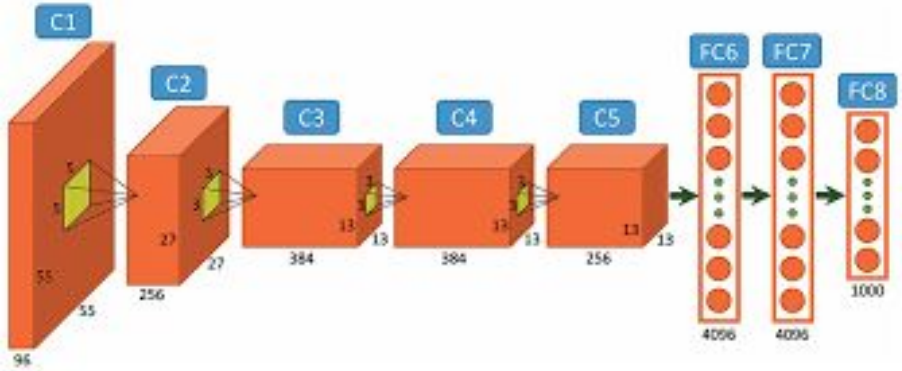
Introducción al Procesamiento de Lenguaje Natural

Clase 6. Transformers, LLMs y loros aleatorios

factor~data
EIDAES_UNSAM

Un poquito de historia…

- Hacia 2010-2011, la “vedette” de Deep Learning era Computer Vision: AlexNet, DanNet
- NLP siempre iba un poco atrás: pocas aplicaciones de Deep Learning
- Primer hito: word embeddings



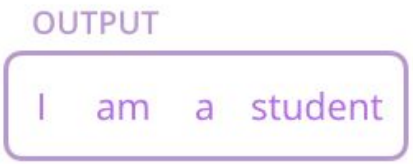
Un poquito de historia…

- A partir de allí NLP y Redes Neuronales se empezaron a llevar bien.
- Redes Neurnales Recursivas (RNN) y Long-Short Term Memory (LSTM)
- Las RNN aprenden una función autorregresiva

$$Y_t = f(x_t, y_{t-1})$$

Un ejemplo

Traducción
Problema Sequence to Sequence

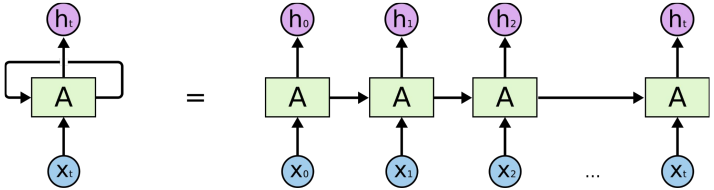


http://jalammar.github.io/illustrated-transformer/

Un ejemplo

INPUT

Je suis étudiant

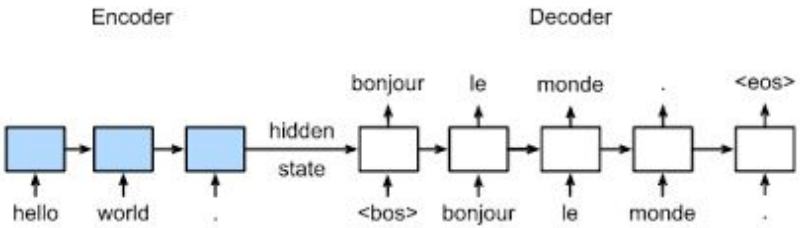


OUTPUT

I am a student

Un ejemplo

- 2014: un modelo para atacar el problema de Machine Translation.
- Idea: una red recurrente (encoder) codifica la oración de entrada en un vector. Luego, otra red recurrente (decoder) decodifica ese vector como un modelo de lenguaje condicionado.
- El decoder recibe como entrada este vector como hidden state y un token de “begin of sentence” (bos) y mediante



Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

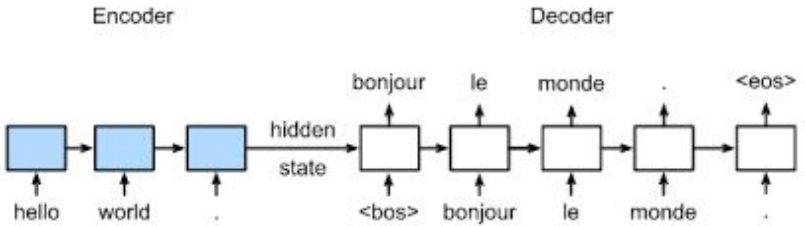
Quoc V. Le
Google
qvl@google.com

Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever

Un ejemplo - RNN

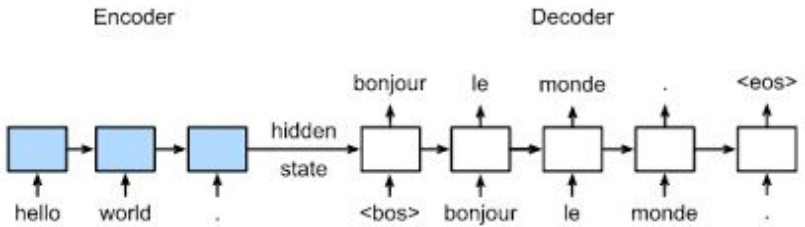
- Aprendizaje secuencial, tiene loop interno y va aprendiendo sobre lo que ya vio.
  - Sigue un loop interno. En cada iteración considera el estado actual del input y lo introduce (hidden state) para obtener output.



