Arquitetura Transformer e seus derivados: BERT e GPT

Redes Neurais e Aprendizado Profundo

Moacir A. Ponti

www.icmc.usp.br/~moacir — moacir@icmc.usp.br

São Carlos-SP/Brasil

Agenda

Transformer Network

BERT: pré-treinamento de encoders de transformers bidirecionais

GPT: generative pretrained transformer

Agenda

Transformer Network

BERT: pré-treinamento de encoders de transformers bidirecionais

GPT: generative pretrained transformer

RNNs vs Transformer Networks

RNNs

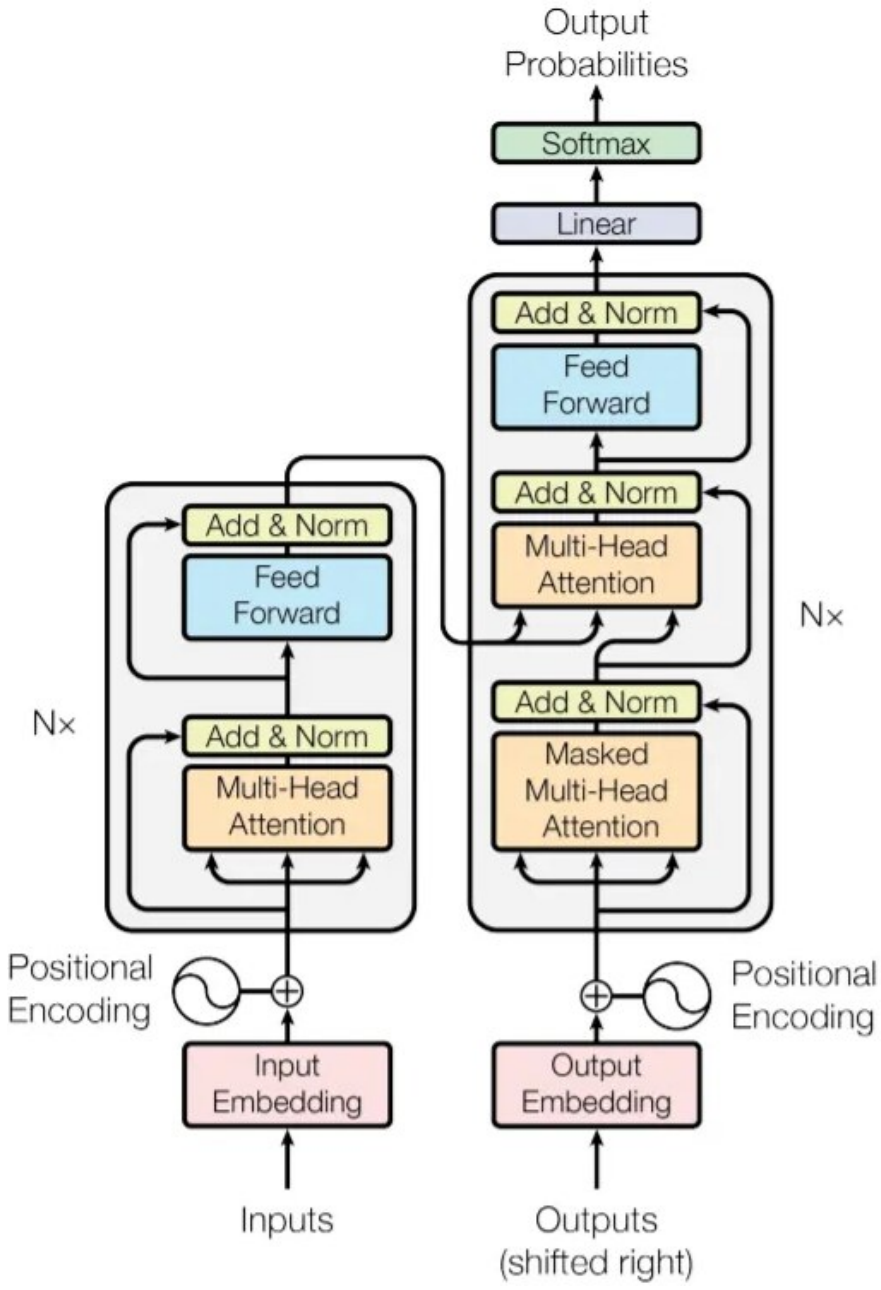
- podem não funcionar com dependências longas
- recorrência dificulta computação paralela (pontos da sequência não podem ser processados em paralelo)
- podem sofrer com explosão ou desaparecimento de gradiente

Transformer Networks

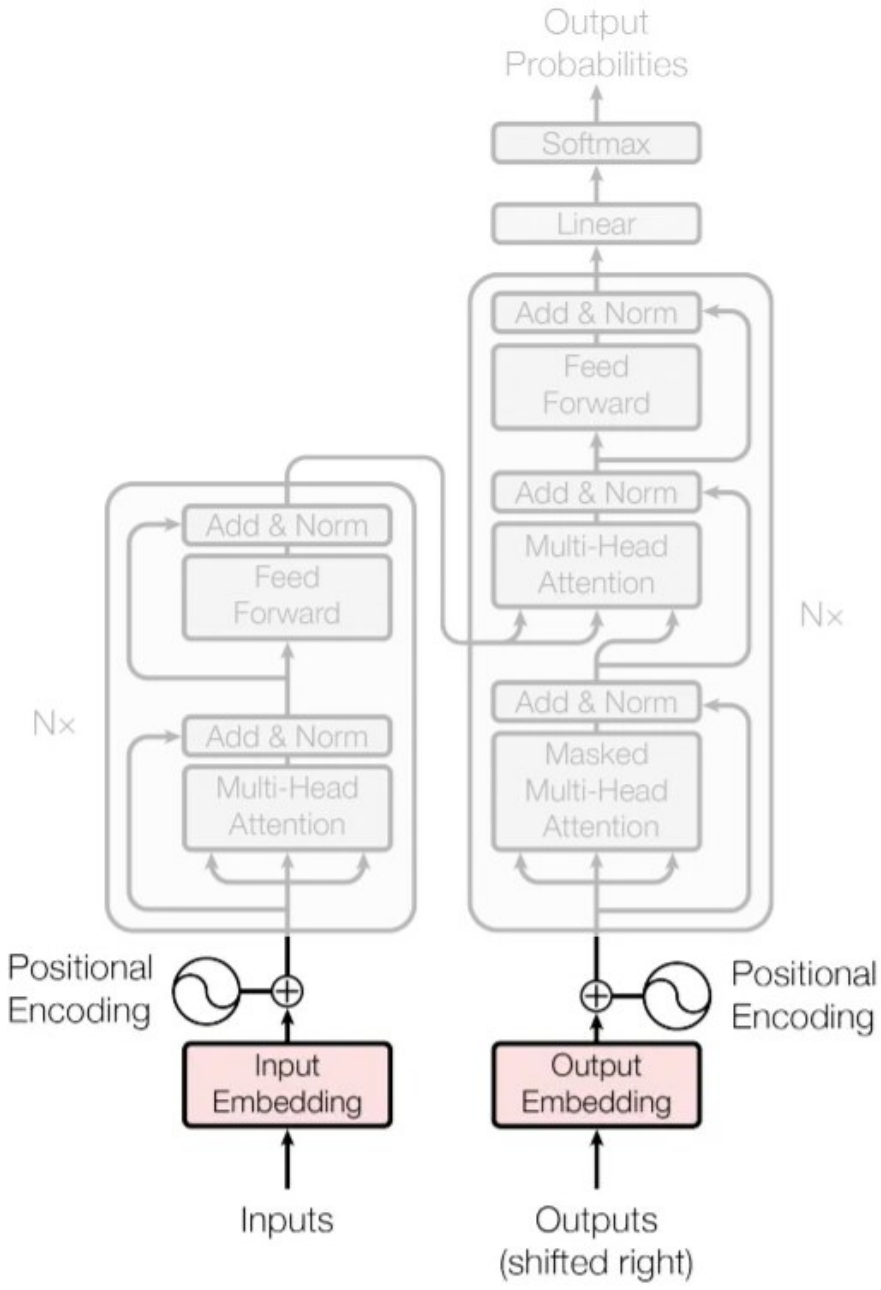
- não possui recorrência, apenas atenção
- facilita capturar dependências longas
- construído para ser sequence-to-sequence

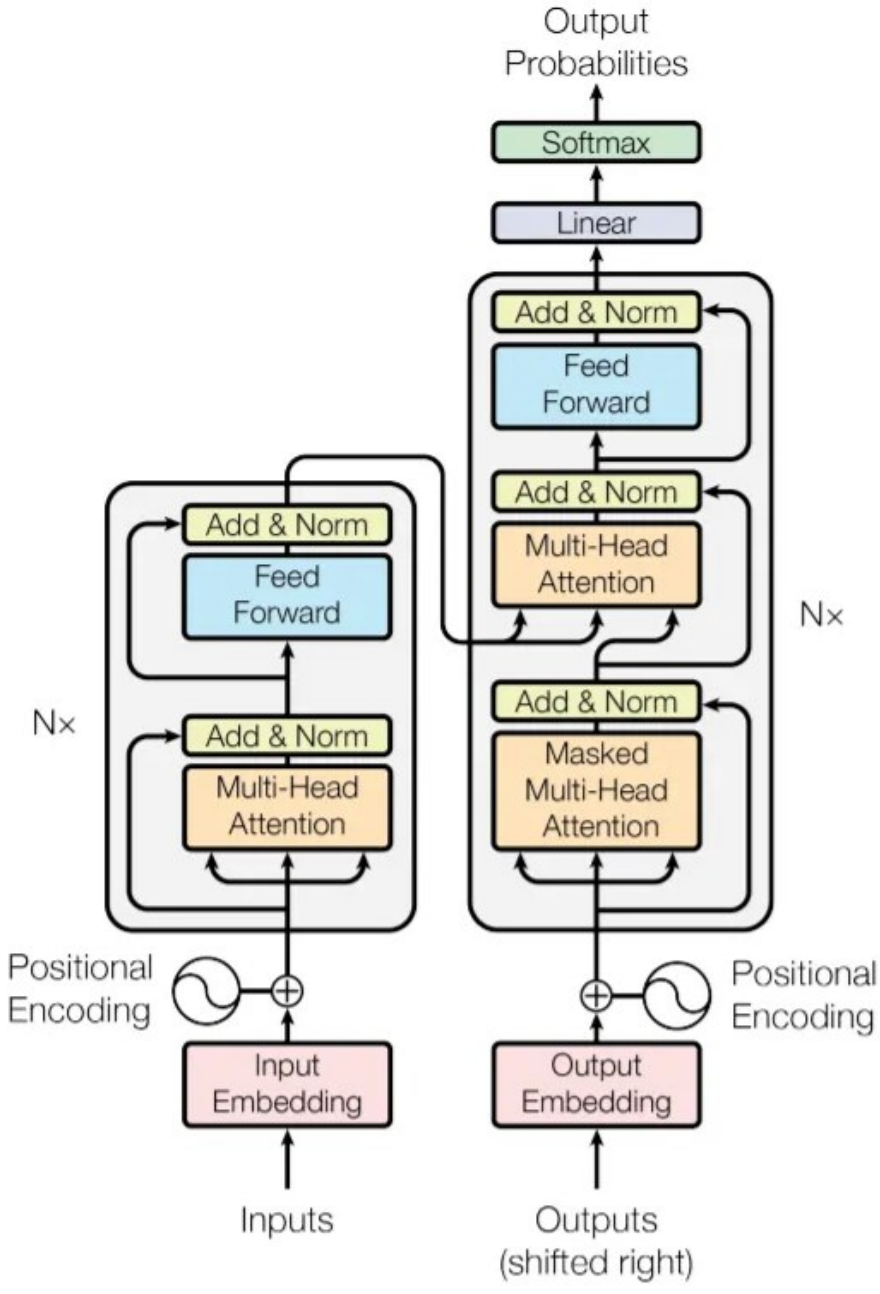
Artigo: "Attention is all you need" Vaswani, NeurIPS 2017.

