Transformers

Métodos Generativos, curso 2025-2026

Guillermo Iglesias, guillermo.iglesias@upm.es
Jorge Dueñas Lerín, jorge.duenas.lerin@upm.es
Edgar Talavera Muñoz, e.talavera@upm.es

5 de noviembre de 2025

Escuela Técnica Superior de Ingeniería de Sistemas Informáticos | UPM

Contenidos

Contenidos

Transformers

Motivación

Hasta la aparición de los transformers, los problemas secuenciales se trataban (casi) siempre con **redes recurrentes**.

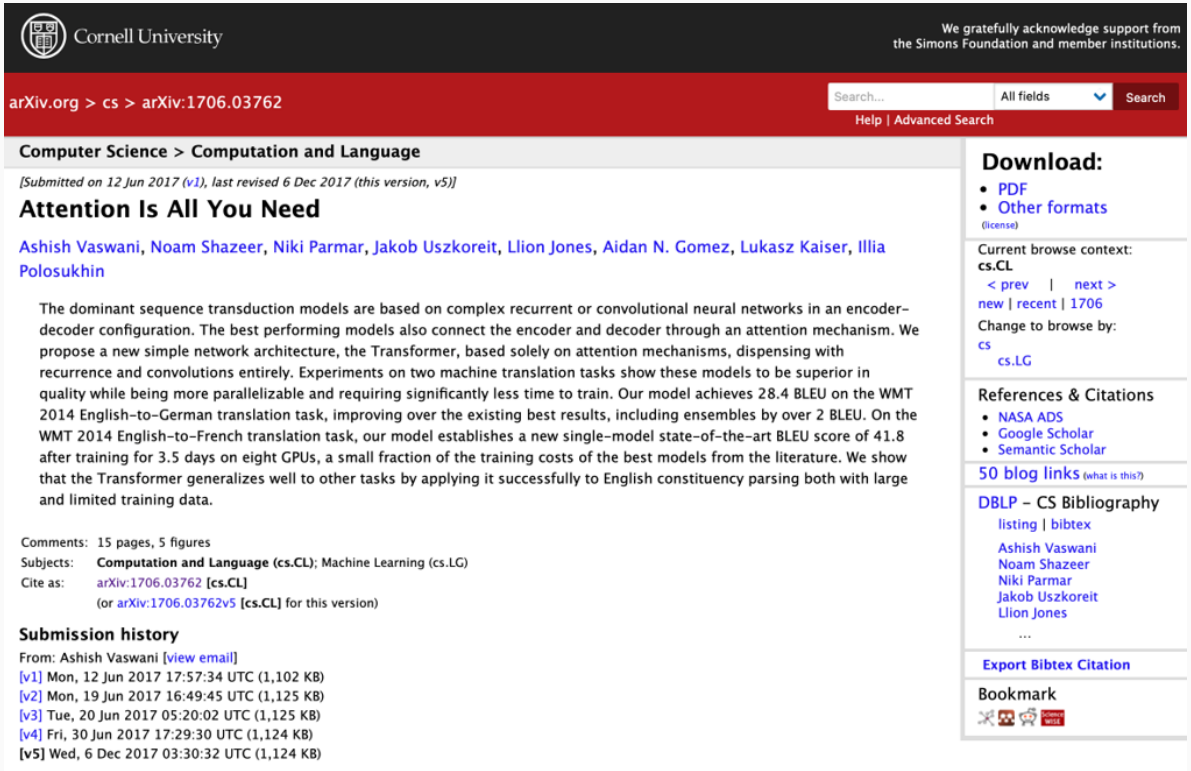
Sin embargo, dichas arquitecturas tienen dos problemas bien conocidos:

- Su entrenamiento no se puede paralelizar, ya que un instante depende del anterior, y así sucesivamente, por lo que su entrenamiento es **lento**.
- Aunque aparecieron para “recordar”, su memoria es bastante limitada, empeorando sus resultados cuanto más larga es la secuencia con la que tienen que trabajar.

¿Qué son los Transformers?

- Los **transformers** son un tipo de arquitectura de redes neuronales revolucionaria en el campo de la inteligencia artificial.
- A mediados de 2017 aparece un artículo científico que demuestra que existe una alternativa a las redes recurrentes para trabajar con **problemas secuenciales**.
- En concreto, habla del concepto de "**atención**", que básicamente significa "**ponderar**" los elementos de las secuencias según afecten más o menos a las predicciones.
- Habla sobre todo de NLP (Natural Language Processing), pero también es aplicable a series temporales (y a imágenes con ligeras modificaciones).
- Elimina el concepto de **recursión**, con lo que el entrenamiento puede ser mucho más **rápido**.
- Además, es capaz de "**memorizar**" secuencias más largas.

¿Qué son los Transformers?



Cornell University

We gratefully acknowledge support from the Simons Foundation and member institutions.

arXiv.org > cs > arXiv:1706.03762

Search... All fields Search

Help | Advanced Search

Computer Science > Computation and Language

[Submitted on 12 Jun 2017 (v1), last revised 6 Dec 2017 (this version, v5)]

Attention Is All You Need

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder-decoder configuration. 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 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Comments: 15 pages, 5 figures
Subjects: **Computation and Language (cs.CL)**; Machine Learning (cs.LG)
Cite as: arXiv:1706.03762 **[cs.CL]**
(or arXiv:1706.03762v5 **[cs.CL]** for this version)

Submission history

From: Ashish Vaswani [view email]
[v1] Mon, 12 Jun 2017 17:57:34 UTC (1,102 KB)
[v2] Mon, 19 Jun 2017 16:49:45 UTC (1,125 KB)
[v3] Tue, 20 Jun 2017 05:20:02 UTC (1,125 KB)
[v4] Fri, 30 Jun 2017 17:29:30 UTC (1,124 KB)
[v5] Wed, 6 Dec 2017 03:30:32 UTC (1,124 KB)

Download:
- PDF
- Other formats (license)

Current browse context: cs.CL
< prev | next >
new | recent | 1706
Change to browse by:
cs
cs.LG

References & Citations
- NASA ADS
- Google Scholar
- Semantic Scholar

50 blog links (what is this?)

