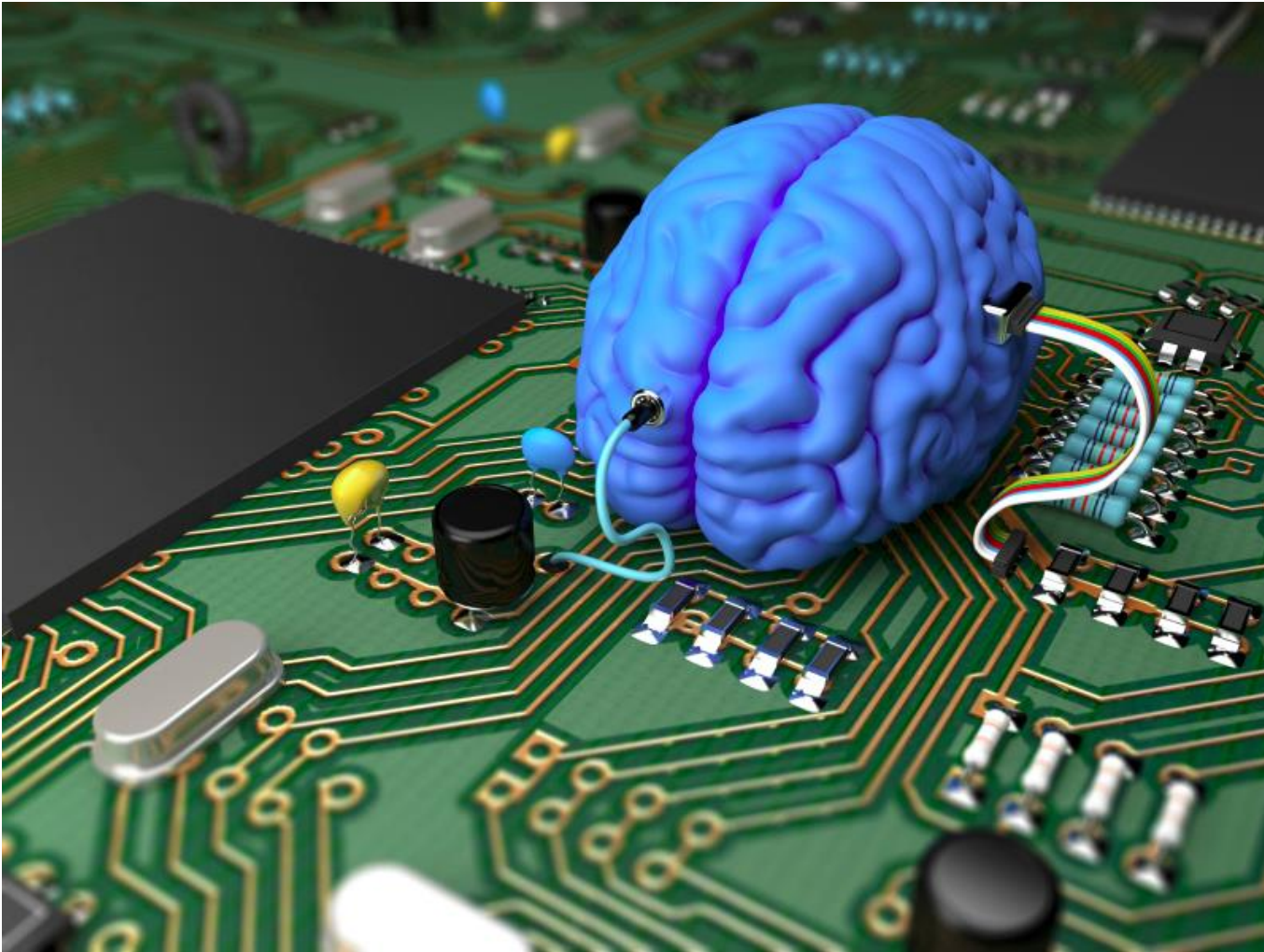


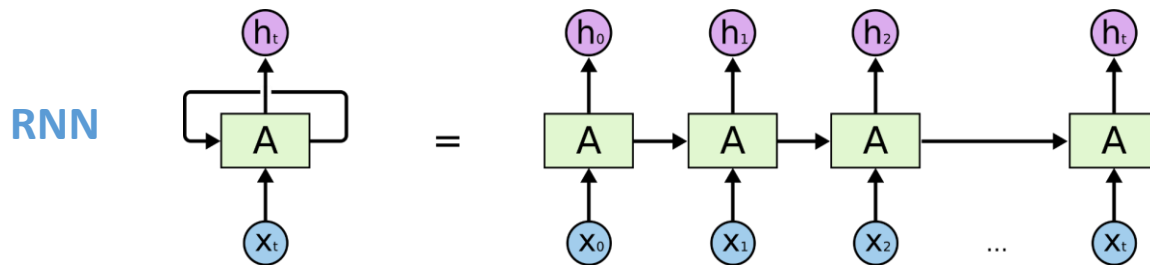
# Machine learning I : RNN



# Introducción

## Los pensamientos son persistentes


- Uno de los problemas de las redes neuronales tradicionales es que **no pueden** emular esta persistencia
- **Problema:** Intentar clasificar la acción de cada fotograma en una película  $\rightarrow$  No está claro como una red tradicional puede meter la información de los primeros fotogramas de la película para clasificar los siguientes.



# RNN: Aplicaciones

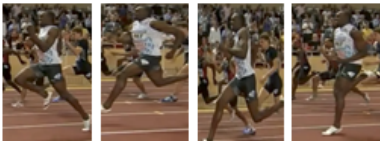
Grabación de audio → Texto

Estilo musical → Melodía

“Esta película no tiene nada especial” → 

AGCAAACGTGAATCGGA → AGCAAACGTGAATCGGA ¿Qué parte de esta secuencia es una proteína?

Хотели как лучше, а получилось как всегда → Lo intentamos hacer bien pero salió como siempre