Arquitetura Transformer

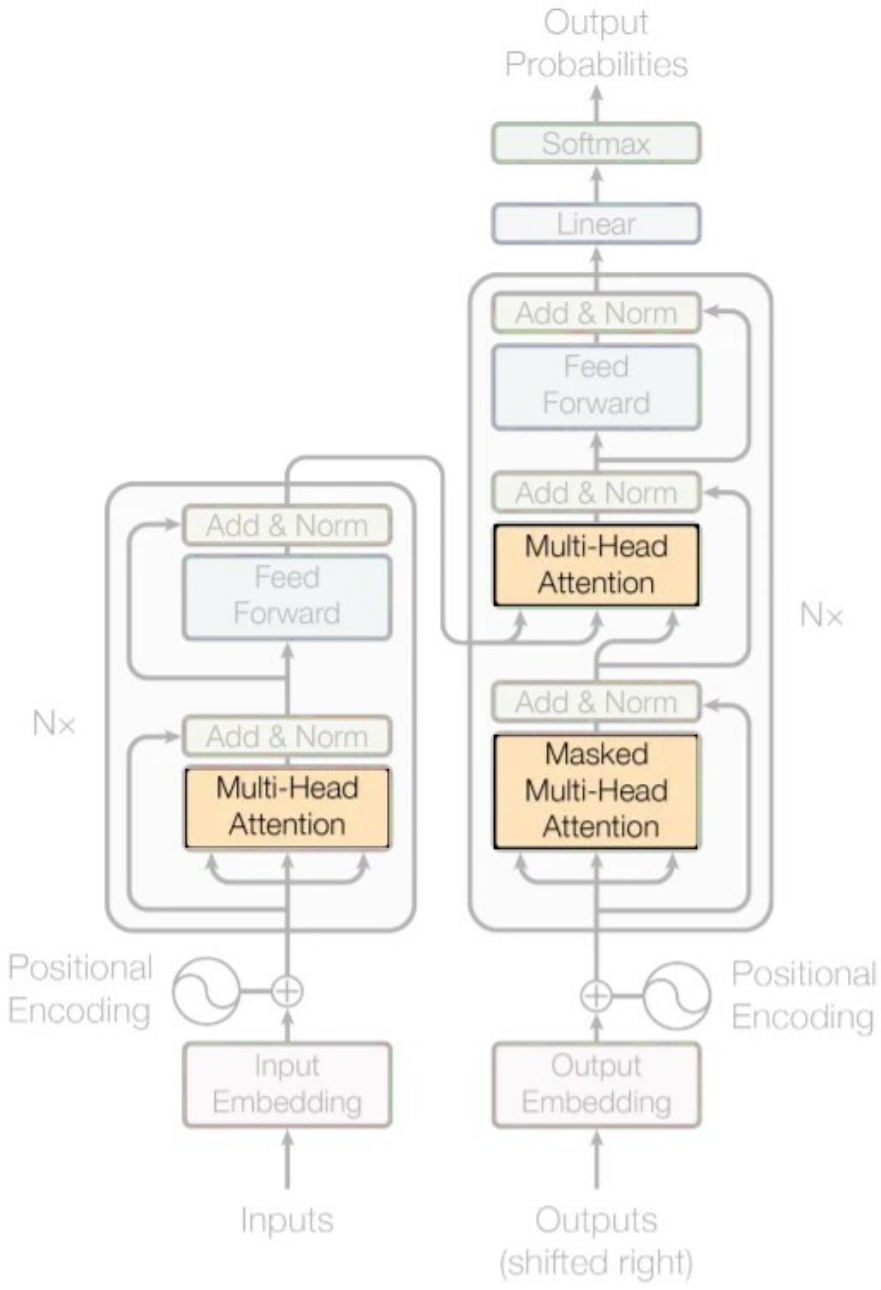


Positional Encoding: adiciona informação de posicionamento

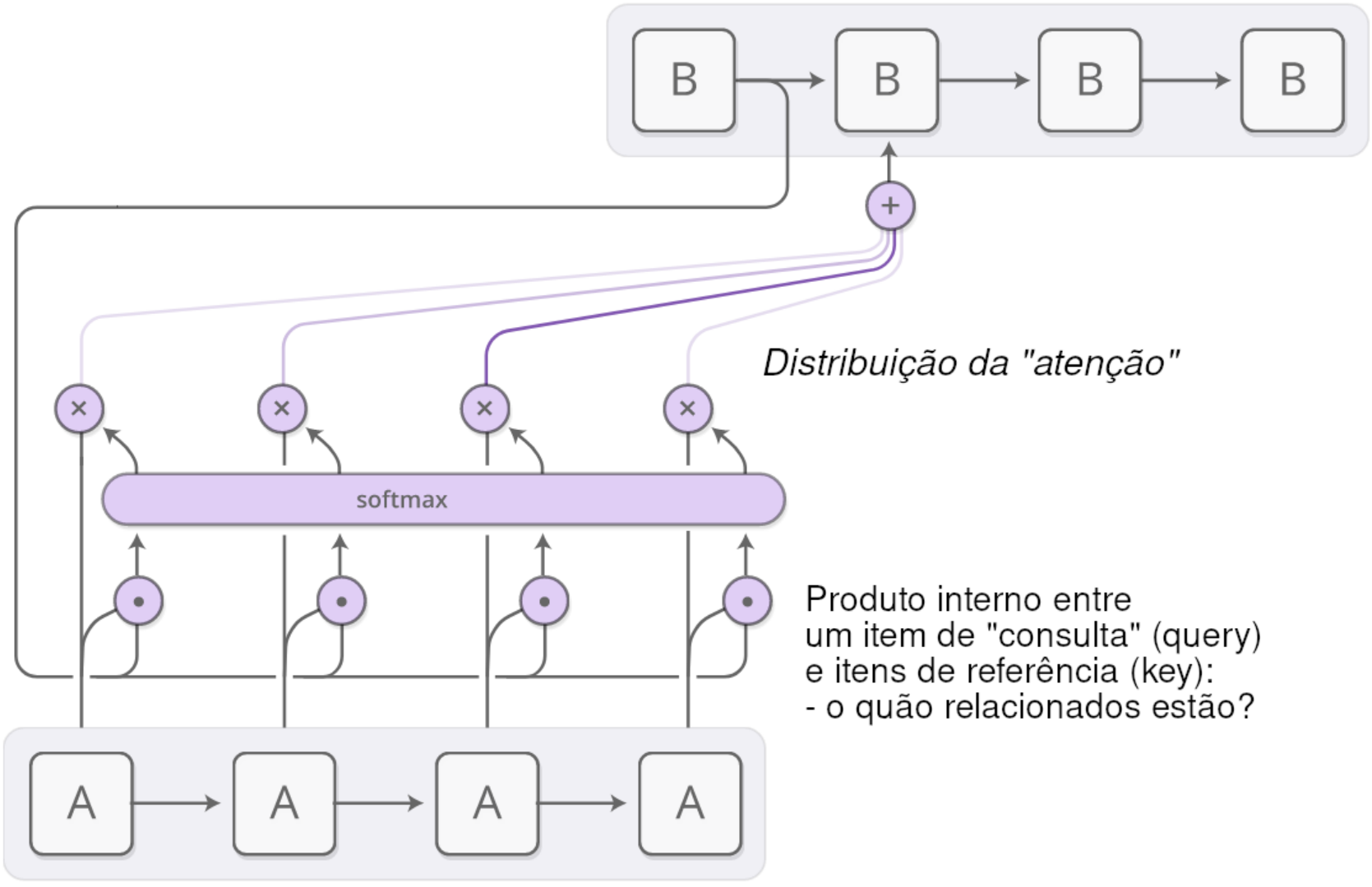



Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
N×
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
N×
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Arquitetura Transformer: multi-atenção



Distribuição da atenção com relação ao estado atual



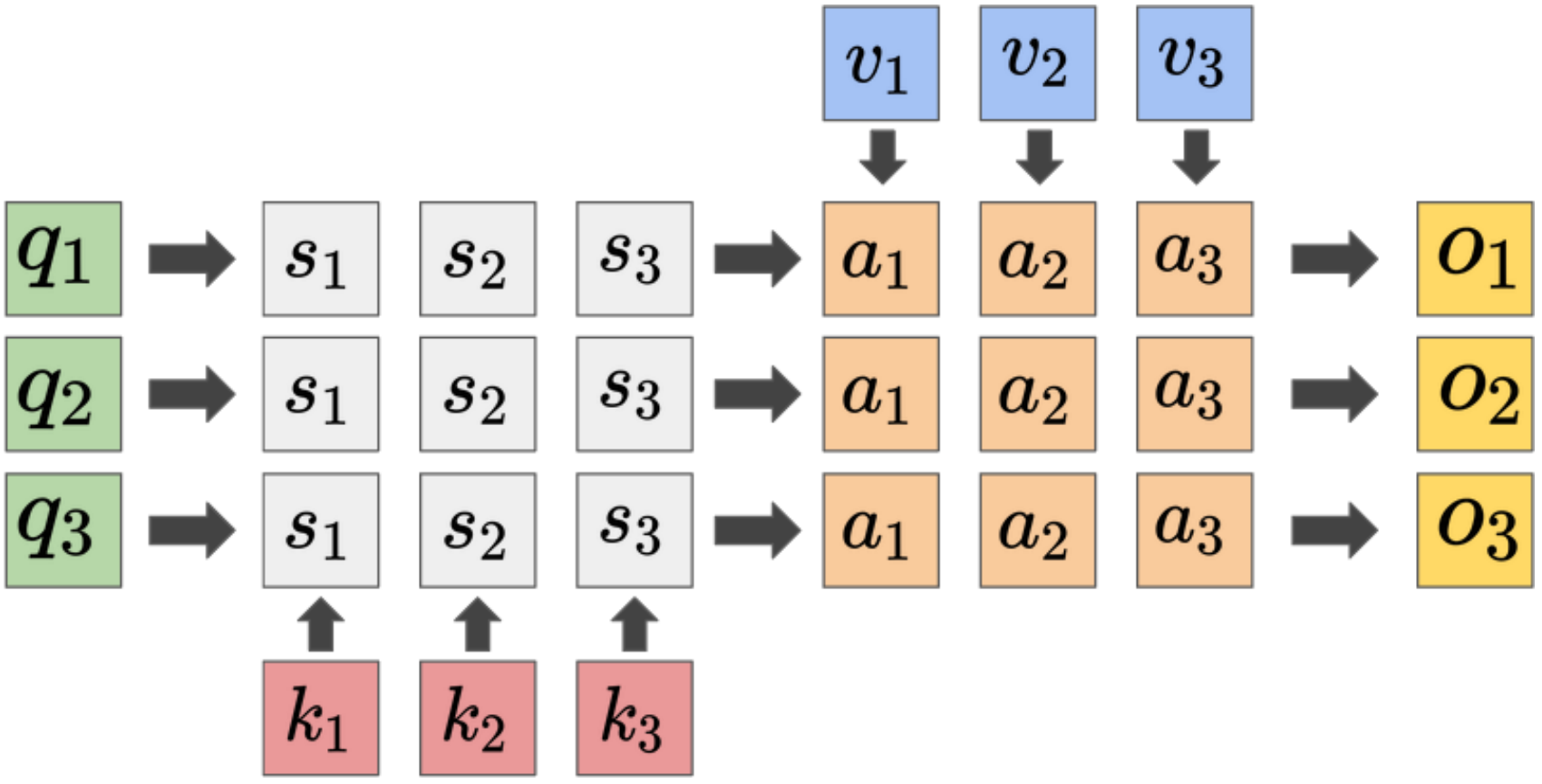
Mecanismo de atenção em Transformer Networks

