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

Agenda

Word2Vec: representações para texto

Sequence-to-Sequence e Mecanismo de Atenção

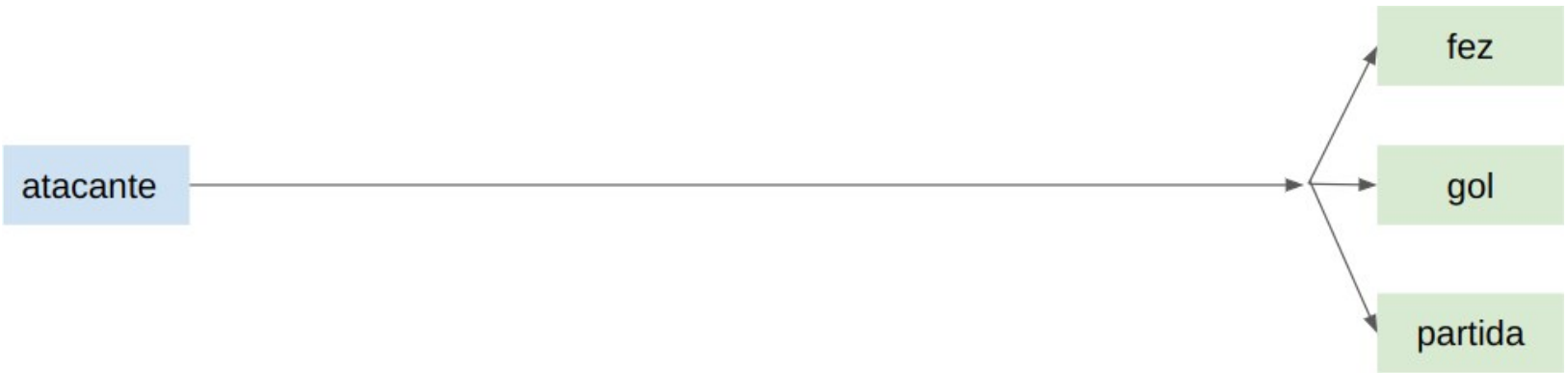
Word2Vec

- 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 outras palavras no contexto de t
- Otimiza em função de palavras que devem estar próximas se estiverem no mesmo contexto

CBOW



Skipgram

Word2Vec: skipgrams

Skip-grams (SG)

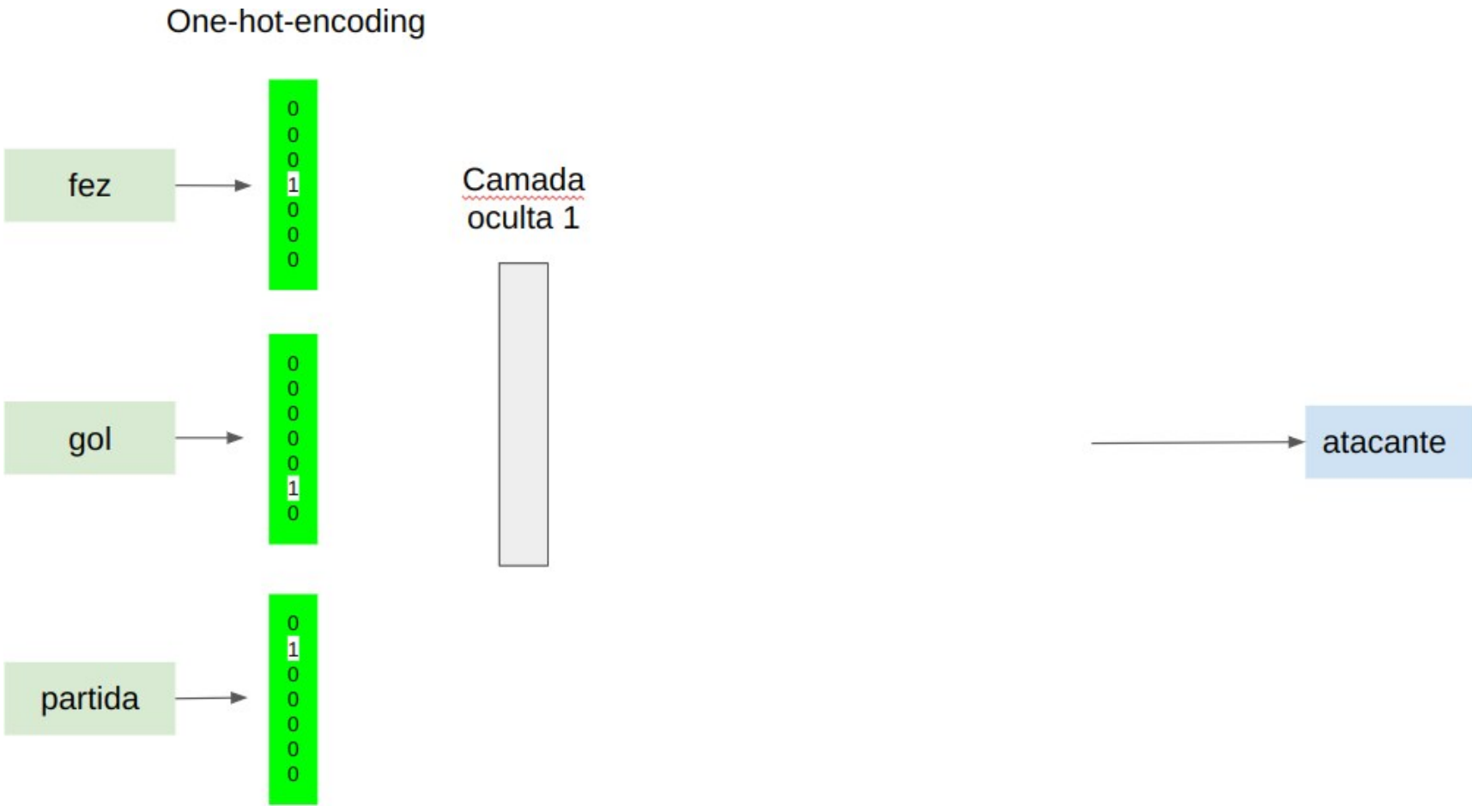
Predição de palavras em uma certa "janela" de proximidade m de uma palavra t

