

MÓDULO 3

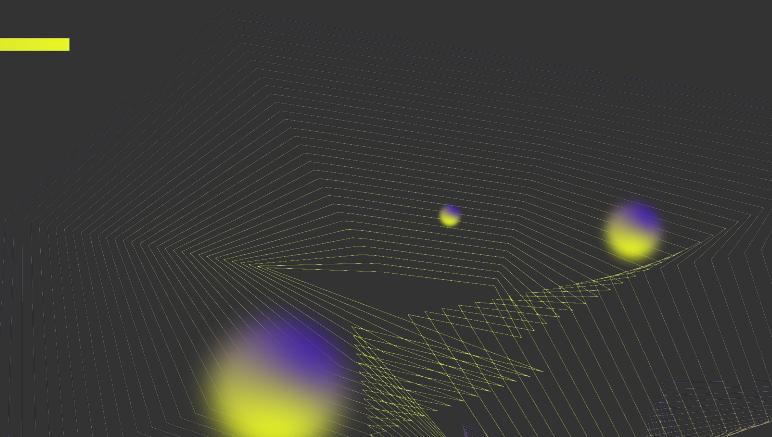
O Modelo Transformer

Prof. Filipe Wall Mutz

ESPECIALIZAÇÃO
INTELIGÊNCIA ARTIFICIAL
& CIÉNCIA DE DADOS



Superintendência de
Educação a Distância



Histórico

Boom do *Deep Learning* a partir de 2012

- MLPs, CNNs, RNNs (ex.: LSTMs)

Em 2017, Vaswani et al. introduzem a arquitetura

Transformer no artigo “Attention Is All You Need”

- Dominante em PLN, atualmente
- Equivalente ou melhor em outras tarefas (e.g., Visão)
- Cerne de Sistemas de IA Generativa

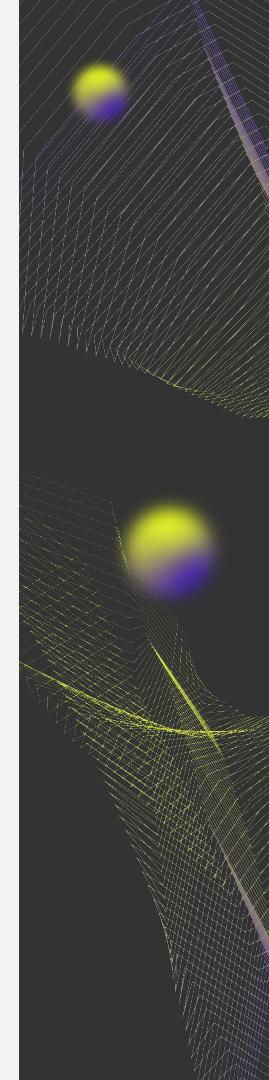
Motivação

Tornar modelos Seq2Seq mais eficientes

- RNNs processam dados sequencialmente
- Isto torna operações pouco paralelizáveis

Aumentar a performance preditiva

- RNNs codificam a entrada em um vetor.
- **Mecanismos de atenção** permitem que o modelo **foque seletivamente** em partes da entrada.

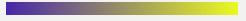


Visão Geral

Principais Contribuições

- Auto-atenção (*self-attention*)
- Arquitetura totalmente baseada em mecanismos de atenção
- Demonstrar que esta arquitetura é competitiva

Foco em tarefas sequência-para-sequência não pareadas (e.g., **tradução automática**).



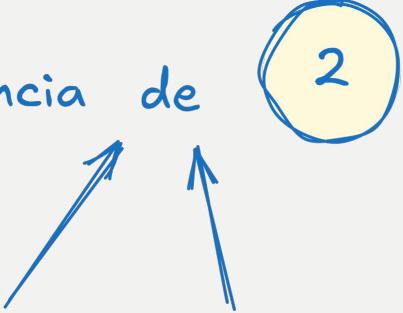
Exemplo

ciência
data science is awesome

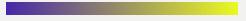
A diagram illustrating a sequence of words: 'ciência', 'data', 'science', 'is', and 'awesome'. An arrow points from the word 'ciência' to a yellow circle containing the number '1'.

Exemplo

ciência de
data science is awesome



The diagram illustrates a sequence of words: "ciência de", "data", "science", "is", and "awesome". Two blue arrows originate from the words "data" and "science", pointing upwards and to the right towards a yellow circle containing the number "2".

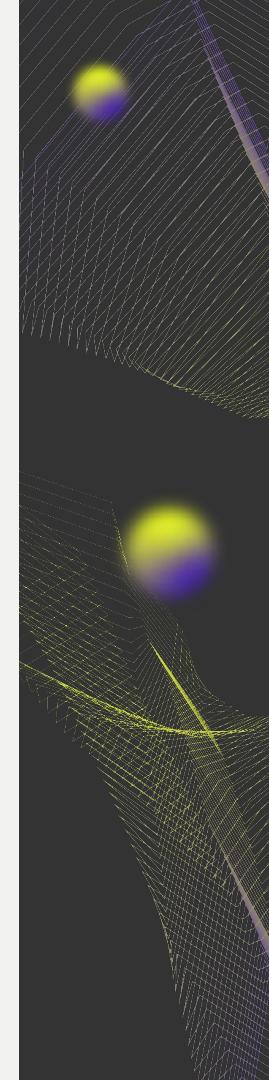


Exemplo

ciência de dados



data science is awesome



Exemplo

ciência de dados é

4



data science is awesome

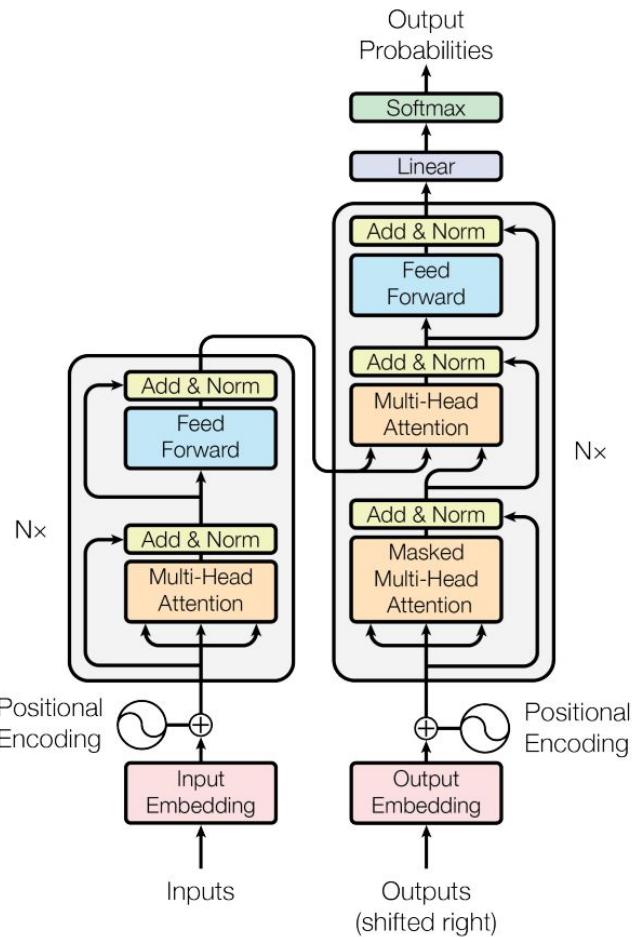
Exemplo

ciência de dados é massa
data science is awesome

5

Exemplo

		saída			
		ciência de dados é massa			
		data			
entrada	data				
	science	█	█		
	is			█	
	awesome				█

