

NLP

Modelos secuenciales: RNNs Y LSTM

Docentes:

Dr. Rodrigo Cardenas Szigety

Dr. Nicolás Vattuone

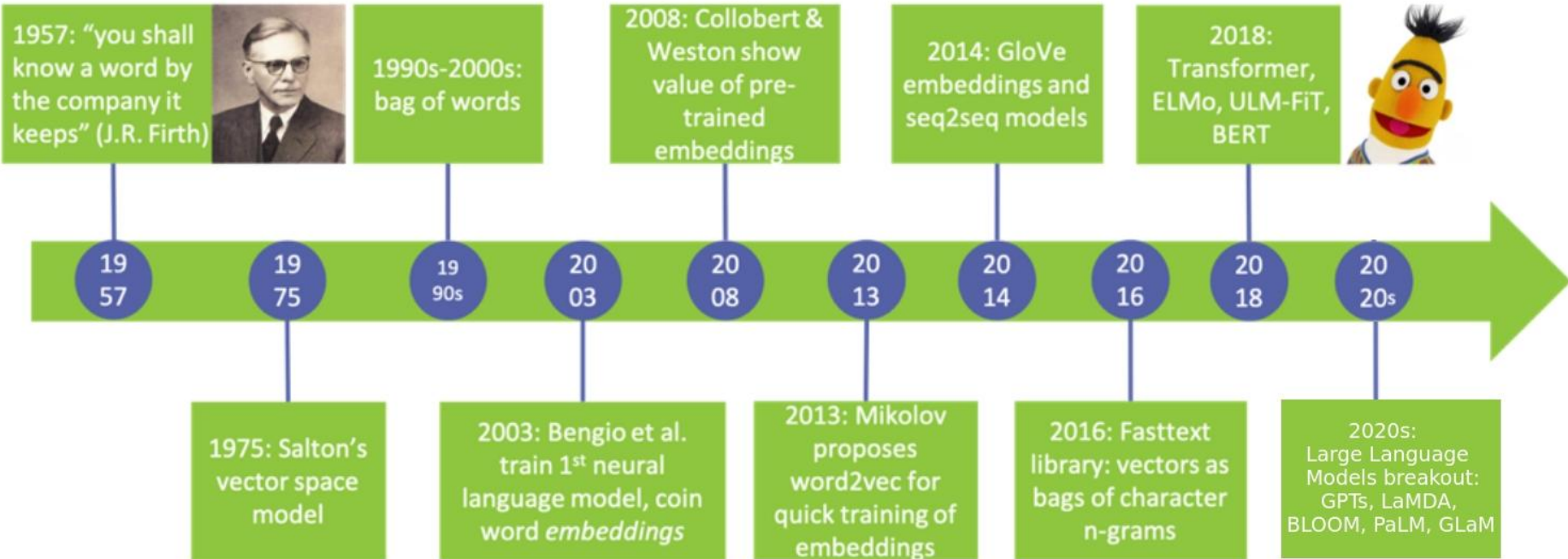
emails: rodrigo.cardenas.sz@gmail.com

nicolas.vattuone@gmail.com

Timeline



1990: Celda RNN básica (Elman)
1997: LSTM



2016: LSTMs se convierten en SotA para traducción automática

Redes Neuronales Recurrentes (RNNs)



Es un tipo de neurona con un estado interno (o memoria) de manera que la información del pasado influye en los resultados futuros.



Se utiliza principalmente para resolver problemas de secuencia, en donde el valor anterior está relacionado con el valor futuro.



Permite construir modelos cuyos tamaños de secuencia sean potencialmente arbitrarios.

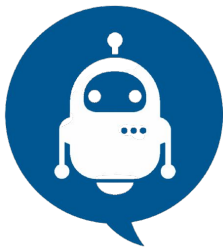


Implementa modelos de lenguaje de la forma:

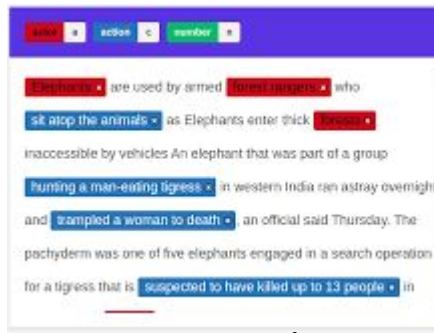
$$\prod_{i=1}^{i=m} P(w_i | w_{i-(n-1)}, \dots, w_{i-1})$$

*"Hoy el **día** está **hermoso** y **despejado**, se puede ver un hermoso **cielo... azul**"*

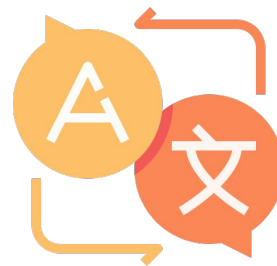
Algunos problemas de secuencia



Bots
Conversacionales



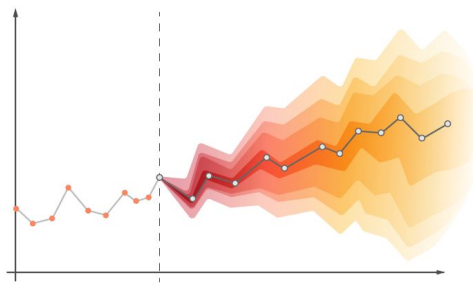
Name entity
recognition



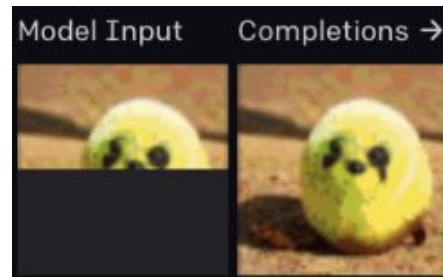
Traducción de
idiomas



Speech to text



Series
temporales



Completar una
imagen

Celda RNN básica (Elman)

[LINK](#)

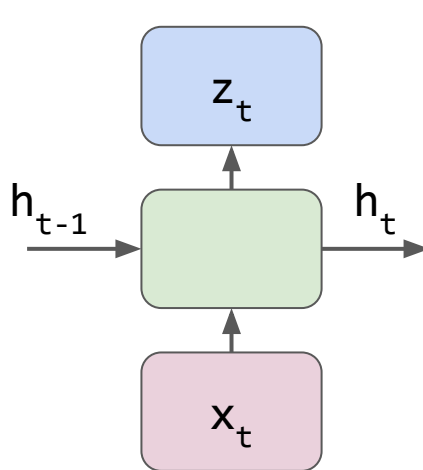
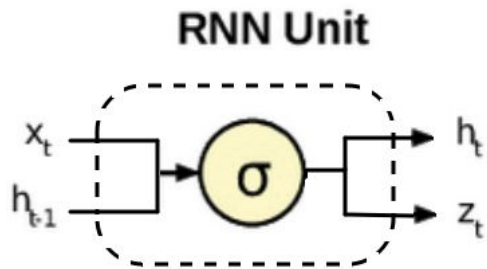
[API KERAS](#)