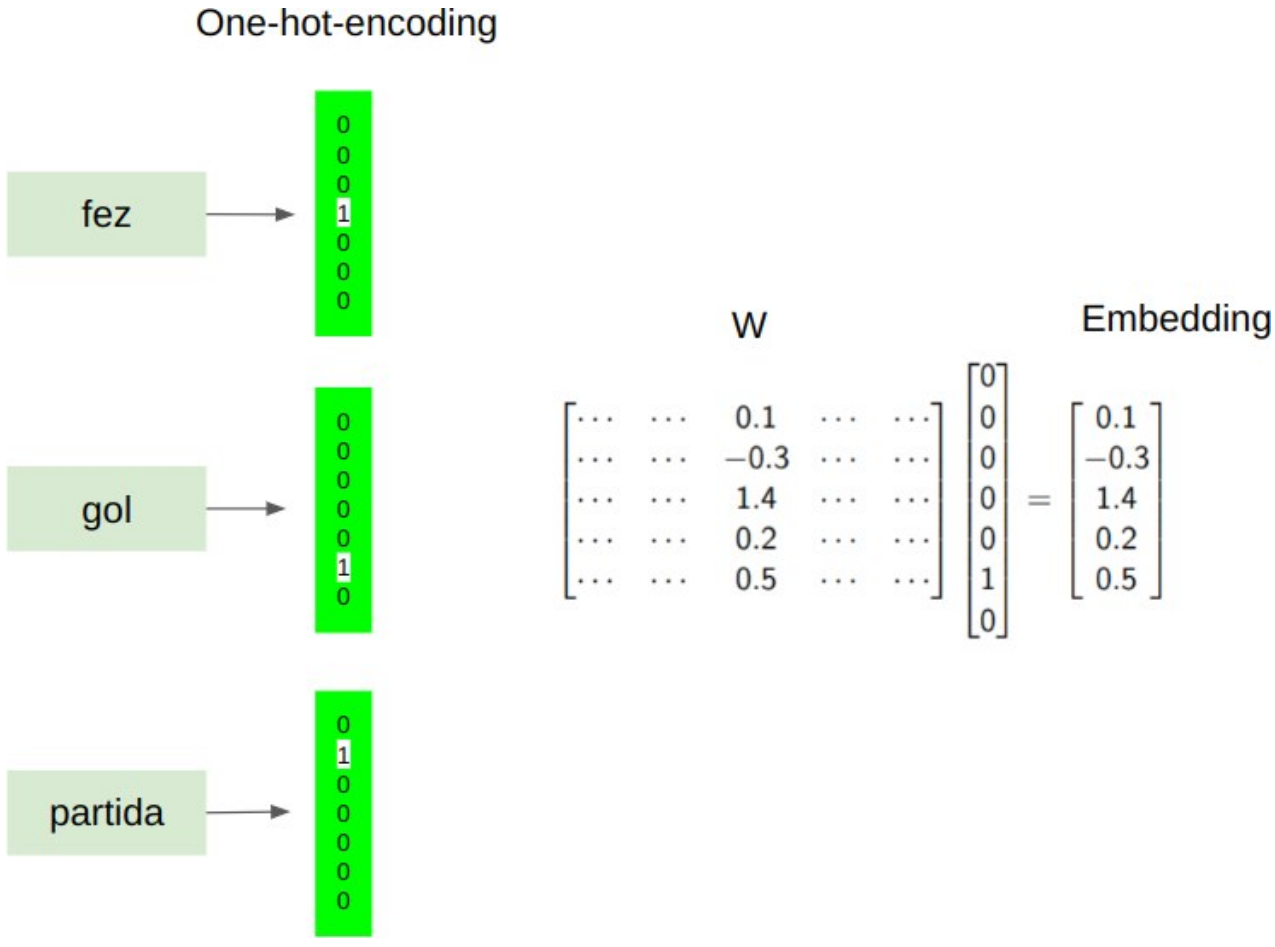
- Formulação "softmax":

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

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)


Token numérico
fez
04
gol
06
partida
02
atacante


One-hot-encoding
fez
0 0 0 1 0 0 0
gol
0 0 0 0 0 1 0
partida
0 1 0 0 0 0 0
Camada oculta 1
atacante

