

A Survey of Transformers

Nator e Pedro

December 12, 2023



- 1 Transform Vanilla;
- 2 Encode.
- 3 Decode.

Apresentação de algumas arquiteturas transformes (X-forms) sobre as perspectivas de:

- Modificação arquitetônica;
- pré-treinamento;
- aplicações.

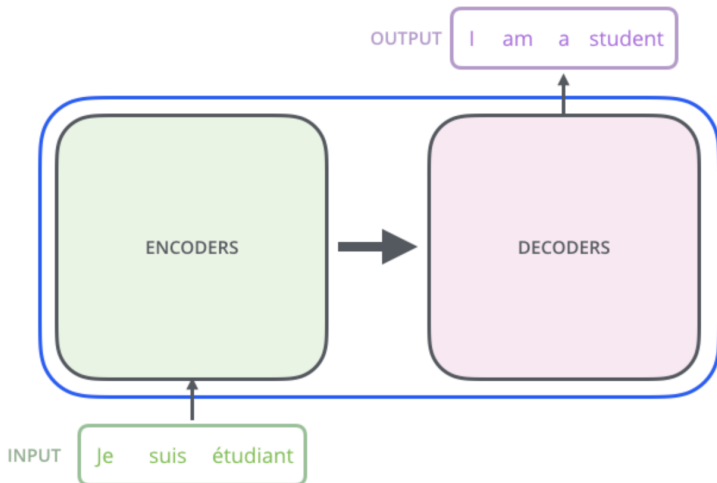
Melhorias dos transformes:

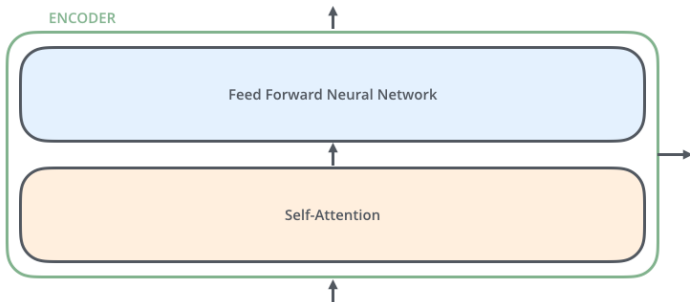
- **Eficiência do modelo:** Um dos principais desafios da aplicação do Transformer é a sua ineficiência no processamento de sequências longas.
- **Generalização do Modelo:** O Transformer é uma arquitetura flexível e faz poucas suposições sobre o viés estrutural dos dados de entrada. No entanto, treinar o modelo com dados de pequena escala pode ser desafiador.

Modelo Vanilla Transformes



Resumo sobre o artigo







$$FFW(x) = \max(0, x * W_1 + bias_1) * W_2 + bias_2 \quad (1)$$

- A codificação de uma palavra com embedding envolve a representação numérica de uma palavra em um espaço vetorial contínuo.

```
1 -> "gato": [0.2, 0.5, -0.8, 0.3]
2 -> "cachorro": [-0.6, 0.9, 0.1, -0.4]
```

- Esses vetores capturam características semânticas das palavras.

- **Vetor de Consulta Q:** Cada palavra de entrada é associada a um vetor de consulta. Este vetor representa a palavra e será usado para calcular a importância (atenção) dessa palavra em relação às outras palavras na sequência.
- **Vetor de Chave K:** Similarmente, cada palavra tem um vetor de chave. Este vetor é usado para calcular a similaridade entre a palavra atual e outras palavras na sequência.
- **Vetor de Valor V:** Cada palavra também tem um vetor de valor. Este vetor é uma representação da palavra que será usada para formar a saída da operação de autoatenção.

