Transformers: Self Attention

Orlando Ramos Flores

Contenido

- Transformers
  - Encoder
  - Self-Attention
  - Multi-Head Attention
  - Positional Encoding
  - Residual connections
  - Decoder

Self-Attention

- La **Attention** opera en queries, keys y values.
  - queries: $q_1, q_2, \ldots, q_T$ donde $q_i \epsilon \mathbb{R}^d$.
  - keys: $k_1, k_2, \ldots, k_T$ donde $k_i \epsilon \mathbb{R}^d$.
  - values: $v_1, v_2, \ldots, v_T$ donde $v_i \epsilon \mathbb{R}^d$.
  - **Nota**: En la práctica el número de queries puede diferir del número de keys y values
- En **self-attention**, las queries, keys y values provienen de la misma fuente de información (la misma oración).
  - Por ejemplo, si la salida de una capa previa es $x_1, \ldots, x_T$ (un vector por palabra) podríamos dejar $v_i = k_i = q_i = x_i$ (usan el mismo vector).

Self-Attention

- La operación de **self-attention** (dot-product):

$$e_{ij} = q_i^T \cdot k_j$$

Calcular afinidades de query-key

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{j'} exp(e_{ij'})}$$

Calcular calcular pesos de atención a partir de las afinidades (softmax)

$$output_i = \sum_j \alpha_{ij} \cdot v_j$$

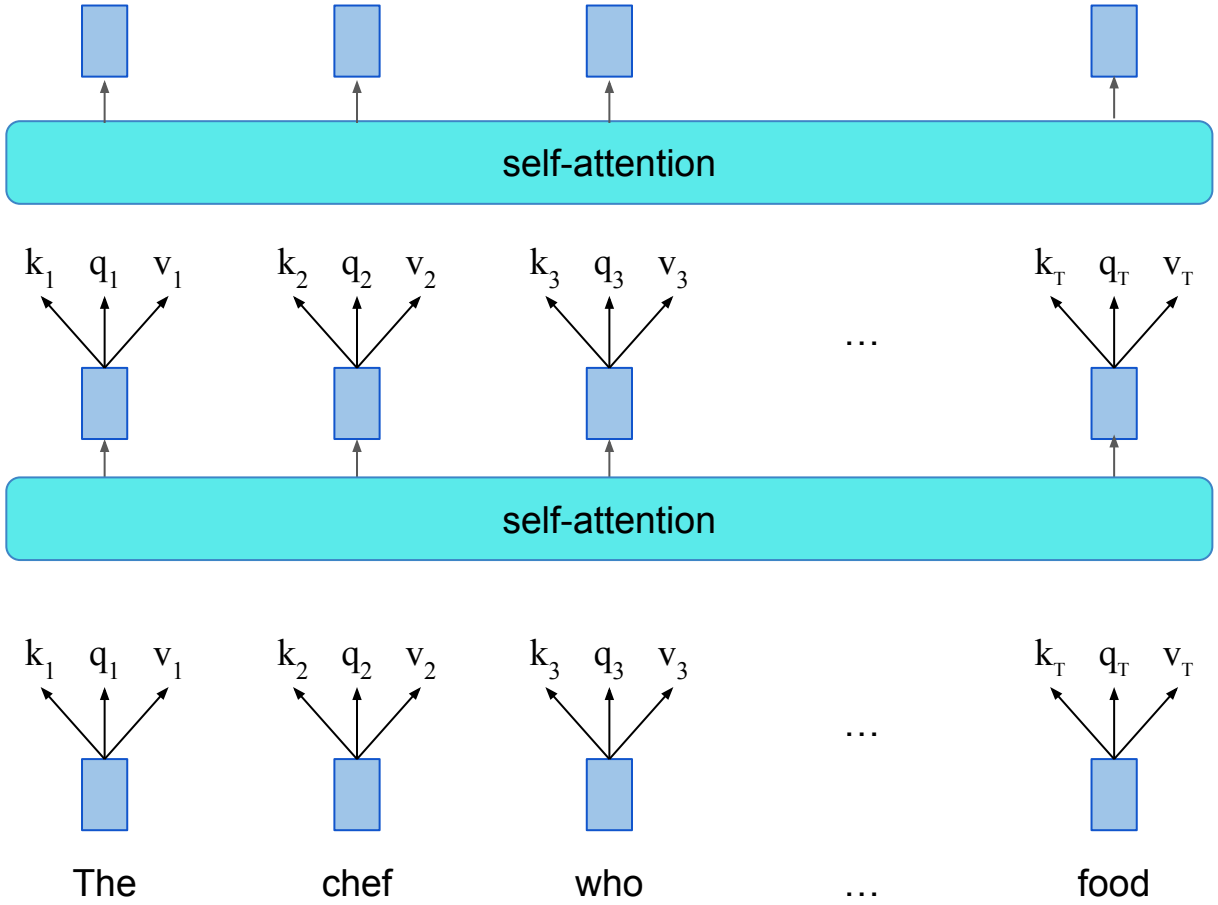
Calcular salidas como suma ponderada de valores

