MAC5921 – Deep Learning

Aula 11 – 26/09/2023

RNN

Nina S. T. Hirata

Redes neurais recorrentes

Usadas para dados cujos elementos são organizados em uma estrutura sequencial

- série temporal
- sequência de genes
- áudio
- vídeo
- texto

Redes neurais convolucionais

- explora o fato de um mesmo padrão poder ocorrer em qualquer parte de uma imagem
- convolução considera uma vizinhança restrita; múltiplas camadas para "ampliar" a vizinhança abarcada — relação espacial
- o compartilhamento de pesos (entre diferentes posições na imagem) permite a aplicação em imagens de variados tamanhos

Redes neurais recorrentes

- explora o fato de um mesmo padrão poder ocorrer em qualquer parte de uma sequência
- recorrência considera todos os elementos prévios na sequência; não considera elementos posteriores na sequência — relação sequencial
- o compartilhamento de pesos permite a aplicação em sequências de variados tamanhos

Entrada:

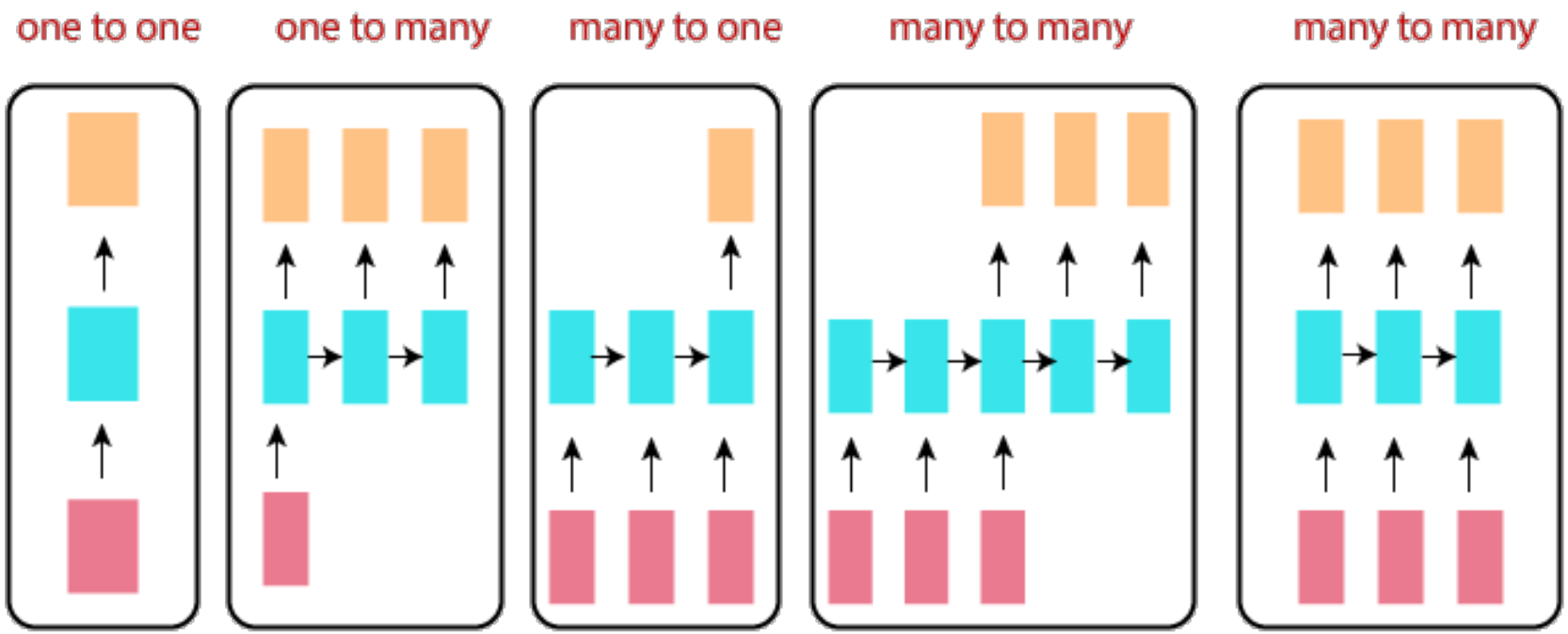
Sequência de elementos; elementos são processados um a um

$\mathbf{x} = x_1, x_2, \ldots, x_t, \ldots, x_T$

Saída pode ser:

- scores para rótulos de classe (problema de classificação)
- sequência de elementos (por exemplo, tradução)
- scores ou valores para próximo elemento (predição de próximo elemento)
- ...

Diferentes formatos de saída e formas de mapear entrada para saída



https://www.javatpoint.com/tensorflow-types-of-rnn

t: índice na sequência

x_t: elemento na posição t da sequência de entrada $\mathbf{x}$

Um novo conceito:

Estado: $h_t = f(x_t, h_{t-1})$

Exemplo: $h_t = \tanh(U x_t + W h_{t-1})$

Saída: $o_t = V h_t, \quad \hat{y}_t = \mathrm{softmax}\Big(V h_t\Big)$

t: índice na sequência

x_t: elemento na posição t da sequência de entrada $\mathbf{x}$

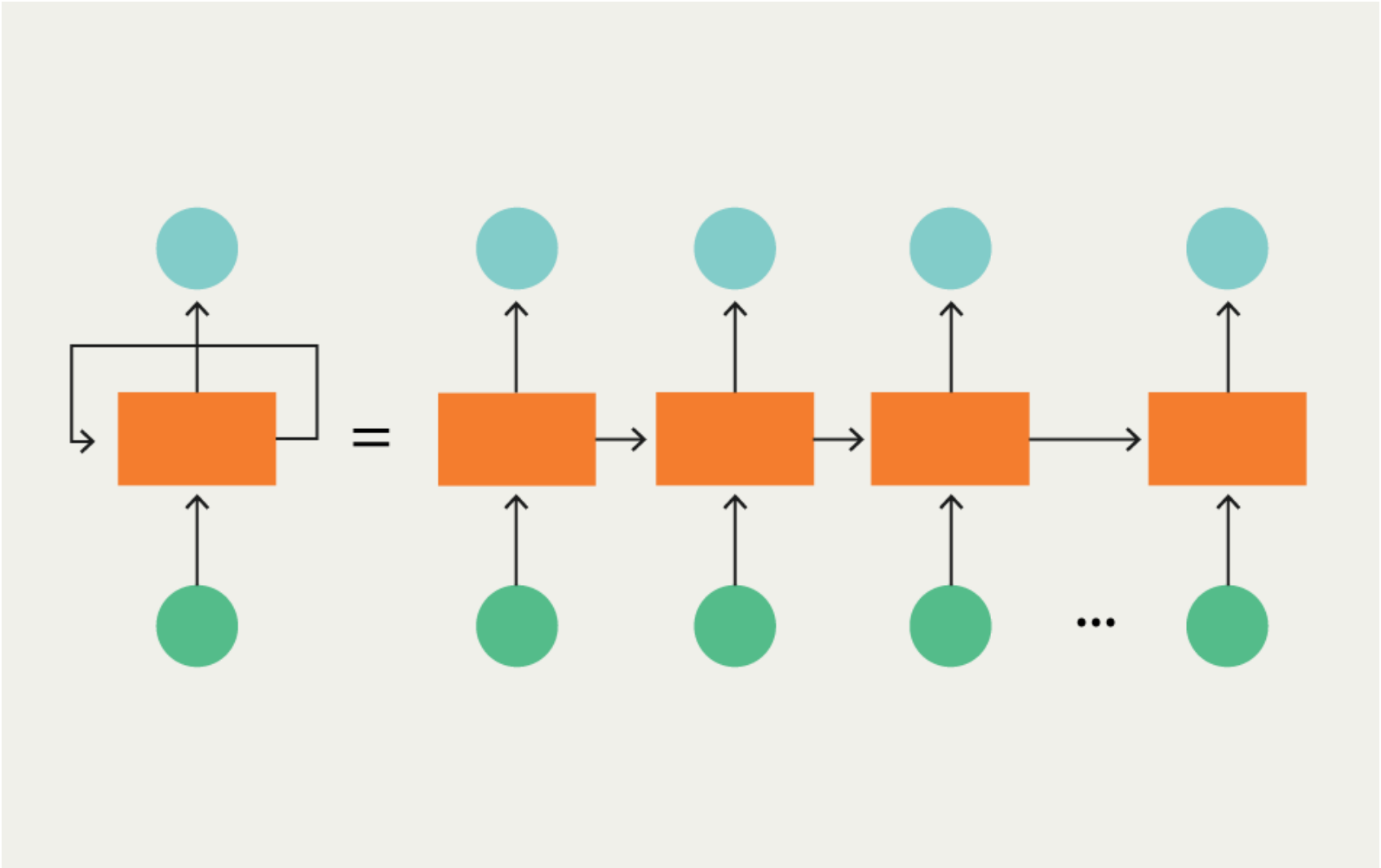
Um novo conceito:

Estado: $h_t = f(x_t, h_{t-1})$

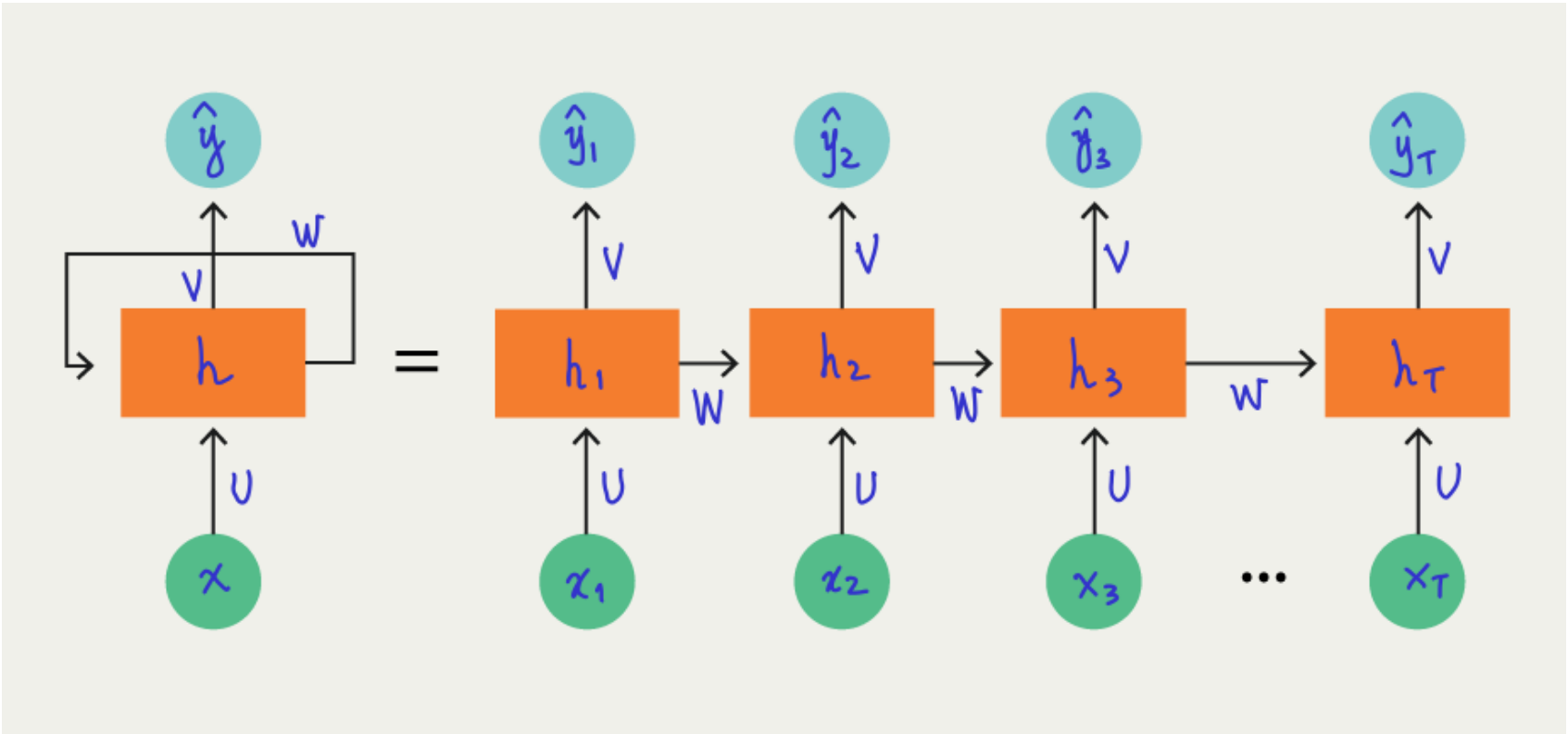
Exemplo: $h_t = \tanh(U x_t + W h_{t-1})$

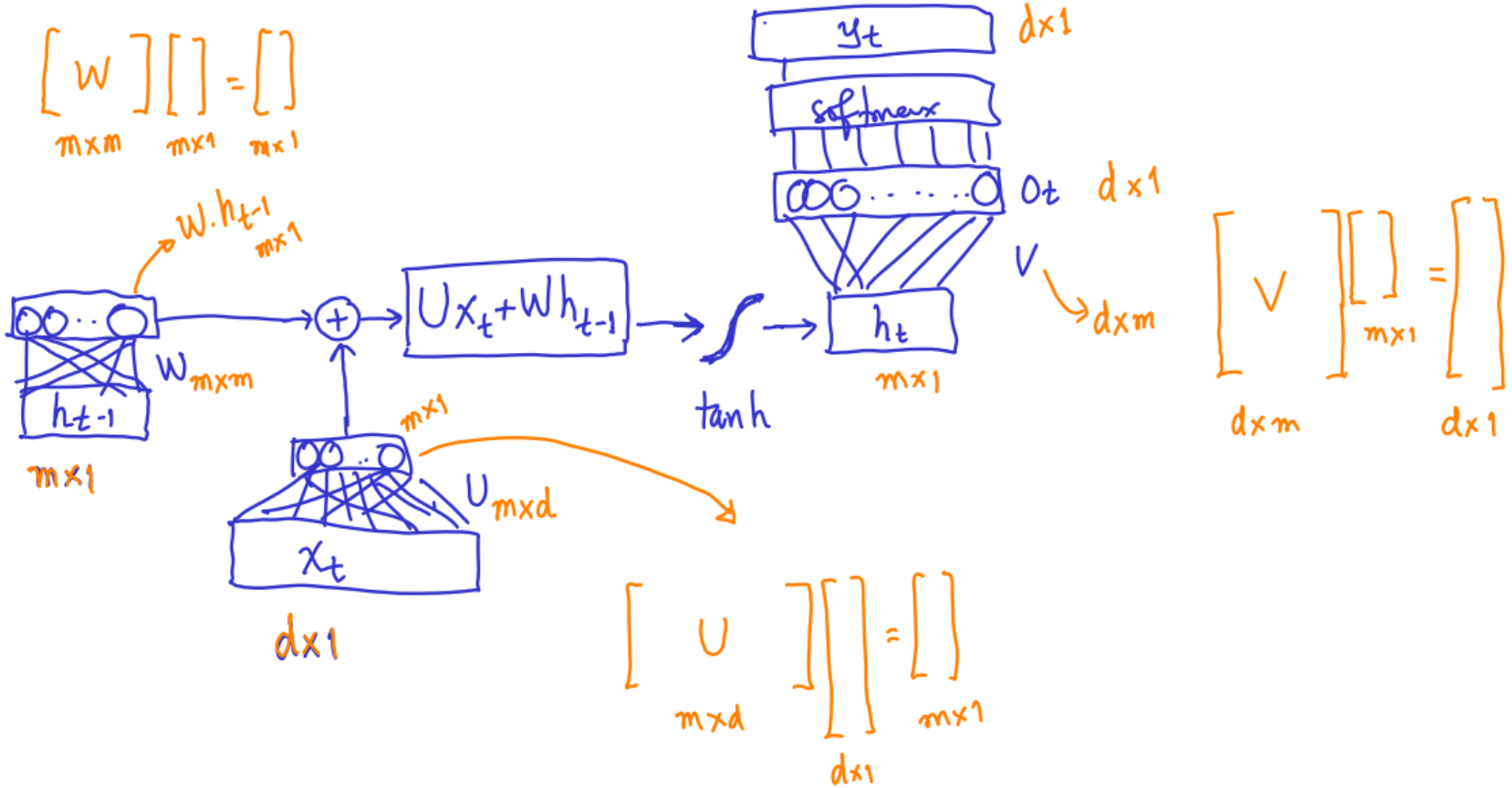
Saída: $o_t = V h_t, \quad \hat{y}_t = \mathrm{softmax}\left(V h_t\right)$