Un ejemplo - RNN

Limitaciones

- Es secuencial, loop que pasa de una etapa a la otra.
- No hay una *paralelización* del aprendizaje
- Se van “olvidando” algunas cosas
- Toda la información está comprimida en un solo vector.
- Solución: usar toda la info del encoder => Attention

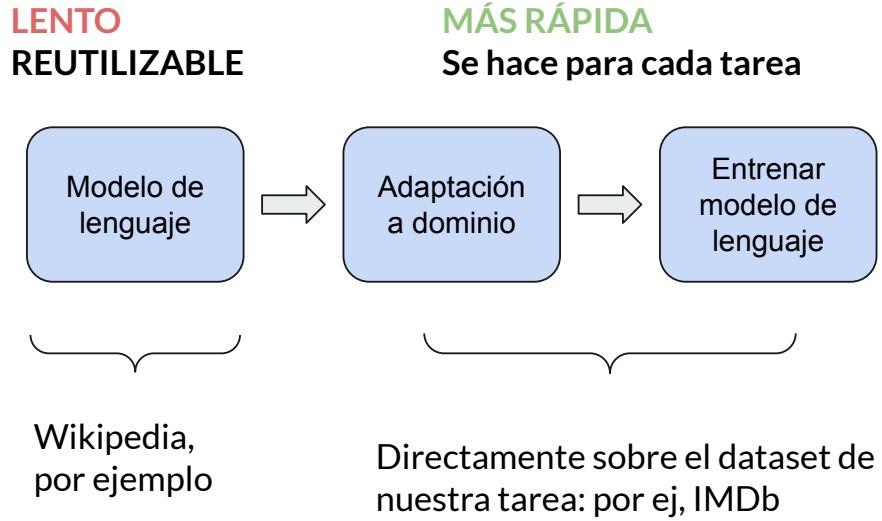


Cambio en el “paradigma” de NLP

Transfer Learning

- Enternar un modelo sobre un dataset grande
- Adaptar a nuevo dominio (por ejemplo, sentiment analysis de reseñas tipo IMDB)
- Entrenar para la nueva tarea, agregando una última capa al modelo grande.

- Reutilizamos casi la totalidad de la red
- Eso vamos a hacer con BERT



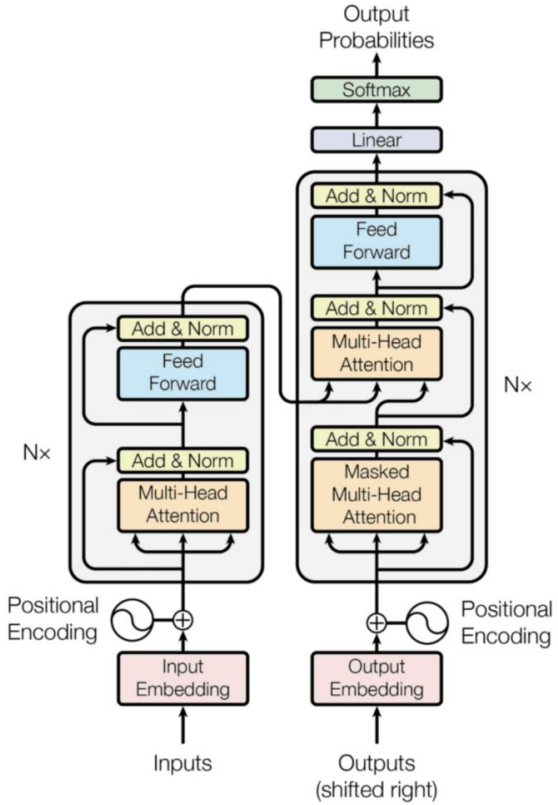
Un ejemplo

http://jalammar.github.io/illustrated-transformer/

Transformers

- Modelo paralelizable → puede procesar varias partes de una secuencia al mismo tiempo, lo que acelera considerablemente el entrenamiento y la inferencia.
- Capta las dependencias a largo plazo en el texto, lo que permite comprender mejor el contexto general y generar textos más coherentes.
- Utiliza mecanismos de **self-attention**.

Transformers



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 lukaszkaiser@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 41.8 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.

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

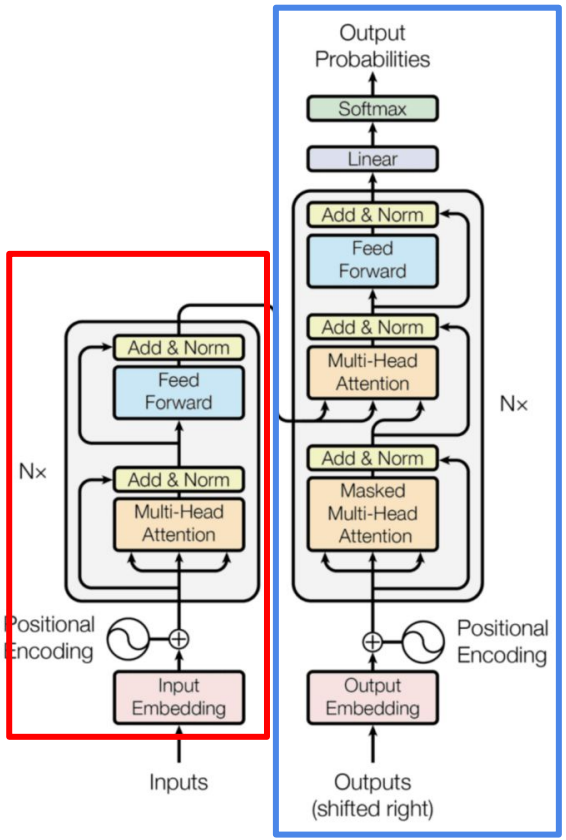
‡Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

