Redes Neuronales para Lenguaje Natural

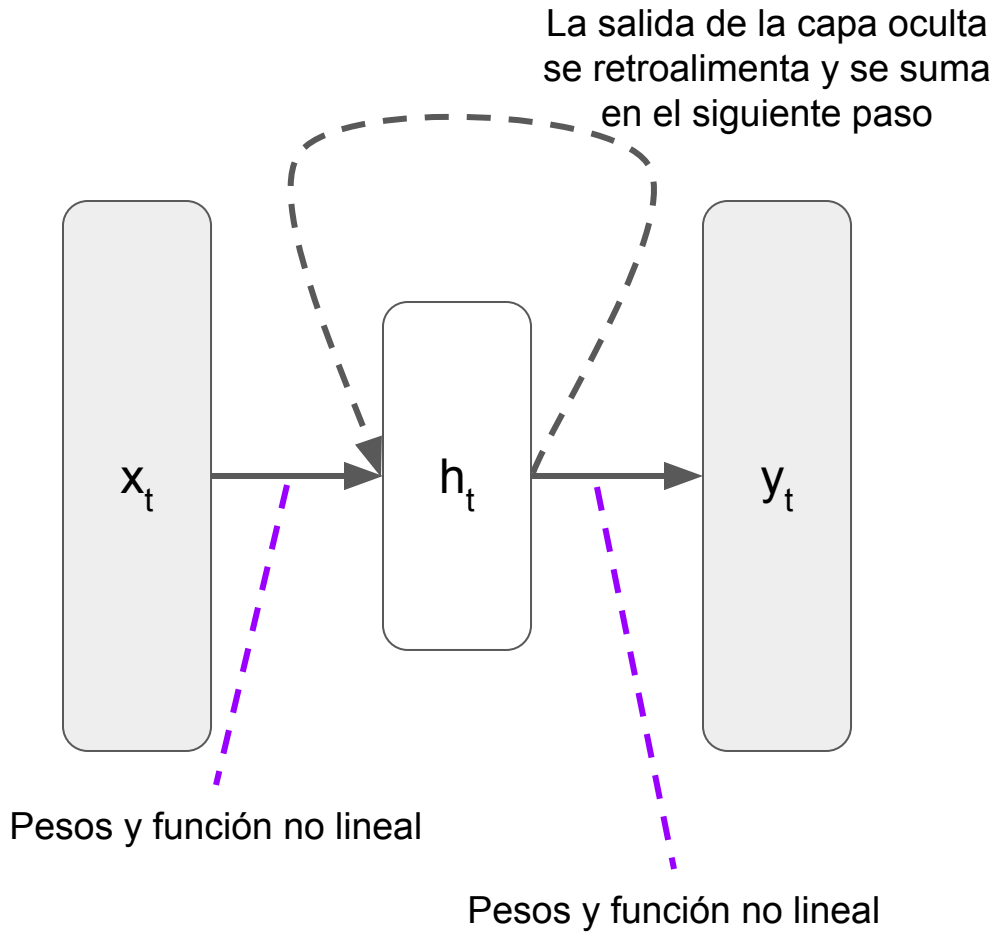
2024

Grupo de Procesamiento de Lenguaje Natural
Instituto de Computación

Bibliografía

- Jurafsky & Martin, 3rd Ed. (draft) - Capítulo 8
- Understanding LSTMS - Christopher Olah
  https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Papers…

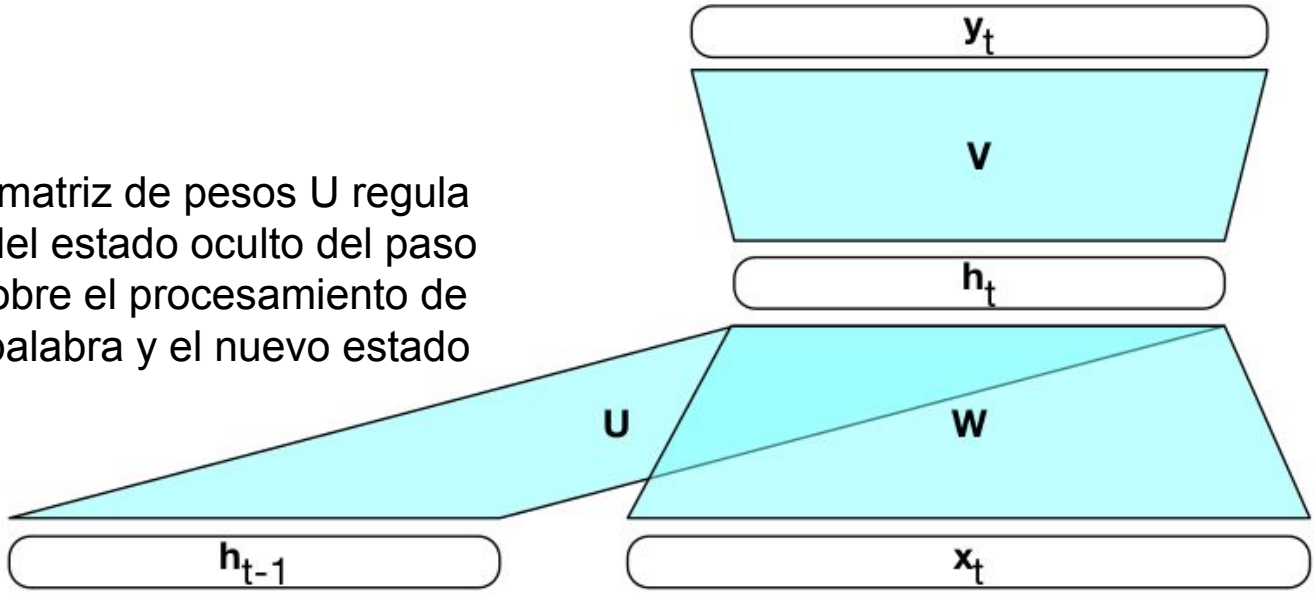
Red Recurrente Simple (Red de Elman)



Finding Structure in Time
Elman (1990)

Red Recurrente Simple

La nueva matriz de pesos U regula el efecto del estado oculto del paso anterior sobre el procesamiento de la nueva palabra y el nuevo estado oculto



$$h_t = g(Uh_{t-1} + Wx_t)$$

$$y_t = f(Vh_t)$$

Por ejemplo:

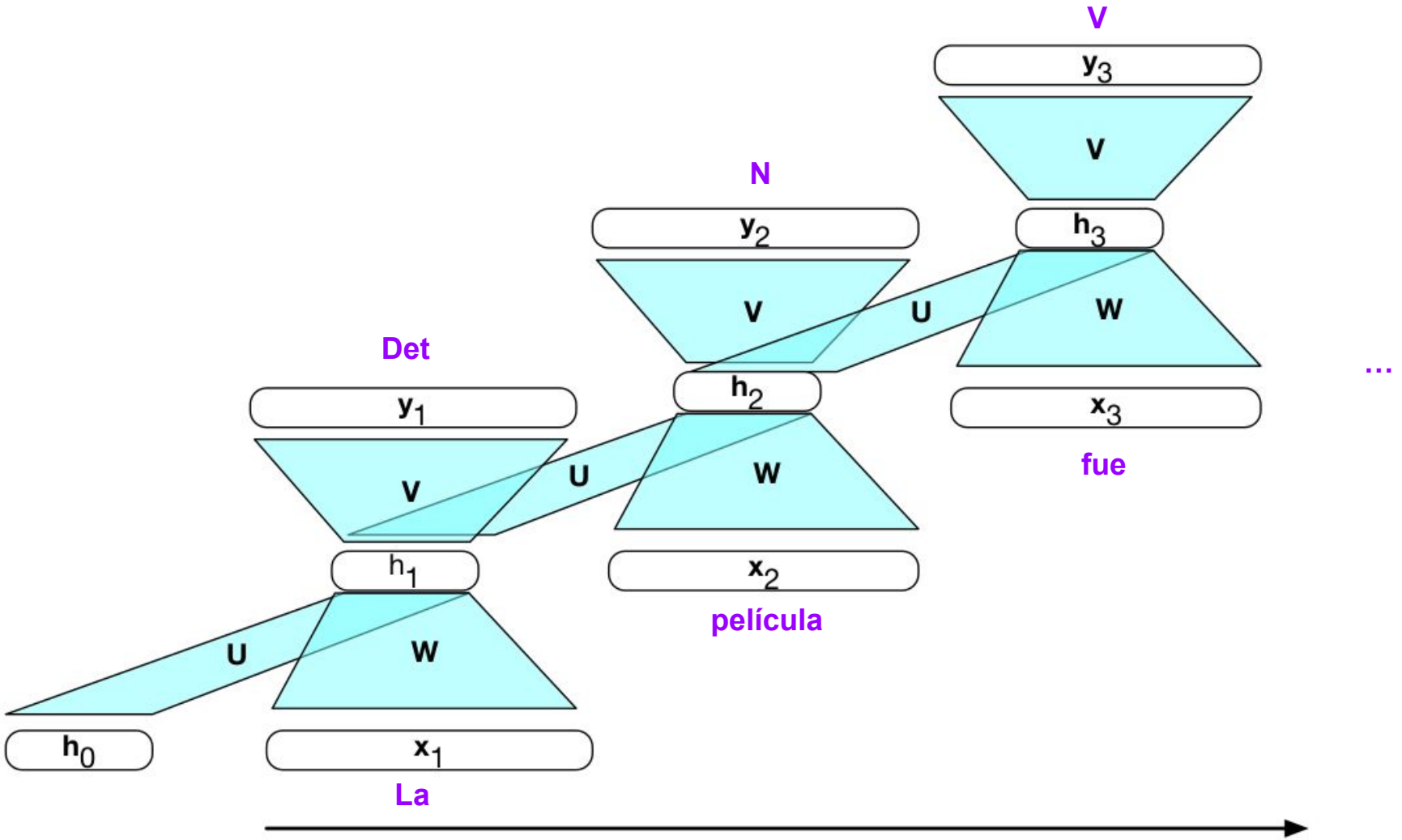
$$y_t = \text{softmax}(Vh_t)$$

$U \rightarrow dim_h \times dim_h$

$W \rightarrow dim_h \times dim_{in}$

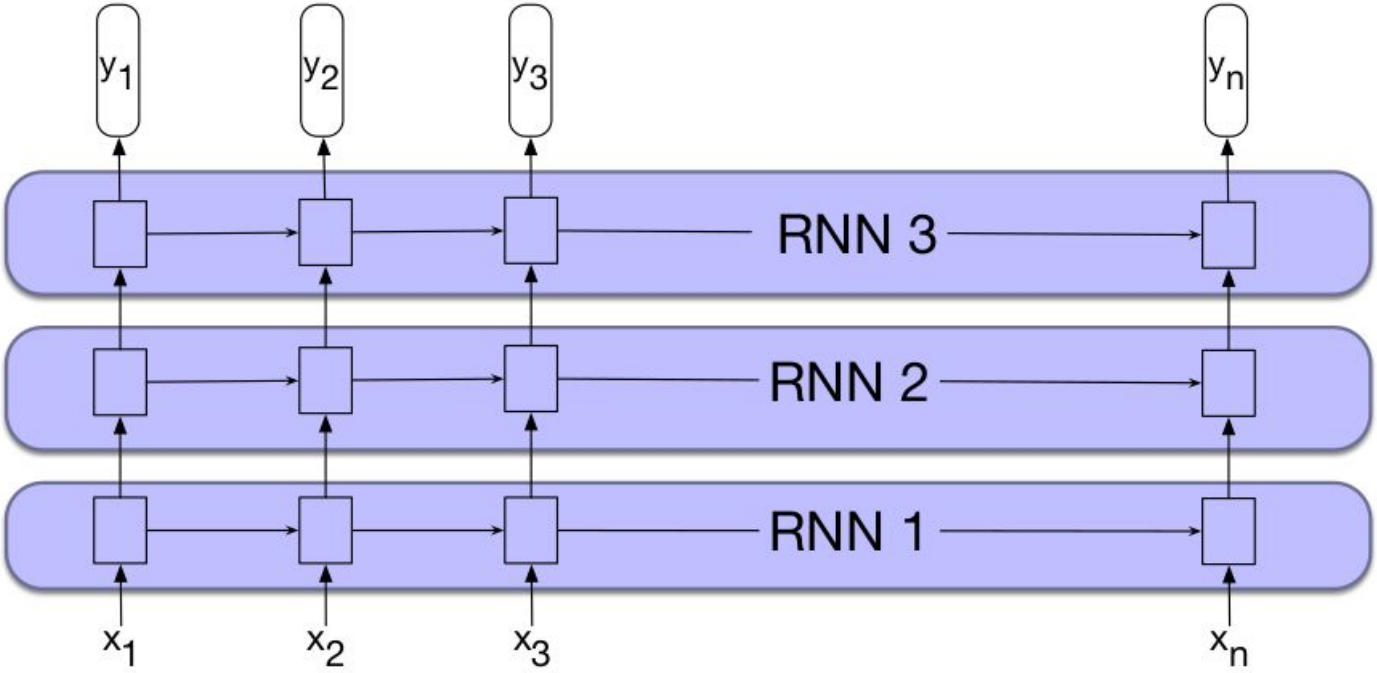
$V \rightarrow dim_{out} \times dim_h$

Desarrollo en el Tiempo



Variantes de la Arquitectura RNN

Stack de RNN

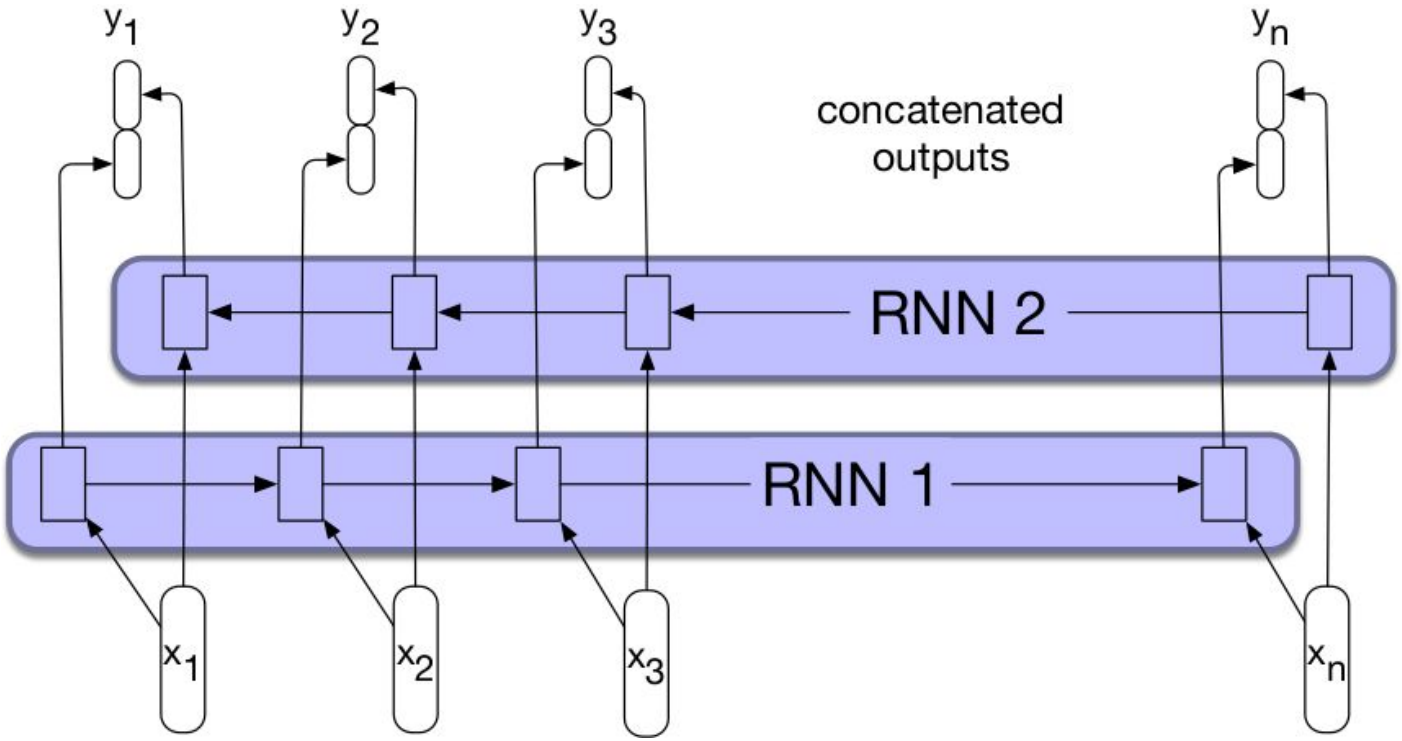


Podemos enganchar las salidas de la capa oculta y usarlas como la entrada de otra RNN

Distintas capas inducen representaciones con diferente nivel de abstracción (palabras, frases, sintagmas…)

Ojo: Más parámetros, más difícil de entrenar

RNN bidireccional



Dos RNNs: una toma la secuencia normal y la otra la secuencia en reversa

