

Generative AI - Large Language Models

College of Computing

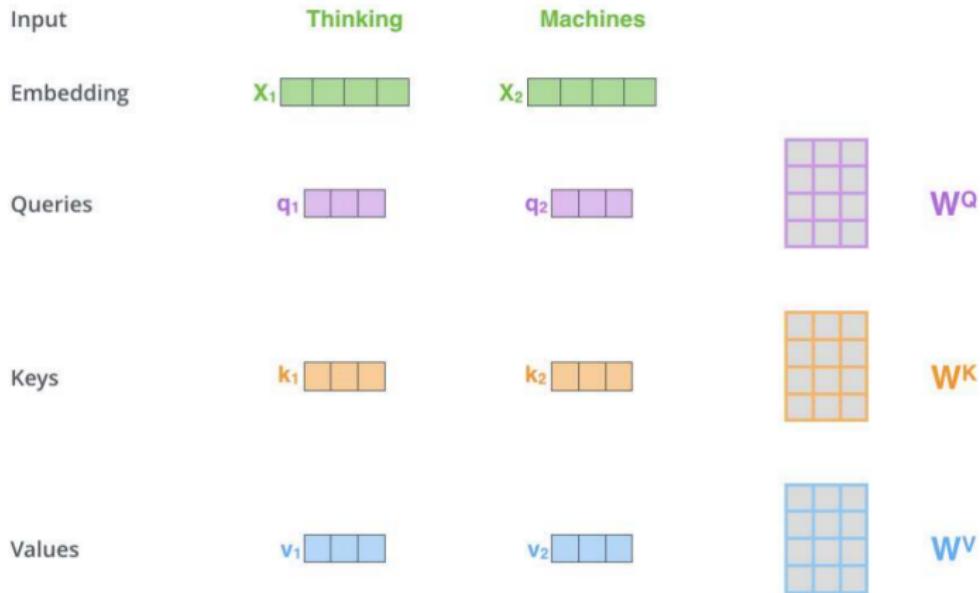
Weekly Lab - Week 4

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

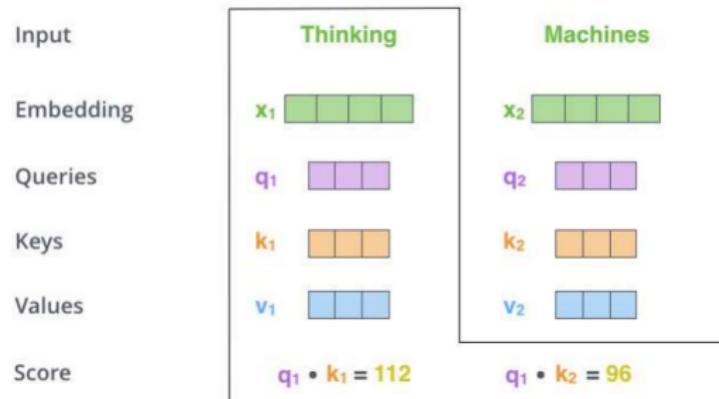
Attention Illustration - Reminder



Multiplying x_1 by the WQ 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



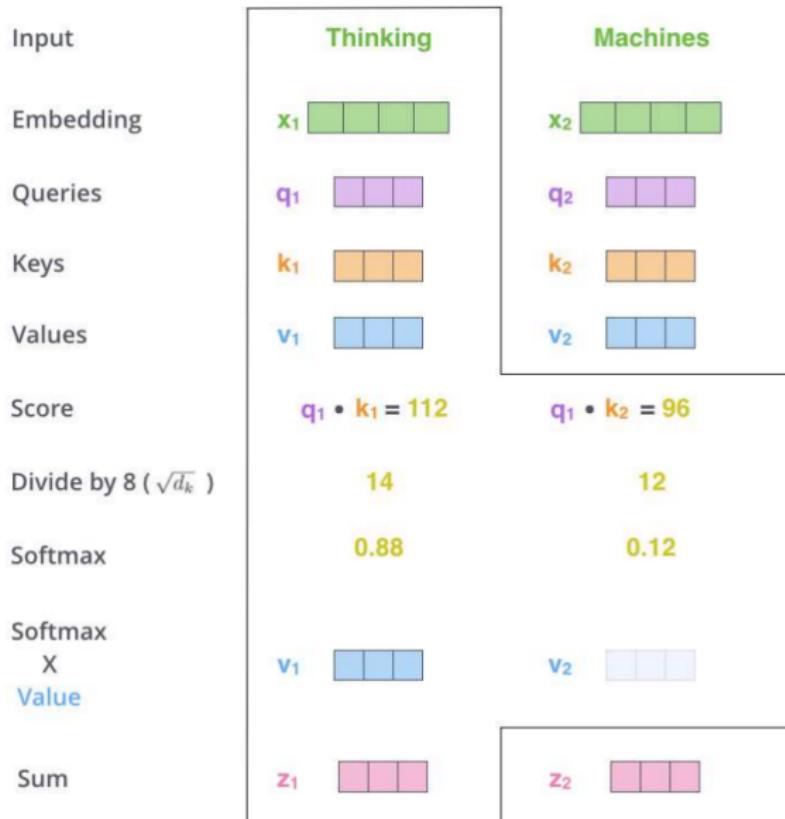
▶ jalamar

Attention Illustration - Reminder

Input	Thinking		Machines	
Embedding	x_1		x_2	
Queries	q_1		q_2	
Keys	k_1		k_2	
Values	v_1		v_2	
Score	$q_1 \cdot k_1 = 112$		$q_1 \cdot k_2 = 96$	
Divide by 8 ($\sqrt{d_k}$)	14		12	
Softmax	0.88		0.12	

