

Generative AI - Large Language Models

College of Computing

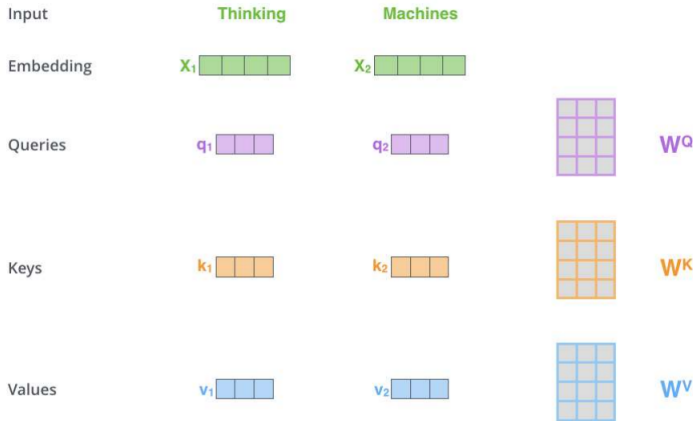
Weekly Lab - Week 4

February 13, 2026

Aim of the Tutorial

- In this weekly lab, we aim to:
 - Dive into the details of attention mechanism
 - Implement self-attention layer and build your own transformer model from scratch if interested
 - Use the public resources available that can help in implementing. Some of these are:
 - ▶ [MLFSTransformerTutorial](#)
 - ▶ [Attentionisallyouneed](#)
 - ▶ [Tensor2Tensor](#)
- The expected outcome is:
 - Match the attention's implementation with the fundamental concepts learned
 - Implement a transformer architecture on a benchmark task and discuss the effect of different hyperparameters (batch size, learning rate,...) on the performance

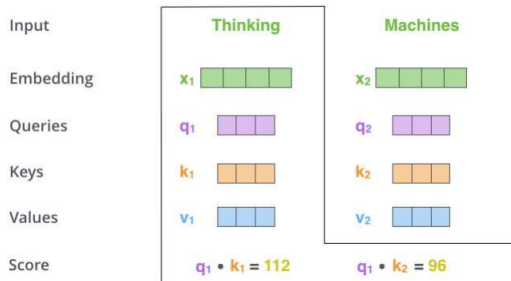
Attention Illustration - Reminder



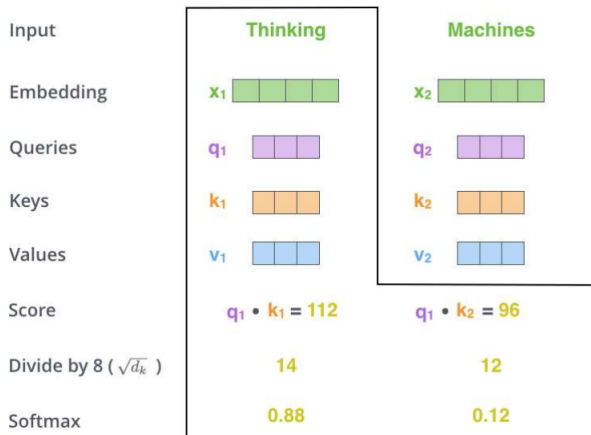
Multiplying x_1 by the W^Q weight matrix produces q_1 , the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

Attention Illustration - Reminder

- The embedding and encoder input/output vectors have a dimensionality of 512
- The created query, key and value vectors have a dimensionality of 64



Attention Illustration - Reminder



Attention Illustration - Reminder

Input

Embedding

Queries

Keys

Values

Score

Divide by 8 ($\sqrt{d_k}$)

