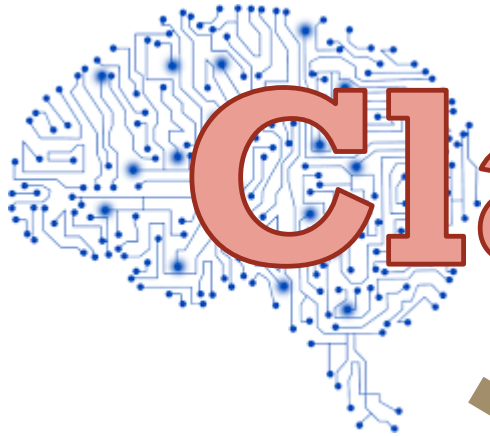
SECUENCIAS

Javier Diaz Cely, PhD

UNIVERSIDAD ICESI

AGENDA

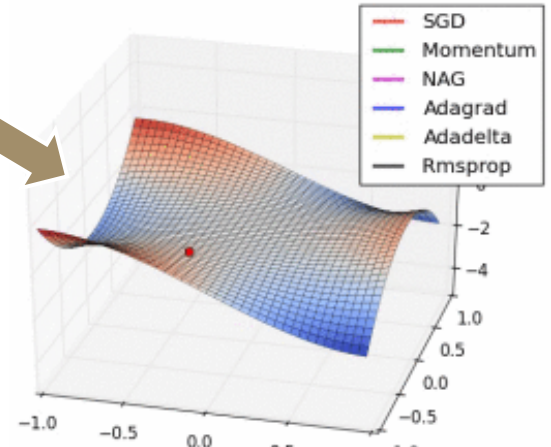


Deep Learning

Keras

TensorFlow

Clase anterior

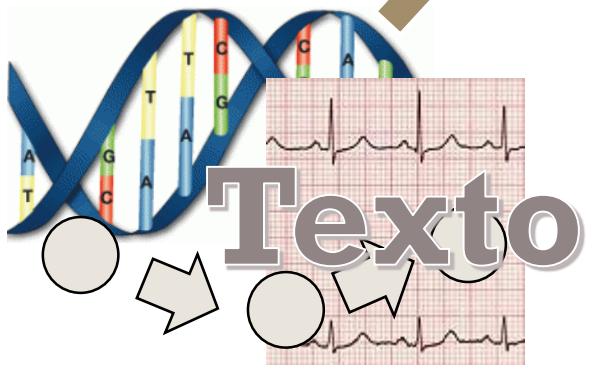


Optimizadores

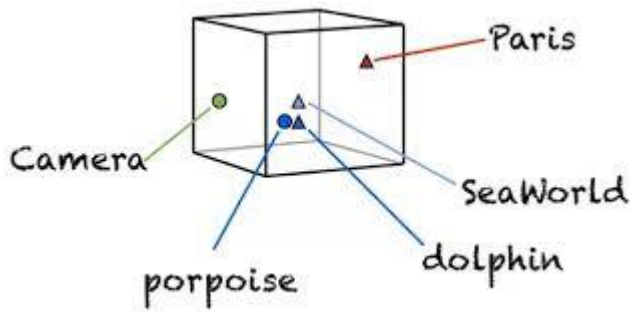
UNIVERSIDAD ICESI

AGENDA

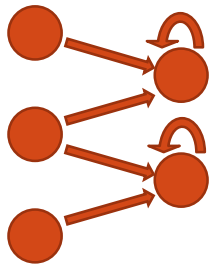
Deep Learning



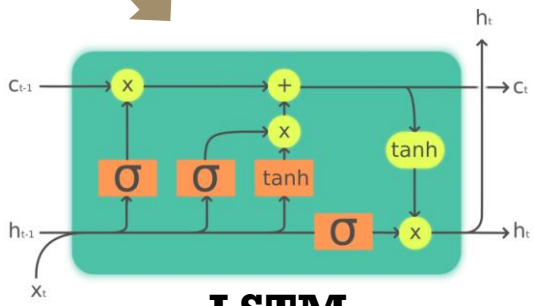
Secuencias



Embeddings



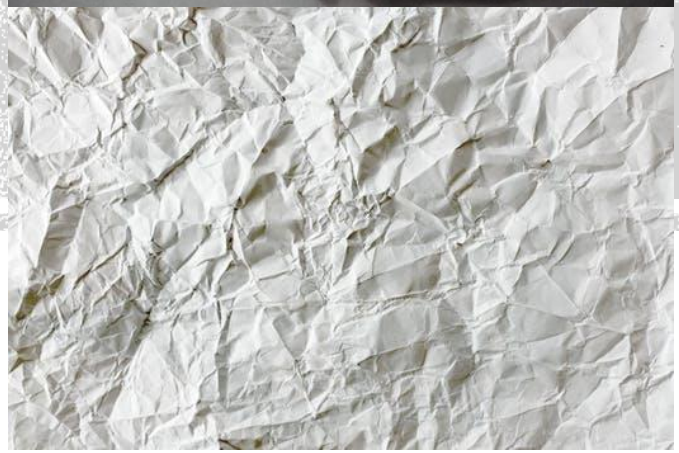
Redes recurrentes



LSTM

UNIVERSIDAD ICESI

EMBEDDINGS



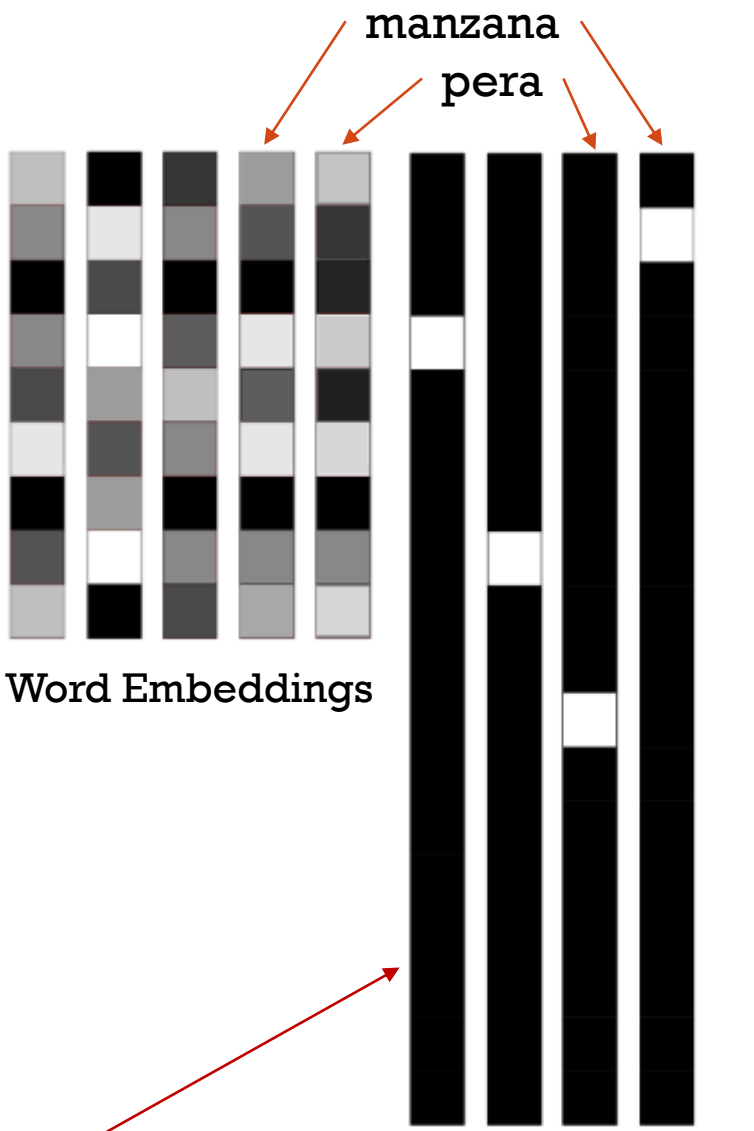
UNIVERSIDAD ICESI

SECUENCIAS

- Si se quiere procesar una secuencia en redes feed-forward, es necesario codificarla como una sola instancia que se procesa de manera atómica.
- Al igual que con las imágenes, vamos a explotar la estructura inherente de los datos analizados (secuencia)
- Cada nuevo elemento aporte nueva información a la obtenida a través de los elementos anteriores (actualización del conocimiento).
- Datos secuenciales:
  - Texto y procesamiento de lenguaje natural (NLP): análisis de sentimiento, chatbots, clasificación, traducción automática, descripción de imágenes
  - Series de tiempo: finanzas, señales de audio (habla, música), salud
  - ADN (Secuencias de genes)
  - Videos

EMBEDDINGS

- Para poder procesar y analizar secuencias de términos hay primero que definir el tipo de estructura que se va a utilizar para representarlas
- Representaciones:
  - Vectores de palabras con codificación one-hot
    - Datos dispersos
    - Alto número de dimensiones
    - Espacio de representación en "código duro"
  - Embeddings de términos
    - Datos densos
    - Dimensionalidad reducida
    - Espacio de representación aprendido desde los datos



¿Qué problemas tiene la representación one-hot?

