Carrera de Especialización en Inteligencia Artificial

APRENDIZAJE PROFUNDO

CLASE 6

REDES NEURONALES RECURRENTES

RECURRENT NEURAL NETWORK (RNN)

Docente: Dr. Ing. Marcos Uriel Maillot

Redes Neuronales Recurrentes

Recurrent Neural Network (RNN)

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

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

Recurrent Neural Network (RNN)

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

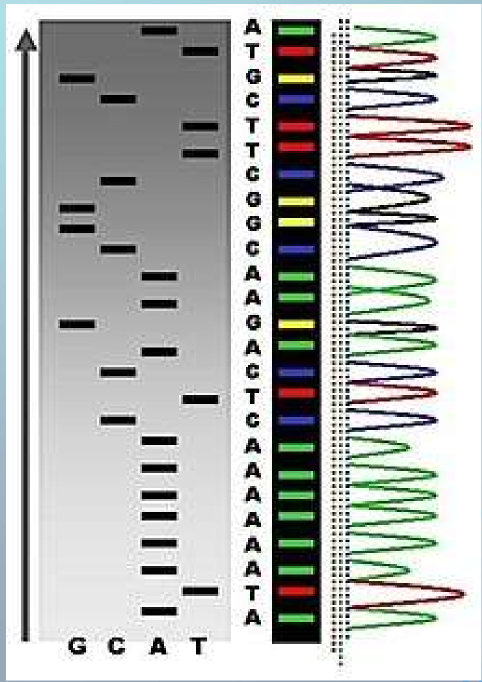
Los pasos, no son necesariamente en unidad tiempo!!

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:

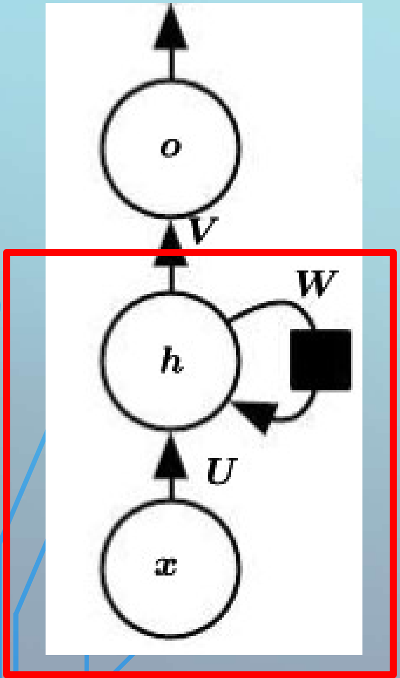
Este	es	un	mensaje	para	la	red	neuronal



wikipedia

Recurrent Neural Network (RNN)

Red recurrente básica



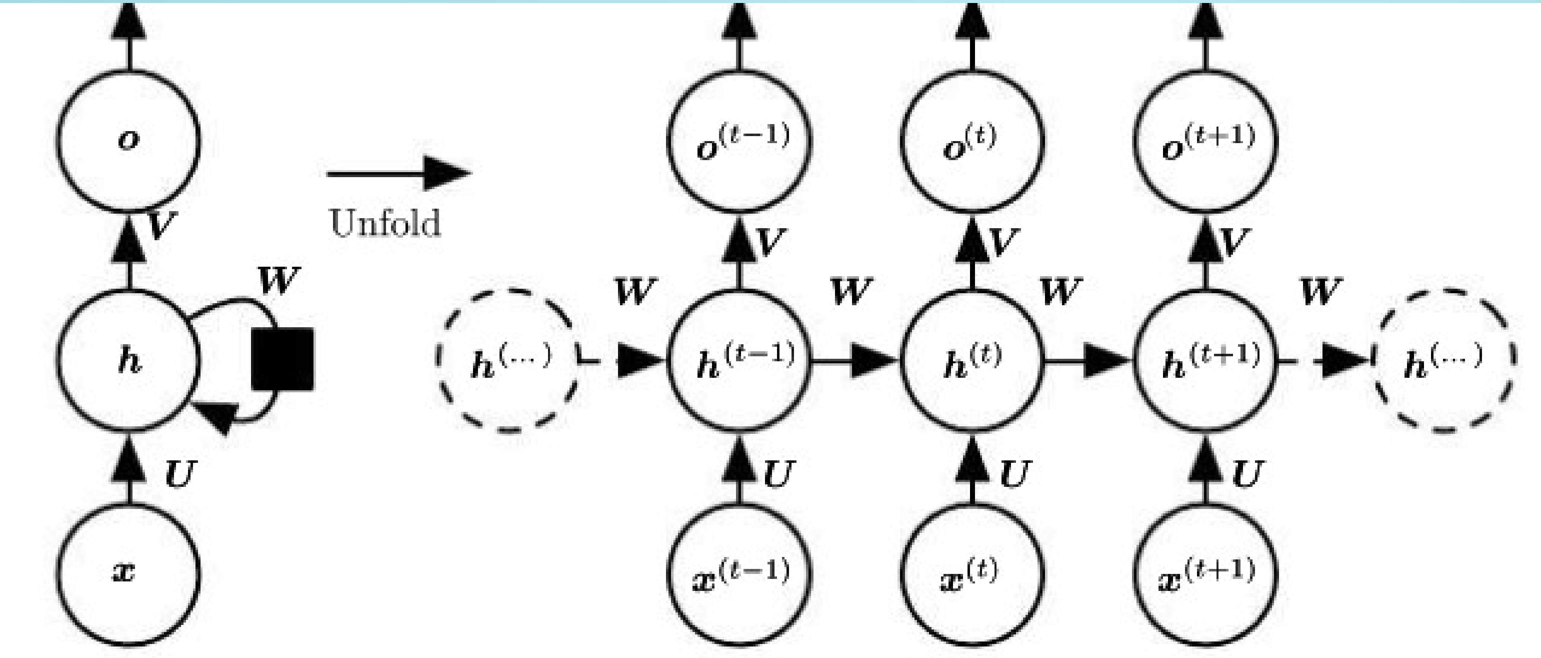
Ecuaciones

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

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

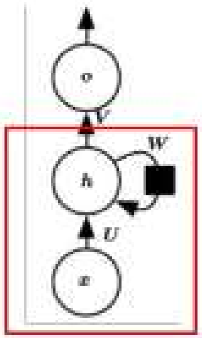
Recurrent Neural Network (RNN)

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



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

Parameters sharing



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

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

$$\left\{ \begin{aligned} &U = 0,2 \quad ; \quad b = -0,1 \\ &W = 0,3 \end{aligned} \right\}$$ Parámetros inventados

$$\left[\begin{aligned} h[0] &= \tanh\left[-0,1 + 0,3 \cdot \underbrace{h[-1]}_{=0 \text{ para la } 1^{\circ} \text{ muestra}} + 0,2 \cdot 0,7\right] = 0,0399 \\ h[1] &= \tanh\left[-0,1 + 0,3 \cdot 0,0399 + 0,2 \cdot 0,9\right] = 0,0917 \\ h[2] &= \tanh\left[-0,1 + 0,3 \cdot 0,0917 + 0,2 \cdot 1,1\right] = 0,1464 \end{aligned} \right]$$

Valido para cada neurona hidden

Los hidden se conectan todas contra todas

Recurrent Neural Network (RNN) - 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', …

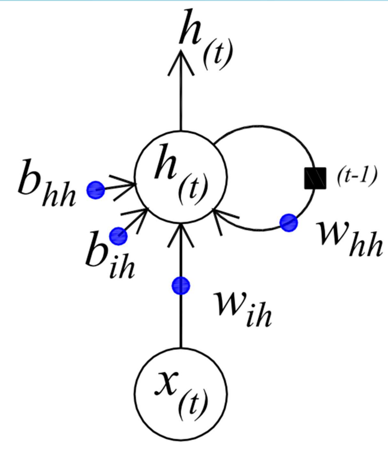
bias=True, batch_first=False, dropout=0, bidirectional=False **)**

