Tópicos em Ciência de Dados

Revisão de Deep Learning e Introdução aos LLMs

Pontifícia Universidade Católica de Campinas

Prof. Dr. Denis M. L. Martins

Objetivo de aprendizado

Relembrar os conceitos fundamentais que sustentam as arquiteturas modernas de IA, preparando-nos para a construção de modelos avançados (Transformers, LLMs, etc.).

Redes Neurais – Visão Geral

- **Neurônio artificial**: soma ponderada de entradas + bias → função de ativação → saída
- **Rede**: conjunto de neurônios conectados em camadas (entrada, ocultas, saída)
- **Propagação direta** (`forward pass`): computa a predição
- **Retro-propagação** (`backward pass`): calcula gradientes para ajuste de pesos

Estrutura de uma Rede Neural

Estrutura Básica

1. Camada de Entrada (input layer)
2. Camadas Ocultas (hidden layers): **Linear + Non-linearity**
3. Camada de Saída (output layer)

Fórmula geral da saída de uma camada: $\mathbf{h}^{(l)} = \sigma\left(\mathbf{W}^{(l)}\,\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right)$

- σ: função de ativação (ReLU, sigmoid, tanh, GELU, etc.)
- $\mathbf{W}^{(l)}$, $\mathbf{b}^{(l)}$: pesos e bias aprendidos

Funções de Ativação

Função	Fórmula	Quando usar
ReLU	$\max(0, x)$	Camadas ocultas profundas (evita gradiente morto)
Sigmoid	$\frac{1}{1+e^{-x}}$	Saída binária ou probabilidades
Tanh	$\tanh(x)$	Quando se deseja saída centrada em zero
GELU	$x\Phi(x)$ (onde Φ é a CDF normal)	Transformers, BERT, GPT

- ReLU é padrão para CNNs e RNNs; GELU aparece nas arquiteturas Transformer.

Perda (Loss) e Otimização

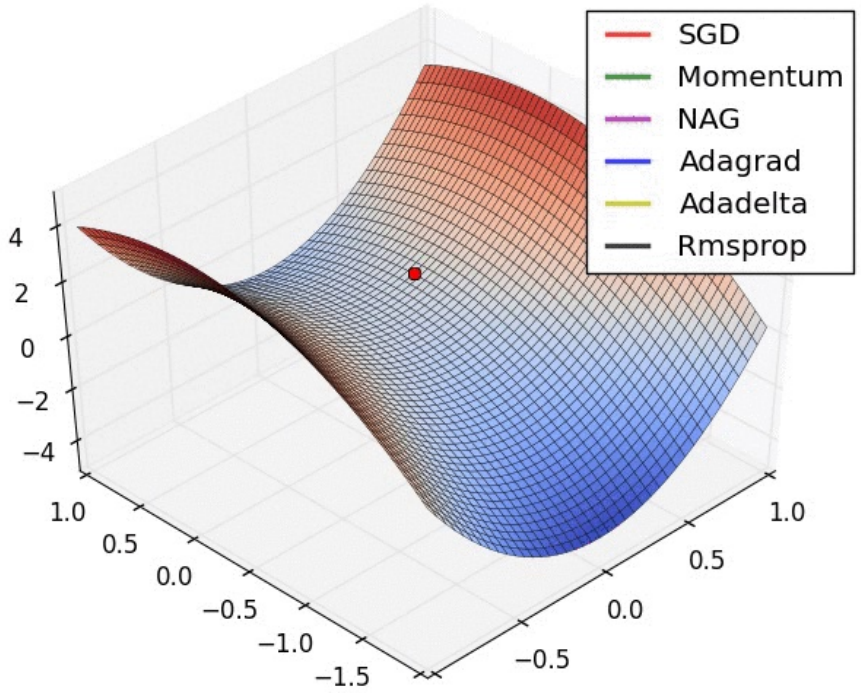
Funções de Perda

- MSE (regressão): $\mathcal{L}_{\mathcal{MSE}} = \frac{1}{N}\sum_i (y_i - \hat{y}_i)^2$
- Cross-Entropy (classificação): $\mathcal{L}_{\mathcal{CE}} = -\frac{1}{N}\sum_i \log p_\theta(y_i|x_i)$

Otimizadores

- SGD + momentum:
  - $\mathbf{v}_{t+1} = \gamma \mathbf{v}_t + \eta \nabla_\theta \mathcal{L}$
  - $\theta_{t+1} = \theta_t - \mathbf{v}_{t+1}$
- Adam: combina RMSProp e momentum

Algoritmo de Atualização (Adam)



