Métodos analíticos, primavera 2020
Laura Gómez Bustamante 191294
Francisco Paz Cendejas 175764
Marco Julio Monroy 187825

Neural Machine Translation y mecanismos de atención para la traducción de enunciados español-inglés

Con base en los papers de Bahdanau et al., 2014 [1] y Luong et al., 2015 [2] se explica e implementa la idea de usar mecanismos de atención en los modelos de *Neural Machine Translation* (NMT) como solución al problema de traducción de lenguaje. Los códigos e instrucciones para replicar este proyecto se encuentran en el siguiente enlace de github github-NeuralMachineTranslation_esp-ing.

1. Introducción

La tarea de leer un texto y generar la traducción en otro idioma (*Machine Translation*) es muy antigua; sin embargo las primeras aproximaciones usando redes neuronales aparecieron alrededor del año 2013. La idea básica consistía en introducir un enunciado o secuencia de palabras, pasarlas por una red y esta última regresaría la traducción, es decir, una nueva secuencia de palabras.

Dichas redes se implementaron a través de una familia de *encoders* y *decoders*, que aunque tienen un *performance* bueno para la traducción de enunciados, su capacidad se reducía conforme aumentaba el número de palabras. Para solucionar este inconveniente, se introduce la idea de mecanismos de atención, los cuales buscan crear un contexto entorno a la palabra y agregarlo como información al momento de traducir el texto.

El uso de redes recurrentes (RNN) tomó relevancia para resolver esta tarea; y aunque actualmente existen métodos más eficientes como el uso de arquitecturas *transformers* [3] (que dicho sea de paso también usan mecanismos de atención), las RNN siguen siendo objeto de investigación.

En el presente trabajo se exponen las ideas, metodología y un ejemplo de la implementación que se usó en el año 2015 para realizar la tarea de traducción de máquina. En la sección 2 se hablará de la arquitectura y los modelos que se necesitan entender para llevar cabo la implementación de la solución, específicamente se revisarán en detalle los encoders y decoders de las modelos de NMT clásicos y con Atención. Con respecto a la sección, 3 se expondrá la forma en la que se desarrolló la implementación así como la base de datos

utilizada. Finalmente se presentarán algunas conclusiones sobre los hallazgos tanto en la investigación como en el desarrollo del proyecto.

2. Arquitectura/Modelos

2.1 Modelos *Sequence-to-sequence*

Los modelos *sequence-to-sequence* pertenecientes al conjunto de modelos de aprendizaje profundo, han alcanzado gran éxito en tareas como traducción máquina, resumen de texto y descripción de imágenes. De manera específica, estos modelos toman como entrada una serie de elementos (palabras, letras, características de una imagen, etc), los procesan y generan como resultado otra serie de elementos [6]. Puntualmente, la arquitectura de tales modelos está compuesto por un *encoder* y un *decoder*. El *encoder* procesa cada elemento de la secuencia de entrada y compila la información capturada en un vector llamado *contexto*. Una vez que la secuencia de entrada fue procesada, el *encoder* envía el vector *contexto* al *decoder*, el cual empieza a producir la secuencia de salida.

Dentro del contexto de modelos *sequence-to-sequence* y en particular para resolver la tarea de traducción de máquina, se encuentran las llamadas máquinas neuronales de traducción (*Neural Machine Translation* NMT), en las cuales tanto la secuencia de entrada como de salida es una serie de palabras.

2.2 Neural Machine Translation

La manera en la que los seres humanos traducimos de un lenguaje a otro implica leer oraciones completas dentro de un párrafo, entender su significado y después generar una traducción, esto se hace para todas las oraciones contenidas. Las máquinas neuronales de traducción (*Neural Machine Translation* NMT) intentan imitar este comportamiento. [5]

Dado que NMT es un modelo *sequence-to-sequence*, su estructura está formada por los siguientes elementos:

- Un *encoder* que lee una oración y construye un vector de *contexto* de longitud fija (es decir, una secuencia de números que representan el significado de la oración).
  .
- Y un decoder que *procesa* el mencionado vector para emitir una traducción.

En la Figura 1 se muestra un diagrama de la estructura general de un sistema NMT.

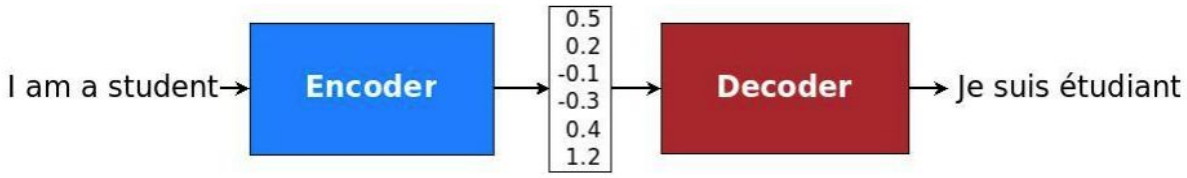


Fig 1. Arquitectura *encoder-decoder* - diagrama general de un sistema NMT [5].

Existen varias arquitecturas que conducen a variaciones en los modelos de NMT. La mayoría de ellas utilizan redes neuronales recurrentes (RNN) tanto en el *encoder* como en el *decoder*; y las variaciones se presentan en cuanto a dirección, pues pueden ser unidireccionales o bidireccionales; profundidad, referido a una o varias capas; y tipo de red en el sentido que puede utilizarse una vanillla RNN, una *Long Short-Term Memory* (LSTM) o una *Gated Recurrent Unit* (GRU) [5].

Planteando el problema de traducción desde una perspectiva probabilística, se busca la oración objetivo y tal que maximice la probabilidad condicional $y|x$, es decir, $arg\ max_y\ p(y|x)$, en donde x representa la oración de entrada. Conforme a lo descrito previamente, la solución de este problema utiliza una RNN con *encoder* y *decoder*; para introducir estos conceptos se hará uso de la estructura planteada por [1] para después agregar mecanismos de atención, también establecidos por [1].

2.2.1 Encoder

El *encoder* es una RNN encargada de leer el enunciado *input*, el cual es una secuencia de vectores $x = (x_1, ..., x_{Tx})$ dentro de un vector c, cuya longitud en muchas ocasiones es fija. La definición más común dentro de las RNN es la siguiente:

$$c = q(\{h_1.h_2, ..., h_{Tx}\}) \tag{1}$$

donde $h_t = f(x_t, h_{t-1})$ representa el estado oculto al tiempo t, y c es la secuencia formada por los estados ocultos que también es conocida como vector contexto[1]. Las funciones f y q son funciones no lineales; una propuesta común es tomar f como una LSTM y $q(\{h_1.h_2, ..., h_{Tx}\}) = h_T$

2.2.2 Decoder

El *decoder* es entrenado para predecir la palabra y_t dados el vector contexto c y las palabras antes predichas $\{y_1, .., y_{t-1}\}$. De esta forma, el *decoder* es entonces el encargado de definir la

probabilidad de traducción $\vec{y} = (y_1, ..., y_{Ty})$ descomponiendo la probabilidad condicional de la siguiente forma:

$$p(\vec{y}) = \prod_{t=1}^{T} p(y_t \mid \{y_1, ..., y_{t-1}\}, c) \quad (2)$$

Para una RNN $p(y_t \mid \{y_1, ..., y_{t-1}\}, c) = g(y_{t-1}, s_t, c)$ donde g es una función no lineal y s_t representa un estado oculto de una RNN.

2.3 Mecanismos de Atención

El vector de *contexto* de longitud fija se convirtió en un cuello de botella de los modelos tipo NMT, pues la red neuronal debía ser capaz de comprimir toda la información necesaria proveniente de la secuencia de entrada en un vector de longitud fija. Esto condujo a que fuera difícil para la red neuronal lidiar con la traducción de frases largas; específicamente con aquellas cuya longitud era mayor que las oraciones contenidas en el *corpus* de entrenamiento [1]. Una solución a este problema fue propuesta en Bahdanau et al., 2014 [1] y Luong et al., 2015 [2]. Estos papers introdujeron una técnica llamada *Attention*, la cual mejoró la calidad de las máquinas de traducción. La Atención le permite al modelo concentrarse en las partes relevantes de la secuencia de entrada conforme va siendo necesario.

La diferencia principal entre un modelo clásico *sequence-to-sequence* y el modelo con *Atención* radica en que este último no pretende codificar toda la sentencia de entrada en un único vector de longitud fija. En cambio, codifica la sentencia de entrada en una secuencia de vectores y elige un subconjunto de estos vectores adaptativamente mientras decodifica la traducción [1]. Lo anterior implica que el *encoder* transmite mucha más información en un modelo de Atención; pues en lugar de pasar únicamente el último estado oculto de la etapa *encoding*, el *encoder* pasa todos los estados ocultos hacia el *decoder*.

2.3.1 Modificaciones al Encoder para obtener Mecanismos de Atención

En contraste con el *encoder* definido en la sección 2.2.1, cuya idea es leer una secuencia que inicia en x_1 y termina en x_{Tx}, es decir, en una sola dirección; el mecanismo de atención propone hacerlo de manera bidireccional, de esta forma una palabra no se ve afectada únicamente por sus predecesoras si no que también es influenciada por la palabras consiguientes. Con base en el argumento anterior, la ecuación (1) se modifica conforme a $h_t = [\rightarrow h'_j, \ h'_j \leftarrow]$, de esta forma h_t contiene información tanto de las palabras anteriores como de las posteriores. En este caso h_t es conocido como *annotation*.

2.3.2 Modificaciones al Decoder para obtener Mecanismos de Atención

En relación al mecanismo de atención, el componente que se ve afectado es el *decoder*, en donde la definición que utilizada para la probabilidad condicional $p(y_t \mid \{y_1, ..., y_t\}, x)$ incorpora ahora un contexto diferente para cada $y_{i,}$ es decir,

$$p(y_t \mid \{y_1, ..., y_t\}, x) = g(y_{i-1}, s_i, c_i) \tag{3}$$

donde s_i representa el estado oculto de una RNN al tiempo i, y está dado por $s_i = f(s_{i-1}, y_{i-1}, c_i)$. Nótese que f representa la función de activación de la RNN; y de acuerdo con [1], se utiliza una *gated hidden unit*. Por otro lado, el cálculo del vector de contexto c_i queda determinado por la siguiente expresión:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{4}$$

Aquí es posible observar que el vector contexto es una suma de pesos de los estados escondidos (o *annotations*) resultantes del *encoder*. Es importante hacer notar que cada uno de estos estados escondidos contiene información acerca de la oración de entrada, con un fuerte enfoque en las partes que rodean la i-ésima palabra de dicha secuencia. Para obtener cada c_i es necesario obtener los pesos de atención α_{ij}, para lo cual es necesario revisar un concepto más, el *score*. En Bahdanau et al., 2014 [1] se introduce una primera definición; y un año más tarde Luong et al., 2015 [2] proporciona otra definición para el cálculo del score:

$$score(s_{i-1}, h_j) = \vartheta_a tanh(W_1 s_{i-1} + W_2 h_j) \quad [1]$$
$$score(s_{i-1}, h_j) = s'_{i-1} W h_j \quad [2]$$

donde W_1 y W_2 representan matrices de pesos. Una vez obtenido el score, basta pasarlo a través de una softmax y finalmente obtener los pesos α_{ij} a través de:

$$\alpha_{ij} = \frac{exp(score(s_{i-1}, h_j))}{\sum_{k=1}^{T_x} exp(score(s_{i-1}, h_k))} \tag{5}$$

Vale la pena enfatizar que cada α_{ij} refleja la importancia de la *annotation* h_j con respecto al estado oculto previo s_{i-1} en decidir el siguiente estado s_i y generar la traducción y_i. Intuitivamente lo anterior implementa un mecanismo de atención en el *decoder*. El *decoder* decide a qué partes de la secuencia de entrada pone atención [1].

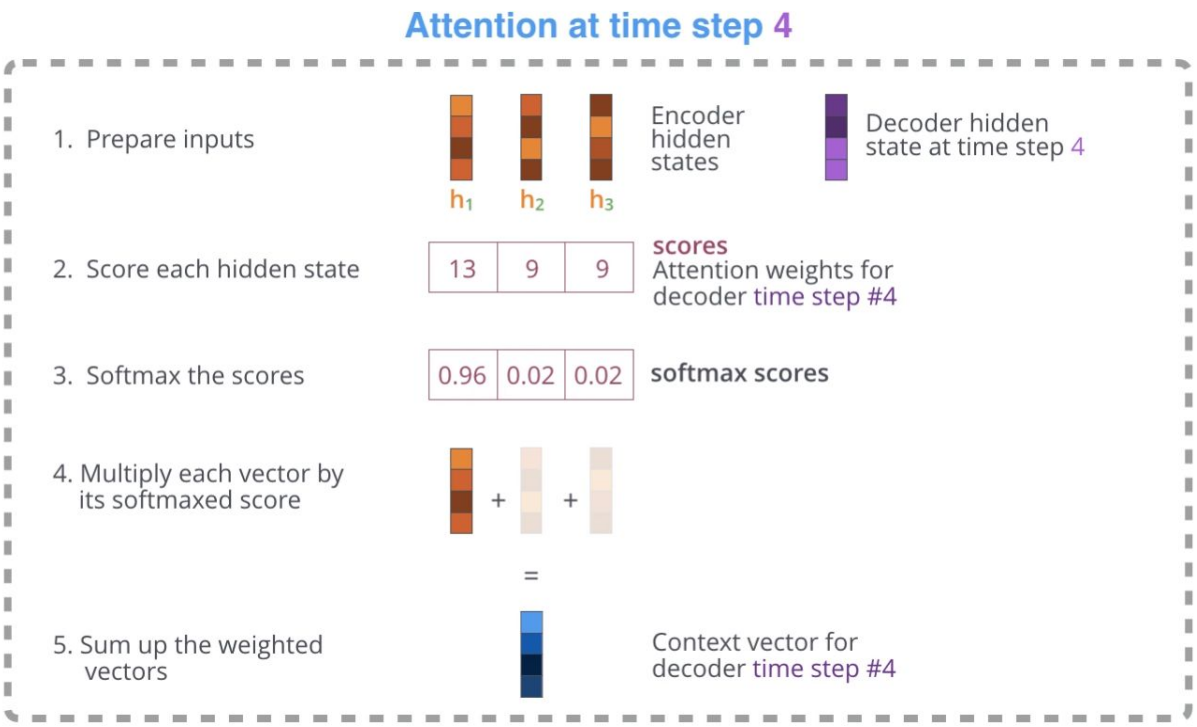


Fig 2. Mecanismos de atención y cálculo del vector contexto [6].

En la Figura 2 observa un diagrama del funcionamiento del mecanismo de Atención descrito anteriormente.

2.4 Neural Machine Translation con Mecanismos de Atención

En las secciones anteriores se presentaron los conceptos necesarios para entender el funcionamiento de las NMT, así como el concepto de Mecanismo de Atención. A continuación se explica el funcionamiento de estos elementos en su conjunto.

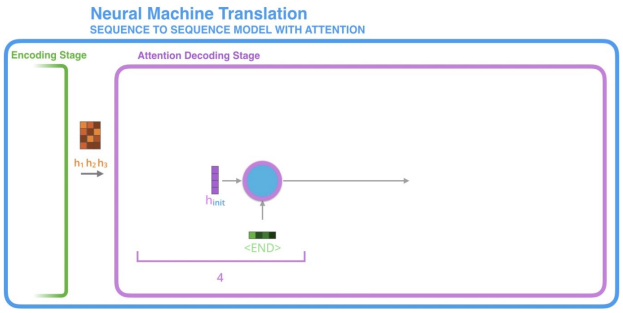
Sobre el *encoder*

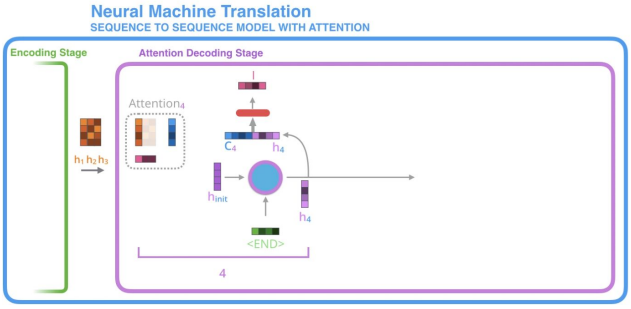
1. Dada una secuencia de entrada, para este caso palabras a las cuales se les aplicará un *embedding* con un token de inicio y uno de fin .
2. Esto alimenta al *encoder* y crea los estados ocultos que ocupará el *decoder*.

Sobre el *decoder*

Como ya se mencionó, esta es la parte que más cambia en la arquitectura, por lo que en los párrafos siguientes se detalla cómo es que los componentes antes mencionados se ven en el diseño del *decoder*.

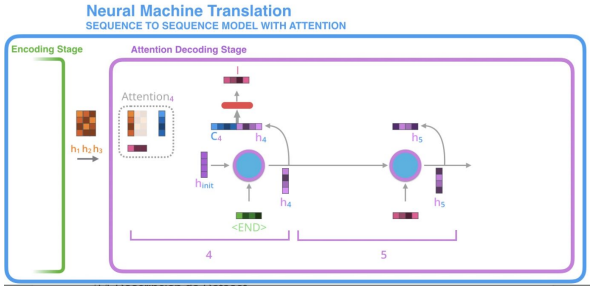
Se presenta el diseño del *decoder* en color morado. La salida del *encoder* (color verde) son estados latentes h_j que se usan como entrada del *decoder*. En la primera instancia se tiene una RNN (círculo morado con azul) la cual es alimentada con un estado inicial del *decoder*, mismo que se define de la siguiente forma $tanh(W_s, \leftarrow h_1)$ y con el *embedding* del estado final del *encoder* <end>. De esta forma se entrena la RNN.

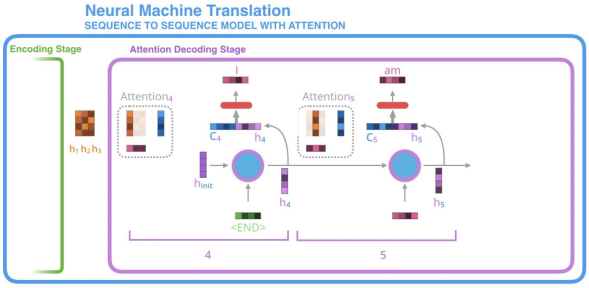




La salida de la RNN es un nuevo estado oculto, el cual es utilizado para el cálculo del vector contexto, que como se vio antes, se hace mediante el uso del *score*. Una vez obtenido el vector contexto, se concatena junto con la salida obtenida de la RNN; y con este nuevo tensor se alimenta una red densa la cual devuelve la probabilidad de que la palabra sea y_{Ty} para T_y palabras en el embedding de salida, es decir, como si fuera la salida de un clasificador con varias categorías.

En el siguiente paso, se toma el vector de contexto concatenado con el vector de *embedding* de salida y esto lo usa como entrada la RNN para calcular un nuevo estado oculto.





Finalmente,se utiliza el estado oculto resultante para hacer los cálculos del vector de contexto y obtener el vector de probabilidades. De esta forma se obtiene la nueva palabra de la traducción. Esta idea continúa hasta que obtener el <end> correspondiente al *embedding* de la traducción.[1]

[1]Las imágenes utilizadas en esta página fueron obtenidas de [6].

3. Prueba Dataset/Resultados

En línea con el objetivo establecido, en esta sección se describe la implementación realizada de NMT con mecanismos de atención para lograr obtener traducciones del español al inglés.

3.1 Descripción de Dataset

El conjunto de datos utilizado fue tomado de [4] y contiene 118,964 pares de traducciones Inglés-Español que van desde una sola palabra hasta frases formadas por más de 40 palabras. Todas estas frases tuvieron que ser procesadas conforme a los siguientes pasos, para poder ser usadas en el modelo:

1. Las palabras fueron convertidas a minúsculas y a formato ascii.
2. Se crearon espacios entre signos de puntuación y palabras.
3. Los caracteres especiales de cada frase fueron eliminados, excepto letras de la a a la z, puntos, comas, signos de exclamación y de interrogación.
4. Se añadió un token al inicio <start> y fin de cada oración <end>.
5. Adicionalmente fue necesario generar un tokenizador para frases en inglés y otro para frases en español. La manera de construir estos tokenizadores fue a través de la frecuencia de las palabras en el conjunto de datos. El número 1 fue asignado a la palabra con mayor frecuencia, el número 2 a la segunda palabra más frecuente y así sucesivamente. Estos tokenizadores permiten la búsqueda por id o por palabra.
6. Se completó cada oración de forma que todas tuvieran el tamaño de la frase de longitud máxima. Esto *padding* es realizado hacia la derecha.

El 80% de las frases fue utilizado para formar el conjunto de datos de entrenamiento y el resto formó parte del conjunto de datos de prueba.

3.2 Fase de entrenamiento

El modelo implementado consideró una red neuronal recurrente GRU de 1,024 unidades tanto para el *encoder* como para el *decoder*. Se fijó un *batch size* de tamaño 64, un *embedding* de dimensión 256, optimizador Adam y se corrió durante 10 épocas. La función de pérdida utilizada corresponde a una *sparse categorical cross entropy* debido a que las frases de tamaño pequeño tienen un *padding* de ceros para poder tener una longitud igual a la frase de longitud máxima.

De manera general se puede resumir el procesamiento por cada batch de la siguiente manera:

ENCODER.

1. Recibe como *input* dos elementos; un tensor del tamaño del *batch* con frases en español; y un estado oculto inicial que se inicializa en ceros. Dentro de esta etapa se aplica un *embedding* a las frases que fueron recibidas como *input*. El output del encoder comprende también dos elementos; por un lado devuelve un tensor que será llamado *encoder output* y por otro un estado oculto. El *encoder output* tiene dimensiones (tamaño del *batch*, longitud de la secuencia, número de unidades de la red).

De manera iterativa se recorrerán los elementos de la secuencia correspondiente al batch target.

DECODER.

2. Recibe como input tres elementos: output del *encoder*, estado oculto del *decoder* y *decoder input*.. En la primera iteración, el estado oculto del *decoder* corresponde al estado oculto del *encoder*, para las siguientes iteraciones, el *decoder* producirá sus propios estados ocultos y se alimentará de ellos. Asimismo en la primera iteración el *decoder input* corresponde al token <start> de la secuencia *target*; para iteraciones posteriores este input será el token correspondiente de la secuencia *target*.
3. A través del mecanismo de atención introducido por Bahdanau [1] se calcula el *score* del estado oculto, se aplica *softmax* a dicho *score* y finalmente el mecanismo devuelve tanto el vector contexto como los pesos de atención. El cálculo del *score* lo realiza a través de: $FC\{tanh(FC(EO)+FC(H))\}$, en donde FC representa una capa densa, EO se refiere al *output* del *encode*r en la primera iteración y en posteriores corresponde al propio estado oculto del *decoder*. Finalmente H representa el estado oculto del *encoder*. El número de unidades de las capas densas son 1, 1024 y 1024 leyendo el *score* de izquierda a derecha.
4. El *decoder* devuelve las predicciones y el estado oculto del *decoder*.
5. Se calcula la función de pérdida sobre las predicciones y este valor se va acumulando.
6. El estado oculto del *decoder* es reutilizado por el modelo para la siguiente iteración.

Al finalizar las iteraciones sobre el batch completo, se calcula la pérdida del batch, los gradientes y se optimizan los pesos de la red a través del *backpropagation*.

3.3 Pruebas del modelo

Con el fin de conocer el desempeño del modelo, se decidió llevar a cabo dos tipos de pruebas. En primer lugar se consideraron frases pertenecientes al conjunto de datos de prueba (*test*) y como segunda opción se incluyeron frases arbitrarias, es decir, frases que no formaron parte ni

del conjunto de entrenamiento ni del conjunto de prueba. Es importante hacer notar que las frases arbitrarias deben estar formadas únicamente por palabras que forman parte del *corpus* de entrenamiento.

3.3.1 Frases del conjunto de prueba

La primera frase que se considera es una frase corta, compuesta de 7 tokens:

- **input:** ¿Qué tal estuvo tu paseo?
- **output:** how was your walk?

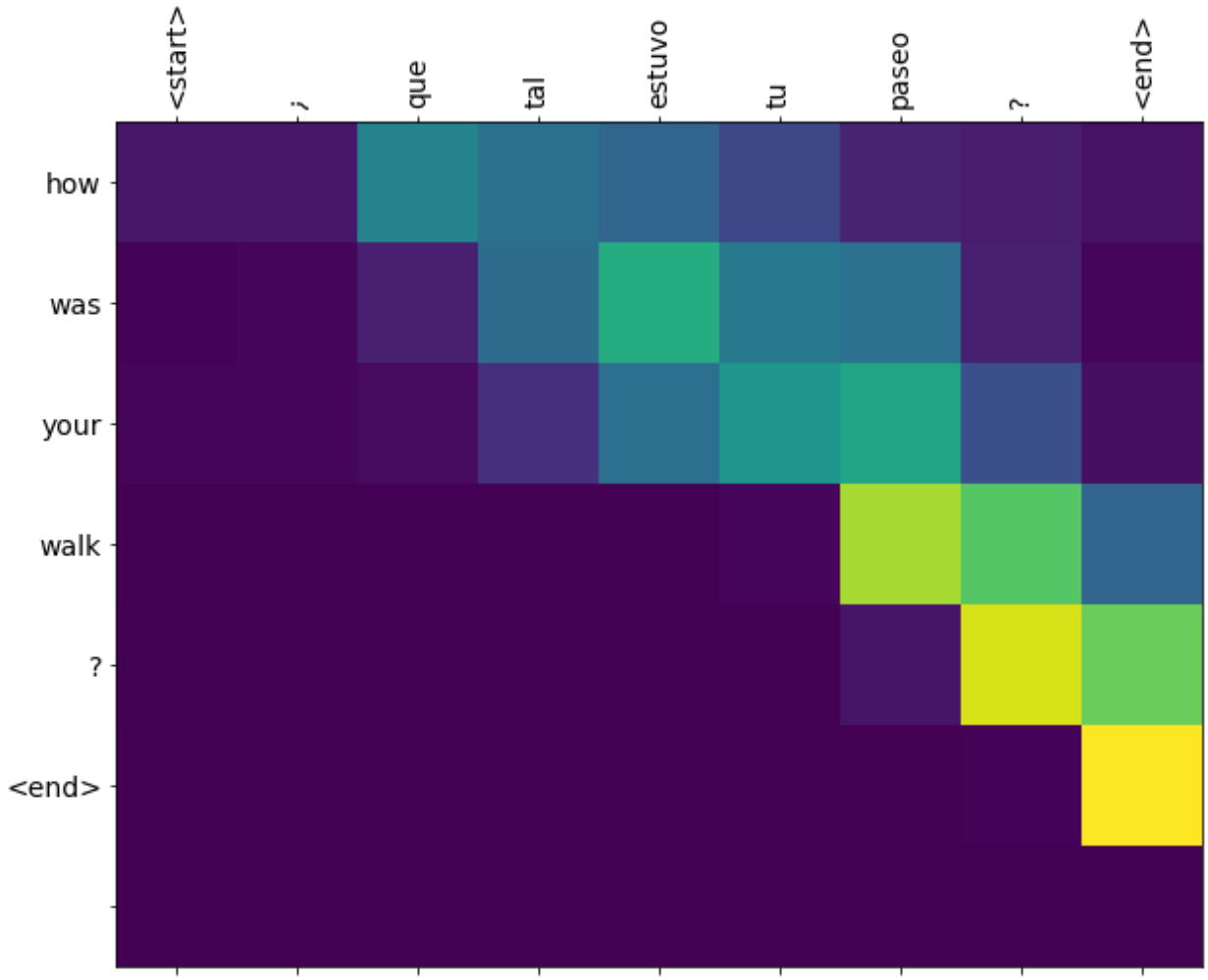


Fig 3. Matriz de pesos de atención de una frase de entrada de 6 tokens.

La interpretación de la matriz de la Figura 3 implica reconocer que cada fila muestra los pesos asociados con los estados ocultos. A partir de esto es posible observar cuáles posiciones de la

secuencia de entrada fueron consideradas más importantes cuando se generó la palabra *target*. Con respecto a los colores, el color morado representa pesos de atención igual a cero y el color amarillo indica pesos de atención iguales a 1. Los colores intermedios toman valores entre 0 y 1; en particular los tonos azules son más cercanos a 0, mientras que los tonos verdes son cercanos a 1.
Particularmente, la palabra *target* *walk* pone más atención en la palabra de entrada *paseo*. La RNN en este caso fue capaz de alinear correctamente la palabra *walk* con la palabra *paseo*, saltando la palabra *tu*.
Este ejemplo demuestra que el modelo es capaz de lidiar con frases de entrada y de salida de distintas longitudes.

Adicionalmente se revisa el resultado obtenido al considerar una frase de 24 tokens:

- **input:** una manera de reducir el número de errores en el corpus de tatoeba sería fomentar que la gente tradujera únicamente a su lengua materna.
- **output:** a game of atomic number , the tatoeba corpus would be to encourage people to only translate into their native languages.

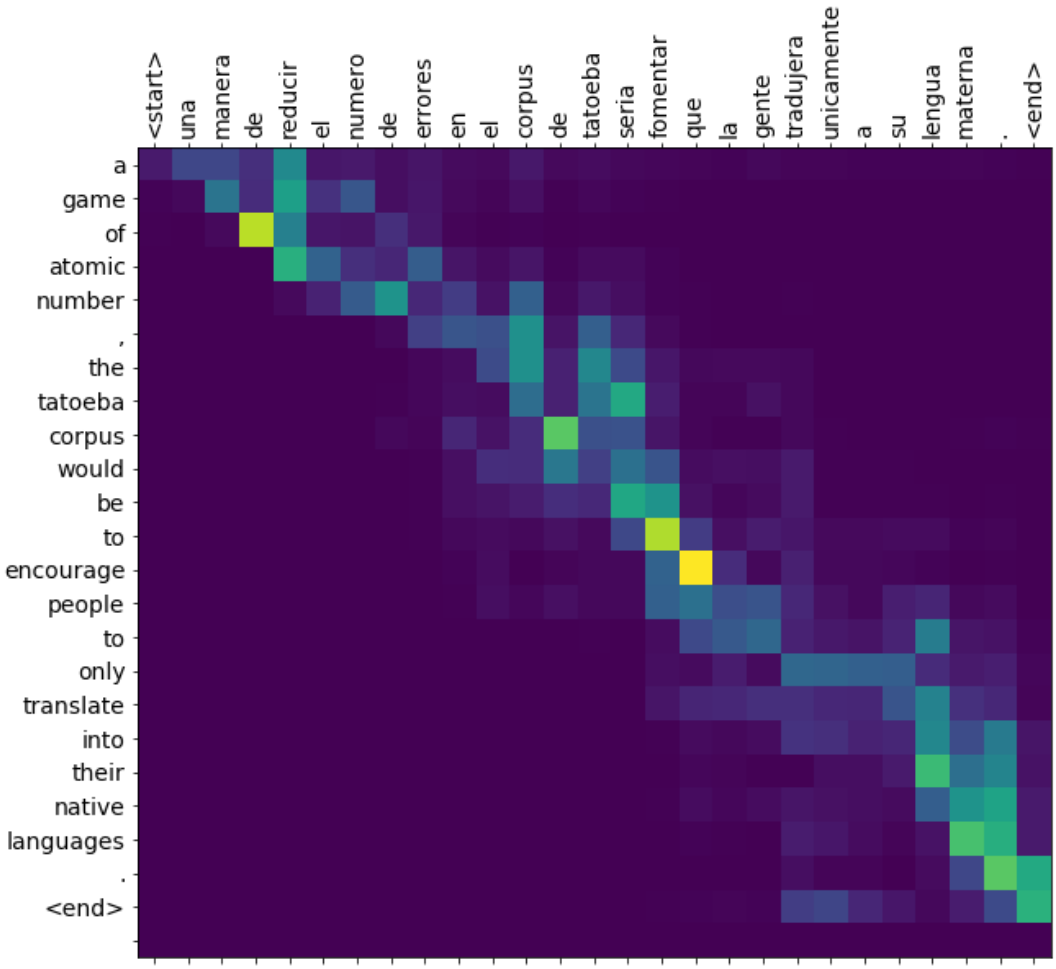


Fig 4. Matriz de pesos de atención de una frase de entrada de 24 tokens.

La frase mostrada en la Figura 4 es seis veces más larga que la de la Figura 3, por lo tanto es un poco más tedioso interpretar la matriz de pesos asociada. Es notable que la traducción devuelta por el modelo no corresponde totalmente a la traducción de la secuencia de entrada; sin embargo vale la pena notar que la frase *output* incluye una coma que no fue dada en el *input*. La parte previa a la coma es la que no coincide con la traducción de las primeras palabras de la frase *input*; pero la parte posterior a la coma muestra una mejor traducción de la frase inicial.
Otro punto importante radica en que la palabra *target translate* es la que tiene asociado el mayor número de pesos de atención distintos de cero, es decir, es la que tomó en cuenta más elementos para poder ser generada.

3.3.2 Frases arbitrarias

De manera análoga a las frases anteriores, en esta sección se consideran también frases de distinta longitud.

Un primer caso consiste en comparar dos frases que tienen las mismas palabras; pero la primera de ellas es ingresada como pregunta mientras que la segunda es una afirmación. En esta prueba se demuestra que la traducción no solo es realizada por las palabras contenidas en el enunciado, si no que el contexto que las rodea también influye :

Frase 1

- **input:** ¿todavía están en casa?
- **output:** are you still at home?

Frase 2

- **input:** todavía están en casa
- **output:** they re still at home

En la Figura 5 es posible apreciar las matrices de pesos asociadas a cada frase. En ambas situaciones, el modelo NMT fue capaz de traducir correctamente las secuencias de entrada. Nótese que los alineamientos entre las palabras del español y del inglés no se encuentran completamente en la diagonal, es decir, este alineamiento es no monotónico. Al momento de realizar una pregunta, la posición de los pronombres en inglés varía con respecto a la posición de los mismos en español.

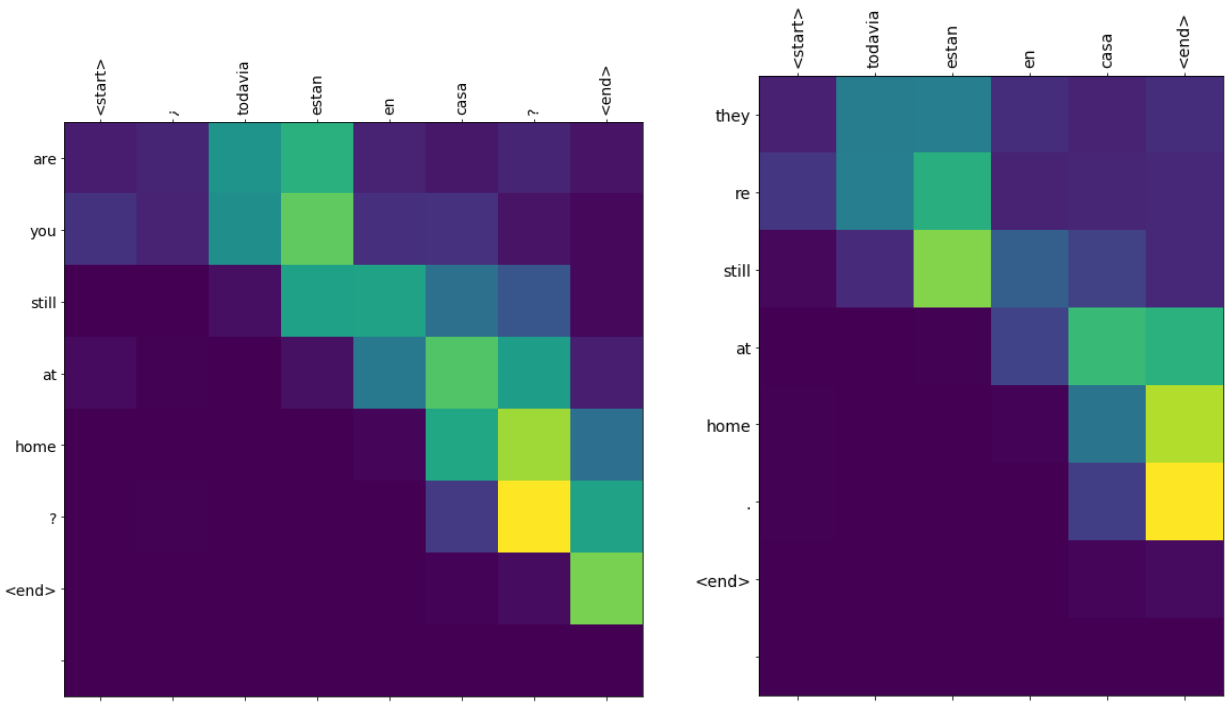


Fig 5. Matrices de pesos de atención de una frase de entrada de 6 y 4 tokens respectivamente.

Para la última prueba mostrada en el reporte se eligió una frase formada por 12 palabras, de forma que fuera posible probar el *performance* de la red en situaciones de longitud media.

- **input:** es mejor estar aproximadamente en lo cierto que estar completamente equivocado.
- **output:** it is better to be approximately right than to be completely wrong.

La matriz de pesos de la frase de longitud media se muestra en la Figura 6. Nuevamente se observa que la longitud de la frase de entrada no es la misma que la de la frase traducida.
Por otro lado los alineamientos de las palabras de ambas frases pareciera ser monotónico, es decir, los pesos de atención más altos se concentran en la diagonal. Por último; pero no menos importante, se observa que la traducción realizada por el modelo es correcta.

Los ejemplos aquí mostrados sugieren que el modelo NMT con Atención captura el contexto de las frases de longitud corta (alrededor de 6 tokens). Con respecto a las frases de mayor longitud se observan deficiencias en algunas partes de la traducción. Adicionalmente, el modelo logra aprender la estructura gramatical del idioma de traducción.

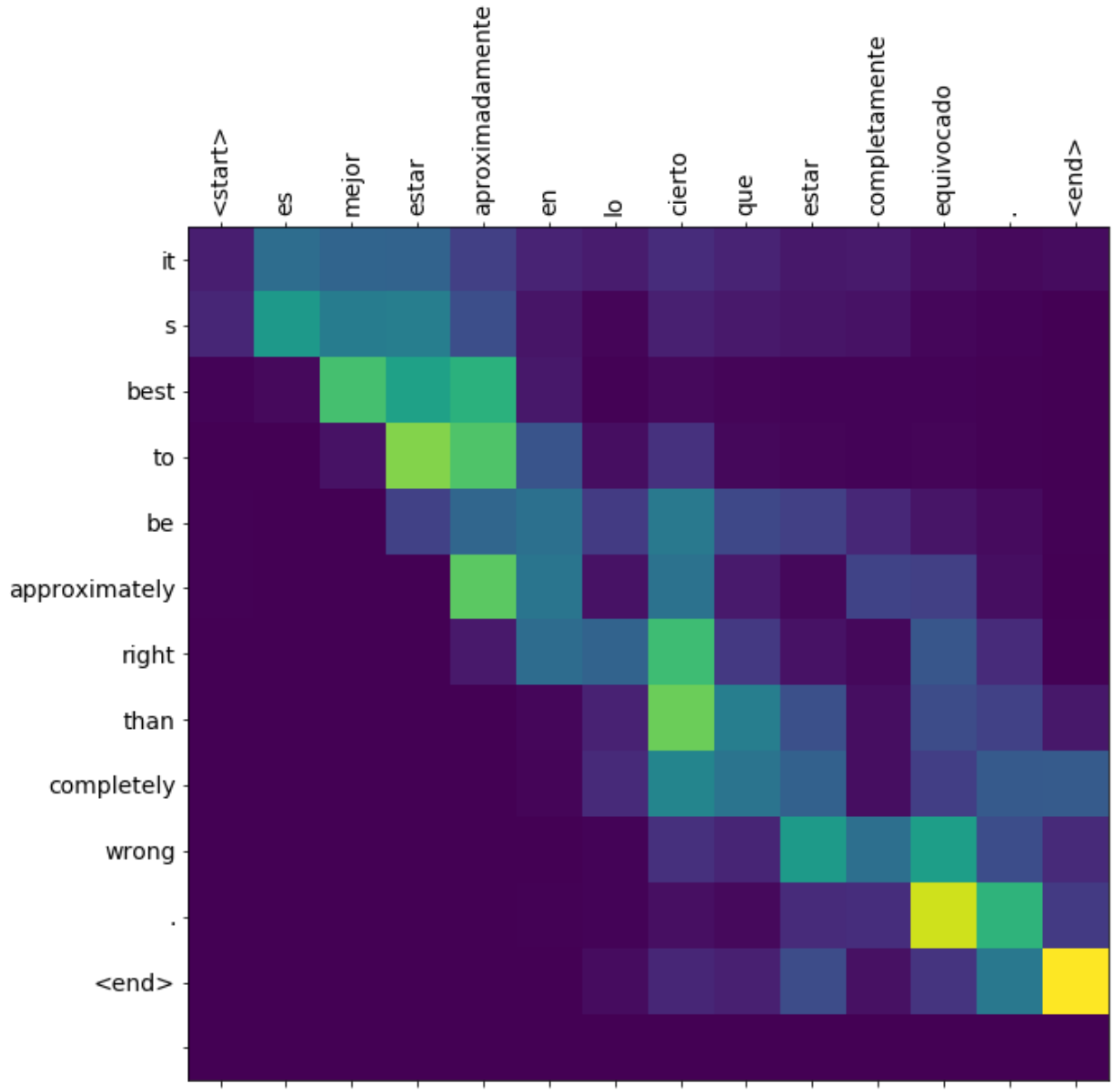


Fig 6. Matriz de pesos de atención de una frase de entrada de 12..

4. Conclusiones

La idea planteada por Bahdanau et al., 2014 [1] implicó una revolución en la forma de traducir los textos, llevándolos al estado del arte en ese momento. Cambiar la idea que se tenía del vector contexto por uno donde ahora considera una suma ponderada de todos los estados ocultos del encoder, condujo a mejoras de traducción en textos de larga longitud e incluso al día de hoy los mecanismos de atención siguen siendo utilizados en arquitecturas más modernas, como son las transformers. Esta idea no solo es útil en traducción de textos,

también ha sido empleada en la tarea para descripción de imágenes y otras arquitecturas que usen modelos *sequence-to-sequence.*

Aunque los resultados expuestos en este trabajo no tienen el mejor rendimiento, si son capaces de desempeñarse de una manera aceptable con poco costo computacional (~4hrs con GPU proporcionada por colab) y reflejan de buena forma las bondades de utilizar mecanismos de atención; como son, entender reglas gramaticales y cambiar el sentido de una traducción en el caso en el que preguntamos o afirmamos algo.

Referencias

[1] D. Bahdanau, Ky. Cho and Y. Bengio (2015). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv*

[2] M. Luong, H. Pham and C. D. Manning(2015). Effective Approaches to Attention-based Neural Machine Translation. arXiv

[3] Alammar, Jay (2018). The Illustrated Transformer [Blog post]. *Retrieved from https://jalammar.github.io/illustrated-transformer/*

[4] Kelly, Charles (2020). Tab-delimited Bilingual Sentence Pairs. *Retrieved from http://www.manythings.org/anki/*

[5] Luong, Thang (2017). Neural Machine Translation (se2seq) Tutorial. *Retrieved from https://github.com/tensorflow/nmt*

[6] Alammar, Jay (2018). Visualizing A Neural Machine Translation Model (Mechanisc of Seq2seq Models With Attention). [Blog post]. *Retrieved from https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/*

Recursos:
Github: NeuralMachineTranslation_esp-ing