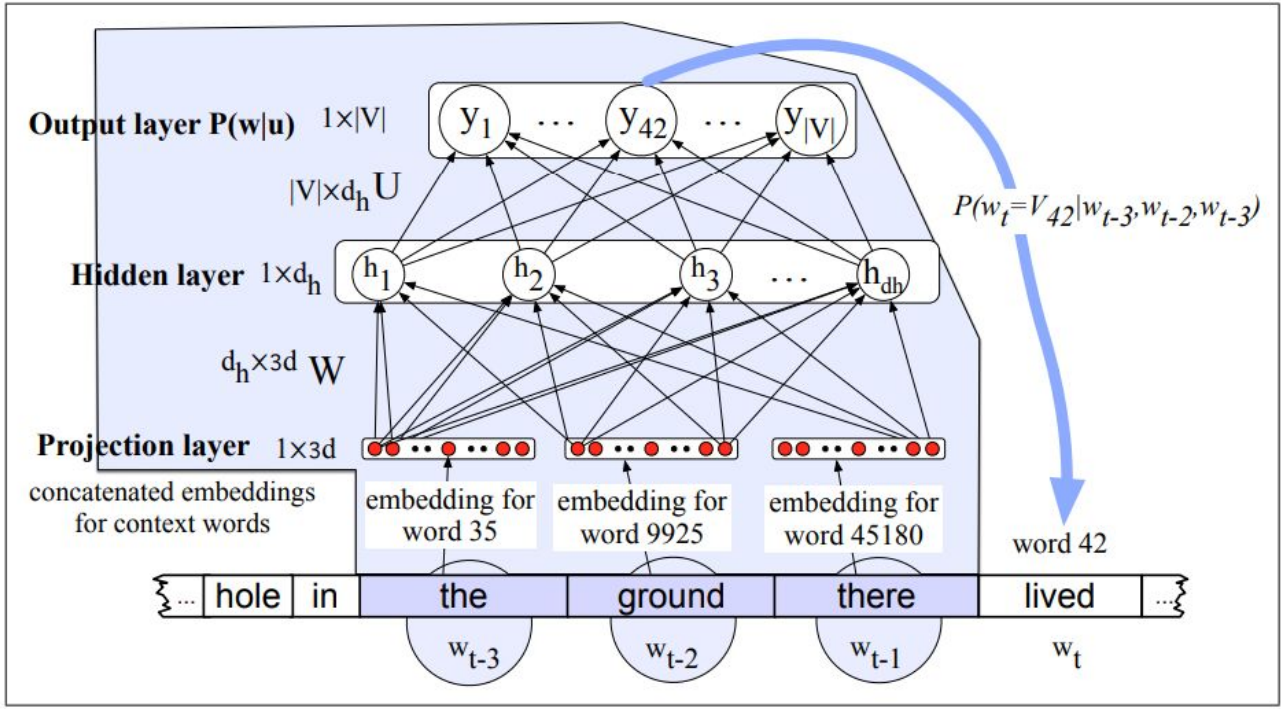
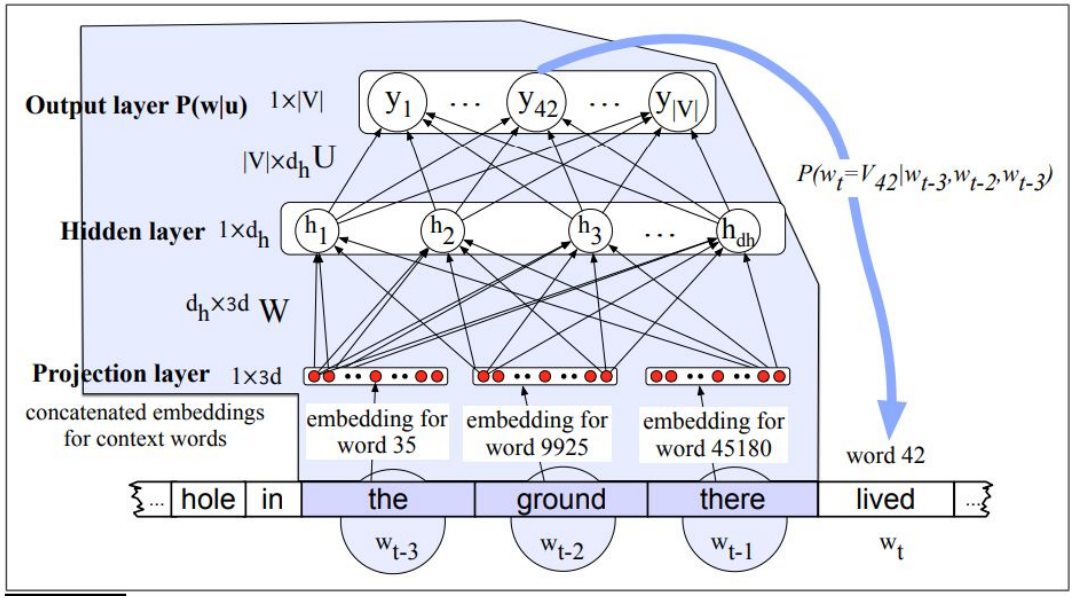
# Recurrent Neural Networks

# Neural Language Models




Jurafsky & Martin. (2019) Speech and Language Processing 3rd edition

# Feed-forward neural language models

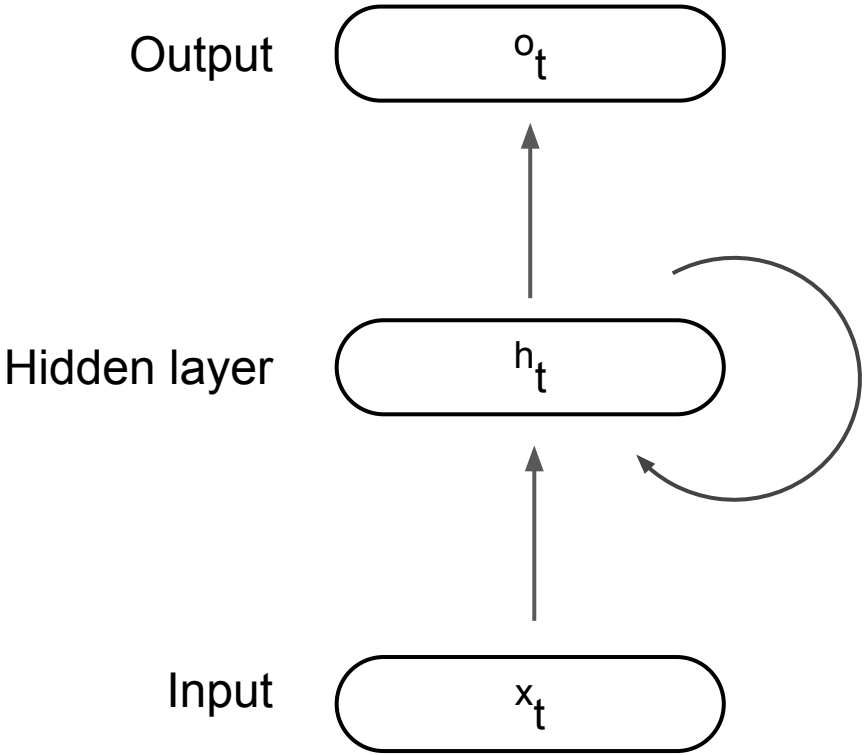



Limitaciones:

- No usa información de toda la secuencia

Jurafsky & Martin. (2019) Speech and Language Processing 3rd edition

# Recurrent Neural Network

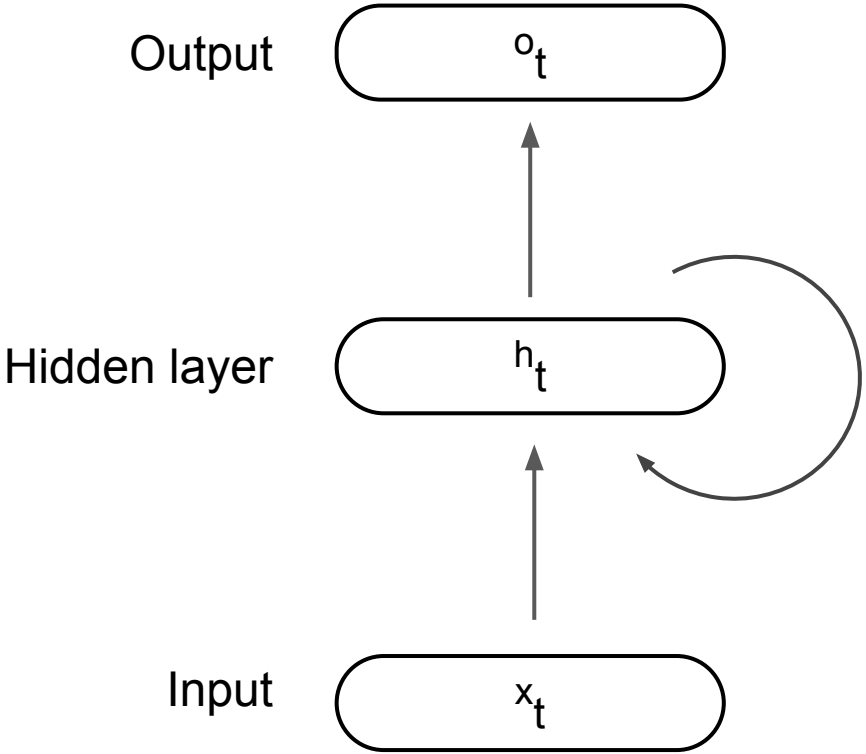



## Idea

El hidden layer del tiempo *t* ($h_t$) se alimenta del input del tiempo *t* ($x_t$) y del hidden layer del tiempo anterior ($h_{t-1}$)

## Ventajas

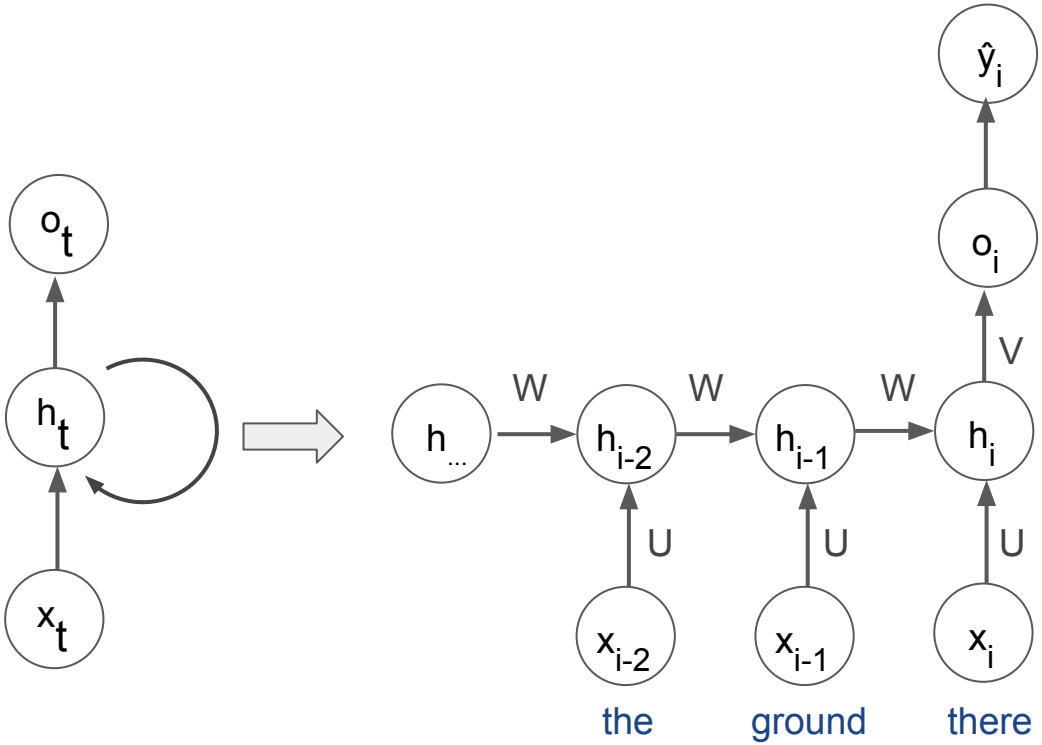
- En principio puede usar información de toda la secuencia.
- El tamaño del modelo no depende del largo del input




embedding of word *t*

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

Goodfellow, Bengio, and Courville (2016) *Deep learning*.