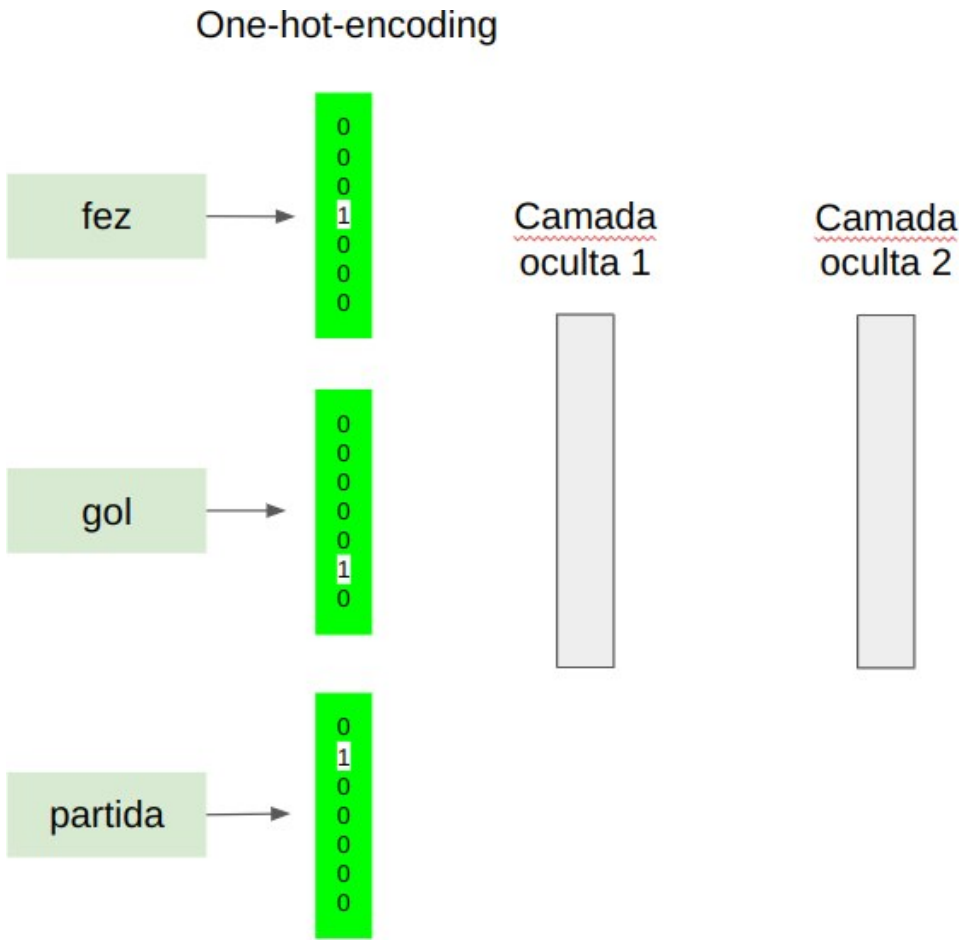

One-hot-encoding
fez
0 0 0 1 0 0 0
gol
0 0 0 0 0 1 0
partida
0 1 0 0 0 0 0
W
Embedding
$\begin{bmatrix} \cdots & \cdots & 0.1 & \cdots & \cdots \\ \cdots & \cdots & -0.3 & \cdots & \cdots \\ \cdots & \cdots & 1.4 & \cdots & \cdots \\ \cdots & \cdots & 0.2 & \cdots & \cdots \\ \cdots & \cdots & 0.5 & \cdots & \cdots \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.3 \\ 1.4 \\ 0.2 \\ 0.5 \end{bmatrix}$