factor~dat_
EIDAES_UNSAM

https://arxiv.org/abs/1706.03762

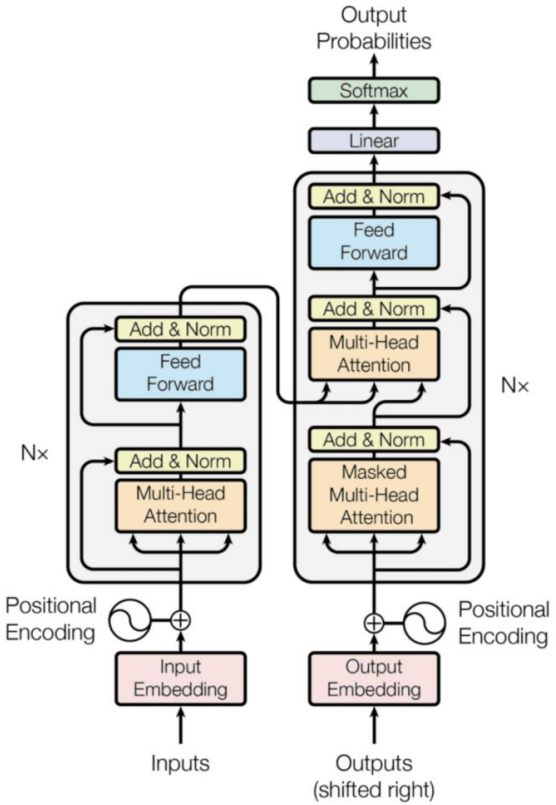
Transformers



Dos componentes:

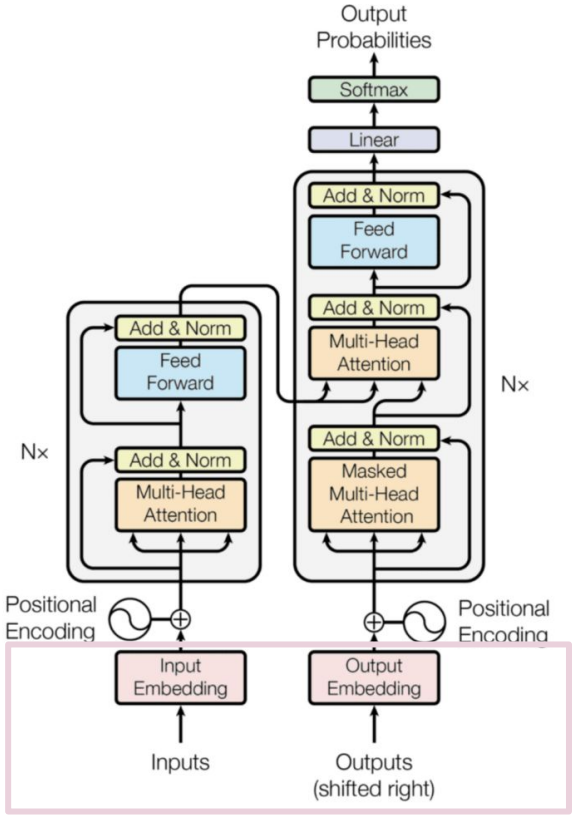
- Encoder: se centra en generar una representación “abstracta” del lenguaje => útil para tareas de clasificación, NER, identificación de párrafos
- Decoder: generación, útiles para generar frases