Recurrent Neural Network (RNN) - Dimensiones

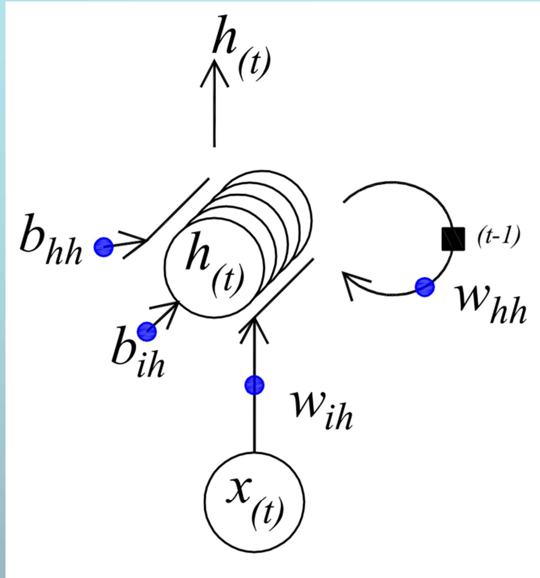
Pytorch

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

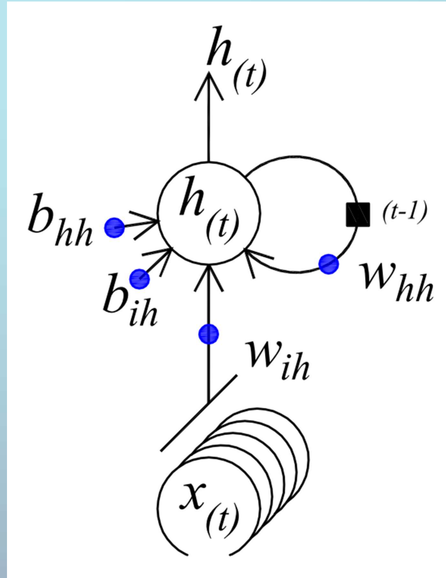
Ver colab RNN_teoria.ipynb



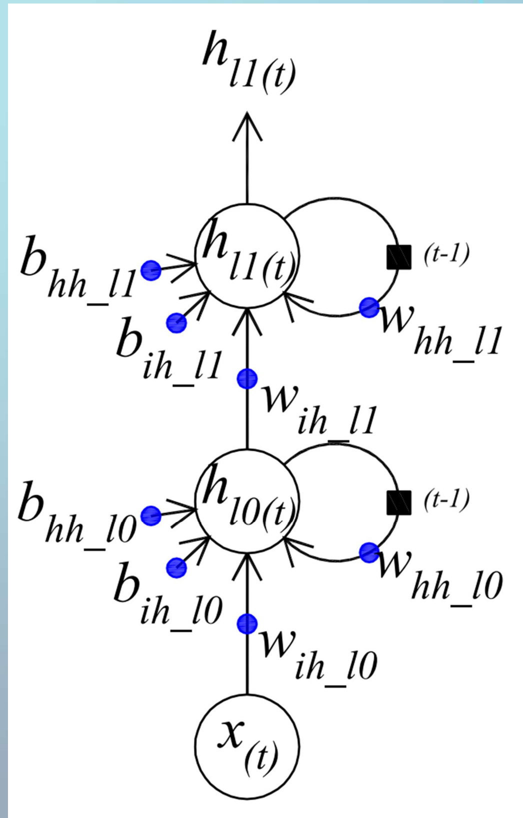
Básica



Varias hidden



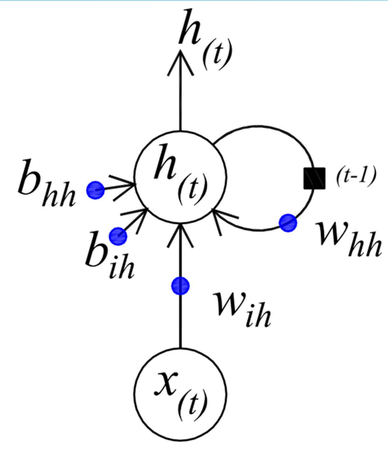
Input multivariable



2 layers

Recurrent Neural Network (RNN) - Dimensiones

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

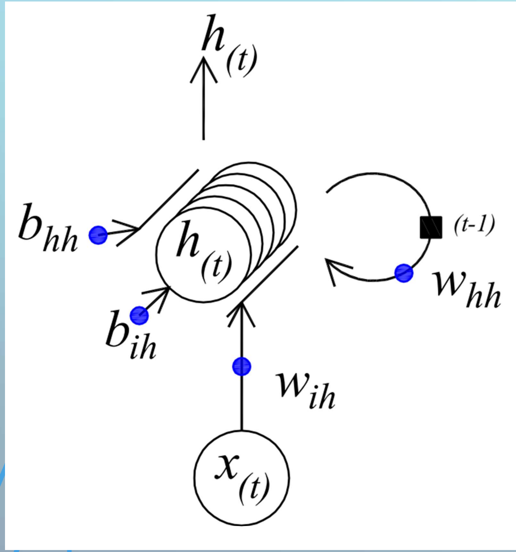


Variable	Tamaño	Parámetro	Tamaño
x		Wih	
h		Bih	
		Whh	
		bhh	

Básica

Recurrent Neural Network (RNN) - Dimensiones

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

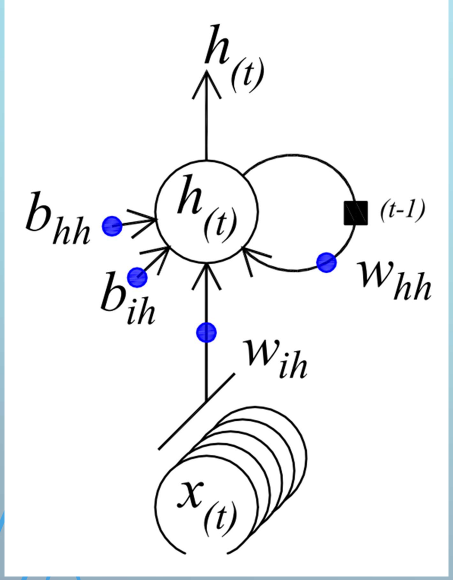


Variable	Tamaño	Parámetro	Tamaño
x		Wih	
h		Bih	
		Whh	
		bhh	

Varias hidden

Recurrent Neural Network (RNN) - Dimensiones

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

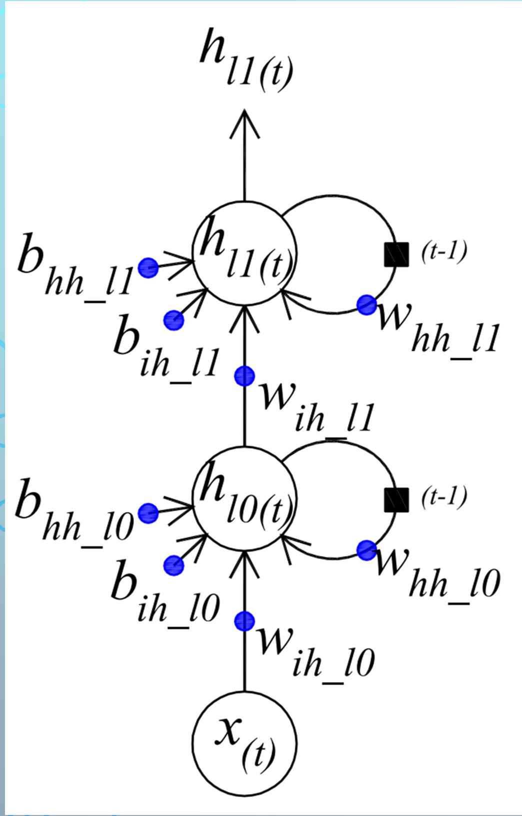


Input multivariable

Variable	Tamaño	Parámetro	Tamaño
