

Modelos de secuencias: aplicaciones

César Olivares

Pontificia Universidad Católica del Perú

Maestría en Informática

INF659 - Técnicas avanzadas de data mining y sistemas inteligentes

2017

Tipos de secuencias modeladas por RNNs

- Las RNNs pueden modelar secuencias de entrada y de salida, así como mapear secuencias a puntos de datos individuales (y viceversa).

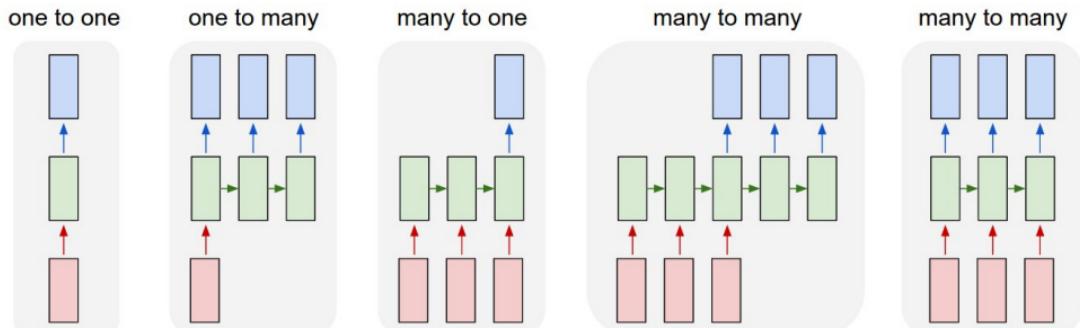
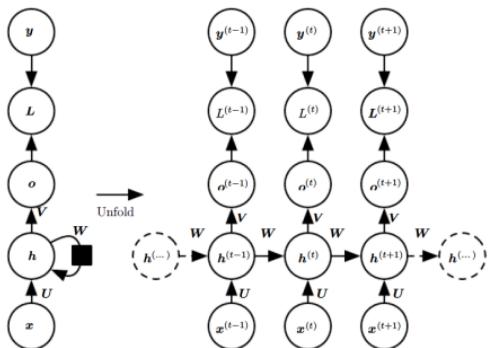


Figura 1: Modelos de secuencias modeladas por RNNs. Cada rectángulo es un vector de nodos. Cada flecha representa una función (p.ej. multiplicación de matrices). De izquierda a derecha: (1) Modelo de red feed-forward, no RNN. (p.ej. clasificación de imágenes). (2) Secuencia de salida (p.ej. anotación de imágenes). (3) Secuencia de entrada (p.ej. análisis de sentimiento). (4) Secuencias de entrada y salida (p.ej. traducción automática). (5) Secuencias sincronizadas (p.ej. clasificación de cada fotograma de video). (Karpathy 2015)

RNN «many-to-many» con conexiones «hidden-to-hidden»



$$\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W}h^{(t-1)} + \mathbf{Ux}^{(t)} \quad (1)$$

$$\mathbf{h}^{(t)} = \tanh(\mathbf{a}^{(t)}) \quad (2)$$

$$\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)} \quad (3)$$

$$\hat{y}^{(t)} = \text{softmax}(\mathbf{o}^{(t)}) \quad (4)$$

El reto de las dependencias de largo plazo

- En las RNNs, los gradientes propagados por muchos pasos de tiempo tienden a desvanecerse (la mayoría del tiempo) o explotar.
- Aún cuando esto no se da por completo, las dependencias de largo plazo reciben un peso exponencialmente menor que las dependencias de corto plazo.

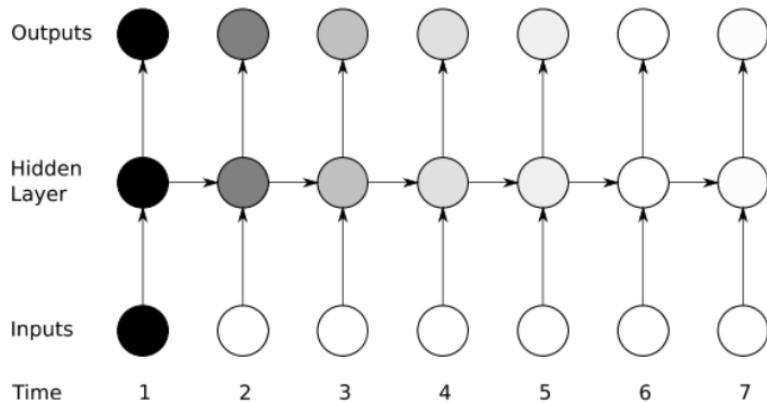


Figura 2: El problema de gradientes desvanecientes en las RNNs. La oscuridad de los nodos indica la sensibilidad a las entradas de $time = 1$ (mayor sensibilidad cuanto más oscuros). La sensibilidad decae en el tiempo conforme nuevas entradas sobrescriben la activación de la capa oculta y la red «olvida» las primeras entradas (Graves 2012)