Transformers



Tres mecanismos importantes

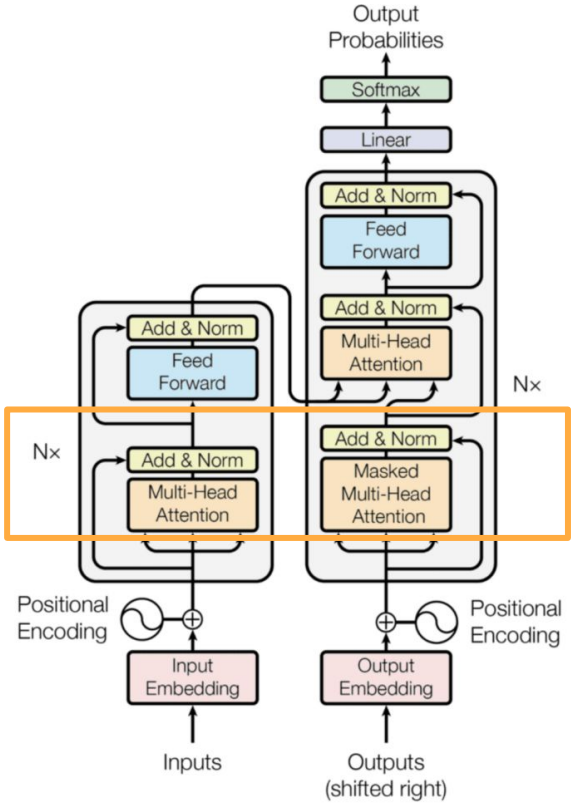
Transformers



Tres mecanismos importantes

- Input/Output Embeddings

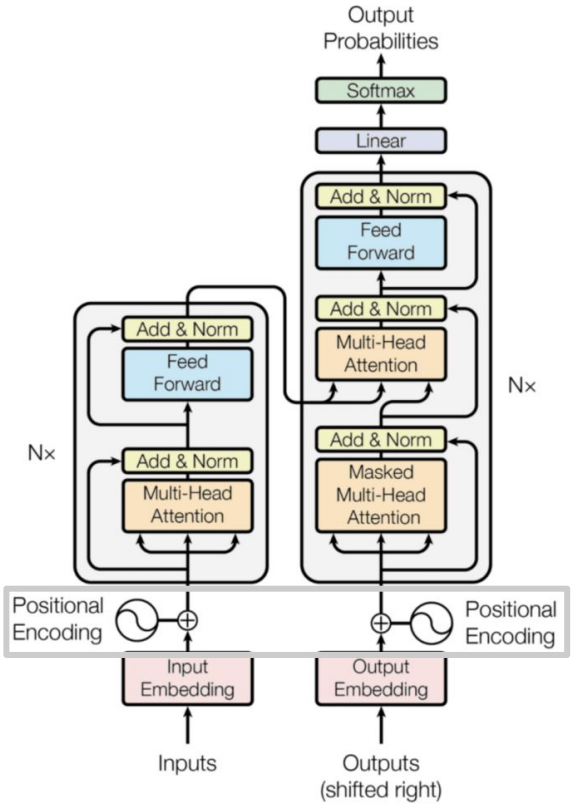
Transformers



Tres mecanismos importantes

- Input/Output Embeddings
- Multi-head Attention