x		Wih	
h		Bih	
		Whh	
		bhh	

Recurrent Neural Network (RNN) - Dimensiones

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

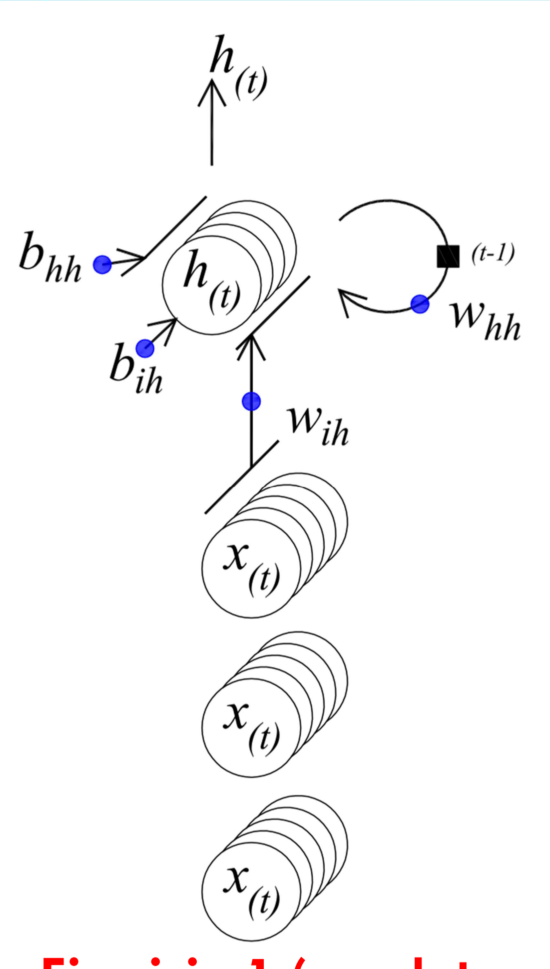


2 layers

Variable	Tamaño	Parámetro	Tamaño
x		Wih	
h		Bih	
		Whh	
		bhh	

Recurrent Neural Network (RNN) - Dimensiones

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



Ejemplo A

Variable	Tamaño	Parámetro	Tamaño
x		Wih	
h		Bih	
		Whh	
		bhh	

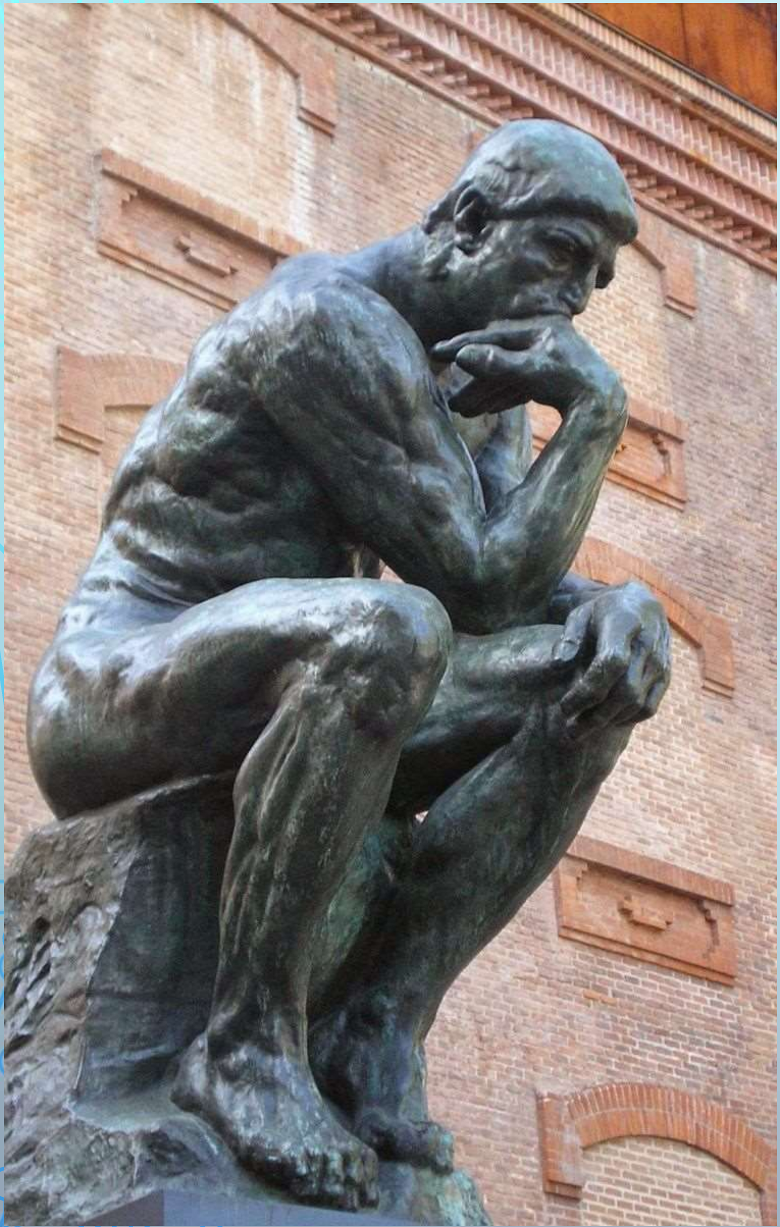
Ejemplo B

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)



“El pensador” de Rodin

A pensar!

Recurrent Neural Network (RNN)

BACK PROPAGATION THROUGH TIME [BPTT]

Ver desarrollo teórico

Recurrent Neural Network (RNN)

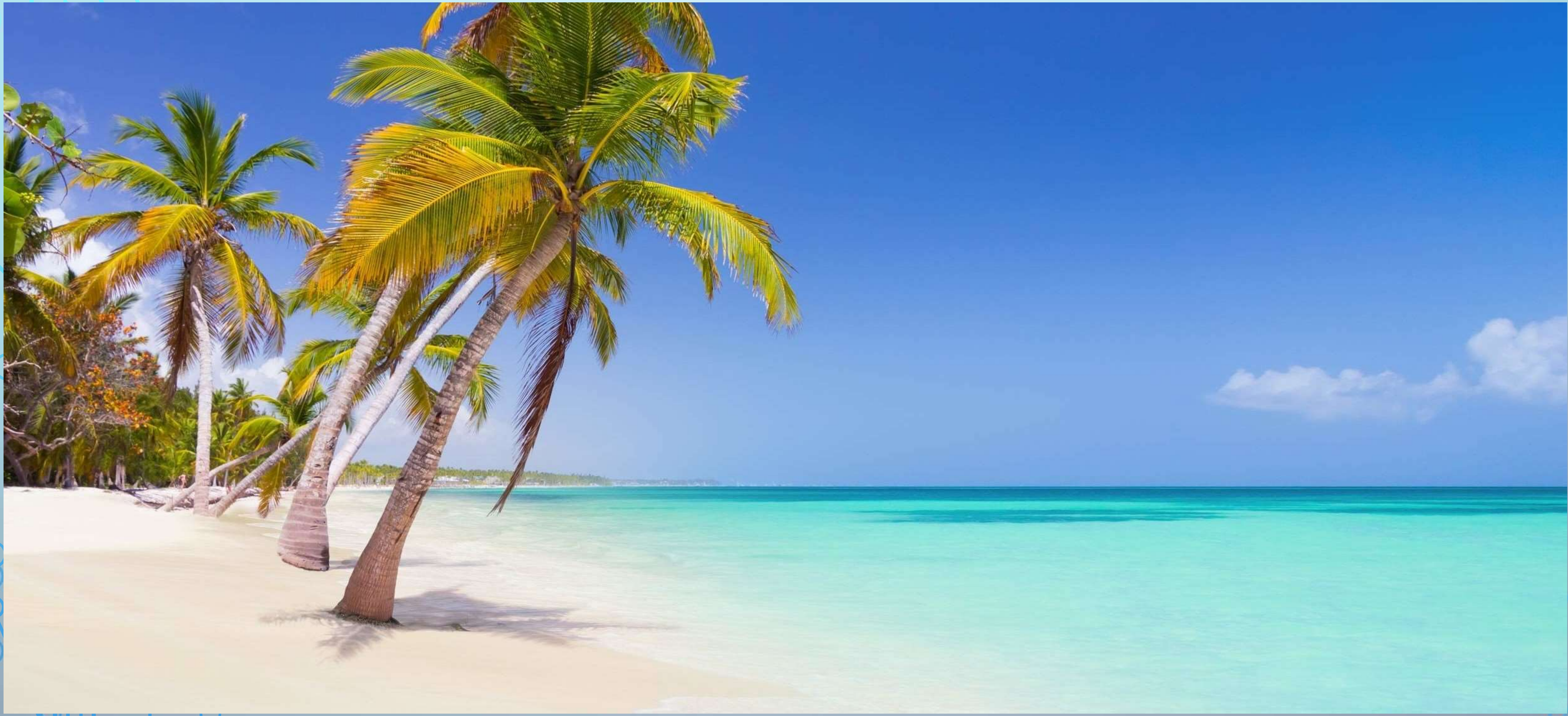
Problemas de la RNN básica con el BPTT

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

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

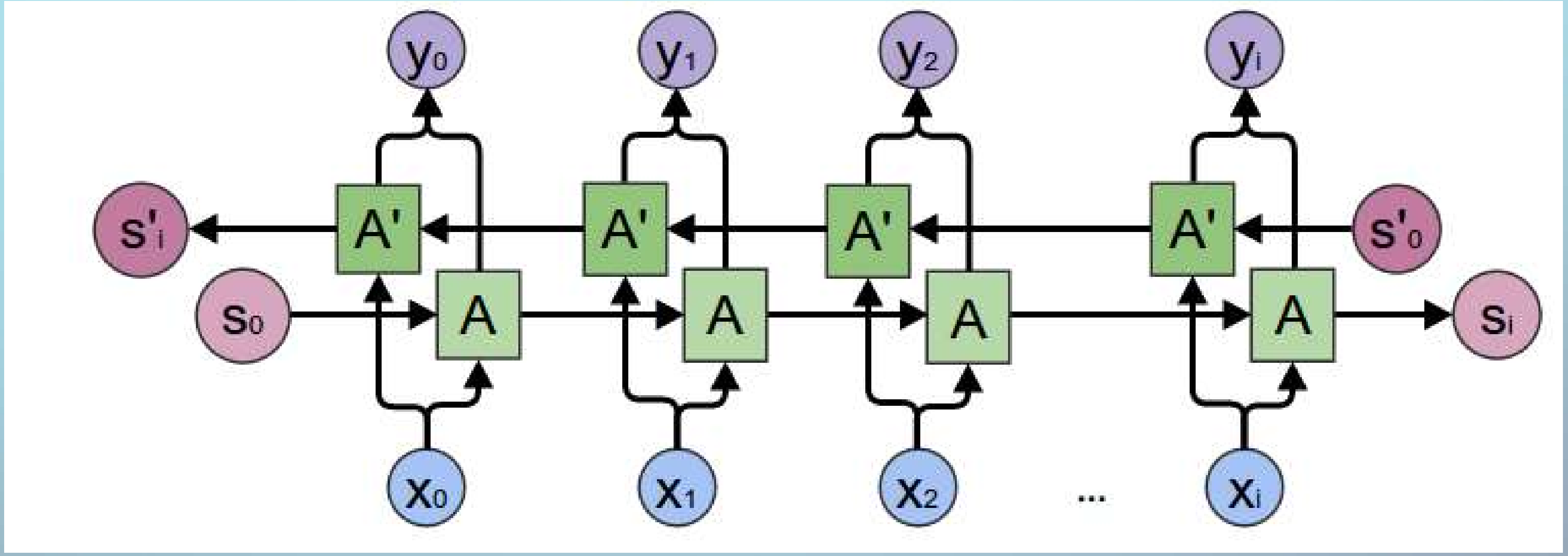
Solución con otras RNN mas avanzadas (LSTM y GRU)

¡Un merecido descanso!



