Aprendizaje Profundo

Facultad de Ingeniería
Universidad de Buenos Aires

Profesores:
Alfonso Rafael
Marcos Maillot

Redes Recurrentes

Recurrent Neural Network (RNN)

. Introducción

. Neurona recurrente básica

. Implementación en pytorch

. Back propagation through time (BPTT)

. Birideccionalidad

. Arquitectura enconder-decoder (seq to seq)

. Mecanismos de atención

Redes recurrentes - Introducción

Red neuronal **favorita** para el trabajo secuencias (datos que en cuya naturaleza exista un **comportamiento secuencial**):

- señales temporales
- series temporales
- texto
- habla
- música
- etc

En **cada paso**, se repiten los **mismos cálculos**, empleando **datos del paso actual** y **datos del pasado**.

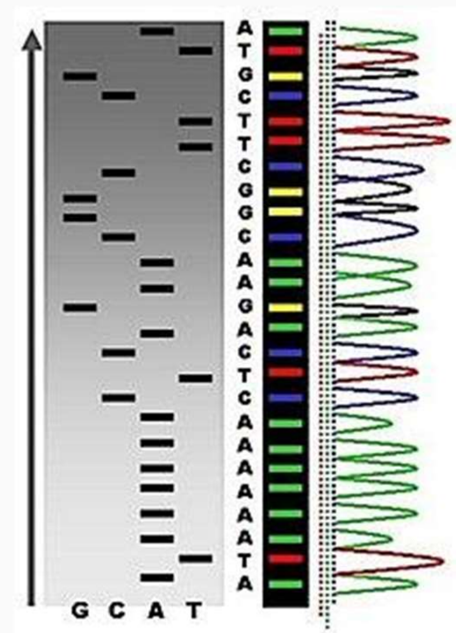
Los pasos, no son necesariamente en unidad tiempo!!

Temperatura f(t):

X(t=0)	X(t=1)	X(t=2)	X(t=3)	X(t=4)	X(t=5)	…	X(t=τ)

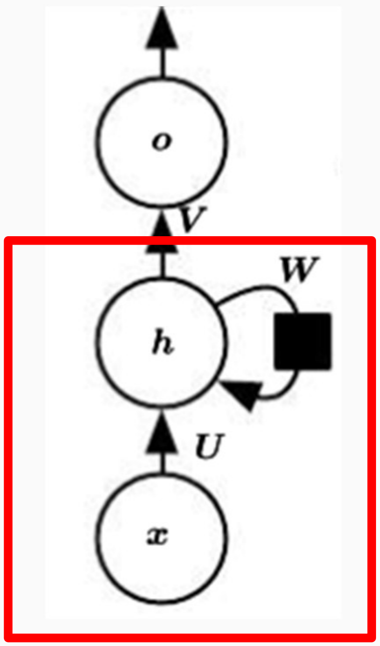
Mensaje:

Este	es	un	mensaje	para	la	red	neuronal



wikipedia

Redes recurrentes - Neurona recurrente básica



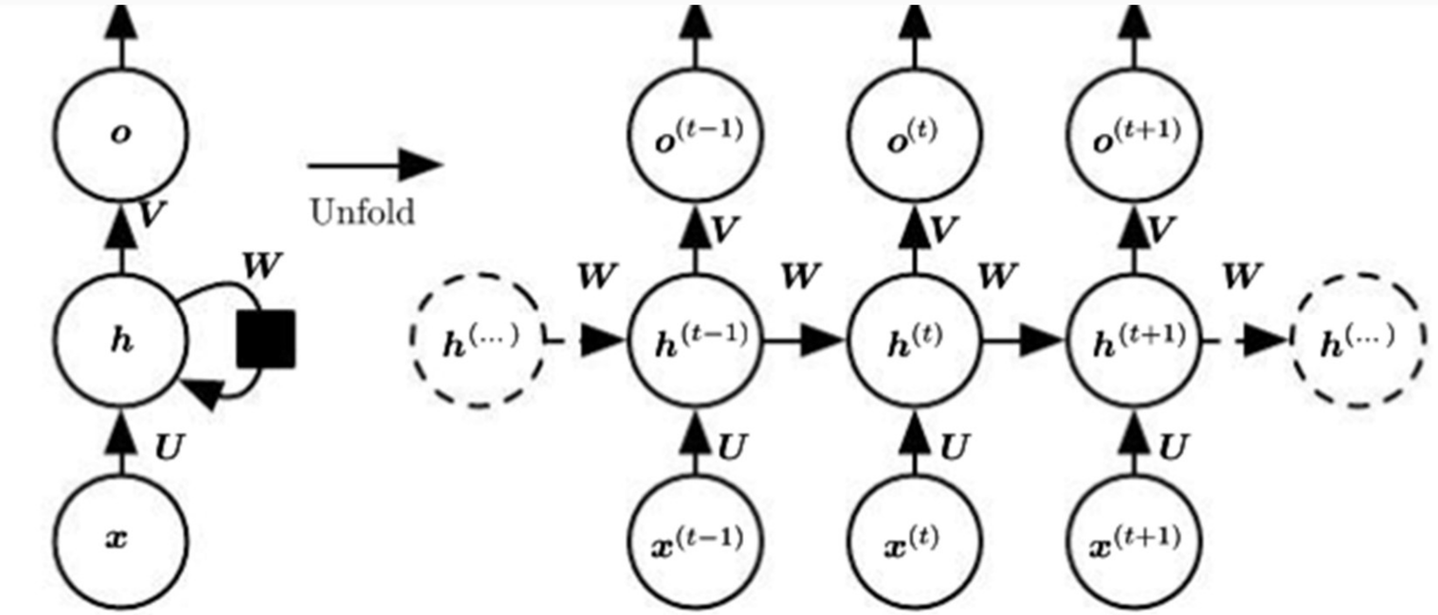
Ecuaciones

$$\boxed{\begin{aligned} \boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \\ \boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}), \end{aligned}}$$

$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)},$$

Redes recurrentes - Neurona recurrente básica

$$\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}), \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)},
\end{aligned}$$



***U*, *W* son los mismos!!**

Parameters sharing

Redes recurrentes - Neurona recurrente básica

$x[n] = [0{,}7;\ 0{,}9;\ 1{,}1]$

$\left\{ U = 0{,}2 \ ;\ b = -0{,}1 \ ;\ W = 0{,}3 \right\}$ Parámetros inventados

$$h[0] = \tanh[-0{,}1 + 0{,}3 \cdot \underbrace{h[-1]}_{=0 \text{ para la } 1^{\circ} \text{ muestra}} + 0{,}2 \cdot 0{,}7] = 0{,}0399$$

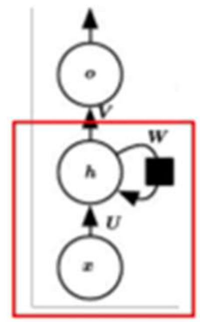
$$h[1] = \tanh[-0{,}1 + 0{,}3 \cdot 0{,}0399 + 0{,}2 \cdot 0{,}9] = 0{,}0917$$

$$h[2] = \tanh[-0{,}1 + 0{,}3 \cdot 0{,}0917 + 0{,}2 \cdot 1{,}1] = 0{,}1464$$

Válido para cada neurona hidden

Las hidden se conectan todas contra todas

$$\begin{aligned} \boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \\ \boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}), \\ \boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \end{aligned}$$



Redes recurrentes - Implementación en pytorch

RNN

CLASS `torch.nn.RNN(*args, **kwargs)` [SOURCE]

Applies a multi-layer Elman RNN with $\tanh$ or ReLU non-linearity to an input sequence.

For each element in the input sequence, each layer computes the following function:

$$h_t = \tanh(W_{ih} x_t + b_{ih} + W_{hh} h_{(t-1)} + b_{hh})$$

where h_t is the hidden state at time *t*, x_t is the input at time *t*, and $h_{(t-1)}$ is the hidden state of the previous layer at time *t-1* or the initial hidden state at time *0*. If `nonlinearity` is `'relu'`, then ReLU is used instead of $\tanh$.

torch.nn.RNN(input_size, hidden_size, num_layers=1, nonlinearity=‘tanh’, …
bias=True, batch_first=False, dropout=0,

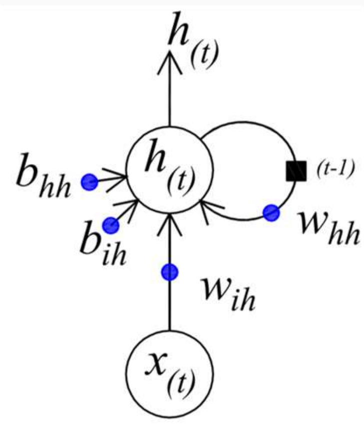
bidirectional=False)