→ Correr

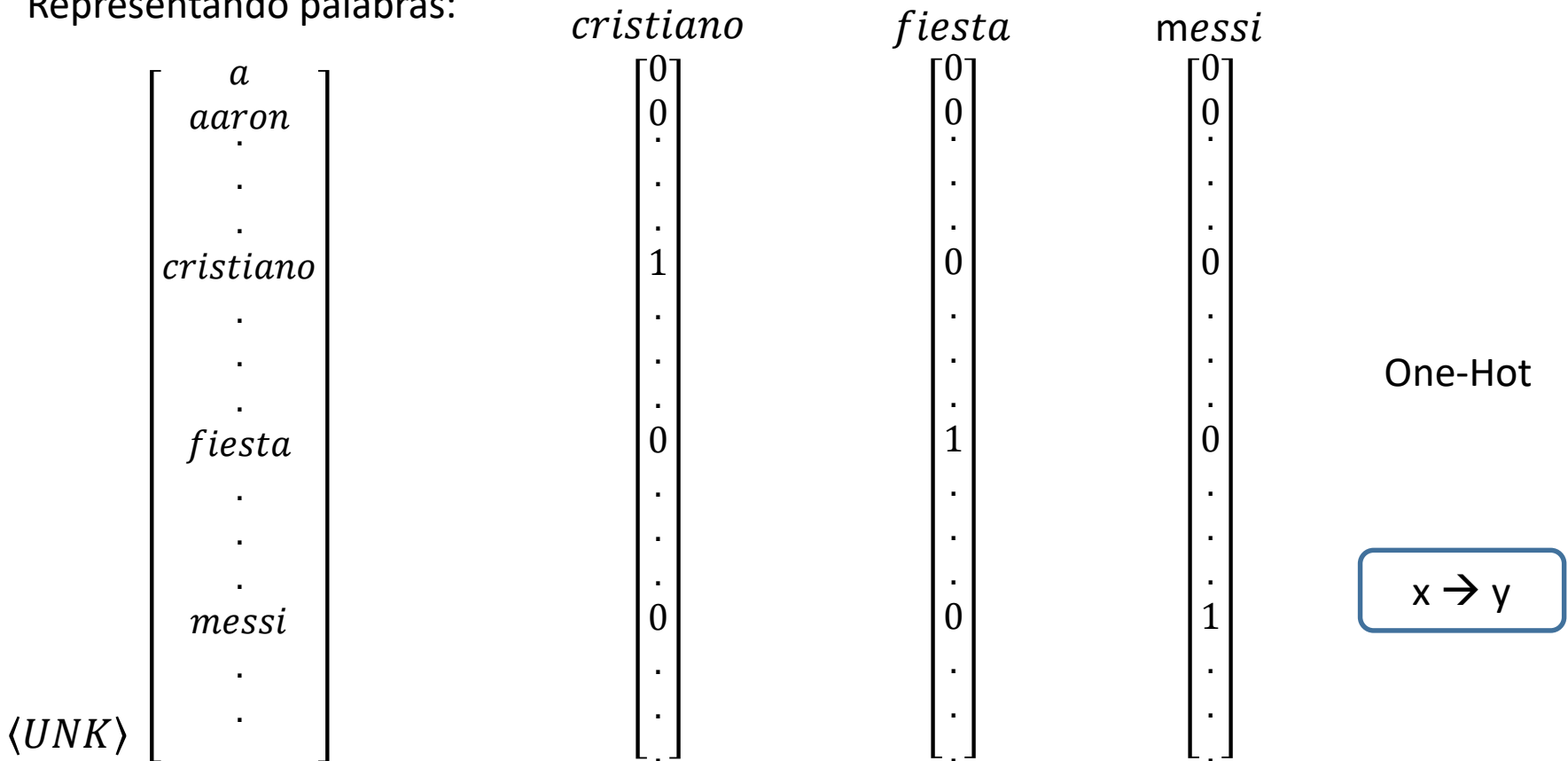
Todos estos problemas pueden ser tratados como problemas de aprendizaje supervisado

# Ejemplo: NER

## Reconocimiento de entidades nombradas (NER)

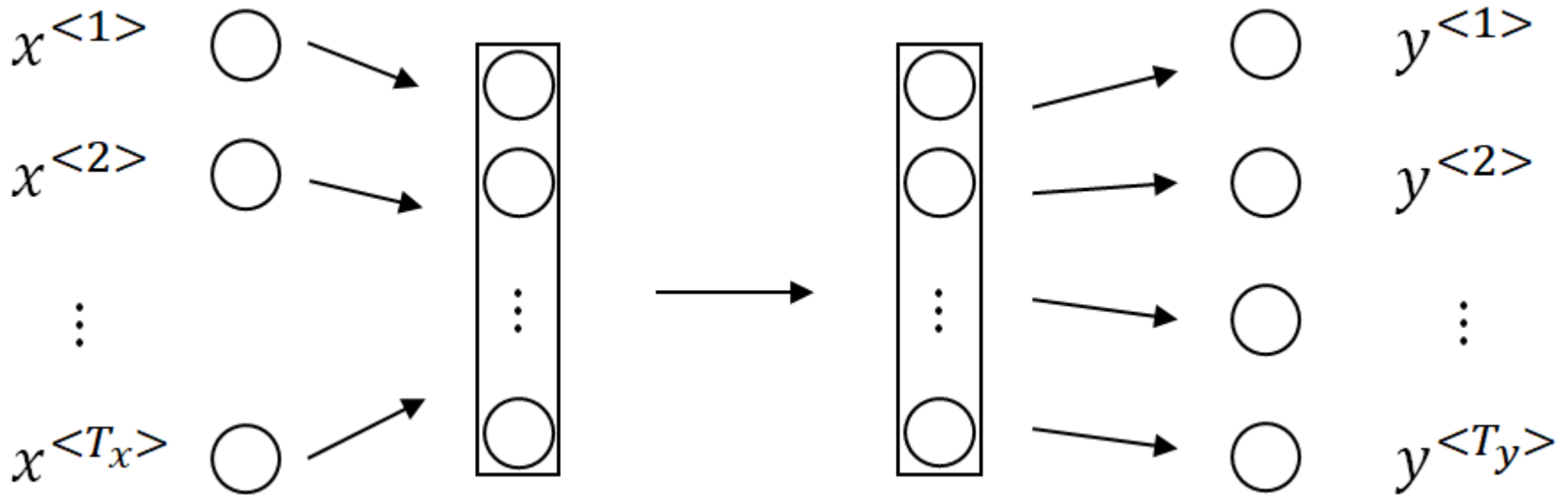
x <=>	Cristiano	Ronaldo	Sabotea	La	Fiesta	De	Cumpleaños	De	Lionel	Messi	T <sub>x</sub> =10
y <=>	1	1	0	0	0	0	0	0	1	1	T <sub>y</sub> =10

Representando palabras:



# Ejemplo: NER

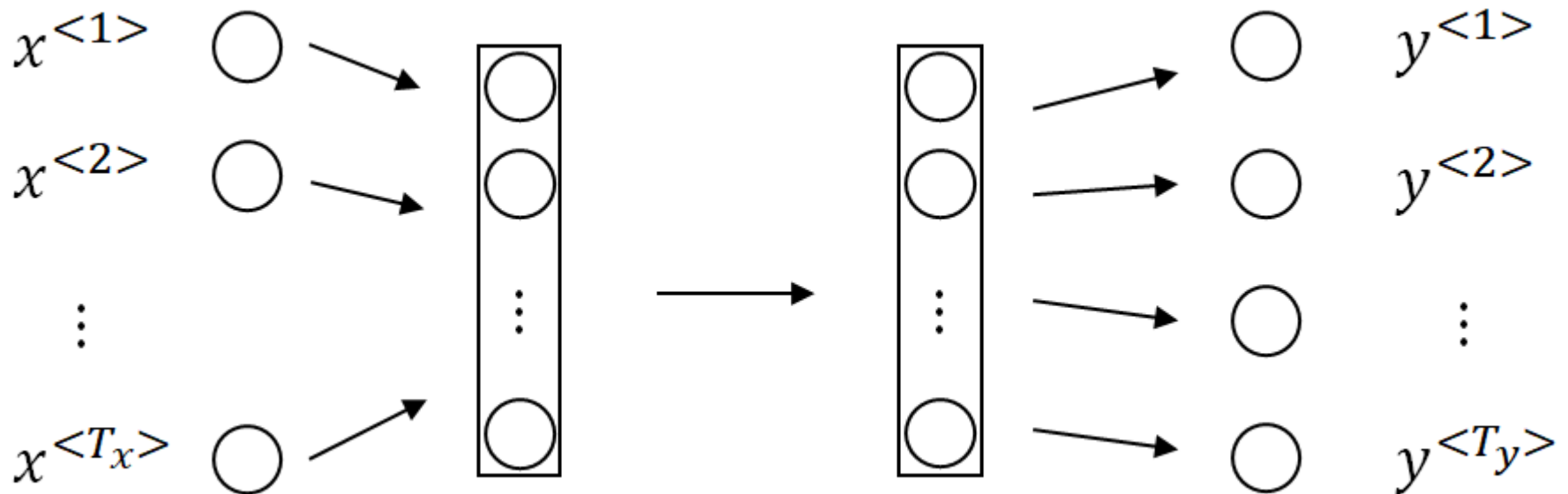
- ¿Cómo construir un modelo para hacer un mapping  $x \rightarrow y$ ?
- Podemos intentar usar una red neuronal clásica



