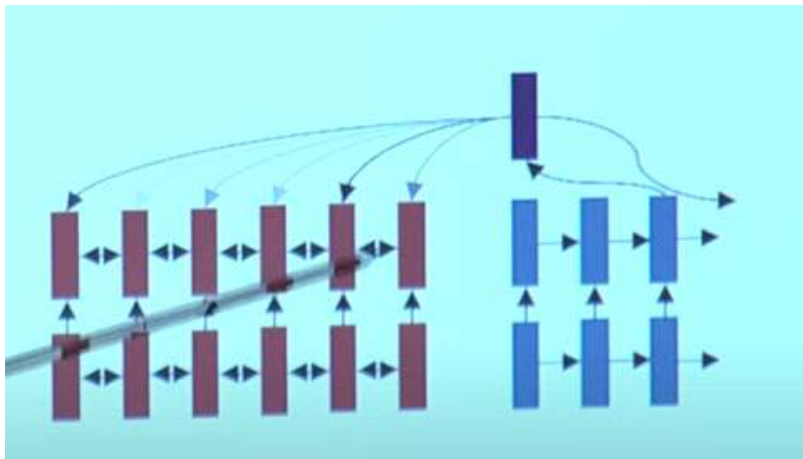


Lec9. NLP processing with Deep learning - Self-Attention and Transformers

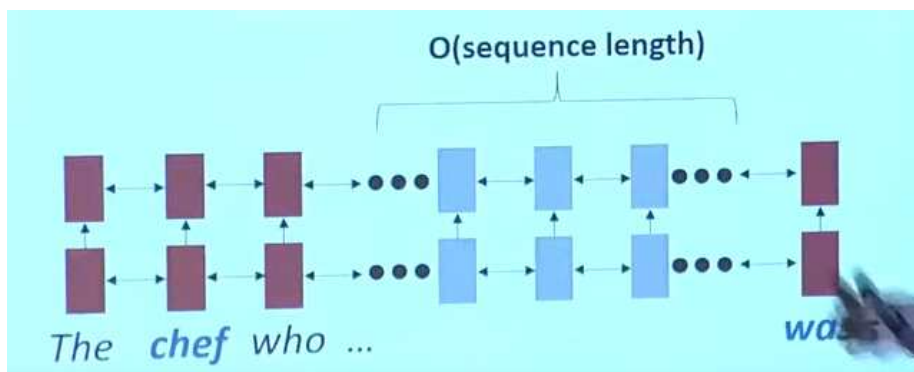
1. recurrent models for NLP

LSTM → RNN → Attention

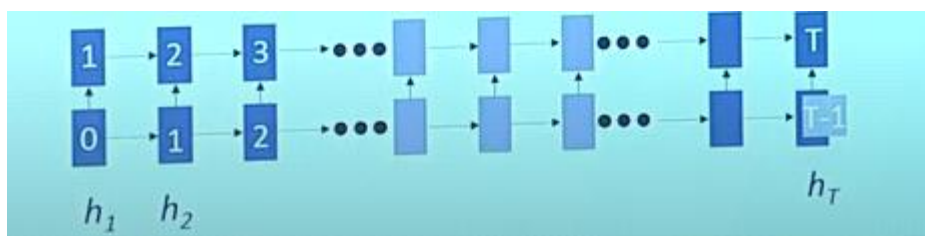


Issues with recurrent models: Linear interaction distance

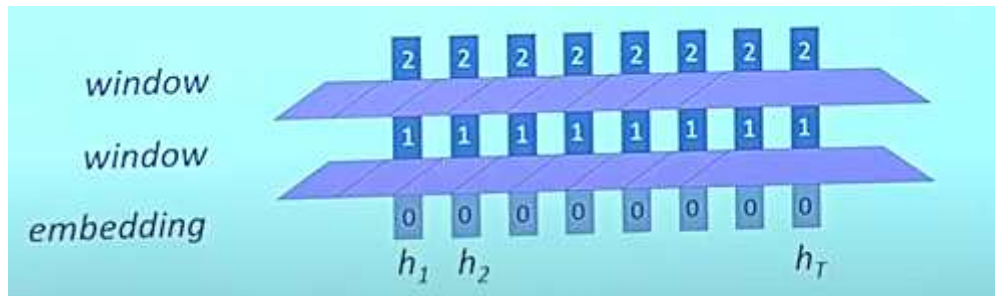
- RNNs are unrolled “left to right”