Redes recurrentes - Implementación en pytorch

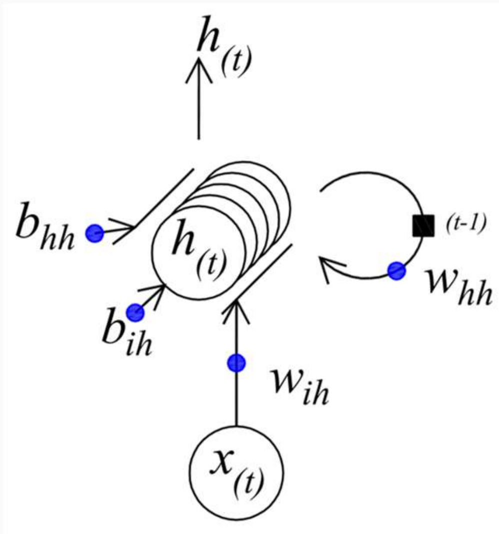
Pytorch

Ver colab RNN_teoria.ipynb

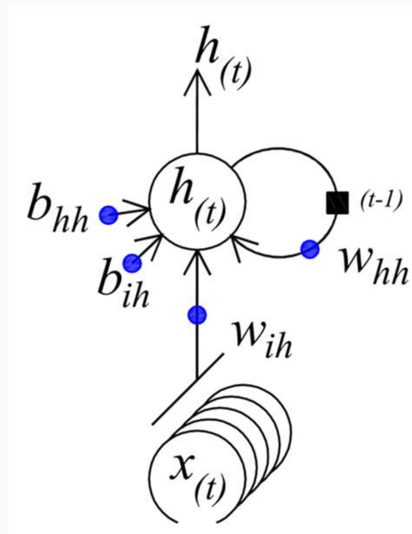
$$h_t = \tanh(W_{ih}x_t + b_{ih} + W_{hh}h_{(t-1)} + b_{hh})$$