Recurrent Neural Network (RNN)

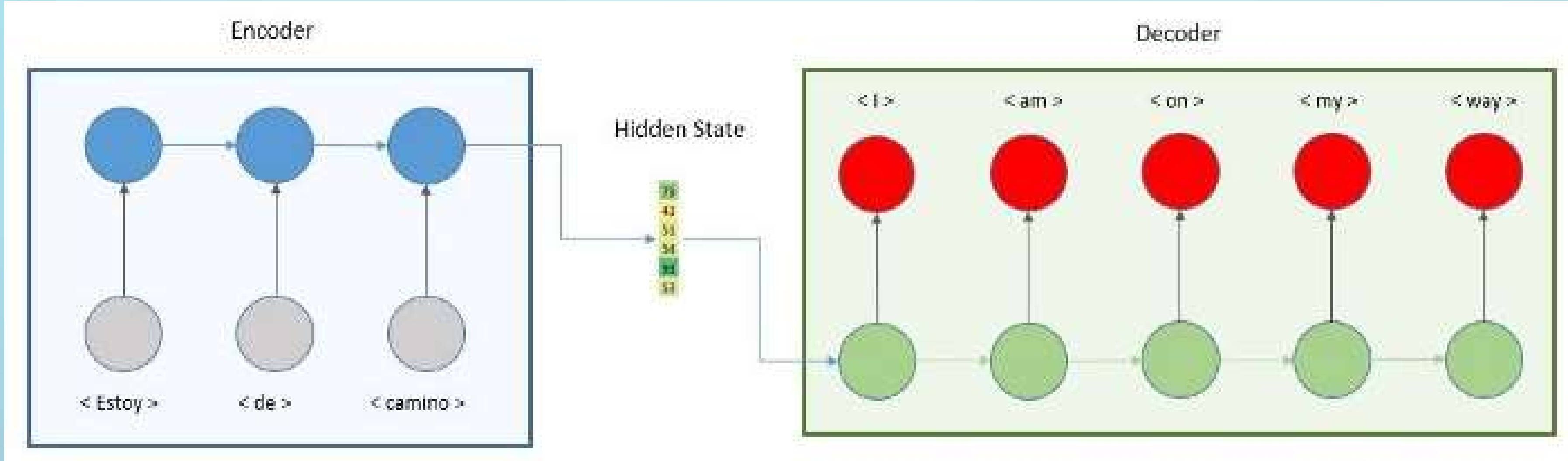
Bi-directional RNN

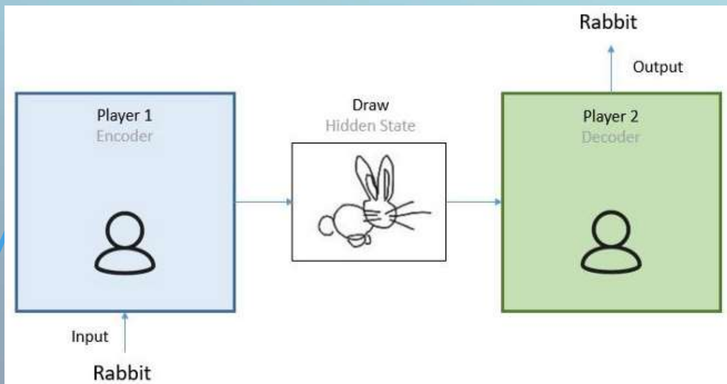


Para la traducción, suele ser útil tener la frase entera.

Recurrent Neural Network (RNN)

Arquitectura **encoder/decoder** o **seq-to-seq**





2 RNN de distinto tamaño

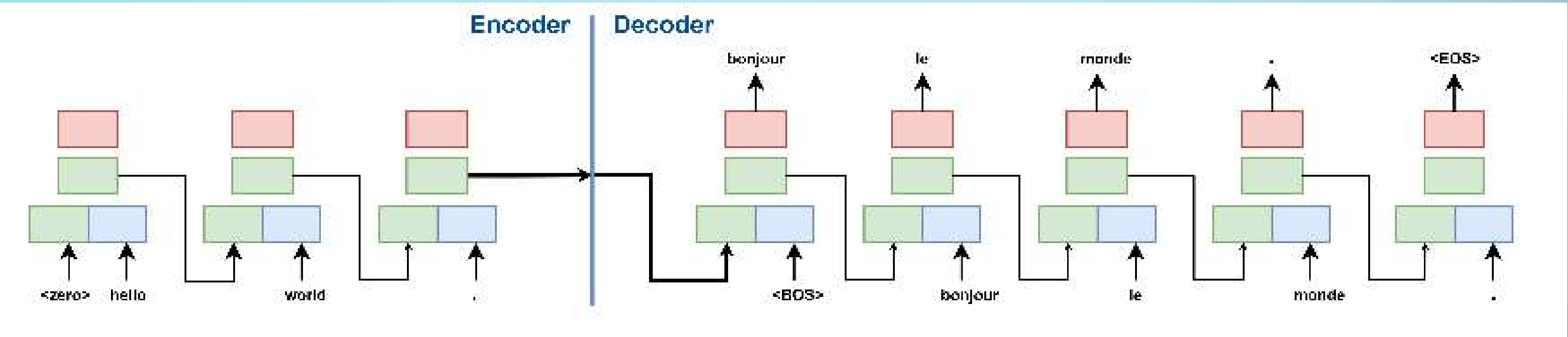
1 Hidden state que "resume" toda la información de la input.

Flexibilidad máxima para inputs/outputs de distinta longitud

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

Recurrent Neural Network (RNN)

Arquitectura **encoder/decoder** o **seq-to-seq**

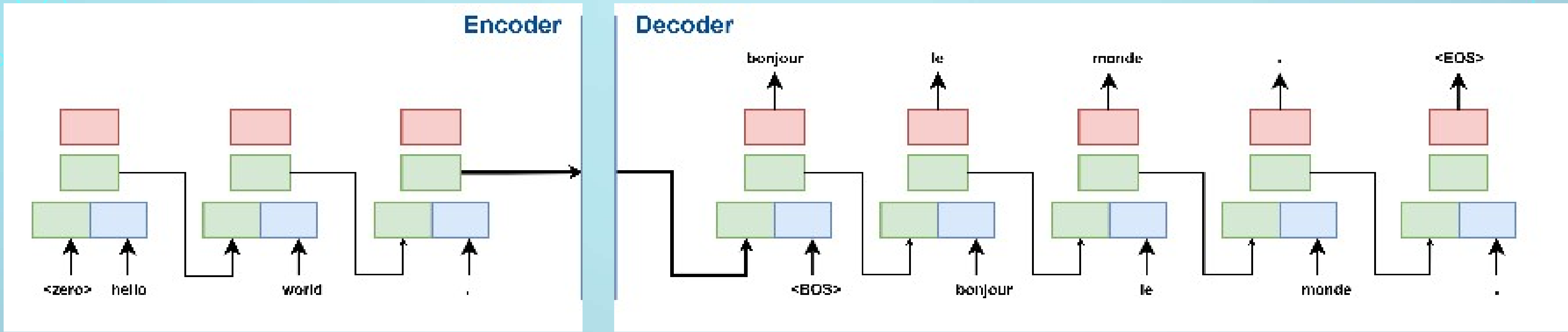


Unfolded!!!!

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

Recurrent Neural Network (RNN)

Entrenamiento y uso del **encoder/decoder** o **seq-to-seq**



ENCODER

Siempre leen la secuencia entera

Emiten un hidden state final

DECODER

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

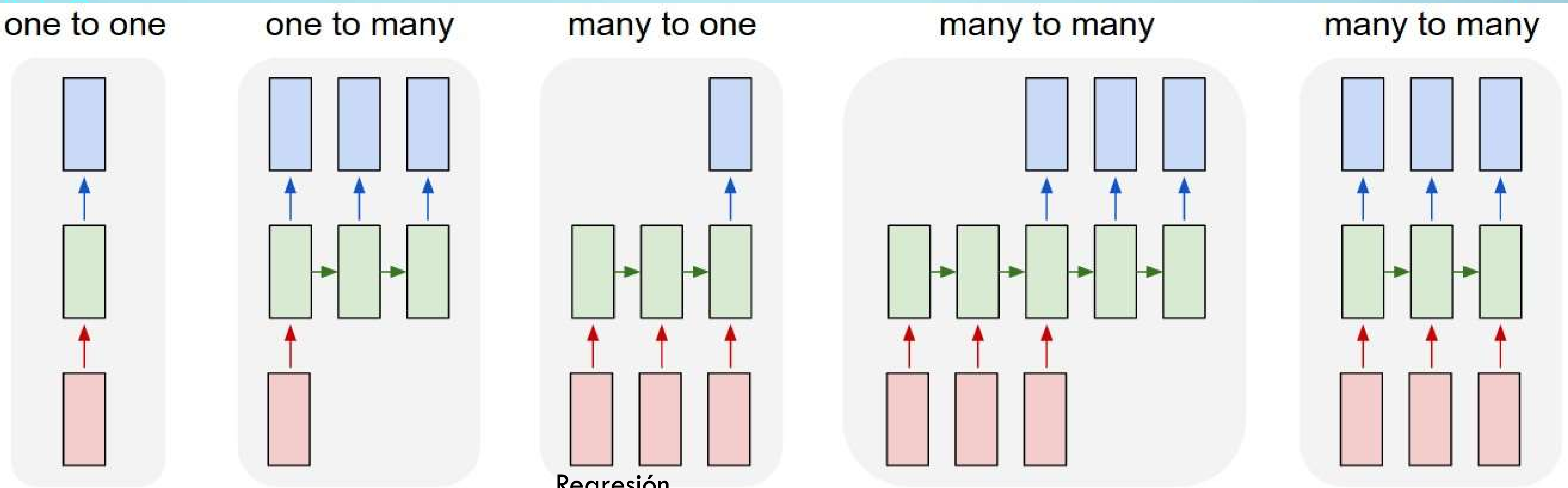
Uso → while last_token =! <EOS>:
genero_token

TEACHER FORCING!!!