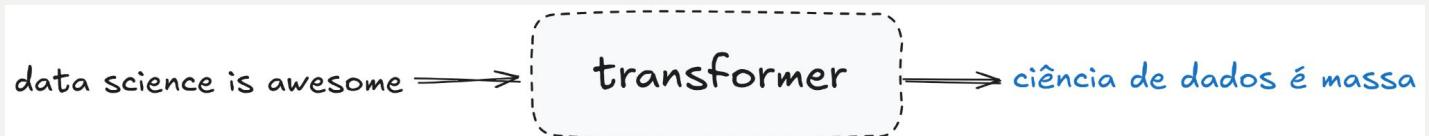


A arquitetura é um pouco complexa e teremos que fazer um passeio, com calma, por todos os componentes

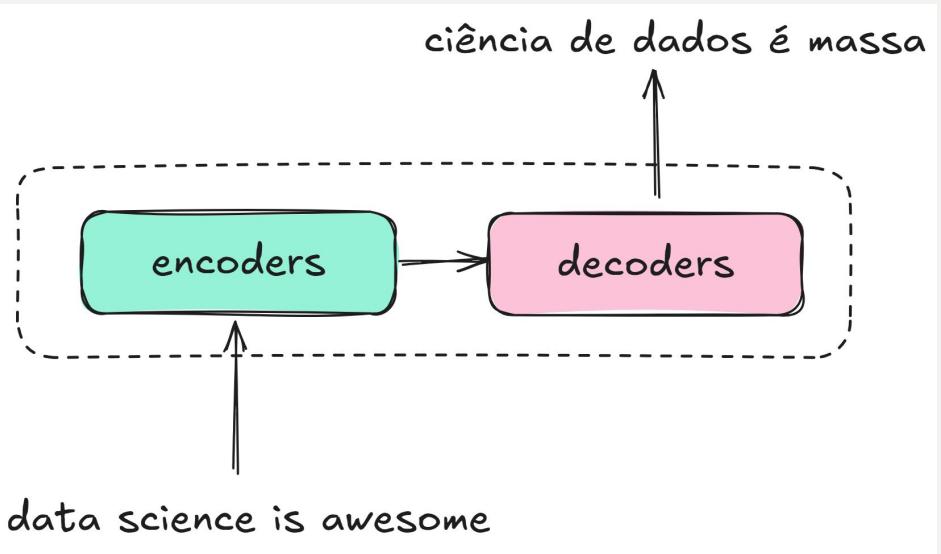
Figure 1: The Transformer - model architecture.

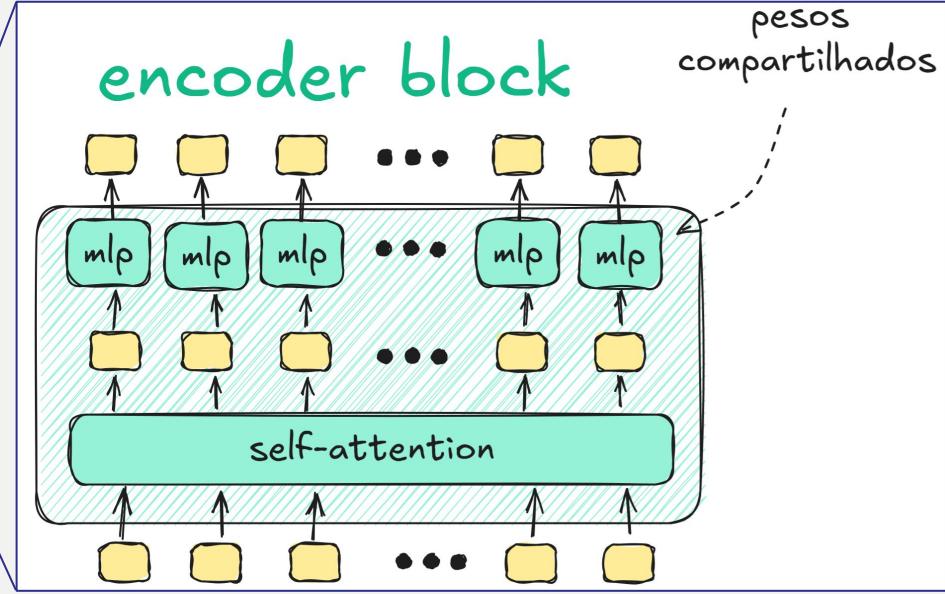
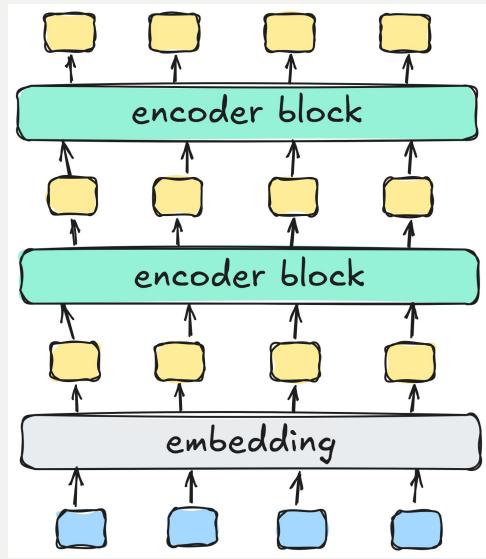


