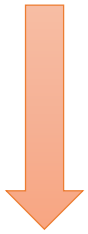
Visión Computacional

Ivan Sipiran

Machine Translation

El objetivo de machine translation es traducir la sentencia x en un lenguaje (source language) a la sentencia y en otro lenguaje (target language)

Juan limpia la casa para la fiesta de su cumpleaños



Juan cleans the house for his birthday party

Machine Translation

Antes de tener redes neuronales, la idea era aprender un modelo probabilístico desde los datos

Encontrar la mejor sentencia en inglés y , para una sentencia en español dada x

$$\underset{y}{\text{argmax}} P(y|x)$$

Usando la regla de Bayes

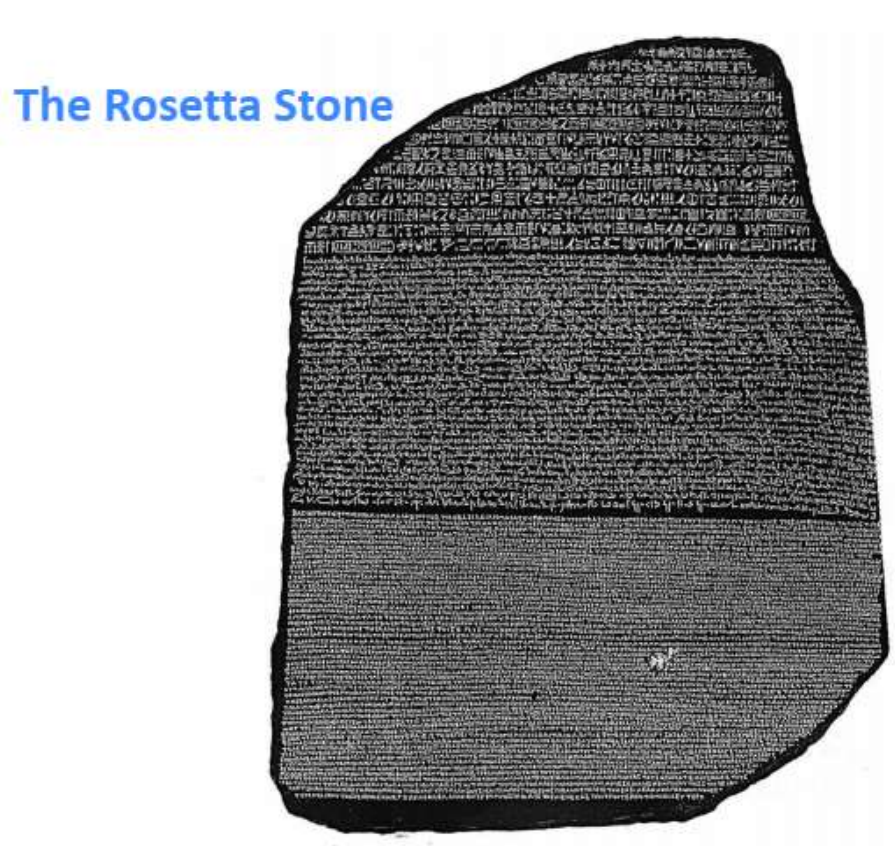
$$\underset{y}{\text{argmax}} P(x|y)P(y)$$

Modelo de traslación
Fidelidad de la traslación.
Aprendida desde pares de datos

Modelo de lenguaje
Fluidez de buen español.
Aprendida desde data en español

Machine Translation

El modelo de traslación se aprende desde un corpus que asocie cómo traducir sentencias de un lenguaje a otro

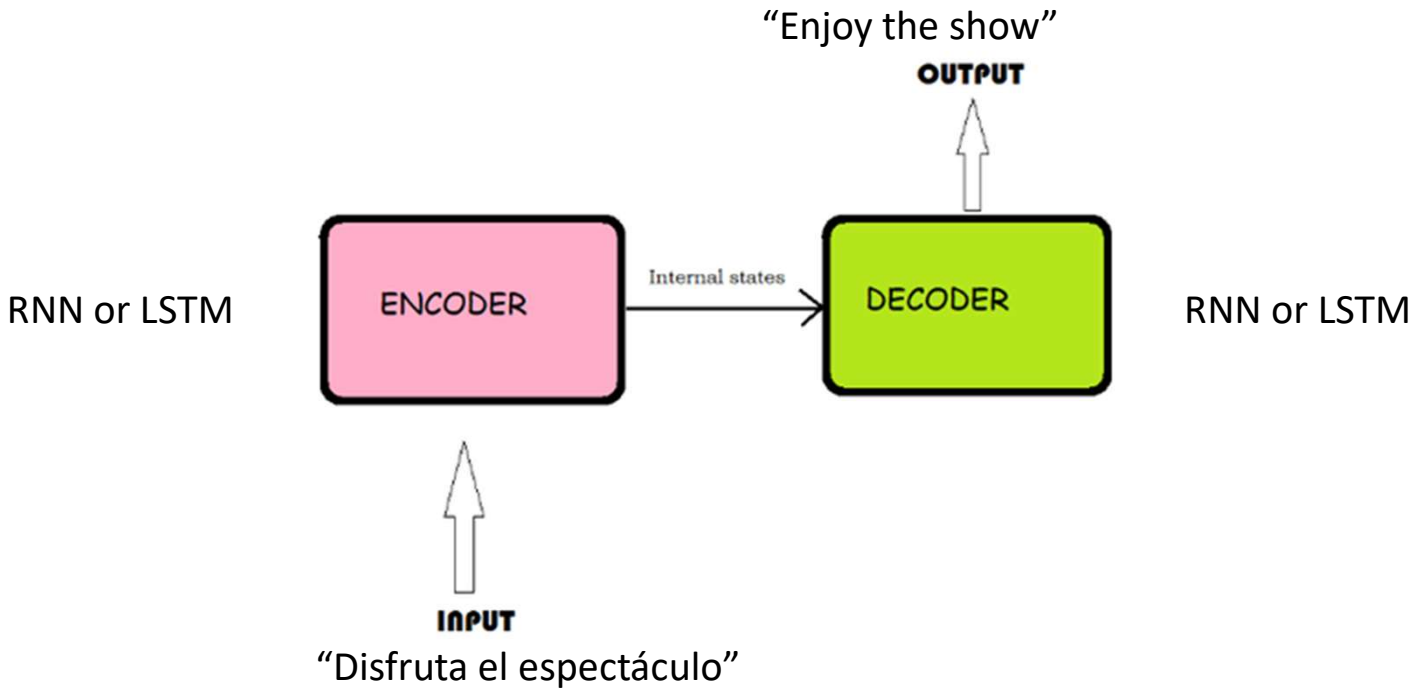


- Problema complejo
- Feature engineering
- Un modelo distinto para cada par de lenguajes

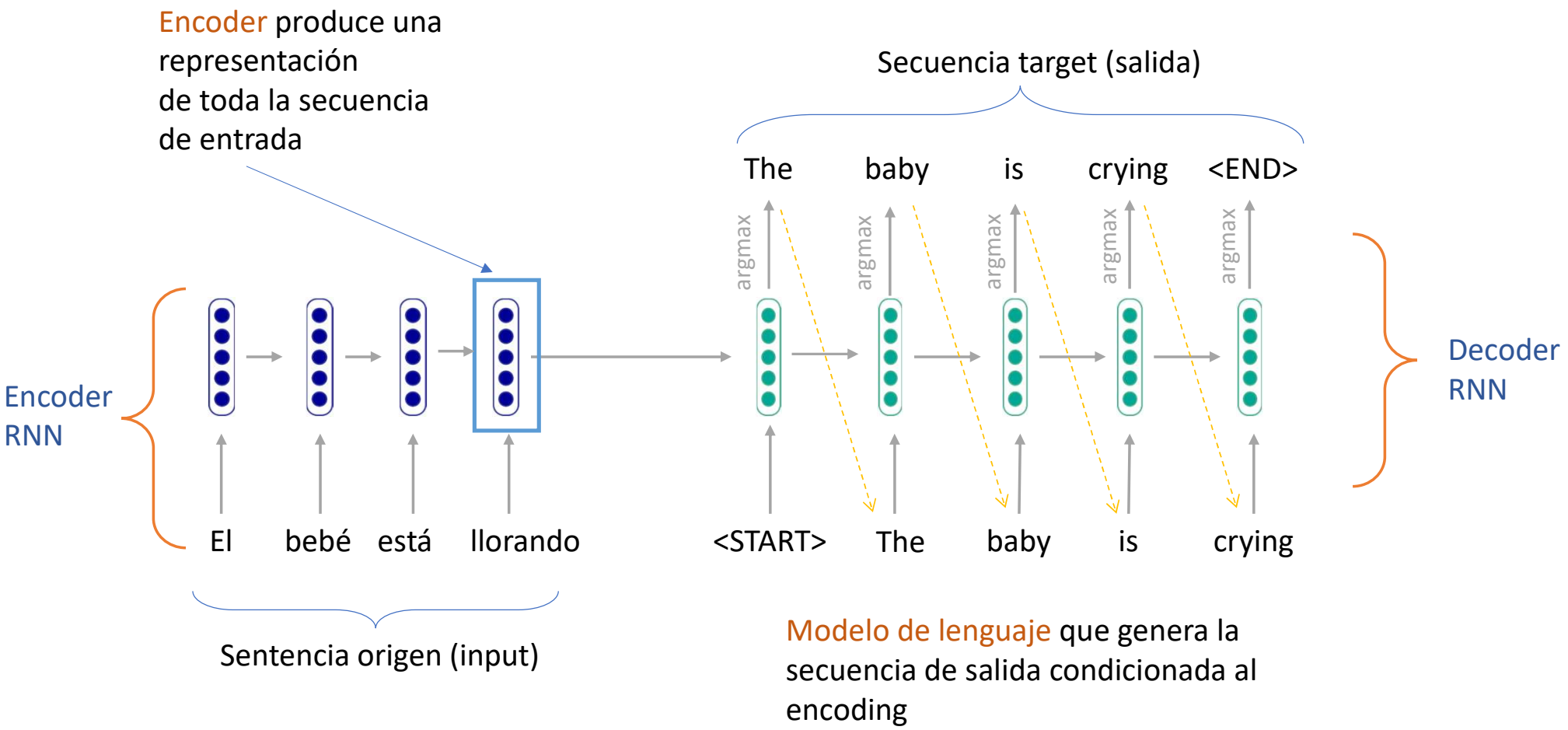
Neural Machine Translation

Una red neuronal que traduzca una sentencia en otra

Modelo comúnmente llamado Sequence-to-sequence o seq2seq que involucra dos RNN's



Neural Machine Translation



Neural Machine Translation

Modelo de lenguaje condicional

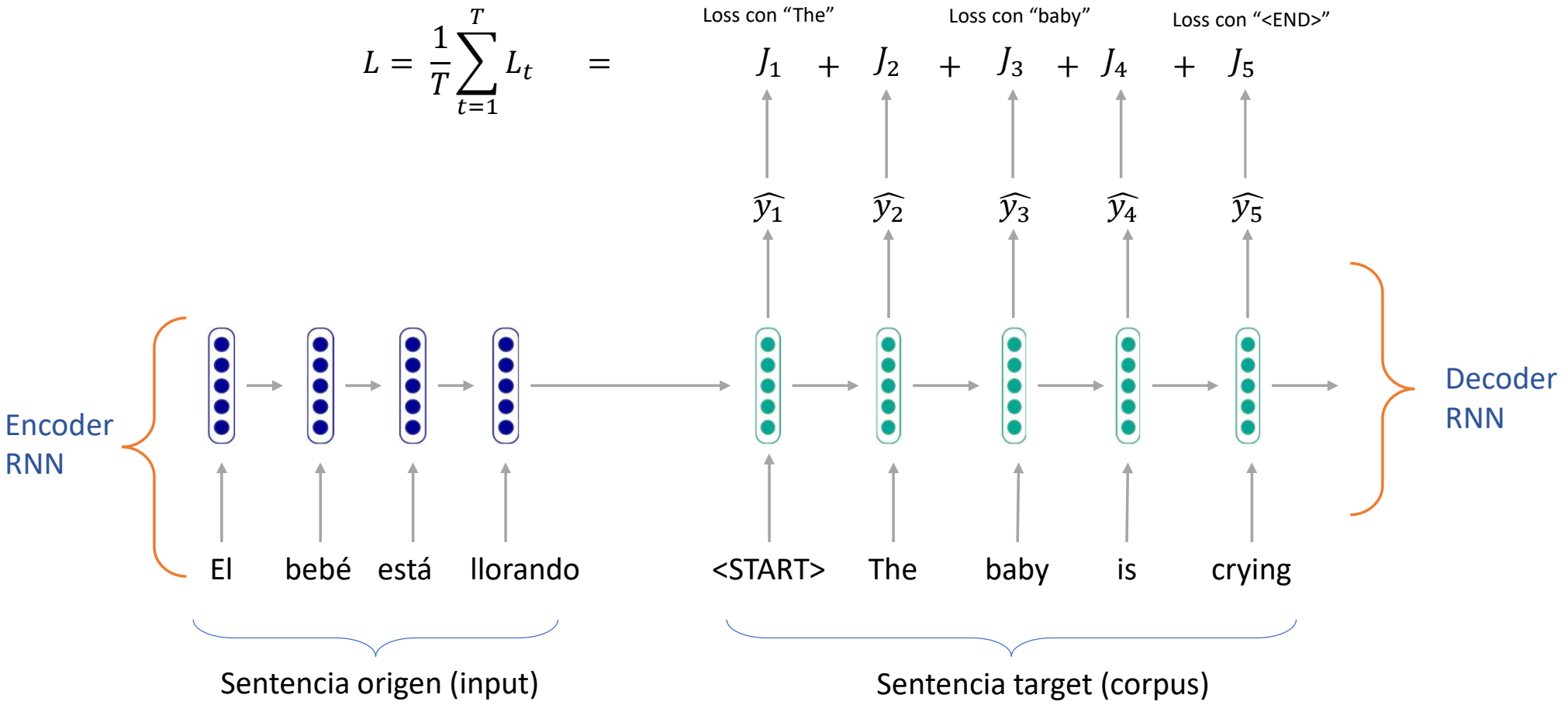
- Modelo de lenguaje porque el decoder predice una palabra a la vez
- Condicional porque las predicciones usan la información de la sentencia origen