U, W e V são matrizes de peso, compartilhados para todos os t



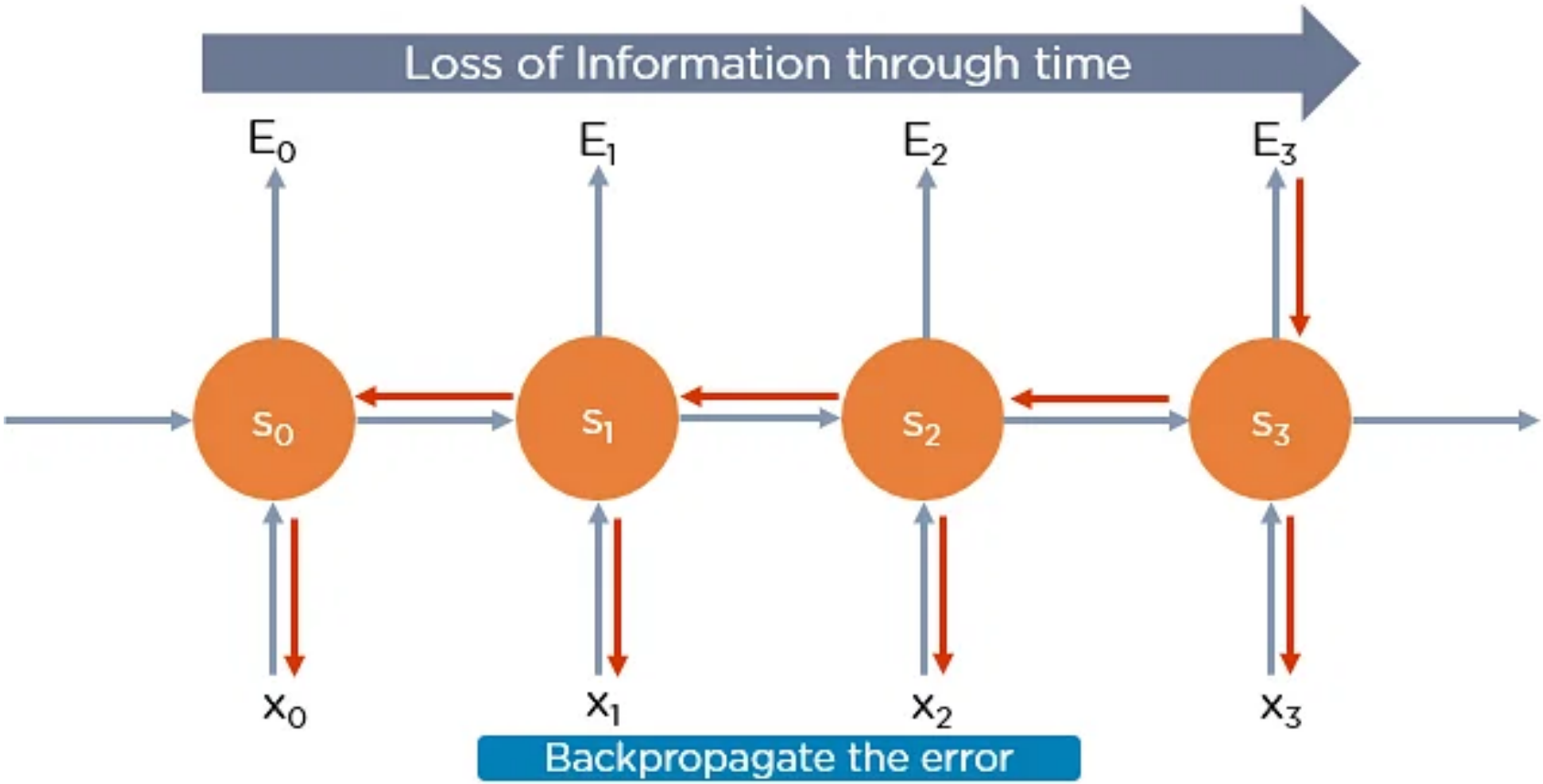
Fonte: https://www.telusinternational.com/articles/difference-between-cnn-and-rnn


ŷ
W
V
h
U
x
=
ŷ1
ŷ2
ŷ3
ŷT
h1
h2
h3
hT
x1
x2
x3
xT
...

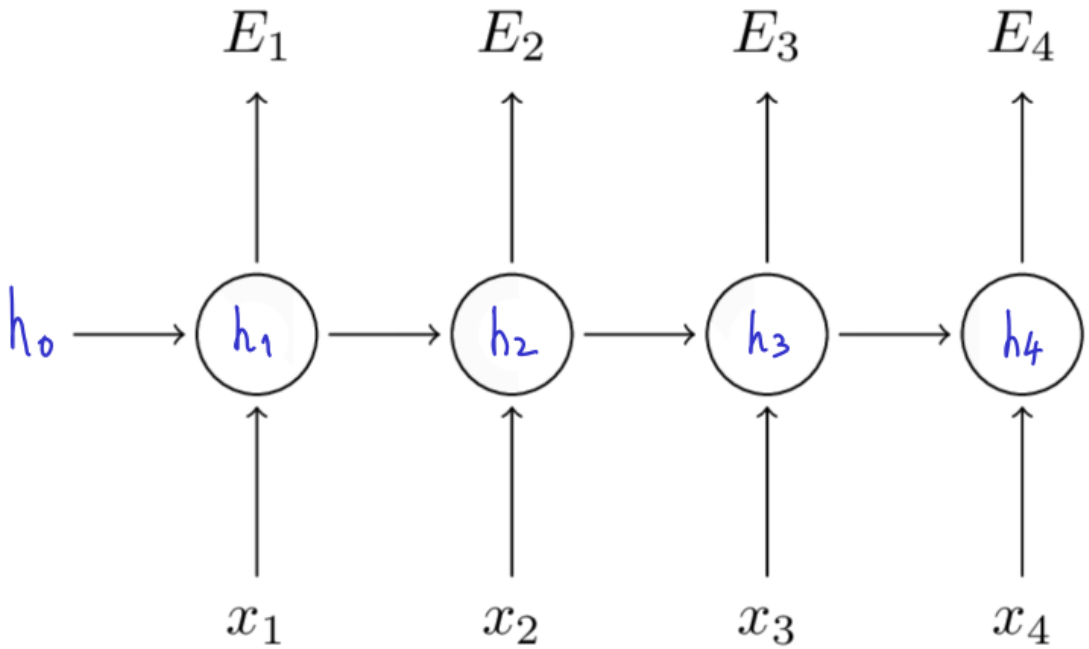


Aqui, podemos pensar que $\mathtt{d}$ é o tamanho do vocabulário (*one-hot-encoding*)

Forward e Backward pass em RNNs



Fonte: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn


E_1
E_2
E_3
E_4
h_0
h_1
h_2
h_3
h_4
x_1
x_2
x_3
x_4

Erro de predição na posição t (lembrar que y_t é one-hot-encoded target):

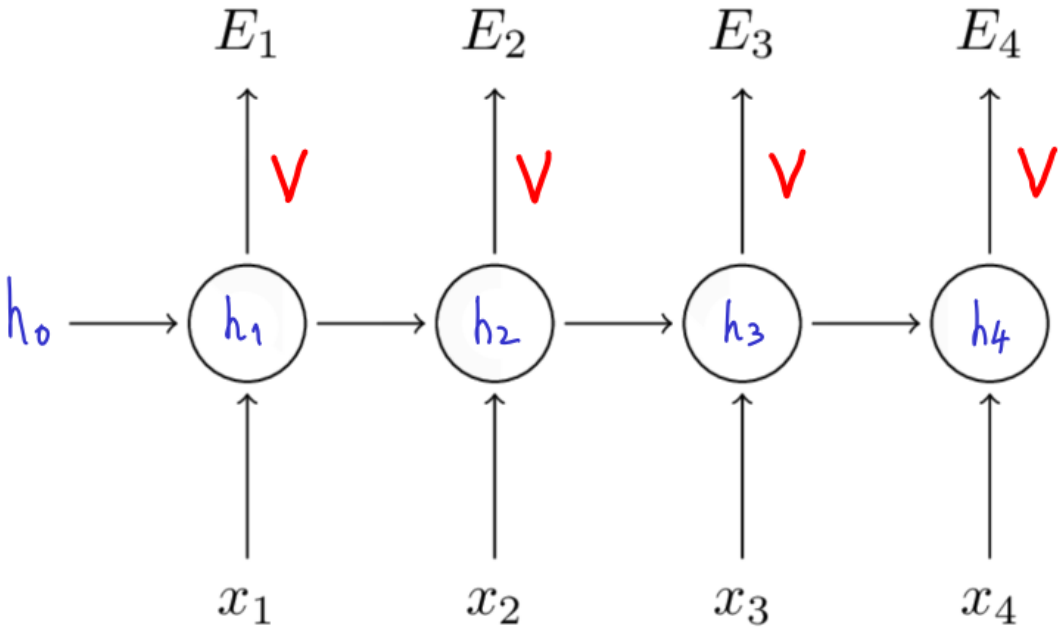
$$E_t(y_t, \hat{y}_t) = -\sum_c y_t \log(\hat{y}_t)$$

