TEC | Tecnológico de Costa Rica

Aprendizaje automático

Redes neuronales recurrentes (RNN) aplicadas a procesamiento de lenguaje natural (NLP)

Contenidos

- Introducción a NLP
- Introducción a RNN
- Arquitectura
- Grafo computacional
- El error o pérdida
- Ejemplos

Procesamiento de lenguaje natural (NLP)

El procesamiento del lenguaje natural (NLP) es el conjunto de métodos para hacer que el lenguaje humano sea accesible a las computadoras. (Eisenstein, 2018)

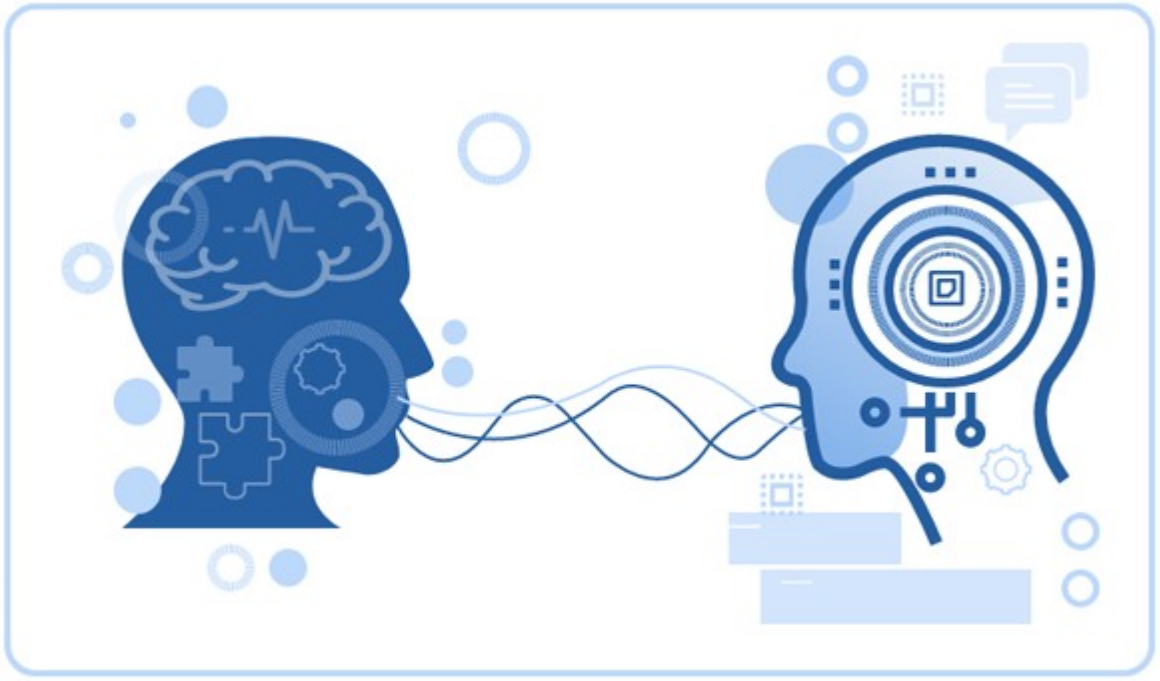
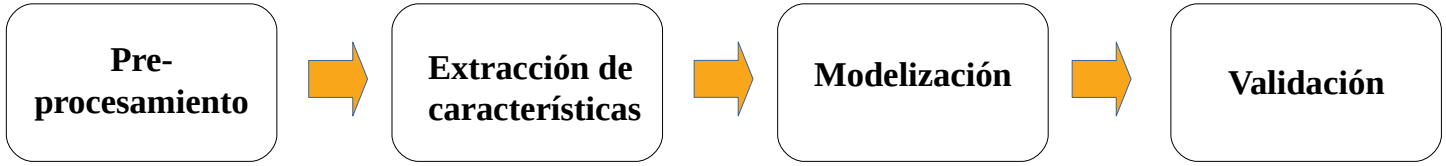


Imagen: SaaSworthy

Etapas básicas de un sistema de aprendizaje automático



Preprocesamiento

- Antes de realizar cualquier tarea de procesamiento de lenguaje natural, el texto debe ser **normalizado**.
- **Normalizar un texto consiste en transformarlo a una forma más conveniente**
- **Consiste de varias etapas como**
  - Separar el texto en oraciones
  - Separar en tokens
  - Lematizar
  - Stop words o palabras vacías
  - Otras

Extracción de características

Vectorización

- N-gramas
- Bolsas de palabras
- TF-IDF
- Word embedding (incrustaciones de palabras)

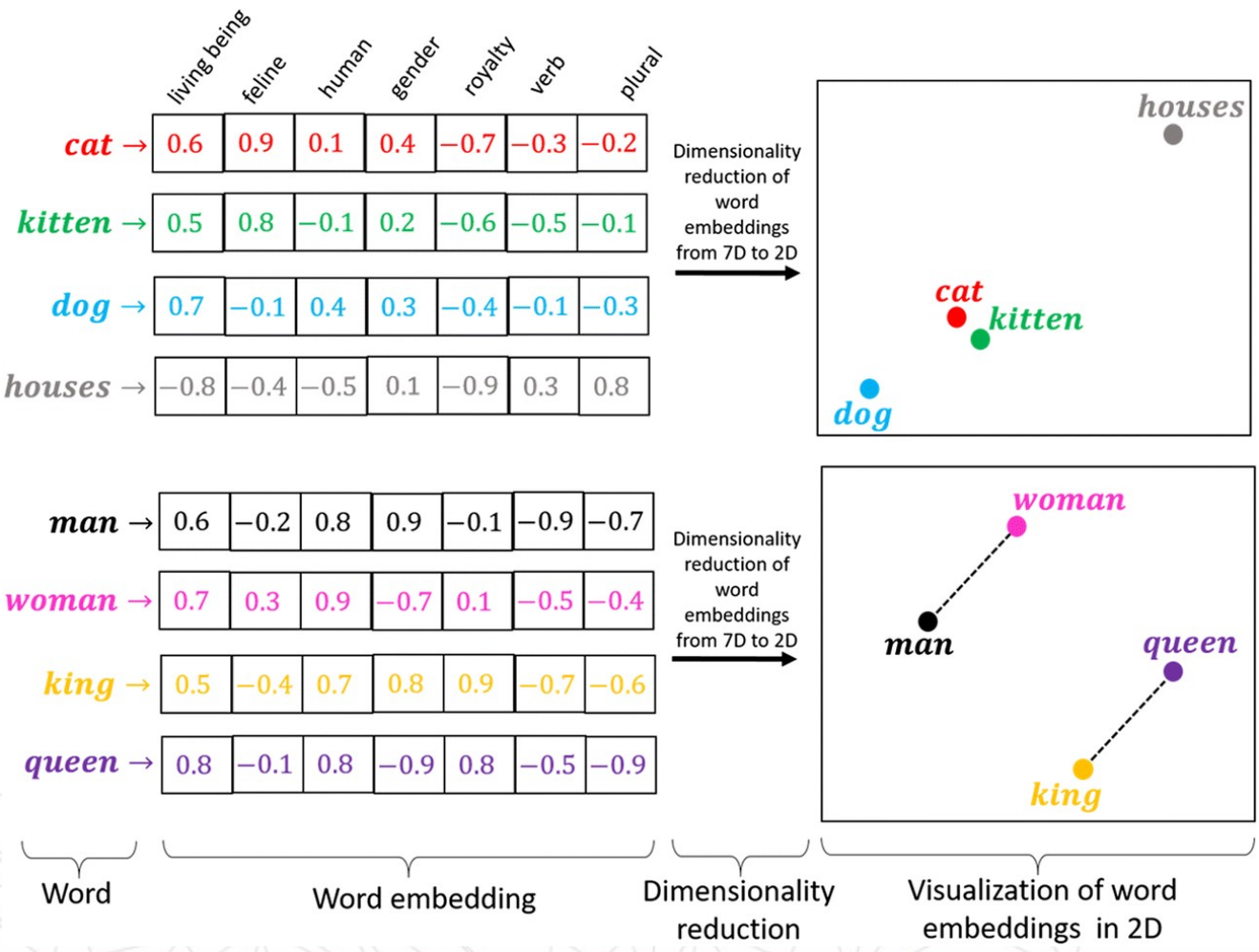


Imagen: (Siddharth, 2021)

Ejemplo de aplicación en NLP
Clasificación de textos

Clasificación de textos

- Una de las tareas más básica y parte del núcleo del NLP.
- Consiste en **asignar una etiqueta o categoría** a un texto o documento completo.

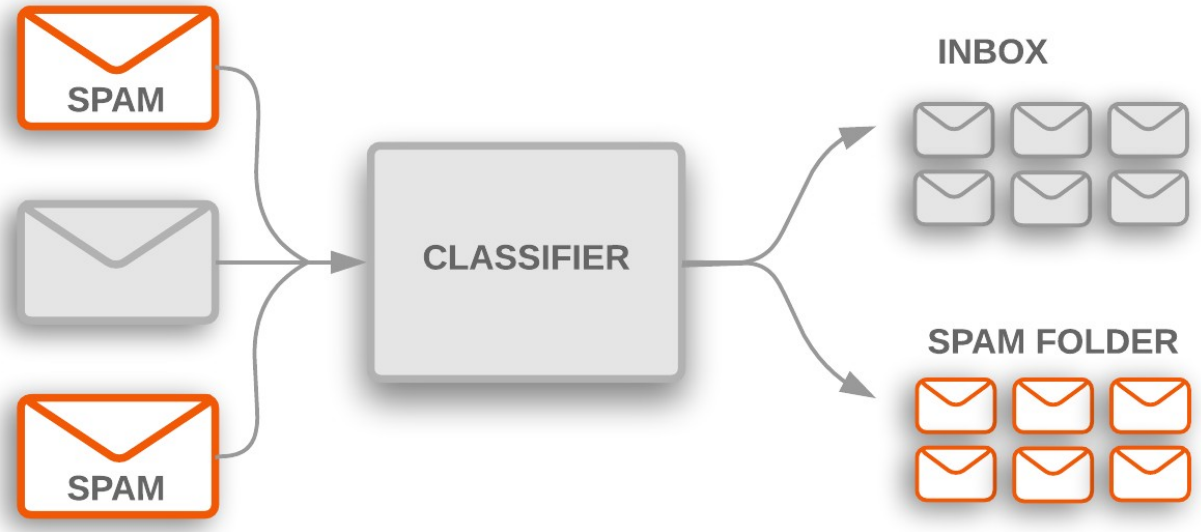


Imagen: Google Developers

Clasificación de textos

Ejemplo: Análisis de sentimientos:

- Extracción de la **orientación positiva o negativa** que un escritor expresa hacia algún objeto.
- Muy relevante para campos como: el **mercadeo, la investigación, la seguridad informática, la política**.
- Ejemplos:
  - Clasificación de reseñas de películas, libros o productos que expresan el sentimiento del autor hacia el producto.
  - Clasificación de un texto editorial o político que expresa el sentimiento hacia un candidato o una acción política.
  - Clasificación de un tweet.

