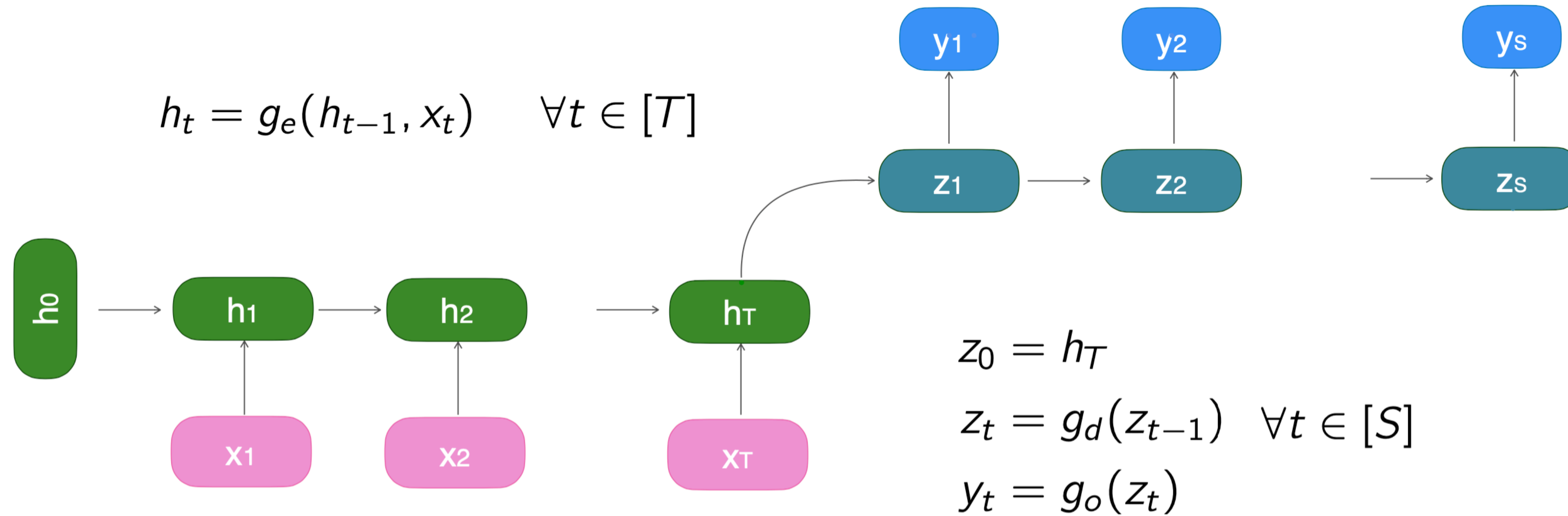


Transformers

“Attention is All you Need” - Vaswani et al.

Recordando modelos seq2seq



$$\begin{aligned} h_t &= g_e(h_{t-1}, x_t) = \sigma_e(W_e h_{t-1} + U x_t + b_e) \\ z_t &= g_d(z_{t-1}) = \sigma_d(W_d z_{t-1} + b_d) \\ y_t &= g_o(z_t) = \sigma_o(V z_t + b_o) \end{aligned}$$

Implementación vía RNN
Simple FC (densa)

Problemas de las RNN

Discutimos también algunos problemas asociados a las implementaciones basadas en RNN, principalmente 2:

Dependencias de largo plazo

Napoleon

From Wikipedia, the free encyclopedia


This article is about Napoleon I. For other uses, see [Napoleon \(disambiguation\)](#) and [Napoleon Bonaparte \(disambiguation\)](#).

Napoleon Bonaparte^[a] (born **Napoleone Buonaparte**; 15 August 1769 – 5 May 1821), and later known by his regnal name **Napoleon I**, was a French military and political leader who rose to prominence during the [French Revolution](#) and led [several successful campaigns](#) during the [Revolutionary Wars](#). He was the *de facto* leader of the French Republic as First Consul from 1799 to 1804. As Napoleon I, he was [Emperor of the French](#) from 1804 until 1814 and again in 1815. Napoleon's political and cultural legacy has endured, and he has been one of the most celebrated and controversial leaders in world history.^{[3][4]}

Napoleon was born on the island of [Corsica](#) not long after its annexation by the [Kingdom of France](#).^[6] He supported the [French Revolution](#) in 1789 while serving in the French army, and tried to spread its ideals to his native Corsica. He rose rapidly in the Army after he saved the governing [French Directory](#) by [firing on royalist insurgents](#). In 1796, he began a [military campaign](#) against the Austrians and their Italian allies, scoring decisive victories and becoming a national hero. Two years later, he led a [military expedition to Egypt](#) that served as a springboard to political power. He engineered a [coup in November 1799](#) and became *First Consul of the Republic*. Differences with the British meant that the French faced the [War of the Third Coalition](#) by 1805. Napoleon shattered this coalition with victories in the [Ulm Campaign](#), and at the [Battle of Austerlitz](#), which led to the dissolving of the [Holy Roman Empire](#). In 1806, the [Fourth Coalition](#) took up arms against him because [Prussia](#) became worried about growing French influence on the continent. Napoleon knocked out Prussia at the [battles of Jena and Auerstedt](#), marched the *Grande Armée* into [Eastern Europe](#), annihilating the Russians in June 1807 at [Friedland](#), and forcing the defeated nations of the Fourth Coalition to accept the [Treaties of Tilsit](#). Two years later, the Austrians challenged the French again during the [War of the Fifth Coalition](#), but Napoleon solidified his grip over Europe after triumphing at the [Battle of Wagram](#).

Hoping to extend the [Continental System](#), his embargo against Britain, Napoleon invaded the [Iberian Peninsula](#) and declared his brother [Joseph](#) King of Spain in 1808. The Spanish and the Portuguese revolted in the [Peninsular War](#), culminating in defeat for Napoleon's marshals. Napoleon launched an [invasion of Russia](#) in the summer of 1812. The resulting campaign witnessed the catastrophic retreat of Napoleon's *Grande Armée*. In 1813, Prussia and Austria joined Russian forces in a [Sixth Coalition](#) against France. A chaotic military campaign resulted in a large coalition army defeating Napoleon at the [Battle of Leipzig](#) in October 1813. The coalition [invaded France](#) and captured Paris, forcing Napoleon to abdicate in April 1814. He was exiled to the island of [Elba](#), between Corsica and Italy. In France, the [Bourbons](#) were [restored to power](#). However, Napoleon escaped Elba in February 1815 and took control of France.^{[6][7]} The Allies responded by forming a [Seventh Coalition](#), which defeated Napoleon at the [Battle of Waterloo](#) in June 1815. The British exiled him to the remote island of [Saint Helena](#) in the [Atlantic](#), where he died in 1821 at the age of 51. Napoleon had an extensive impact on the modern world, bringing liberal reforms to the many countries he conquered, especially the [Low Countries](#), Switzerland, and parts of modern Italy and Germany. He implemented liberal policies in France and Western Europe.^[b]

Napoleon

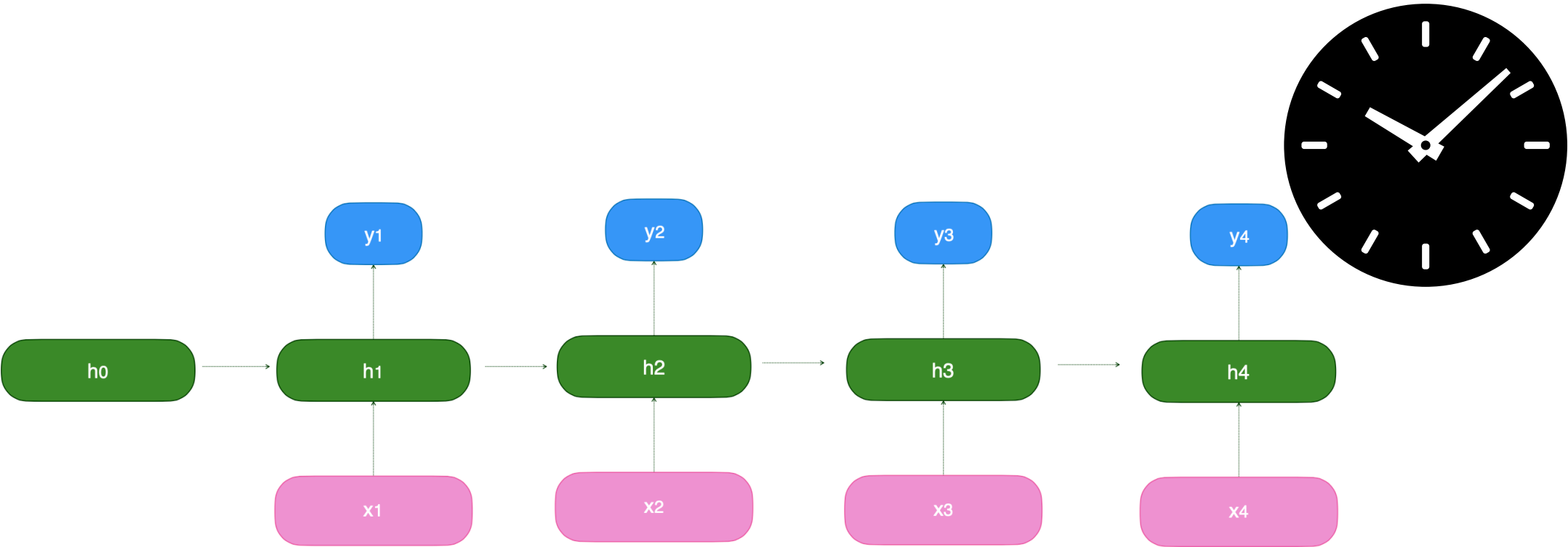


The Emperor Napoleon in His Study at the Tuilleries, by Jacques-Louis David, 1812

Emperor of the French (more...)

1st reign	18 May 1804 – 6 April 1814
Coronation	2 December 1804 Notre-Dame Cathedral
Successor	Louis XVIII (as King of France)
2nd reign	20 March 1815 – 22 June 1815

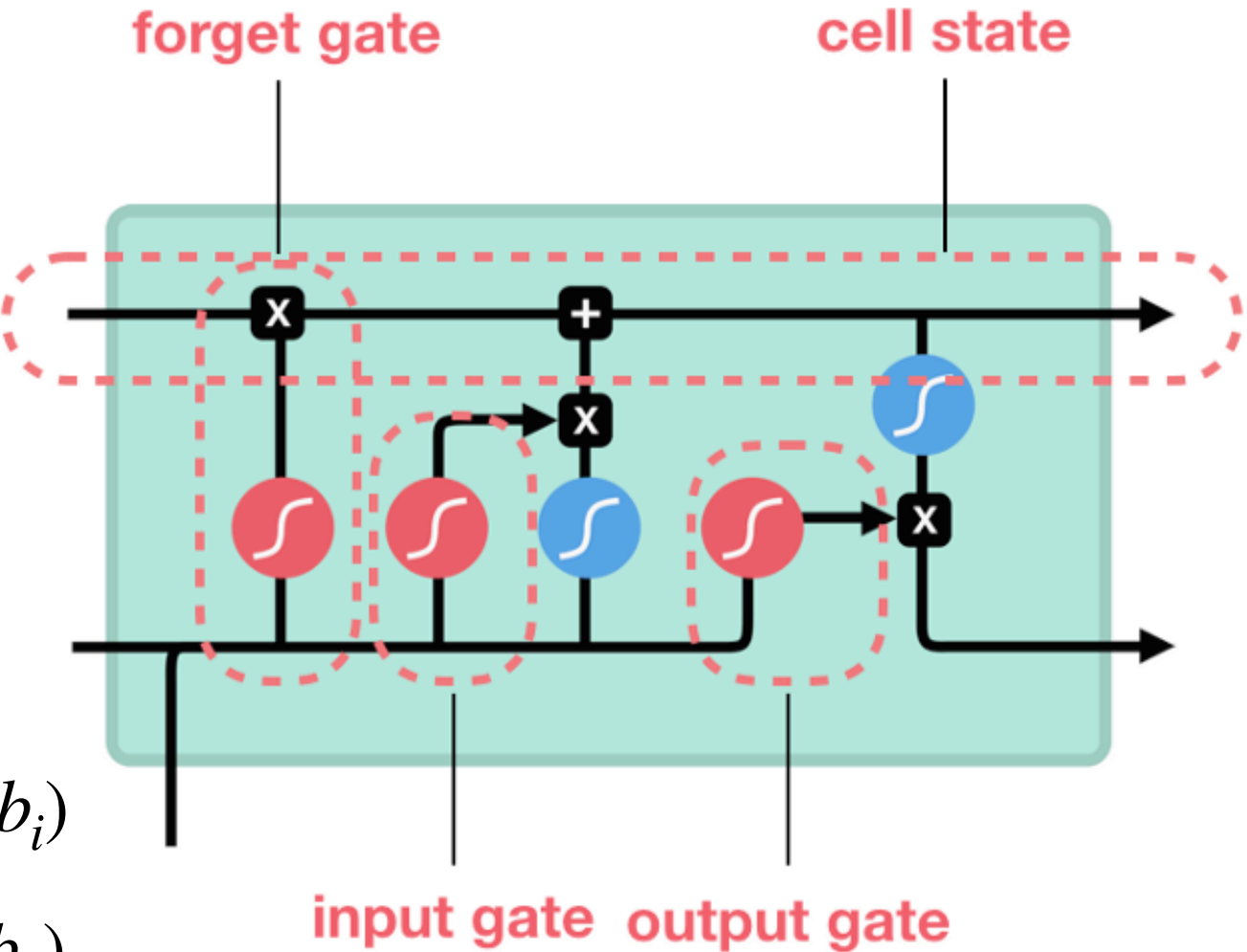
Lentitud en el entrenamiento



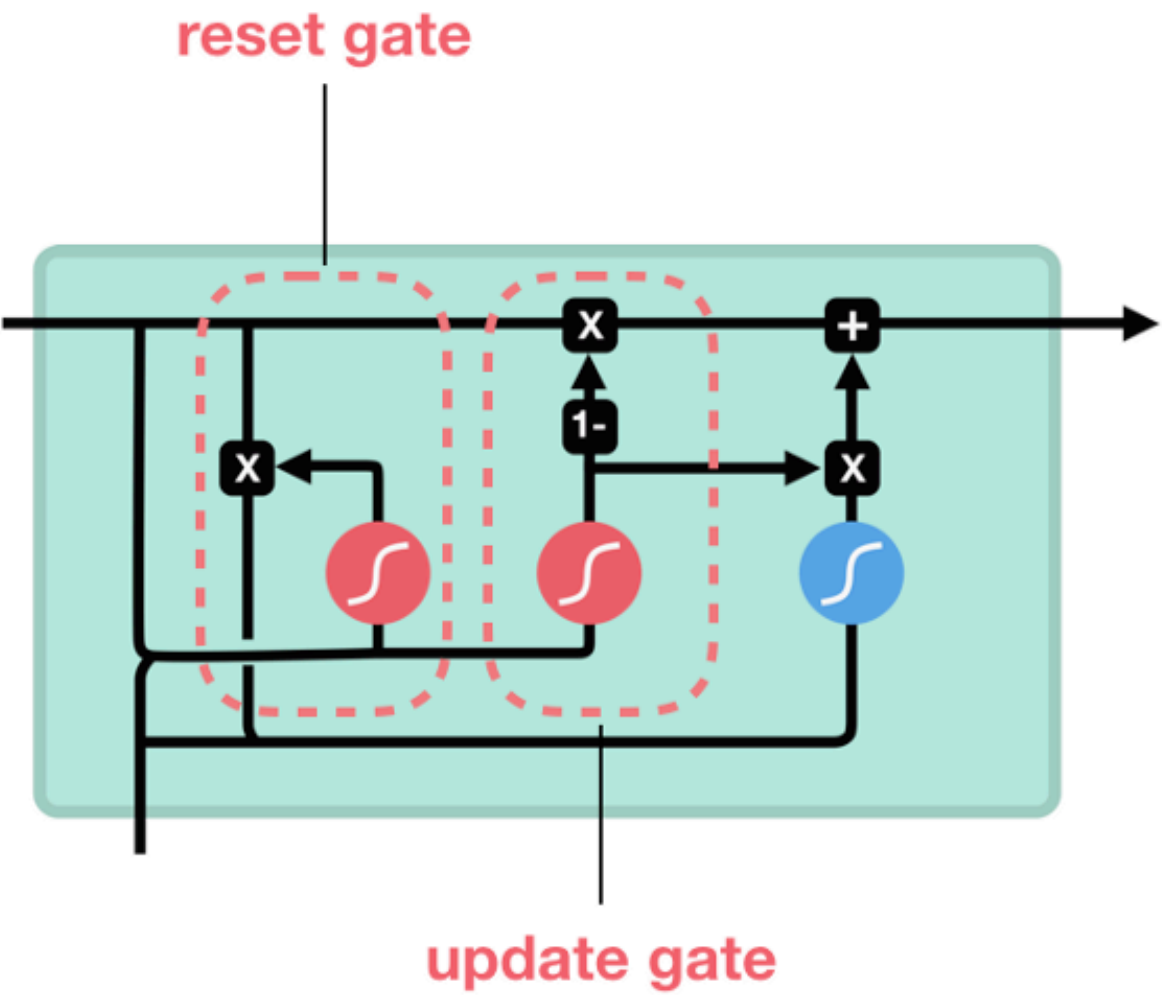
Solución a secuencias largas: Memoria

Una solución al problema de aprender dependencias de largo plazo/ aprender de secuencias largas es incorporar un mecanismo de “memoria” en las neuronas que pueda expresarse en su estado interno (h_t) y dotarlas de una capacidad de actualizar esa memoria u olvidarla (mediante las compuertas).