Las salidas de las capas ocultas de ambas redes en el tiempo *t* se concatenan para formar la representación de la palabra x_t

RNN bidireccional

- La red “hacia adelante”

$$h_t^f = \text{RNN}_{\text{forward}}(x_1, ..., x_t)$$

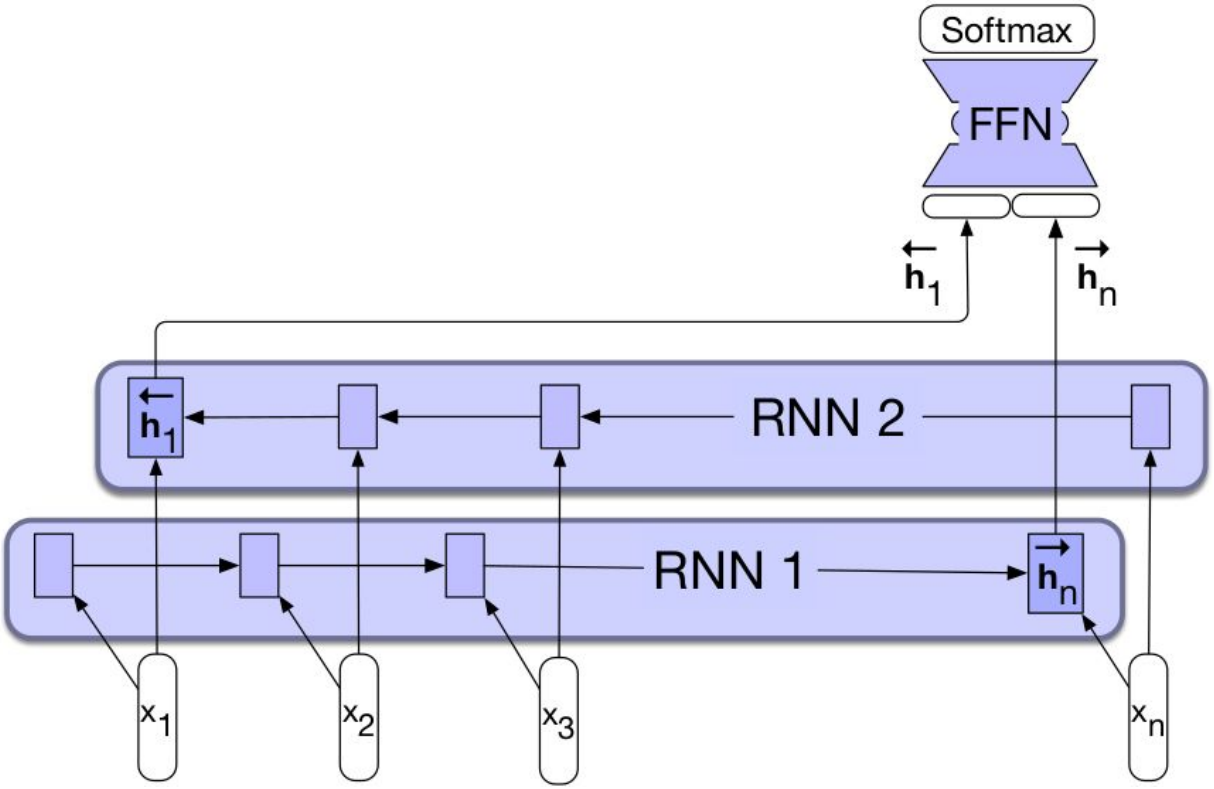
- La red “en reversa”

$$h_t^b = \text{RNN}_{\text{backward}}(x_t, ..., x_n)$$

- La nueva salida de la capa oculta se forma concatenando las salidas de las dos para el tiempo *t*

$$\begin{aligned} h_t &= [h_t^f; h_t^b] \\ &= h_t^f \oplus h_t^b \end{aligned}$$

RNN bidireccional



Para clasificación de una secuencia entera, es usual concatenar las salidas de la capa oculta que consideran todas las palabras: h_n de la red directa y h_1 de la red reversa

