Transformer

Artificial Intelligence @ Allegheny College

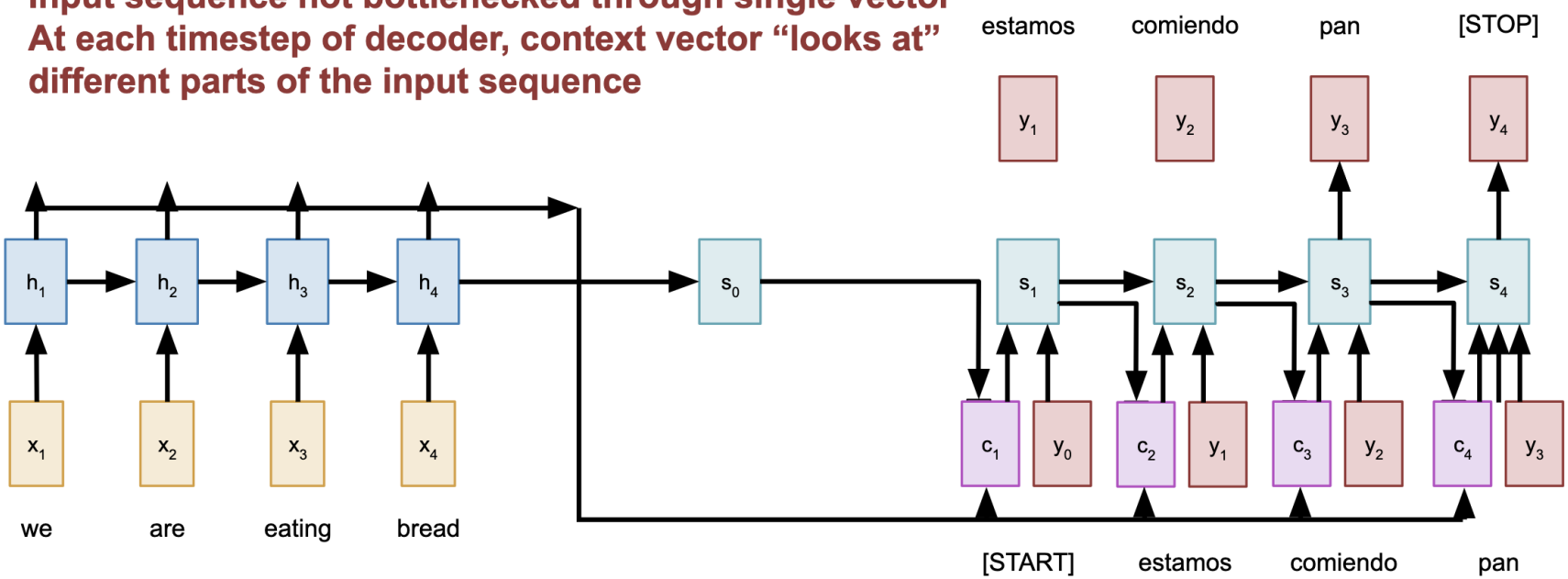
Janyl Jumadinova

April 12, 2023

RNN and Attention - Sequence to Sequence

Use a different context vector in each timestep of decoder

- **Input sequence not bottlenecked through single vector**
- **At each timestep of decoder, context vector "looks at" different parts of the input sequence**



Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

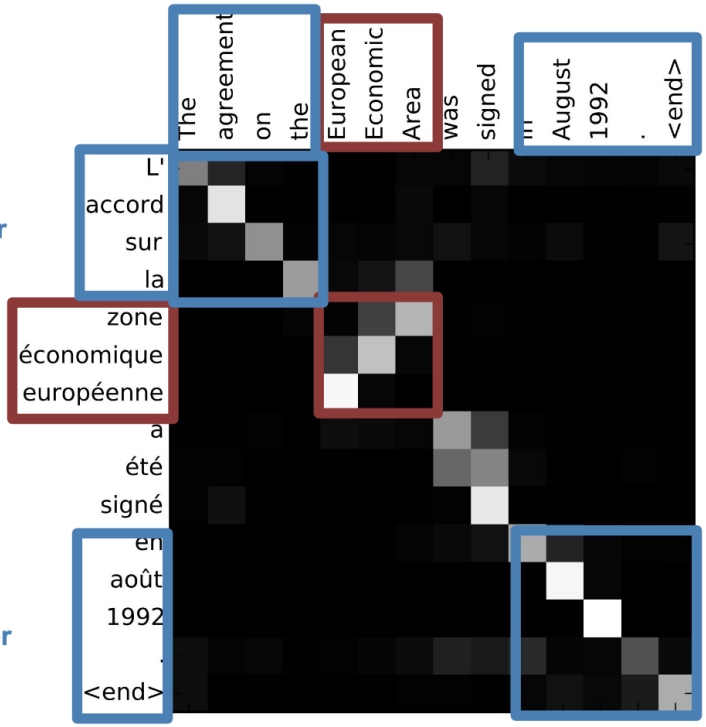
RNN and Attention - Sequence to Sequence

Example: English to French translation

Input: "The agreement on the European Economic Area was signed in August 1992."

Output: "L'accord sur la zone économique européenne a été signé en août 1992."

Visualize attention weights $a_{t,i}$



Diagonal attention means words correspond in order

Attention figures out different word orders

Diagonal attention means words correspond in order

Bahdanau et al, "Neural machine translation by jointly learning to align and translate", ICLR 2015

Self-Attention

- Use the concept of attention to encode sequences instead of using RNNs.
- Both the encoder and decoder use attention mechanisms.

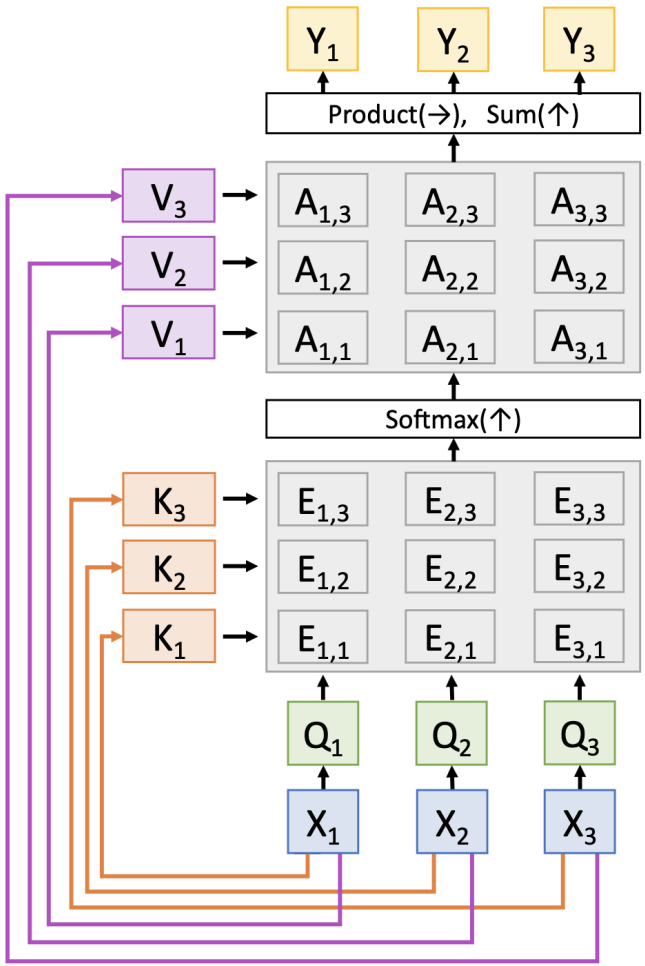
Self-Attention

Self-Attention Layer

One **query** per **input vector**

Inputs:
Input vectors: X (Shape: $N_X \times D_X$)
Key matrix: W_K (Shape: $D_X \times D_Q$)
Value matrix: W_V (Shape: $D_X \times D_V$)
Query matrix: W_Q (Shape: $D_X \times D_Q$)