LSTM



GRU



$$i_t(x_t, h_{t-1}) = \sigma(W_i X_t + U_i h_{t-1} + b_i)$$
$$o_t(x_t, h_{t-1}) = \sigma(W_o X_t + U_o h_{t-1} + b_o)$$
$$f_t(x_t, h_{t-1}) = \sigma(W_f X_t + U_f h_{t-1} + b_f)$$
$$c_t = f_t \odot c_{t-1} + i \odot \tanh(WX + Uh_{t-1})$$
$$h_t = o_t \odot \tanh(c_t)$$



sigmoid



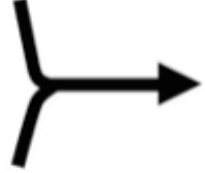
tanh



pointwise multiplication



pointwise addition

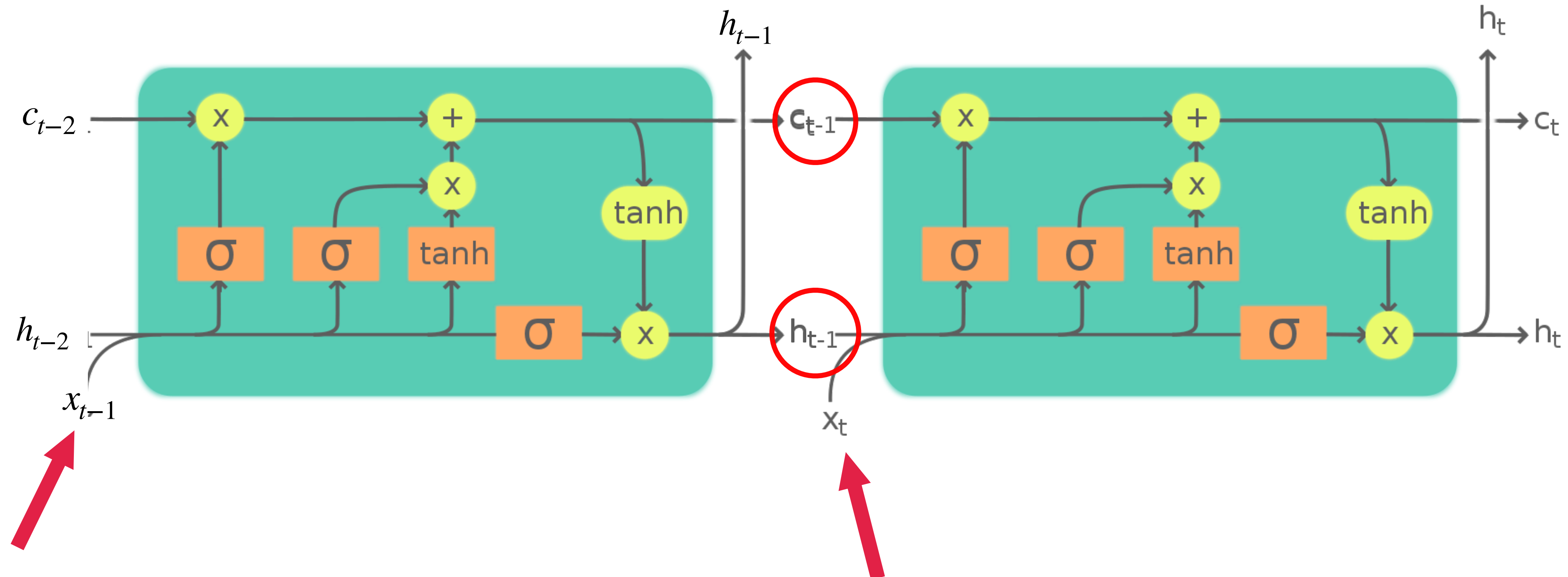


vector concatenation

$$z_t(x_t, h_{t-1}) = \sigma(W_z X_t + U_z H_{t-1} + b_z)$$
$$r_t(x_t, h_{t-1}) = \sigma(W_r X_t + U_r h_{t-1} + b_r)$$
$$g_t = \phi(W_g X_t + U_g(r \odot h_{t-1}) + b_g)$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot g_t$$

Algunos problemas persisten: Entrenamiento lento

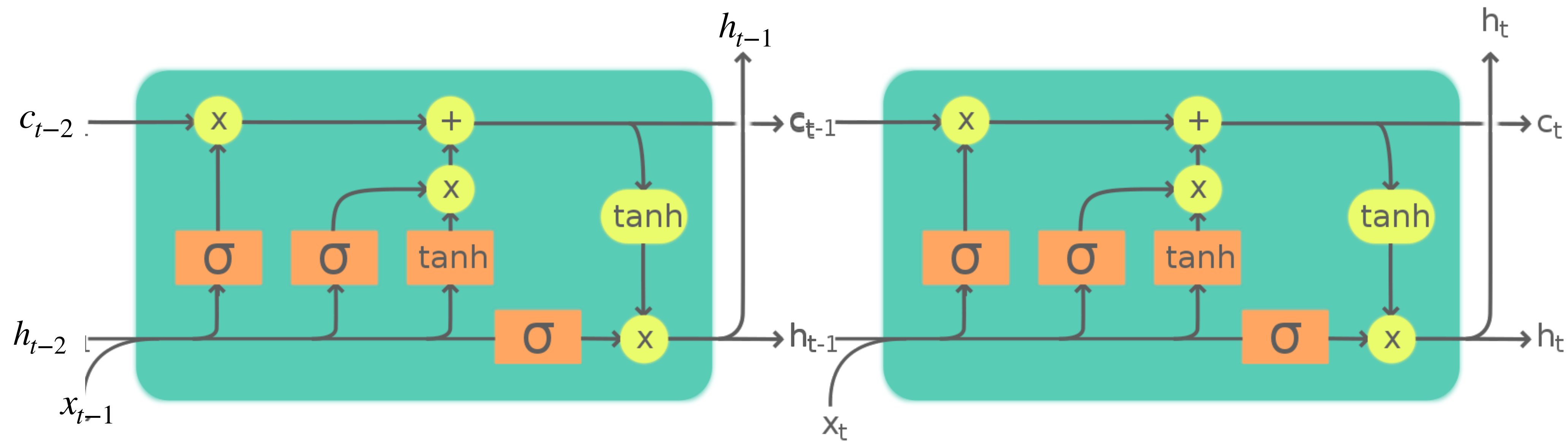
- Los datos de input aún deben computarse procesarse secuencialmente.
- Para computar h_t necesitamos haber terminado de computar h_{t-1} .



Algunos problemas persisten: Entrenamiento lento

- Los datos de input aún deben computarse procesarse secuencialmente.
- Para computar h_t necesitamos haber terminado de computar h_{t-1} .

Sería ideal poder ingestar los datos de manera no secuencial, todo de una sola vez



Transformer Model

Un transformer se estructura de manera encoder-decoder pero aplicando un tipo especial de mecanismo de atención, además, para poder sortear el problema de la “secuencialidad” del input (y de la generación de los estados internos) este modelo va a representar las secuencias de entrada y salida de una manera particular

Attention Is All You Need

Ashish Vaswani*

Google Brain

avaswani@google.com

Noam Shazeer*

Google Brain

noam@google.com

Niki Parmar*

Google Research

nikip@google.com

Jakob Uszkoreit*

Google Research

usz@google.com

Llion Jones*

Google Research

llion@google.com

Aidan N. Gomez* †

University of Toronto

aidan@cs.toronto.edu

Łukasz Kaiser*

Google Brain

lukaszkaizer@google.com

Illia Polosukhin* ‡

illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

The diagram illustrates the internal structure of the Transformer model, divided into an encoder (left) and a decoder (right), both consisting of N stacked layers.

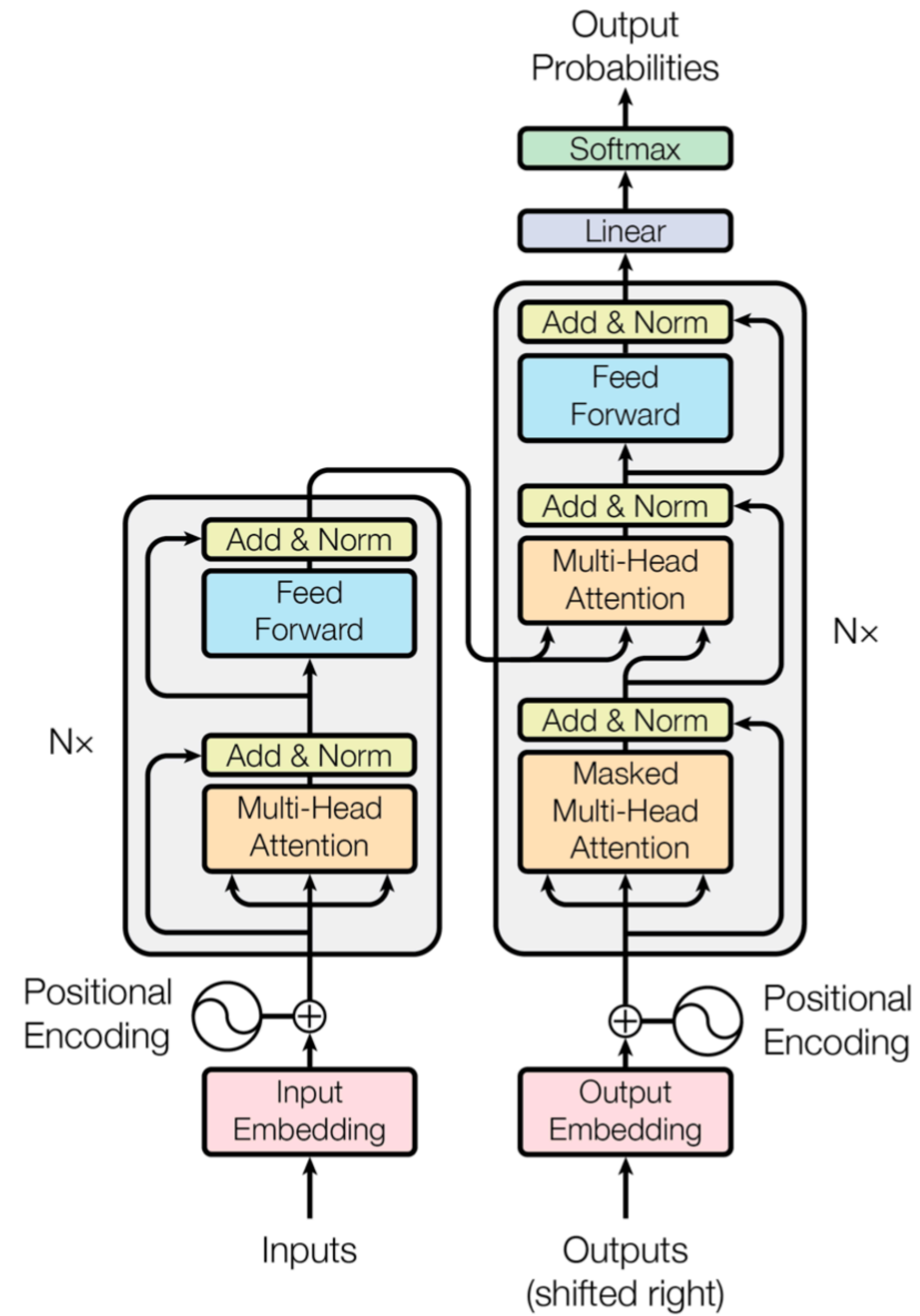
Encoder (Left): Each layer contains a **Multi-Head Attention** sub-layer followed by a **Feed Forward** sub-layer. Both sub-layers are followed by an **Add & Norm** block. A residual connection bypasses the attention and feed-forward layers, adding the input of the layer to the output of the **Add & Norm** block. The input to the encoder is **Inputs**, which are passed through an **Input Embedding** layer and added to **Positional Encoding**.

Decoder (Right): Each layer contains a **Masked Multi-Head Attention** sub-layer, followed by a **Multi-Head Attention** sub-layer that takes input from the encoder's output, and finally a **Feed Forward** sub-layer. Each sub-layer is followed by an **Add & Norm** block. There are two residual connections: one bypassing the masked self-attention layer, and another bypassing the entire decoder block (adding the input of the layer to the output of the final **Add & Norm** block). The input to the decoder is **Outputs (shifted right)**, which are passed through an **Output Embedding** layer and added to **Positional Encoding**.

The final output of the decoder is passed through a **Linear** layer and a **Softmax** layer to produce **Output Probabilities**.

A portrait of Ashish Vaswani, one of the authors of the Transformer paper. He is a man with glasses, wearing a blue denim jacket over a dark shirt, standing with his arms crossed in front of a balcony railing with a cityscape and water in the background.