**Problemas:**

# Ejemplo: NER

- ¿Cómo construir un modelo para hacer un mapping  $x \rightarrow y$ ?
- Podemos intentar usar una red neuronal clásica

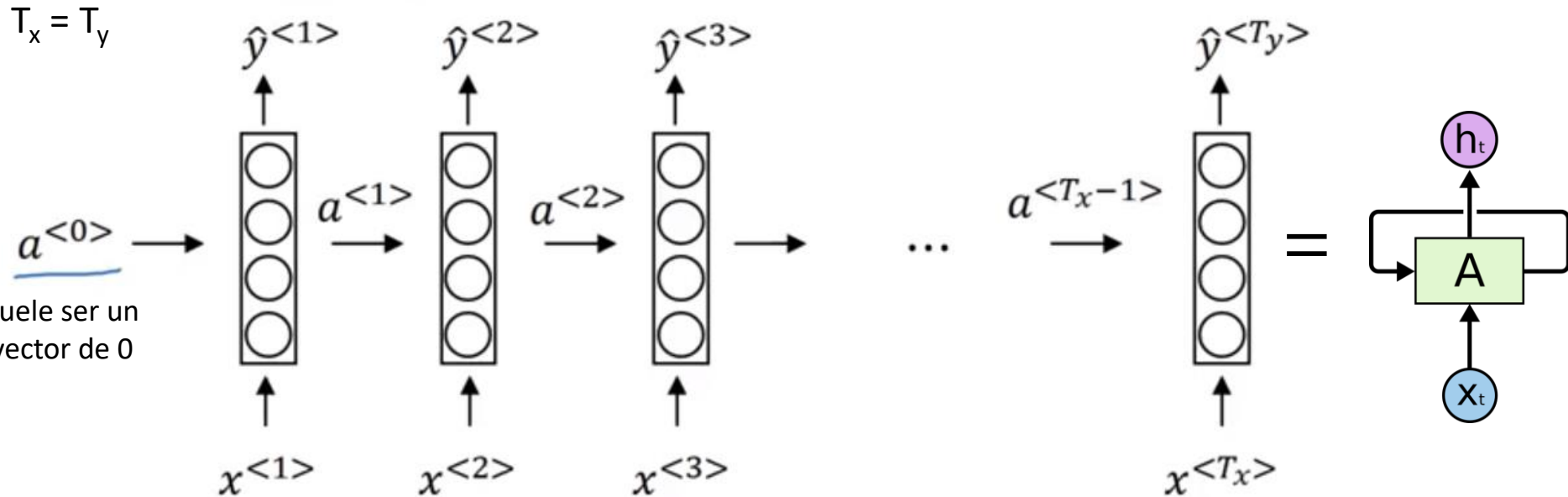


## Problemas:

- Inputs y outputs pueden tener tamaños distintos para distintas muestras.
- No comparte características aprendidas en distintas posiciones del texto.

# RNN

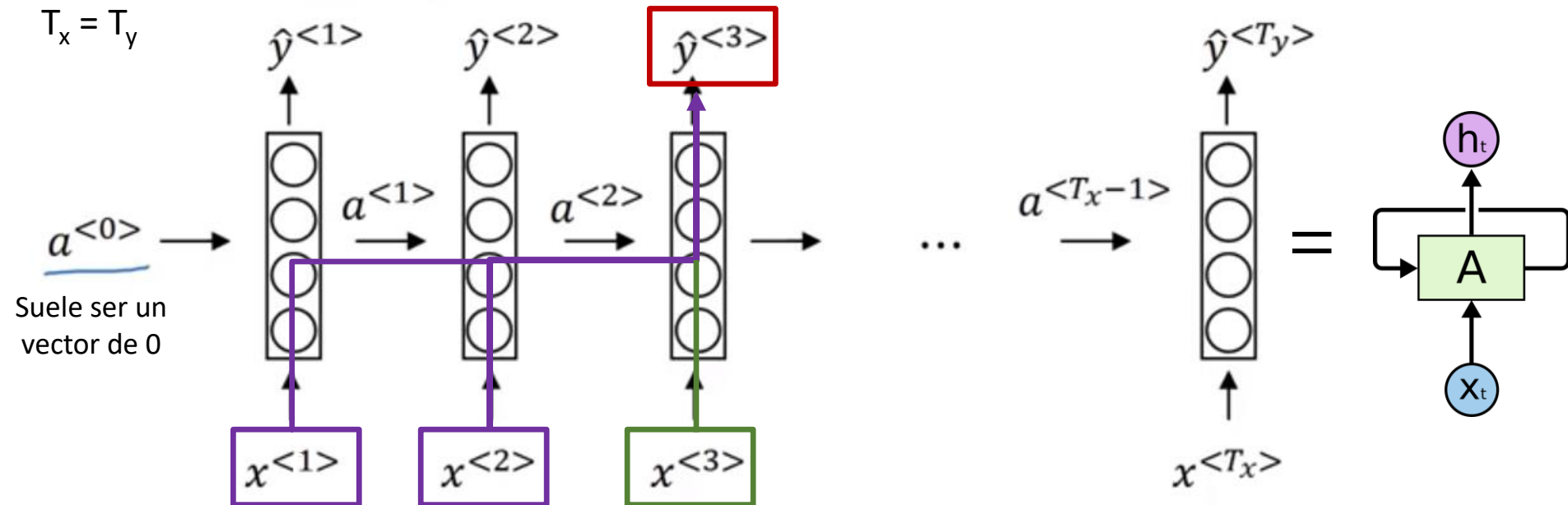
- Si leemos de izquierda a derecha:





# RNN

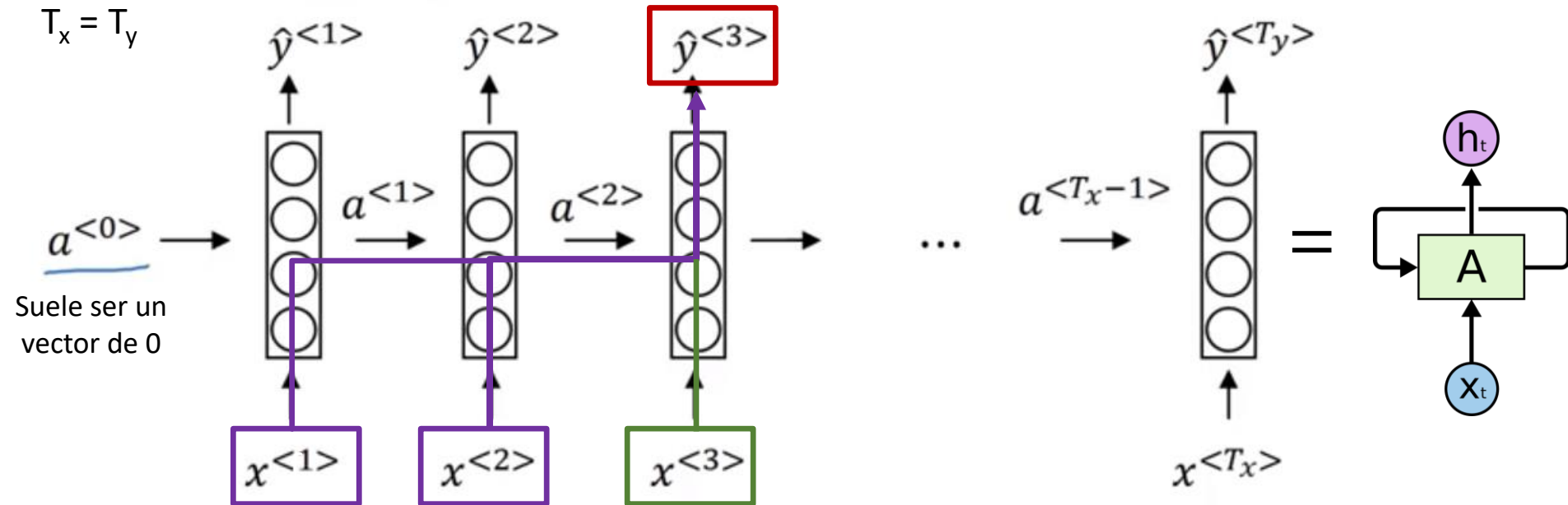
- Si leemos de izquierda a derecha:





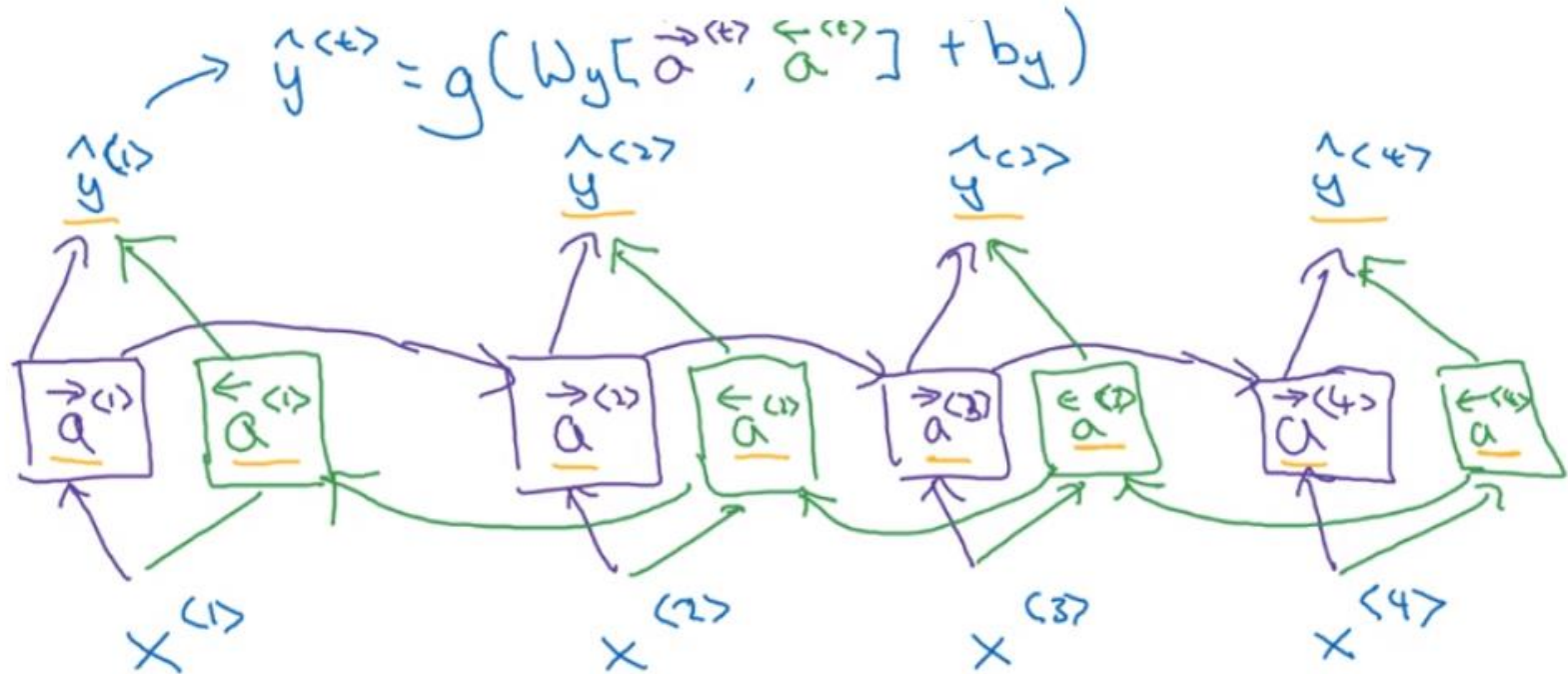
# RNN

- Si leemos de izquierda a derecha:

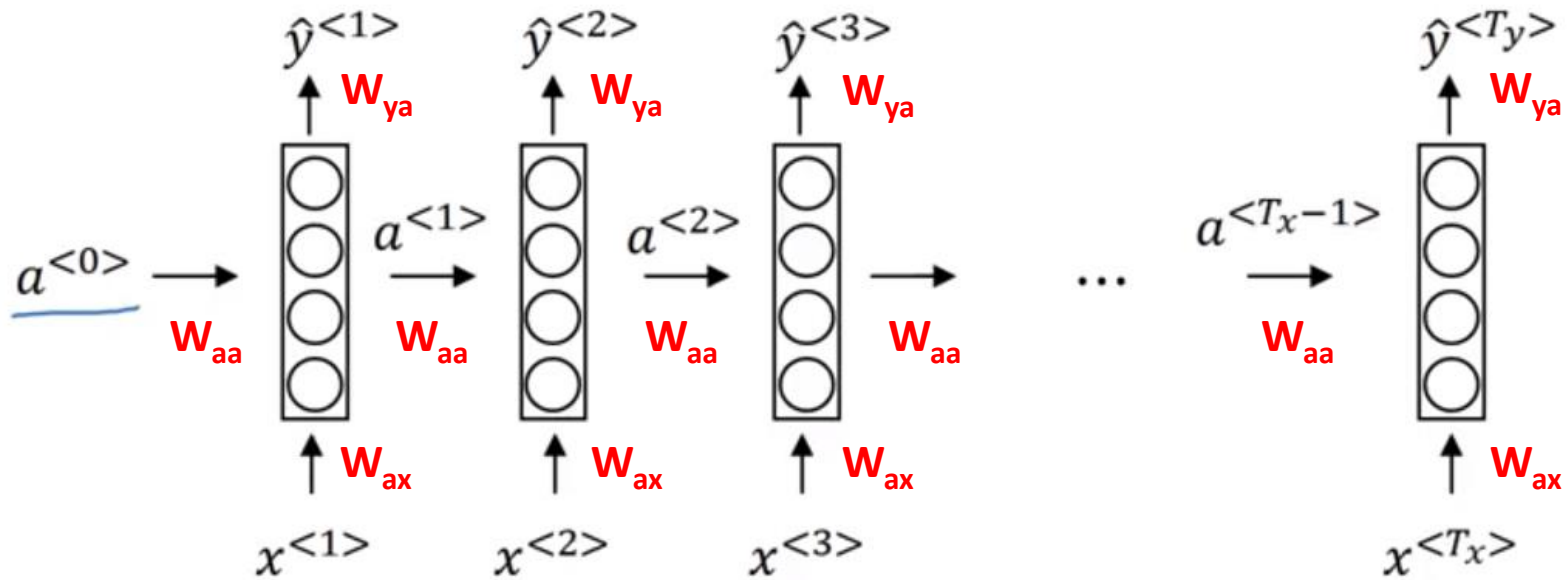


- El empleado dijo “Paloma García es clienta nuestra”
- El empleado dijo “Paloma o gorrión, no me gusta ninguno”