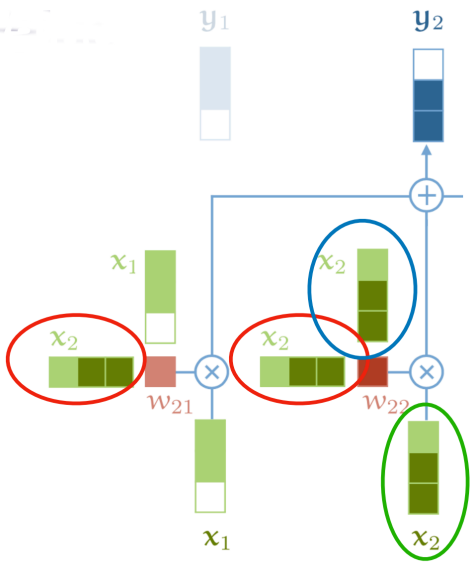
Computation:
Query vectors: $Q = XW_Q$
Key vectors: $K = XW_K$ (Shape: $N_X \times D_Q$)
Value Vectors: $V = XW_V$ (Shape: $N_X \times D_V$)
Similarities: $E = QK^T$ (Shape: $N_X \times N_X$) $E_{i,j} = Q_i \cdot K_j / \text{sqrt}(D_Q)$
Attention weights: $A = \text{softmax}(E, \text{dim}=1)$ (Shape: $N_X \times N_X$)
Output vectors: $Y = AV$ (Shape: $N_X \times D_V$) $Y_i = \sum_j A_{i,j} V_j$



Self-Attention Layer

Every input vector:

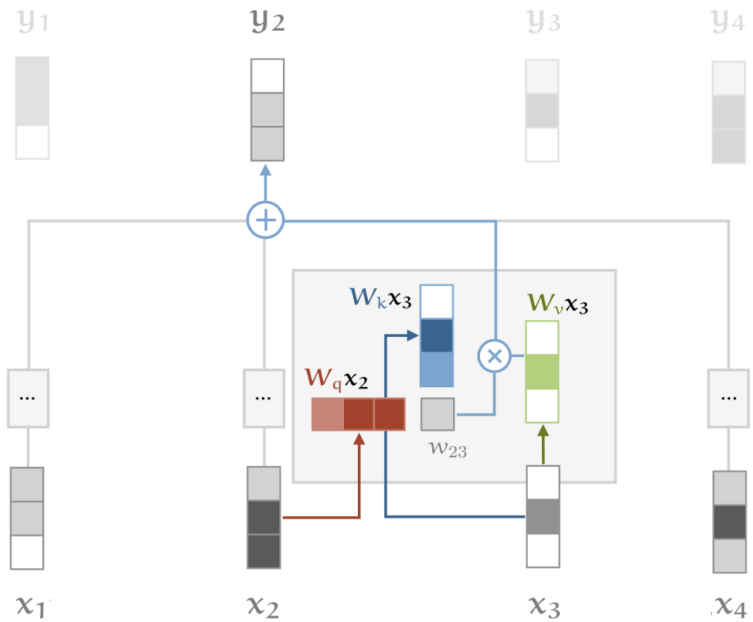
- compared to every other vector to compute attention weights for its own output (query)
- compared to every other vector to compute attention weight w_{ij} for output (key)
- summed with other vectors to form the result of the attention weighted sum (value)



Ref: `https://peterbloem.nl/blog/transformers`

Self-Attention

- Process each input vector to fulfill the three roles with matrix multiplication.
- Learning the matrices $->$ learning attention

