Word2Vec, Sequence2Sequence e Mecanismo de Atenção

Redes Neurais e Aprendizado Profundo

Moacir A. Ponti

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

São Carlos-SP/Brasil

Agenda

Word2Vec: representações para texto

Sequence-to-Sequence e Mecanismo de Atenção
- Transformer Network

Agenda

Word2Vec: representações para texto

Sequence-to-Sequence e Mecanismo de Atenção

- Transformer Network

Word2Vec

- Representação (embedding) para palavras
- A função de custo para aprender essa representação:

$$p(w_{t+j}|w_t)$$

  t é a palavra, $t + j$ são todas as palavras no contexto de t
- Otimiza em função de palavras que devem estar próximas se estiverem no mesmo contexto

Word2Vec: skipgrams

Skip-grams (SG)

Predição de palavras em uma certa "janela" de proximidade m de uma palavra t

- Formulação "softmax":

$$\frac{\exp(u_o^T v_c)}{\sum_w^V \exp(u_w^T v_c)}$$

V é o total de palavras no vocabulário
u_o é a representação de uma palavra de "saída" (Que queremos prever
v_c é a representação de uma palavra de entrada (central)

Word2Vec: skipgram

Dada uma representação one-hot de uma palavra $w_t \in R^V$, calculamos sua representação vetorial $v_c \in R^d$ (central)

$$W \cdot w_t = v_c$$

$$\begin{bmatrix} \cdots & \cdots & 0.1 & \cdots & \cdots \\ \cdots & \cdots & -0.3 & \cdots & \cdots \\ \cdots & \cdots & 1.4 & \cdots & \cdots \\ \cdots & \cdots & 0.2 & \cdots & \cdots \\ \cdots & \cdots & 0.5 & \cdots & \cdots \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.3 \\ 1.4 \\ 0.2 \\ 0.5 \end{bmatrix}$$

Word2Vec: skipgram

v_c é filtrada por representações u_o das palavras de saída (no contexto, que queremos prever) em diferentes posições $t - i$

$$u_o^T \cdot v_c$$

Para todas as palavras do vocabulário isso é codificado em uma matriz:

$$U_o \cdot v_c$$

$$\begin{bmatrix} 0.0 & 2.0 & 0.1 & 2.0 & 0.1 \\ 0.0 & 1.0 & 2.0 & -0.5 & 1.0 \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix} \begin{bmatrix} 0.1 \\ -0.3 \\ 1.4 \\ 0.2 \\ 0.5 \end{bmatrix} = softmax \left(\begin{bmatrix} 00 \\ 2.9 \\ 0.1 \\ 1.4 \\ -0.5 \\ 0.0 \\ 0.0 \end{bmatrix} \right) = \begin{bmatrix} 0.04 \\ 0.67 \\ 0.04 \\ 0.15 \\ 0.02 \\ 0.04 \\ 0.04 \end{bmatrix}$$

Word2Vec: skipgram

- W aprende representações (nas colunas) para cada palavra quando são "centrais"
- U_o aprende representações (nas linhas) para cada palavra quando são "contexto"

$$W = \begin{bmatrix} \cdots & \cdots & 0.1 & \cdots & \cdots \\ \cdots & \cdots & -0.3 & \cdots & \cdots \\ \cdots & \cdots & 1.4 & \cdots & \cdots \\ \cdots & \cdots & 0.2 & \cdots & \cdots \\ \cdots & \cdots & 0.5 & \cdots & \cdots \end{bmatrix} \quad U_o = \begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots \\ 0.0 & 1.0 & 2.0 & -0.5 & 1.0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

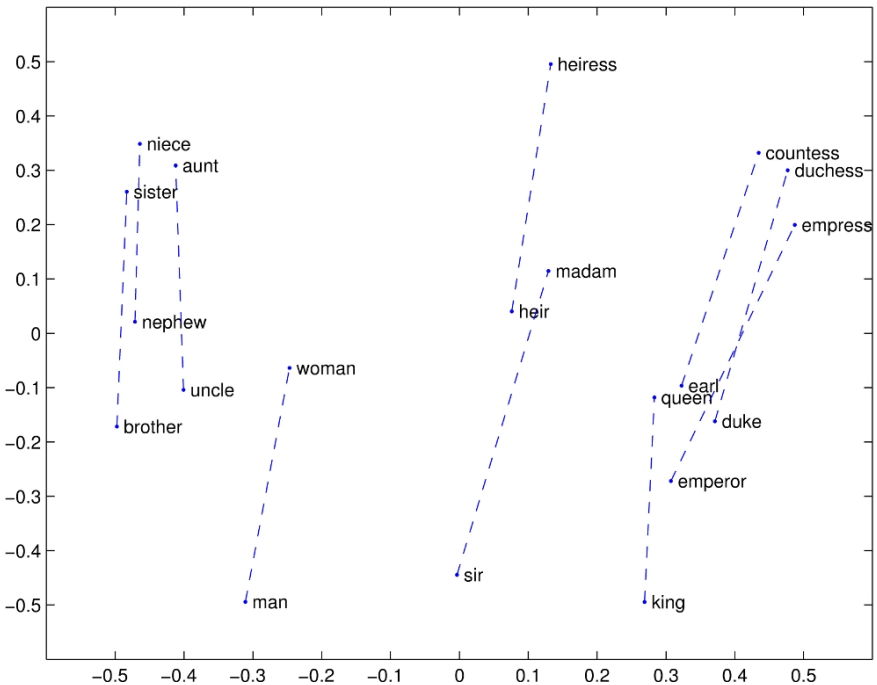
Word2Vec: skipgram

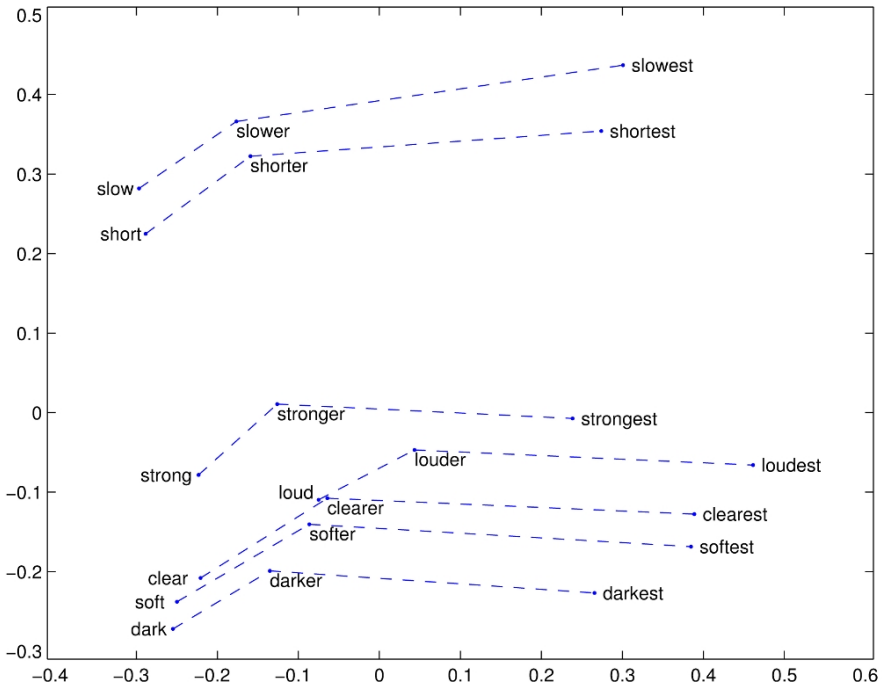
- Palavras que aparecem num mesmo contexto terão representações similares

Deixa o menino jogar
Deixa o moleque jogar
Deixa o piá jogar
Deixa seu filho jogar

Word2Vec: GloVe (Global Vectors for Word Representation)

https://nlp.stanford.edu/projects/glove/





Word Embeddings em Português - NILC/ICMC

http://www.nilc.icmc.usp.br/embeddings

Agenda

Word2Vec: representações para texto

Sequence-to-Sequence e Mecanismo de Atenção

- Transformer Network

RNNs e Sequence-to-Sequence (seq2seq)

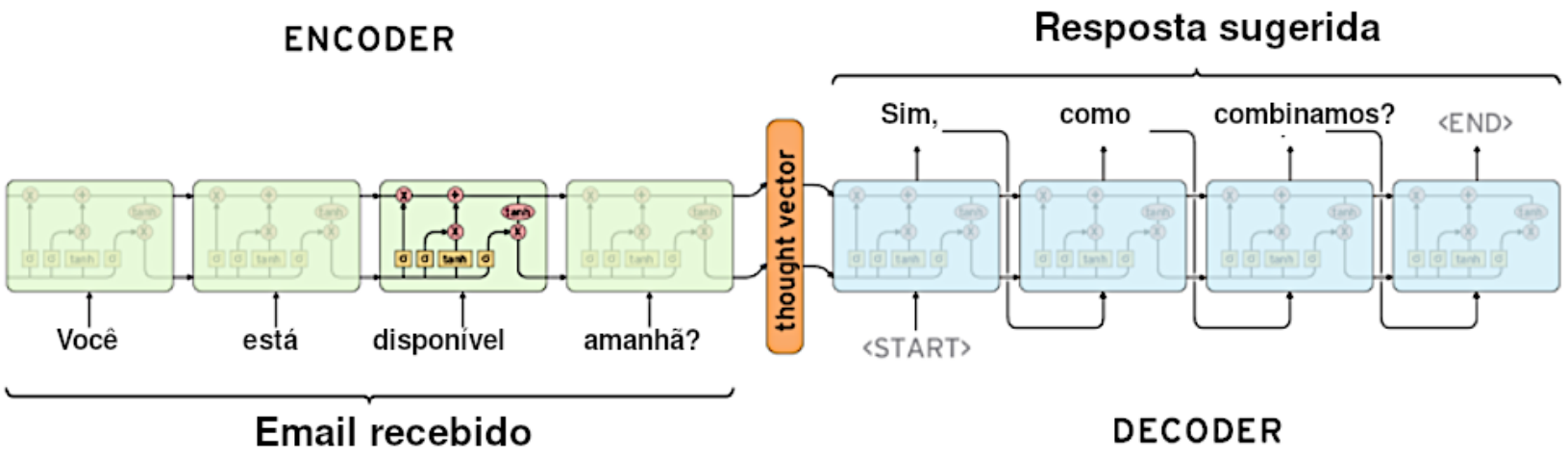


Figura adaptada de: Sachin Abeywardana

- (3 vídeos de Jay Alammar)