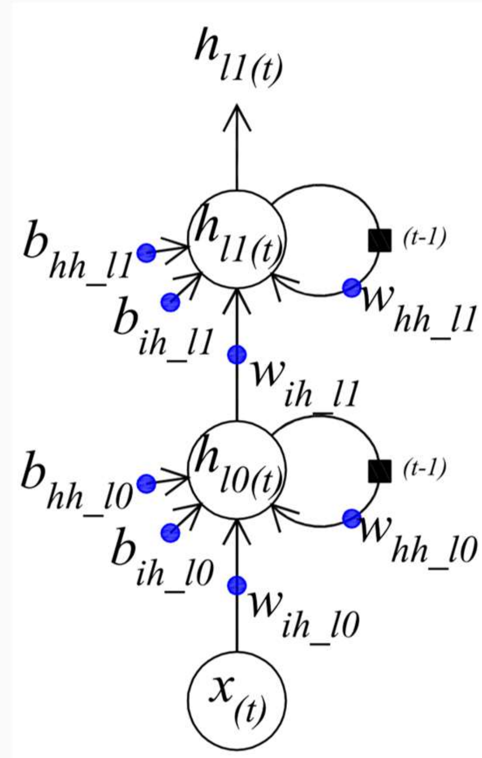
Básica



Varias hidden



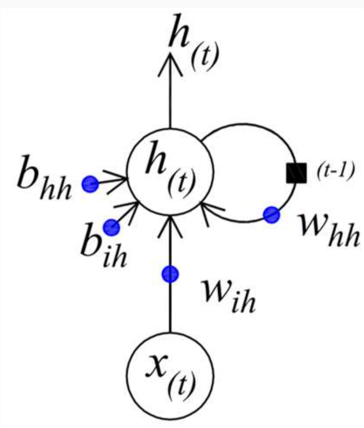
Input multivariable



2 layers

Redes recurrentes - Implementación en pytorch

$$h_t = \tanh(W_{ih} x_t + b_{ih} + W_{hh} h_{(t-1)} + b_{hh})$$

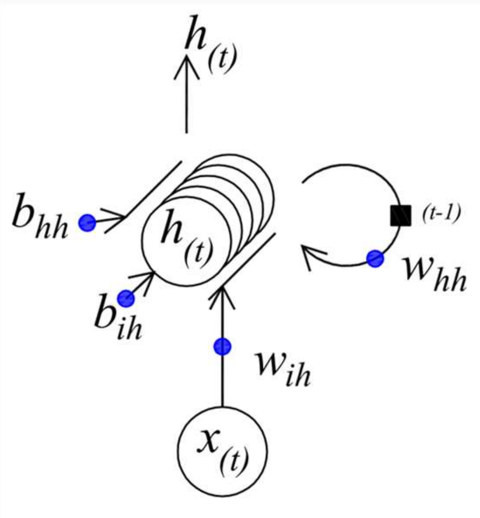


Variable	Tamaño	Parámetro	Tamaño
x		Wih	
h		Bih	
		Whh	
		bhh	

Básica

Redes recurrentes - Implementación en pytorch

$$h_t = \tanh(W_{ih} x_t + b_{ih} + W_{hh} h_{(t-1)} + b_{hh})$$

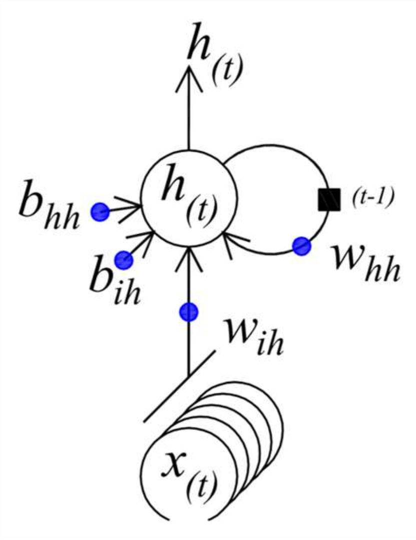


Varias hidden

Variable	Tamaño	Parámetro	Tamaño
x		Wih	
h		Bih	
		Whh	
		bhh	

Redes recurrentes - Implementación en pytorch

$$h_t = \tanh(W_{ih}x_t + b_{ih} + W_{hh}h_{(t-1)} + b_{hh})$$

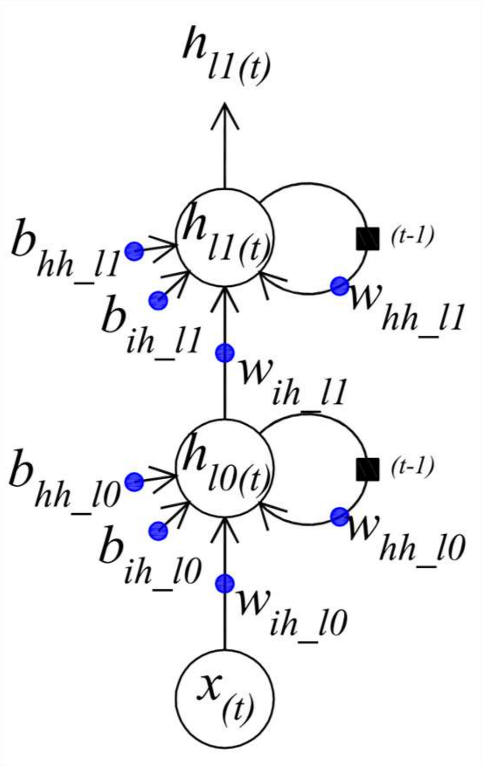


Input multivariable

Variable	Tamaño	Parámetro	Tamaño
x		Wih	
h		Bih	
		Whh	
		bhh	

Redes recurrentes - Implementación en pytorch

$$h_t = \tanh(W_{ih}x_t + b_{ih} + W_{hh}h_{(t-1)} + b_{hh})$$

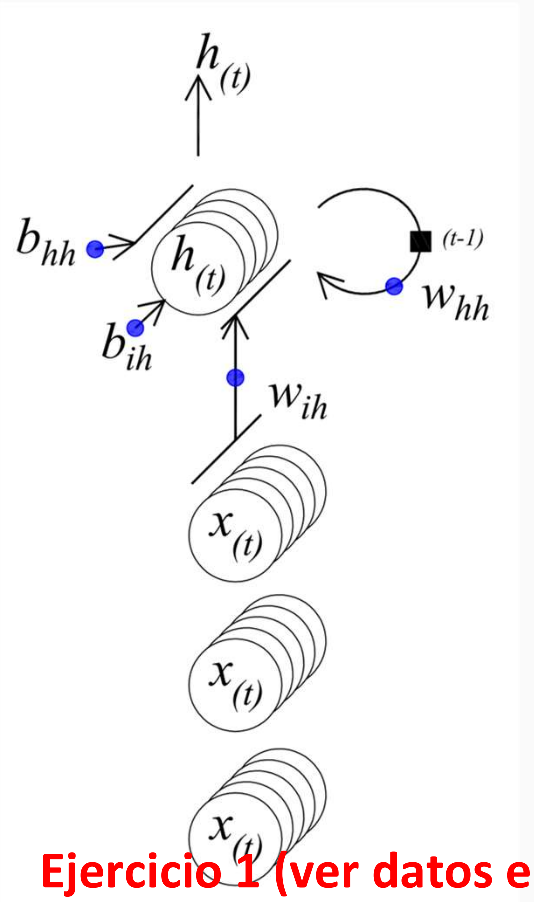


2 layers

Variable	Tamaño	Parámetro	Tamaño
x		Wih	
h		Bih	
		Whh	
		bhh	

Redes recurrentes - Implementación en pytorch

$$h_t = \tanh(W_{ih}x_t + b_{ih} + W_{hh}h_{(t-1)} + b_{hh})$$



Ejemplo A

Variable	Tamaño	Parámetro	Tamaño
x		Wih	
h		Bih	
		Whh	
		bhh	

Ejemplo B

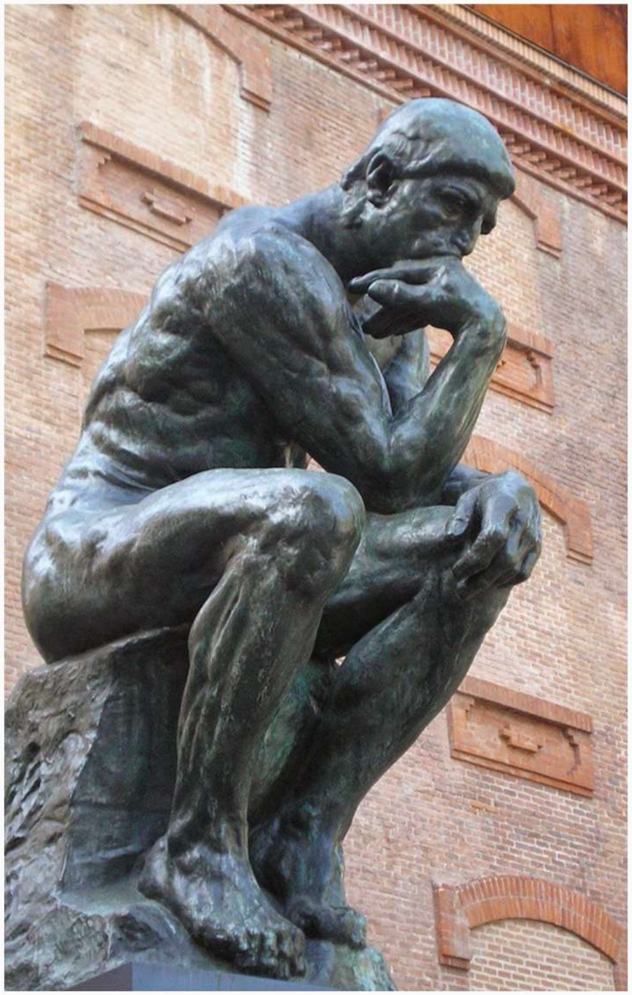
Variable	Tamaño	Parámetro	Tamaño
x		Wih	
h		Bih	
		Whh	
		bhh	

Ejemplo C

Variable	Tamaño	Parámetro	Tamaño
x		Wih	
h		Bih	
		Whh	
		bhh	

Ejercicio 1 (ver datos en colab)

Redes recurrentes - Implementación en pytorch



“El pensador” de Rodin

A pensar!

Redes recurrentes - Back propagation through time (BPTT)

Ver desarrollo teórico

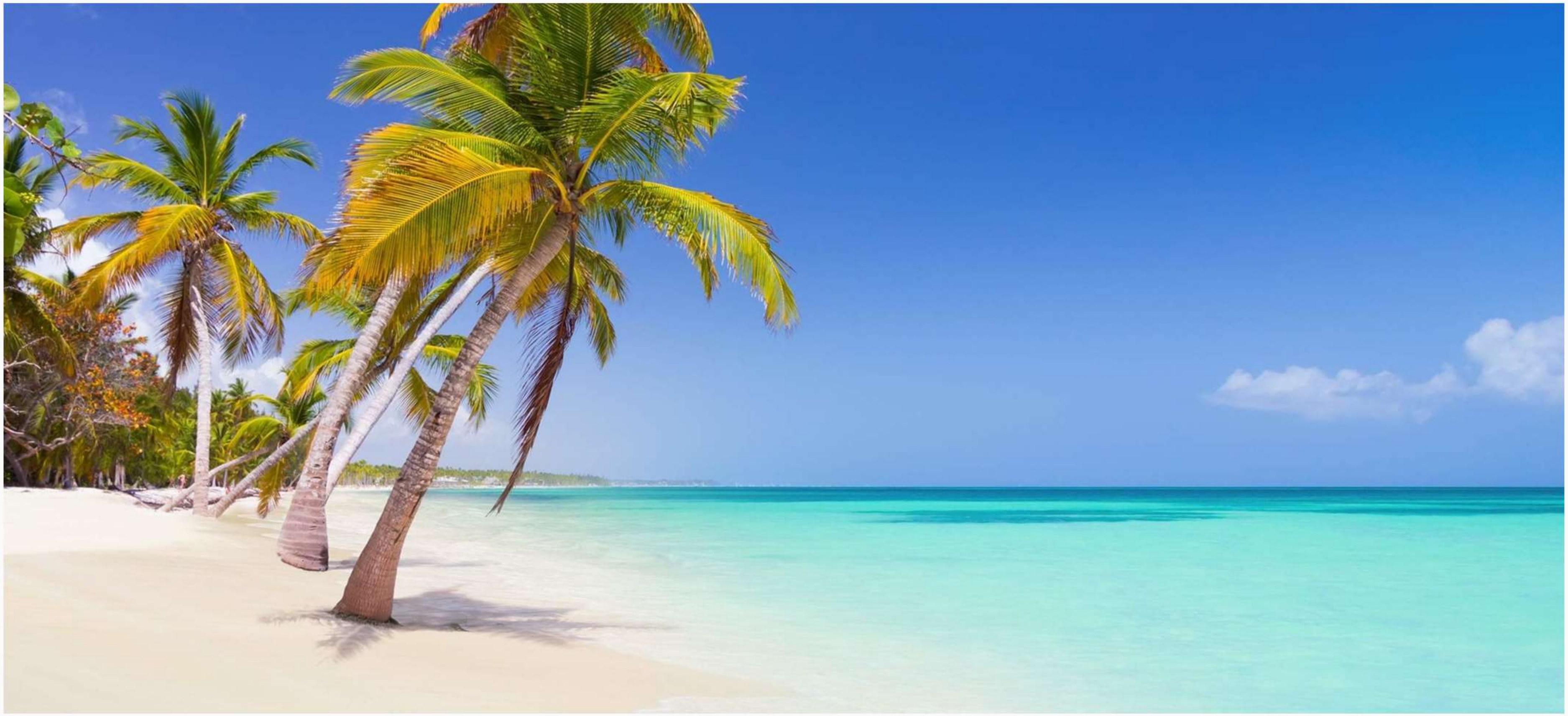
Problemas de la RNN básica con el BPTT

Vanishing gradient → pérdida de aportes de long-term states (gradientes próximos a cero)

Exploding gradient → se soluciona con clipping gradient (gradientes mayores a 1)

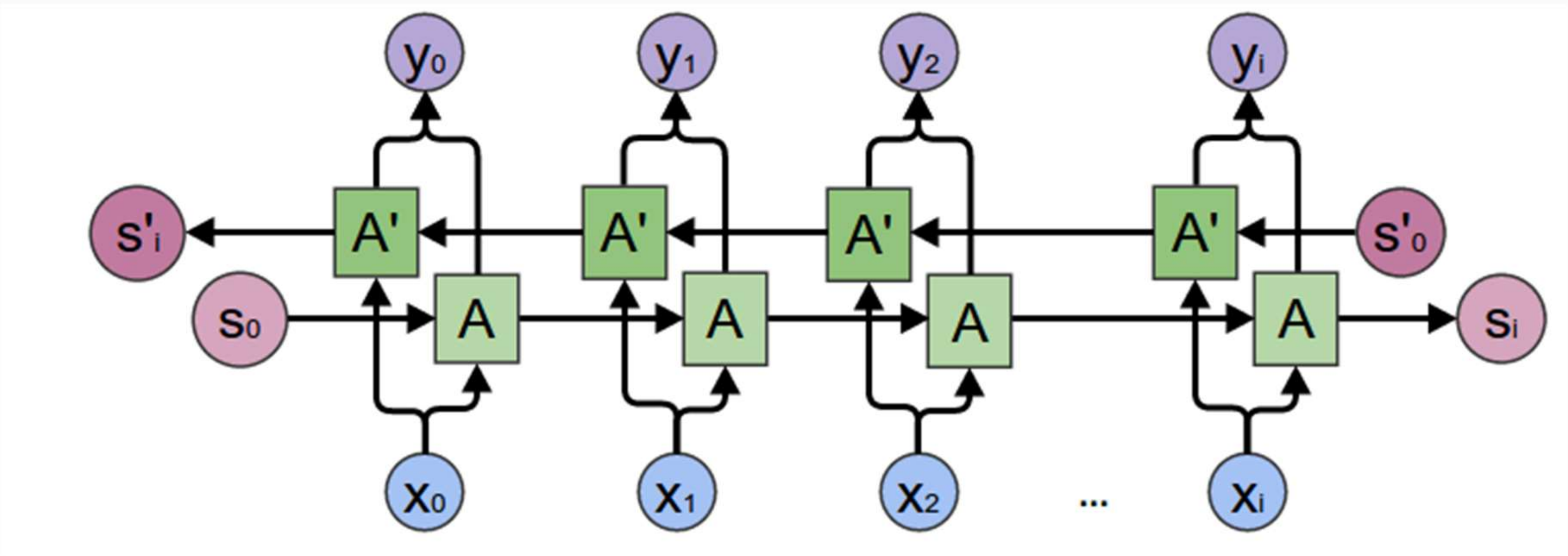
Solución con otras RNN mas avanzadas (LSTM y GRU)

¡Un merecido descanso!



