

RNNs y LSTM

Sihuinta Perez Luis Armando
Bustamante Torres Luis Angel
Silva Corrales Brian Emanuel

Contenido I

- 1 Introducción
- 2 Posibles soluciones
- 3 Introducción a las RNN
- 4 Arquitectura tradicional de una RNN
- 5 Funciones de activacion mas usadas
- 6 Tipos de RNN
- 7 Arquitecturas de RNN
- 8 Problemas con las RNN
- 9 Introducción a LSTM

Contenido II

- 10 Arquitectura de una LSTM
- 11 Tipos de LSTM
- 12 Tabla comparativa
- 13 Gated Recurrent Unit - GRU
- 14 Limitaciones de las LSTM

“Se refiere como información secuencial a una serie de elementos que se suceden unos a otros y guardan relación entre sí”

Oxford Languages

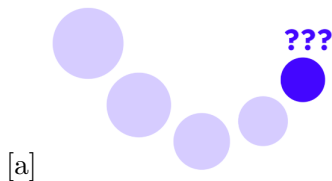
Esta tipología está presente en casi cualquier problema de la vida.

- Reconocimiento de actividad en vídeo (Saltar, correr, hablar)
- Análisis de secuencia de ADN
- Generación de música, dialogo, etc.
- Reconocimiento de voz
- Análisis de sentimiento
- Traducción de texto
- Natural language processing

¿Por qué no utilizar modelos de redes neuronales normales?

Las redes neuronales estándar están diseñadas para tener un tamaño de input y de output específico. A comparación de entradas de información secuencial estos pueden tener tamaños diferentes dependiendo de la observación. Como lo es una traducción de 5 palabras o de un párrafo. No comparte las características entre diferentes posiciones.

En el modelo MLP no se comparten parámetros



Las entradas no son independientes las unas de las otras, el modelo MLP no toma en cuenta el vínculo entre las entradas.

- Utilizar arquitecturas neuronales que son utilizadas junto a información secuencial.
- Procesar entradas y salidas de cualquier longitud, sin que estas longitudes afecten al modelo.
- Acceder a la información de estados más antiguos

Introducción a las RNN

Las redes neuronales recurrentes son diseñadas para tomar series de entradas sin un límite predeterminado de tamaño.

Estas *recuerdan el pasado y sus decisiones son influenciadas por lo que se aprendió del pasado*".

No solo aprenden durante el entrenamiento si no que también recuerdan cosas aprendidas anteriores mientras generan salidas, así que la misma entrada podría producir una salida diferente dependiendo a las entradas anteriores.

Introducción a las RNN

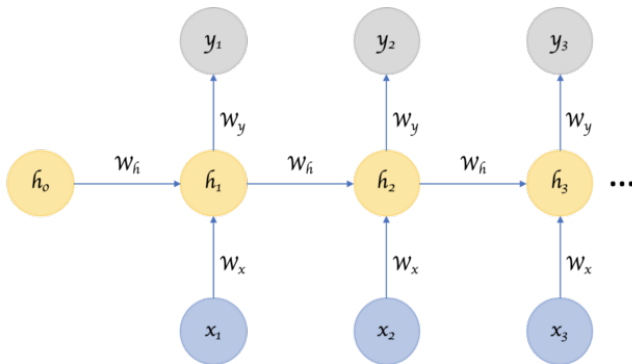
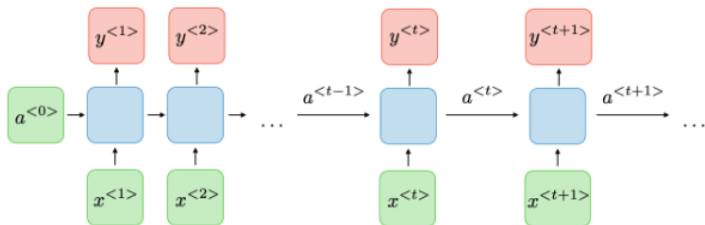
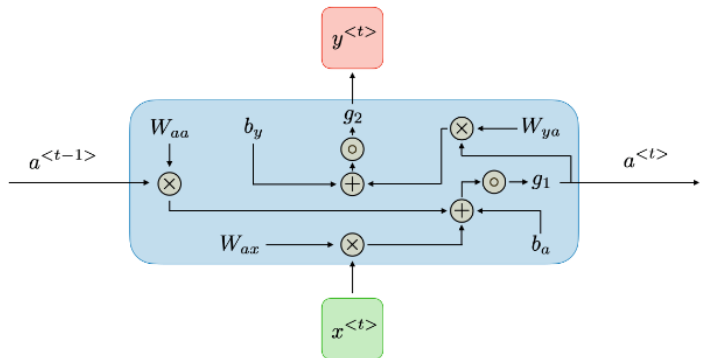


Figura 1: RNN con un estado oculto el cual transporta información de un elemento de entrada en la serie de otros

Arquitectura tradicional de una RNN



Arquitectura tradicional de una RNN



$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \quad \text{and} \quad y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

Arquitectura tradicional de una RNN

$$h_t = f_a(\underbrace{W_x X_t}_{\text{Actual}} + \underbrace{W_h h_{t-1}}_{\text{Pasado}} + b_h)$$

$$y_t = f_a(W_t h_t + b_t)$$

La **función de pérdida** **L** de todos los pasos de tiempo se define en función de la pérdida en cada paso de tiempo de la siguiente manera:

$$L(\hat{y}, y) = \frac{1}{T} \sum_{i=1}^T L(\hat{y}_i, y_i)$$

$$\frac{\partial L}{\partial W_h} = \frac{\partial}{\partial W_h} \left(\frac{1}{T} \sum_{i=1}^T L_i \right)$$

Retro propagación a través del tiempo: La retro propagación se realiza en cada punto de tiempo. En el tiempo T , la derivada de la pérdida L con respecto a la matriz de peso W se expresa como sigue:

$$\frac{\partial L_T}{\partial W_h} = \frac{\partial L_T}{\partial y_T} \frac{\partial y_T}{\partial h_T} \frac{\partial h_T}{\partial W_h}$$

Arquitectura tradicional de una RNN

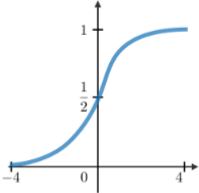
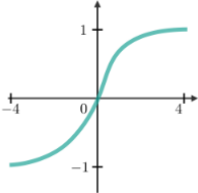
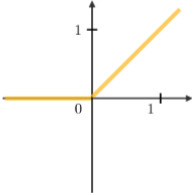
$$\underbrace{\frac{\partial h_T}{\partial W_h}}_{a_t} = \underbrace{\frac{\partial f_h(W_x X_t + W_h h_{t-1})}{\partial W_h}}_{b_t} + \underbrace{\frac{\partial f_h(W_x X_t + W_h h_{t-1})}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial W_h}}_{c_t a_{t-1}}$$

$$a_t = b_t + \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^t c_j \right) b_i$$

$$a_t = b_t + \sum_{i=1}^{t-1} \underbrace{\left(\prod_{j=i+1}^t \frac{\partial f_h(W_x X_t + W_h h_{t-1})}{\partial h_{t-1}} \right)}_{>1.1 \text{ Exploding Gradient, } < 0.9 \text{ Vanishing Gradient}} b_i$$

>1.1 Exploding Gradient, < 0.9 Vanishing Gradient

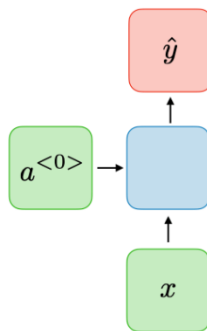
Funciones de activación más usadas

| Sigmoid | Tanh | RELU |
|---|---|--|
| $g(z) = \frac{1}{1 + e^{-z}}$ | $g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ | $g(z) = \max(0, z)$ |
|  |  |  |

Tipos de RNN

One to one

One-to-one
 $T_x = T_y = 1$

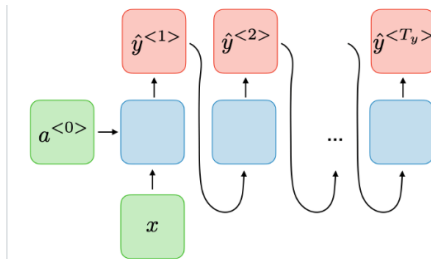


Típica red neuronal

One to many

Una entrada con muchas salidas.

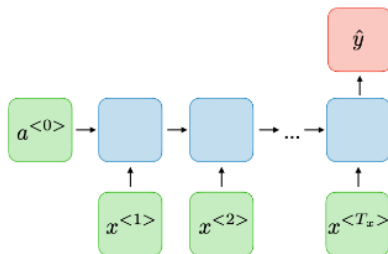
One-to-many
 $T_x = 1, T_y > 1$



Many to one

Muchas entradas con una única salida.

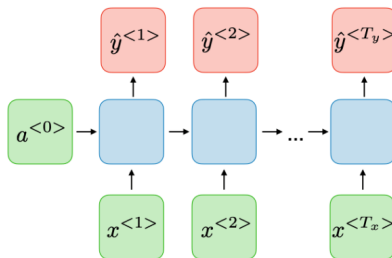
Many-to-one
 $T_x > 1, T_y = 1$



Many to many

Cada entrada tiene una respectiva salida.

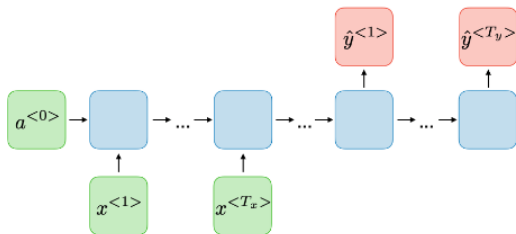
Many-to-many
 $T_x = T_y$



Many to many

Existen muchas entradas y muchas salidas, pero no son iguales.

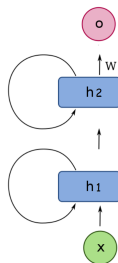
Many-to-many
 $T_x \neq T_y$



Arquitecturas de RNN

Deep (DRNN)

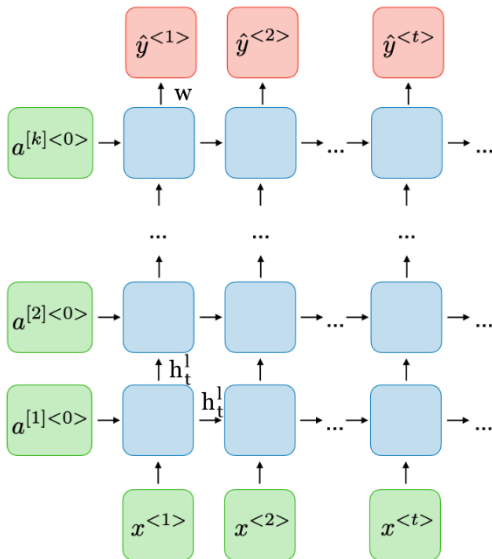
Es posible concatenar los bloques de construcción que hemos visto anteriormente para crear una red más larga. La siguiente imagen muestra la representación de lo que sería una red con dos capas RNN.



No es común concatenar muchos de estos bloques RNN ya que convertiría la red demasiado exigente computacionalmente para ser utilizada en la práctica. Las arquitecturas tienden a usar hasta 3 capas RNN concatenadas, pero es raro usar mucho más que eso.

Deep (DRNN)

DRNN extendida



Cada **estado oculto** (h_t) se pasa continuamente tanto al siguiente **paso de tiempo** ($t + 1$) de la **capa actual** (l) como al paso de **tiempo actual** (t) de la **siguiente capa** ($l + 1$)

$$h_t^l = f_a(\underbrace{W_x^l h_t^{l-1}}_{\text{Actual}} + \underbrace{W_h^l h_{t-1}^l}_{\text{Pasado}} + b_h^l)$$

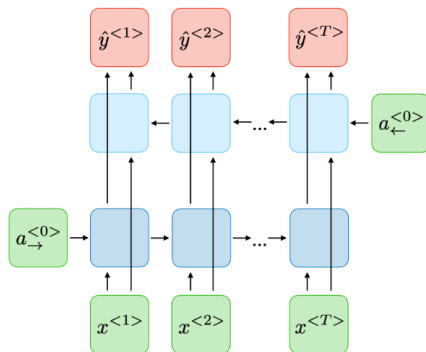
l :Representa a la capa de la RNN

Cuando se tienen los estados ocultos de una capa, si es necesario se puede generar una **salida final** y_t para ese tiempo.

$$y_t = f_a(W_y h_t^l + b_t)$$

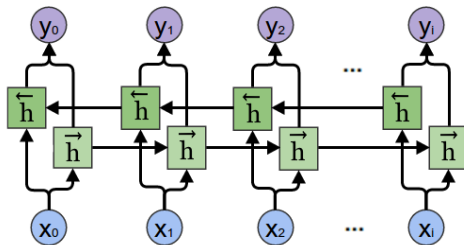
Bidirectional (BRNN)

Un BRNN es una combinación de dos RNN: un RNN se mueve hacia adelante, comenzando desde el inicio de la secuencia de datos, y el otro, se mueve hacia atrás, comenzando desde el final de la secuencia de datos.



Bidirectional (BRNN)

Se considera todas las secuencias de entrada disponibles tanto en el pasado como en el futuro para la estimación del vector de salida.



El estado oculto en el momento t (h_t) viene dado por una combinación de \overrightarrow{h}_t (**forward**) y \overleftarrow{h}_t (**backward**).

Bidirectional (BRNN)

$$h_t \text{ (Forward)} = \vec{h}_t = f_a(W_x^{\rightarrow} x_t + W_h^{\rightarrow} h_{t-1}^{\rightarrow} + b_h^{\rightarrow})$$

$$h_t \text{ (Backward)} = \overleftarrow{h}_t = f_a(W_x^{\leftarrow} x_t + W_h^{\leftarrow} h_{t-1}^{\leftarrow} + b_h^{\leftarrow})$$

$$y_t = f_a(W_{\vec{h}_t}^{\rightarrow} \vec{h}_t + W_{\overleftarrow{h}_t}^{\leftarrow} \overleftarrow{h}_t + b_y)$$

BRNN es útil para las siguientes aplicaciones:

- Reconocimiento de escritura a mano
- Reconocimiento de voz
- Análisis de dependencias
- Procesamiento del lenguaje natural
- Análisis de sentimientos

Explosión o desvanecimiento de gradientes. La gradiente es la derivada de la función de pérdida con respecto a los pesos, usada para minimizar la función de pérdida durante el back propagation.

Problemas con las RNN

El desvanecimiento de gradiente ocurre con las funciones de activación *sigmoidea* y *tanh*, este problema se puede evadir con una función de activación *relu* porque la gradiente es 0 para entradas cero y negativas, y 1 para entradas positivas.

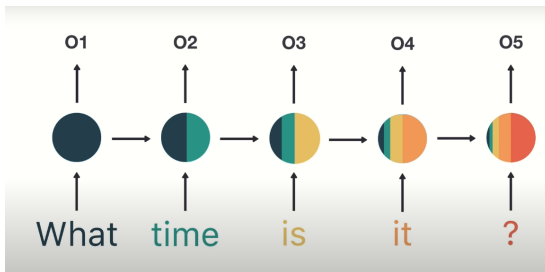


Figura 2: Chat bot clasificador

Problemas con las RNN

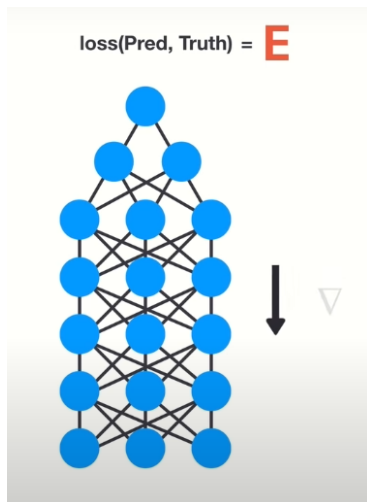


Figura 3: Back propagation

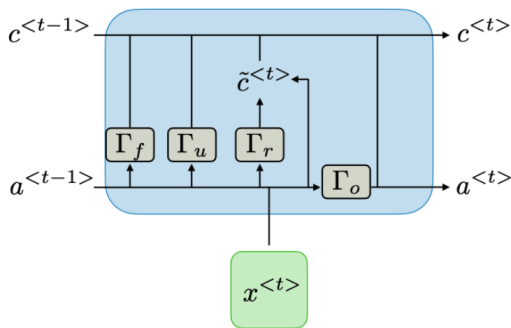
- Procedimientos de entrenamiento lentos y complejos
- Dificultad para procesar secuencias más largas

LSTM

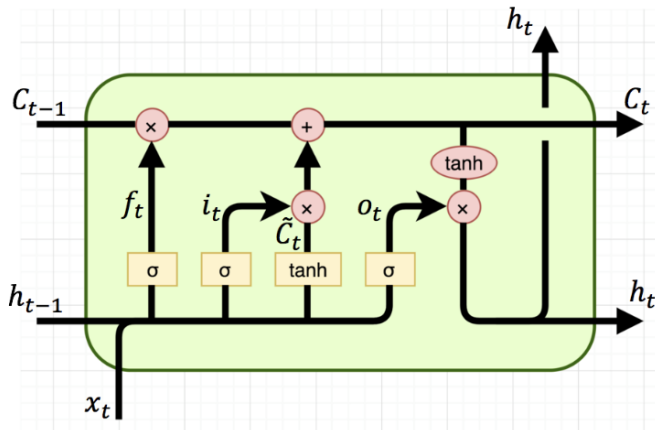
Long Short Term Memory

Introducción a LSTM

Hochreiter and Schmidhuber [1997] Introdujeron el modelo LSTM para superar el problema de las gradientes que se desvanecían. Este modelo es parecido a las redes neuronales recurrentes estándar con una capa oculta, pero cada nodo en la capa oculta se reemplaza por una celda de memoria. Una de sus principales características es la presencia de puertas cuya función es mantener información significativa u olvidar datos inútiles.

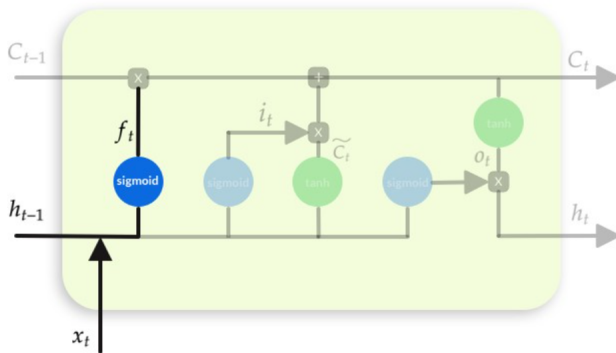


Arquitectura de una LSTM



Forget gate

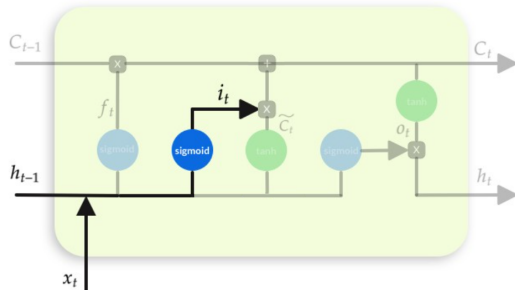
Forget gate determina qué información no es relevante y no debe considerarse, los valores cercanos a 0 se considerarán información a descartar y los valores cercanos a 1 se consideran información útil que se debe conservar.



$$\Gamma_f = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

Input gate

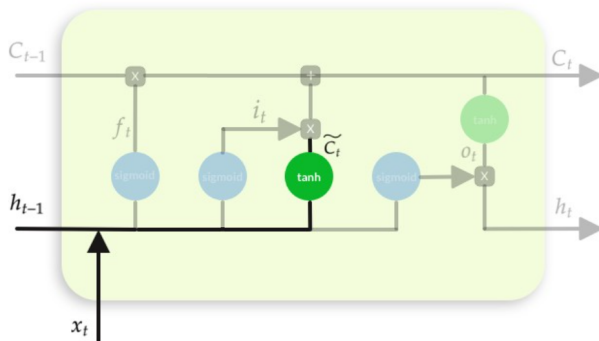
La puerta de entrada cuantifica la importancia de la nueva información transportada por la entrada \tilde{C}_t para formar parte del estado de la celda (la memoria del LSTM)



$$\Gamma_i = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

Relevance gate o \tilde{C}

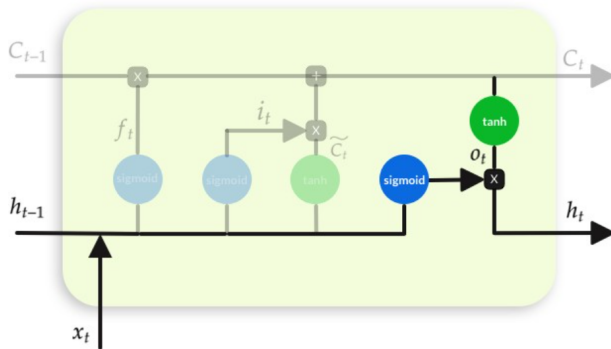
Representa un vector de información relevante entre la información actual y la anterior, sus valores van desde -1 hasta 1 por lo que ayudan a regular la red. Al multiplicarse con Γ_i se agrega al estado de la celda.



$$\Gamma_{\tilde{C}} = \Gamma_r = \tanh(W_r x_t + U_r h_{t-1} + b_r)$$

Output gate

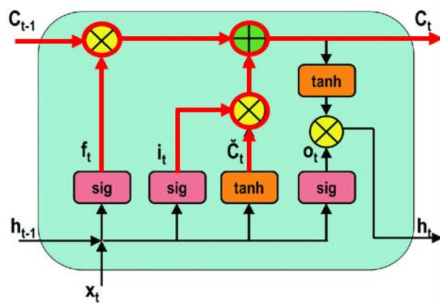
Puerta de salida que determina la información relevante que será usada para calcular el nuevo estado oculto H_t



$$\Gamma_o = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

Arquitectura de una LSTM

El estado de la celda (C_t) se conoce como la memoria del LSTM, se actualiza mediante la puerta de olvido y la puerta de entrada



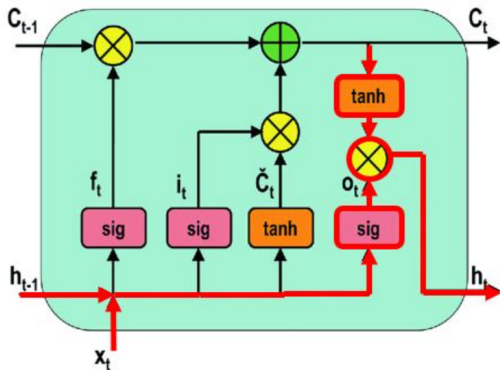
$$C_t = \underbrace{\Gamma_f C_{t-1}}_{\text{Información pasada}} + \underbrace{\Gamma_i \Gamma_{\tilde{C}}}_{\text{Información nueva}}$$

Si: $\Gamma_f = 0$, Olvida todo

Si: $\Gamma_f = 1$, No olvida nada

Arquitectura de una LSTM

El nuevo estado oculto (h_t) se obtiene a partir de la salida (o_t) y aplicando la función \tanh al estado de celda actual (C_t).



$$h_t = \Gamma_o \tanh(C_t)$$

Arquitectura de una LSTM

Una lstm al manejar un vector de estado interno cuyos valores aumentan o disminuyen cuando agregamos la salida de una función de activación.

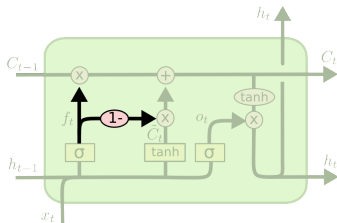
- Sigmoid: Siempre incrementa, ya que sus valores son positivos
- Tanh: Sus valores pueden ser positivos o negativos pudiendo aumentar o disminuir sus valores.

Es por esto que se usa una función Tanh para determinar los valores a agregar, además de la función sigmoide la cual su uso se justifica por el *gating* el cual ayuda a la red a actualizar la información que se debe olvidar.

Variantes de LSTM

Coupled forget and input gates

Esta variación utiliza la idea de que solo olvidaremos algo cuando haya nueva información para reemplazar la información olvidada.

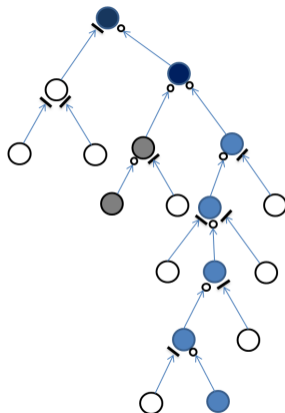


$$C_t = f_t * C_{t-1} + (1 - f_t) * \tilde{C}_t$$

Una LSTM se extiende a estructuras de árbol, en las que una celda de memoria puede reflejar los recuerdos históricos de varias celdas secundarias y, por lo tanto, de varias celdas descendientes en una estructura jerárquica.

S-LSTM

El nodo raíz considera información de las interacciones que se encuentran a larga distancia. Los círculos indican el pase de información y las líneas el bloqueo de esta. En modelos reales se aplica una versión de *gating* con el rango de 0 a 1 y comúnmente utilizado con una función logística sigmoide.



Bloque de memoria

Cada nodo se compone de un bloque de memoria S-LSTM, cada uno se compone de:

- 1 input gate
- 1 output gate
- N forget gates(Depende de la estructura) por ejemplo un nodo con 2 hijos tendría 2 forget gates.
- \otimes Producto Hadamard
- S Función(Tanh)
- Hidden vectors h_{t-1}^*
- Cell vectors c_{t-1}^*

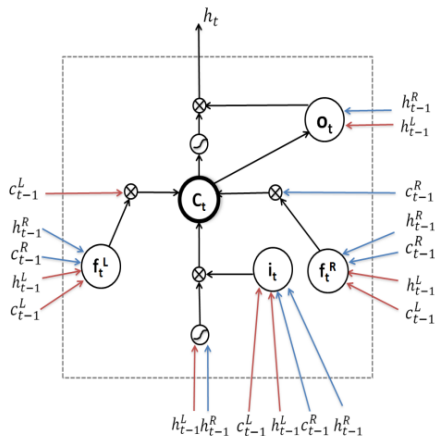


Figura 5: Estructura del bloque de memoria

$$i_t = \sigma(W_{hi}^L h_{t-1}^L + W_{hi}^R h_{t-1}^R + W_{ci}^L c_{t-1}^L + W_{ci}^R c_{t-1}^R + b_i)$$

$$f_t^L = \sigma(W_{hf_l}^L h_{t-1}^L + W_{hf_l}^R h_{t-1}^R + W_{cf_l}^L c_{t-1}^L + W_{cf_l}^R c_{t-1}^R + b_{f_l})$$

$$f_t^R = \sigma(W_{hf_r}^L h_{t-1}^L + W_{hf_r}^R h_{t-1}^R + W_{cf_r}^L c_{t-1}^L + W_{cf_r}^R c_{t-1}^R + b_{f_r})$$

$$x_t = W_{hx}^L h_{t-1}^L + W_{hx}^R h_{t-1}^R + b_x$$

$$c_t = f_t^L \otimes c_{t-1}^L + f_t^R \otimes c_{t-1}^R + i_t \otimes \tanh(x_t)$$

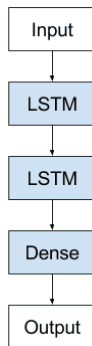
$$o_t = \sigma(W_{ho}^L h_{t-1}^L + W_{ho}^R h_{t-1}^R + W_{co} c_t + b_o)$$

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

Es una extensión del modelo LSTM que tiene múltiples capas LSTM ocultas donde cada capa contiene múltiples celdas de memoria. Dado que los LSTM operan con datos secuenciales, significa que la adición de capas agrega niveles de abstracción de las observaciones de entrada a lo largo del tiempo. En efecto, fragmentando las observaciones a lo largo del tiempo o representando el problema en diferentes escalas de tiempo.

Stacked LSTM

La arquitectura se puede definir como un modelo compuesto por múltiples capas LSTM. Una capa LSTM superior proporciona una **salida de secuencia** en lugar de una salida de valor único a la capa LSTM inferior. Específicamente, una salida por paso de tiempo de entrada, en lugar de un paso de tiempo de salida para todos los pasos de tiempo de entrada. Donde "*Dense*" es un perceptrón multicapa.



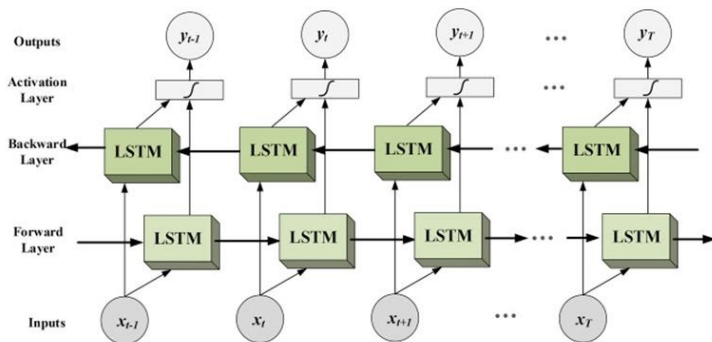
$$h_t^l = f_H(W_{IH}h_t^{l-1} + W_{HH}h_{t-1}^l + b_h^l)$$

Una capa oculta l en un stack de L LSTM utilizando la hidden layer, donde la secuencia del hidden vector se calcula sobre el tiempo $t = (1, \dots, T)$ para $l = (1, \dots, L)$

En una LSTM estándar solo se considera el contexto previo de datos para el entrenamiento. Si bien simplemente mirar el contexto anterior es suficiente para la mayoría de aplicaciones, puede ser también importante conocer el contexto futuro, se ha intentado el uso de información futura como contexto para la predicción actual en la arquitectura básica de RNN al retrasar la salida en un cierto número de pasos de tiempo. Sin embargo, este método requiere que se elija un retraso óptimo cuidadosamente seleccionado para cualquier implementación.

Bidirectional LSTM

Nuestro input es usado para entrenar dos LSTM en lugar de uno solo, de esta manera la memoria fluye en dos direcciones permitiendo preservar la información futura y pasada.



Multiplicative LSTM

Se basa en la idea de una mRNN(Multiplicative RNN) que utiliza una matriz de transición factorizada para los pesos.

Las arquitecturas mRNN y LSTM no son excluyentes, por lo que se pueden combinar agregando conexiones desde el estado intermedio m_t^* de mRNN a cada gate, lo que da como resultado el siguiente sistema:

$$m_t = (W_{mx}x_t) \odot (W_{mh}h_{t-1}) \quad (17)$$

$$\hat{h}_t = W_{hx}x_t + W_{hm}m_t \quad (18)$$

$$i_t = \sigma(W_{ix}x_t + W_{im}m_t) \quad (19)$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_t) \quad (20)$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_t). \quad (21)$$

El modelo estándar de LSTM es efectivo en tareas de aprendizaje de secuencias unidimensionales, su desempeño es alto como tareas de reconocimiento de voz o reconocimiento de escritura, etc. Pero muchas áreas poseen datos que no son unidimensionales.

- Procesamiento de video.
- Reconocimiento facial.
- Procesamiento de imágenes médicas.

Multidimensional LSTM

Aquí las entradas no están dispuestas en una secuencia, sino en una cuadrícula de n -dimensiones.

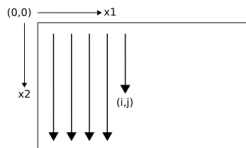


Figure 3: 2D sequence ordering. The MDRNN forward pass starts at the origin and follows the direction of the arrows. The point (i,j) is never reached before both $(i-1,j)$ and $(i,j-1)$.

Multidimensional LSTM (MDLSTM)

Para cada input x se reciben un total de N vectores ocultos (h_1, h_2, \dots, h_N) y la misma cantidad de vectores de memoria (m_1, m_2, \dots, m_N) generando un total de N puertas de olvido. Las entradas son computadas y generan un nuevo vector oculto y de memoria los cuales serán pasados al siguiente estado por cada dimensión.

A pesar de que MDLSTM sea capaz de leer datos multidimensionales este, debido a su crecimiento combinatorio se vuelve inestable a medida que el tamaño de la cuadrícula y la dimensionalidad crecen.

Grid LSTM modela secuencias multidimensionales con un aumento del tamaño de la cuadrícula.

Los LSTM han mostrado una mejor capacidad de aprendizaje en la comprensión de las dependencias secuenciales a largo plazo. Sin embargo, se ha argumentado que sus mecanismos de activación no tienen forma de discriminar de manera exhaustiva entre la información destacada y no destacada en una secuencia.

Differential RNN

Por lo tanto, los LSTM no logran capturar patrones dinámicos espacio-temporales en tareas como el reconocimiento de acciones, en los que las secuencias pueden contener muchos cuadros irrelevantes.



push



pushup



ride
bike



ride
horse



run



shake
hands



shoot
ball



shoot
bow



shoot
gun



sit



situp



smile



smoke



somersault

Las DRNN se refieren a la detección y captura de secuencias espacio-temporales para aprender el dinamismo de las acciones de entrada. Se monitorea la alternancia en la ganancia de información de data importante entre frames sucesivos.

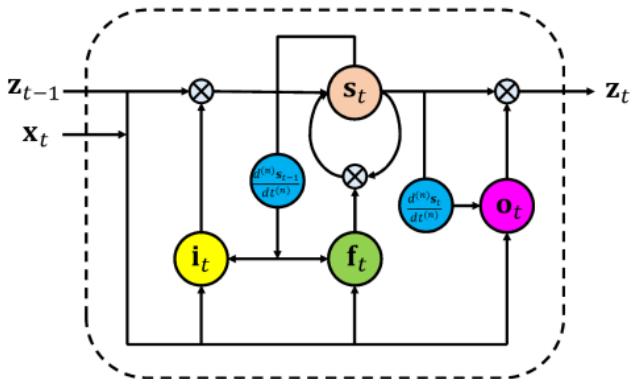
Este cambio de información se detecta calculando la derivada de los hidden states(DoS). Se utiliza back propagation truncado para prevenir explosiones o desvanecimiento de gradiente.

Así que el DoS $\frac{ds_t}{dt}$ cuantifica el cambio de información a cada tiempo t , mientras sea más grande la magnitud del DoS es un indicador de que la información que contiene es sobresaliente y es causada por un abrupto cambio de acción, de lo contrario sería información no sobresaliente o relevante no tendría muchos cambios.

Por lo tanto:

- A mayor magnitud de DoS las gates permiten mayor información para la célula de memoria.
- A menor magnitud de DoS las gates permiten menor información para la célula de memoria.
- Input y Forget gate se controlan por $\frac{ds^{(n)}_{t-1}}{dt^{(n)}}$
- Output gate se controla por $\frac{ds^{(n)}_t}{dt^{(n)}}$

Differential RNN



También se puede involucrar derivadas de orden superior para detectar más patrones de acciones, como por ejemplo en un video con un objeto en movimiento el primer orden captura la velocidad y el segundo orden captura la aceleración.

$$\frac{ds^n s_t}{dt^n} | n \geq 2$$

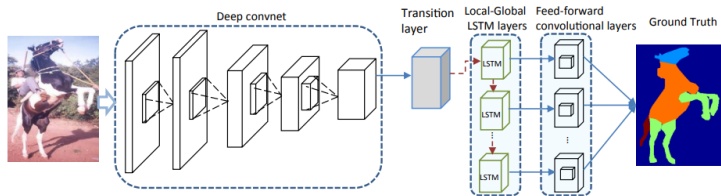
Propuesta para el análisis de objetos semánticos (todo aquello sobre lo que un sujeto puede llevar a cabo una acción), ya que es una tarea fundamental para comprender a los objetos en la comunidad de visión computacional donde incorporar información contextual multinivel es necesario para lograr un reconocimiento preciso.

Local-Global LSTM - LG LSTM



Está inspirado en Grid LSTM. Las características de cada posición se basan en la información corta-local por lo cual la información se propaga por los pixeles vecinos y retiene el contexto, como por ejemplo los límites de los objetos, mientras que la información global obtenida del mapeo de características provee el contexto en largas distancias.

Local-Global LSTM - LG LSTM



Una imagen pasa por varias capas convolucionales para generar un mapeo de características. Luego pasa por la capa de transición y por stacked LG-LSTM, por último, teniendo este mapeo de características mejorado por LG-LSTM, la capa convolucional feed-forward produce el análisis final del objeto.

Matching LSTM

Se propuso inicialmente para la inferencia del lenguaje natural (NLI). (NLI) es la tarea de clasificar un par de oraciones de premisa e hipótesis en tres clases: contradicción, neutral e implicación. Por ejemplo.

| Premise | Hypothesis | Label |
|--|--|---------------|
| A man inspects the uniform of a figure in some East Asian country. | The man is sleeping. | contradiction |
| An older and younger man smiling. | Two men are smiling and laughing at the cats playing on the floor. | neutral |
| A soccer game with multiple males playing. | Some men are playing a sport. | entailment |

Matching LSTM

El mecanismo de coincidencia almacena (recuerda) los resultados críticos para la predicción final y olvida las coincidencias menos importantes. El último estado oculto del mLSTM es útil para predecir la relación entre la premisa y la hipótesis.

La diferencia con otros métodos es que, en lugar de incrustar la premisa y la hipótesis en una oración completa, el mLSTM realiza una comparación palabra por palabra de la hipótesis con la premisa.

Este LSTM permite poner más énfasis en resultados importantes de coincidencia a nivel de palabra identificando importantes desajustes que son críticos para predecir la etiqueta de contradicción o la etiqueta neutral.

Este modelo genera un resumen de la información espectral escaneando bandas de frecuencia con un LSTM de frecuencia (F - LSTM) de modo que la información de evolución de la frecuencia se resuma mediante la salida del F-LSTM.

Todas las salidas de F-LSTM pasaran como entrada para hacer el análisis del tiempo de manera normal en un LSTM de tiempo (T-LSTM).

Frequency-Time LSTM

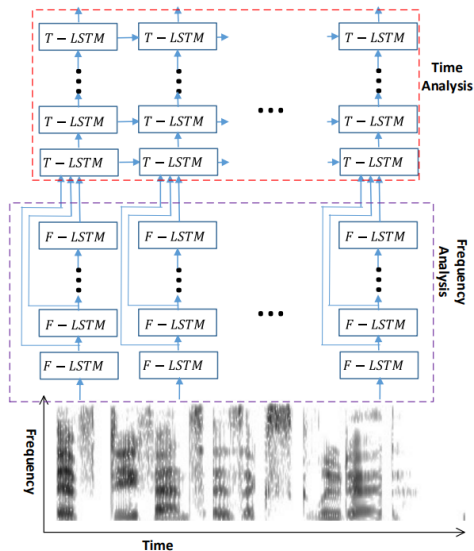
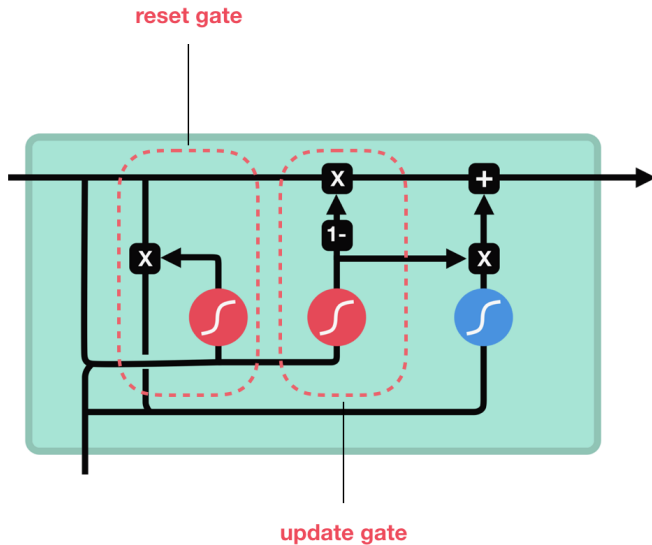


Tabla comparativa

| Variante | Ventajas | Desventajas |
|-----------------------|--|--|
| S-LSTM | Maneja entradas complicadas mejor que un LSTM | Mayor complejidad computacional que un LSTM |
| Stacked LSTM | Usa dependencias secuenciales de largo plazo por su arquitectura mas profunda | Mayor complejidad computacional y requerimiento de memoria por ser una pila |
| Bidireccional LSTM | Captura mejor el contexto pasado y futuro de la entrada mejor que un LSTM y S-LSTM | Mayor complejidad computacional por el aprendizaje en ambas direcciones |
| Multidimensional LSTM | Puede modelar secuencias de información multidimensional | -Mayor complejidad computacional y requerimiento de memoria por los múltiples hidden vectors. -Inestabilidad mientras que el tamaño y profundidad aumenta |
| Grid LSTM | Puede modelar secuencias de información multidimensional con tamaños de grid incrementado | Mayor complejidad computacional por conexiones recurrentes |
| Differential RNN | Discrimina la información en secuencia relevante de la que no Captura mejor patrones espacio-temporales | Mayor complejidad computacional a comparación de las LSTM por los operadores diferenciales |
| Local-Global LSTM | Aprovecha mejor la información contextual global y local | Mayor complejidad computacional por mayor número de parámetros para las representaciones globales y locales |
| Matching LSTM | Optimiza el LSTM para tareas de inferencia de lenguaje natural | Mayor complejidad computacional debido al matching de palabra por palabra de hipótesis y premisa |
| Frequency-Time LSTM | Usa tiempo y frecuencia | Mayor complejidad computacional por más parámetros para modelar tiempo y frecuencia |

Introducida por *Cho, et al. (2014)*., una Gru es parecida a una LSTM aunque también se puede considerar una variante con la diferencia de que una GRU tiene menos parámetros ya que presenta la ausencia de una output gate y algunos otros cambios.

Gated Recurrent Unit-GRU



La Update Gate actúa de manera similar a la unión de forget e input gate de un LSTM. Puede decidir cuánta información de pasos previos tiene que ser pasada al futuro. Esto indica que puede decidir si pasar toda la información del pasado, eliminando el riesgo de un desvanecimiento de gradiente.

$$Z_t \sigma(W^{(z)} x_t + U^{(z)} h_{t-1})$$

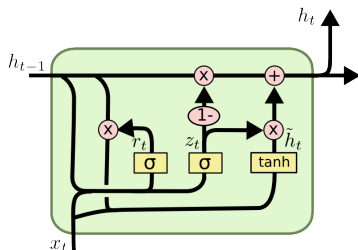
Donde Z_t es la puerta update, cuando x_t se conecta a la unidad se multiplica por su propio peso $W^{(z)}$, al igual que con h_{t-1} que contiene la información de las unidades anteriores $t - 1$. Ambos resultados se suman y se activa la función sigmoide para aplastar los valores entre 0 y 1.

La reset gate es otra puerta que se utiliza para decidir cuánta información pasada olvidar.

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

Esta fórmula es similar a la de Update gate, donde se multiplicaran el input por sus pesos, de la misma manera con h_{t-1} . Ambos resultados se suman y son aplastados con la función de activación sigmoide.

Gated Recurrent Unit-GRU



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

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

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

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

Limitaciones de las LSTM

- Aun se tiene una ruta secuencial desde las celdas pasadas más antiguas hasta la actual. De hecho, el camino ahora es aún más complicado, porque tiene puertas aditivas y de olvido adjuntas.
- Al poseer 3 compuertas, es computacionalmente más caro.
- No es amigable con el hardware ya que para el entrenamiento rápido de estas redes se necesitan una gran cantidad de recursos.