Mecanismo de atenção: intuição e motivação

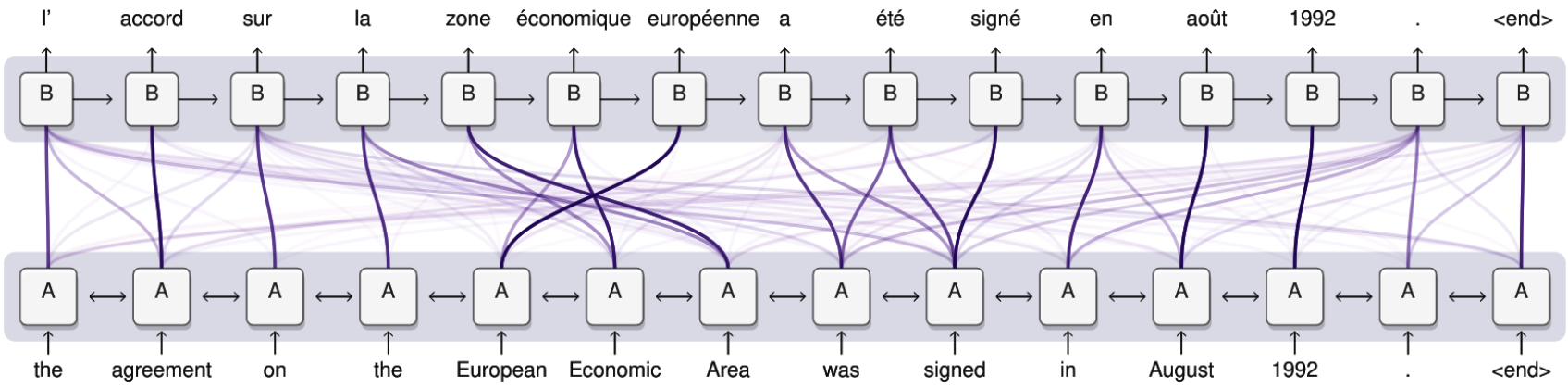
- Encontrar qual parte de uma sequência é mais importante para predizer uma certa saída
- Em unidades recorrentes, cada entrada perturba a memória prejudicando conhecimento de dados anteriores

Mecanismo de Atenção: imagens

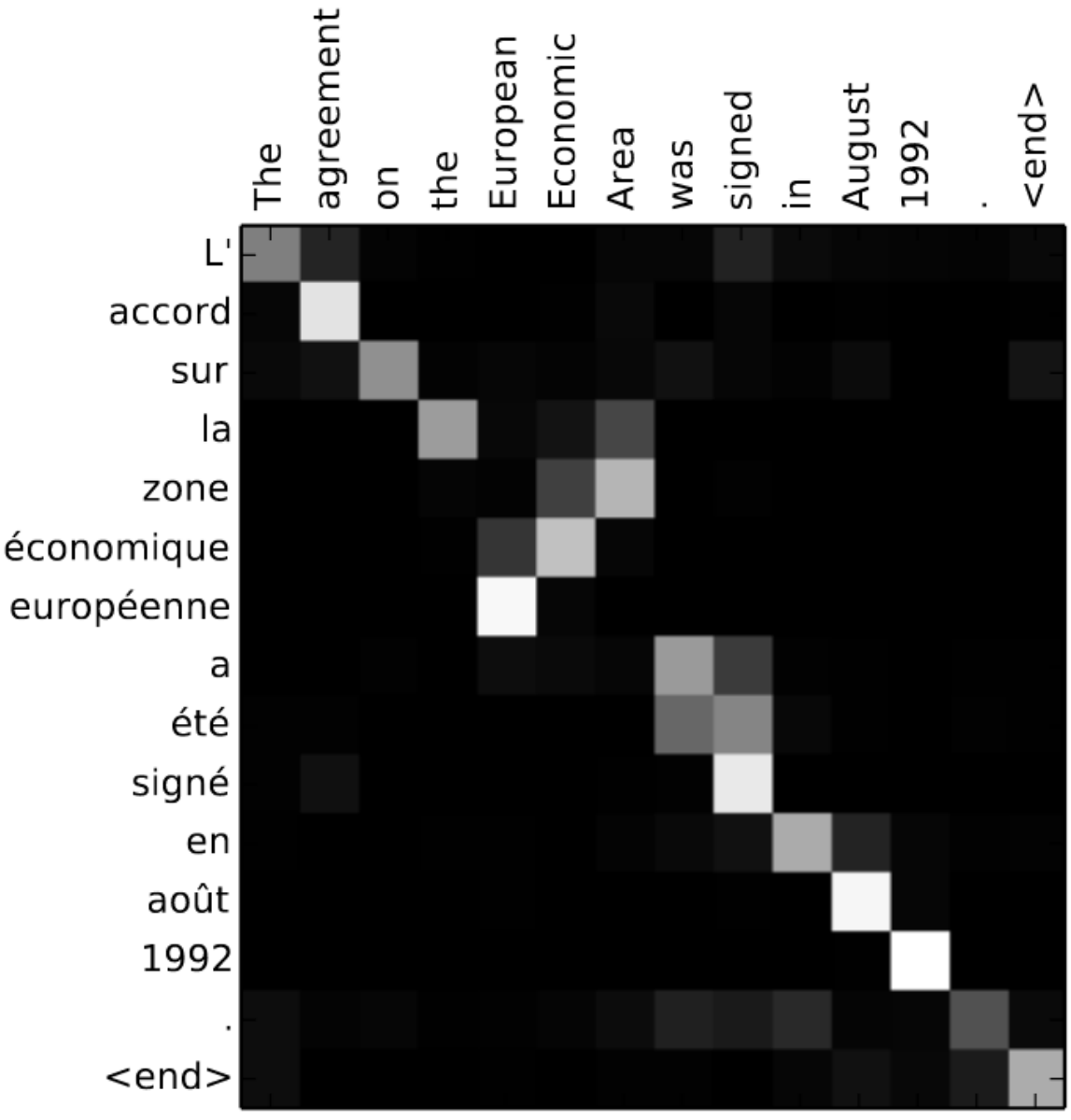
A man wearing a hat and
a hat on a skateboard.

A man is talking on his cell phone
while another man watches.

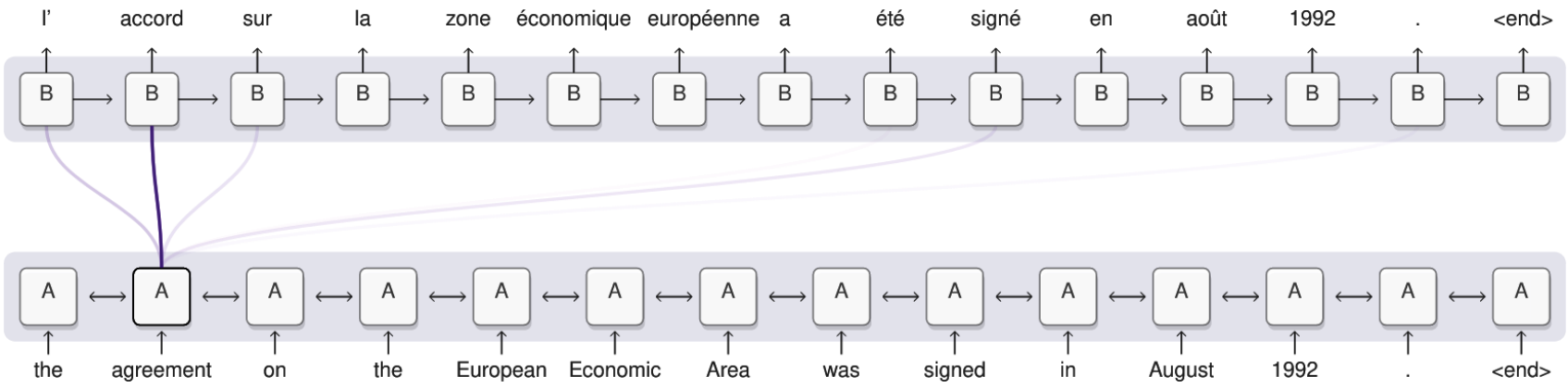
Mecanismo de Atenção: texto



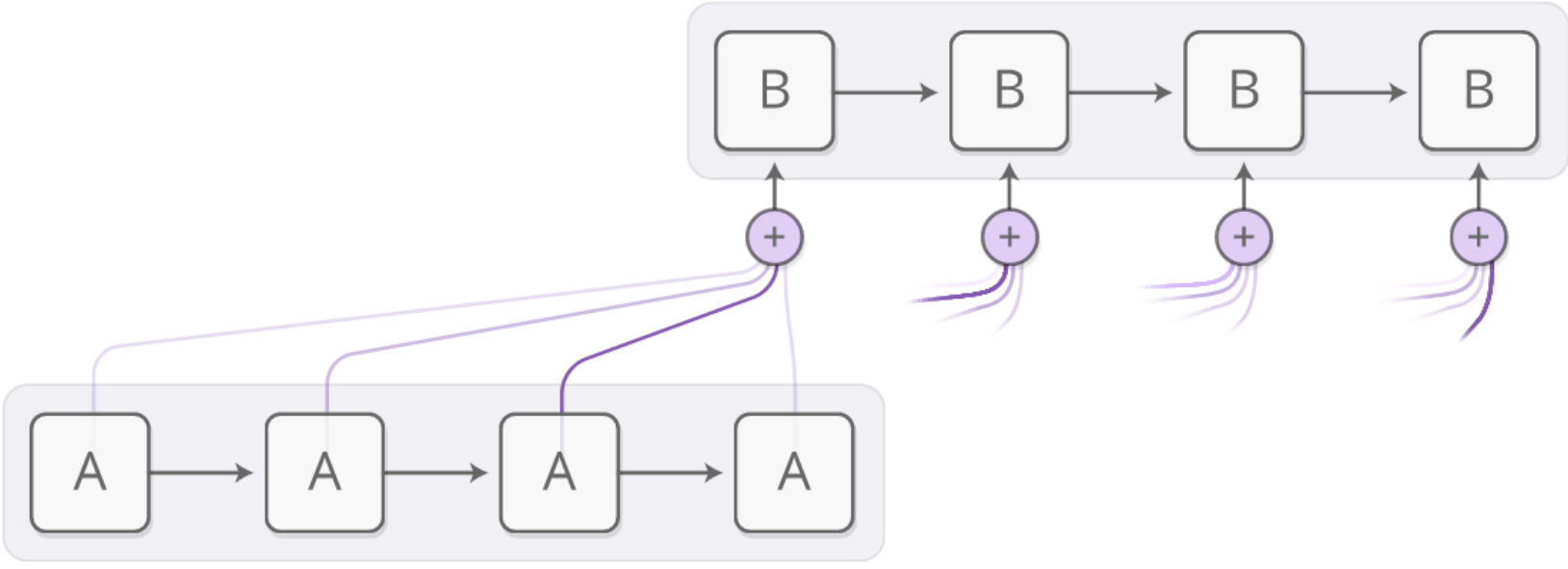
Mecanismo de Atenção: texto



Mecanismo de Atenção: texto



RNNs seq2seq e atenção global



Adaptado de Olah & Carter, "Attention and Augmented Recurrent Neural Networks", Distill, 2016.