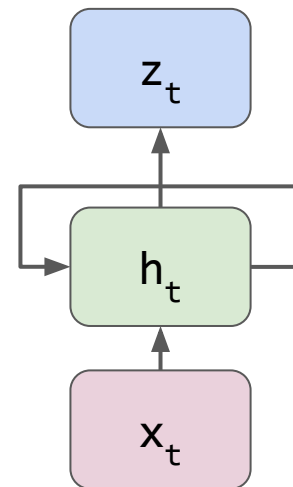
En Keras: SimpleRNN

$$h_t = \sigma(W_{hh} * h_{t-1} + W_{hx} * x + b_h)$$

$$z_t = h_t$$



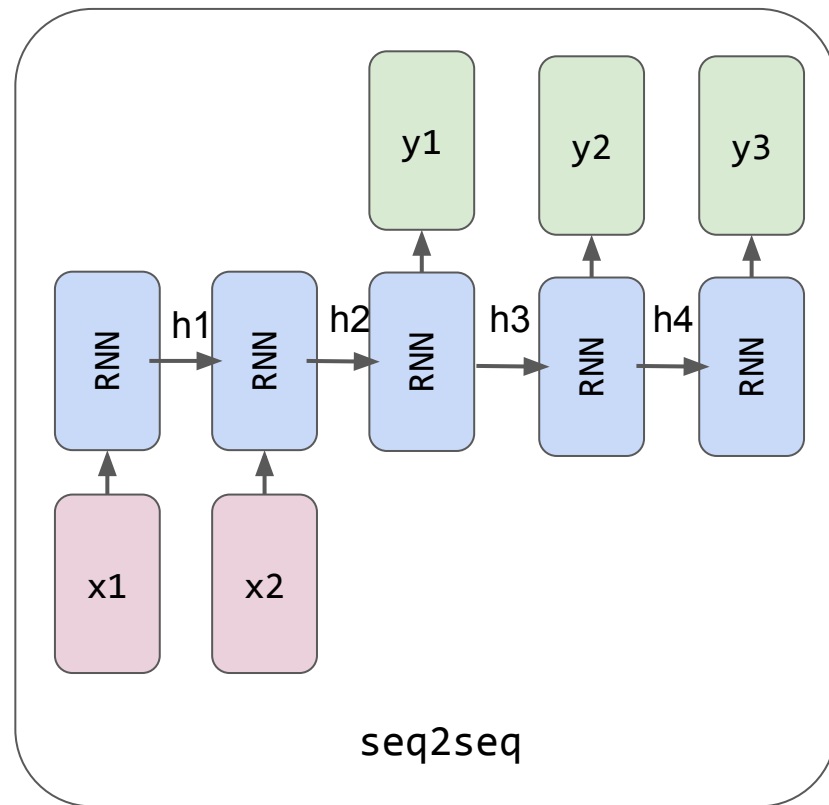
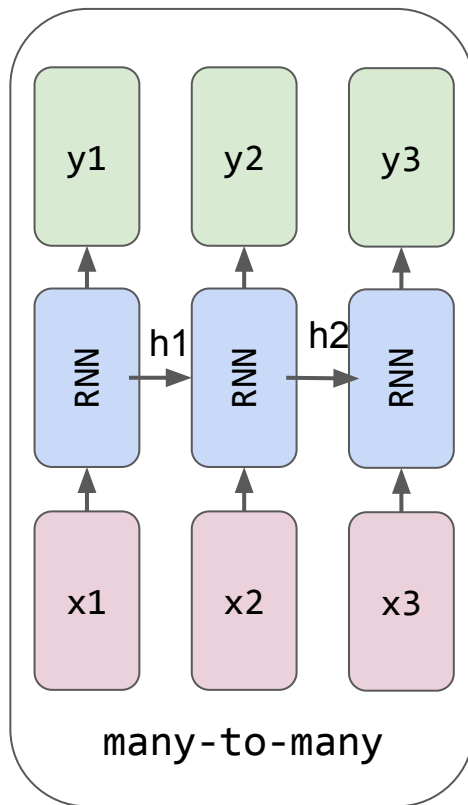
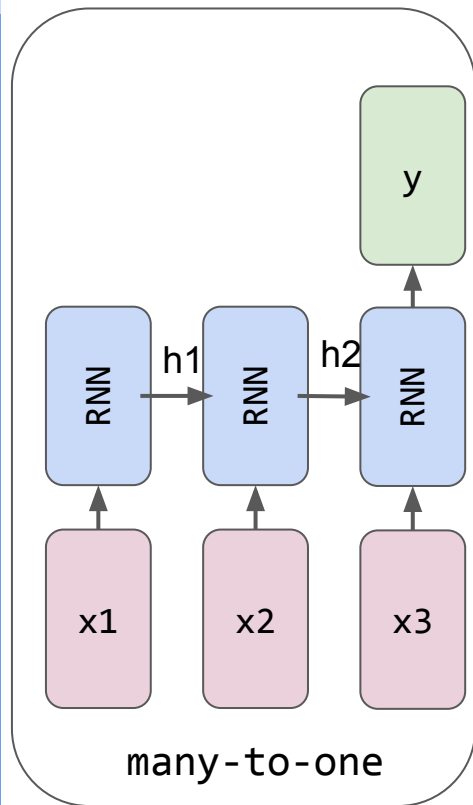
Unidad básica