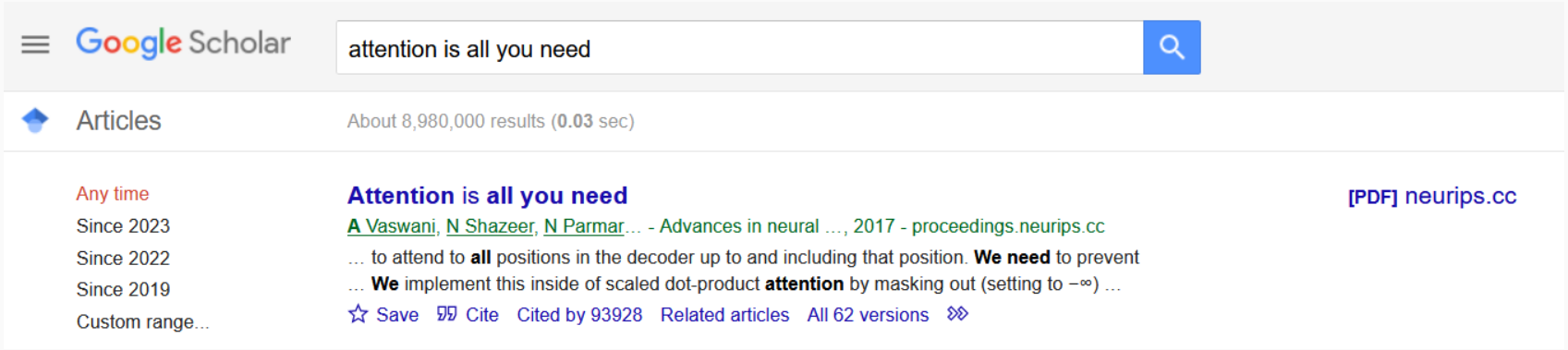
DBLP - CS Bibliography
listing | bibtex
Ashish Vaswani
Noam Shazeer
Niki Parmar
Jakob Uszkoreit
Llion Jones
...

Export Bibtex Citation

Bookmark

Captura de arxiv.org del artículo “Attention is all you need”.

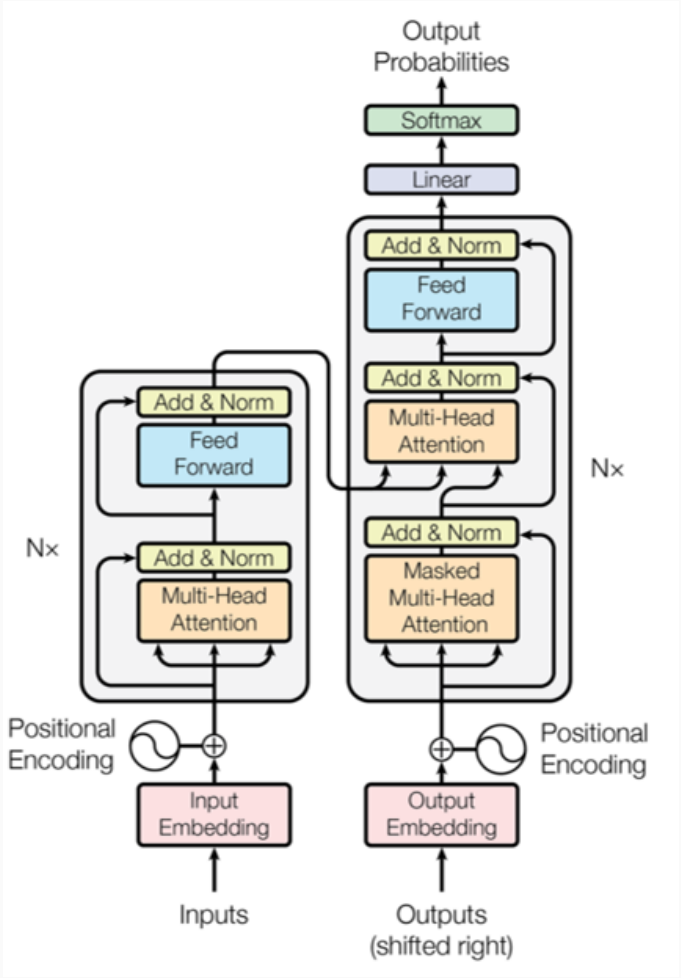
¿Qué son los Transformers?



Captura de Google Scholar mostrando el número de citas a fecha Octubre de 2023.

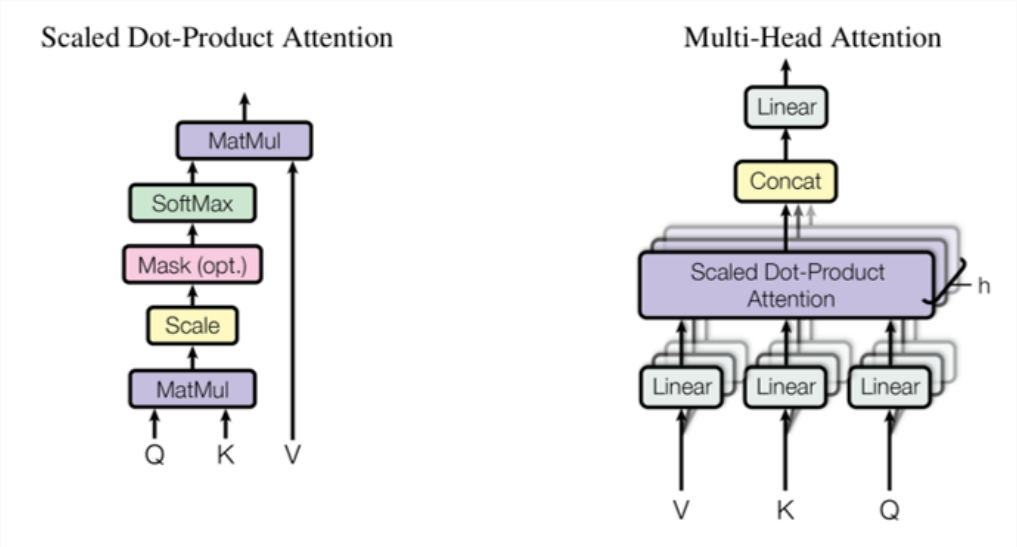
Arquitectura de los Transformers

- Una arquitectura encoder-decoder con atención.
- La atención “mira” a la secuencia de entrada y decide para cada instante qué otras partes de la misma son importantes (podéis entenderlo como el “contexto”).
- Consiste en bloques encoder y decoder apilables (*Nx* bloques)
- Capas densas y bloques Multi-Head Attention (MHA).
- Positional embbeding (para introducir la información “temporal” en el caso de las series temporales o “situacional” en el caso de NLP)



Arquitectura de los Transformers

- El bloque **Multi-Head Attention** es el que permite al modelo "fijarse" en las cosas más importantes.
- Funciona creando varios **mapas de activación** (Q: query, K: key, V: value) y comprobando sus **relaciones**.