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

Recurrent Neural Network (RNN)

Arquitecturas flexibles IN/OUT



MLP

Img captioning

Regresión

Classificacion

Sentiment classification

Resumen de videos

Regresión varios futuros

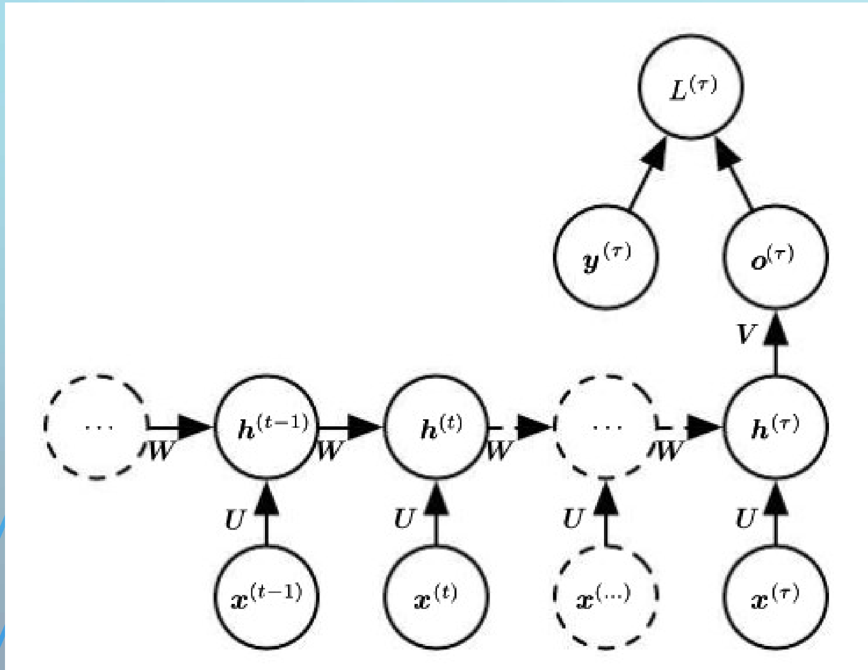
Traducción

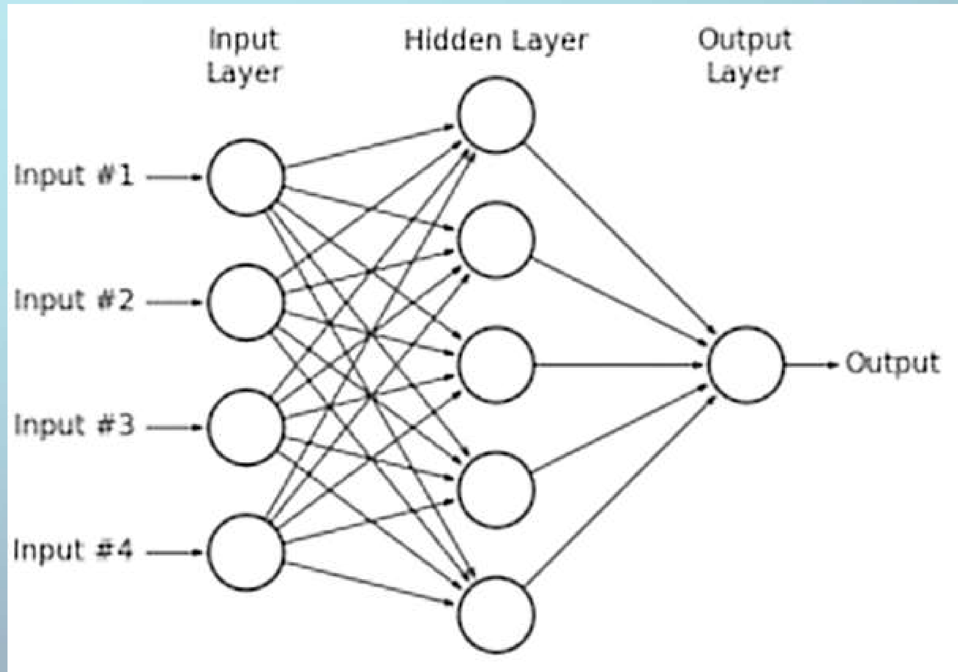
Video class frame lvl

Recurrent Neural Network (RNN) vs Tapped Delayed MLP

TP

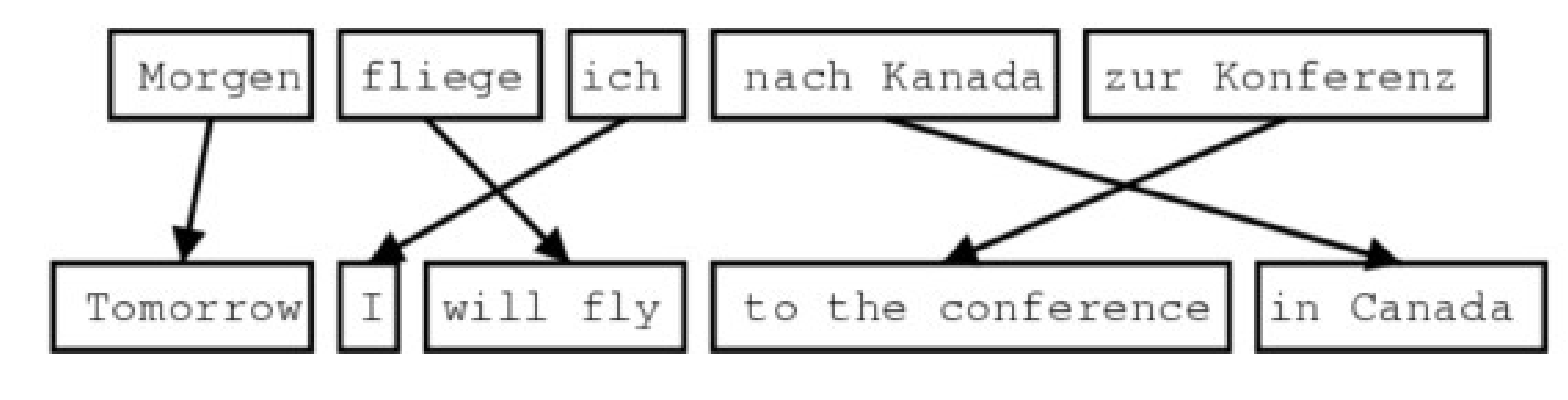
X(t=0)	X(t=1)	X(t=2)	X(t=3)	X(t=4)	X(t=5)	X(t=6)	X(t=7)	…	X(t=τ)





Ver colab RNN_signal_TP.ipynb

Recurrent Neural Network (RNN) – Self Attention!!



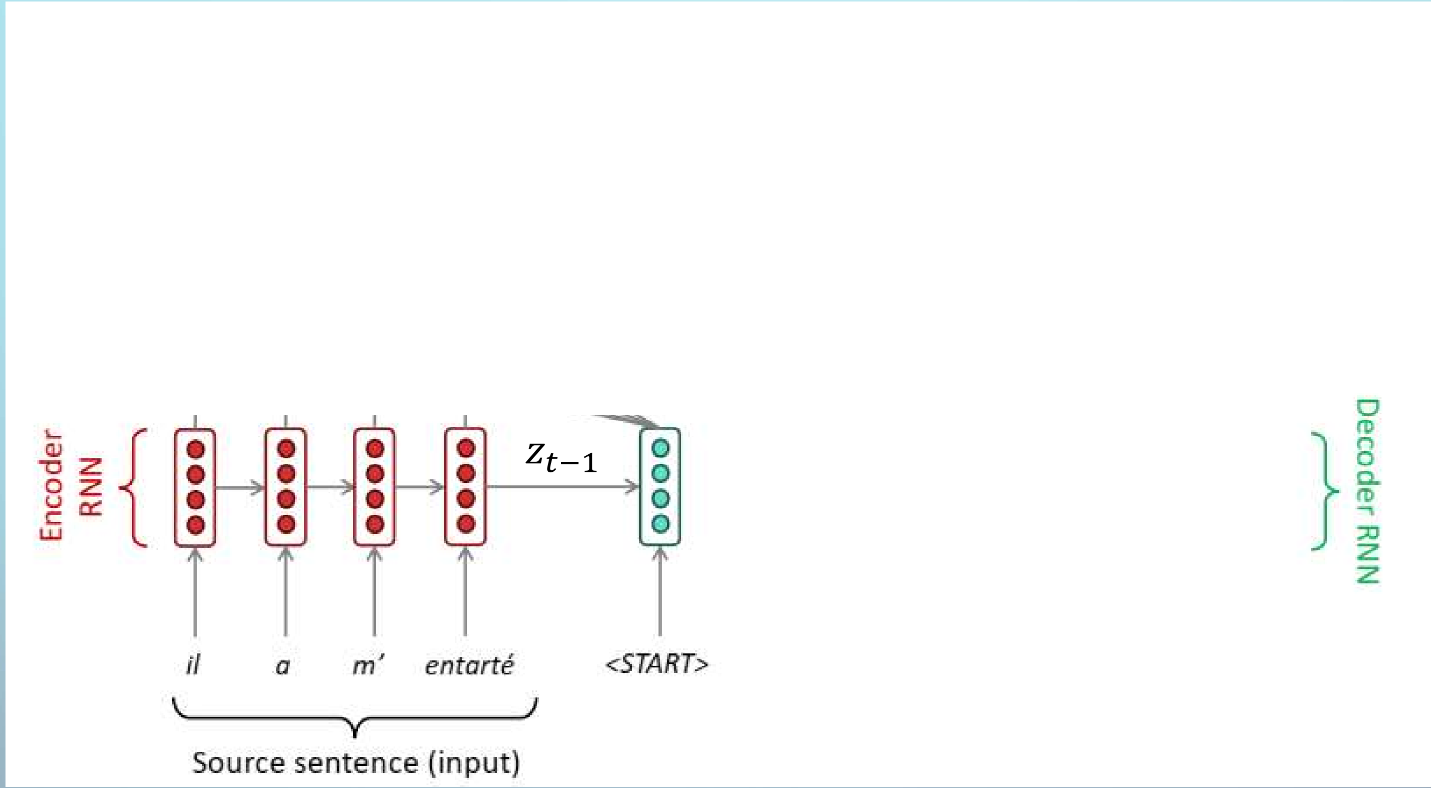
*El mecanismo de atención permite al decoder **utilizar las partes más relevantes** de la entrada **como una suma ponderada** del vector de entrada codificados para predecir la siguiente palabra.*