Chollet, 2018

EMBEDDINGS

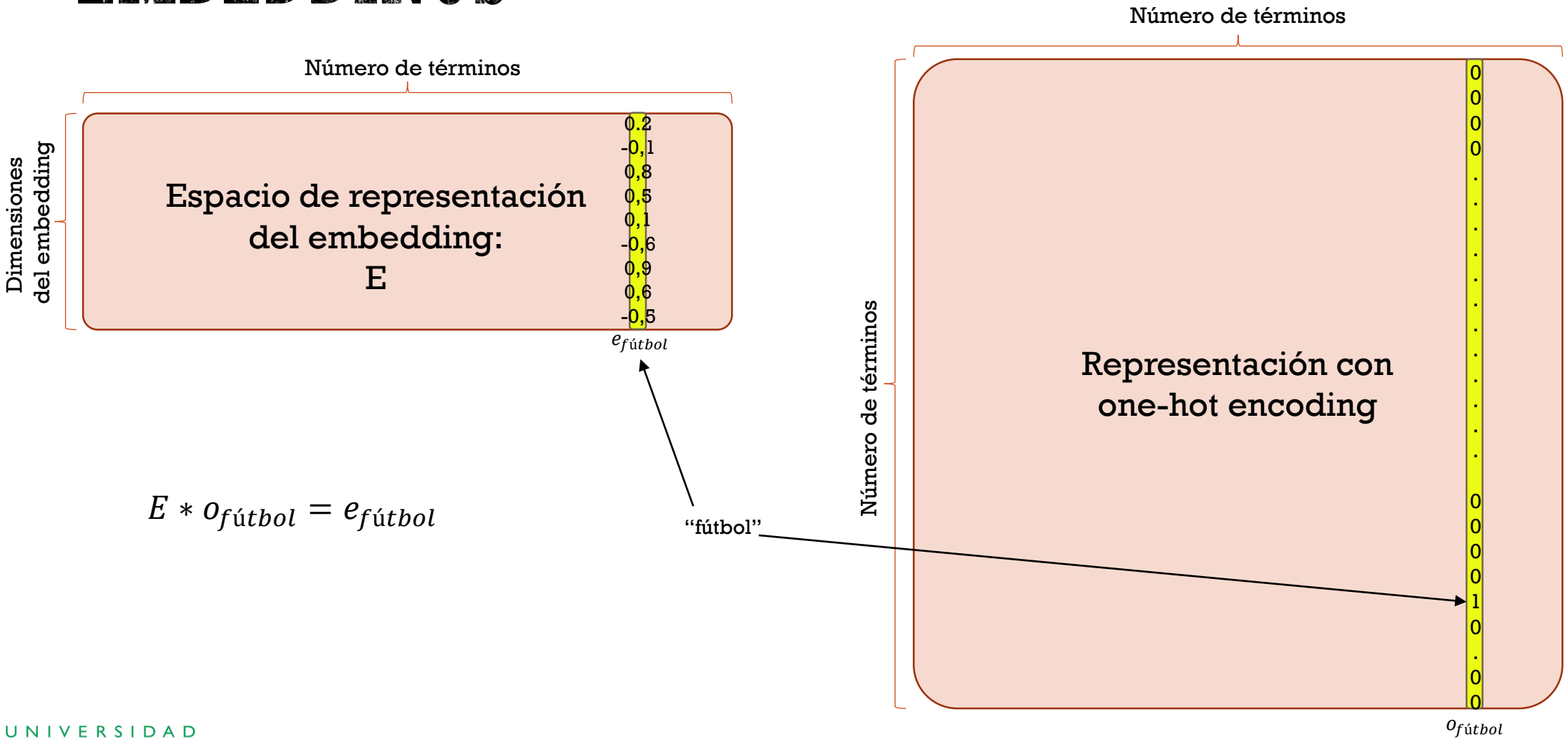
- La representación de los términos debe tener nociones semánticas
  - Cada término se va a **incrustar** en un nuevo espacio de representación que se debe **aprender** teniendo en cuenta unas secuencias de entrenamiento
  - No debe ser producto de un orden arbitrario (i.e. alfabético, frecuencia, aleatorio)
  - Co ocurrencia de términos en contextos similares (con otros tipos de término en común):

    "Felipe se quebró la pierna jugando fútbol"

    "James se quebró el ligamiento jugando _____"

    fútbol, basketball, volleyball
  - En un embedding, se utiliza el coseno como medida de similitud entre términos
  - Un proceso de reducción de la dimensionalidad final escogida, como t-SNE, debe reconocer la proximidad semántica de los términos

EMBEDDINGS



UNIVERSIDAD ICESI

EMBEDDINGS

- Para aprender un embedding se necesita establecer:
  - Un **corpus** de secuencias (e.g. textos)
  - La granularidad de los términos para lo **tokenización** de las secuencias
  - Los términos que se van a considerar como entrada (e.g. #de términos más **frecuentes**)
  - El número de **dimensiones** de representación del nuevo espacio vectorial

medium.com/@jayeshbahire/introduction-to-word-vectors-ea1d4e4b84bf

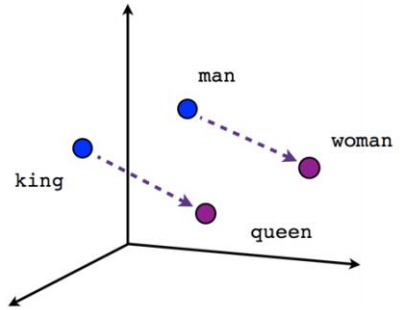
UNIVERSIDAD ICESI

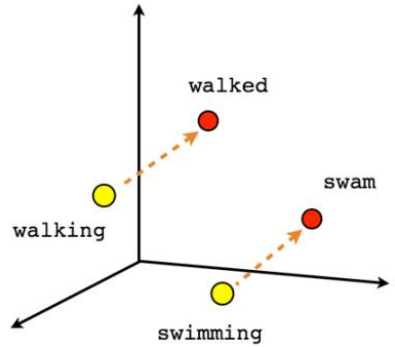
EMBEDDINGS

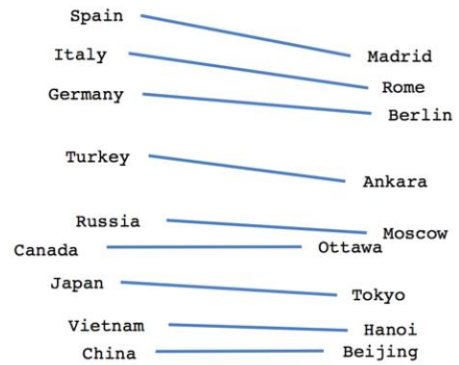
- No hay un embedding perfecto: **dependen del contexto** de las secuencias (e.g. un buen embedding para analizar los sentimientos de las críticas de películas de pronto es muy malo para analizar documentos legales)
- La calidad de un embedding depende del **tamaño del conjunto de secuencias de entrenamiento**. Entra más grande, mejor.
- Una capa embedding:
  - se aprende como una capa inicial de una red, a través de **back-propagation**, conectada a otras capas para el aprendizaje de una **tarea definida** (e.g. capas densas o recurrentes)
  - constituye un diccionario que **mapea** los índices de los tokens a los vectores en el nuevo espacio de representación

EMBEDDINGS

- En el nuevo espacio de representación las relaciones **semánticas** se expresan a partir de relaciones **geométricas vectoriales.**
  - Distancias geométricas representan distancias semánticas
  - Los sinónimos deben estar cerca (coseno como similitud)
  - Las direcciones tienen significado
  - Vectorialmente podemos identificar relaciones como: pluralidad, género, tamaño, conjugaciones
  - Relaciones geométricas y analogías:
    - $\overrightarrow{rey} - \overrightarrow{hombre} + \overrightarrow{mujer} = \overrightarrow{reina}$
    - $\overrightarrow{animal\ salvaje}$ es a $\overrightarrow{mascota}$ lo que $\overrightarrow{lobo}$ es a $\overrightarrow{perro}$