Principales mecanismos de la arquitectura transformer.

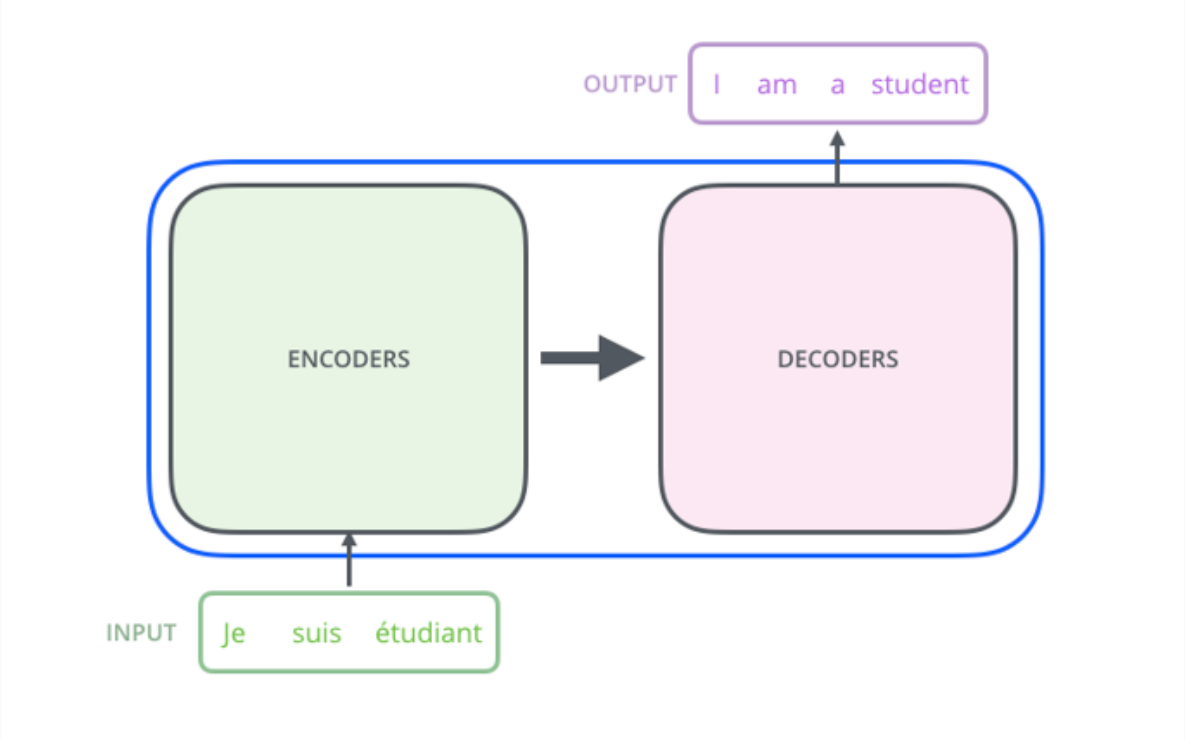
Arquitectura de los Transformers

- Vamos a ver la arquitectura transformer explicada en el campo del procesamiento del lenguaje natural, ya que es el caso de uso más típico.
- Para el caso de series temporales o imágenes es muy similar, lo único que cambia es la codificación de los datos de entrada.

Ejemplo simplificado de una tarea de traducción con un transformer (fuente).

Arquitectura de los Transformers

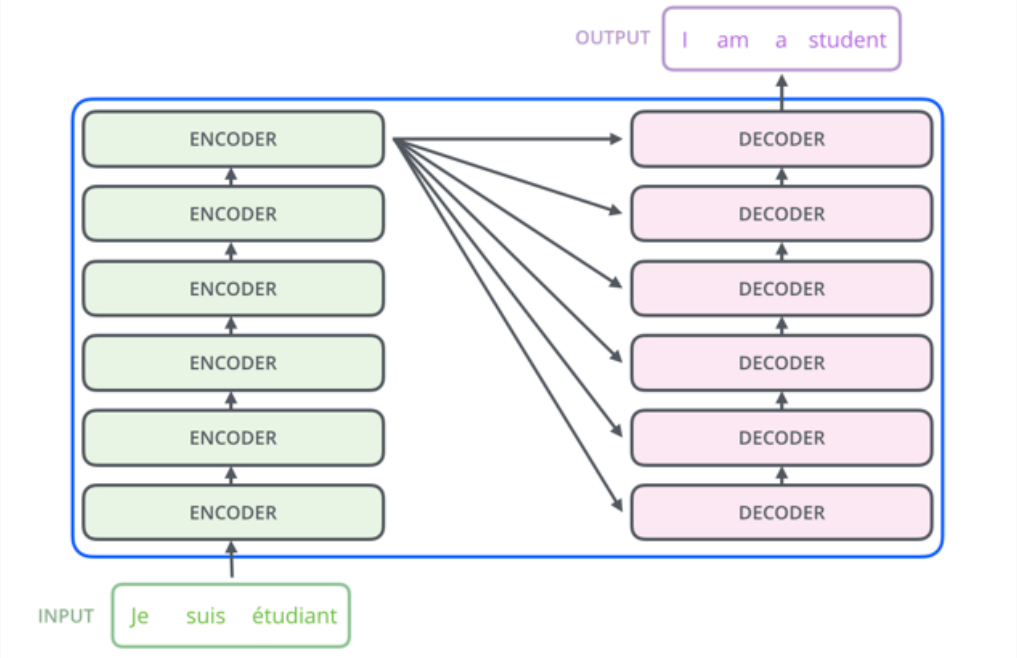
- Los transformers se componen de un **encoder** y un **decoder**.



Arquitectura encoder-decoder (fuente).