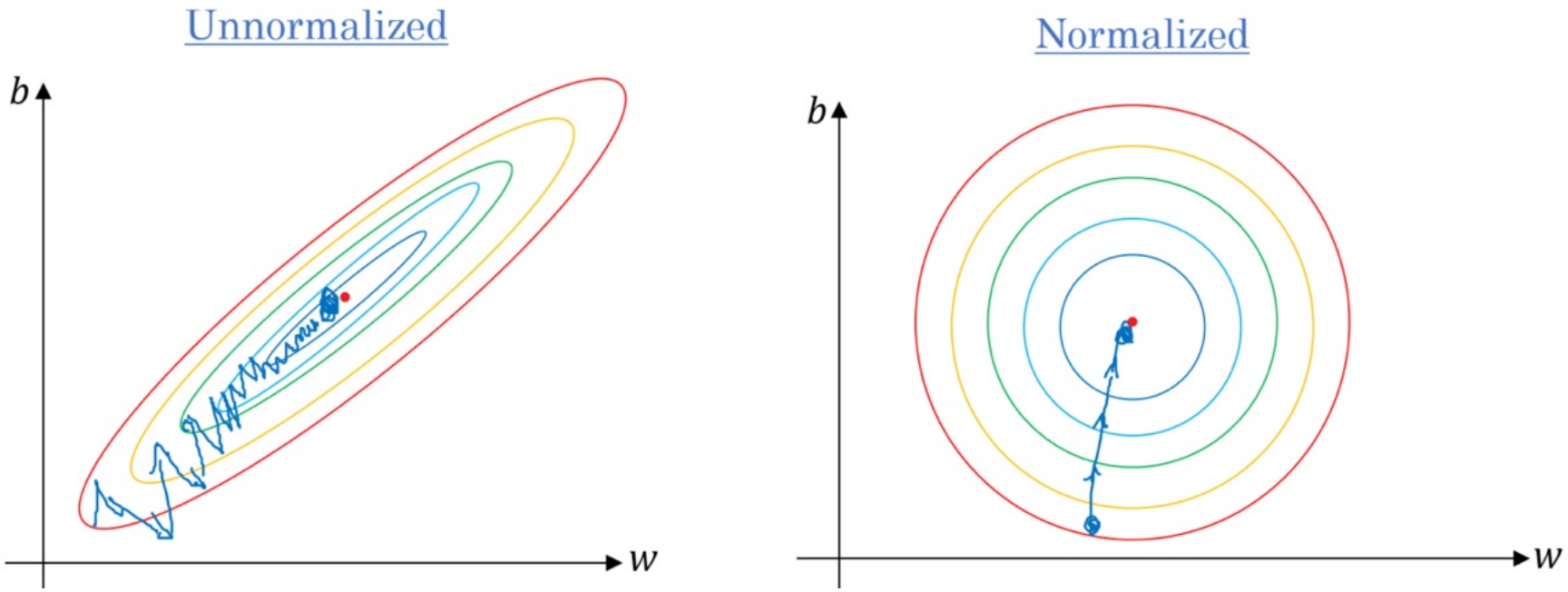
$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta \mathcal{L}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla_\theta \mathcal{L})^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Gradient Descent e Feature Scaling



Regularização

- **Dropout**[1]: zero out neurônios aleatoriamente, $z^{(l)} = \mathbf{h}^{(l)} \odot \text{Bernoulli}(p)$
- **Weight Decay (L2)**: adicionar $\lambda ||\theta||_2^2$ à função de loss
- **Batch Normalization**: normaliza saída de cada mini-batch
  $\hat{x} = \frac{x - \mu_{\text{batch}}}{\sqrt{\sigma^2_{\text{batch}} + \epsilon}}$,
  $y = \gamma \hat{x} + \beta$
- **Early Stopping**: interromper iterações (épocas) quando validação piora

[1] Dropout em Transformers é usado principalmente nas camadas feed-forward e no final das cabeças de atenção.

Arquiteturas Convencionais (Resumo)

Tipo	Exemplo	Principais Componentes
CNN	LeNet, ResNet, EfficientNet	Convoluções, pooling, skip-connections
RNN	LSTM, GRU	Gating, memória recorrente
Transformers	BERT, GPT, T5	Multi-head self-attention, positional encoding, feed-forward

Em várias aplicações, Transformers substituíram RNNs em NLP por paralelização e capacidade de capturar dependências longas.

Veja também:

- https://poloclub.github.io/cnn-explainer/
- https://poloclub.github.io/transformer-explainer/

Preparação para LLMs

- Attention: mecanismo de foco (self-attention).
- Positional Encoding: informação de ordem em sequências.
- Pretraining & Fine-tuning: aprendizado não supervisionado + ajuste fino.

Exercício Rápido para Revisão e Estudo

1. Implemente uma camada Linear + ReLU no PyTorch e treine em MNIST.
2. Adicione Dropout, por exemplo $p = 0.3$, e compare a acurácia de validação.
3. Documente os efeitos do dropout nos gráficos de loss/accuracy.