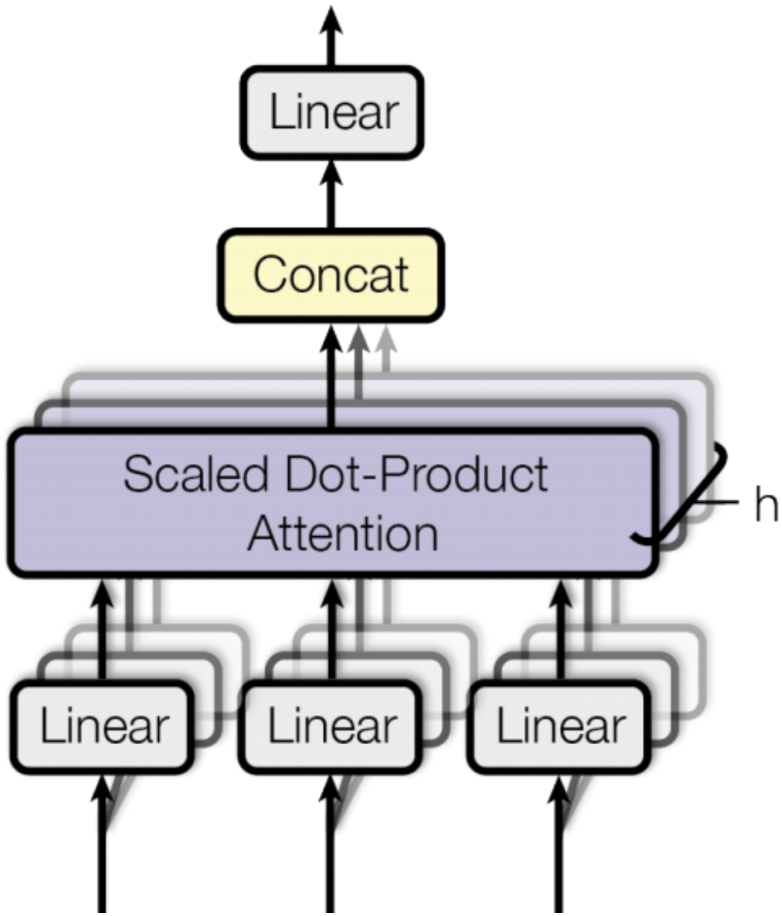


Softmax operation over the weights is not shown above. Ref:

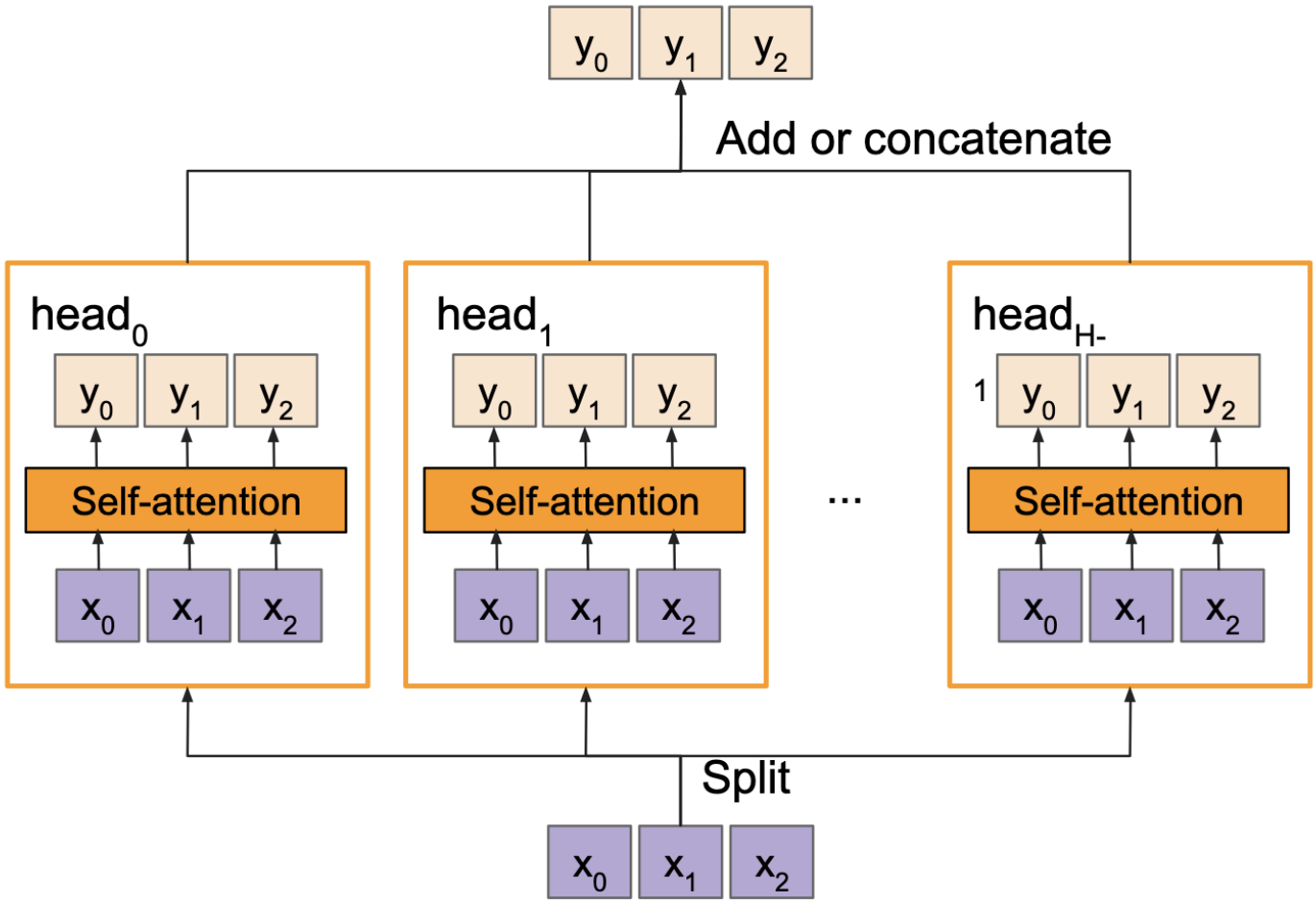
`https://peterbloem.nl/blog/transformers`