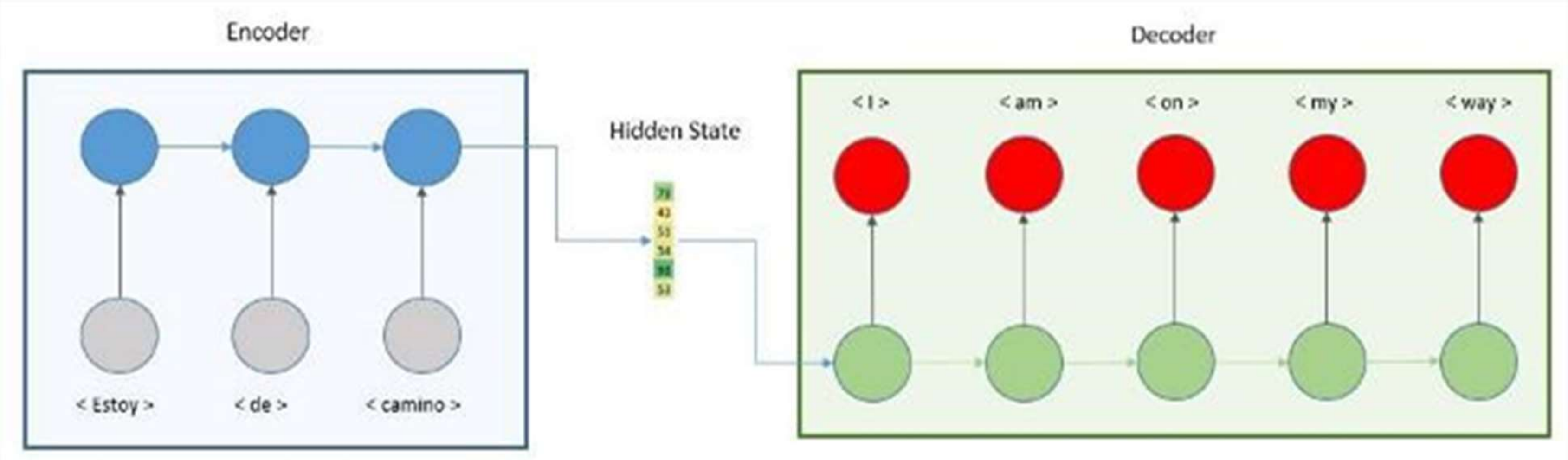
Redes recurrentes - Birideccionalidad

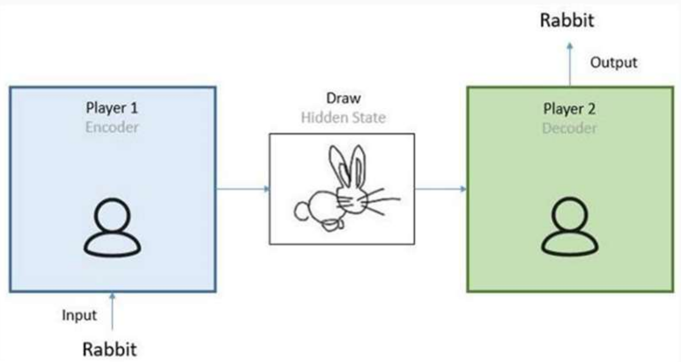
Bi-directional



Para la traducción, suele ser útil tener la frase entera.

Redes recurrentes - Arquitectura enconder-decoder (seq to seq)





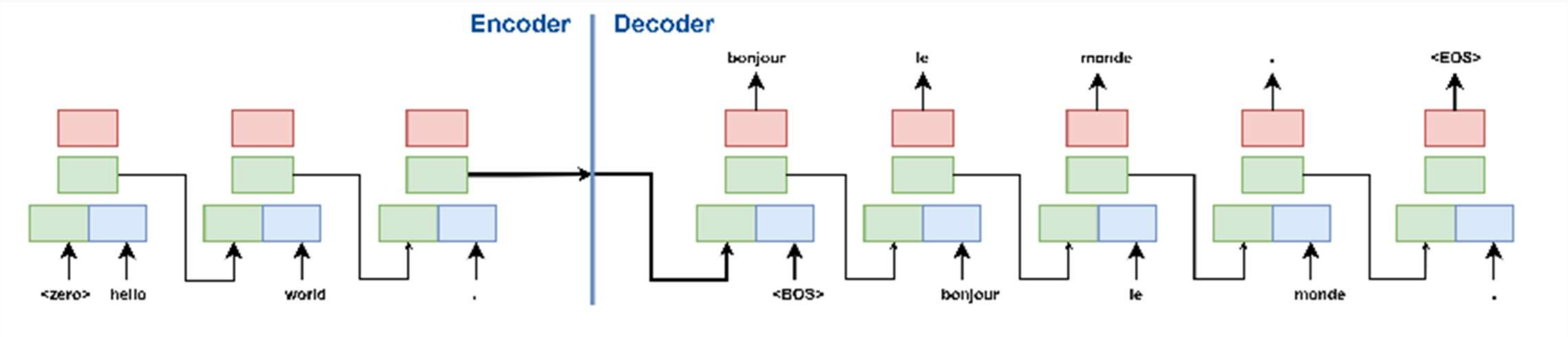
2 RNN de distinto tamaño

1 Hidden state que “resume” toda la información de la input.

Flexibilidad máxima para inputs/outputs de distinta longitud

https://towardsdatascience.com/what-is-an-encoder-decoder-model-86b3d57c5e1a

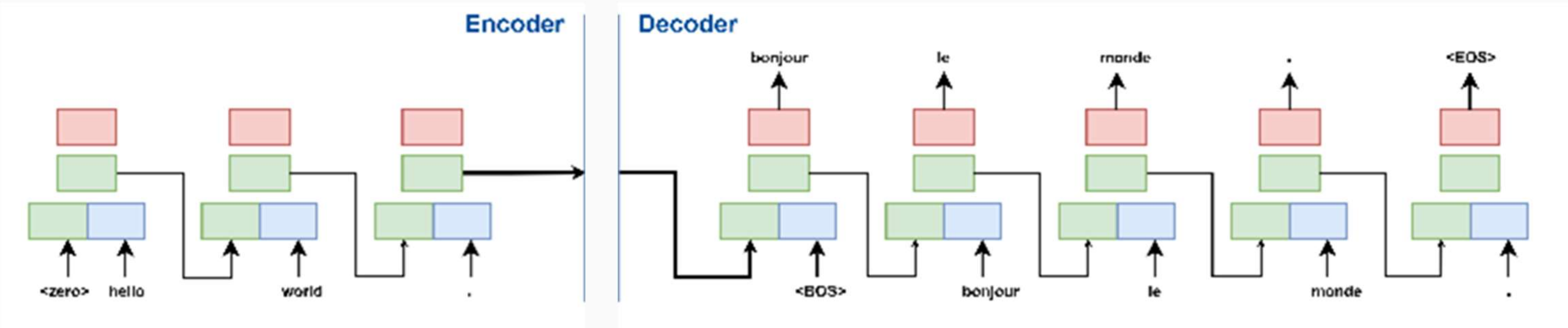
Redes recurrentes - Arquitectura enconder-decoder (seq to seq)



Unfolded!!
!!

https://brunomaga.github.io/AI-Supercomputing-2

Redes recurrentes - Arquitectura enconder-decoder (seq to seq) entrenamiento



ENCODER
Siempre leen la secuencia entera
Emiten un hidden state final

DECODER
Entrenamiento → for i in range(len(y_deseado):
genero_token

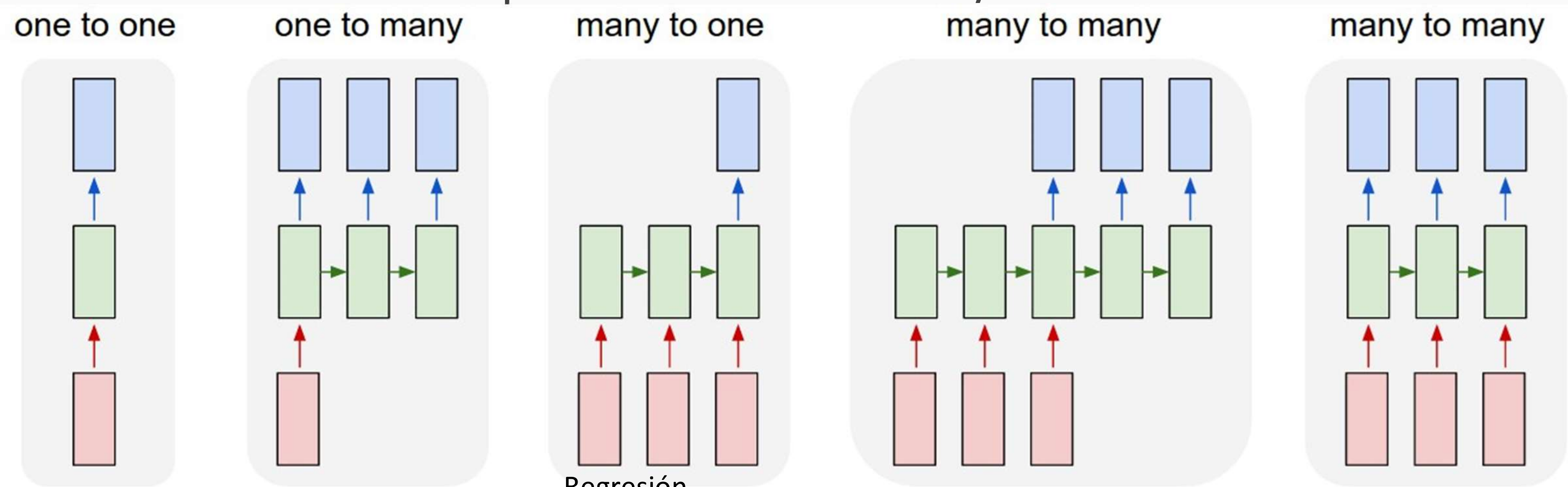
Uso → while last_token =! <EOS>:
genero_token

https://brunomaga.github.io/AI-Supercomputing-2

TEACHER FORCING!!!