Clasificación de textos

Análisis de sentimientos: comercio electrónico

-
-

Fuente: Amazon

Clasificación de textos

Otros ejemplos

- Detección del idioma — ES, EN, etc
- Tópico de un texto — Química, Biología, Medicina, etc
- Detección de correo spam — Spam, No-Spam
- Urgencia de los tickets de soporte — Nivel de urgencia de 1-5.

Clasificación de textos

Definición:

d

Clasificador

Predicción
C_i
(en multi-etiqueta, el resultado puede ser más de una clase.)

Conjunto finito de clases:
$C = \{ c_1, c_2, \ldots c_n \}$

Modelización

Redes recurrentes (RNN)

- Las redes recurrentes (Rumelhart et al., 1986) son una familia de redes neuronales diseñadas para procesar datos secuenciales.
- Proponen el procesamiento de secuencias con un largo τ arbitrario, implementando el concepto de parámetros compartidos.
  - Imagine el problema de reconocer la vocalización de la letra “a” en una grabación. Un modelo debe reconocer tal grabación como la pronunciación de la letra “a”, sin importar cuánto tardó la vocalización y la posición de la misma.
  - Imagine el problema de traducir un texto, cada palabra tiene un contexto que se pierde si se procesa individualmente.

Redes neuronales

Una red neuronal básica toma un vector de tamaño fijo como entrada, lo que limita su uso en situaciones que involucran una entrada de tipo "serie" sin tamaño predeterminado.

Perceptrón multicapa

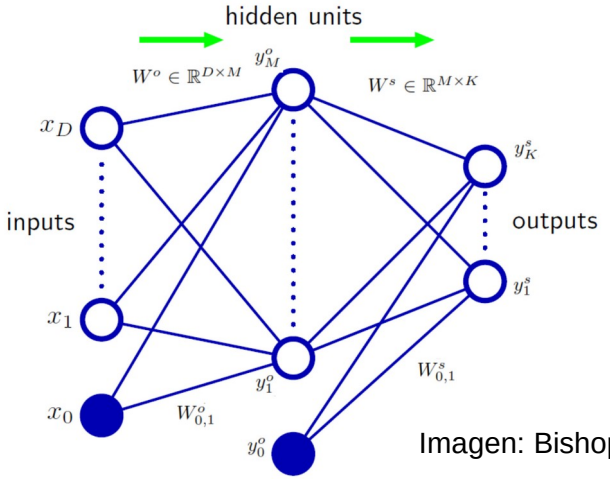


Imagen: Bishop, 2006

ConvNet Alex Net

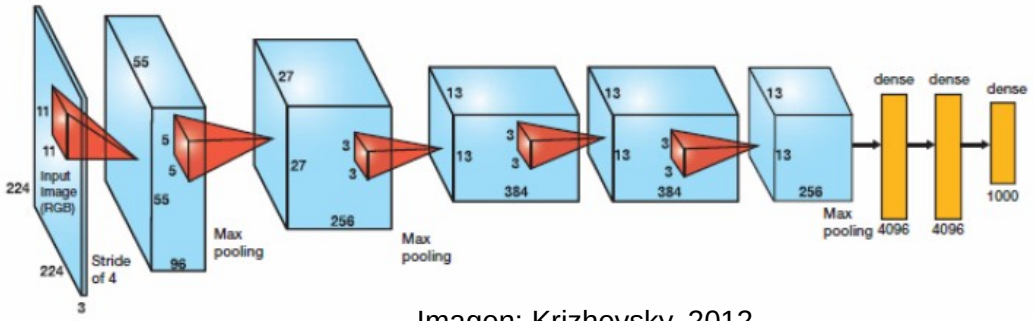
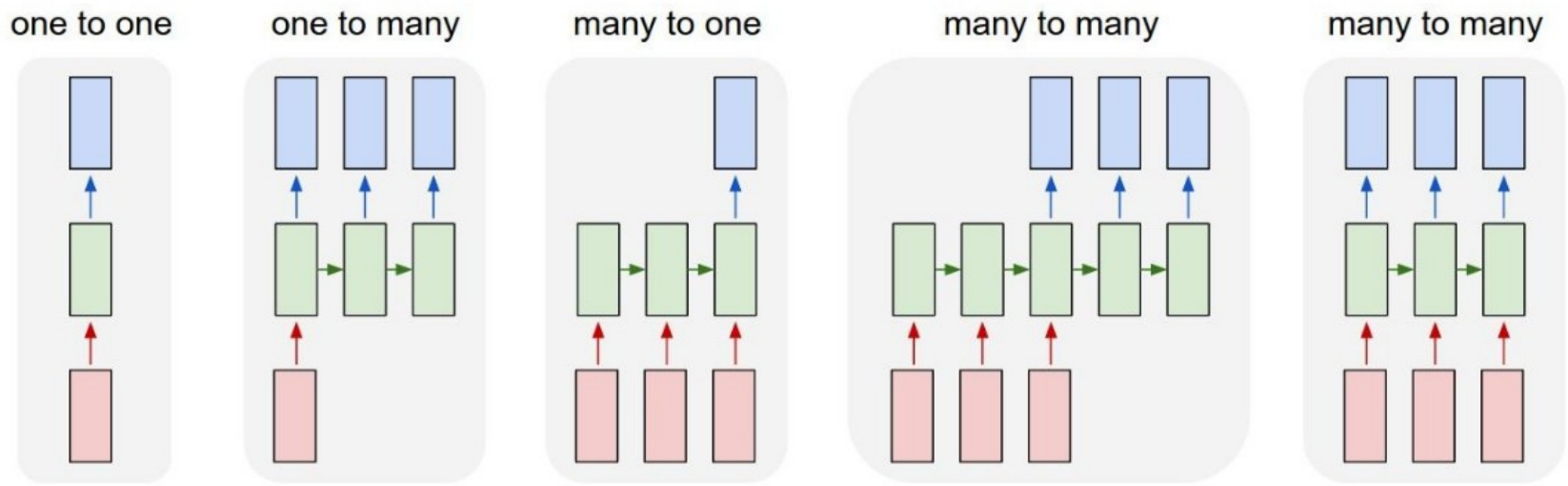


Imagen: Krizhevsky, 2012

Las redes recurrentes procesan secuencias



Ej. Subtítulos en imágenes

Ej. clasificación de textos.

Ej. Traducción de textos

Ej. Clasificación de vídeos

Imagen: (Li, Johnson & Yeung, 2017)

Redes recurrentes

Usualmente se necesita predecir sobre una entrada en un momento en el tiempo o parte de una serie de datos

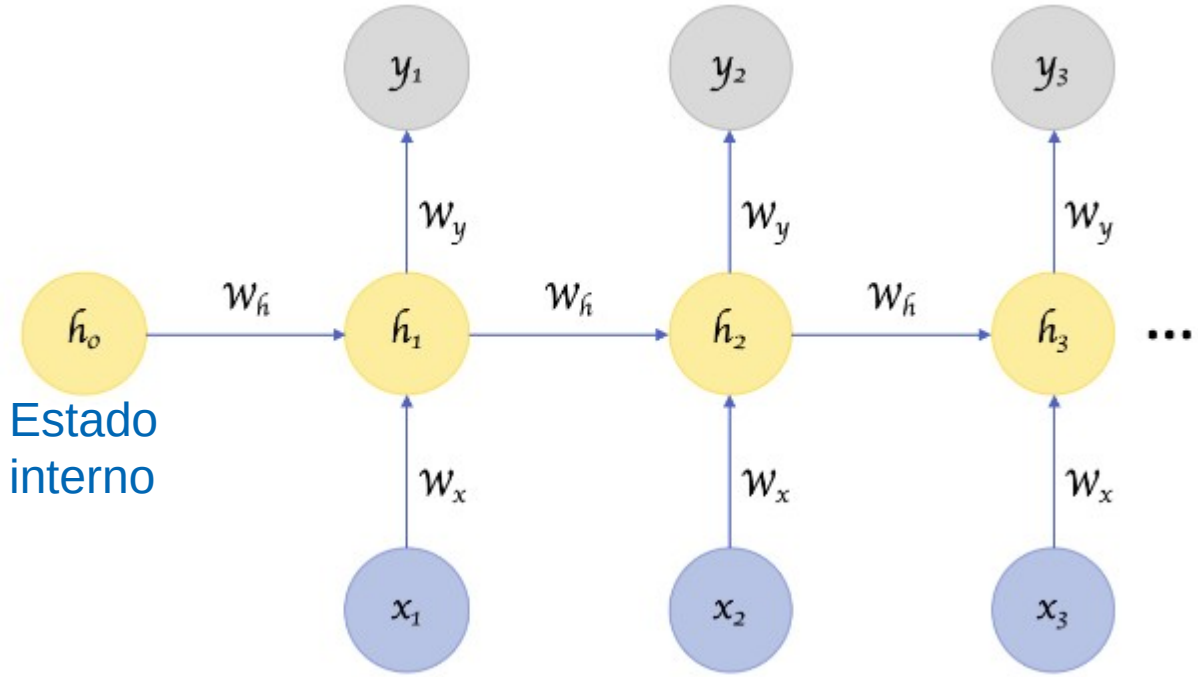
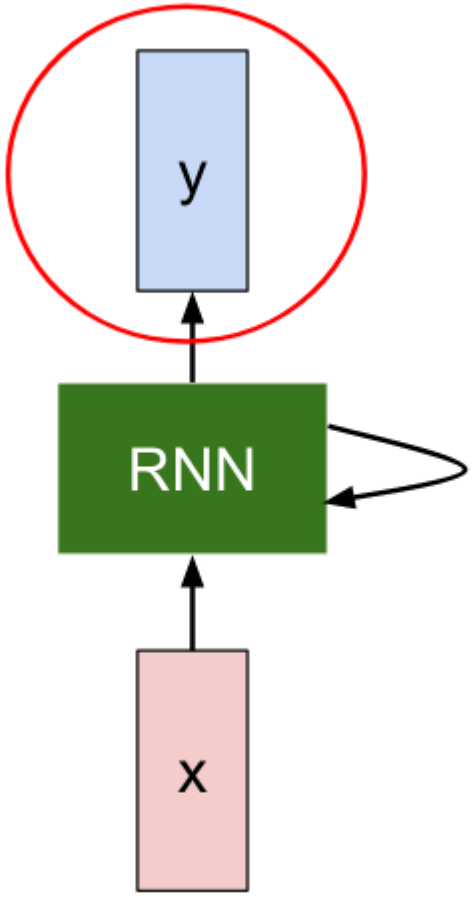


