Aula 3: Treino e inferência Transformer Decoder-Only

Prof. Marcelo Keese Albertini

Universidade Federal de Uberlândia

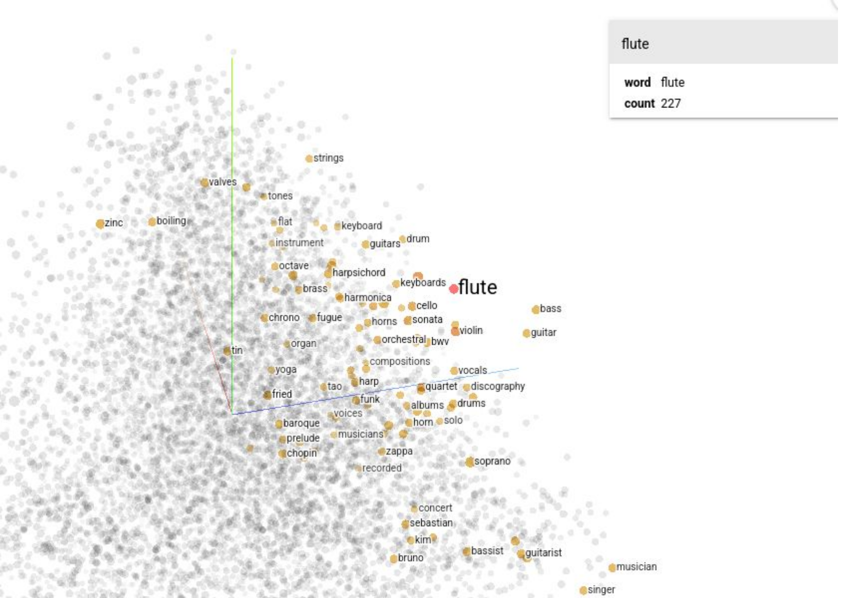
Roteiro

1. Revisão Transformer
2. Transformer Decoder-Only
3. Embedding
4. Codificação de posições
5. Inferência
6. Prática
7. Referências

Transformer

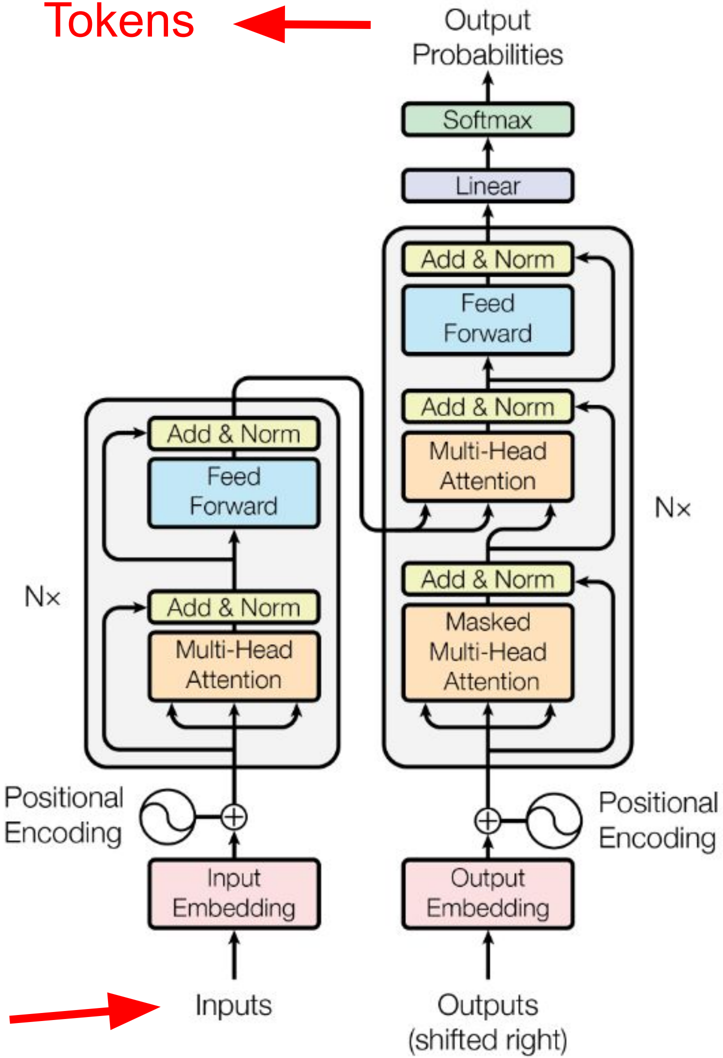
“Attention Is All You Need” (2017)