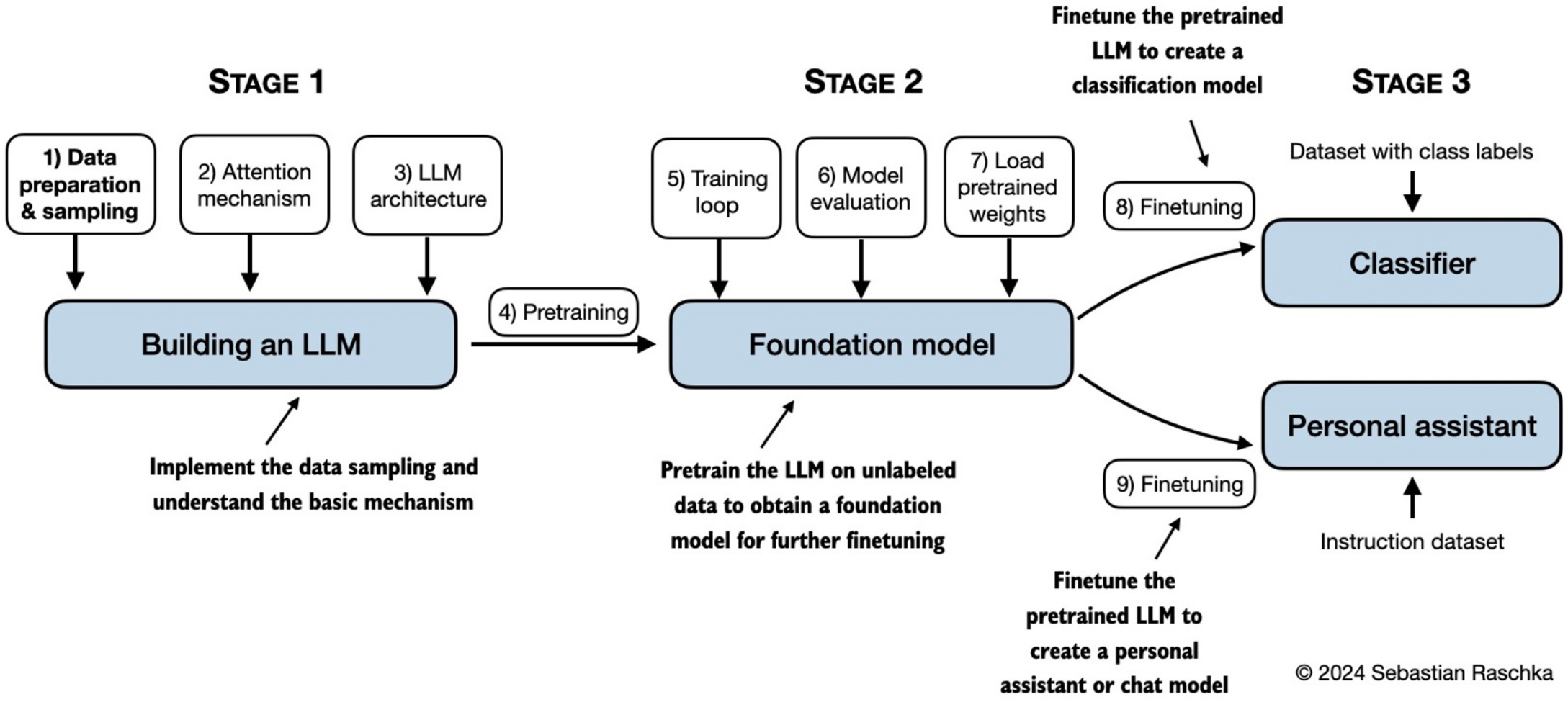
Leitura Recomendada

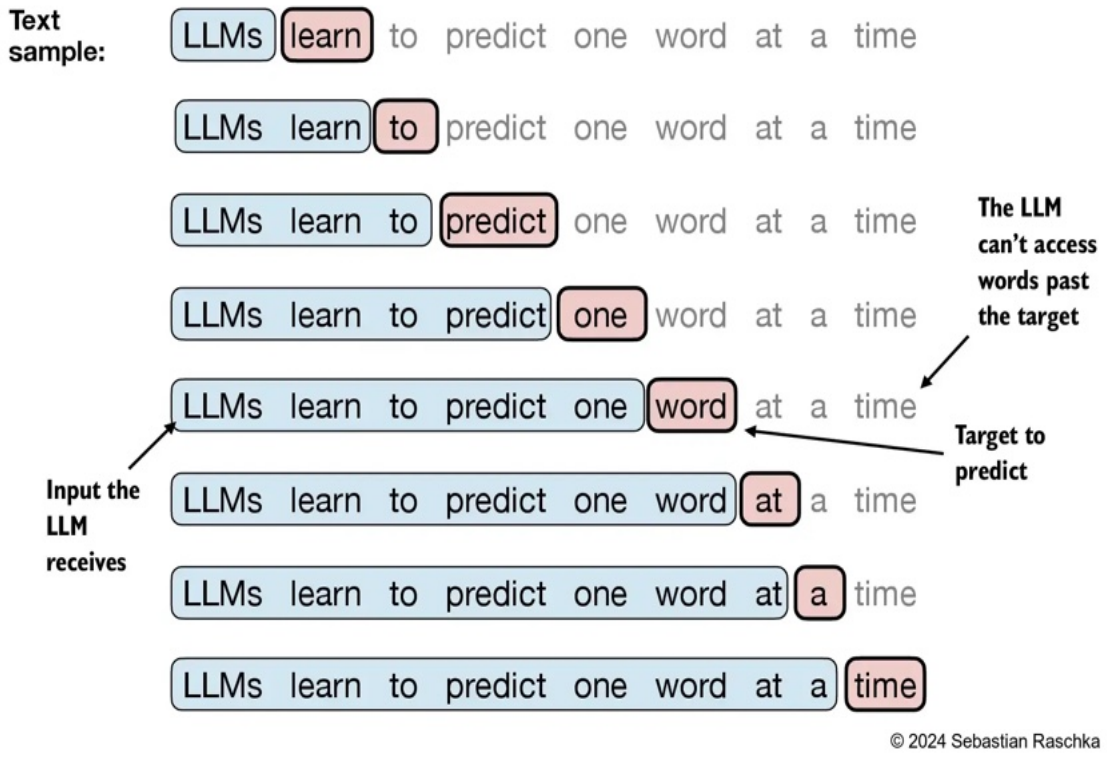
- **Deep Learning** – Ian Goodfellow, Yoshua Bengio, Aaron Courville. Capítulos 3–4 (Neural Networks) e 6 (Regularization & Optimization)
- **Attention Is All You Need** – Vaswani et al. (2017)
- **Transformers in PyTorch** – Hugging Face tutorial

Perguntas e Discussão

1. Por que a ReLU é preferida em camadas ocultas profundas?
2. Como o Dropout funciona em uma arquitetura Transformer?
3. Quais são as limitações dos otimizadores clássicos (SGD) quando se trabalha com LLMs?