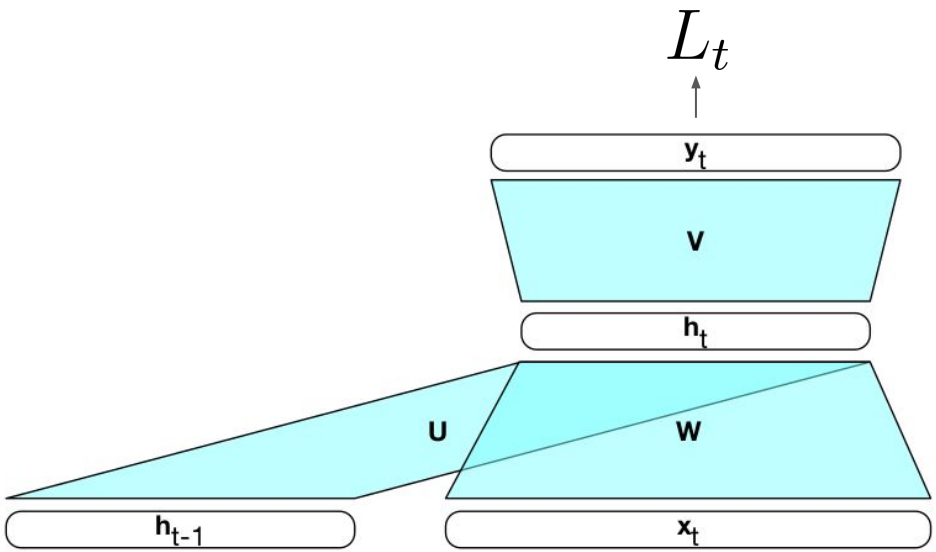

LSTM

Modelos de Lenguaje

Algunas rutas se mantienen cerradas desde el fin de semana debido a las copiosas *lluvias*

Viví en Francia toda mi infancia y hasta los dieciocho años, por lo tanto hablo un perfecto *francés*

Desvanecimiento del gradiente en RNN



$$L = \sum_{t=1}^{T} L_t$$

$$\frac{\partial L_t}{\partial \mathbf{U}} = \frac{\partial L_t}{\partial y_t} \cdot \frac{\partial y_t}{\partial \mathbf{h}_t} \cdot \left(\sum_{k=1}^{t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \cdot \frac{\partial \mathbf{h}_k}{\partial \mathbf{U}} \right)$$

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}}$$

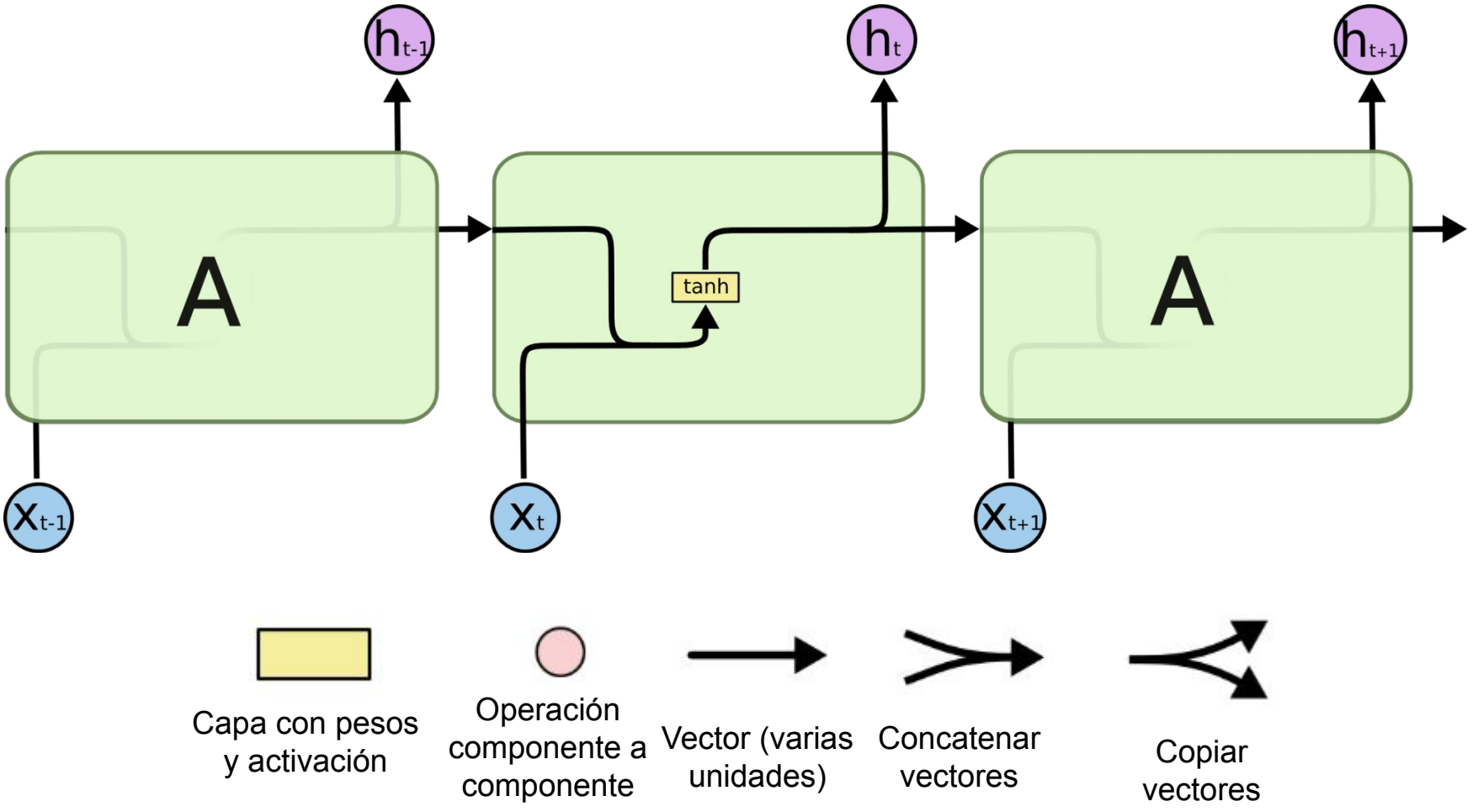
Problema del desvanecimiento/ explosión del gradiente!

Backpropagation through time: what it does and how to do it
P. J. Werbos (1990)