https://arxiv.org/abs/1706.03762



https://projector.tensorflow.org/

Tokens



Tokens

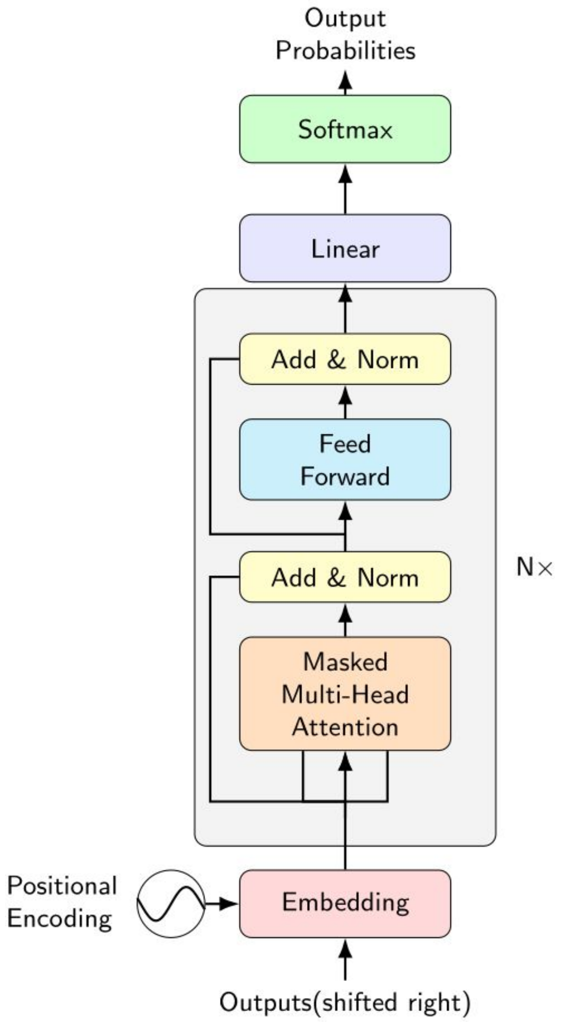
Transformer Decoder-Only

Texto → Tokenizer.encode → Tokens →

Modelo:

1. Embedding + Positional Encoding (PE) →
2. [Multi-head Causal Self-Attention → FFN] × N →
3. Linear →
4. Softmax →

Inferência do próximo token→Tokenizer.decode → Texto



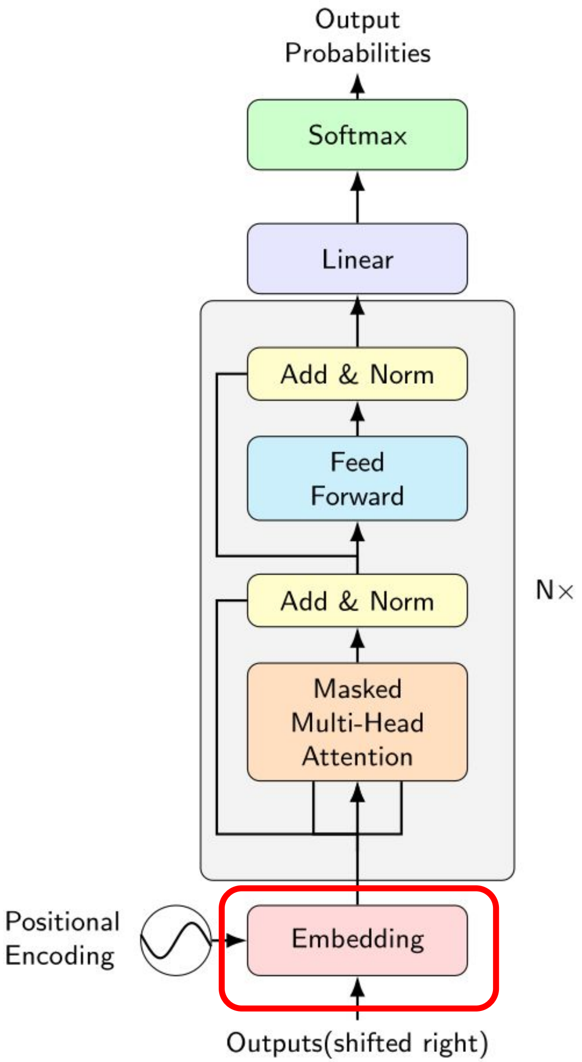
Embedding

```
>>> # an Embedding module containing 10 tensors of size 3
>>> embedding = nn.Embedding(10, 3)
>>> # a batch of 2 samples of 4 indices each
>>> input = torch.LongTensor([[1, 2, 4, 5], [4, 3, 2, 9]])
>>> embedding(input)
tensor([[[-0.0251, -1.6902,  0.7172],
         [-0.6431,  0.0748,  0.6969],
         [ 1.4970,  1.3448, -0.9685],
         [-0.3677, -2.7265, -0.1685]],

        [[ 1.4970,  1.3448, -0.9685],
         [ 0.4362, -0.4004,  0.9400],
         [-0.6431,  0.0748,  0.6969],
         [ 0.9124, -2.3616,  1.1151]]])
```

https://docs.pytorch.org/docs/stable/generated/torch.nn.Embedding.html

https://zackproser.com/demos/embeddings



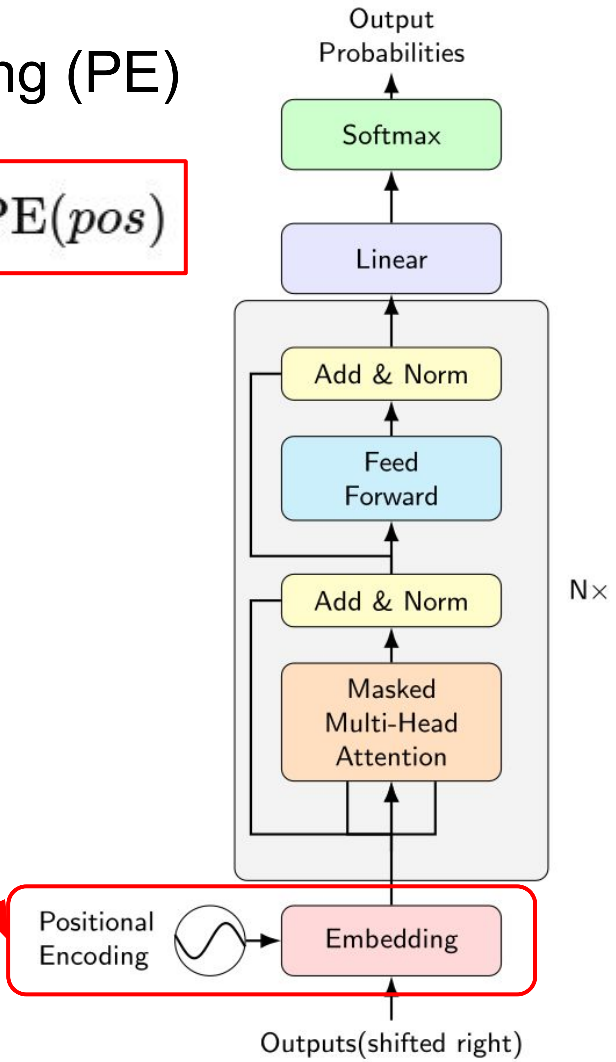
Codificação de posições: Positional Encoding (PE)

$$\boxed{\text{Input} = \text{Embedding}(token) + \text{PE}(pos)}$$

- ***Attention*** não tem como representar a ordem dos tokens por si só !