Transformer Model

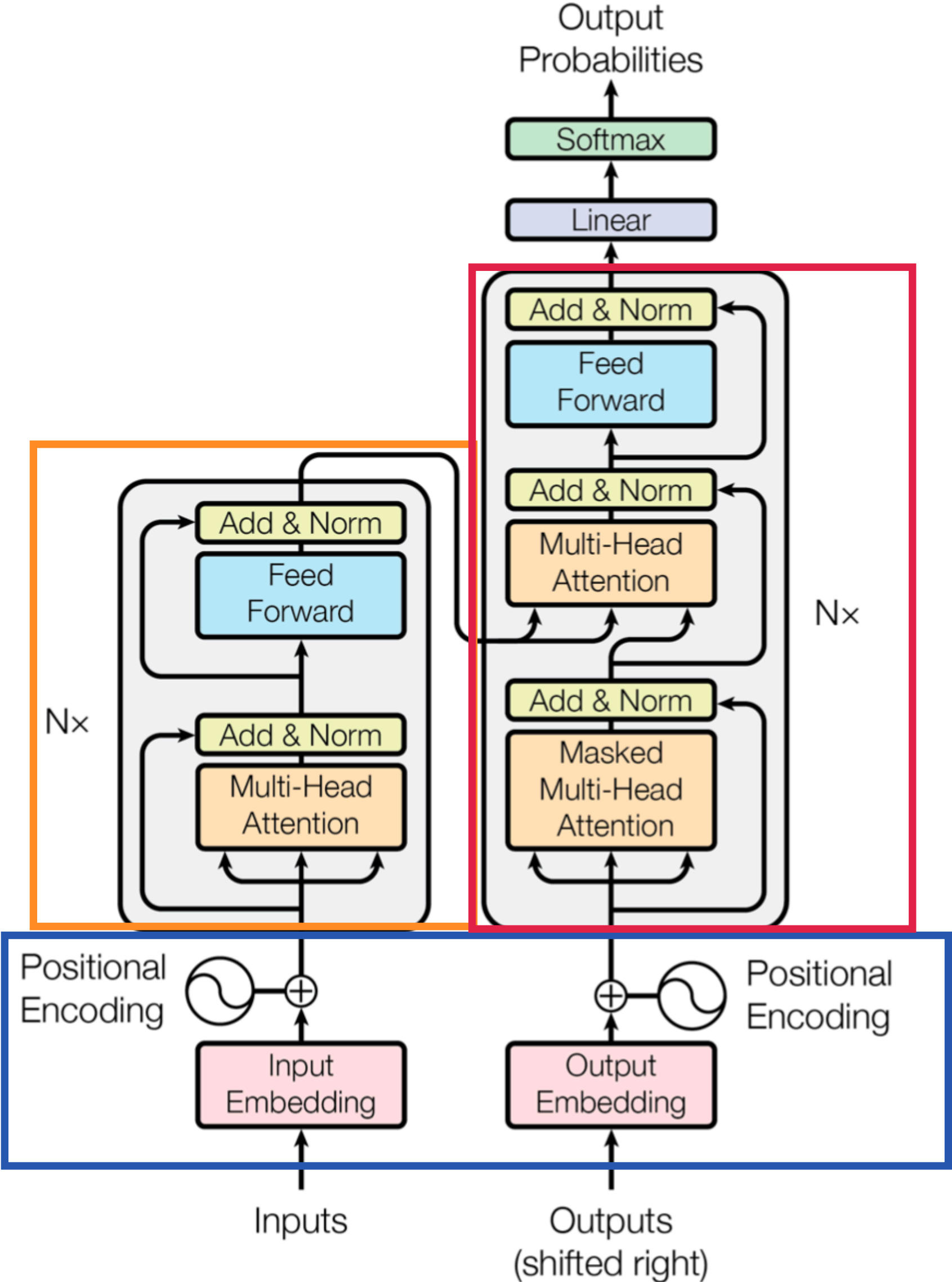


Transformer Model

Bloque de Encoder 2

Embeddings 1

3 Bloque de Decoder



1. Embeddings

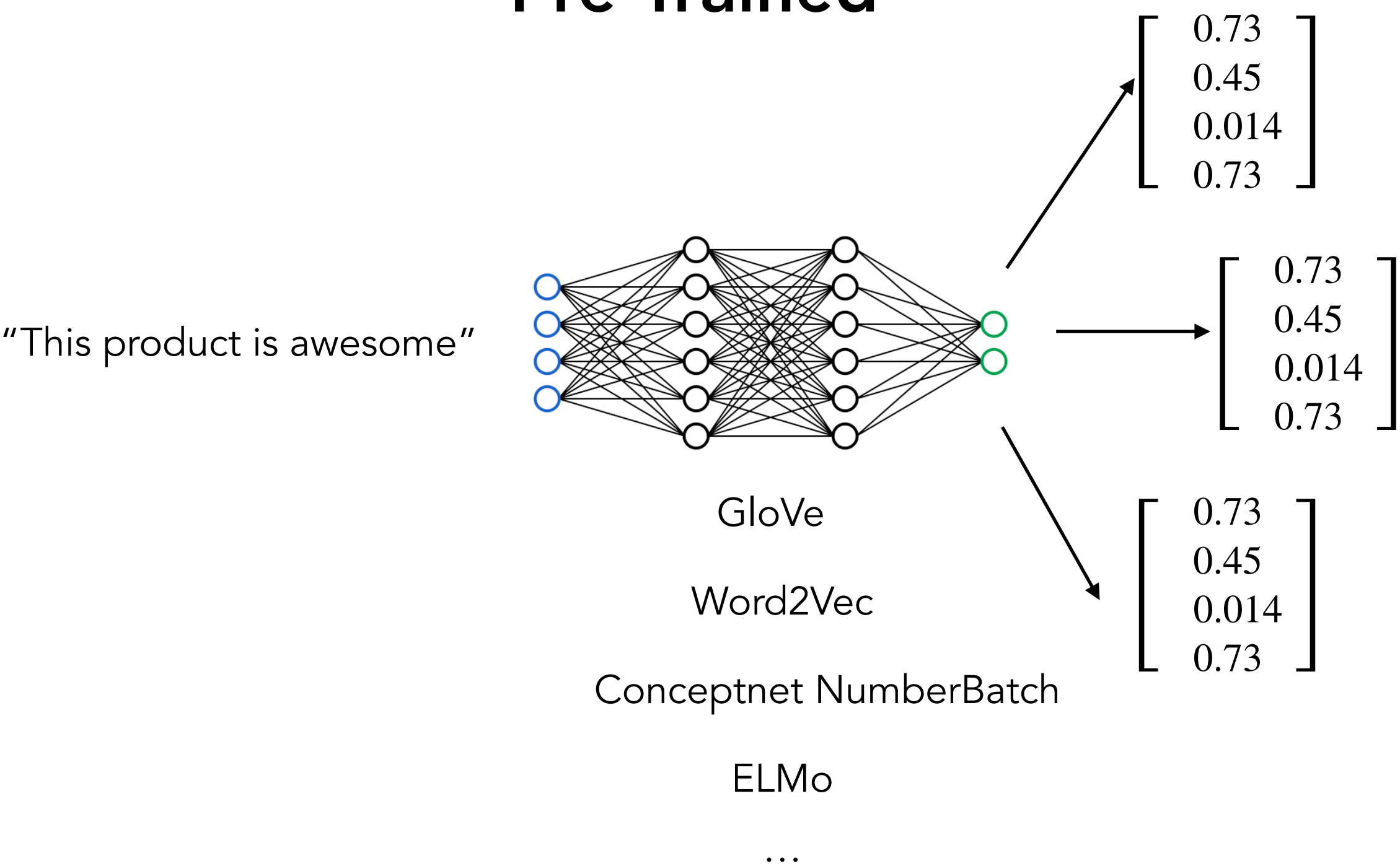
Antes de explicar el mecanismo usado por un transformer, evaluemos las opciones clásicas para enfrentar este problema (nos ayudará a entender el por qué de la decisión de los autores)

Como aproximaciones “clásicas” para representar palabras como vectores de embedding podríamos considerar dos:

BoW: Bag of Words

	are	call	from	hello	home	how	me	money	now	tomorrow	win	you
0	1	0	0	1	0	1	0	0	0	0	0	1
1	0	0	1	0	1	0	0	1	0	0	2	0
2	0	1	0	0	0	0	1	0	1	0	0	0
3	0	1	0	1	0	0	0	0	0	1	0	1

Pre-Trained



1. Embeddings

El contexto de una palabra es tanto local como global: Por un lado una palabra toma un determinado significado semántico dependiendo de su uso en una frase específica, así como también dentro de su significado en el lenguaje al que pertenece.

El significado completo de una palabra depende del contexto donde se ocupe:

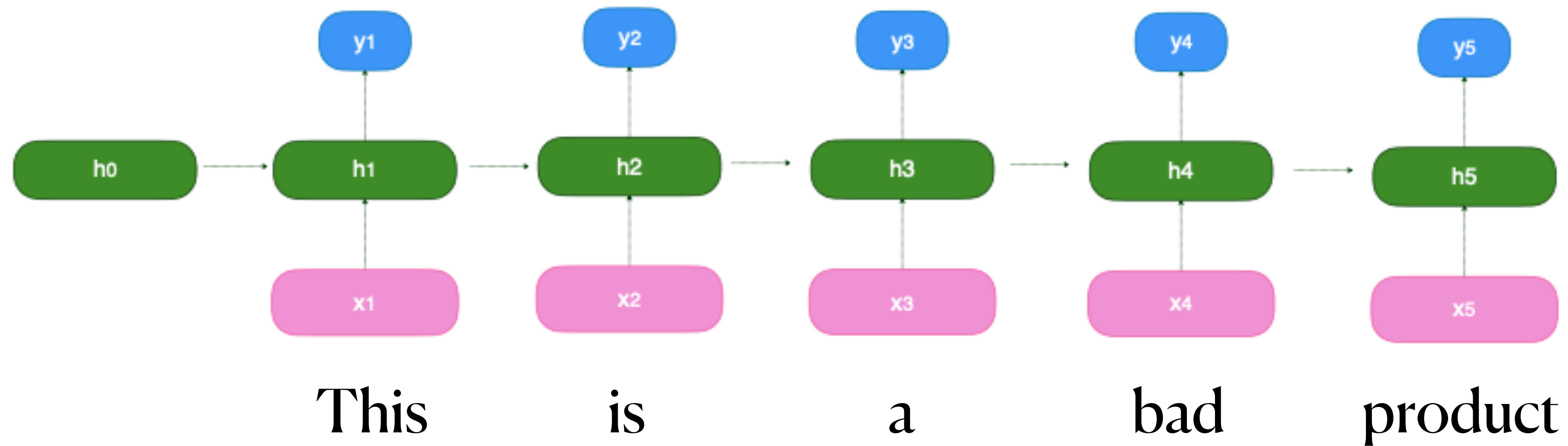
“This is a really **bad** product”

“This product is actually not **bad** at all!”

Ambos tipos de embeddings como los propuestos anteriormente tienen uno u otro problema, ya sea ignoran el contexto donde se encuentra la palabra al momento de entregarnos el embedding (GloVe, Word2Vec) o son demasiado dispersos, ignoran el orden de las palabras en la sentencia (BoW).

1. Embeddings

Los modelos recurrentes lidian con el problema anterior imbuyendo en su arquitectura la lógica para computar sobre la lógica secuencial de los datos de entrada mediante su mecanismo de ingesta secuencial del input y la adición del vector de estado interno previo $h_t(x_t, h_{t-1})$ (u otro tipo de paso/regulación de información a través de los pasos temporales como la LSTM y GRU)



1. Embeddings

Para poder ingestar la secuencia completamente (y tener el contexto completo de cada palabra inmediatamente) el modelo Transformer trata con este problema mediante algo denominado **“positional encoding”**, que es un vector que da información de la posición de un token en la oración.

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

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

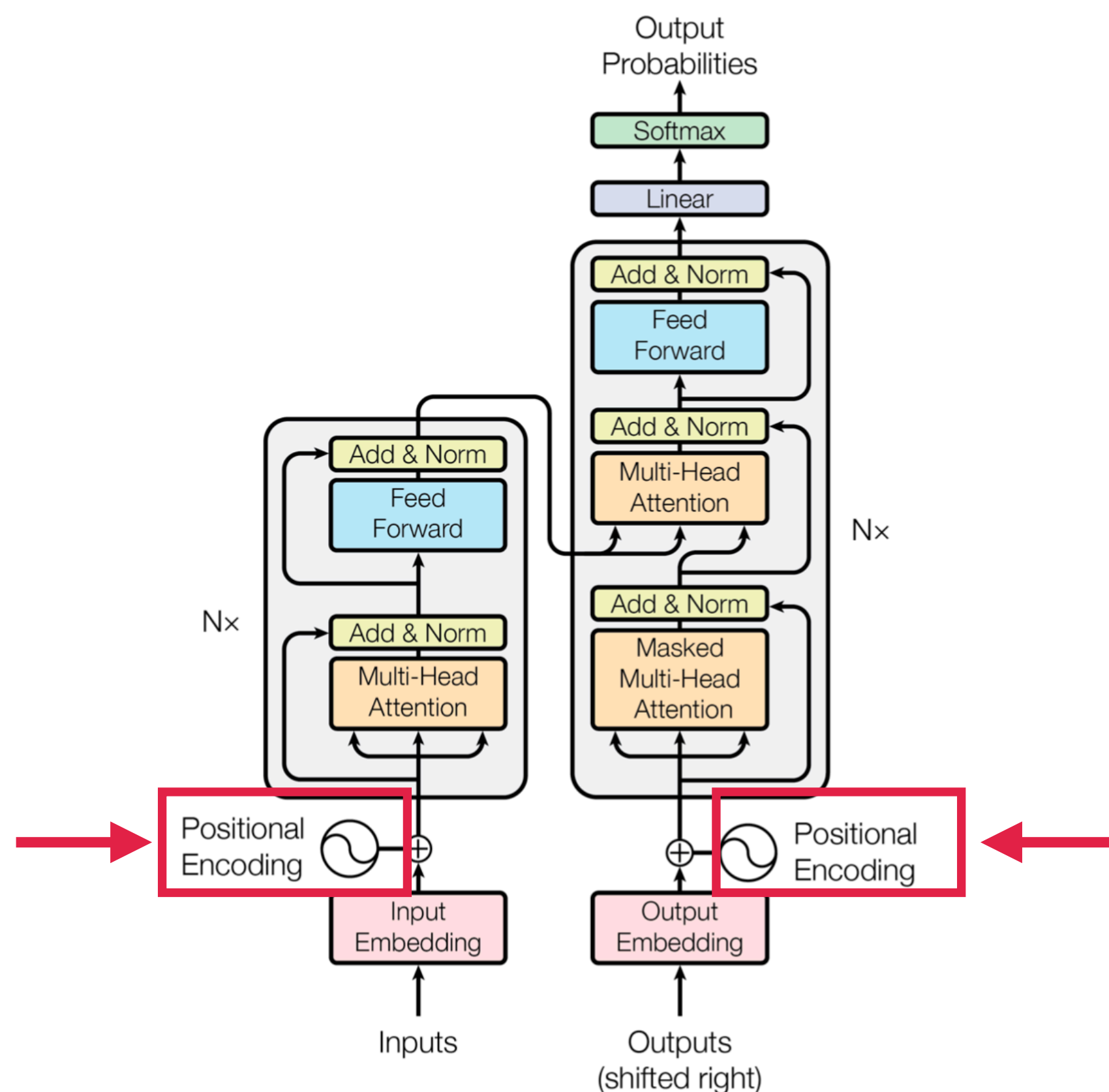
Donde:

pos : Es la posición del token en la secuencia .

d_{model} : Dimensionalidad del espacio de embedding, es un hiperparámetro .

$P(pos, \cdot)$: Función de mapeo de posición para una posición pos en la secuencia de entrada la posición (pos, \cdot) en la matriz de posiciones.

i : Usado para describir los índices de columnas, toma valores $0 \leq i \leq d/2$.



1. Embeddings

Para poder ingestar la secuencia completamente (y tener el contexto completo de cada palabra inmediatamente) el modelo Transformer trata con este problema mediante algo denominado **“positional encoding”**, que es un vector que da información de la posición de un token en la oración.

