



IIC 3800 Tópicos en CC NLP

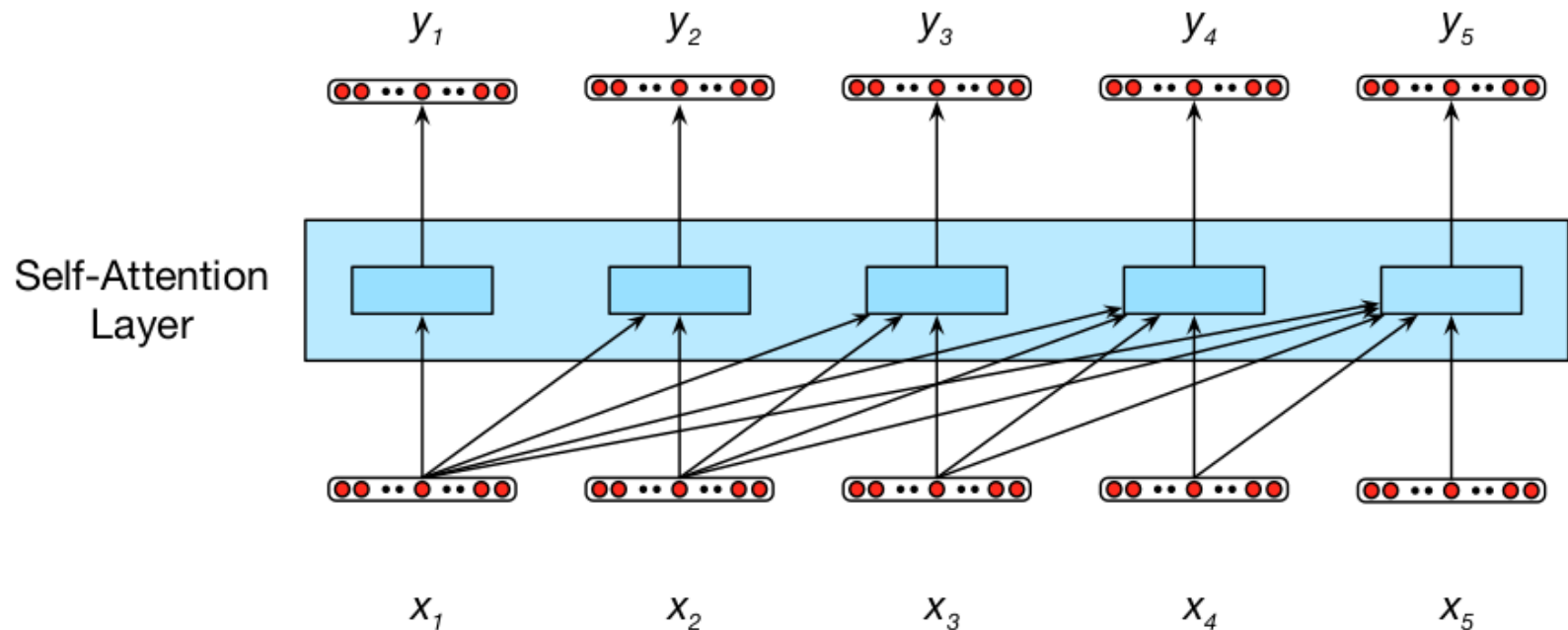
<https://github.com/marcelomendoza/IIC3800>

- TRANSFORMER -

Mecanismo de auto-atención

La auto-atención permite a la red extraer información desde contextos de largo variable sin la necesidad de pasar por una red recurrente.

En una capa de auto-atención, el modelo (FFNN) tiene acceso a todas las entradas procesadas hasta ese step:



Este enfoque se llama 'masked self-attention'.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., Polosukhin, I., Attention Is All You Need, NIPS, 2017

Mecanismo de auto-atención

La base del mecanismo está en la comparación de un ítem de interés con otros de manera que revele su relevancia en el contexto del step actual.

La forma más simple de comparación es el producto punto de vectores. Por ejemplo, para calcular y_3 debemos comparar $x_3 \cdot x_1$, $x_3 \cdot x_2$ y $x_3 \cdot x_3$. Los productos se normalizan con una softmax para crear un vector de pesos α_{ij} , que indica la relevancia de cada entrada (j) al símbolo de posición i .

$$\begin{aligned}\alpha_{ij} &= \text{softmax}(\text{score}(x_i, x_j)) \quad \forall j \leq i \\ &= \frac{\exp(\text{score}(x_i, x_j))}{\sum_{k=1}^i \exp(\text{score}(x_i, x_k))} \quad \forall j \leq i\end{aligned}$$

Mecanismo de auto-atención

Los Transformers agregan a la capa de self-attention parámetros adicionales que operan sobre los embeddings de entrada. La motivación se basa en tres roles que juegan los embeddings de entrada en el proceso de atención:

- Query: la input embedding se considera como foco actual de atención al compararse con las entradas precedentes.
- Key: En su rol como entrada precedente, el input embedding del step anterior se compara con el siguiente.
- Valor: se usa el input embedding para calcular el valor de salida en el time step actual.

Los parámetros que se agregan son:

$$q_i = W^Q x_i; \quad k_i = W^K x_i; \quad v_i = W^V x_i$$

En el Transformer original, el input embedding es de dim=1024 pero se reducen a 64 al pasarlos por las matrices W (1024 x 64).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., Polosukhin, I., Attention Is All You Need, NIPS, 2017

Mecanismo de auto-atención

Los Transformers en lugar de ingestar los input embeddings en la capa de self-attention, usan el vector q_i para el símbolo de step i , y lo comparan con los vectores k_j precedentes:

$$y_i = \sum_{j \leq i} \alpha_{ij} v_j$$

$\nearrow \text{score}(x_i, x_j) = q_i \cdot k_j$
 \searrow Vector value

Los Transformers normalizan el producto punto por la raíz de la dimensionalidad de los vectores (empírico). Dado que la capa de atención es FFNN, los cálculos en cada step se pueden realizar de forma simultánea (paralelo). Usando notación matricial:

$$Q = W^Q X; \quad K = W^K X; \quad V = W^V X$$

La capa de auto-atención puede representarse según:

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

Tienen el contexto hacia ambos lados.