# RNN bidireccional



# RNN: Forward Propagation



$$a^{<0>} = \vec{0}$$

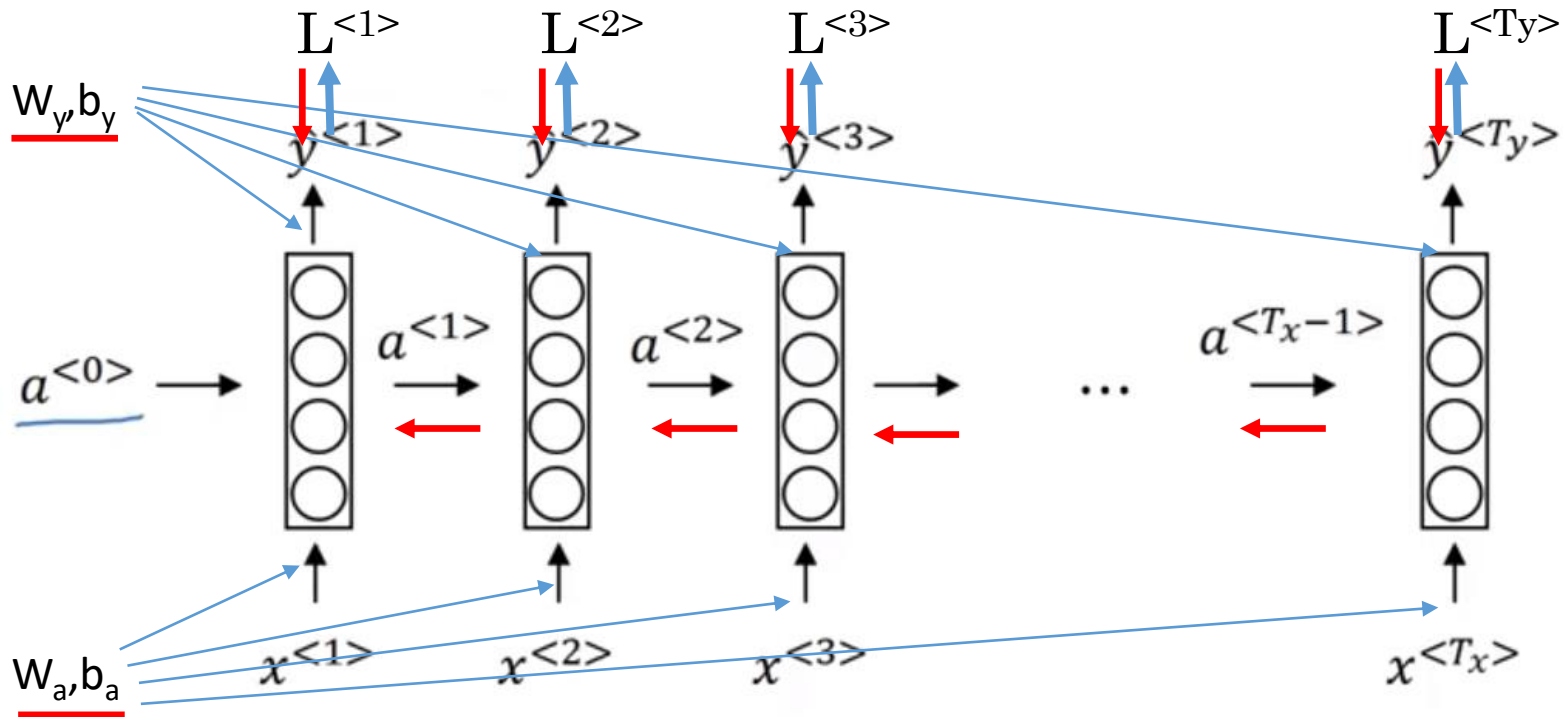
$$a^{<1>} = g_1(W_{aa} a^{<0>} + W_{ax} x^{<1>} + b_a) \leftarrow \text{ReLU}$$

$$\hat{y}^{<1>} = g_2(W_{ya} a^{<1>} + b_y) \leftarrow \text{sigmoid}$$

perché problema binario

$$\boxed{\begin{aligned} a^{<t>} &= g(W_{aa} a^{<t-1>} + W_{ax} x^{<t>} + b_a) \\ \hat{y}^{<t>} &= g(W_{ya} a^{<t>} + b_y) \end{aligned}}$$

# RNN: Backward Propagation



$$\mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{<t>} \log \hat{y}^{<t>} - (1 - y^{<t>}) \log (1 - \hat{y}^{<t>})$$

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

Backward Propagation en el tiempo

# Modelado lingüístico

- Uno de los problemas más importantes en NLP
- ¿Cómo modelizar el lenguaje con una RNN?

Comencemos por definir qué es un modelo lingüístico.  
Supongamos que estamos haciendo un sistema de reconocimiento del habla:

Elena no quiere salir con Juan.

# Modelado lingüístico

- Uno de los problemas más importantes en NLP
- ¿Cómo modelizar el lenguaje con una RNN?

Comencemos por definir qué es un modelo lingüístico.  
Supongamos que estamos haciendo un sistema de reconocimiento del habla:

Elena no quiere salir con Juan.

El enano quiere salir con Juan.

# Modelado lingüístico

- Uno de los problemas más importantes en NLP
- ¿Cómo modelizar el lenguaje con una RNN?

Comencemos por definir qué es un modelo lingüístico.  
Supongamos que estamos haciendo un sistema de reconocimiento del habla:

Elena no quiere salir con Juan.

El enano quiere salir con Juan.

Un modelo lingüístico nos diría cual de estas dos frases es más probable:

$$P(\text{Elena no quiere salir con Juan.}) = 6 \times 10^{-3}$$

$$P(\text{El enano quiere salir con Juan}) = 6 \times 10^{-5}$$



# ¿Cómo se construye un modelo lingüístico?

- **Set de entrenamiento:** Necesitamos un corpus muy grande (un set muy grande de frases en la lengua a modelar)

En el máster de Data Science aprendemos mucho<EOS>

**Lo primero es *tokenizar*:**

- Creamos un vocabulario tal y como hemos visto previamente
- Mapeamos cada palabra con un vector one-hot
- Viene bien tener en nuestro vocabulario también el signo de parada (“.”) que llamaremos <EOS>

# ¿Cómo se construye un modelo lingüístico?

- **Set de entrenamiento:** Necesitamos un corpus muy grande (un set muy grande de frases en la lengua a modelar)

En el máster de Data Science aprendemos mucho<EOS>

$y^{<1>}$   $y^{<2>}$     $y^{<3>}$     $y^{<4>}$     $y^{<5>}$     $y^{<6>}$     $y^{<7>}$     $y^{<8>}$     $y^{<9>}$

**Lo primero es *tokenizar*:**

- Creamos un vocabulario tal y como hemos visto previamente
- Mapeamos cada palabra con un vector one-hot
- Viene bien tener en nuestro vocabulario también el signo de parada (“.”) que llamaremos <EOS>
- **Si hay alguna palabra que no tenemos en nuestro vocabulario, conviene tener un token apropiado <UNK>**

# Modelo lingüístico

- Una vez que ya hemos *tokenizado* las frases de nuestro training set, vamos a construir una RNN para ver cual es la probabilidad de una frase concreta.
- $x^{<t>} = y^{<t-1>}$