Multi-head Attention

- Multiple "heads" of attention means learning different sets of W_q, W_k, and W_v matrices simultaneously.
- Implemented as a single matrix.

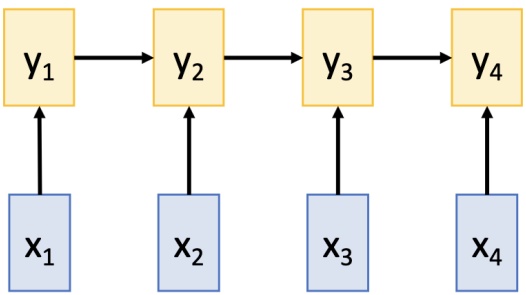


Multi-head Self-Attention



Three Ways of Processing Sequences

Recurrent Neural Network

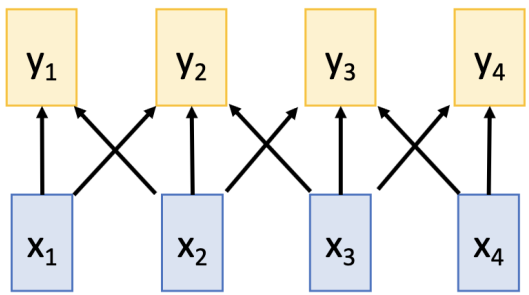


Works on **Ordered Sequences**

(+) Good at long sequences: After one RNN layer, h_T "sees" the whole sequence

(-) Not parallelizable: need to compute hidden states sequentially

1D Convolution

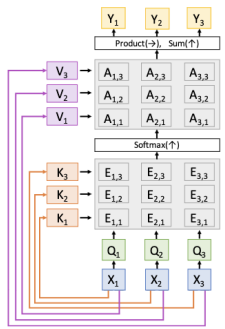


Works on **Multidimensional Grids**

(-) Bad at long sequences: Need to stack many conv layers for outputs to "see" the whole sequence

(+) Highly parallel: Each output can be computed in parallel

Self-Attention



Works on **Sets of Vectors**

(-) Good at long sequences: after one self-attention layer, each output "sees" all inputs!

(+) Highly parallel: Each output can be computed in parallel

(-) Very memory intensive

Transformer

Any architecture designed to process a connected set of unit - such as the tokens in a sequence or the pixels in an image - where the only interaction between units is through self-attention.