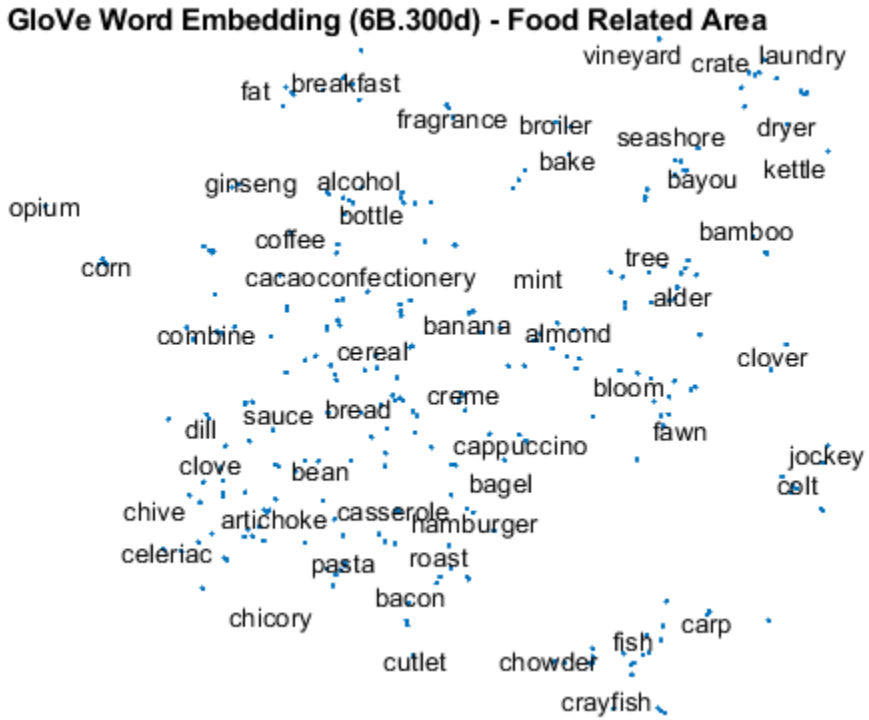
Towardsdatascience.com

EMBEDDINGS

- Una capa embedding:
  - se inicializa aleatoriamente
  - considera secuencias de entrada del mismo largo, por lo que se **truncan** si son más largas (de lo contrario se completan con 0s – -"**padding**"). Tanto la truncada como el padding se pueden definir al comienzo o al final de la secuencia.
  - recibe tensores de rango 2 como entrada: instancias x largo de la secuencia de términos
  - produce tensores de rango 3 como salida: instancias x largo de secuencia x dimensionalidad del embedding

T-SNE

- Transformación no lineal de datos representados en alta dimensionalidad a una dimensionalidad reducida (2D o 3D)
- Diferente a PCA (no lineal): los puntos que están cerca tienen una semántica asociada, pero el mapeo aprendido por T-SNE no conserva las relaciones geométricas de los embeddings ni se pueden usar como base para aprendizaje automático (e.g. no se pueden tomar como base de K-NN o K-Means)



blogs.mathworks.com/loren/2017/09/21/math-with-words-word-embeddings-with-matlab-and-text-analytics-toolbox/

UNIVERSIDAD ICESI

TALLER EMBEDDINGS

Desarrollar la parte 1 del taller de embeddings, dedicado a la representación de se secuencias utilizando:

- one hot encoding (básico)
- aprendiendo un embedding sobre un conjunto de reviews de películas específico para la tarea de análisis de sentimiento

UNIVERSIDAD ICESI

EMBEDDINGS

Transfer Learning:

- El entrenamiento de un embedding tiene **requerimientos** importantes en **tiempo** de procesamiento, **tamaño** del corpus de entrenamiento, y eventualmente de una **infraestructura** computacional importante (GPUs).
- Se puede **reutilizar** embeddings aprendidos para una tarea original, con el fin de realizar tareas análogas con conjuntos de entrenamiento mucho más pequeños.
- Se pueden **afinar** (fine-tune) embeddings transferidos considerando la nueva tarea a aprender.

TALLER EMBEDDINGS

Desarrollar la parte 2 del taller de embeddings, dedicado a:

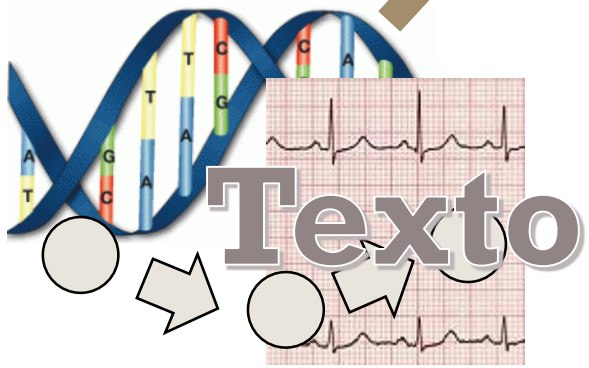
- el aprendizaje de embeddings desde un corpus de textos
- la reutilización de un embedding a través del transfer learning