Self-Attention: en bloques

La self-attention es una operación sobre conjuntos.

No tiene una noción inherente de orden.

¡El orden en que aparecen las palabras en las oraciones es importante!



Self-Attention: sequence order

- Dado que la self-attention no se construye con información en orden, se necesita codificar el orden de la oración en las keys, queries y values.
- Se Considera representar cada índice de secuencia como un vector:

$$p_i \epsilon \mathbb{R}^d \text{ para } i \epsilon \{1, 2, 3, \ldots T\} \text{ son vectores de posicion}$$

- Fácil de incorporar esta información en el bloque self-attention: simplemente agregar (sumar) el p_i a las entradas.

Sean $\tilde{v}, \tilde{k}$ y $\tilde{q}$ los valores antiguos de values, keys y queries:

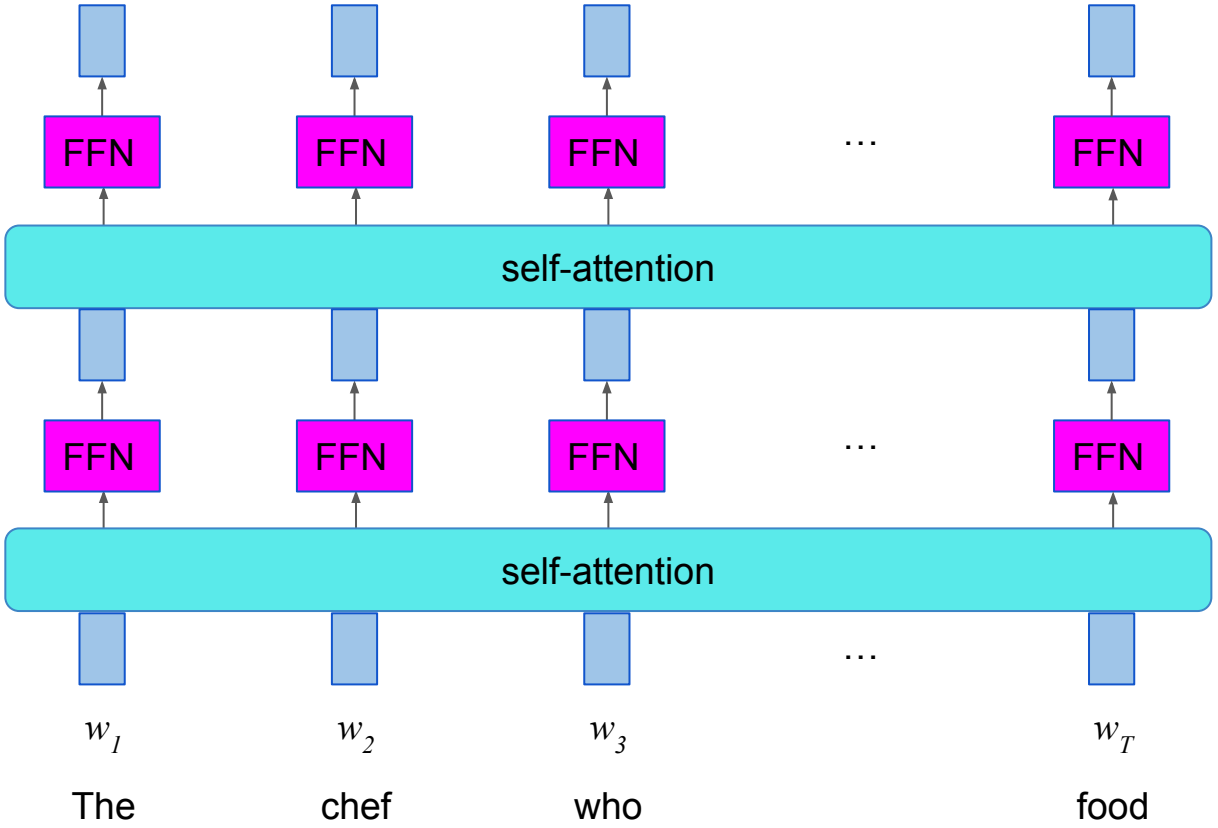
$$v_i = \tilde{v} + p_i$$
$$k_i = \tilde{k} + p_i$$
$$q_i = \tilde{q} + p_i$$