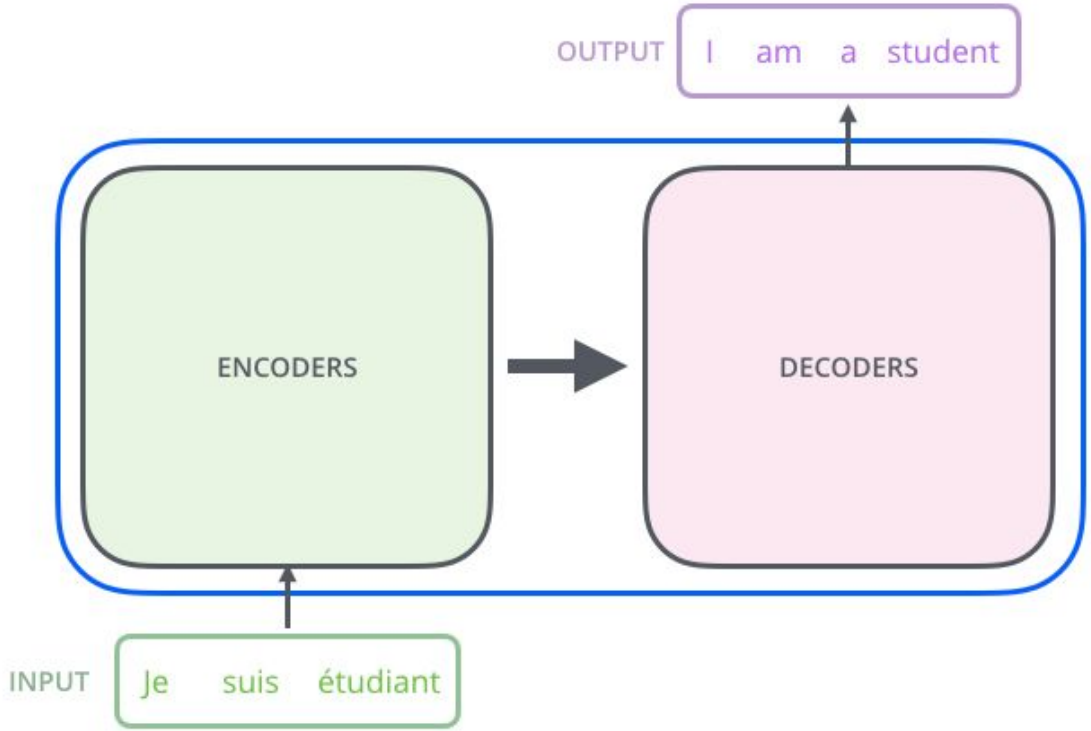
Transformers



Tres mecanismos importantes

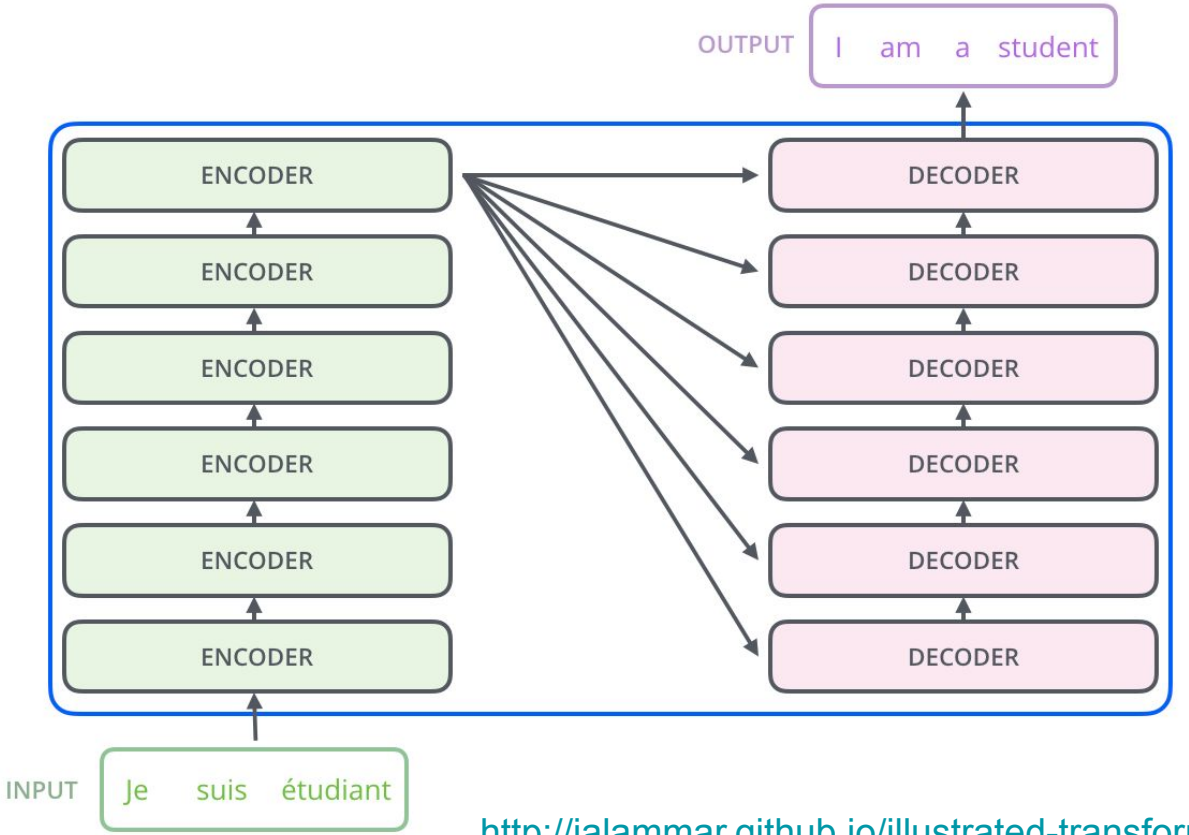
- Input/Output Embeddings
- Multi-head Attention
- Positional encoding

Abriendo la caja



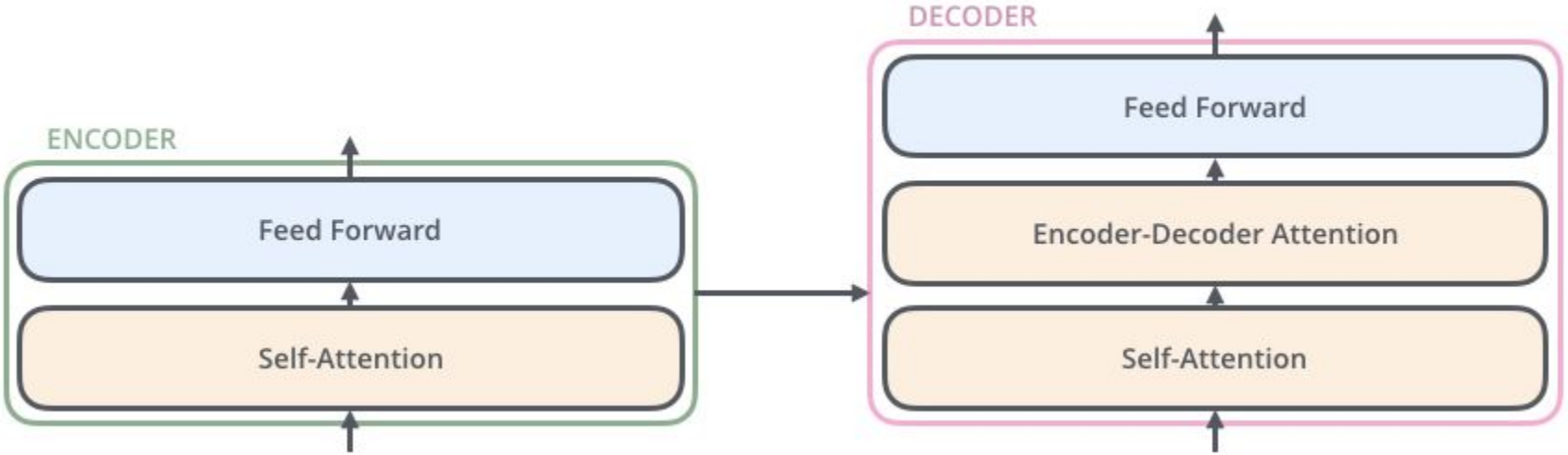
http://jalammar.github.io/illustrated-transformer/