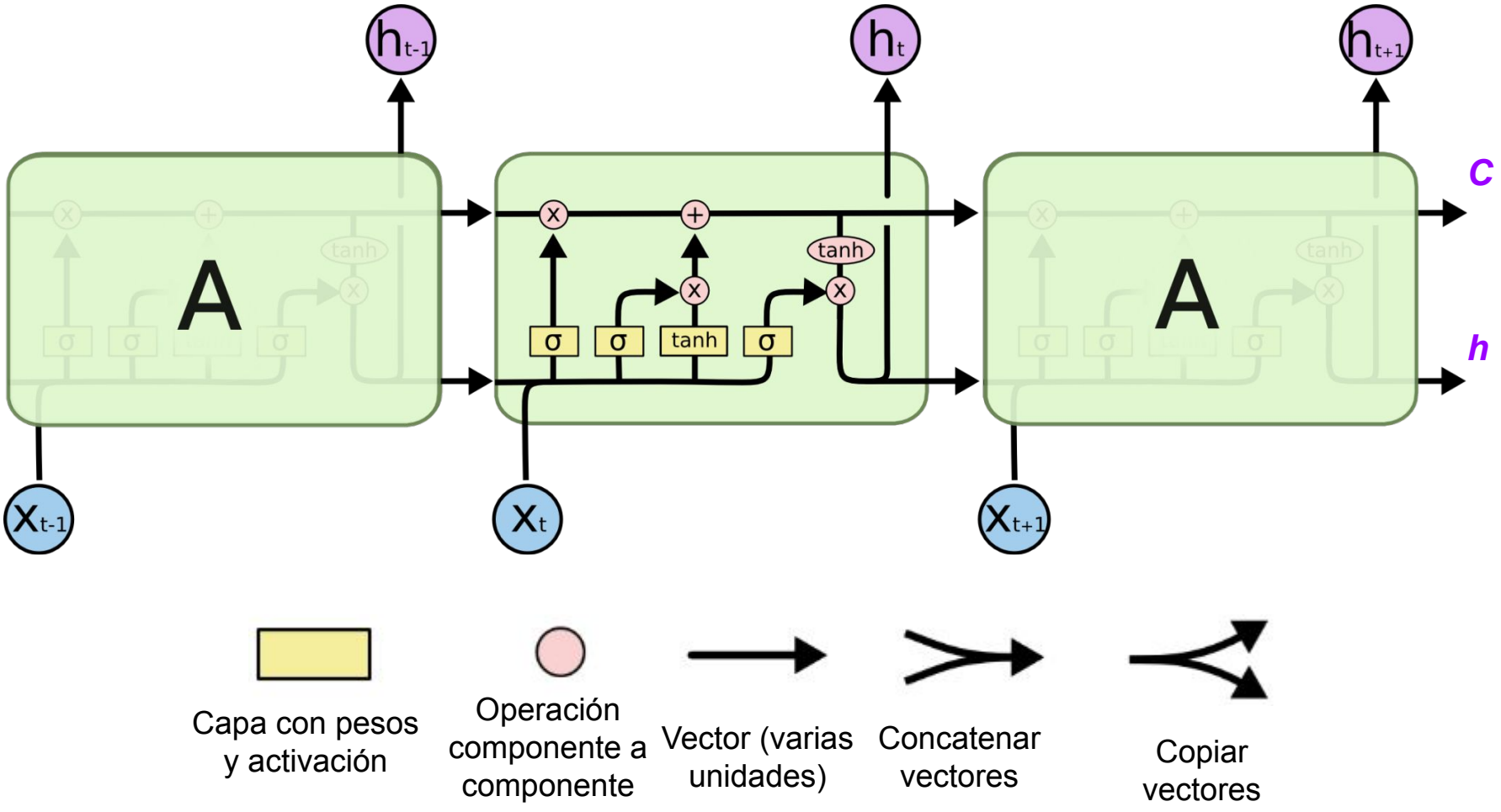
LSTM

- *Long Short-Term Memory* (Hochreiter y Schmidhuber, 1997)
- Intentan solucionar el problema de que la información más lejana es más difícil de acceder (recordar)
- Utiliza dos “estados ocultos”
  - uno de ellos mantiene la información “vieja” — celda (*C*)
  - el otro es el que se usa para calcular las salidas — capa oculta (*h*)
- En cada paso, los dos estados y la nueva palabra de entrada (*x*) se recombinan

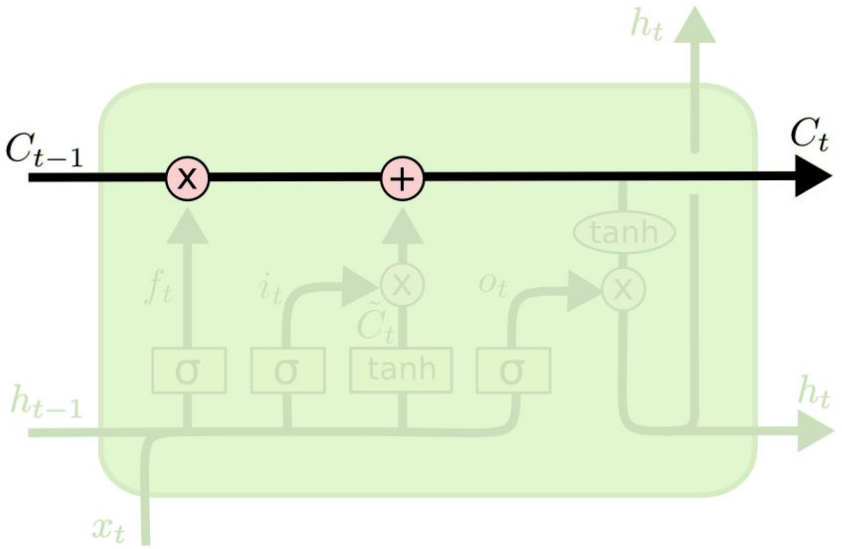
RNN Simple


h_{t-1}
h_t
h_{t+1}
A
tanh
A
X_{t-1}
X_t
X_{t+1}
Capa con pesos y activación
Operación componente a componente
Vector (varias unidades)
Concatenar vectores
Copiar vectores

LSTM


h_{t-1}
h_t
h_{t+1}
A
A
tanh
σ
σ
tanh
σ
tanh
C
h
X_{t-1}
X_t
X_{t+1}
Capa con pesos y activación
Operación componente a componente
Vector (varias unidades)
Concatenar vectores
Copiar vectores

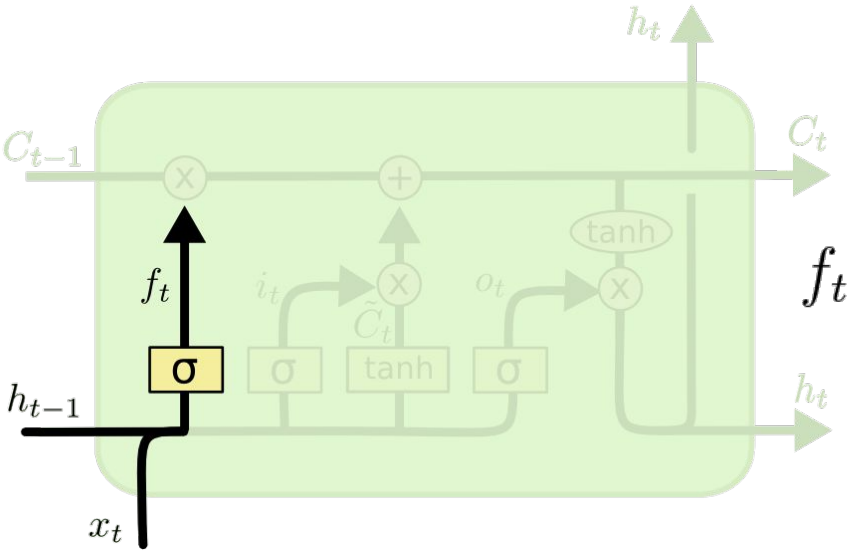
LSTM



- *C*: celda o *“cell state”*, tamaño d_h
- Se conecta directamente de una unidad a otra
- En cada paso puede “olvidar” algunos datos y “aprender” algunos datos nuevos