- **Positional Encoding** demarca a posição dos tokens nas frases ao longo das dimensões do **Embedding.**

https://colab.research.google.com/drive/1gvdxzcMYQwooPaSH4_ifh5Wc8Sn6CXJw?usp=sharing

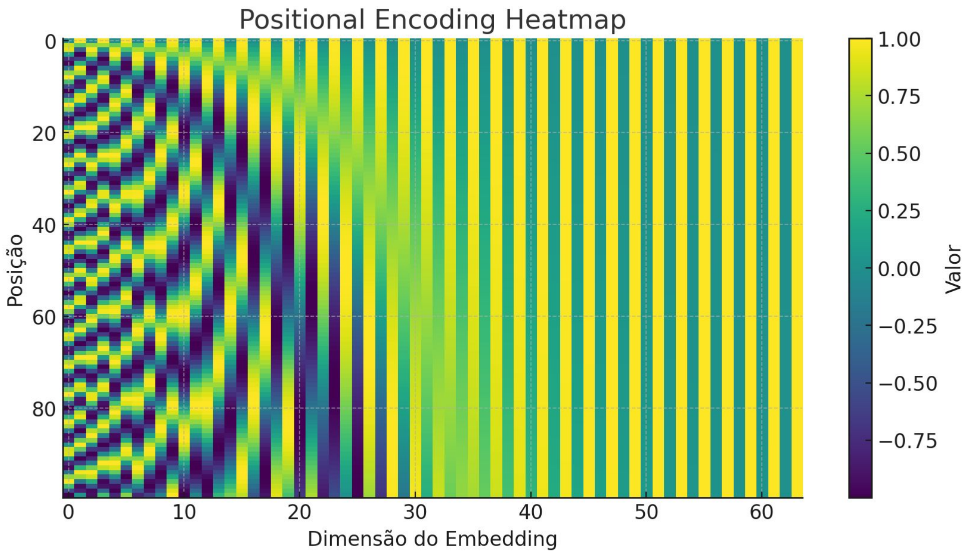


Positional Encoding (PE)