http://doi.org/10.23915/distill.00001

- (vídeos de Jay Alammar)

Mecanismo de atenção: implementação básica

- Computar o alinhamento/similaridade entre o sumário atual do decoder, s_i, com sumários anteriores do encoder, h_j

  Usa softmax para obter pesos na forma de probabilidades

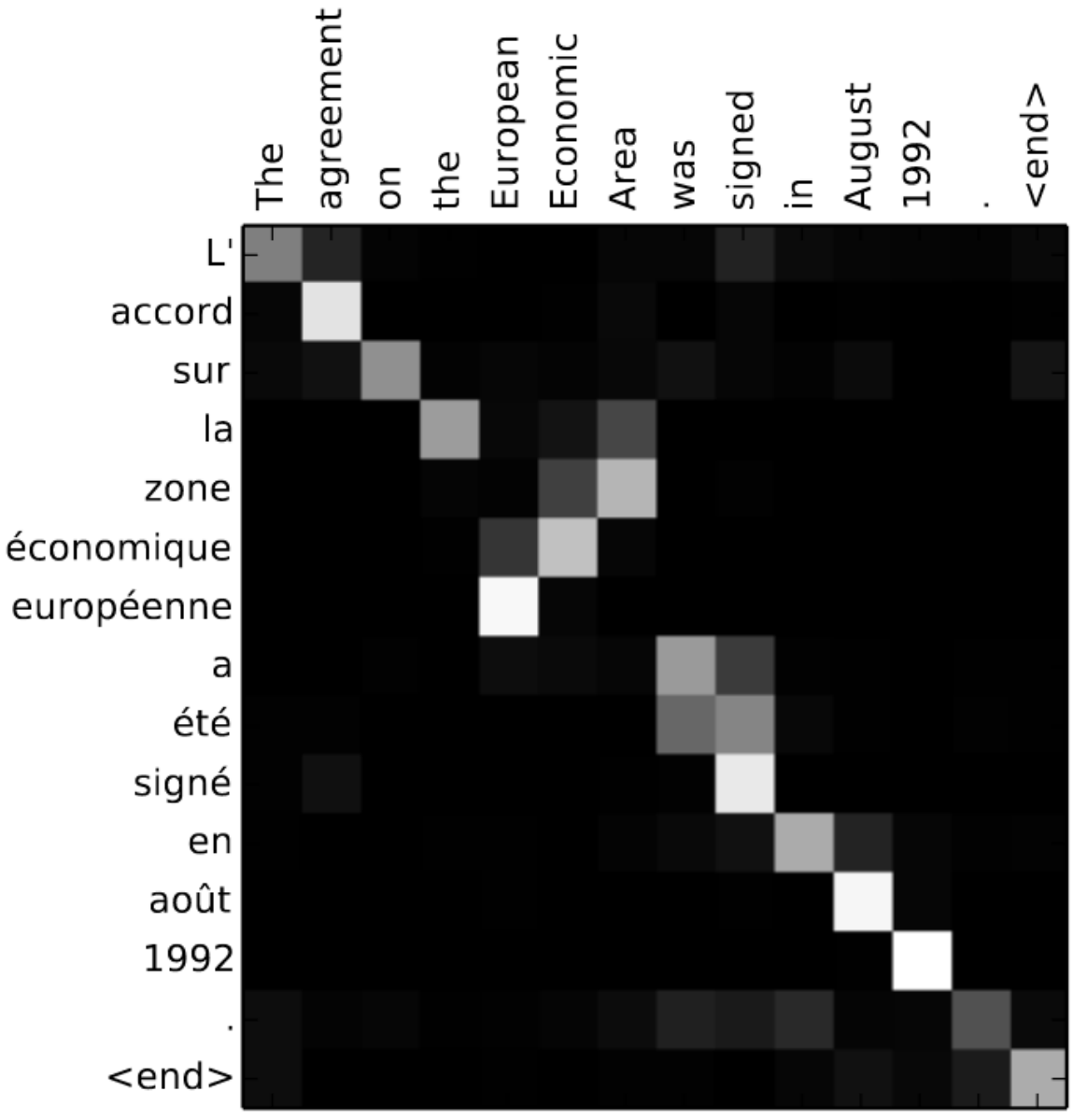
  $$a_{i,j} = \frac{\exp(alinhamento(s_i, h_k))}{\sum_k \exp(alinhamento(s_i, h_k))},$$

  "alinhamento" é um tipo de similaridade, e.g. produto interno:

  $$alinhamento(s_i, h_k)) = s_i^T h_j$$

- Atenção produz um vetor de "contexto" $c_i = \sum_j a_{i,j} h_j$ a ser usado para produzir a saída atual.

Mecanismo de Atenção: exemplo de vetores de alinhamento



Agenda

Word2Vec: representações para texto

Sequence-to-Sequence e Mecanismo de Atenção
- Transformer Network

RNNs vs Transformer Networks

RNNs

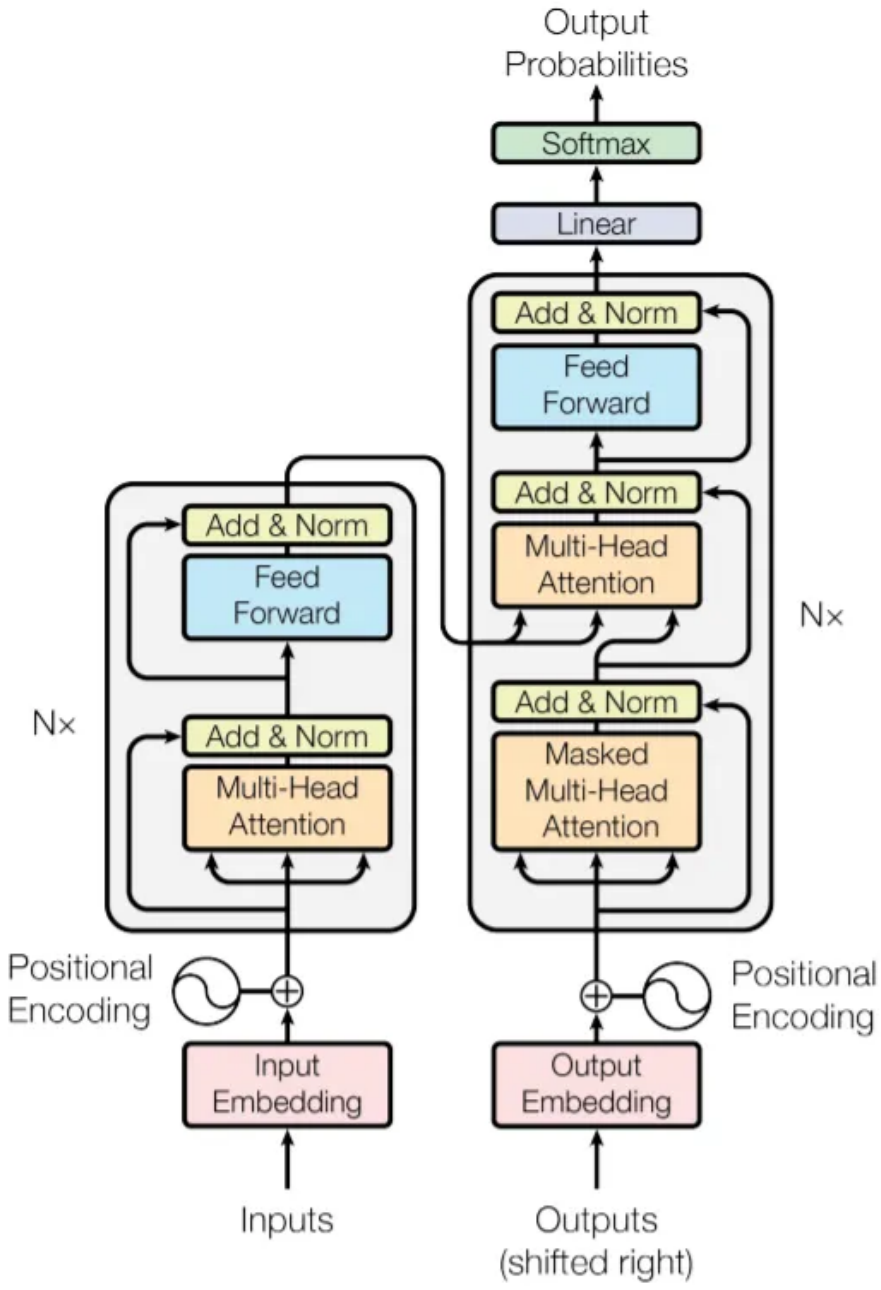
- podem não funcionar com dependências longas
- recorrência dificulta computação paralela (pontos da sequência não podem ser processados em paralelo)
- podem sofrer com explosão ou desaparecimento de gradiente