LSTM

$W_f \rightarrow d_h \times (d_h + d_w)$



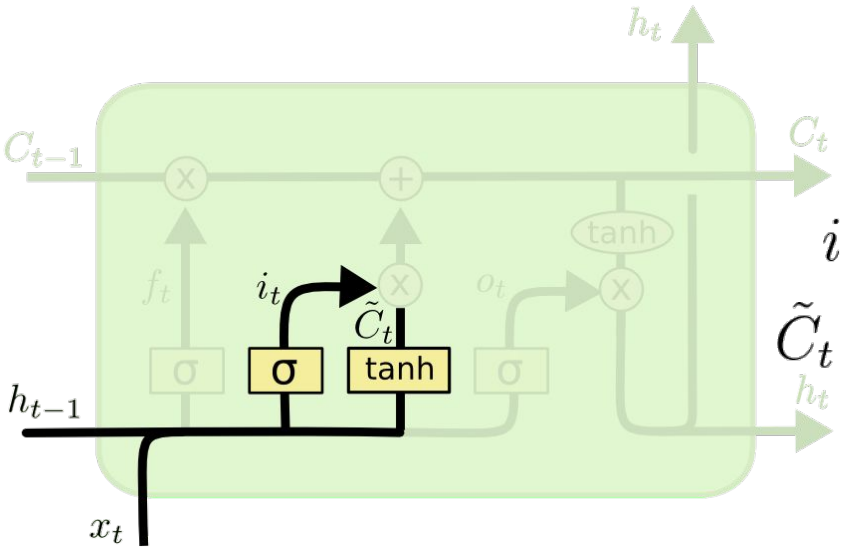
$$f_t = \sigma(W_f . [h_{t-1}; x_t])$$

- *h*: estado oculto o *"hidden state"*, tamaño d_h
- Se concatenan h_{t-1} y la nueva entrada x_t (de tamaño d_w)
- f_t: *"forget gate"*, indica qué información será "olvidada" por la celda
- Se calcula con una matriz de pesos W_f y activación sigmoide

LSTM

$W_i \rightarrow d_h \times (d_h + d_w)$

$W_c \rightarrow d_h \times (d_h + d_w)$

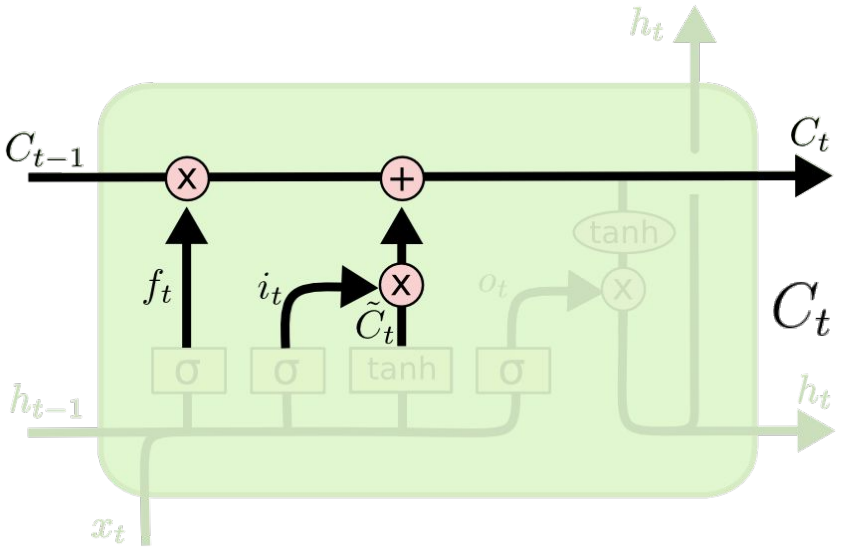


$$i_t = \sigma(W_i.[h_{t-1}; x_t])$$

$$\tilde{C}_t = \tanh(W_c[h_{t-1}; x_t])$$

- i_t: *"input gate"*, indica qué partes de la celda serán modificadas
- $\tilde{C}$ es el vector de valores que se incluirán en esa nueva celda modificada (nuevos valores a "recordar")

LSTM

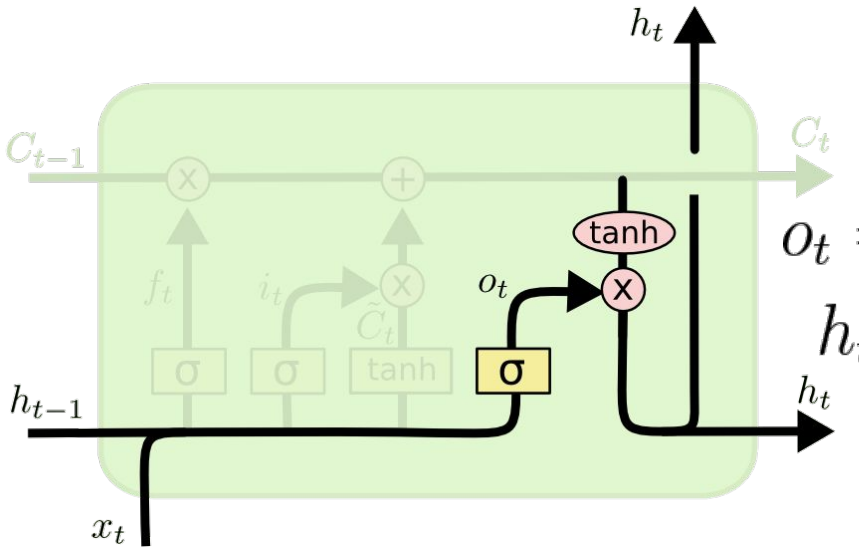


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- El nuevo estado de celda del paso siguiente C_t se calcula a partir de la celda en el paso anterior C_{t-1}
  - “olvidando” los valores indicador por f_t
  - “aprendiendo” valores nuevos indicados por i_t

LSTM

$W_o \rightarrow d_h \times (d_h + d_w)$



$$o_t = \sigma(W_o.[h_{t-1}; x_t])$$

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

- El nuevo *hidden state* h_t es una combinación de:
  - el nuevo *cell state*
  - el *hidden state* viejo
  - la nueva palabra de entrada