Calculando os vetores



Vanilla


Input

Thinking

Machines

Embedding

x_1 

x_2 

Queries

q_1 

q_2 



W^Q

Keys

k_1 

k_2 



W^K

Values

v_1 

v_2 



W^V

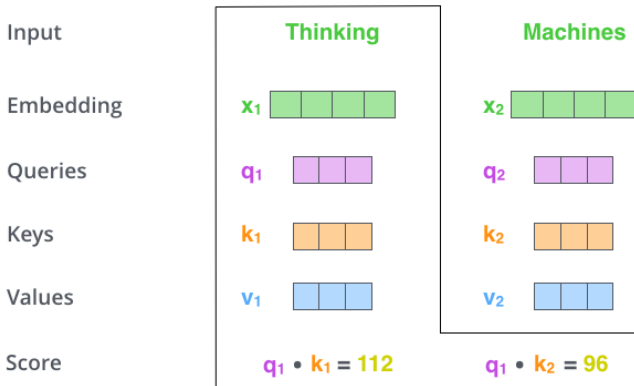
$$Q_i = X_i * W^Q$$

$$K_i = X_i * W^K$$

$$V_i = X_i * W^V$$

(2)

Self-attention



$$\text{score}(Q_i, K) = Q_i * k_j^T$$

(3)

Input

Embedding

Queries

Keys

Values

Score

Divide by 8 ($\sqrt{d_k}$)

Softmax

Thinking

x_1

q_1

k_1

v_1

$q_1 \cdot k_1 = 112$

14

0.88

Machines

x_2

q_2

k_2

v_2

$q_2 \cdot k_2 = 96$

12

0.12

$$\text{score}(Q_i, K) = \frac{Q_i * k_j^T}{\sqrt{d_k}}$$

(4)

Input

Embedding

Queries

Keys

Values

Score

Divide by 8 ($\sqrt{d_k}$)

Softmax

Softmax

X

Value

Sum

Thinking

x_1

q_1

k_1

v_1

$q_1 \cdot k_1 = 112$

14

0.88

v_1

z_1

Machines

x_2

q_2

k_2

v_2

$q_2 \cdot k_2 = 96$

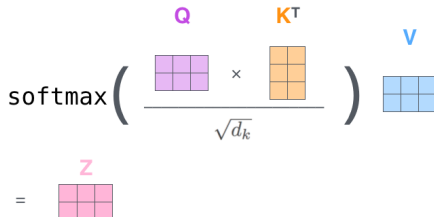
12

0.12

v_2

z_2

- Calculando a saída (z) para a palavra i

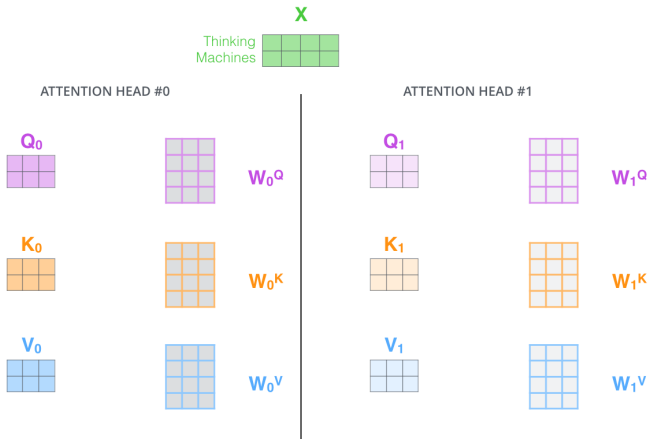

$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \times V = Z$$

$$z_i = \sum_{j=0}^{d_k} \text{softmax}\left(\frac{Q_i * K_j^T}{\sqrt{d_k}}\right) * V_j \quad (5)$$