Abriendo la caja



http://jalammar.github.io/illustrated-transformer/

Abriendo la caja



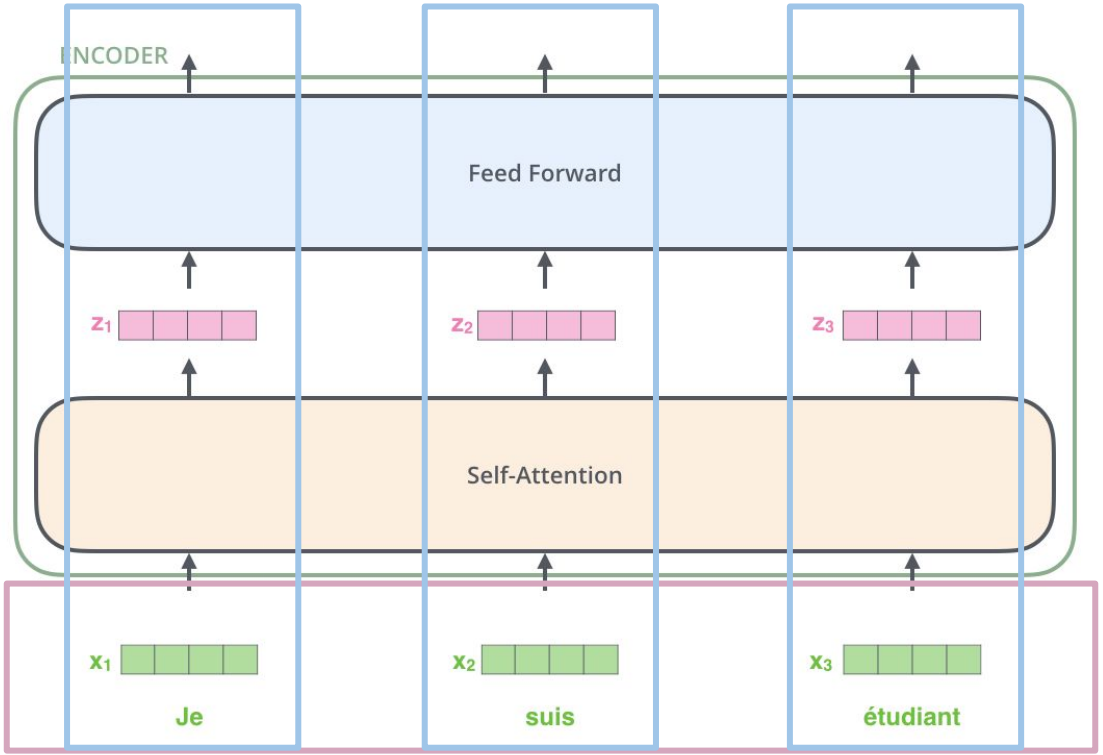
http://jalammar.github.io/illustrated-transformer/

Abriendo la caja

Cada palabra “fluye” de forma paralela a través del encoder.

¿Cómo se recuperan las dependencias de palabras? =>
Self-Attention mechanism

Word Embedding
(d = hiperparámetro)
Se entrena con el modelo



http://jalammar.github.io/illustrated-transformer/

Self-attention

“El perro no jugó con el niño porque él tenía pulgas”

- ¿A quién remite el término “él”? ¿Al perro o al niño?
- Para nosotros es evidente, pero para un modelo no.
- Cuando el modelo procesa la palabra "él", la atención propia le permite asociarla con “perro”.
- A medida que el modelo procesa cada palabra (cada posición en la secuencia de entrada), *self-attention* le permite buscar otras posiciones en la secuencia de entrada en busca de pistas que puedan ayudar a codificar mejor esta palabra.

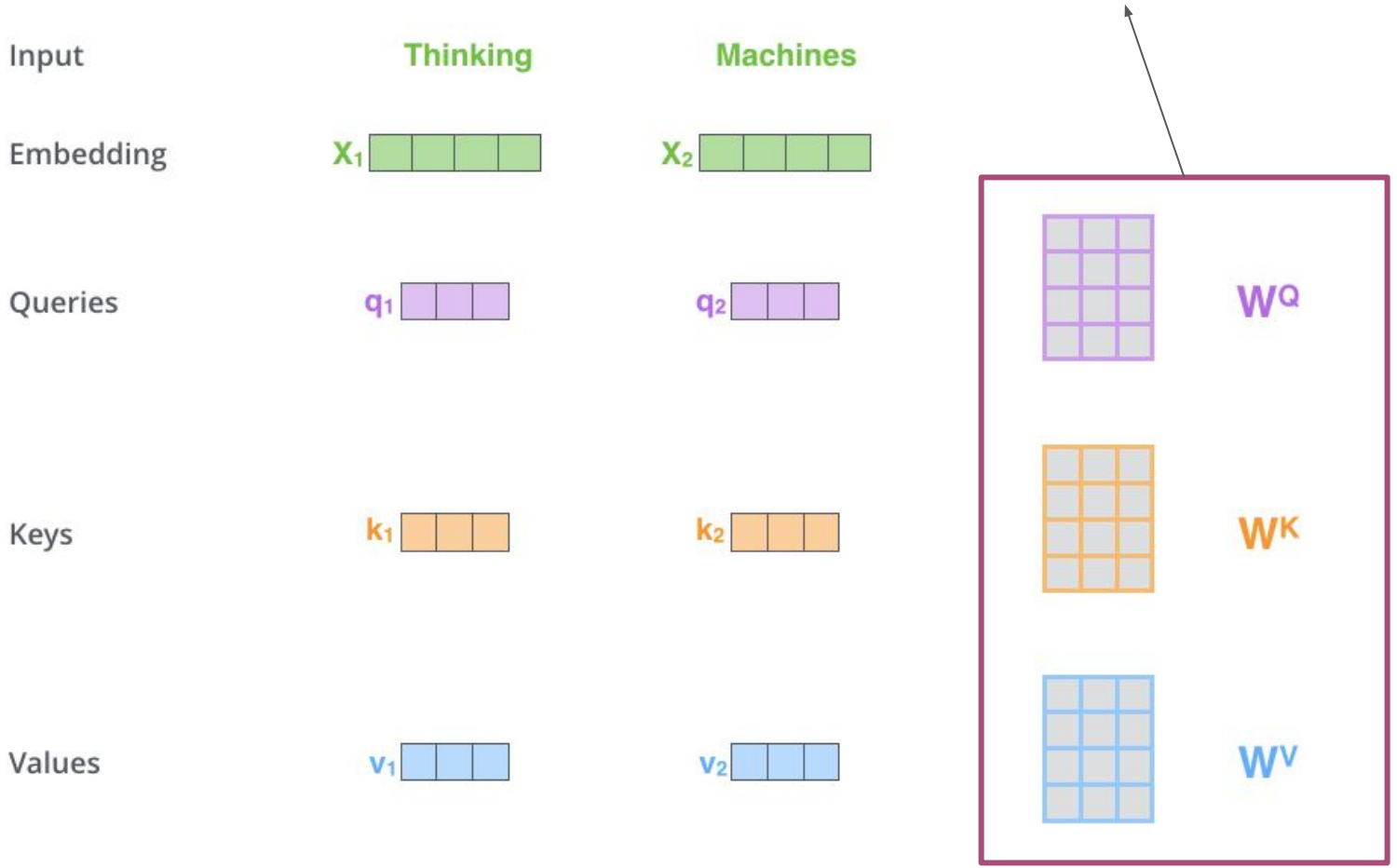
http://jalammar.github.io/illustrated-transformer/