Imagen: https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

Redes recurrentes

Se utiliza una secuencia para **representar el estado de un sistema en distintos momentos**, mediante los arreglos h_t, h_2, . . . h_t, para modelar y estimar el comportamiento de un sistema dinámico:



$$h_t = f_W(h_{t-1}, x_t)$$

Nuevo estado

Función de activación

Estado anterior

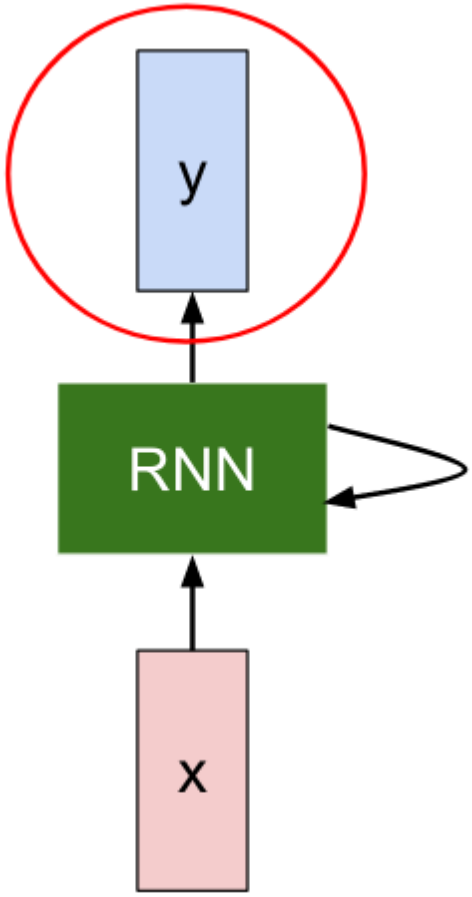
Entrada en el tiempo t

Fuente: Li, Johnson, Yeung, (2017)

Redes recurrentes

Se procesa una secuencia de vectores x aplicando una fórmula de recurrencia en cada paso de tiempo

$$h_t = f_W(h_{t-1}, x_t)$$

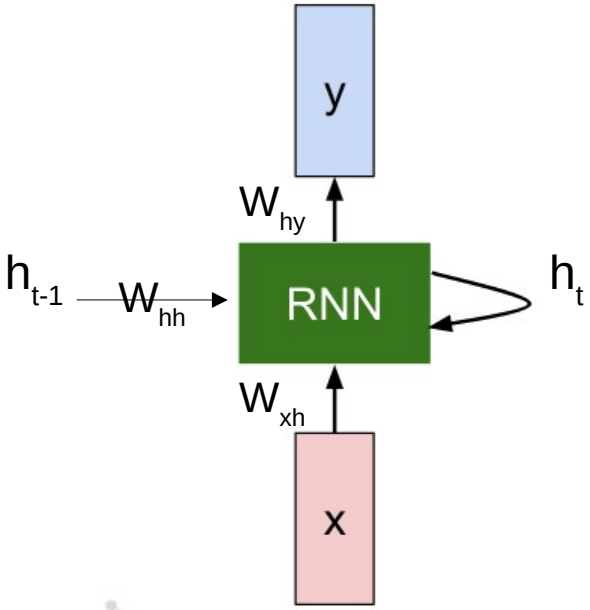


Se utiliza la misma función en cada paso de tiempo.

Fuente: Li, Johnson, Yeung, (2017)

Redes recurrentes

Ejemplo con función de activación tanh:



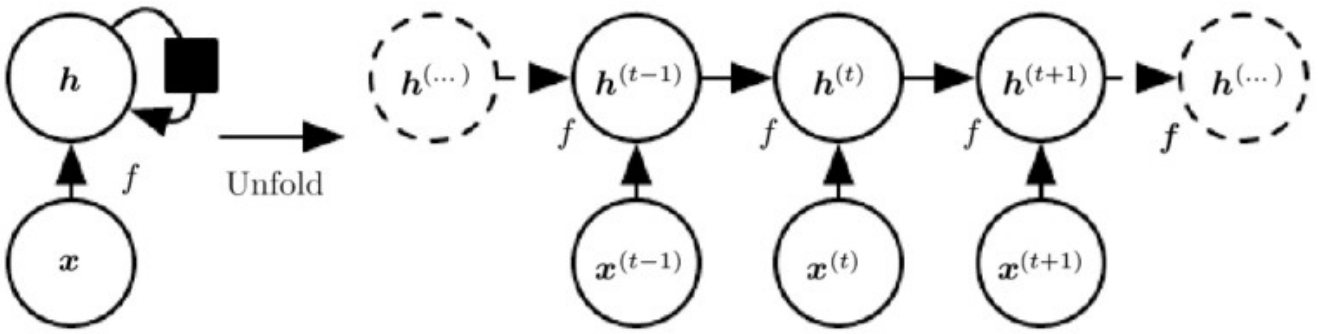
$$h_t = f_W(h_{t-1}, x_t)$$

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

$$y_t = W_{hy}h_t$$

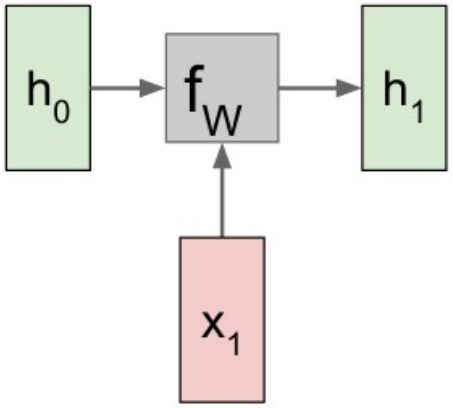
Fuente: Li, Johnson, Yeung, (2017)

Red recurrente sencilla



Fuente: (LeCun, Bengio, Hinton, 2015)

RNN: Grafo computacional

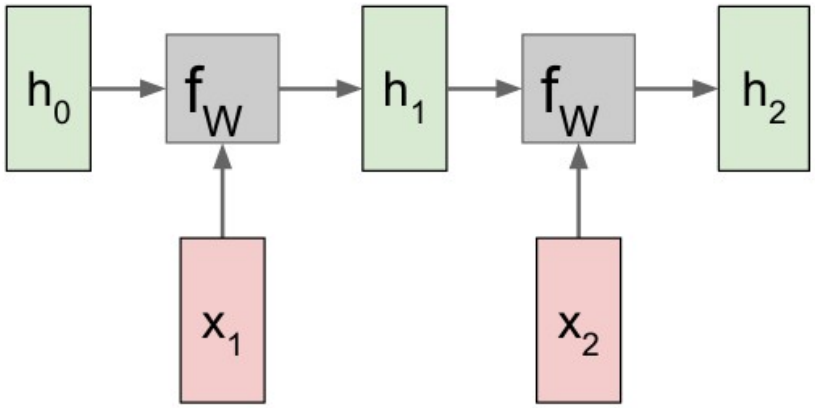


Ejemplo

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

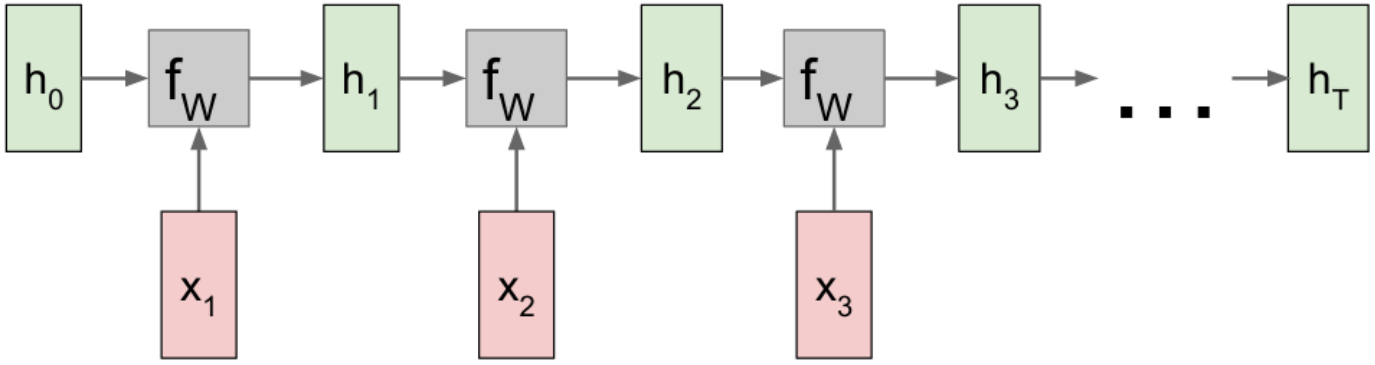
Fuente: Li, Johnson, Yeung, (2017)

RNN: Grafo computacional



Fuente: Li, Johnson, Yeung, (2017)