RNNs con múltiples escalas de tiempo

- Lidiar con dependencias de largo plazo requiere operar en múltiples escalas de tiempo, con algunas partes del modelo atendiendo a detalles finos de corto alcance, mientras que otras atiendan a información más general desde el pasado distante.
- Algunas estrategias para lidiar con múltiples escalas de tiempo son:
 - Añadir conexiones diferidas luego de d pasos.
 - Emplear unidades permeables (*leaky units*) con autoconexiones lineales del tipo $\mathbf{h}^{(t)} \leftarrow \alpha\mathbf{h}^{(t-1)} + (1 - \alpha)\tanh(\mathbf{a}^{(t)})$
 - Reemplazar conexiones de distancia 1 por conexiones más largas.

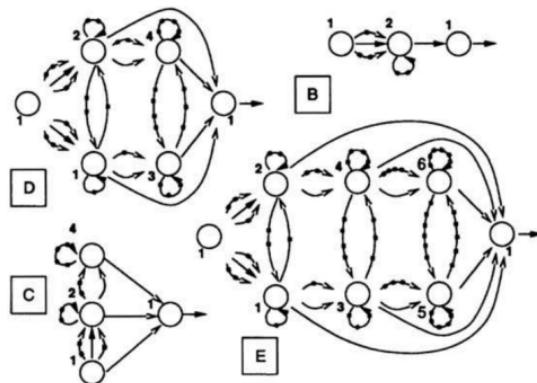


Figura 3: Ejemplo de arquitecturas con conexiones en múltiples escalas. Los pequeños cuadrados representan una dilación discreta, los números representan la escala de tiempo de las unidades. Las arquitecturas B a E tienen 2, 3, 4 y 6 escalas de tiempo respectivamente. (El Hihi & Bengio 1996)

RNNs con compuertas

- El uso de RNNs con compuertas generaliza la idea de las unidades permeables empleando pesos de conexión α variables en el tiempo.
- Adicionalmente, mientras que las unidades permeables permiten *acumular* información, las RNNs con compuertas añaden la capacidad de *olvidar* información que ya haya sido usada y no sea ya relevante.
- Los principales tipos de RNNs con compuertas son:
 - Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber 1997; Gers 2000)
 - Gated Recurrent Units (GRU) (Cho 2014; Chung 2014)

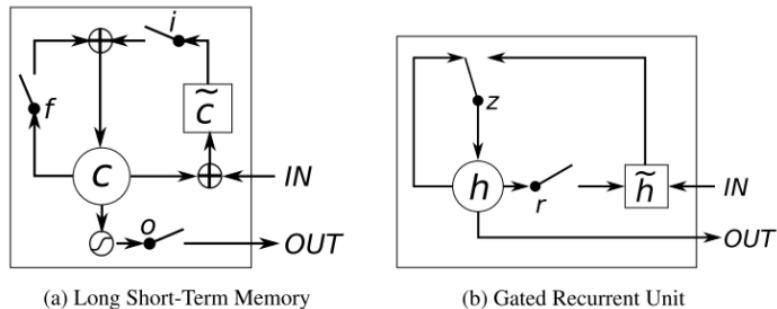


Figura 4: Long Short-Term Memory (LSTM) y Gated Recurrent Units (GRU). (Chung 2014)

Long Short-Term Memory (LSTM)