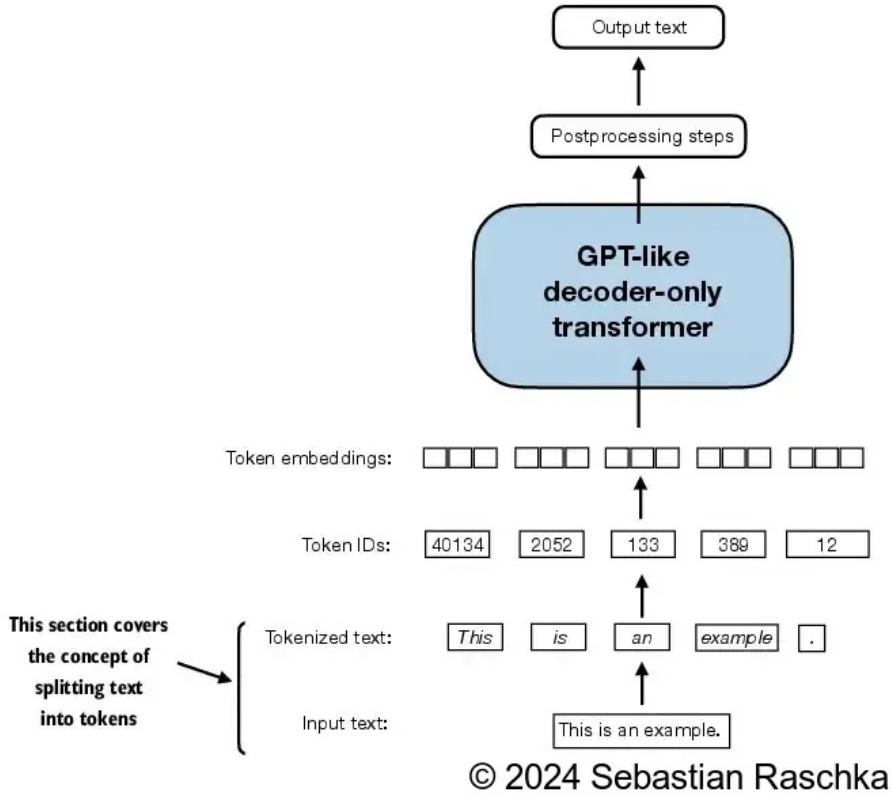
Construindo um LLM: Visão Geral

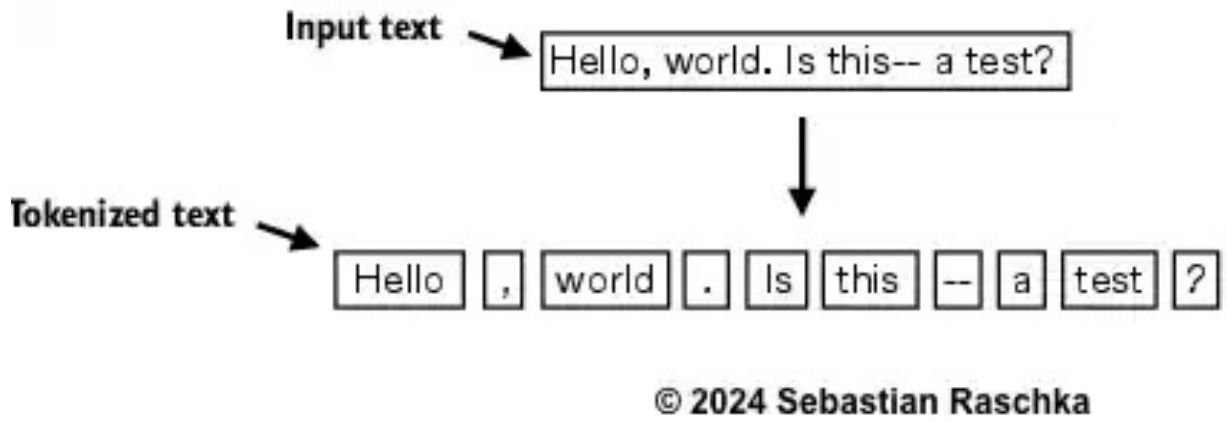




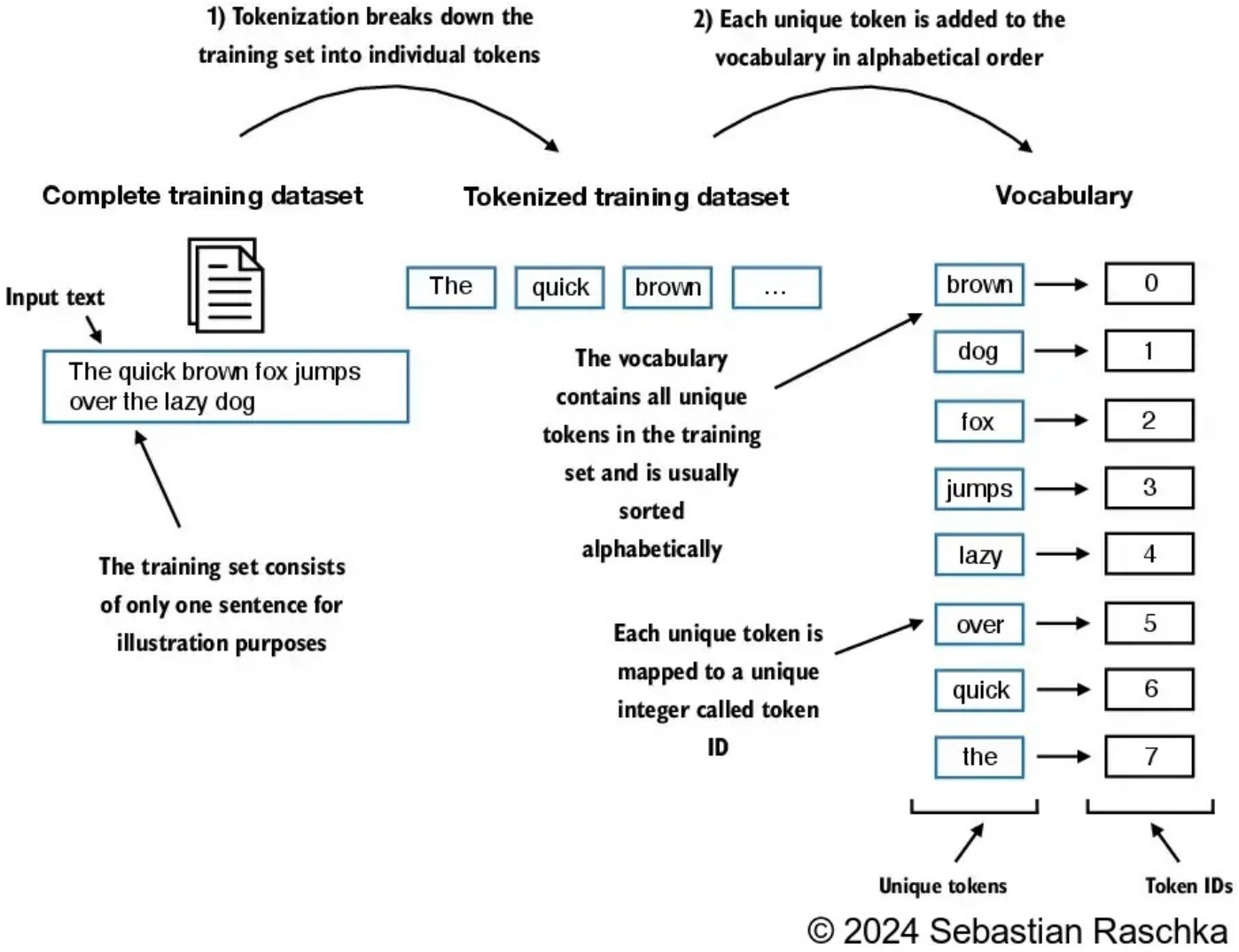
LLMs são pré-treinadas para prever a próxima... **palavra**.

Tokenizer





Use https://www.bpe-visualizer.com/ e https://tiktokenizer.vercel.app com o texto The Verdict (1908) de Edith Wharton: https://en.wikisource.org/wiki/The_Verdict



Token IDs

- LLMs operam sobre sequências numéricas; os ids permitem representar qualquer token com um número inteiro que pode ser processado por matrizes de embeddings.
- A tabela de embeddings converte IDs em vetores densos que carregam semântica aprendida durante o pré-treino.
- Token IDs são apenas endereços para os embeddings.