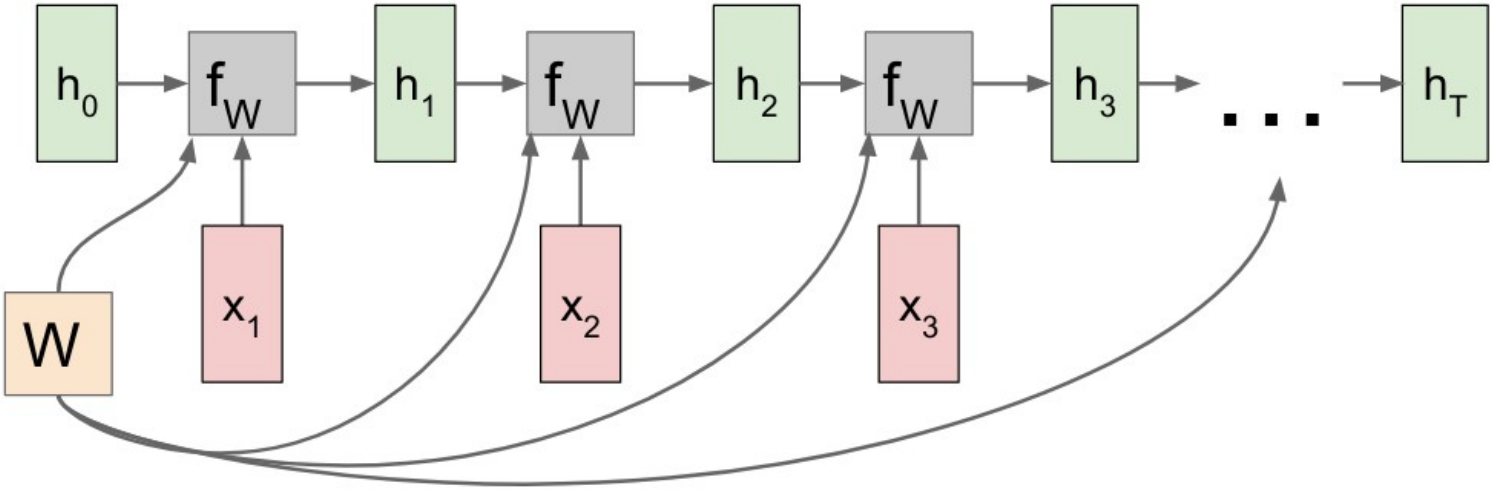
RNN: Grafo computacional



Fuente: Li, Johnson, Yeung, (2017)

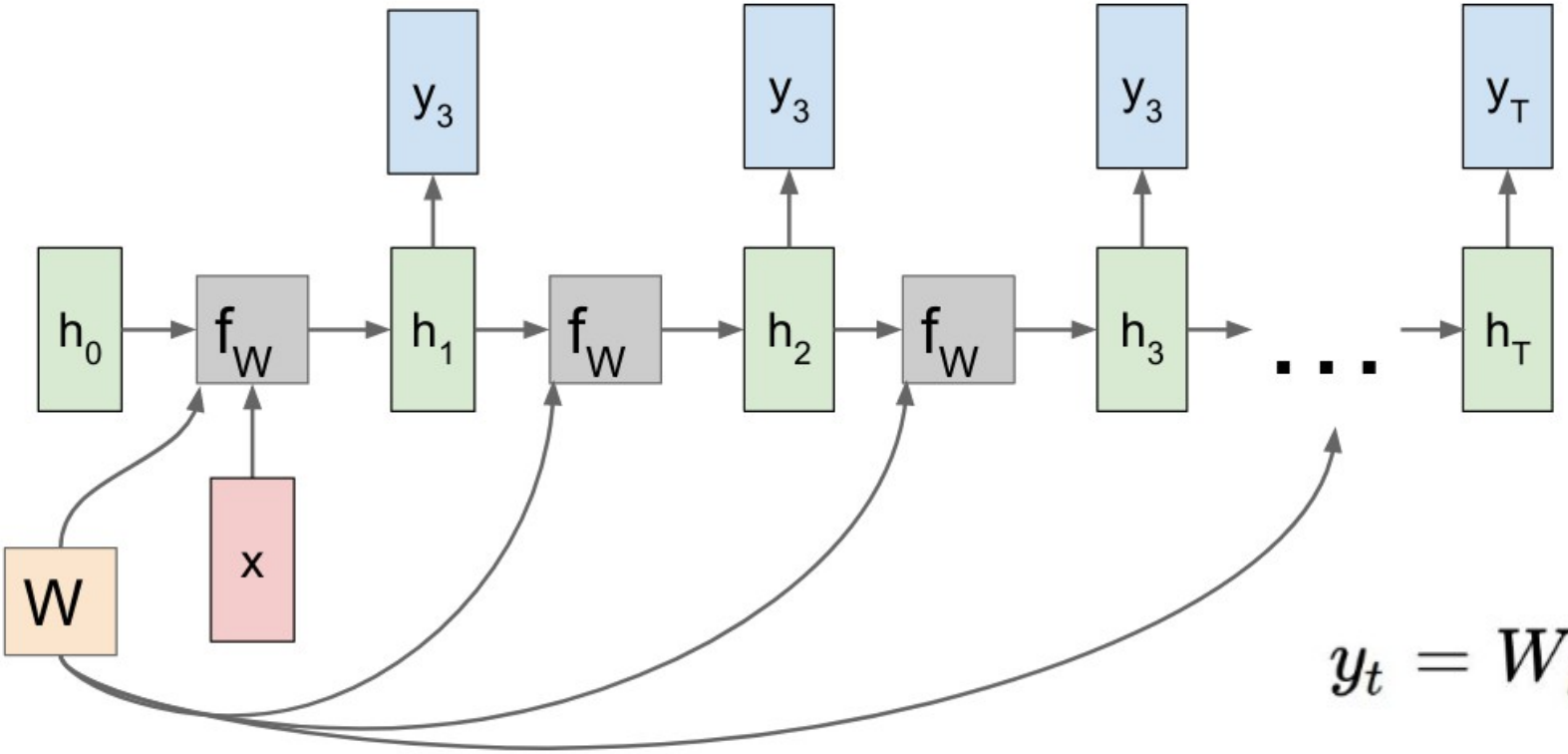
RNN: Grafo computacional

Se utiliza la misma matriz de pesos a través del tiempo



Fuente: Li, Johnson, Yeung, (2017)

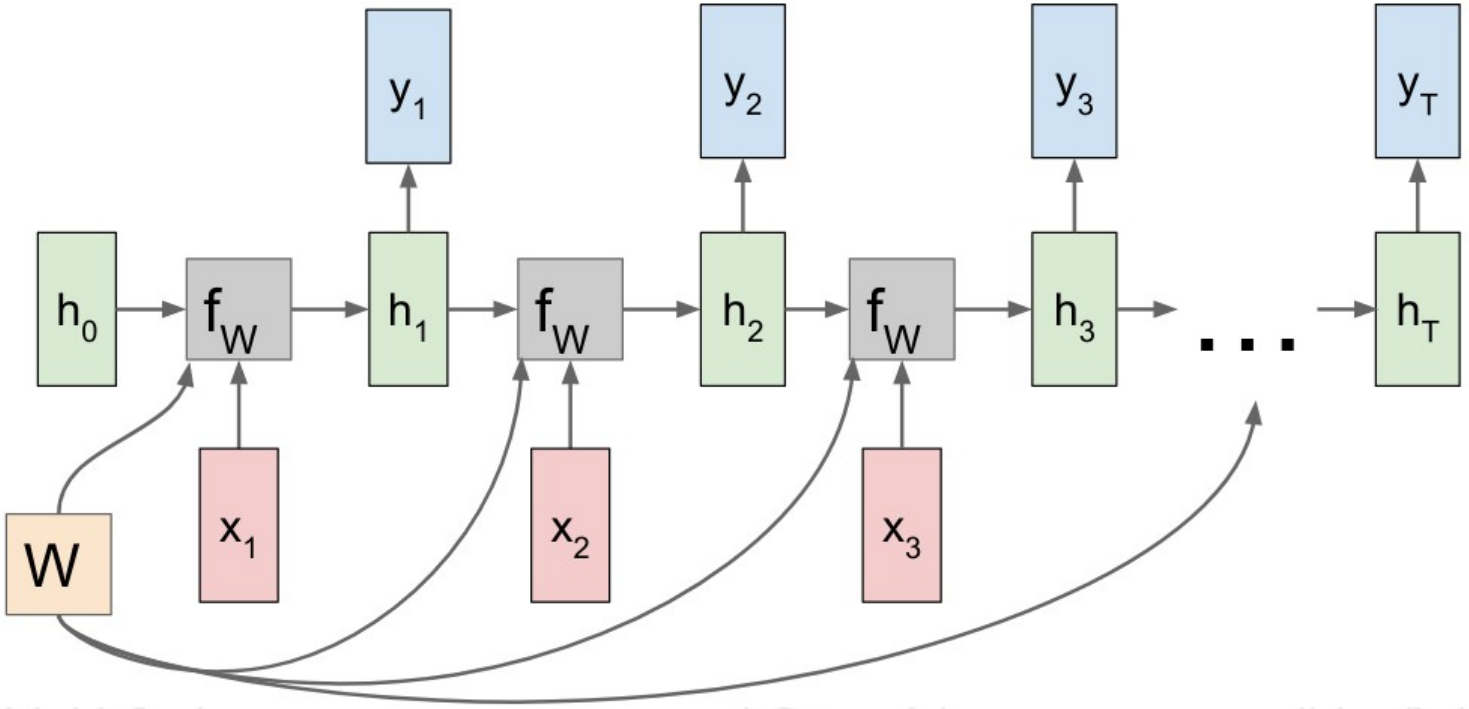
RNN: Grafo computacional. De uno a muchos



$$y_t = W_{hy} h_t$$

Fuente: Li, Johnson, Yeung, (2017)