- Recuperar um valor v_i para uma consulta/query q baseada numa chave/key k_i

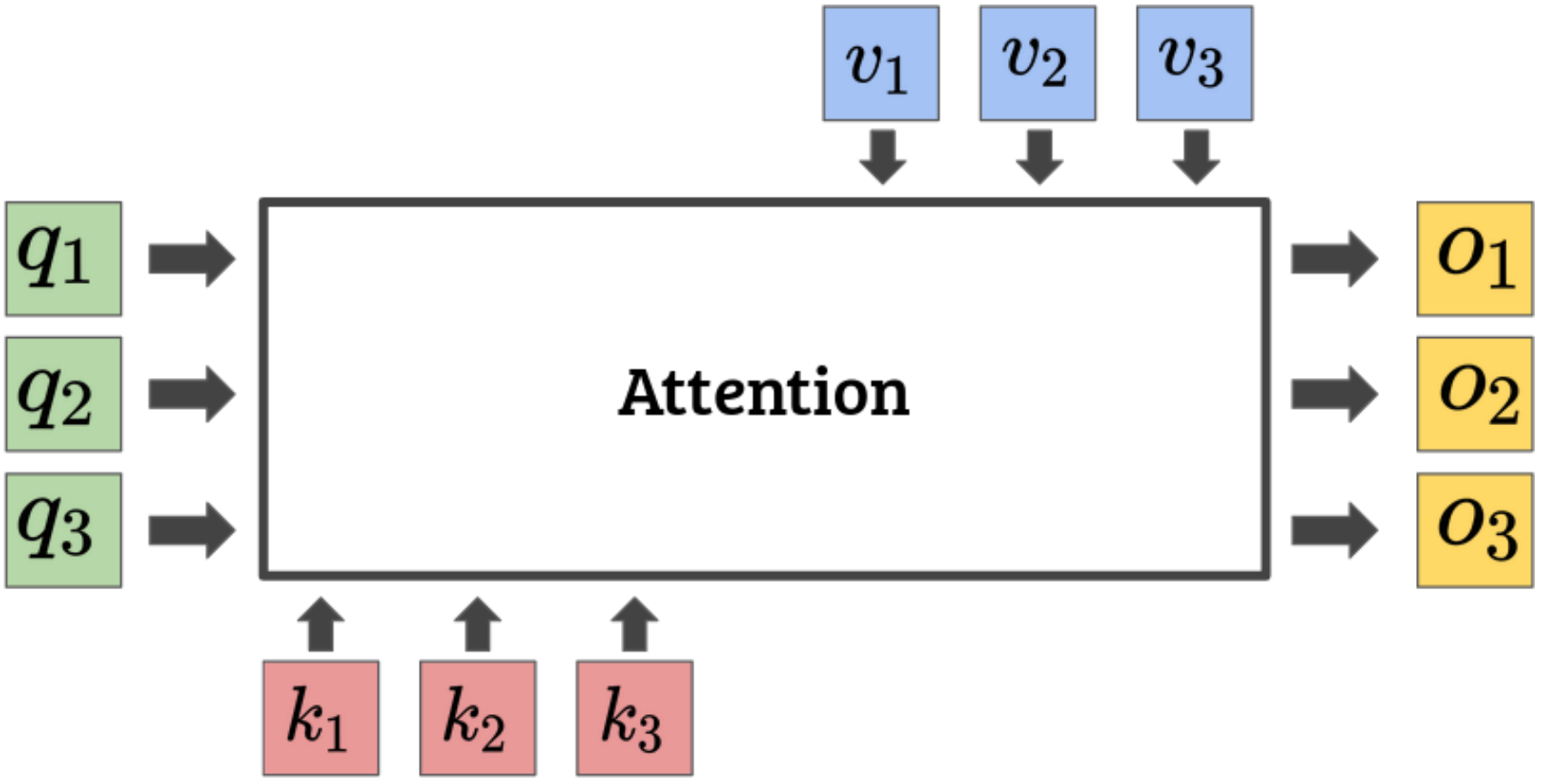
$$Attention(q, \mathrm{k}, \mathrm{v}) = \sum_i similarity(q, k_i) \times v_i$$

- A **similaridade** entre uma consulta e todas as chaves, ponderadas pelos valores
- Somar ao longo de todas as chaves/valores, produz uma **distribuição** de pesos relacionando consulta e todos os valores

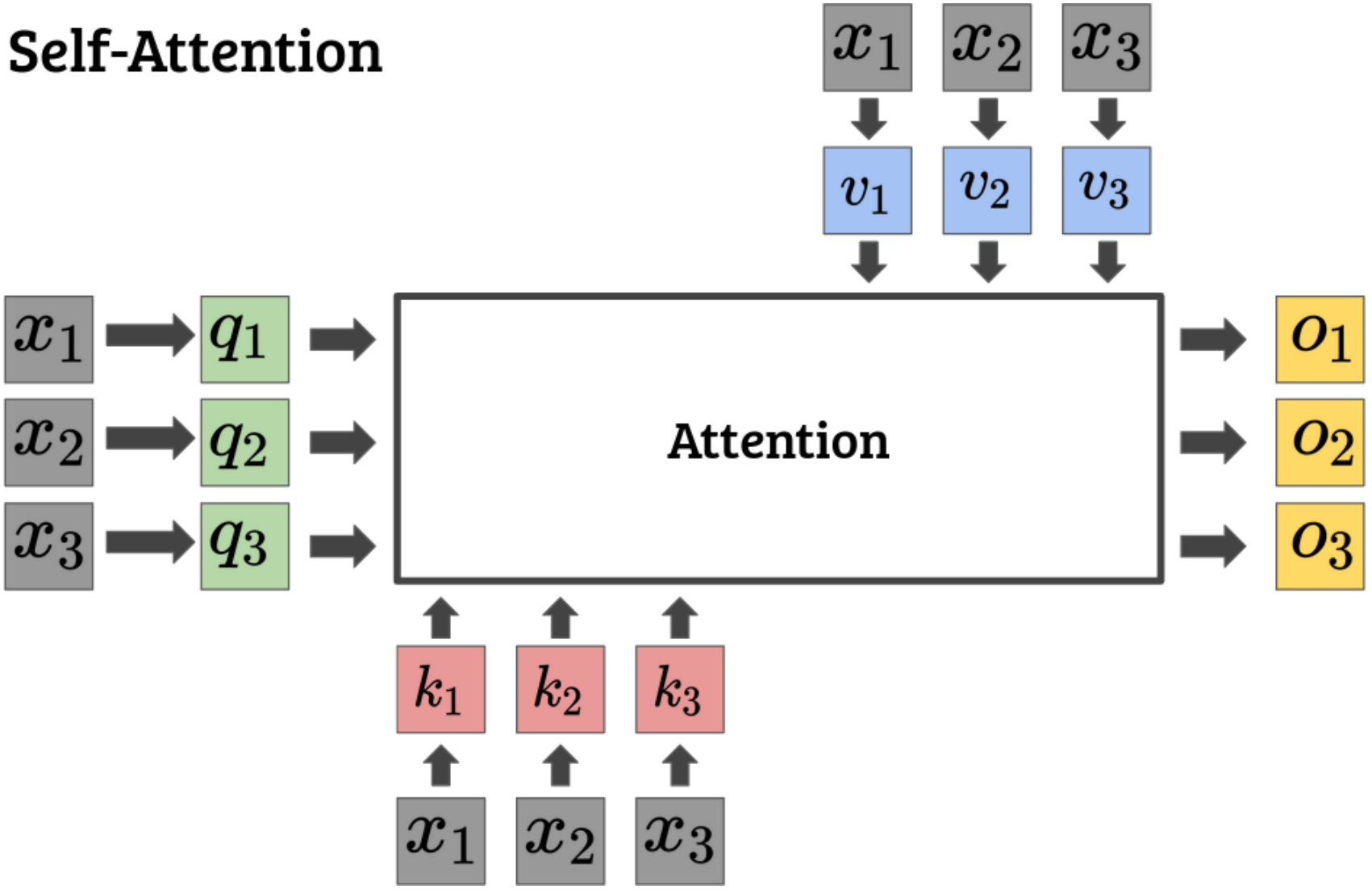
Atenção


v_1 v_2 v_3
q_1 s_1 s_2 s_3 a_1 a_2 a_3 o_1
q_2 s_1 s_2 s_3 a_1 a_2 a_3 o_2
q_3 s_1 s_2 s_3 a_1 a_2 a_3 o_3
k_1 k_2 k_3

Atenção



Atenção (auto-atenção)



Para texto: representações para cada palavra

1. Word embeddings: w
2. Word embeddings + Positional encoding: $\mathrm{w} + \mathrm{p} = \mathrm{e}$
3. Embeddings contextuais com base na saída da atenção