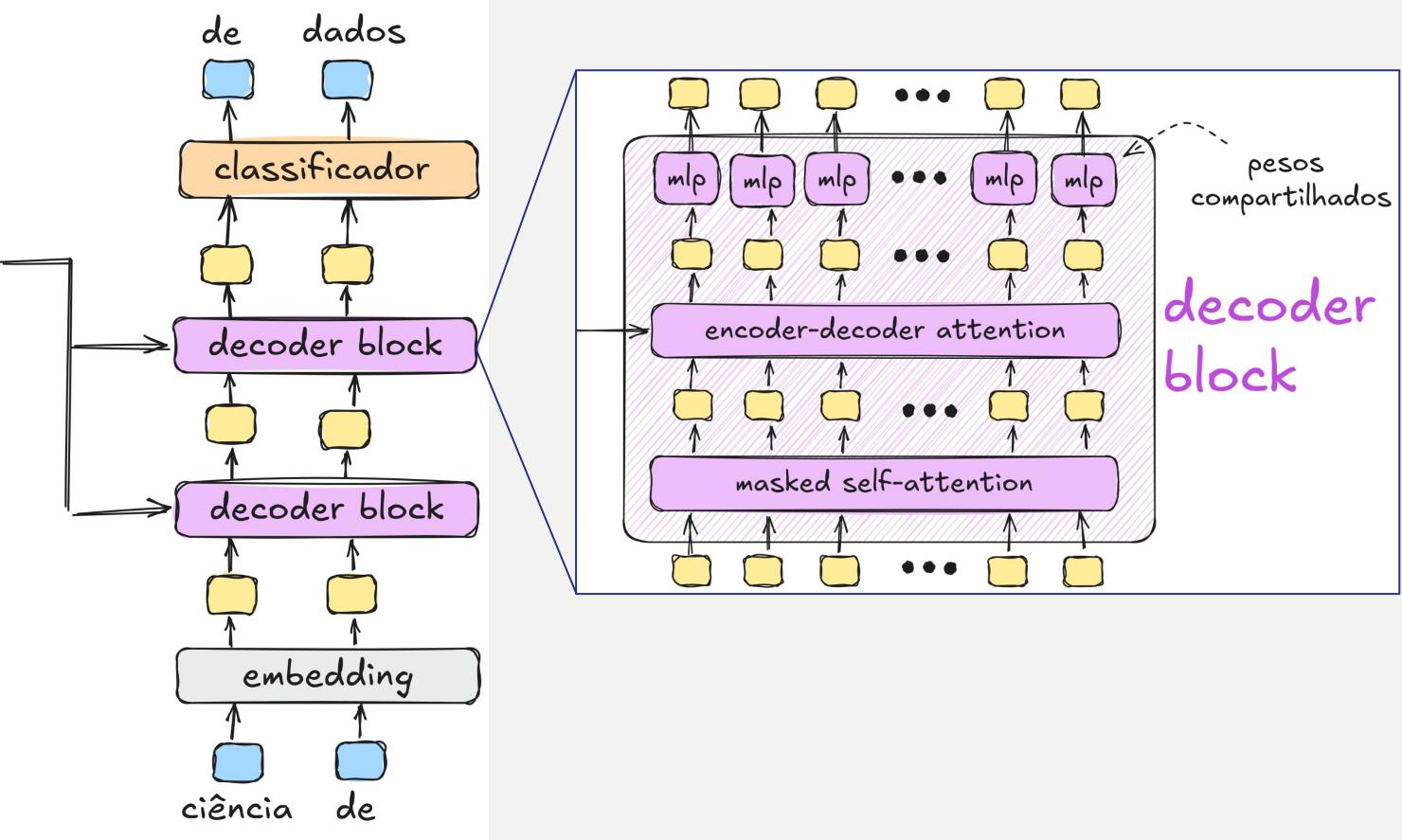
Arquitetura



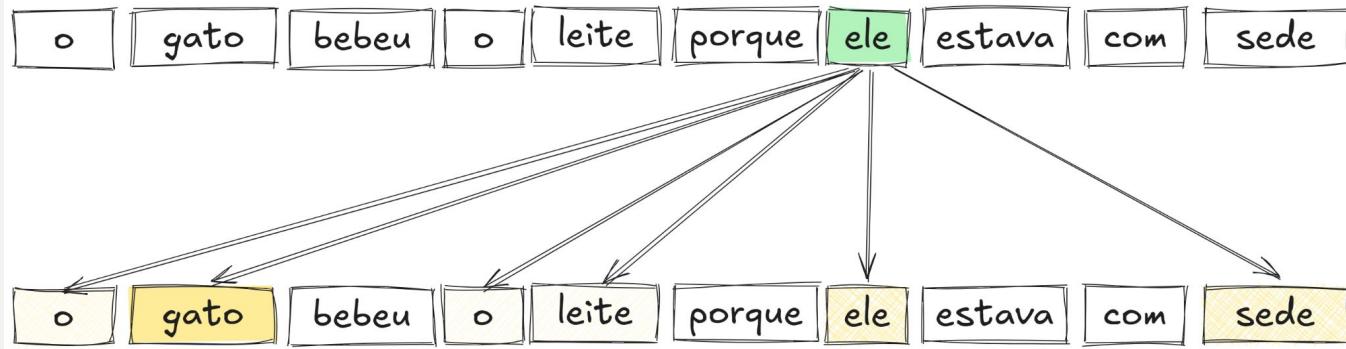
Arquitetura



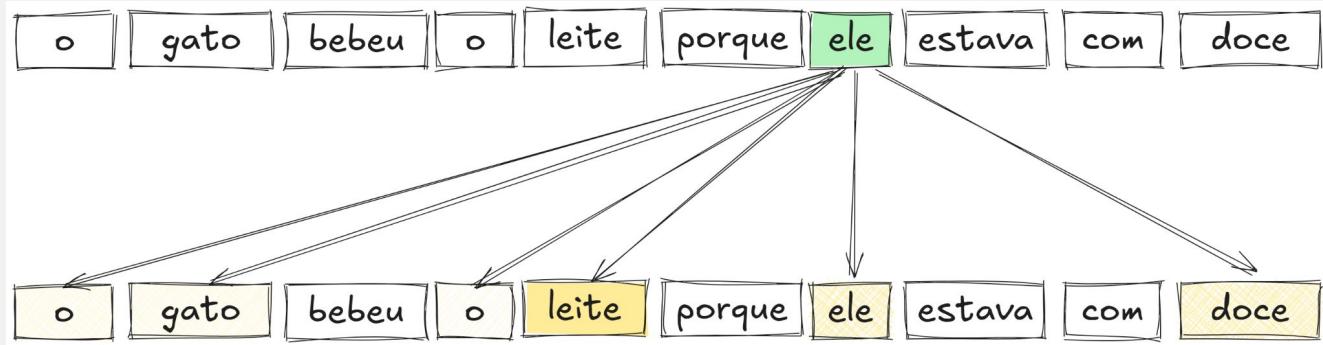




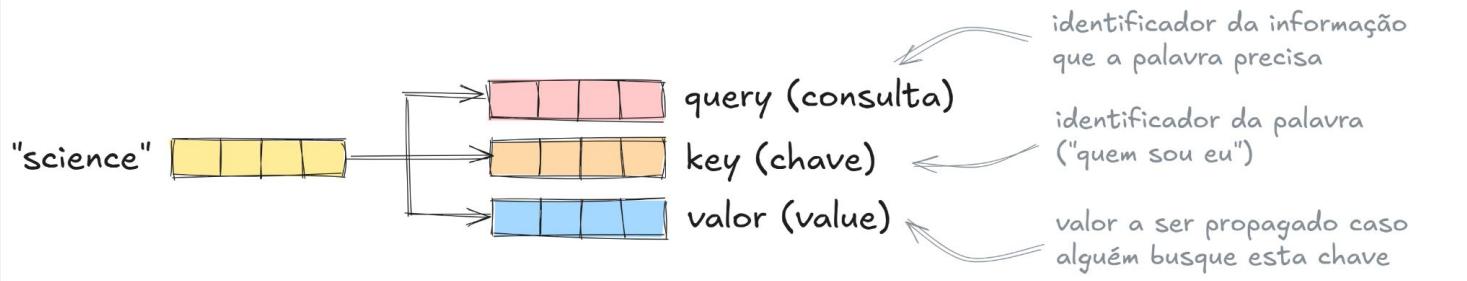
Self-Attention



Self-Attention

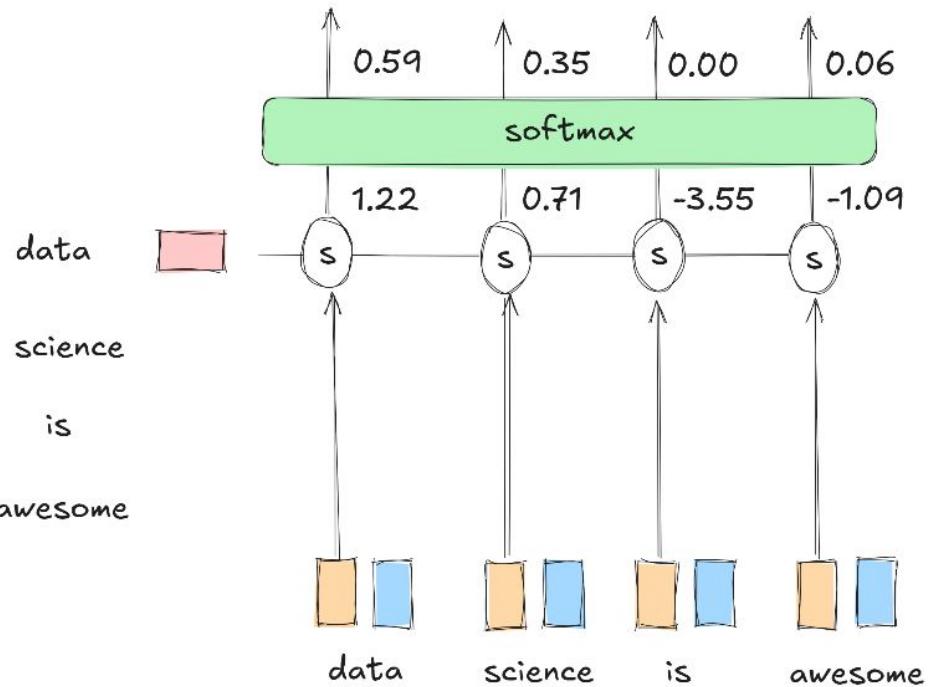


Self-Attention