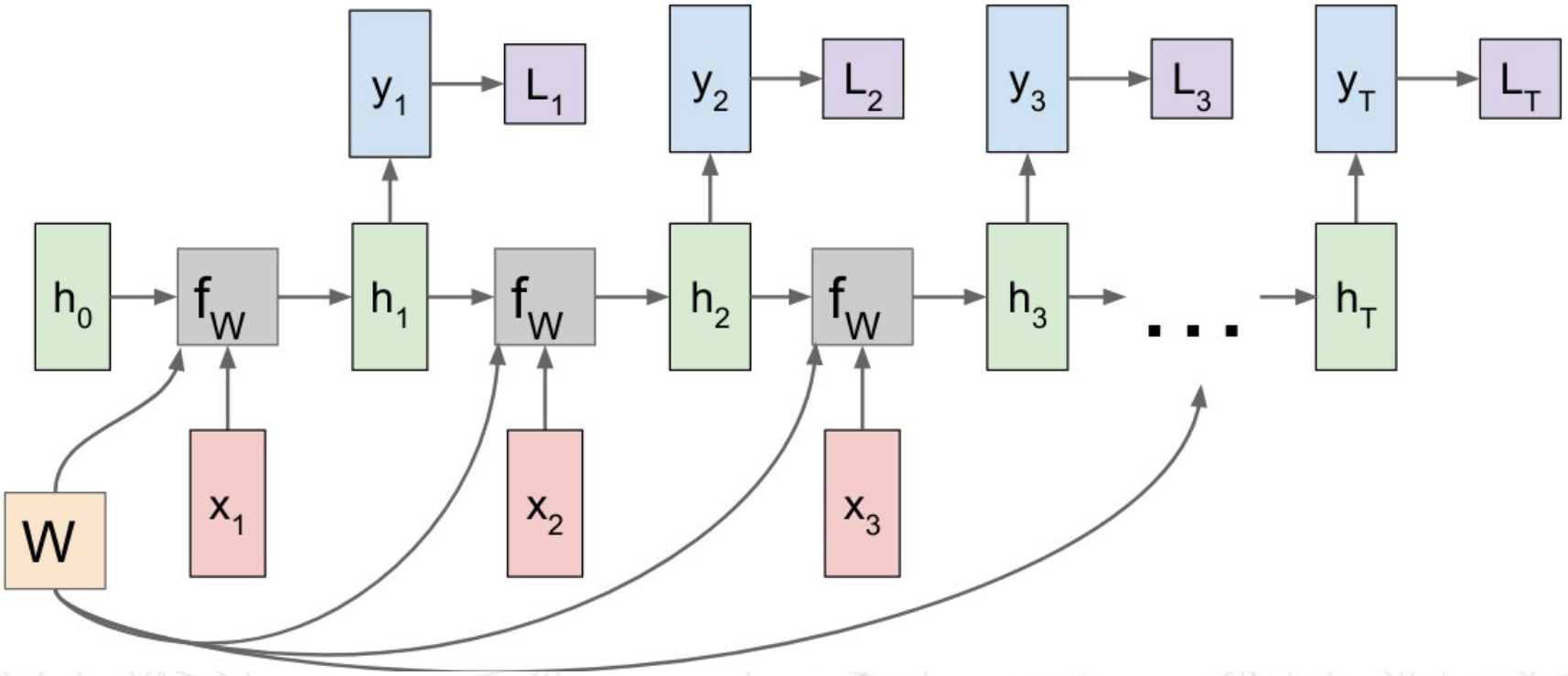
RNN: Grafo computacional. De muchos a muchos



Fuente: Li, Johnson, Yeung, (2017)

RNN: Grafo computacional. De muchos a muchos

Cálculo de la pérdida



Fuente: Li, Johnson, Yeung, (2017)

RNN: La función de pérdida

La función de pérdida mide qué tan lejos está la predicción, y en general viene dada por:

$$L\left(\overrightarrow{\hat{y}}, \overrightarrow{y}\right) = L\left(\left\{\overrightarrow{\hat{y}}^{(1)}, \overrightarrow{\hat{y}}^{(2)}, \ldots, \overrightarrow{\hat{y}}^{(\tau)}\right\}, \left\{\overrightarrow{y}^{(1)}, \overrightarrow{y}^{(2)}, \ldots, \overrightarrow{y}^{(\tau)}\right\}\right) = \sum_t L^{(t)}$$

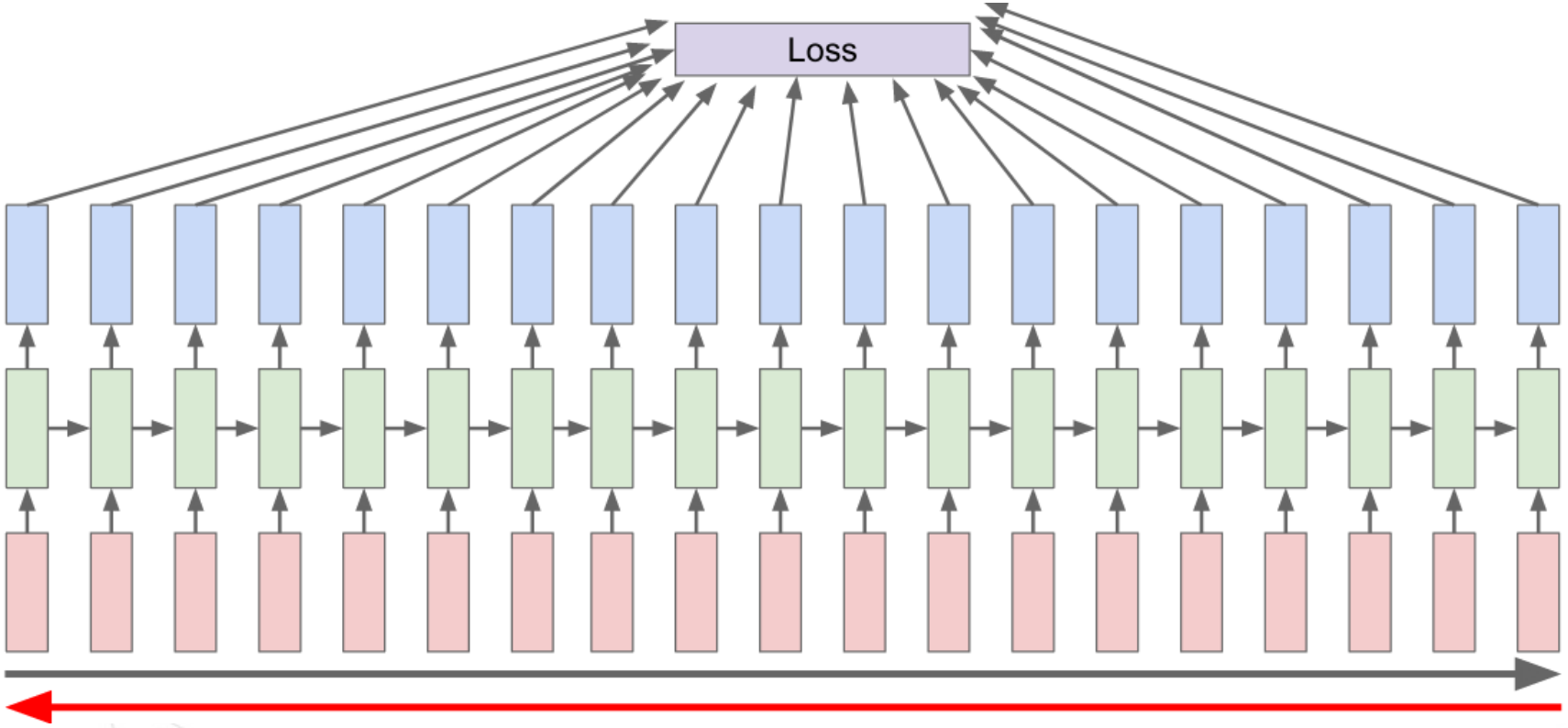
En caso de usar entropía cruzada como función de pérdida se tiene que:

$$L\left(\overrightarrow{\hat{y}}, \overrightarrow{y}\right) = \overrightarrow{y} \log\left(\overrightarrow{\hat{y}}\right) = -\sum \overrightarrow{y} \log\left(\overrightarrow{\hat{y}}\right)_t$$

Con: $\overrightarrow{\hat{y}}^{(t)}$ = valor estimado por el modelo

$\overrightarrow{y}^{(t)}$ = valor real o target

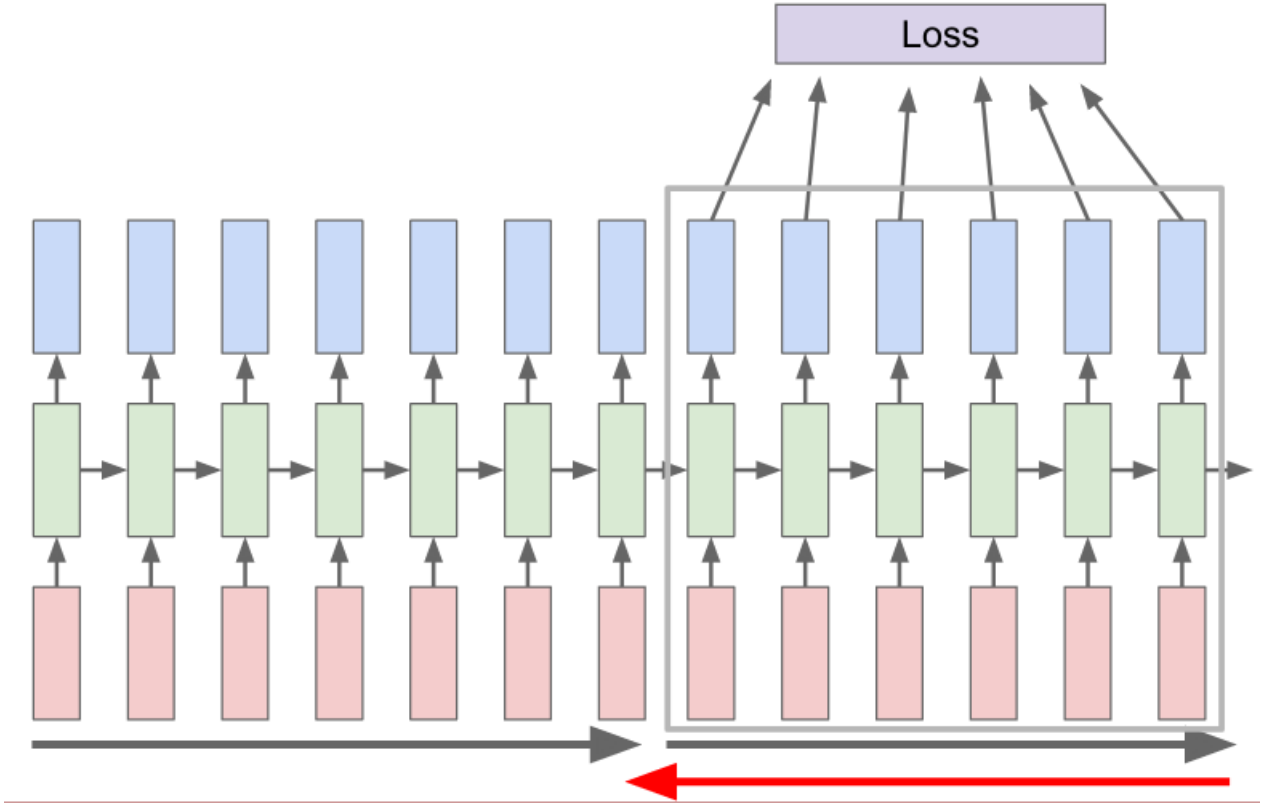
RNN: Retropropagación completa



Forward de toda la secuencia para calcular la pérdida, luego backward a través de toda la secuencia para calcular el gradiente y ajustar los pesos.