Total loss

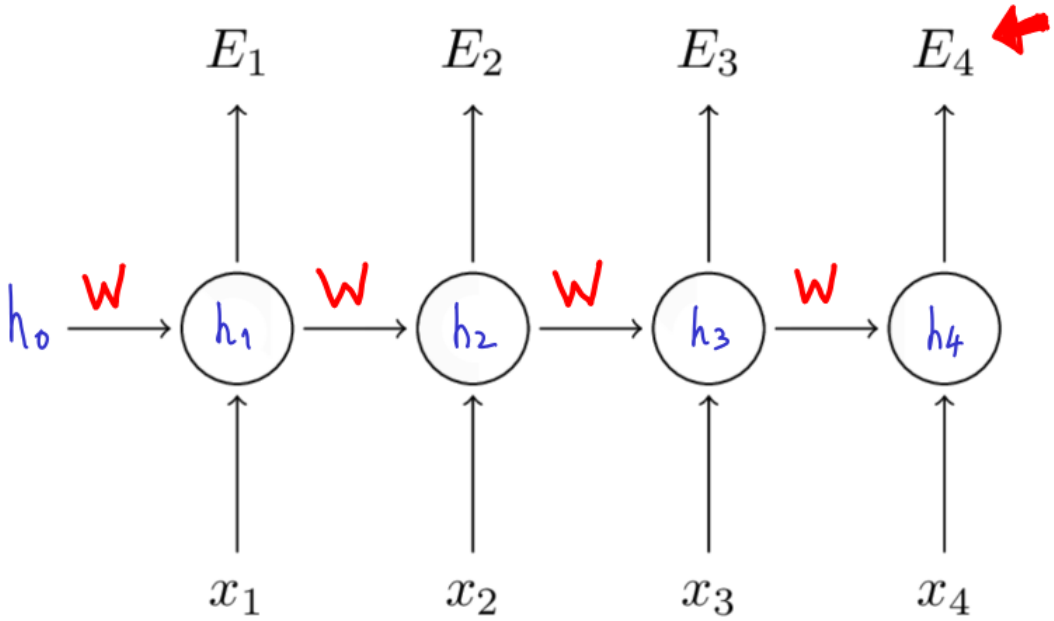
$$E(y, \hat{y}) = \sum_t E_t(y_t, \hat{y}_t)$$

Derivadas parciais que precisamos calcular

$$\frac{\partial E}{\partial U} = \sum_t \frac{\partial E_t}{\partial U} \qquad \frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W} \qquad \frac{\partial E}{\partial V} = \sum_t \frac{\partial E_t}{\partial V}$$



$$\frac{\partial E}{\partial V} = \sum_t \frac{\partial E_t}{\partial V}$$

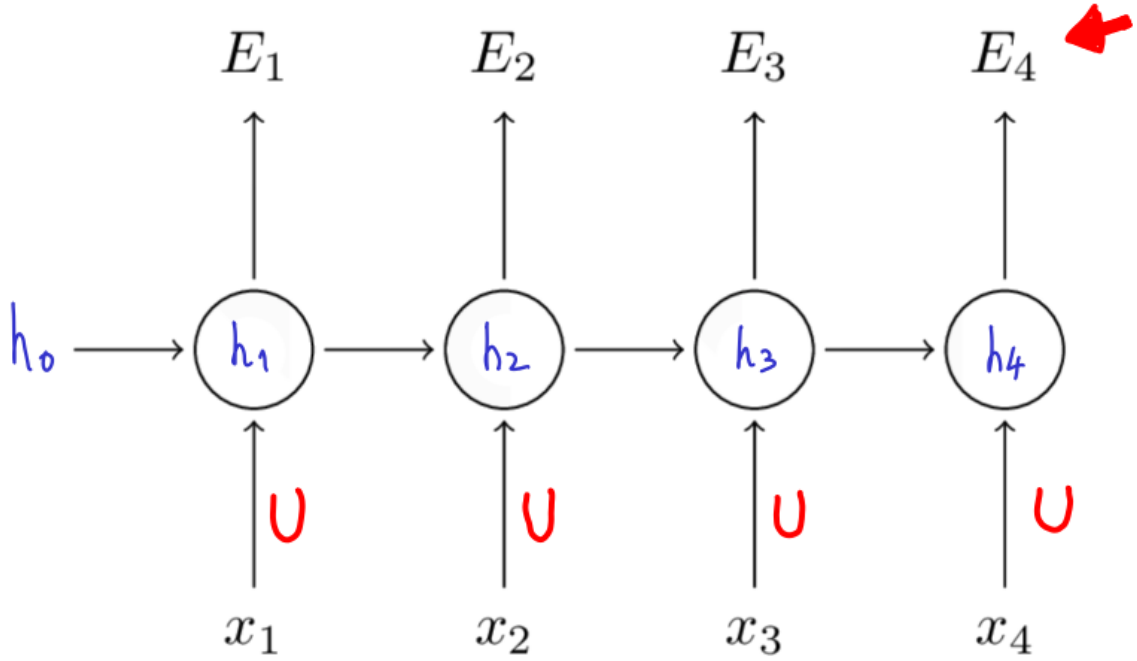


$$\frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\frac{\partial h_t}{\partial W} + \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial W} + \cdots + \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_1}\frac{\partial h_1}{\partial W}$$

$$\frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\frac{\partial h_t}{\partial W} + \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial W} + \ldots + \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_1}\frac{\partial h_1}{\partial W}$$

$$\begin{aligned}
\frac{\partial E_t}{\partial W} &= \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\frac{\partial h_t}{\partial h_t}\frac{\partial h_t}{\partial W} + \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial W} + \ldots + \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\frac{\partial h_t}{\partial h_1}\frac{\partial h_1}{\partial W} \\
&= \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\left[\frac{\partial h_t}{\partial h_t}\frac{\partial h_t}{\partial W} + \frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial W} + \ldots + \frac{\partial h_t}{\partial h_1}\frac{\partial h_1}{\partial W}\right] \\
&= \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\sum_{i=1}^{t}\frac{\partial h_t}{\partial h_i}\frac{\partial h_i}{\partial W}
\end{aligned}$$

Produto de gradientes pode levar a vanishing/exploding gradient



$$\frac{\partial E}{\partial U} = \sum_t \frac{\partial E_t}{\partial U}$$

Muito similar ao caso dos pesos W

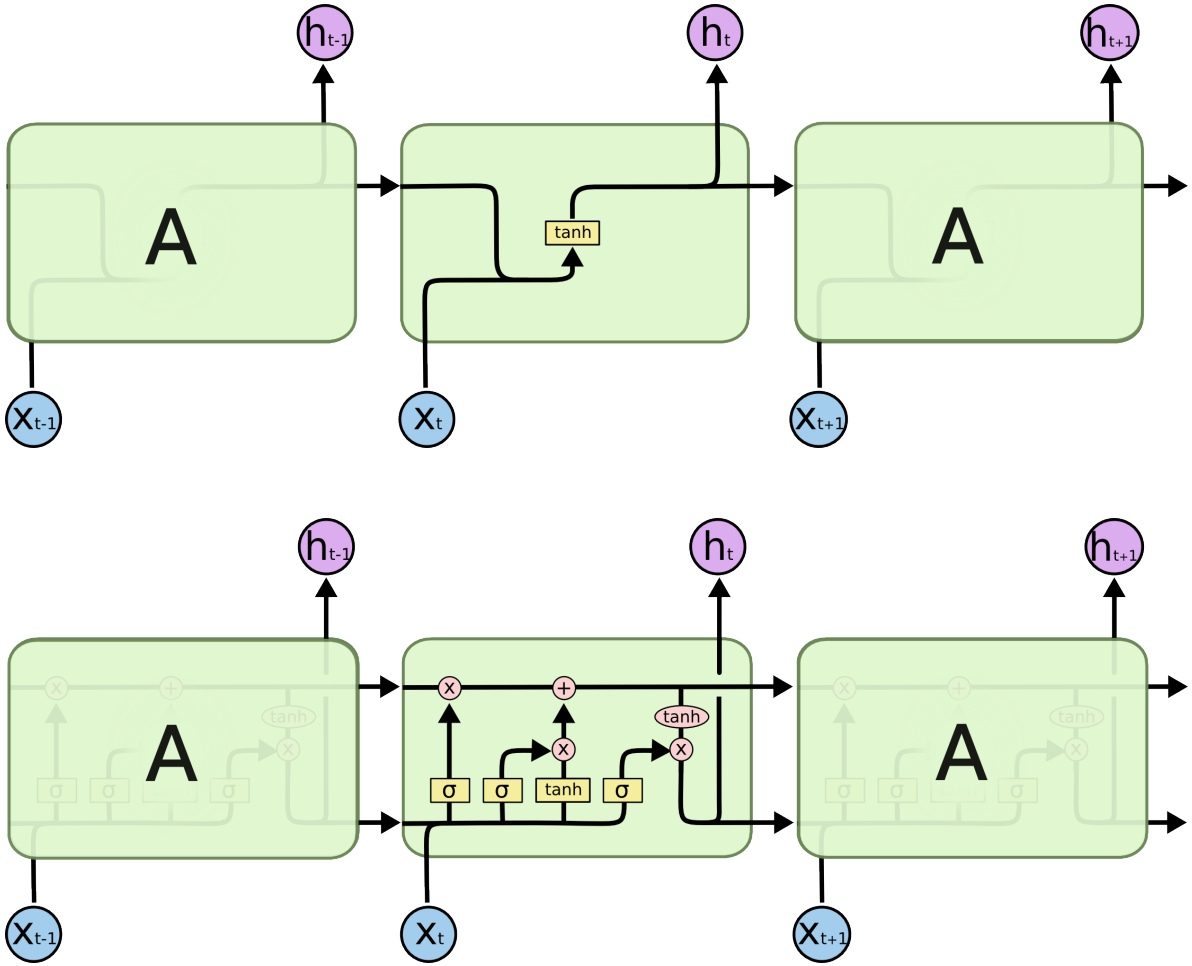
Exemplo de código (Python only):
https://gist.github.com/karpathy/d4dee566867f8291f086