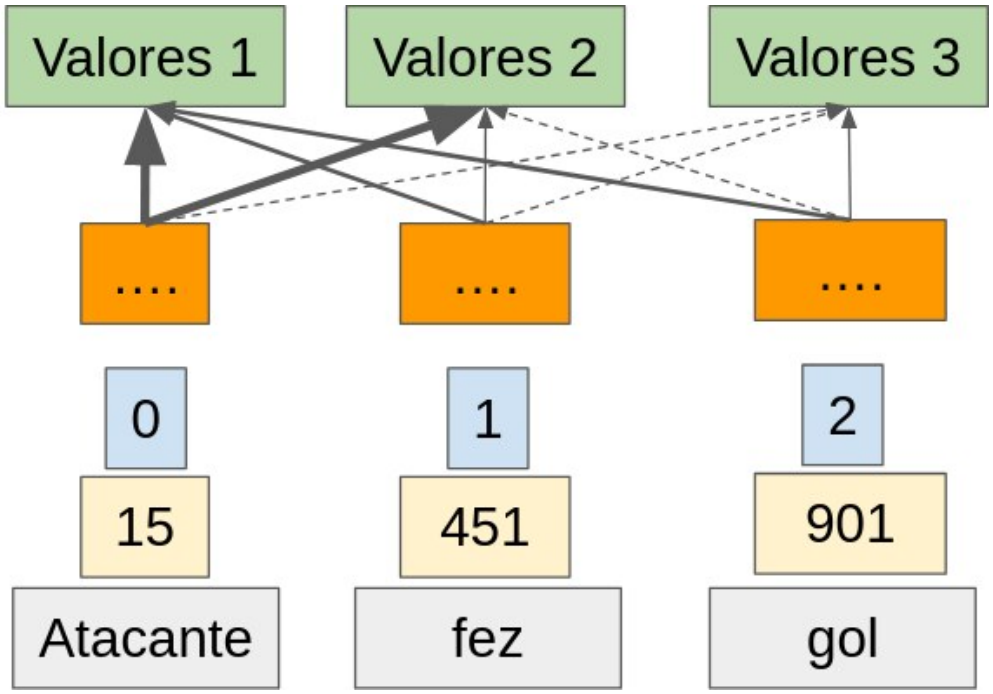
Encoder

Contextual Embeddings

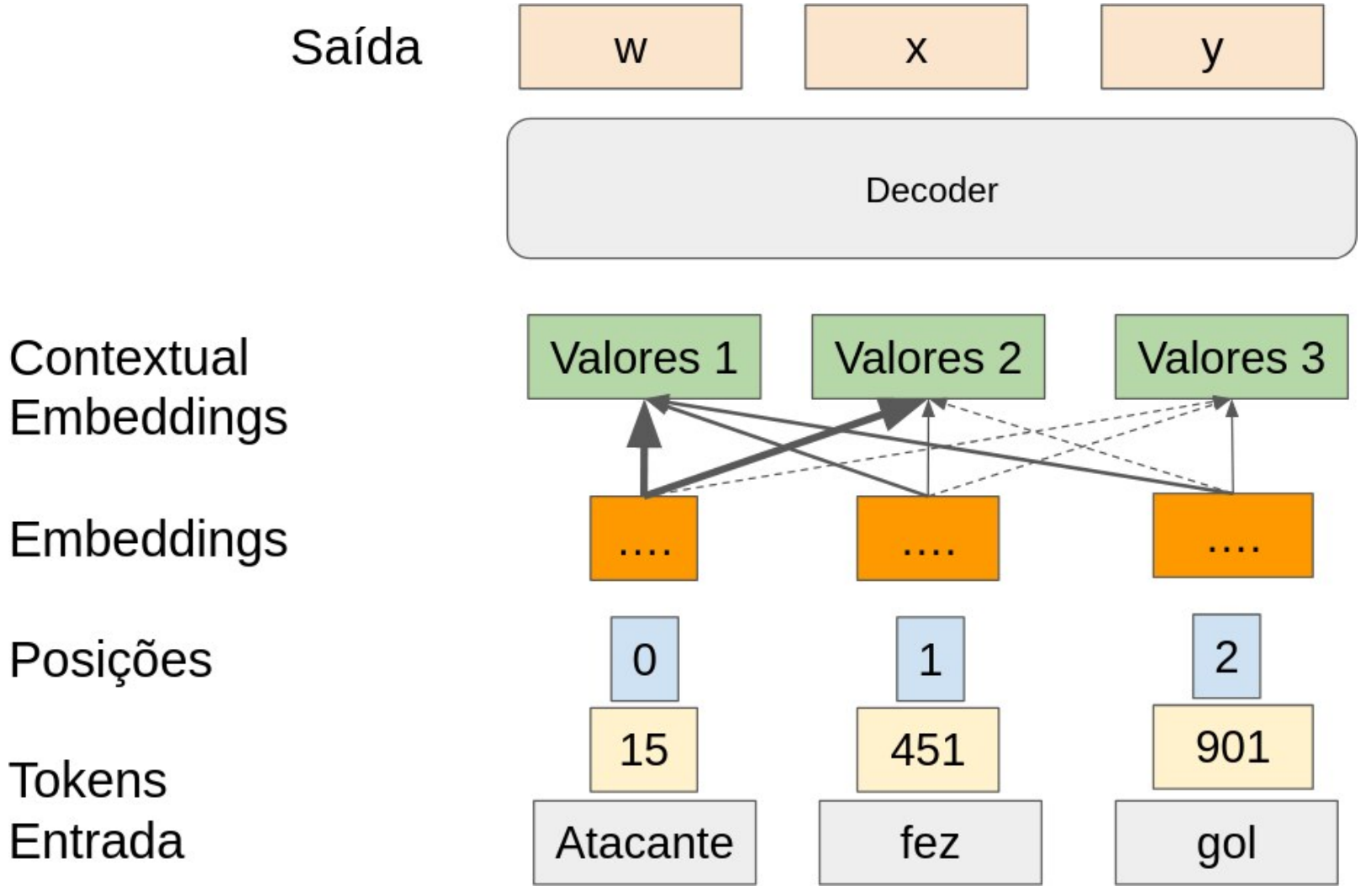
Embeddings

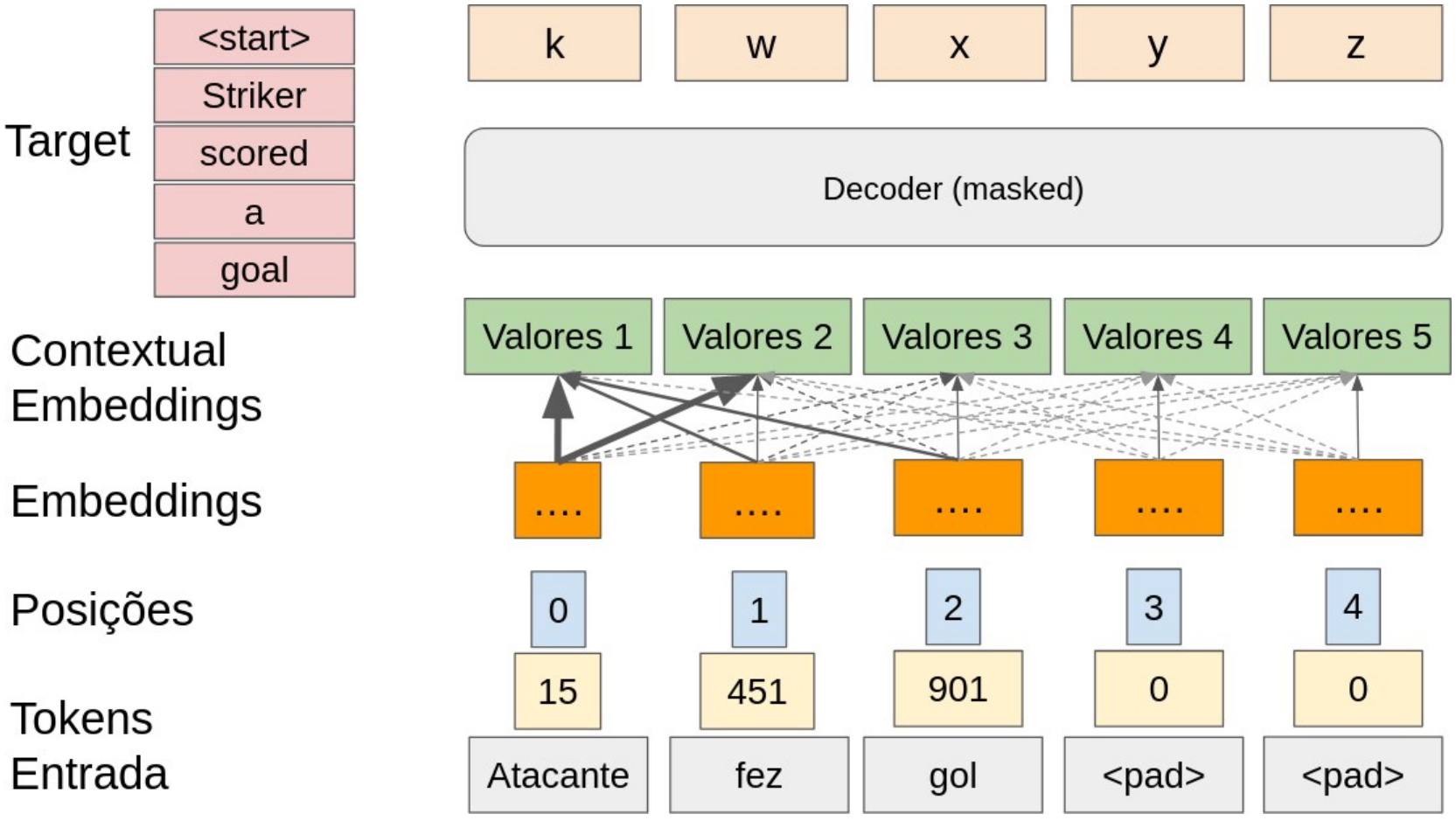
Posições

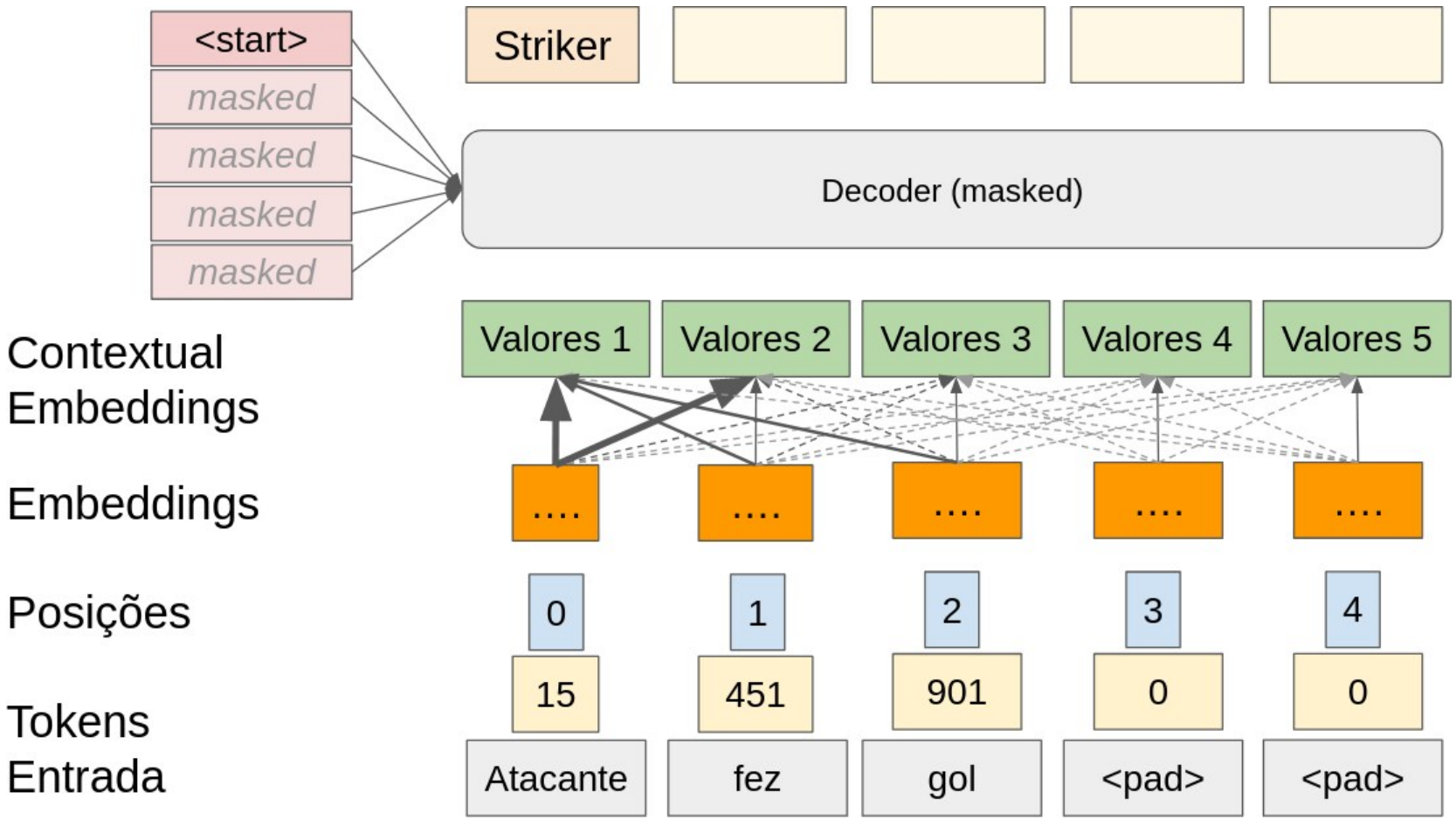
Tokens Entrada

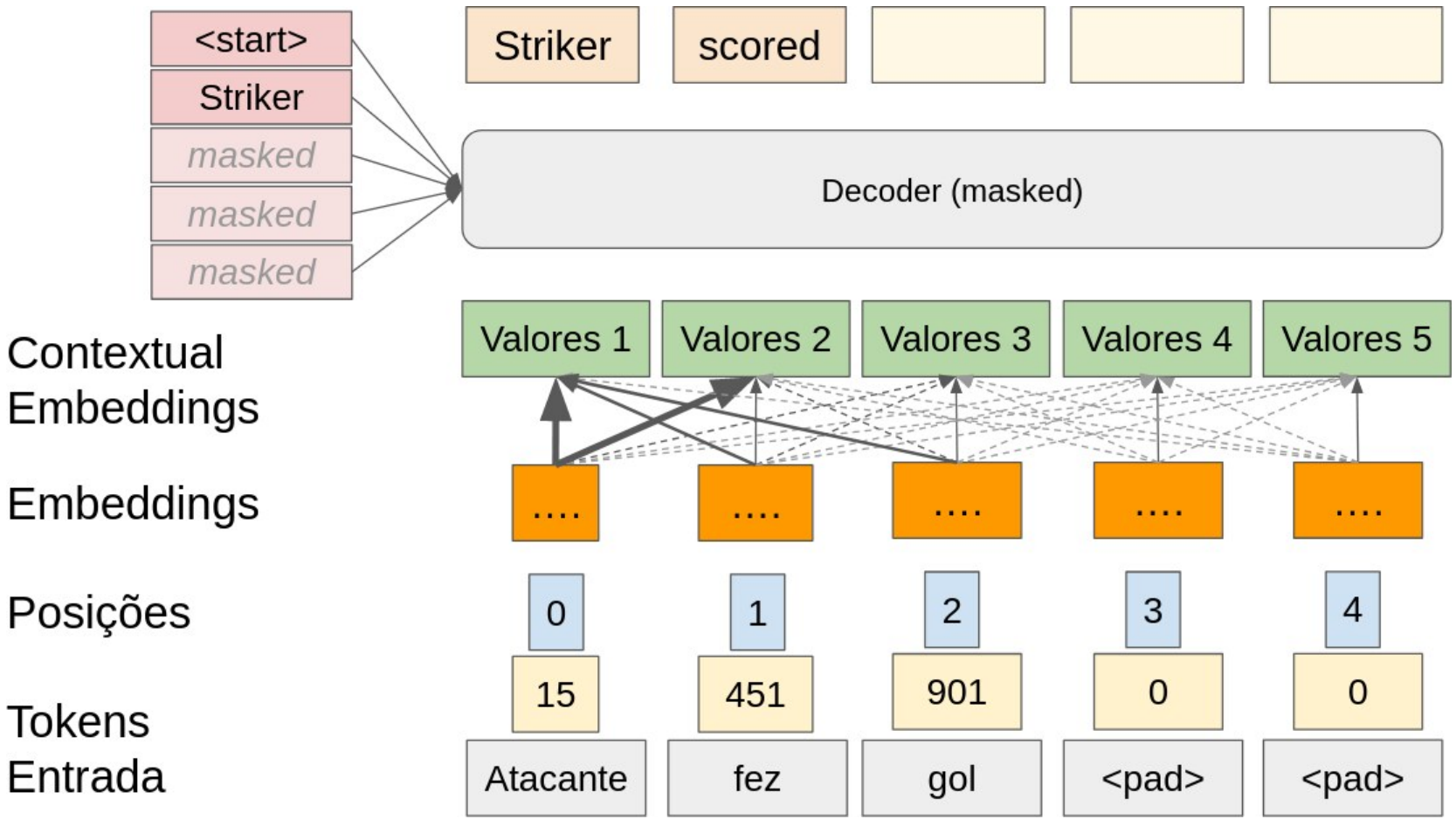


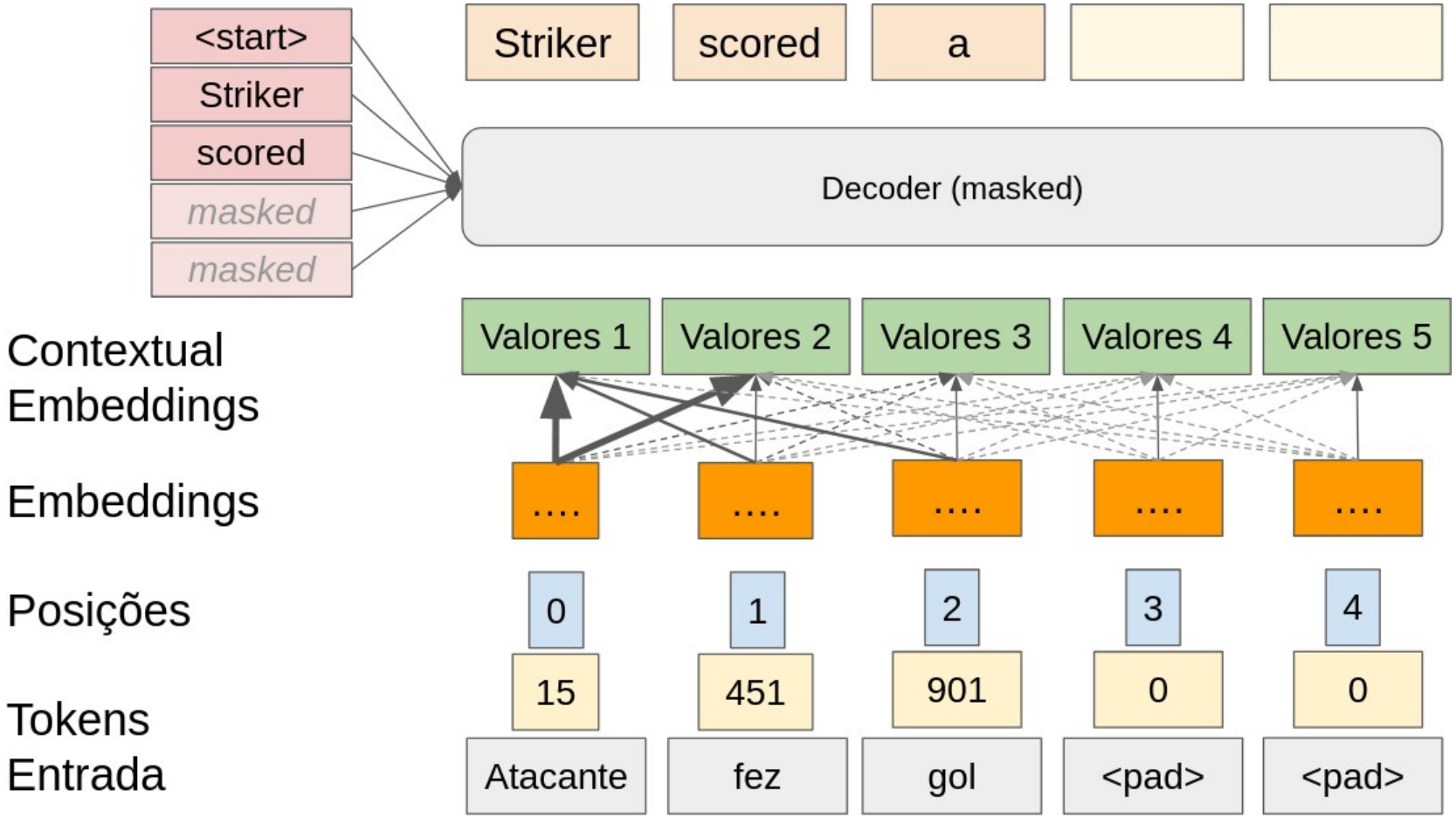
Encoder-Decoder

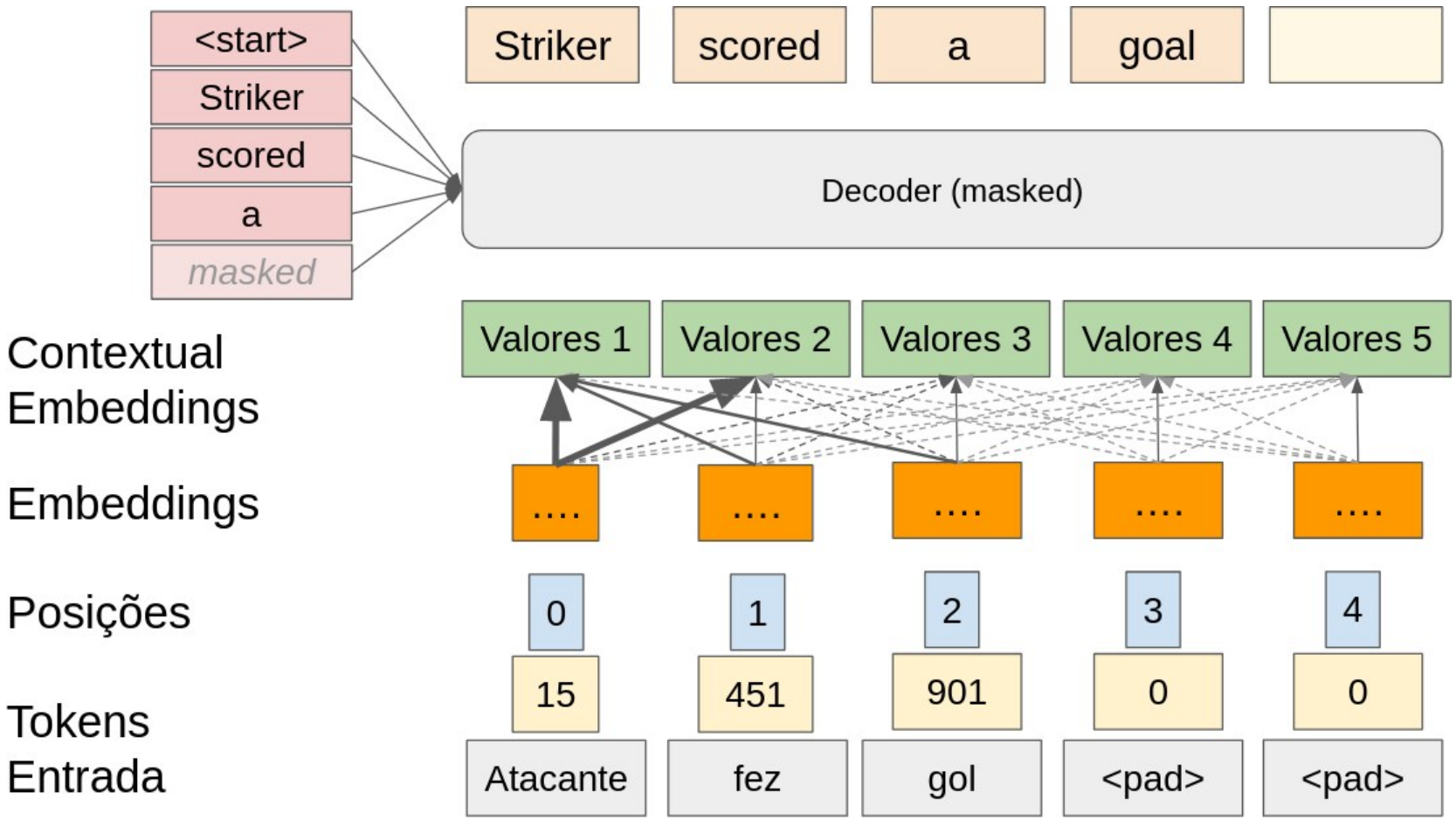


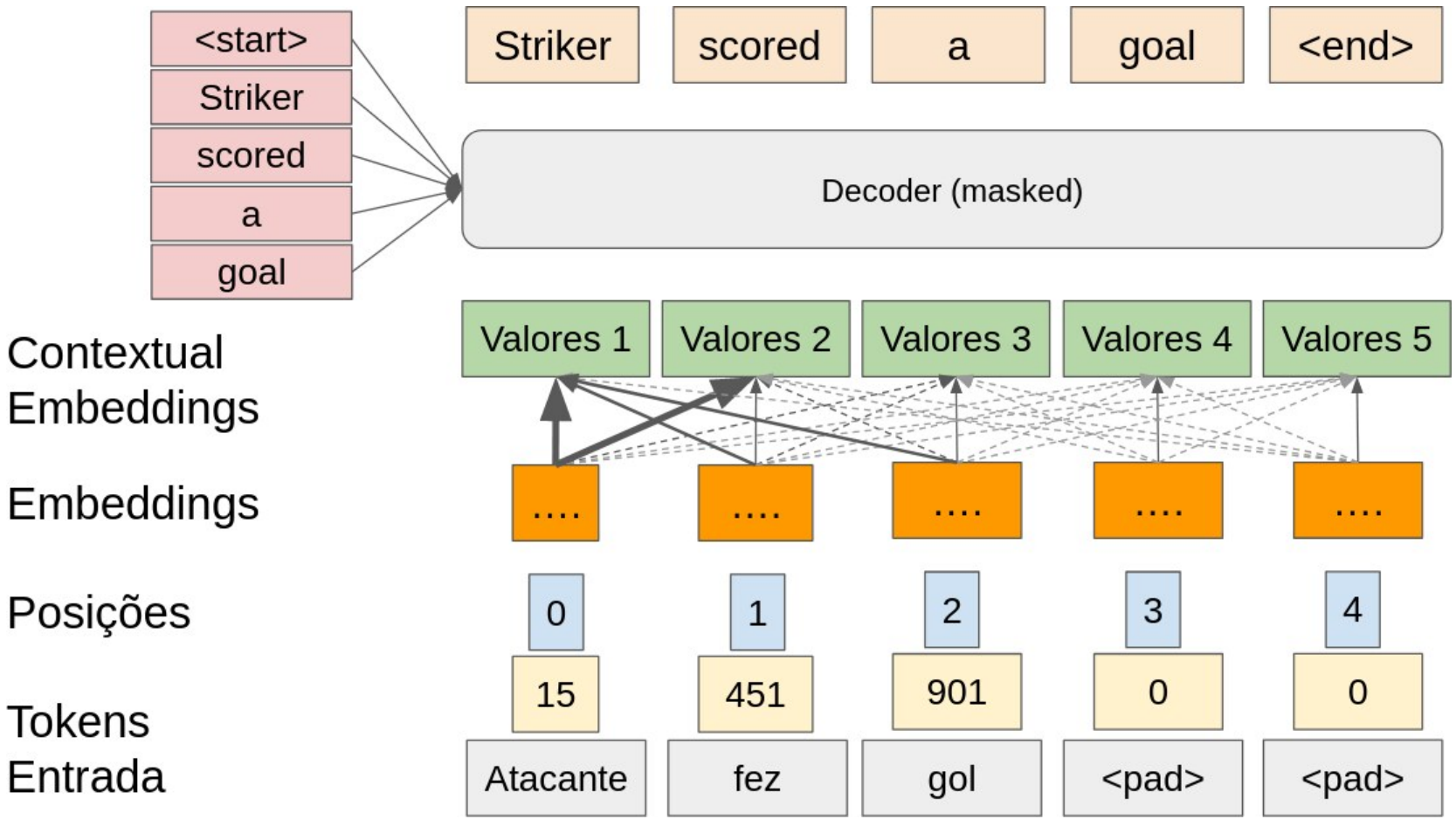