Redes recurrentes - Arquitectura enconder-decoder (seq to seq)

Arquitecturas flexibles IN/OUT



MLP

Img captioning

Regresión

Classificacion

Sentiment classification

Resumen de videos

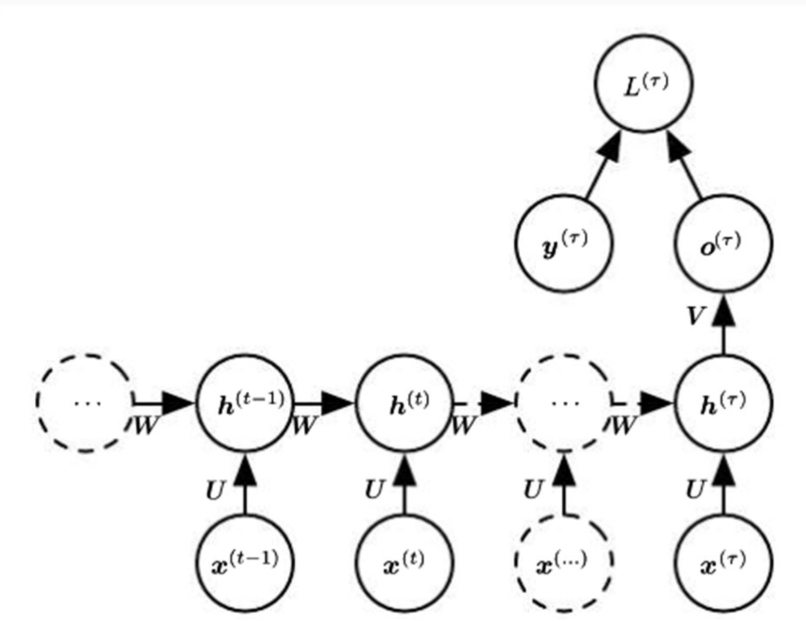
Regresión varios futuros

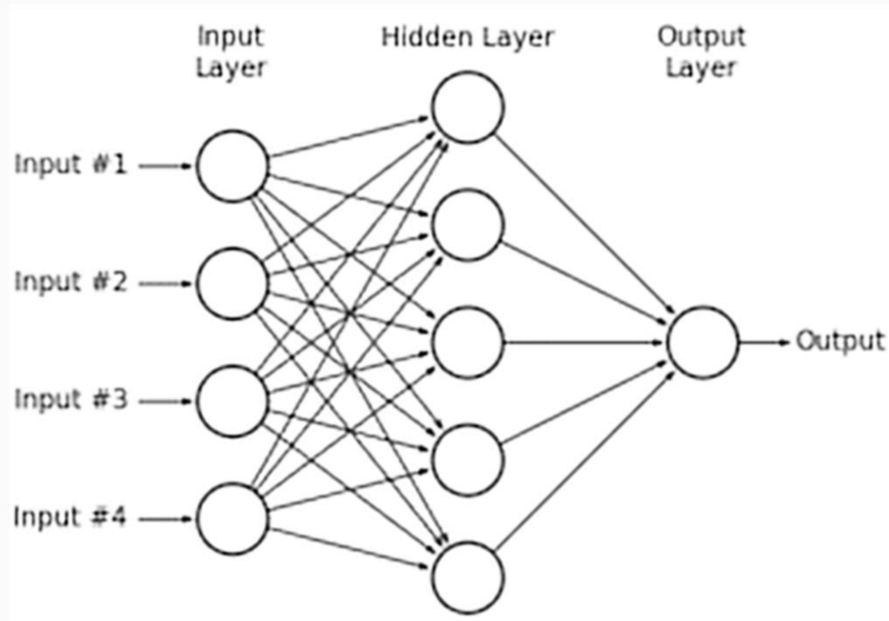
Traducción

Video class frame lvl

Redes recurrentes - Implementación de modelos

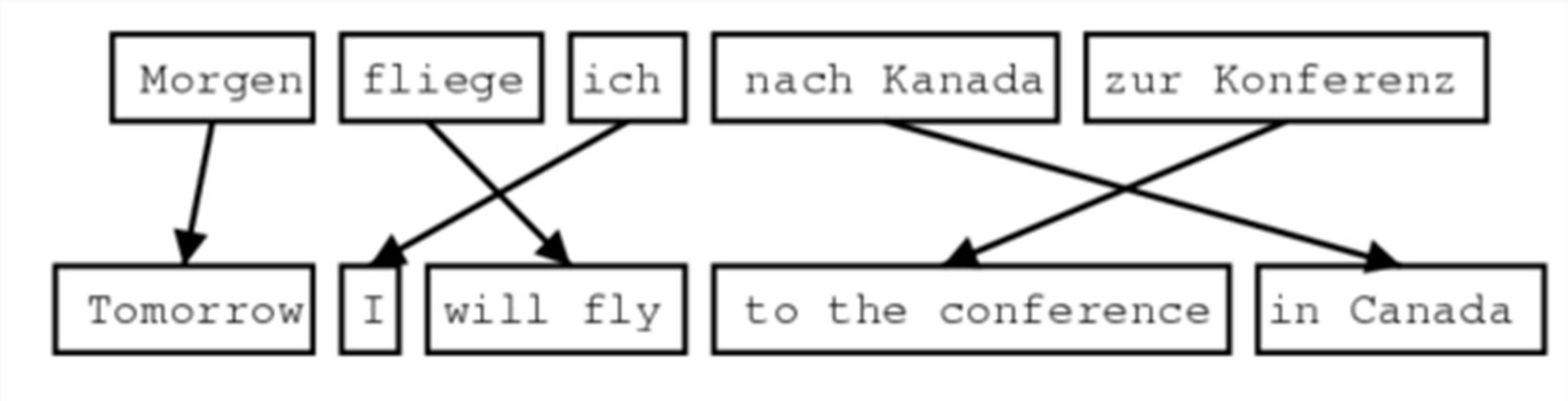
X(t=0)	X(t=1)	X(t=2)	X(t=3)	X(t=4)	X(t=5)	X(t=6)	X(t=7)	...	X(t=τ)





Ver colab RNN_signal_TP.ipynb

Redes recurrentes - Mecanismos de atención

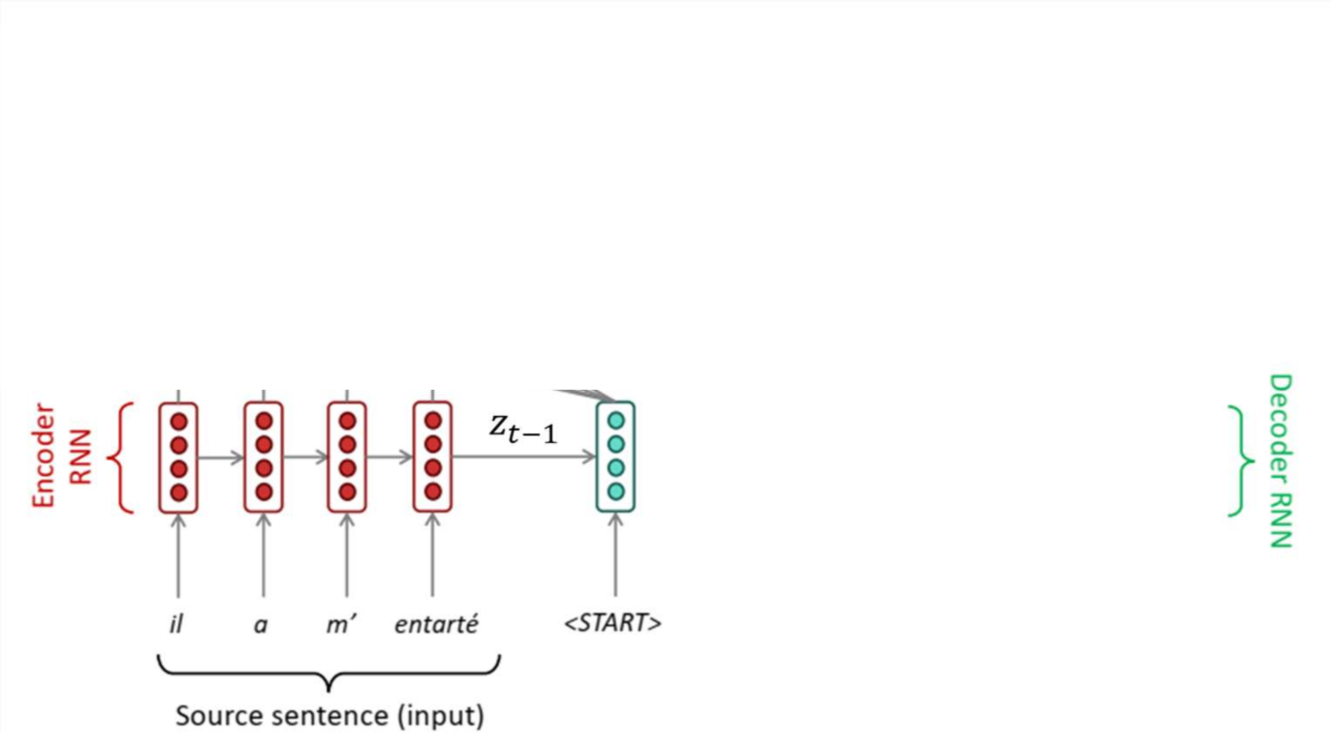


El mecanismo de atención permite al decoder ***utilizar las partes más relevantes*** *de la entrada* ***como una suma ponderada*** *del vector de entrada codificados para predecir la siguiente palabra.*

Una ***palabra relevante*** *tendrá un* ***mayor peso*** *que una palabra no relevante*