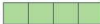
Softmax

Softmax

X
Value

Sum

Thinking

x_1 

q_1 

k_1 

v_1 

$q_1 \cdot k_1 = 112$


14

0.88

v_1 

z_1 

Machines

x_2 

q_2 

k_2 

v_2 

$q_2 \cdot k_2 = 96$

12

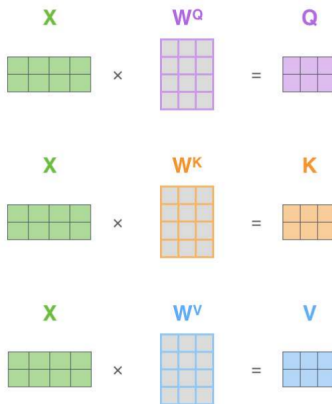
0.12

v_2 

z_2 

Attention Illustration - Reminder

- Matrix Calculation of Self-Attention



Every row in the X matrix corresponds to a word in the input sentence. We again see the difference in size of the embedding vector (512, or 4 boxes in the figure), and the $q/k/v$ vectors (64, or 3 boxes in the figure)

Attention Illustration - Reminder

- Matrix Calculation of Self-Attention

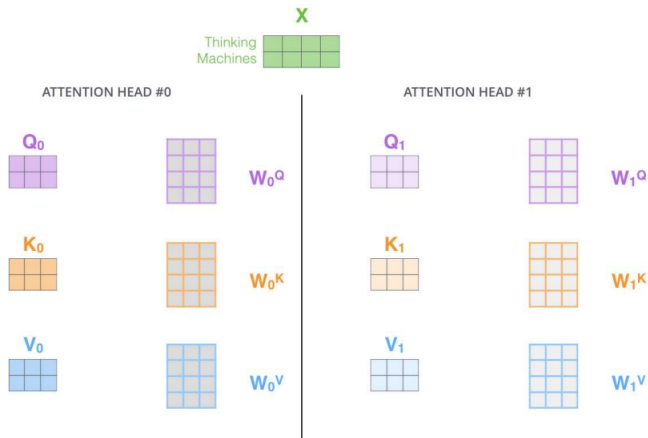
$$\text{softmax}\left(\frac{\begin{matrix} \text{Q} \\ \begin{matrix} \text{3x3 grid} \end{matrix} \end{matrix} \times \begin{matrix} \text{K}^T \\ \begin{matrix} \text{3x3 grid} \end{matrix} \end{matrix}}{\sqrt{d_k}}\right) \begin{matrix} \text{V} \\ \begin{matrix} \text{3x3 grid} \end{matrix} \end{matrix}$$
$$= \begin{matrix} \text{Z} \\ \begin{matrix} \text{3x3 grid} \end{matrix} \end{matrix}$$

The self-attention calculation in matrix form

► jalammar

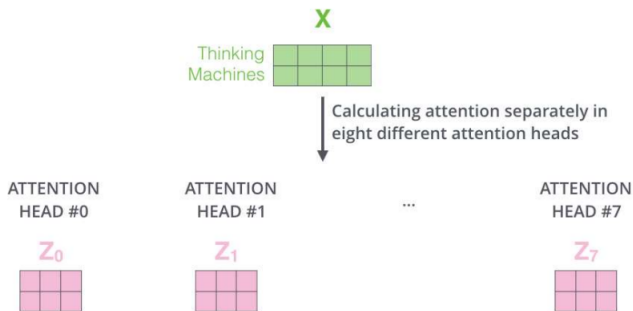
Attention Illustration - Reminder

- Matrix Calculation of Self-Attention



Attention Illustration - Reminder

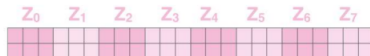
- Matrix Calculation of Self-Attention



Attention Illustration - Reminder

- Matrix Calculation of Self-Attention

1) Concatenate all the attention heads



2) Multiply with a weight matrix W^O that was trained jointly with the model

X



3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN



Attention Illustration - Reminder

• Matrix Calculation of Self-Attention

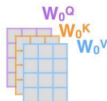
1) This is our input sentence*

Thinking
Machines

2) We embed each word*



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



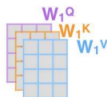
4) Calculate attention using the resulting $Q/K/V$ matrices



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



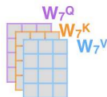
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one



...

...

...



W^O