- O steps for distant word pairs to interact means:
- Hard to learn long-distance-dependencies(기울기 문제)
- Lack of parallelizability

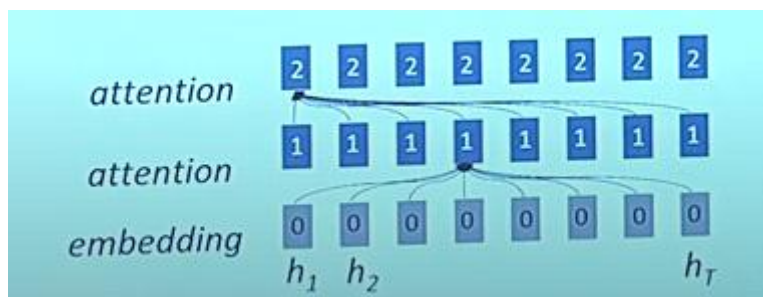


- Word window models aggregate local contexts



- How about attention?

- Attention treats each word's representation as a query to access and incorporate information from a set of values



- Self-Attention

• The (dot product) self-attention operation is as follows:

$$e_{ij} = q_i^\top k_j$$

Compute key-query affinities

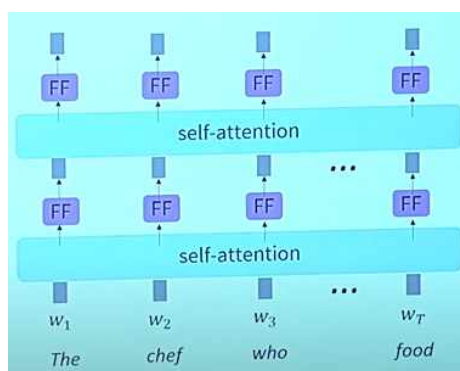
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

Compute attention weights from affinities

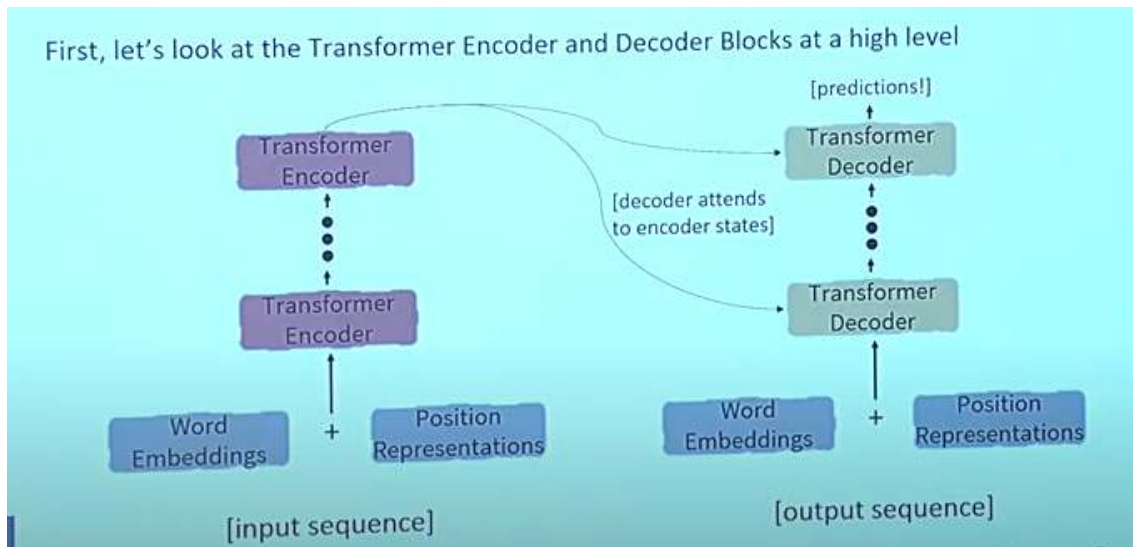
$$\text{output}_i = \sum_j \alpha_{ij} v_j$$

Compute outputs as weighted sum of values

- Fixing the first self-attention problem: sequence order(consider sequence index as a vector)
- Position representation vectors through sinusoids
- Position representation vectors learned from scratch
- Adding nonlinearities in self-attention



2. Introducing Transformer model



- The Transformer Encoder: Key-Query-Value Attention

The Transformer Encoder: Key-Query-Value Attention

- Let's look at how key-query-value attention is computed, in matrices.
 - Let $X = [x_1; \dots; x_T] \in \mathbb{R}^{T \times d}$ be the concatenation of input vectors.
 - First, note that $XK \in \mathbb{R}^{T \times d}$, $XQ \in \mathbb{R}^{T \times d}$, $XV \in \mathbb{R}^{T \times d}$.
 - The output is defined as $\text{output} = \text{softmax}(XQ(XK)^T) \times XV$.