Arquitectura de los Transformers

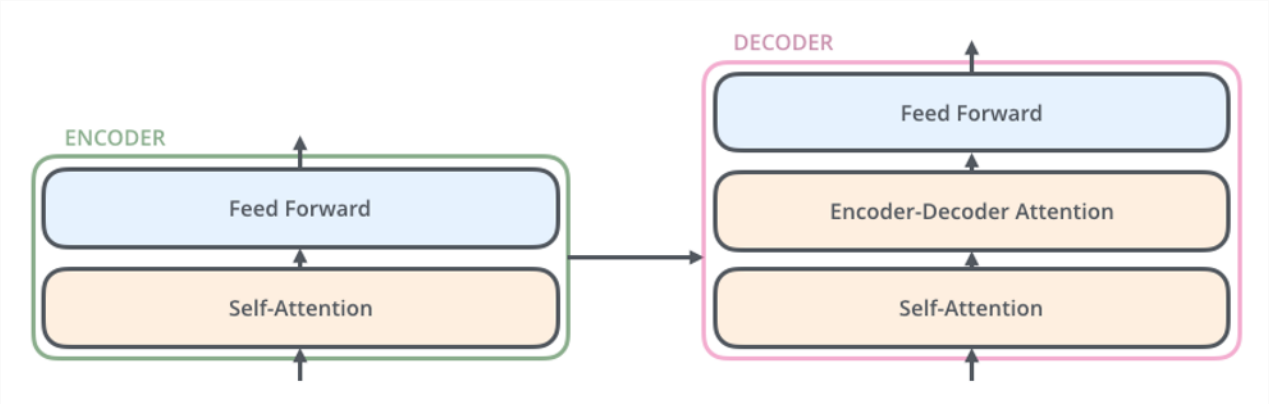
- O para ser más exactos, de *N* bloques de encoders apilados seguidos de *N* bloques de decoders apilados.



Arquitectura general del transformer (fuente).

Arquitectura de los Transformers

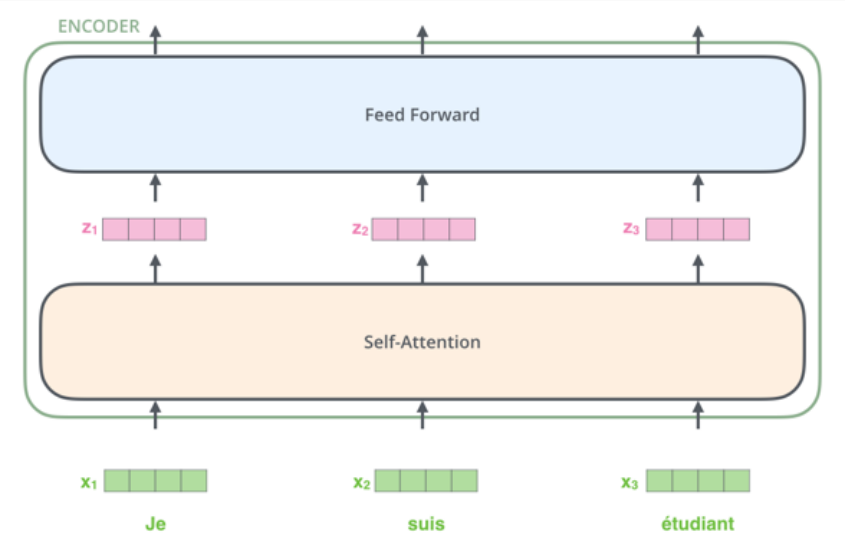
- Dentro de cada **encoder** tendremos un bloque de capas **feed-forward** (densas) y otro bloque de **Self-Attention**, el cual ayuda al encoder a fijarse no solo en la palabra que está procesando en ese momento, sino también en las demás palabras de la secuencia de entrada.
- En los **decoders**, tendremos además de eso, un **bloque de Encoder-Decoder attention** que le permita decirle qué partes de la secuencia de entrada son **importantes**.



Arquitectura general del encoder y el decoder (fuente).

Arquitectura de los Transformers

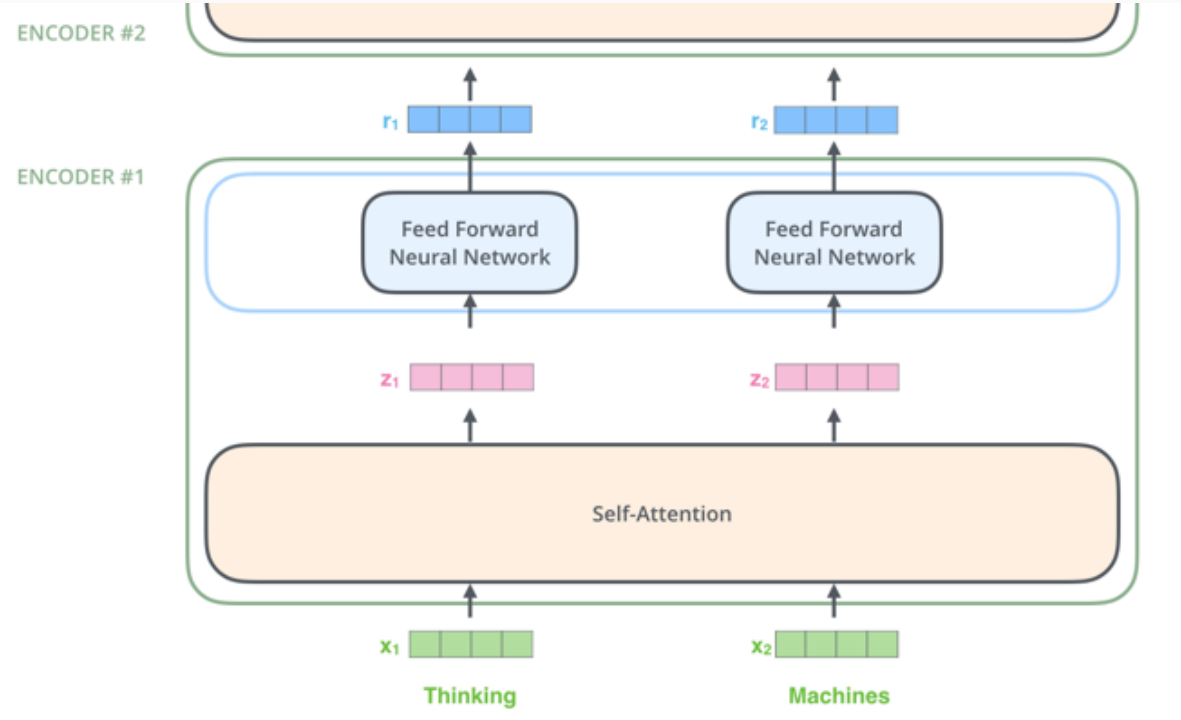
- Los vectores de entrada *fluyen* a través de los bloques cada uno en una posición fija.
- El encoder crea dependencias entre estos “caminos” en los bloques de **self-attention**.
- ¡Esto se puede **paralelizar**, ya no existen dependencias temporales!



Arquitectura de los bloques de tipo **encoder** en detalle (fuente).