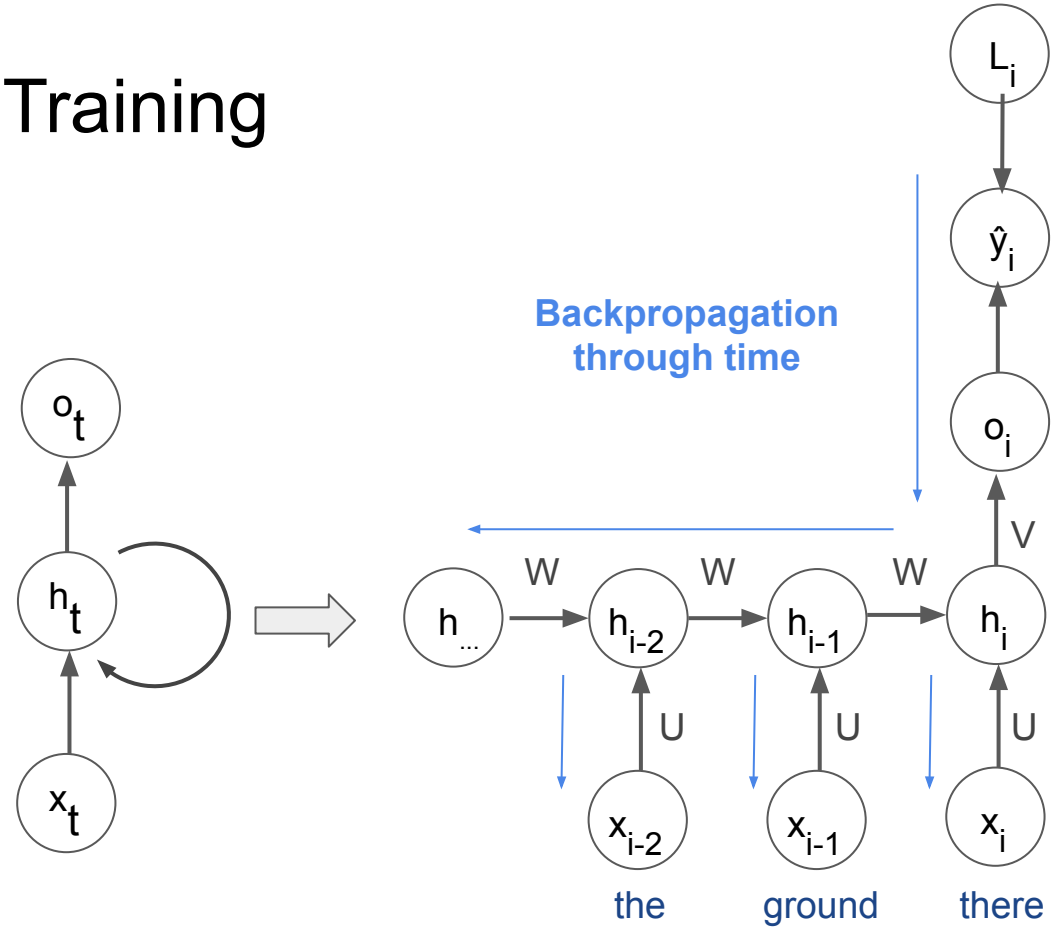



$$\hat{y}_i = p(x_{i+1} | \text{...the ground there})$$

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

sentence: “... hole in the ground there lived …”

# Training



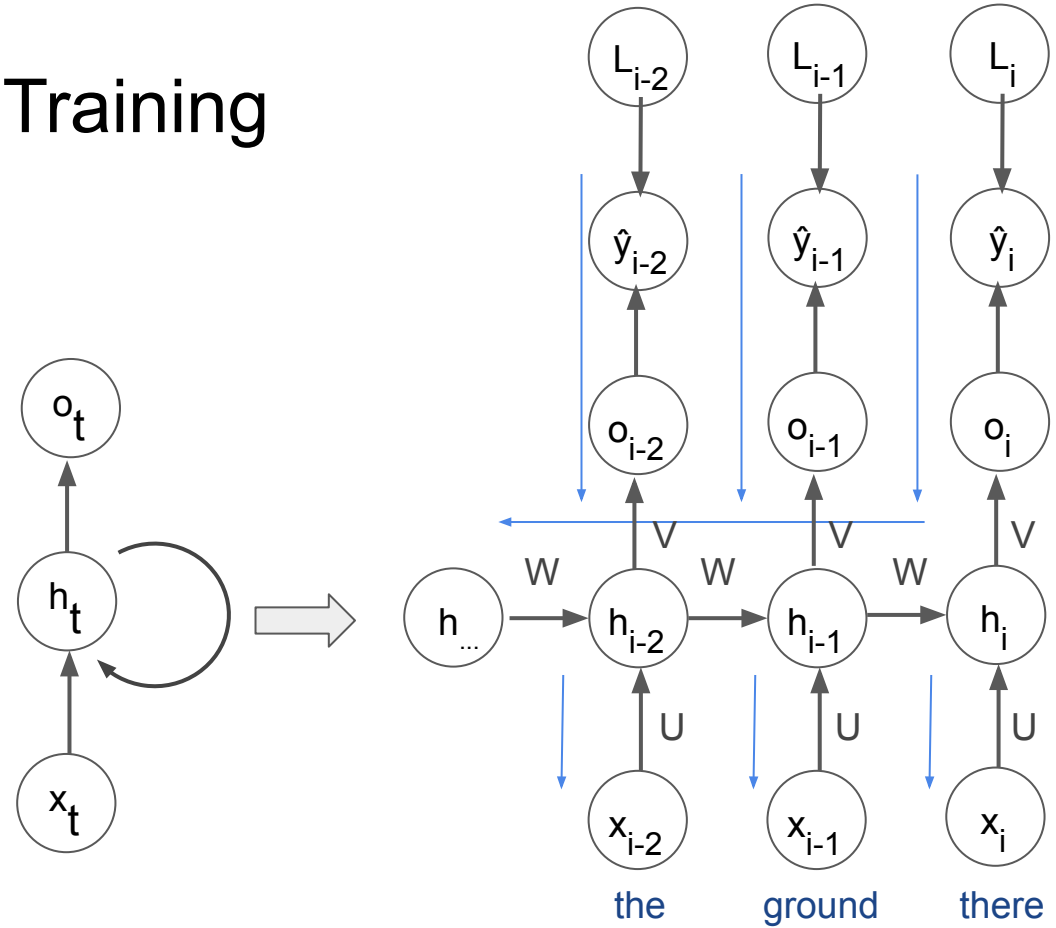

$$\mathcal{L}_i = -\sum_{w \in V} y_i^w \log(\hat{y}_i^w)$$

$$\hat{y}_i = p(x_{i+1} | \text{...the ground there})$$

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

sentence: “... hole in the ground there lived …”

# Training

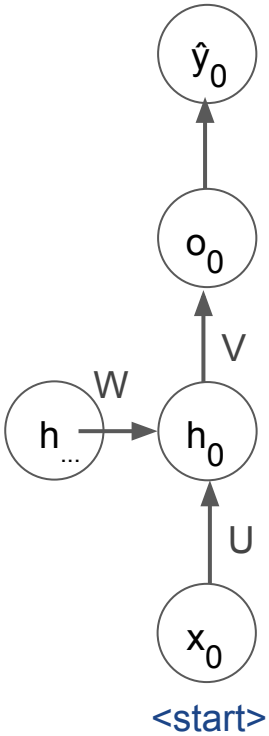



$$\mathcal{L}_i = -\sum_{w \in V} y_i^w \log(\hat{y}_i^w)$$

$$\mathcal{L} = \sum_i \mathcal{L}_i$$

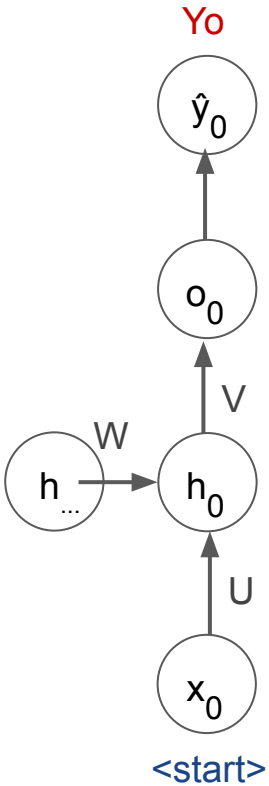
sentence: “... hole in the ground there lived …”

