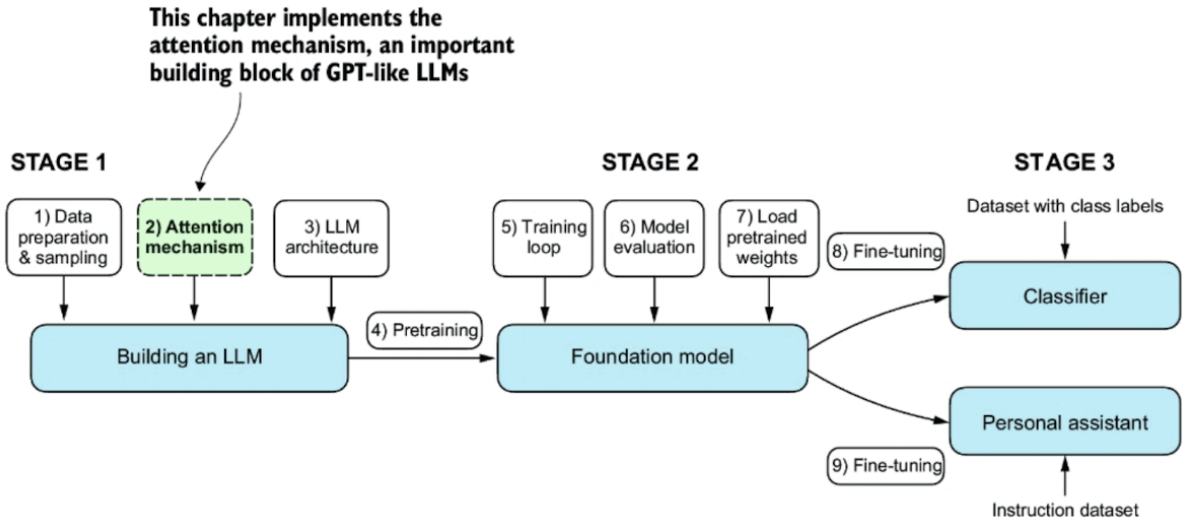
Ch03: Coding Attention Mechanisms

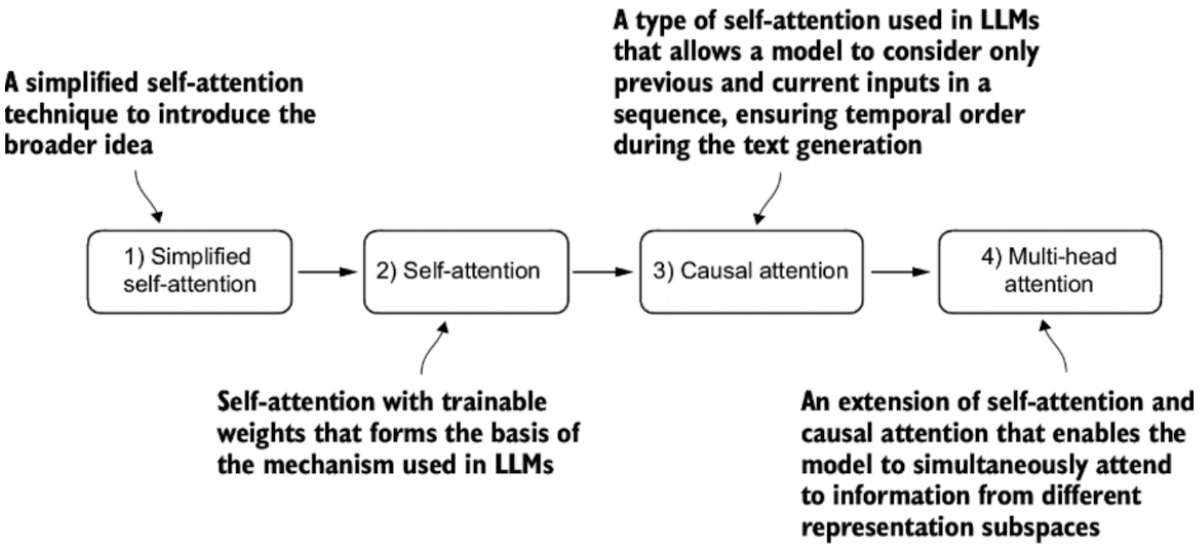


There are 3 main stages of Training an LLM, In this chapter we will focus on `Step-2` of `Stage-1`