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

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

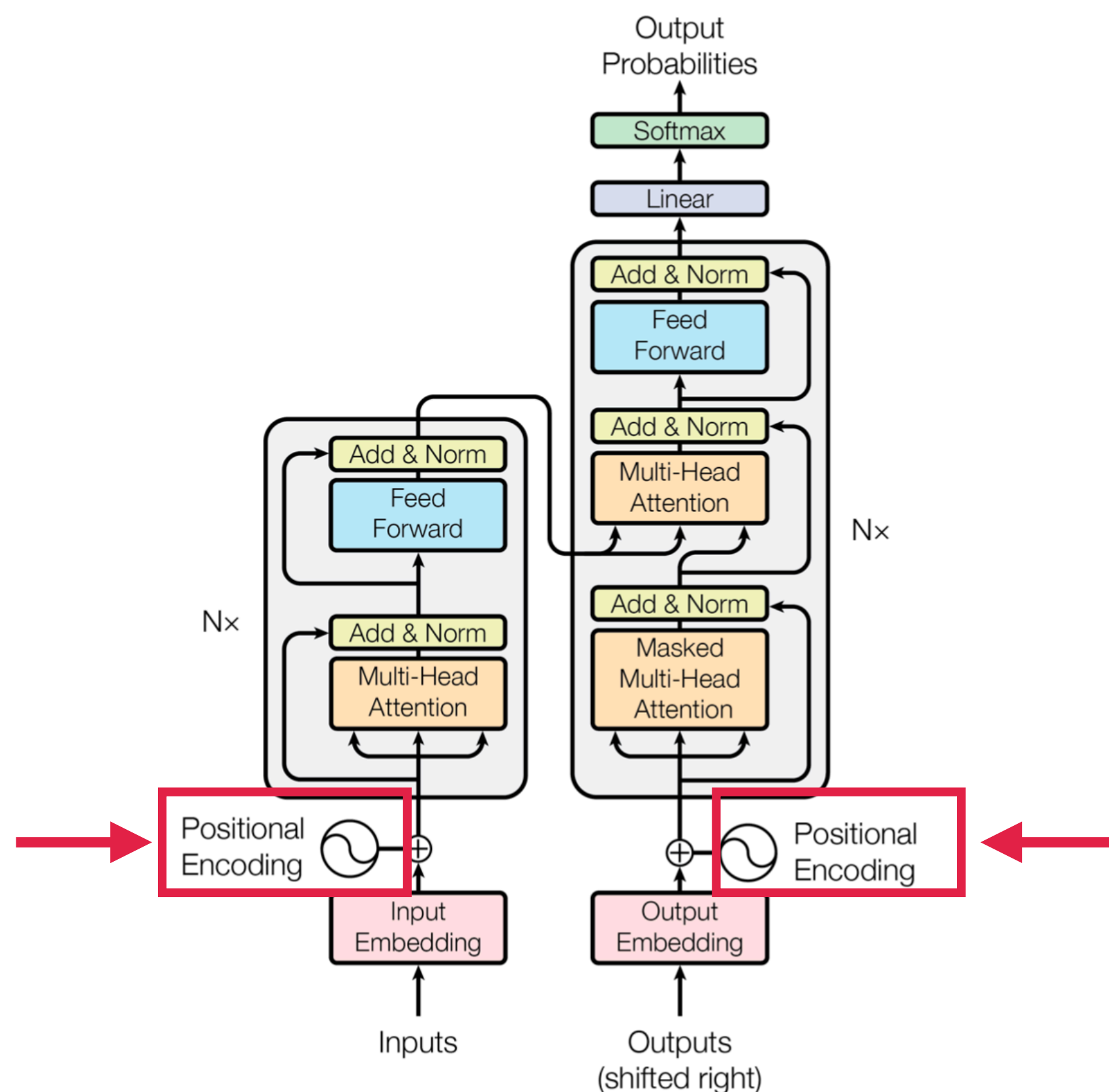
Por ejemplo, para el primer token de una sentencia, el vector de encodings posicionales será:

$$P_{00} = \sin(0/10000^{2 \cdot 0/4}) = \sin(0) = 0$$

$$P_{01} = \cos(0/10000^{2 \cdot 0/4}) = \cos(0) = 1$$

$$P_{02} = \sin(0/10000^{2 \cdot 1/4}) = \sin(0) = 0$$

$$P_{04} = \cos(0/10000^{2 \cdot 1/4}) = \cos(0) = 1$$

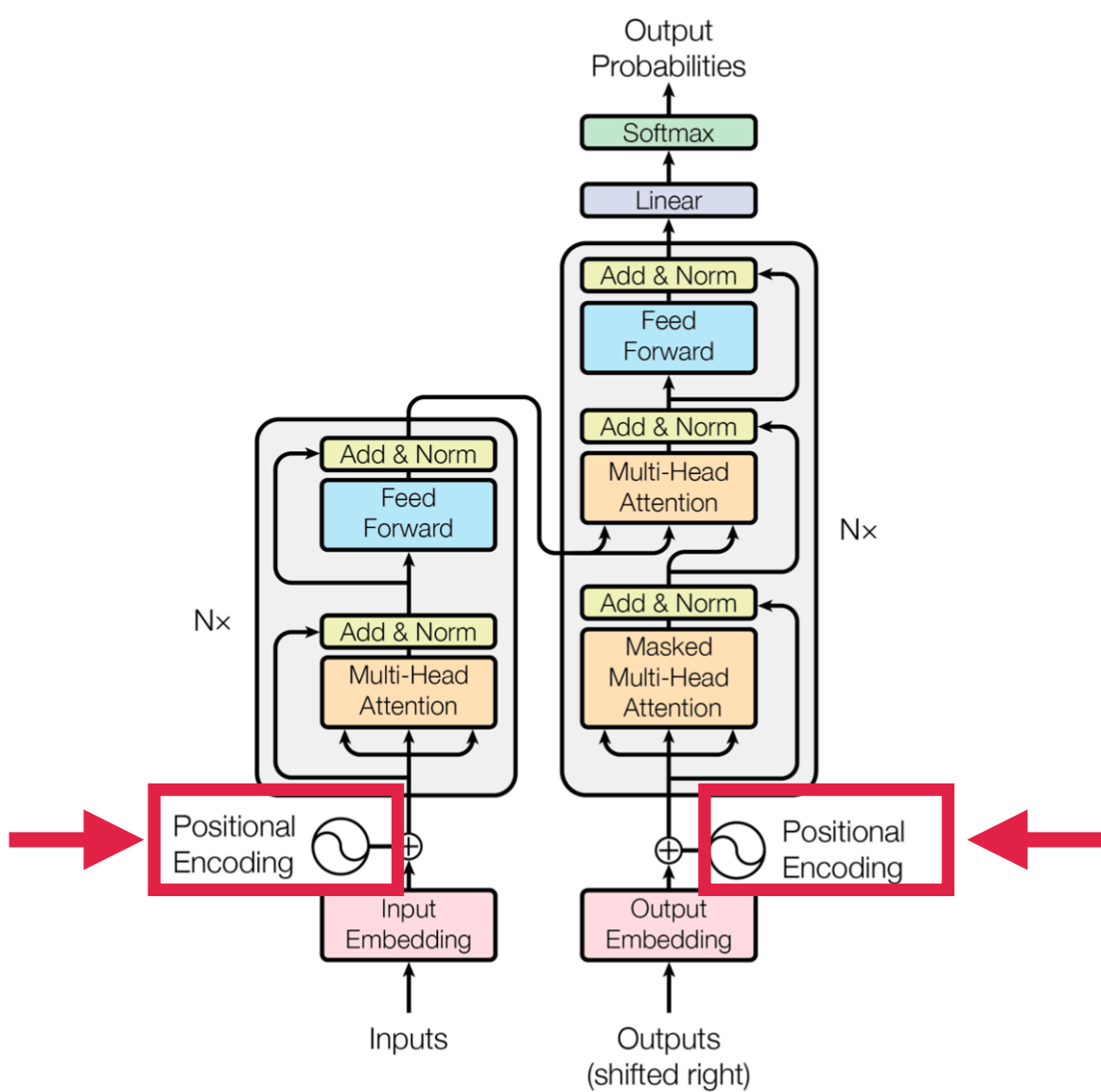


1. Embeddings

Para poder ingestar la secuencia completamente (y tener el contexto completo de cada palabra inmediatamente) el modelo Transformer trata con este problema mediante algo denominado **“positional encoding”**, que es un vector que da información de la posición de un token en la oración.

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

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



Indice en
la secuencia

0 Omelette \rightarrow $\begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ 0 & 1 & 0 & 1 \end{bmatrix}$

1 du \rightarrow $\begin{bmatrix} P_{10} & P_{11} & P_{12} & P_{13} \\ 0.01745 & 0.9998 & 1.74e^{-6} & 0.9999 \end{bmatrix}$

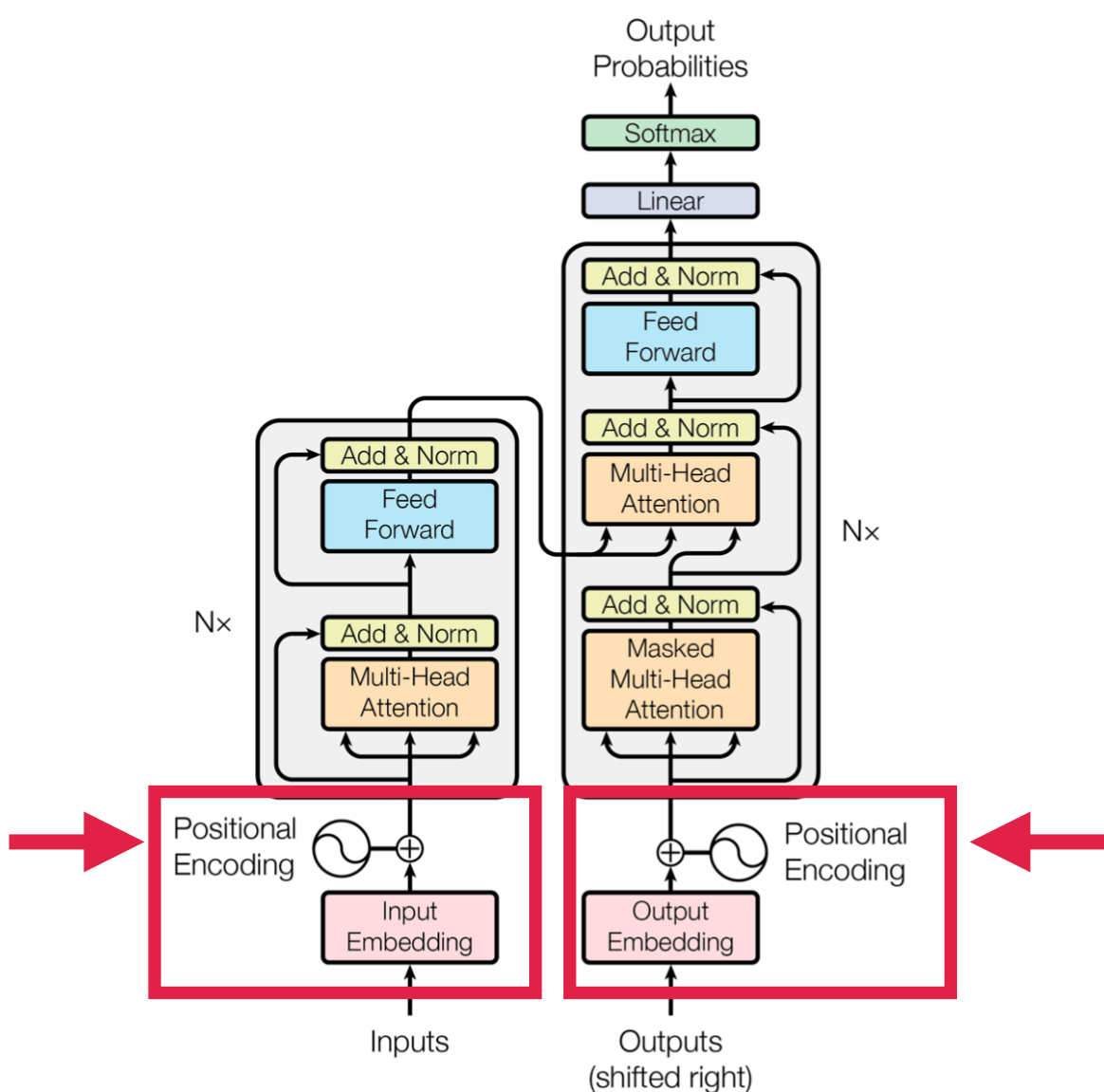
2 fromage \rightarrow $\begin{bmatrix} P_{20} & P_{21} & P_{22} & P_{23} \\ 0.03489 & 0.99939 & 3.4906e^{-4} & 0.9999 \end{bmatrix}$

1. Embeddings

Con esto podemos enriquecer un embedding “clásico” de un token, sumando a su embedding “clásico” su correspondiente embedding posicional, obteniendo un embedding que representa no solo la palabra en un contexto global de uso en el lenguaje, sino que un matiz específico a la sentencia que se tiene en el conjunto de entrenamiento.

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

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



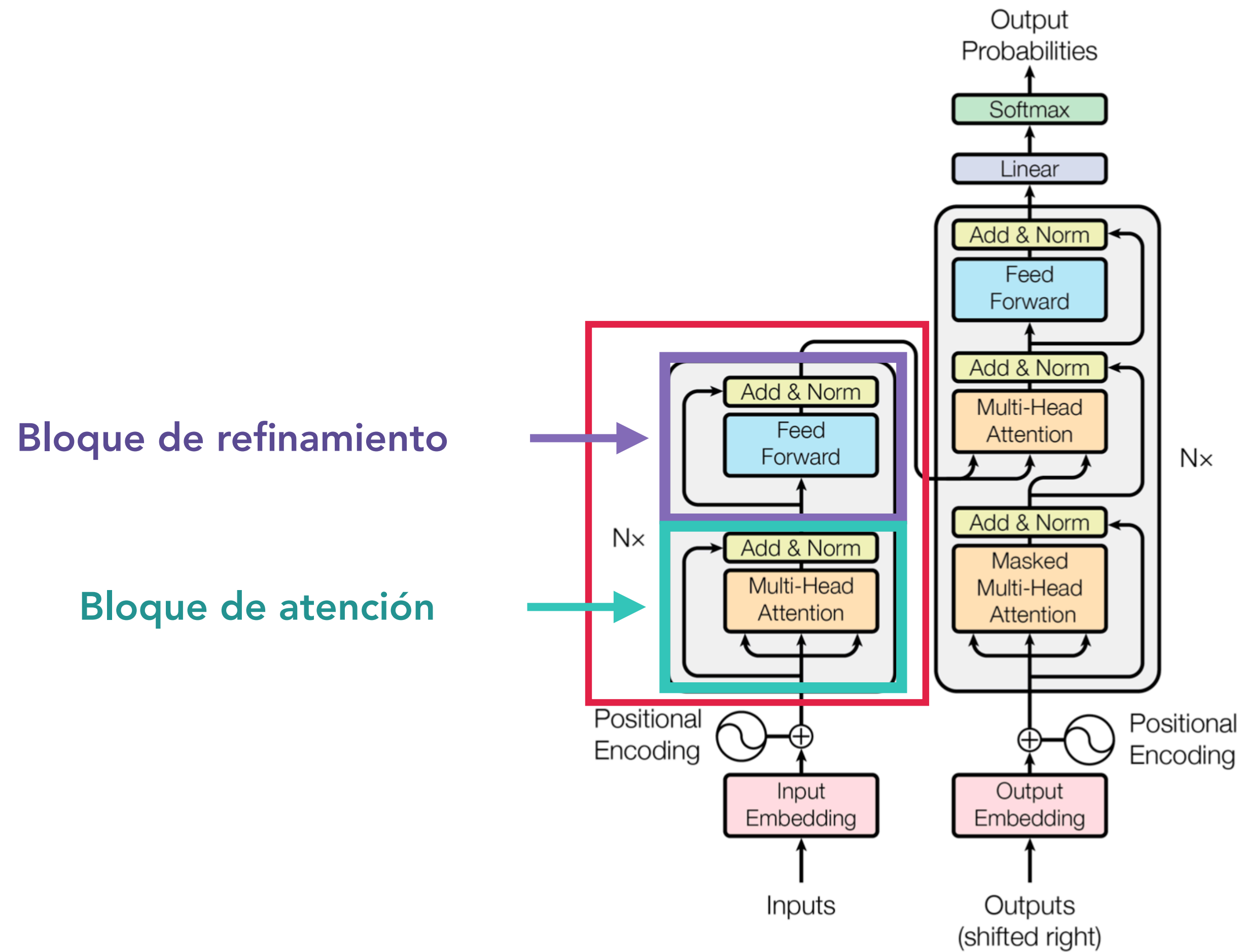
Omelette

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \oplus \begin{bmatrix} 0.0174 \\ 0.9998 \\ 1.74e^{-6} \\ 0.9999 \end{bmatrix} = \begin{bmatrix} 0.0174 \\ 1.9998 \\ 1.74e^{-6} \\ 1.9999 \end{bmatrix}$$

Encoding posicional Embedding del token Embedding enriquecido

2. Bloque de encoder

Para discutir el bloque del encoder vamos a distinguir dos “partes” principales, nombrémoslas como: El bloque de atención y el bloque de “refinamiento”.



2. Bloque de encoder - Sub-Bloque de atención

¿En qué parte de x_t nos tenemos que centrar?