En redes profundas de self-attention, esto se aplica solo en la primera capa. También se podrían concatenar, pero la mayoría de las personas solo los suma

Self-Attention: Agregando no linealidad

Para no solo tener el "promedio ponderado" (weighted average), se agregar una FFN para el post procesamiento de la salida de cada vector

La FFN procesa el resultado de la atención

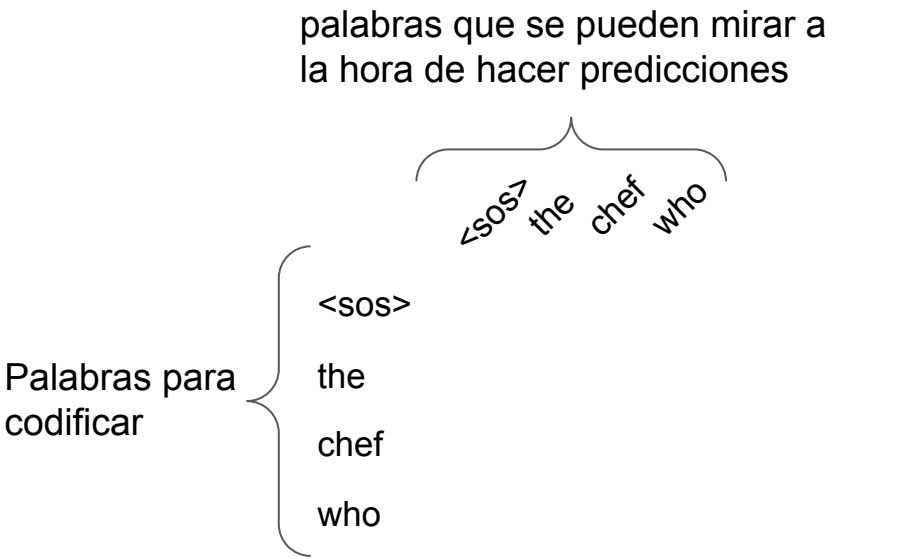


Self-Attention: Enmascarar el futuro

- Para usar la **self-attention** en los **decoders**, se debe asegurar que no podemos mirar al futuro.
- En cada paso de tiempo se van a enmascarar las palabras futuras a través de los propios pesos de atención

$$e_{ij} = \begin{cases} q_i^T \cdot k_j, & k < j \\ -\infty, & k \geq j \end{cases}$$