Se calcula lo siguiente:

$$P(y|x)$$

$$P(y|x) = P(y_1|x)P(y_2|y_1, x)P(y_3|y_2, y_1, x) \dots P(y_T|y_1, \dots, y_{T-1}, x)$$

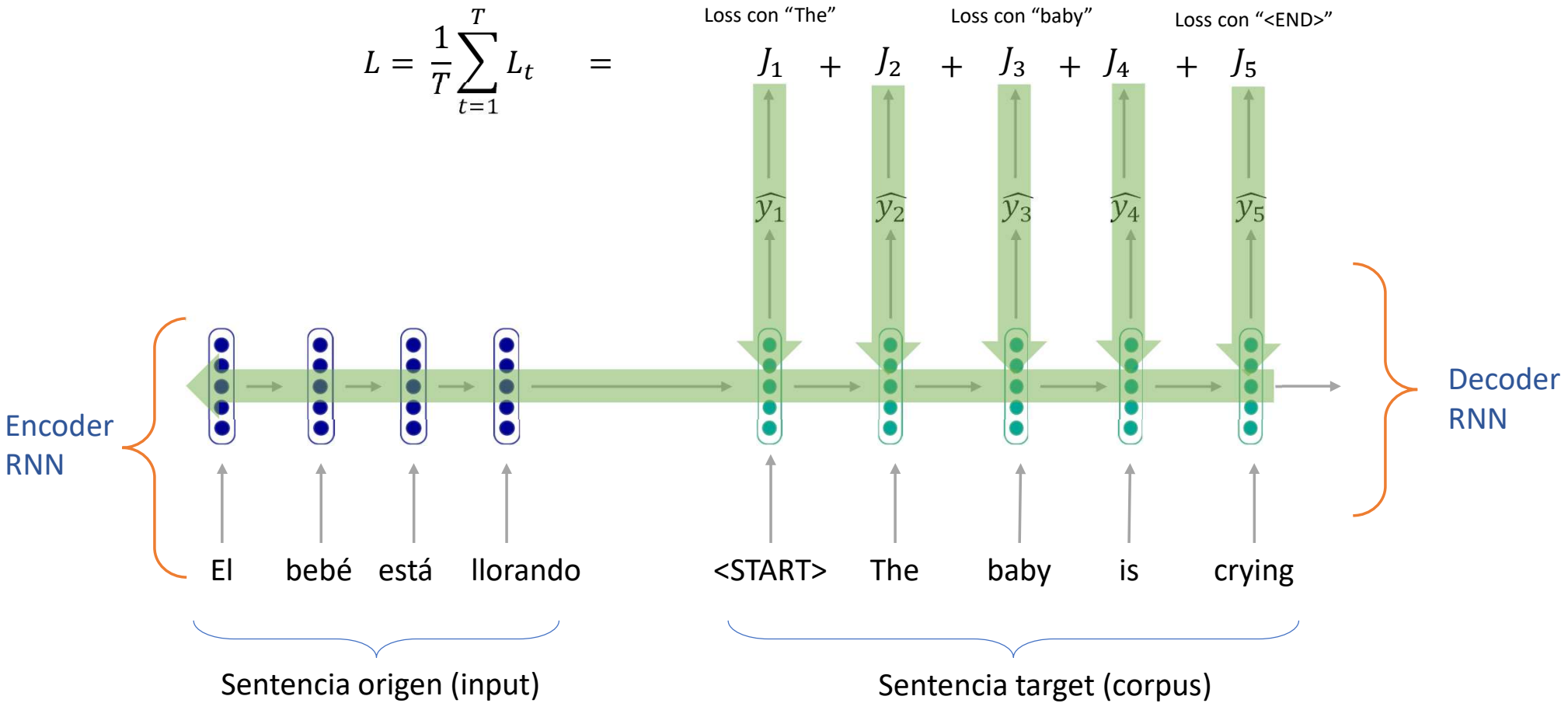
Neural Machine Translation - Training



Neural Machine Translation - Training

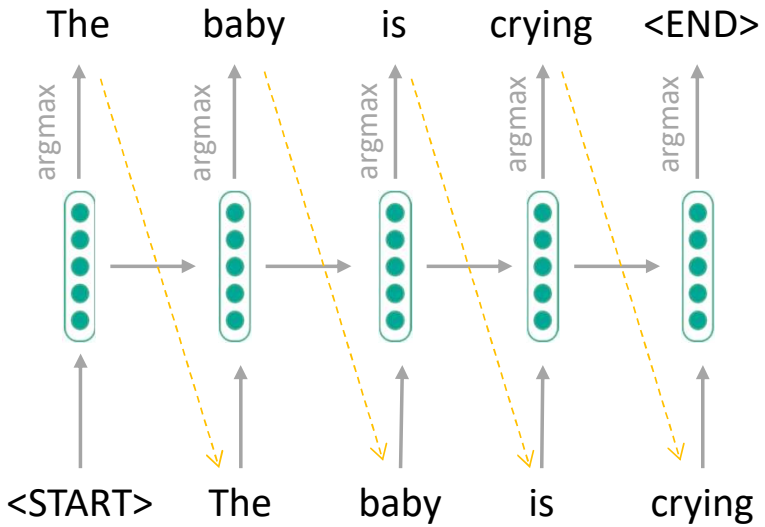
$$L = \frac{1}{T}\sum_{t=1}^{T} L_t = J_1 + J_2 + J_3 + J_4 + J_5$$

Loss con "The" Loss con "baby" Loss con "<END>"



Neural Machine Translation - Inferencia

En el ejemplo de antes, usamos $argmax$. Es decir, se toma la palabra más probable en cada tiempo como resultado de la inferencia



Conocido como greedy decoding (tomar la palabra más probable cada vez)

Una vez tomada una decisión, no hay forma de deshacerla

Neural Machine Translation - Inferencia

Queremos encontrar la secuencia y (de longitud T) que maximiza

$$P(y|x) = P(y_1|x)P(y_2|y_1, x)P(y_3|y_2, y_1, x) \dots P(y_T|y_1, \dots, y_{T-1}, x)$$

$$= \prod_{t=1}^{T} P(y_t|y_1, \dots, y_{t-1}, x)$$

Podríamos evaluar todas las potenciales secuencias, pero eso equivale a $O(V^T)$ secuencias. V es el tamaño del vocabulario

Una posible solución es Beam Search Decoding

Beam Search Decoding

En cada paso del decoder, mantener registro de los k traducciones más probables (k es el tamaño del beam)

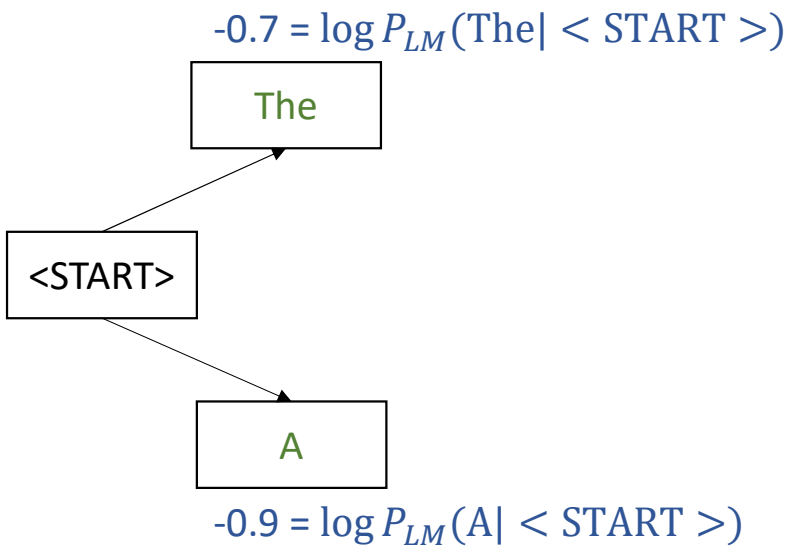
Una potencial secuencia $y_1, \dots, y_t$ tiene un score que calculamos como el logaritmo de su probabilidad

$$score(y_1, \dots, y_t) = \log P_{LM}(y_1, \dots, y_t|x) = \sum_{i=1}^{t} \log P_{LM}(y_i|y_1, \dots, y_{i-1}, x)$$