as mesmas matrizes são utilizadas para todos os *embeddings* de entrada

$$\begin{aligned} q &= W^Q x \\ k &= W^K x \\ v &= W^V x \end{aligned}$$

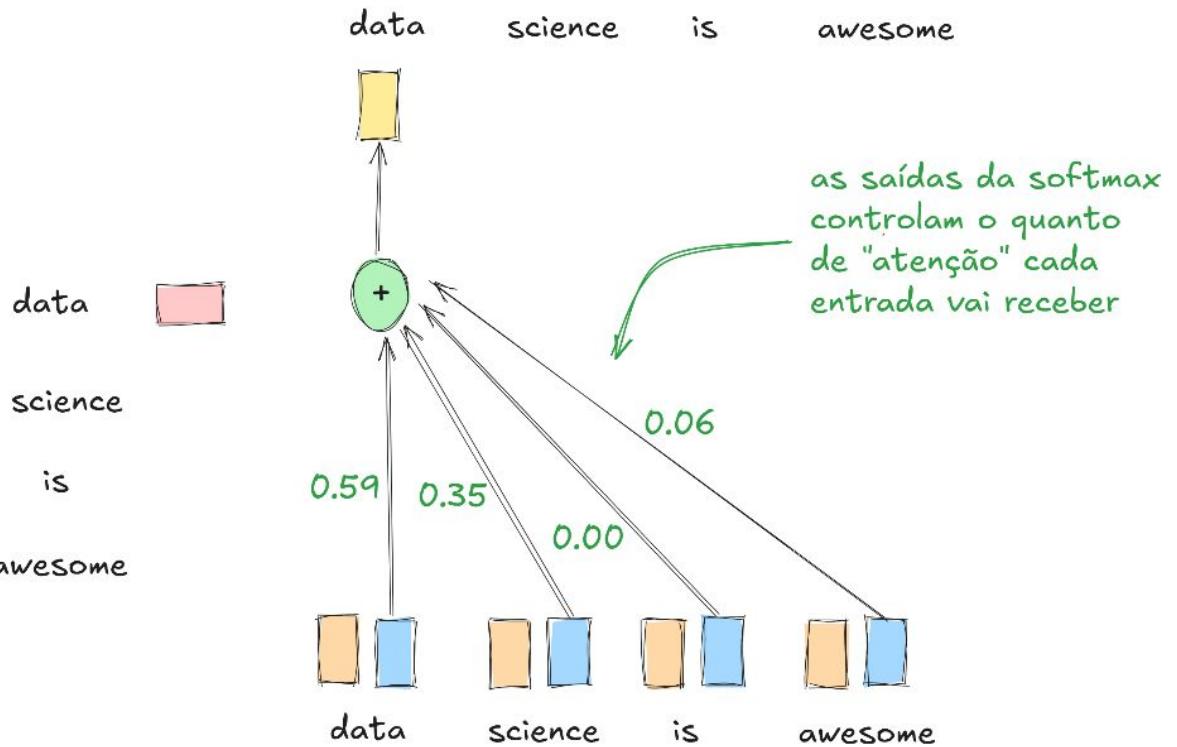


1

comparamos a query de uma palavra com as keys das outras palavras usando o produto escalar escalado (scaled dot-product)