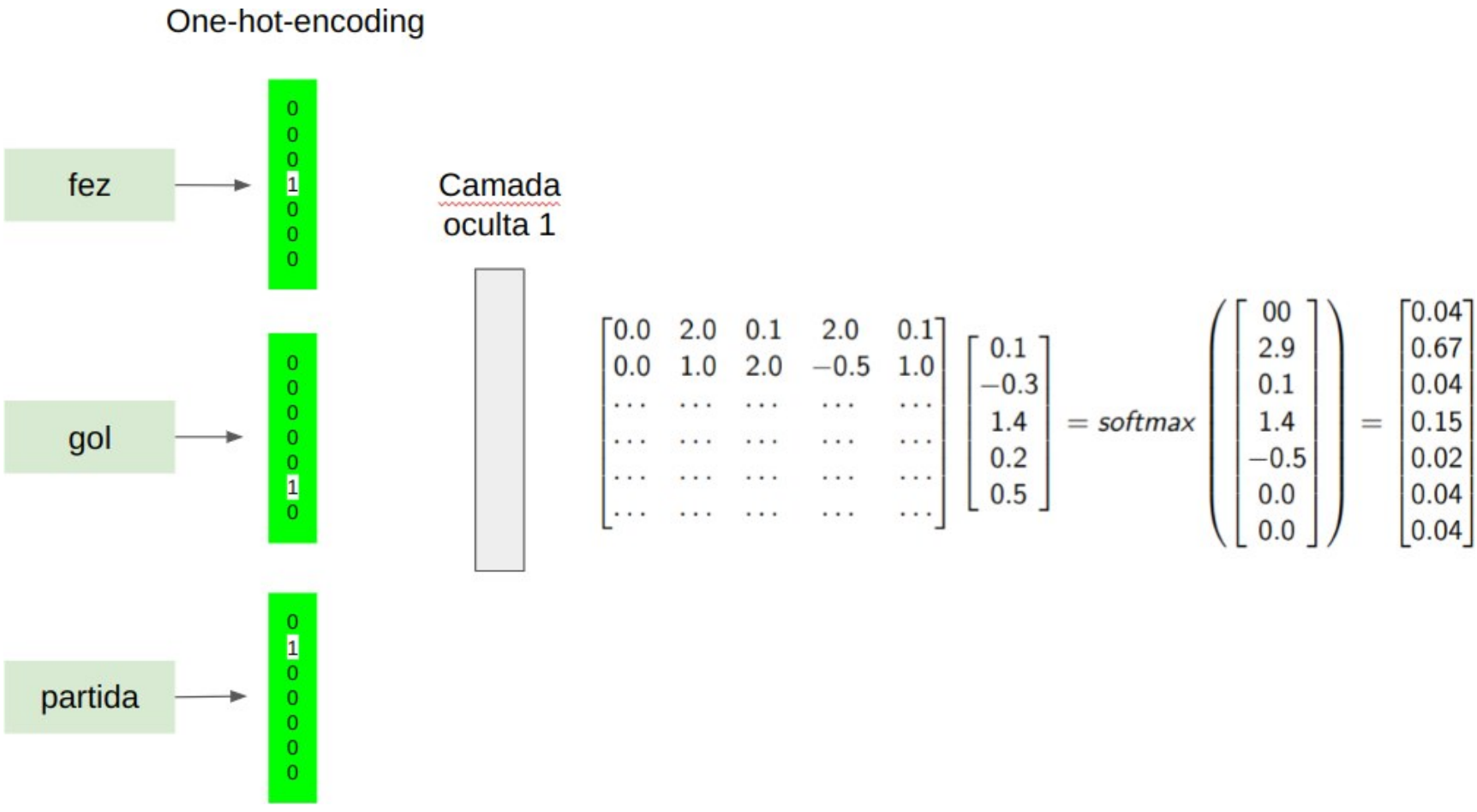
Word2Vec: skipgram

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

$$W \cdot w_t = v_c$$

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


One-hot-encoding
fez
0
0
0
1
0
0
0
gol
0
0
0
0
0
1
0
partida
0
1
0
0
0
0
0
Camada oculta 1
Camada oculta 2


One-hot-encoding
fez
gol
partida
Camada oculta 1
softmax

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"

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

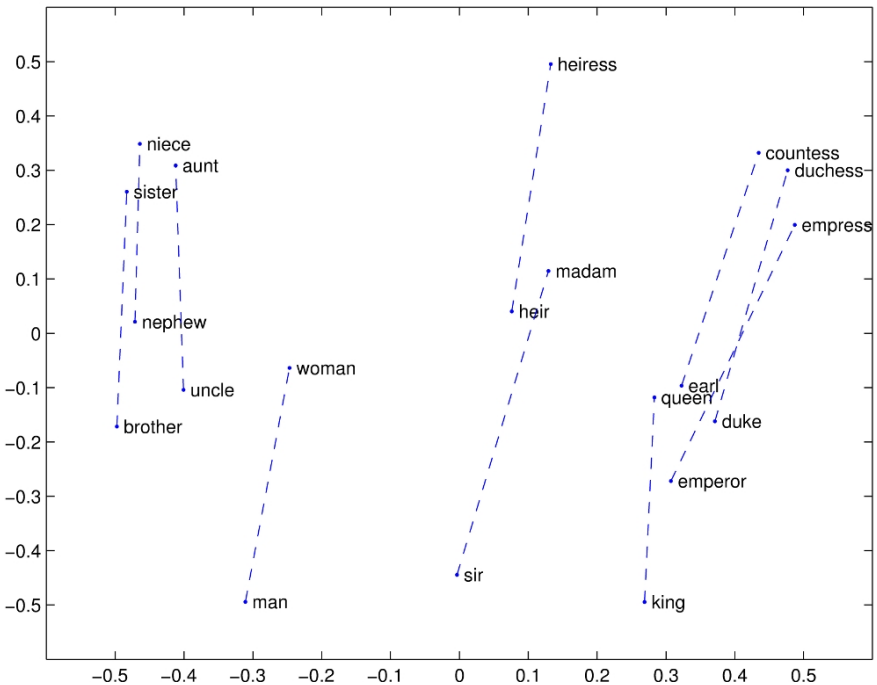
Word2Vec: skipgram

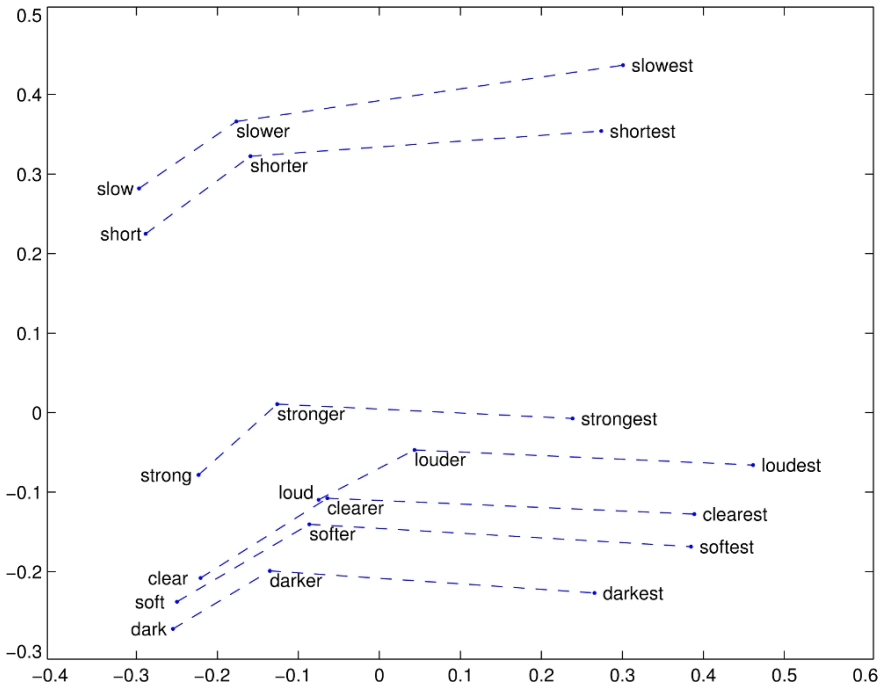
- Palavras que aparecem num mesmo contexto terão representações similares