# Text generation



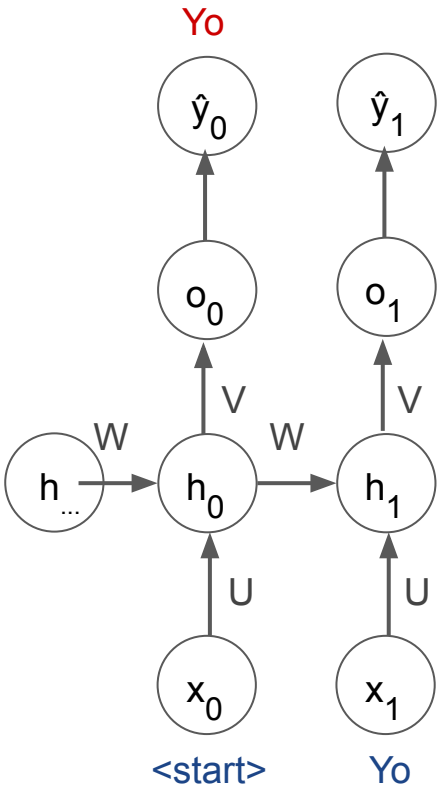

Sampleo de la distribución de probabilidad

# Text generation

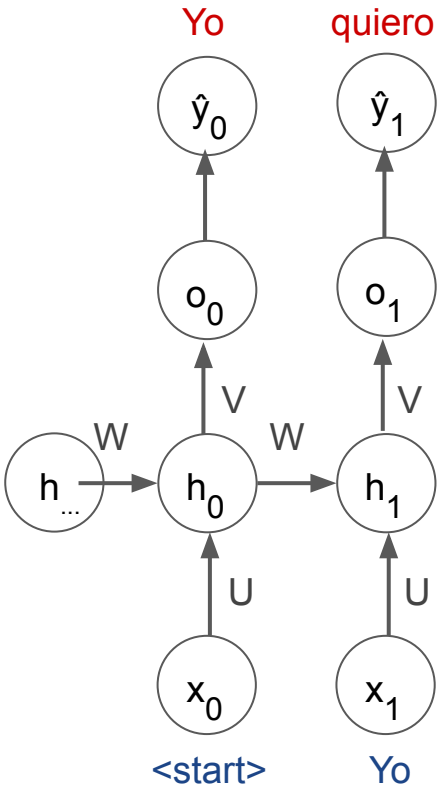



Sampleo de la distribución de probabilidad

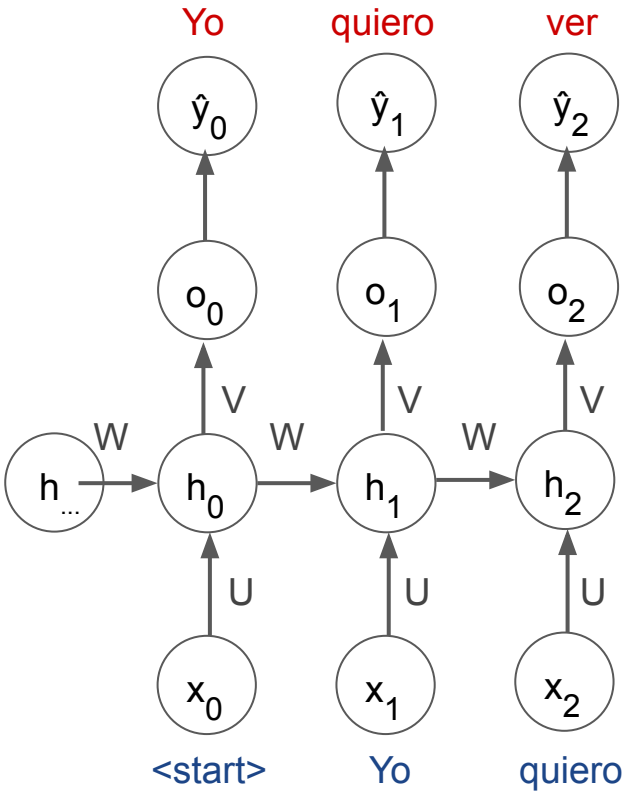
# Text generation



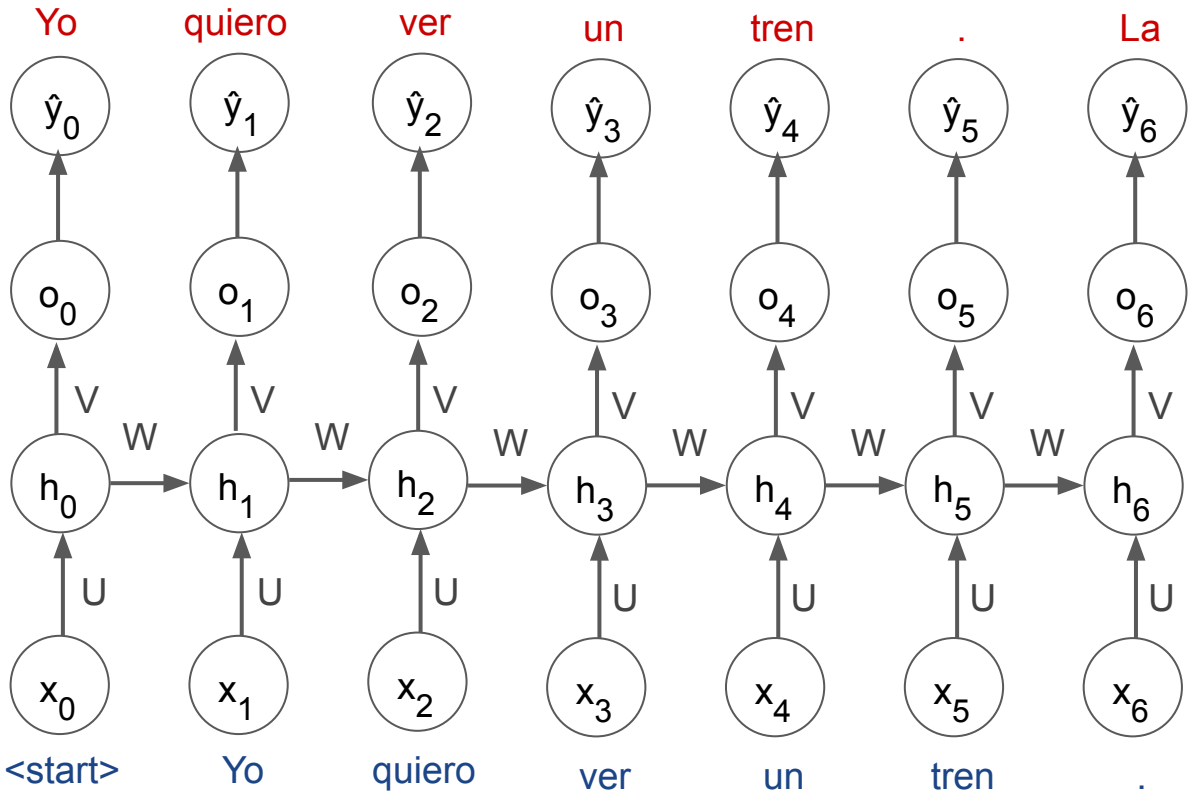
# Text generation



# Text generation



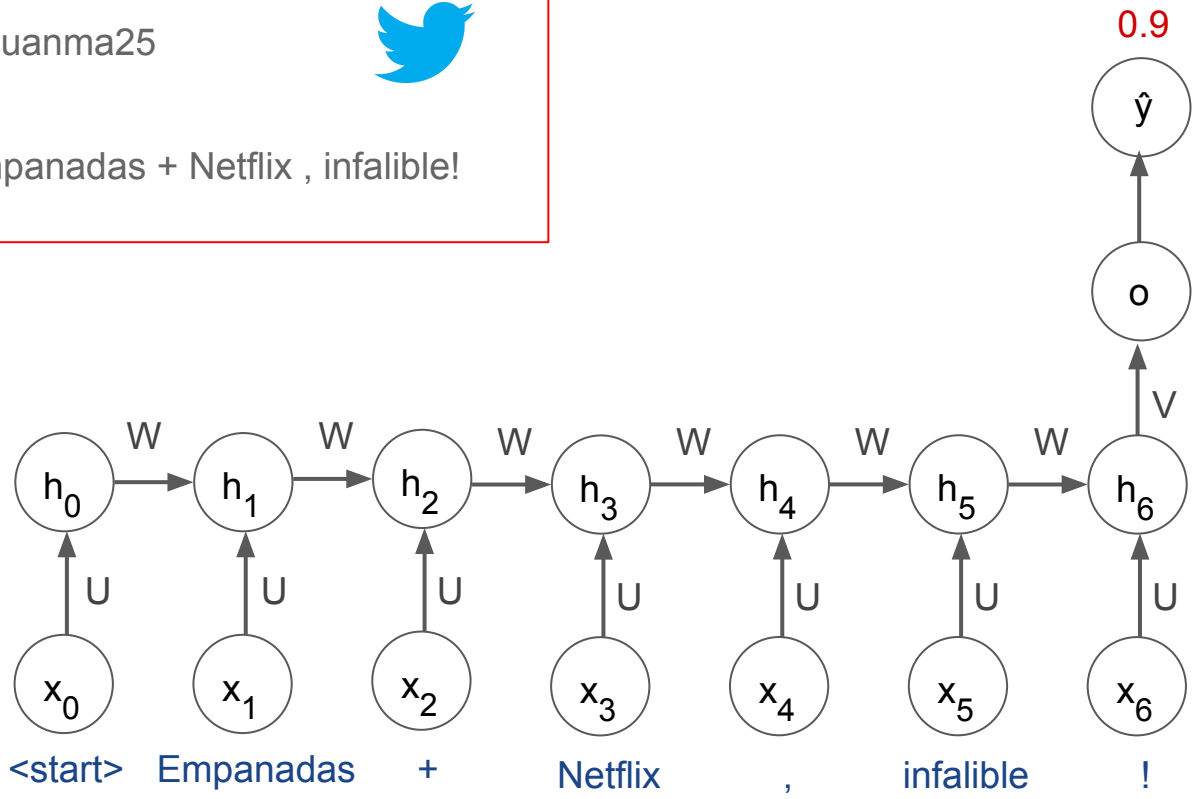
# Text generation



# Clasificación

@juanma25

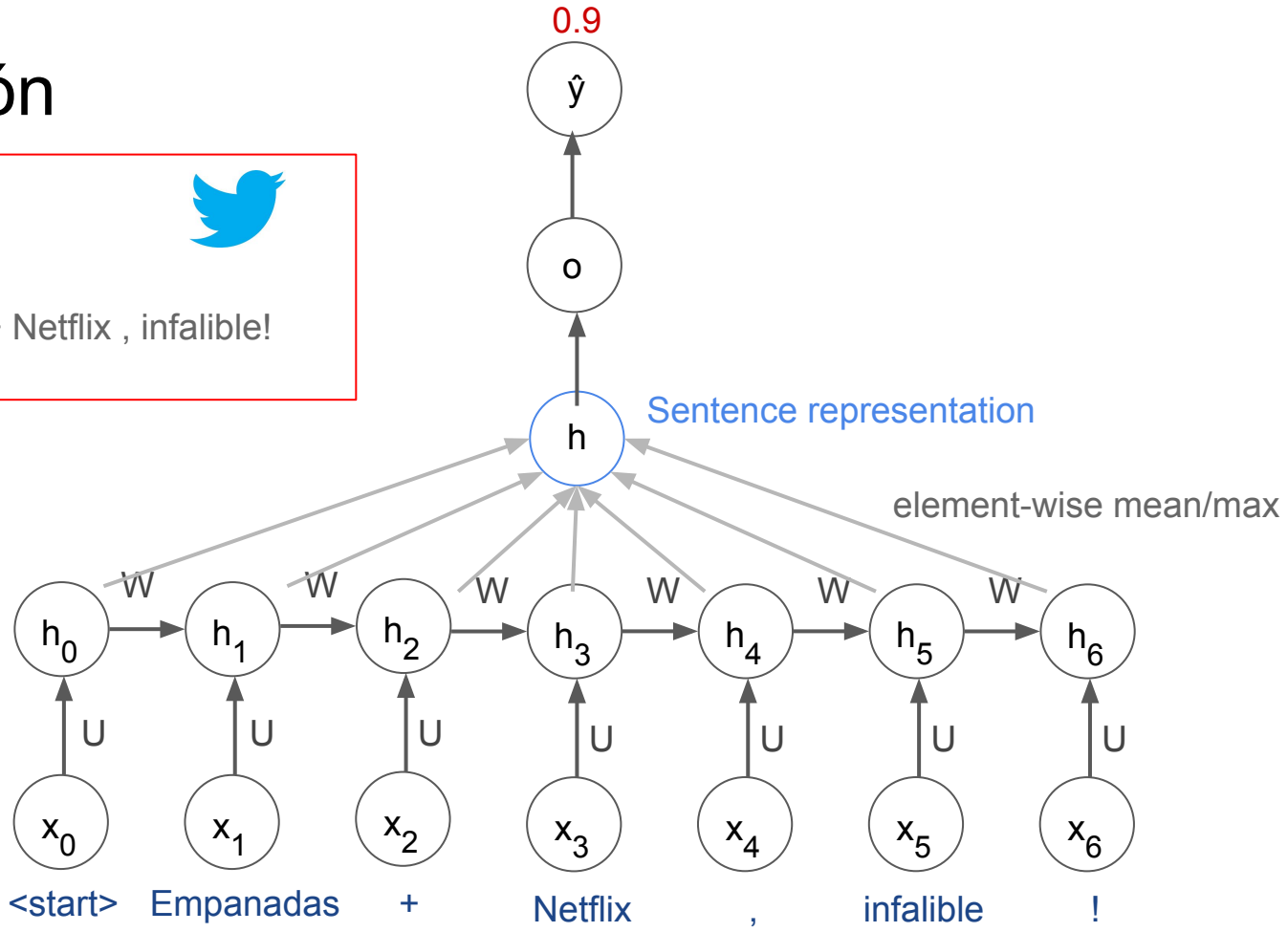
Empanadas + Netflix , infalible!