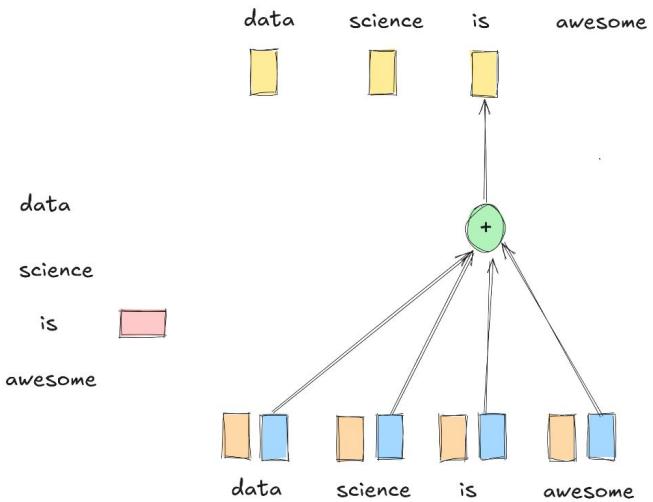
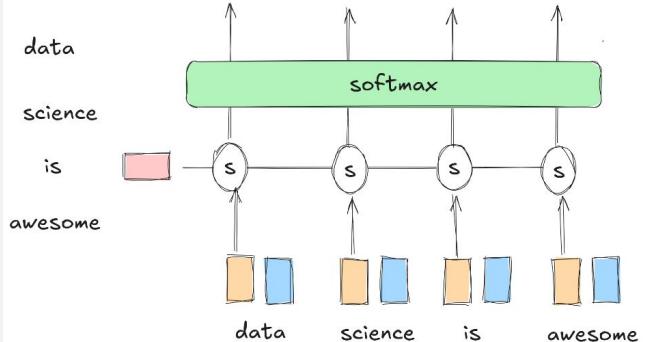
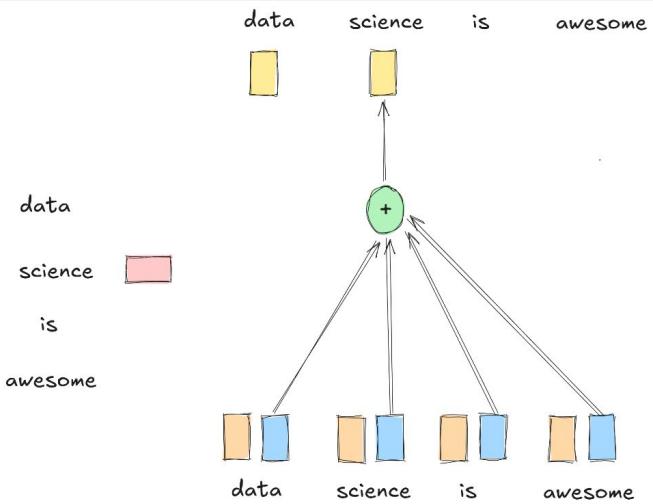
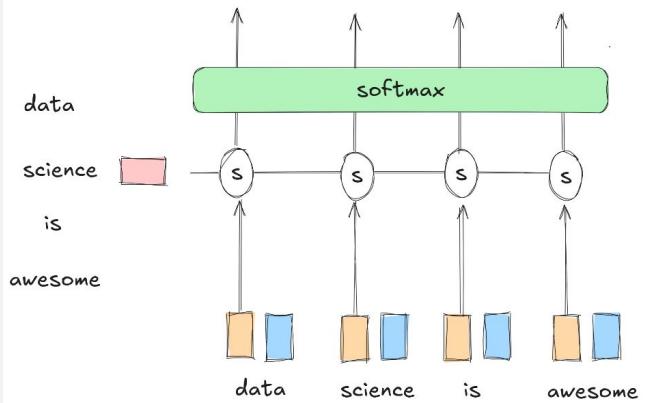
$$s(q, k) = (q \cdot k) / \sqrt{d}$$

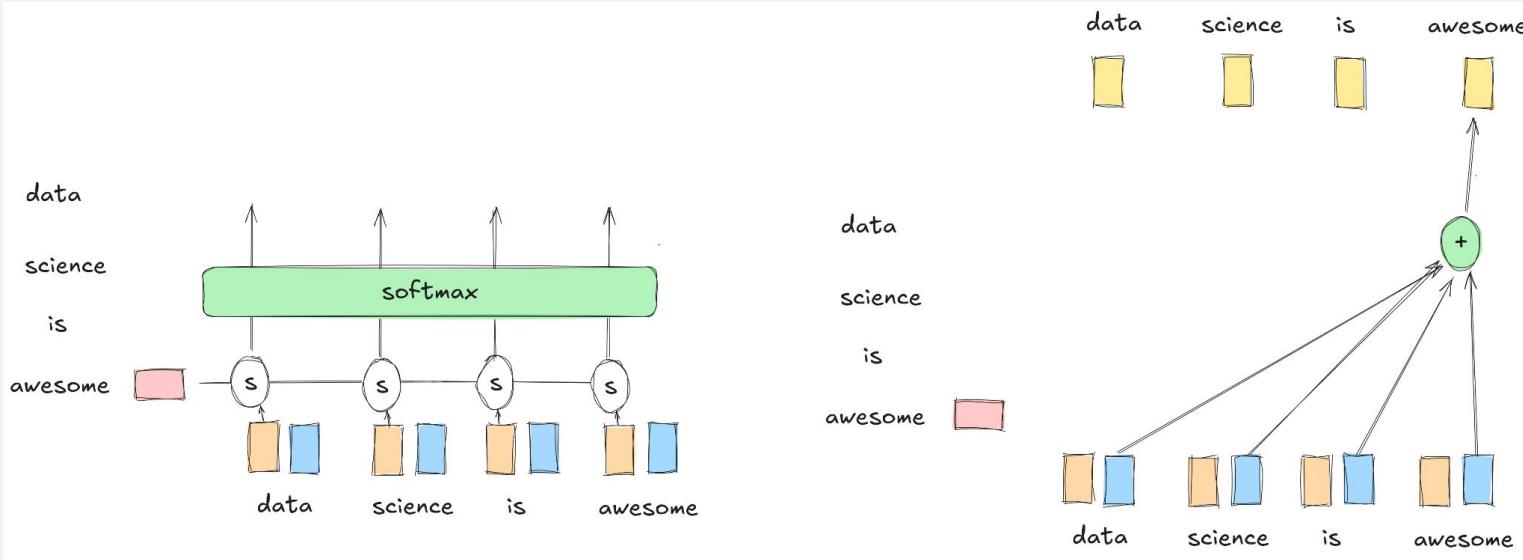
d é o tamanho dos vetores q e k



2

o embedding de saída para o primeiro token será a soma dos values ponderada pelas saídas da softmax



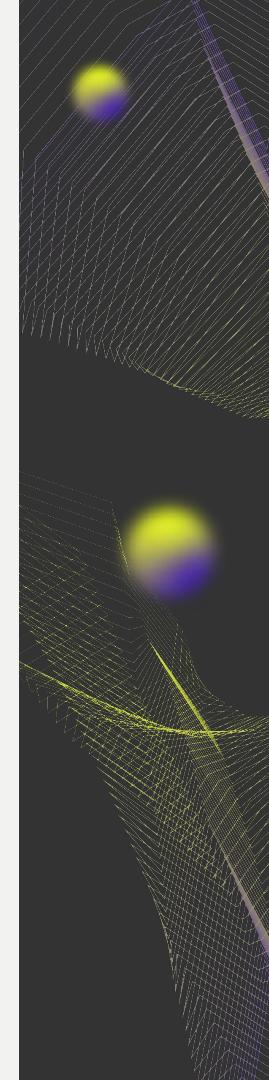




Pontos de Atenção !