Self-Attention: Enmascarar el futuro



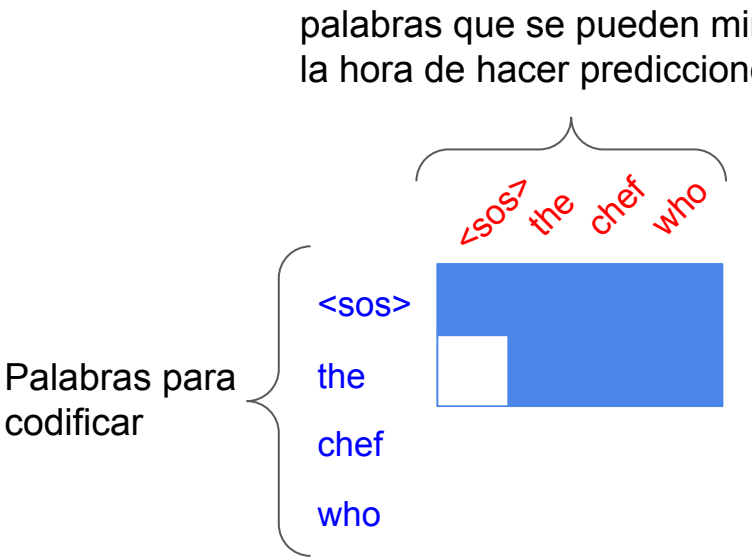
Self-Attention: Enmascarar el futuro

palabras que se pueden mirar a
la hora de hacer predicciones

Palabras para codificar

Para iniciar la predicción se comienza con el token **<sos>**

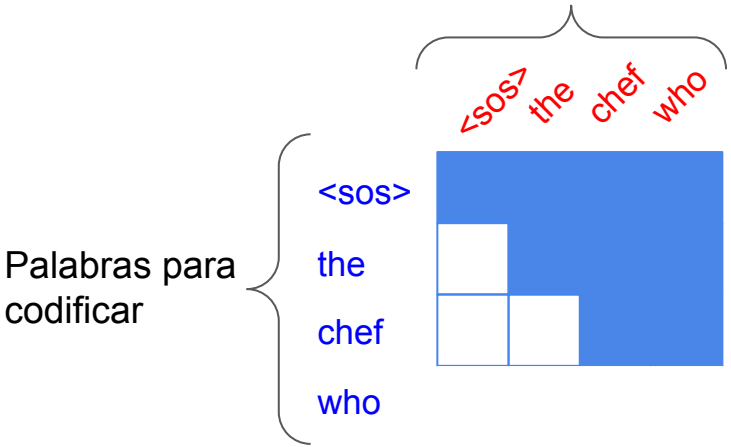
Self-Attention: Enmascarar el futuro



Predecir la primer palabra **the**, no se permite mirar la palabra **the** así como ninguna otra futura palabra. Solo se permite mirar el token **<sos>**