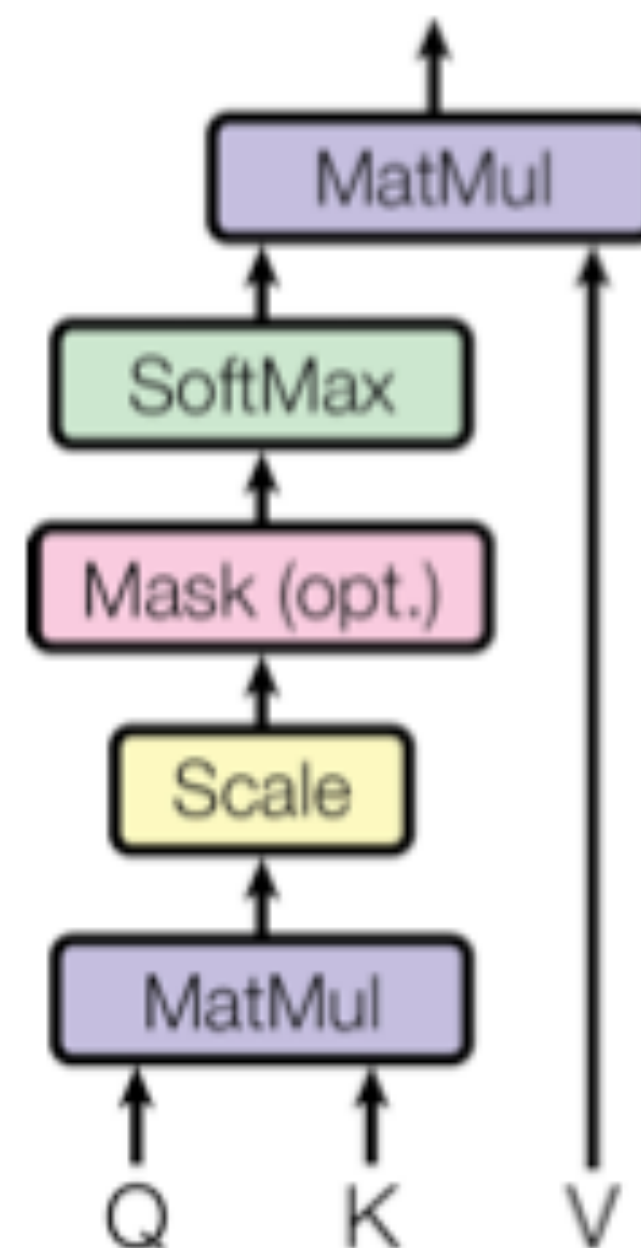
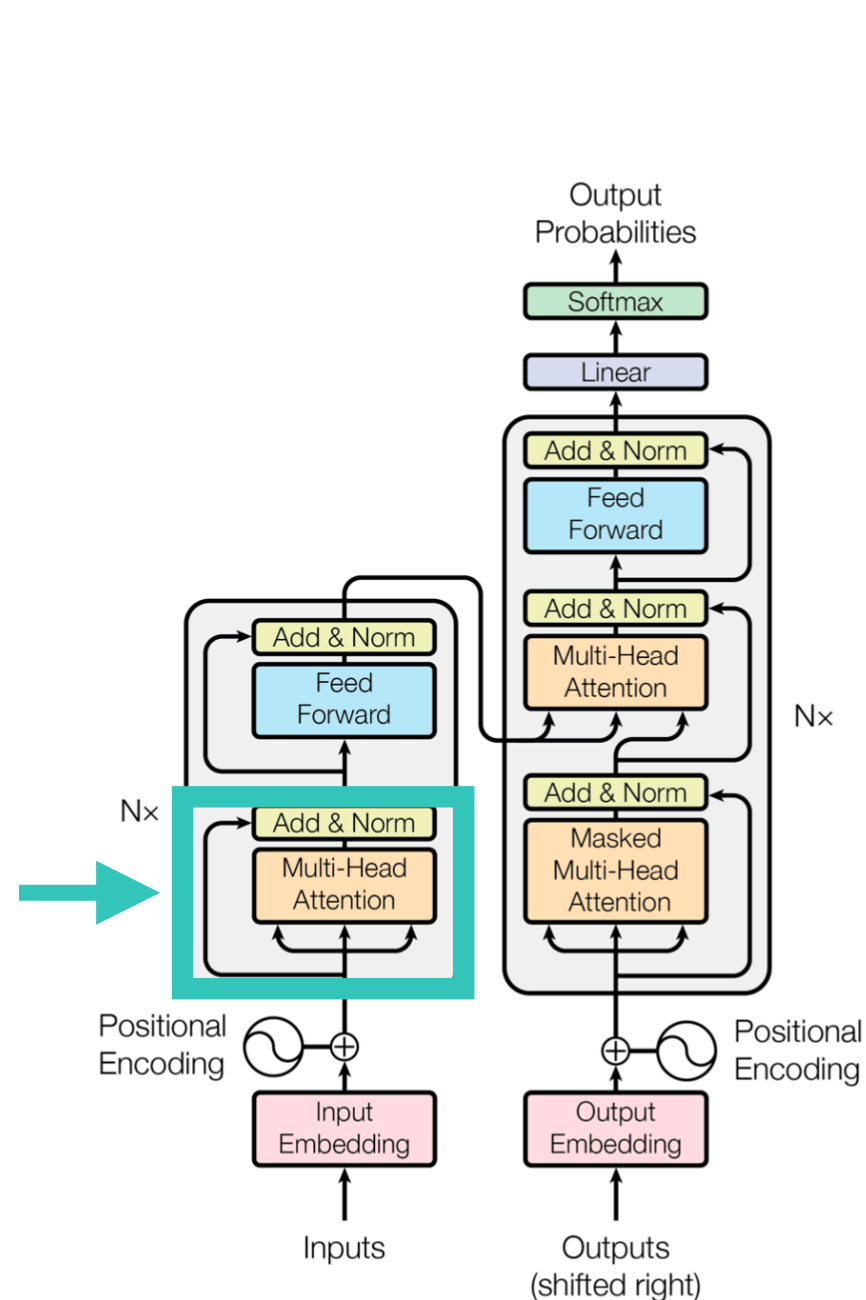
El modelo Transformer implementa un tipo particular de atención (presentado en el mismo paper) llamado “*Scaled Dot-Product Attention*” que requiere de tres vectores:

Q : Matriz de vectores de “consulta”, cada uno de dimensión d_k (misma que para K).

K : Matriz de vectores de “llave”, cada uno de dimensión d_k .

V : Matriz de vectores de “valor”, cada uno de dimensión d_v .

Scaled Dot-Product Attention



$$attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

2. Bloque de encoder - Sub-Bloque de atención

Entendiendo Q, K y V

Recordemos lo que hacía Bahdanau et al:

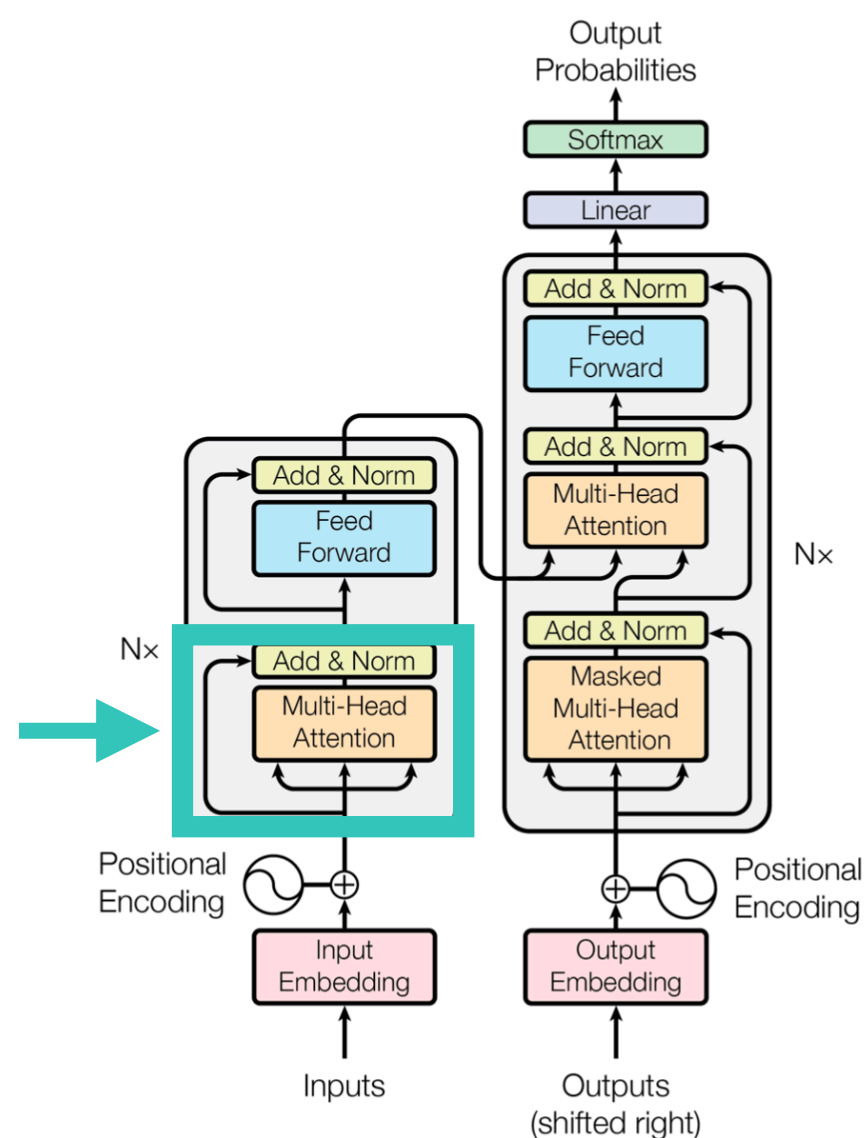
Se basaba en computar un puntaje de atención e_{ij} que represente la importancia relativa de los inputs en la secuencia (keys) para el output particular (query), al multiplicar los alineamientos (puntajes α_j) con la secuencia de input h_j (values) estamos ponderando la secuencia de entrada para generar el vector de contexto.

$$\text{attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$e_{ij} = a(s_i, h_j)$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_k \exp(e_{ik})}$$

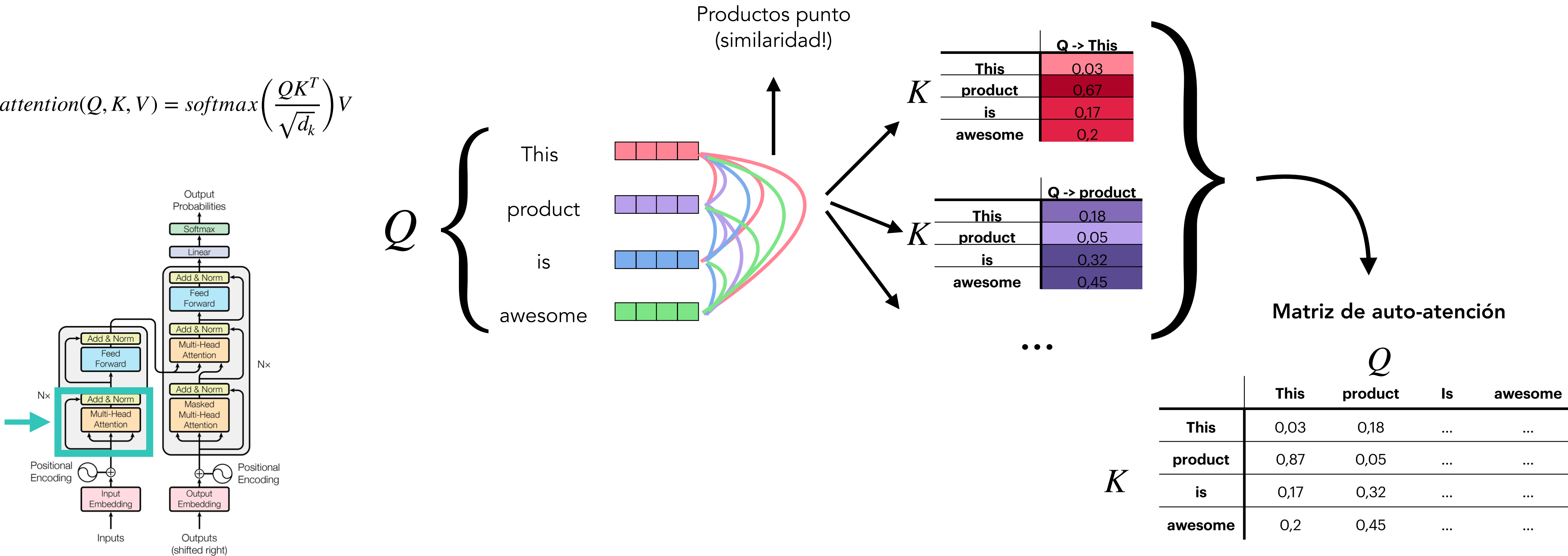
$$c = \sum_j \alpha_j h_j$$



2. Bloque de encoder - Sub-Bloque de atención

Entendiendo Q, K y V

Veamos un par de ejemplos gráficos, si estamos analizando una oración, es de nuestro interés determinar qué tokens (palabras) están más relacionados entre sí para determinar el significado de uno de estos tokens, por ejemplo la frase: “*This product is awesome*”

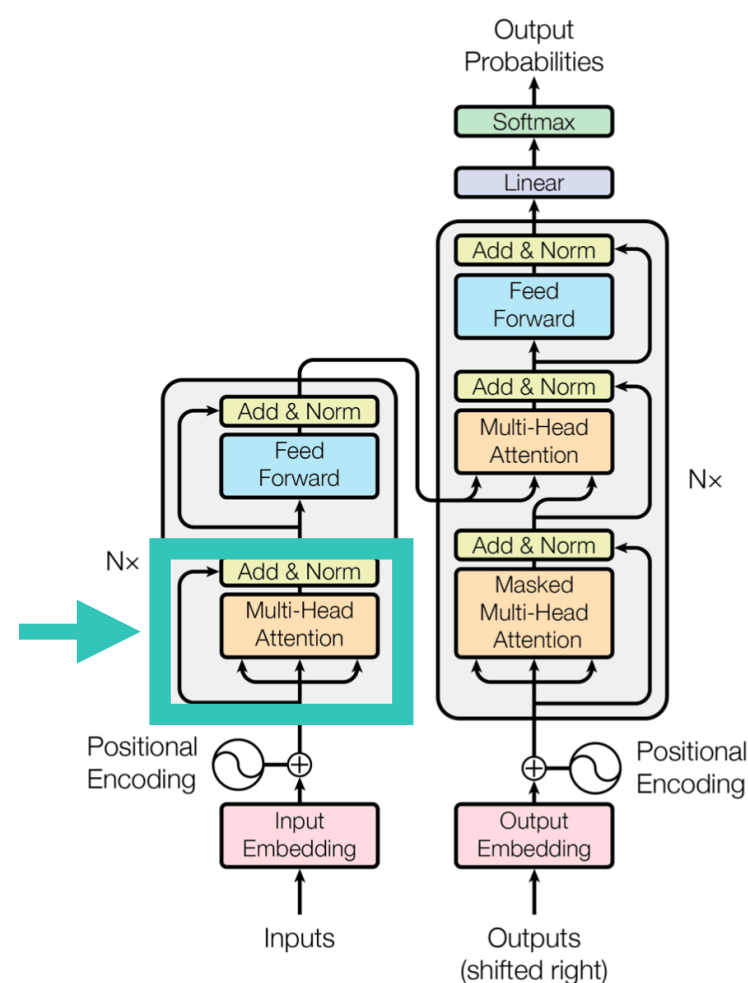


2. Bloque de encoder - Sub-Bloque de atención

Entendiendo Q, K y V

Mientras que un mecanismo de atención (por ejemplo, para NLP) puede ser usado para relacionar dos sentencias entre sí (las llaves con los valores), la autoatención es usada para relacionar diferentes posiciones de una misma sentencia de entrada entre sí.

Los mecanismos de atención pueden ser adaptados eligiendo (para el problema en cuestión) los valores de las llaves (secuencia objetivo) y las llaves (la misma secuencia de entrada).



Q

The
animal
didn't
cross
the
street
because
it
was
too
tired
.

K

The
animal
didn't
cross
the
street
because
it
was
too
wide
.