- As operações de comparação de queries e keys e a propagação dos valores é feita **independentemente para todos os embeddings de entrada.**
- Isto proporciona **maior paralelismo!**
- A saída do mecanismo de atenção pode ser computado eficientemente fazendo:

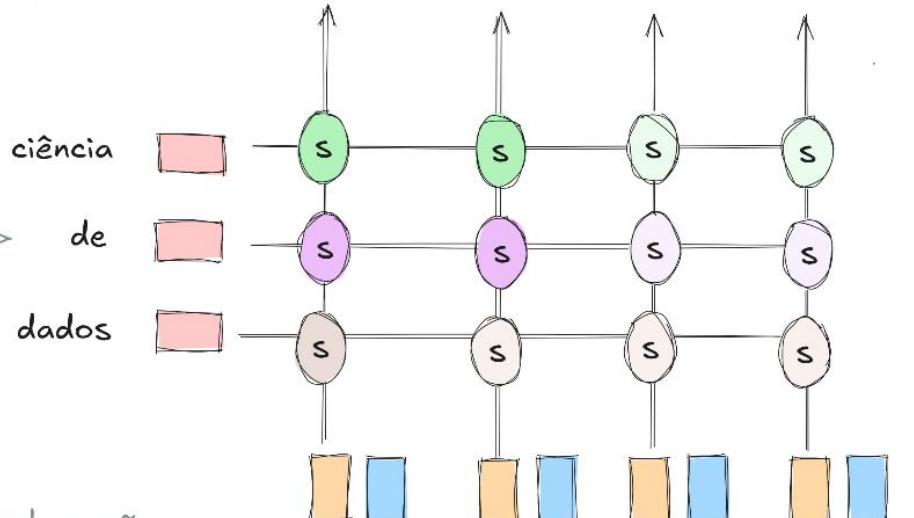
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Encoder-Decoder Attention

- 1 compararmos queries do decoder com keys do encoder usando scaled dot product seguido de softmax

queries vêm
do decoder

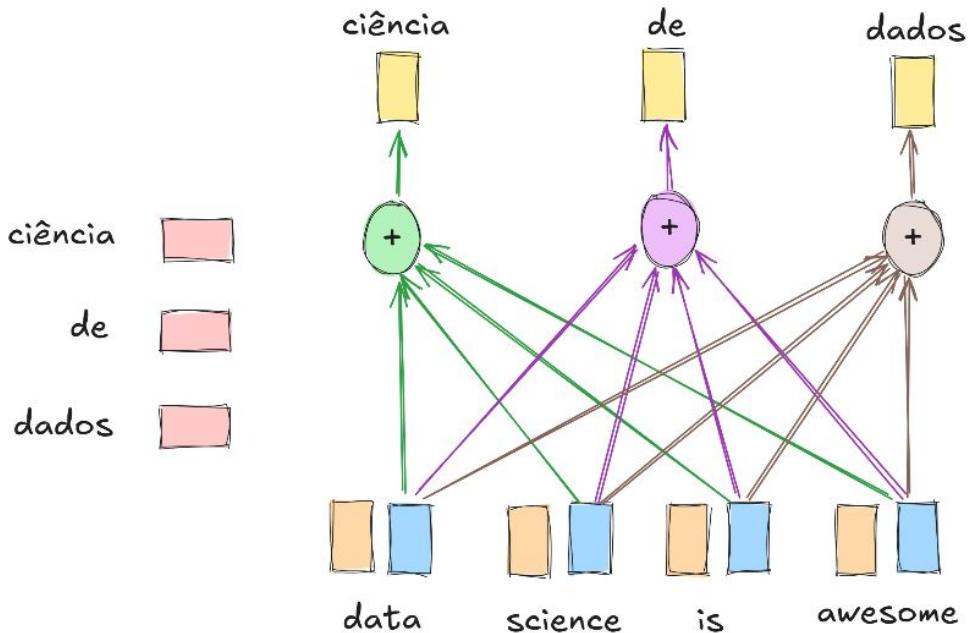


keys e values são
calculados a partir dos
embeddings de saída
do encoder

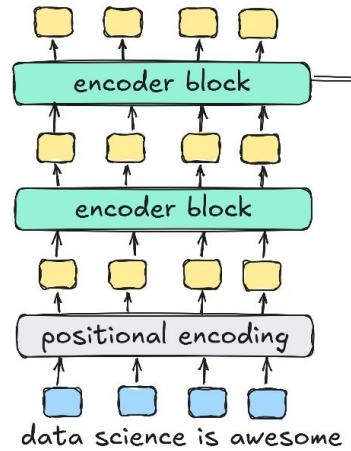
Encoder-Decoder Attention

2

os tokens de saída são obtidos somando os values do encoder ponderados pelos coeficientes de atenção.



a saída para cada embedding de entrada é a próxima palavra



ciência de dados é massa <fim>

Um token especial é usado para indicar o fim da tradução

ciência: 0.53
dados: 0.21
de: 0.13
massa: 0.09
...

token predictor (linear + softmax)