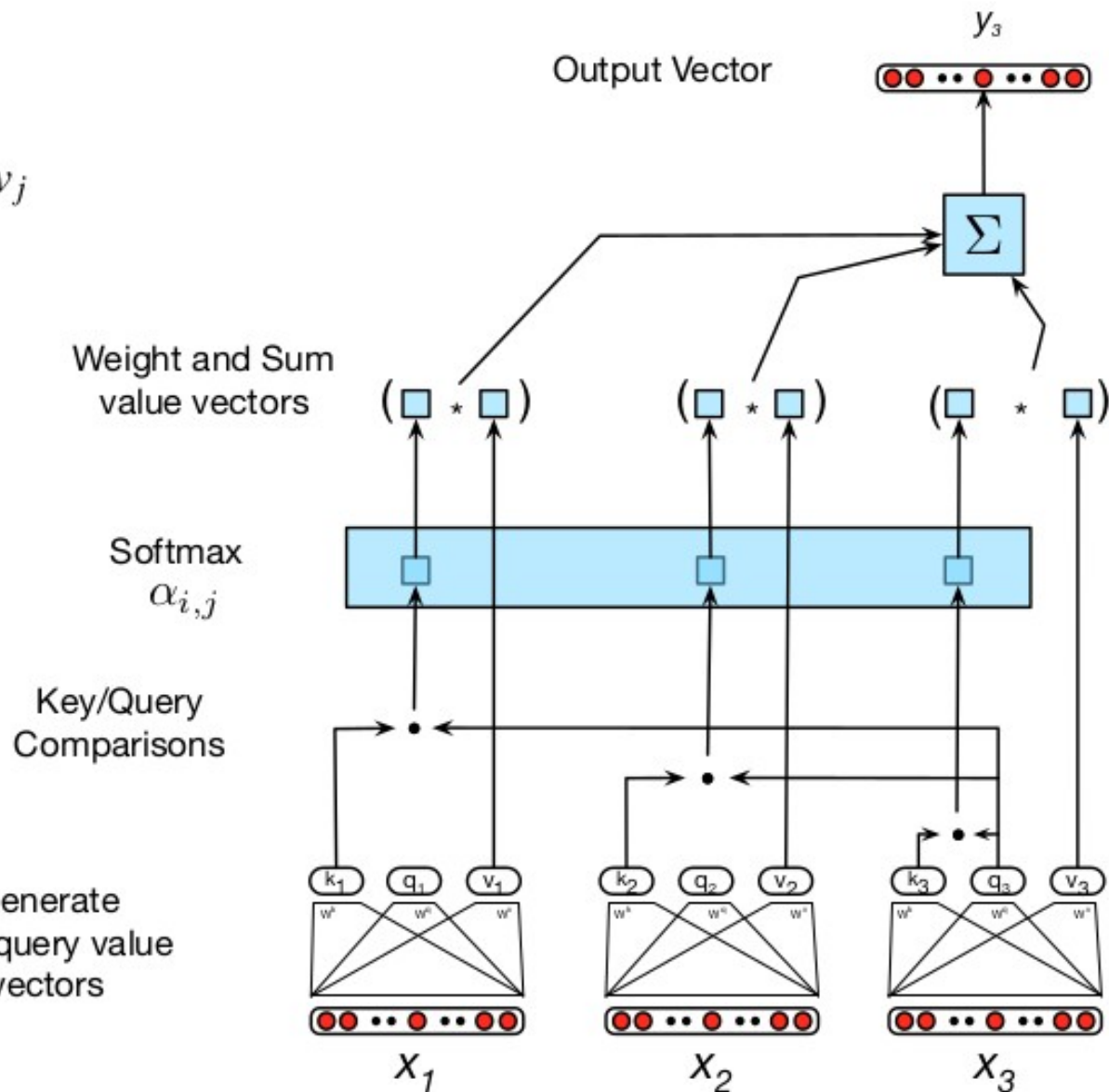
Mecanismo de auto-atención

$$y_i = \sum_{j \leq i} \alpha_{ij} v_j$$

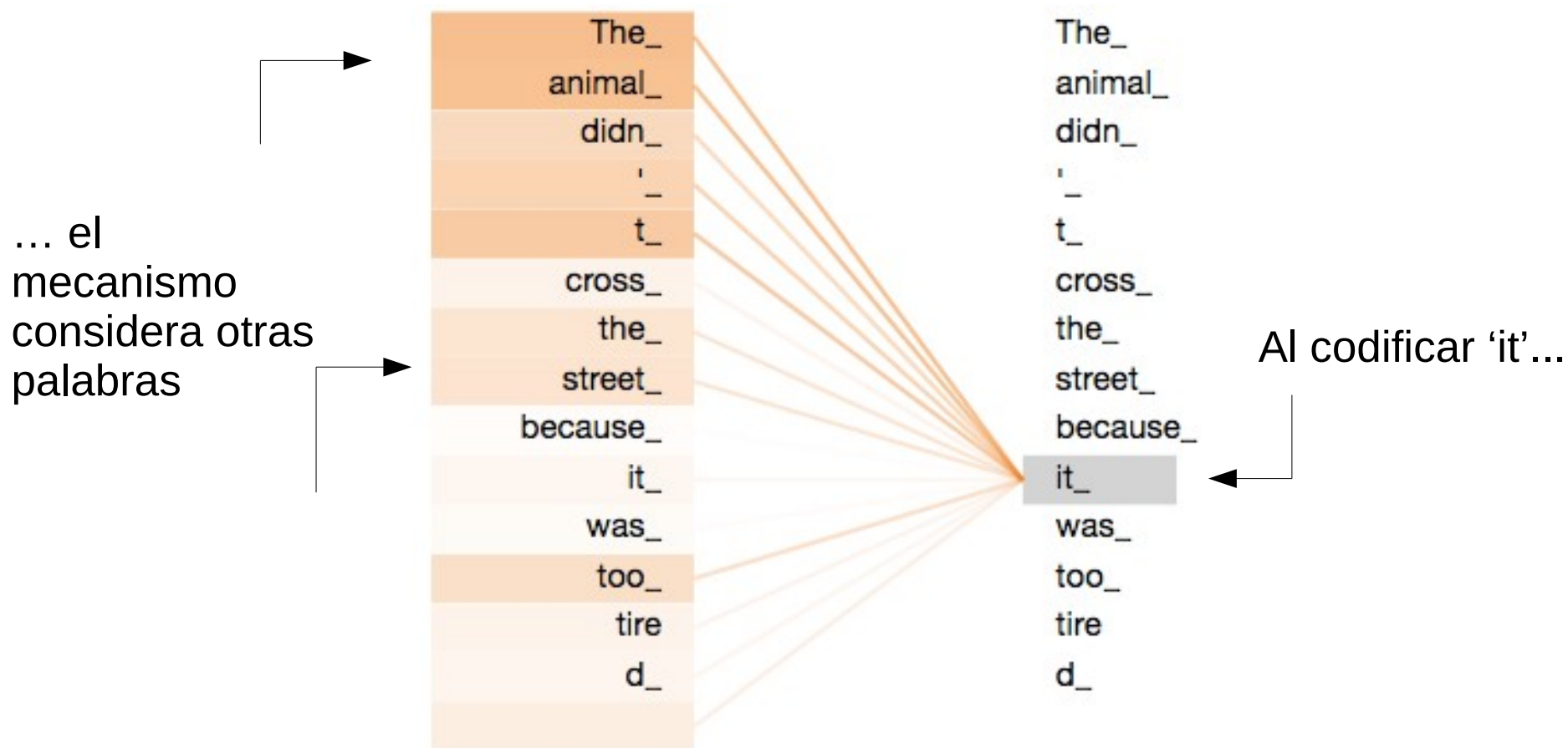
$$\text{score}(x_i, x_j) = q_i \cdot k_j$$

$$Q = W^Q X; \quad K = W^K X; \quad V = W^V X$$

Generate
key, query value
vectors



Mecanismo de auto-atención



Mecanismo de auto-atención

Auto-atención en el **encoder** (query, key y value):