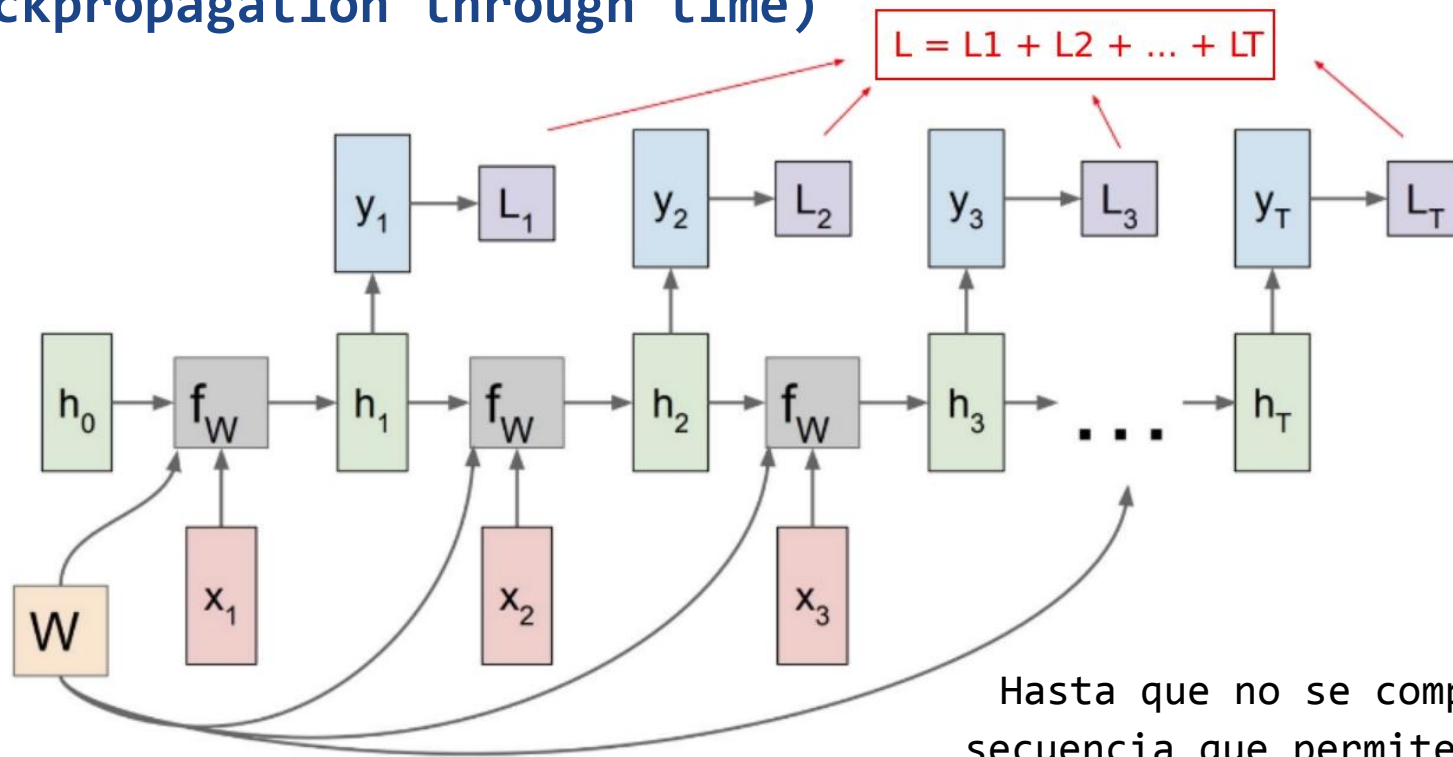
Representación
compacta

Tipos de problemas de secuencia

[LINK](#)



Grafo de cómputo de una RNN y BPTT (Backpropagation through time)

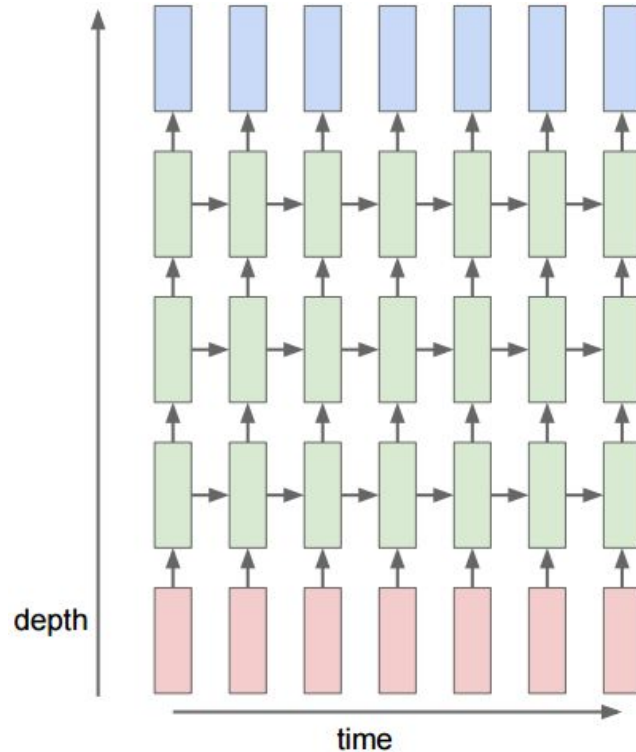


Hasta que no se complete la
secuencia que permite calcular
el loss no se actualizan los
pesos (W) de la/s celda/s

Multi-layer RNN



Tal como se vio en los ejemplos se trata de apilar layers RNN en donde la salida de una se traslada a la entrada de la siguiente



Problema de una RNN tradicional

[LINK](#)



"Una RNN tradicional solo usa información del pasado y no de las futuras palabras para predecir"

Ejemplo: Data una sentencia determinar si existe una entidad que represente al nombre de una persona utilizando (name entity recognition)

Ejemplo 1:

*"Hoy escuche que **Victoria** terminó su bot para NLP" → persona*

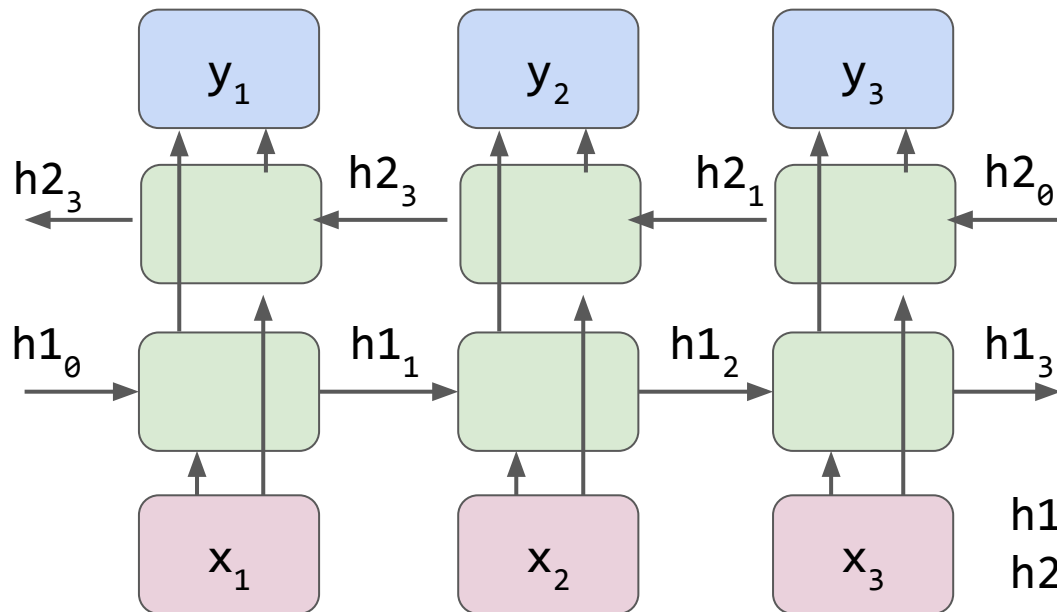
Ejemplo 2:

*"Hoy escuche que **Victoria** cambió de intendente" → ciudad*

La
contextualización
de la palabra es
a futuro



“La palabra anterior y la palabra futura tienen impacto en la presente predicción”



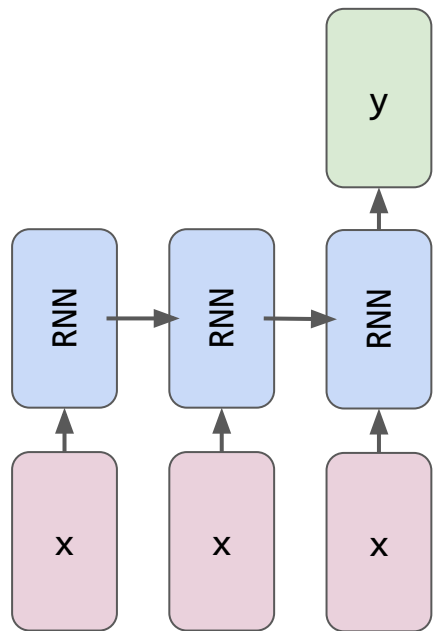
$$h1_t = \sigma(W_{hh1} * h1_{t-1} + W_{hx1} * x + b_1)$$
$$h2_t = \sigma(W_{hh2} * h2_{t+1} + W_{hx2} * x + b_2)$$

Las dos salidas pueden concatenarse, sumarse o promediarse

many-to-one



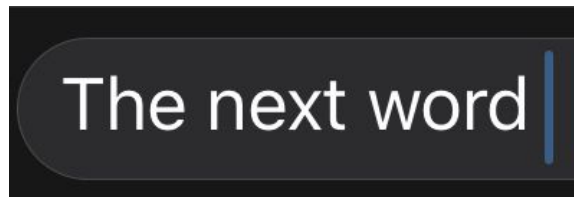
"Dada una sentencia o oración de entrada de tamaño fijo, el sistema arroja un único resultado que la representa".



many-to-one



Este tipo de estructuras se utilizan para determinar cuál es la siguiente palabra o elemento en la secuencia o para clasificación (sentiment analysis).



Predicción de
próxima palabra



Análisis de
sentimientos



Link al Colab



LINK

Long short term memory (LSTM)

[LINK](#)



Se introduce este tipo de celda neuronal con mayor persistencia de memoria para lograr capturar relaciones de palabras a largo plazo.



Se crearon en 1997. Se adoptó como la layer principal para problemas de secuencia en 2014 hasta la aparición de los transformers en 2017.



Desplazaron completamente a las capas RNN simples (Elman), ya que el costo adicional de las LSTM es marginal respecto al beneficio que otorgan

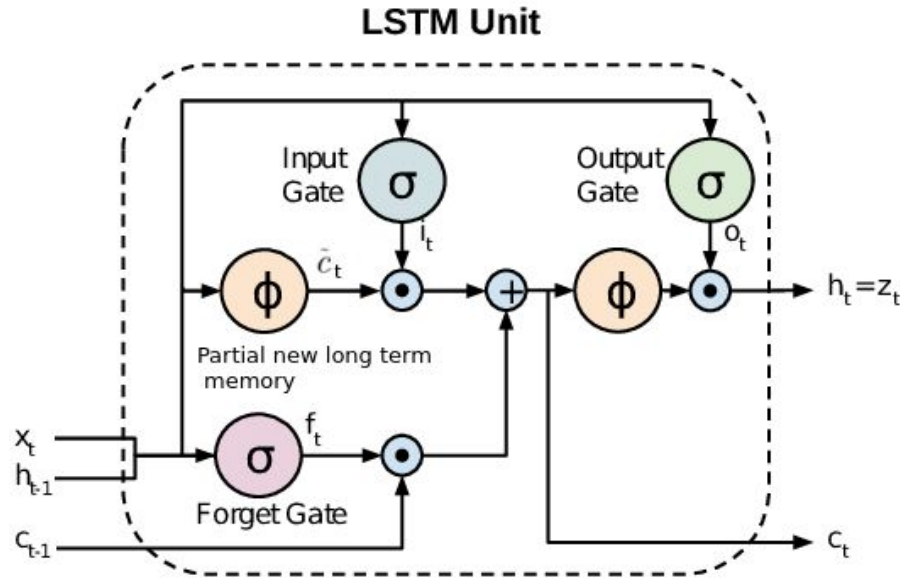


Se basan en el principio de ponderar la importancia de una palabra respecto al contexto futuro/pasado (key words).

¿Comprarías este producto?

*"**Incredible!** El producto es lo que venden, hace lo que tiene que hacer y me **ayudó mucho** a resolver los problemas que tenía. Lo **volvería a comprar** sin dudas"*

LSTM approach - Idea de la arquitectura



C: 'Memoria a largo plazo'

Input (i): Dado el último estado (h_{t-1}) evalúa cuánto de la nueva entrada (x) se incluirá en la memoria de largo plazo (C_t).

Forget (f): Dada la entrada (X) cuánto del estado de memoria anterior (C_{t-1}) tiene importancia en el nuevo estado.

Output (o): Qué parte del estado de memoria (C_t) pasa al próximo estado de memoria h_t .