First, take the query-key dot products in one matrix multiplication: $XQ(XK)^T$

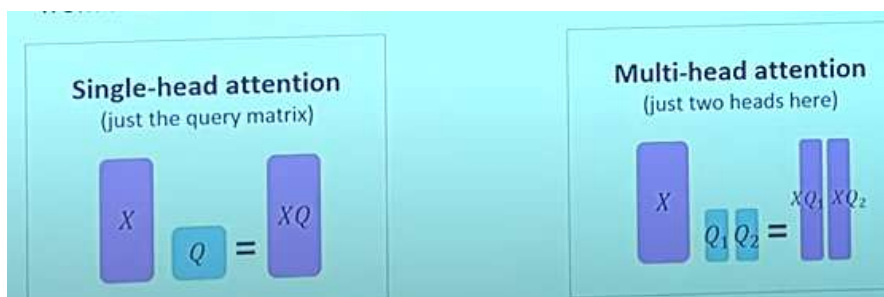
$$XQ \cdot K^T X^T = XQK^T X^T \in \mathbb{R}^{T \times T}$$

All pairs of attention scores!

Next, softmax, and compute the weighted average with another matrix multiplication.

$$\text{softmax}\left(XQK^T X^T\right) \cdot XV = \text{output} \in \mathbb{R}^{T \times d}$$

- The Transformer Encoder: Multi-headed attention

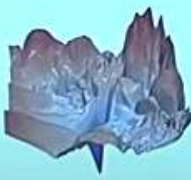



- The Transformer Encoder: Residual connections

- **Residual connections** are a trick to help models train better.
 - Instead of $X^{(i)} = \text{Layer}(X^{(i-1)})$ (where i represents the layer)

$X^{(i-1)} \rightarrow \boxed{\text{Layer}} \rightarrow X^{(i)}$
 - We let $X^{(i)} = X^{(i-1)} + \text{Layer}(X^{(i-1)})$ (so we only have to learn “the residual” from the previous layer)

$X^{(i-1)} \rightarrow \boxed{\text{Layer}} \xrightarrow{+} X^{(i)}$
 - Residual connections are thought to make the loss landscape considerably smoother (easier to train!)

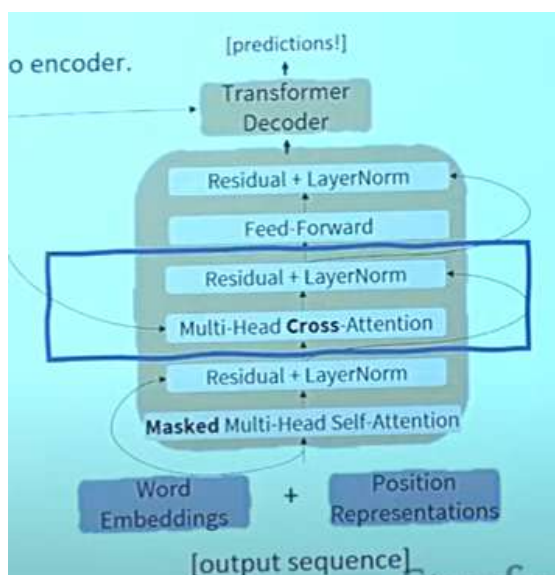



[no residuals]

[residuals]

- The Transformer Encoder: Scaled Dot Product

- The Transformer Encoder: Cross-attention



First, take the query-key dot products in one matrix multiplication: $ZQ(HK)^T$

ZQ

$K^T H^T$

=

$ZQK^T H^T$

$\in \mathbb{R}^{T \times T}$

All pairs of attention scores!

Next, softmax, and compute the weighted average with another

softmax

$ZQK^T H^T$

HV

=

$\text{output} \in \mathbb{R}^{T \times d}$