Comentário detalhado sobre o código:
https://mkffl.github.io/2019/07/08/minimalist-RNN.html

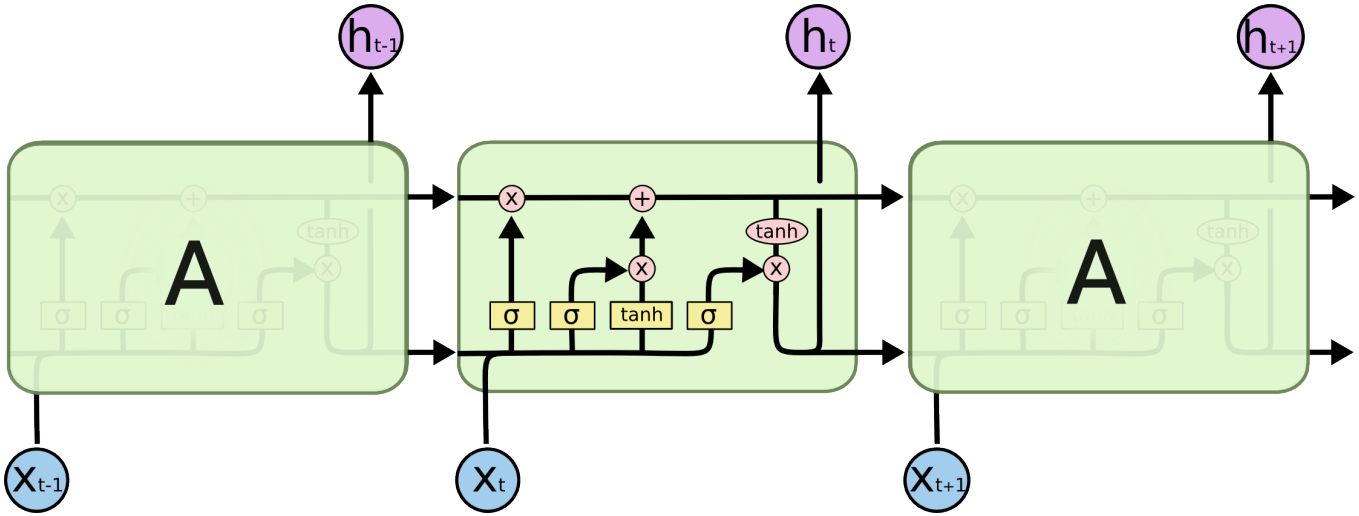
A estrutura da rede pode ser bem mais complexa

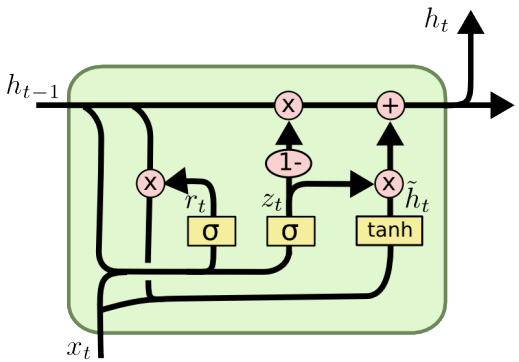
Por exemplo, múltiplas camadas ocultas

Padrão e LSTM



LSTM e GRU





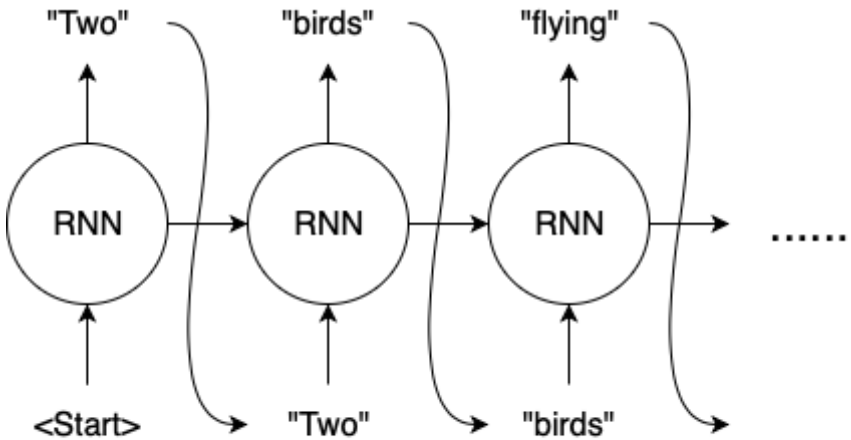
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

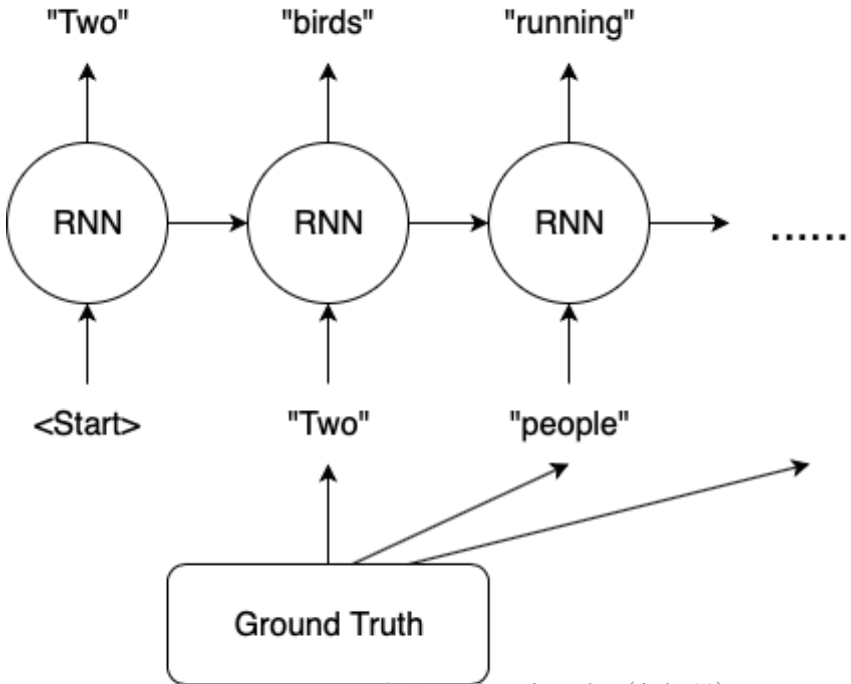
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