# Clasificación

@juanma25

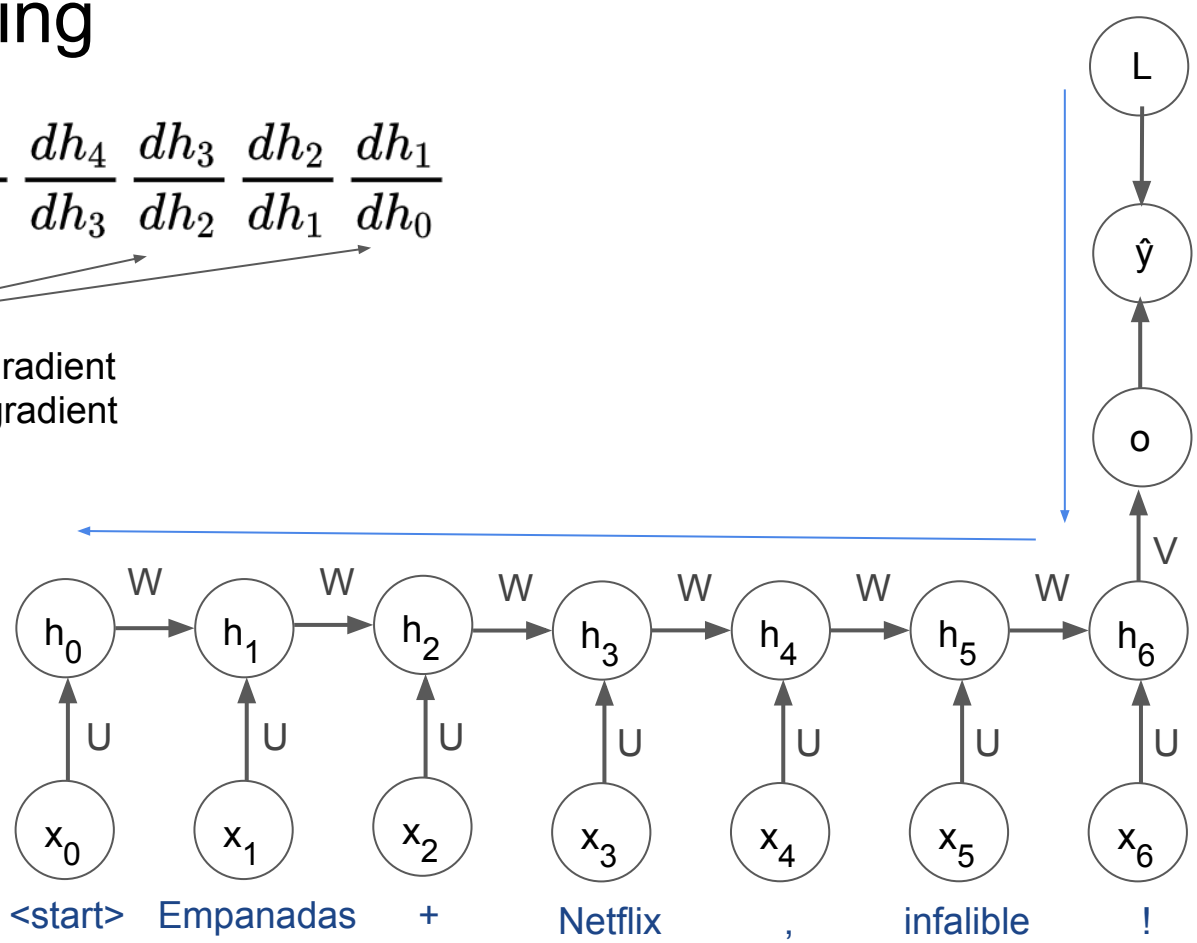
Empanadas + Netflix , infalible!



# Gradient in training

$$\frac{d\mathcal{L}}{dh_0} = \frac{d\mathcal{L}}{dh_6}\frac{dh_6}{dh_5}\frac{dh_5}{dh_4}\frac{dh_4}{dh_3}\frac{dh_3}{dh_2}\frac{dh_2}{dh_1}\frac{dh_1}{dh_0}$$

- Si son chicos: vanishing gradient
- Si son grandes: exploding gradient



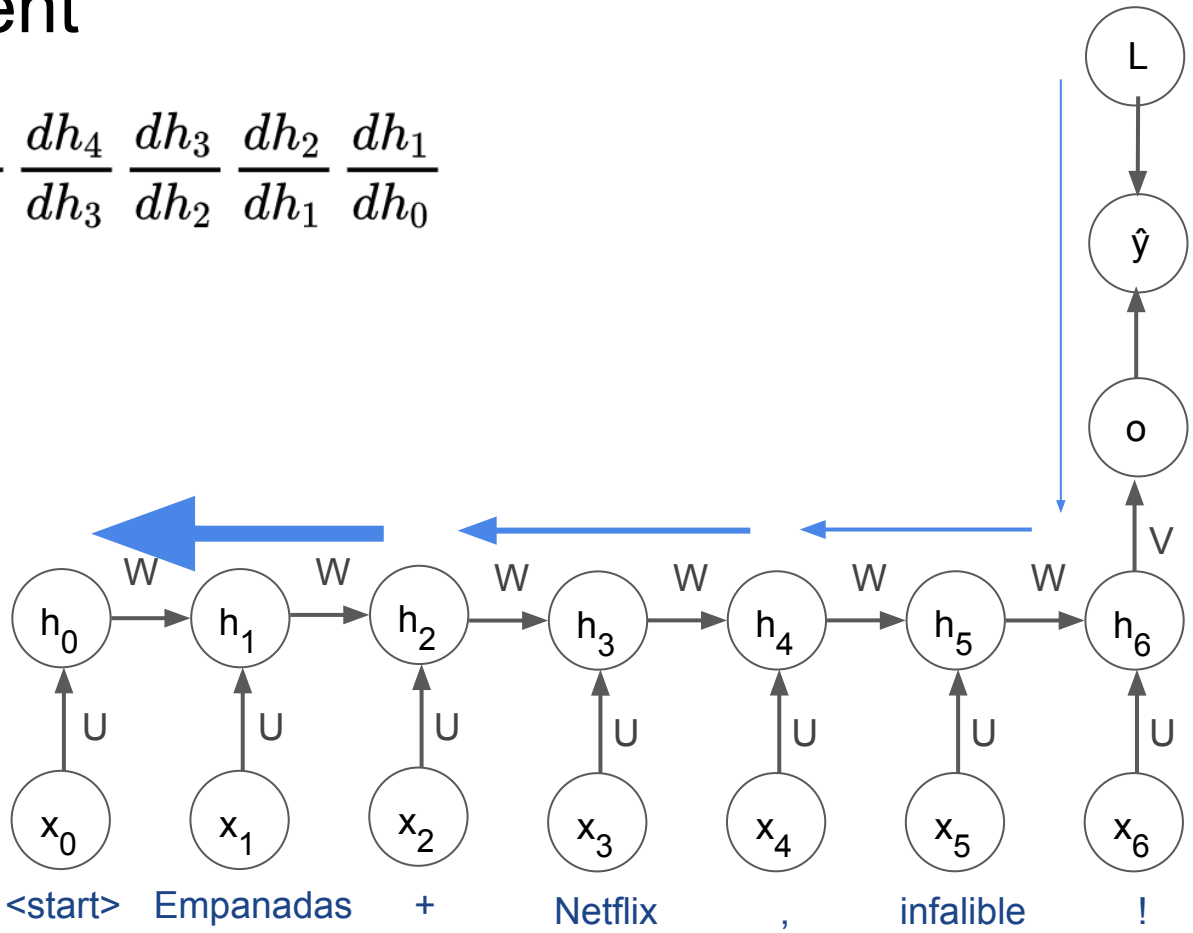
# Exploding gradient

$$\frac{d\mathcal{L}}{dh_0} = \frac{d\mathcal{L}}{dh_6}\frac{dh_6}{dh_5}\frac{dh_5}{dh_4}\frac{dh_4}{dh_3}\frac{dh_3}{dh_2}\frac{dh_2}{dh_1}\frac{dh_1}{dh_0}$$

**parameter update**

$$\theta_{new} = \theta_{old} - \lambda\frac{d\mathcal{L}}{d\theta}$$

- puede generar pasos muy grandes en el espacio de parámetros
- puede generar *Inf*



# Exploding gradient

## Gradient clipping

$$\vec{\theta}_{new} = \vec{\theta}_{old} - \lambda \vec{\nabla}_\theta \mathcal{L}$$

$$if \;\; ||\vec{\nabla}_\theta \mathcal{L}|| > v_{max} :$$

$$\vec{\nabla}_\theta \mathcal{L} \leftarrow \frac{\vec{\nabla}_\theta \mathcal{L}}{||\vec{\nabla}_\theta \mathcal{L}||} v_{max}$$

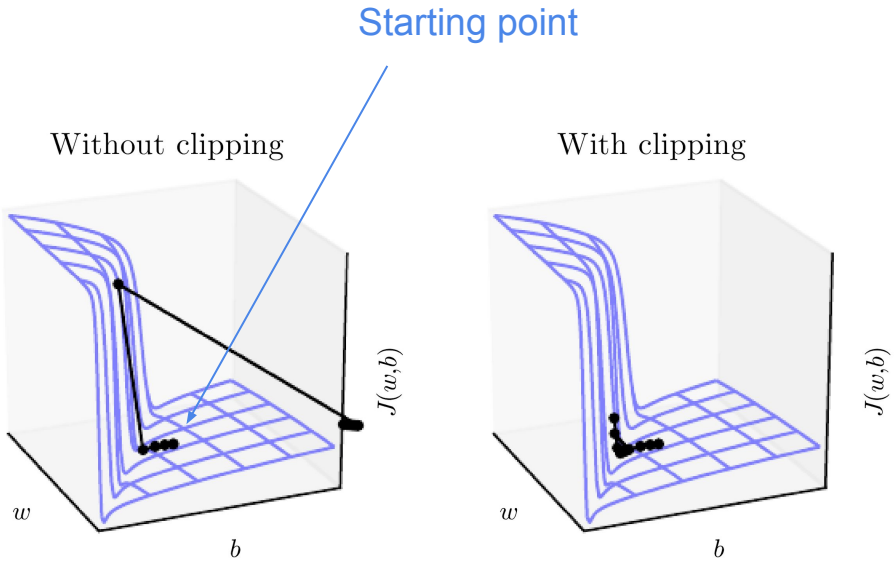
# Exploding gradient

## Gradient clipping

$$\vec{\theta}_{new} = \vec{\theta}_{old} - \lambda \vec{\nabla}_\theta \mathcal{L}$$

$$if \;\; ||\vec{\nabla}_\theta \mathcal{L}|| > v_{max} :$$

$$\vec{\nabla}_\theta \mathcal{L} \leftarrow \frac{\vec{\nabla}_\theta \mathcal{L}}{||\vec{\nabla}_\theta \mathcal{L}||} v_{max}$$




**Panel izquierdo**
La primer pendiente abrupta (a la derecha del mínimo) permite subir por el “acantilado” para luego dar un paso muy largo

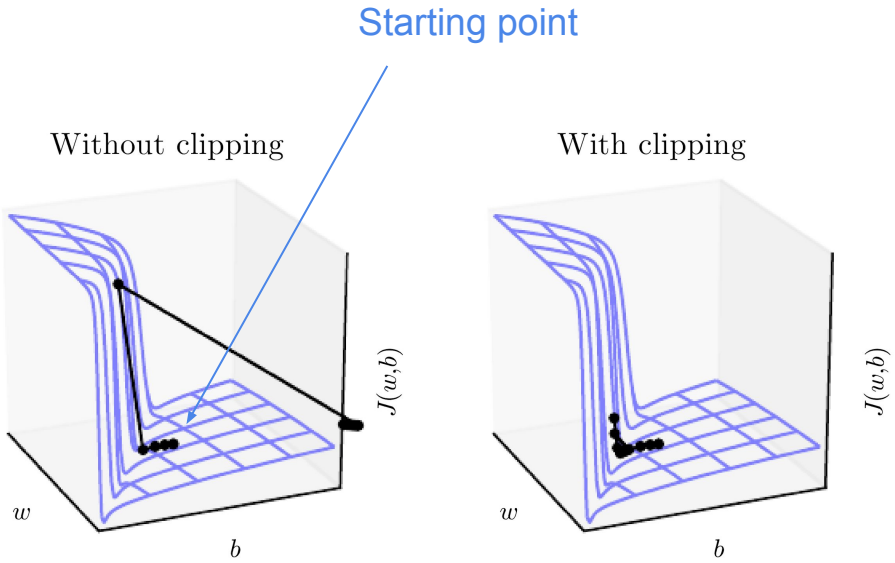
# Exploding gradient

## Gradient clipping

$$\vec{\theta}_{new} = \vec{\theta}_{old} - \lambda \vec{\nabla}_\theta \mathcal{L}$$

$$if \;\; ||\vec{\nabla}_\theta \mathcal{L}|| > v_{max} :$$

$$\vec{\nabla}_\theta \mathcal{L} \leftarrow \frac{\vec{\nabla}_\theta \mathcal{L}}{||\vec{\nabla}_\theta \mathcal{L}||} v_{max}$$