Scores son negativos, pero mientras más grandes mejor

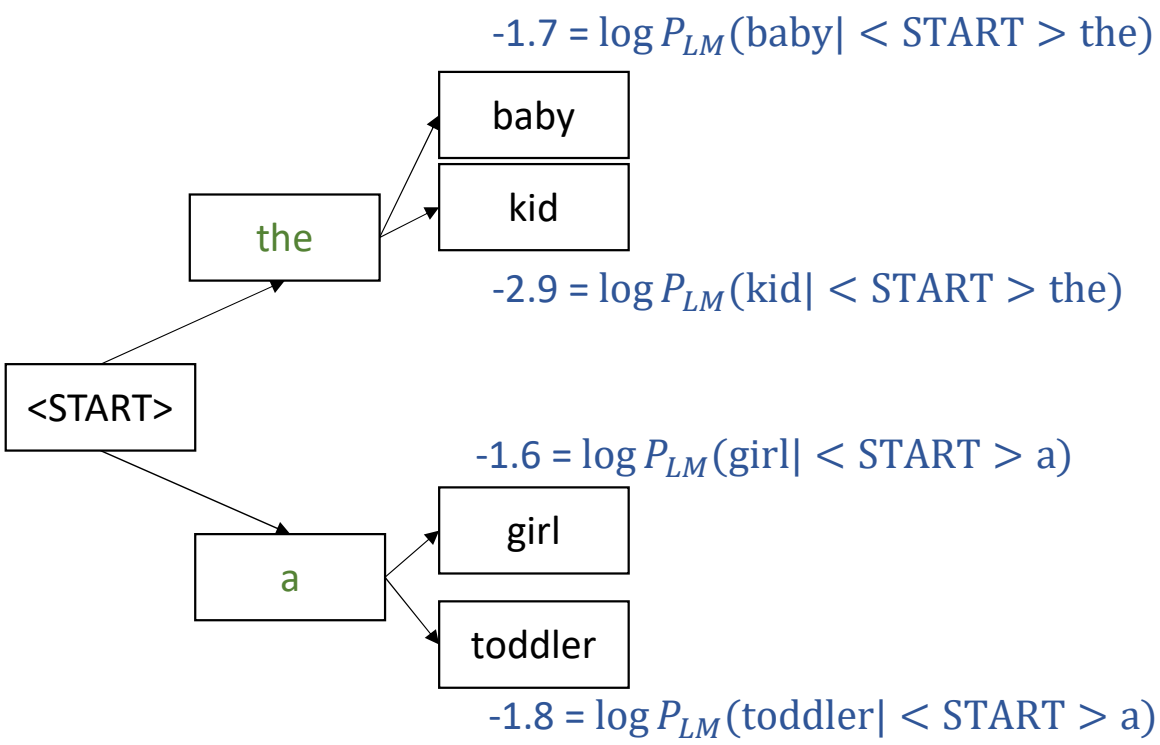
Beam search decoding

Tamaño de beam $k = 2$



Beam search decoding

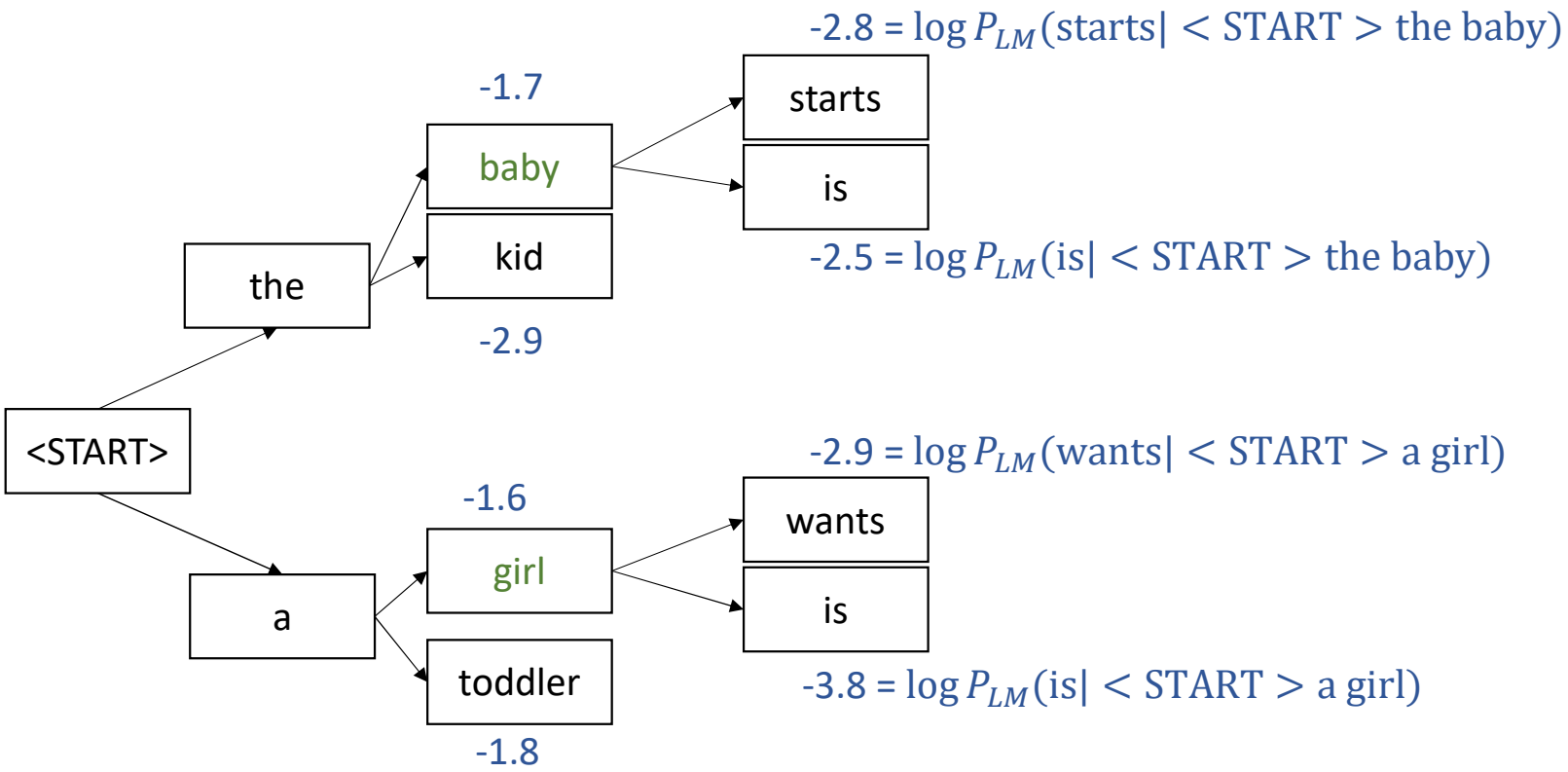
Tamaño de beam $k = 2$



Escogemos las k palabras con mayor score

Beam search decoding

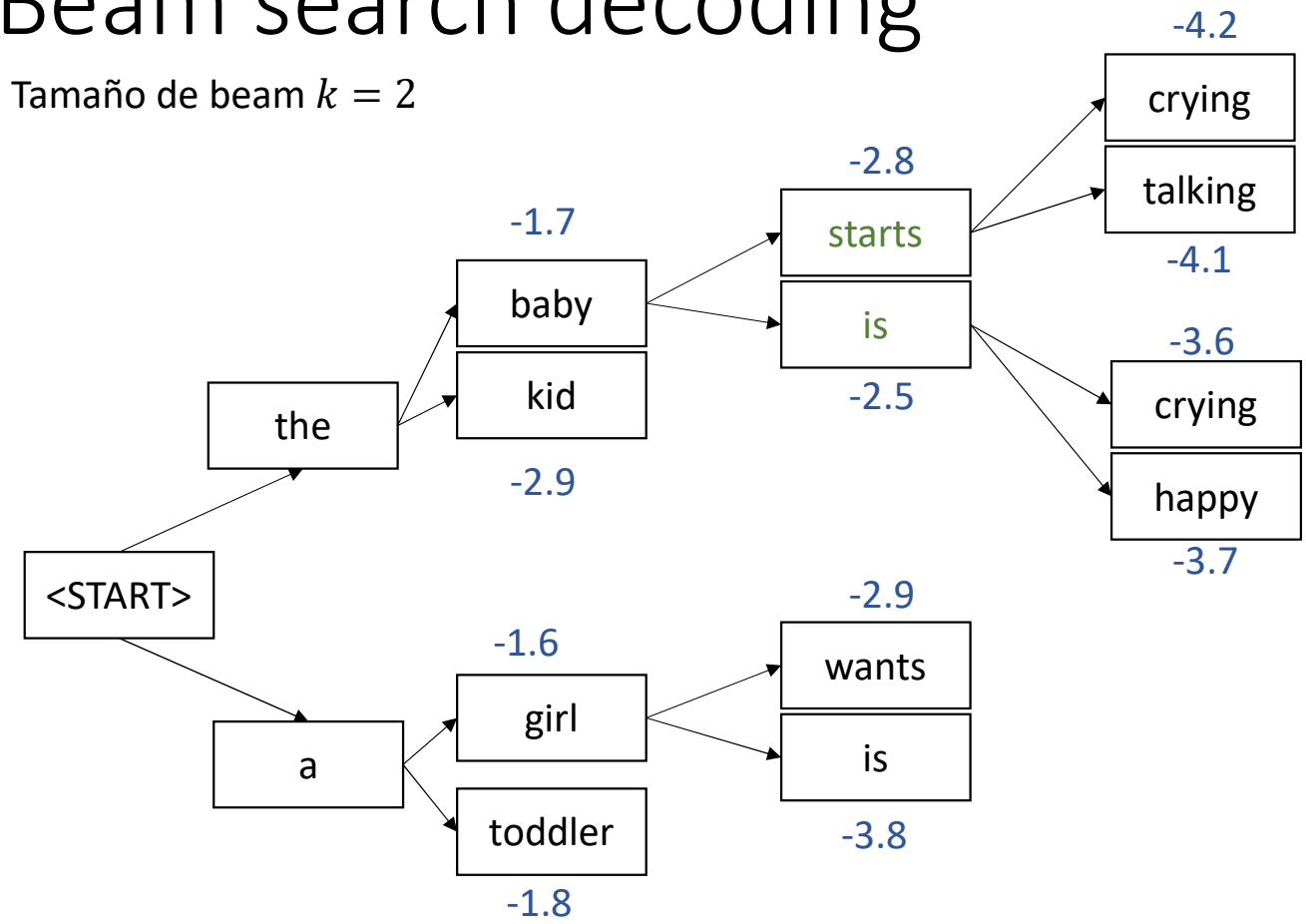
Tamaño de beam $k = 2$



Escogemos las k palabras con mayor score

Beam search decoding

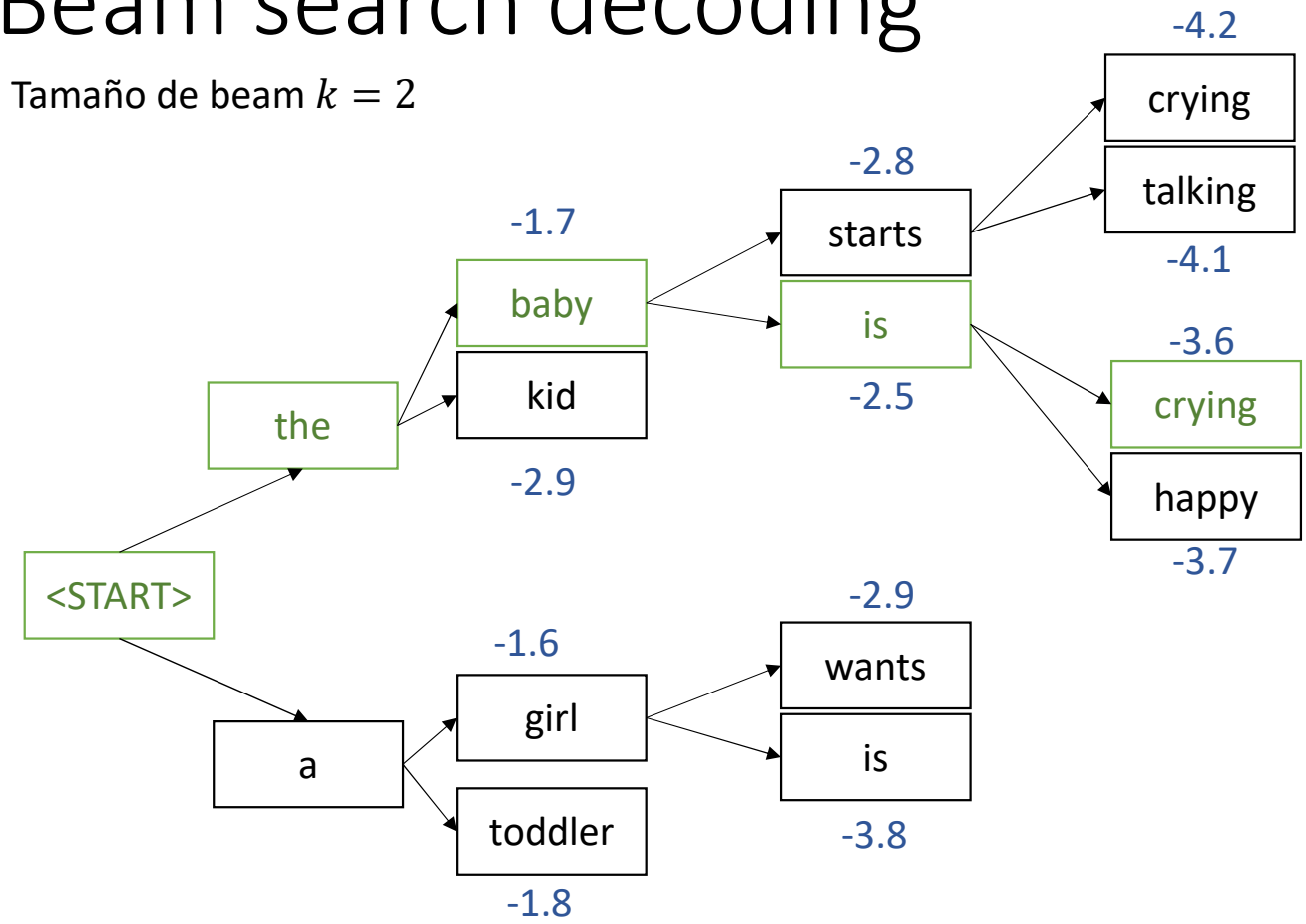
Tamaño de beam $k = 2$



Escogemos las k palabras con mayor score

Beam search decoding

Tamaño de beam $k = 2$



Cuando se llega al final, se hace backtracking para seleccionar la secuencia

Evaluación de Neural Machine Translation

Es necesario contar con una métrica para medir la similitud entre una traducción automática y corpus obtenidos por humanos.

La métrica más común es BLEU (Bilingual Evaluation Understudy). Se define como:

$$BLEU = PB \cdot \exp\left(\sum_{n=1}^{N} w_n \log P_n\right)$$

Donde P_n es la proporción de n-gramas que coinciden. Esto es P_1 es la proporción de 1 −grama (palabras) que coinciden y P_2 es la proporción de 2 −gramas (pares de palabras) que coinciden, y así sucesivamente.

Cada tipo de n −grama tiene un peso asociado w_n.

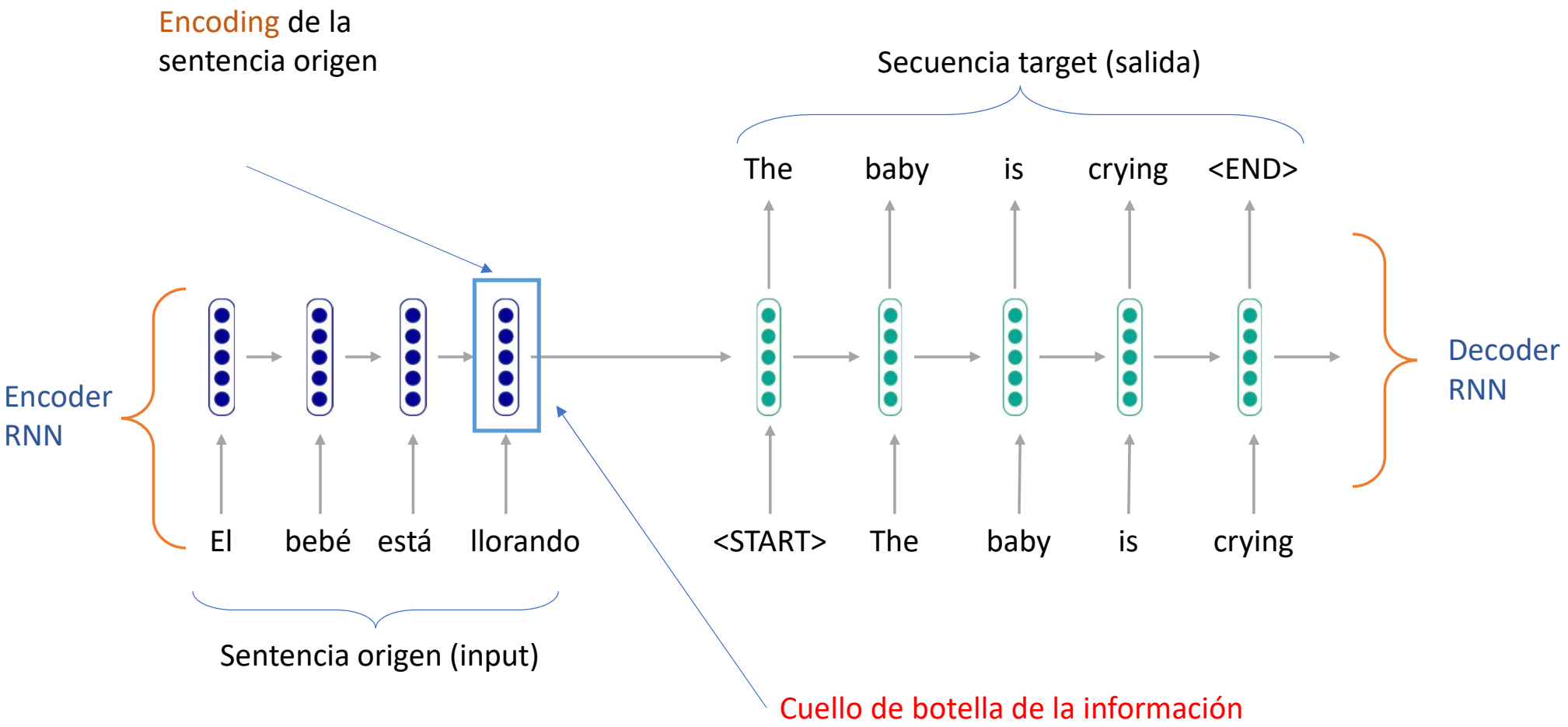
$$PB = \begin{cases} 1 & \text{Si } c > r \\ e^{1-\frac{r}{c}} & \text{Caso contrario} \end{cases}$$