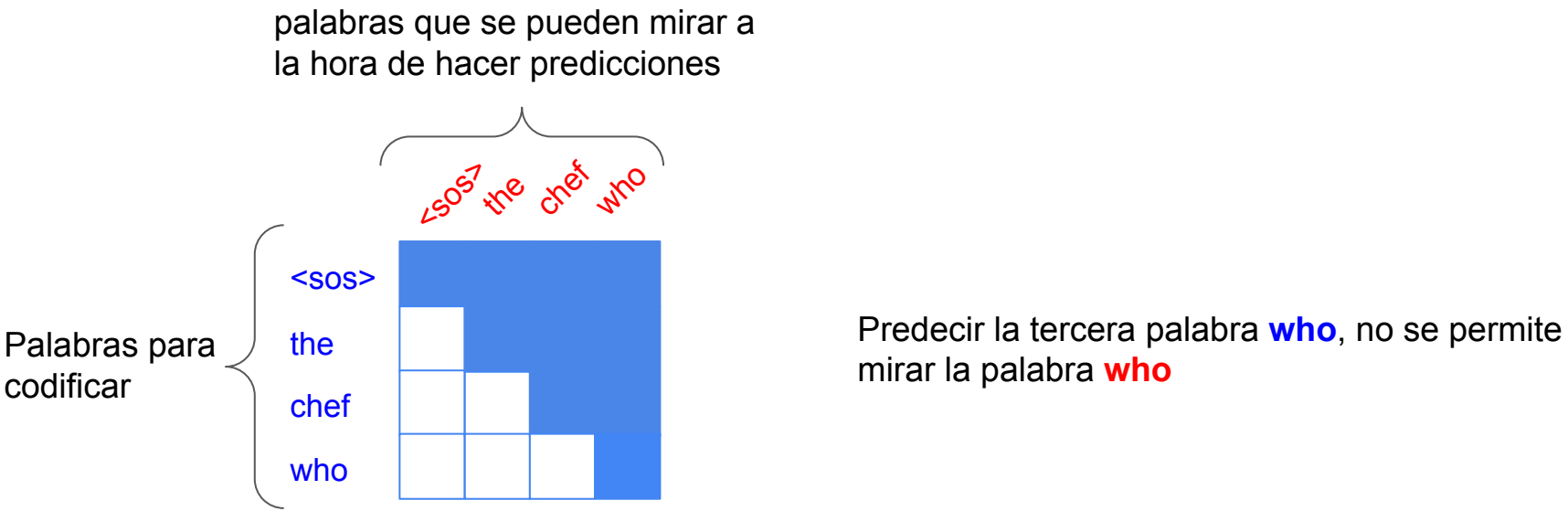
Self-Attention: Enmascarar el futuro

palabras que se pueden mirar a
la hora de hacer predicciones



Predecir la segunda palabra **chef**, no se permite mirar la palabra **chef**

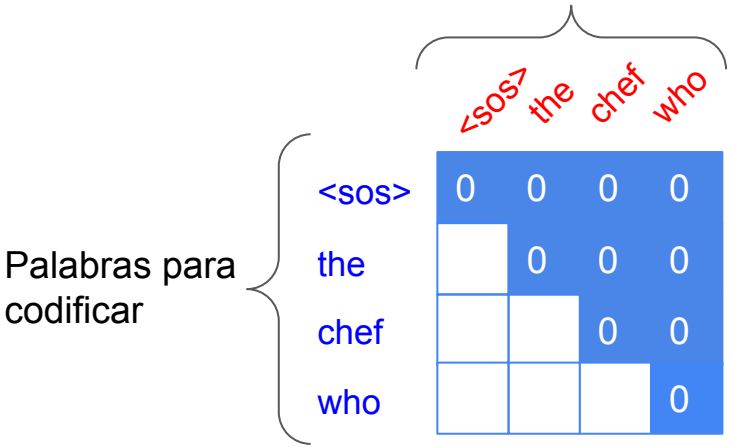
Self-Attention: Enmascarar el futuro



Predecir la tercera palabra **who**, no se permite mirar la palabra **who**

Self-Attention: Enmascarar el futuro

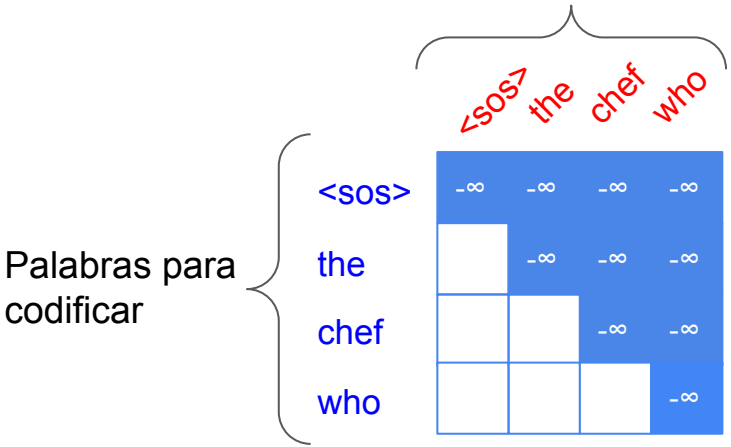
palabras que se pueden mirar a
la hora de hacer predicciones



Lo que se desea es que todos los pesos en el área sombreada sean ceros (0)

Self-Attention: Enmascarar el futuro

palabras que se pueden mirar a
la hora de hacer predicciones



Así cuando se calcule la afinidad se suman infinito negativo a todos ellos.

Y eso garantiza que no se pueda mirar hacia el futuro

Multi-Head Attention

- ¿Qué sucede si se desea mirar en múltiples lugares en la oración a la vez?
- Entonces se tienen que definir múltiples “cabezas” de atención a través de múltiples matrices Q, K, V que codifican diferentes cosas sobre X, todas aprenden diferentes *transformaciones*.
- Sea, $Q_l, K_l, V_l \in \mathbb{R}^{d \times (d/h)}$ donde h es el número de cabezas de atención, y l va del rango de 1 a h.
- Así, cada cabeza de atención realiza la atención de forma independiente: $output_l = softmax(XQ_l K_l^T X^T) * XV_l$ donde $output \in \mathbb{R}^{d/h}$
- Finalmente, todas las outputs de las cabezas se combinan:

  $output = Y[output_1, output_2, ..., output_h]$ donde $Y \in \mathbb{R}^{d \times d}$n

- Cada cabeza puede "mirar" cosas diferentes y construir vectores de valores de manera diferente

Recursos sobre transformers

1. Stanford CS224N NLP with Deep Learning | Winter 2021 | Lecture 9 - Self- Attention and Transformer
2. CS 182: Lecture 12: Part 1: Transformers
3. Attention? Attention!
4. Transformers from Scratch
5. MIT 6.S191: Recurrent Neural Networks and Transformers