Arquitectura de los Transformers

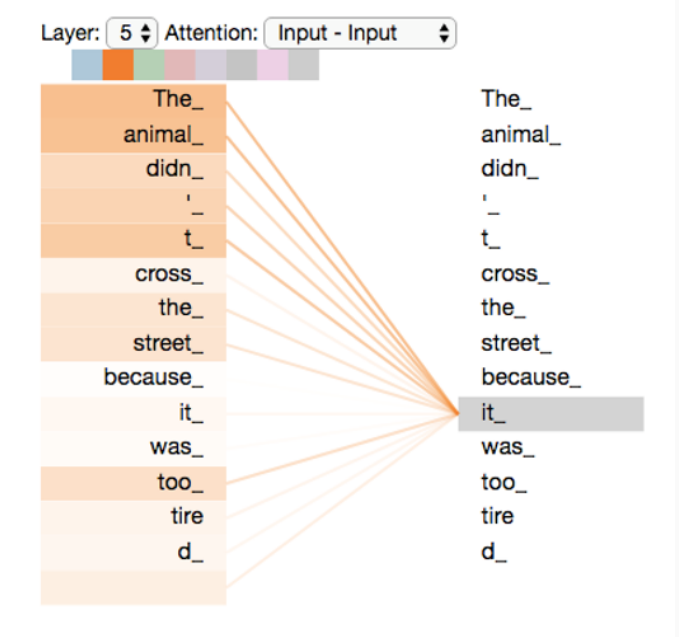
- Este proceso se repite para cada bloque “encoder”.



Arquitectura del **encoder** en detalle (fuente).

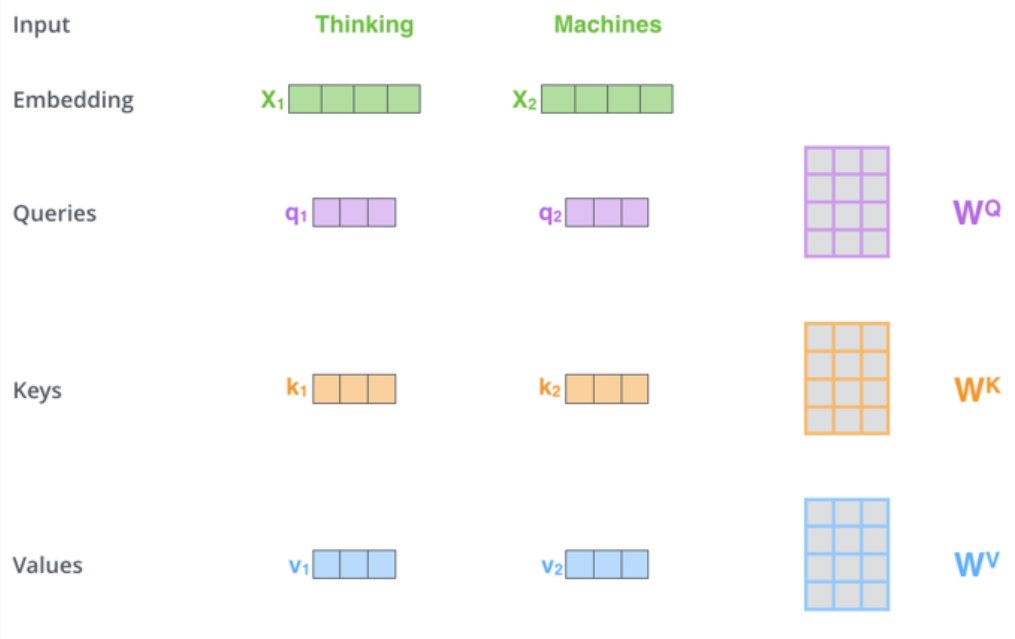
¿Cómo funciona la atención?

- En la frase “The **animal** didn’t cross the street because it was too tired”, el modelo no sabe a priori que “it” se refiere a “The animal” -> ¡Esto es lo que hace el bloque de **self-attention**!
- Conforme el modelo procesa **cada palabra** (o cada posición en la secuencia de entrada), el módulo de self-attention le permite **mirar** al **resto de las posiciones** en la secuencia de entrada para pistas que le permitan **codificar mejor** el elemento **actual**.



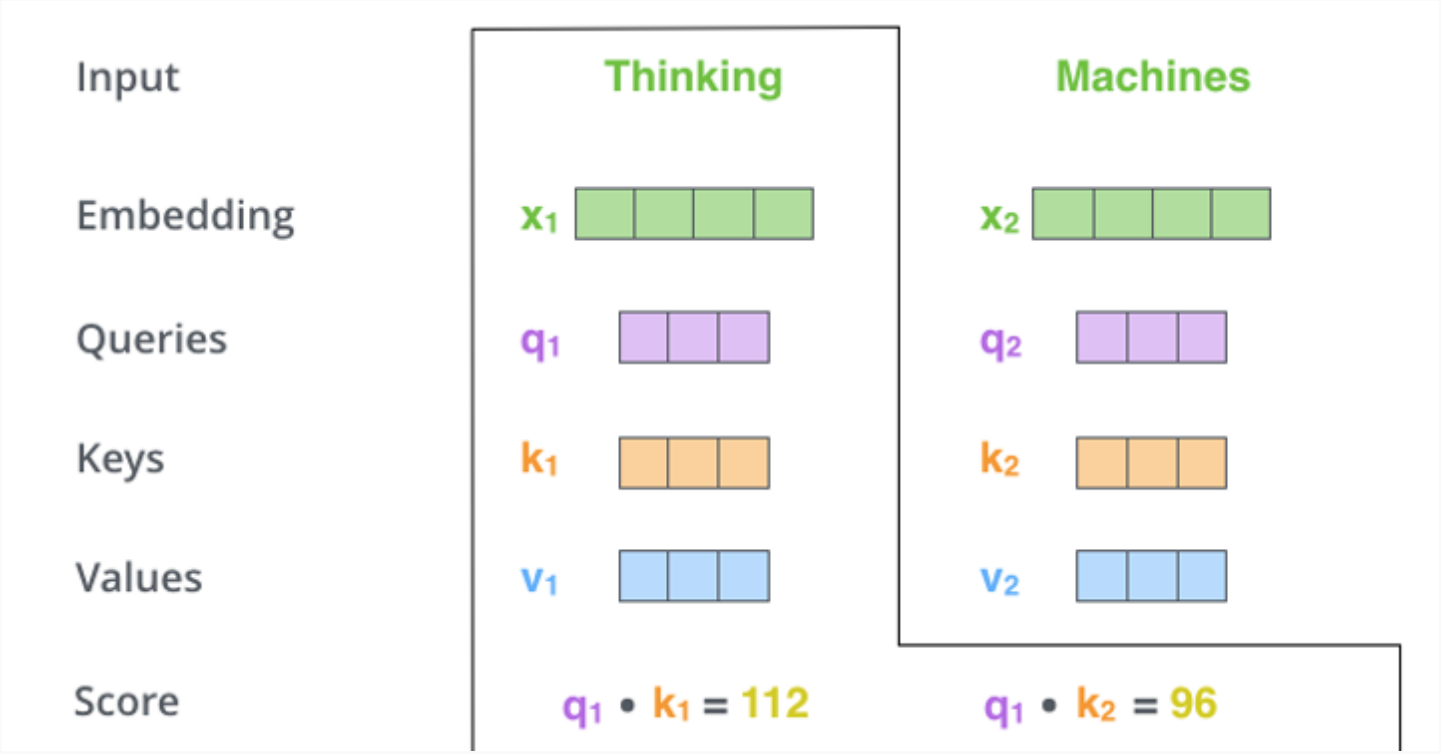
¿Cómo funciona el bloque de self-attention?