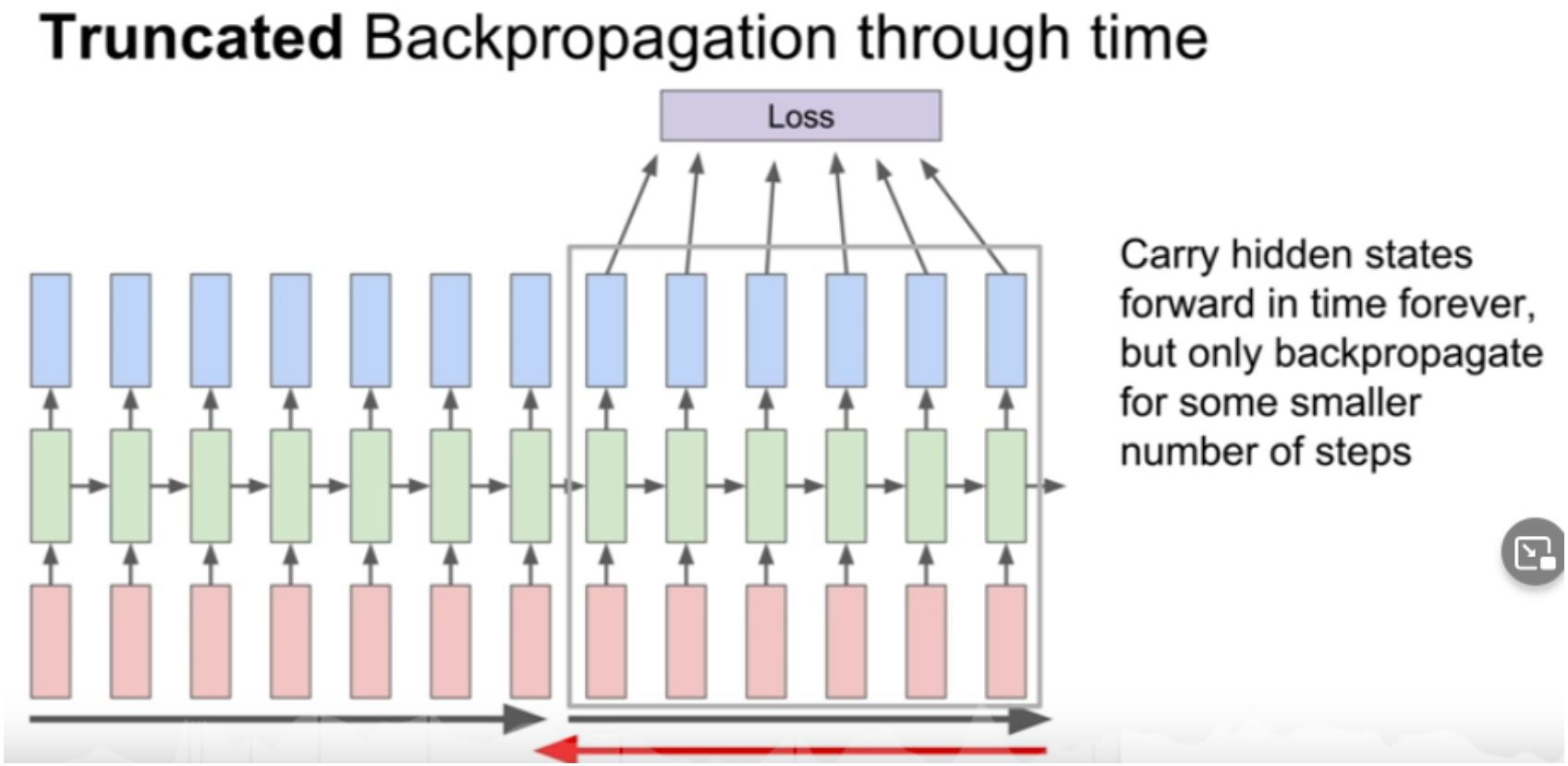
Teacher forcing no treinamento



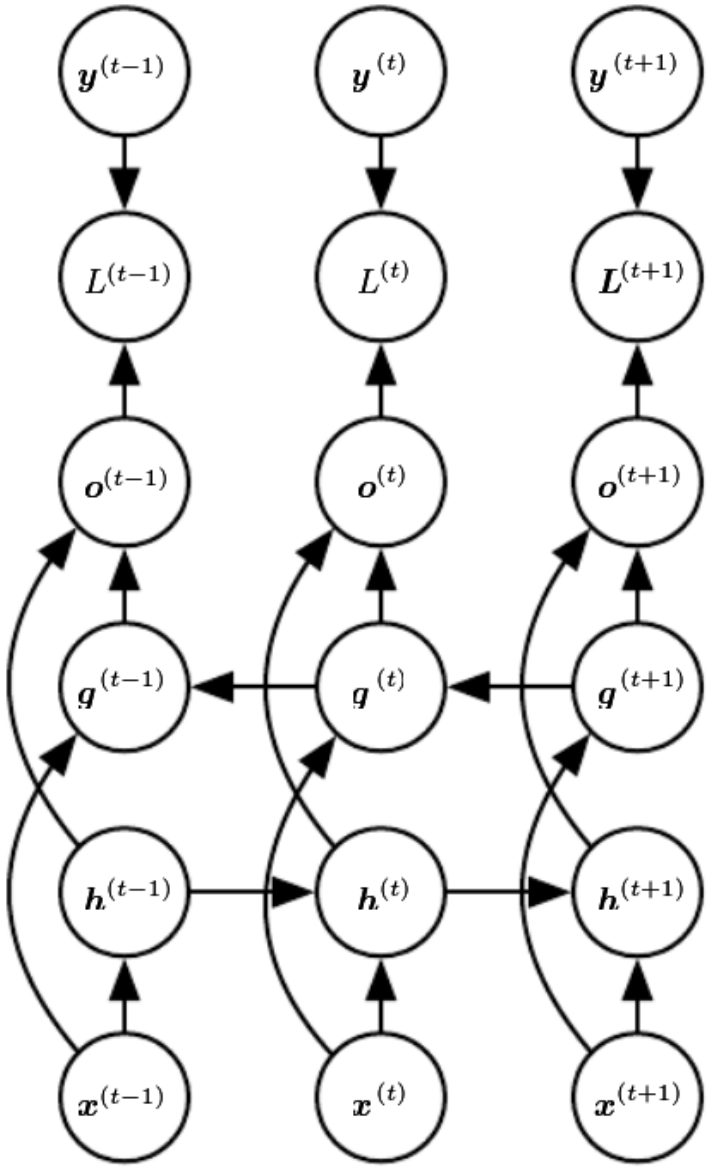
Without Teacher Forcing



Truncated BPTT (back propagation through time)



Bidirectional RNN – recorrência bidirecional



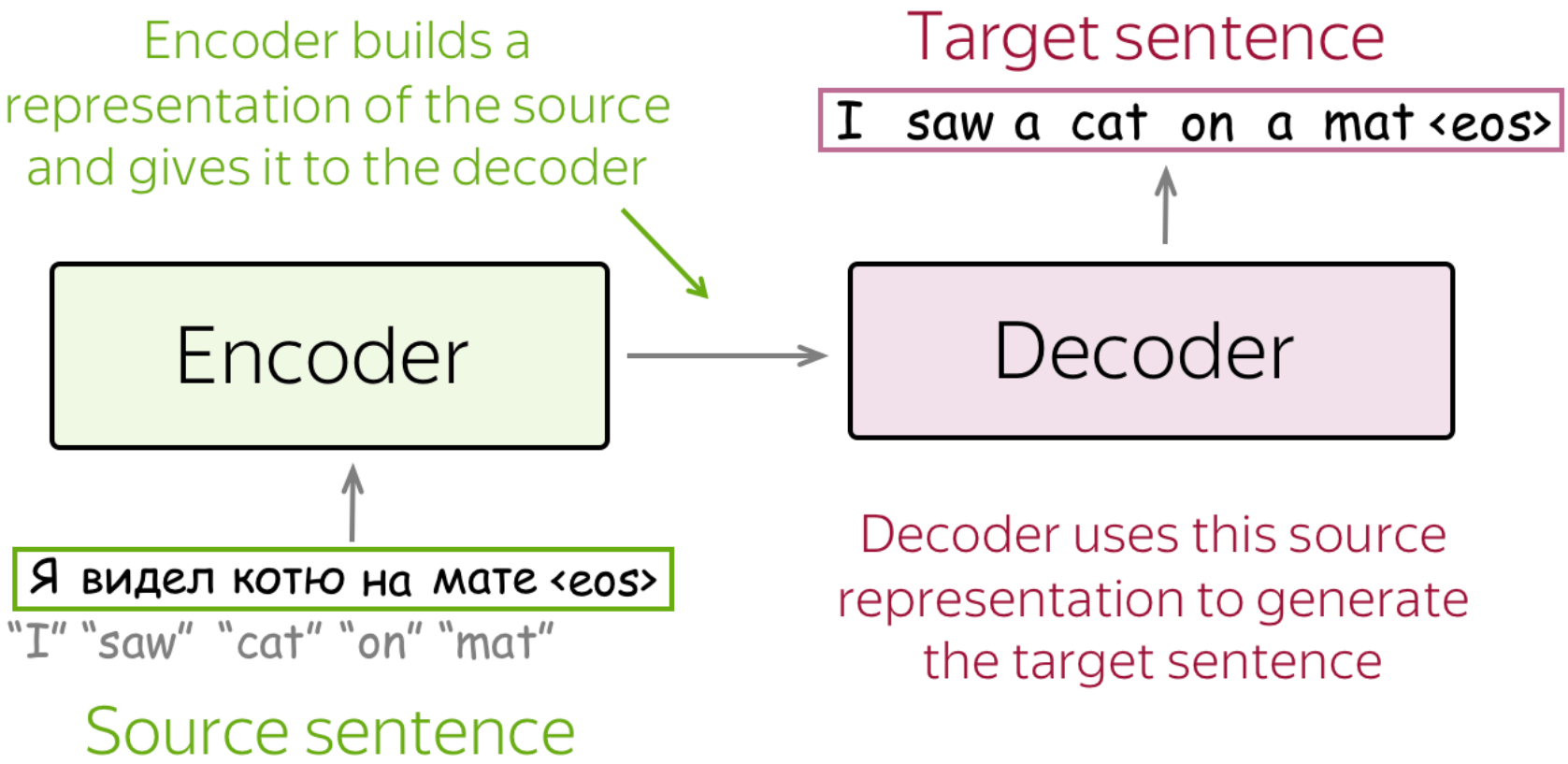
seq2seq model (2014)