<start>
Striker
scored
a
goal
Target
k
w
x
y
z
Decoder (masked)
Valores 1
Valores 2
Valores 3
Valores 4
Valores 5
Contextual Embeddings
....
....
....
....
....
Embeddings
0
1
2
3
4
Posições
15
451
901
0
0
Tokens Entrada
Atacante
fez
gol
<pad>
<pad>


<start>
masked
masked
masked
masked
Striker
Decoder (masked)
Valores 1
Valores 2
Valores 3
Valores 4
Valores 5
....
....
....
....
....
0
1
2
3
4
15
451
901
0
0
Atacante
fez
gol
<pad>
<pad>
Contextual Embeddings
Embeddings
Posições
Tokens Entrada

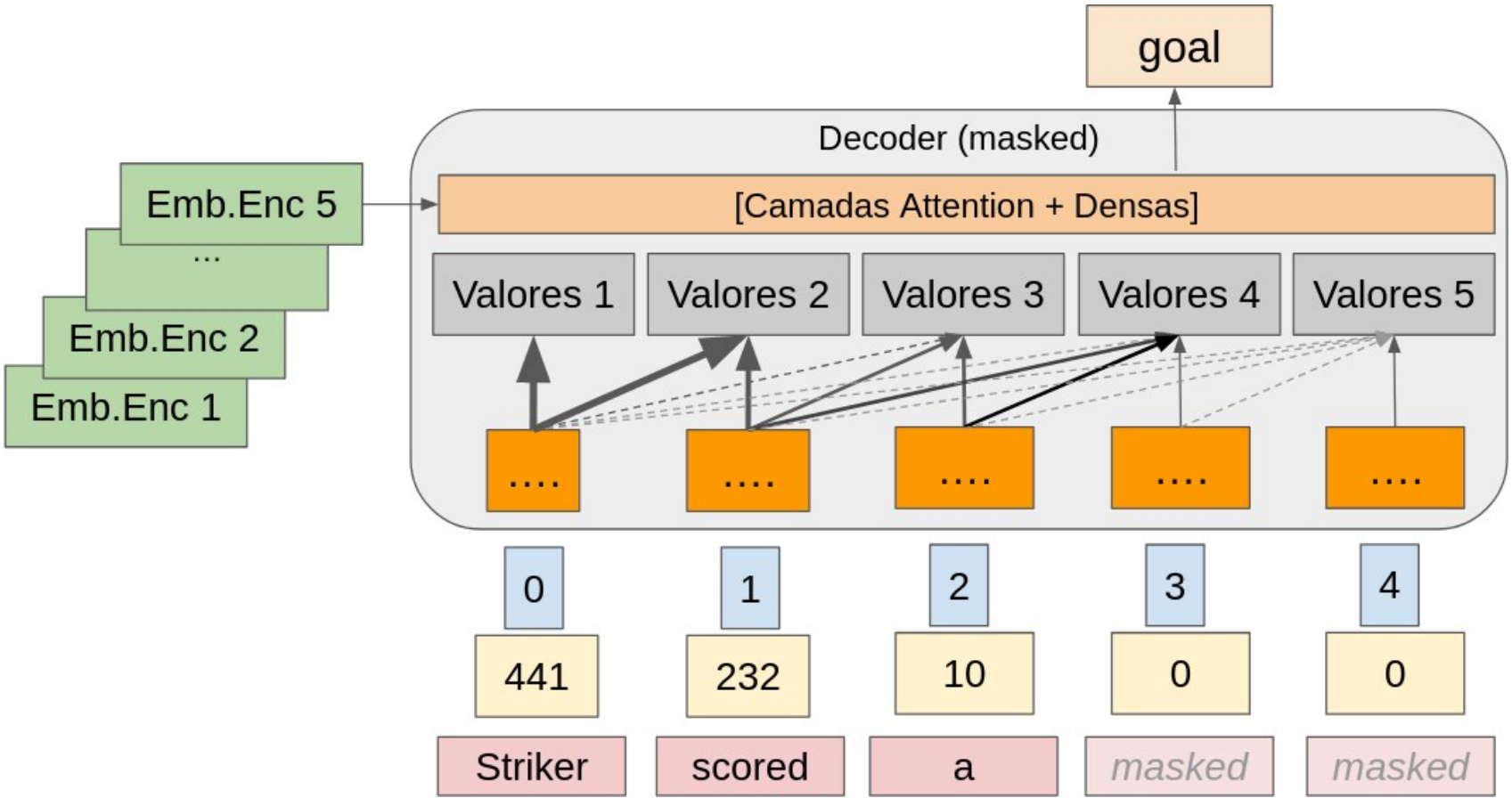

<start>
Striker
masked
masked
masked
Striker
scored
Decoder (masked)
Valores 1
Valores 2
Valores 3
Valores 4
Valores 5
Contextual Embeddings
Embeddings
Posições
Tokens Entrada
....
....
....
....
....
0
1
2
3
4
15
451
901
0
0
Atacante
fez
gol
<pad>
<pad>


