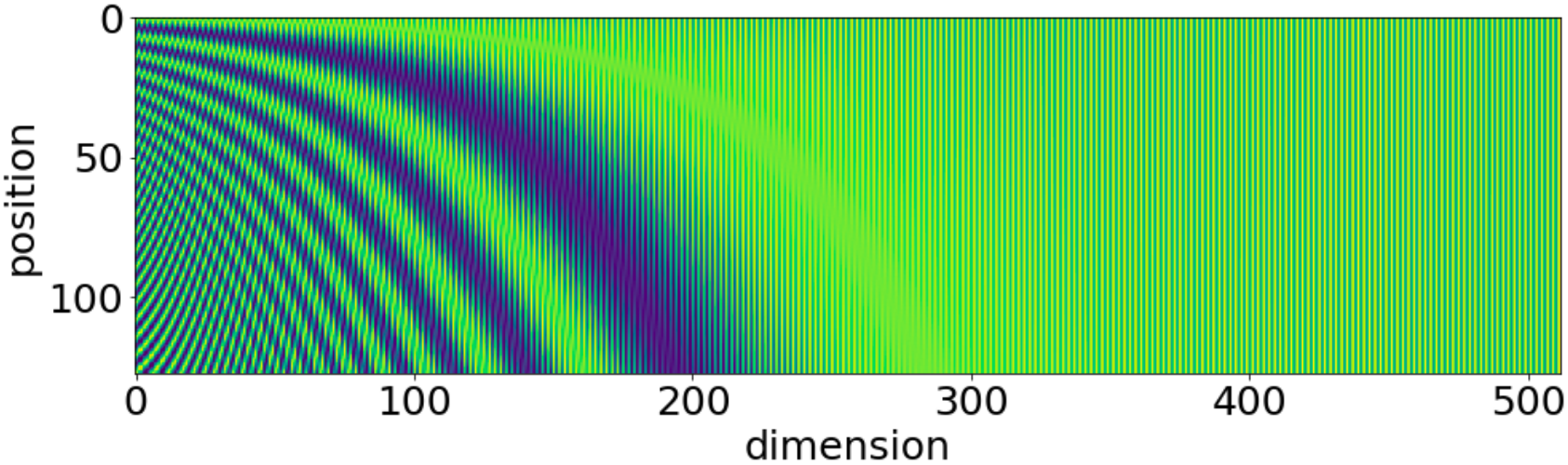
- Frequência decresce ao longo do vetor de enconding

$$\vec{p_t} = \begin{bmatrix} \sin(\omega_1 \cdot t) \\ \cos(\omega_1 \cdot t) \\ \\ \sin(\omega_2 \cdot t) \\ \cos(\omega_2 \cdot t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} \cdot t) \\ \cos(\omega_{d/2} \cdot t) \end{bmatrix}_{d \times 1}$$



- Tamanho do encoding $\mathrm{d} = 128$ (eixo x)
- Tamanho da sequência 50 (eixo y)
- A linha t corresponde ao $\vec{\mathrm{p}_\mathrm{t}}$

Com outra coloração ...