c: longitud de frase candidate
r: longitud de frase referencia

Deep Learning

Atención neuronal

Neural Machine Translation

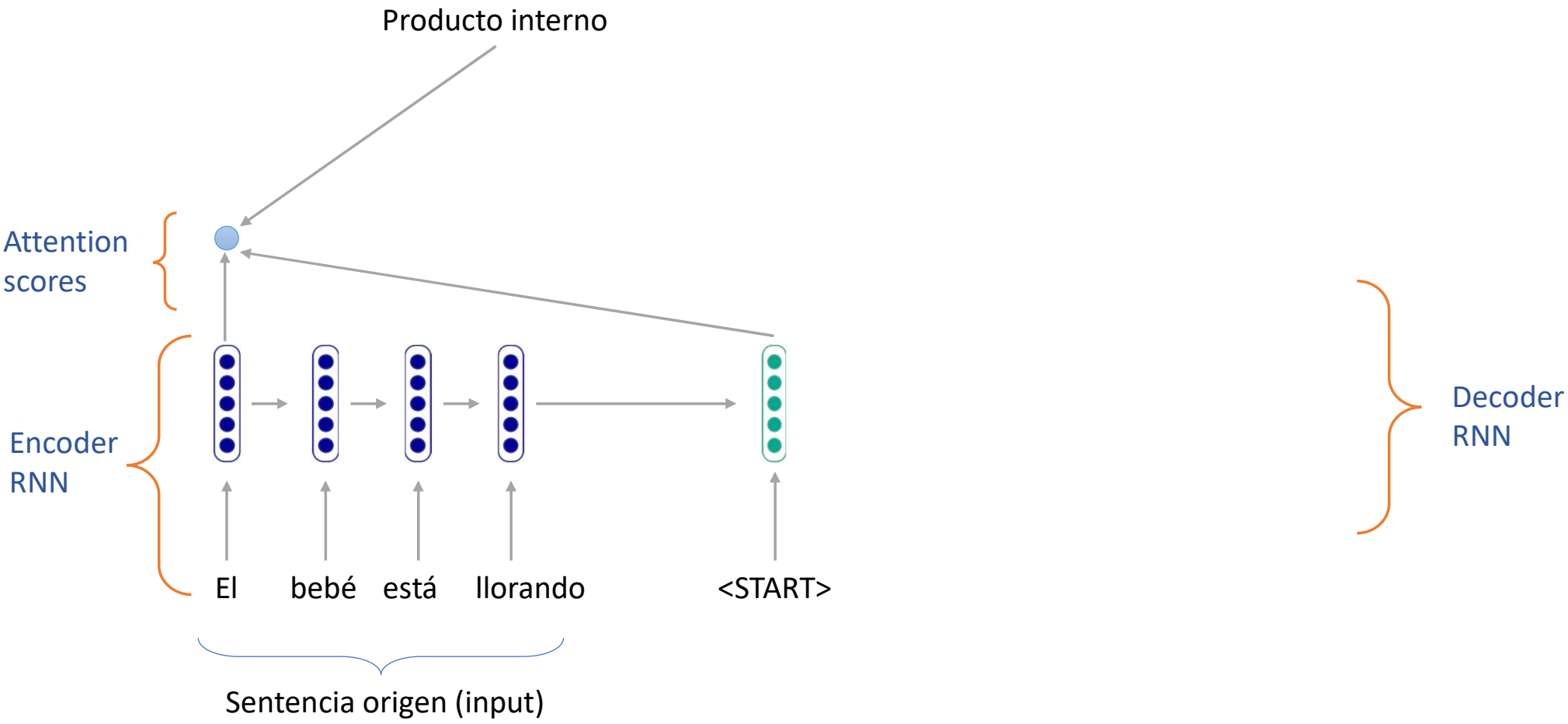


Atención neuronal

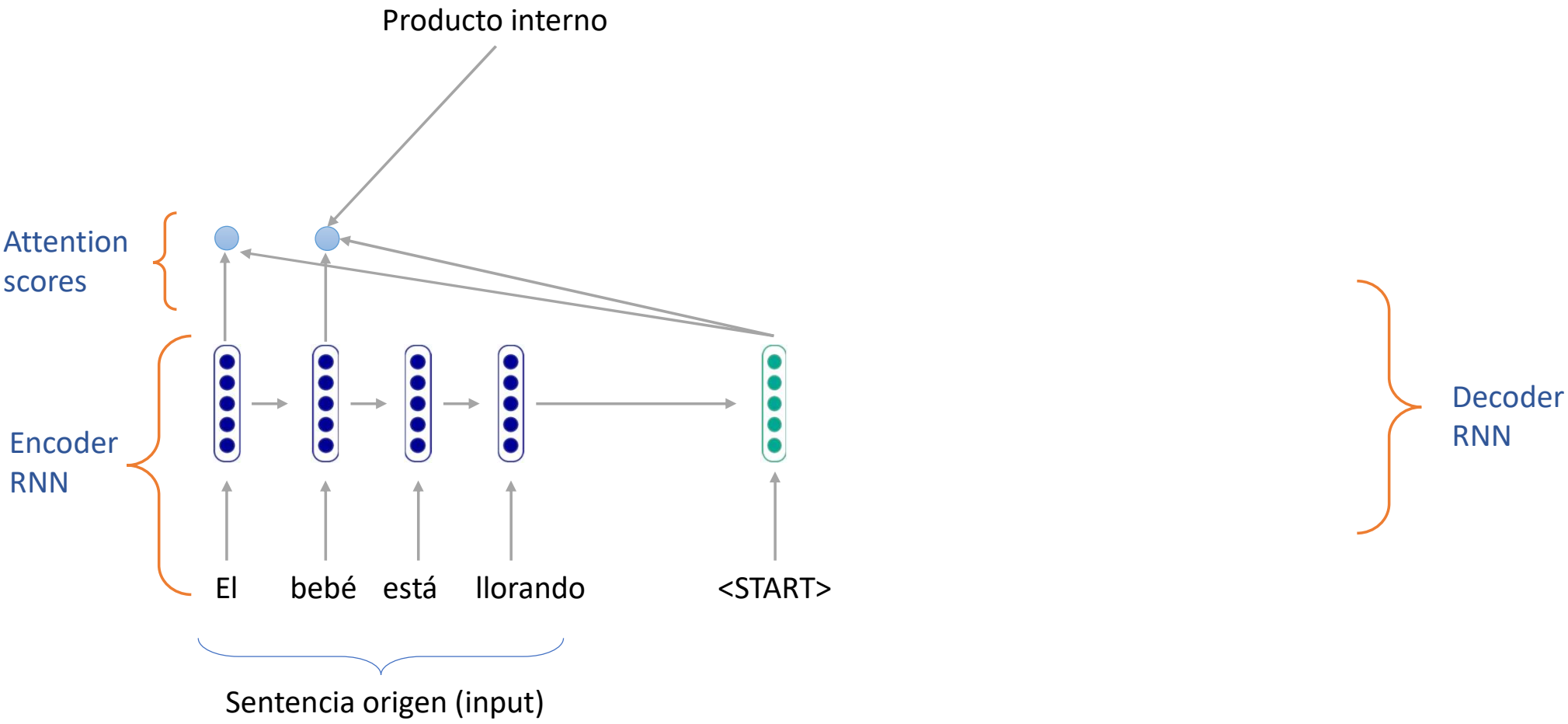
Es un mecanismo que previene el cuello de botella de la información

Idea: porqué no usamos todas las salidas intermedias del encoding para tener mejor información de las relaciones entre las entradas y las salidas?