**Panel izquierdo**
La primer pendiente abrupta (a la derecha del mínimo) permite subir por el “acantilado” para luego dar un paso muy largo

**Panel derecho**
el *gradient clipping* acota los pasos, por lo que no sube el “acantilado”

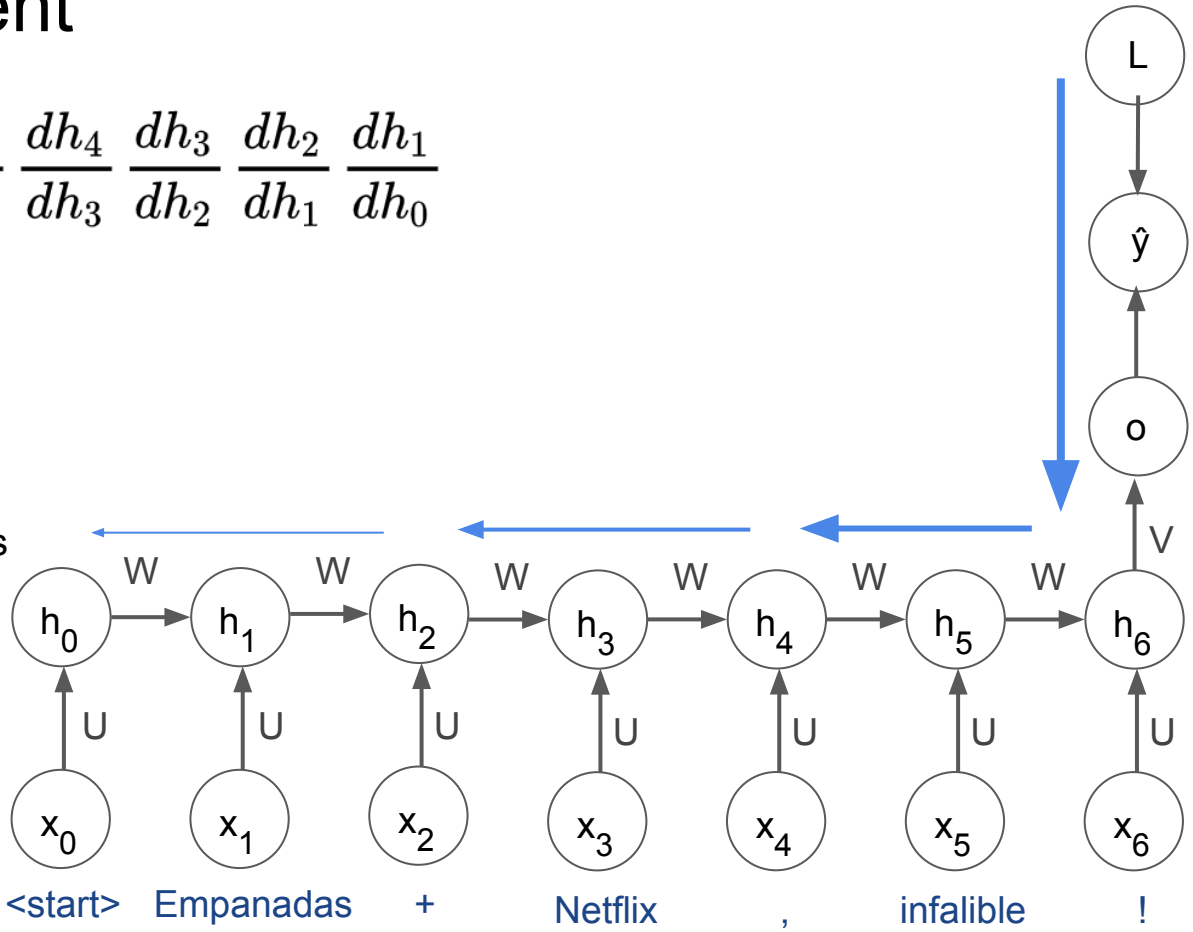
# Vanishing gradient

$$\frac{d\mathcal{L}}{dh_0} = \frac{d\mathcal{L}}{dh_6}\frac{dh_6}{dh_5}\frac{dh_5}{dh_4}\frac{dh_4}{dh_3}\frac{dh_3}{dh_2}\frac{dh_2}{dh_1}\frac{dh_1}{dh_0}$$

**parameter update**

$$\theta_{new} = \theta_{old} - \lambda\frac{d\mathcal{L}}{d\theta}$$

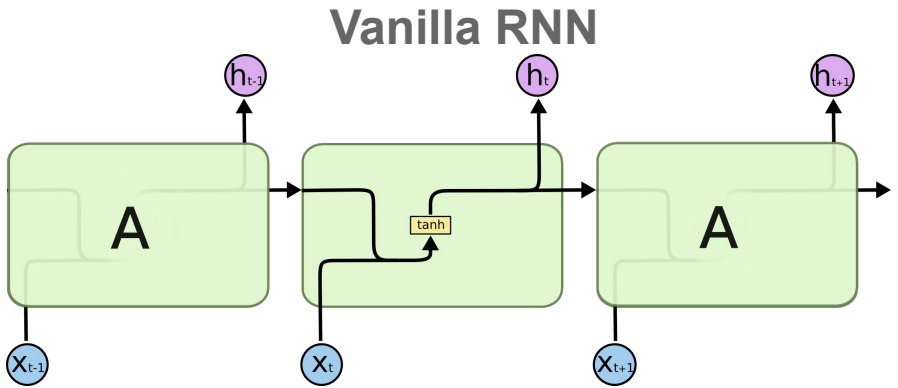
- Se pierden las dependencias de largo alcance



# Gated Recurrent Units (GRU)

## Idea:

- Facilitar la retención de información de largo alcance
- los estados ocultos $h_i$ pueden ser actualizados o no en cada iteración
- La actualización de los estados se controlan con *gates.* los *gates* toman valores entre 0 (cerrado) y 1 (abierto).
- Tanto el contenido de la actualización como los *gates* están determinados tanto por el estado anterior como por el input.

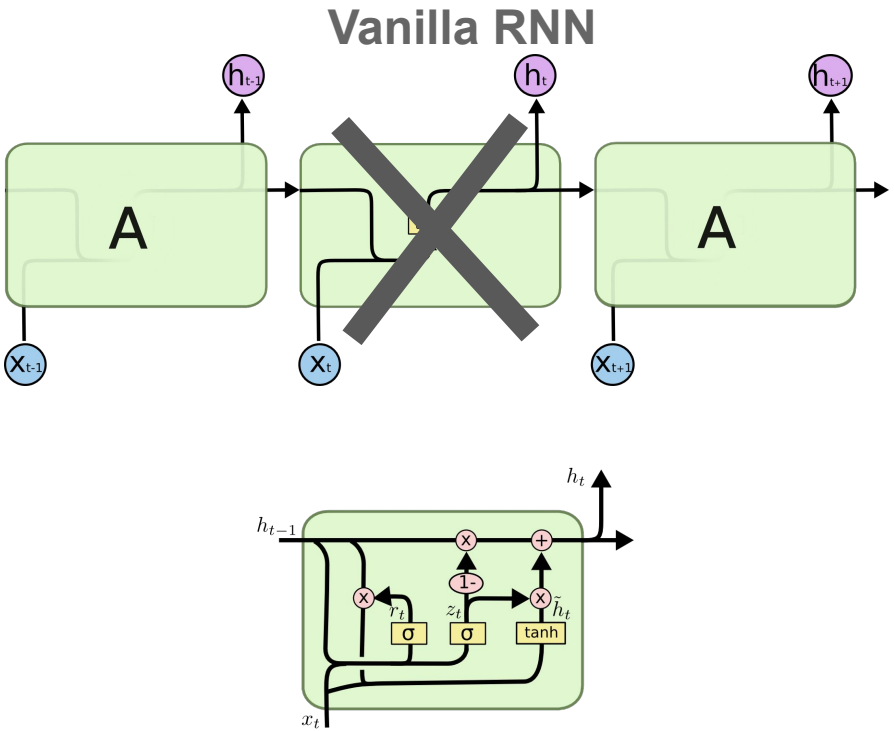



Cho et al (2014) Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# Gated Recurrent Units (GRU)

## Idea:

- Facilitar la retención de información de largo alcance
- los estados ocultos $h_i$ pueden ser actualizados o no en cada iteración
- La actualización de los estados se controlan con *gates*. los *gates* toman valores entre 0 (cerrado) y 1 (abierto).
- Tanto el contenido de la actualización como los *gates* están determinados tanto por el estado anterior como por el input.




Cho et al (2014) Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation

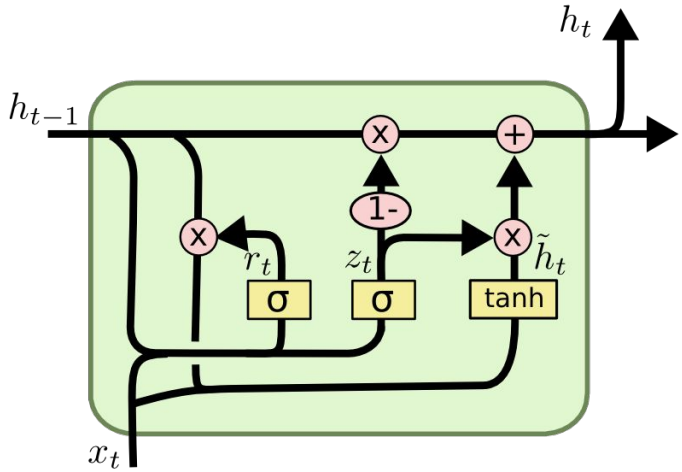
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# Gated Recurrent Units (GRU)

new hidden state content

reset gate

$$\tilde{\mathbf{h}}^{(t)} = \tanh(\mathbf{b}_h + \mathbf{W}_h(\mathbf{r}^{(t)} \circ \mathbf{h}^{(t-1)}) + \mathbf{U}_h\mathbf{x}^{(t)})$$