decoder block

decoder block

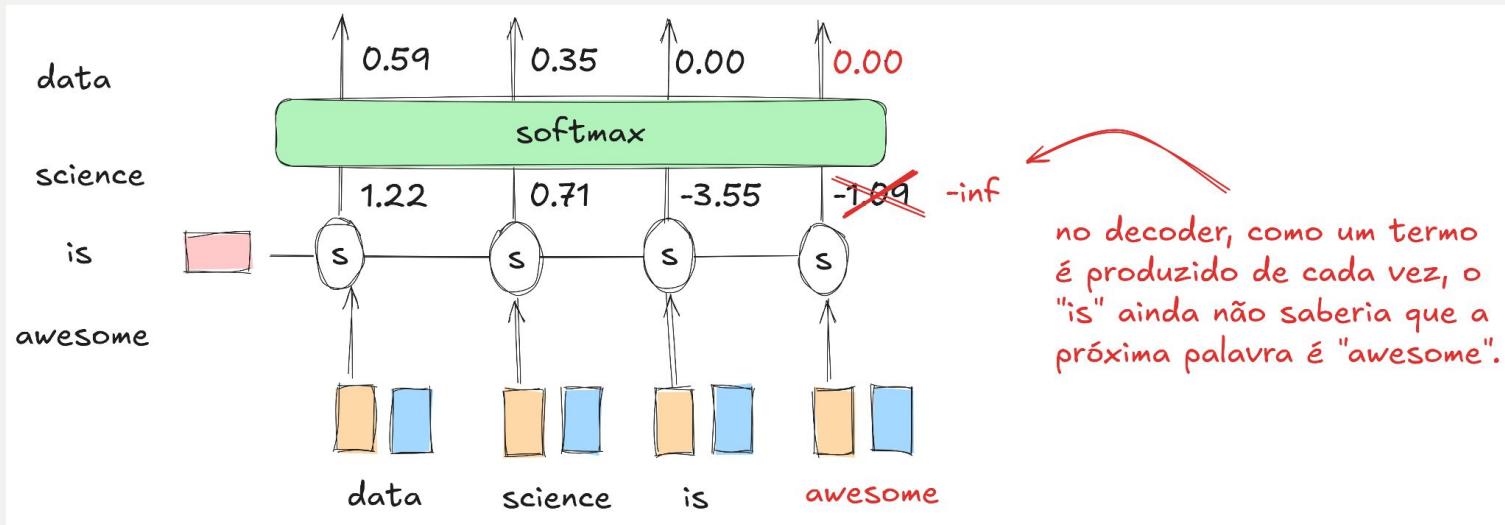
positional encoding

<start> ciência de dados é massa

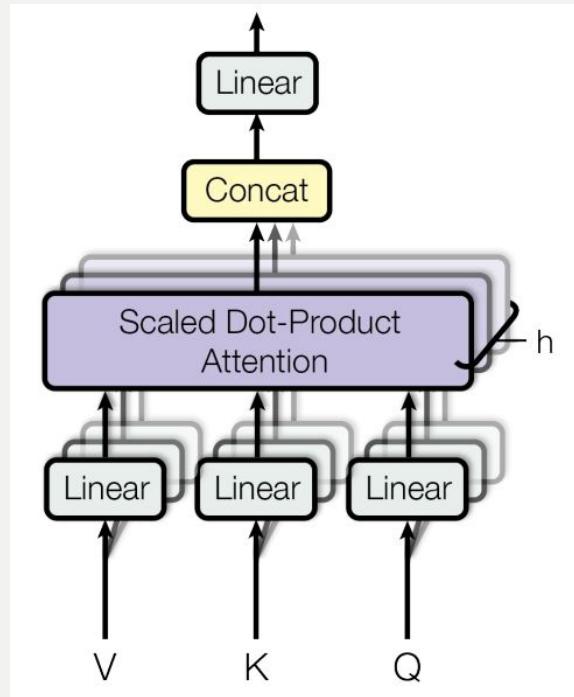
Um token especial é usado para iniciar a tradução

Durante o treino, podemos treinar todas as previsões de próximas palavras ao mesmo tempo! Só precisamos tomar cuidado para na self-attention, um embedding não veja o futuro usando masked self-attention.

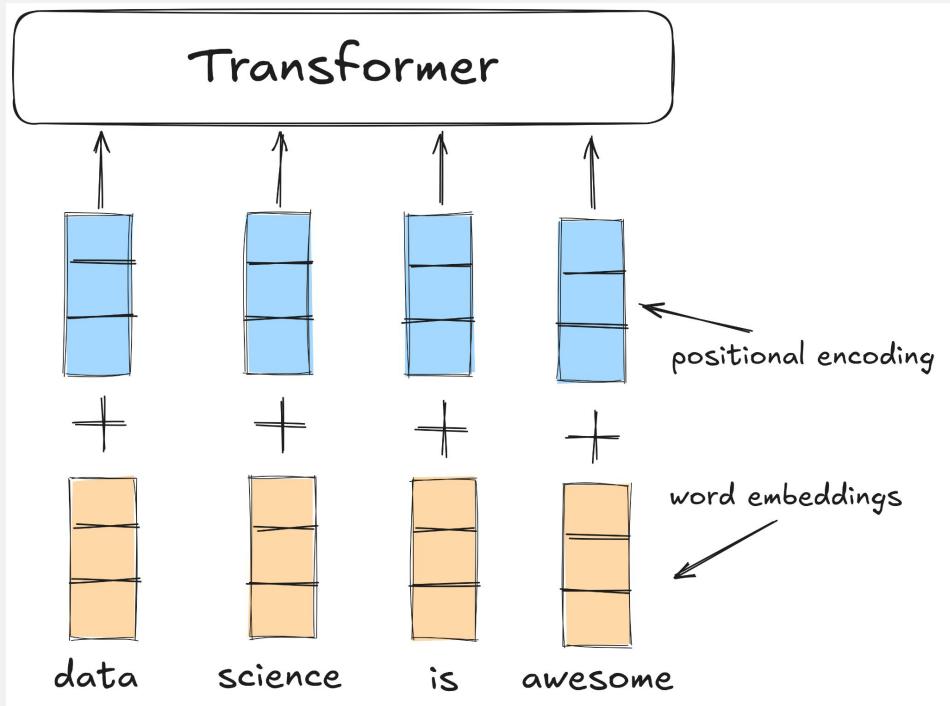
Masked Self-Attention



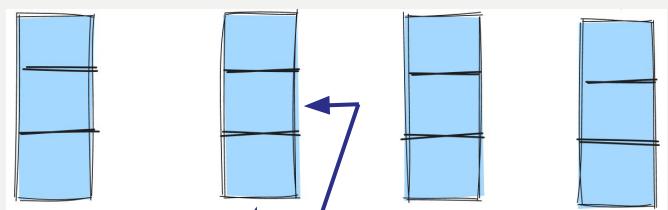
Multi-head Attention



Positional Encoding

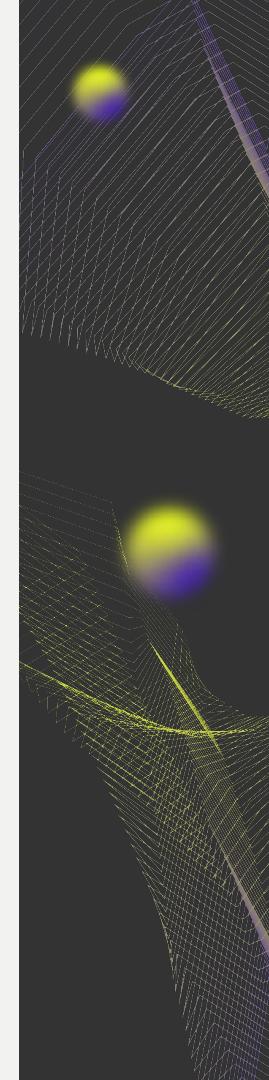
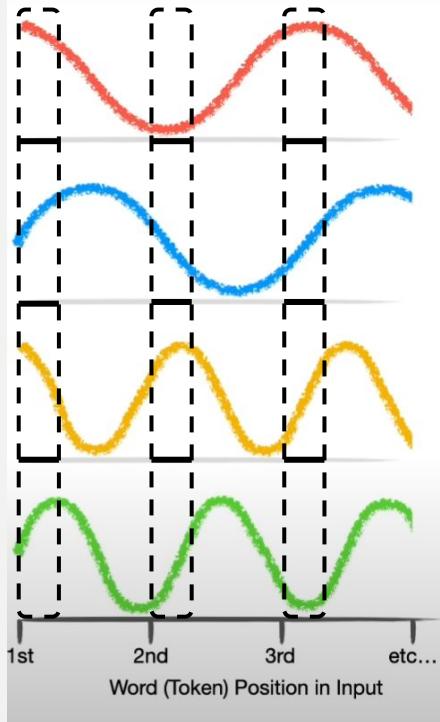


Positional Encoding



$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$



Layer Normalization

Tem como objetivo uniformizar os valores das ativações das camadas ao longo do treinamento.