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

Recurrent Neural Network (RNN) – Self Attention!!

Ver en bibliografía

cs224n-2021-lecture07-nmt.pdf

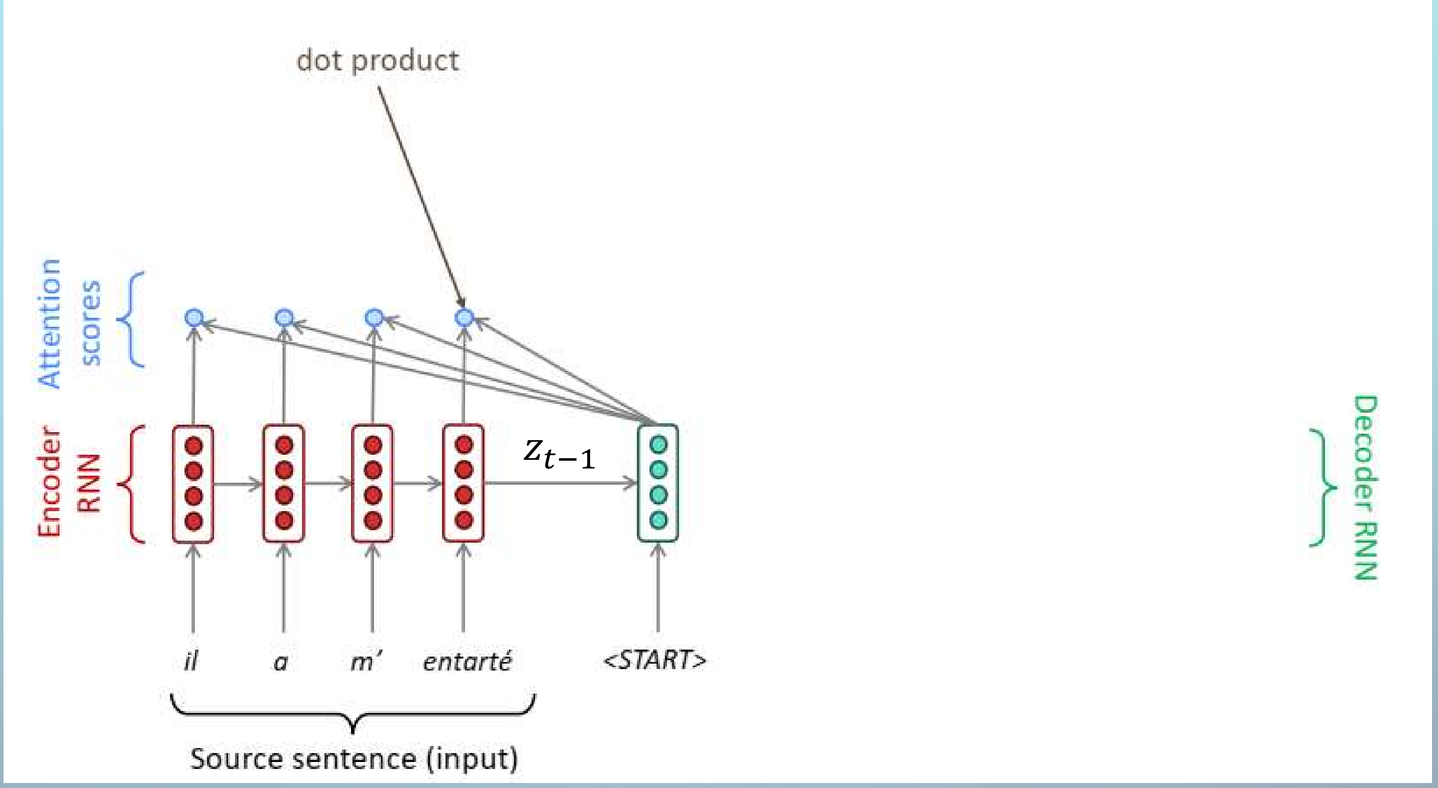


Recurrent Neural Network (RNN) – Self Attention!!

Ver en bibliografía

cs224n-2021-lecture07-nmt.pdf

$att\_score_{t,j}$

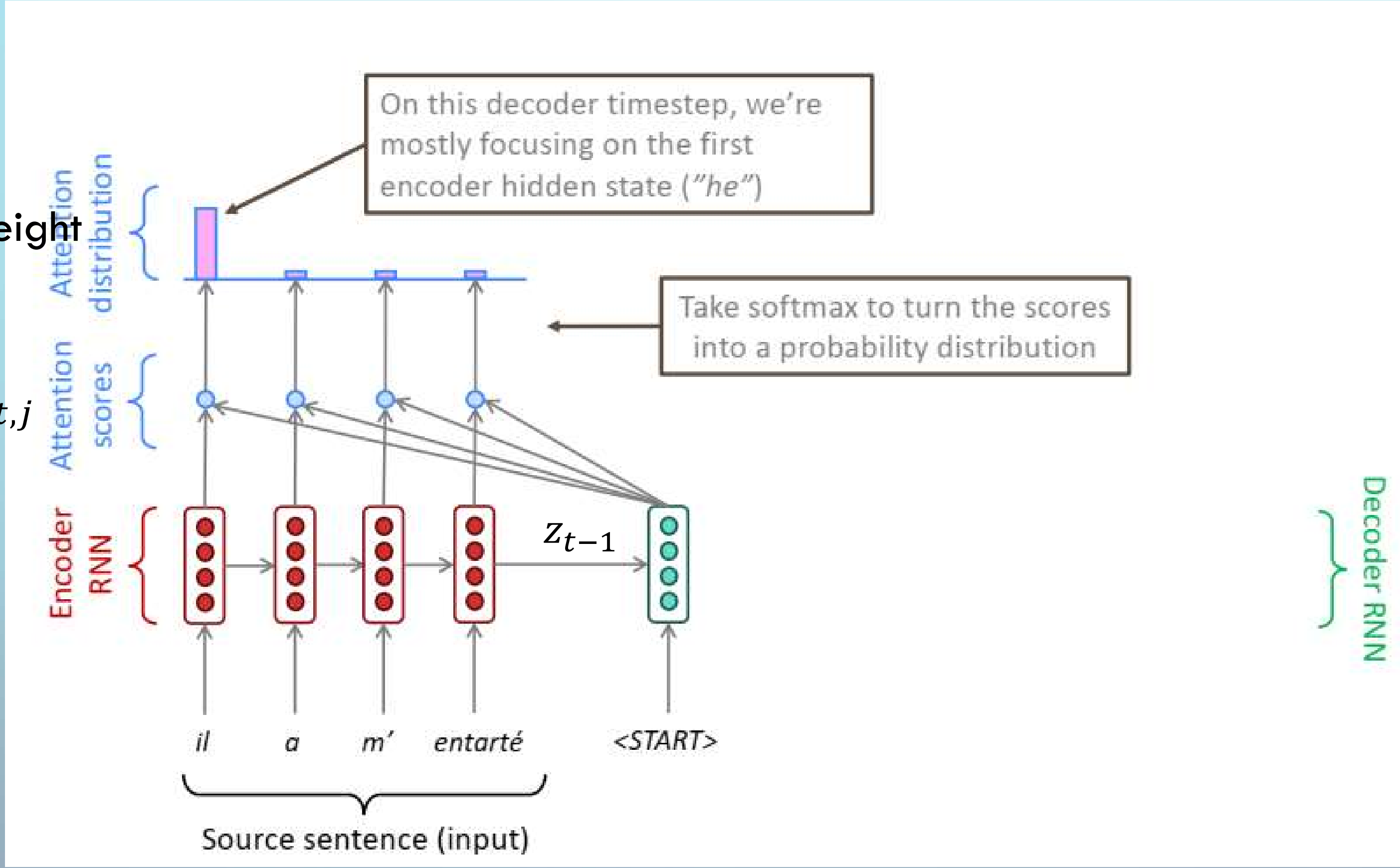


$att\_score_{t,j} = \mathrm{dot}(z_t, h_j)$

Recurrent Neural Network (RNN) – Self Attention!!