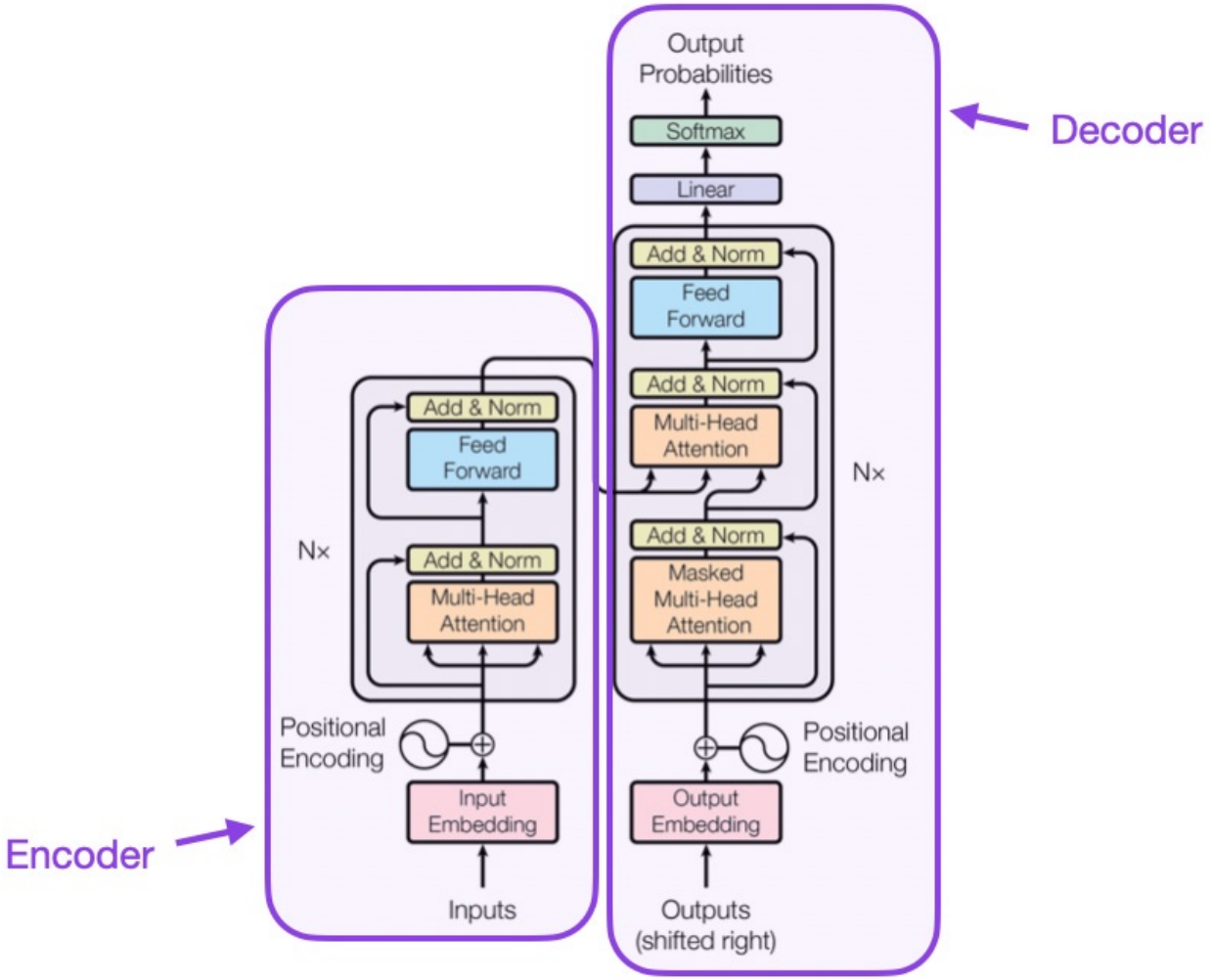
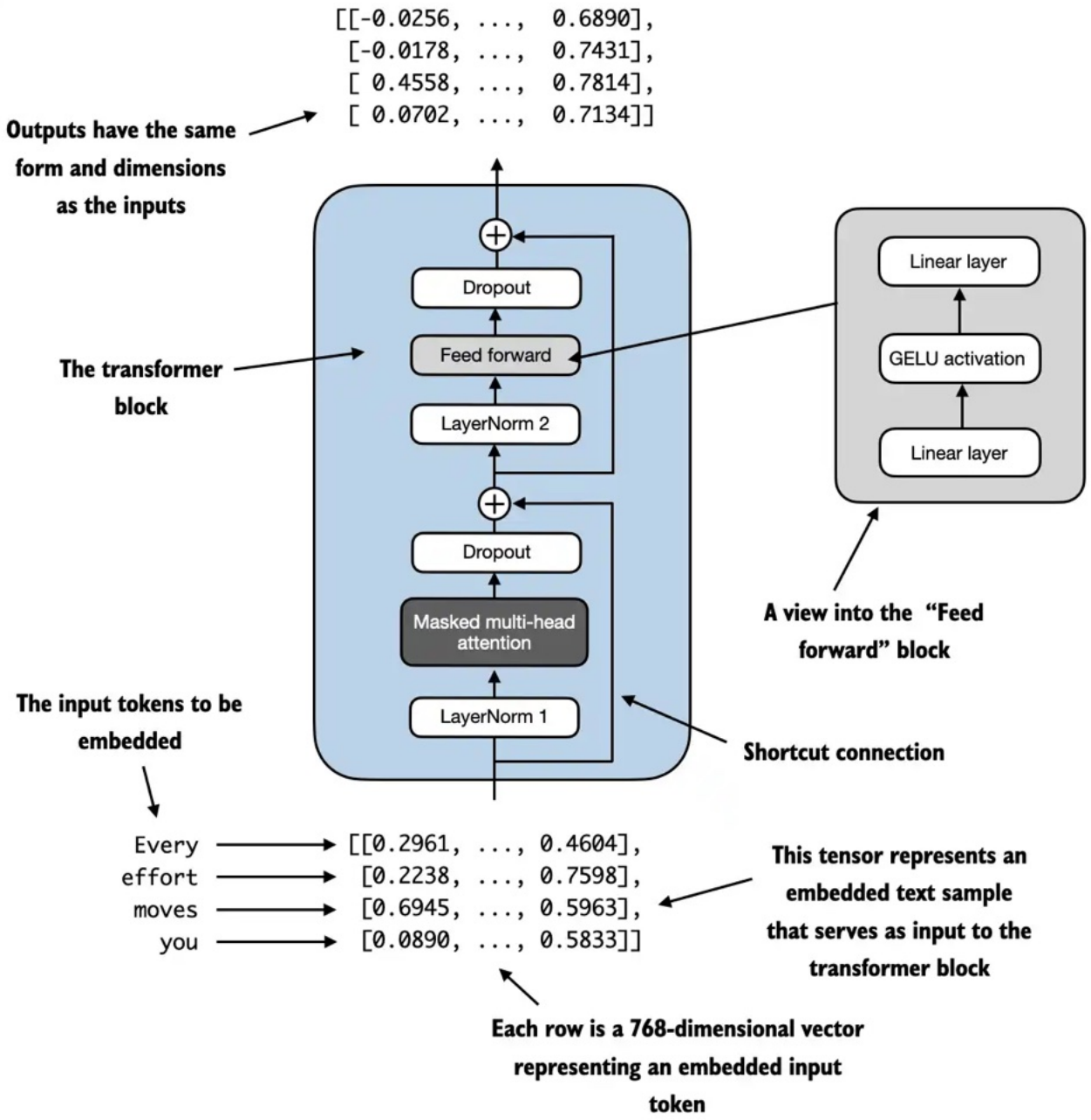


Figure 1: The Transformer - model architecture.