1. Creamos 3 matrices entrenables (W^Q, W^K, W^V)
2. Multiplicamos cada vector de entrada por cada una de las 3 matrices entrenables (W^Q, W^K, W^V)
3. Obtenemos 3 nuevos vectores (q_n, k_n, v_n), que además son de menor dimensionalidad que los de entrada, ya que las matrices se eligen así para reducir el coste computacional



¿Cómo funciona el bloque de self-attention?

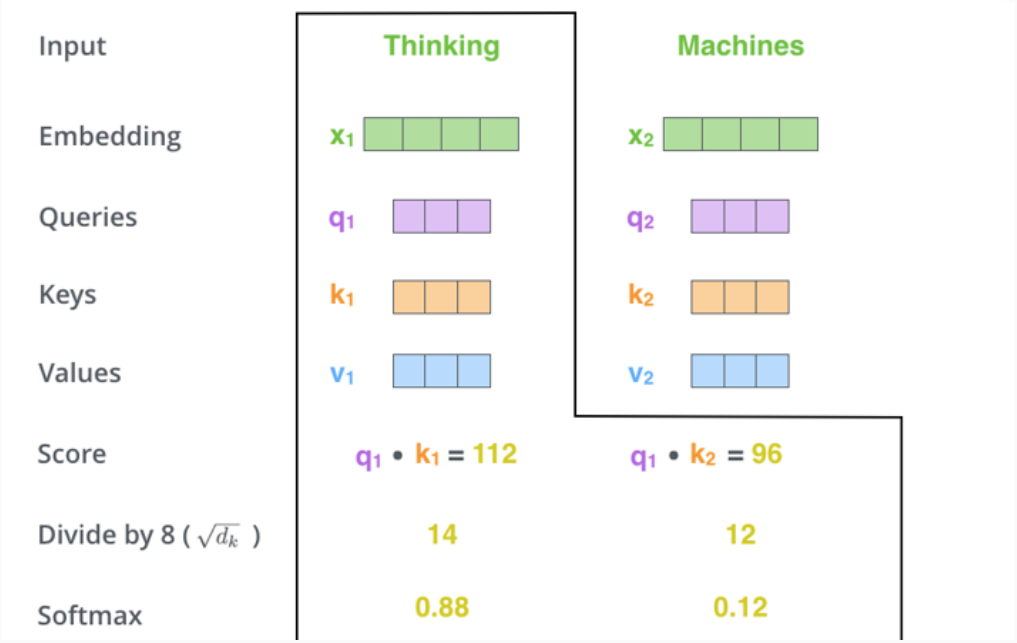
4. Calculamos la self-attention como el producto escalar de $q_n \cdot k_n$.



Ejemplo del producto escalar en el mecanismo de self-attention (fuente).

¿Cómo funciona el bloque de self-attention?

5. Dividimos entre la raiz cuadrada del número de dimensiones escogidas para los vectores **key** (para estabilizar los gradientes durante el entrenamiento)
6. Aplicamos la funcion **softmax** para normalizar los valores y que sumen 1.