Transformer Networks

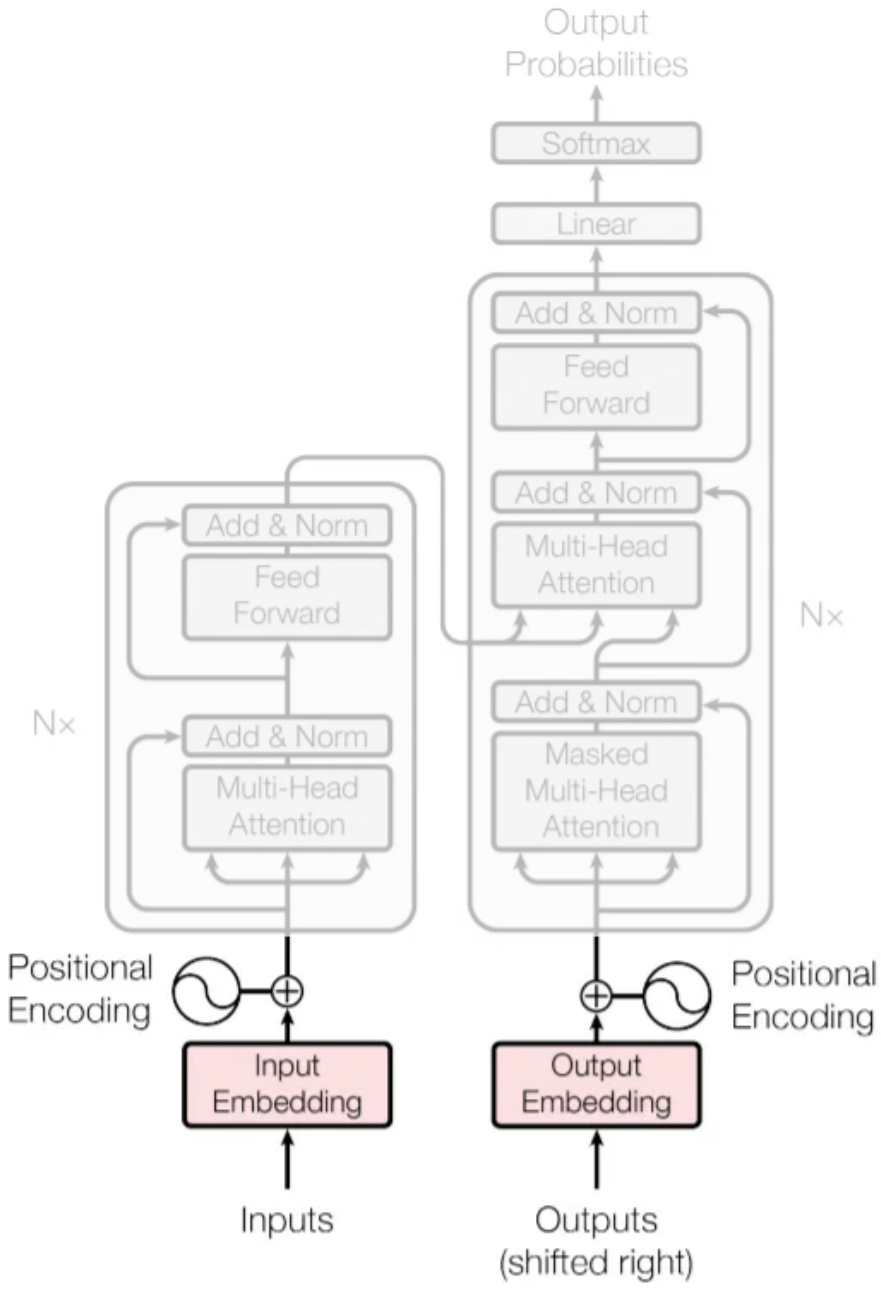
- não possui recorrência, apenas atenção
- facilita capturar dependências longas
- facilita processamento paralelo: atenção é invariante à permutação

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

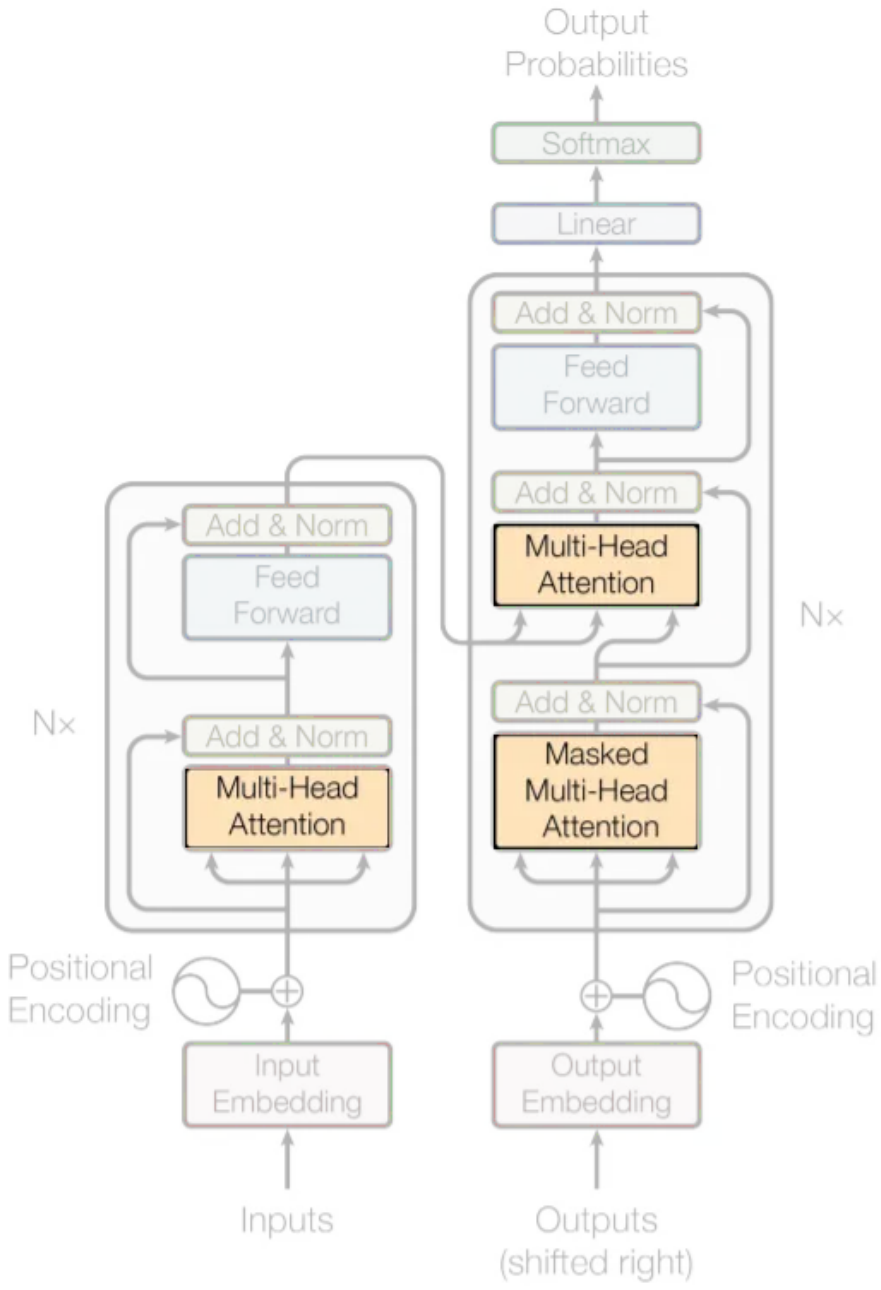
Arquitetura Transformer



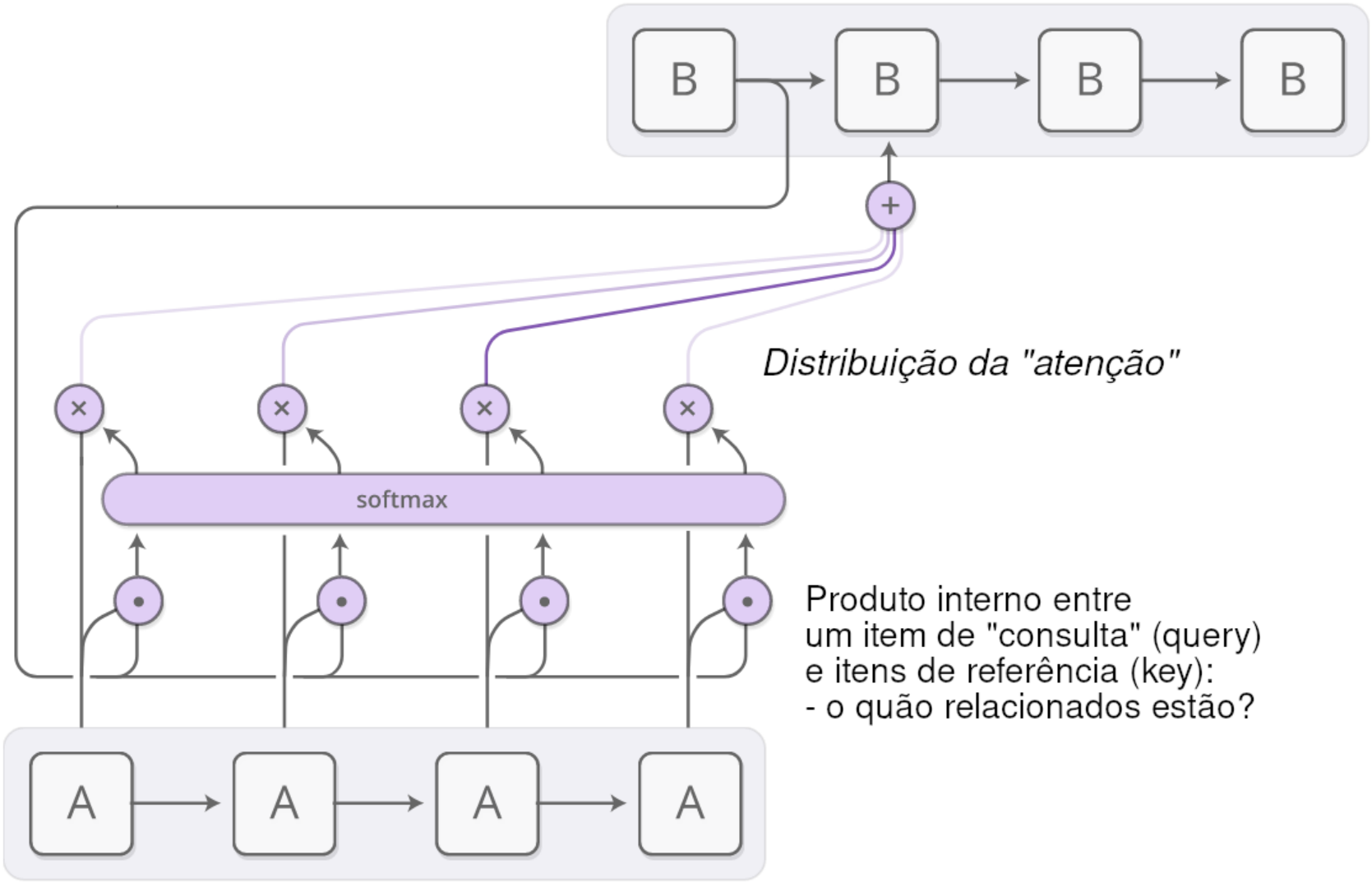
Positional Encoding: adiciona informação de posicionamento