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

RNNs e Sequence-to-Sequence (seq2seq)

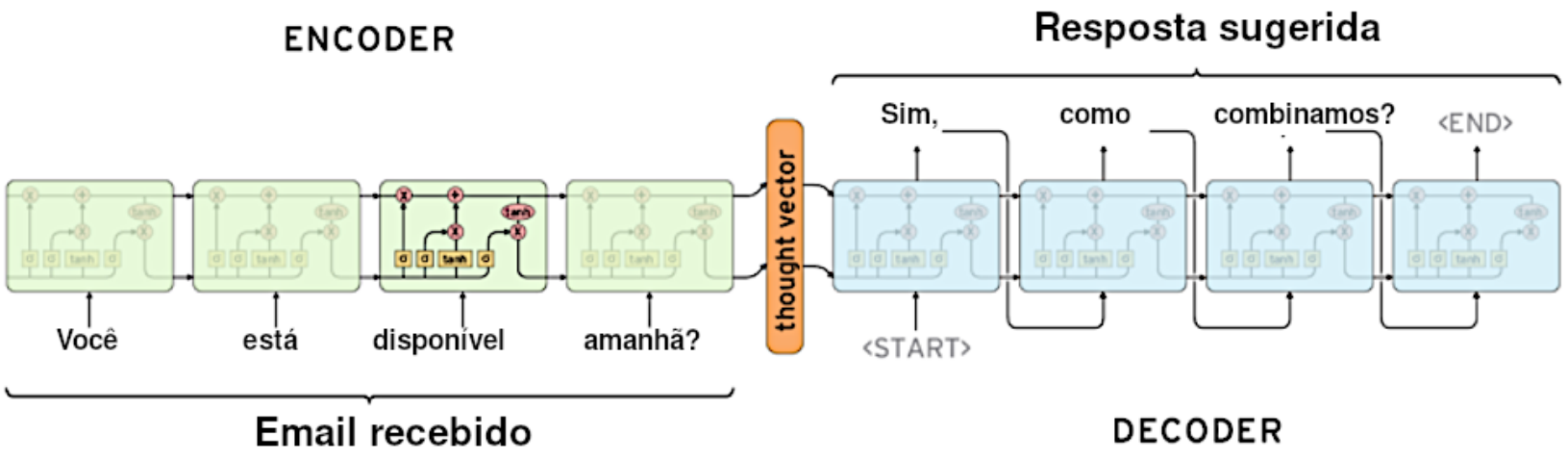


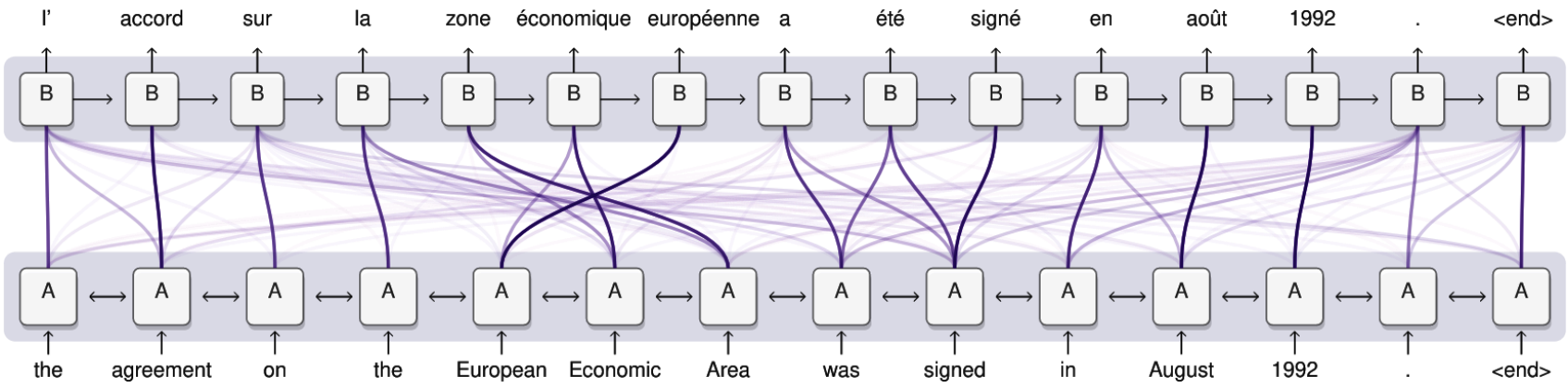
Figura adaptada de: Sachin Abeywardana

- (3 vídeos de Jay Alammar)