Fuente: (Li, Johnson & Yeung, 2017)

RNN: Retropropagación truncada



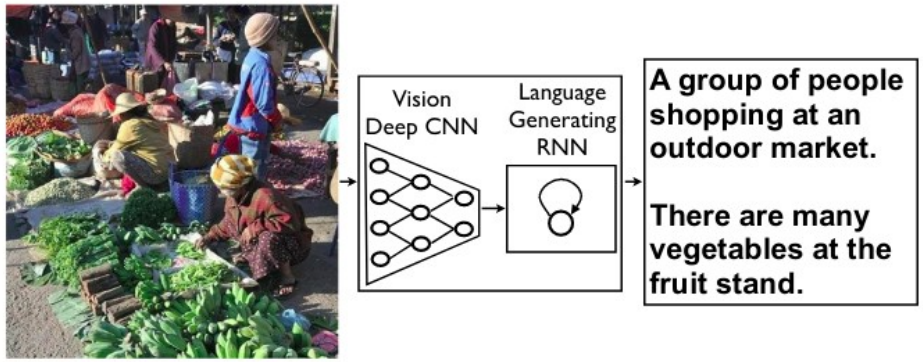
Solo se retropropaga un pequeño número de pasos.

Imagen: (Li, Johnson & Yeung, 2017)

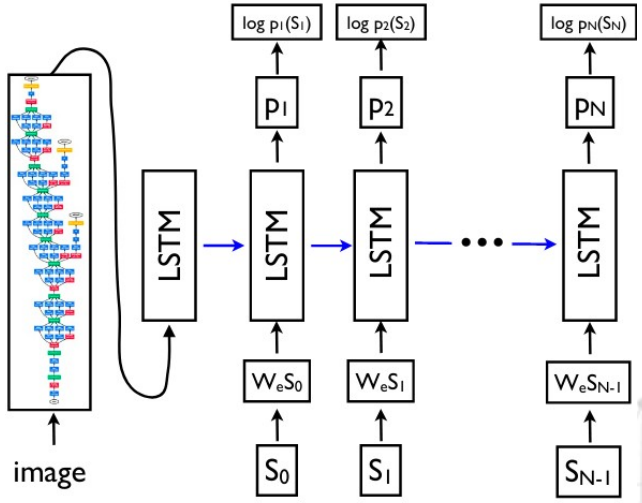
Long Short Term Memory (LSTM)

- Las RNN convencionales tienen el problema del desvanecimiento de gradientes y no son buenas para procesar secuencias largas porque sufren de memoria a corto plazo.
- Las redes LSTM son un tipo especial de RNN, capaces de aprender dependencias a largo plazo. Hochreiter & Schmidhuber (1997)
- Lo hacen manteniendo un estado de memoria interna llamado "*cell state*" y tienen reguladores llamados "*gates*" para controlar el flujo de información dentro de cada unidad LSTM.

Ejemplo de LSTM: generador de subtítulos de imágenes



Entrenamiento: El modelo LSTM está entrenado para predecir cada palabra de la oración después de haber visto la imagen y todas las palabras anteriores como se define por p (S_t | I, S_0, ..., S_{t-1}).



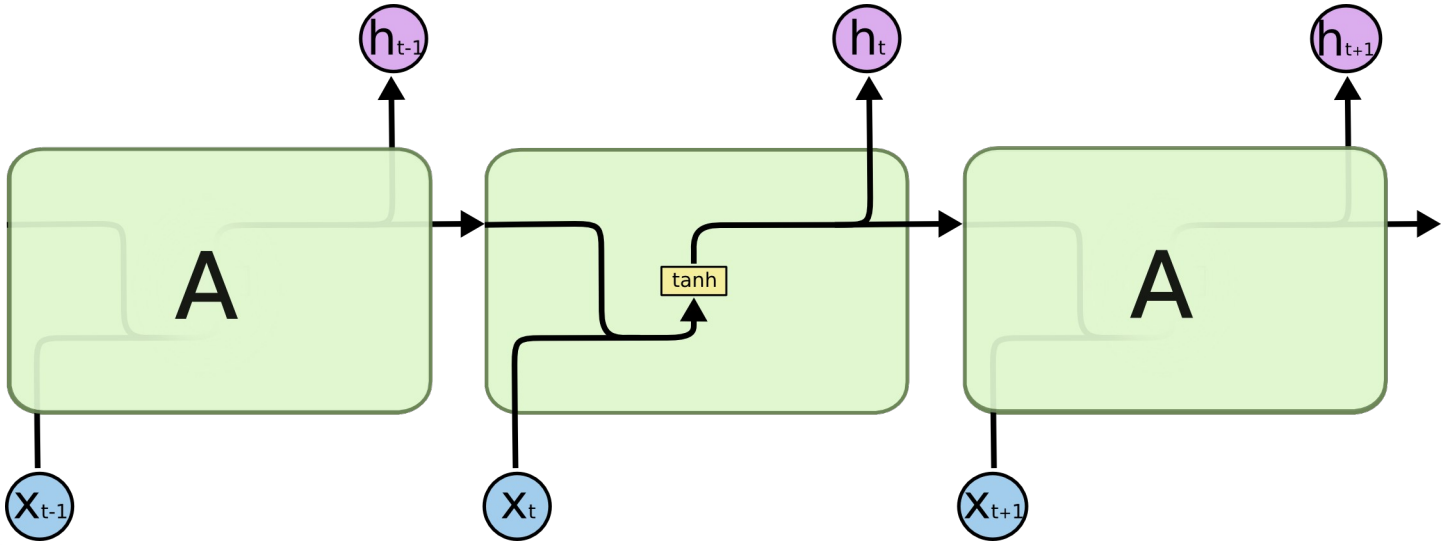
(Vinyals et al., 2015).

Ejemplo de LSTM: generador de subtítulos de imágenes

Vinyals, O. et al (2015). Show and Tell: A Neural Image Caption Generator. Recuperado de https://arxiv.org/pdf/1411.4555.pdf

La redes recurrentes (RNN)

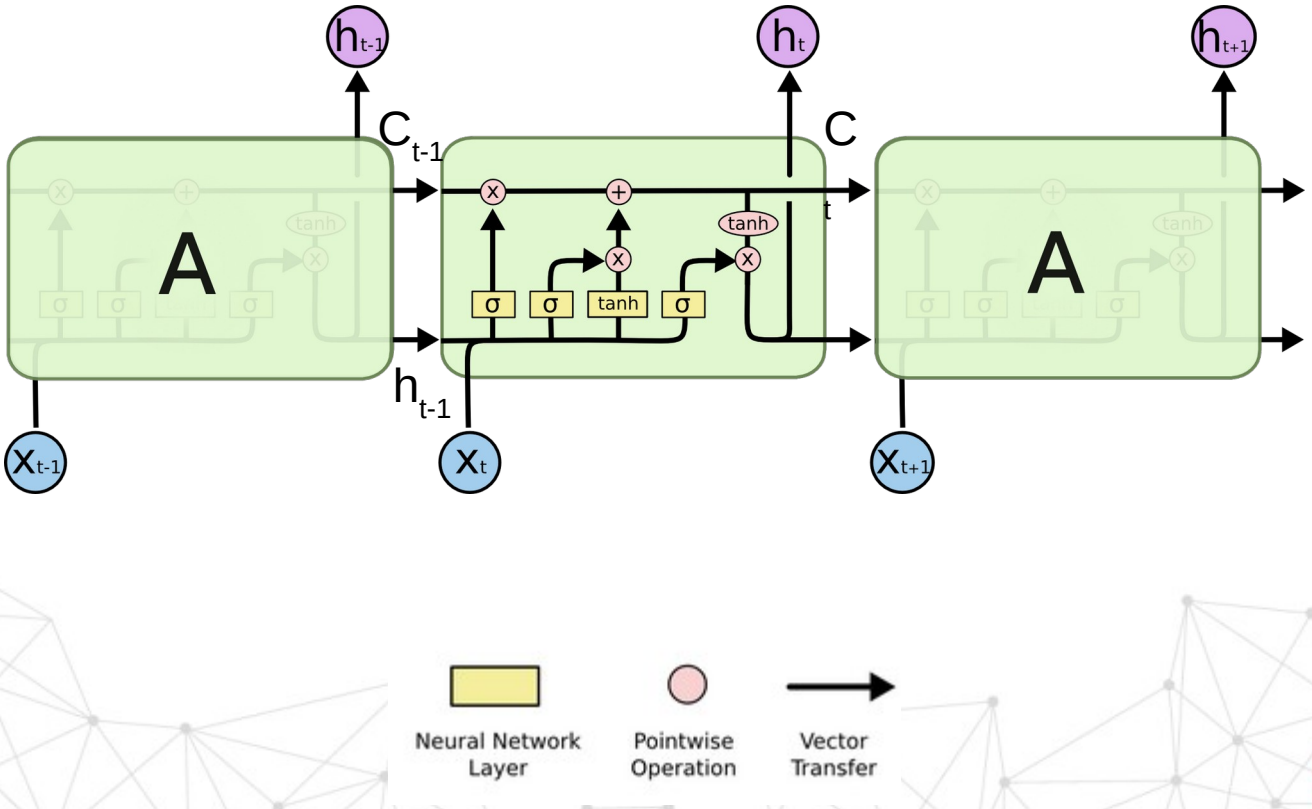
Las RNN estándar, repiten una estructura muy simple, como una sola capa de tanh



(Imagen Olah, 2015)

Long Short Term Memory (LSTM)

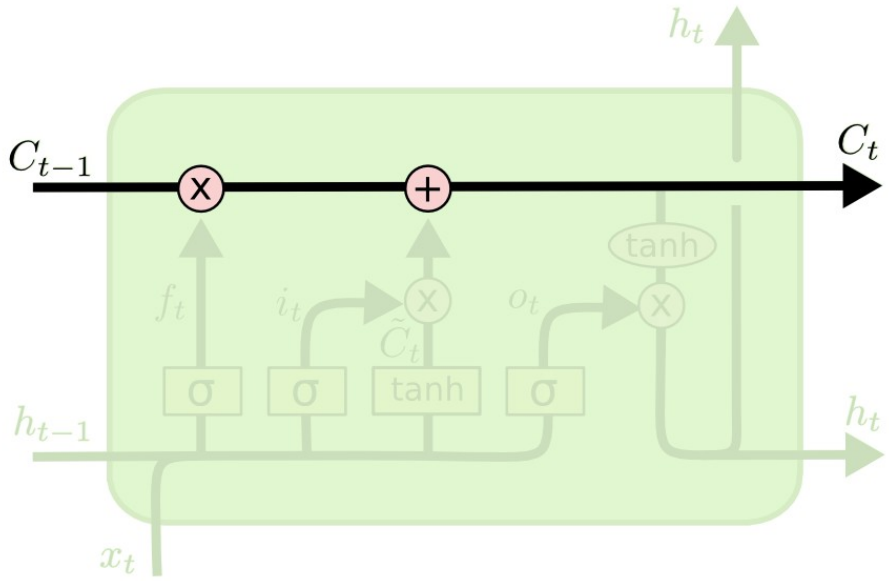
Las LSTM en lugar de tener una sola capa de red neuronal, tienen cuatro que interactúan entre ellas.