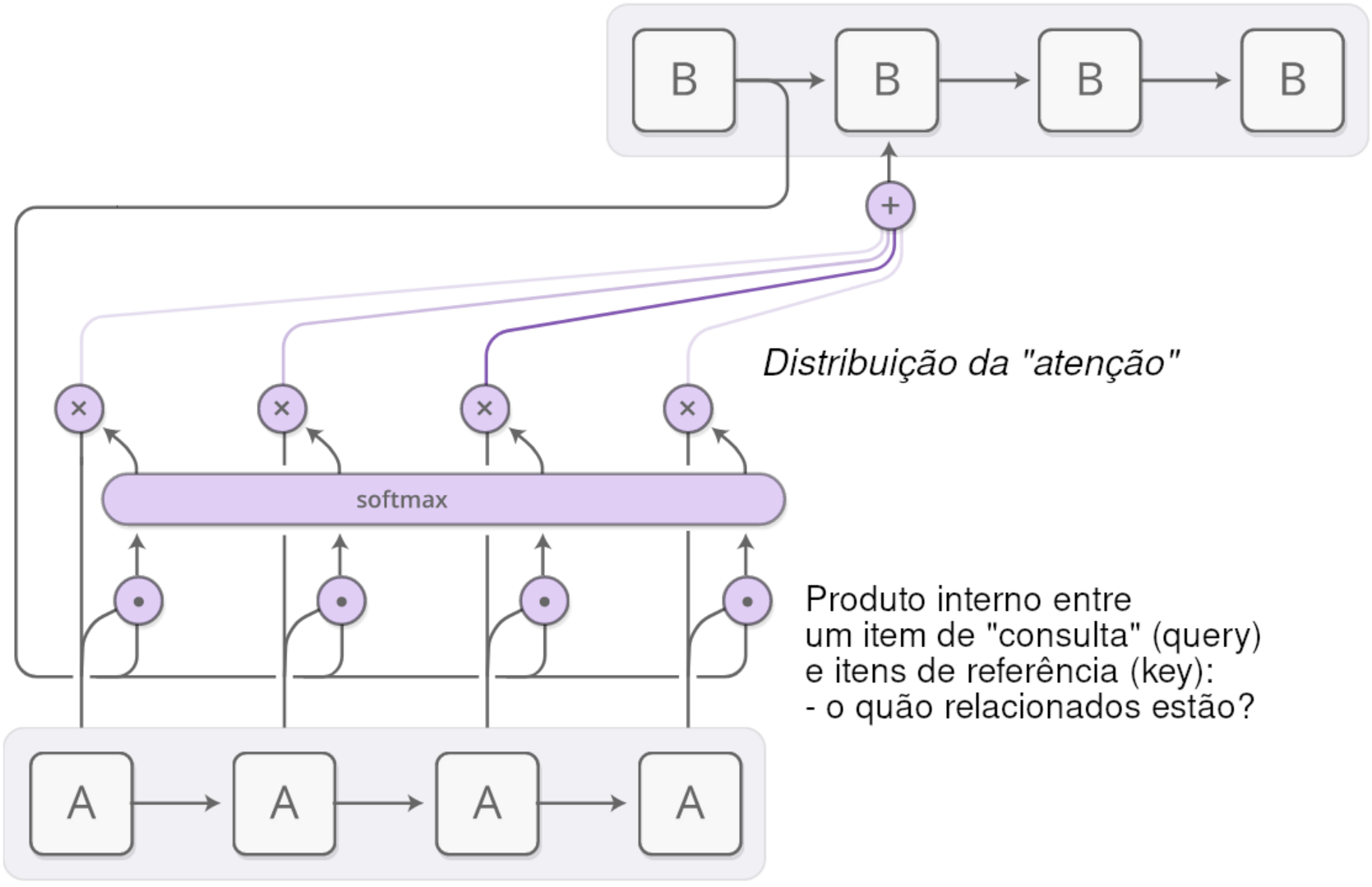
Arquitetura Transformer: multi-atenção



Distribuição da atenção com relação ao estado atual



Distribuição da atenção com relação ao estado atual



Mecanismo de atenção em Transformer Networks

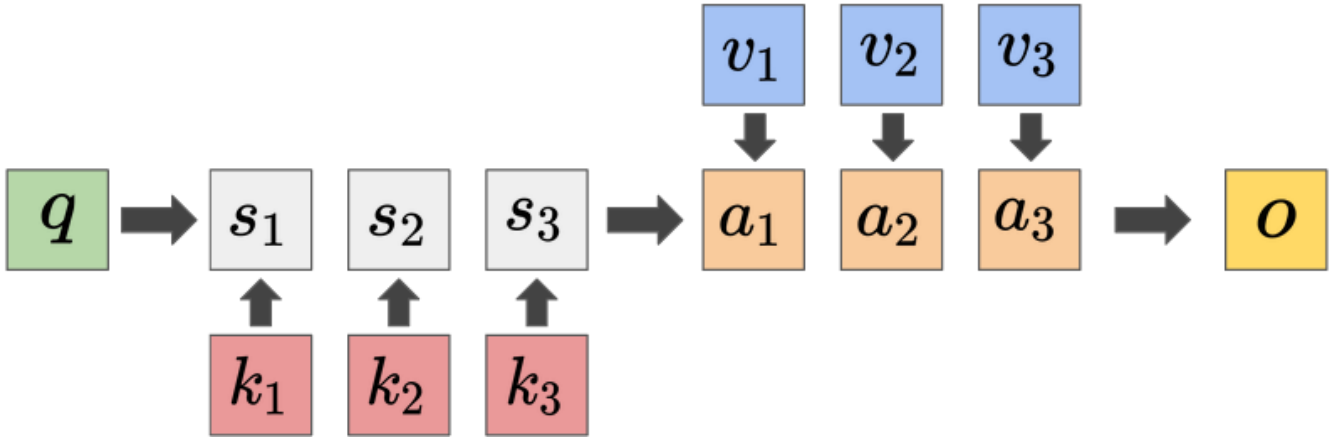
- Recuperar um valor v_i para uma consulta/query q baseada numa chave/key k_i

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

- 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

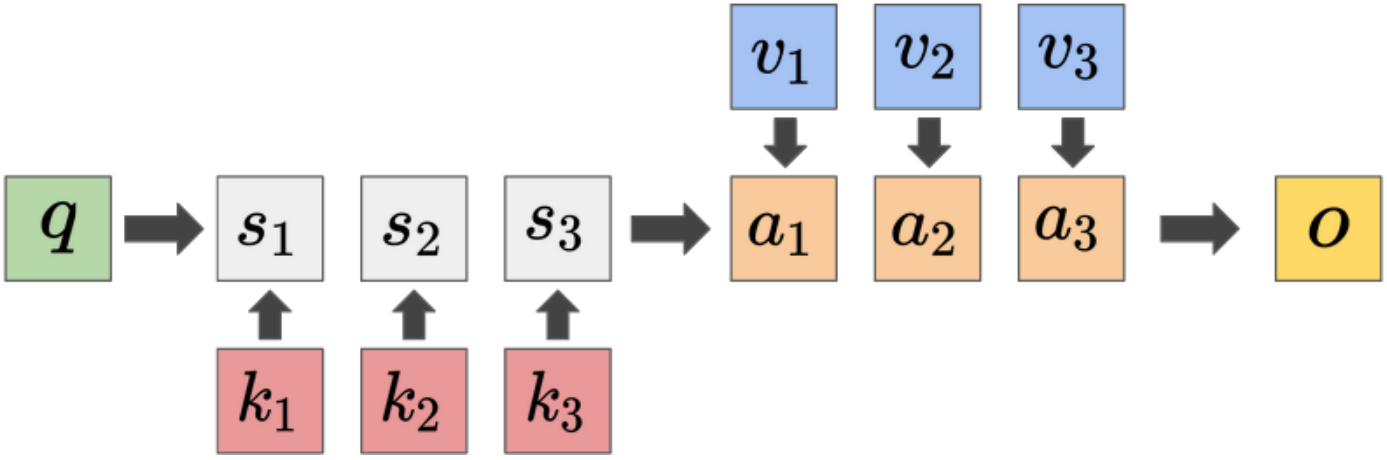
Atenção

$$s_i = \text{sim}(\boldsymbol{q}, \boldsymbol{k_i}) \qquad a_i = \frac{e^{s_i}}{\sum_{j=1}^{N} e^{s_j}} \qquad \boldsymbol{o} = \sum_{i=1}^{N} a_i \boldsymbol{v_i}$$