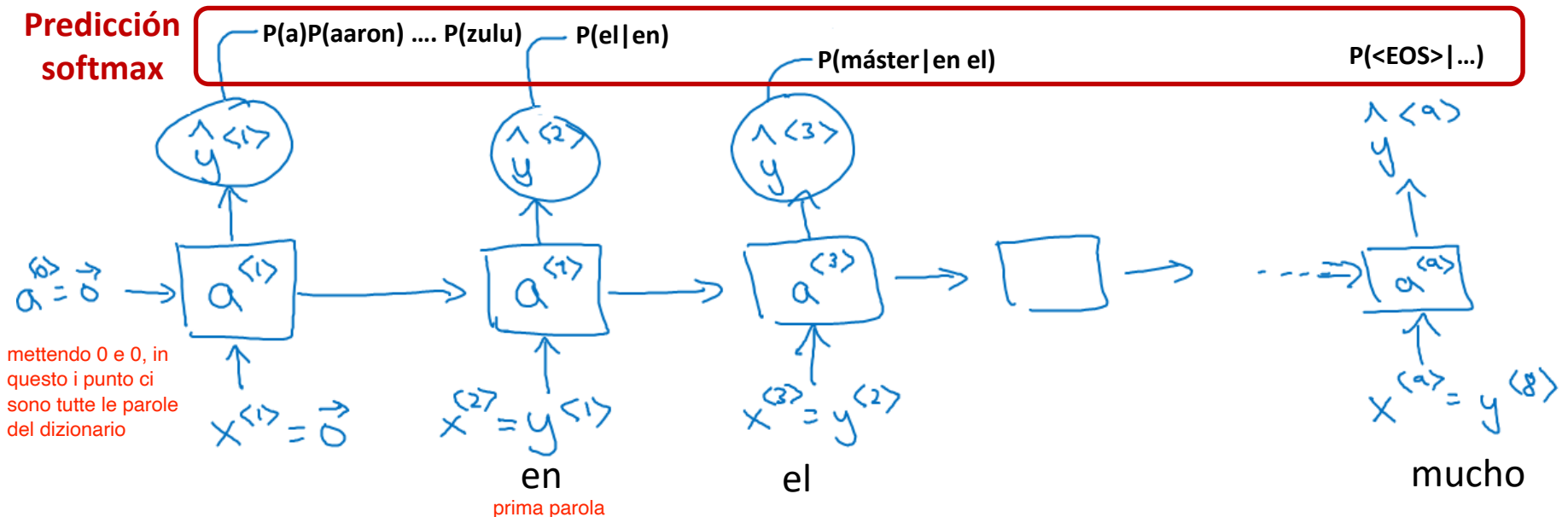
# Modelo lingüístico

- Una vez que ya hemos *tokenizado* las frases de nuestro training set, vamos a construir una RNN para ver cual es la probabilidad de una frase concreta.
- $x^{<t>} = y^{<t-1>}$

En el máster de Data Science aprendemos mucho<EOS>

# Modelo lingüístico

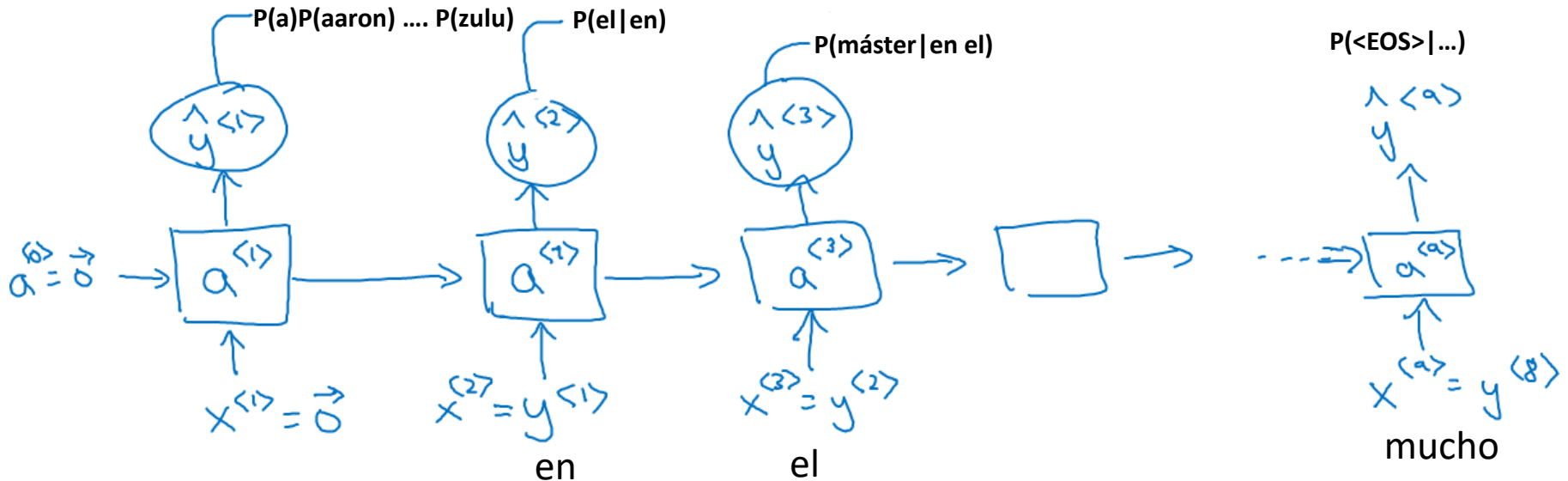
- Una vez que ya hemos *tokenizado* las frases de nuestro training set, vamos a construir una RNN para ver cual es la probabilidad de una frase concreta.
- $x^{<t>} = y^{<t-1>}$



En el máster de Data Science aprendemos mucho<EOS>

# Modelo lingüístico

En el máster de Data Science aprendemos mucho<EOS>



$$\mathcal{L}(\hat{y}^{<t>}, y^{<t>}) = - \sum_i y_i^{<t>} \log \hat{y}_i^{<t>}$$

$$\mathcal{L} = \sum_t \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

Así que dada una frase

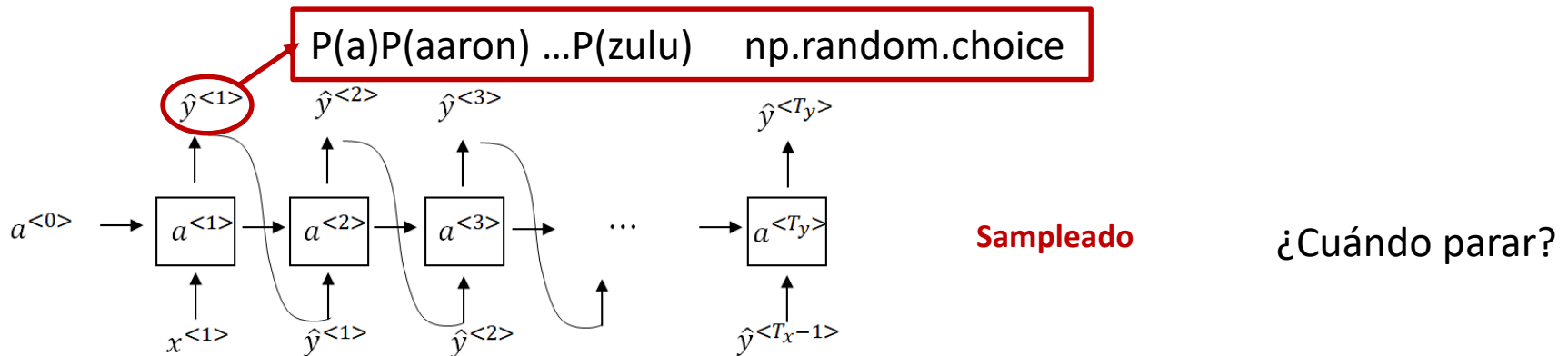
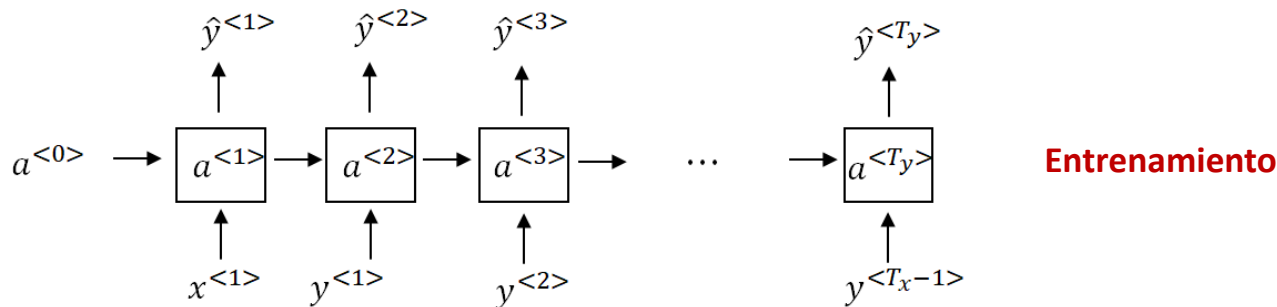
$y^{<1>} y^{<2>} y^{<3>}$

Su probabilidad será:

$P(y^{<1>})P(y^{<2>} | y^{<1>})P(y^{<3>} | y^{<1>}y^{<2>})$

# Muestreando secuencias nuevas

- Modelo lingüístico  $\rightarrow$  Modela la probabilidad de cualquier secuencia de palabras.
- Podemos *samplear* este modelo lingüístico para generar secuencias nuevas.