<start>
Striker
scored
masked
masked
Striker
scored
a
Decoder (masked)
Valores 1
Valores 2
Valores 3
Valores 4
Valores 5
Contextual Embeddings
Embeddings
Posições
Tokens Entrada
....
....
....
....
....
0
1
2
3
4
15
451
901
0
0
Atacante
fez
gol
<pad>
<pad>


<start>
Striker
scored
a
masked
Striker
scored
a
goal
Decoder (masked)
Valores 1
Valores 2
Valores 3
Valores 4
Valores 5
....
....
....
....
....
0
1
2
3
4
15
451
901
0
0
Atacante
fez
gol
<pad>
<pad>
Contextual Embeddings
Embeddings
Posições
Tokens Entrada


<start>
Striker
scored
a
goal
Striker
scored
a
goal
<end>
Decoder (masked)
Valores 1
Valores 2
Valores 3
Valores 4
Valores 5
Contextual Embeddings
Embeddings
Posições
Tokens Entrada
....
....
....
....
....
0
1
2
3
4
15
451
901
0
0
Atacante
fez
gol
<pad>
<pad>

Visão do Decoder



Attention e Masked Attention

Attention

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

d_k dimensões da chave

Masked Attention

- o decoder deve anular atenção com relação a palavras de entrada futuras, senão não há aprendizado

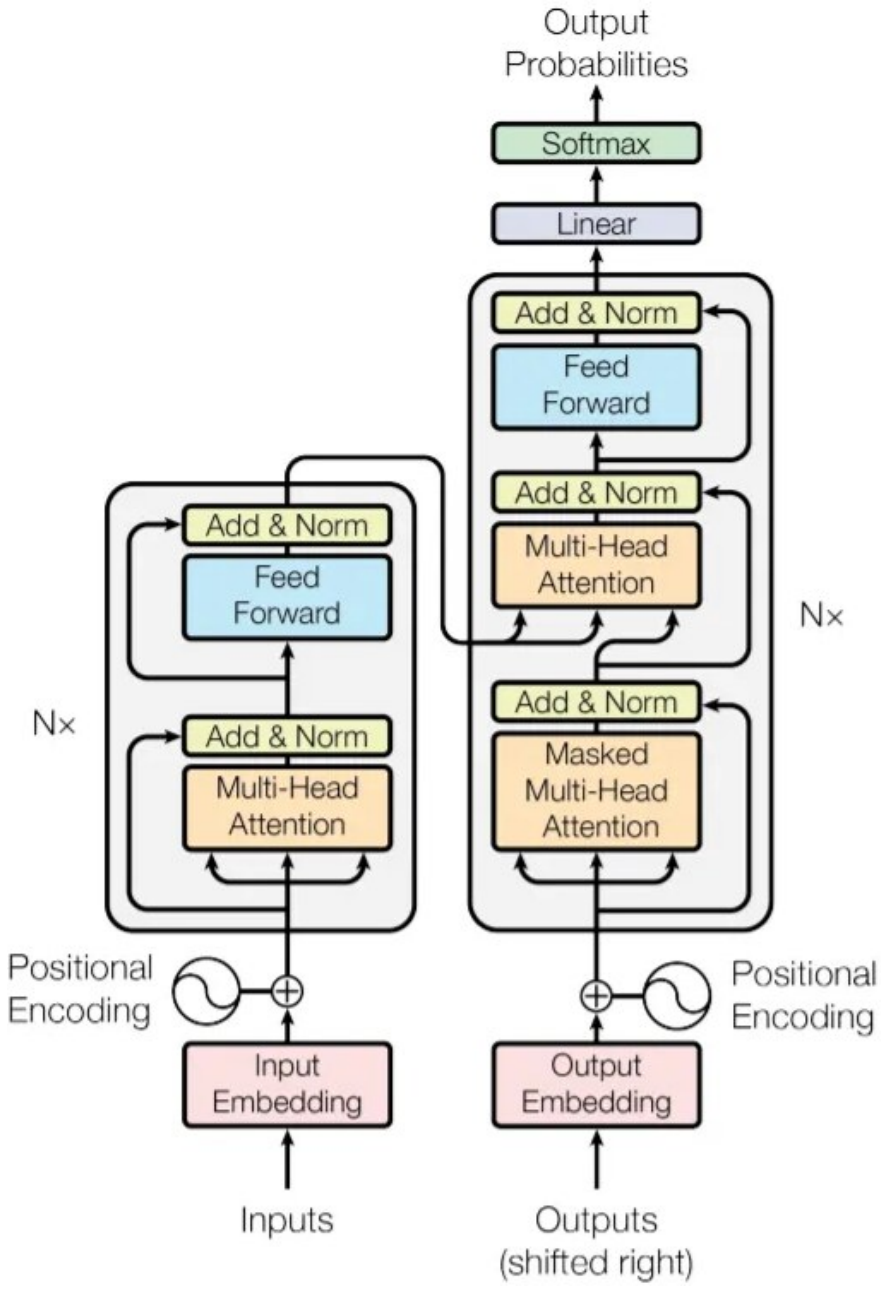
$$maskedattention(Q, K, V) = softmax\left(\frac{Q^T K + M}{\sqrt{d_k}}\right) V$$

M é uma matriz com `-inf` nas posições de palavras futuras

Representações para cada palavra

1. Word embeddings: w
2. com positional encoding: $\mathrm{w} + \mathrm{p} = \mathrm{e}$
3. Query: $\mathrm{Q}_h = W_Q \cdot \mathrm{e}_h$
4. Key: $\mathrm{K}_h = W_K \cdot \mathrm{e}_h$
5. Value: $\mathrm{V}_h = W_V \cdot \mathrm{e}_h$
6. Attention heads: $Z_h = softmax\left(\frac{Q_h^T K_h + M}{\sqrt{d_k}}\right) V_h$
7. Attention output: $Z = W_O concat((Z_1, Z_2, \cdots Z_H))$

Arquitetura Transformer



Agenda

Transformer Network

BERT: pré-treinamento de encoders de transformers bidirecionais

GPT: generative pretrained transformer

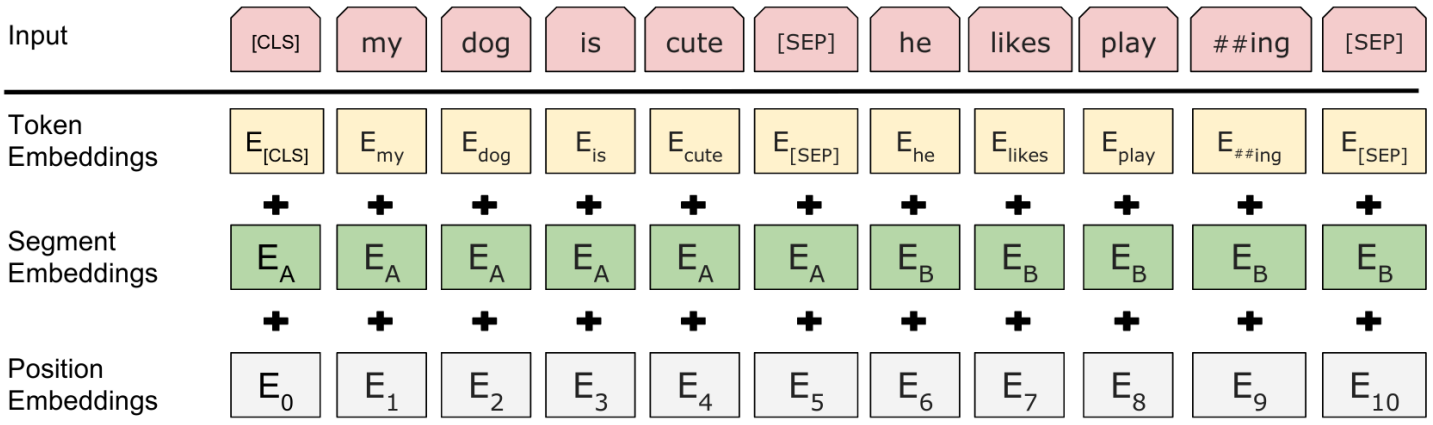
BERT e Transformer

Método para **pré-treinar** encoders to tipo Transformer

— BERT Base e BERT Large, modelos com blocos Transformer:

- Base:
  - 12 camadas (blocos Transformer)
  - Embedding com 768 dimensões
  - 110 milhões de parâmetros
- Large:
  - 24 camadas (blocos Transformer)
  - Embedding com 1024 dimensões
  - 336 milhões de parâmetros

BERT Treinamento e Embeddings

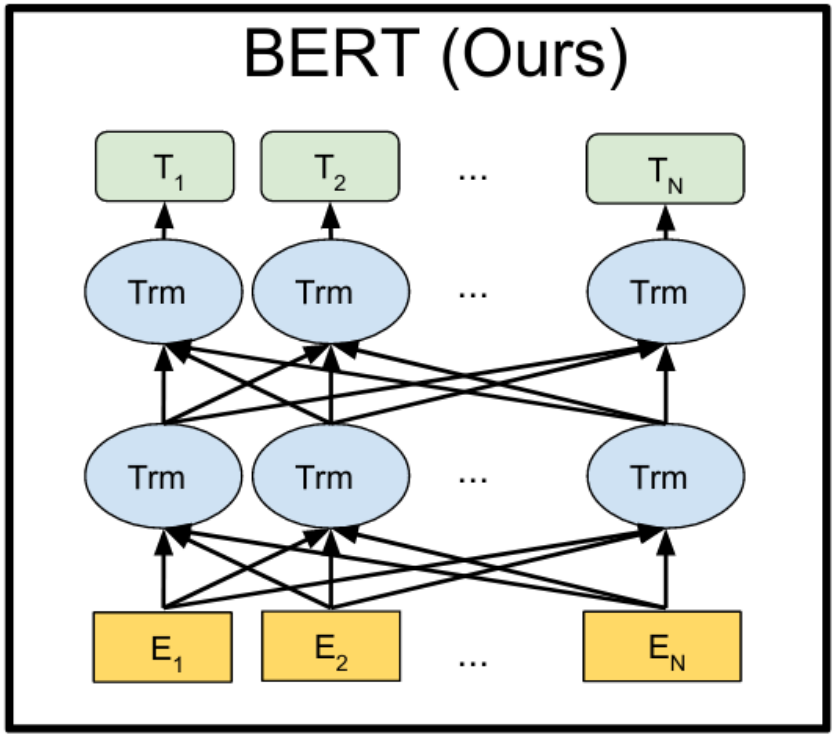


BERT: pré-treinamento inspirado em ELMo

- GloVe e similares
  - vetor fixo por palavra independente do contexto
  - ex. manga, bateria, pilha.
- Olha para toda a sentença antes de atribuir vetor
  - aprende (sem labels) a predizer a próxima palavra (e a anterior)

vamos deixar o menino jogar bola
$\rightarrow$ vamos deixar o menino jogar [mask]
[mask] deixar o menino jogar bola $\leftarrow$
vamos deixar o $\rightarrow$ [mask] $\leftarrow$jogar bola