(Hochreiter and Schmidhuber, 1997; Imagen Olah, 2015)

Long Short Term Memory (LSTM)

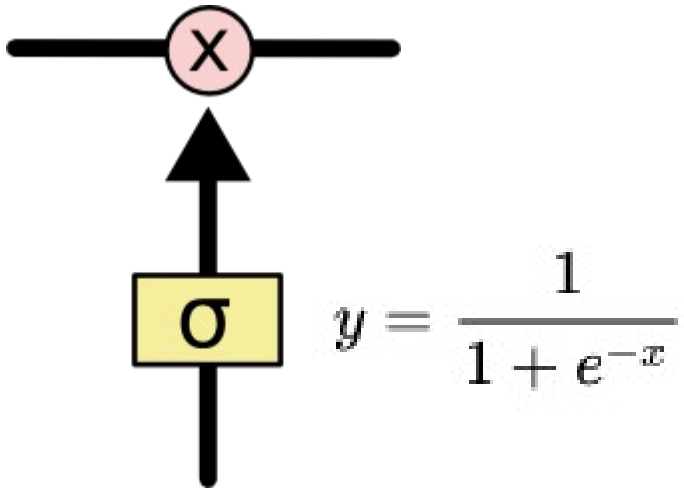
El valor candidato es actualizado a lo largo de toda la cadena, con solo algunas operaciones lineales menores.



(Hochreiter and Schmidhuber, 1997; Imagen Olah, 2015)

Long Short Term Memory (LSTM)

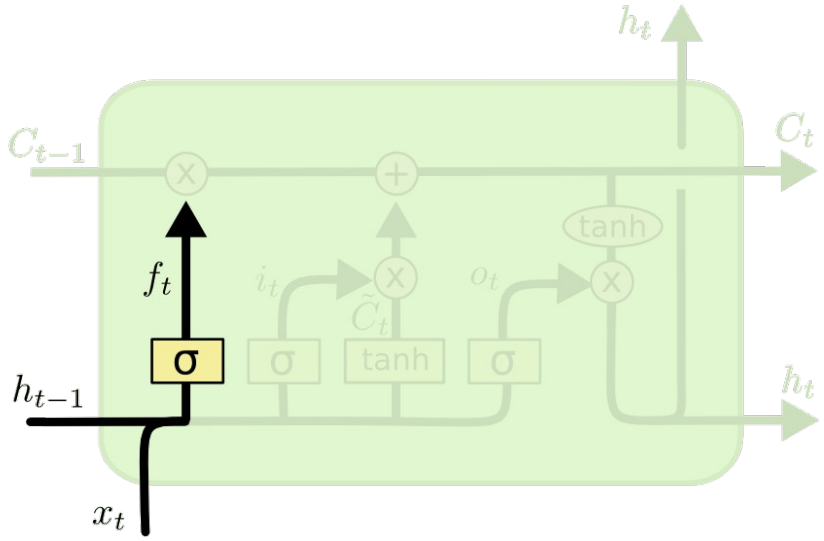
La LSTM tiene la capacidad de eliminar o agregar información a la celda de estado, regulada por estructuras llamadas **gates** (con función de activación sigmoide).



(Hochreiter and Schmidhuber, 1997; Imagen Olah, 2015)

Long Short Term Memory (LSTM)

El primer paso lo ejecuta la “forget gate layer” formada por una capa con función de activación sigmoide.



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

$$f_t = \sigma\left(W_{hh} h_{t-1} + W_{xh} x_t \; + \; b_f\right)$$

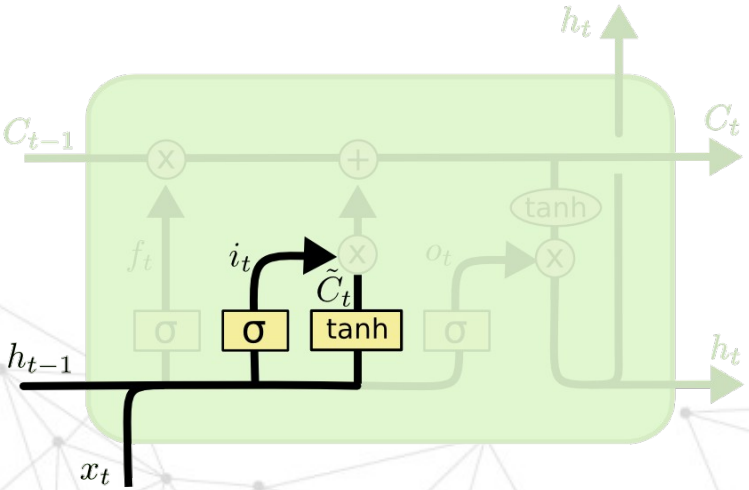
Bias

(Hochreiter and Schmidhuber, 1997; Imagen Olah, 2015)

Long Short Term Memory (LSTM)

El siguiente paso es decidir qué información nueva se va a agregar a la celda de estado.
Dividido en dos partes:

- Una capa sigmoidea llamada "input gate layer" que decide qué valores se actualizarán.
- Una capa tanh que crea un vector de nuevos valores candidatos, C'_t que podrían agregarse al estado.



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

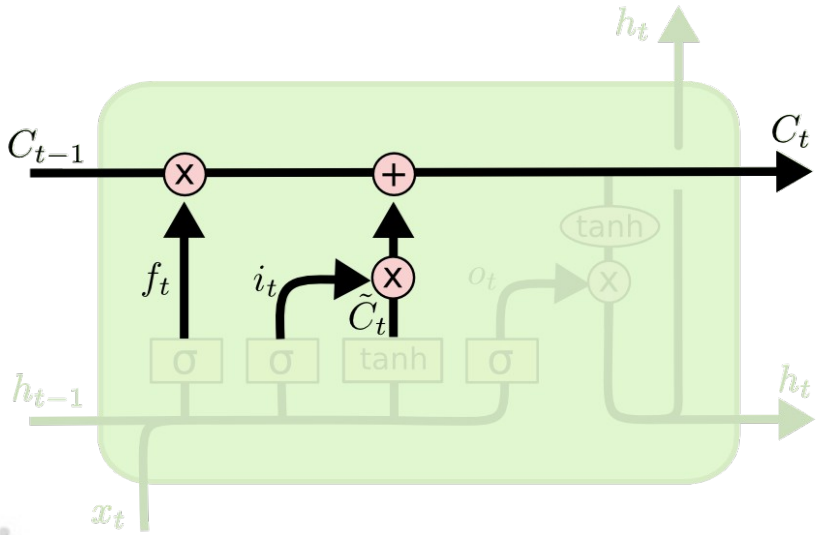
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

(Hochreiter and Schmidhuber, 1997; Imagen Olah, 2015)

Long Short Term Memory (LSTM)

Se multiplica el estado anterior por f_t, que representa el olvido. Luego se suma a $i_t * C'_t$. Estos corresponden a los nuevos valores candidatos, escalados.



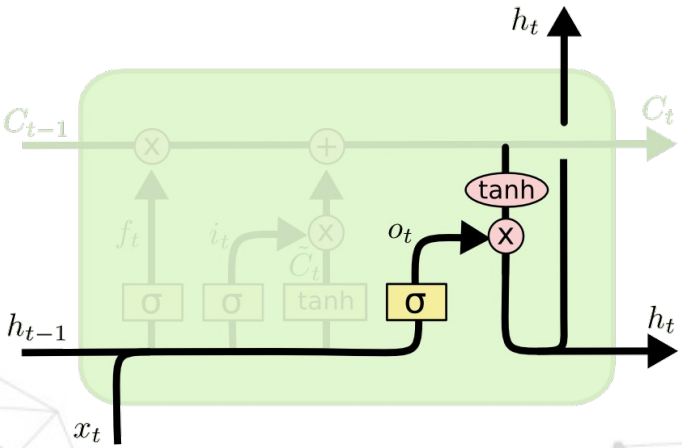
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

(Hochreiter and Schmidhuber, 1997; Imagen Olah, 2015)

Long Short Term Memory (LSTM)

Por último, se construye la salida del estado actual. Esta salida se basa en el estado de la celda, pero será una versión filtrada construida de la siguiente forma:

- Una capa sigmoidea que decide qué partes del estado se van a generar.
- Luego, se pasa la solución candidata por la función de tanh y se multiplica por la salida del gate sigmoideo.