$a_{t,j}$
attention weight

$att\_score_{t,j}$



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

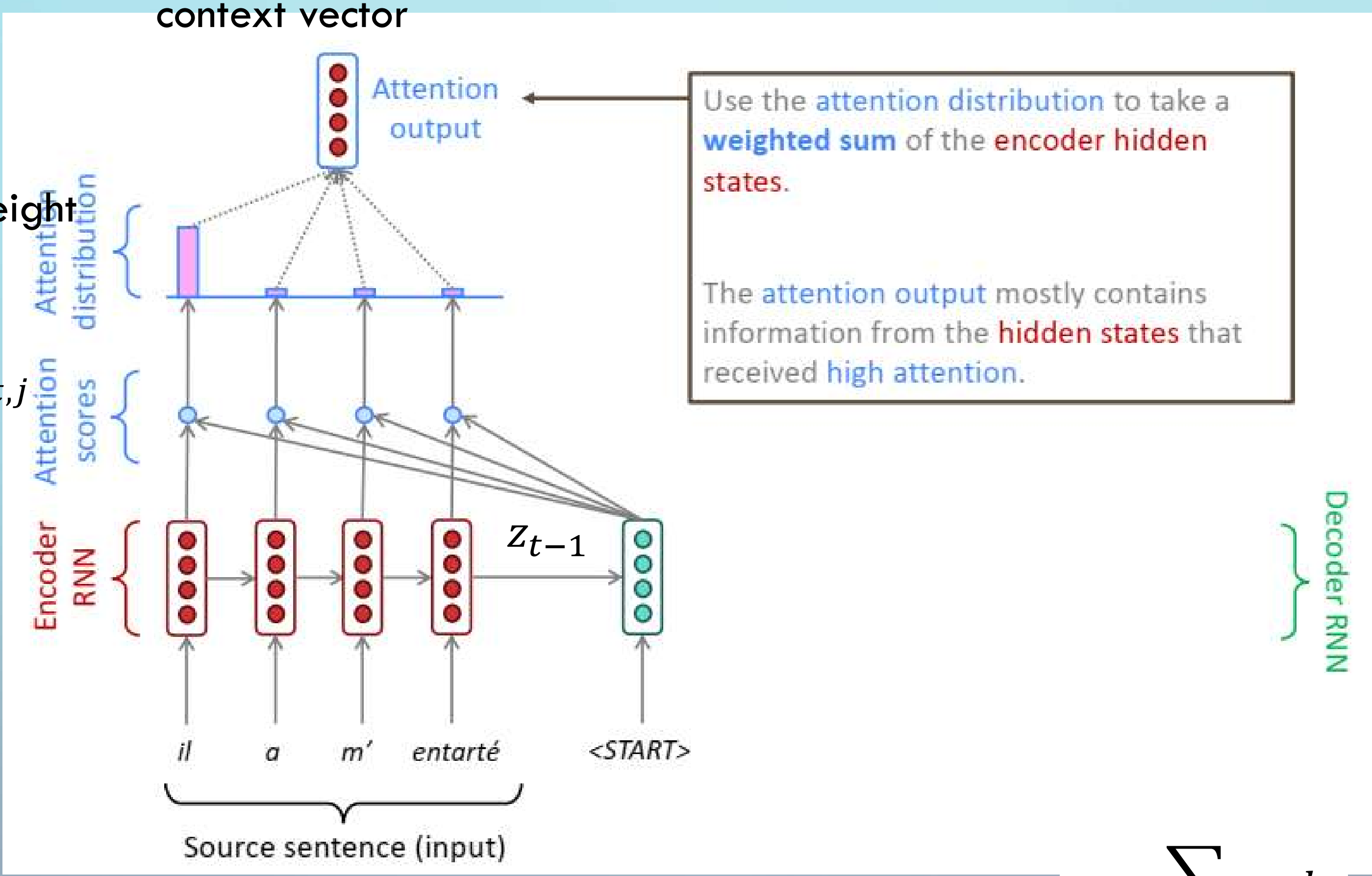
$a_{t,j} = softmax(att\_score_{t,j})$

Recurrent Neural Network (RNN) – Self Attention!!

c_t
context vector

$a_{t,j}$
attention weight

$att\_score_{t,j}$

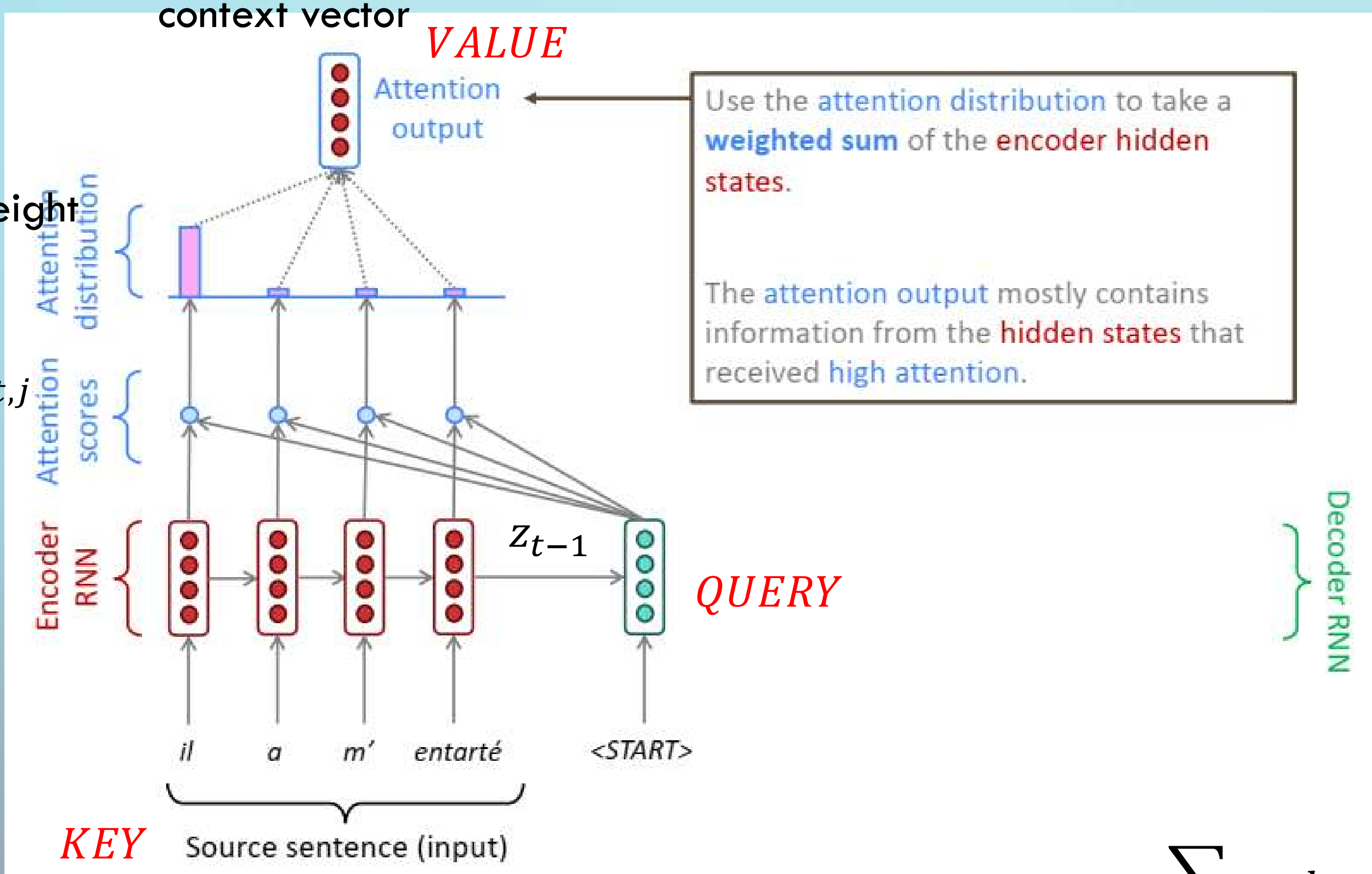


$att\_score_{t,j} = \mathrm{dot}(z_t, h_j)$

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

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

Recurrent Neural Network (RNN) – Self Attention!!