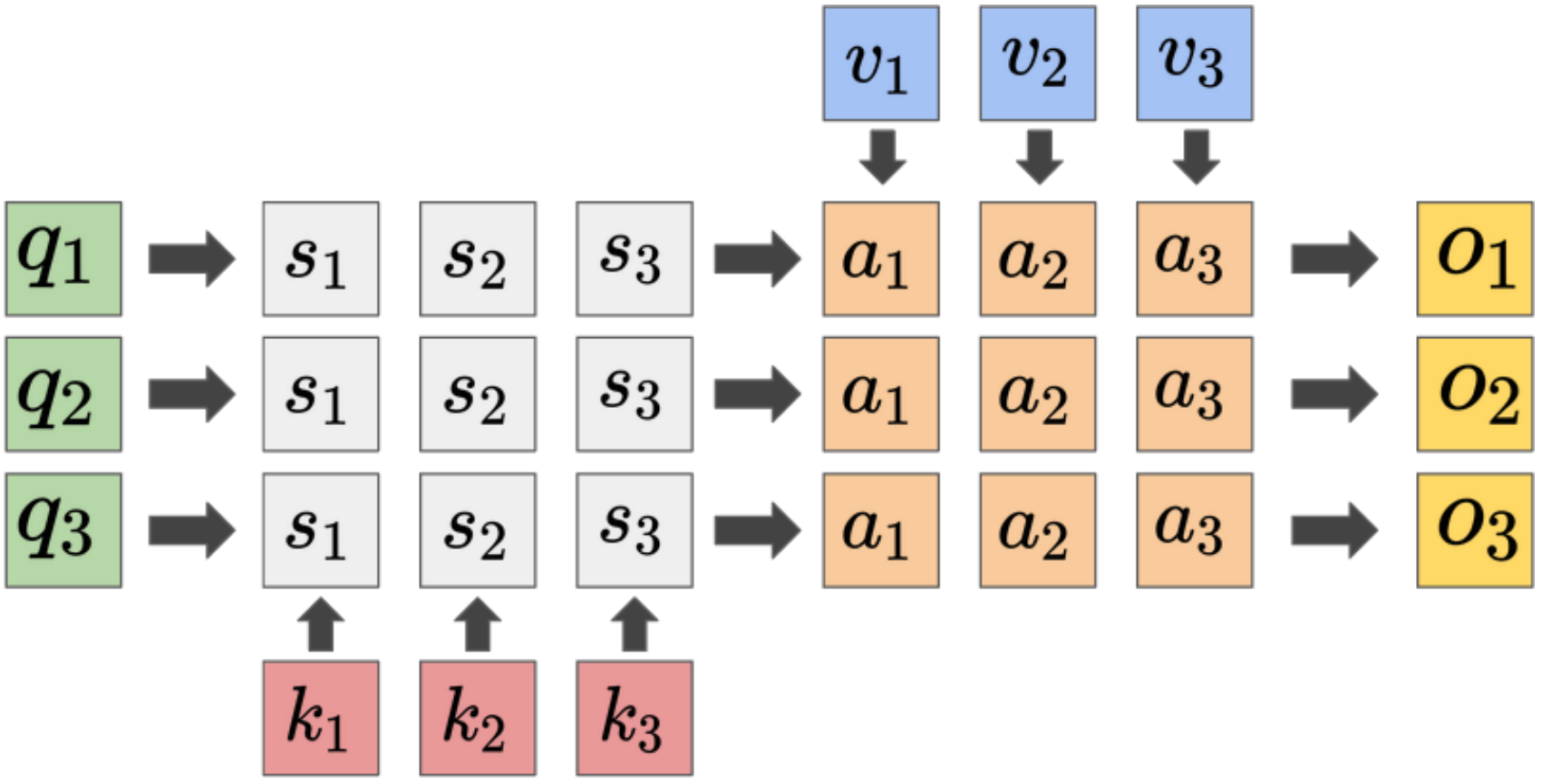


Atenção

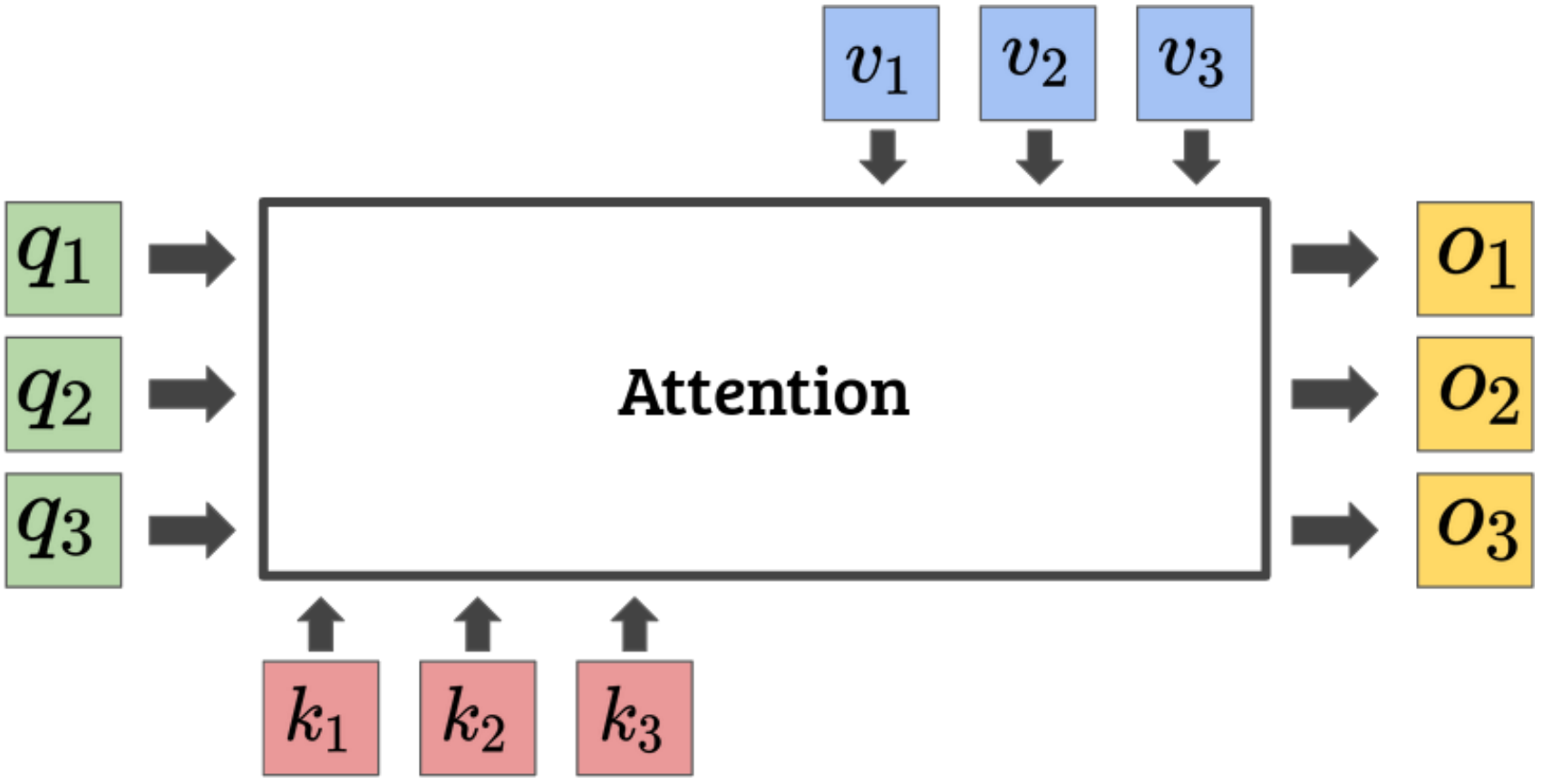
$$s_i = \boldsymbol{q}^T \boldsymbol{k_i} \qquad a_i = \frac{e^{s_i}}{\sum_{j=1}^{N} e^{s_j}} \qquad o = \sum_{i=1}^{N} a_i \boldsymbol{v_i}$$