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

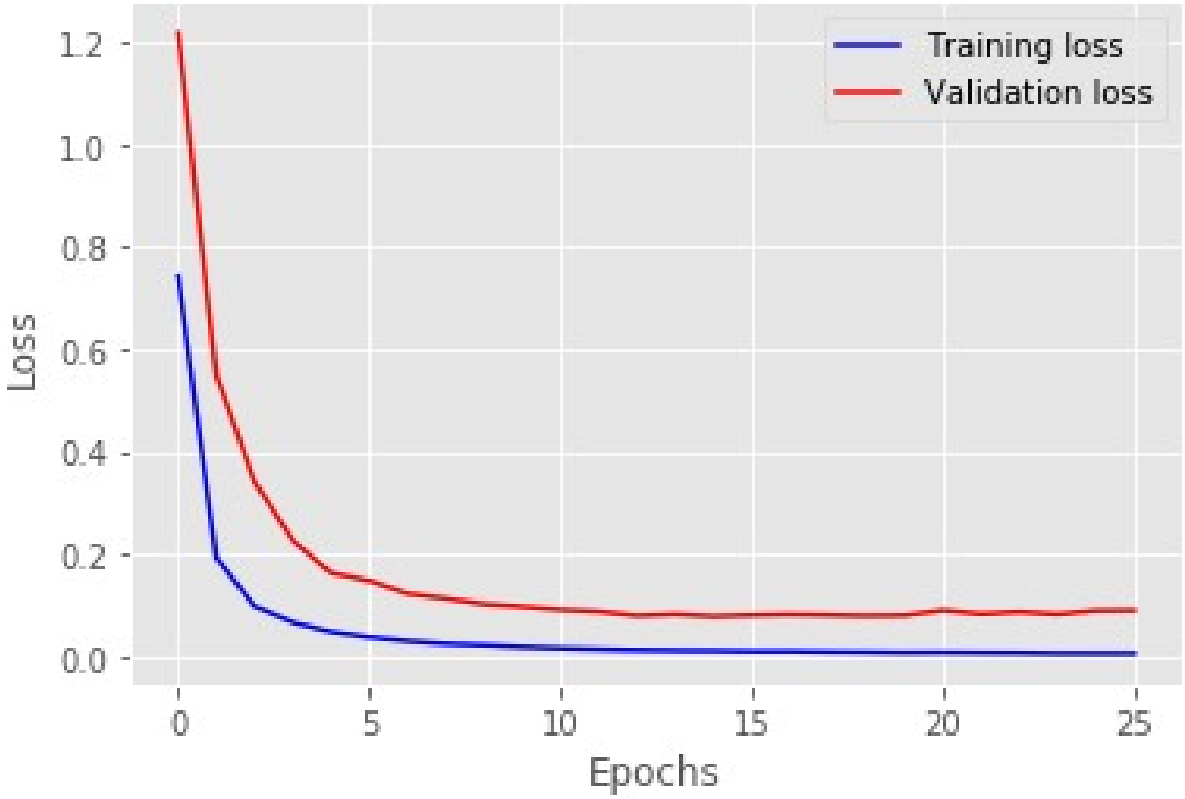
$$h_t = o_t * \tanh\left(C_t\right)$$

(Hochreiter and Schmidhuber, 1997; Imagen Olah, 2015)

Entrenamiento de modelos y la curva de error

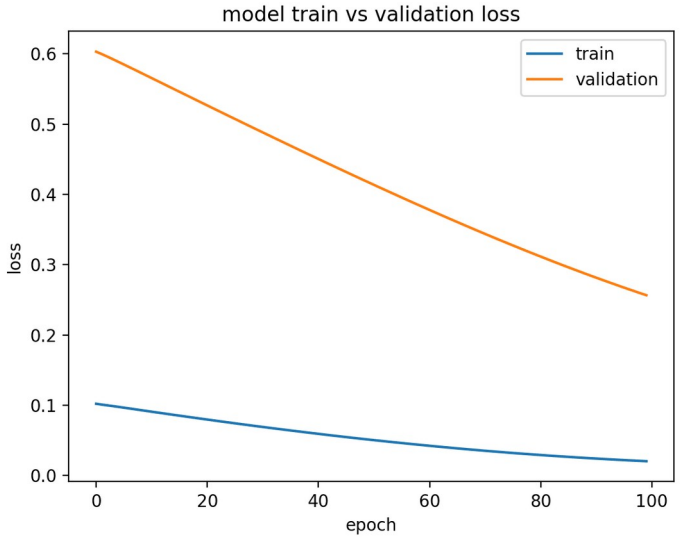
Análisis de la curva de error

Error acumulado calculado durante la fase de entrenamiento.

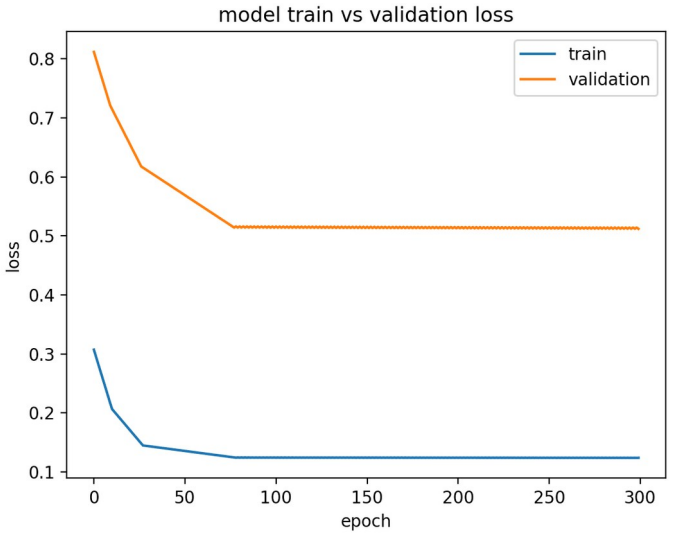


Análisis de la curva de error

Sub-ajuste del modelo



Sugerencia: Incremente el número de épocas

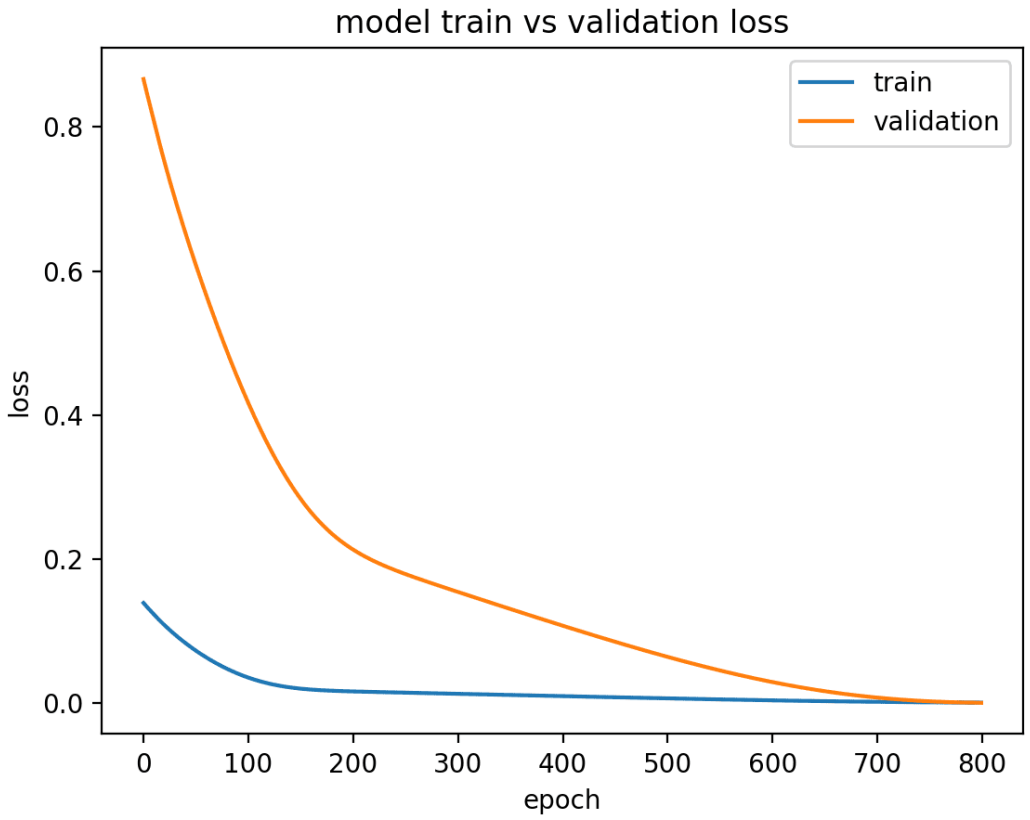


Sugerencia: Incremente la capacidad del modelo

Fuente: (Brownlee. 2017)

Análisis de la curva de error

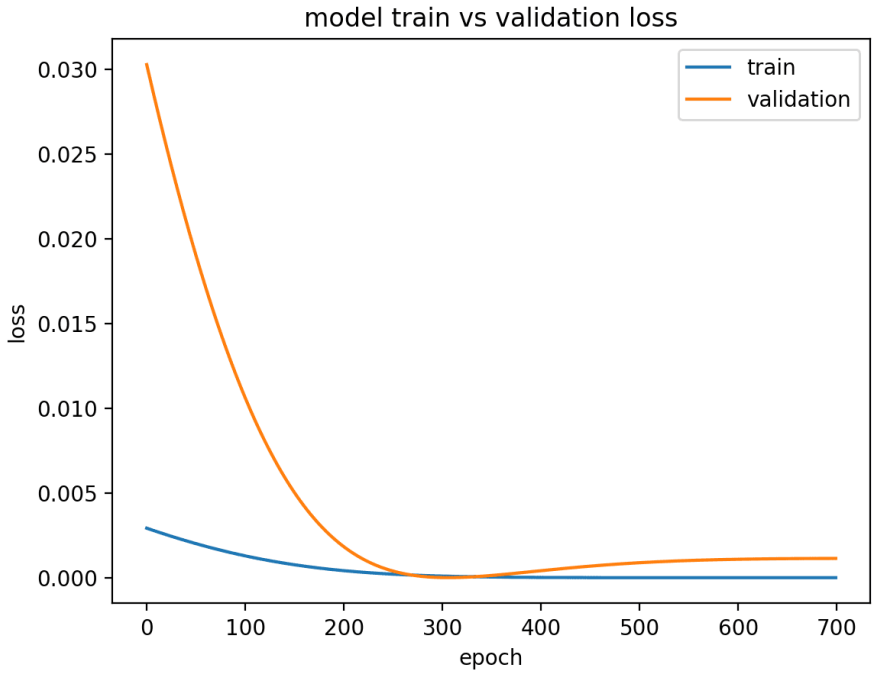
Ajuste esperado



Fuente: (Brownlee. 2017).

Análisis de la curva de error

Sobre-ajuste



Sugerencia: detener el entrenamiento en el punto de inflexión

Fuente: (Brownlee. 2017)

Referencias

- Li, F., Johnson, J. y Yeung, S. (2017). Recurrent Neural Network. Diapositivas. Recurepado de http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf
- LeCun,Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553):436,.
- Hinton, Geoffre (2013). Recurrent Neural Networks. Recuperado de https://www.cs.toronto.edu/~hinton/csc2535/notes/lec10new.pdf
- Olah, W. (2015). Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Bishop, C (2006). Pattern Recognition and Machine Learning. (S. Calderón, Trans.). Springer.
- Krizhevsky, A., Ilya, S., and Hilton, G. (2012). Imagenet Classification with Deep Convolutional Neural Networks. Advanced in Neural Information Processing Systems. Recuperado de https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
- Siddharth, M. (2021). Feature Extraction and Embeddings in NLP: A Beginners guide to understand Natural Language Processing. Recuperado de https://www.analyticsvidhya.com/blog/2021/07/feature-extraction-and-embeddings-in-nlp-a-beginners-guide-to-understand-natural-language-processing/