# Modelo lingüístico a nivel de caracteres

Mismo concepto pero el vocabulario en lugar de ser:

**Vocabulario: [a, aaron, ..., casa, ....., merienda,..., zulu]**

Va a ser:

**Vocabulario: [a,b,c,d,...., A, B, C, D, .....,0,1,2,3.....]**

# LSTM

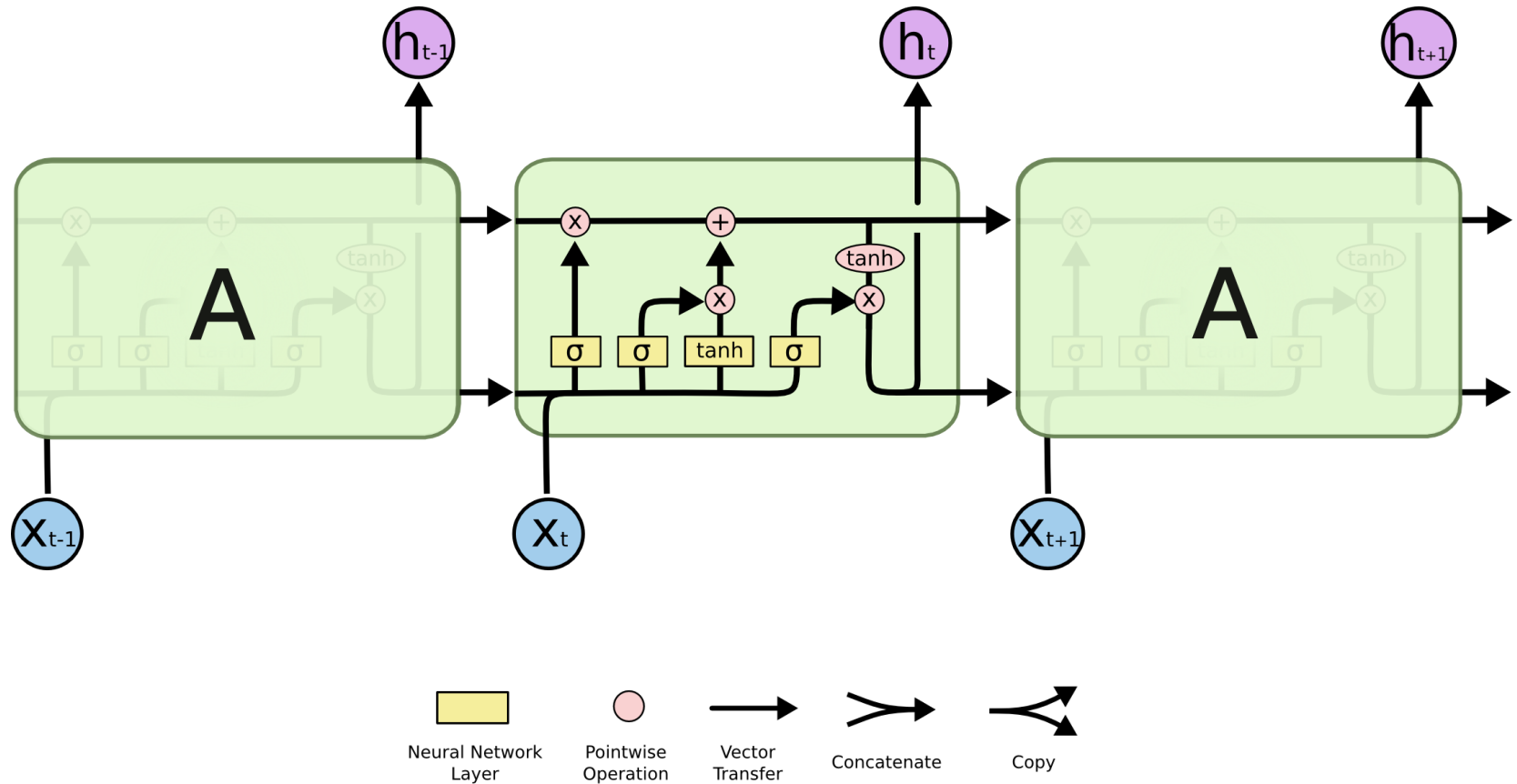
- Long Short Term Memory (LSTM)
- Te permiten conectar información muy lejana

El **niño**, que salió del colegio a las cinco y su...., **come** filetes.

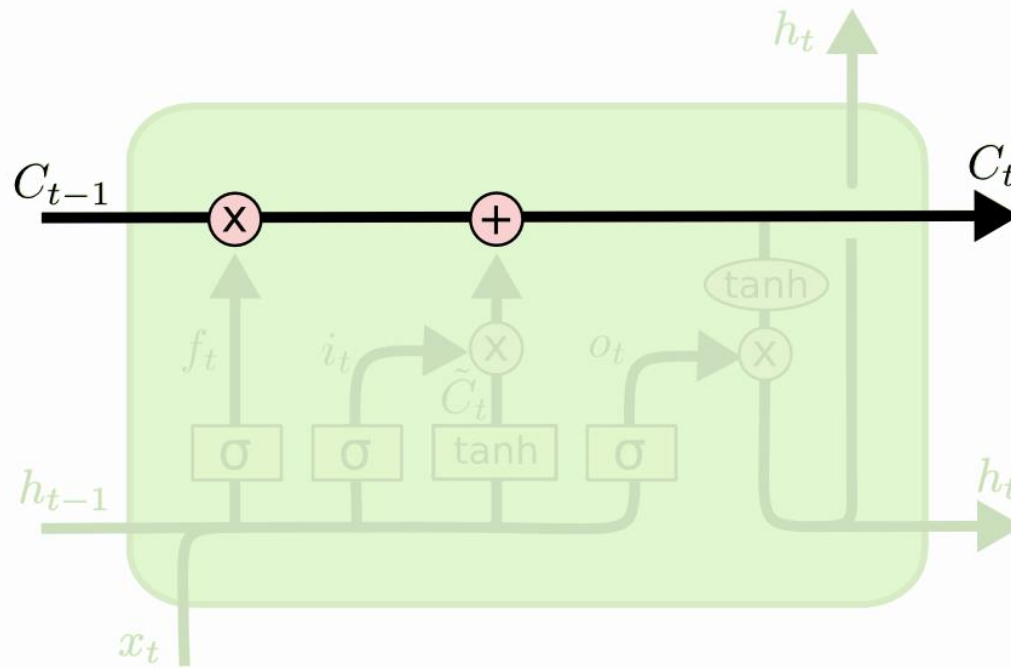
Nací en **Polonia**, crecí en el seno de...., hablo **polaco**.