LSTM - Resumen

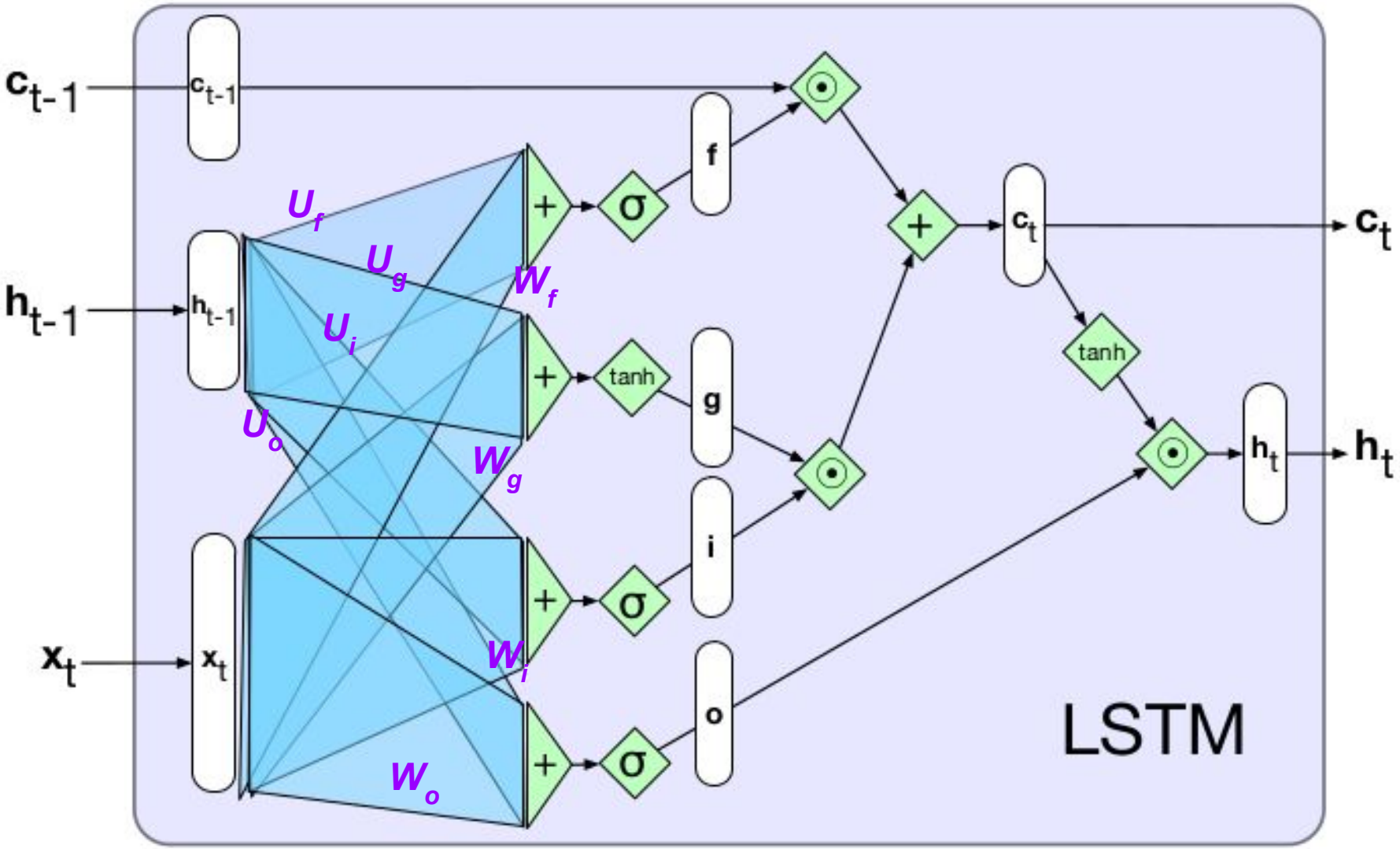
- Cuatro matrices de pesos, todas de tamaño $d_h \; x \; (d_h + d_w)$:
  - W_f: controla qué elementos se van a “olvidar”
  - W_i: controla qué elementos de la celda de modificarán
  - W_c: controla qué valores nuevos “recordará” la celda
  - W_o: controla cómo combinar la celda y los valores de entrada para obtener la salida
- La salida de la capa oculta h_t, es la que se utilizará para seguir el procesamiento, por ejemplo con un *softmax*

LSTM desde el punto de vista de Jurafsky

Jurafsky & Martin, 3rd Ed. ve la LSTM de forma análoga, pero:

- Cada entrada y estado se multiplica por matrices de pesos independientes
- Luego se suman los resultados de esas capas
  - $U_f W_f$: controla qué elementos se van a “olvidar”
  - $U_i W_i$: controla qué elementos de la celda de modificarán
  - $U_g W_g$: controla qué valores nuevos “recordará” la celda
  - $U_o W_o$: controla cómo combinar la celda y los valores de entrada para obtener la salida
  - Las matrices *W* operan sobre la entrada, las *U* sobre el estado oculto

LSTM desde el punto de vista de Jurafsky



LSTM desde el punto de vista de Jurafsky

Todas las matrices *U* son d_h x d_h

Todas las matrices *W* son d_h x d_w

$$f_t = \sigma(U_f h_{t-1} + W_f x_t)$$
$$k_t = c_{t-1} * f_t$$

forget gate

$$g_t = \tanh(U_g h_{t-1} + W_g x_t)$$
$$i_t = \sigma(U_i h_{t-1} + W_i x_t)$$
$$j_t = g_t * i_t$$

add gate

$$c_t = j_t + k_t$$

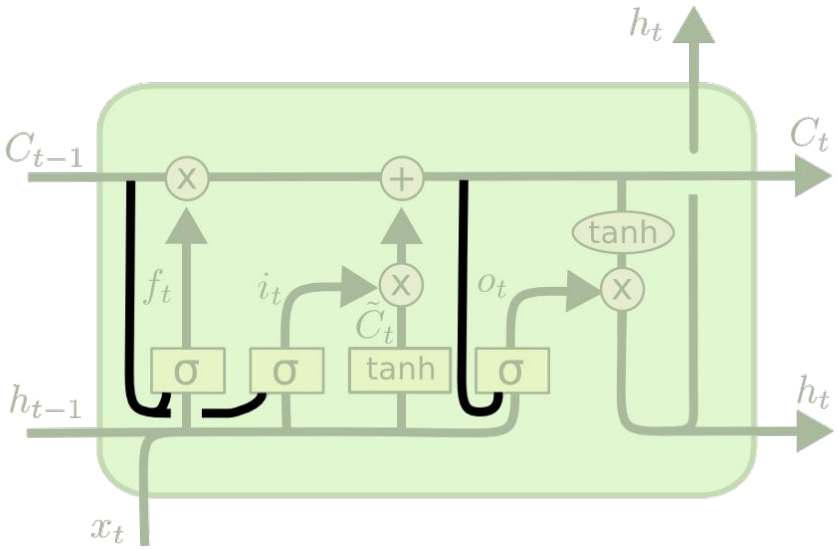
siguiente valor de celda

$$o_t = \sigma(U_o h_{t-1} + W_o x_t)$$
$$h_t = o_t * \tanh(c_t)$$

output gate


GRU y otras variantes

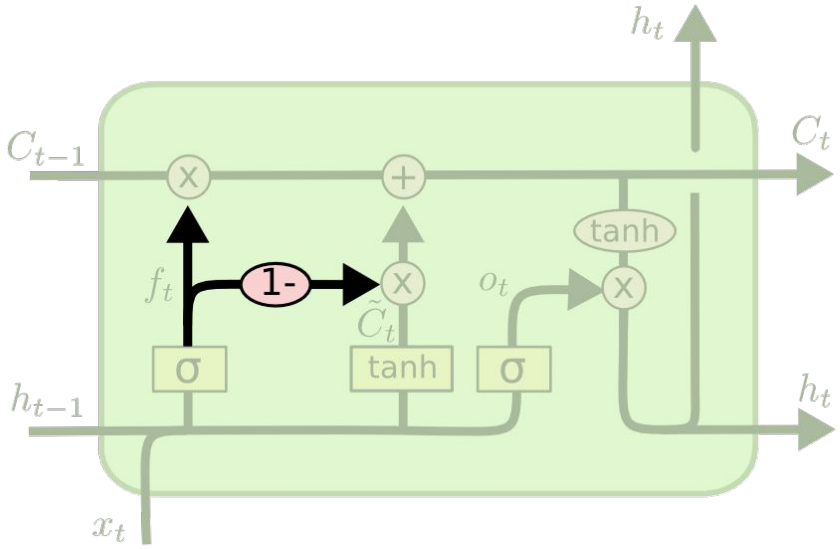
LSTM con peepholes



$$f_t = \sigma(W_f.[C_{t-1}; h_{t-1}; x_t])$$
$$i_t = \sigma(W_i.[C_{t-1}; h_{t-1}; x_t])$$
$$o_t = \sigma(W_o.[C_t; h_{t-1}; x_t])$$