Para la tarea de clasificar los sentimiento de las críticas de las películas

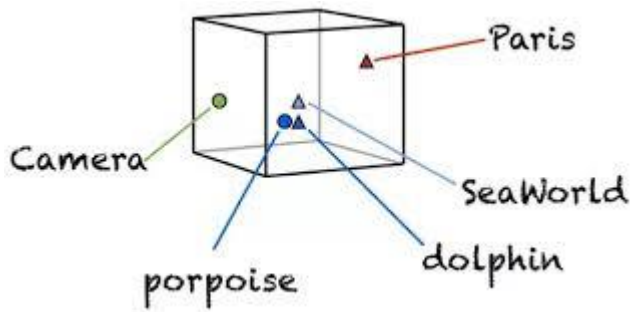
UNIVERSIDAD ICESI

AGENDA

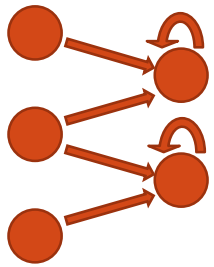
Deep Learning



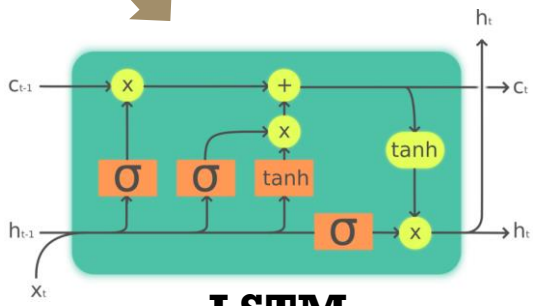
Secuencias



Embeddings



Redes recurrentes



LSTM

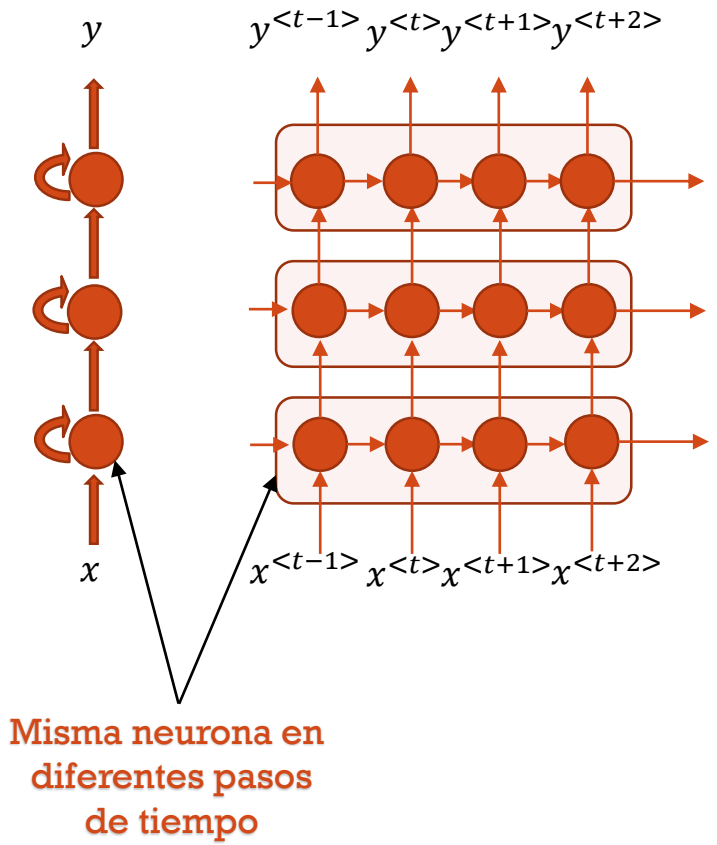
UNIVERSIDAD ICESI

REDES RECURRENTES BÁSICAS

UNIVERSIDAD ICESI

REDES RECURRENTES

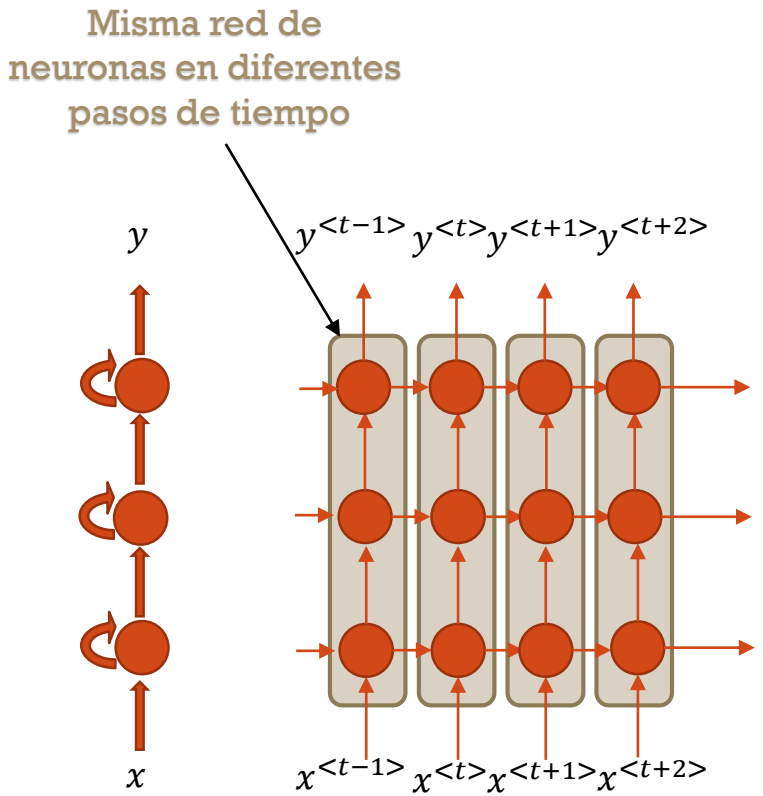
- Datos secuenciales: considerar información generada por estados anteriores
- Las activaciones generadas por un dato en una neurona pueden ser consideradas como entradas de neuronas al procesar el dato siguiente → **estado** de las activaciones anteriores: **red con memoria**
- 2 Matrices de parámetros: 1 para las entradas nuevas, 1 para las activaciones anteriores
- Las células recurrentes se conectan unas con otras y con sus propias salidas
- Una capa (“layer”) recurrente está compuesta por varias neuronas (“cell”), conectadas entre ellas y con ellas mismas



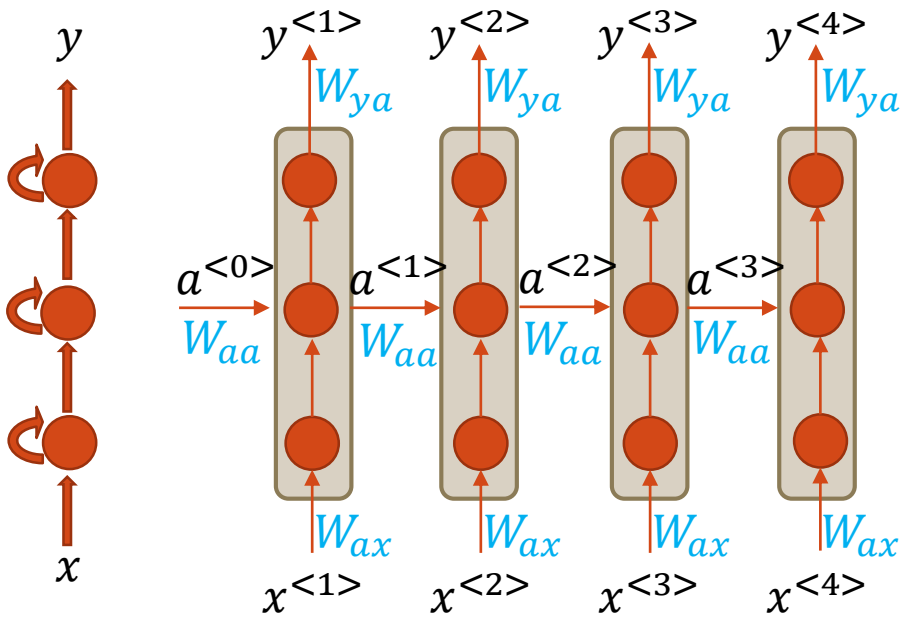
UNIVERSIDAD ICESI

REDES RECURRENTES

- Datos secuenciales: considerar información generada por estados anteriores
- Las activaciones generadas por un dato en una neurona pueden ser consideradas como entradas de neuronas al procesar el dato siguiente → **estado** de las activaciones anteriores: **red con memoria**
- 2 Matrices de parámetros: 1 para las entradas nuevas, 1 para las activaciones anteriores
- Las células recurrentes se conectan unas con otras y con sus propias salidas
- Una capa (“layer”) recurrente está compuesta por varias neuronas (“cell”), conectadas entre ellas y con ellas mismas



REDES RECURRENTES – BACKPROPAGATION THROUGH TIME



Parámetros compartidos por cada paso de tiempo:

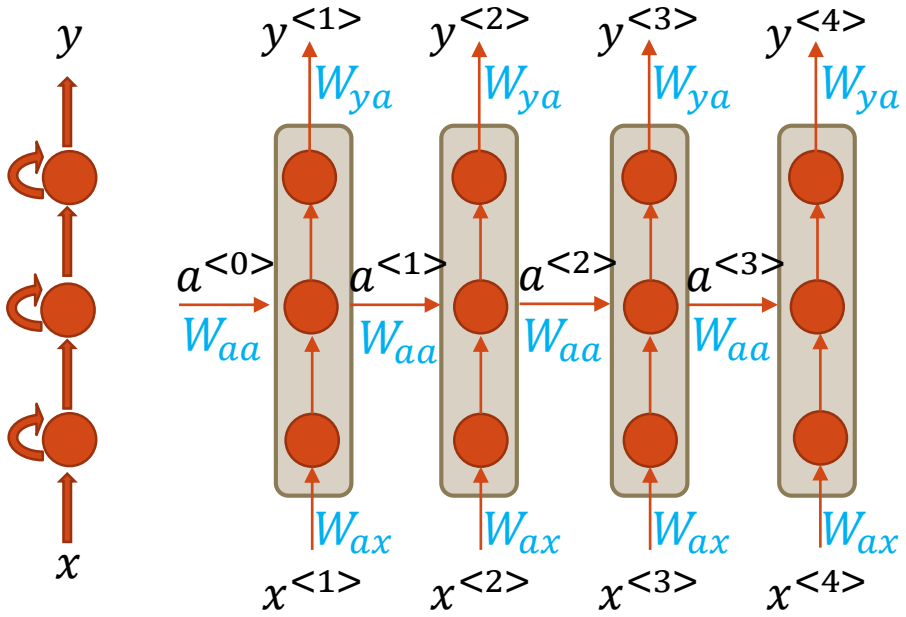
- W_{ax}: considera la entrada $x^{<t>}$ actual para calcular $a^{<t>}$
- W_{aa}: considera la entrada $a^{<t-1>}$ anterior para calcular $a^{<t>}$
- W_{ya}: considera la entrada $a^{<t>}$ actual para calcular $y^{<t>}$

Dos flujos de información, cada uno con su propia función de activación

- Flujo de propagación:
  - Activaciones: tanh, ReLU
  - $a^{<t>} = g_1(W_{aa} * a^{<t-1>} + W_{ax} * x^{<t>} + b_a)$
- Flujo de salida:
  - Activaciones: sigmoid, softmax, lineal
  - $y^{<t>} = g_2(W_{ya} * a^{<t>} + b_y)$

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

REDES RECURRENTES – BACKPROPAGATION THROUGH TIME



$$a^{<t>} = g_1\left(\begin{bmatrix} W_{aa} & W_{ax} \end{bmatrix} * \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} + b_a \right)$$

W_a: # neuronas × (# neuronas + # instancias); $a^{<t-1>}$: # neuronas × # instancias; b_a: # neuronas × 1

$$y^{<t>} = g_2\left(W_{ya} * a^{<t>} + b_y \right)$$

W_y: # salidas × # neuronas; $a^{<t>}$: # neuronas × # instancias; b_y: # salidas × 1

