Thesis Title

by
Student Name

Professor SuperProf, Advisor

A thesis submitted in partial fulfillment
of the requirements for the
Degree of Bachelor of Arts with Honors
in Physics

WILLIAMS COLLEGE
Williamstown, Massachusetts
2 de agosto de 2021

Abstract

Your abstract will summarize your thesis in one or two paragraphs. This brief summary should emphasize methods and results, not introductory material.

Executive Summary

Your executive summary will give a detailed summary of your thesis, hitting the high points and perhaps including a figure or two. This should have all of the important take-home messages; though details will of course be left for the thesis itself, here you should give enough detail for a reader to have a good idea of the content of the full document. Importantly, this summary should be able to stand alone, separate from the rest of the document, so although you will be emphasizing the key results of your work, you will probably also want to include a sentence or two of introduction and context for the work you have done.

Acknowledgments

The acknowledgment section is optional, but most theses will include one. Feel free to thank anyone who contributed to your effort if the mood strikes you. Inside jokes and small pieces of humor are fairly common here ...

Índice general

Índice de figuras

Capítulo 1

Introduction

Capítulo 2

Transformers

2.1. De RNN's a Transformers

Las **Redes Neuronales Recurrentes** o **RNN** (por sus siglas en Inglés) datan del año 1986, basadas en el trabajo de Rumelhart [2]. Este tipo de redes están especializadas en el procesamiento de datos que contienen información temporal, mejorando los resultados obtenidos por otros tipos de redes como *Redes FeedForward* o *Redes Convolucionales*.

La principal idea detrás de estos modelos de red es el concepto de *Parameter Sharing*. Con *Parameter Sharing* un modelo un modelo puede generalizar mejor cuando la información que esta contenida en diferentes partes de una secuencia. Así, el modelo no necesita aprender independientemente todas las reglas que forman la secuencias, sino que ahora, la salida para cada elemento en el tiempo esta determinada por la salida del elemento anterior. Resultando en una recurrencia con las mismas reglas de actualización aplicadas a cada elemento en el tiempo. La ecuación 2.1 representa este proceso; $h^{(t)}$ es el estado de la recurrencia aplicada por alguna función f a un elemento $x^{(t)}t$ de la secuencia x en el tiempo, t y θ son los parámetros compartidos.

$$h^{(t)} = f(x^{(t)}, h^{(t-1)}; \theta) \tag{2.1}$$

En una *RNN* vista como un *gráfo computacional dirigído y acíclico*, cada nodo representa un estado en la recurrencia y procesa la información de la secuencia x con los mismos parámetros θ en cada paso, observe la figura 2.1.

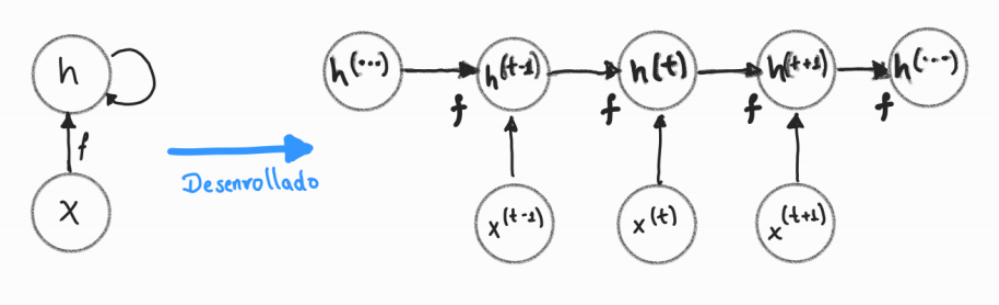
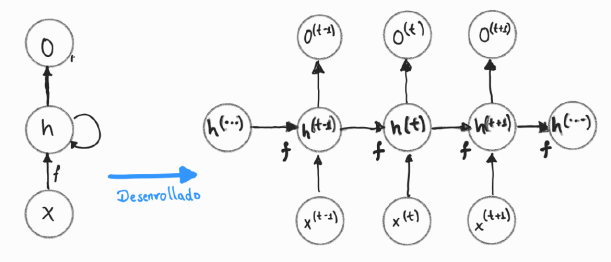


Figura 2.1: Grafo computacional generado por una *RNN* al "desenrollar"la recurrencia. Usando los parámetros compartidos en cada nodo, con cada elemento $x^{(t)}$ de la secuencia genera un nuevo estado oculto $h^{(t)}$ para retroalimentar nuevamente la entrada del siguiente nodo.

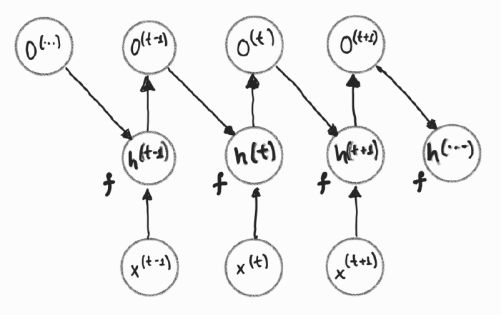
2.1.1. Redes Neuronales Recurrentes más comunes

Existen diversas formas como construir *Redes neuronales Recurrentes*, pueden producir una salida en cada paso de tiempo o tener solo una al final de la recurrencia y también pueden tener conexiones entre unidades ocultas. La manera más común de implementar una *RNN* está ilustrada en la figura 2.2a. En eta figura, cada etapa de la recurrencia es retroalimentada por la activación del estado oculto previo. Así, $h^{(t)}$ contiene información codificada de elementos previos de la secuencia que puede ser usada en el futuro para obtener una salida $O^{(t+1)}$. En la figura 2.2b se cambia la retroalimentación de $h^{(t)}$ por $o^{(t)}$. Nótese que en este caso, la red es entrenada para obtener un valor en específico $o^{(t)}$ lo que provocaría que gran parte de la información de los estados ocultos pasados $h^{(t-1)}, h^{(t-2)}, ...$ no se transmita. En el esquema anterior 2.2a la red es entrenada para decidir que información debe transmitir en el futuro a través de los estados ocultos, en cambio, en la figura 2.2b cada estado esta conectado con el pasado a través de la predicción del paso anterior, perdiendo así gran parte de la información codificada en los estados ocultos, a menos que la salida $O^{(t-1)}$ sea lo suficientemente rica y esté en altas dimensiones.

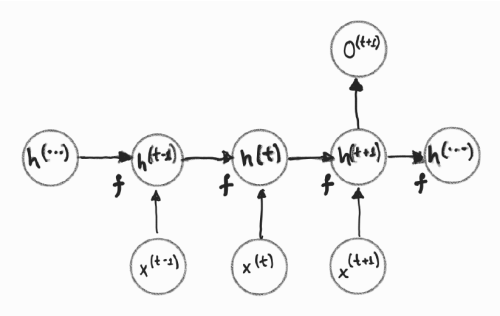
Por otro lado, la *RNN* representada en la figura 2.2c tiene una sola salida al final de la recurrencia. Al contrario de las anteriores, este tipo de redes pueden ser usadas para resumir información contenida en la secuencia para finalmente predecir un único valor final. El *Análisis de Sentimiento* en textos es una tarea común que puede ser representada con este esquema de red. En la figura 2.2d vemos un modelo de *RNN* entrenado mediante el proceso de Teacher Forcing; durante el entrenamiento la red es retroalimentada con las salidas esperadas del modelo $y^{(t)}$ en el tiempo $t+1$. La ventaja de esta red es que al ser eliminadas las conexiones entre estados ocultos, las funciones de pérdida basadas en comparar la predicción en el tiempo t con el valor objetivo $y^{(t)}$ pueden ser desacopladas. Por tanto, el entrenamiento puede ser paralelizado al calcular el gradiente para cada tiempo t por separado, puesto que ya tenemos el valor ideal para esta salida.



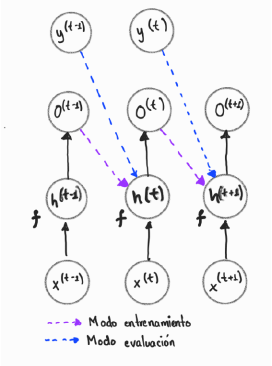
(a) Las activaciones de las capas ocultas $h^{(t)}$ alimentan al nodo siguiente, cada etapa de la recurrencia genera una salida $o^{(t)}$.



(b) Cada nodo es alimentado por las salidas de cada estado oculto $o^{(t)}$.



(c) Al final de la recurrencia solo tiene una salida $o^{(T)}$, puede ser usada para resumir/predecir un valor de una secuencia.



(d) Teacher Forcing. En modo de entrenamiento cada nodo en el tiempo t es retroalimentado por la salida correcta $y^{(t-1)}$, en modo evaluación es retroalimentado por las salidas del modelo $O^{(t-1)}$.

Figura 2.2: Distintos tipos de RNNs.

Finalmente, en la figura 2.3 la Red Neuronal Recurrente es modificada para esta vez no procesar una secuencia, sino que, procesa un solo vector en cada paso. El estado oculto previo $h^{(t-1)}$ retroalimenta al siguiente paso t así como la predicción esperada $y^{(t)}$ que también es usada para calcular la función de costo del paso anterior $L^{(t-1)}$. Esta estructura de red puede ser usada en tareas como Image Captioning, en donde la entrada es una imagen y la salida una secuencia de palabras que describen esta misma.

Si bien, los modelos ejemplificados anteriormente son construidos de forma *causal*, es decir, la secuencia es procesada en un solo sentido en donde la información pasada es transmitida hacia el estados futuros, no siempre este flujo de información es suficientemente para resolver ciertas tareas. Un *Modelo de Lenguaje* aprende la estructura estadística del lenguaje con el que fue entrenado y su meta es predecir la siguiente palabra, n-grama o letra dado un contexto antes visto. En otros términos, dada una secuencia de texto de longitud T $x^{(1)}, x^{(2)}, ..., x^{(T)}$ con $x \in \mathcal{R}^{1\times d}$ donde d es la dimensión de la codificación de las palabras, la

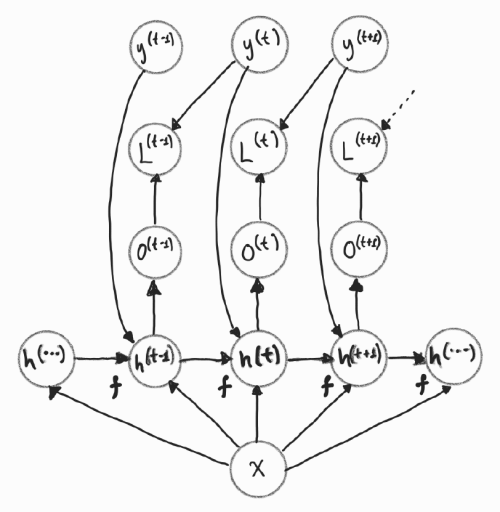


Figura 2.3: Modelo usado para tareas de *Image Captioning*, la entrada es una sola imagen y la red predice una secuencia de palabras que describen dicha imagen La salida esperada $y^{(t)}$ sirve como objetivo para la función de costo del paso anterior y como entrada en cada paso.

meta es predecir la probabilidad conjunta de la secuencia:

$$P(x^{(1)}, x^{(2)}, ..., x^{(T)}) = \prod_{t=1}^{T} P(x^{(t)}|x^{(t)}, \ldots, x^{(t-1)}) \tag{2.2}$$

Con ello, un modelo de lenguaje basado en *Redes Neuronales Recurrentes* es capaz de predecir un siguiente elemento $\hat{x}^{(t)}$ simplemente obteniéndolo de la secuencia mediante:

$$\hat{x}^{(t)} \approx P(x^{(t)}|x^{(t-1)}, \ldots, x^{(1)}) \approx P(x^{(t)}|h^{(t-1)}) \tag{2.3}$$

donde $h^{(t-1)}$ es el estado oculto que almacena la información pasada hasta el tiempo t tal y como se definió en 2.1.

Sin embargo, la información previa de la secuencia codificada en $h^{(t)}$ no siempre contiene los elementos necesarios para que el modelo pueda predecir correctamente el siguiente elemento, observe la siguiente oración:

« *Ella estaba muy _____, después de que Alejandra vió el amanecer en la playa* »

En la oración anterior, el espacio en blanco puede ser completado con algún adjetivo calificativo; *contenta*, *enojada*, *maravillada*, etc. Gracias a la información provista por la parte final de la oración, podemos deducir que de las 3 opciones la menos probable de elegir

es *enojada*. Es decir, usamos información del futuro que no pudo haber sido vista por una red (que procesa la información en forma causal) para tomar la mejor elección. Una ligera modificación fácilmente aplicable a estos modelos es que las secuencias sean procesadas en ambas direcciones, a lo que llamamos **Redes Neuronales Recurrentes Bidireccionales** [3].

Así una *RNN Recurrente* procesa la secuencia en ambos sentidos (una *RNN* en un sentido y otra en el otro), capturando información del pasado en el estado oculto $\overrightarrow{h}^{(t)}$ cuando la recurrencia es del inicio al final de la secuencia e información del futuro en $\overleftarrow{h}^{(t)}$ cuando la recurrencia es del final al inicio de la secuencia. Finalmente, el estado oculto $h^{(t)}$ es una concatenación de ambos estados $\overrightarrow{h}^{(t)}$ y $\overleftarrow{h}^{(t)}$, vea la ecuación 2.4. Por lo cual, la salida $o^{(t)}$ ahora puede ser calculada con información tanto del futuro como del pasado 2.5.

$$\begin{aligned} \overrightarrow{h}^{(t)} &= f(x^{(t)}, \overrightarrow{h}^{(t-1)}; \theta_f) \\ \overleftarrow{h}^{(t)} &= f(x^{(t)}, \overleftarrow{h}^{(t+1)}; \theta_b) \\ h^{(t)} &= Concat(\overrightarrow{h}^{(t)}, \overleftarrow{h}^{(t)}) \end{aligned} \tag{2.4}$$

$$o^{(t)} = g(h^{(t)}; \theta_{out}) \tag{2.5}$$

2.1.2. Compuertas LSTM y GRU

Hasta el momento, se ha hecho mención de las salidas $o^{(t)}$ y estados ocultos $h^{(t)}$ solo como el resultado de operaciones aplicadas por dos funciones; g y f respectivamente. Existen varias alternativas de construir una *RNN*, una de las maneras más comunes es usando 2.6 y 2.7:

$$h^{(t)} = \phi(x^{(t)}W_x + h^{(t-1)}W_h + b) \tag{2.6}$$

$$o^{(t)} = x^{(t)}W_{out} + b \tag{2.7}$$

Donde los parámetros compartidos de la red ahora son descritos por las matrices $W_x \in \mathbb{R}^{d\times k}$, $W_h \in \mathbb{R}^{k\times k}$ y $W_{out} \in \mathbb{R}^{k\times q}$ con k como la dimensión del estado oculto, q la dimensión de las salidas $o^{(t)}$, $b \in \mathbb{R}^{1\times q}$ el parámetro de sesgo y ϕ es la función de activación. De esta manera, los pesos de los parámetros aprendidos en la matriz W_h determinan cómo será usada la información del pasado, codificada en $h^{(t-1)}$. Posteriormente es incluida a la codificación

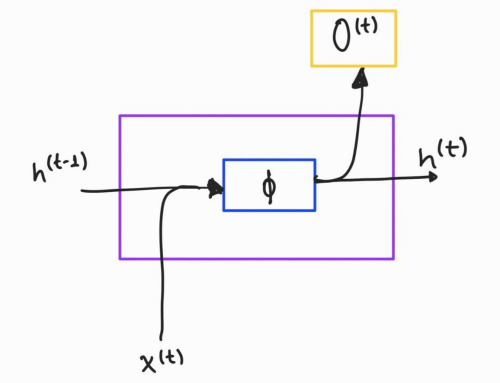


Figura 2.4: Computo del estado oculto y salida de una Red Neuronal Recurrente.

de la información del tiempo actual t calculada por W_x. La figura 2.4 representa gráficamente la lógica usada para calcular los estados ocultos y las salidas de la red.

Sin embargo el cálculo de los estados ocultos mediante 2.6 presenta algunos problemas. La interacción entre la información del pasado y la actual siempre es "plana", es decir, la información fluye a través del tiempo de la misma manera sin forma de dar prioridad o ignorar parte de esta. Por lo que resulta una tarea un poco más complicada preservar información relevante a en cada paso o desechar información que ya no es util para la red. También, causado por este mismo flujo de los datos, la información del pasado poco a poco es opacada por nueva información, impidiendo que se puedan encontrar dependencias de información en secuencias largas en tiempos distantes; comúnmente se hace referencia a este problema como *The Short-term Memory Problem* en inglés [4]. Aunado a problemas como el *Desvanecimiento o Explosión del Gradiente* [5] [6], acentuándose aun más debido a las matrices de pesos compartidos en la recurrencia. Por ejemplo, podemos imaginar las multiplicaciones en la recurrencia como un problema simplificado en donde dicha matriz es multiplicada por si misma muchas veces (una similitud al método de potencia donde cualquier componente en la matriz inicial que no esté alineada con el vector propio asociado al mayor valor propio son eventualmente descartados [7, pp. 390-392]), así los resultados de este producto tendrán a ser cercanos a cero (desvanecerse) o explotar dependiendo de la magnitud de la matriz de pesos.

Una manera de solventar los problemas anteriores son las **Redes Neuronales con Compuertas**, creadas con la idea de crear conexiones a través del tiempo de tal manera de tener gradientes que no se desvanezcan o exploten, convirtiéndose además en un mecanismo para olvidar información pasada y decidiendo automáticamente cuándo y cuánto de la información debe prevalecer.

LSTM

Long Short-Term Memory, **LSTM** por sus siglas en inglés, fue propuesta en 1997 por *Hochreiter y Schmidhuber* [8], como un método de preservar dependencias de información relevante distantes a corto plazo. Las *LSTM* introducen un nuevo componente la *Celda de Memoria* cuya función es guardar información a través del tiempo y es controlada por distintas compuertas, las cuales aprenden a distinguir que información es relevante y cual no. Contiene tres compuertas, la *Compuerta de Entrada*, la *Compuerta de Olvido* y la *Compuerta de Salida*. La *Compuerta de Entrada* $I^{(t)}$ (véase la figura 2.5) determina cuanta información actual debe ser contemplada a través de la *Memoria Candidata* $\tilde{C}^{(t)}$ para actualizar la *Celda de Memoria* $C^{(t)}$. La *Compuerta de Olvido* indica qué información del pasado debe ser desechada de la *Celda de Memoria* $C^{(t-1)}$ y la *Compuerta de Salida* ayuda a determinar el nuevo estado $h^{(t)}$ via la *Celda de Memoria* actual $C^{(t)}$.

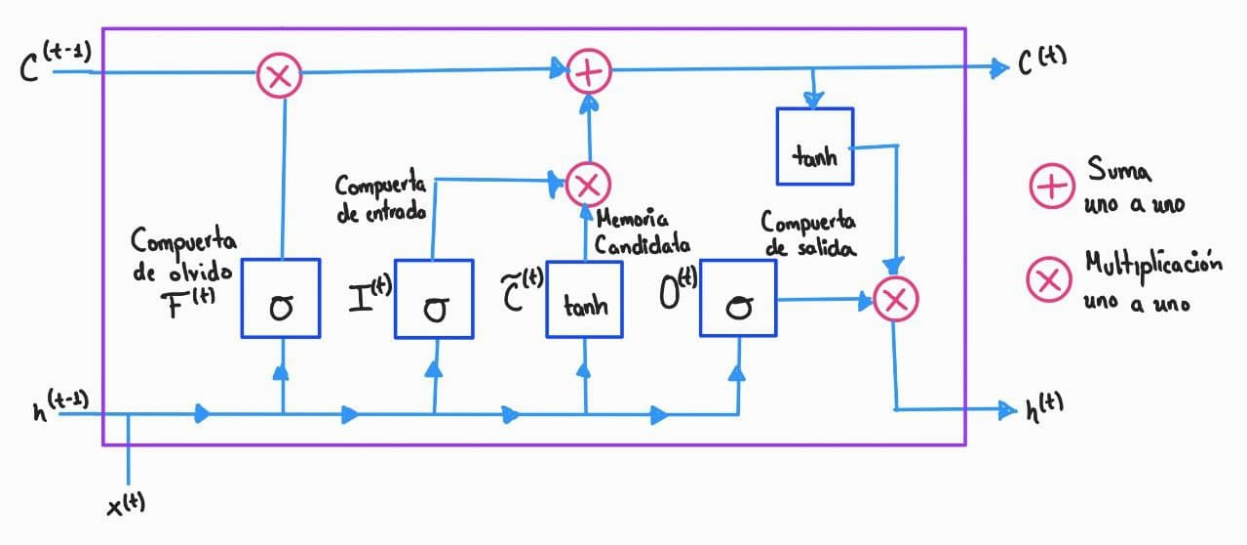


Figura 2.5: Descripción.

Las ecuaciones 2.8 rigen el comportamiento de Compuertas de Entrada, Salida y Olvido.

$$\begin{aligned} I^{(t)} &= \sigma(x^{(t)}W_{xi} + h^{(t-1)}W_{hi} + b_i) \\ F^{(t)} &= \sigma(x^{(t)}W_{xf} + h^{(t-1)}W_{hf} + b_f) \\ O^{(t)} &= \sigma(x^{(t)}W_{xo} + h^{(t-1)}W_{ho} + b_o) \end{aligned} \tag{2.8}$$

Donde $W_{xi}, W_{xf}, W_{xo} \in \mathbb{R}^{d\times k}$, $W_{hi}, W_{hf}, W_{ho} \in \mathbb{R}^{k\times k}$ y $b_i, b_f, b_o \in \mathbb{R}^{1xk}$

La Memoria Candidata y la Celda de memoria son actualizadas mediante:

$$\begin{aligned} \tilde{C}^{(t)} &= \tanh\Big(x^{(t)}W_{xi} + h^{(t)}W_{hc} + b_c\Big) \\ C^{(t)} &= F^{(t)} \odot C^{(t-1)} + I^{(t)} \odot \tilde{C}^{(t)} \end{aligned} \tag{2.9}$$

Donde $W_{xi} \in \mathbb{R}^{d\times k}$, $W_{hc} \in \mathbb{R}^{k\times k}$ y $b_c \in \mathbb{R}^{1xk}$

Y finalmente el estado oculto $h^{(t)}$ esta dado por:

$$h^{(t)} = O^{(t)} \odot \tanh\Big(C^{(t)}\Big) \tag{2.10}$$

σ y $\odot$ denotan la función de activación sigmoide y la multiplicación uno a uno respectivamente.

GRU

Gated Recurrent Units o **GRU** por sus siglas en inglés, fueron propuestas en 2014 [9] como una alternativa computacionalmente más rápida y con similar rendimiento que las *LSTM* [10]. A diferencia de anteriores, las *GRU* prescinden de la *Celda de Memoria* y utilizan un par de compuertas (la *Compuerta de Actualización* y la de *Olvido*) para decidir que información aún es necesario que esté codificada dentro del estado oculto 2.11. La *Compuerta de Olvido* permite decidir que del pasado aún debe ser transmitido a futuros estados o de otro modo ser desechada. La *Compuerta de Actualización* indica que información nueva es relevante y necesita ser incorporada al no estar codificada dentro del estado oculto 2.12.

$$\begin{aligned} R^{(t)} &= \sigma(x^{(t)}W_{xR} + h^{(t-1)}W_{hR} + b_R) \\ Z^{(t)} &= \sigma(x^{(t)}W_{xZ} + h^{(t-1)}W_{hZ} + b_Z) \end{aligned} \tag{2.11}$$

$$\begin{aligned} \tilde{h}^{(t)} &= \tanh\Big(x^{(t)}W_{xh} + (R^{(t)} \odot h^{(t-1)})W_{hh} + b_h\Big) \\ h^{(t)} &= Z^{(t)} \odot h^{(t-1)} + (1 - Z^{(t)}) \odot \tilde{h}^{(t-1)} \end{aligned} \tag{2.12}$$

2.1.3. Mecanismos de Atención

Una de las arquitecturas comunes vistas previamente es la 2.2c cuya información procesada es resumida en una sola salida. Este tipo de red es usada como parte de las soluciones a

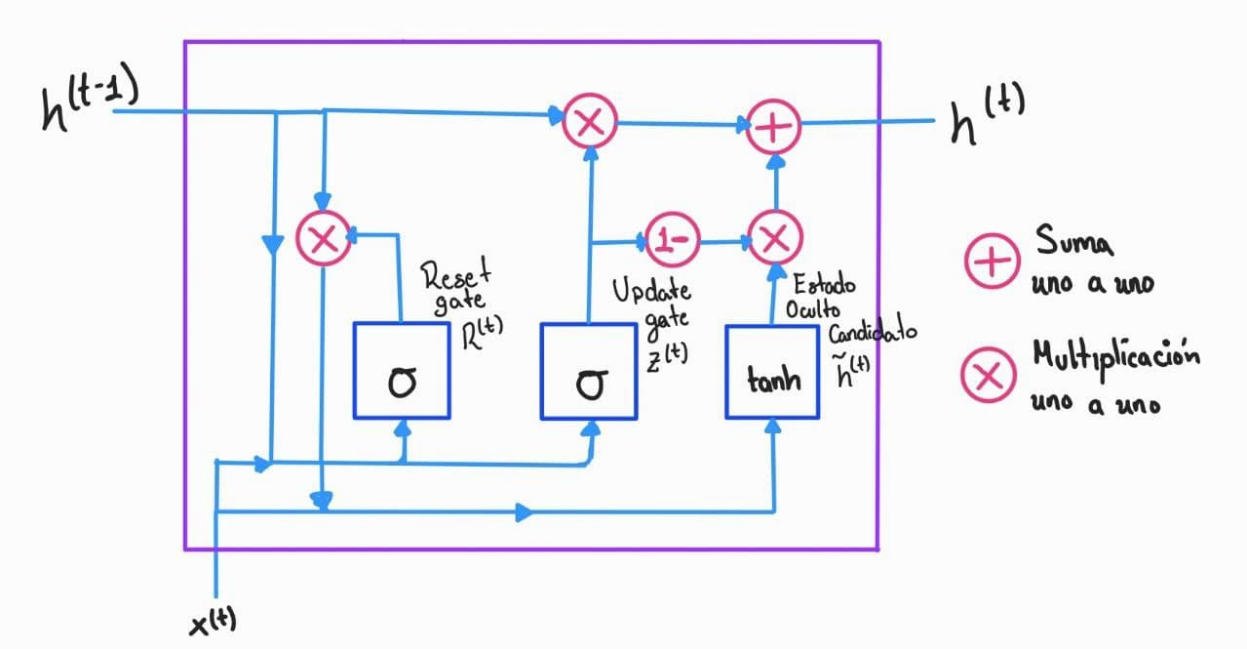


Figura 2.6: Descripción.

tareas de como reconocimiento de voz (*Speech Recognition*), traducción de lenguaje (*Machine Translation*) o asistencia en respuestas automáticas (*Question Answering*) típicamente bajo modelos Secuencia a Secuencia (*Sequence to Sequence, Seq2Seq*). Los modelos *Seq2Seq* están formados por dos redes neuronales como la mostrada en 2.7. La primera se comporta como *codificador* al resumir la entrada para producir un vector de salida de tamaño fijo llamado *vector de contexto*. La segunda red se comporta como un *decodificador*, este es inicializado y condicionado con el *vector de contexto* para obtener una transformación de la entrada no necesariamente del mismo tamaño de secuencia, ya sea por ejemplo, traducir una oración de español a inglés en donde la traducción no siempre contiene las misma cantidad de palabras usadas en el idioma original.

Por ejemplo, en tareas de *Machine Translation* el *encoder* esta formado por una *RNN* Bidireccional que lee y procesa un conjunto de vectores $X = (x^{(1)}, x^{(2)}, \ldots, x^{(T_x)})$ para obtener un vector de contexto C. La forma más común es como en 2.13:

$$\begin{aligned} h^{(t)} &= f_{bi}(x^{(t)}, h^{(t-1)}; \theta_f, \theta_b) \\ C &= q(h^{(1)}, h^{(2)}, \ldots, h^{(T)}) \end{aligned} \tag{2.13}$$

Recordemos que $h^{(t)}$ es el estado oculto generado por la concatenación de los dos estados ocultos generados por la *RNN Bidireccional*, f_{bi} y q son funciones no lineales, ya sea, una *LSTM* para f_{bi} y $q(h^{(1)}, h^{(2)}, \ldots, h^{(T)}) = h^{(T)}$, equivalente a tomar solo el ultimo estado oculto

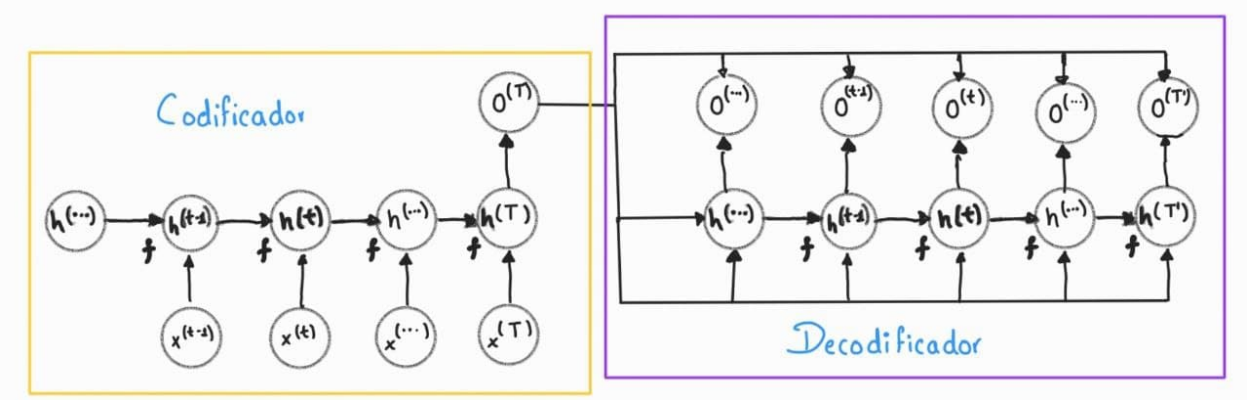


Figura 2.7: Descripción.

como vector de contexto C. El *Decoder* es entrenado para predecir la siguiente palabra $y^{(t')}$ dado el vector de contexto C y todas las palabras previas predichas. En otras palabras, el decoder define la probabilidad conjunta modelada por una *RNN*:

$$p(Y) = \prod_{t=1}^{T_y} p(y^{(t)}|\{y^{(1)}, \ldots, y^{(t-1)}\}, C) \tag{2.14}$$

$$p(y^{(t)}|\{y^{(1)}, \ldots, y^{(t-1)}\}, C) = g(y^{(t-1)}, s^{(t)}, C; \theta_g) \tag{2.15}$$

donde g es una función no lineal que emite la probabilidad de $y^{(t)}$ y $s^{(t)}$ es el estado oculto del *decoder* 2.16.

$$s^{(t)} = f(s^{(t)}, y^{(t-1)}, C; \theta_s) \tag{2.16}$$

Sin embargo, cuando las secuencias son bastante largas el *vector de contexto* emitido por el *codificador* no es lo suficientemente grande como para resumir correctamente la secuencia y por tanto, la información inicial de la entrada es olvidad, por lo que su presencia es escasa en estados ocultos más lejanas. Así, en 2015 [1, Bahdanau et. al] observaron estos efectos y propusieron una forma de minimizarlos, surgiendo los **Mecanismos de Atención**.

Ahora las palabras predichas no son calculadas por un único *vector de contexto* generado por el codificador, sino que para cada objetivo $y^{(t)}$ se calcula un *vector de contexto* $c^{(t)}$:

$$p(y^{(t)}|\{y^{(1)}, \ldots, y^{(t-1)}\}, c^{(t)}) = g(y^{(t-1)}, s^{(t)}, c^{(t)}; \theta_g) \tag{2.17}$$

$$s^{(t)} = f(s^{(t)}, y^{(t-1)}, c^{(t)}; \theta_s) \tag{2.18}$$

Dado que cada estado oculto $h^{(t)}$ contiene mucho mejor la información que se encuentran alrededor del t-ésimo término, se puede generar cada vector de contexto como una suma pesada de sobre los estados ocultos del codificador. Estos pesos nos ayudan a determinar que tan importante es la información codificada por cada estado oculto y al momento de obtener la salida del t-ésimo valor "prestar atención" a aquellos que son más relevantes para obtener la predicción:

$$c^{(t)} = \sum_{i=1}^{T_x} \alpha_{t,i} h^{(i)} \tag{2.19}$$

cada peso $\alpha_{t,i}$ indica que tan bien se "alinean" los términos $y^{(t)}$ y $x^{(i)}$

$$\alpha_{t,i} = align(y^{(t)}, x^{(i)}) = \frac{\exp\Big(score(s^{(t-1)}, h^{(i)})\Big)}{\sum_{k=1}^{T_x} \exp(score(s^{(t-1)}, h^{(k)}))} \tag{2.20}$$

Bahdanau propone aprender esta alineación usando una *Red feed-forward* con una sola capa oculta y la función tanh como activación:

$$score(s^{(t)}, h^{(i)}) = v_a^\top \tanh\Big(W_a[s^{(t)}; h^{(i)}]\Big) \tag{2.21}$$

con v_a y W_a como matrices de pesos a aprender durante el entrenamiento. En la figura 2.8 podemos ver gráficamente el modelo usado por *Bahdanau*.

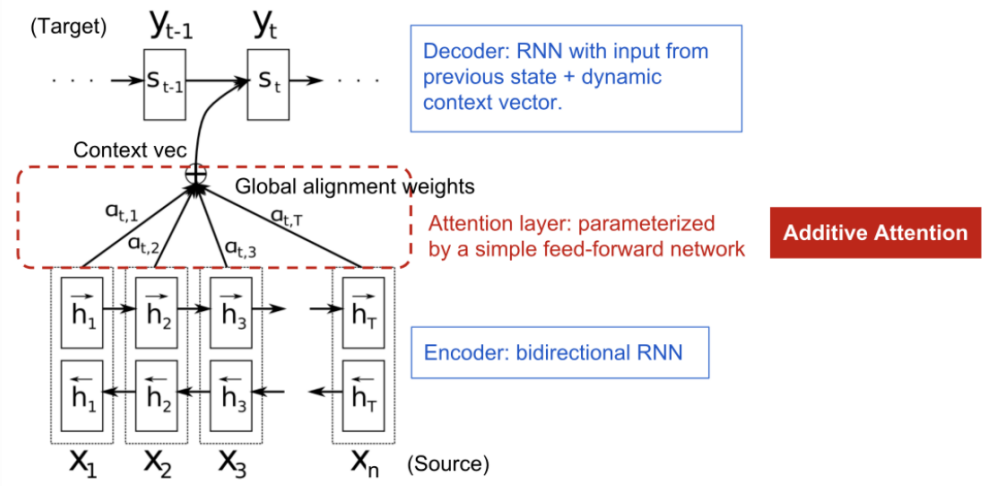


Figura 2.8: Modelo seq2seq propuesto por Bahdanau et. al [1] con *Additive/Concat Attention*

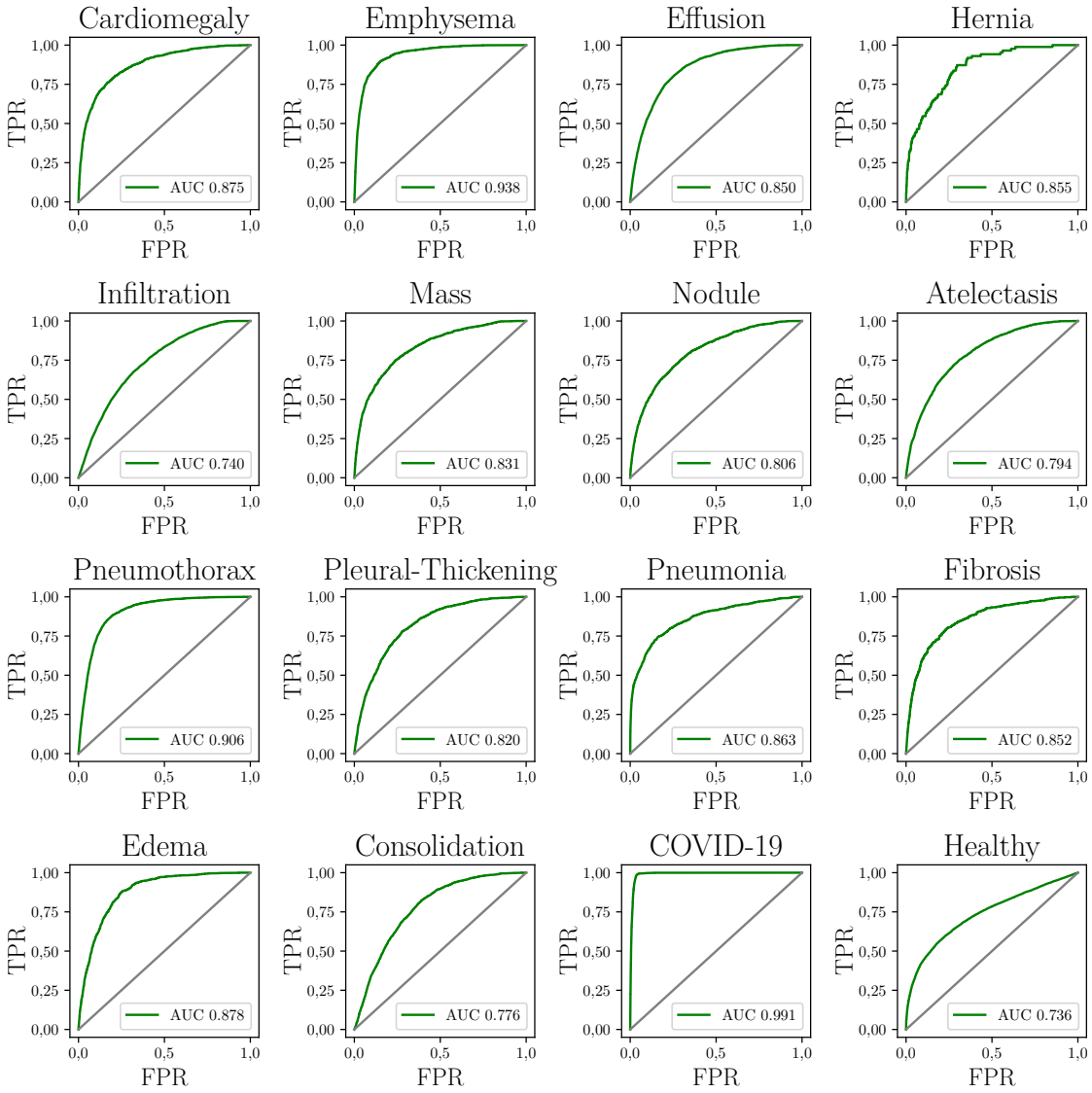
Así, diversas investigaciones comenzaron surgir con distintos*Mecanismo de Atención*. La tabla 2.1 resume estos métodos.

Nombre	Función de Puntaje de Alineación	Cita
Content-Base Attention	$score(s^{(t)}, h^{(i)}) = cosine[s^{(t)}, h^{(i)}]$	[11, Graves 2014]
Additive [1]	$score(s^{(t)}, h^{(i)}) = v_a^\top \tanh\Big(W_a[s^{(t)}; h^{(i)}]\Big)$	[1, Bahdanau 2015]
Location-Base	$a_{t,i} = softmax(W_a s^{(t)})$	[12, Luong 2015]
General	$score(s^{(t)}, h^{(i)}) = s^{(t)\top} W_a h^{(i)}$	[12, Luong 2015]
Dot Product	$score(s^{(t)}, h^{(i)}) = s^{(t)\top} h^{(i)}$	[12, Luong 2015]
Scaled Dot-Product [2]	$score(s^{(t)}, h^{(i)}) = \frac{s^{(t)\top} h^{(i)}}{\sqrt{d_k}}$	[13, Vaswani 2017]

Cuadro 2.1: Tipos de funciones para calcular el Puntaje de Alineación. (Tabla basada en [14]).
[1] También llamado *Concat* Luong, et al., 2015 y "Additive attention"" en Vaswani, et al., 2017.
[2] El factor de escala $\frac{1}{\sqrt{d_k}}$ ayuda a estabilizar cuando el gradiente es muy pequeño. d_k es el tamaño de la cabeza de atención.

2.2. El modelo Transformer

A finales del año 2017 se presentó un nuevo modelo que vino a revolucionar el área de Procesamiento de Lenguaje Natural, El Transformer [15]. Una de sus principales características es la capacidad de procesar la información de alguna secuencia de forma paralela, caso contrario a las Redes Neuronales Recurrentes, donde la información se procesa recurrentemente. Gracias a ello la capacidad de *recuerdo* no se ve afectado por el problema de *El desvanecimiento del Gradiente* cuando se trabaja con secuencias bastante largas.


Cardiomegaly
AUC 0.875
Emphysema
AUC 0.938
Effusion
AUC 0.850
Hernia
AUC 0.855
Infiltration
AUC 0.740
Mass
AUC 0.831
Nodule
AUC 0.806
Atelectasis
AUC 0.794
Pneumothorax
AUC 0.906
Pleural-Thickening
AUC 0.820
Pneumonia
AUC 0.863
Fibrosis
AUC 0.852
Edema
AUC 0.878
Consolidation
AUC 0.776
COVID-19
AUC 0.991
Healthy
AUC 0.736
TPR
FPR
0,00
0,25
0,50
0,75
1,00
0,0
0,5
1,0

Apéndice A

An appendix

Appendices are a good idea for almost any thesis. Your main thesis body will likely contain perhaps 40-60 pages of text and figures. You may well write a larger document than this, but chances are that some of the information contained therein, while important, does *not* merit a place in the main body of the document. This sort of content - peripheral clarifying details, computer code, information of use to future students but not critical to understanding your work . . . - should be allocated to one or several appendices.

A.1. About the bibliography

What follows this is the bibliography. This has its own separate environment and syntax; check out the comments in the .tex files for details. Worth nothing, though, is that you may find it helpful to use automated bibliography management tools. BibTeX will automatically generate a bibliography from you if you create a database of references. Other software - for example JabRef on a pc - can be used to make managing the reference database easy. Regardless, once you've created a .bib file you can cite it in the body of your thesis using the `\cite` tag. For example, one might wish to cite a reference by Bermudez [16]. If you use BibTeX, you can put the relevant information into a referencedatabase (called bibliography.bib here), and then BibTeX will compile the references into a .bbl file ordered appropriately for your thesis based on when the citations appear in the main document.

Bibliografía

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2016.

[2] R. D. E., H. G. E., and R. J. Williams, "Learning representations by back-propragation errors.," *Nature*, vol. 323, pp. 533,536, 1986.

[3] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks.," *IEEE Transactions on Signal Processing*, vol. 45(11), pp. 2673–2681, 1997.

[4] Y. Bengio, P. Simard, and P. Fraconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5(2), pp. 157–166, 1994.

[5] S. Hochreiter, Y. Bengio, and P. Frasconi, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," *Field Guide to Dynamical Recurrent Networks*, 2001.

[6] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on Machine Learning* (S. Dasgupta and D. McAllester, eds.), vol. 28(3) of *Proceedings of Machine Learning Research*, (Atlanta, Georgia, USA), pp. 1310–1318, 17–19 Jun 2013.

[7] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2017. `http://www.deeplearningbook.org`.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9(8), p. 1735–1780, 1997.

[9] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *CoRR*, vol. abs/1409.1259, 2014.

[10] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014.

[11] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *CoRR*, vol. abs/1410.5401, 2014.

[12] M. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *CoRR*, vol. abs/1508.04025, 2015.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[14] L. Weng, "Attention? attention!," *lilianweng.github.io/lil-log*, 2018.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gómez, L. Kaiser, and I. Polosukhin, "Attention is all you need.," *NIPS*, 2017.

[16] A. Bermudez, M. Bruderer, and M. B. Plenio, "Controlling and measuring quantum transport of heat in trapped-ion crystals," *Phys. Rev. Lett.*, vol. 111, p. 04091, 2013.