Sequence to Sequence Learning with Neural Networks

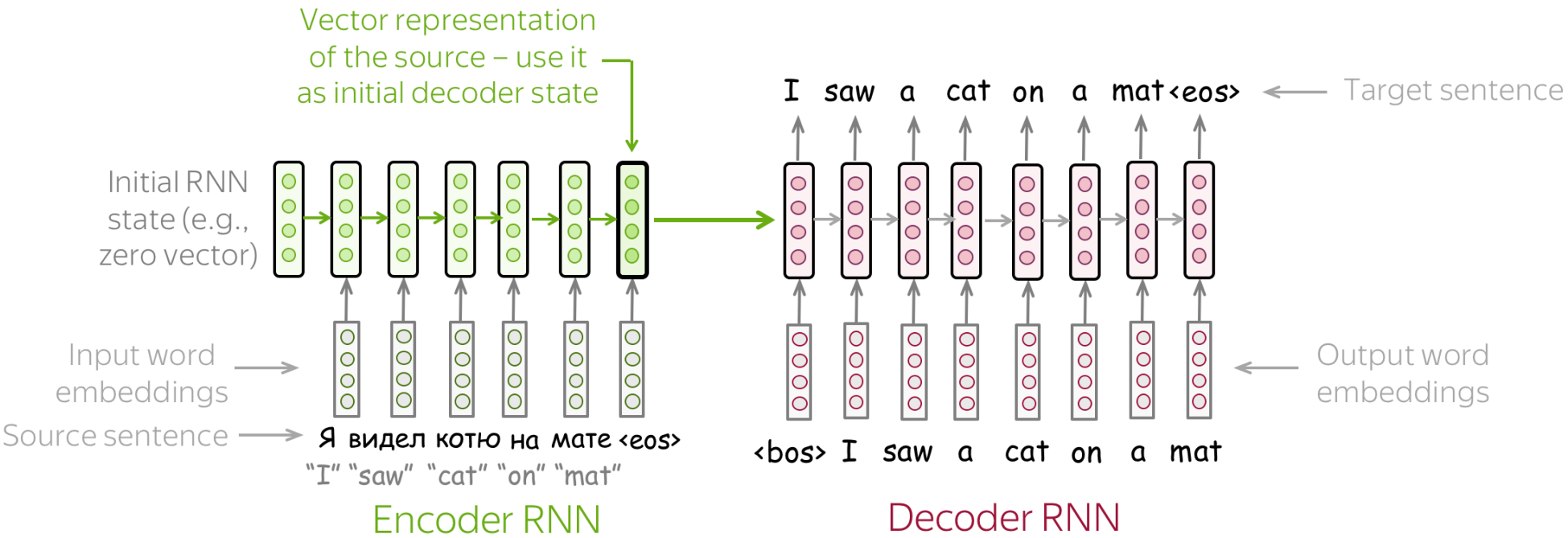
Ilya Sutskever, Oriol Vinyals, Quoc V. Le

https://arxiv.org/abs/1409.3215

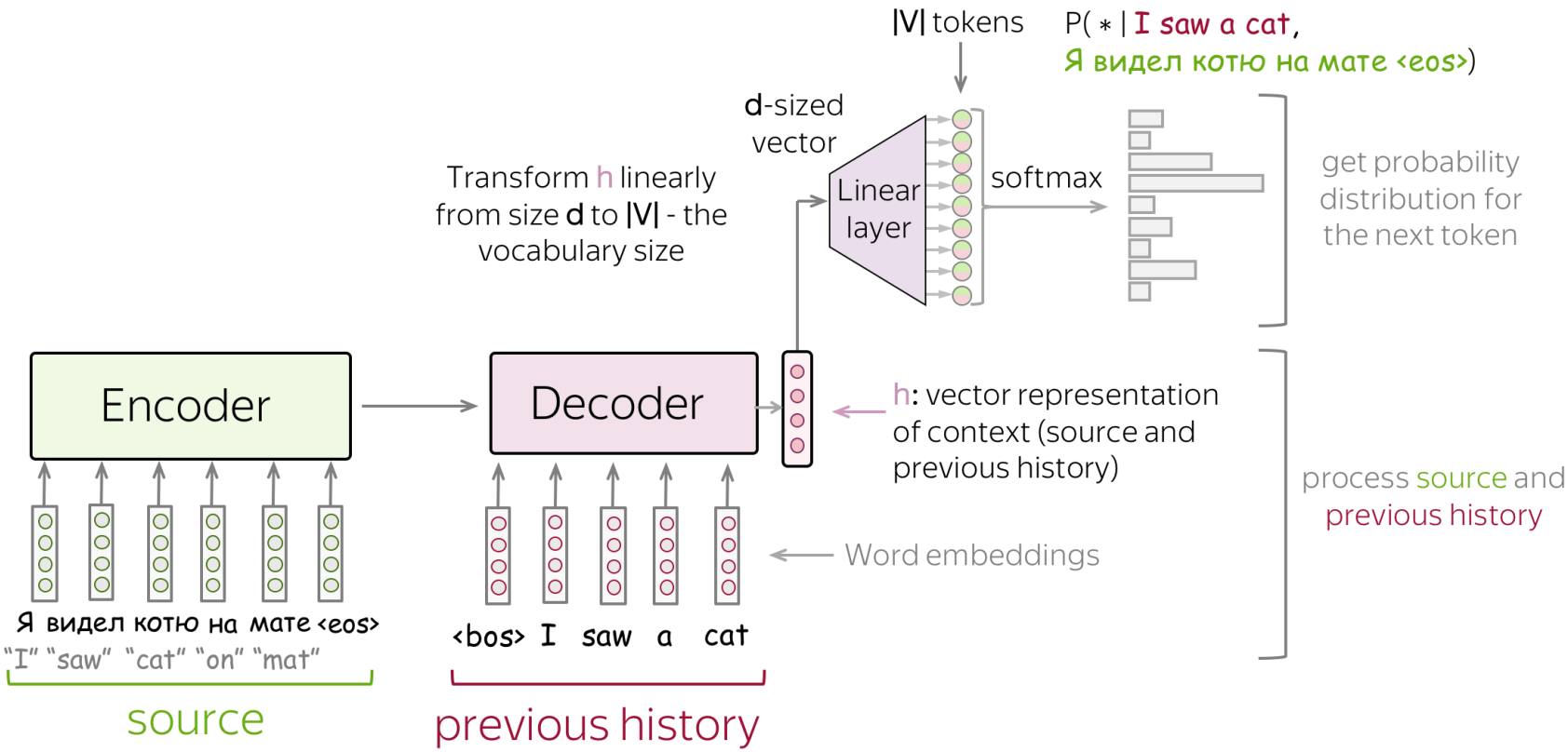
- arquitetura encoder-decoder de RNNs



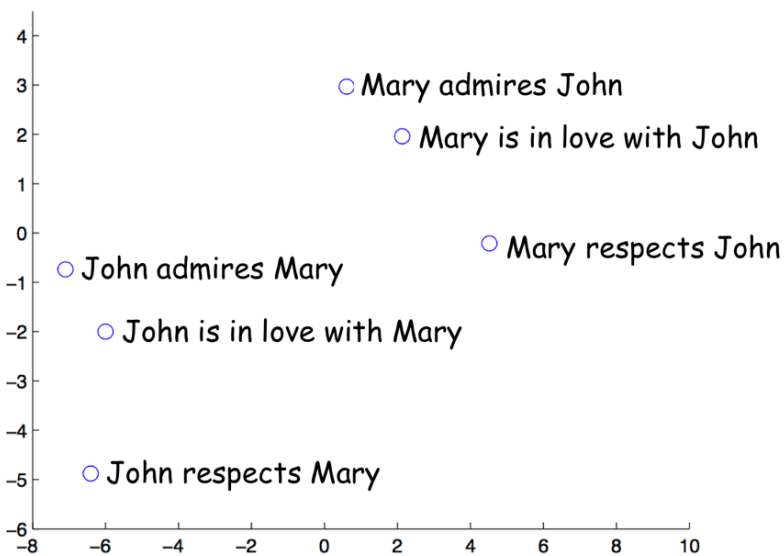
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