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

UNIVERSIDAD ICESI

Andrew Ng, Coursera, 2017

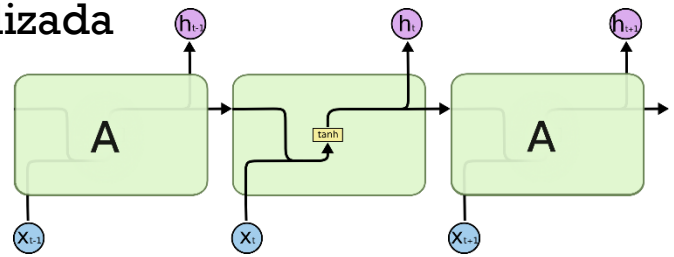
REDES RECURRENTES – BACKPROPAGATION THROUGH TIME

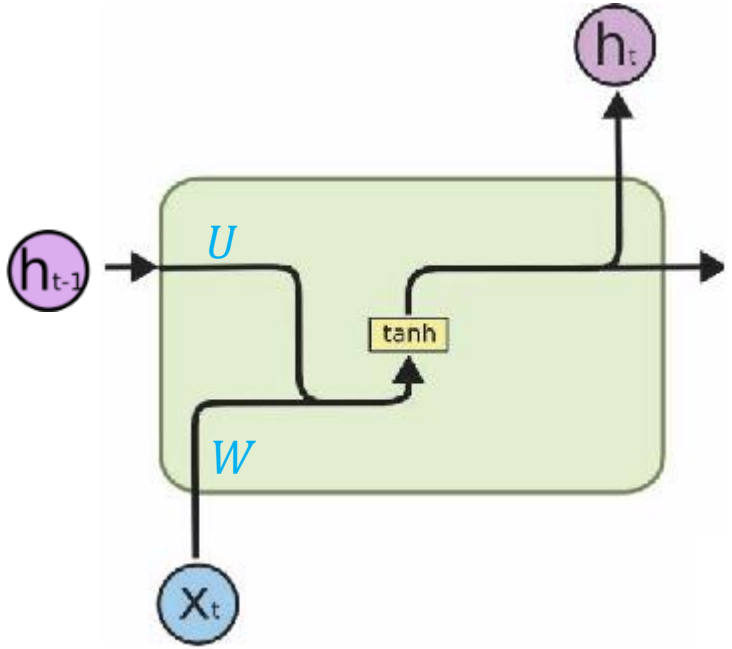
- El proceso de actualización es análogo al del back propagation normal.
- Se calcula para cada paso de tiempo t la función de perdida $L^{<t>}(\hat{y}^{<t>}, y^{<t>})$.
- Se agregan las perdidas de toda la secuencia (suma de los $L^{<t_i>}$)
- Se propagan los errores desde el último paso de tiempo hasta el primero, siguiendo el descenso de gradiente con las derivadas parciales de la perdida con respecto a los parámetros W_a, b_a, W_y, b_y → "Through time"

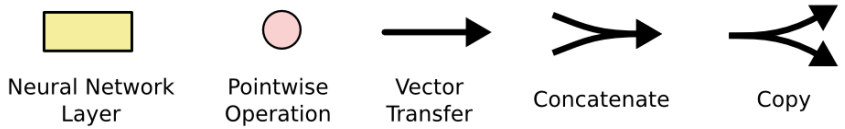
REDES RECURRENTES – SIMPLE RNN

SimpleRNN: Se trata de un modelo de RNN cuya salida anterior ($h^{<t-1>}$) es el estado que se comunica al siguiente paso de la red, combinada con la entrada ($x^{<t>}$) para obtener la salida actual ($h^{<t>}$):

- $h^{<t>} = g(W * x^{<t>} + U * h^{<t-1>} + b)$
- Parámetros:
  - W: matriz de pesos aplicada a los inputs
  - U: matriz de pesos aplicada a las salidas anteriores
  - b: sesgo
- Utiliza una función de activación g; siendo *tanh* la más comúnmente utilizada







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

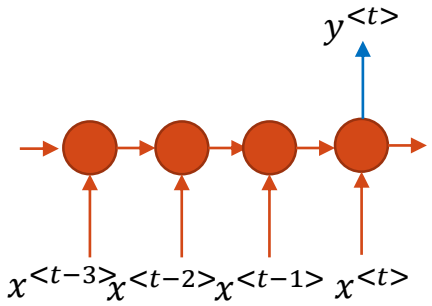
UNIVERSIDAD ICESI

TALLER ANÁLISIS DE SENTIMIENTO CON SIMPLERNN EN KERAS

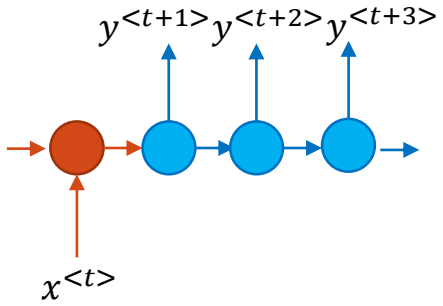
Utilizar un modelo SimpleRNN en Keras para analizar críticas de películas representadas a través de Embeddings

UNIVERSIDAD ICESI

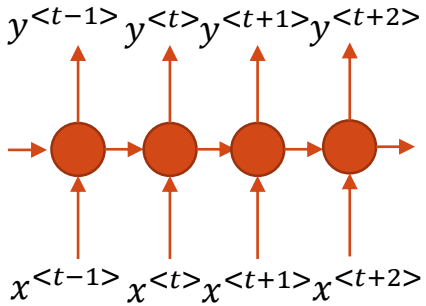
REDES RECURRENTES



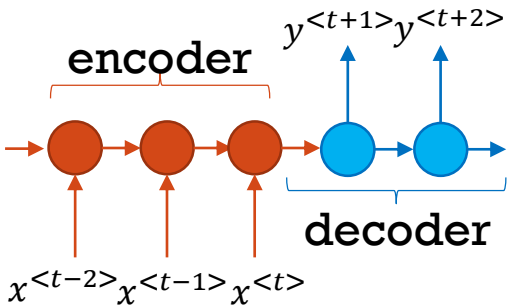
Secuencia a valor



Valor a secuencia (generación)



Secuencia a secuencia (mismo largo)



Secuencia a secuencia (largos diferentes)

Diferentes arquitecturas para diferentes tipos de problemas:

- Series de tiempo
- Traducción de textos
- Chatbots
- Creación automática (texto, audio)
- Reconocimiento de habla
- Descripción de imágenes
- Reconocimiento de género musical
- …

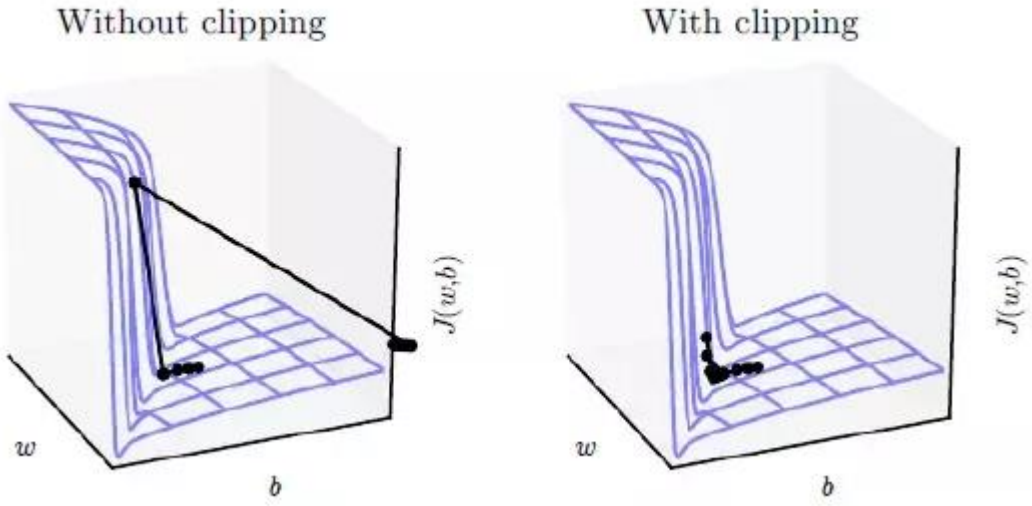
UNIVERSIDAD ICESI

REDES RECURRENTES

Gradient clipping:

- Permite controlar la **explosión del gradiente**, que ocurre cuando hay pendientes muy importantes en la superficie de optimización, situación común en las RNN
- **Limita la magnitud de la actualización** de los parámetros a partir del gradiente al propagar a través del tiempo durante el entrenamiento, definiendo un umbral
- En **Keras**, los optimizadores cuentan con argumento clipnorm (un buen valor es 1.0) y clipvalue (un buen valor es 0.5)