\cdot : Producto elemento a elemento

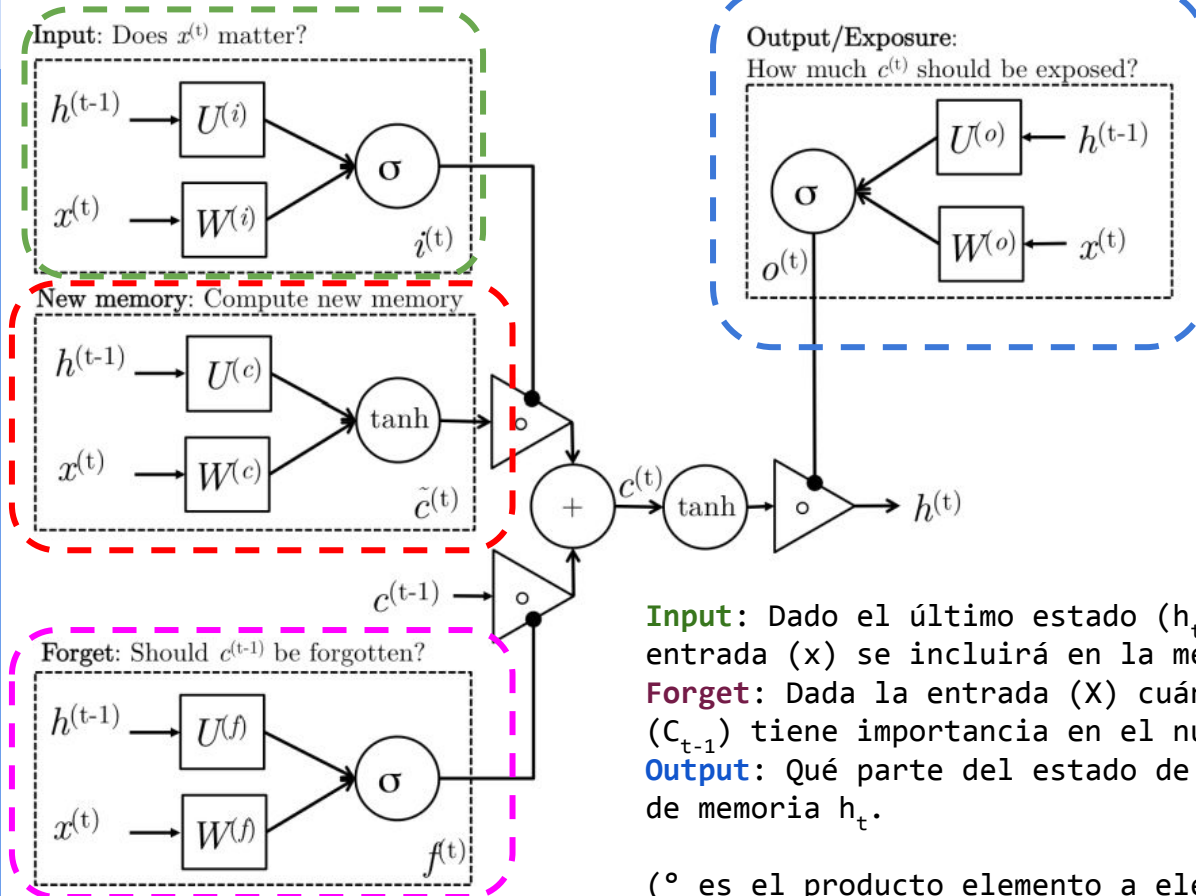
ϕ : Función Tanh

$+$: Suma vectorial

\rightarrow : Conexiones

LSTM approach

[LINK](#)



$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1})$$

$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1})$$

$$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1})$$

$$\tilde{c}_t = \tanh(W^{(c)}x_t + U^{(c)}h_{t-1})$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$

$$h_t = o_t \circ \tanh(c_t)$$

Input: Dado el último estado (h_{t-1}) evalúa cuánto de la nueva entrada (x) se incluirá en la memoria de largo plazo (C_t).

Forget: Dada la entrada (X) cuánto del estado de memoria anterior (C_{t-1}) tiene importancia en el nuevo estado.

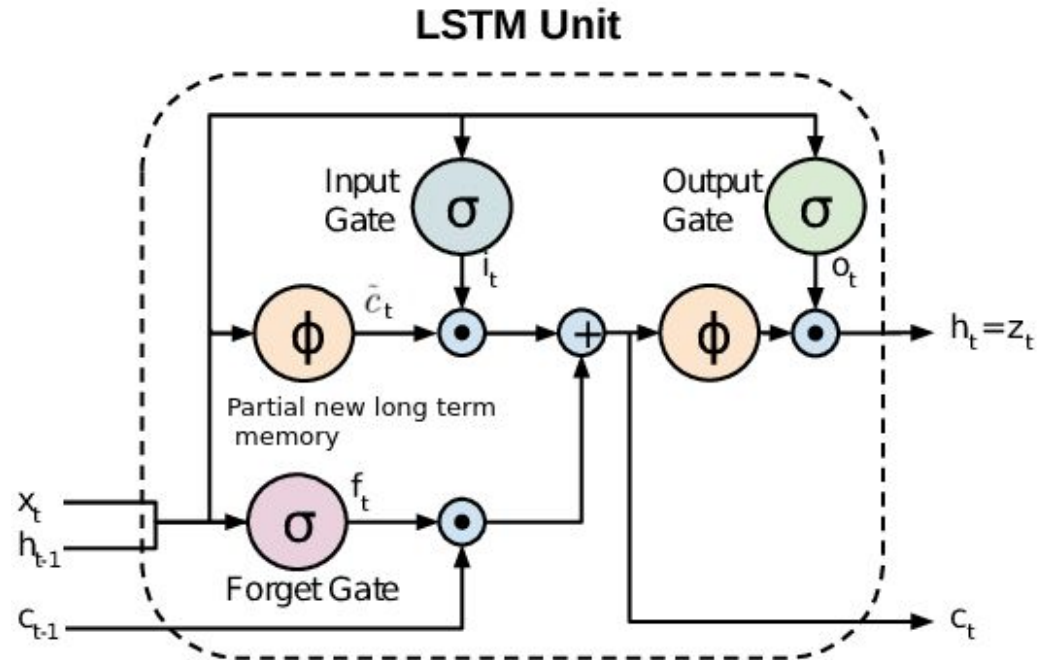
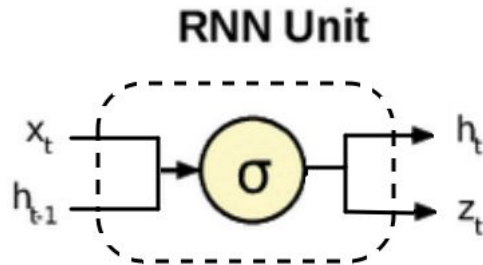
Output: Qué parte del estado de memoria (C_t) pasa al próximo estado de memoria h_t .

(\circ es el producto elemento a elemento)

LSTM vs RNN



La memoria de largo plazo c_t permite propagar gradiente eficientemente a mayor “profundidad temporal”.