Cho et al (2014) Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation

# Gated Recurrent Units (GRU)

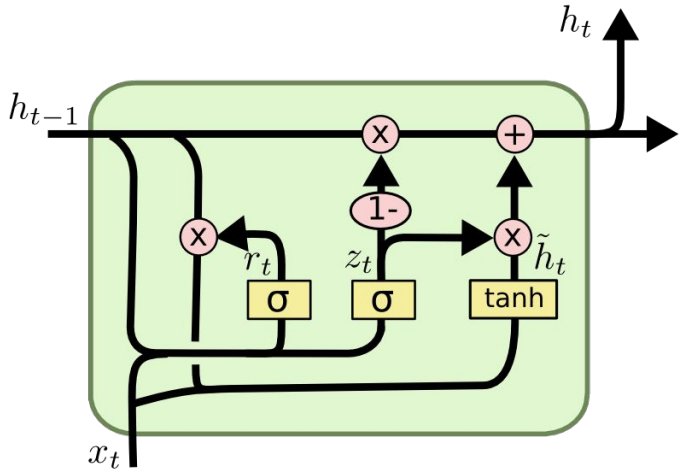
new hidden state content

reset gate

$$\tilde{\mathbf{h}}^{(t)} = \tanh(\mathbf{b}_h + \mathbf{W}_h(\mathbf{r}^{(t)} \circ \mathbf{h}^{(t-1)}) + \mathbf{U}_h\mathbf{x}^{(t)})$$

$$\mathbf{h^{(t)}} = (1 - \mathbf{z}^t) \circ \mathbf{h}^{(t-1)} + \mathbf{z}^{(t)} \circ \tilde{\mathbf{h}}^{(t)}$$

update gate




Cho et al (2014) Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation

# Gated Recurrent Units (GRU)

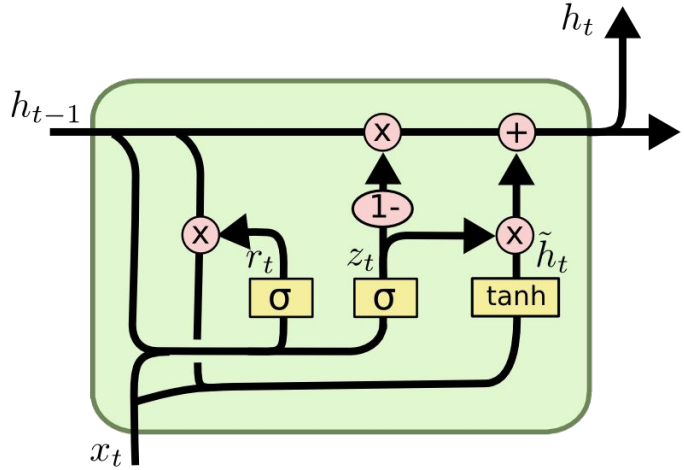
new hidden state content

reset gate

$$\tilde{\mathbf{h}}^{(t)} = \tanh(\mathbf{b}_h + \mathbf{W}_h(\mathbf{r}^{(t)} \circ \mathbf{h}^{(t-1)}) + \mathbf{U}_h\mathbf{x}^{(t)})$$

$$\mathbf{h}^{(t)} = (1 - \mathbf{z}^t) \circ \mathbf{h}^{(t-1)} + \mathbf{z}^{(t)} \circ \tilde{\mathbf{h}}^{(t)}$$

update gate




$$\mathbf{z}^{(t)} = \sigma(\mathbf{b}_u + \mathbf{W}_u\mathbf{h}^{(t-1)} + \mathbf{U}_u\mathbf{x}^{(t)})$$

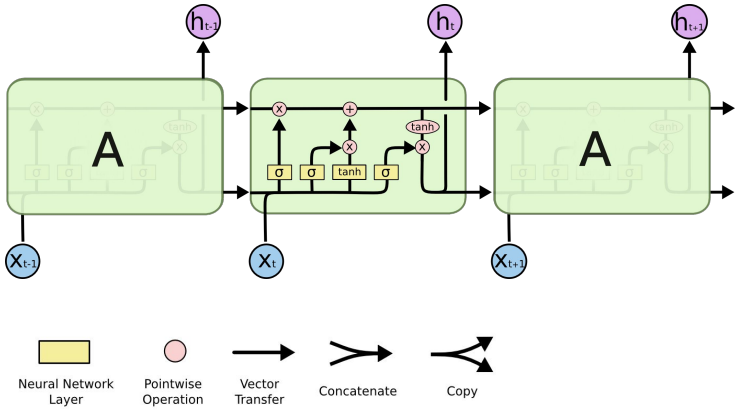
$$\mathbf{r}^{(t)} = \sigma(\mathbf{b}_r + \mathbf{W}_r\mathbf{h}^{(t-1)} + \mathbf{U}_r\mathbf{x}^{(t)})$$

Cho et al (2014) Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation

# Long short-term memory (LSTM)

## Idea:

- Ademas de estado oculto $h_i$, el LSTM guarda información de largo alcance en otra unidad llamada *cell state* $c_i$




S Hochreiter, J Schmidhube, 1997 Long short-term memory
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# Long short-term memory (LSTM)

cell state

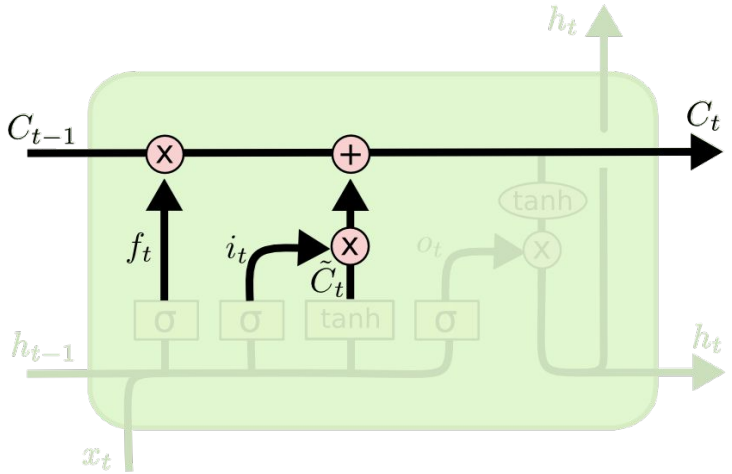
forget gate

input gate

new cell content

$$\mathbf{c}^{(t)} = \mathbf{f}^{t} \circ \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \circ \tilde{\mathbf{c}}^{(t)}$$

$$\tilde{\mathbf{c}}^{(t)} = \sigma(\mathbf{b}_c + \mathbf{W}_c \mathbf{h}^{(t-1)} + \mathbf{U}_c \mathbf{x}^{(t)})$$



# Long short-term memory (LSTM)

cell state

forget gate

input gate

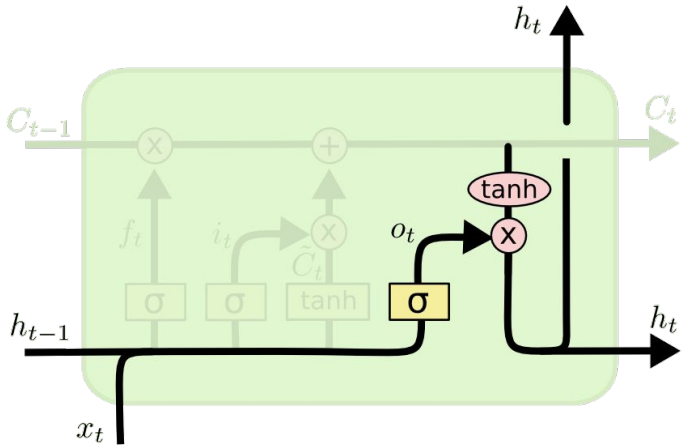
new cell content

$$\mathbf{c}^{(t)} = \mathbf{f}^{t} \circ \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \circ \tilde{\mathbf{c}}^{(t)}$$

$$\tilde{\mathbf{c}}^{(t)} = \sigma(\mathbf{b}_c + \mathbf{W}_c \mathbf{h}^{(t-1)} + \mathbf{U}_c \mathbf{x}^{(t)})$$

$$\mathbf{h}^{(t)} = \mathbf{o}^{t} \circ \tanh(\mathbf{c}^{(t)})$$

output gate



# Long short-term memory (LSTM)

$$\mathbf{c^{(t)}} = \mathbf{f}^t \circ \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \circ \tilde{\mathbf{c}}^{(t)}$$

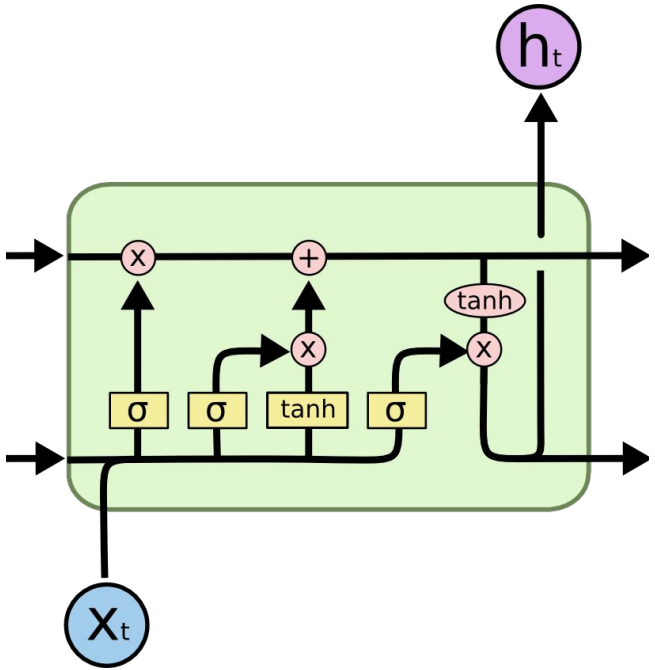
$$\tilde{\mathbf{c}}^{(t)} = \sigma(\mathbf{b}_c + \mathbf{W}_c \mathbf{h}^{(t-1)} + \mathbf{U}_c \mathbf{x}^{(t)})$$

$$\mathbf{h}^{(t)} = \mathbf{o}^t \circ \tanh(\mathbf{c^{(t)}})$$

$$\mathbf{f}^{(t)} = \sigma(\mathbf{b}_f + \mathbf{W}_f \mathbf{h}^{(t-1)} + \mathbf{U}_f \mathbf{x}^{(t)})$$

$$\mathbf{i}^{(t)} = \sigma(\mathbf{b}_i + \mathbf{W}_i \mathbf{h}^{(t-1)} + \mathbf{U}_i \mathbf{x}^{(t)})$$

$$\mathbf{o}^{(t)} = \sigma(\mathbf{b}_o + \mathbf{W}_o \mathbf{h}^{(t-1)} + \mathbf{U}_o \mathbf{x}^{(t)})$$

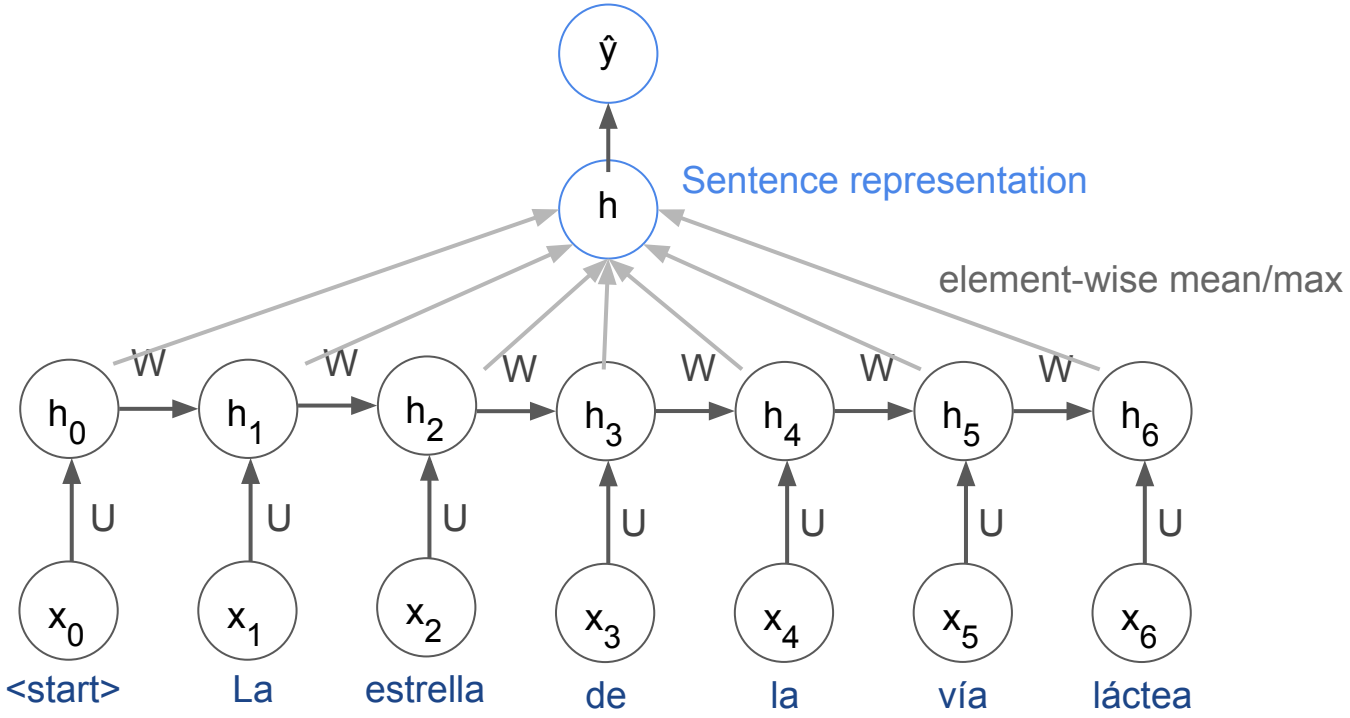


# LSTM vs GRU

- No hay evidencias fuertes de que consistentemente un modelo sea mejor a otro (entre LSTM y GRU)