→ Estabiliza el aprendizaje de los modelos



Ian Goodfellow et. al, MIT press, 2016

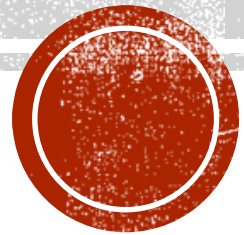
UNIVERSIDAD ICESI

REDES RECURRENTES

Consideraciones de las RNN básicas

- Las secuencias sirven para representar las entradas y/o salidas de las RNNs
- Aplicaciones múltiples: forecasting, traducción, …
- Sufren de una posible **explosión del gradiente**:
  - Uso de **gradient clipping**
- Sufren del **desvanecimiento del gradiente**:
  - Imposibilidad de capturar patrones y dependencias en las secuencias
  - Olvido de patrones de corto plazo en el largo plazo (memoria de corto plazo)
  - → **Redes recurrentes con memoria de largo plazo (e.g. GRUs, LSTMs)**

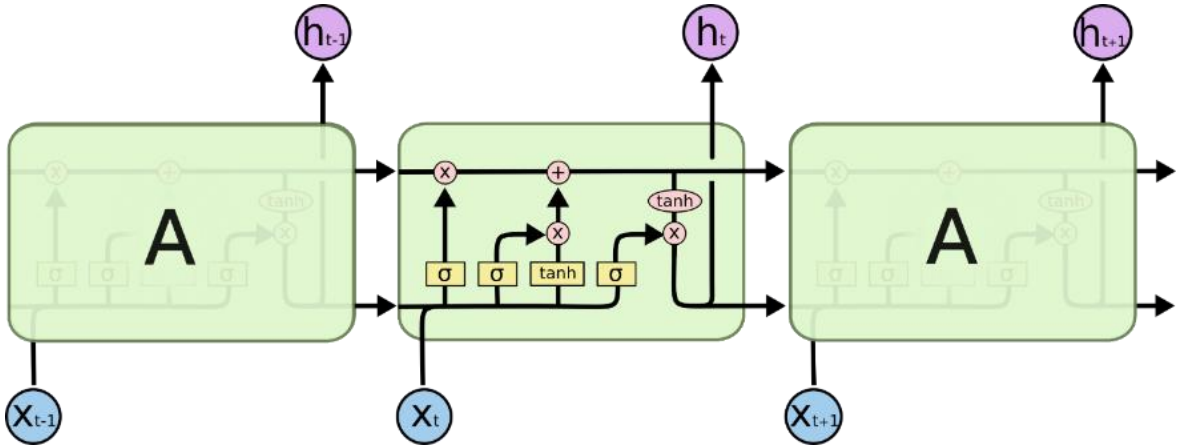
REDES RECURRENTES CON MEMORIA



UNIVERSIDAD ICESI

REDES RECURRENTES CON MEMORIA

- Las informaciones de estados anteriores lejanos influyen poco en las activaciones de estados futuros → desvanecimiento del gradiente
- Patrones de corto plazo encontrados no son olvidados en el largo plazo
- Se define una célula o módulo (GRU o LSTM), que sirve después como bloques de base que al concatenarlos forman un modelo predictivo.



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

UNIVERSIDAD ICESI

GRU (GATED RECURRENT UNITS) SIMPLE

Las redes recurrentes con memoria se construyen a partir de la repetición de celdas que internamente incurren en varios cálculos te tensores representando diferentes conceptos:

- $\tilde{h}^{<t>}$: Valor candidato a convertirse en el estado de la memoria en el tiempo t
- $h^{<t>}$: Valor de la memoria en el tiempo t.

Compuertas controlan si se deja o no pasar información, se aplican a cada posición del espacio de representación:

- $r^{<t>}$: Compuerta que considera la relevancia del estado de la memoria $\tilde{h}^{<t-1>}$ anterior en el cálculo del valor candidato de la memoria $\tilde{h}^{<t>}$
- $z^{<t>}$: Compuerta que considera si se debe actualizar o no la memoria con el valor candidato $\tilde{h}^{<t>}$

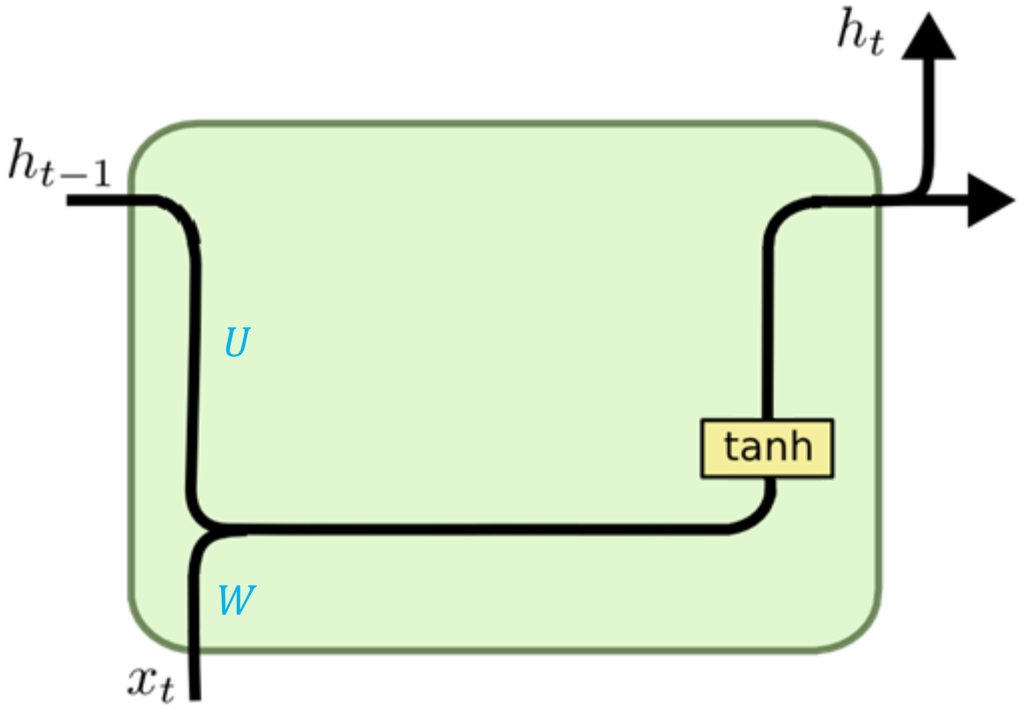
GRU (GATED RECURRENT UNITS) SIMPLE

Simple RNN

$$h^{<t>} = g(W * x^{<t>} + U * h^{<t-1>} + b)$$

$$h^{<t>} = g(W_a * [x^{<t>}, h^{<t-1>}] + b)$$

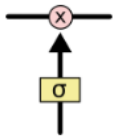
$$h^{<t>} = \underbrace{\begin{bmatrix} W & U \end{bmatrix}}_{W_a} * \begin{bmatrix} x^{<t>} \\ h^{<t-1>} \end{bmatrix}$$



UNIVERSIDAD ICESI

GRU (GATED RECURRENT UNITS) SIMPLE

GRU simplificada:



Define una compuerta (gate) $z^{<t>}$:

- $z^{<t>} = \sigma(W_z * [h^{<t-1>}, x^{<t>}] + b_{\tilde{h}})$, controla la actualización de la memoria

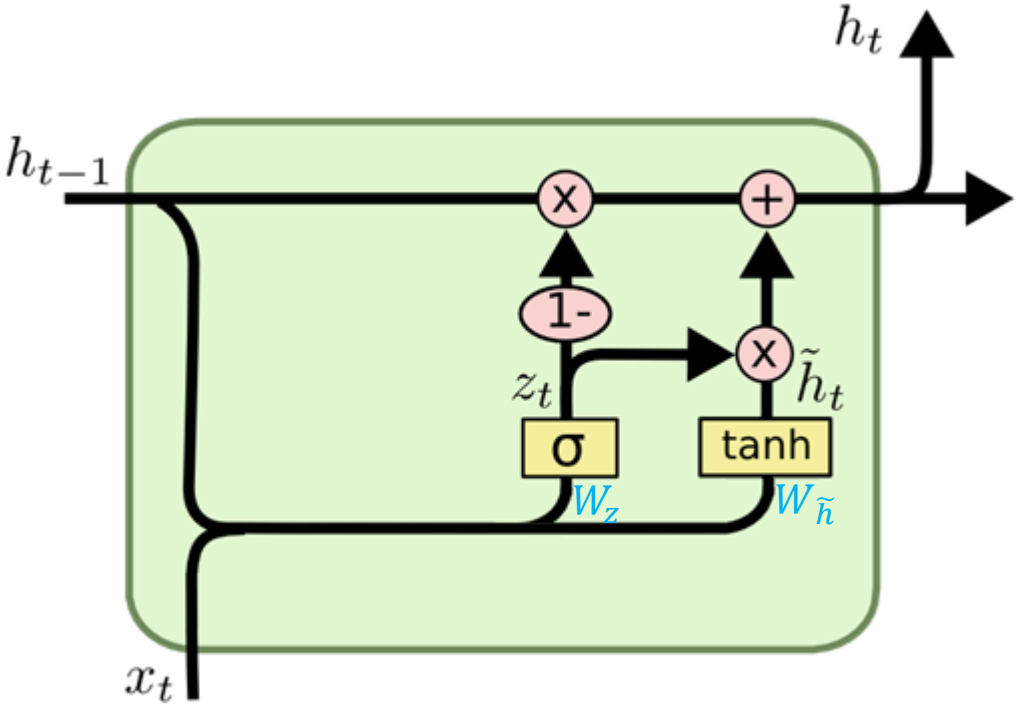
Se calcula un valor de memoria candidato:

- $\tilde{h}^{<t>} = \tanh(W_{\tilde{h}} * x^{<t>} + b_{\tilde{h}})$

Que se pondera con el valor anterior, teniendo en cuenta la relevancia

- $h^{<t>} = (1 - z^{<t>}) \circledast h^{<t-1>} + z^{<t>} \circledast \tilde{h}^{<t>}$

Elemento por elemento



UNIVERSIDAD ICESI

GRU (GATED RECURRENT UNITS)

GRUs (Cho, et al. (2014)):

Define dos compuertas (gates)

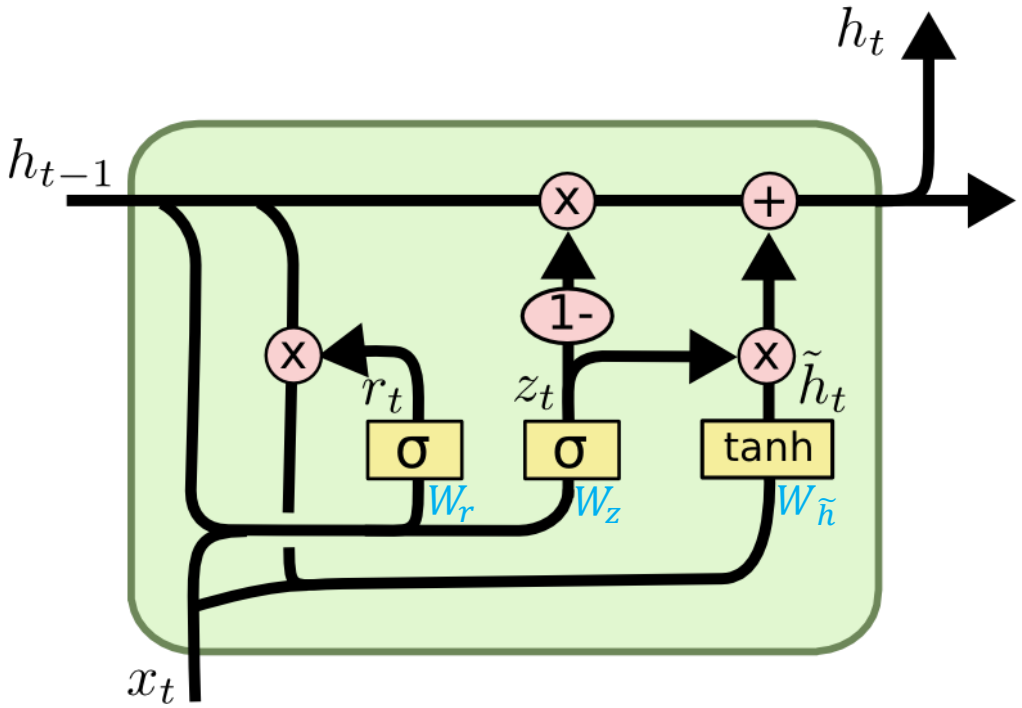
- $z^{<t>} = \sigma(W_z * [h^{<t-1>}, x^{<t>}] + b_{\tilde{h}})$, controla la actualización de la memoria
- $r^{<t>} = \sigma(W_r * [h^{<t-1>}, x^{<t>}] + b_r)$, controla la relevancia del estado anterior de la memoria

Se calcula un valor de memoria candidato:

- $\tilde{h}^{<t>} = \tanh(W_{\tilde{h}} * (r^{<t>} * [h^{<t-1>}, x^{<t>}]) + b_{\tilde{h}})$

Que se pondera con el valor anterior, teniendo en cuenta la relevancia

- $h^{<t>} = (1 - z^{<t>}) \circledast h^{<t-1>} + z^{<t>} \circledast \tilde{h}^{<t>}$

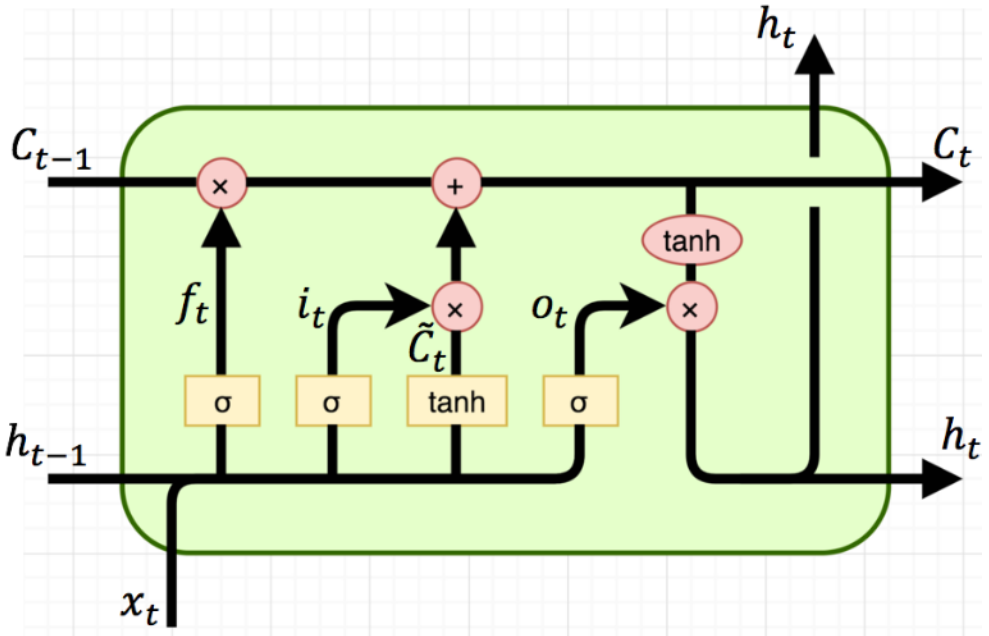


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM (LONG SHORT-TERM MEMORY)

LSTM: (Hochreiter & Schmidhuber (1997))

- Diferenciación entre $c^{<t>}$ y $h^{<t>}$
  - $c^{<t>}$ es el estado de la celda de memoria, que se actualiza después de cada paso
  - $h^{<t>}$ es la salida de cada paso secuencial
  - Ya no tenemos que $h^{<t>} = c^{<t>}$ como con las GRUs
- Se tienen 3 compuertas
  - de actualización del input ($i^{<t>}$) del estado anterior.
  - de olvido ($f^{<t>}$) del estado anterior.
  - de salida o output ($o^{<t>}$)
- Se tiene un valor candidato del nuevo estado de la memoria $\tilde{c}^{<t>}$



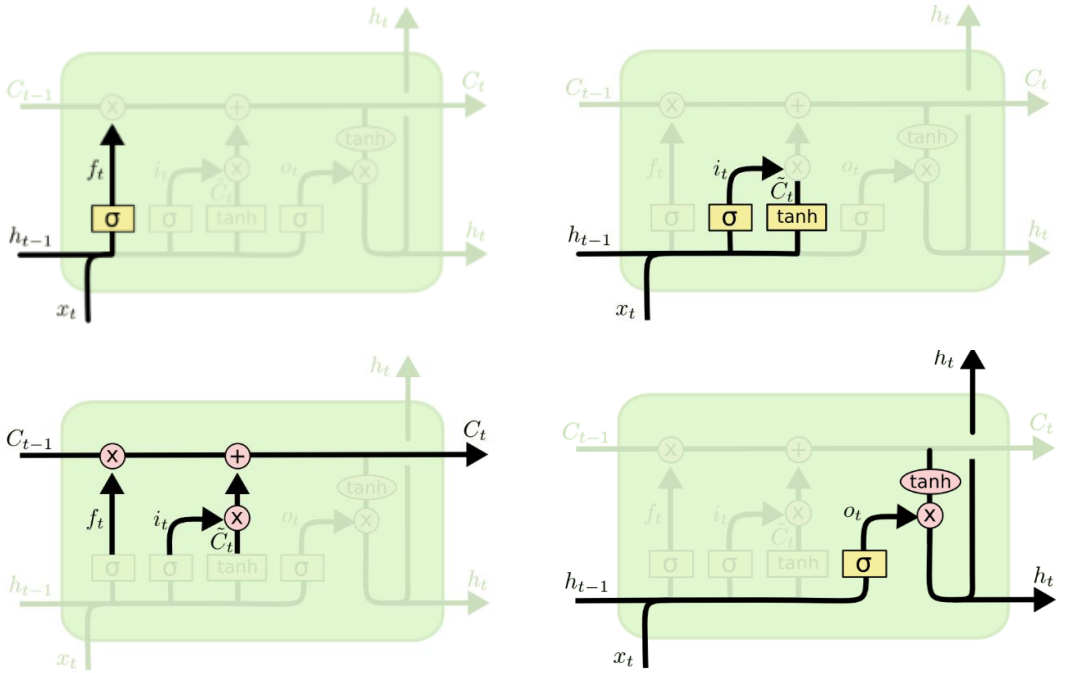
https://isaacchanghau.github.io/post/lstm-gru-formula/