Self-attention

- Cada input se asocia a tres vectores:
  - Query (Q), Key (K) y Value (V).
  - Los vectores surgen de multiplicar cada embedding de cada palabra por una matriz de pesos (WQ, WK y WV) que se aprenden durante el entrenamiento.
- Se calculan las puntuaciones de similitud entre los vectores de Q y K.
  - Indican cuánta atención debe prestarse a cada elemento de la secuencia al procesar el elemento actual.
- Suma ponderada: Las puntuaciones de atención se utilizan para calcular una suma ponderada de los vectores. Esta suma ponderada representa el contexto o la información de toda la secuencia de entrada relevante para el elemento actual.

Self-attention

Son producto del entrenamiento…



Self-attention

- Atención multicabezal: La autoatención se aplica normalmente en paralelo varias veces con diferentes conjuntos de vectores Q, K y V aprendidos, creando múltiples "cabezas de atención".
- Esto permite al modelo centrarse en diferentes aspectos de los datos de entrada y capturar varios tipos de relaciones.

1) This is our input sentence*

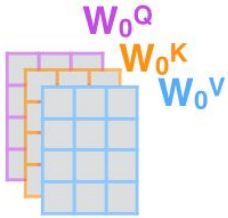
2) We embed each word*

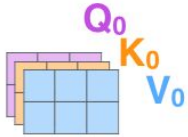
3) Split into 8 heads. We multiply X or R with weight matrices

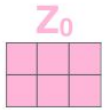
4) Calculate attention using the resulting Q/K/V matrices

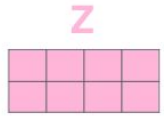
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

Thinking
Machines

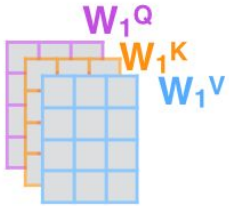


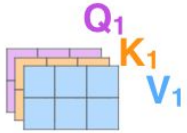


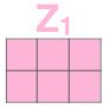


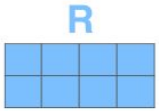


* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

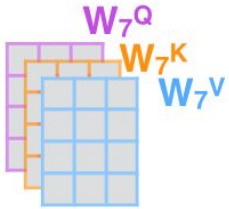




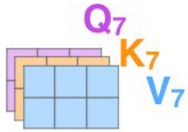




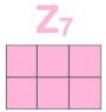
...



...



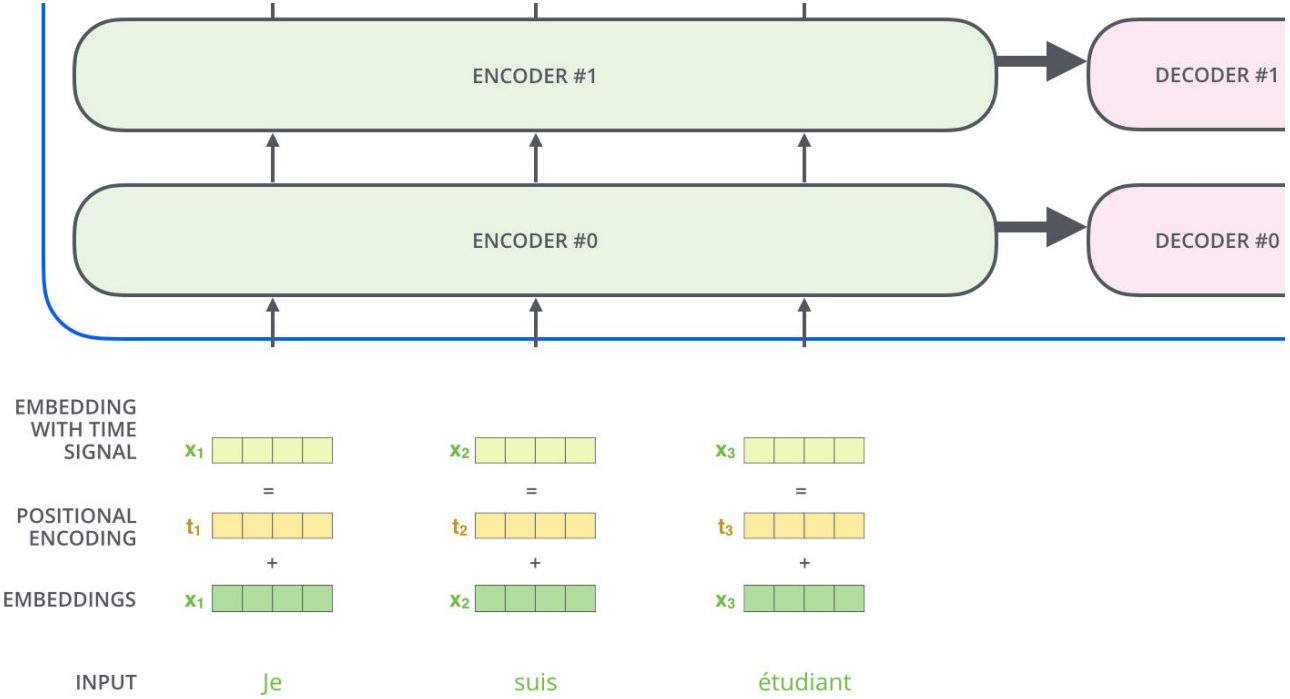
...