Por esto en muchos problemas donde se usan Transformers las llaves son iguales a los valores

2. Bloque de encoder - Sub-Bloque de atención

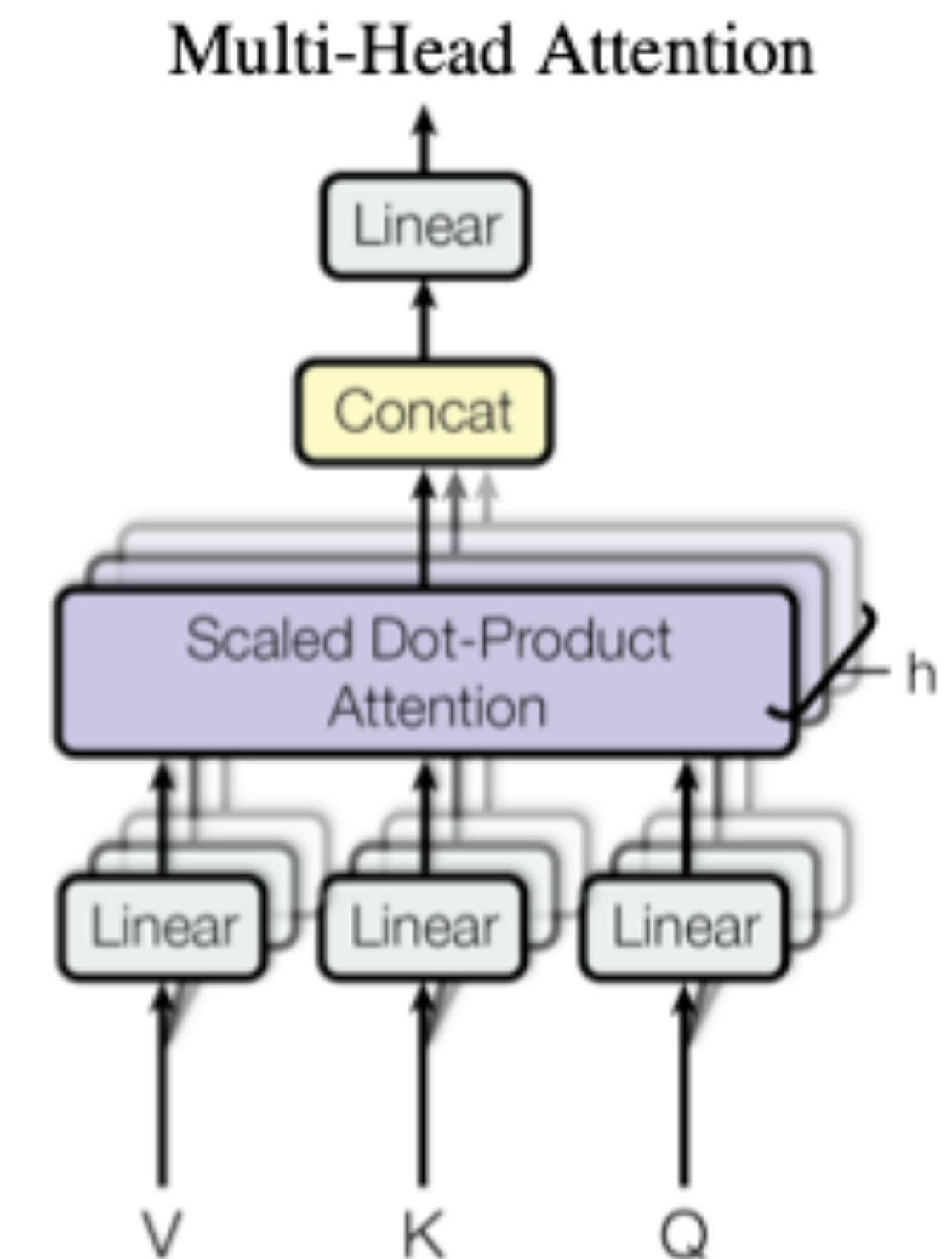
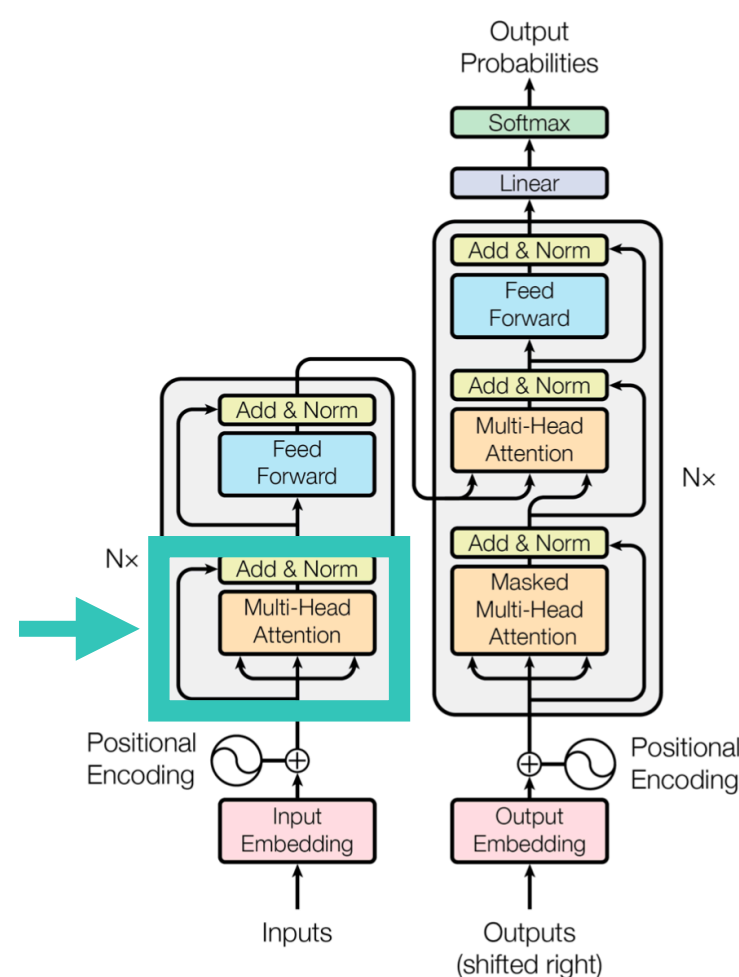
Multi-Head Attention

El bloque de atención implementado efectivamente por el Transformer usa el sub-bloque de atención ("producto punto") para conformar lo denominado Multi-Head Attention. Para esto se proyectan las llaves, valores y consultas h veces con diferentes proyecciones lineales: d_k , d_v y d_q

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O$$

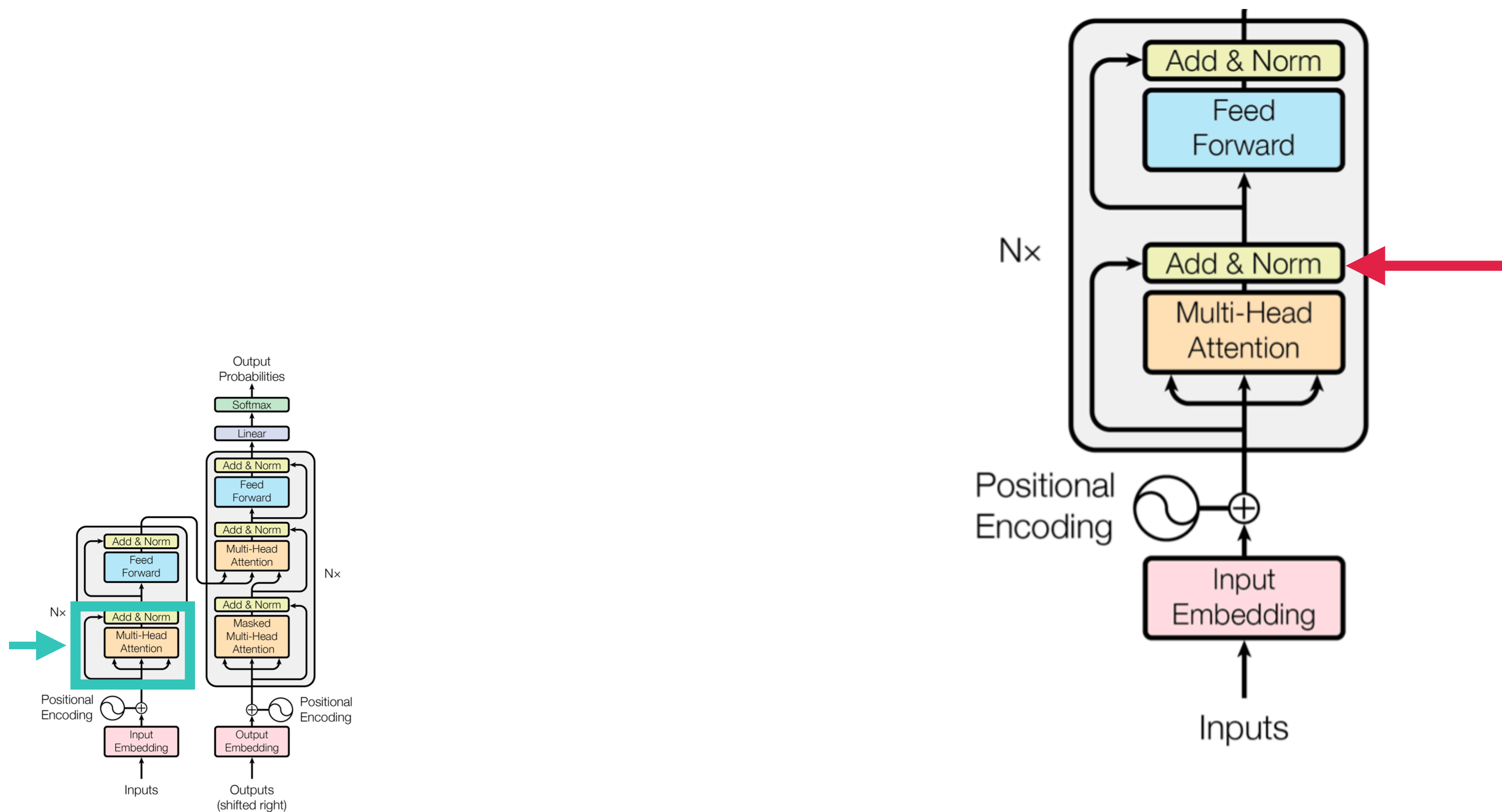
$$\text{Donde } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Además, $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ y $W^O \in \mathbb{R}^{hd_v \times d_{model}}$ son matrices de parámetros para las correspondientes proyecciones.



2. Bloque de encoder - Sub-Bloque de atención

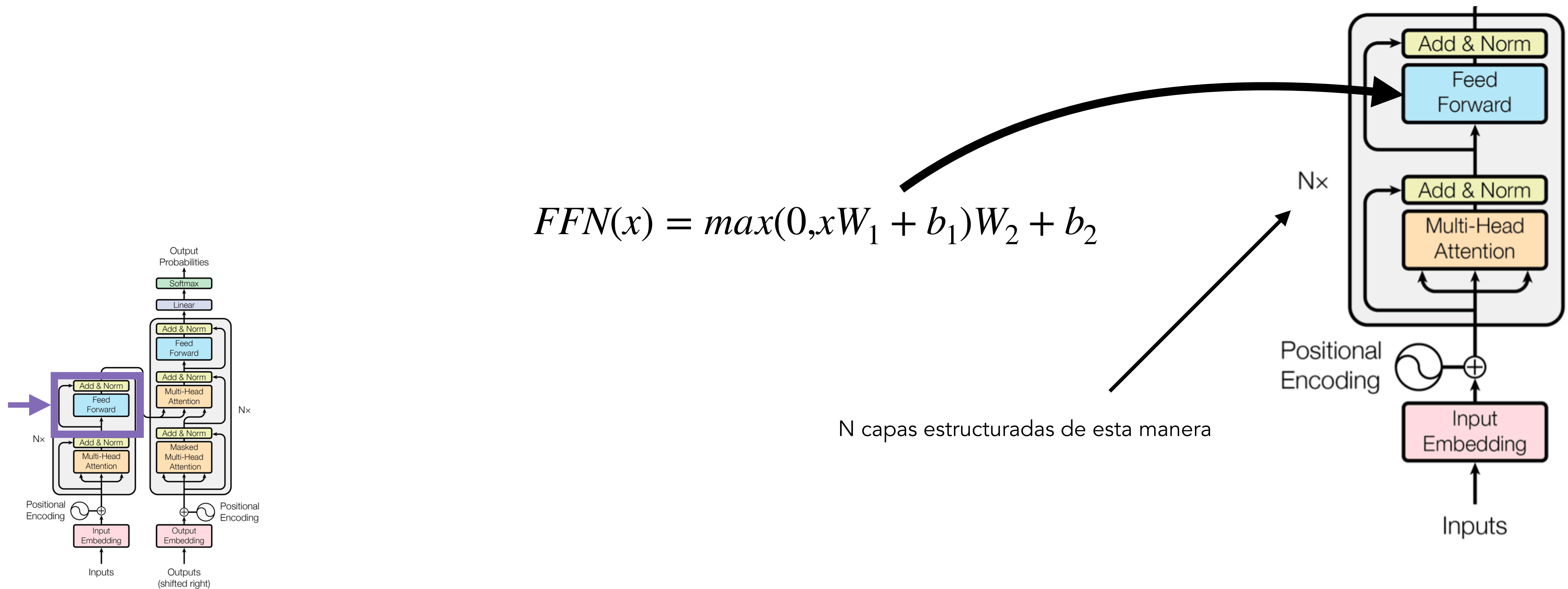
Una vez realizado el proceso de auto-atención en el encoder, se hace una suma y normalización (layer norm) incluyendo la conexión residual.



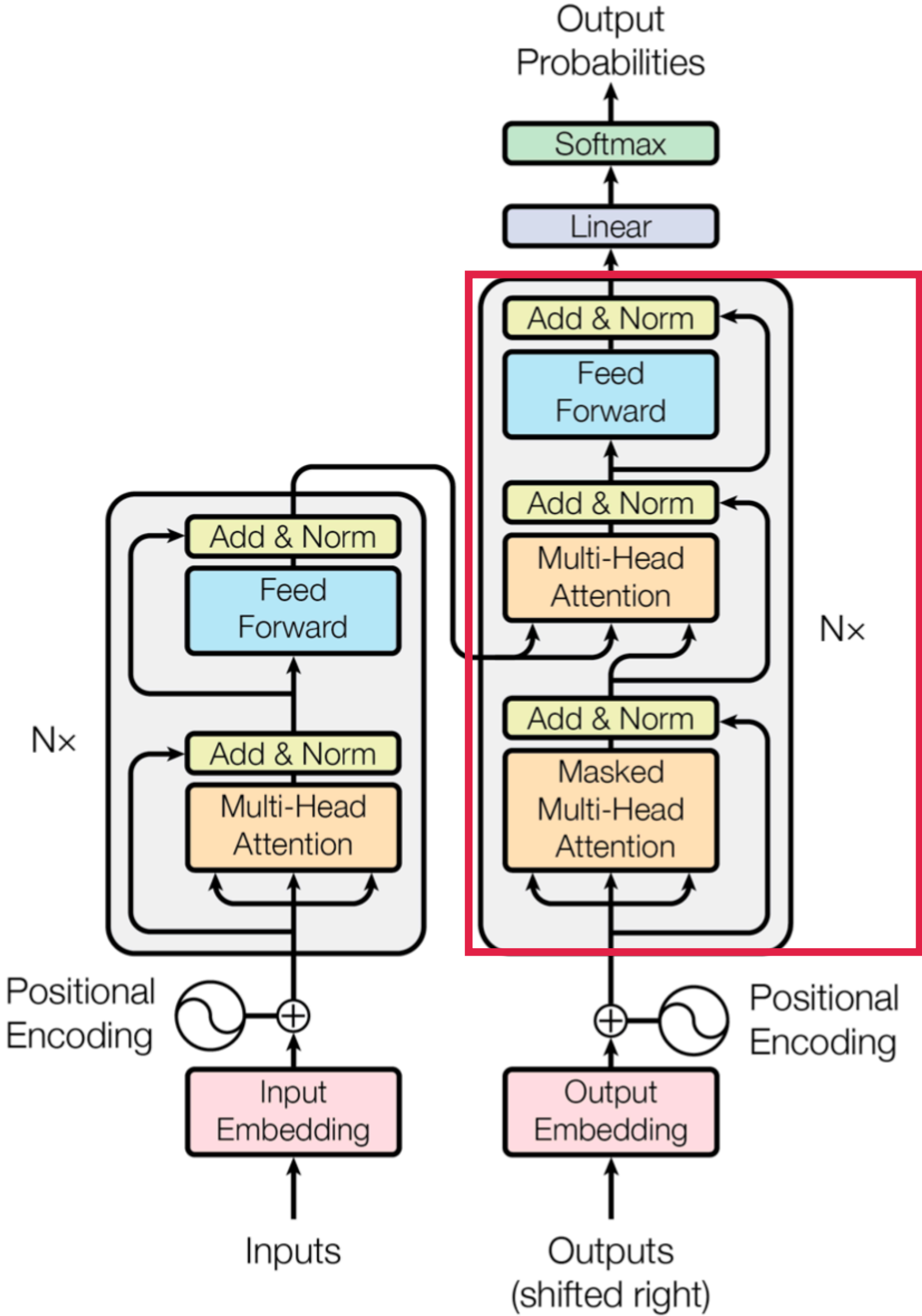
2. Bloque de encoder - Sub-Bloque de refinamiento

Position-wise Feed-Forward Networks

Los autores colocan al final del bloque del encoder una red neuronal densa (fully connected), la cual se aplica a cada posición de manera independiente, esta capa lineal es implementada usando una transformación lineal sobre una activación ReLU.