Input

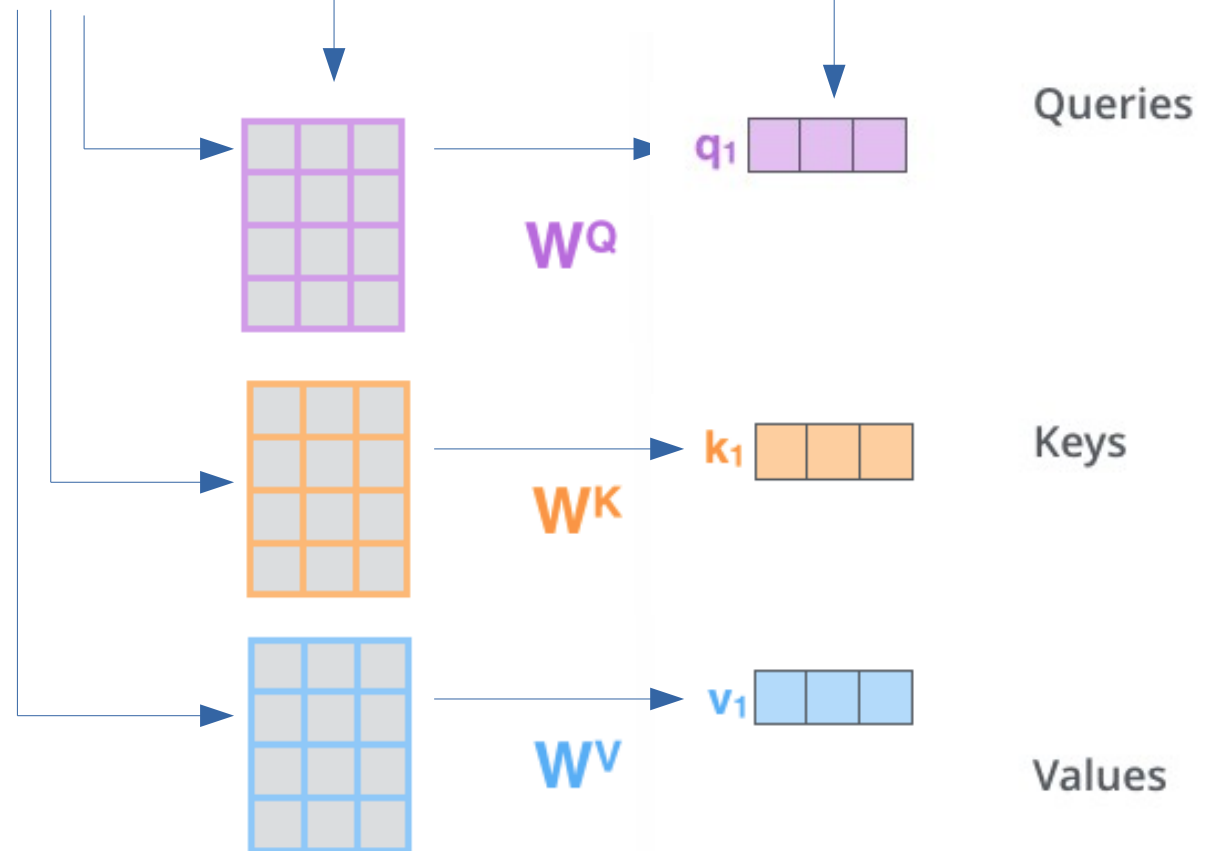
Thinking

Embedding

X_1

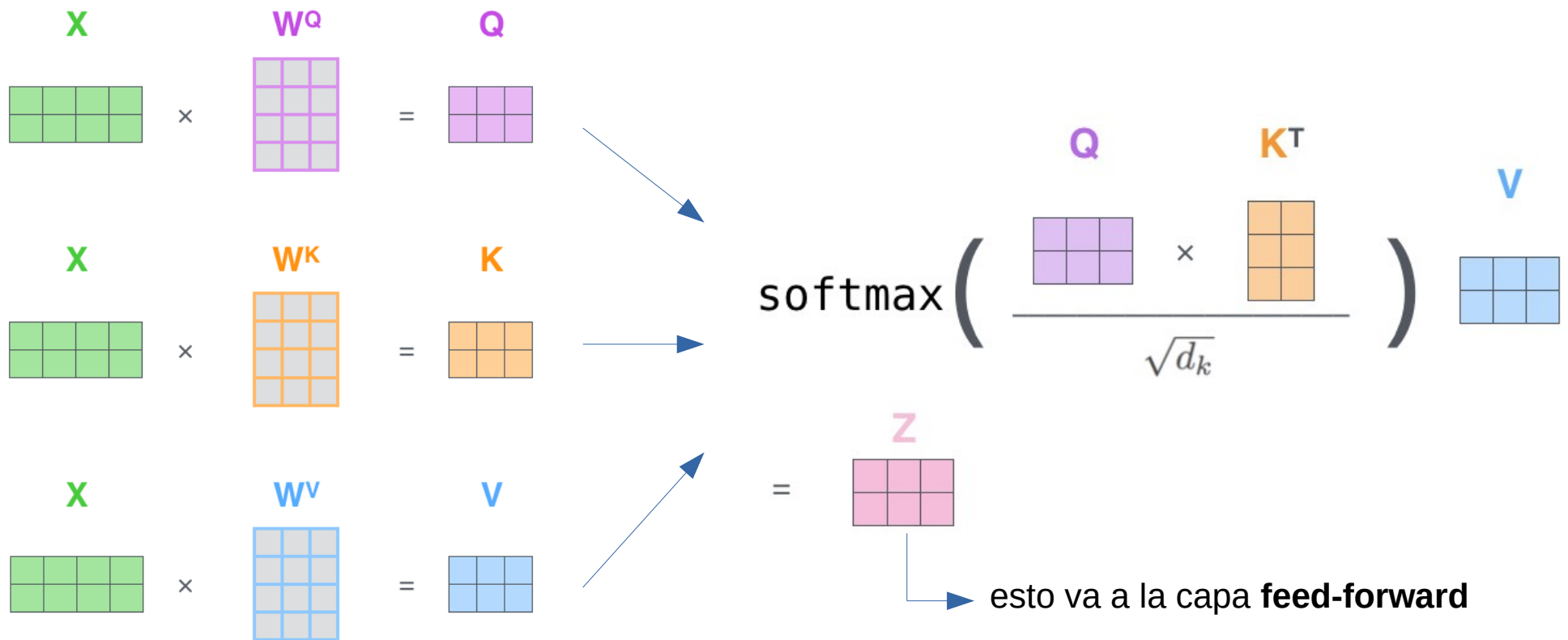
W^Q , W^K y W^V se aprenden durante el entrenamiento

estos vectores son de baja dimensionalidad



Mecanismo de auto-atención

Auto-atención en el **encoder** (cálculo matricial):