Transformer

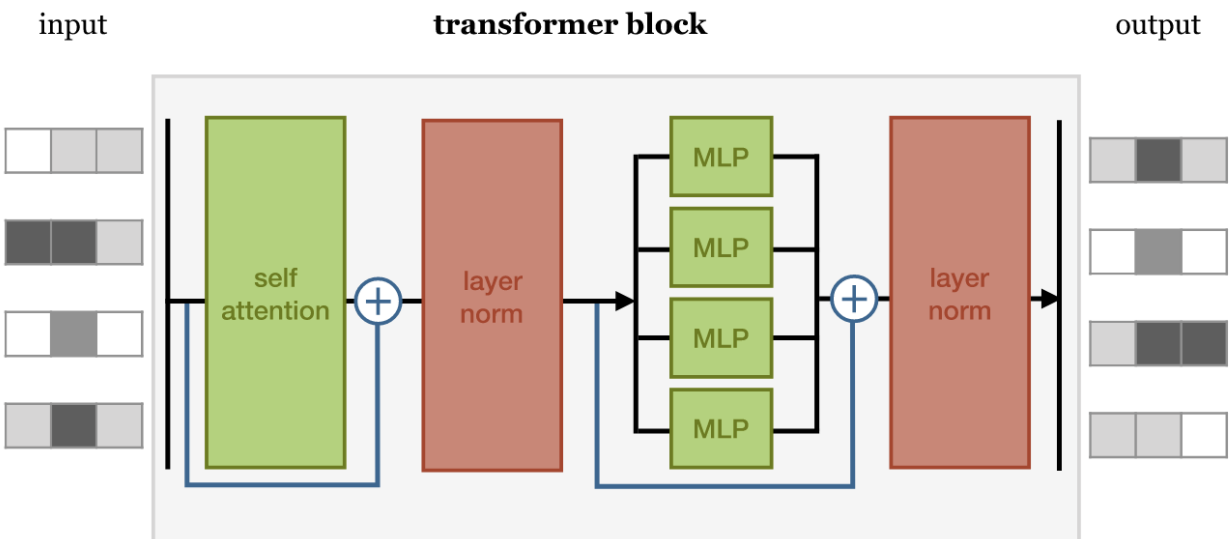
Any architecture designed to process a connected set of unit - such as the tokens in a sequence or the pixels in an image - where the only interaction between units is through self-attention.

Transformer

is a sequence of transformer blocks.
"Attention is all you need" paper by Vaswani et al: 12 blocks, D_Q(query dimension)=512, 6 heads