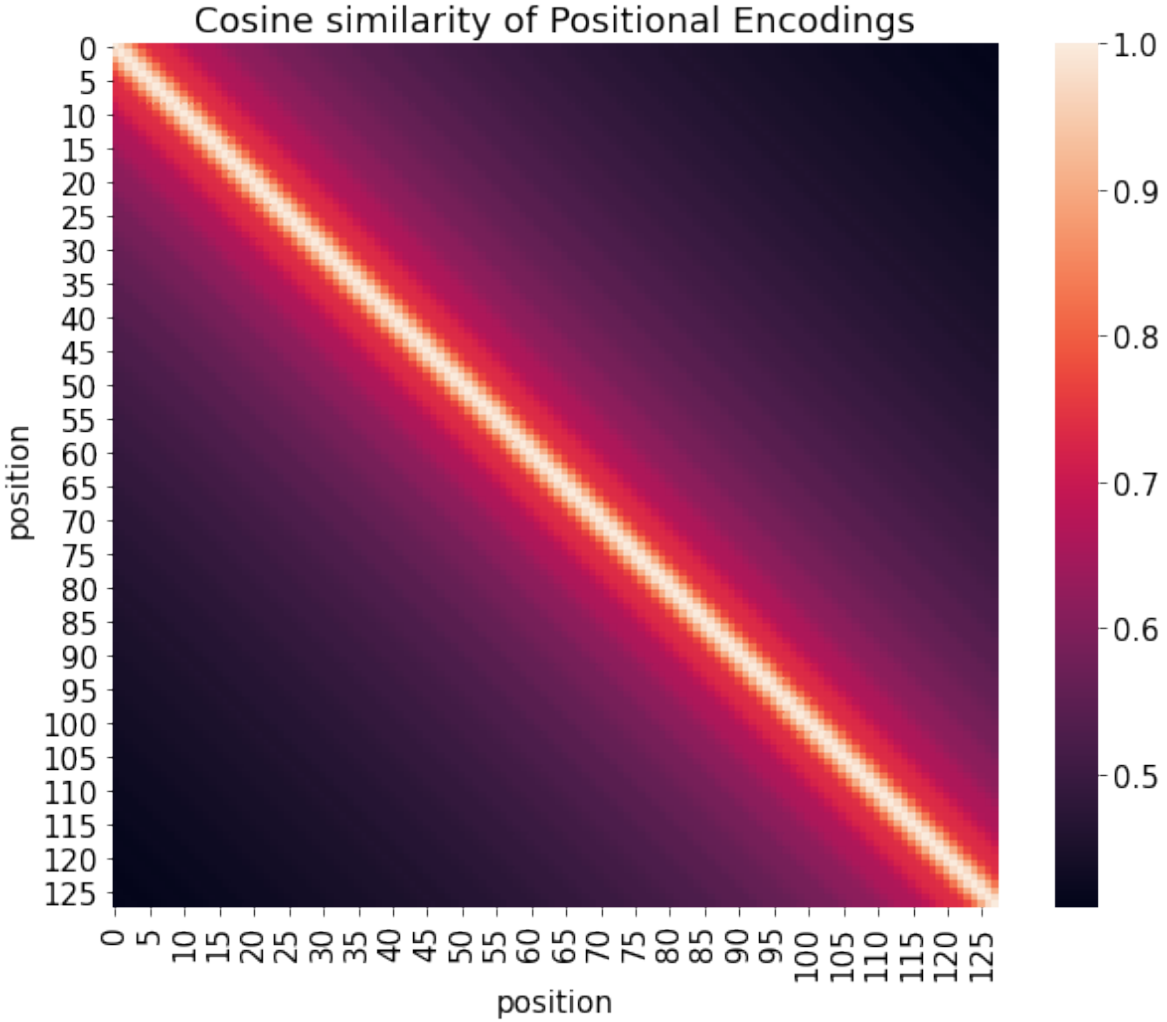
Propriedades

https://tungmphung.com/the-transformer-neural-network-architecture/

- For each pair (i, d), the corresponding encoding is fixed. (This addresses the problem of relative-distance.)

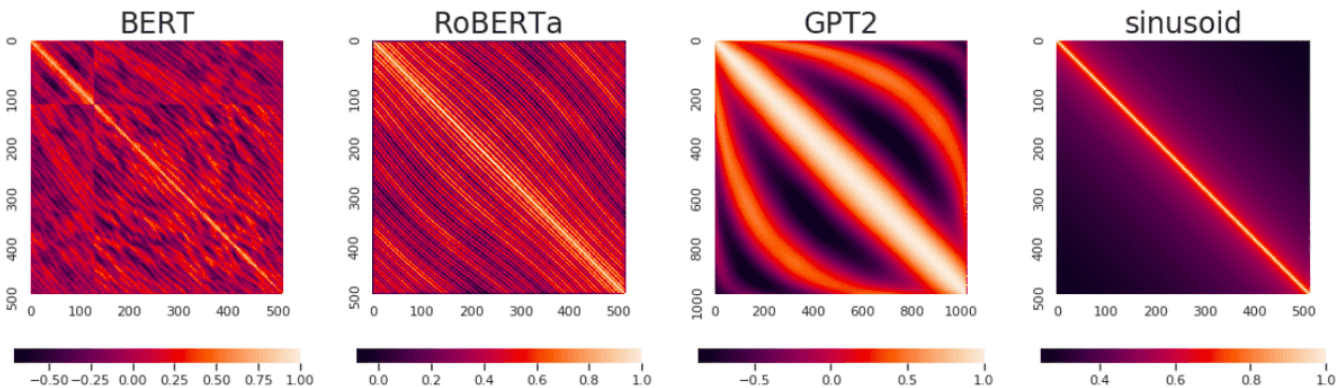
- The encoding values are always in the range $[-1, 1]$ since they are the results of sin and cos functions. (This solves the extrapolation problem.)

- The encoding values for the earlier dimensions (say dimension indexes < 300) have strong positional effects. In other words, they are different for different positions, and the difference is bigger when the positions are further away from each other. For latter dimensions, the encoding values are very similar for all positions. Thus, this region acts as a free-positional space, allowing a part of the embedding to pass to subsequent layers without being affected by positional information. With that to say, the successive layers have options to use the semantic of input sequences with or without positional effect.


Cosine similarity of Positional Encodings
position
position
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
1.0
0.9
0.8
0.7
0.6
0.5

Positional encoding $\times$ positional embedding

https://theaisummer.com/positional-embeddings/



embedding: learnable