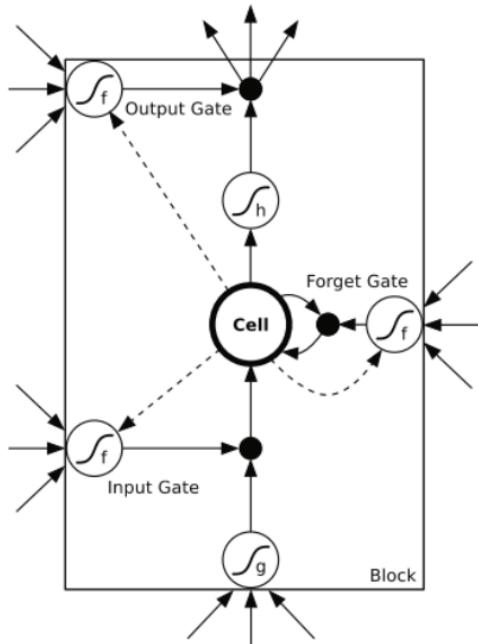


Figura 5: LSTM (Graves 2012)

- Salida (*output*):

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

- Unidad de estado (*state unit*):

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

$$\tilde{c}_t = \tanh(U_c x_t + W_c h_{t-1} + b_c)$$

- Compuertas de salida (o), entrada (i) y olvido (f):

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

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

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

LSTM para dependencias de largo plazo

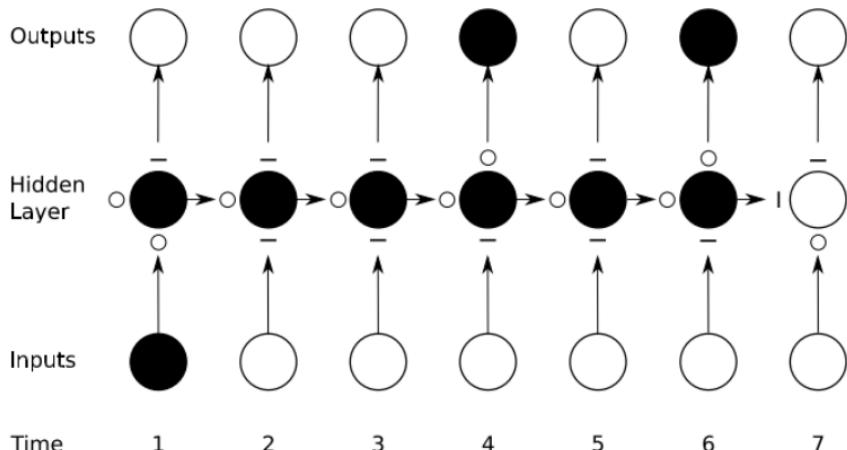


Figura 6: Conservación de la información del gradiente en las LSTM. La oscuridad de los nodos indica su sensibilidad a las entradas en cada paso. Los nodos negros tienen sensibilidad máxima y los nodos blancos son completamente insensibles. Se muestra el estado de las compuertas de entrada, olvido y salida por debajo, izquierda y arriba de la capa oculta, respectivamente. Por simplicidad, todas las compuertas están completamente abiertas ('o') o cerradas ('-'). La célula de memoria «recuerda» la primera entrada mientras la compuerta de olvido está abierta y la de entrada esté cerrada. La sensibilidad de la capa de salida puede ser controlada por la compuerta de salida sin afectar a la celda. (Graves 2012)

Comparación de las RNNs simples y LSTM

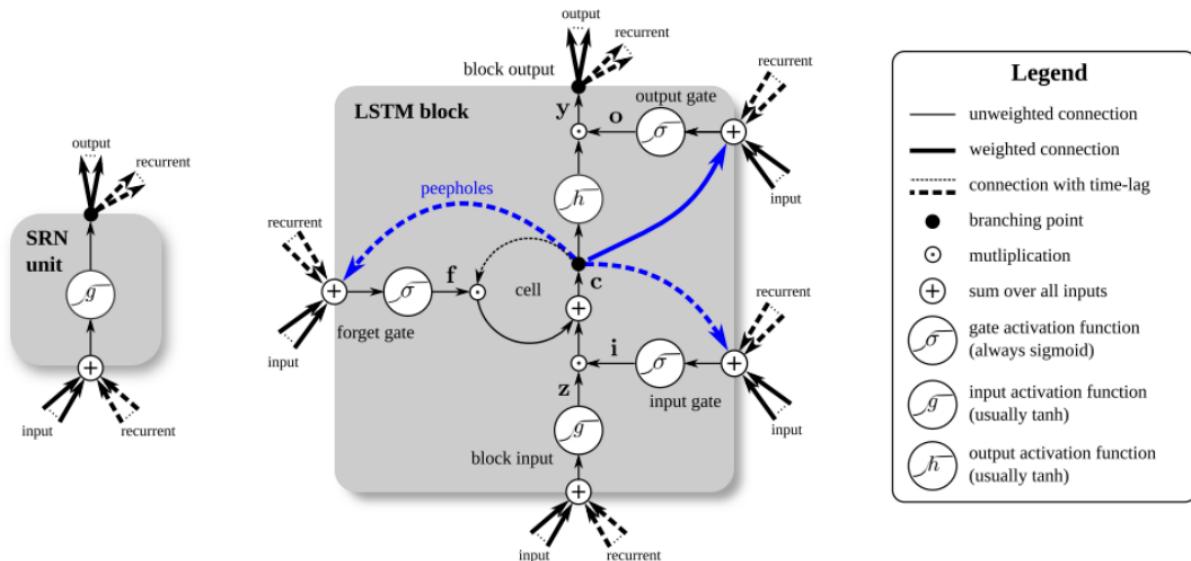


Figura 7: Esquema de una unidad recurrente simple (izquierda) y de un bloque LSTM (derecha) usados en la capa oculta de una RNN. (Greff 2015)

Gated Recurrent Units (GRU)

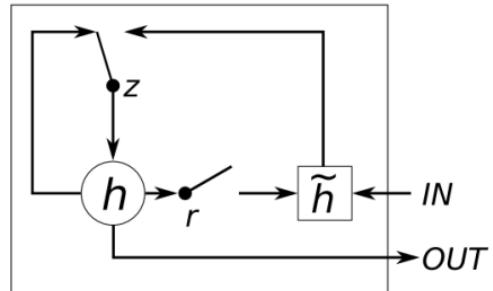


Figura 8: Gated Recurrent Units
(Chung 2014)

- Salida (*output*):

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

$$\tilde{h}_t = \tanh(Ux_t + W(r_t \odot h_{t-1}) + b)$$

- Compuertas de actualización (z) y restablecimiento (r):

$$z_t = \sigma(U_z x_t + W_z h_{t-1} + b_z)$$

$$r_t = \sigma(U_r x_t + W_r h_{t-1} + b_r)$$

Comparación de LSTM y GRU

Long Short-Term Memory (LSTM)

(Hochreiter & Schmidhuber 1997; Gers 2000)

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

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

$$\tilde{c}_t = \tanh(U_c x_t + W_c h_{t-1} + b_c)$$

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

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

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

Gated Recurrent Units (GRU)

(Cho 2014; Chung 2014)

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

$$\tilde{h}_t = \tanh(U_x x_t + W_h h_{t-1} + b)$$

$$z_t = \sigma(U_z x_t + W_z h_{t-1} + b_z)$$

$$r_t = \sigma(U_r x_t + W_r h_{t-1} + b_r)$$

- A diferencia de las RNN simples, tanto LSTM como GRU hacen una composición **aditiva** en la recurrencia de t a $t+1$.
- LSTM controla la **exposición del contenido** de su memoria; GRU, no.
- GRU controla independientemente de la compuerta de olvido la **cantidad de información del paso anterior** que es recibida; LSTM no.
- LSTM controla por separado las **compuertas de ingreso (i) y de olvido (f)**; GRU las controla con una sola compuerta de actualización (z).

Visualización de LSTM (1)

Cell sensitive to position in line:

```
The sole importance of the crossing of the Berezina lies in the fact  
that it plainly and indubitably proved the fallacy of all the plans for  
cutting off the enemy's retreat and the soundness of the only possible  
line of action--the one Kutuzov and the general mass of the army  
demanded--namely, simply to follow the enemy up. The French crowd fled  
at a continually increasing speed and all its energy was directed to  
reaching its goal. It fled like a wounded animal and it was impossible  
to block its path. This was shown not so much by the arrangements it  
made for crossing as by what took place at the bridges. When the bridges  
broke down, unarmed soldiers, people from Moscow and women with children  
who were with the French transport, all--carried on by vis inertiae--  
pressed forward into boats and into the ice-covered water and did not,  
surrender.
```

Cell that turns on inside quotes:

```
"You mean to imply that I have nothing to eat out of.... On the  
Contrary, I can supply you with everything even if you want to give  
dinner parties," warmly replied Chichagov, who tried by every word he  
spoke to prove his own rectitude and therefore imagined Kutuzov to be  
animated by the same desire.
```

```
Kutuzov, shrugging his shoulders, replied with his subtle penetrating  
smile: "I meant merely to say what I said."
```

Cell that robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,  
    siginfo_t *info)  
{  
    int sig = next_signal(pending, mask);  
    if (sig) {  
        if (current->notifier) {  
            if (sigismember(current->notifier_mask, sig)) {  
                if (!!(current->notifier)(current->notifier_data)) {  
                    clear_thread_flag(TIF_SIGPENDING);  
                    return 0;  
                }  
            }  
        }  
        collect_signal(sig, pending, info);  
    }  
    return sig;  
}
```

A large portion of cells are not easily interpretable. Here is a typical example:

```
/* unpack a filter field's string representation from user-space  
 * buffer. */  
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)  
{  
    char *str;  
    if (!*bufp || (len == 0) || (len > *remain))  
        return ERR_PTR(-EINVAL);  
    /* Of the currently implemented string fields, PATH_MAX  
     * defines the longest valid length.  
    */
```

Figura 9: Ejemplos de celdas LSTM con activaciones interpretables entrenadas con «Linux Kernel» y «War and Peace». El texto en color corresponde a $\tanh(c)$, donde -1 es rojo y +1 es azul. (Karpathy 2016)

Visualización de LSTM (2)

Cell that turns on inside comments and quotes:

```
/* Duplicate LSM field information. The lsm_rule is opaque, so
 * re-initialized... */
static inline int audit_dupe_lsm_field(struct audit_field *df,
    struct audit_field *sf)
{
    int ret = 0;
    char lsm_str[1];
    /* our own copy of lsm_str */
    lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
    if (unlikely(!lsm_str))
        return -ENOMEM;
    df->lsm_str = lsm_str;
    /* our own (refreshed) copy of lsm_rule */
    ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
        (void *)+1); /* don't return */
    /* Keep currently invalid fields around in case they
     * become valid after a policy reload. */
    if (ret == -EINVAL) {
        pr_warn("Audit rule for LSM '\\%s\\' is invalid\n",
            df->lsm_str);
        ret = 0;
    }
    return ret;
}
```

Cell that is sensitive to the depth of an expression:

```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
    int i;
    if (!classes[class]) {
        for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
            if (mask[i] & classes[class][i])
                return 0;
    }
    return 1;
}
```

Cell that might be helpful in predicting a new line. Note that it only turns on for some ")":

```
char *audit_unpack_string(void *bufp, size_t *remain, si
{
    char *str;
    if (!bufp || (*len == 0) || (*len > *remain))
        return ERR_PTR(-EINVAL);
    /* of the currently implemented string fields, PATH_MAX
     * defines the longest valid length. */
    if (*len > PATH_MAX)
        return ERR_PTR(-EAMETOOLONG);
    str = kmalloc(*len + 1, GFP_KERNEL);
    if (unlikely(!str))
        return ERR_PTR(-ENOMEM);
    memcpy(str, bufp, len);
    str[len] = 0;
    *bufp += len;
    *remain -= len;
    return str;
}
```

Figura 10: Ejemplos de celdas LSTM con activaciones interpretables entrenadas con «Linux Kernel» y «War and Peace». El texto en color corresponde a $\tanh(c)$, donde -1 es rojo y +1 es azul. (Karpathy 2016)

Mecanismos de atención y alineamiento

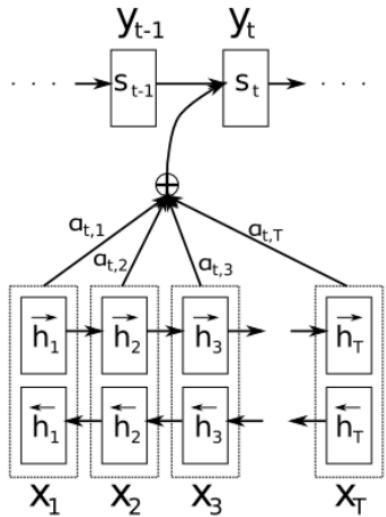


Figura 11: Mecanismo de atención. El contexto $c^{(t)}$ se calcula como un promedio ponderado de los vectores de características $h^{(t)}$ con los pesos $\alpha^{(t)}$. Los pesos $\alpha^{(t)}$ son usualmente calculados aplicando una función *softmax* a un vector de puntajes emitidos por el modelo. (Bahdanau 2014)

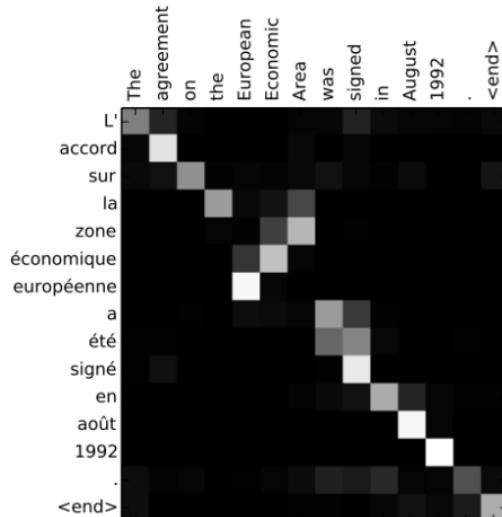


Figura 12: Ejemplo de alineamiento. Los ejes x y y corresponden a las palabras originales (inglés) y su traducción (francés). Cada pixel muestra el peso α_{ij} de la palabra original j para la palabra traducida i en escala de grises (0: negro, 1: blanco). (Bahdanau 2014)

RNNs con memoria explícita

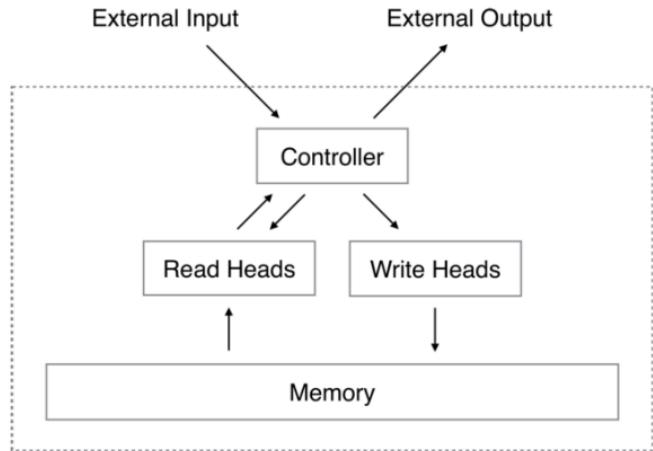


Figura 13: Arquitectura de la Neural Turing Machine Architecture. En cada ciclo, el controlador recibe entradas y emite salidas. Además lee y escribe de una matriz de memoria mediante un conjunto de cabezales de lectura y escritura paralela. La línea punteada indica la división entre el circuito NTM y el mundo exterior (Graves 2014)

Differentiable Neural Computer

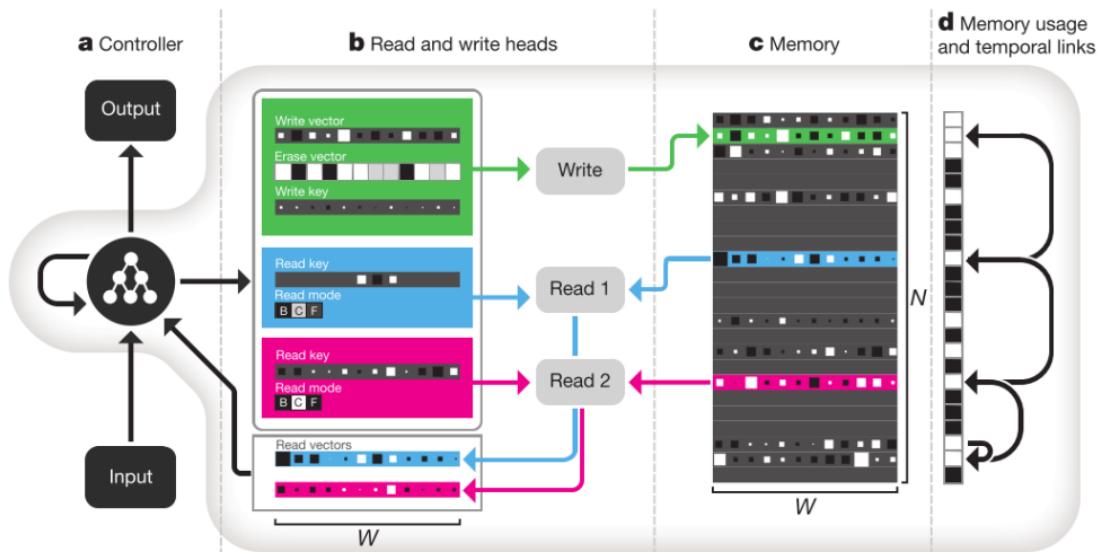
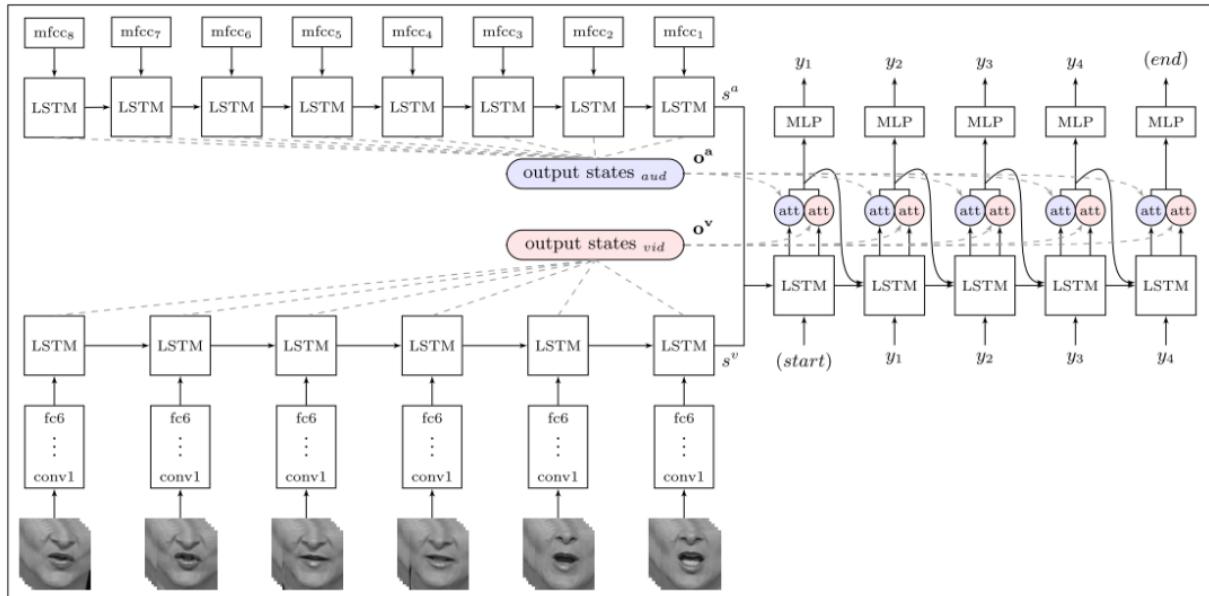


Figura 14: Arquitectura de la Differentiable Neural Computer. a. El controlador recurrente recibe entradas y produce salidas. A recurrent controller network receives input from an external data source and produces output. b, c. El controlador también emite vectores de parámetros para un cabezal de escritura (verde) y múltiples cabezales de lectura (dos en este caso, azul y rosado). El cabezal de escritura define vectores de escritura y borrado usados para editar la matriz de memoria $N \times W$. La clave de escritura ubica direcciones de memoria por editar y contribuye a definir pesos que se concentran en las filas o posiciones de la matriz de memoria en las que se escribirá. Los cabezales de escritura pueden usar compuertas (modos de lectura) para hacer una búsqueda basada en contenido ('C') o en ubicación en el orden de escritura ('F') o en sentido inverso ('B'). d. El vector de uso registra las ubicaciones que ya han sido usadas, y una matriz de enlaces temporales registra el orden en que se escribió las ubicaciones (representadas como flechas dirigidas). (Graves 2016)

Aplicaciones

from his travels it might have been
from his travels - it might have been

Lectura labial



Chung, J. S., Senior, A., Vinyals, O., & Zisserman, A. (2016). Lip reading sentences in the wild.

Procesamiento secuencial de datos no secuenciales

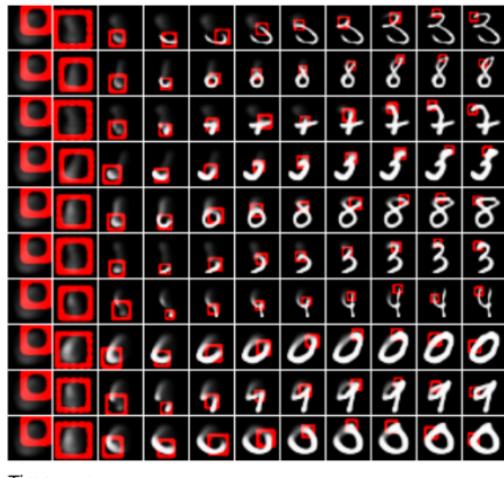


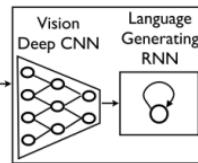
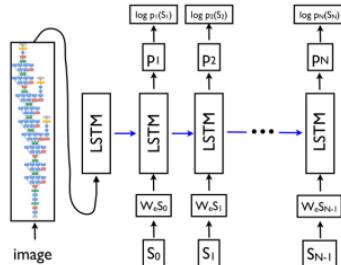
Figura 15: Generación de dígitos MNIST con una red DRAW



Figura 16: Generación de números de casa de Google Street View. La columna de la derecha es la imagen de entrenamiento más cercana.

Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., & Wierstra, D. (2015). DRAW: A recurrent neural network for image generation.

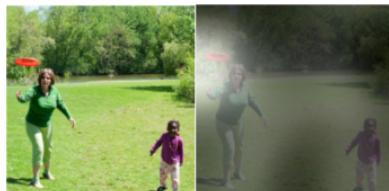
Anotación de imágenes



A group of people shopping at an outdoor market.
There are many vegetables at the fruit stand.

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2017). Show and Tell: Lessons Learned from the 2015 MSCOCO Image Captioning Challenge.

Anotación de imágenes con atención



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.



A group of people sitting on a boat in the water.



A giraffe standing in a forest with trees in the background.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention.

Respuesta a preguntas visuales



Is something under the sink broken?	yes	no
yes	yes	no
yes	no	no
What number do you see?	33	5
33	6	
33	7	



Does this man have children?	yes	yes
yes	yes	yes
yes	yes	yes
Is this man crying?	no	no
no	no	yes
no	no	yes



How many glasses are on the table?	3	2
3	2	6
3	2	6

What is the woman reaching for?
door handle
glass
wine

fruit
glass
remote



Can you park here?	no	no
no	no	yes
no	no	yes



Has the pizza been baked?	yes	yes
yes	yes	yes
yes	yes	yes

What kind of cheese is topped on this pizza?	feta	mozzarella
feta	mozzarella	mozzarella
ricotta	mozzarella	mozzarella



What kind of store is this?	bakery	art supplies
bakery	art supplies	grocery
pastry	grocery	grocery

Is the display case as full as it could be?	no	no
no	yes	yes
no	yes	yes



How many pickles are on the plate?	1	1
1	1	1
1	1	1

What is the shape of the plate?	circle	round
circle	round	round
round	round	round

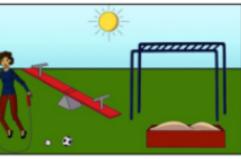


How many bikes are there?	2	3
2	2	4
2	4	12

What number is the bus?	48	46
48	46	number 6
48	number 6	number 6



What does the sign say?	stop	stop
stop	stop	yield
stop	yield	stop



How many balls are there?	2	1
2	2	2
2	2	3

What side of the teeter totter is on the ground?
right
left
right side

left
left
right side

Do you think the boy on the ground has broken legs?	yes	no
yes	yes	yes
yes	yes	yes

Why is the boy on the right freaking out?	his friend is hurt	ghost
his friend fell down	someone fell	lightning
someone fell	sprayed by hose	sprayed by hose

Bibliografía

- Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv Preprint arXiv:1409.0473.
- Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv Preprint arXiv:1409.1259.
- Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.
- Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, 12(10), 2451–2471.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. MIT Press. Retrieved from <http://www.deeplearningbook.org/>
- Graves, A. (2012). Supervised Sequence Labelling with Recurrent Neural Networks (1st ed.). Springer-Verlag Berlin Heidelberg.
- Graves, A., Wayne, G., & Danihelka, I. (2014). Neural Turing Machines. Retrieved from <http://arxiv.org/abs/1410.5401>
- Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. (2015). LSTM: A search space odyssey. arXiv Preprint arXiv:1503.04069, 1–11.
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.
- Karpathy, A., Johnson, J., & Fei-Fei, L. (2016). Visualizing and Understanding Recurrent Networks. *Iclr*, 1–13.
- Lipton, Z. C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning. *CoRR*, abs/1506.0, 1–38. <https://doi.org/10.1145/2647868.2654889>
- Olah, C. Understanding LSTM Networks. Retrieved from <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>