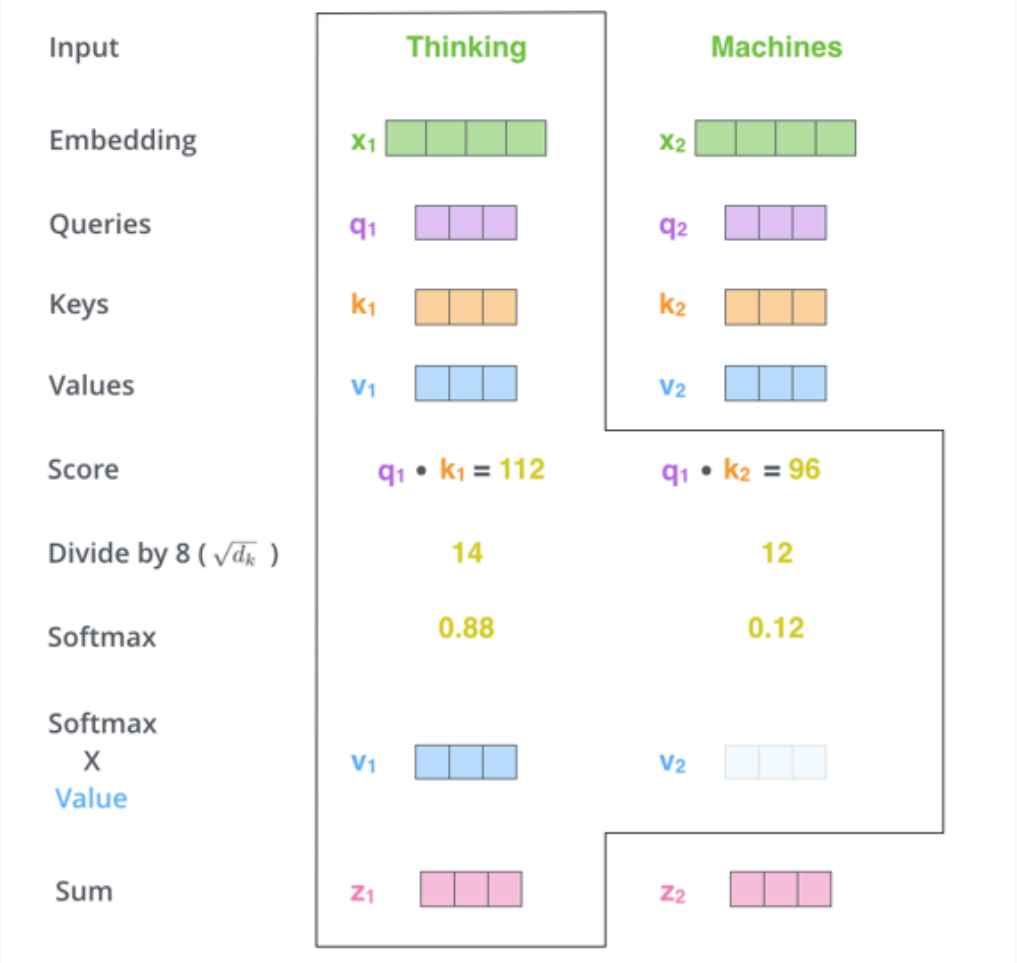


¡Ya tenemos el valor de la atención!
Pero no hemos acabado...

¿Cómo funciona el bloque de self-attention?

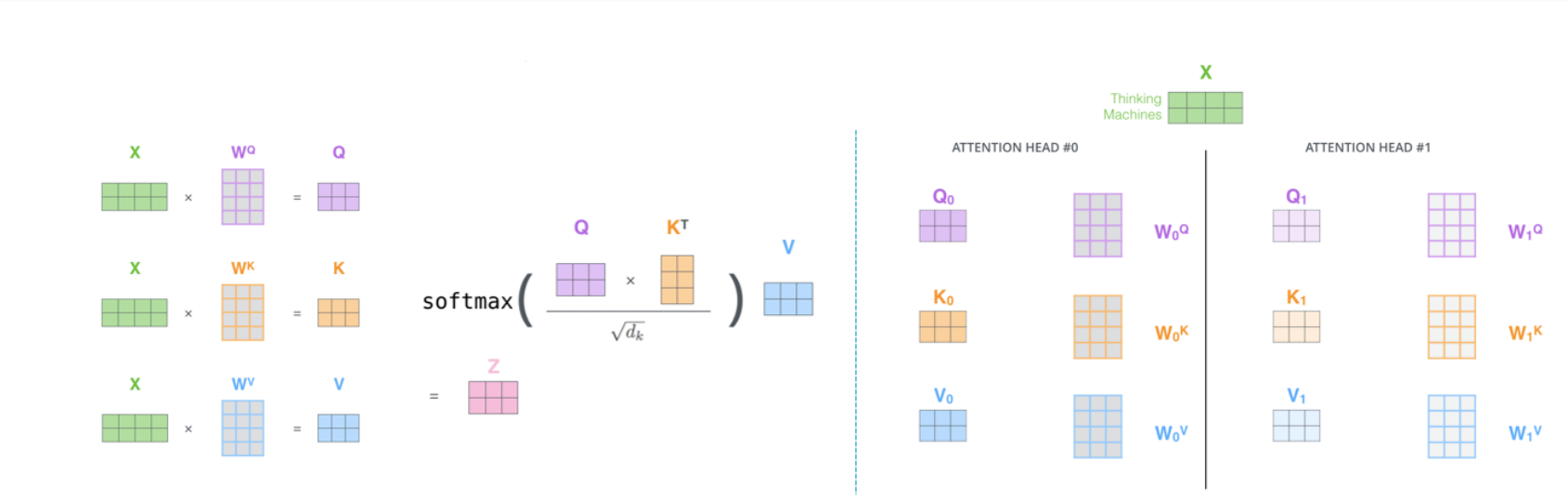
7. Multiplicamos el valor de la atención de cada vector de entrada (a_n) por el vector value v_1 (porque estamos calculando el valor de la atención para el **vector de entrada 1**)
8. Sumamos todos los resultados $a_n \cdot v_n$, obteniendo z_1

Y ahora sí que hemos acabado :-)



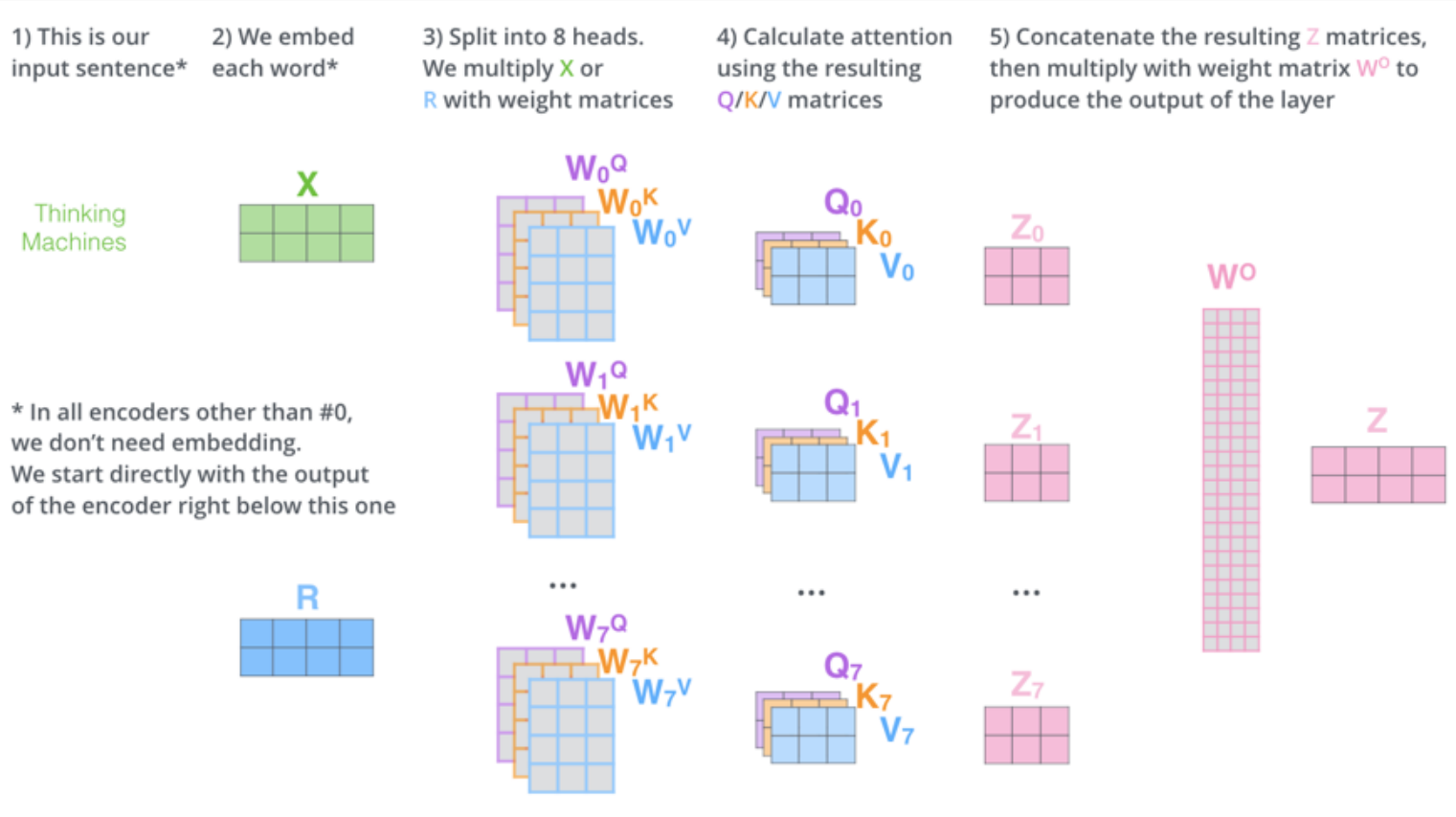
¿Cómo funciona el bloque de self-attention?