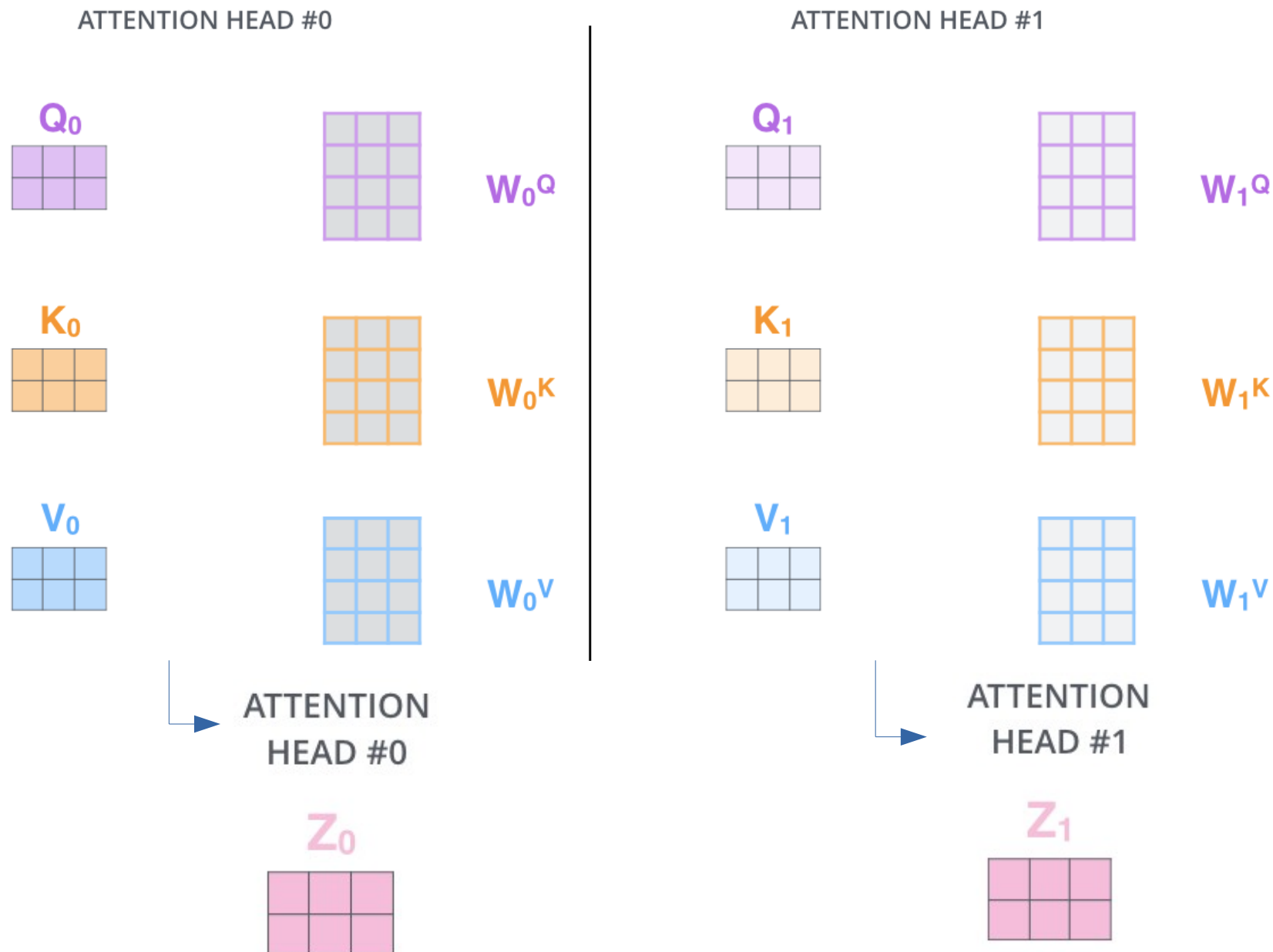
Mecanismo de auto-atención multi-cabezal

Los Transformers usan un mecanismo de auto-atención multi-cabezal para proporcionar al modelo la habilidad de aprender distintos aspectos de las relaciones entre entradas al mismo nivel de abstracción.

Cada cabezal tiene su propio set de parámetros q , k y v para proyectar la entrada a la capa correspondiente. La salida de una multi-head attention layer con h cabezales es de h veces la dimensionalidad de la entrada. Luego, los h vectores son combinados y reducidos a la dimensionalidad original usando una capa de proyección lineal:

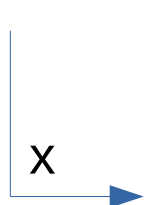
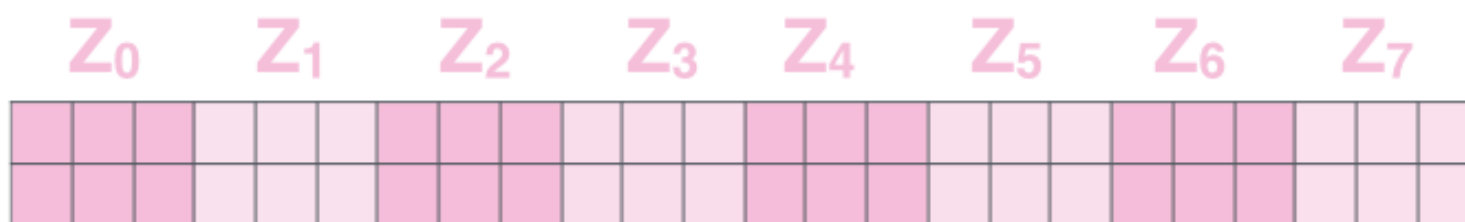
$$\begin{aligned} \text{MultiHeadAttn}(Q, K, V) &= W^O(\text{head}_1 \oplus \text{head}_2 \dots \oplus \text{head}_h) \\ \text{head}_i &= \text{SelfAttention}(W_i^Q X, W_i^K X, W_i^V X) \end{aligned}$$

Mecanismo de auto-atención multi-cabezal



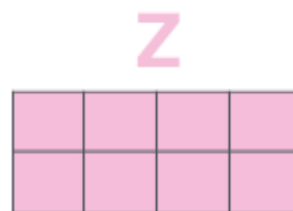
Mecanismo de auto-atención multi-cabezal

Concatenamos los Z ...



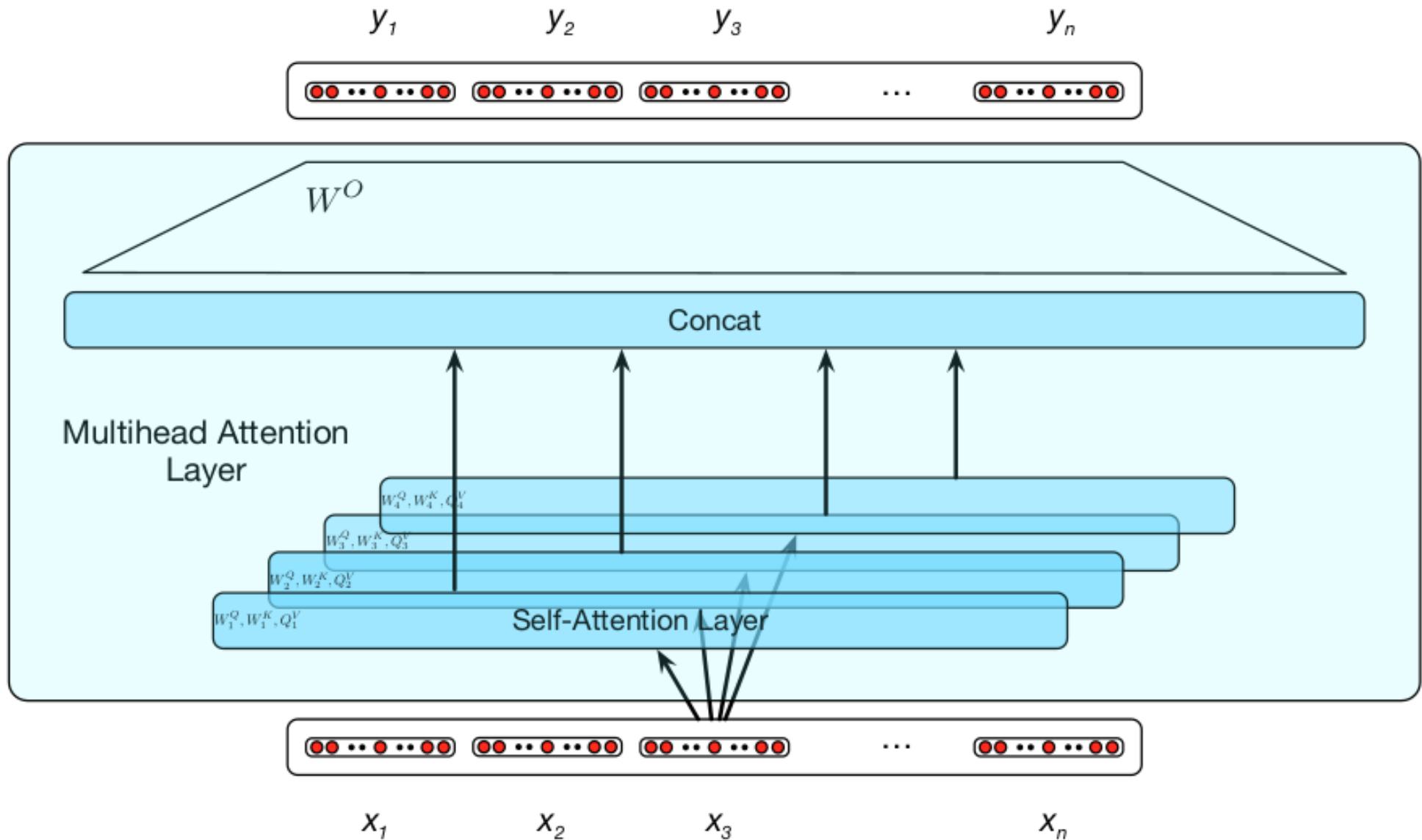
... y la reducimos a la dimensionalidad original.