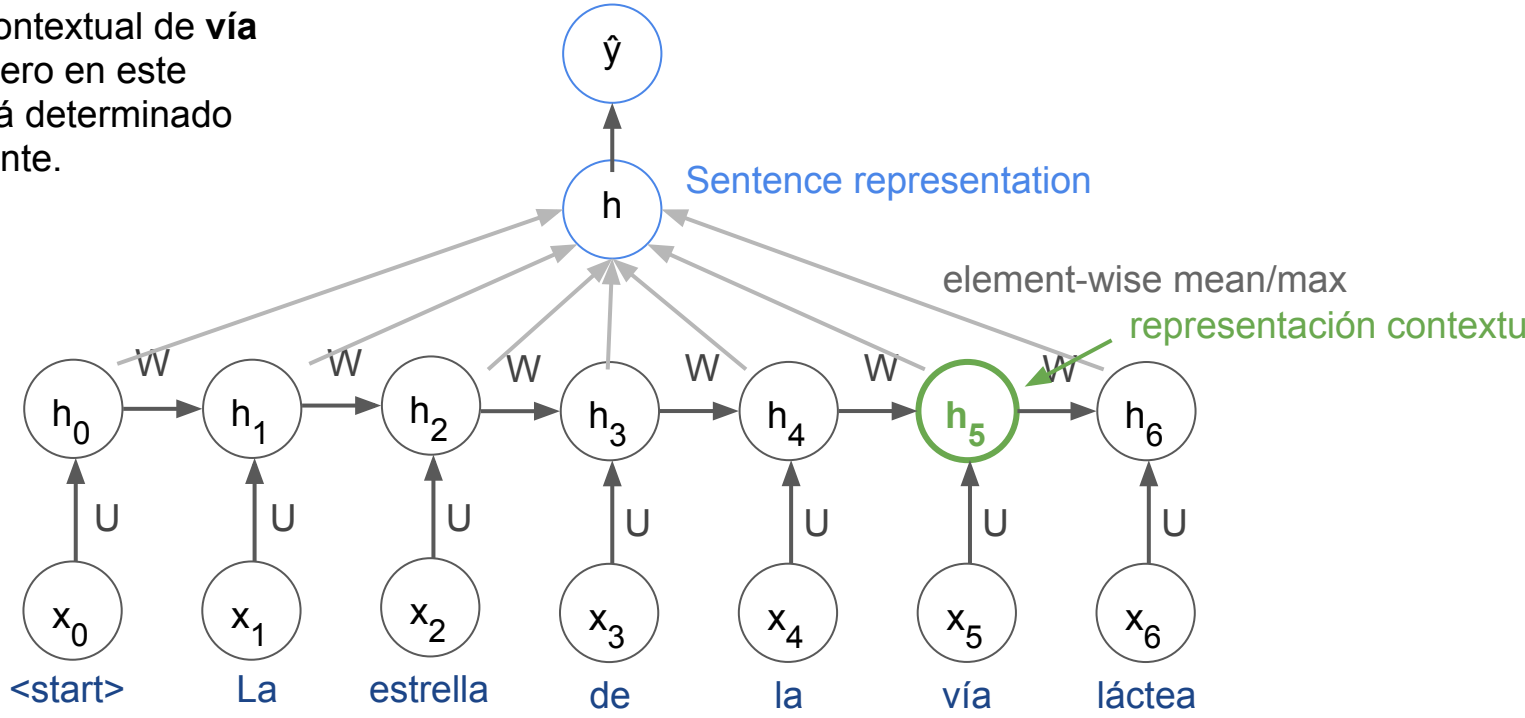
- Dado que GRU tiene menos parámetros, es mas eficiente computacionalmente

# Volviendo a las RNN

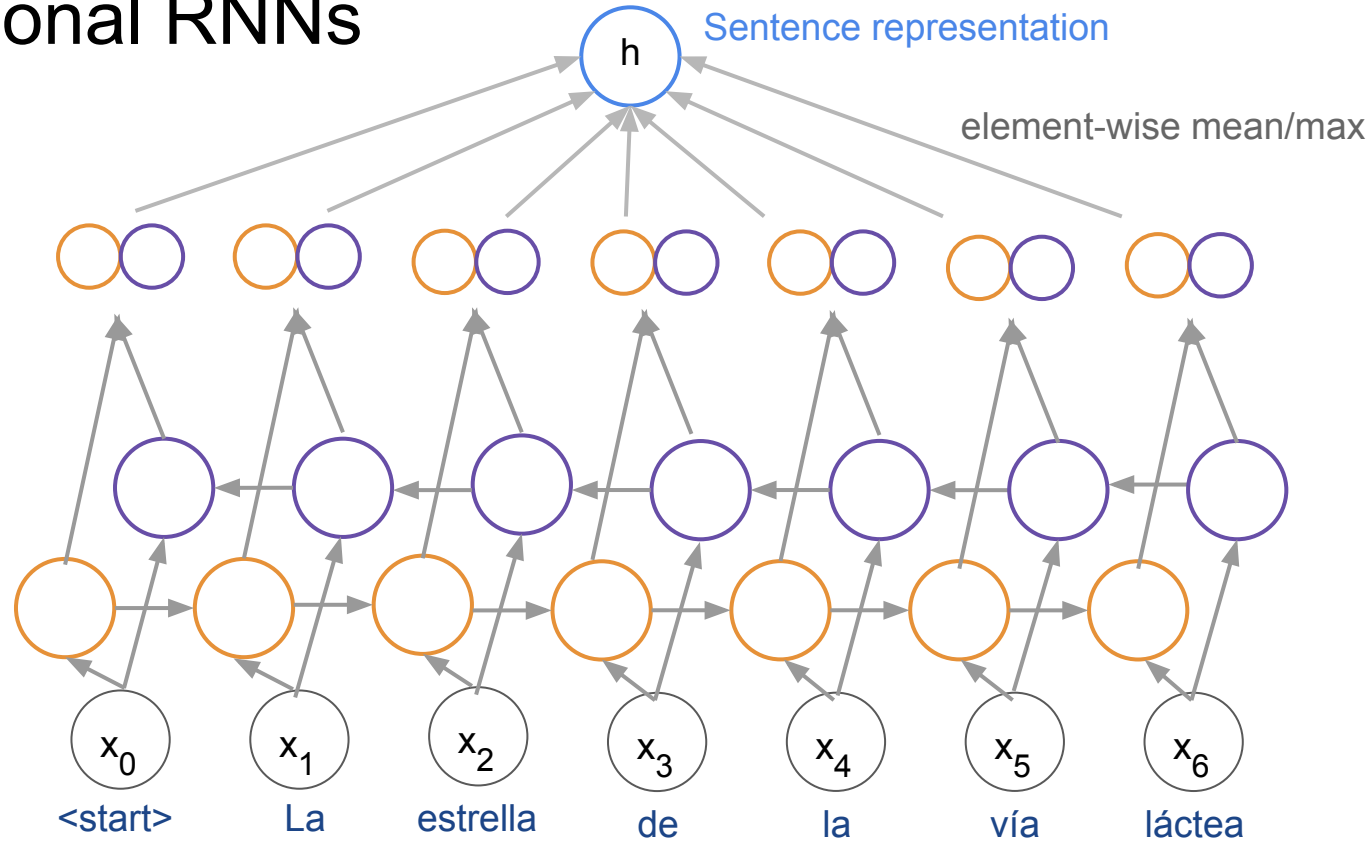


# Volviendo a las RNN

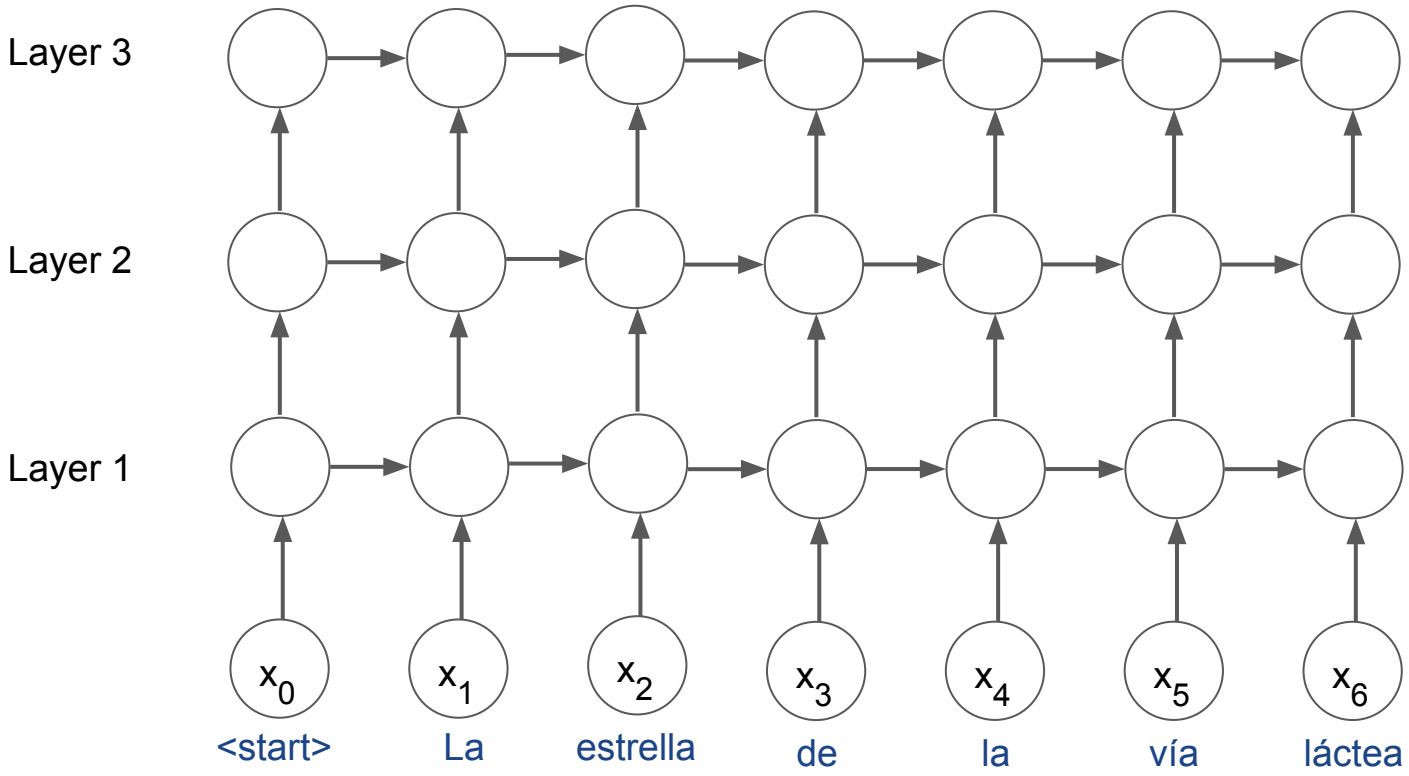
La representación contextual de **vía** solo ve el pasado. Pero en este caso el contexto está determinado por la palabra siguiente.