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

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

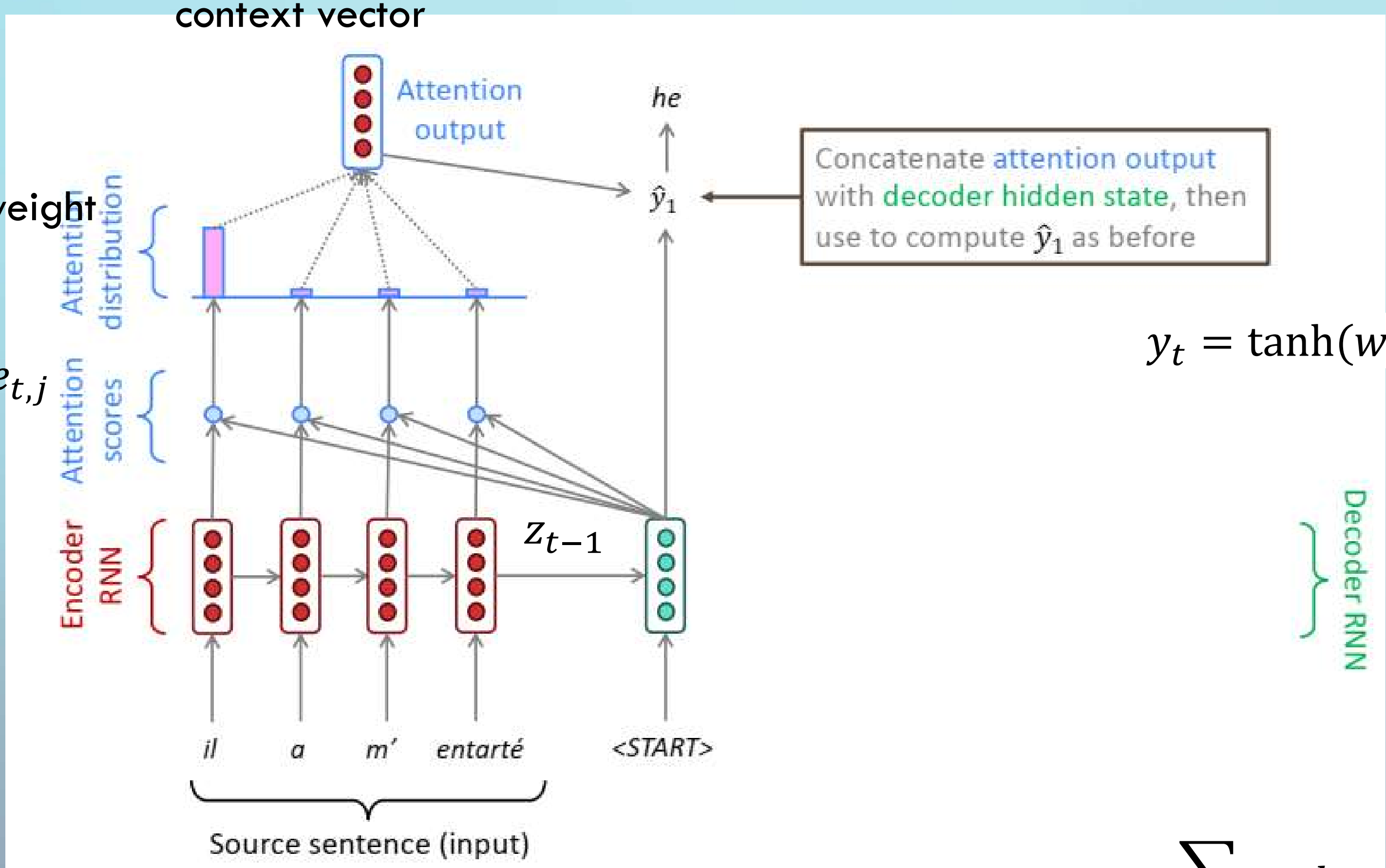
$$c_t = \sum_j a_{t,j} h_j$$

Recurrent Neural Network (RNN) – Self Attention!!

c_t
context vector

$a_{t,j}$
attention weight

$att\_score_{t,j}$



NUEVO!

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

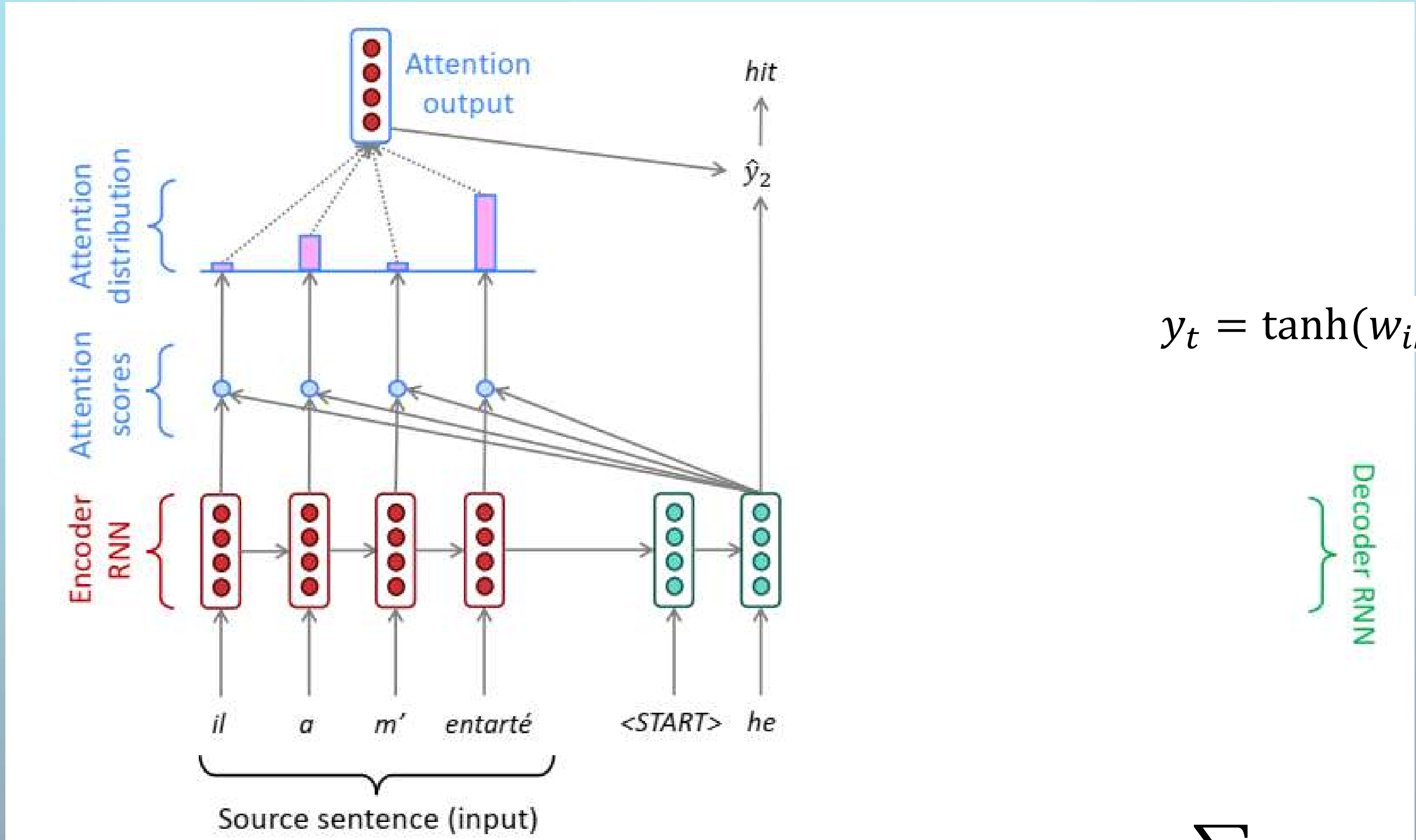
$$att\_score_{t,j} = \mathrm{dot}(z_t, h_j)$$

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

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

SOLO CUENTAS!!!
NO HAY
PARAMETROS

Recurrent Neural Network (RNN) – Self Attention!!



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

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

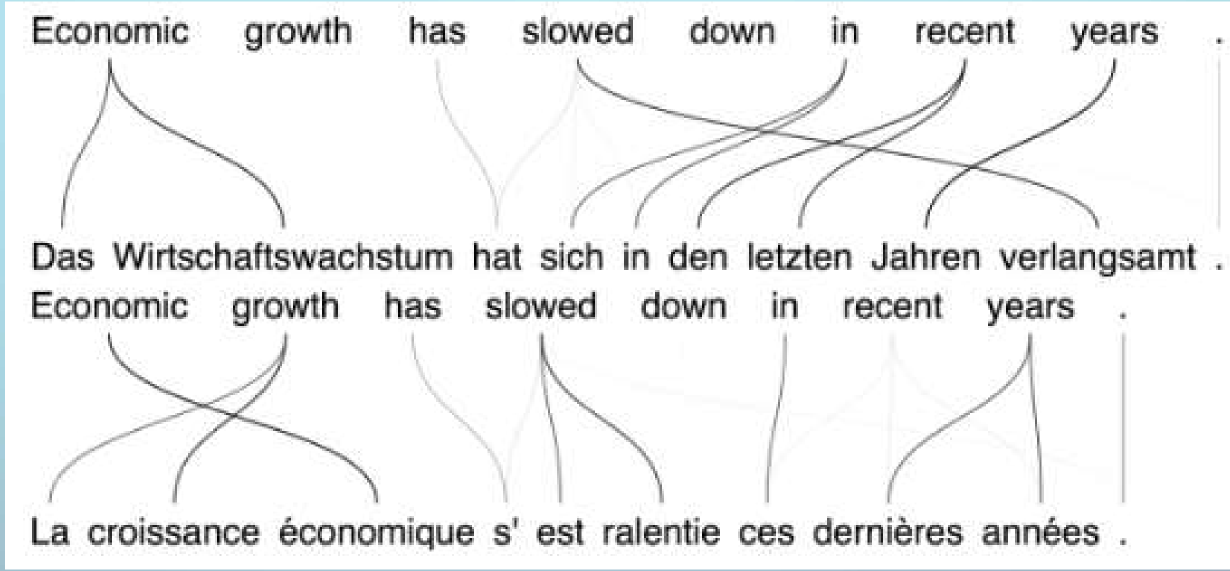
$$a_{t,j} = softmax(att\_score_{t,j})$$

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

Recurrent Neural Network (RNN) – Self Attention!!

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**.

Recurrent Neural Network (RNN)

Attention se aplica a mas que NLP (o traducción)

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