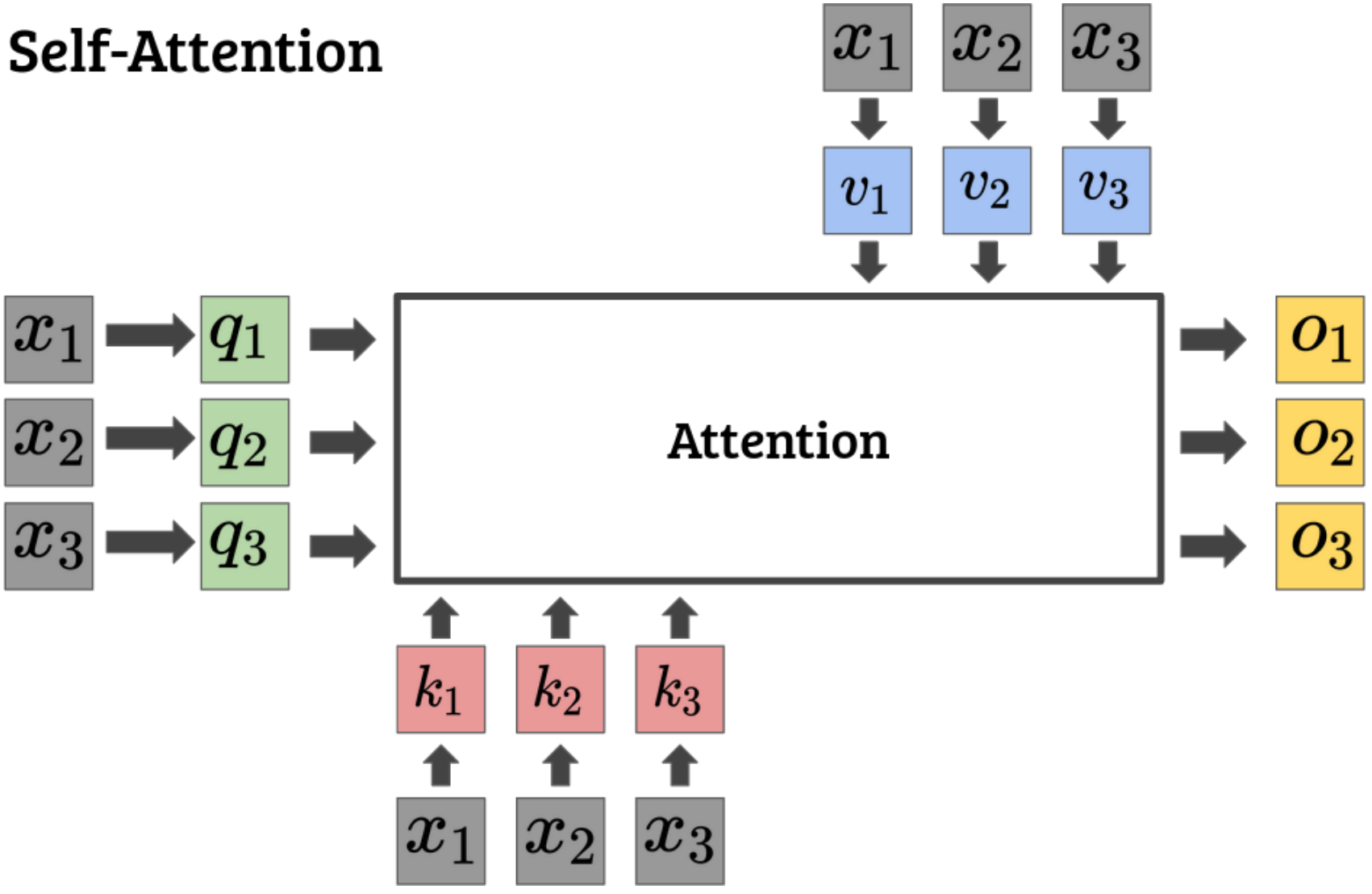
Atenção


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

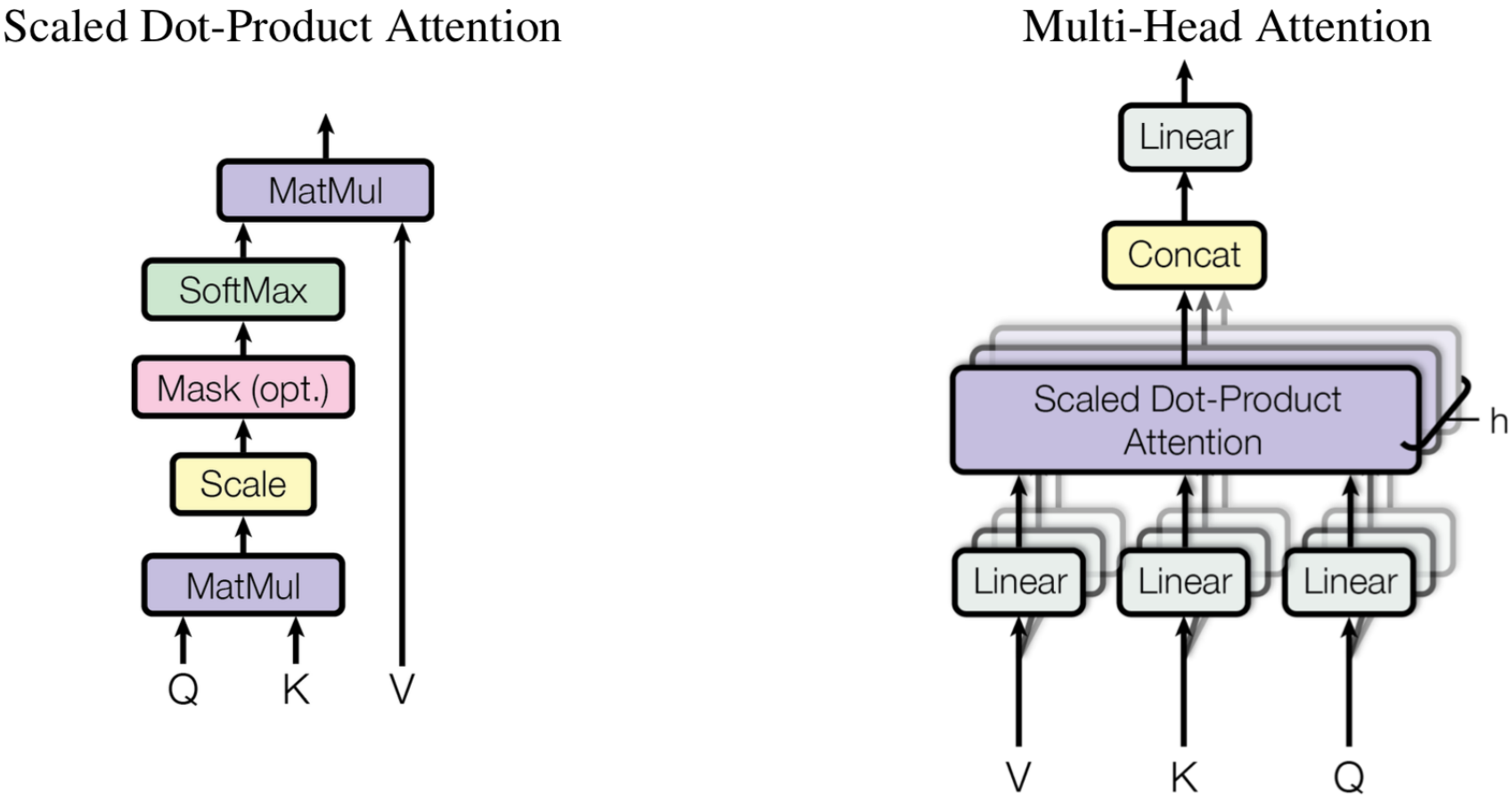
Atenção



Atenção (auto-atenção)



Multi-head attention



Multi-head attention

Cada cabeça de atenção implica em menor custo computacional pois:

- recebe $1/H$ das dimensões originais de V, K, Q.
- após as projeções cada cabeça recebe um tensor no formato $batchsize, seqlen, H, d/H$

Para texto: representações para cada palavra

1. Word embeddings: w
2. Word embeddings + Positional encoding: $\mathrm{w} + \mathrm{p} = \mathrm{e}$
3. Query: $\mathrm{Q}_h = W_Q \cdot \mathrm{e}$
4. Key: $\mathrm{K}_h = W_K \cdot \mathrm{e}$
5. Value: $\mathrm{V}_h = W_V \cdot \mathrm{e}$

h indexa as H "cabeças de atenção"

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

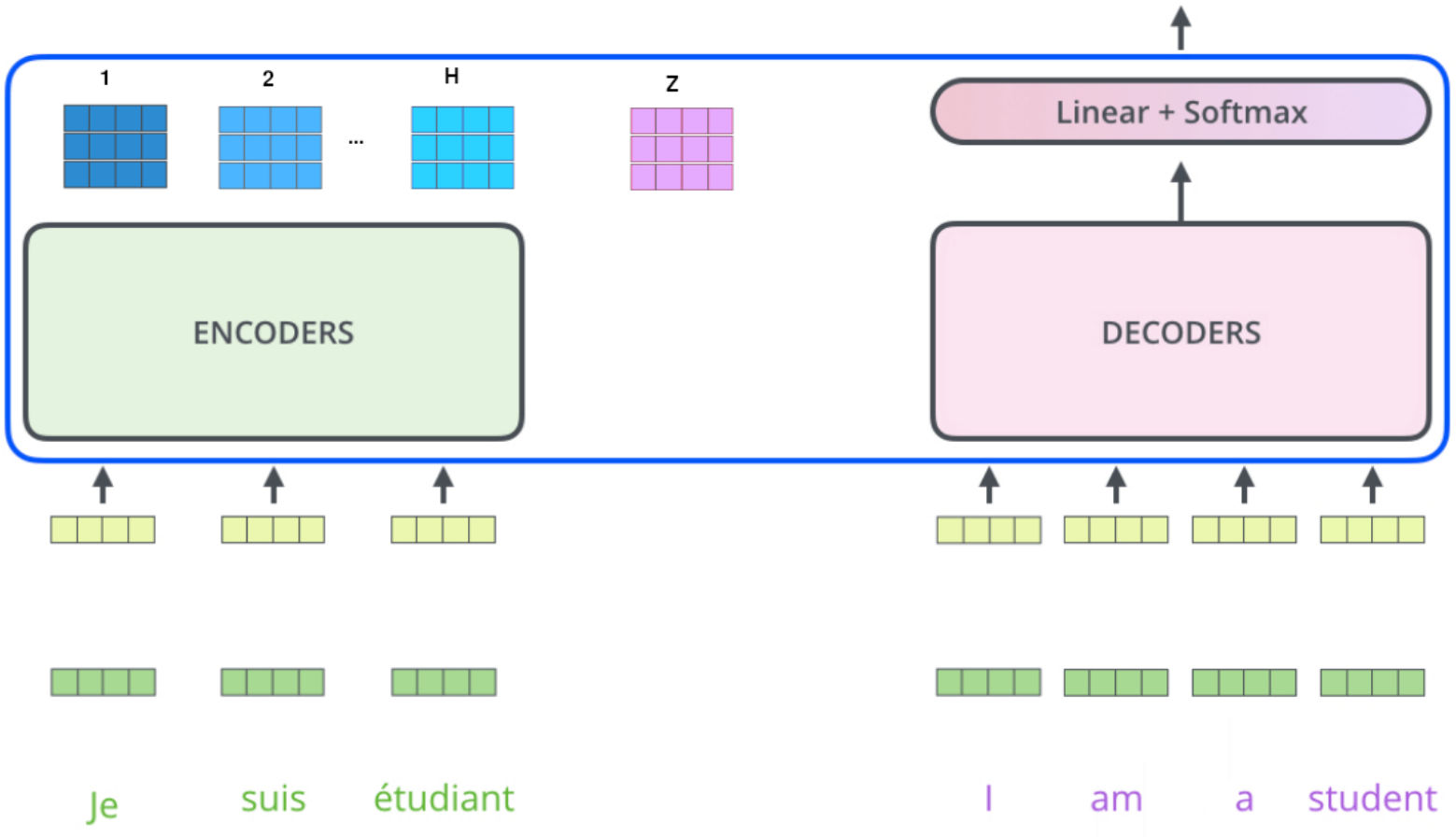
$$maskedattention(Q, K, V) = softmax\left(\frac{Q^T K + M}{\sqrt{d_k}}\right) V$$