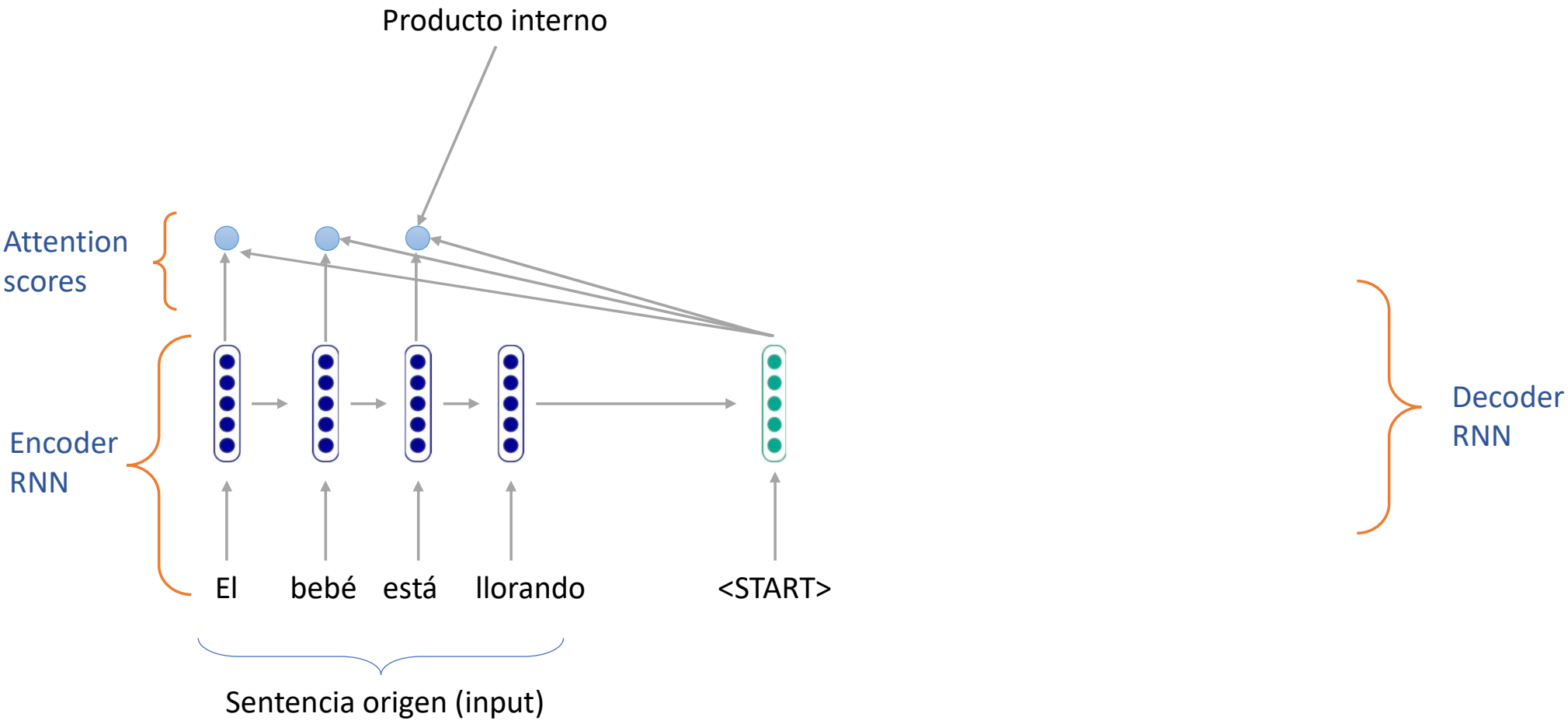
Atención neuronal



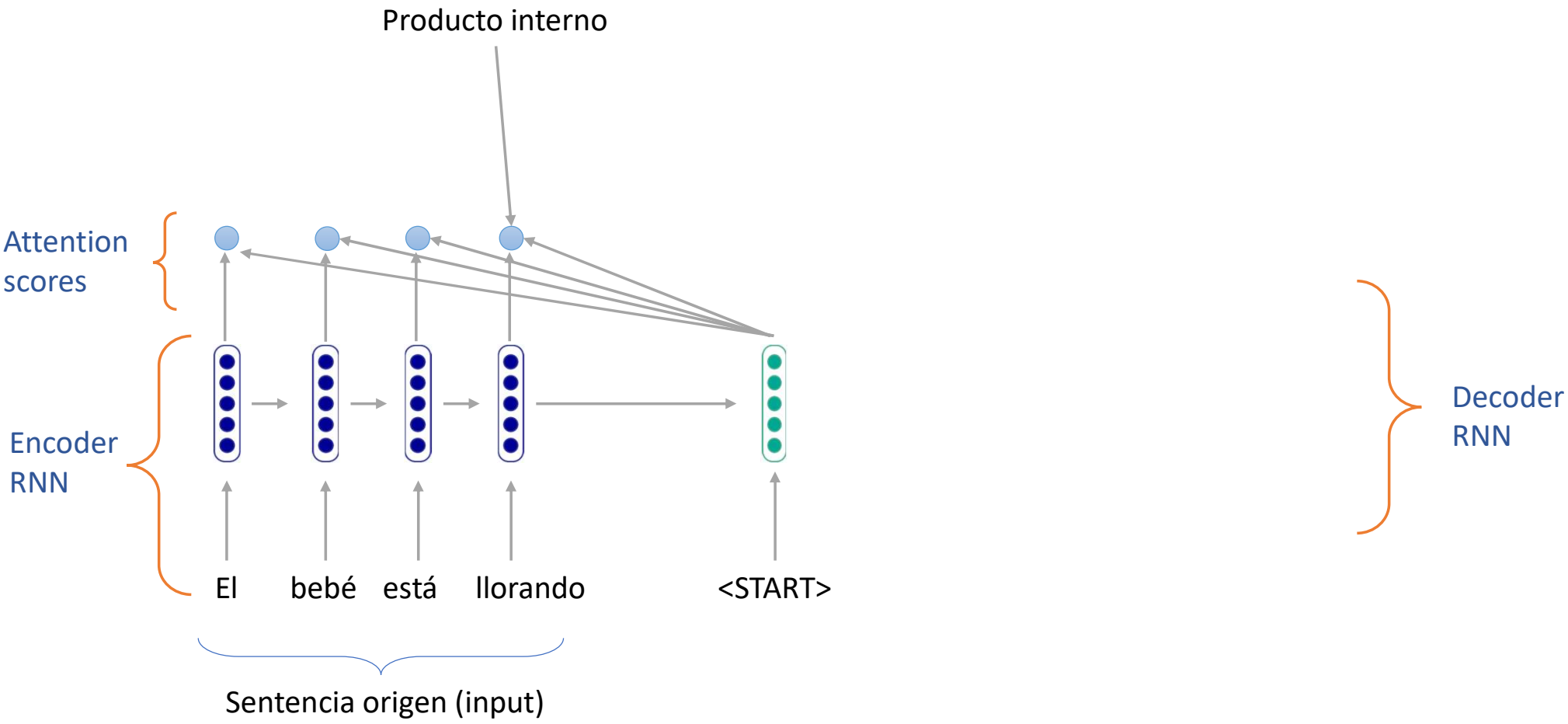
Atención neuronal



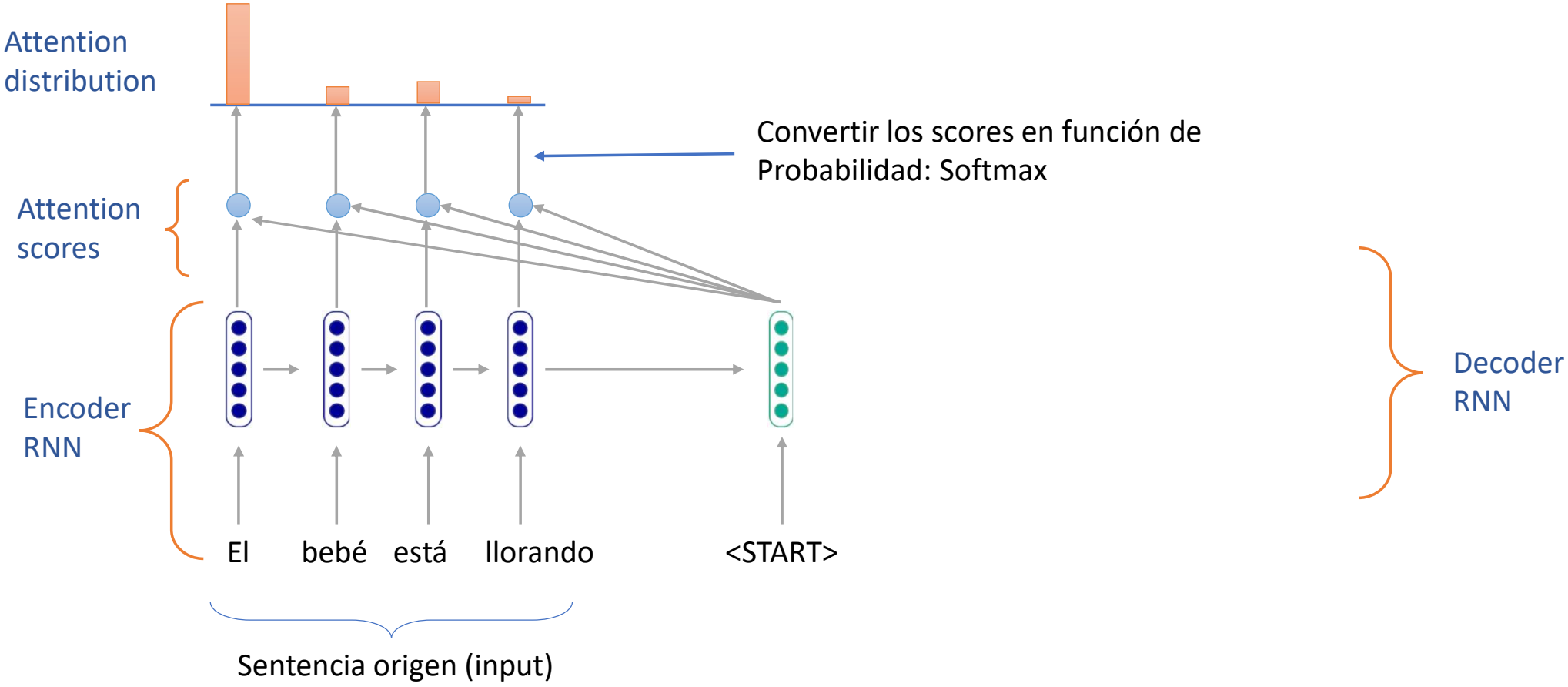
Atención neuronal



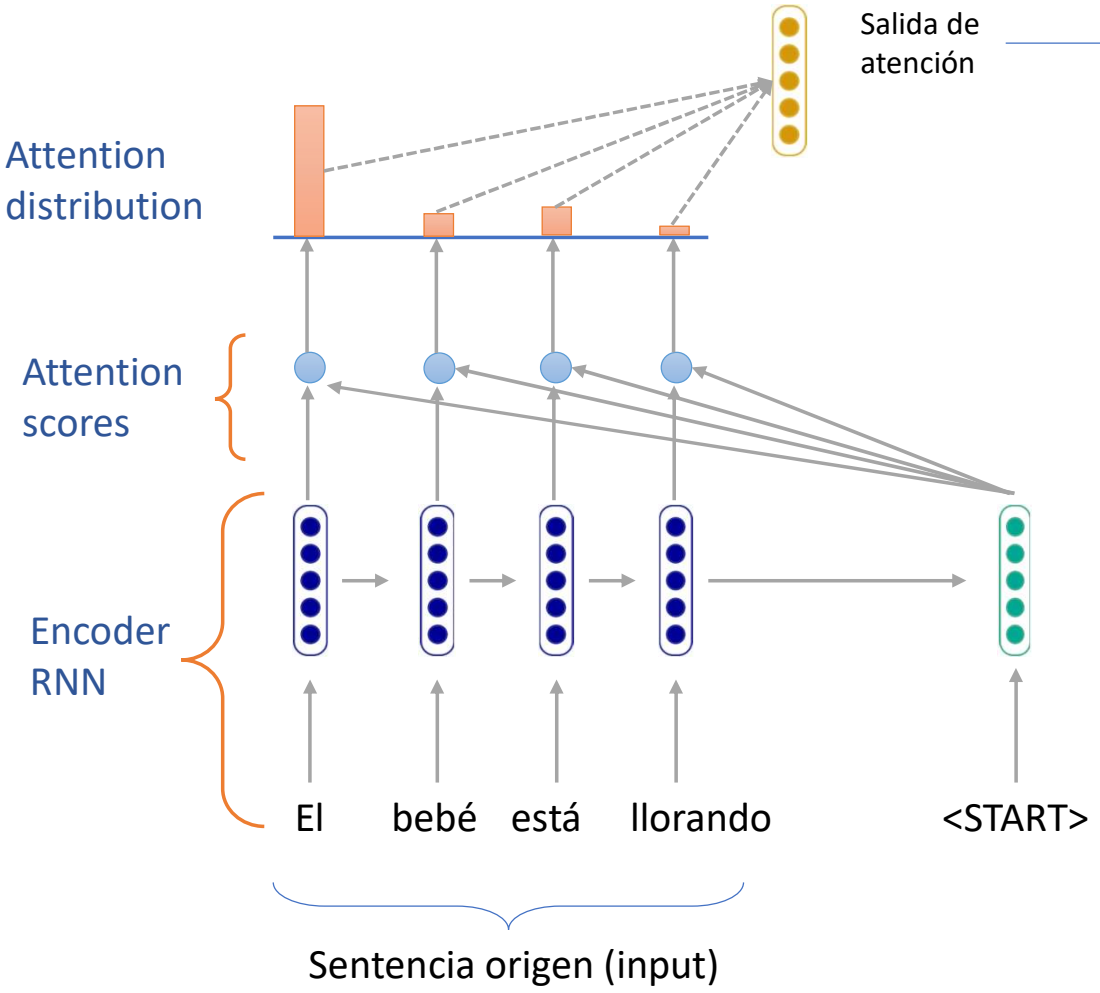
Atención neuronal



Atención neuronal



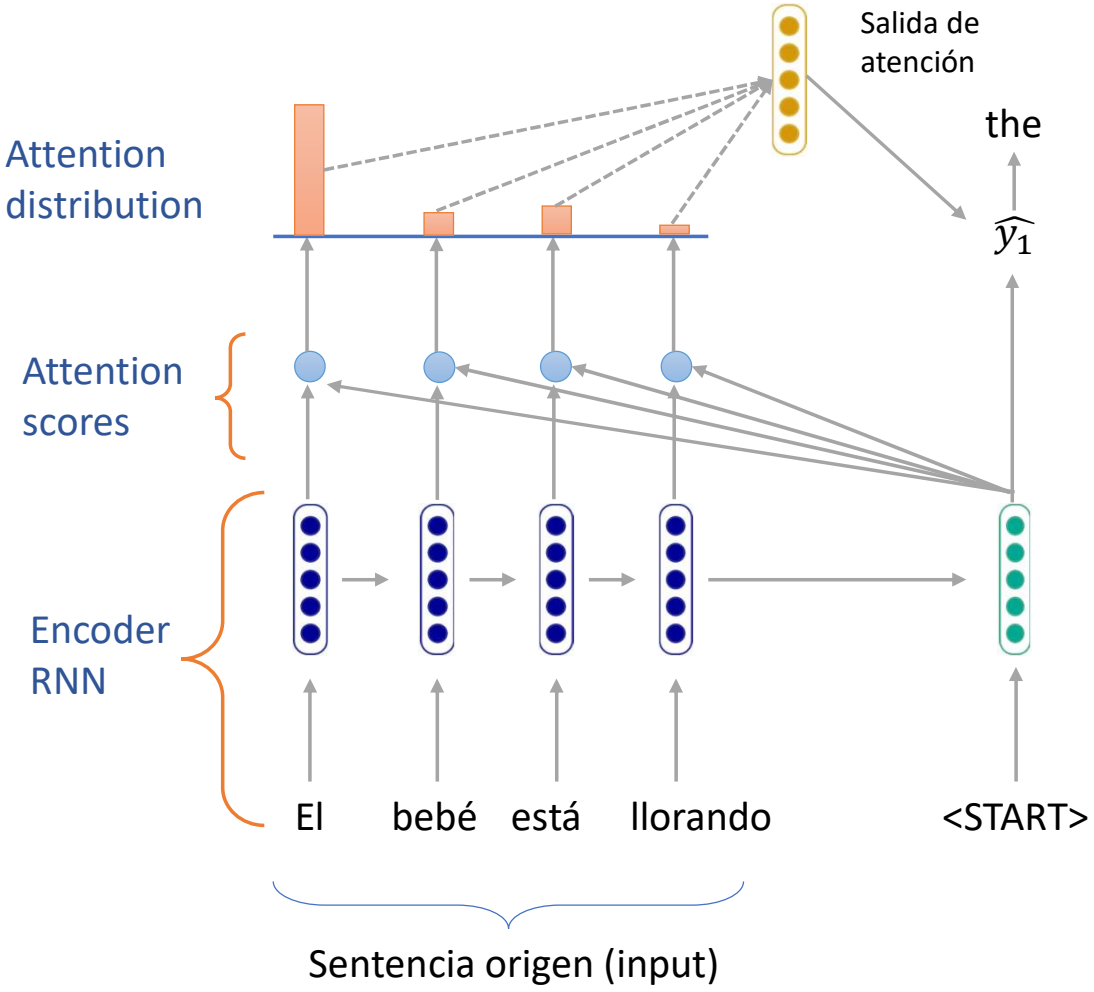
Atención neuronal



Suma ponderada de estados ocultos del encoder
Con los pesos de atención

Decoder RNN

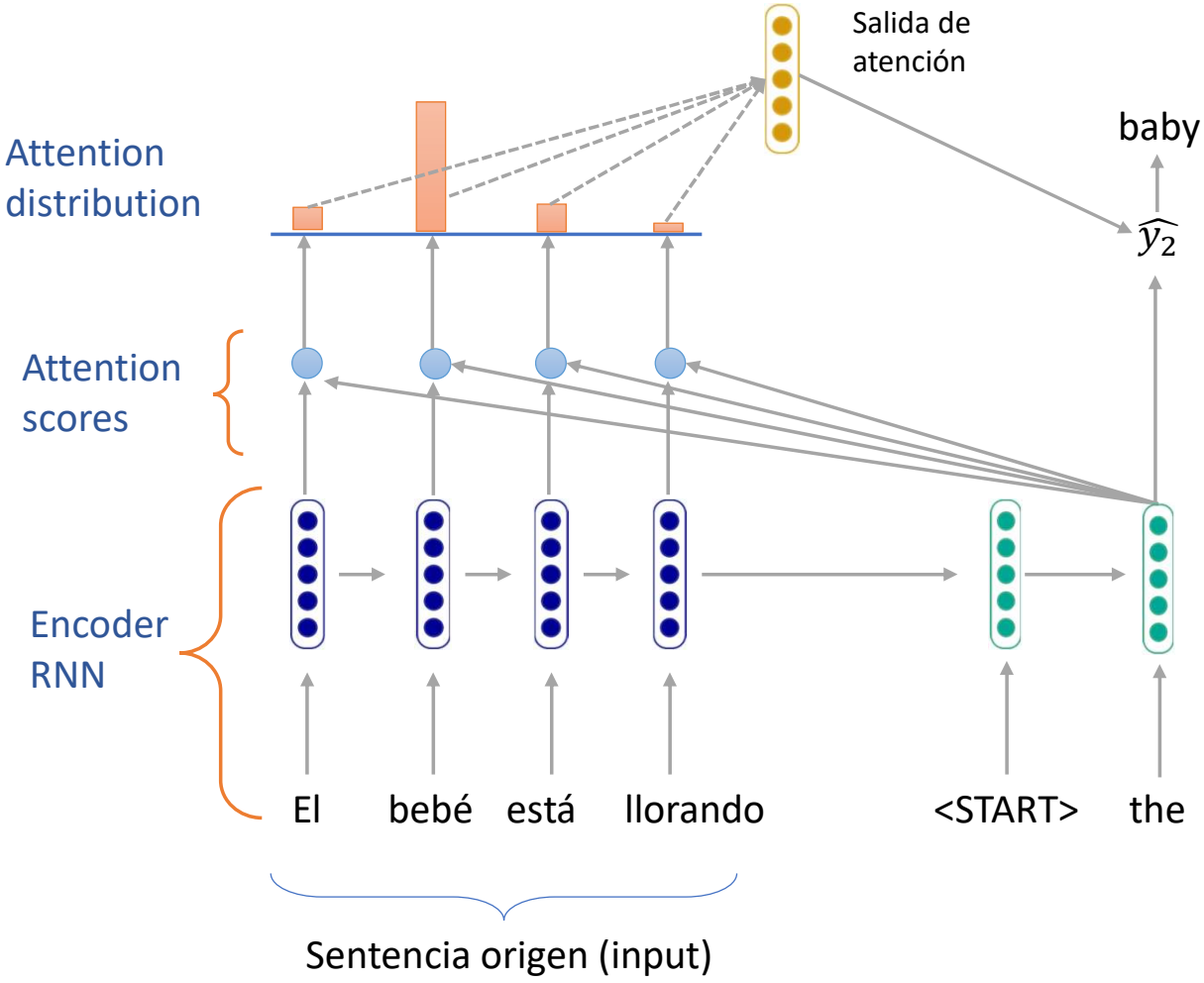
Atención neuronal



Salida de atención se usa en conjunto con
El estado oculto del decoder para calcular la palabra

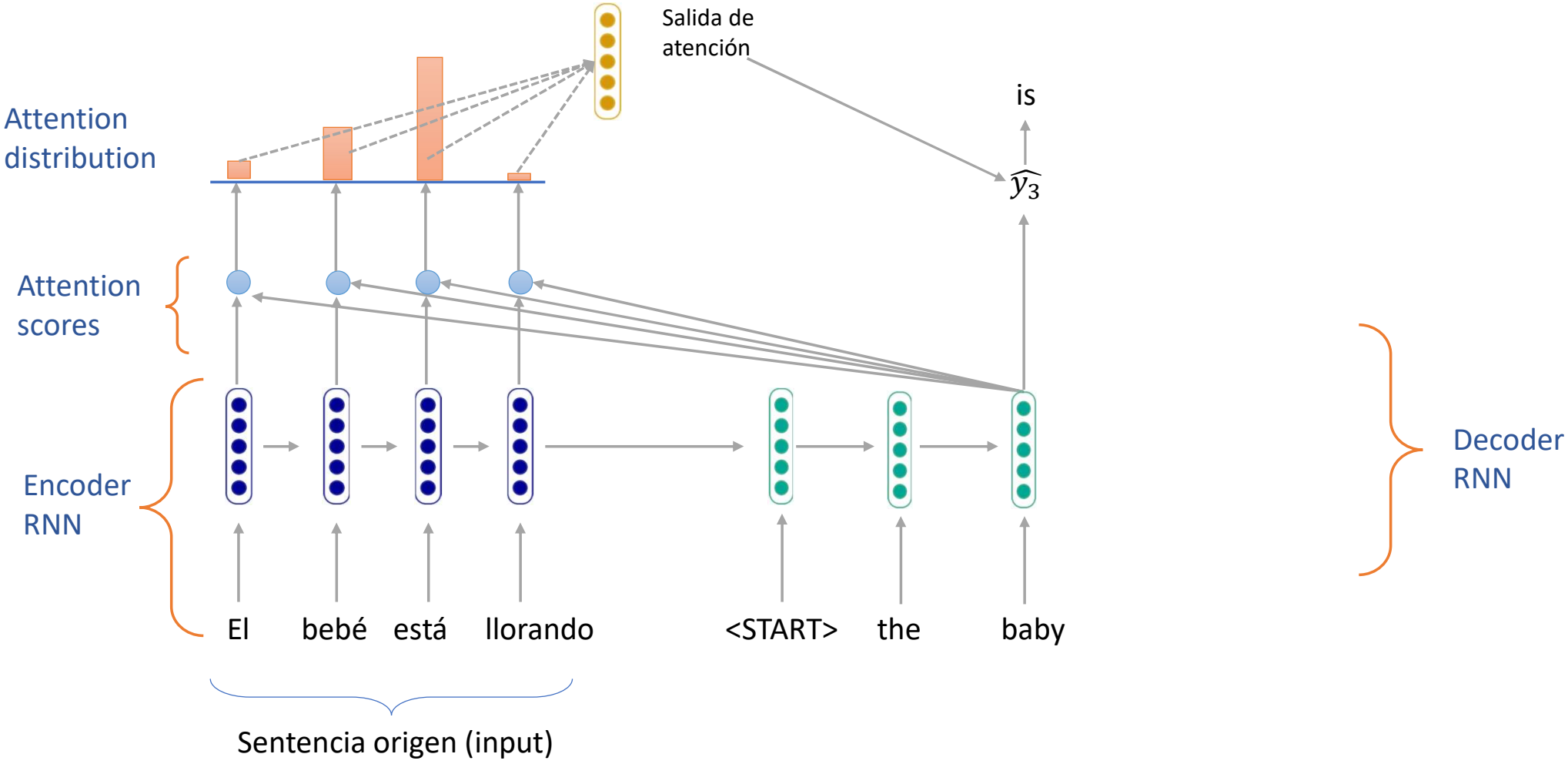
Decoder RNN

Atención neuronal

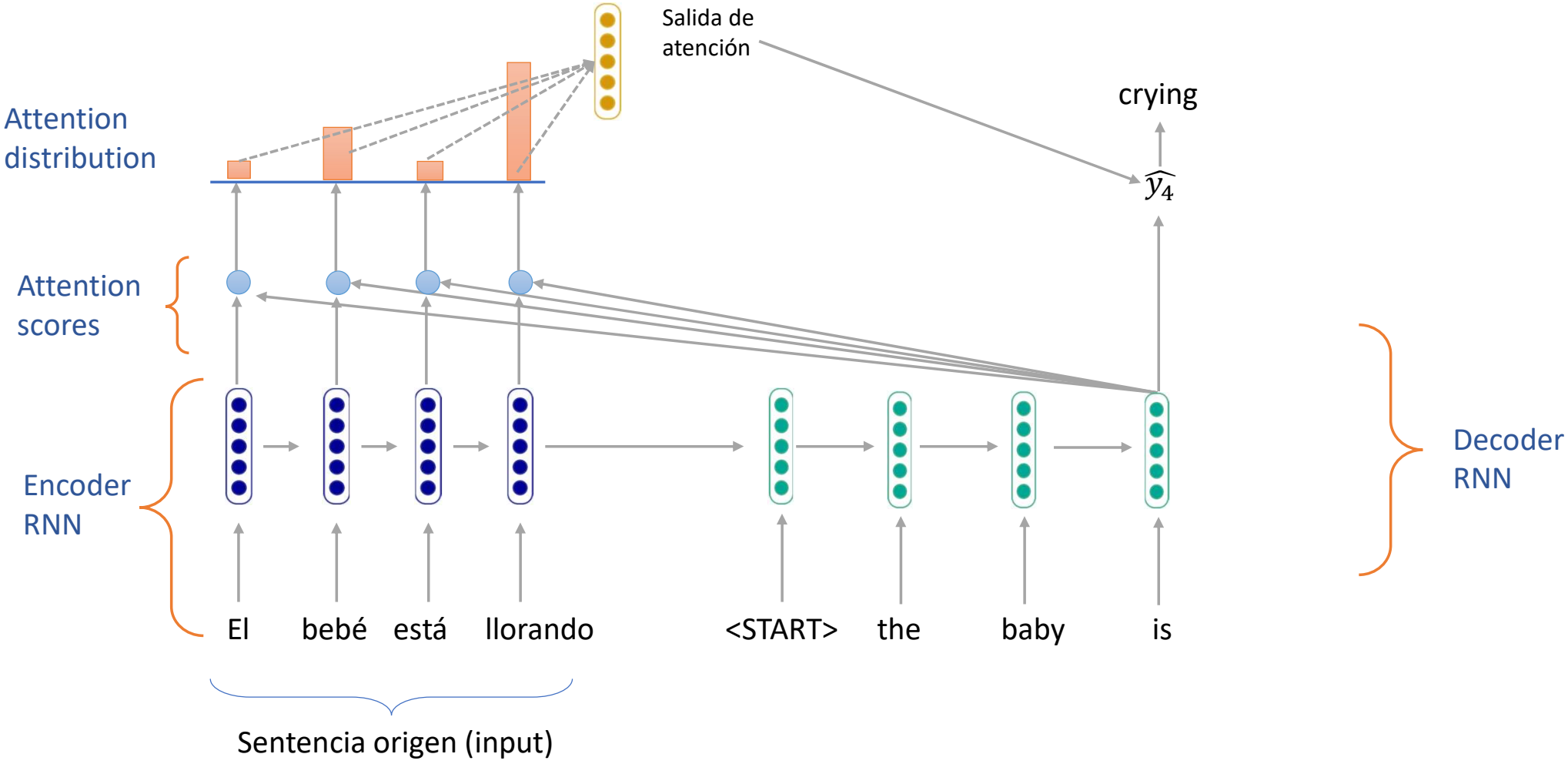


Decoder RNN

Atención neuronal



Atención neuronal



Atención neuronal

- Necesitamos como entrada los estados ocultos del encoder: $h_1, h_2, \dots, h_N \in \mathbb{R}^h$
- En el tiempo t del decoder, se tiene el estado oculto $s_t \in \mathbb{R}^h$
- Obtenemos los scores de atención en este paso:

$$e_t = [s_t^T h_1, s_t^T h_2, \dots, s_t^T h_N] \in \mathbb{R}^N$$

- Calculamos Softmax para convertir los scores a una función de probabilidad

$$\alpha_t = softmax(e_t) \in \mathbb{R}^N$$

- Usamos la distribución para ponderar los estados ocultos del encoder

$$a_t = \sum_{i=1}^{N} \alpha_t^i h_i \in \mathbb{R}^h$$

- Finalmente se concatenan la salida de atención con el estado oculto del encoder

$$[a_t; s_t] \in \mathbb{R}^{2h}$$

Atención neuronal

- Variantes de atención: principalmente en el cómputo de los scores
- Generalizamos las entradas de la atención: estados ocultos del encoder : $h_1, h_2, \dots, h_N \in \mathbb{R}^{d_1}$ y estado oculto del decoder $s_t \in \mathbb{R}^{d_2}$

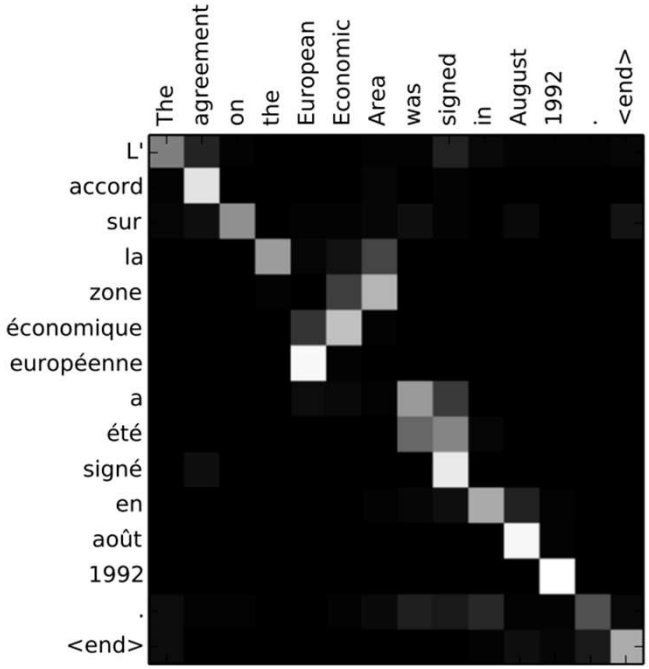
- Atención con producto interno: $e_t^i = s_t^T h_i$

- Atención multiplicativa: $e_t^i = s_t^T W h_i$

- Atención aditiva: $e_t^i = v^T \tanh(W_1 h_i + W_2 s_t)$

Atención neuronal

Permite añadir cierta explicabilidad al proceso de NMT. Se puede saber la alineación de palabras al traducir