$W^O \rightarrow =$



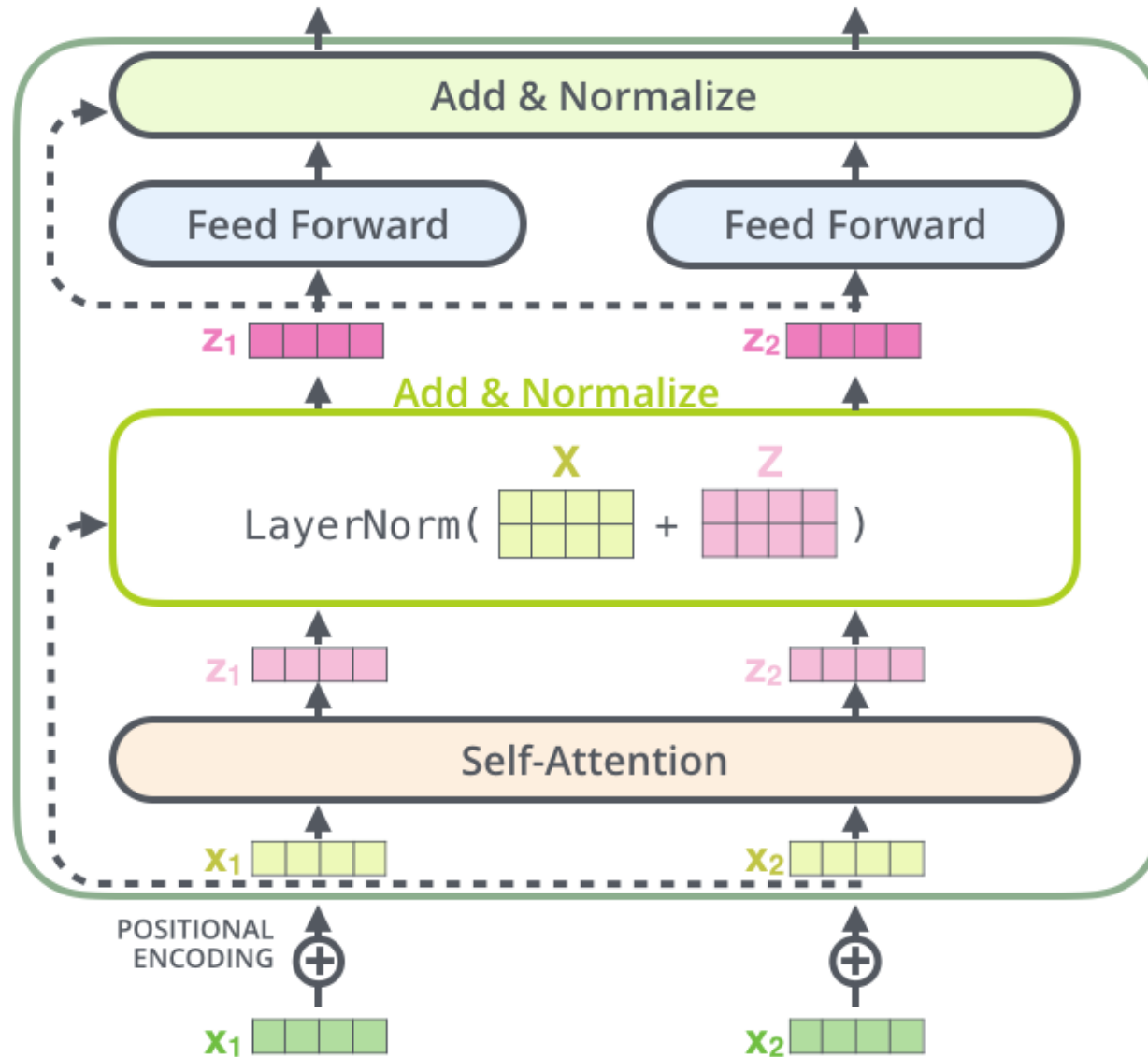
esto va a la capa **feed-forward**

Mecanismo de auto-atención multi-cabezal



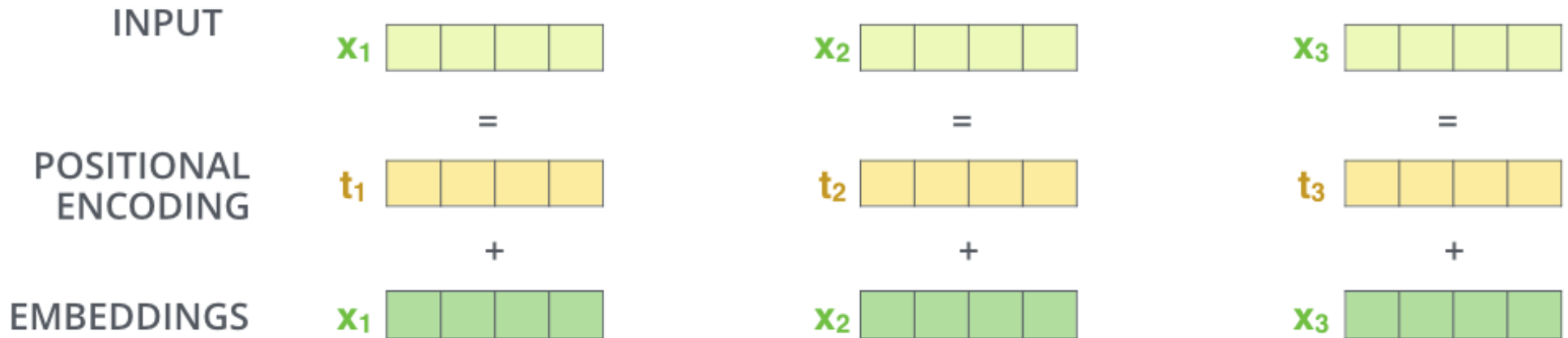
Add & Normalize (transformer block)