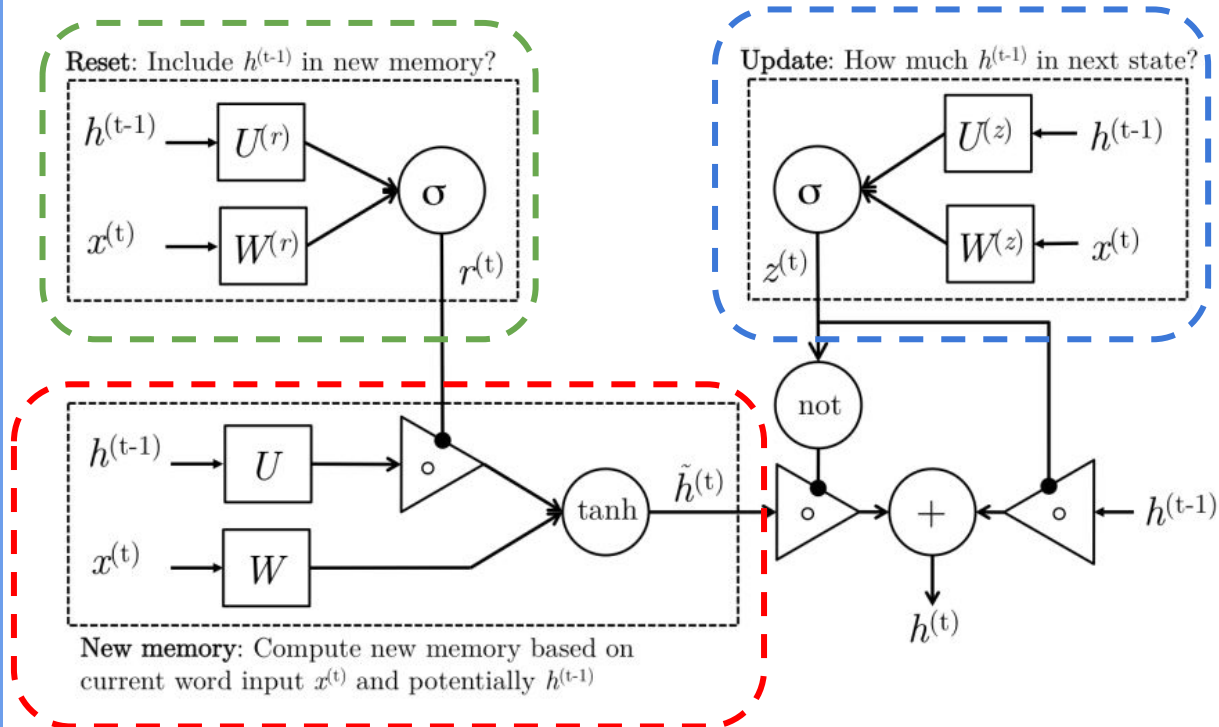


Gated Recurrent Units (GRU)

[LINK](#)



"Evolución de las RNN para superar problemas de "short-memory", versión reducida de una LSTM".



$$\begin{aligned} z_t &= \sigma(W^{(z)}x_t + U^{(z)}h_{t-1}) \\ r_t &= \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}) \\ \tilde{h}_t &= \tanh(r_t \circ U h_{t-1} + W x_t) \\ h_t &= (1 - z_t) \circ \tilde{h}_t + z_t \circ h_{t-1} \end{aligned}$$

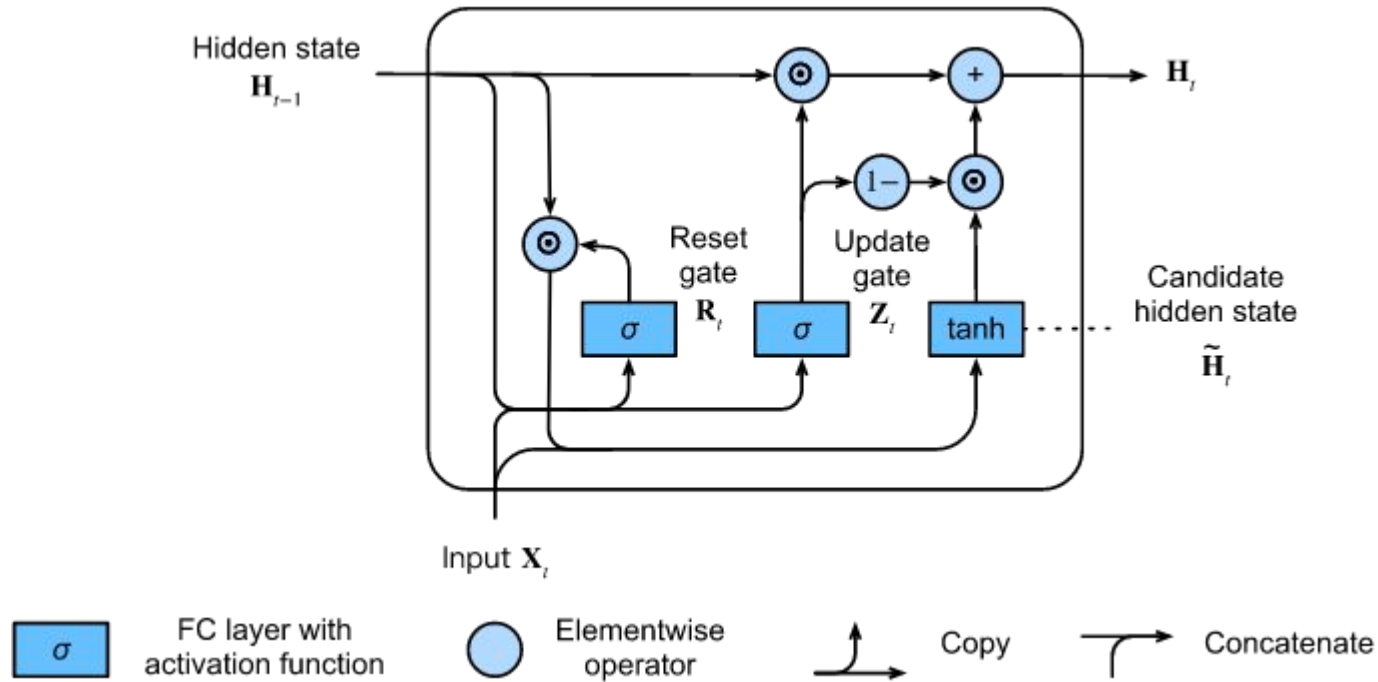
(Update gate)

(Reset gate)

(New memory)