- Concatena el estado de celda en cada uno de los pasos (forget gate, input gate, output gate)
- Quedan matrices más grandes para aprender, más parámetros

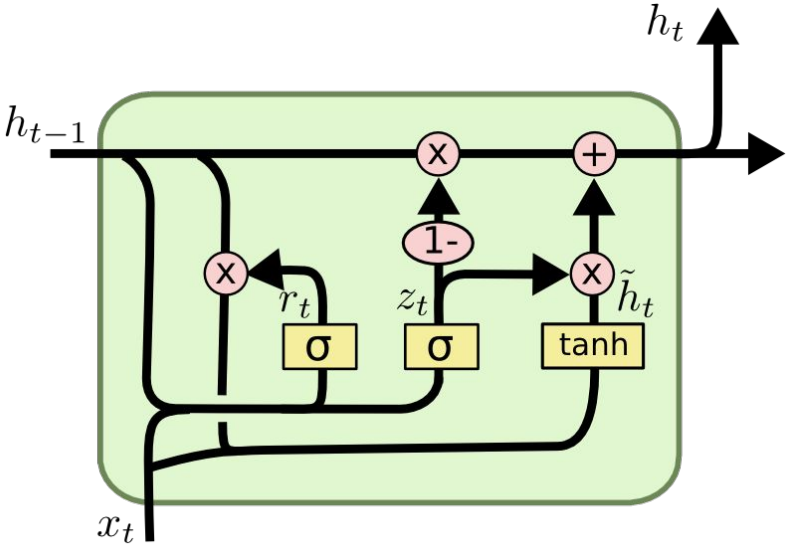
Combinar *i* y *f*



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

- Vimos que el *forget gate* elige qué olvidar, y el *input gate* elige qué modificar
- Se pueden enganchar ambos para que uno sea el complemento del otro
- Quedan menos parámetros para aprender

GRU



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

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

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

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

- *Gated Recurrent Unit* (Cho et al., 2014)
- Combina *C* y *h*, teniendo un solo canal de información que se actualiza con la información nueva en cada paso
- Depende del problema, pero en la práctica se suelen usar más las LSTM originales


ELMO

ELMO

- Embeddings from Language Models
- Descrito en *Deep contextualized word representations* (Peters et al., 2018)
- Una forma de construir embeddings
  - Contextuales: embedding de la palabra en contexto
  - No solo para palabras: embeddings desde caracteres hasta oraciones

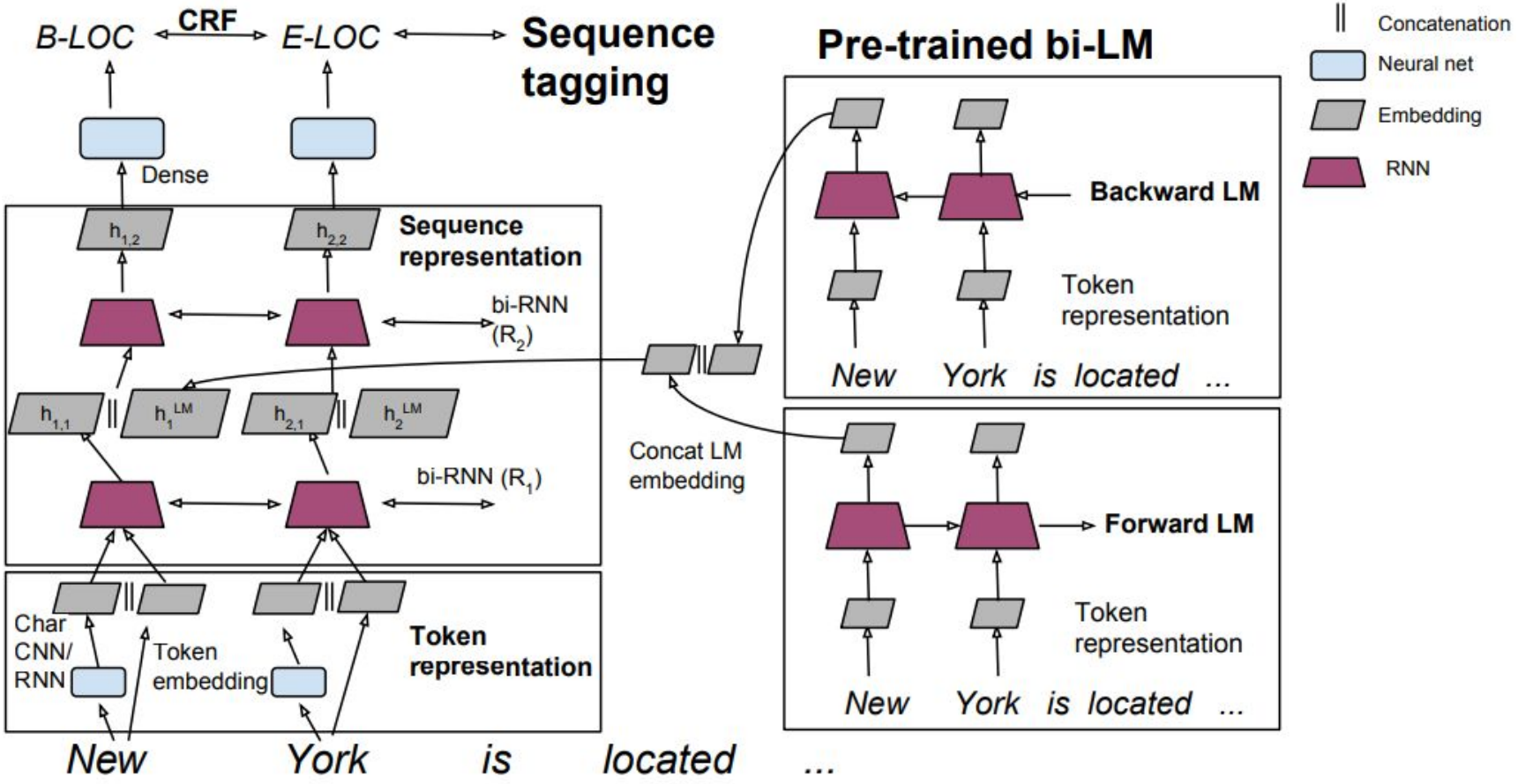
Embeddings contextuales

- Las palabras no tienen significado único
  - El *gato* come pescado
  - Para cambiar la rueda, levanté el auto con el *gato*
- Polisemia: ambigüedad en el sentido de las palabras
- Para desambiguar necesitamos contexto

ELMO no es la única técnica para hacer embeddings contextuales

También está BERT: Bidirectional Encoder Representations from Transformers

ELMO (antecedente)



Semi-supervised sequence tagging with bidirectional language models
Peters et al., (2017)

ELMO

- Usa convolución de caracteres para representar las palabras
- Dos capas de LSTM bidireccionales combinan las palabras para obtener los embeddings
- Hay una “conexión residual” entre la entrada y la segunda capa
- Se entrena minimizando el loss de predicción conjunta de la palabra siguiente (forward LSTM) y la palabra anterior (backward LSTM)
- Una vez preentrenado, se puede adaptar a otras tareas de clasificación de secuencias

ELMO

Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Deep contextualized word representations
Peters et al., (2018)

Redes Neuronales Recurrentes