- Melhora a estabilidade do treinamento
- Permite o uso de taxas de aprendizado maiores
- Acelera a convergência



Layer Normalization

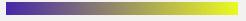
Dadas as ativações de uma camada $x = [x_1, \dots, x_d]$, calculamos a média e desvio padrão e usamos para normalizar as ativações.

$$\mu = \frac{1}{d} \sum_{i=1}^d x_i$$

$$\sigma = \sqrt{\frac{1}{d} \sum_{i=1}^d (x_i - \mu)^2 + \epsilon}$$

$$\hat{x}_i = \frac{x_i - \mu}{\sigma}$$

epsilon é um valor
pequeno usado para
evitar divisões por zero

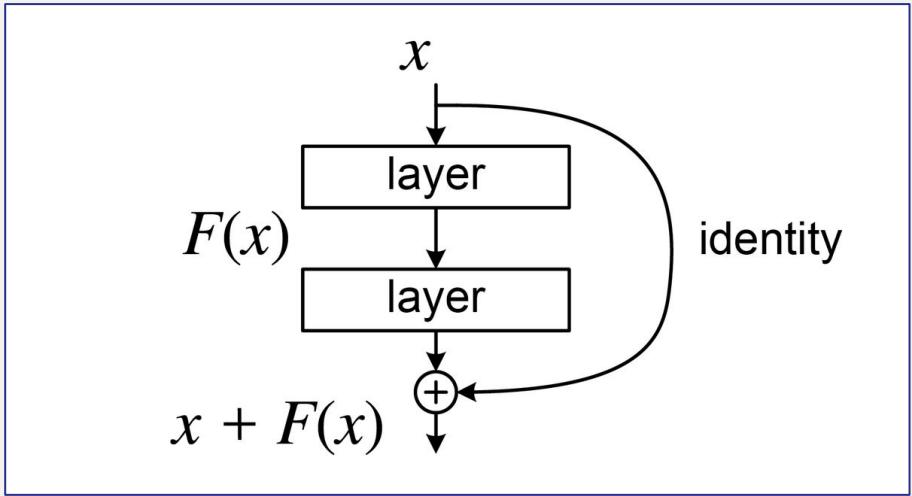


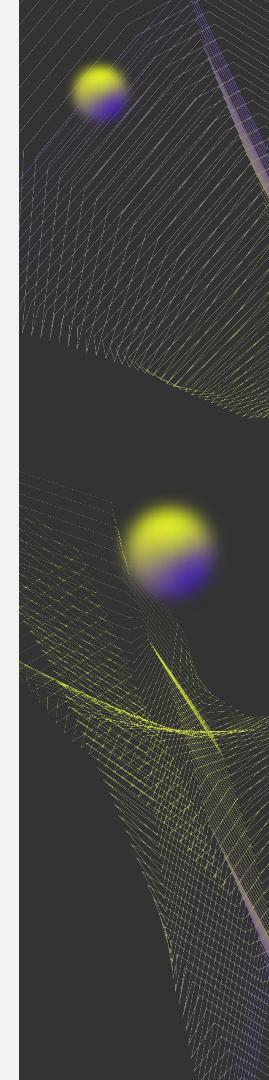
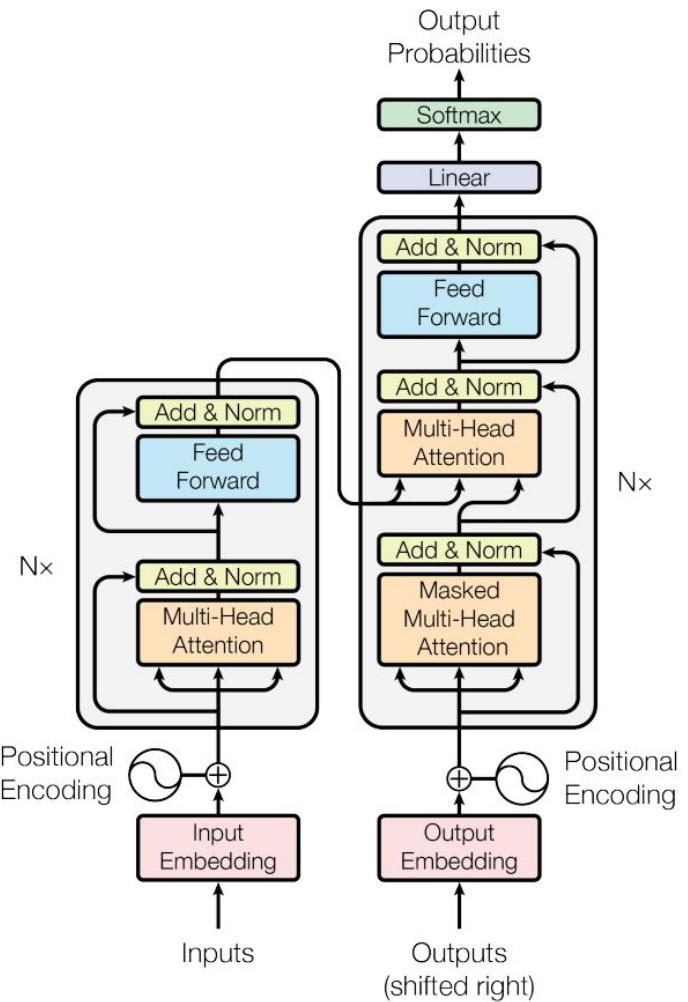
Layer Normalization

A saída da operação, y_i , é obtida realizando uma operação linear na qual os parâmetros são vetores treináveis do mesmo tamanho de entrada.

$$y_i = \gamma_i \hat{x}_i + \beta_i$$

Conexão Residual





Conclusões

- Treinar modelos Transformer do zero requer muitos dados e computação!
- Os mecanismos de atenção são quadráticos no número de *embeddings* da entrada.
- No próximo vídeo, veremos como usar uma arquitetura baseada em Transformers pré-treinada para classificação de textos!

INTELIGÊNCIA ARTIFICIAL & CIÊNCIA DE DADOS



Prof. Filipe Wall Mutz