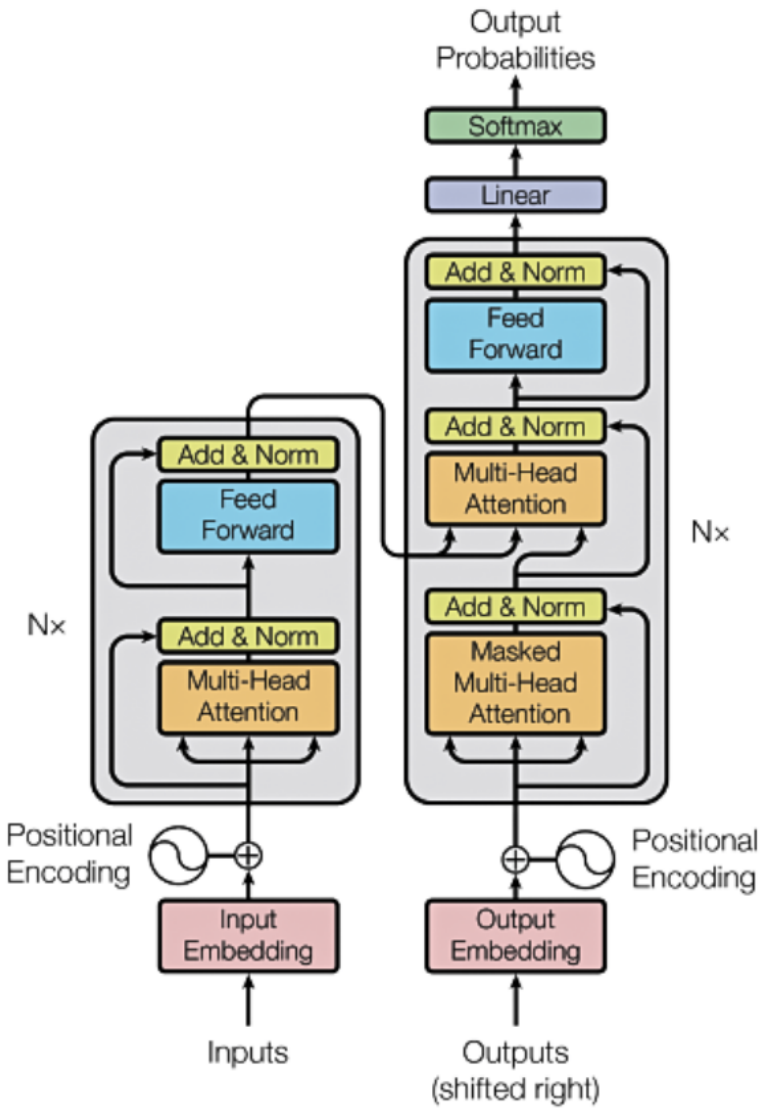
Transformer

Self-attention layer $->$ Layer normalization $->$ Dense layer

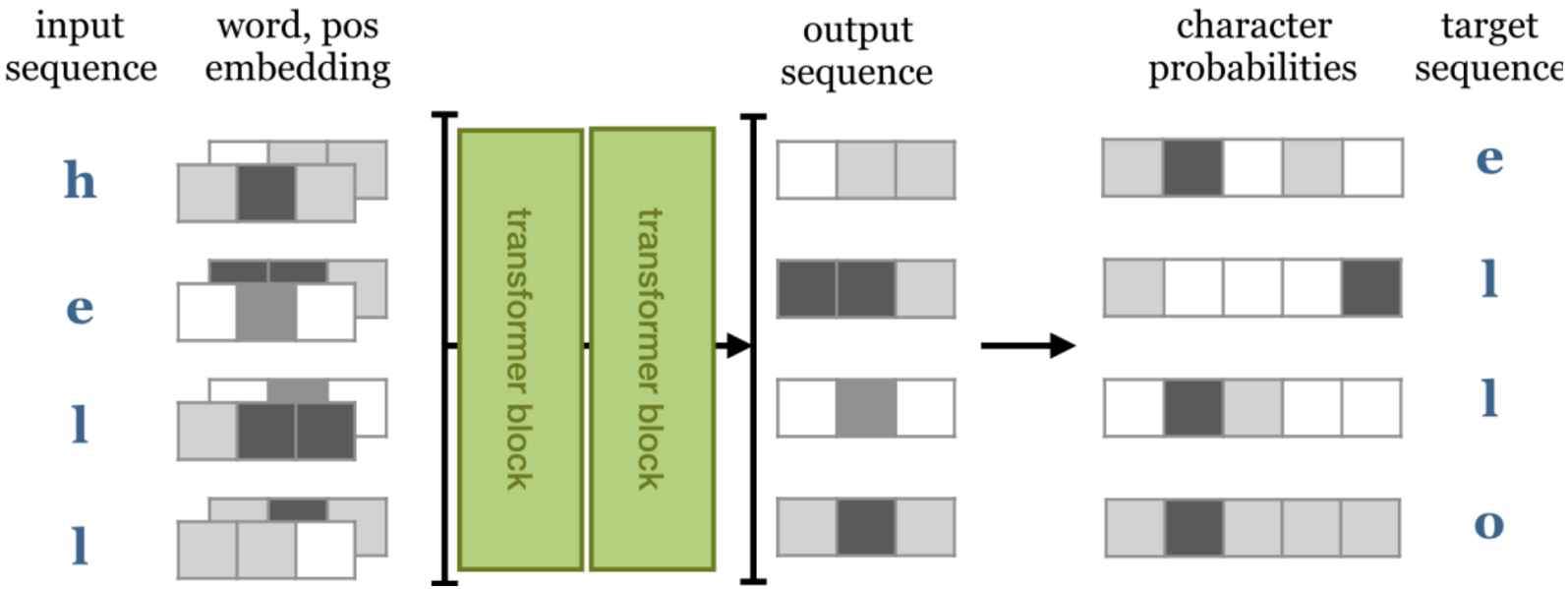


Layer normalization ref: https://arxiv.org/pdf/1803.08494.pdf

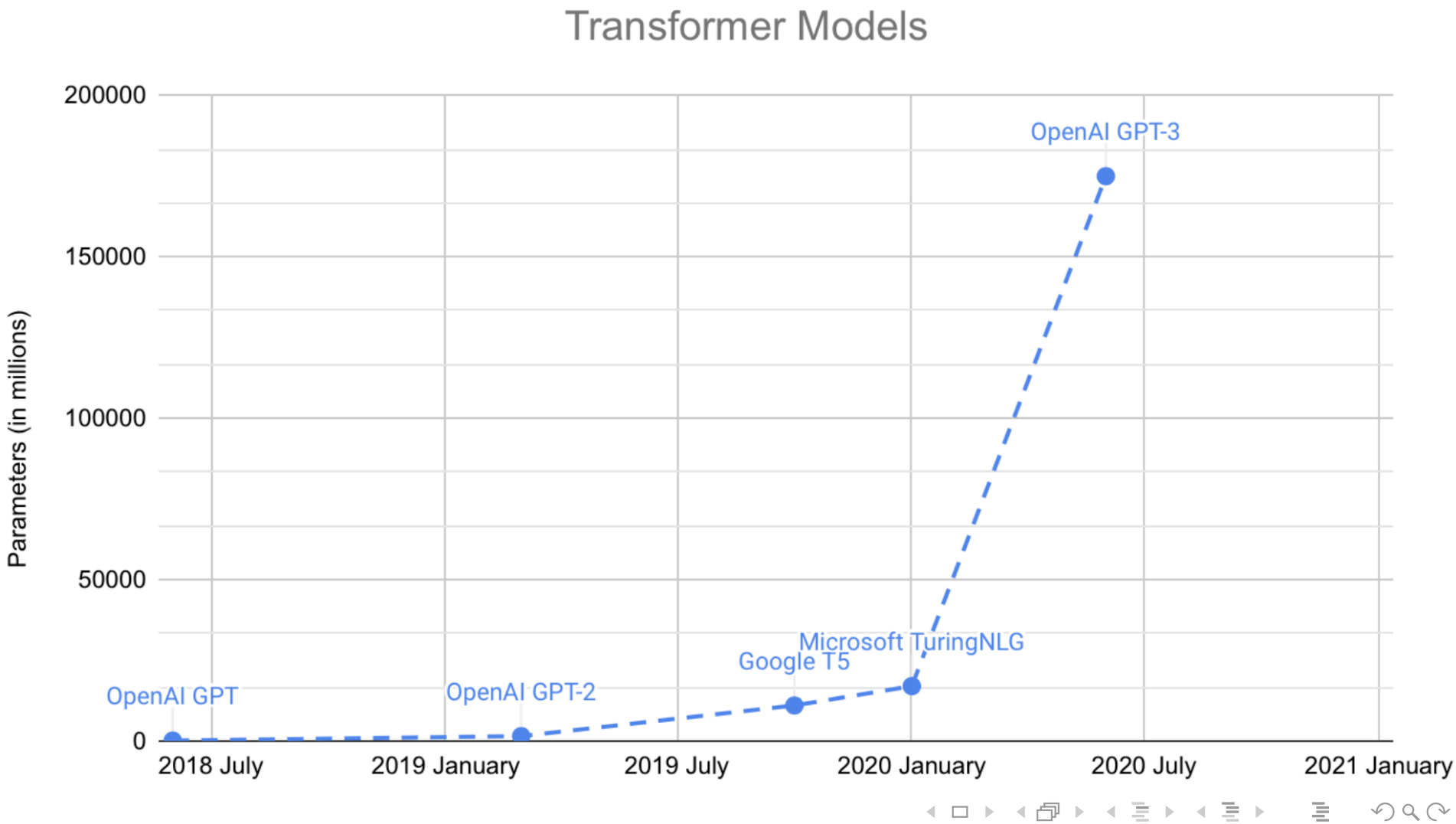
Transformer



Transformer



Transformer Parameters



Activity 15.2: Commit 2

After walking through the tutorial, modify README of activity15 repository to add the following:

- A section called *Data* that explains the data used in the tutorial and how it was pre-processed.
- A section called *Transformer Components* that gives a brief overview of transformer model in your own words and then in separate multiple subsections explains each component of the transformer used in the tutorial in general terms.

The explanation of each section and subsection should be brief (short paragraph) and should be understandable by a general CS student, not only students in this class.