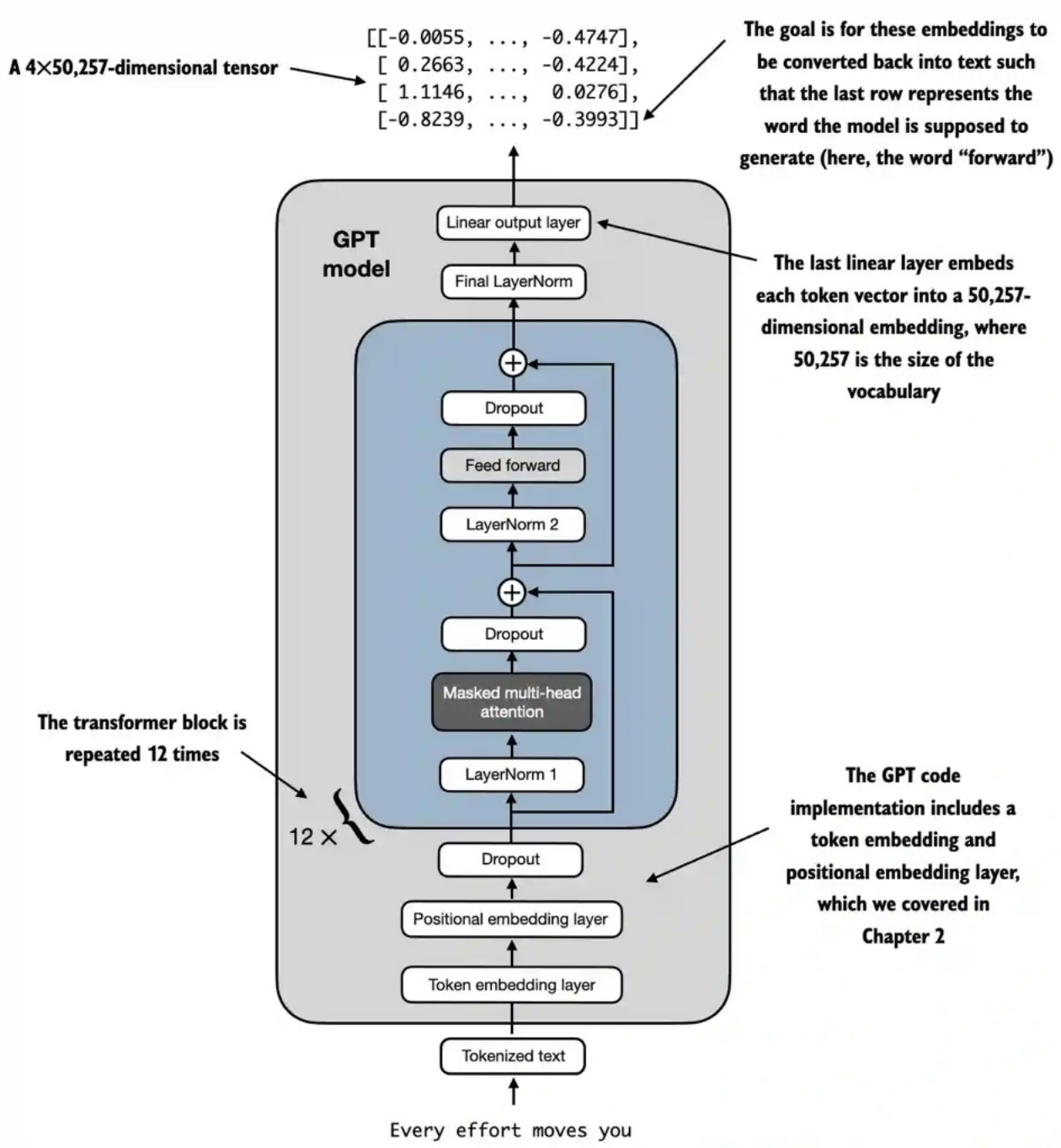
Attention is all you need

- Relações de longa-distância: "o gato que estava no quarto do apartamento 3 foi encontrado por..."
- Qual parte do texto é mais importante agora?
- Associa mais peso às palavras que parecem importantes, menos a aquelas que não fazem sentido.