BERT Bidirectional Transformer

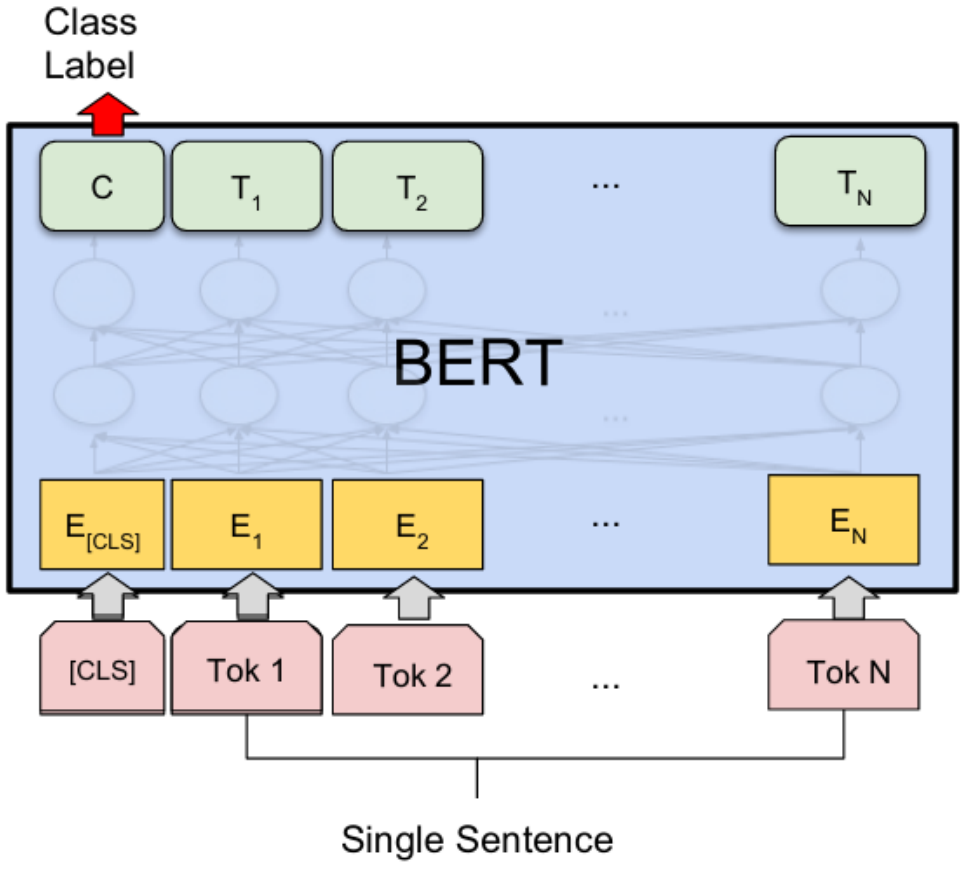


BERT: segunda tarefa de pré-treinamento

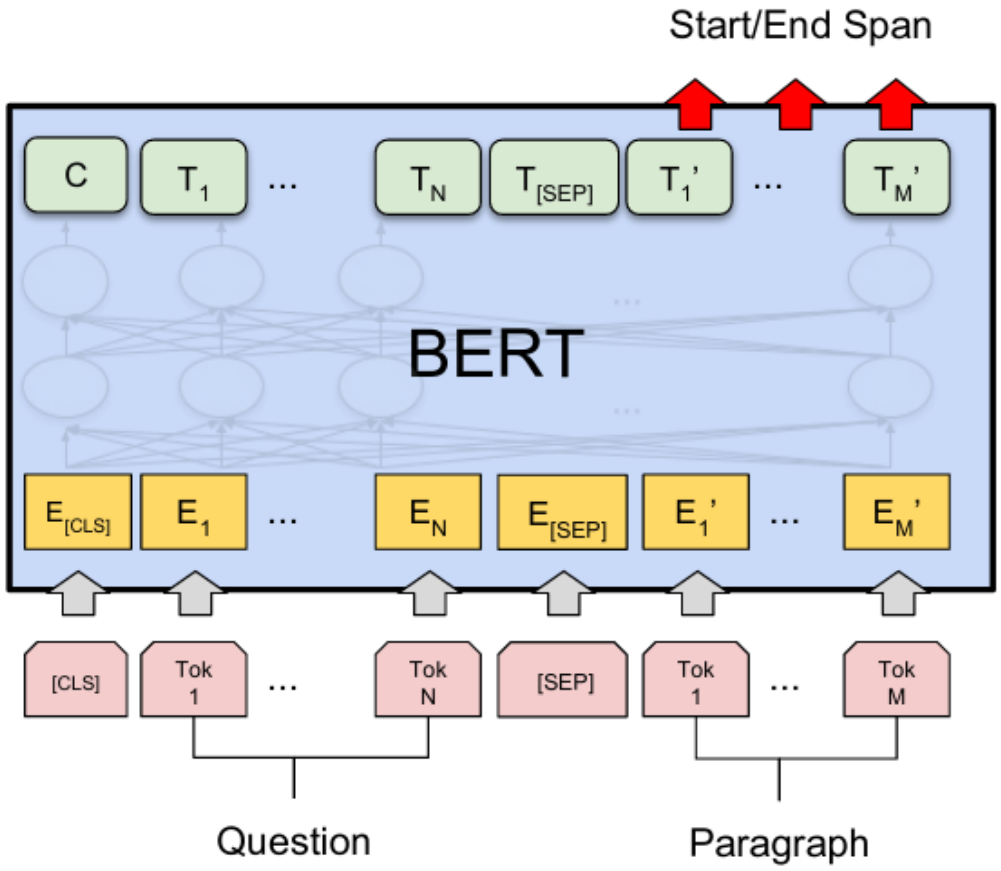
$$\text{Input} = \texttt{[CLS] the man went to [MASK] store [SEP]}$$
$$\texttt{he bought a gallon [MASK] milk [SEP]}$$
$$\text{Label} = \texttt{IsNext}$$

$$\text{Input} = \texttt{[CLS] the man [MASK] to the store [SEP]}$$
$$\texttt{penguin [MASK] are flight ##less birds [SEP]}$$
$$\text{Label} = \texttt{NotNext}$$

BERT: para classificação



BERT: para encontrar resposta em texto



Agenda

Transformer Network

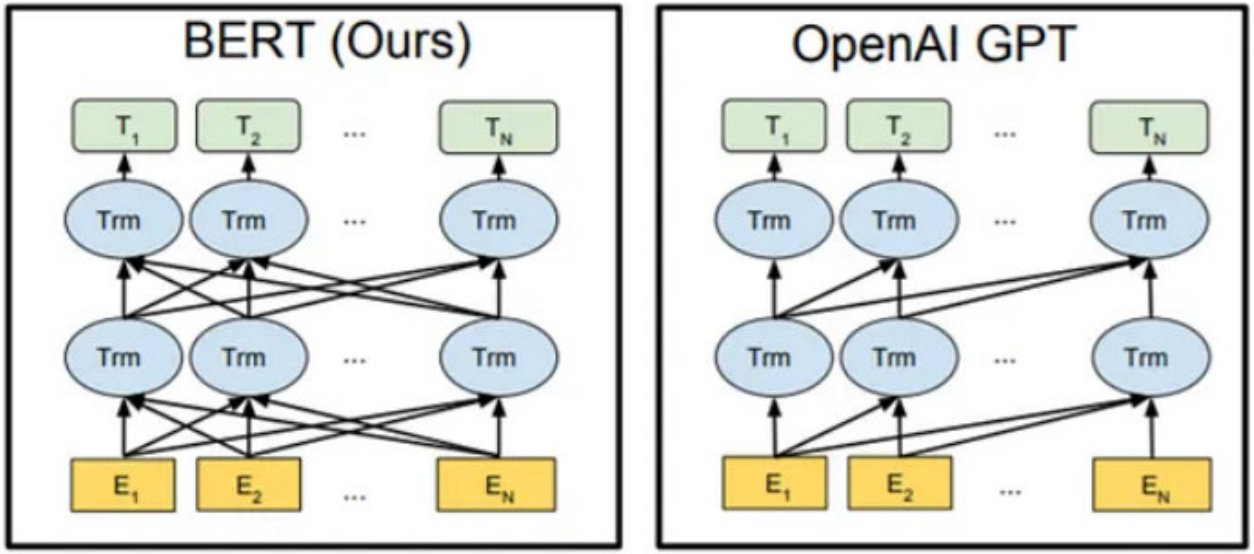
BERT: pré-treinamento de encoders de transformers bidirecionais

GPT: generative pretrained transformer

GPT e Transformer

Método para **pré-treinar** decoders to tipo Transformer

Modelo auto-regressivo: usa a saída atual como entrada



GPT

O atacante fez um

$$p(\text{"gol"}|\text{"O atacante fez um"}) = 0.203$$
$$p(\text{"jogo"}|\text{"O atacante fez um"}) = 0.196$$
$$p(\text{"golaço"}|\text{"O atacante fez um"}) = 0.141$$
$$p(\text{"pedido"}|\text{"O atacante fez um"}) = 0.105$$
$$p(\text{"comentário"}|\text{"O atacante fez um"}) = 0.083$$
$$p(\text{"chute"}|\text{"O atacante fez um"}) = 0.070$$
$$p(\text{"bolo"}|\text{"O atacante fez um"}) = 0.009$$

GPT

O atacante fez um gol **na partida de ontem**

Temperatura

O atacante fez um gol na partida de ontem
O atacante fez um gol invalidado pelo VAR
O atacante fez um gol de fora da área
O atacante fez um jogo ruim no segundo tempo
O atacante fez um pedido à torcida

GPT 3 e superiores

1. Coletar grandes quantidades de dados
2. Pré-treinar com bases massivas para gerar tokens (auto-supervisionado)
3. Ajustar com tarefas específicas e controladas (ex. perguntas e respostas): fine-tuning
4. Realizar ajuste com aprendizado por reforço usando humanos (RLHF - Reinforcement Learning Human Feedback)

Resumo

1. Word / sentence embeddings: espaços com coerência e contexto entre textos
2. Transformer: modelo sequence-to-sequence
3. BERT: modelo que olha todo o texto para gerar a predição, possível de ajustar para diferentes tarefas alvo
4. GPT: modelo autoregressivo para geração de texto

References

- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention is all you need, NeurIPS 2017
- Dzmitry Bahdanau, KyungHyun Cho and Yoshua Bengio. Neural Machine Translation By Jointly Learning To Align And Translate. ICLR 2015.
- A. Karpathy. Understanding LSTM Networks. `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`
- C. Olah. Understanding LSTM Networks `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`