STAGE 1

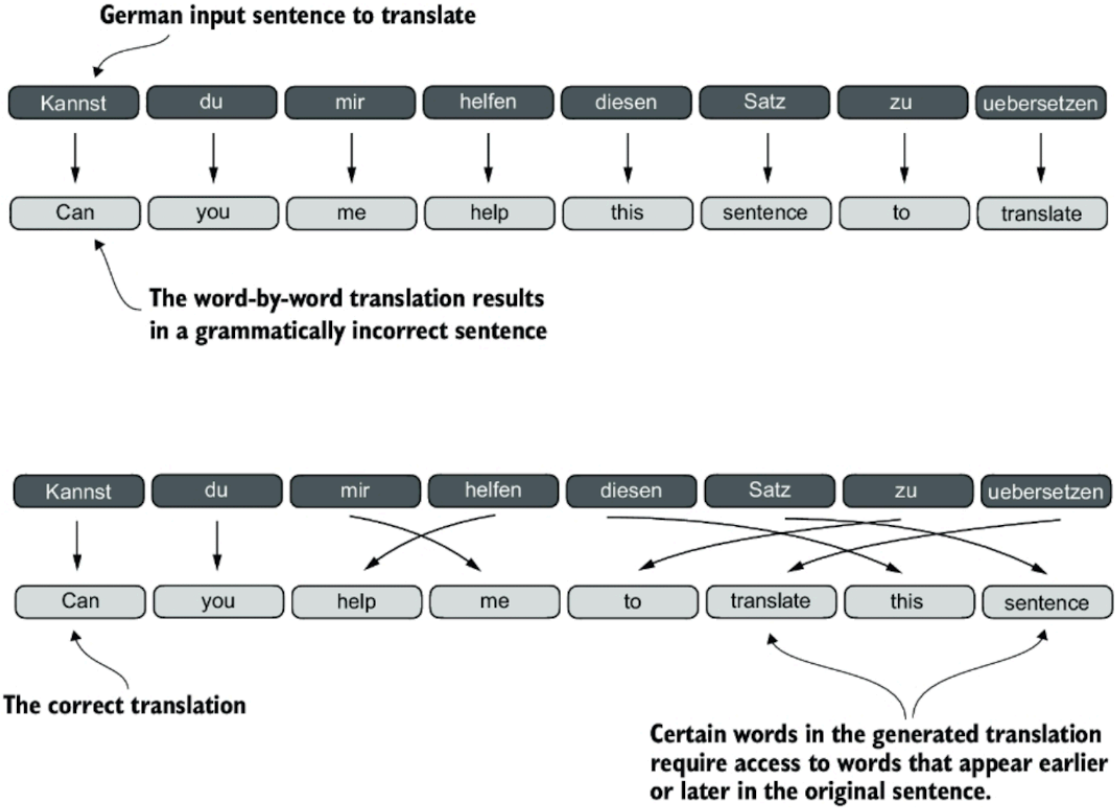
1. Data preparation and sampling
2. **Attention mechanism** 🤌
3. LLM architecture

In this chapter we will implement the 4 different variants of Attention Mechanism which were build on each other. So in the end we make attention mechanism which we can plug in the LLM training.



- **Simplified self-attention** : before adding trainable weights
- casual self attention adds mask to self attention that allow LLM to generate one word at a time
- Finally multi-head attention allows the attention mechanism to organizes in multiple heads that allows model to capture various aspect of input data in parallel.