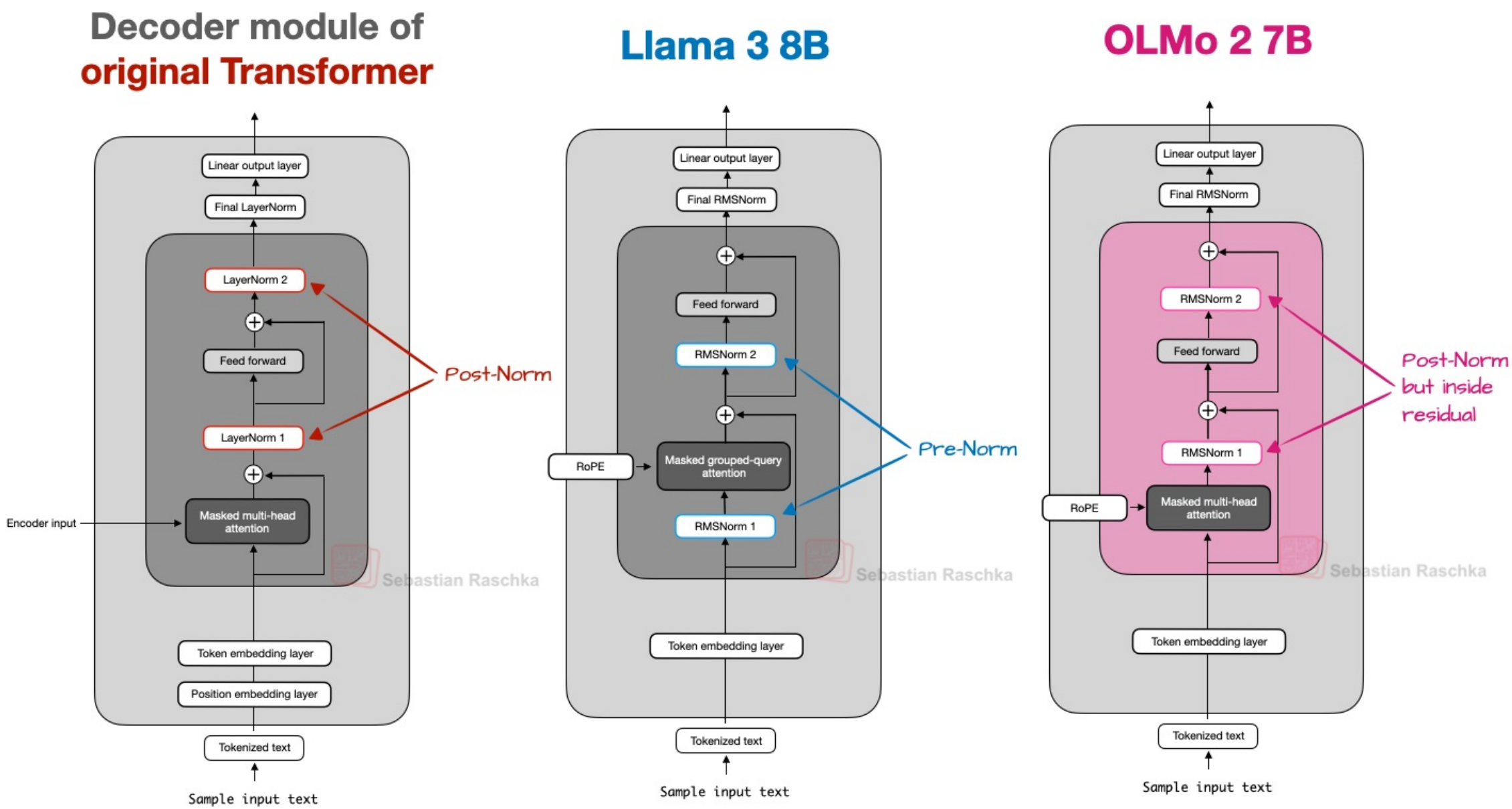
Bloco Transformer

- Multi-head attention
- Linear Layers
- Dropout and shortcut connections
- Camadas de normalização



Arquitetura GPT Original

- Texto tokenizado
- Camadas de Embedding
- Bloco(s) Transformer
  - 12 em `gpt2-small`
  - 36 em `gpt2-large`
- Camada linear na saída
- Figura de S. Raschka


Decoder module of
original Transformer
Linear output layer
Final LayerNorm
LayerNorm 2
Feed forward
LayerNorm 1
Masked multi-head attention
Encoder input
Token embedding layer
Position embedding layer
Tokenized text
Sample input text
Post-Norm
Sebastian Raschka
Llama 3 8B
Linear output layer
Final RMSNorm
Feed forward
RMSNorm 2
RoPE
Masked grouped-query attention
RMSNorm 1
Token embedding layer
Tokenized text
Sample input text
Pre-Norm
Sebastian Raschka
OLMo 2 7B
Linear output layer
Final RMSNorm
RMSNorm 2
Feed forward
RMSNorm 1
RoPE
Masked multi-head attention
Token embedding layer
Tokenized text
Sample input text
Post-Norm but inside residual
Sebastian Raschka

Perguntas e Discussão

1. Qual é a principal diferença entre um modelo pré-treinado em grande escala e um modelo que recebe apenas fine-tuning em uma tarefa específica?
2. Como a arquitetura Transformer, especialmente o mecanismo de atenção, facilita a criação de modelos com bilhões de parâmetros?
3. Em que situações um método tradicional (por ex., TF-IDF + SVM) pode ser mais vantajoso do que um LLM para tarefas de NLP? Quais são os trade-offs?
4. Quais preocupações éticas você identifica quando um LLM é usado como "juiz" em processos de tomada de decisão (ex.: triagem de currículos, avaliação de crédito)? Como mitigá-las?