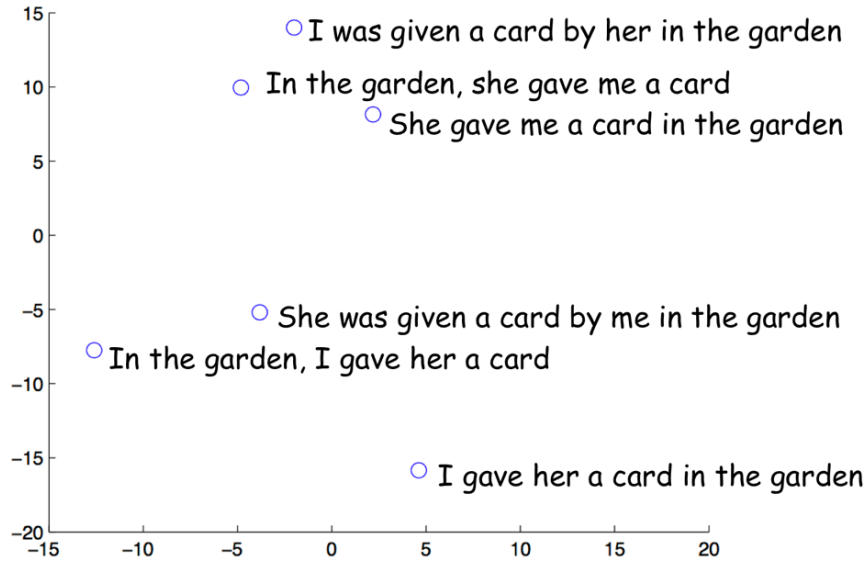


https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html



https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html





https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

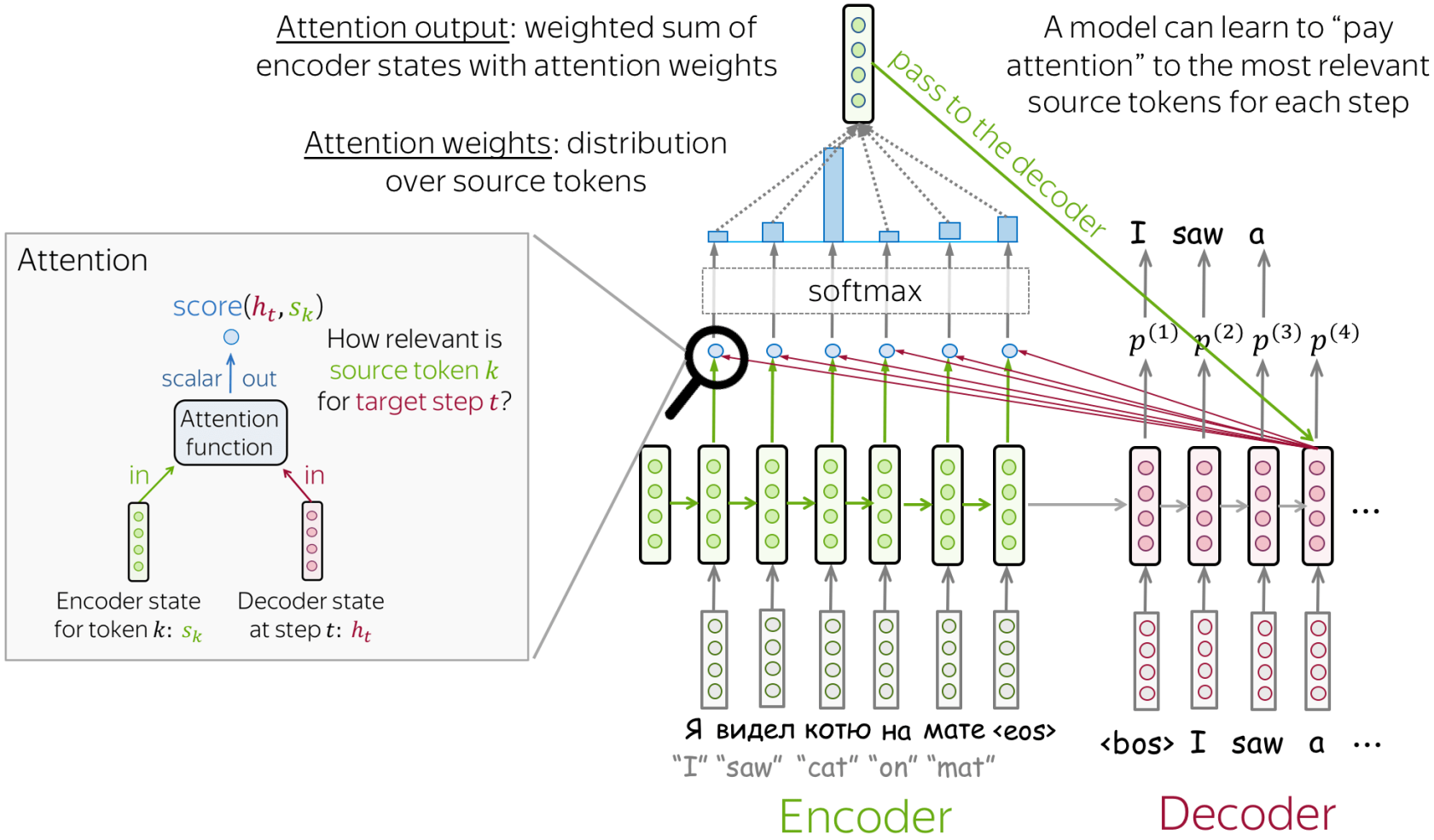
seq2seq model with attention (2014)

Neural Machine Translation by Jointly Learning to Align and Translate

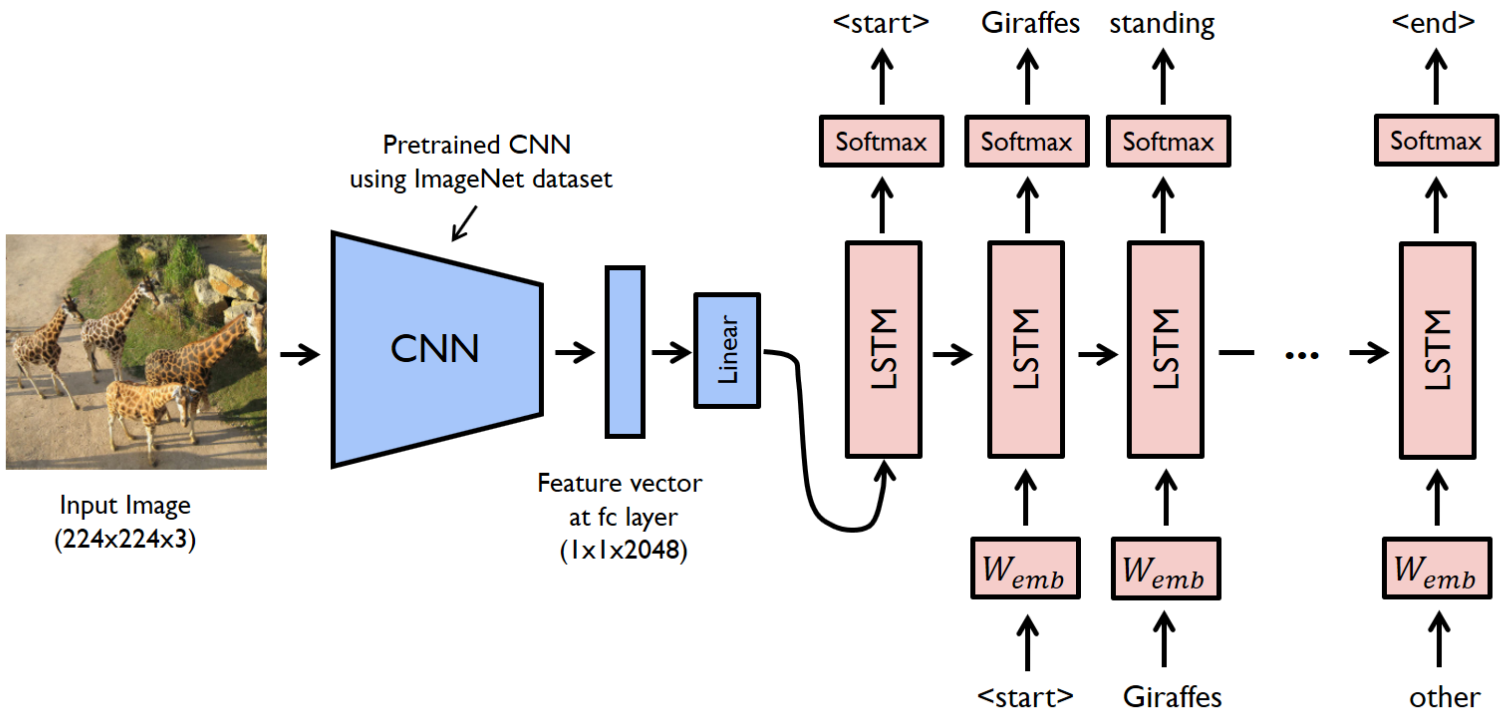
Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio

https://arxiv.org/abs/1409.0473

- trata o problema de dependências longas



https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html


Pretrained CNN
using ImageNet dataset
CNN
Input Image
(224x224x3)
Feature vector
at fc layer
(1x1x2048)
Linear
LSTM
LSTM
LSTM
LSTM
Softmax
Softmax
Softmax
Softmax
<start>
Giraffes
standing
<end>
W_{emb}
W_{emb}
W_{emb}
<start>
Giraffes
other