Multhead attention



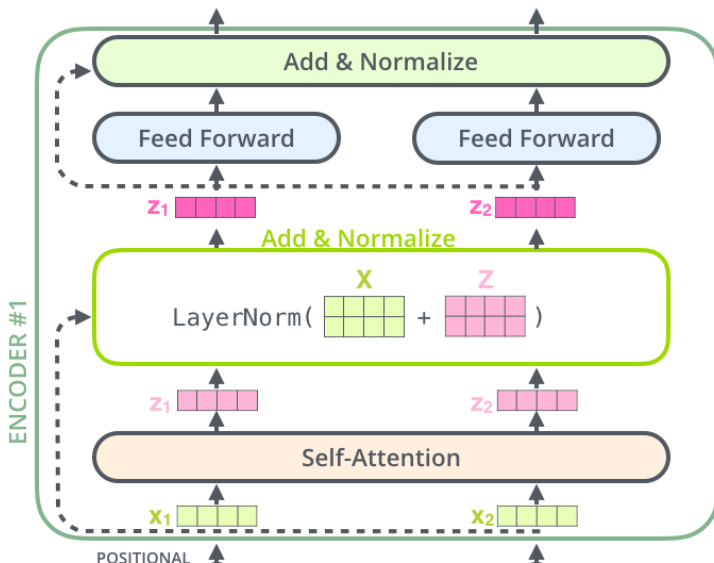
Vanilla

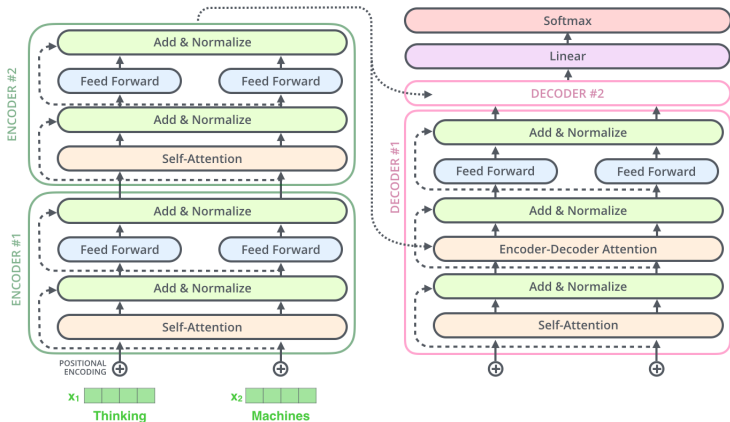


Multithread attention



Vanilla







- **Codificador-Decodificador:** Problemas sequence to sequence;
- **Somente codificador:** Normalmente é usado como entrada em outro modelos para resolução de problemas de classificação;
- **Somente decodificador:** Isso normalmente é usado para geração de sequência, como modelagem de linguagem.



Uma grande variedade de modelos foi proposta até agora com base no Transformer vanilla a partir de três perspectivas: tipos de modificação de arquitetura, métodos de pré-treinamento e aplicações.

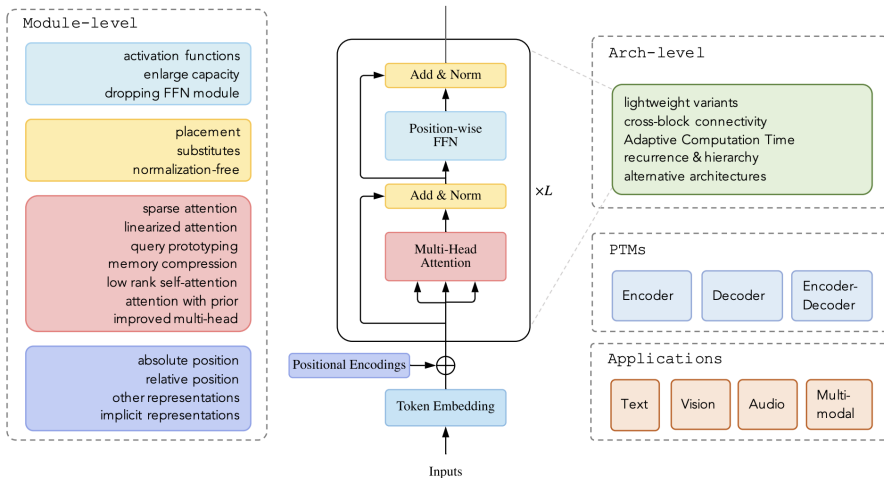


Fig. 2. Categorization of Transformer variants.



A Survey of Transformers

Nator e Pedro

December 12, 2023