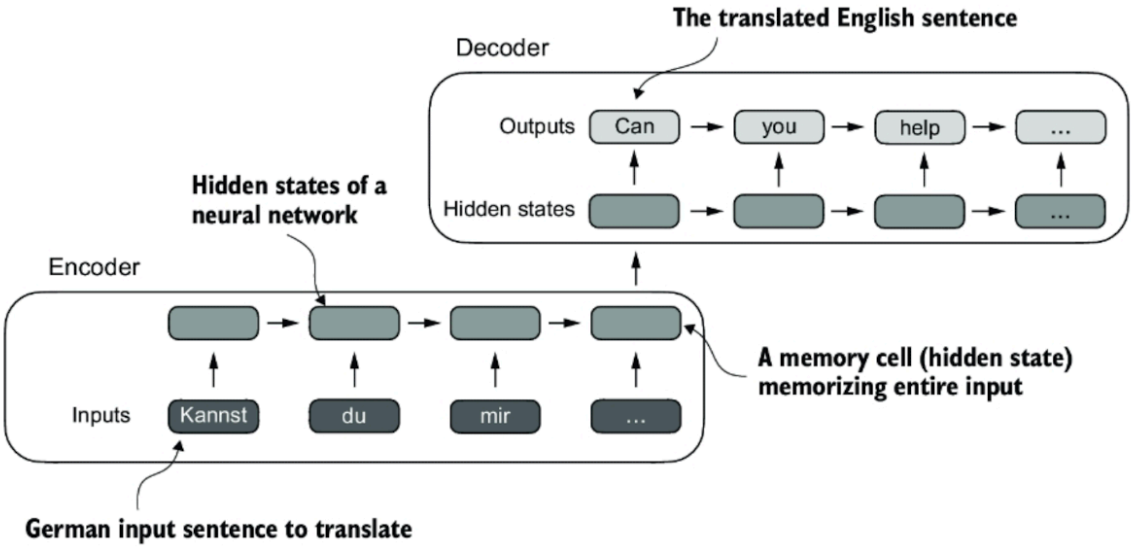
3.1 The problem with modeling long sequences



We can’t simply translate word by word. Instead translation requires the contextual understanding and grammatical alignment.

- To address this problem it is common to use the deep neural networks submodules : *encoder* and *decoder*.
- Encoder : read in and process the text and decoder produce the translated text.

Before Transformer: RNNs Encoder-decoder were popular for machine translation.



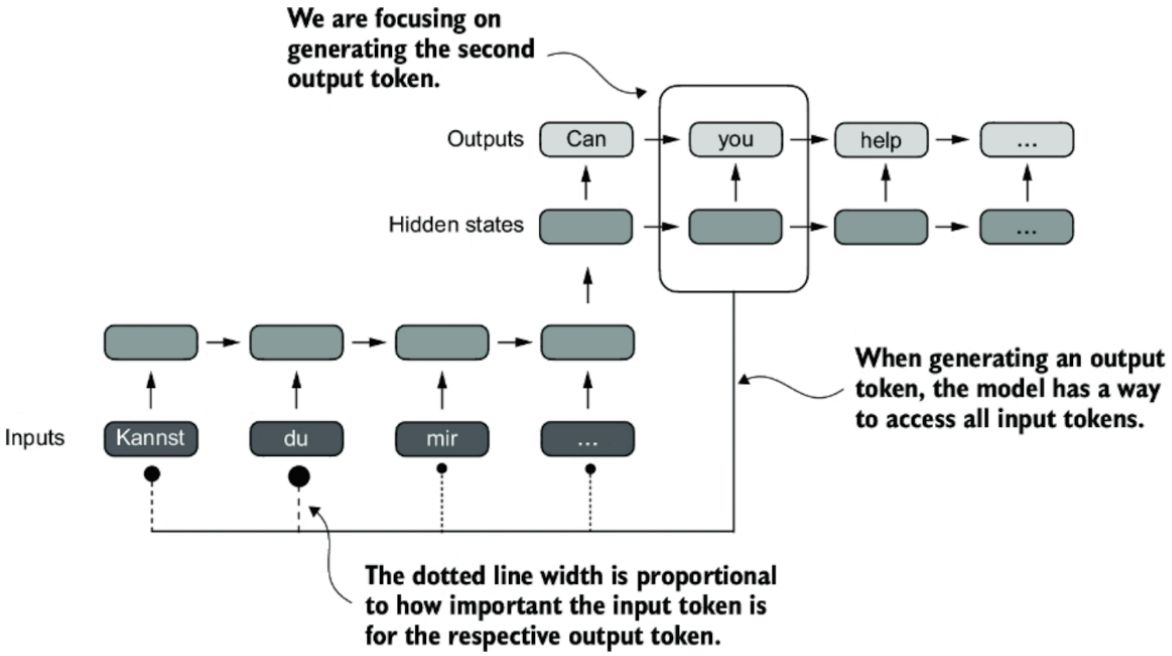
The key idea of the encoder part process is that *"entire input sequence in the hidden state (memory cell)" The decoder takes the hidden states to produce the output.*

We can think the hidden state as the embedding vector as we discussed in the chapter 02.