A la salida de cada capa se suma la entrada y se normaliza (conexiones residuales)

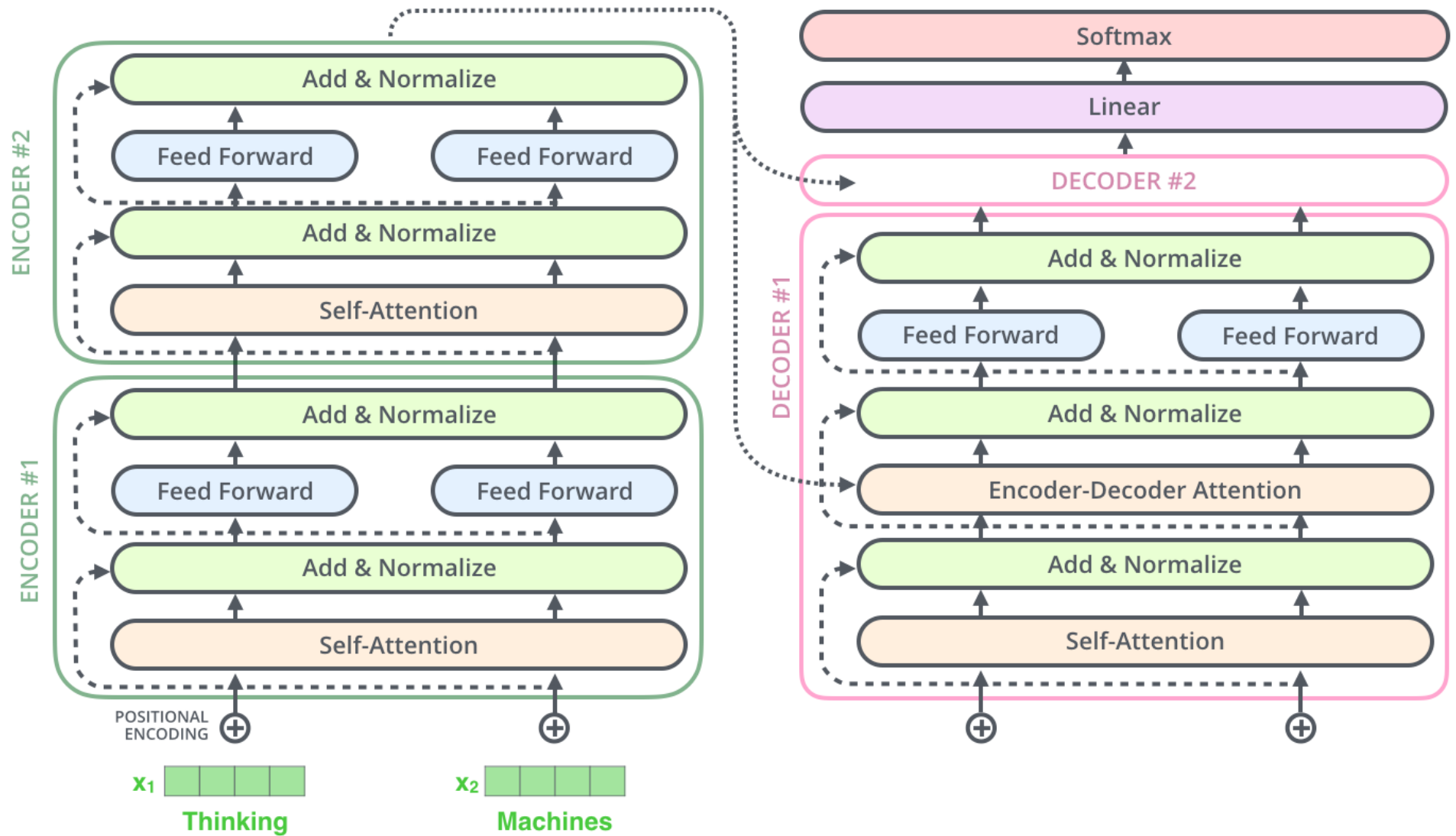


Positional encoding

Se suma un vector a la entrada, el cual codifica la posición de la palabra en la sentencia




Stacked blocks, encoders y decoders del Transformer



- BERT -

Bidirectional Encoder Representations on Transformers (BERT)

- Entrada: WPM


$$\text{tokenize}(\text{text}_i) = x_i = (x_i^1, x_i^2, \dots, x_i^{n_i})$$

Word-piece model →

- **Word:** Jet makers feud over seat width with big orders at stake
- **wordpieces:** _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

LM que minimiza el
número de tokens
necesarios para
representar un
corpus



Basado en Byte-Pair
encoding (BPE)



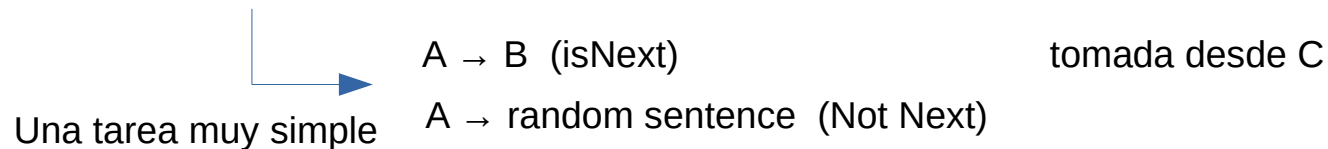
Reemplaza un 2-
char gram frecuente
por un 1-gram poco
usado

Bidirectional Encoder Representations on Transformers (BERT)

- Usa contexto bidireccional
- Usa el **encoder** del **transformer**

Building blocks de **BERT**:

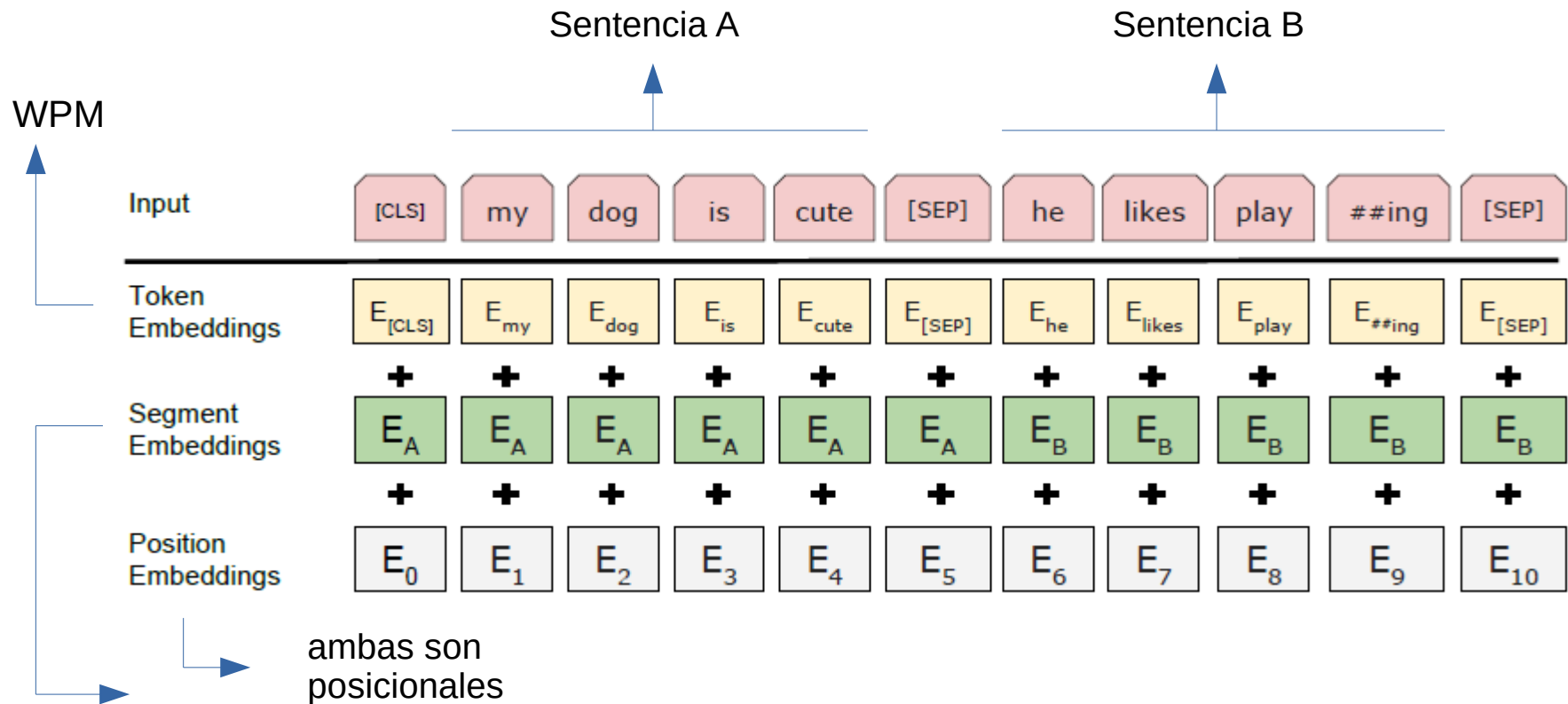
- Text2token usando el **Word-Piece Model** (igual que GPT-1 y GoogleMT)
- Task 1: **MLM** (*masked language model*) para *pre-training* a nivel de *tokens*
- Task 2: **Next sentence prediction (NSP)**



Jacob Devlin, [Ming-Wei Chang](#), [Kenton Lee](#), [Kristina Toutanova](#): BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. [NAACL-HLT \(1\)2019](#): 4171-4186

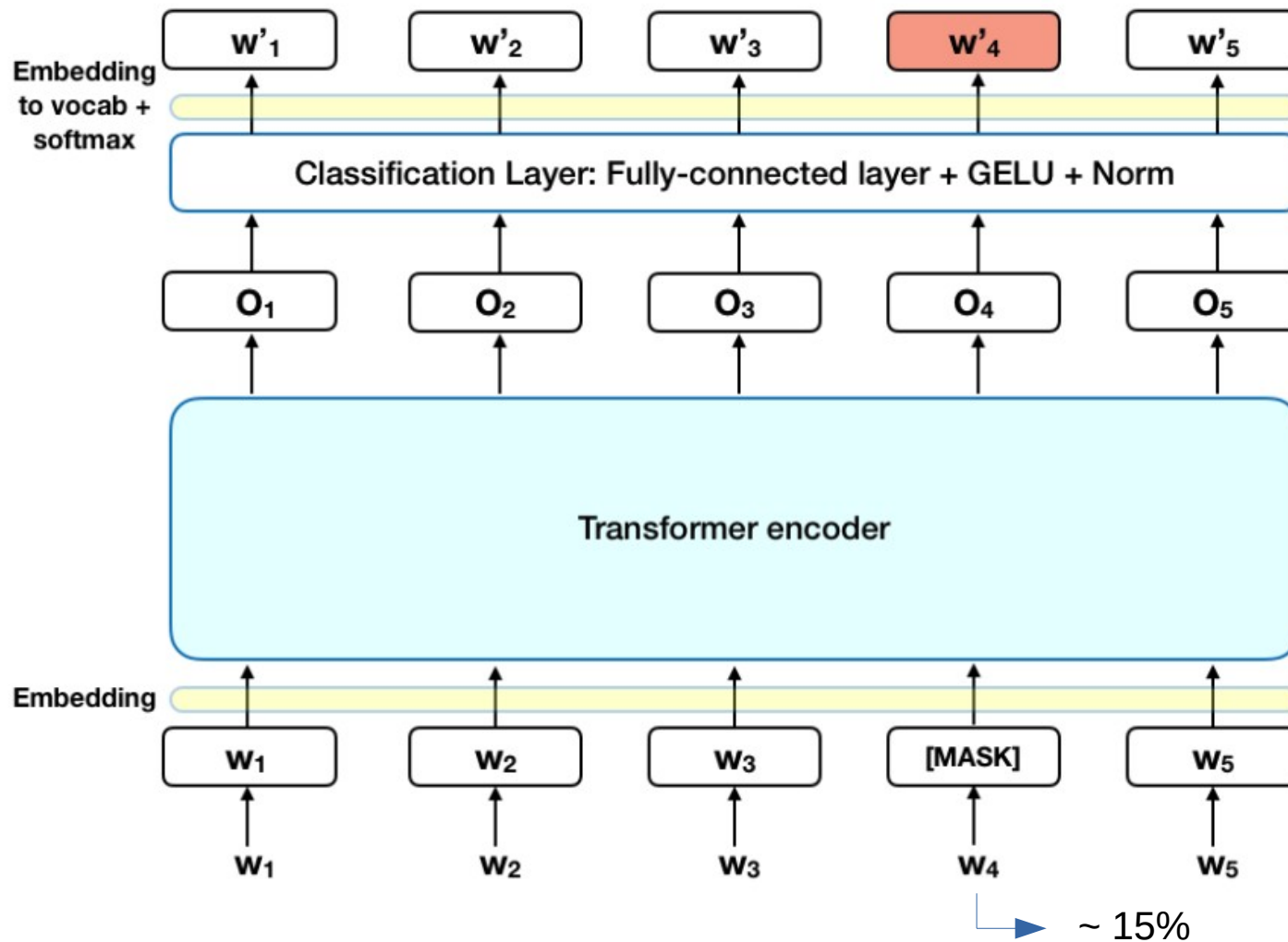
Bidirectional Encoder Representations on Transformers (BERT)

La entrada de **BERT**:



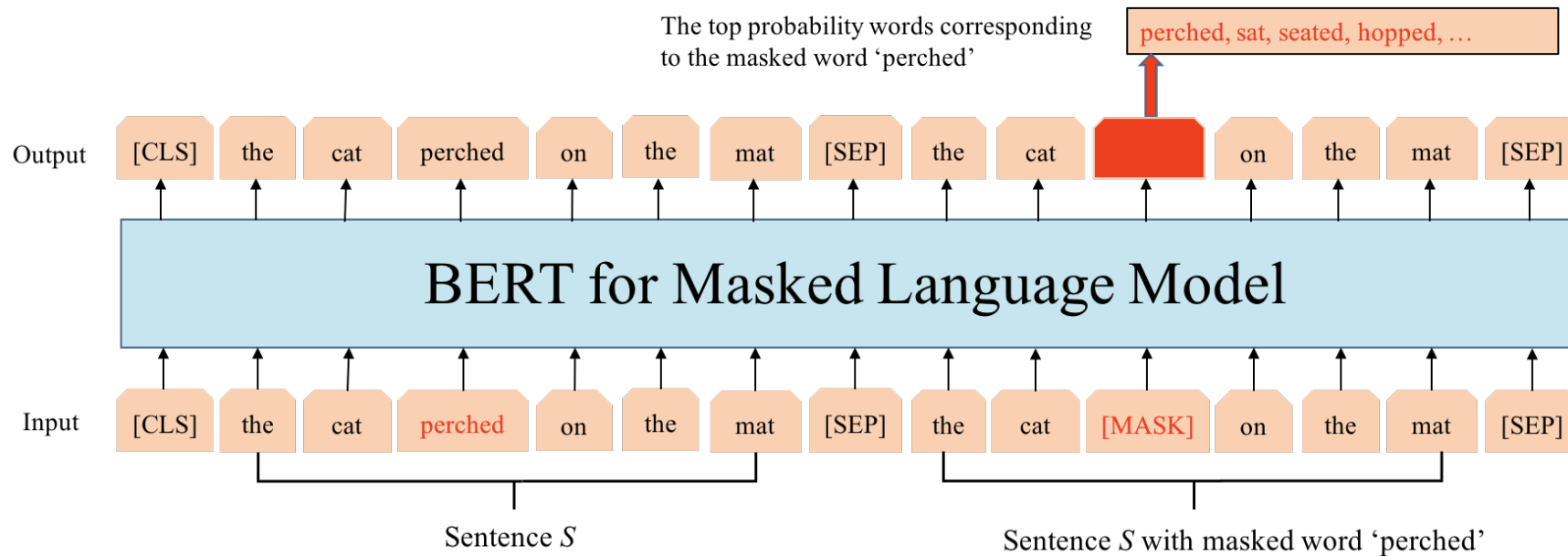
Bidirectional Encoder Representations on Transformers (BERT)

Masked Language Model:



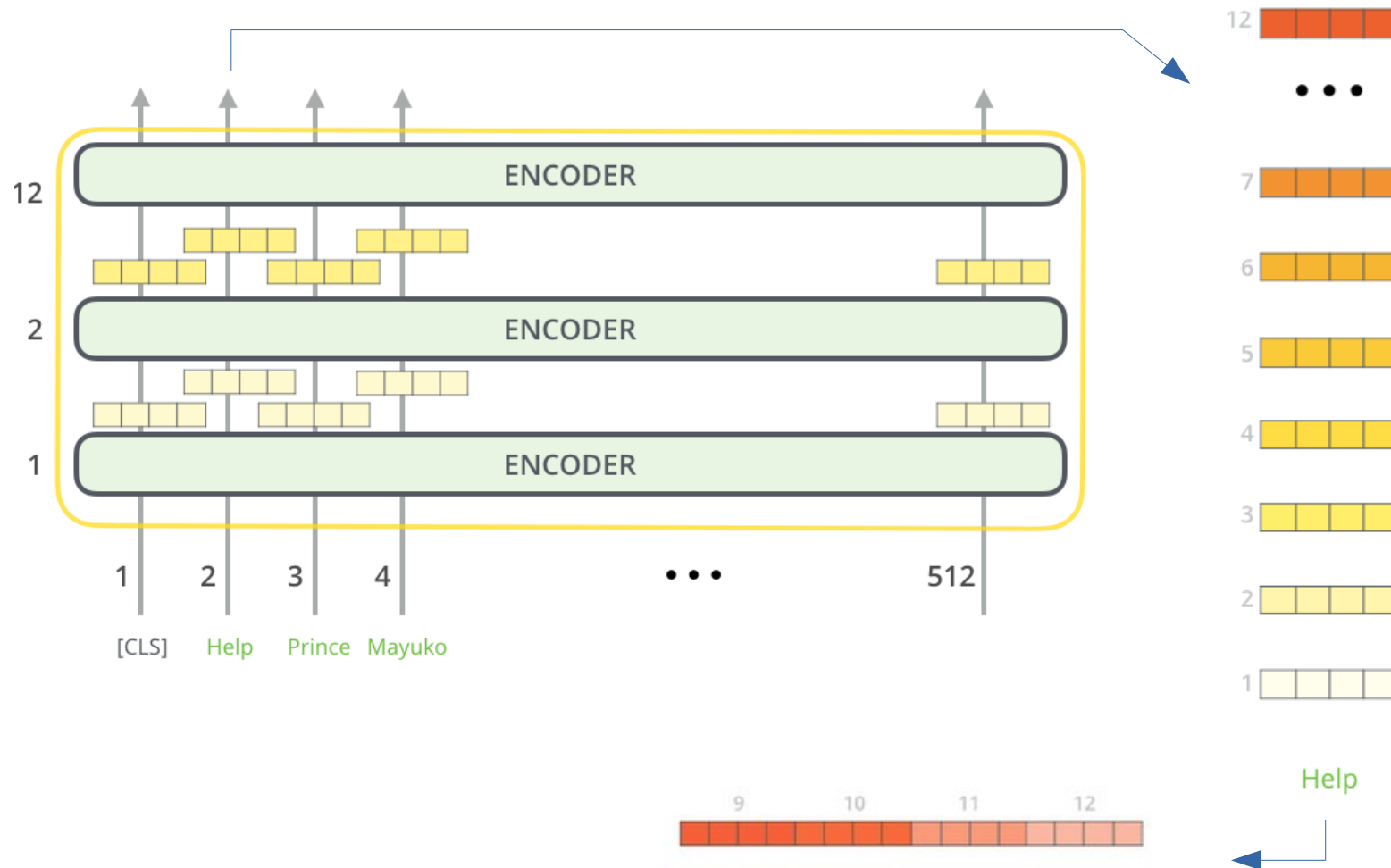
Bidirectional Encoder Representations on Transformers (BERT)

Masked Language Model:



Bidirectional Encoder Representations on Transformers (BERT)

BERT para vectorización de palabras (dependiente del contexto):



Bidirectional Encoder Representations on Transformers (BERT)

Aspectos prácticos

L: bloques de transformer, H: dim de h, A: # multi-heads

- BERT Base: L=12, H=768, A=12 (110M parámetros)
- BERT Large: L=24, H=1024, A=16 (340M parámetros)

Símbolos de BERT para representación de entrada (tokens especiales):

- [CLS]: primer token de cada sentencia
- [SEP]: separador de pares de sentencias

Pretraining: BooksCorpus (800M palabras) y Wikipedia (2,500M palabras)

Fine-tuning para downstream tasks: sobre el modelo preentrenado, se ingestan las entradas y salidas para la tarea específica.

Fine-tuning es menos costoso que pretraining. Por esta razón, BERT proporciona los modelos preentrenados.

Bidirectional Encoder Representations on Transformers (BERT)

Aspectos prácticos

Tareas de pre-entrenamiento

MLM (samplear un token de la sentencia de entrada):

- 80%: reemplazar el token por [MASK].
- 10%: reemplazar el token por un token at random.
- 10%: no reemplazar.

NSP (balance entre IsNext y NotNext)

- 50%: IsNext
- 50%: siguiente sentencia at random

- Training: batches de 256 (sentencias) * 512 tokens (128,000 tokens por batch); 1,000,000 steps (app 40 epochs sobre 3.3 billion word corpus). Implementado en tensor2tensor.

- Tiempo: BERT base en 16 TPU, BERT large en 64 TPU. 4 dias cada uno.