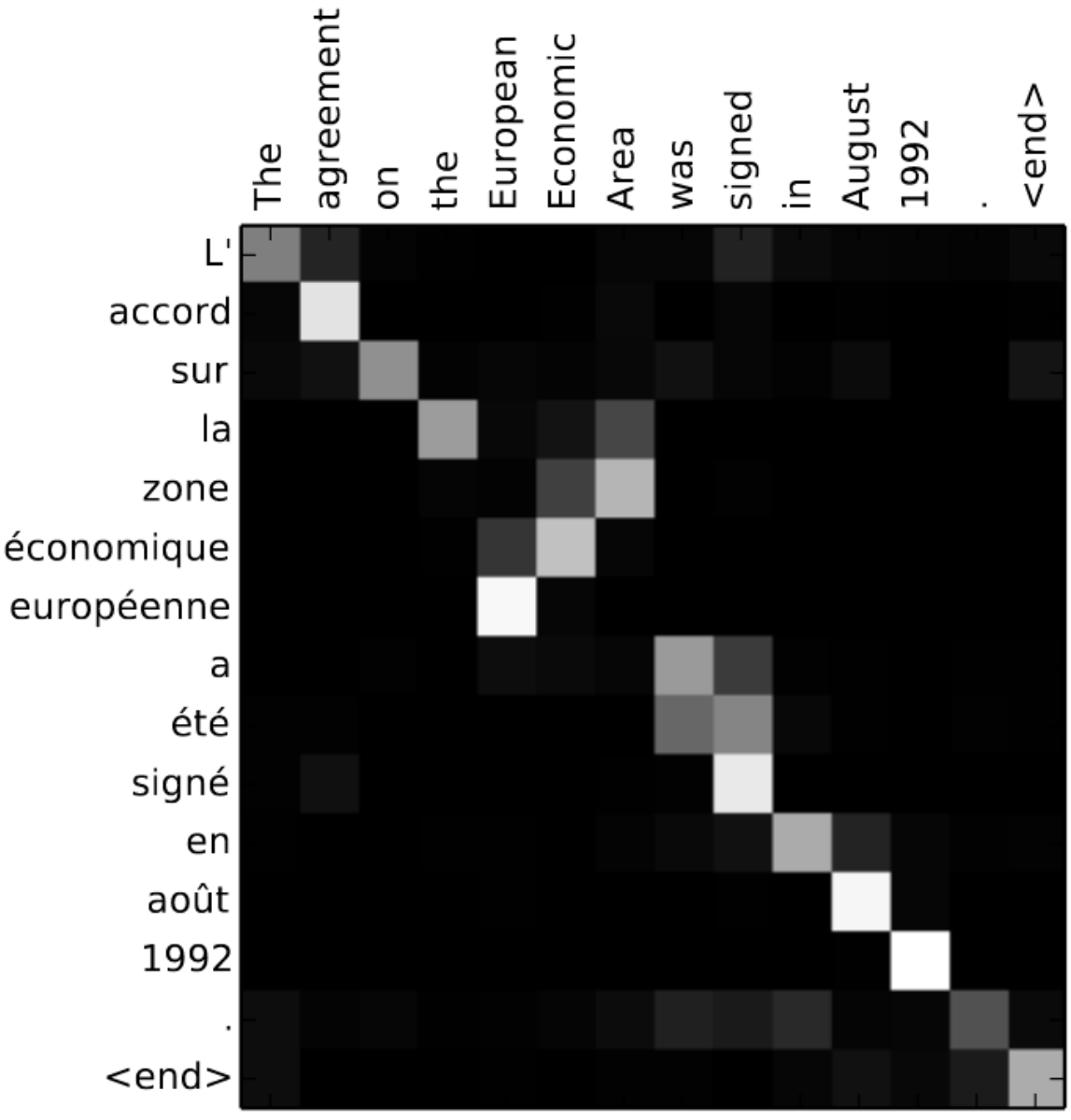
Mecanismo de atenção: intuição e motivação