$$\text{Input} = \text{Embedding}(token) + \text{PE}(pos)$$

$$\text{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

$$\text{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

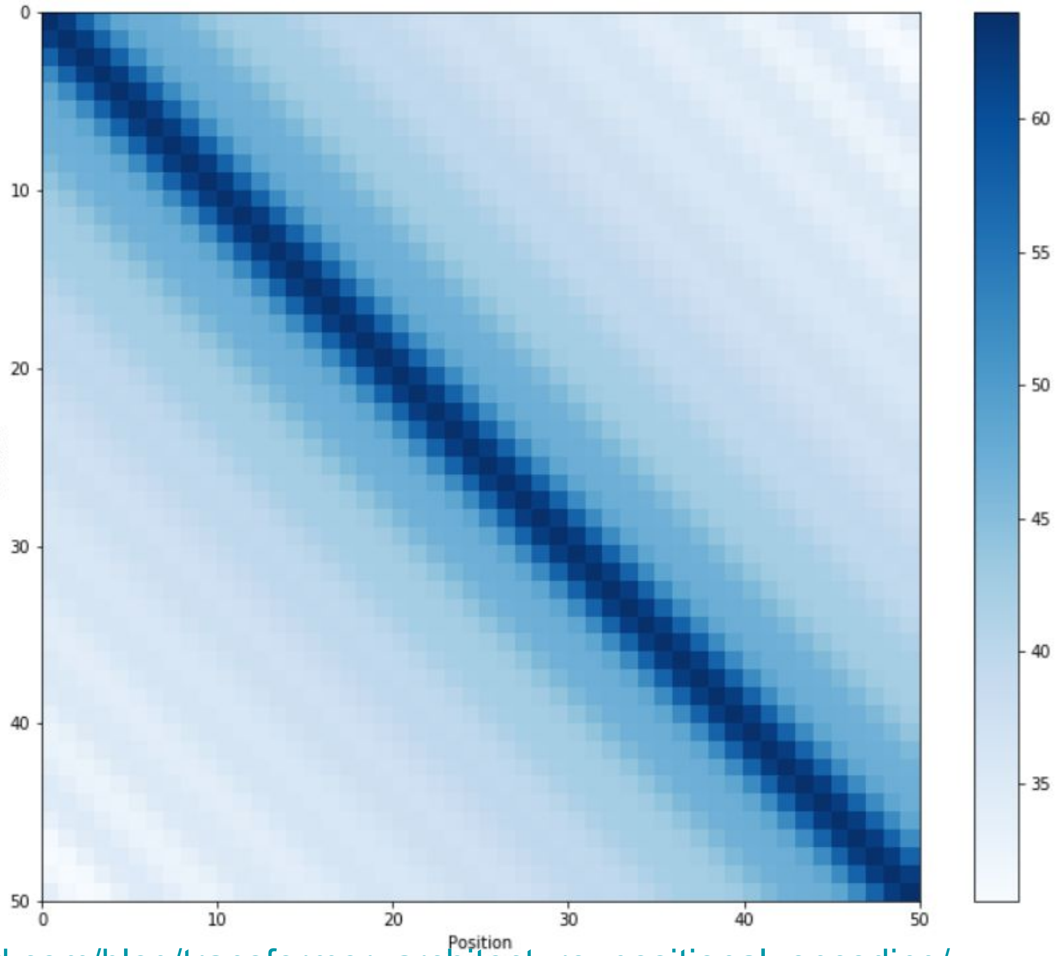


- **pos**: posição do token na sequência de tokens
- **i**: dimensão do embedding
- **d_model**: número total de dimensões no embeddding

```python
def positional_encoding(pos, i, d_model):
    angle = pos / (10000 ** (2 * (i//2) / d_model))
    return np.sin(angle) if i % 2 == 0 else np.cos(angle)
```

Positional Encoding (PE)

- **PE** aumenta similaridade entre time-steps próximos
    - aumento de **similaridade** entre o time 0 e o 50 é **baixo**
    - aumento de **similaridade** entre o time-step 49 e 50 é **alto**



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

Figure 3 - Dot product of position embeddings for all time-steps

https://huggingface.co/learn/llm-course/en/chapter1/8

Inferência: decoding - generate

```python
def generate(prompt, next_token_function, max_new_tokens=20):
   model.eval()
   x = encode(prompt).unsqueeze(0).to(device) # tokenizer

   for _ in range(max_new_tokens):
       logits = model(x)[:, -1, :]  # pega apenas o último passo
       next_token = next_token_function(logits.squeeze(0))

       x = torch.cat([x, next_token.unsqueeze(0)], dim=1)

       if next_token.item() == tokenizer.token_to_id("[EOS]"):
           break

   return decode(x[0])
```

```python
def max_prob_sampling(logits):
   next_token = logits.argmax(dim=-1)
   return next_token.unsqueeze(0)
```

Inferência: decoding - estratégias de amostra de token

```
def sampling(logits, top_k=100, top_p=0.9, temperature=1.0):
```

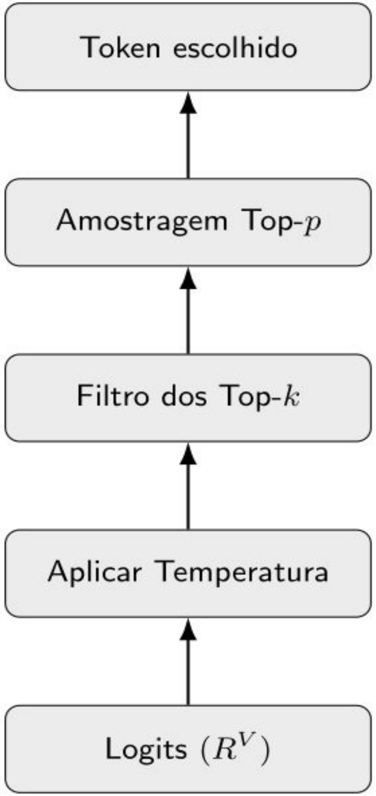
1. **Logits**: probabilidades dos tokens produzidas pelo modelo

2. **Aplicar Temperatura**: modificador de probabilidades.