Positional encoding

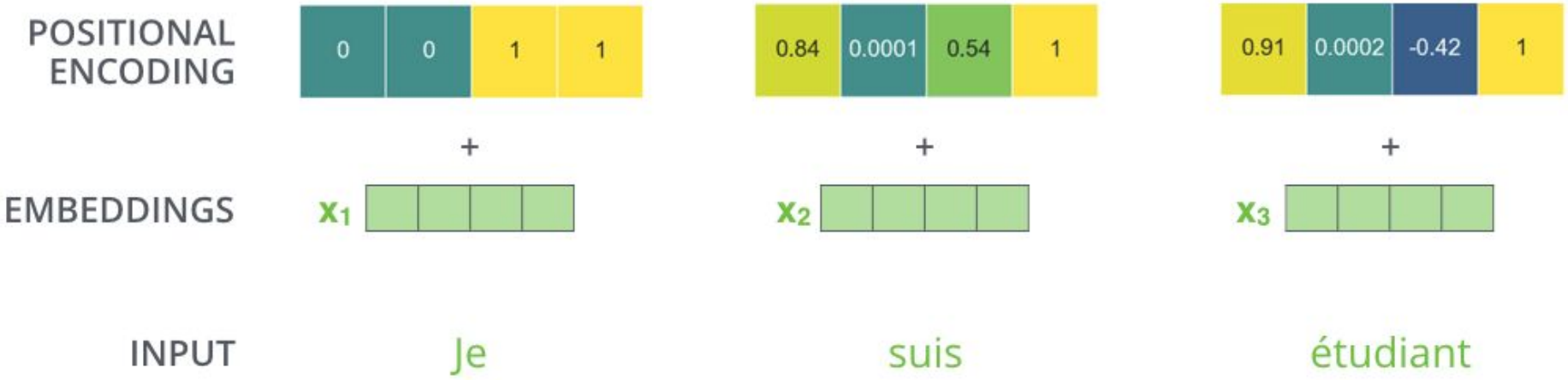
Nos falta algo: necesitamos poder identificar el orden o la posición de cada palabra en la secuencia de input.

Para esto, el modelo agrega un vector a cada uno de los embeddings de input



http://jalammar.github.io/illustrated-transformer/

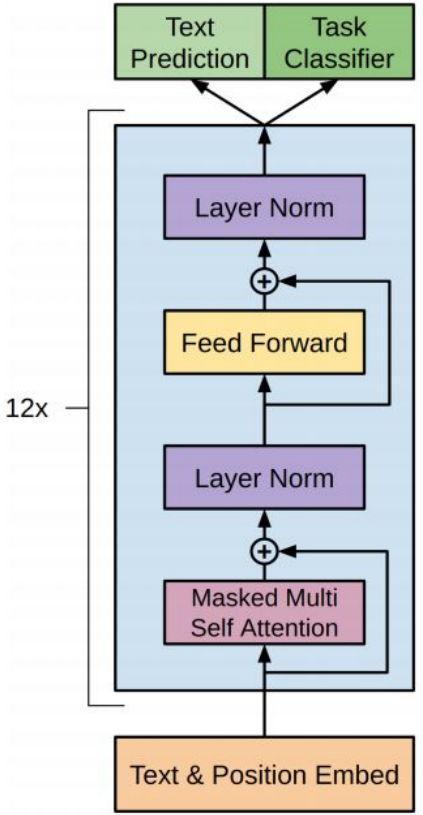
Positional encoding



http://jalammar.github.io/illustrated-transformer/

Modelos basados en Transformers: GPT

- Radford et al (2018) entrenaron un modelo de lenguaje basado en 12 capas de Transformers
- Sacamos la última capa y la reemplazamos por una capa para clasificación
- Entrenamos el modelo completo con un learning rate bajo
- Pequeño detalle: como el modelo de lenguaje sólo puede mirar para atrás, se usan máscaras en la atención
- State-of-the-art en GLUE (benchmark de varias tareas de NLP)



Modelos basados en Transformers: BERT

- Misma idea que GPT, pero en vez de entrenar un modelo de lenguaje, usan otras dos tareas:
  - Predecir un token enmascarado (Masked Language Model)
  - Ver si una oración es la que le sigue a otra (Next-Sentence prediction)
- Entrenado sobre Wikipedia o Common Crawl
- Acá no usamos ninguna máscara ya que todos los tokens pueden ver a los demás
- Sobrepasó a GPT en el GLUE