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

Representações para cada palavra

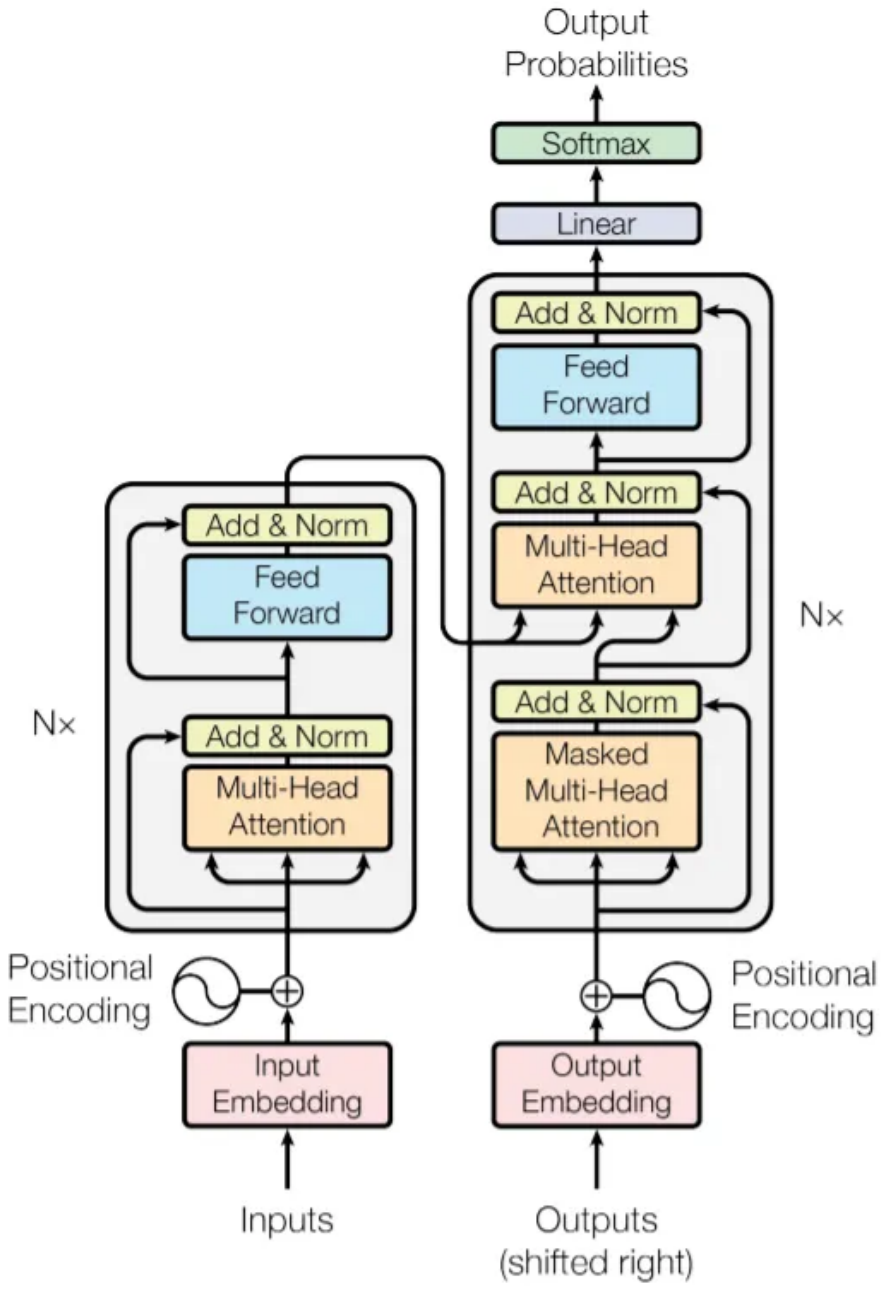
1. Word embeddings: w
2. com positional encoding: $\mathrm{w} + \mathrm{p} = \mathrm{e}$
3. Query: $\mathrm{Q}_h = W_Q \cdot \mathrm{e}_h$
4. Key: $\mathrm{K}_h = W_K \cdot \mathrm{e}_h$
5. Value: $\mathrm{V}_h = W_V \cdot \mathrm{e}_h$
6. Attention heads: $Z_h = softmax\left(\frac{Q_h^T K_h + M}{\sqrt{d_k}}\right)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

Recorrência via "Teacher forcing" no treinamento



Adaptado de Jay Alammar

Arquitetura Transformer



Arquitetura Transformer: alguns resultados

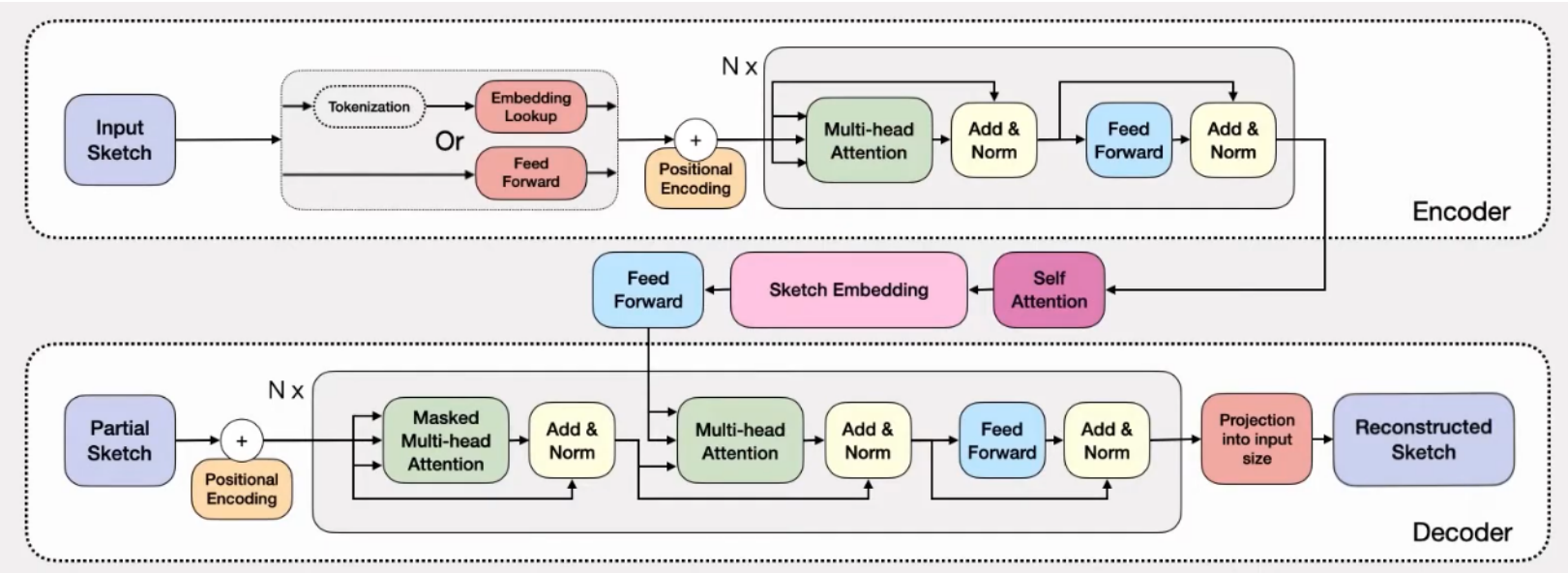
One day, I decided to bake a loaf of bread on my own oven. It was so simple and simple and easy. This bread is my first time baking it in my oven. I don't know what you will have to do to make this loaf.

`https://transformer.huggingface.co/`

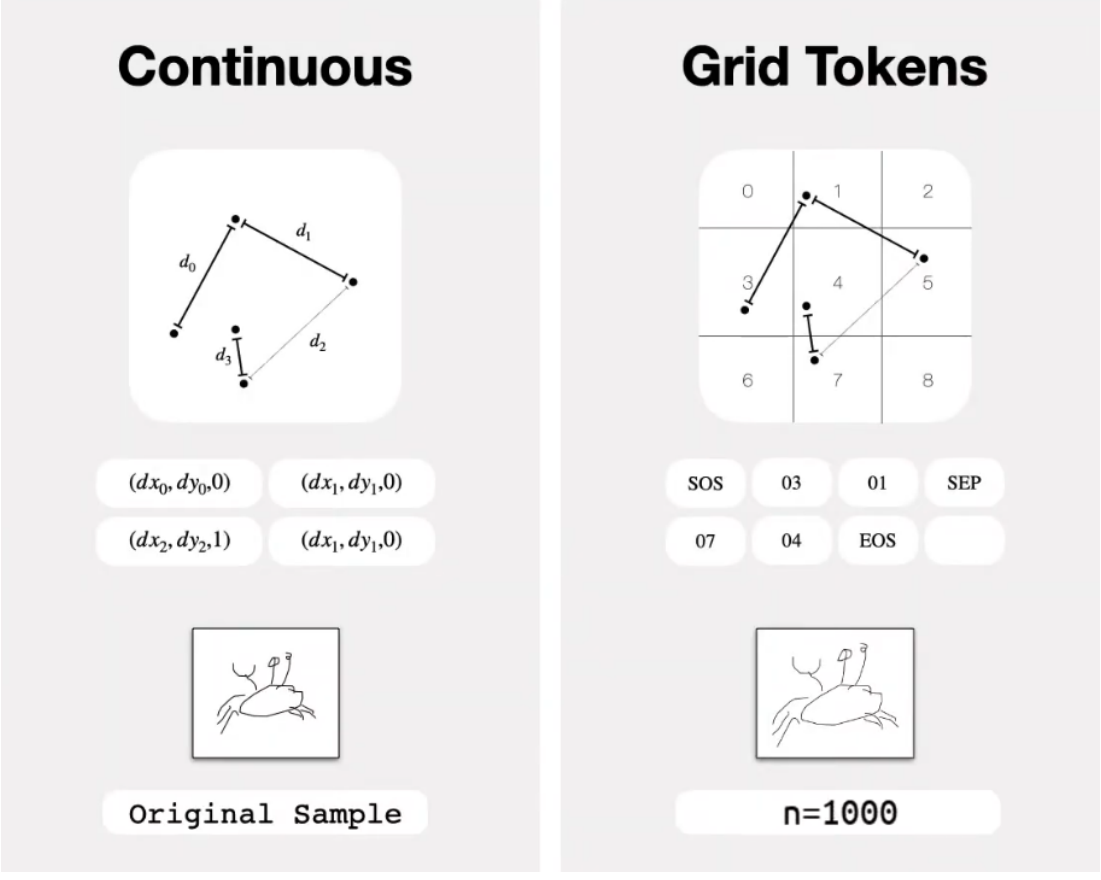
Após Transformer

- BERT
- GPT, GPT2, GPT3

Outras aplicações: para dados visuais



Outras aplicações: para dados visuais



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/`