   Se > 1, aumenta chance de tokens menos prováveis e mais **criativo**. Se < 1, deixa mais **rígido**.

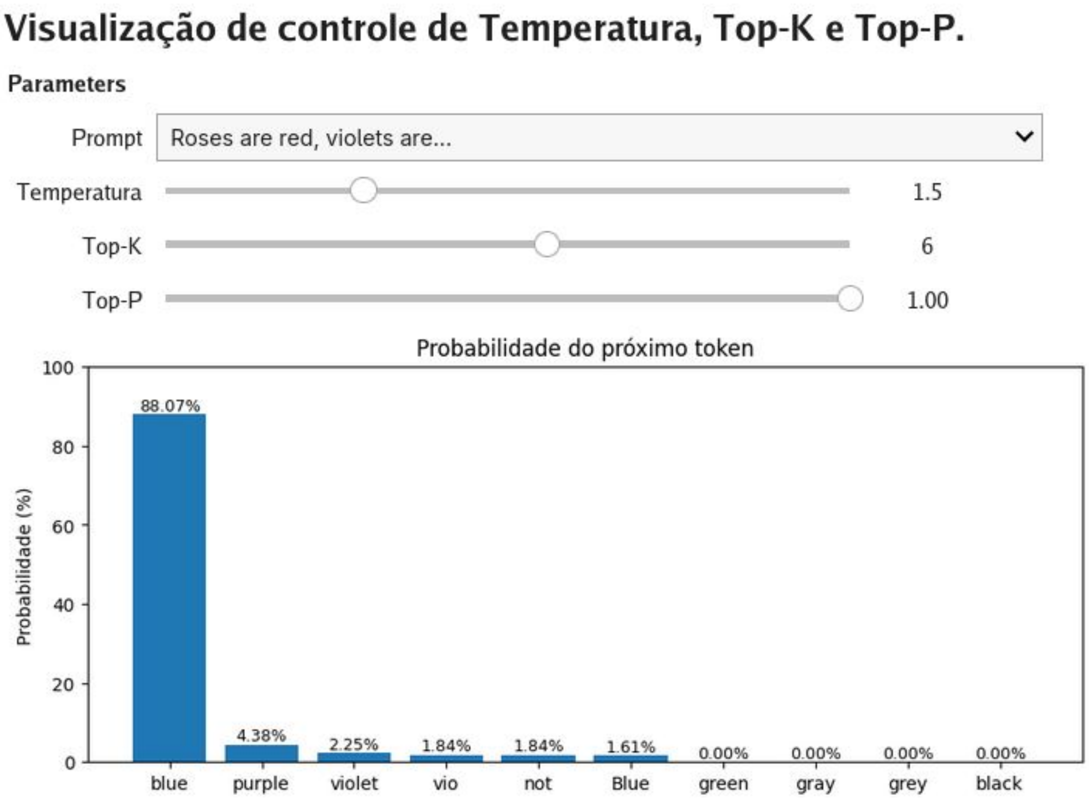
3. **Filtro dos Top-k**: Somente os tokens com as **k** maiores probabilidades são considerados

4. **Amostragem Top-p**: exclui os tokens cuja probabilidade acumulada não inclui p%.



https://huggingface.co/learn/llm-course/en/chapter1/8

Controles: temperatura, top-k, top-P



https://colab.research.google.com/drive/16g7RjiELBvtveKQ7hHxaKgZVAHwDbAH3?usp=sharing

https://andreban.github.io/temperature-topk-visualizer

Referências

- Como inferência de LLMs funciona
  - https://bentoml.com/llm/llm-inference-basics/how-does-llm-inference-work
- Deep dive into Text Generation Inference with LLMs
  - https://huggingface.co/learn/llm-course/en/chapter1/8
- CSCI1470 - Deep learning
  - https://deep-learning-s25.vercel.app/
- “The Annotated Transformer” (Harvard NLP)
  - https://nlp.seas.harvard.edu/2018/04/03/attention.html
- “Transformers from Scratch” — Peter Bloem
  - https://peterbloem.nl/blog/transformers
- Positional encoding
  - https://kazemnejad.com/blog/transformer_architecture_positional_encoding/