- The problem with the encoder-decoder RNNs is that decoder can't access the earlier hidden states during the decoding process.
- It solely relies on the current hidden state, which encapsulate all the relevant information.
- This leads to the loss of information, specially in case of long sequences.
- This shortcoming motivated the design of the self-attention.

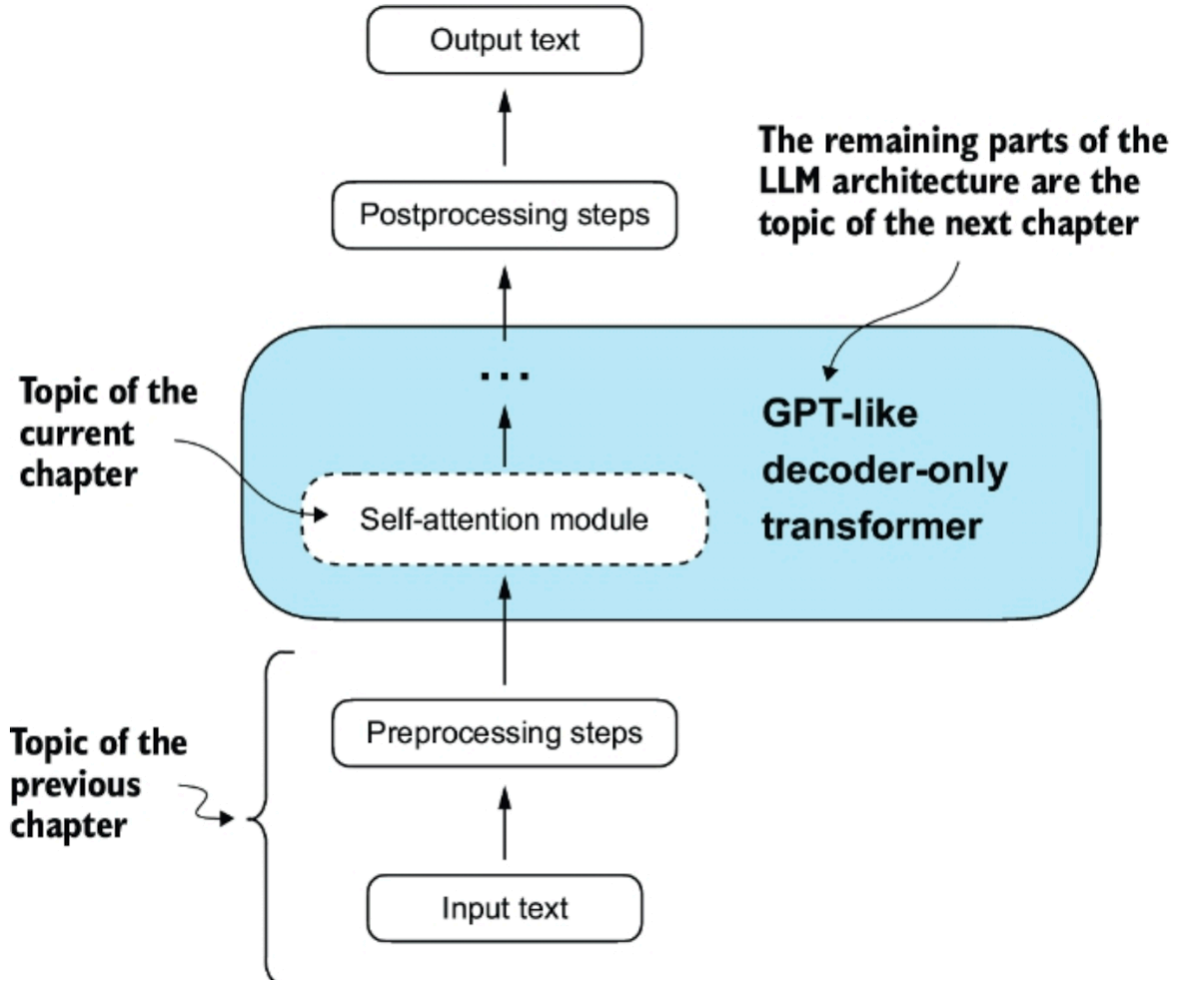
3.2 Capturing the data dependencies with self-attention mechanism

- RNNs works good for translating the short sentences, but not well for the longer sequences.
- One of major shortcoming of RNNs is that it has to remember entire encoded input into a single hidden state before passing it to the decoder.