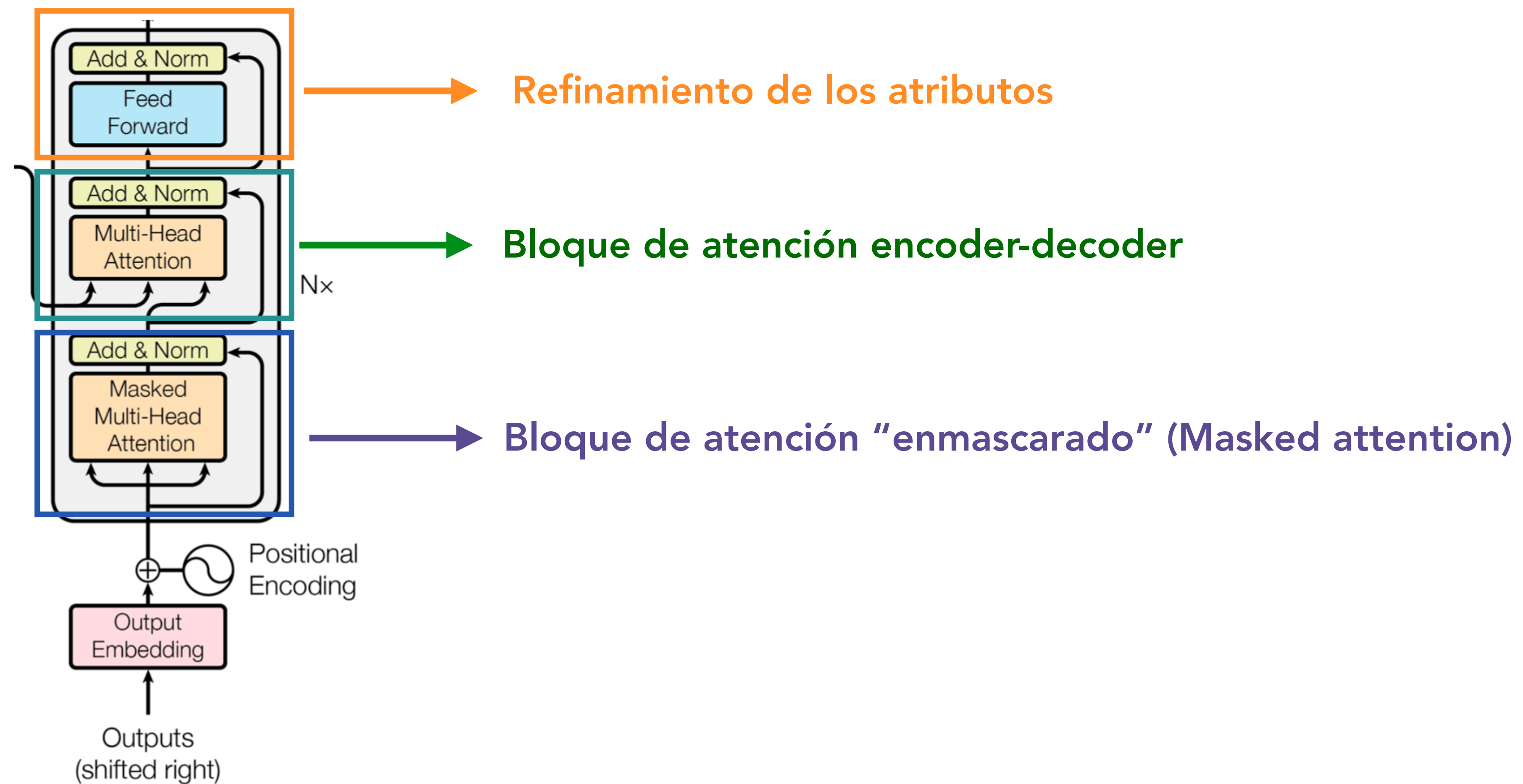


3. Bloque de decoder



3 **Bloque de Decoder**

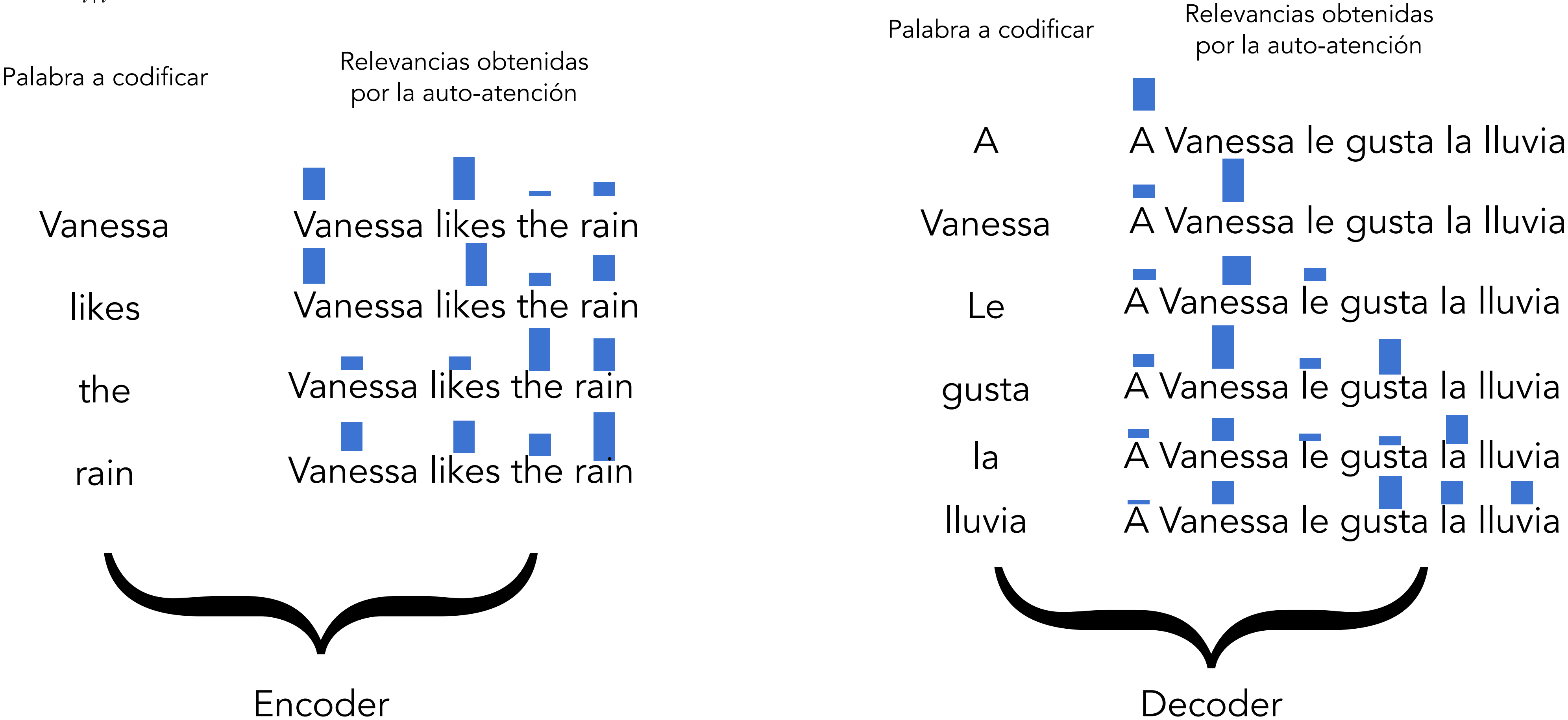
3. Bloque de decoder



3. Bloque de decoder - Masked Attention Block

Consideremos el problema de traducción: Dada una frase en un idioma I_1 , generar la frase análoga en otro idioma I_2 .

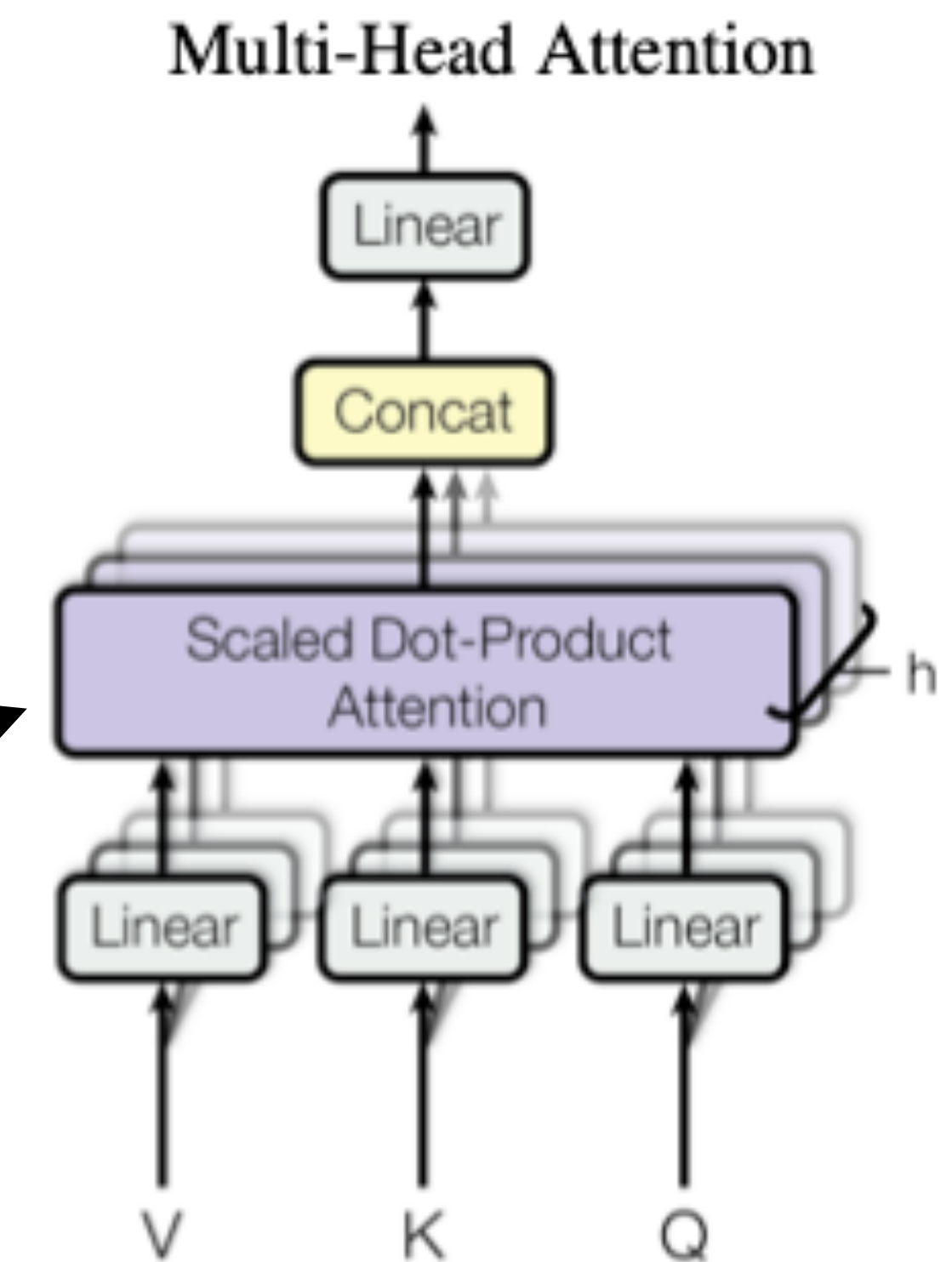
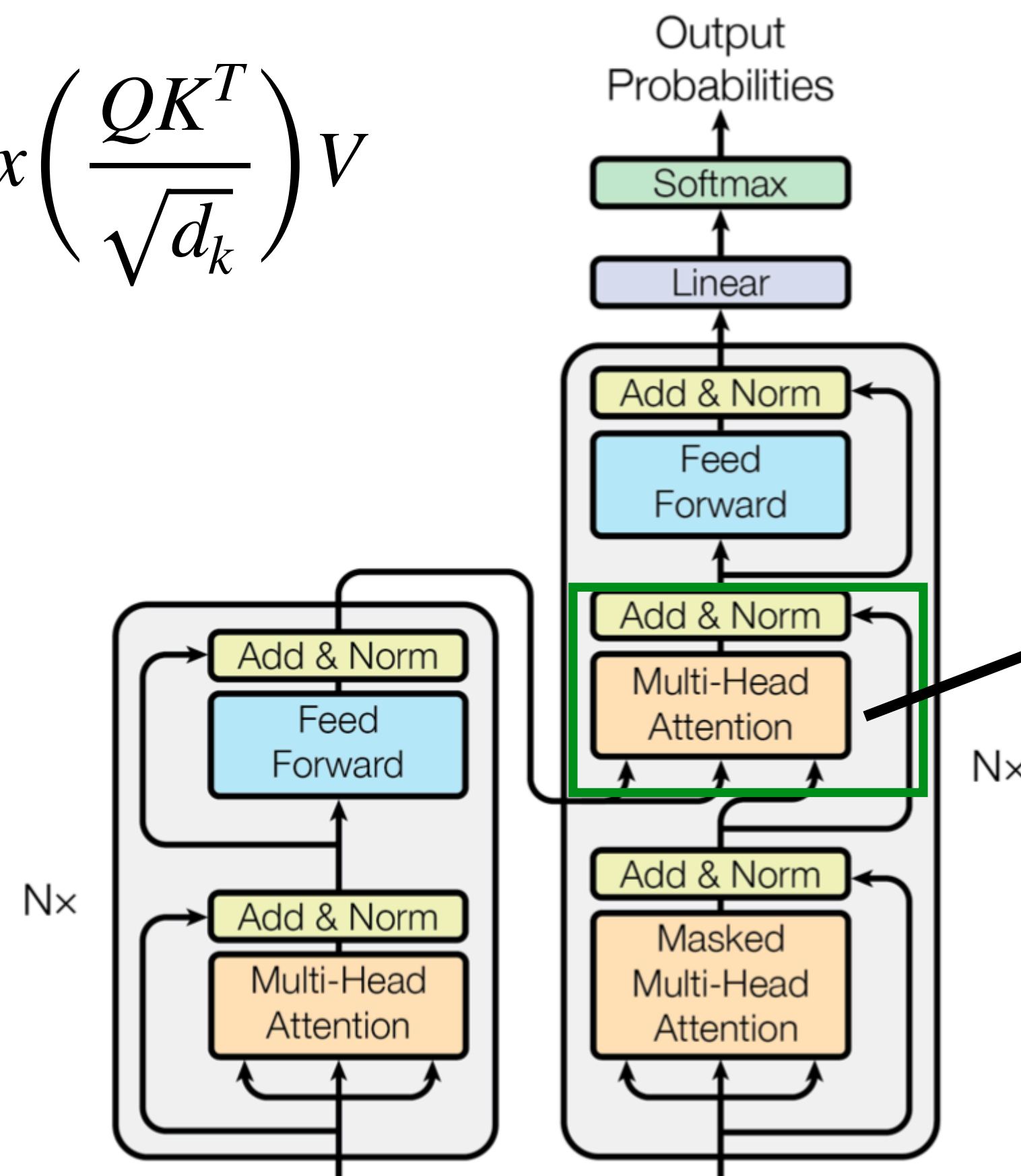
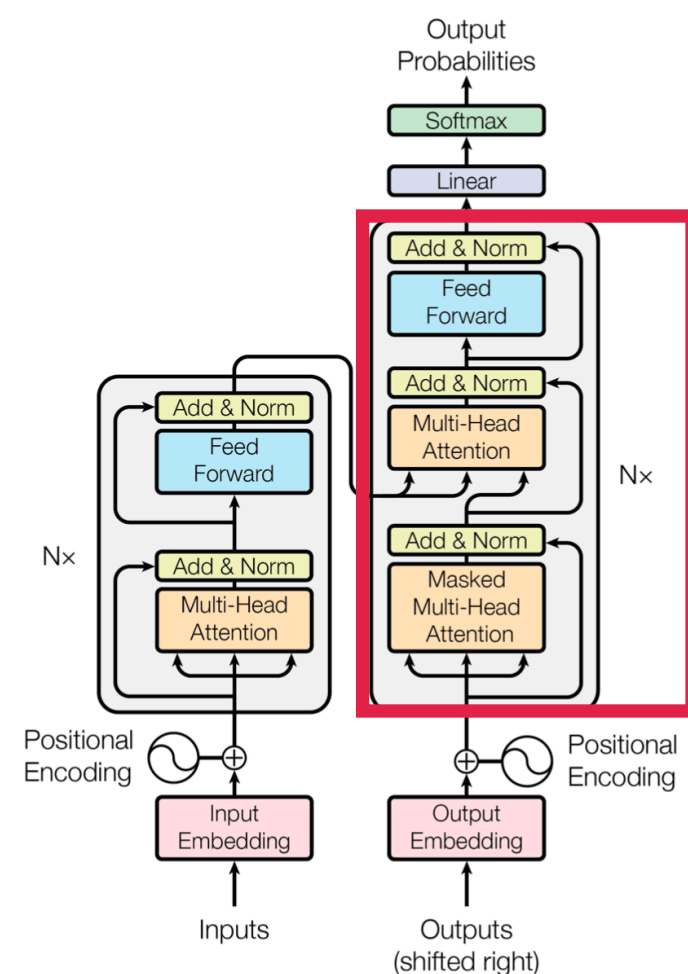
Notemos ahora lo siguiente, mientras que tenemos la frase de input (I_1) completa desde un principio, debemos ir generando la frase de salida una a una, dicho de otra manera, para generar la palabra W_t^2 (palabra $t + 1$ en el idioma 2) podemos usar cualquiera de las w_i^1 (palabras observadas del idioma 1) pero no deberíamos usar las palabras w_{t+i}^2 .