▶ jalammr

Attention Illustration - Reminder



Attention Illustration - Reminder

- Matrix Calculation of Self-Attention

$$\mathbf{X} \times \mathbf{W}^Q = \mathbf{Q}$$

A diagram illustrating the calculation of query vectors (Q) from input vectors (X). It shows a green 4x4 matrix X multiplied by a purple 4x4 weight matrix W^Q , resulting in a purple 4x4 matrix Q .

$$\mathbf{X} \times \mathbf{W}^K = \mathbf{K}$$

A diagram illustrating the calculation of key vectors (K) from input vectors (X). It shows a green 4x4 matrix X multiplied by an orange 4x4 weight matrix W^K , resulting in an orange 4x4 matrix K .

$$\mathbf{X} \times \mathbf{W}^V = \mathbf{V}$$

A diagram illustrating the calculation of value vectors (V) from input vectors (X). It shows a green 4x4 matrix X multiplied by a blue 4x4 weight matrix W^V , resulting in a blue 4x4 matrix V .

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

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

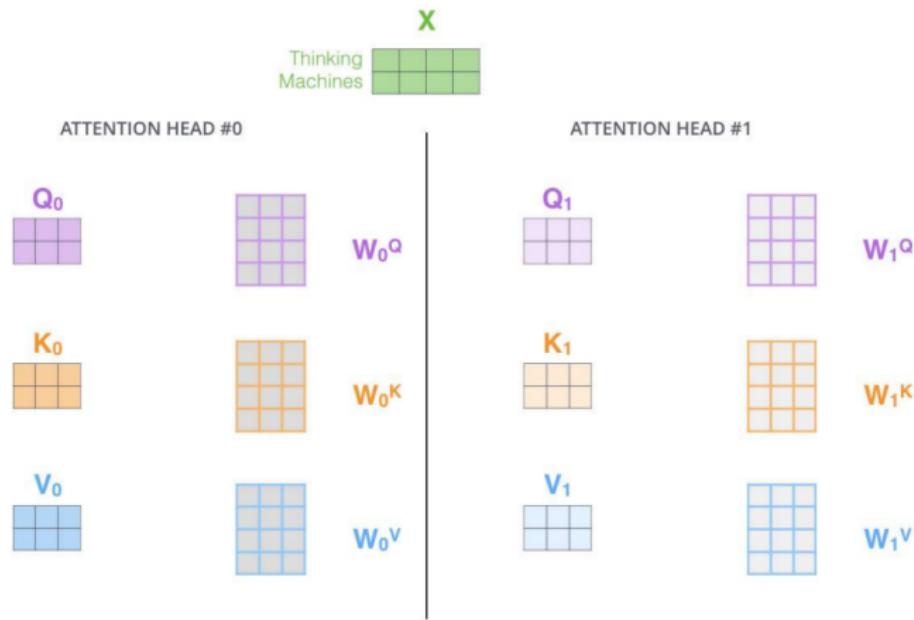
The diagram illustrates the matrix calculation of self-attention. It shows three input matrices: Q (purple, 3x3), K^T (orange, 3x3), and V (blue, 3x3). The calculation involves multiplying Q and K^T , then dividing by $\sqrt{d_k}$, and finally applying the softmax function to the result, followed by a multiplication with V . The result is labeled Z .

The self-attention calculation in matrix form

▶ jalamar

Attention Illustration - Reminder

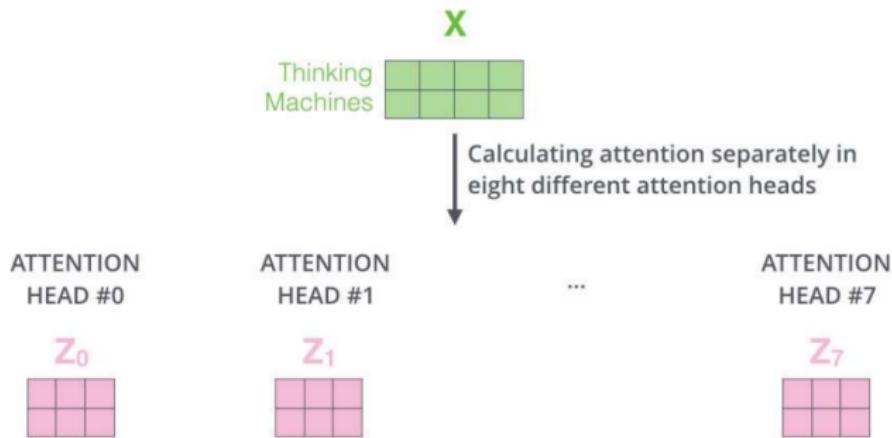
- Matrix Calculation of Self-Attention



▶ jalamar

Attention Illustration - Reminder

- Matrix Calculation of Self-Attention



▶ jalamar

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

$$= \begin{matrix} Z \\ \hline \end{matrix}$$

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