<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

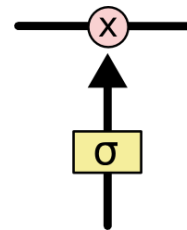
# LSTM



# LSTM

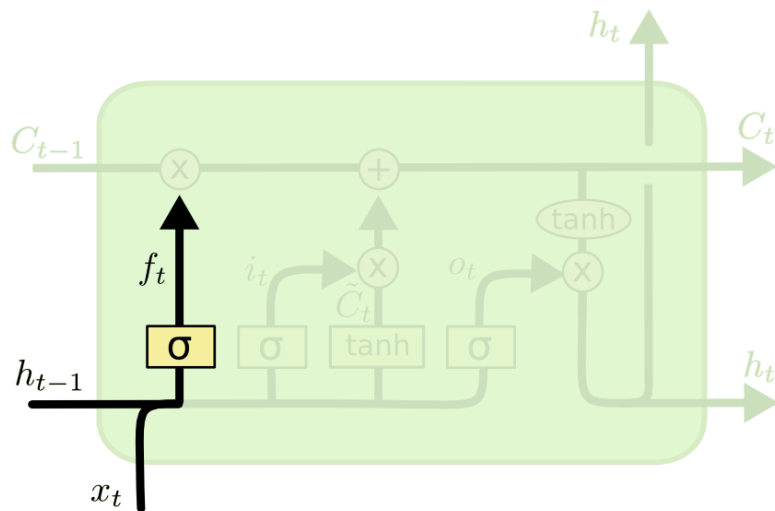


- La clave es la celda de estado (C) que atraviesa de lado a lado el nodo
- La LSTM puede quitar o poner información a esa celda de estado de manera regulada por estructuras llamadas puertas
- La sigmoide saca números entre 0 y 1:
  - 0 no deja pasar información
  - 1 deja pasar la información



# LSTM

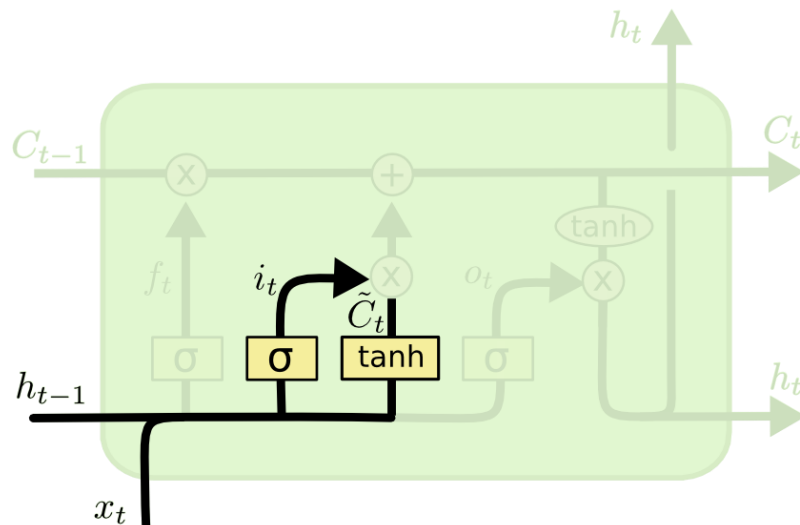
- El primer paso es decidir con qué información nos vamos a quedar y con qué información no: puerta de olvido
- Mira a los valores de  $h_{t-1}$  y  $x_t$  y le asigna un 0 o un 1 a cada valor en  $C_{t-1}$ : si sale un 0 olvidar, si sale un 1 guardar.



$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$

# LSTM

- Ahora nos toca decidir que nueva información guardar en C
- Esto tiene dos pasos:
  - Primero una sigmoide que llamamos “puerta de input” que decide qué valores hay que actualizar
  - Después un tanh que crea un nuevo conjunto de valores  $C_t$  que pueden ser añadidos a C
- Luego los combinamos para actualizar el estado C

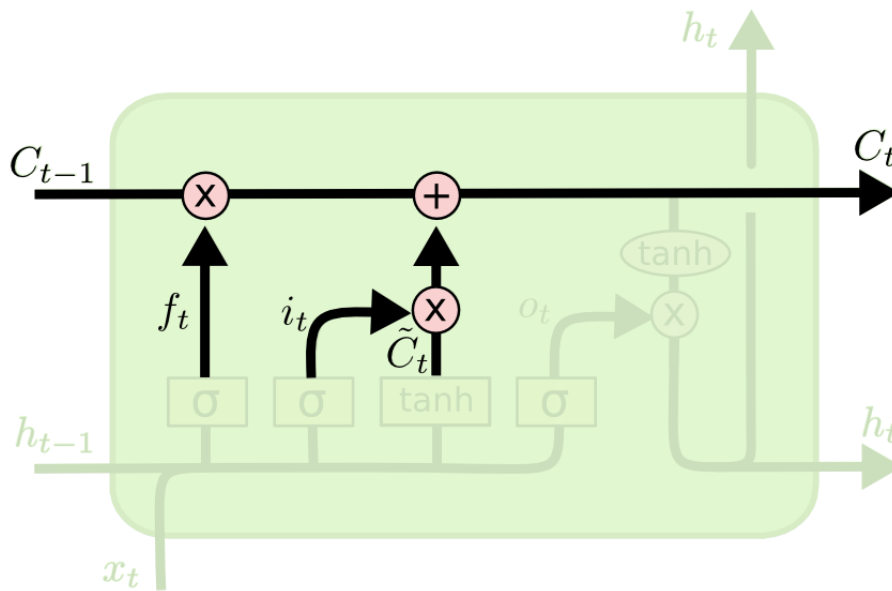


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

# LSTM

- Aquí podemos ver como se actualiza C

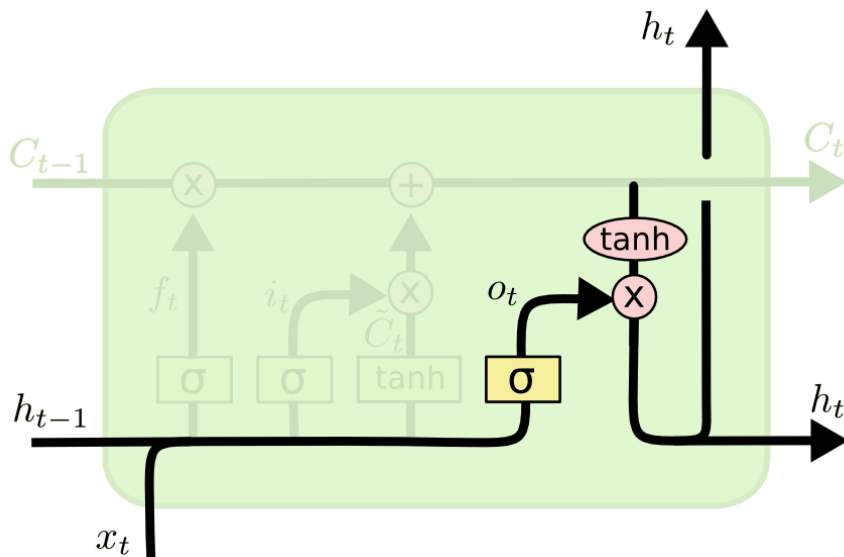


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



# LSTM

- Finalmente decidimos el output: estará basado en (una versión filtrada de)  $C$  y una capa con una *sigmoide* que nos indicará que valores de  $C$  vamos a sacar por el output



$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh (C_t)$$

# Resumiendo

- Para que se pueden usar las redes neuronales recurrentes (RNN)
- Hemos visto a grandes rasgos como es el forward propagation y el backward propagation de una RNN
- Las RNN tienen problemas para recordar dependencias a largo plazo
- La solución para este problema son las LSTM
- Como funciona un LSTM

# Práctica

---

[https://github.com/laramaktub/LSTM\\_Master](https://github.com/laramaktub/LSTM_Master)