3. Bloque de decoder - Sub-Bloque de atención encoder-decoder

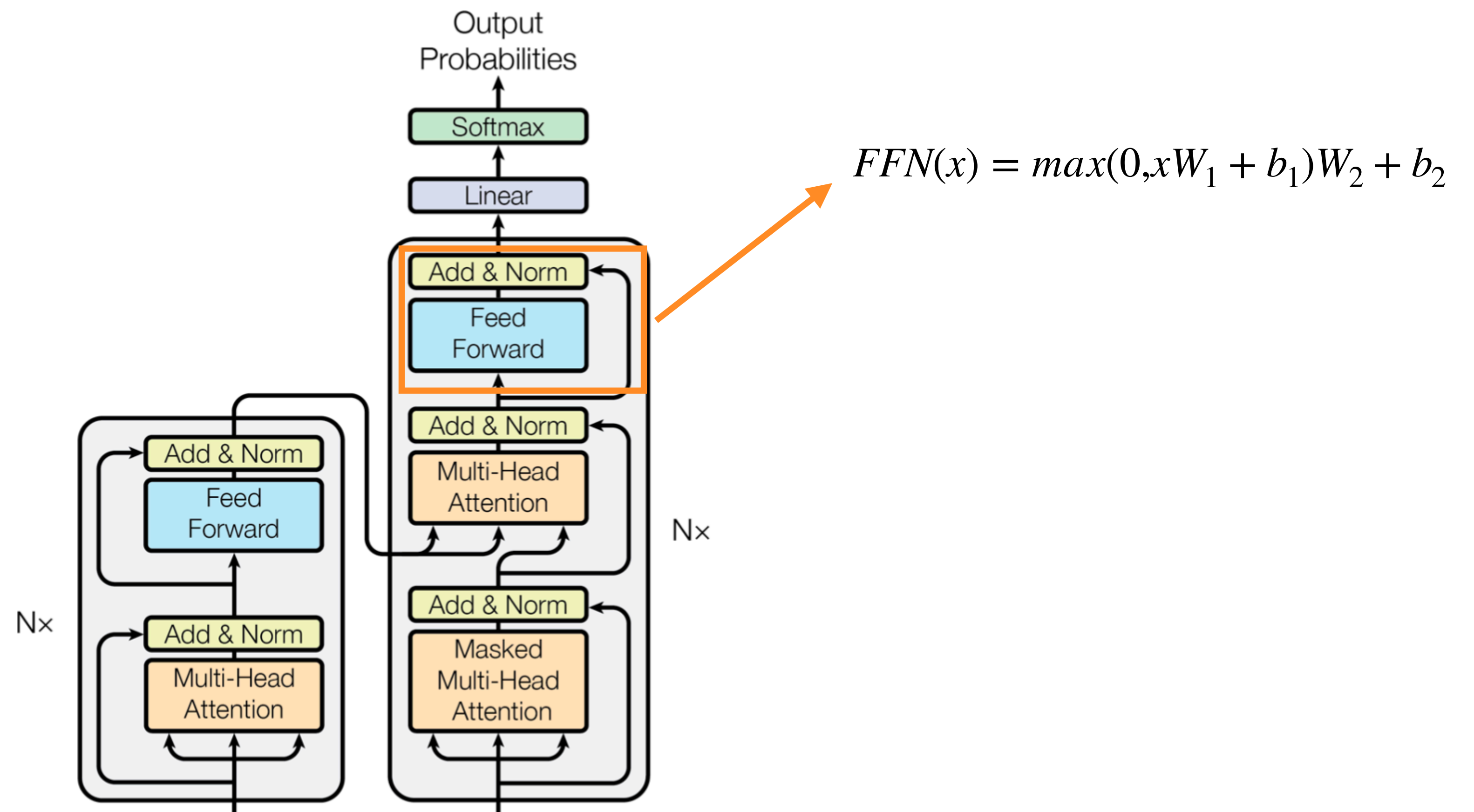
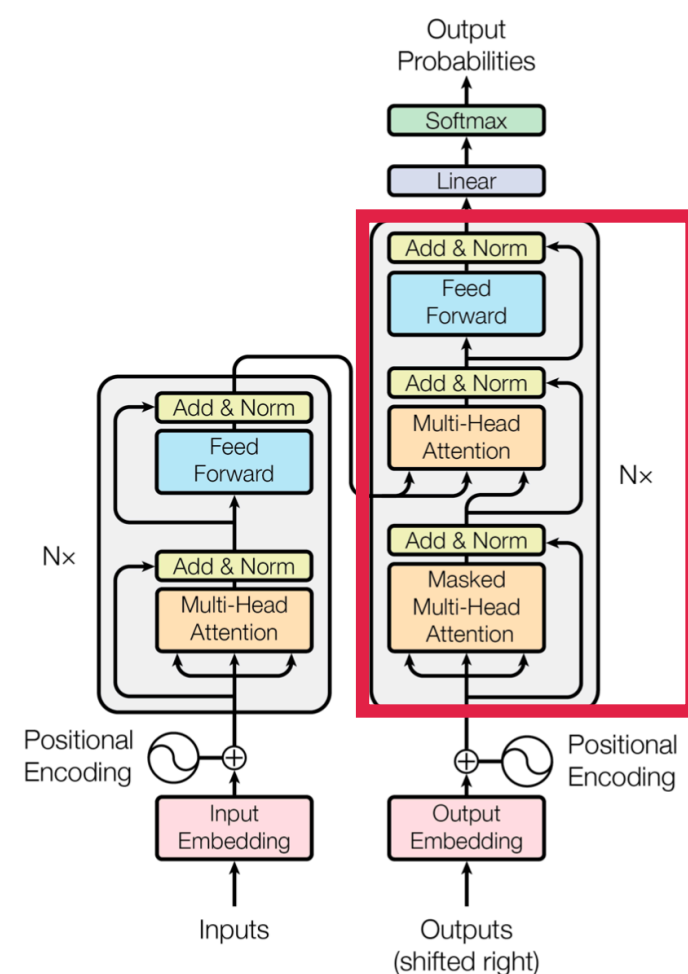
La tarea de este bloque de atención es generar relaciones entre los embeddings aprendidos en el bloque del encoder y los embeddings aprendidos en el sub-bloque de auto-atención del decoder.

$$attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



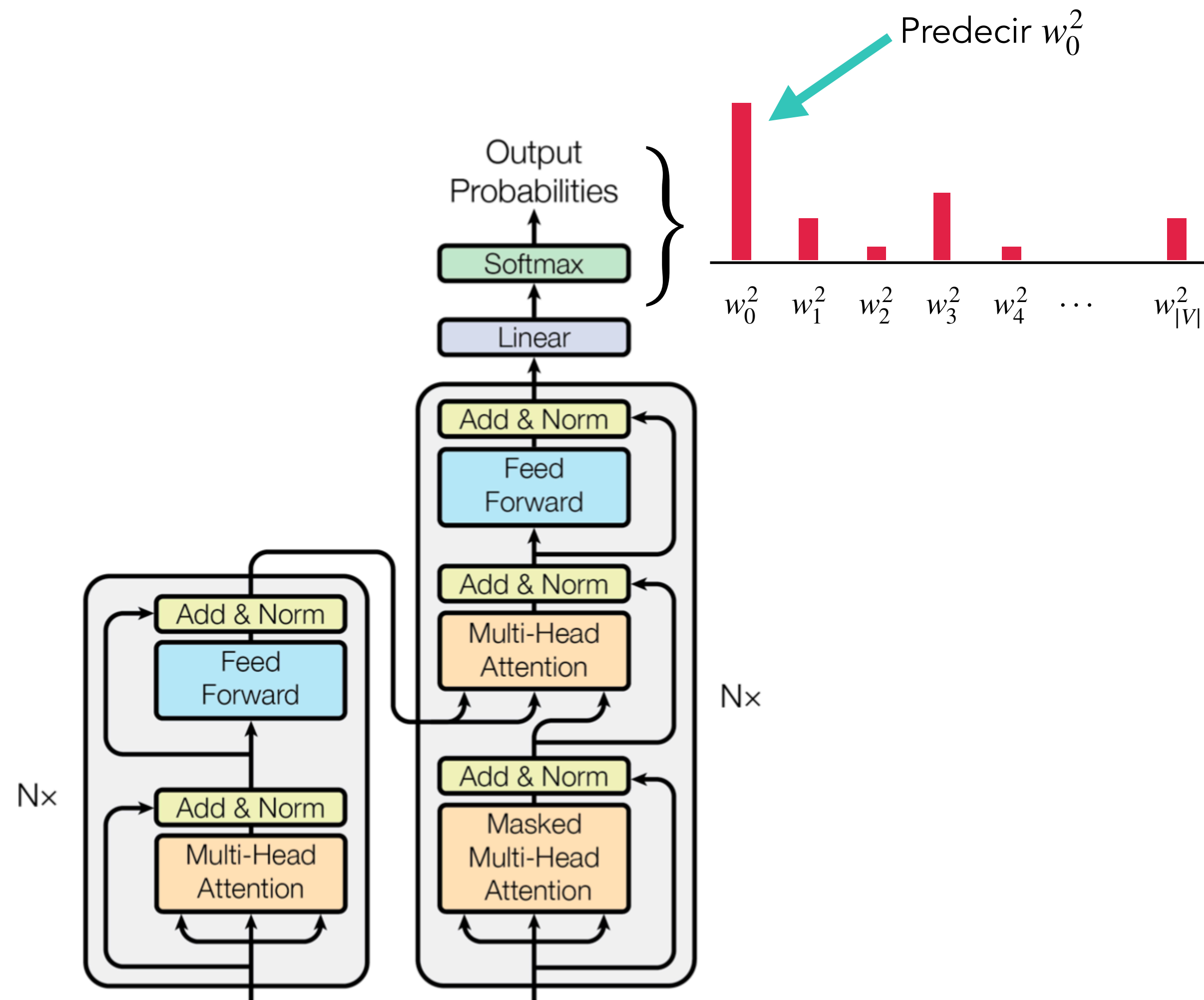
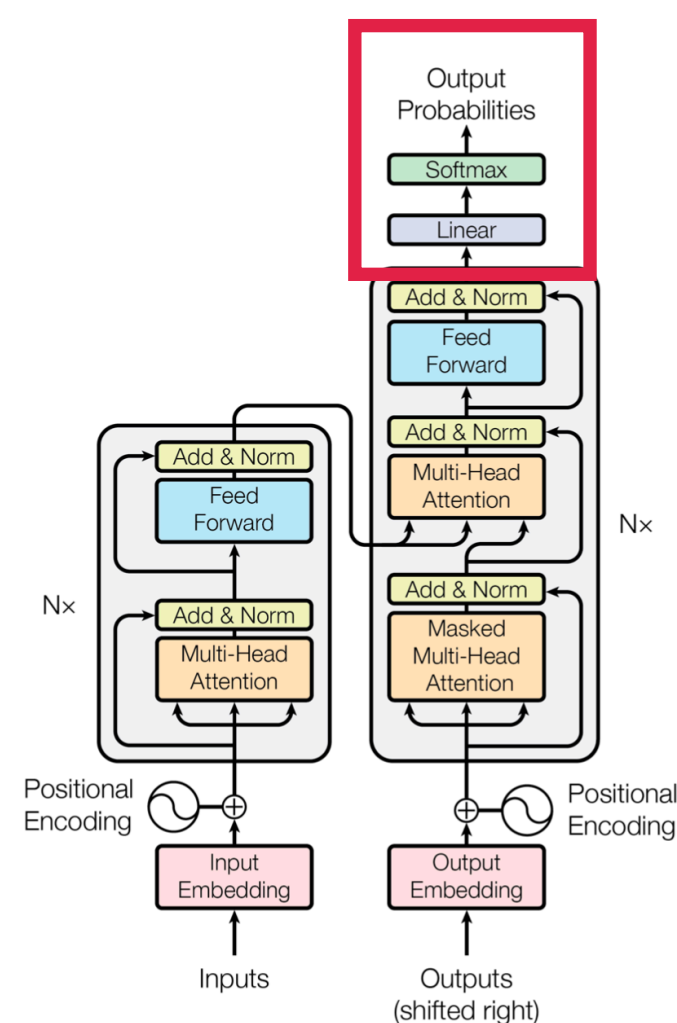
3. Bloque de decoder - Sub-Bloque de refinamiento

De la misma forma que para el bloque del decoder, esta sección aplica una transformación lineal sobre una activación ReLU, para permitir refinar atributos antes de salir del decoder. Finalmente se adiciona el tensor proveniente del bloque de atención encoder-decoder directamente y se aplica una normalización por capa.



Salida del modelo

La parte final del modelo corresponde a una capa lineal densa donde, por ejemplo para el problema de traducción, tendría una dimensionalidad $d_{out} = |V|$, donde $|V|$ es el tamaño del vocabulario de salida, aplicando finalmente una softmax para determinar la palabra adecuada a generar.



En resumen

Transformer Model

Innovaciones:

- Sortea el problema de las dependencias de largo plazo mediante la ingestión de toda la frase de una sola vez, tiñendo el modelo el contexto completo de cada palabra/token en todo momento y, por lo tanto, aliviando a la arquitectura de tener que “recordar” a través del tiempo.
- Mejora la capacidad de entrenamiento en paralelo al no necesitar ingestar el input de manera secuencial (y por lo tanto ya no presenta el cuello de botella al momento de calcular el siguiente estado interno h_{t+1}), lo cual hace al modelo más “paralelizable” durante su entrenamiento.
- Implementa un tipo de mecanismo de atención (Dot-Product Attention con el factor de escala $1/\sqrt{d_k}$) que se desempeña mejor y más rápido que la atención aditiva calculada mediante una FFN.
- Sigue una arquitectura encoder-decoder, donde cada una de estas dos partes tiene su correspondiente sub-bloque de atención Multi-cabezal.