# Bidirectional RNNs



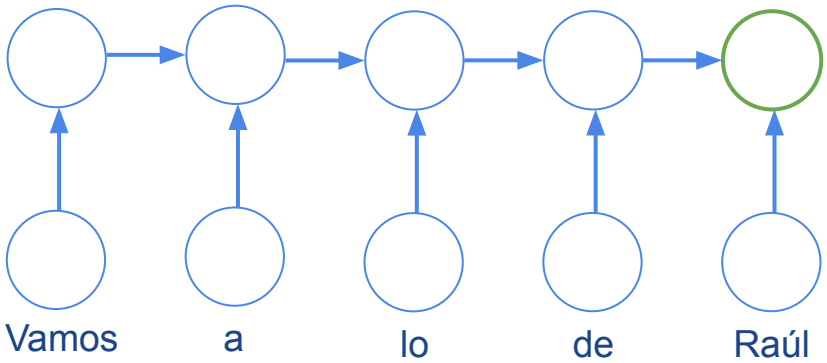
# Multi-layer RNNs



# Sequence to sequence model

## Traducción español -> inglés

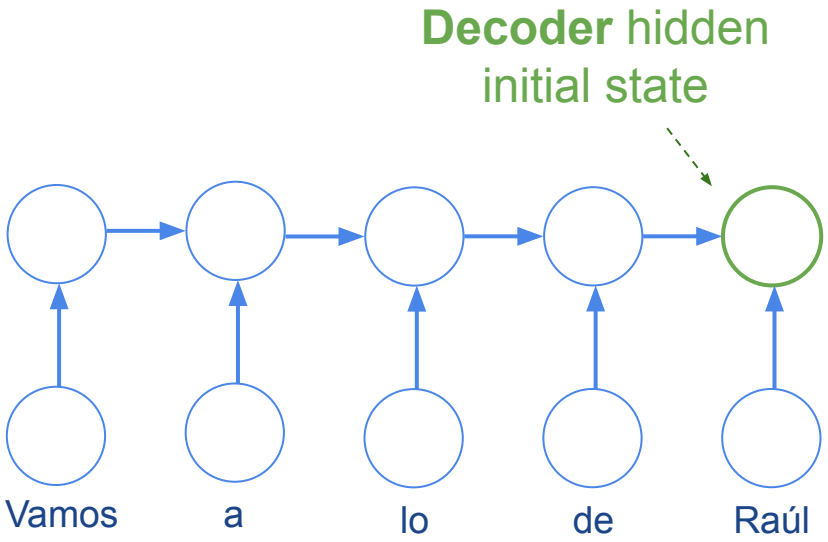
**Encoder** Rnn



# Sequence to sequence model

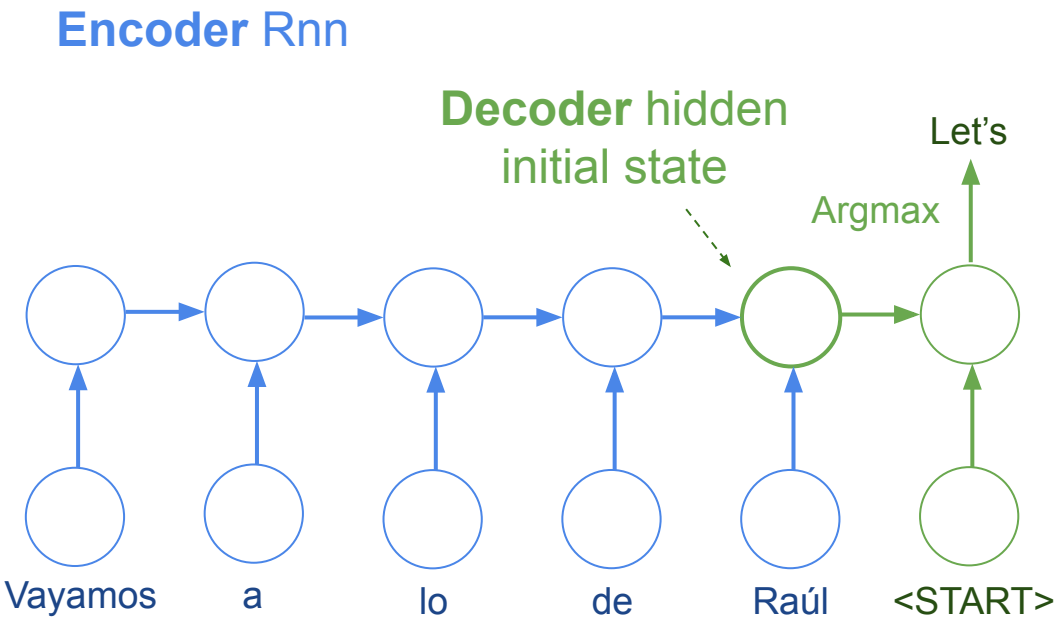
## Traducción español -> inglés

**Encoder** Rnn



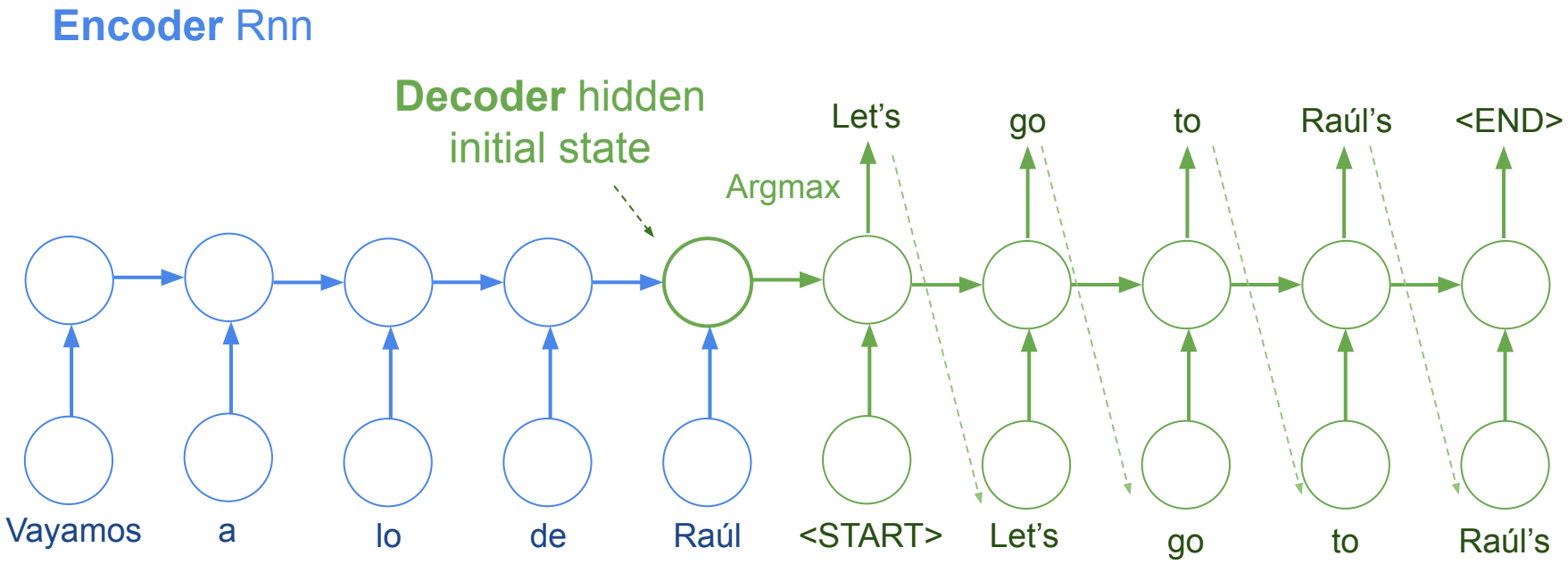
# Sequence to sequence model

## Traducción español -> inglés



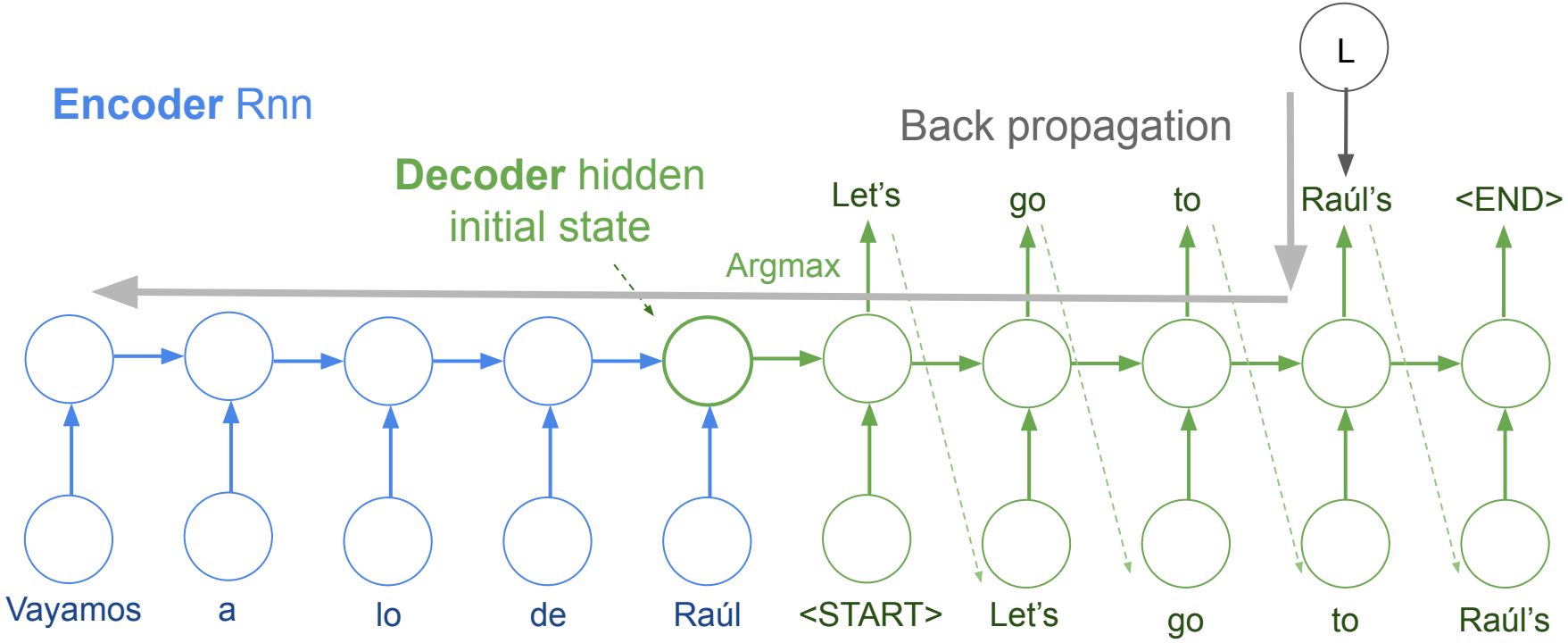
# Sequence to sequence model

## Traducción español -> inglés



# Sequence to sequence model

## Traducción español -> inglés



# Sequence to sequence models

Aplicaciones en NLP:

- Machine Translation
- Summarization
- Dialogue systems
- Image captioning

FIN