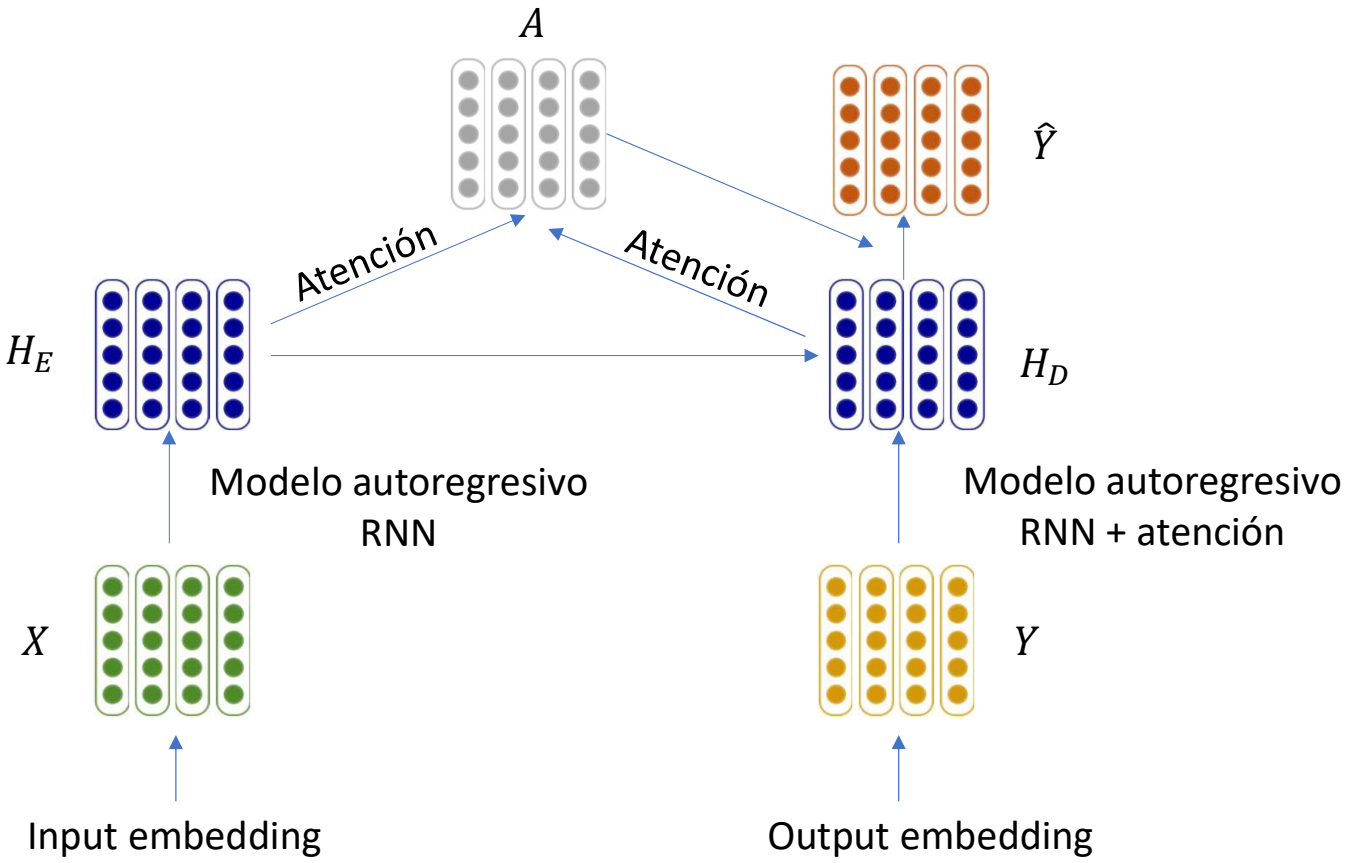


Deep learning

Transformers

Neural Machine Translation + Attention

Input: El bebé está llorando
Output: The baby is crying

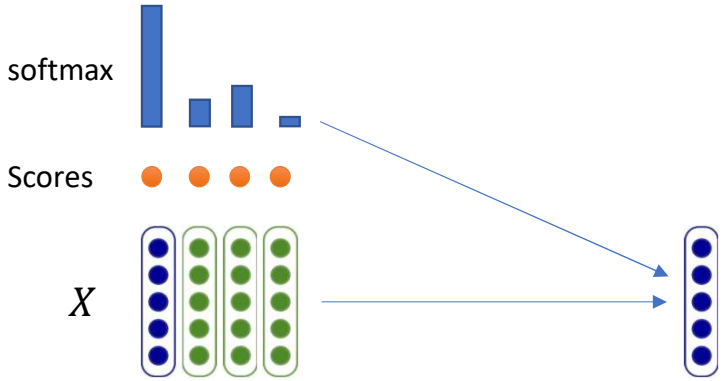


Dados dos conjuntos de vectores (que representan algo, por ejemplo una secuencia), computar nuevos vectores combinados que traten de encontrar asociaciones entre los datos originales.

Self-Attention

Se podrá aplicar atención sobre el mismo conjunto de vectores?

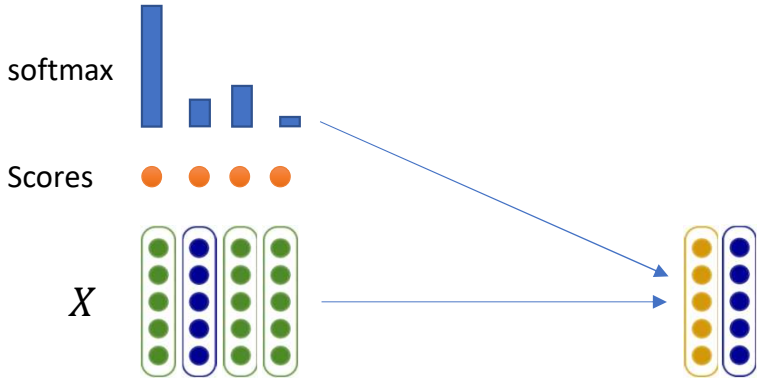
Cada vector de la colección aplica atención sobre los vectores de la colección de entrada y luego combina las atenciones para construir nuevos vectores.



Self-Attention

Se podrá aplicar atención sobre el mismo conjunto de vectores?

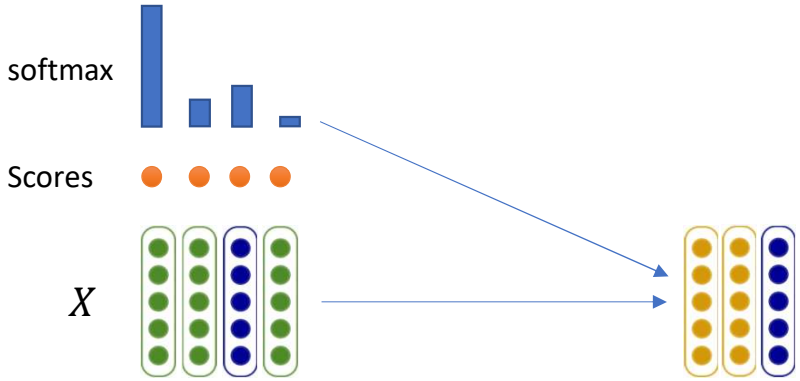
Cada vector de la colección aplica atención sobre los vectores de la colección de entrada y luego combina las atenciones para construir nuevos vectores.



Self-Attention

Se podrá aplicar atención sobre el mismo conjunto de vectores?

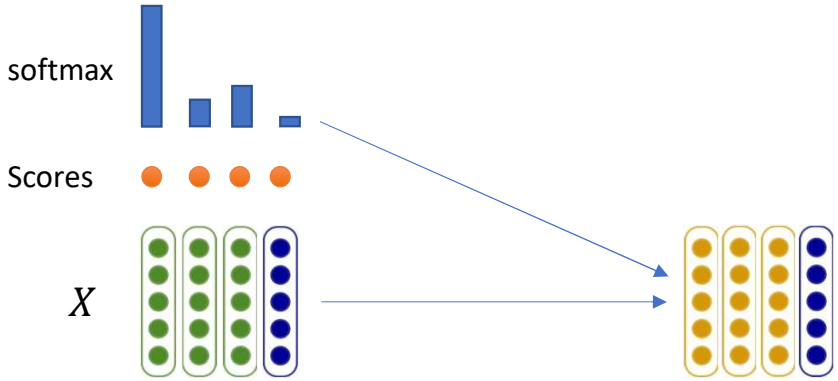
Cada vector de la colección aplica atención sobre los vectores de la colección de entrada y luego combina las atenciones para construir nuevos vectores.



Self-Attention

Se podrá aplicar atención sobre el mismo conjunto de vectores?

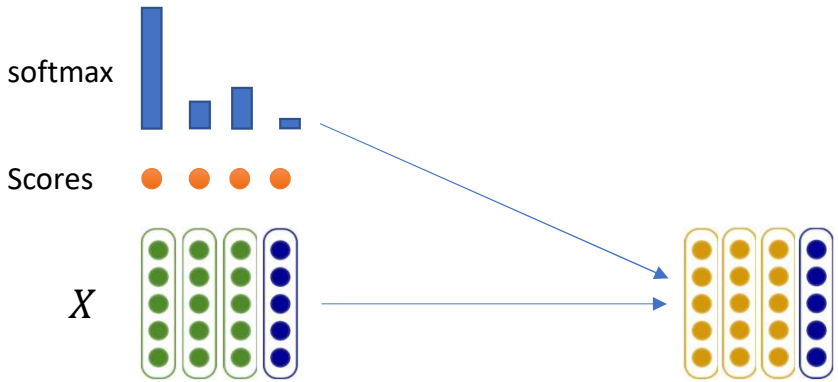
Cada vector de la colección aplica atención sobre los vectores de la colección de entrada y luego combina las atenciones para construir nuevos vectores.



Self-Attention

Se podrá aplicar atención sobre el mismo conjunto de vectores?

Cada vector de la colección aplica atención sobre los vectores de la colección de entrada y luego combina las atenciones para construir nuevos vectores.



Podemos generar nuevos datos a través de la atención
Ya no necesitamos la recurrencia!
Asumimos que los datos vienen en matrices de dimensión $seq \times dim$
Todo se puede calcular en paralelo, así:

$$H = softmax(XX^T)X$$

Pero parece no tener mucho sentido aplicar la auto-atención. Es muy posible llegar a la transformación de identidad entre X y H

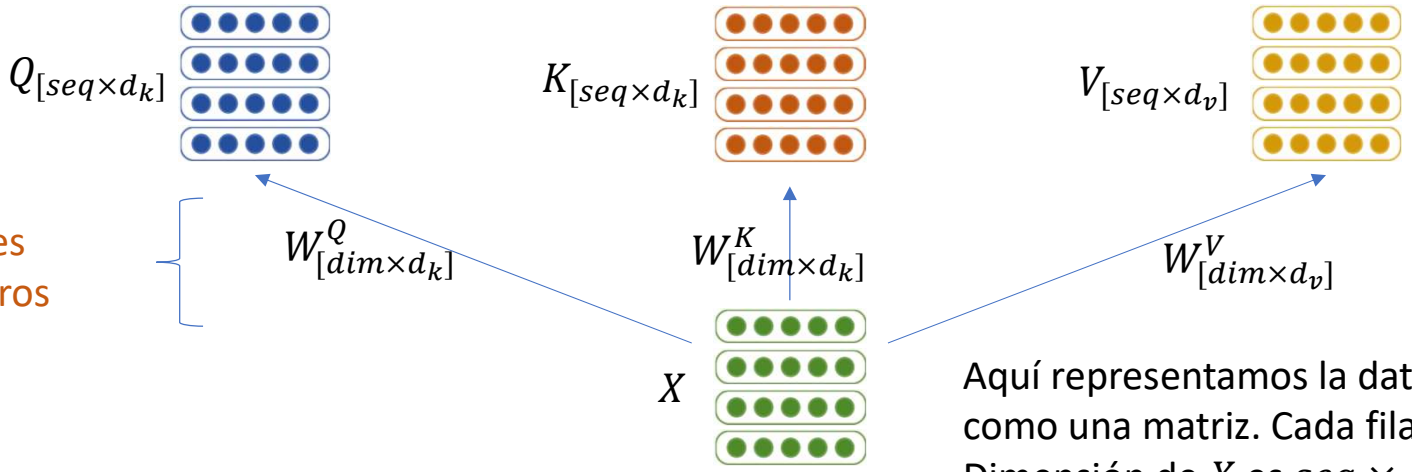
Self-Attention

Solución: Agregar un nivel de abstracción a los vectores que se usan en la auto-atención

$$H = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

QK^T es una matriz cuadrada de scores
"similitud entre cada par de embeddings de la secuencia"

El resultado es $H_{[dim \times d_v]}$



Estas matrices son parámetros

Aquí representamos la data de entrada como una matriz. Cada fila es un word embedding. Dimensión de X es $seq \times dim$

Multi-head Self-attention

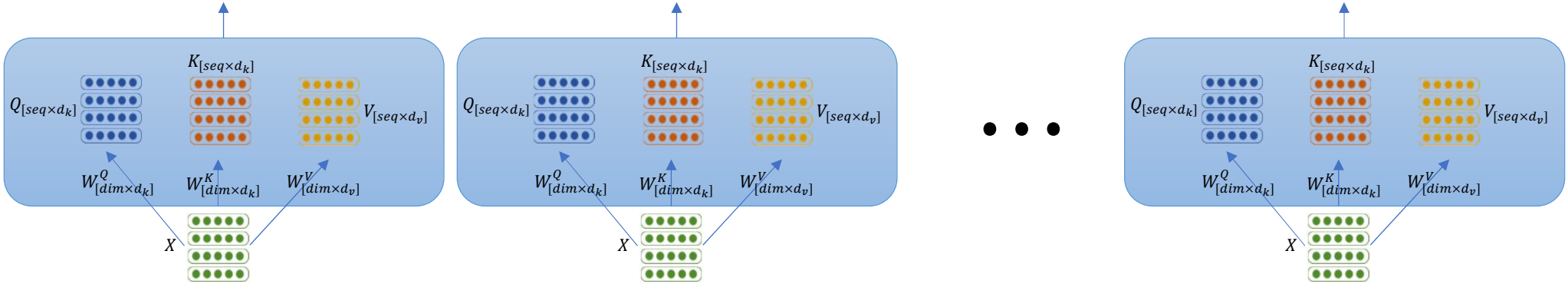
Auto-atención es una transformación de los datos de entrada. Para potenciar su uso, se usa varias veces, esperando que cada aplicación extraiga diferente información.