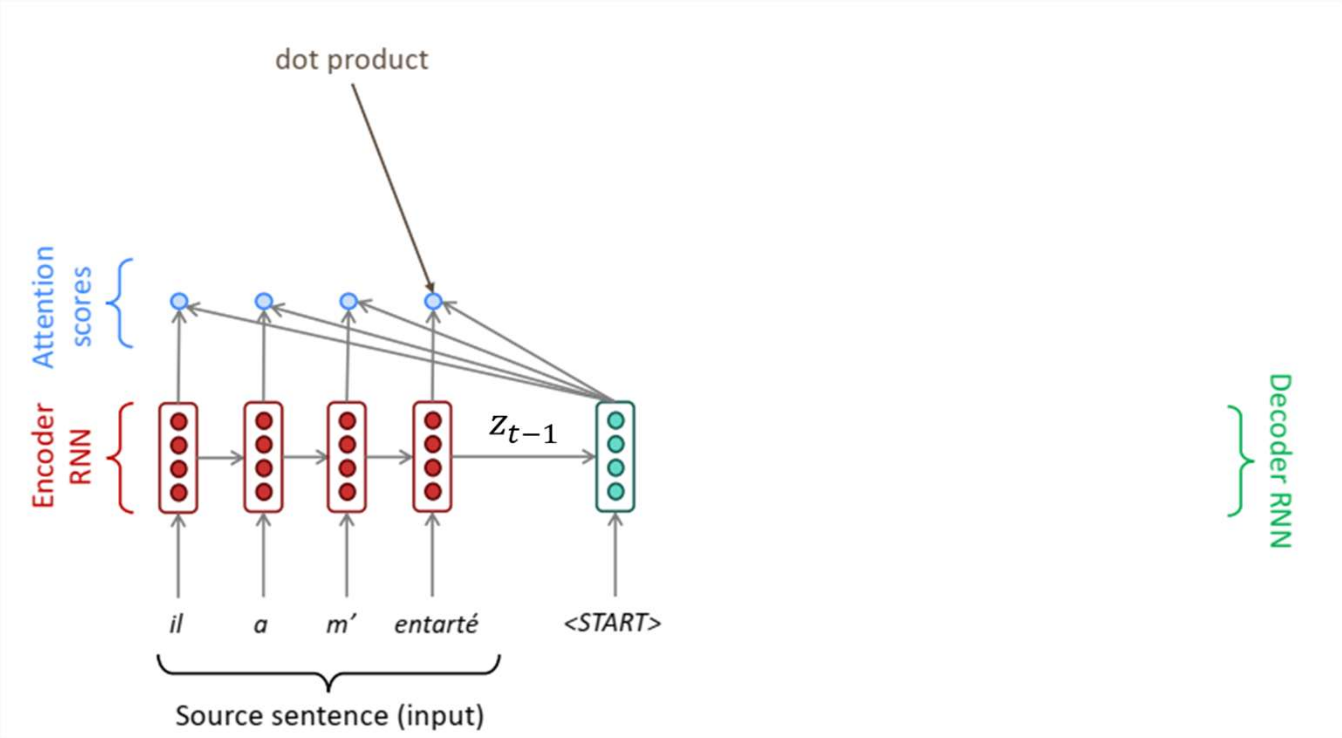
Redes recurrentes - Mecanismos de atención

Ver en bibliografía: cs224n-2021-lecture07-nmt.pdf



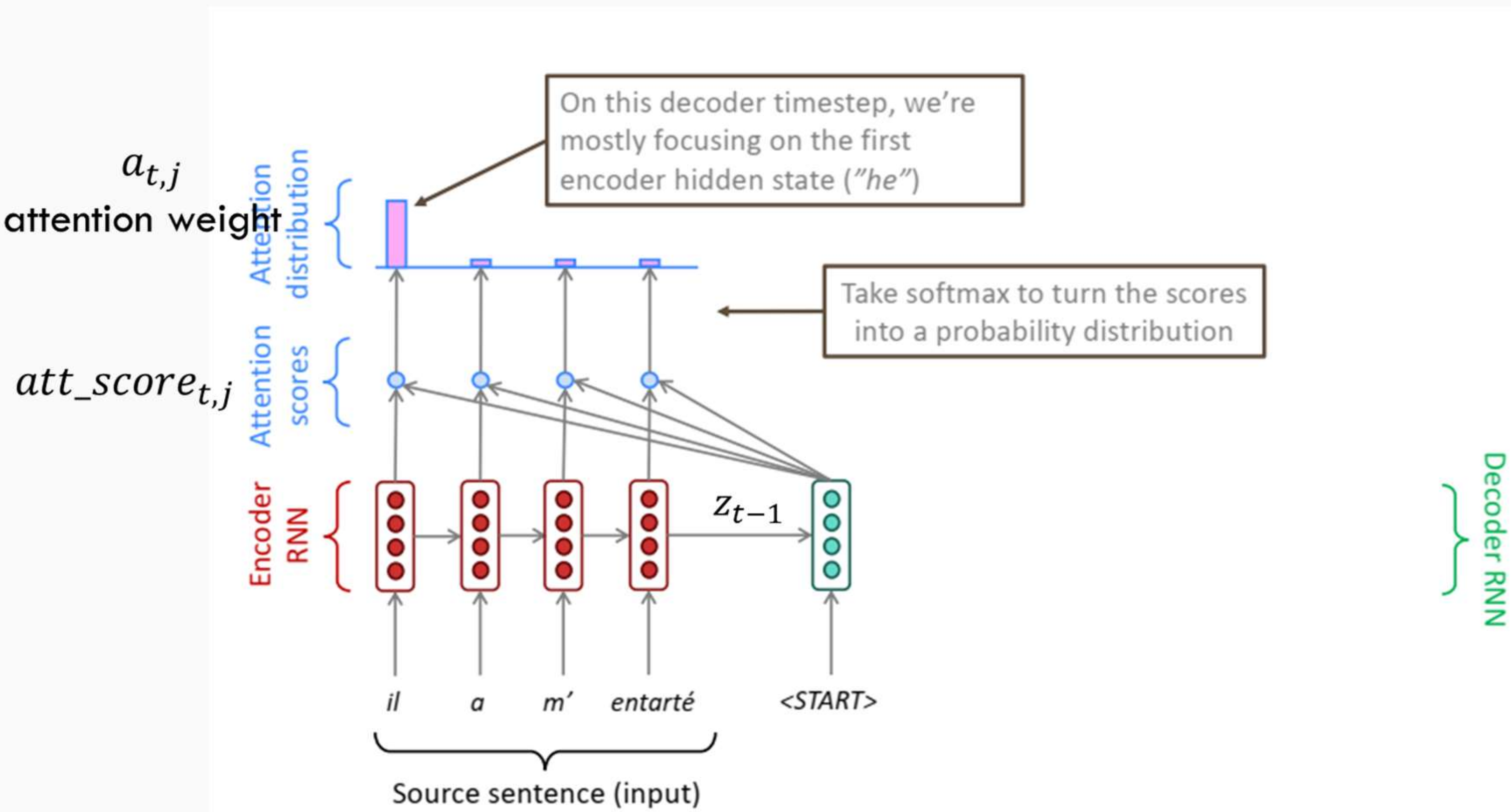
Redes recurrentes - Mecanismos de atención

$att\_score_{t,j}$



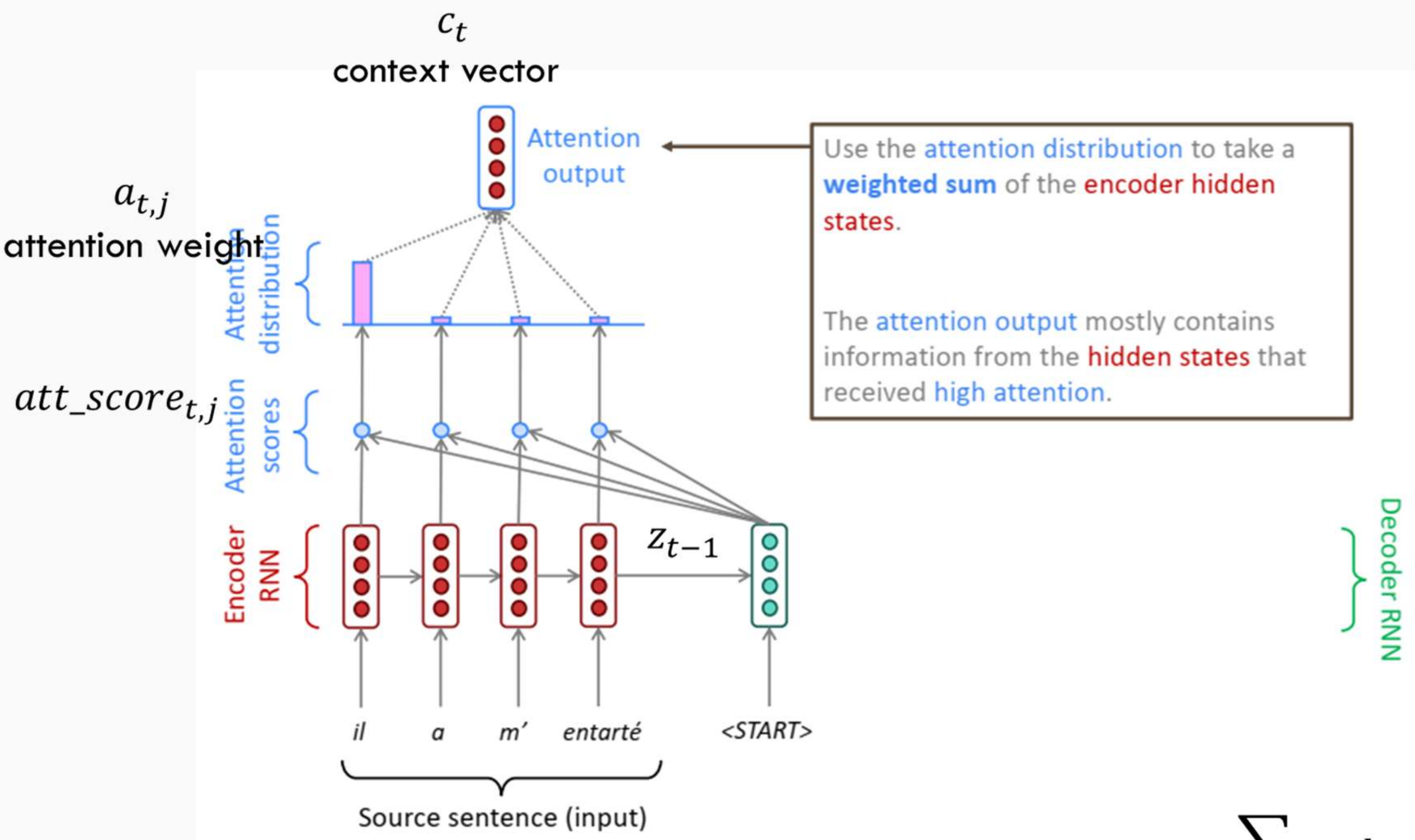
$$att\_score_{t,j} = \mathrm{dot}(z_t, h_j)$$