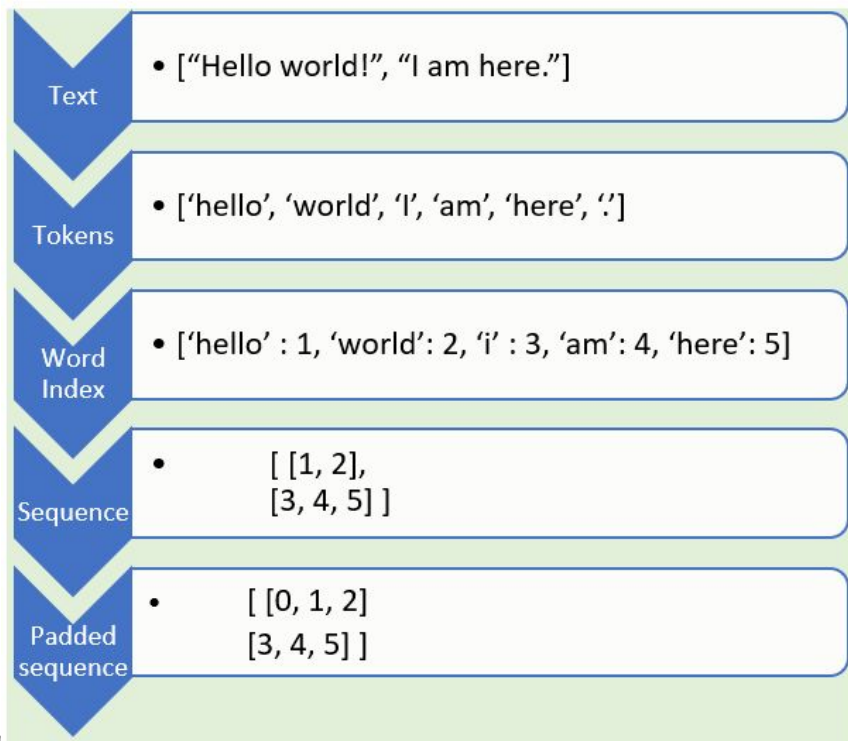
(Hidden state)

Gated Recurrent Units (GRU)





Hoy trabajaremos con sentiment analysis el cual responde a la estructura many-to-one (text_sequence to class/label)



Es necesario garantizar que la longitud de la secuencia de entrada siempre será del mismo largo, para eso se agregan ceros al comienzo o final de las cadenas de texto más cortas (padding)

```
padded_seq =  
[[ 2  6  3  7  8]  
 [ 0  2  9  3  4] <----- 0 Padded at the beginning  
 [ 0  0  0  5 10]  
 [ 0  0  0  5  4]]
```



Link al Colab



LINK

Predicción de texto/modelos de lenguaje



Se utilizará many-to-one, por lo que hay que seleccionar la dimensión de la sentencia de entrada y dividir el texto en grupos:

The next word

Predicción de
próxima palabra

Sentencia

'Yesterday, all my troubles seemed so far away'

Tokens

['yesterday', 'all', 'my', 'troubles', 'seemed', 'so', 'far', 'away']

Vectores de entrada de 4 tokens

$$\prod_{i=1}^{i=m} P(w_i | w_{i-(n-1)}, \dots, w_{i-1})$$

```
[['yesterday', 'all', 'my', 'troubles'],  
 ['all', 'my', 'troubles', 'seemed'],  
 ['my', 'troubles', 'seemed', 'so'],  
 ['troubles', 'seemed', 'so', 'far']]
```

Probabilidad de una secuencia



$$\begin{aligned} P_{LM} (x^{T+1}, x^T, \dots, x^1) &= P_{LM} (x^{T+1} | x^T, \dots, x^1) P_{LM} (x^T, x^{T-1}, \dots, x^1) \\ &= P_{LM} (x^{T+1} | x^T, \dots, x^1) P_{LM} (x^T | x^{T-1}, \dots, x^1) P_{LM} (x^{T-1}, \dots, x^1) \\ &= \prod_{t=1}^T P_{LM} (x^{t+1} | x^t, \dots, x^1) \end{aligned}$$

Midiendo desempeño en modelos de lenguaje: perplexity



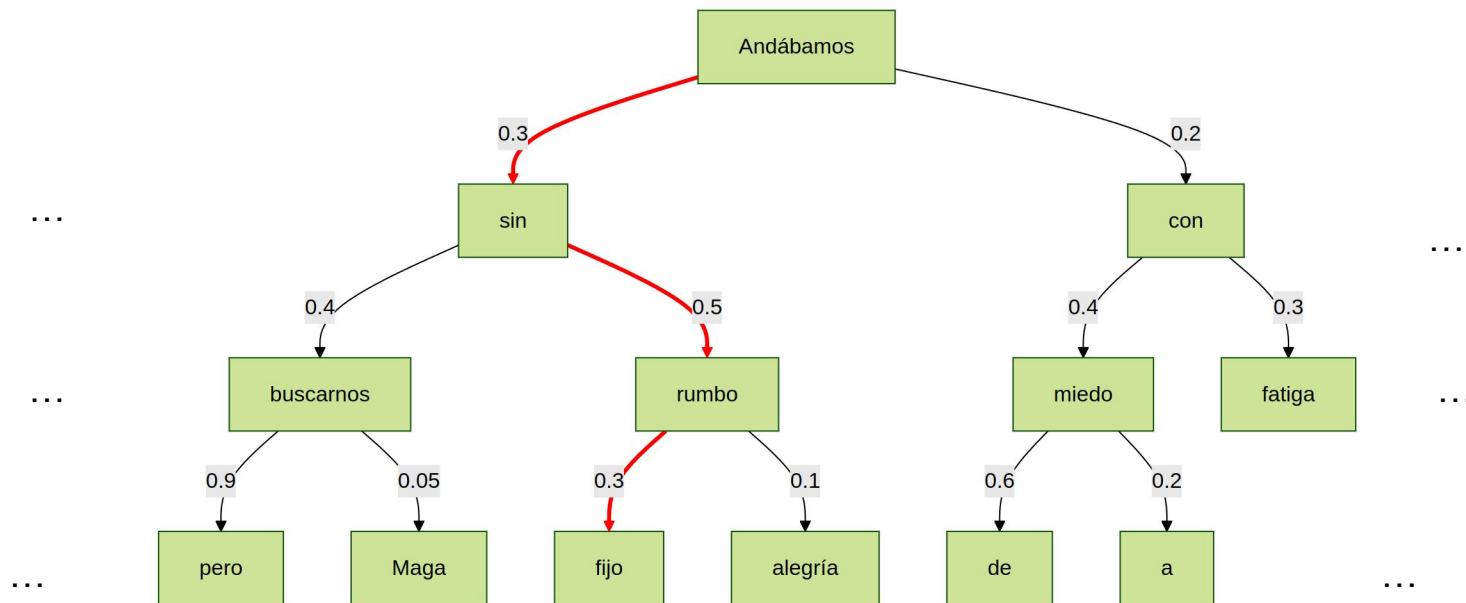
$$\text{perplexity} = \underbrace{\prod_{t=1}^T \left(\frac{1}{P_{\text{LM}}(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(1)})} \right)}_{\text{Inverse probability of corpus, according to Language Model}}^{1/T} \quad \leftarrow \text{Normalized by number of words}$$

Por cuestiones de estabilidad numérica conviene operar sobre los logaritmos de las probabilidades.

$$\text{PPL}(X) = \exp \left\{ -\frac{1}{t} \sum_i^t \log p_{\theta}(x_i | x_{<i}) \right\}$$