- 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

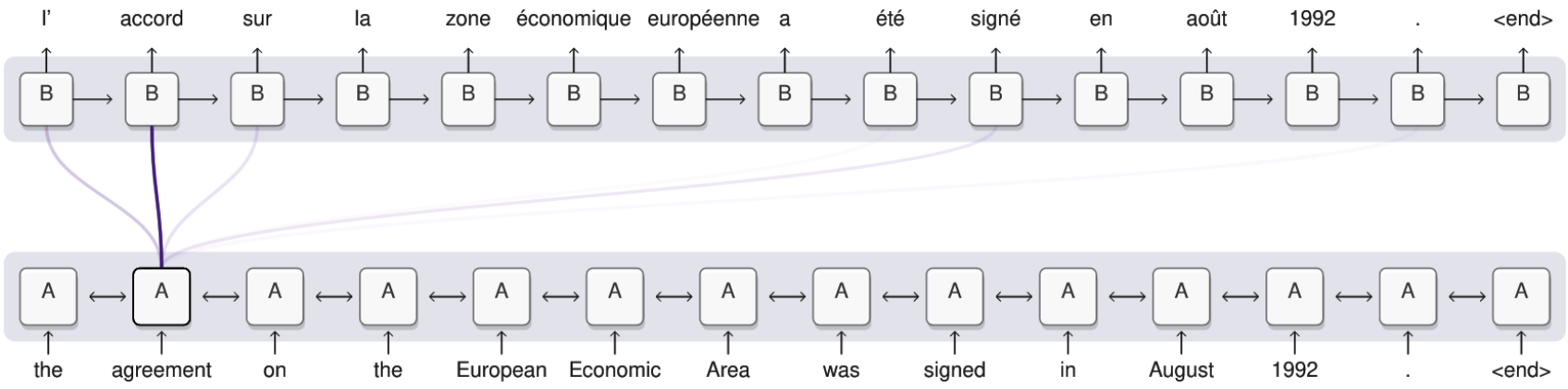
Mecanismo de Atenção: texto



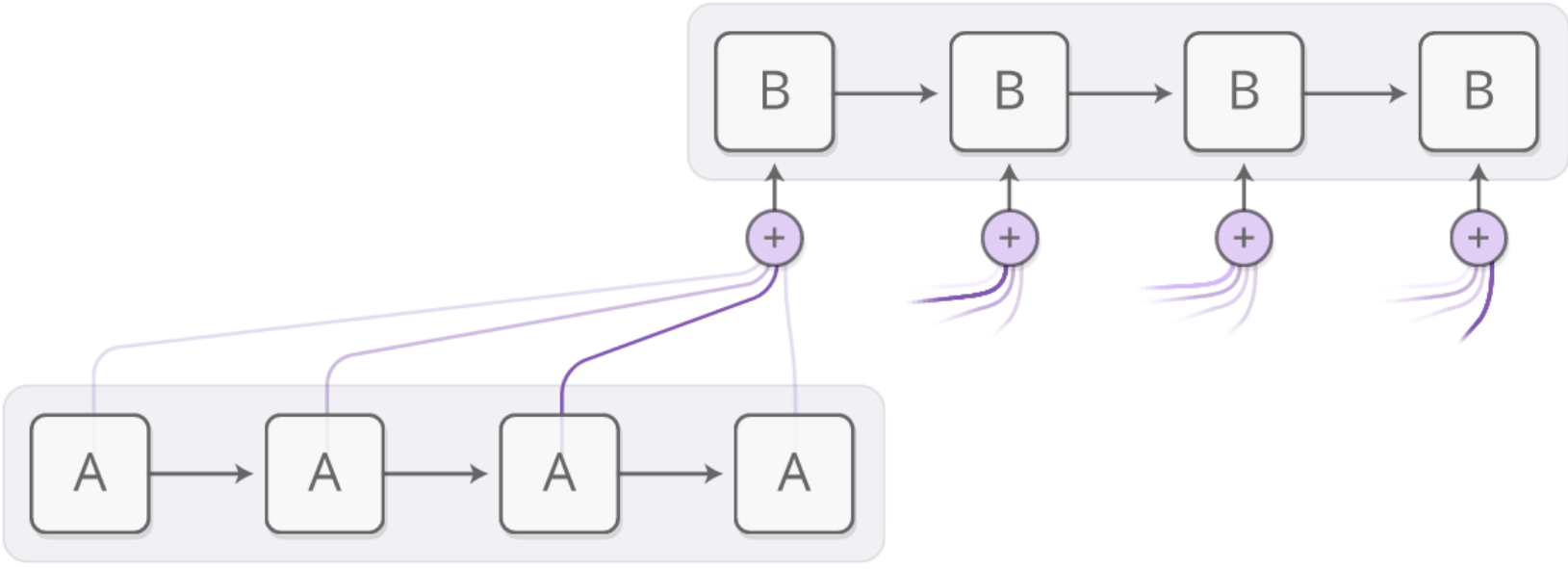
Mecanismo de Atenção: texto



Mecanismo de Atenção: texto



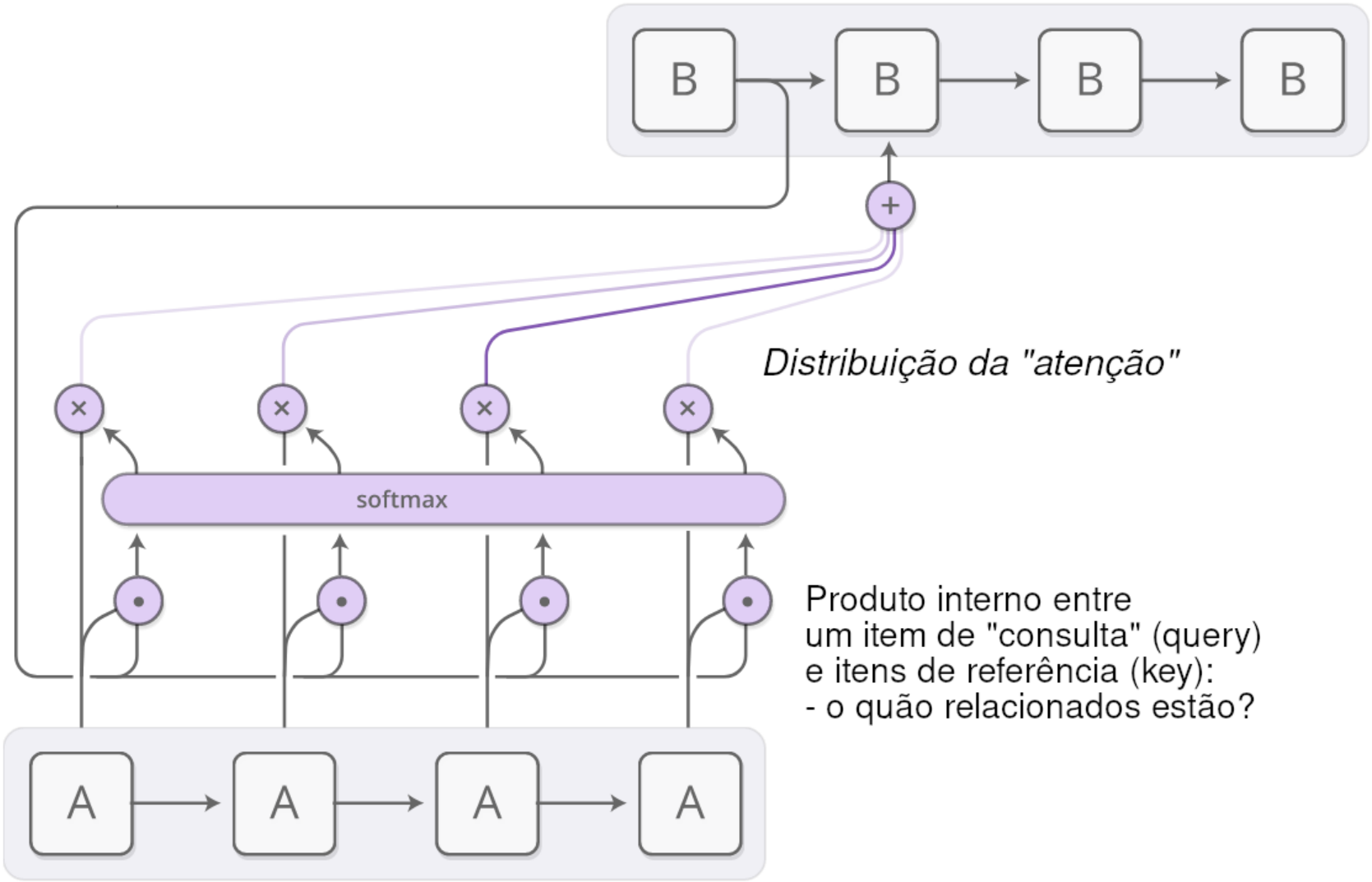
RNNs seq2seq



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