Redes recurrentes - Mecanismos de atención



$att\_score_{t,j} = \mathrm{dot}(z_t, h_j)$ $\quad$ $a_{t,j} = softmax(att\_score_{t,j})$

Redes recurrentes - Mecanismos de atención

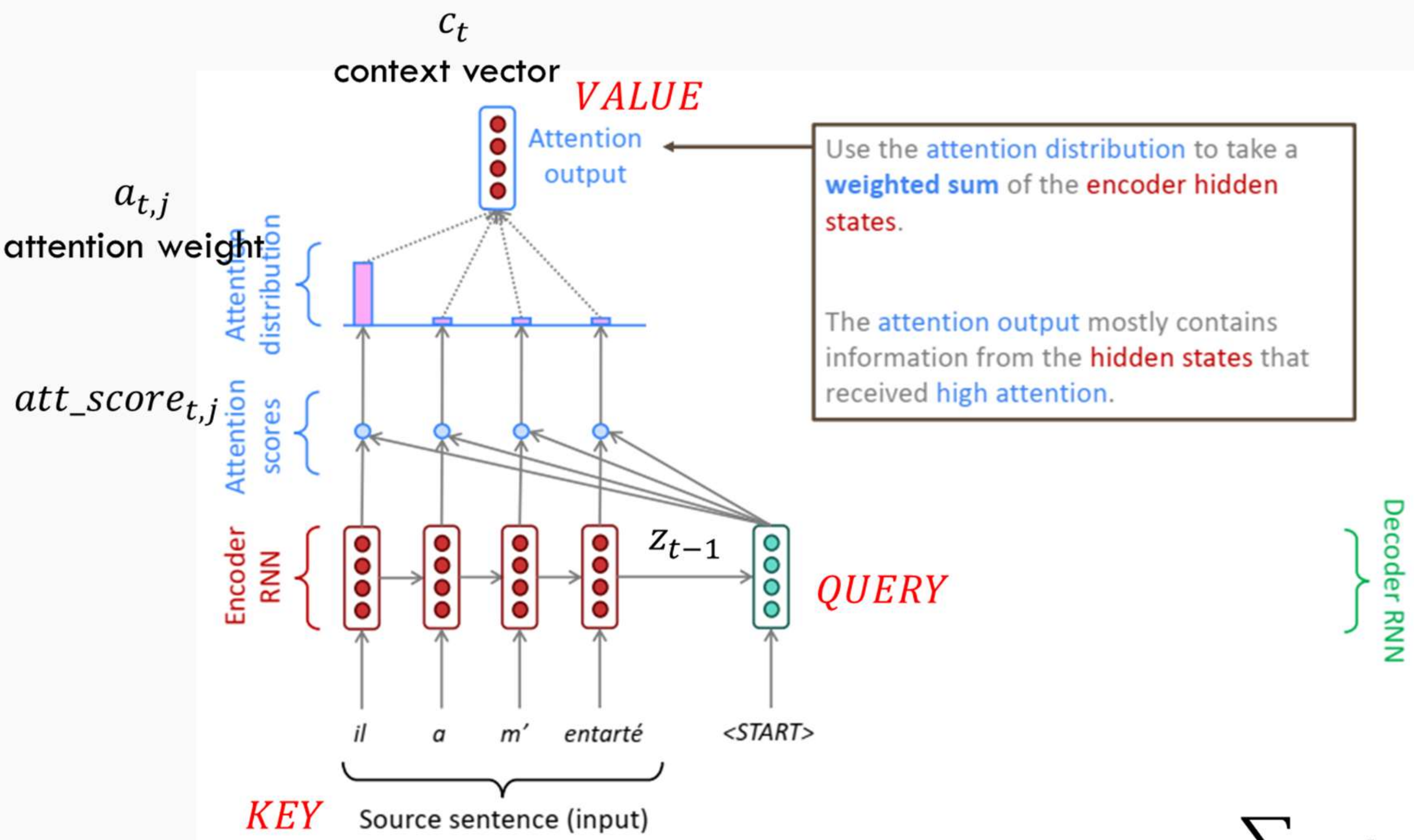


$att\_score_{t,j} = \mathrm{dot}(z_t, h_j)$

$a_{t,j} = softmax(att\_score_{t,j})$

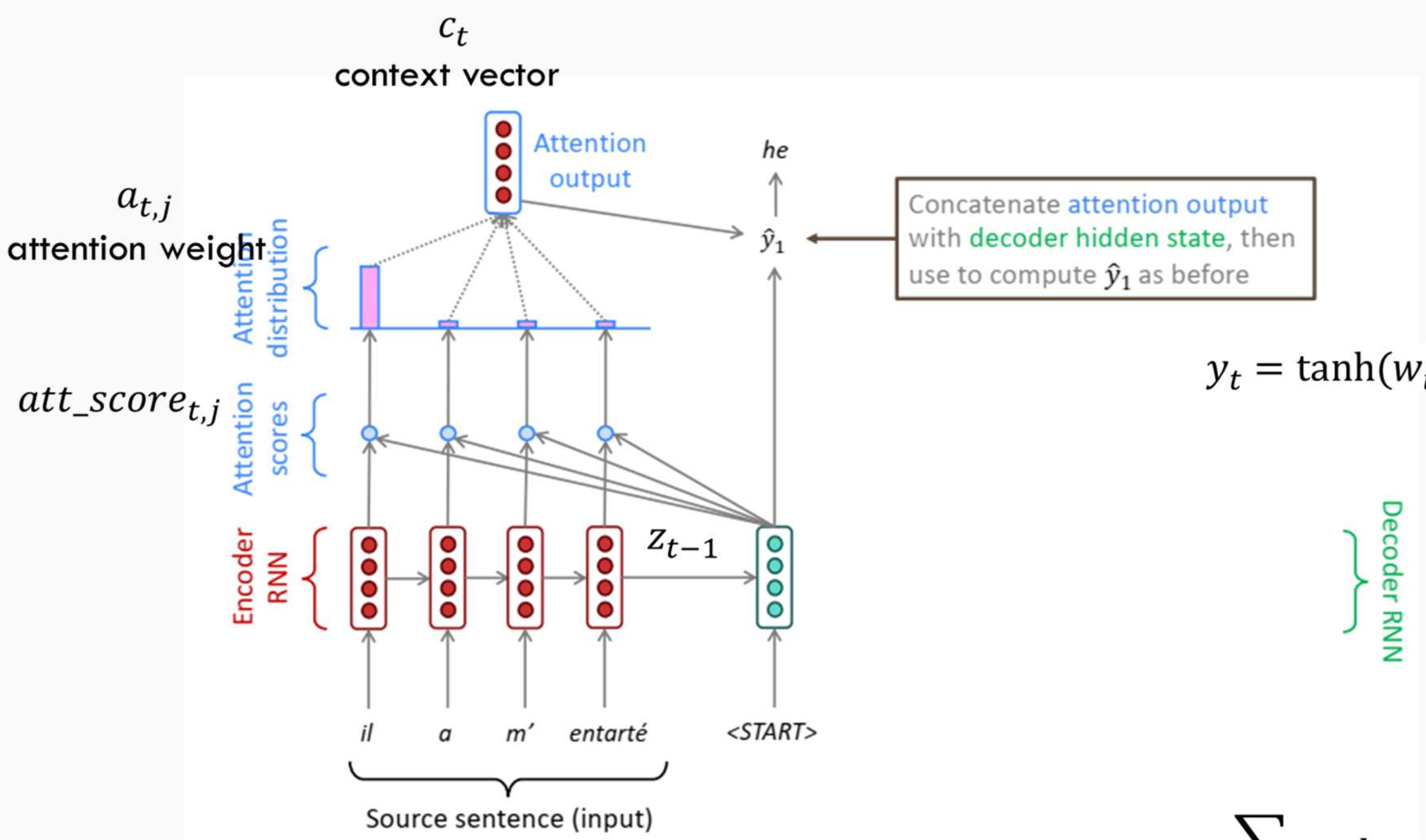
$c_t = \sum_j a_{t,j} h_j$

Redes recurrentes - Mecanismos de atención



$att\_score_{t,j} = \mathrm{dot}(z_t, h_j)$ $\quad a_{t,j} = softmax(att\_score_{t,j})$ $\quad c_t = \sum_j a_{t,j} h_j$

Redes recurrentes - Mecanismos de atención



NUEVO!