Modelo equiprobable: perplejidad V

Generación de texto en modelos de lenguaje: Greedy search

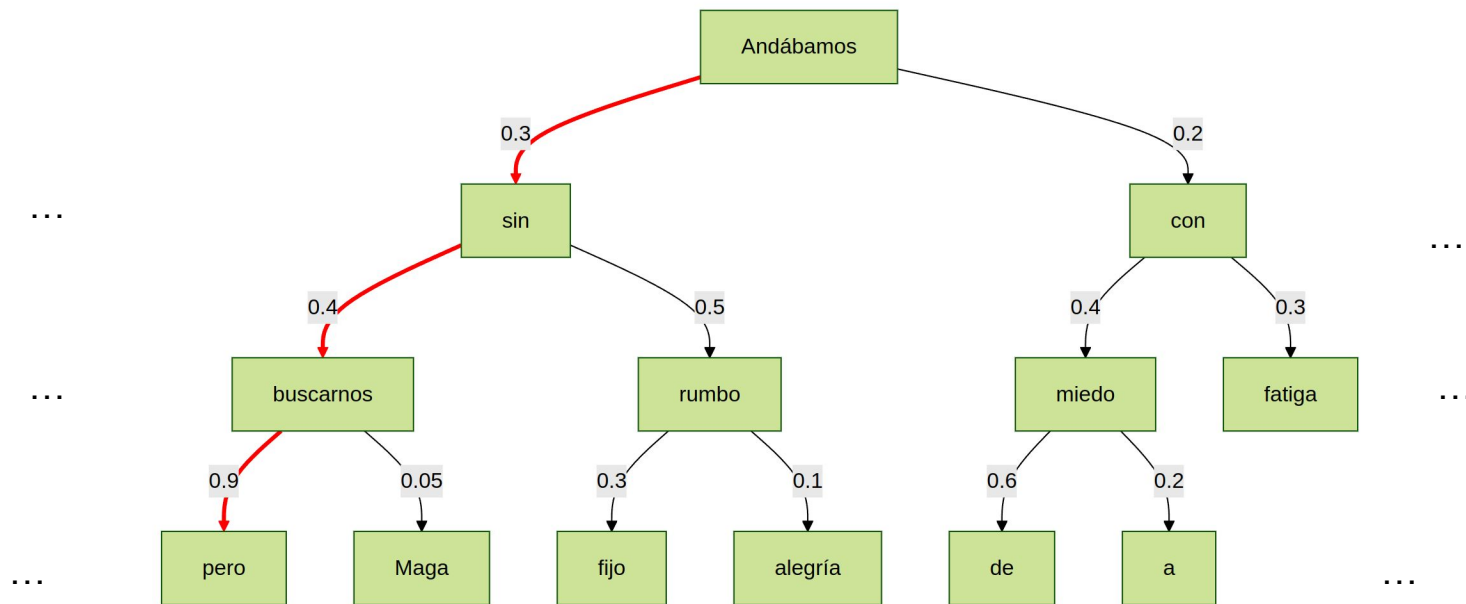


$$\prod_{t=1}^T P_{LM}(x^{t+1}|x^t, \dots, x^1)$$

$P(\text{"Andábamos sin rumbo fijo"}) =$

$$0.3 \times 0.045 \times 0.3 = \mathbf{0.045}$$

Generación de texto en modelos de lenguaje: Beam search



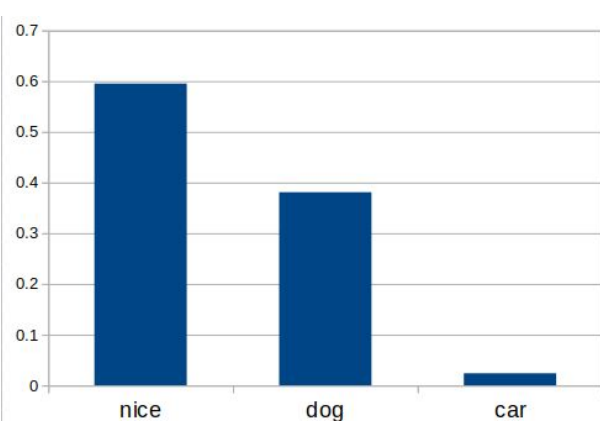
$$\prod_{t=1}^T P_{LM}(x^{t+1} | x^t, \dots, x^1) \longrightarrow$$

$$P(\text{"Andábamos sin buscarnos, pero"}) = \\ 0.3 \times 0.4 \times 0.9 = \mathbf{0.108}$$

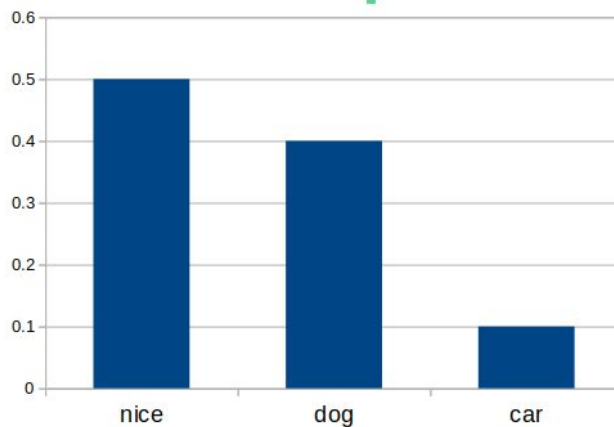
Generación de texto en modelos de lenguaje: Muestreo aleatorio con temperatura



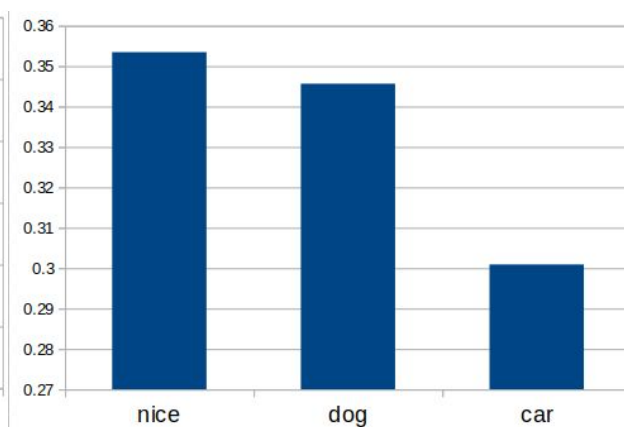
$$P_i = \frac{e^{\frac{y_i}{T}}}{\sum_{k=1}^n e^{\frac{y_k}{T}}}$$



Temperatura = 0.5



Temperatura = 1



Temperatura = 10



Utilizar otro dataset y
poner en práctica
la generación de
secuencias con las
estrategias presentadas.