- Solamente indicar que esto, en realidad, se hace de forma **matricial** para ir más rápido.
- Y con más de un conjunto de W^Q, W^K, W^V (**multi-head**).



Ejemplo del cálculo de la **self-attention** (fuente).

Resumen

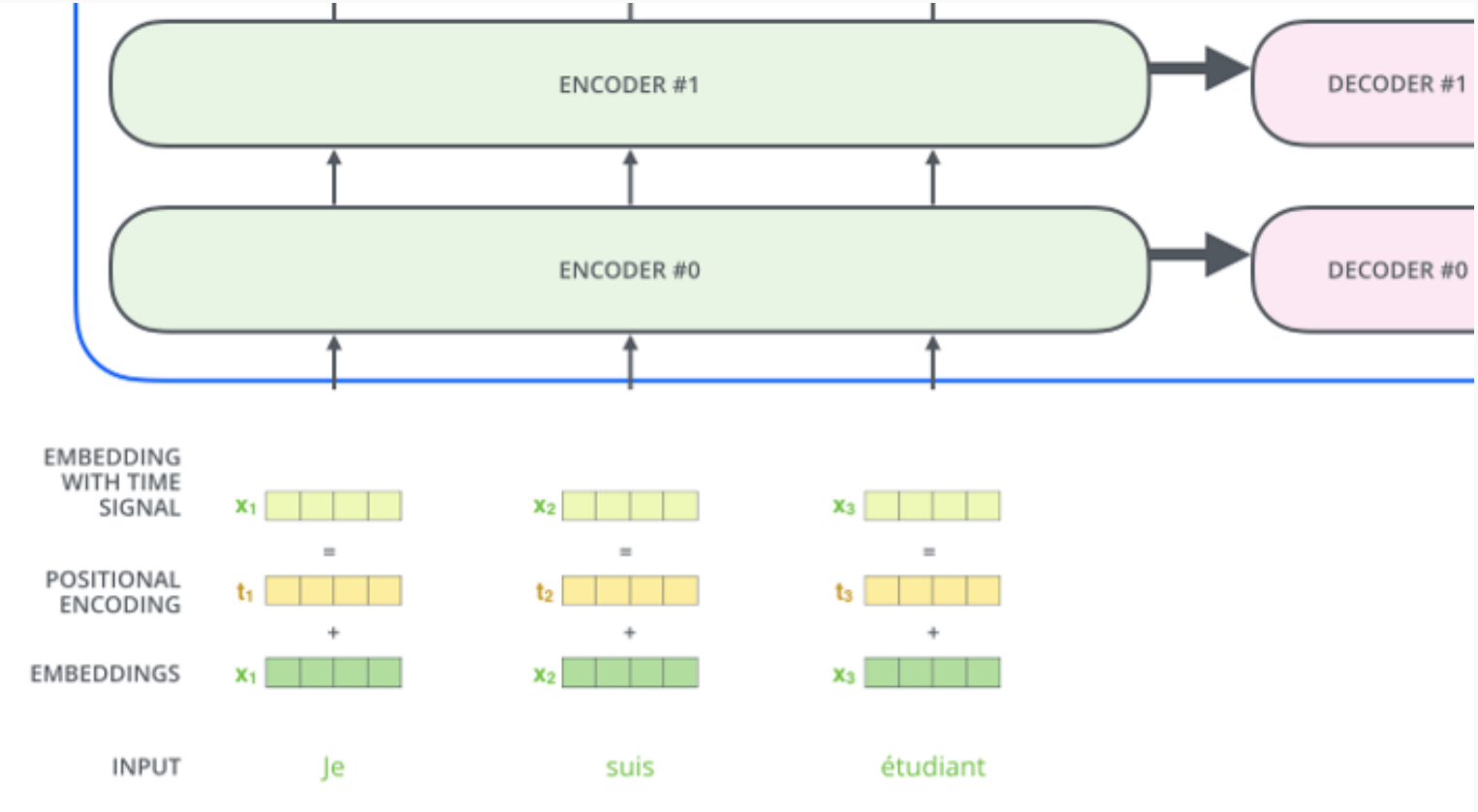


Resumen del mecanismo de **self-attention** (fuente).

Pero... ¿y la información de la posición?

- Si hemos dicho que se procesan todos los inputs de forma paralela, hemos perdido la información del tiempo.
- La solución es añadir lo que se conoce como **positional embedding**, que es lo que le indica a la red la posición de cada elemento de entrada en la secuencia completa.

Pero... ¿y la información de la posición?



Ejemplo de **positional embedding** (fuente).

Más información...

Toda esta explicación está basada en la de “The illustrated Transformer”, desarrollada por Jay Alammar.

La tenéis disponible en:
http://jalammar.github.io/illustrated-transformer/.

Si os habéis quedado con dudas, es **muy recomendable** que os leáis el post y os veáis el video, lo más seguro es que os disipe todas las dudas :-)

Ejemplo de transformer

Recursos

- Diapositivas de Moodle
- Google Collaboratory
- Deep Learning Book (https://www.deeplearningbook.org/)
- https://www.pyimagesearch.com/blog
- https://machinelearningmastery.com/blog