$$y_t = \tanh(w_{ih} y_{t-1} + w_{hh} z_{t-1} + w_{ch} c_t)$$

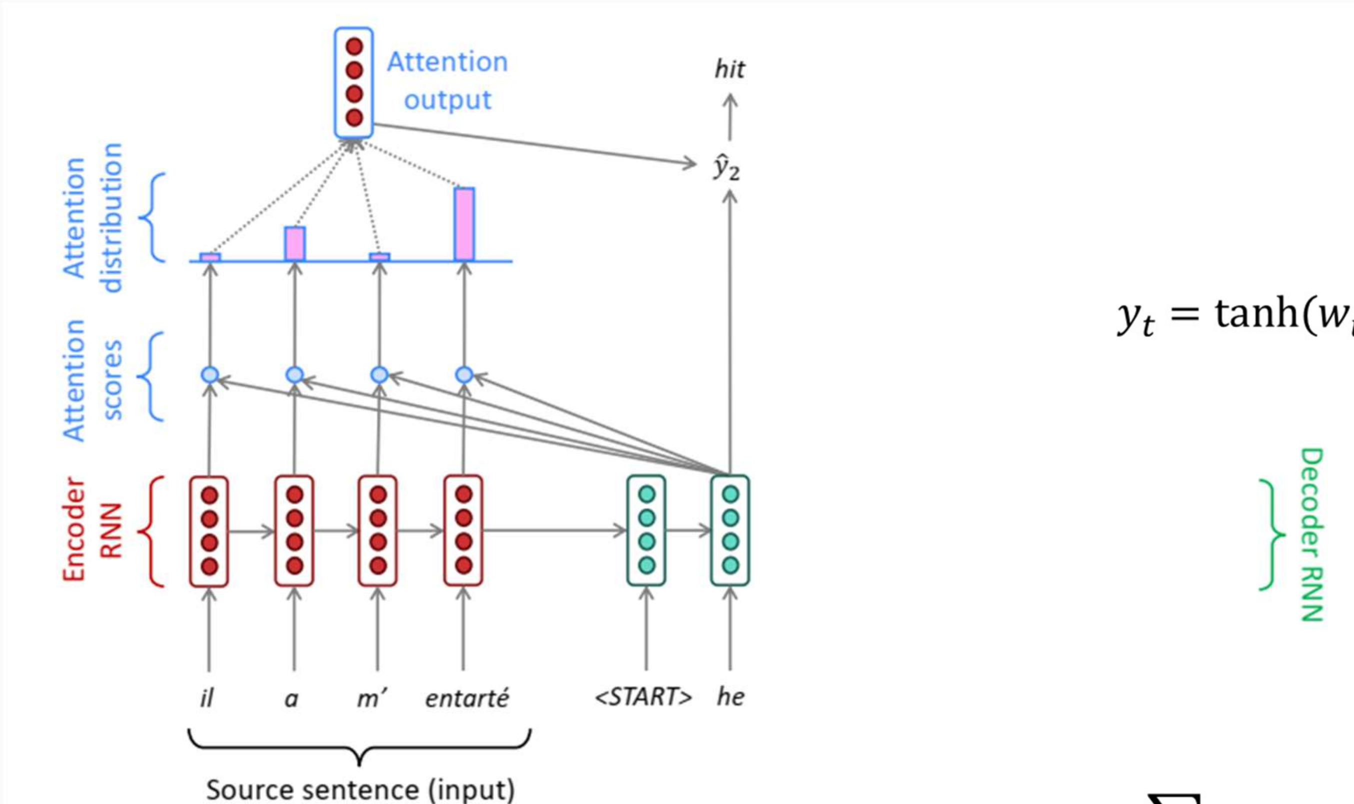
$$att\_score_{t,j} = \mathrm{dot}(z_t, h_j)$$

$$a_{t,j} = softmax(att\_score_{t,j})$$

$$c_t = \sum_j a_{t,j} h_j$$

SOLO CUENTAS!!!
NO HAY
PARAMETROS

Redes recurrentes - Mecanismos de atención



$$y_t = \tanh(w_{ih} y_{t-1} + w_{hh} z_{t-1} + w_{ch} c_t)$$

$$att\_score_{t,j} = \mathrm{dot}(z_t, h_j)$$

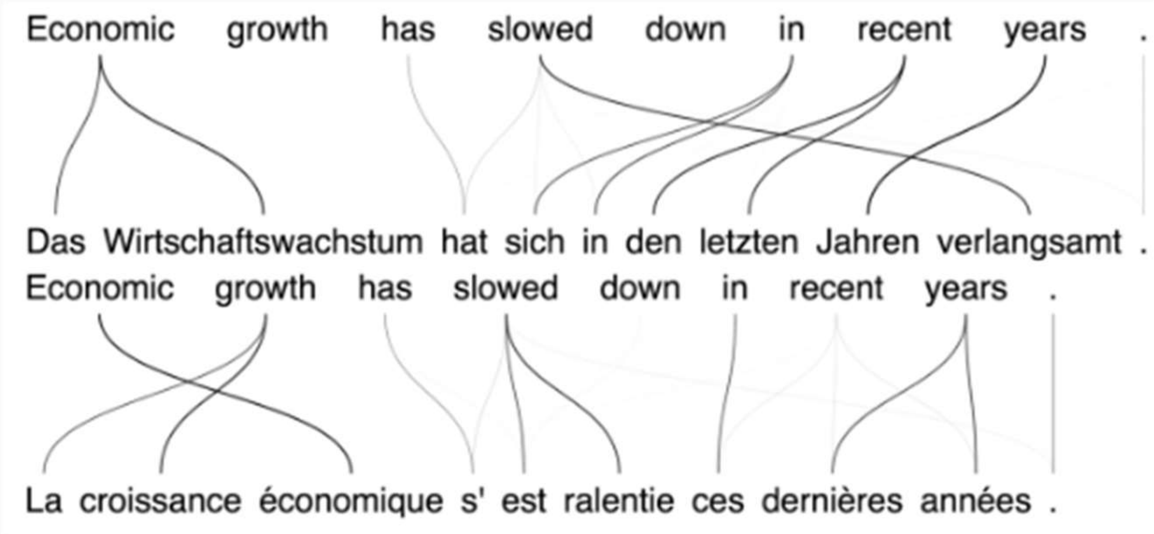
$$a_{t,j} = softmax(att\_score_{t,j})$$

$$c_t = \sum_j a_{t,j} h_j$$

Redes recurrentes - Mecanismos de atención

Desde un punto de vista probabilístico…

el **attention weight** a_j puede ser visto como la probabilidad de que el decoder use esa palabra (representación) para realizar la decodificación del contexto.

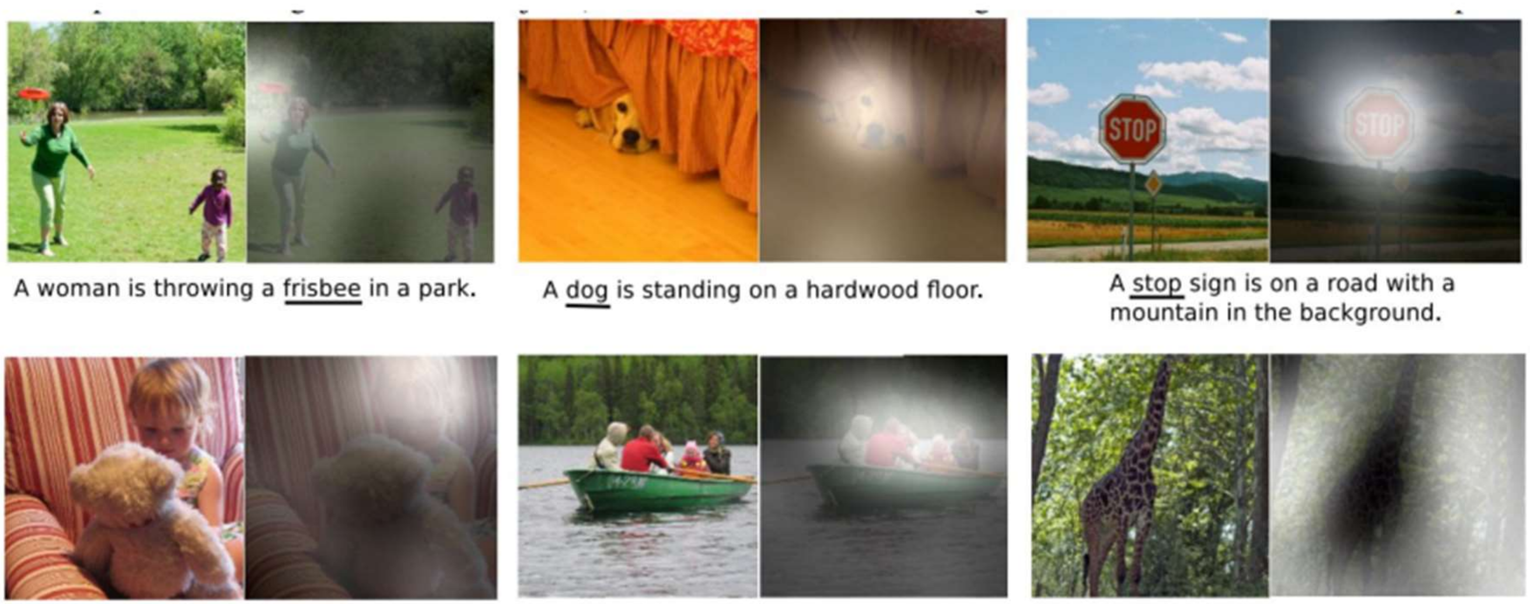


Una definición más general:

Dado un **conjunto de valores** y una **consulta**; el mecanismo de atención devuelve una **suma ponderada** (resumen selectivo) de los valores, **dependiente de la consulta**.

Redes recurrentes - Mecanismos de atención

Image Captioning with Attention



A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf