Maestría en Inteligencia Artificial Aplicada (MNA)

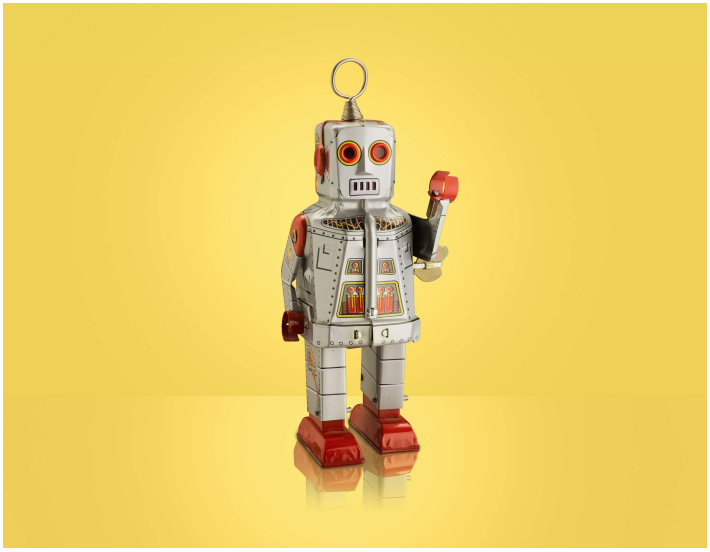
model Seq2Seq : Machine Translation

– Context Vector – Encoder-Decoder

Procesamiento de Lenguaje Natural (NLP)

Luis Eduardo Falcón Morales

Modelos seq2seq, el Vector de Contexto, Encoder-Decoder y la máquina de traducción automática



Actualmente es muy conocido el término de “atención” dentro de los modelos Transformers, pero para llegar a dicho concepto primero surgieron los módulos Encoder-Decoder y el llamado “vector de contexto”, que estudiaremos en esta presentación.

El objtetivo es tartar de entender mejor los modelos secuencia a secuencia, seq2seq, es decir modelos donde la entrada es una oración y la salida es otra oración, es decir, ya no se analizar simplemente token por token.

Cuando observamos una imagen, visualmente recorremos principalmente ciertas regiones de la misma para tener una comprensión general de lo que hay o sucede en ella.

Ahora, dada una oración o un documento ¿en qué palabras nos debemos fijar para tener el significado principal de dicho texto?

Pinocho le dijo al Hada Azul que no había mentido

mayor atención: ——— ——— ———

menor atención: — — ——

Liga de referencia de la imagen de la diapositiva anterior:

https://es.m.wikipedia.org/wiki/Archivo:Eyetracking_heat_map_Wikipedia.jpg

El dilema de la relación de palabras en una oración:

Un dilema en el procesamiento del lenguaje natural es tratar de encontrar las palabras “clave” o más importantes que nos den el contexto general o significado de una oración. Igualmente está el reto de encontrar la manera en que se relacionan todas las palabras entre sí.

Todos los años, por el mes de marzo, una **familia de gitanos** **desarrapados** plantaba **su** carpa cerca de la aldea, y con un grande alboroto de pitos y timbales **daban a conocer** los nuevos inventos. Primero **llevaron** el imán…

¿Cómo diseñar un modelo de IA para que identifique que todas las palabras en rojo hacen referencia en la oración a la “familia de gitanos”?

Una persona lo podrá hacer sin problema, no así una máquina.

En los últimos años se ha generado mucha investigación en relación a este tema.

Extracto de Cien años de soledad (1967)
Gabriel García Márquez (1927-2014)

Sequence-to-Sequence : seq2seq

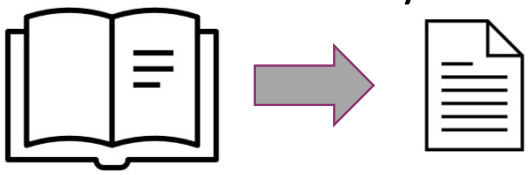
Google Translate

Handwritting Recognition

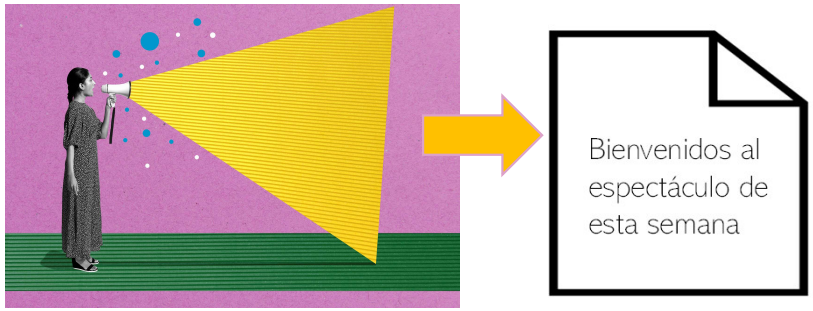
Image Captioning

Un niño en una bicicleta

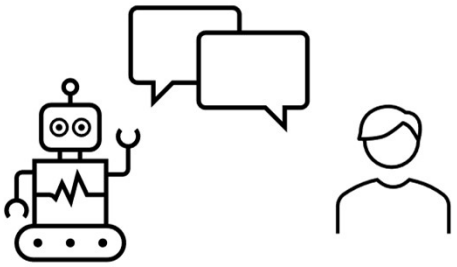
Summary



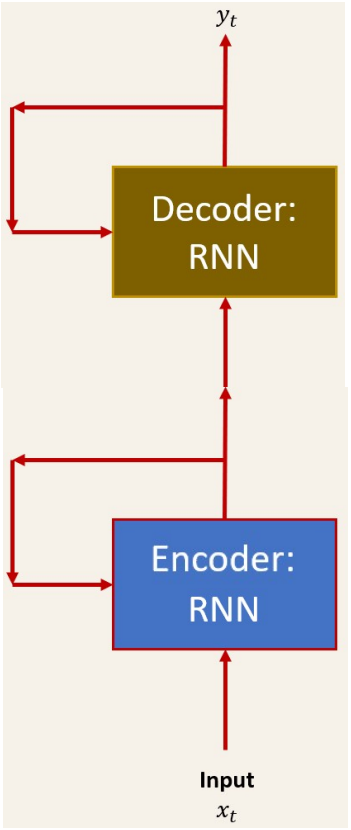
Reconocimiento de voz a texto



chatbot



Sequence-to-Sequence : modelos basados en RNNs : Encoder-Decoder



Decoder

Encoder

- Los primeros modelos para los problemas Seq2Seq están basados en dos RNN: la primera (encoder) encargada de sintetizar la información de cada token de la secuencia de entrada y la segunda (decoder) encargada de generar-predecir los tokens de salida.

- Si se trata de un traductor entre dos idiomas, cada idioma tiene su propio RNN que ayuda a sintetizar o predecir la información de los enunciados de entrada o salida, respectivamente.

- En lugar de seguir abordando el problema desde un punto de vista completamente estadístico, se hace uso ahora de las redes neuronales recurrentes.

Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation

Kyunghyun Cho
Bart van Merriënboer **Caglar Gulcehre** **Dzmitry Bahdanau**
Université de Montréal Jacobs University, Germany
firstname.lastname@umontreal.ca d.bahdanau@jacobs-university.de

Fethi Bougares **Holger Schwenk** **Yoshua Bengio**
Université du Maine, France Université de Montréal, CIFAR Senior Fellow
firstname.lastname@lium.univ-lemans.fr find.me@on.the.web

https://arxiv.org/abs/1406.1078

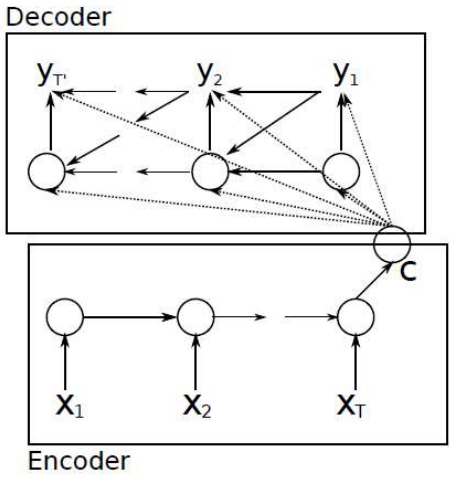


Figure 1: An illustration of the proposed RNN Encoder–Decoder.

Secuencias de palabras y el modelo Encoder-Decoder

Se propone el modelo **Encoder-Decoder** basado en dos RNN/LSTM.

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

Quoc V. Le
Google
qvl@google.com

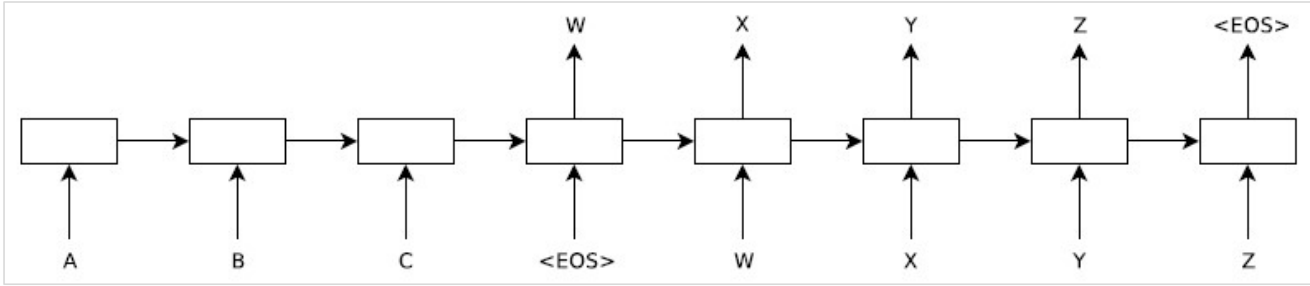
Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French

Seq2Seq

- Aparece el concepto sequence to sequence, seq2seq, en el modelado de lenguaje.
- Propuesta de mejora del modelo seq2seq basado en la pareja Encoder–Decoder.
- Se usan capas LSTM para enfrentar el problema del desvanecimiento de los gradientes.

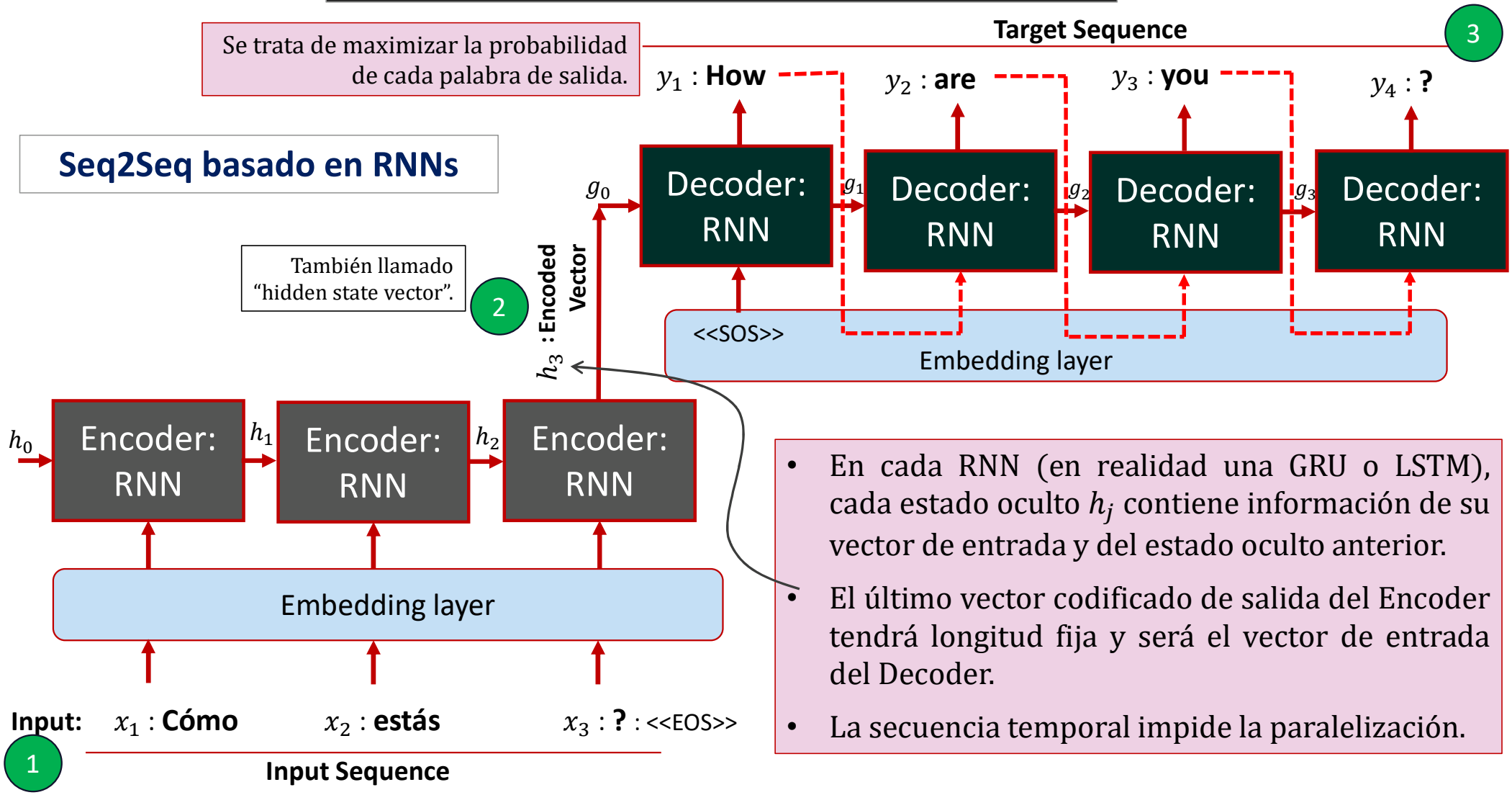
https://arxiv.org/abs/1409.3215



Encoder & Decoder : Codificador & Decodificador

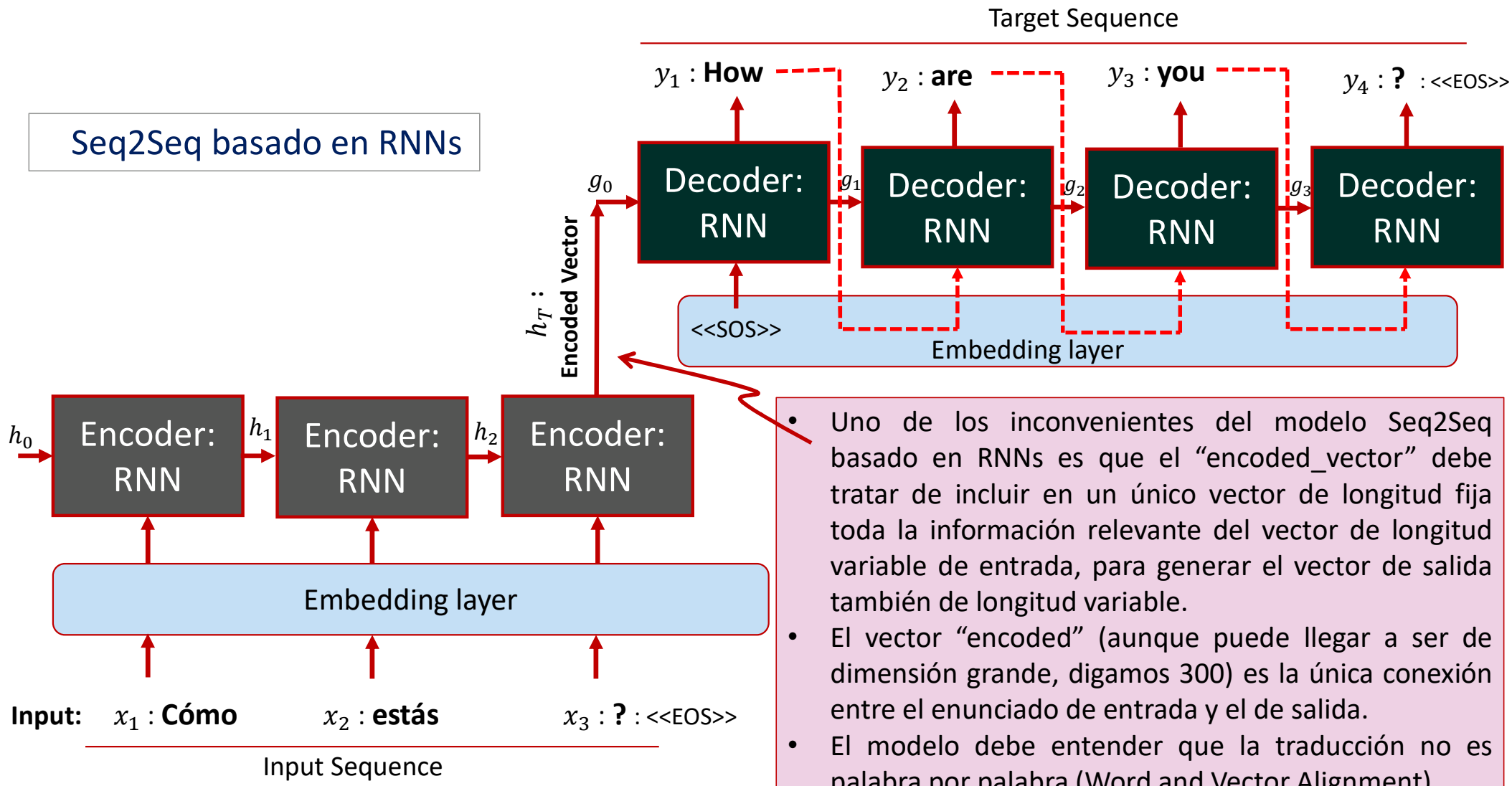
Se trata de maximizar la probabilidad de cada palabra de salida.

Seq2Seq basado en RNNs



- En cada RNN (en realidad una GRU o LSTM), cada estado oculto h_j contiene información de su vector de entrada y del estado oculto anterior.
- El último vector codificado de salida del Encoder tendrá longitud fija y será el vector de entrada del Decoder.
- La secuencia temporal impide la paralelización.

Seq2Seq basado en RNNs

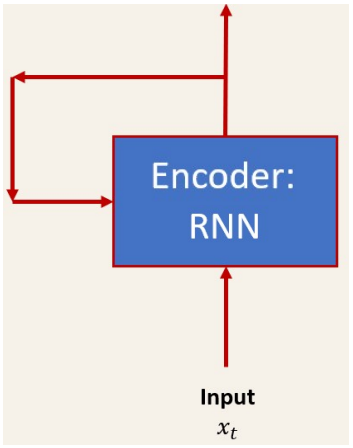


- Uno de los inconvenientes del modelo Seq2Seq basado en RNNs es que el "encoded_vector" debe tratar de incluir en un único vector de longitud fija toda la información relevante del vector de longitud variable de entrada, para generar el vector de salida también de longitud variable.
- El vector "encoded" (aunque puede llegar a ser de dimensión grande, digamos 300) es la única conexión entre el enunciado de entrada y el de salida.
- El modelo debe entender que la traducción no es palabra por palabra (Word and Vector Alignment).

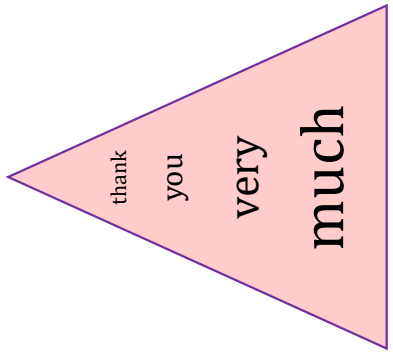
El problema del cuello de botella (information bottleneck):

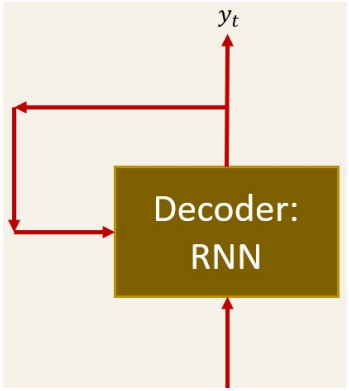
En enunciados muy largos las capas RNN, GRU, LSTM van perdiendo la información de las primeras palabras por el problema de la longitud fija de la memoria (encoded vector) y porque no toma en cuenta la longitud variable del enunciado de entrada y el de salida.

Se sabe que en general el desempeño de un modelo basado en un vector "encoded" de longitud fija, empieza a decaer con enunciados mayores a 20 palabras.



Encoder





Decoder

… esto llevó a buscar alternativas para enfrentar este problema de cuello de botella…

Context Vector & Attention

Introducción del concepto de Attention mediante un vector de contexto de longitud variable y que ajusta la información del enunciado de entrada.

Published as a conference paper at ICLR 2015

https://arxiv.org/abs/1409.0473

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

Se agregará un vector que nos indique cuáles palabras son las más importantes de un enunciado.

The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \quad (5)$$

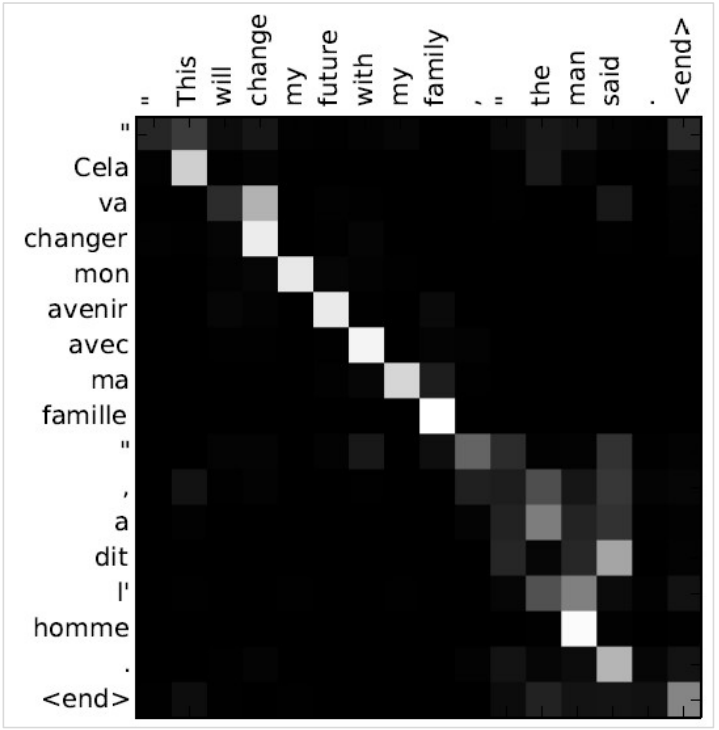
The weight α_{ij} of each annotation h_j is computed by

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

where

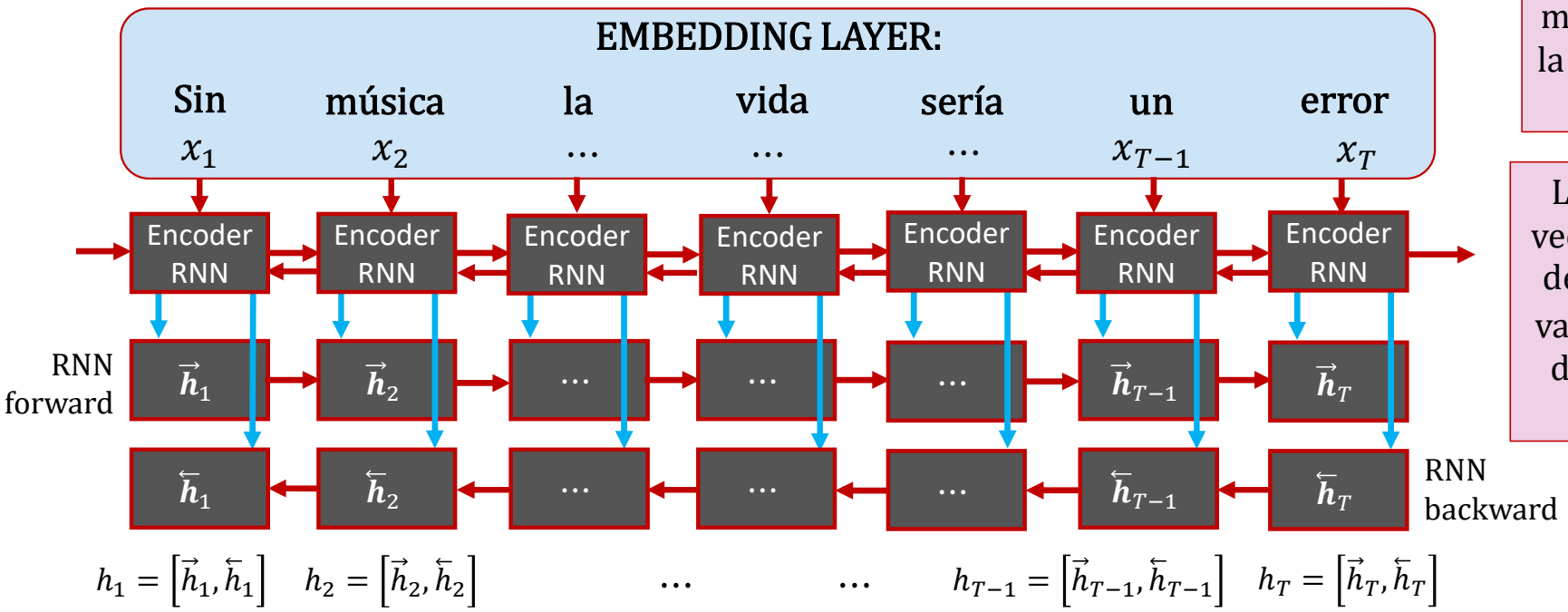
$$e_{ij} = a(s_{i-1}, h_j)$$

Inicialmente se introdujo este concepto de Attention para permitir, en el problema de traducción English-to-French, usar enunciados más largos.



Context Vector : Concepto de Atención : Attention

Encoder bi-direccional: tomará en cuenta no solo las palabras previas a una dada, sino también las que le siguen. Una RNN lee el enunciado hacia adelante y calcula los estados ocultos $\vec{h}_j$ y la otra RNN lo lee en sentido inverso y calcula los $\overleftarrow{h}_j$.



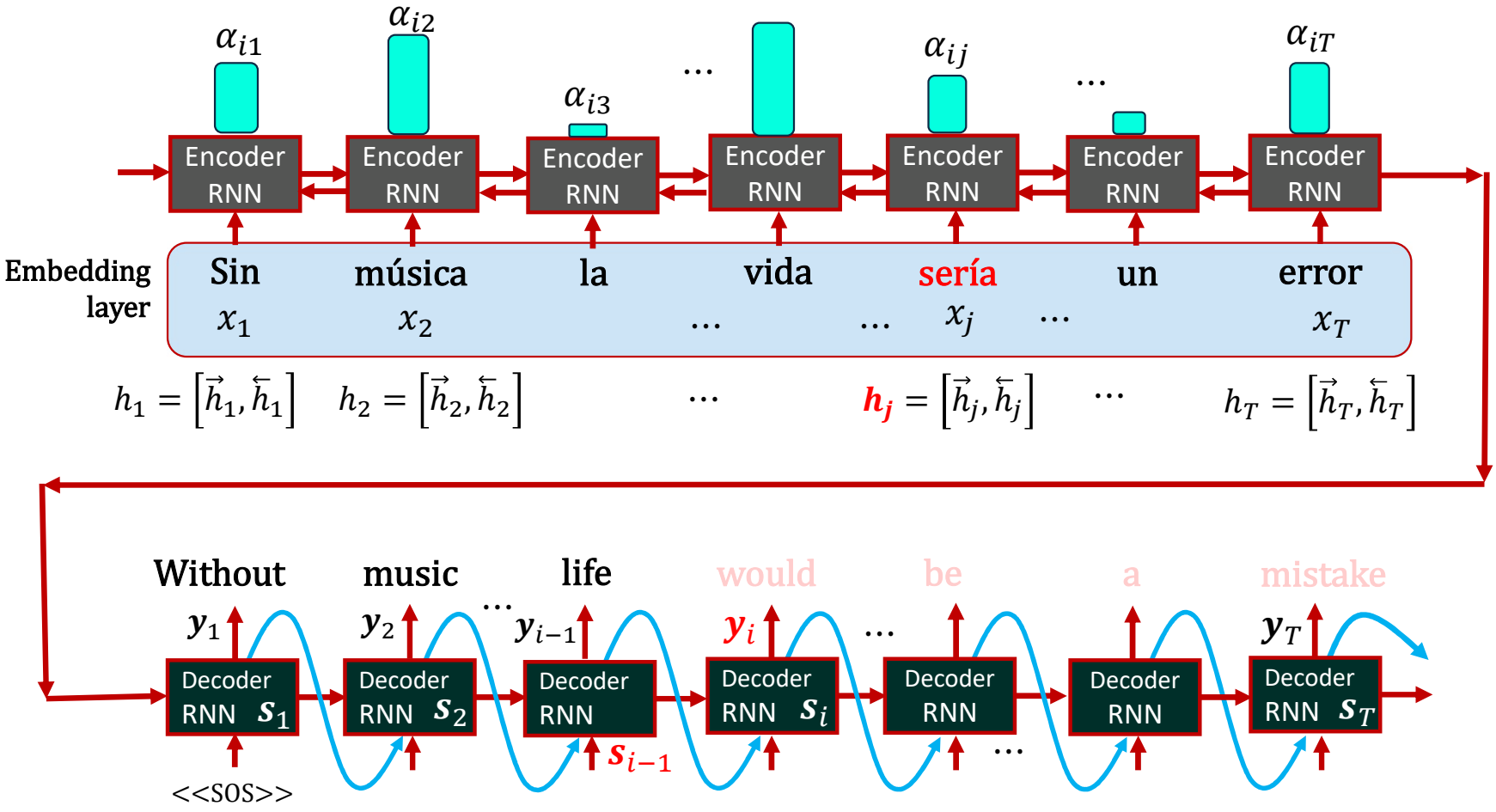
Todos los estados $\vec{h}_j$ y $\overleftarrow{h}_j$ tienen la misma dimensión, la dada por la capa Embedding.

La longitud T del vector concatenado de anotación h_j es variable y depende del enunciado de entrada.

Para cada palabra x_j se obtiene su **anotación** (***annotation***) concatenando ambos estados: $h_j = [\vec{h}_j, \overleftarrow{h}_j]$.
Estas anotaciones h_j serán usadas para definir el vector de contexto c_j de cada palabra de salida.

Para cada palabra a predecir y_i en el Decoder, se calculan **los pesos** $\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}$,

donde $e_{ij} = a(s_{i-1}, h_j)$. Estos pesos varían con cada palabra a predecir y_i.



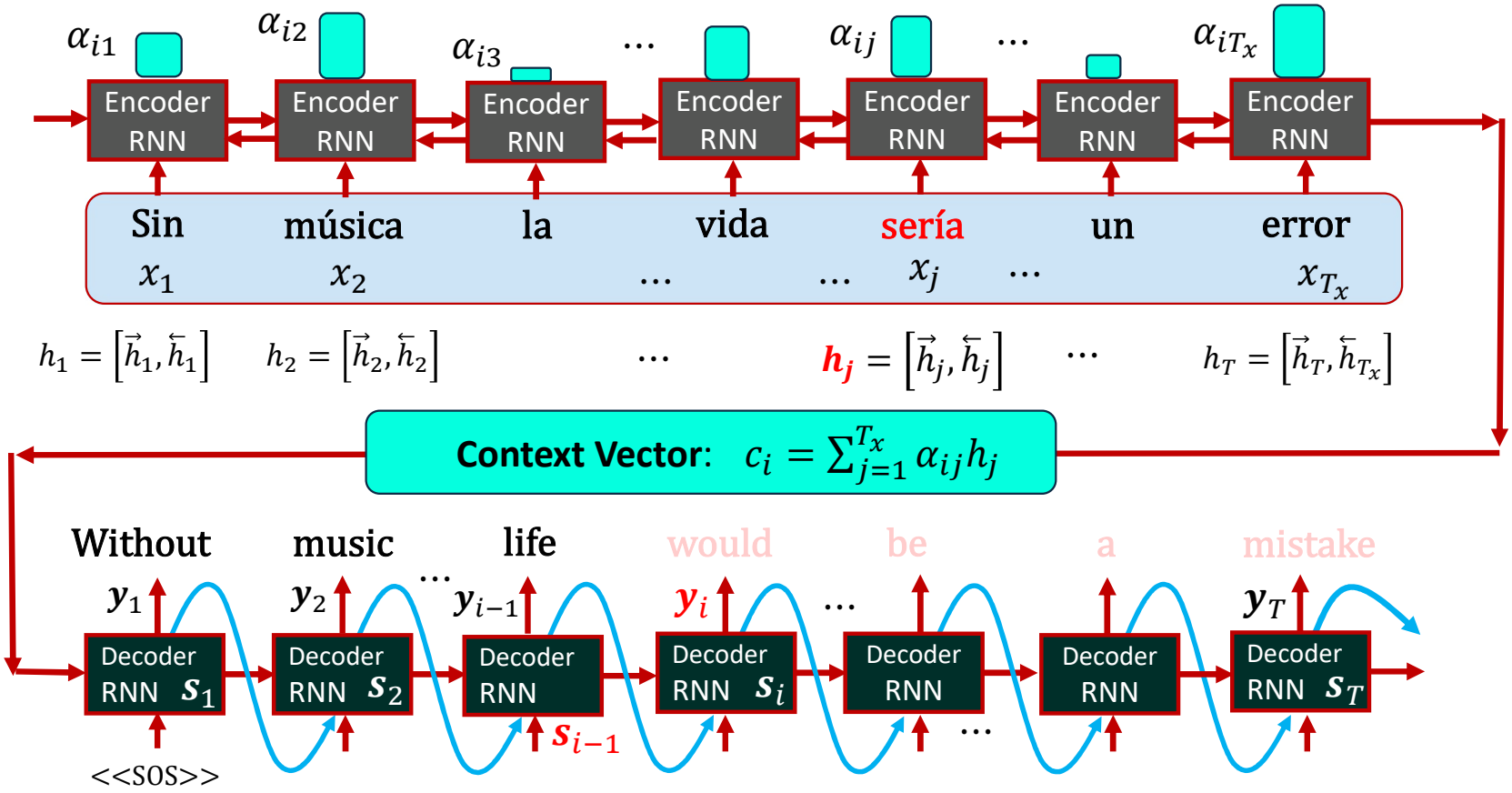
Con cada nueva palabra a predecir y_i, se calculan y combinan los pesos α_{ij} dependientes del estado s_{i-1} del Decoder.

Para cada palabra a predecir y_i en el Decoder, se calculan los pesos $\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}$,

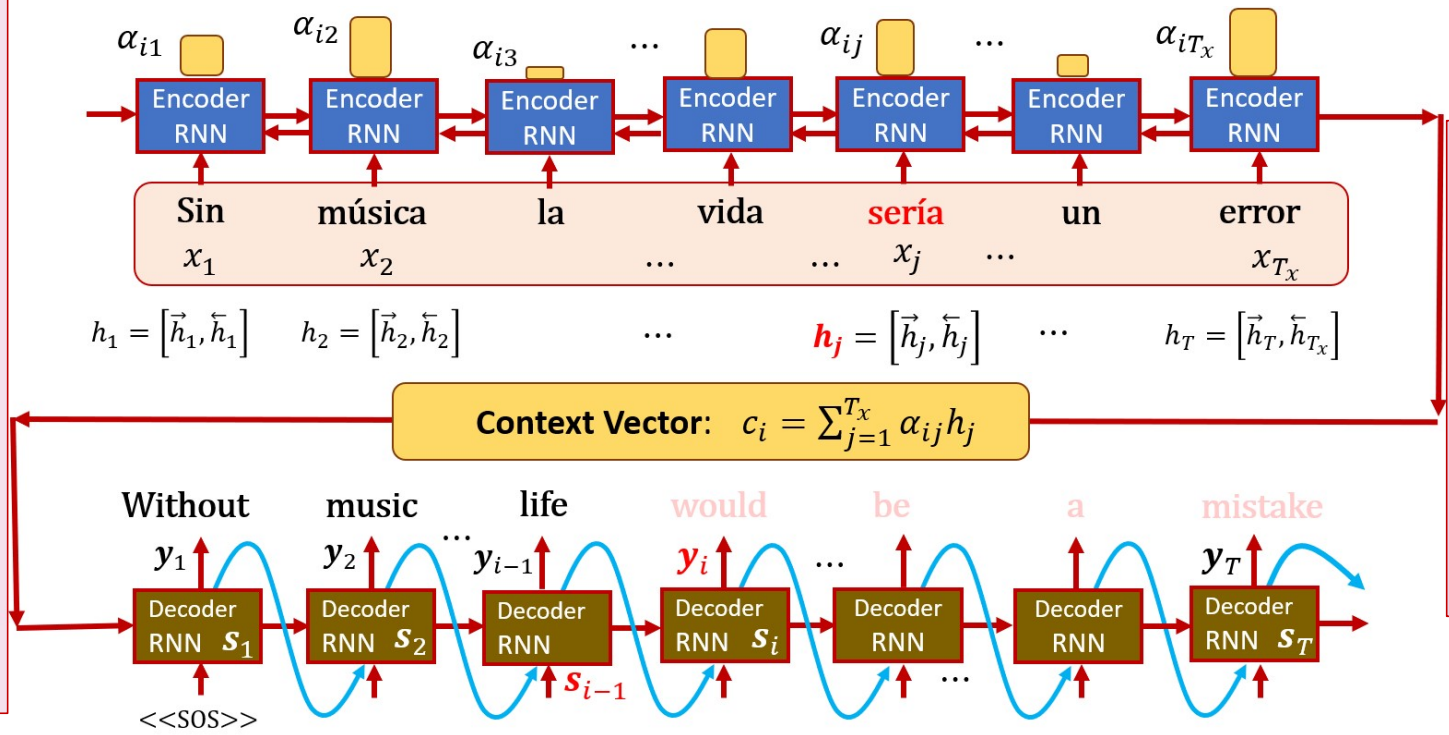
donde $e_{ij} = a(s_{i-1}, h_j)$. Así, cada **vector de contexto** c_i se define como $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$, los cuales se van generando al ir prediciendo cada palabra de salida y_i.

Cada vector c_i permite poner mayor "atención" en aquellas palabras del enunciado de entrada del Encoder que contienen información relevante (verbo, tiempo, sujeto, etc.) en relación a la siguiente palabra a predecir y_i.

Se dice entonces que el vector y_i se **alinea** con el vector x_j.



El Decoder tiene ahora este mecanismo de **atención** mediante el vector de contexto, que empezará a ser clave en los modelos por venir y que se define como un valor esperado sobre todas las anotaciones h_j con sus probabilidades α_{ij}.



De esta manera, el Encoder ya no tiene toda la responsabilidad de poner toda la información del enunciado de entrada en un solo vector de longitud fija.

Cada vez que se desea predecir una nueva palabra (token), se busca en cuál o cuáles de las palabras del enunciado de entrada está la nueva información más relevante, actualizando el vector de contexto. Así, el modelo podrá predecir ahora la siguiente palabra y_i, con base a este vector de contexto c_i, el estado de salida previo $\boldsymbol{s}_{i-1}$ y las palabras que se han predicho hasta ahora $y_1, y_2, \ldots, y_{i-1}$.

La predicción de esta palabra y_i se basa en una red neuronal que se entrena de manera supervisada con toda la etapa de entrenamiento del modelo completo.

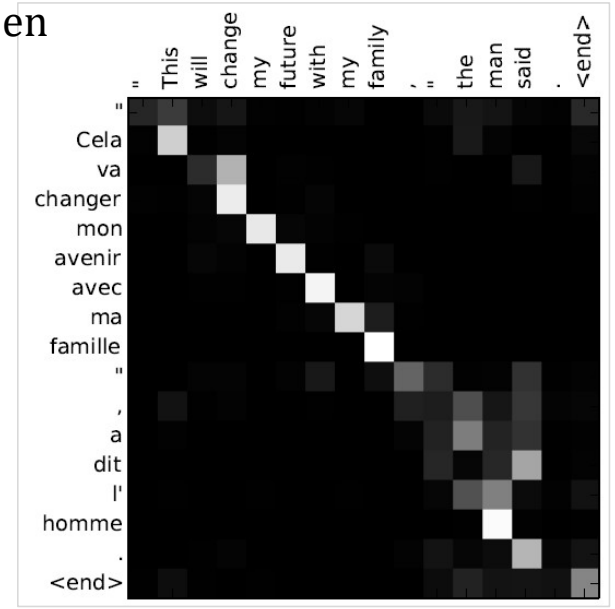
En resumen, el modelo de alineación $e_{ij} = a(s_{i-1}, h_j)$ la definen en el artículo de manera análoga a la manera en que se definen las compuertas del modelo LSTM:

$$e_{ij} = v_a^T tanh(W_a \boldsymbol{s}_{i-1} + U_a h_j)$$

Como el vector $\boldsymbol{s}_{i-1}$ solo es el estado oculto previo del Decoder, se debe repetir las veces que sean suficientes para que iguale la dimensión de h_j.

A partir de estos vectores de alineación se definen los pesos probabilísticos de la siguiente palabra a predecir y_i, con base las palabras alrededor de x_t en el Encoder:

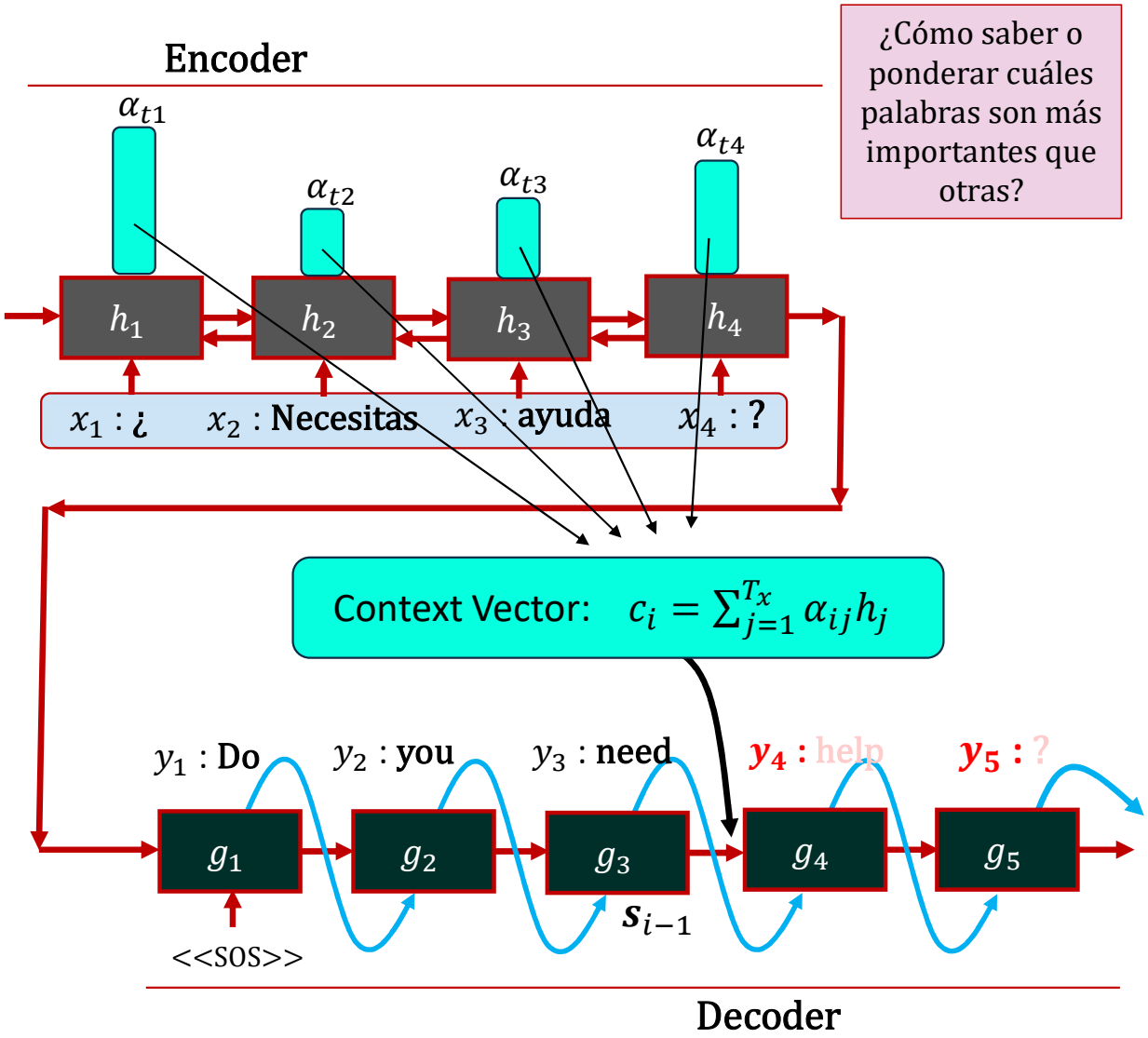
$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}$$



Y finalmente el vector de contexto:

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

Definido como un **valor esperado** sobre todas las anotaciones h_j con sus probabilidades α_{ij}.



El problema de la traducción de un idioma a otro de un enunciado $\vec{y}$ se reduce ahora a un problema de probabilidad condicional: predecir la siguiente palabra (token) y_i, a partir de las palabras predichas previamente $y_1, y_2, \ldots, y_{i-1}$ y del vector de contexto c_i, maximizando el siguiente producto:

$$p(\vec{y}) = \prod_{i=1}^{T} p(y_i \mid y_1, \ldots, y_{i-1}, c_i)$$

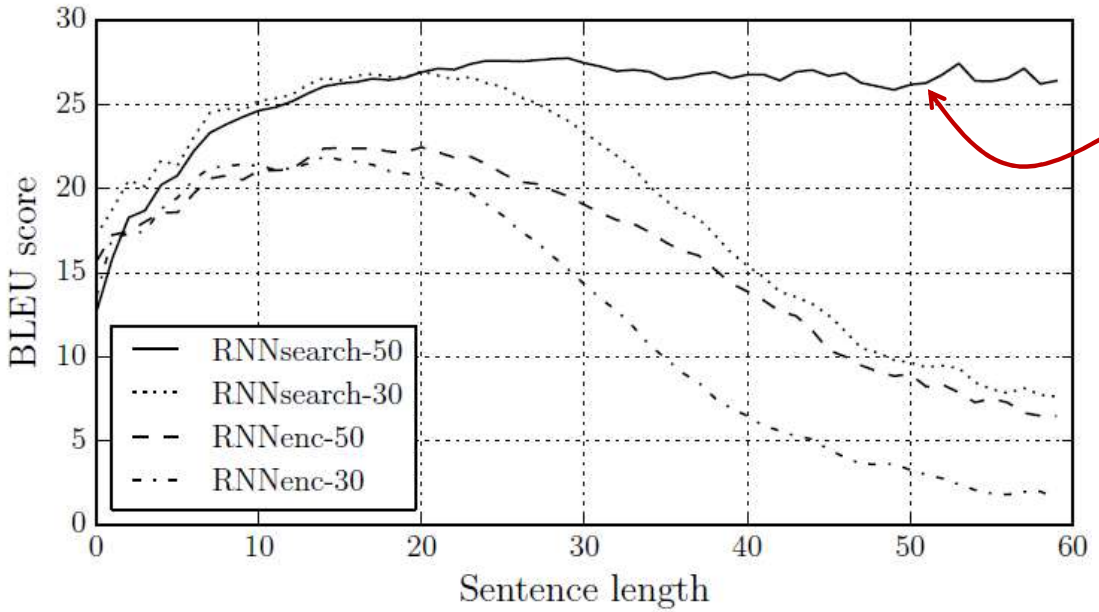
Cada y_i fue la salida probabilística de una red neuronal con 1000 neuronas artificiales en la capa oculta y ajustada mediante backpropagation junto con el entrenamiento de todo el sistema.

El modelo propuesto en el artículo se utilizó para traducción de documentos de inglés al francés. Se entrenó y validó el modelo con cerca de 348 millones de palabras, obtenidas de documentos del parlamento europeo, de las Naciones Unidas, recopilación de documentos de la web, entre otros. El vocabulario de cada idioma se aproximó a las 30,000 palabras más frecuentes.

Palabras que no aparecieron en el vocabulario se sustituyeron por [UNK]. Se conservaron mayúsculas y minúsculas como tales y no se aplicó stemming o algo parecido.

Se utilizó la métrica BLEU y los resultados con el conjunto de prueba se muestran en la siguiente gráfica (valores más grandes del BLEU score implican un mejor modelo):

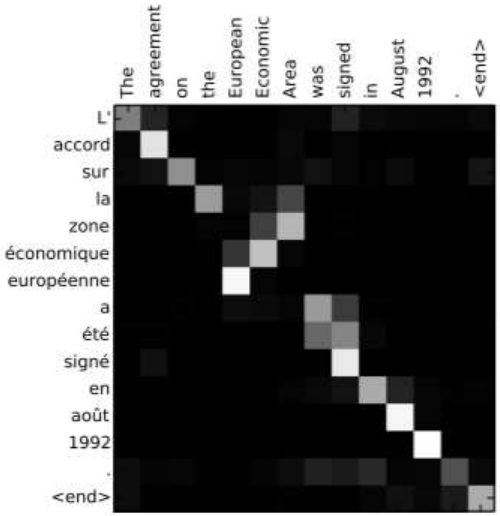
Observamos que a partir de enunciados con longitudes mayores a 20 o 30 palabras, los modelos previos con vector de contexto de longitud fija, empiezan a decaer en su desempeño.



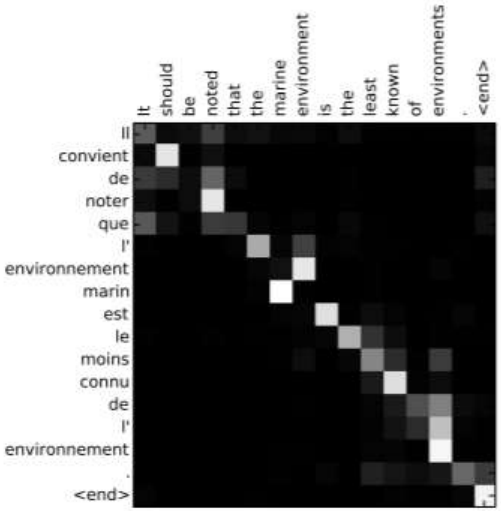
Modelo usando el vector de contexto de longitud variable y que se actualiza con cada nueva palabra a predecir por el Decoder.

Se usaron batches de 80 enunciados y el entrenamiento de cada modelo duró cerca de 5 días.

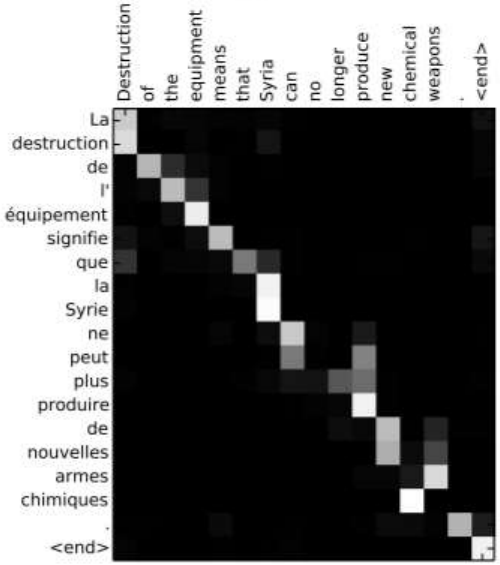
https://arxiv.org/abs/1409.0473



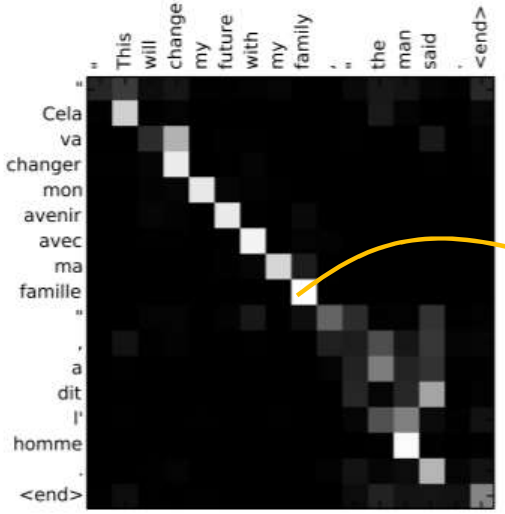
(a)



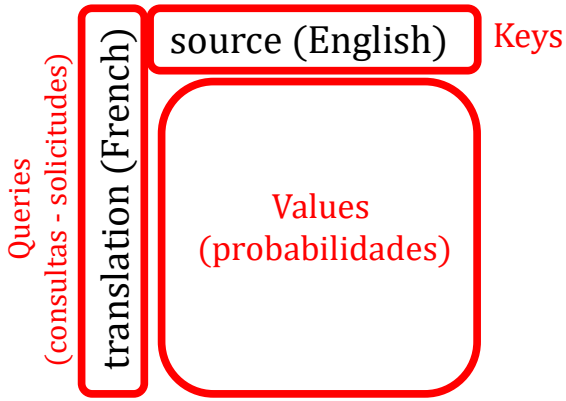
(b)



(c)



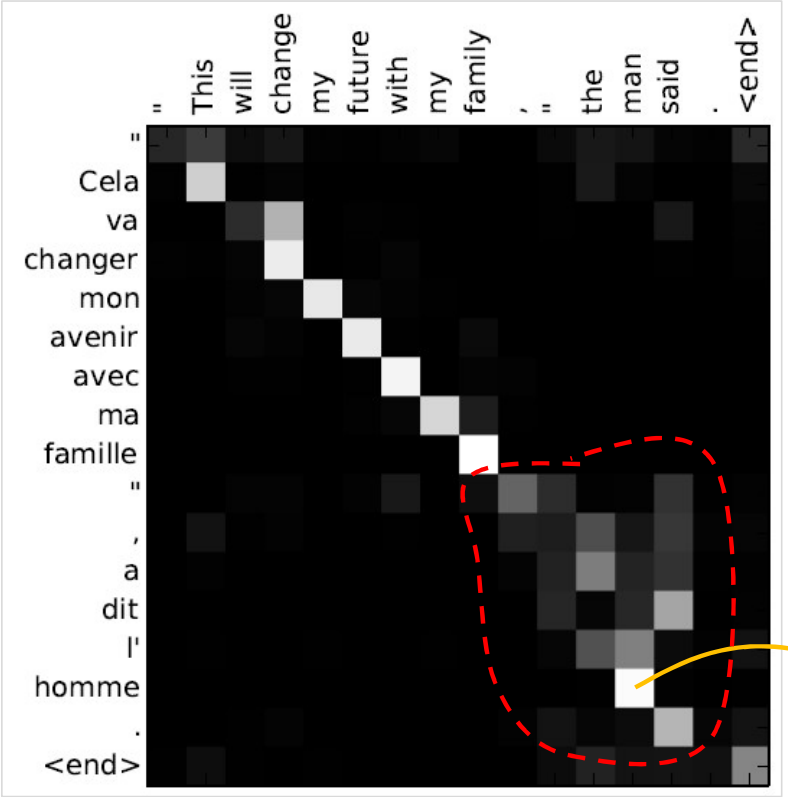
(d)



- Ejemplos de matrices de alineación que ilustran esa correlación existente entre las palabras de una secuencia de entrada y las de secuencia de salida.
- Observa que existe cierta “flexibilidad” en las correspondencias entre palabras.

$$\alpha_{tk} = \frac{exp\left(score\left(g_{t-1}, h_k\right)\right)}{\sum_j exp\left(score\left(g_{t-1}, h_j\right)\right)}$$

https://arxiv.org/abs/1409.0473



Observa cómo el modelo es capaz de identificar el diferente orden del sujeto (man/homme) y el verbo (said/dit) en cada idioma.

Cuando se utiliza el producto interior o punto vemos que palabras similares tendrán resultados numéricos mayores y cuando sean palabras no similares los valores de salida serán menores.

Al normalizarlos, se le da un significado probabilístico a dichas similaridades entre palabras.

Se buscan la o las palabras de salida (queries) que tengan mayor similaridad con las palabras de entrada (keys).

$$\alpha_{tk} = \frac{exp\left(score\left(g_{t-1}, h_k\right)\right)}{\sum_j exp\left(score\left(g_{t-1}, h_j\right)\right)}$$