UNIVERSIDAD ICESI

LSTM (LONG SHORT-TERM MEMORY)

LSTM: (Hochreiter & Schmidhuber (1997))

1. $f^{<t>} = \sigma(W_f * [h^{<t-1>}, x^{<t>}] + b_f)$, se decide que información se va a olvidar
2. $i^{<t>} = \sigma(W_i * [h^{<t-1>}, x^{<t>}] + b_i)$, se decide que información se va a actualizar
3. $\tilde{c}^{<t>} = tanh(W_c * [h^{<t-1>}, x^{<t>}] + b_c)$, se crea un vector candidato a nuevo estado
4. $c^{<t>} = f^{<t>} \circledast c^{<t-1>} + i^{<t>} \circledast \tilde{c}^{<t>}$, se establece el nuevo estado de la memoria
5. $o^{<t>} = \sigma(W_o * [h^{<t-1>}, x^{<t>}] + b_o)$, se calcula un filtro de salida
6. $h^{<t>} = o^{<t>} * tanh(c^{<t>})$, se define la salida del paso de la secuencia



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

UNIVERSIDAD ICESI

LSTM PEEP-HOLE

LSTMs:

- $\tilde{c}^{<t>} = tanh(W_c * [h^{<t-1>}, x^{<t>}] + b_c)$
- $i^{<t>} = \sigma(W_i * [h^{<t-1>}, x^{<t>}] + b_i)$
- $f^{<t>} = \sigma(W_f * [h^{<t-1>}, x^{<t>}] + b_f)$
- $c^{<t>} = f^{<t>} \circledast c^{<t-1>} + i^{<t>} \circledast \tilde{c}^{<t>}$
- $o^{<t>} = \sigma(W_o * [h^{<t-1>}, x^{<t>}] + b_o)$
- $h^{<t>} = o^{<t>} * tanh(c^{<t>})$

LSTMs con peep-hole connection, Gers & Schmidhuber (2000):

- $\tilde{c}^{<t>} = tanh(W_c * [h^{<t-1>}, x^{<t>}] + b_c)$
- $i^{<t>} = \sigma(W_i * [h^{<t-1>}, c^{<t-1>}] + b_i)$
- $f^{<t>} = \sigma(W_f * [h^{<t-1>}, c^{<t-1>}] + b_f)$
- $c^{<t>} = f^{<t>} \circledast c^{<t-1>} + i^{<t>} \circledast \tilde{c}^{<t>}$
- $o^{<t>} = \sigma(W_o * [h^{<t-1>}, c^{<t-1>}] + b_o)$
- $h^{<t>} = o^{<t>} * tanh(c^{<t>})$

Las compuertas consideran el estado de la memoria anterior

GRU VS LSTM

GRUs:

- $\tilde{h}^{<t>} = \tanh(W_{\tilde{h}} * (r^{<t>} * [h^{<t-1>}, x^{<t>}]) + b_{\tilde{h}})$
- $r^{<t>} = \sigma(W_r * [h^{<t-1>}, x^{<t>}] + b_r)$
- $z^{<t>} = \sigma(W_z * [h^{<t-1>}, x^{<t>}] + b_{\tilde{h}})$
- $h^{<t>} = (1 - z^{<t>}) \circledast h^{<t-1>} + z^{<t>} \circledast \tilde{h}^{<t>}$

LSTM:

- $\tilde{c}^{<t>} = tanh(W_c * [h^{<t-1>}, x^{<t>}] + b_c)$
- $i^{<t>} = \sigma(W_i * [h^{<t-1>}, x^{<t>}] + b_i)$
- $f^{<t>} = \sigma(W_f * [h^{<t-1>}, x^{<t>}] + b_f)$
- $c^{<t>} = f^{<t>} \circledast c^{<t-1>} + i^{<t>} \circledast \tilde{c}^{<t>}$
- $o^{<t>} = \sigma(W_o * [h^{<t-1>}, x^{<t>}] + b_o)$
- $h^{<t>} = o^{<t>} * tanh(c^{<t>})$

UNIVERSIDAD ICESI

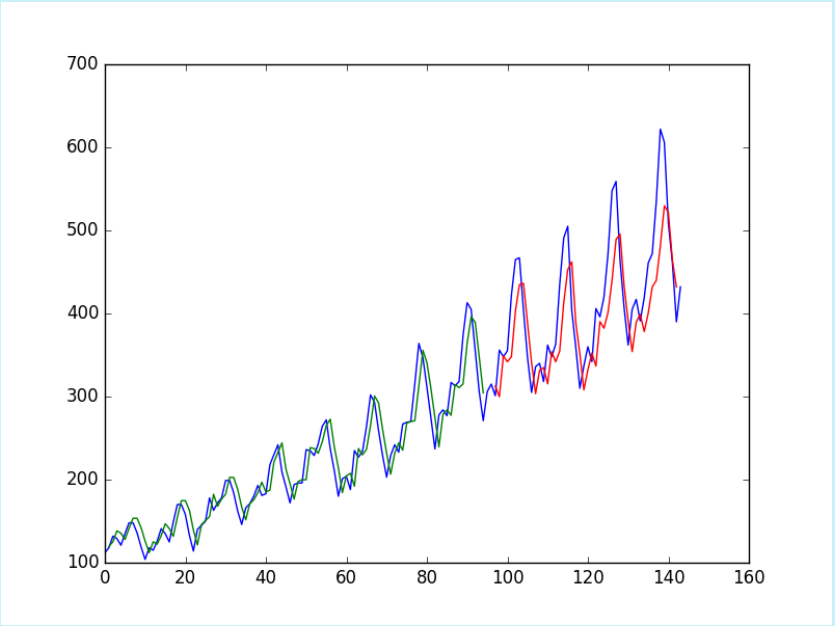
GRU VS LSTM

Consideración

- Primero fueron las LSTMs (1997) que las GRUs (2014)
- Ninguno de los 2 es mejor que el otro,
  - Se han hecho estudios comparativos que no hay una mejor que la otra; en ciertos casos puede que GRU sea mejor que LSTM, en otros se puede encontrar el contrario.
  - Los GRUs son más simples y se escalan más fácilmente
  - Los LSTMs son más flexibles al tener 2 compuertas de memoria en vez de 1
  - Los LSTMs son más usados, al ser más antiguos

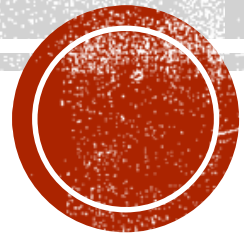
FORECASTING CON LSTM

Utilizar una red recurrente LSTM para predecir el número de pasajeros aéreos futuros.



UNIVERSIDAD ICESI

GRACIAS



UNIVERSIDAD ICESI

REFERENCIAS

- *Learning TensorFlow*, Tom Hope, Yehezkel S. Resheff & Itay Lieder, O'Reilly 2017
- *Introduction to TensorFlow*, Chris Manning & Richard Socher, Lecture 7 of the CS224n course at Stanford University, 2017
- *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, Aurélien Géron, 2017
- *Machine learning with TensorFlow*, Nishant Shukla, Manning, 2018
- *Python Machine Learning (2nd ed.)*, Sebastian Raschka & Vahid Mirjalili, Packt, 2017
- *TensorFlow for Deep Learning*, Charath Ramsundar & Reza Bosagh Zadeh, O'Reilly, 2018
- *Deep Learning with Python*, Francois Chollet, Manning 2018
- *Neural Networks and Deep Learning,* Andrew Ng, Coursera, 2017