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

RNNs seq2seq e atenção



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.

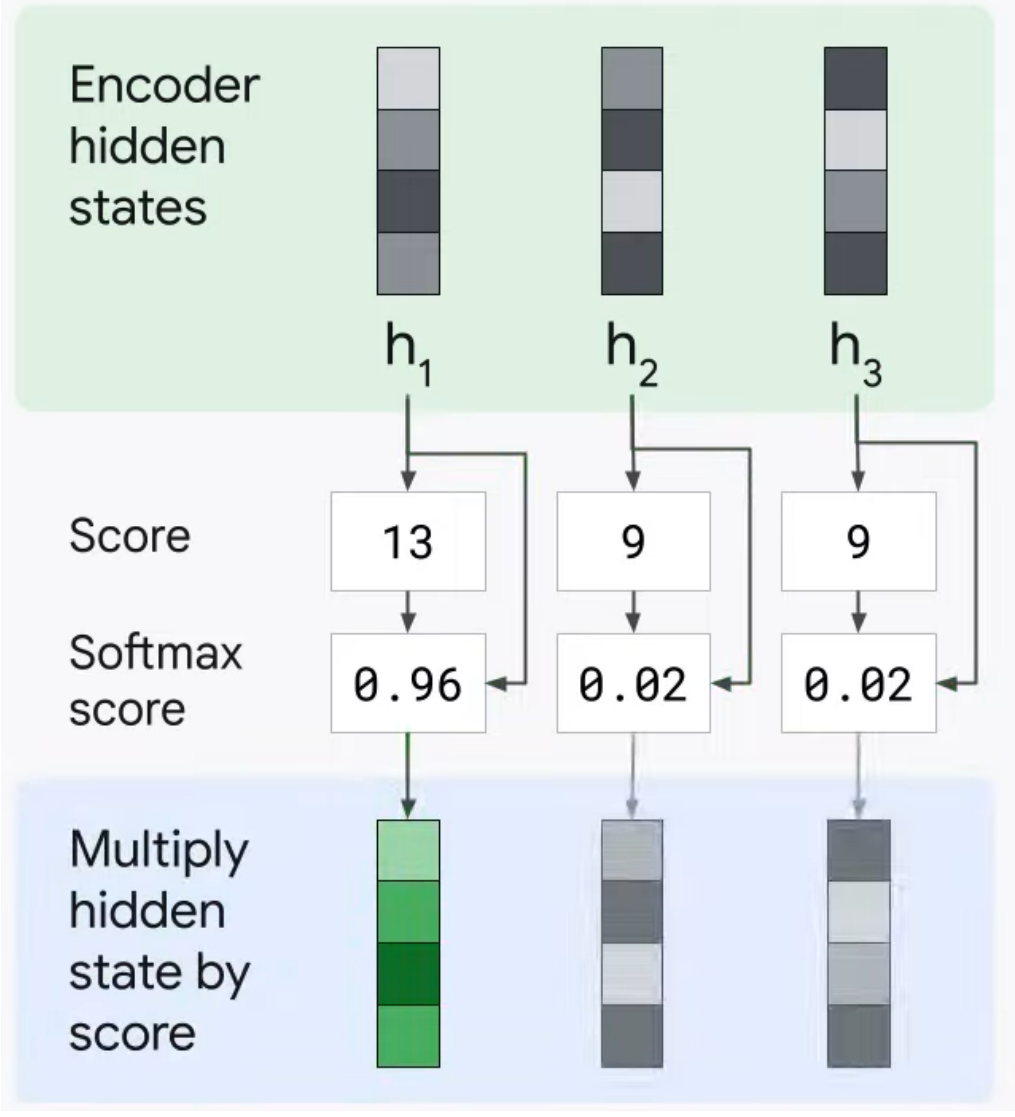
Mecanismo de atenção query/key -> value

- 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

Mecanismo de Atenção



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/`