$$H^{output}_{[seq\times dim]} = H_{[seq\times(h\times d_v)]} W^{O}_{[(h\times d_v)\times dim]}$$

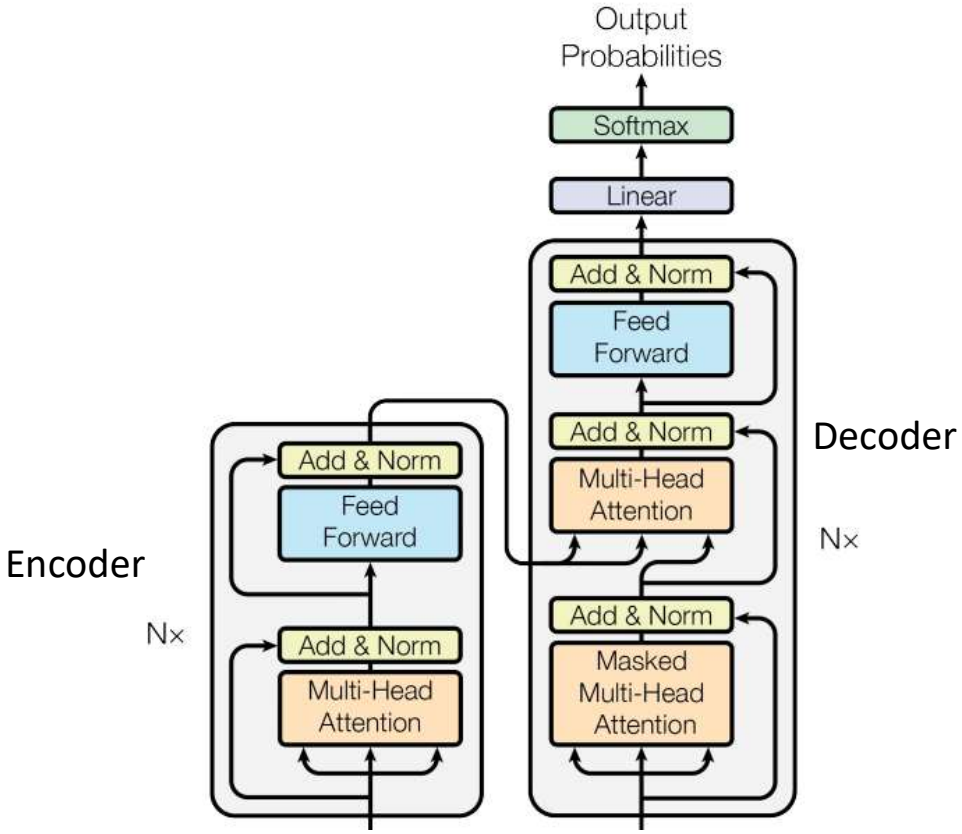
Capa lineal: hace que la salida tenga el mismo tamaño que el input

$$H_{[seq\times(h\times d_v)]} = [H^1_{[seq\times d_v]}, H^2_{[seq\times d_v]}, \dots, H^h_{[seq\times d_v]}]$$

Concatenación



Transformer



Encoder

- Shortcuts y normalización después de cada atención multi-cabeza
- MLP de dos capas (con ReLU)
- Este bloque se repite N veces

Decoder

- Self-attention sobre la secuencia target
- Atención sobre la salida del encoder
- Shortcuts y normalización
- MLP al finalizar el bloque

Un MLP y softmax al final

Transformer

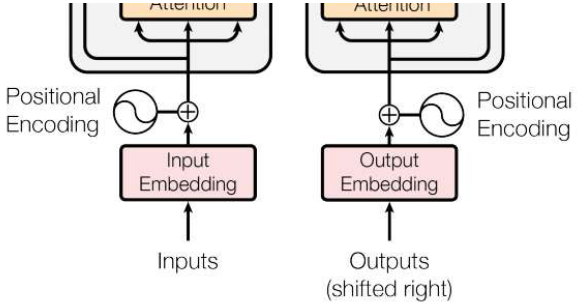
Qué sucede con el orden de la secuencia en esta arquitectura?
Hemos perdido información del orden temporal

Solución: positional encoding

- Agregar información al embedding inicial para saber en qué parte de la secuencia sucede

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

- La propuesta original es usar funciones periódicas en función de posición y dimensión del embedding

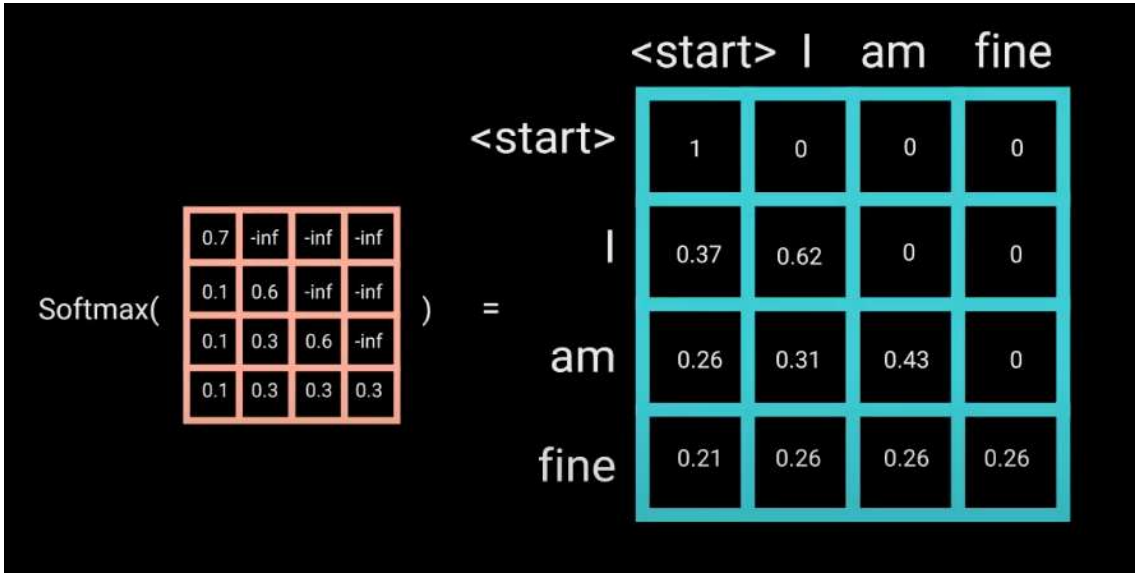


Transformer

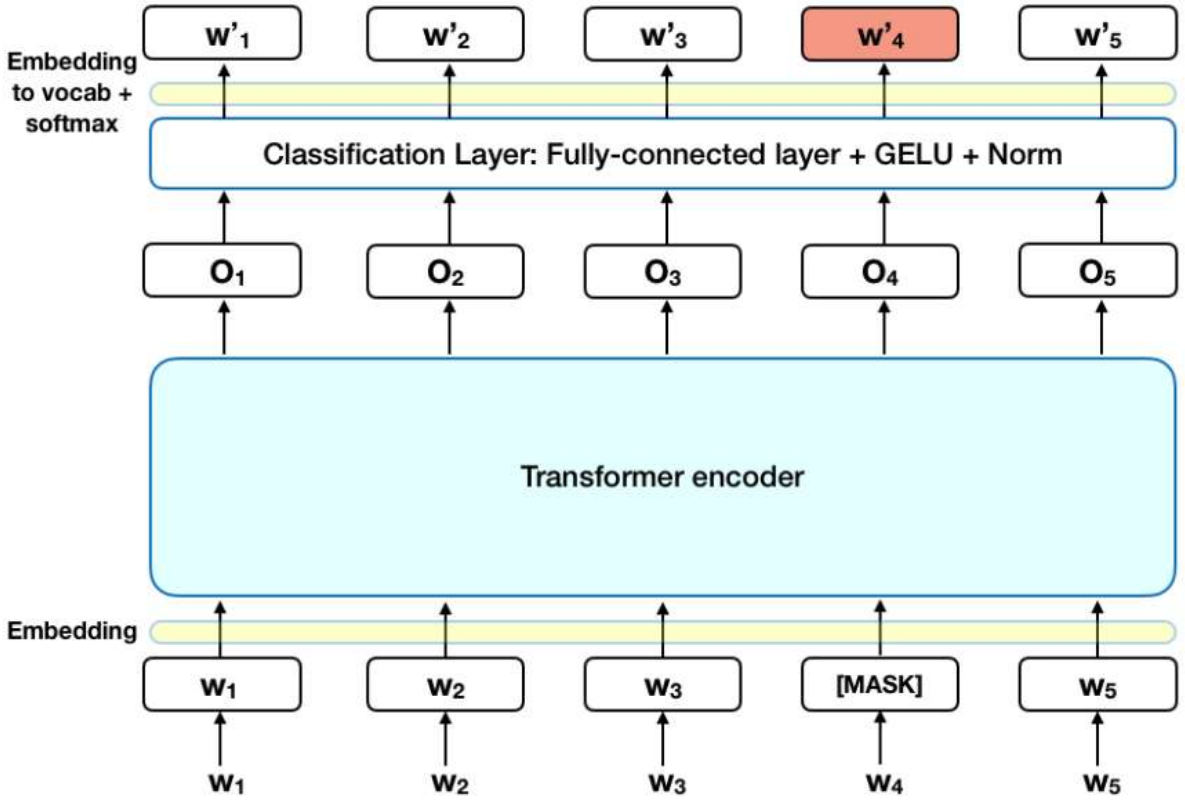
Un detalle más: el decoder debería aprender a poner atención sólo a los tokens que ocurrieron hasta un determinado momento. Al momento de hacer inferencia, no se conoce el futuro.

Esto se puede resolver fácil en el mismo cómputo de la auto-atención.

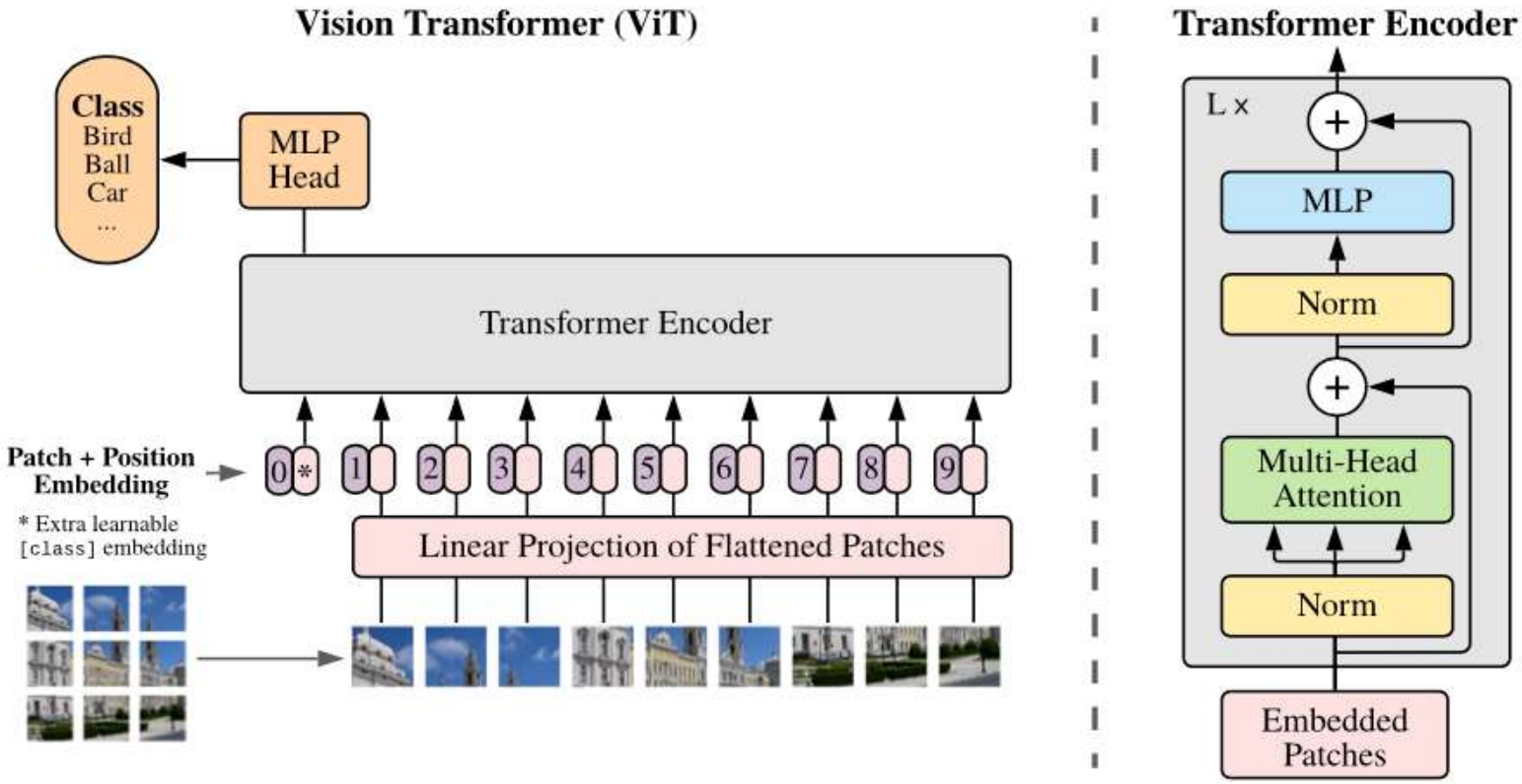
Usar a propósito un score de atención $-inf$ para posiciones futuras. Softmax se hará cargo de poner en cero los scores de atención.



BERT


w'1
w'2
w'3
w'4
w'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

ViT


Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0 * 1 2 3 4 5 6 7 8 9
Linear Projection of Flattened Patches
Transformer Encoder
L x
MLP
Norm
Multi-Head Attention
Norm
Embedded Patches