Hence researcher designed ***Bahdanau attention mechanism*** **(2014)**(refer appendix B) in which decoder can access the selective tokens as shown in above figure.

- 3 years later (2017) researchers proposed the original transformer architecture including the self attention mechanism inspired from Bahdanau attention mechanism.
- Self attention allows each position in the input sequence consider the relevancy of “to attend”.



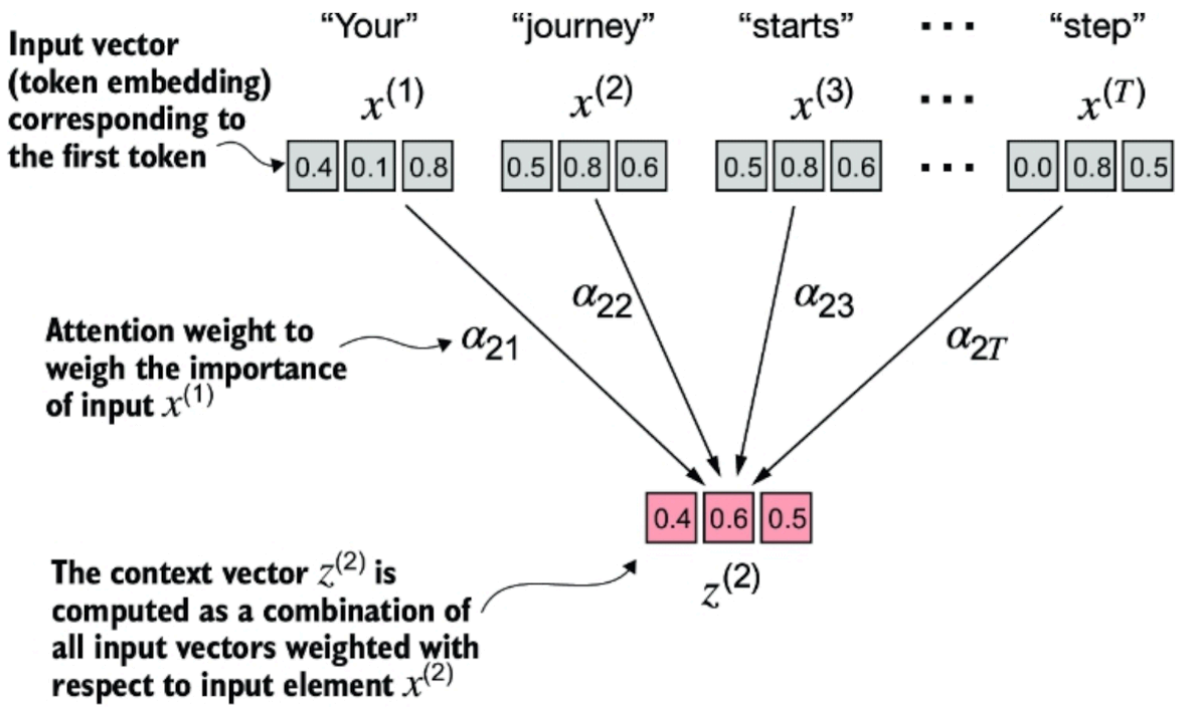
Self attention used to compute more efficient input representation allowing it interact with the position of sequence and weight the importance of others.

3.3 Attending the different parts of Input with self- attention

This part is very important so pay attention and requires focus. As it is the core concept of the LLMs.

👉 The **“Self”** in the Self-attention
- Self refer to the ability to compute the attention weights by relating to different positions with in the single input sequence.
- Learn relationship and dependencies between various parts of input itself, such as word in the sentence or pixels in the image.
- This is contrasts to traditional attention mechanism where relationships and dependencies are between two different sequences.

3.3.1 A simple self-attention mechanism without trainable weights



The goal of self-attention is to compute a context vector for each input element

- In the above figure input sequence is from $x^{(1)}$ to $x^{(T)}$

Example:

- “Your journey starts with one step”
- x(1) corresponds to the d-dimensional embedding to a specific token “Your”. In above figure input embedding is 3 dimensional.

- In self attention our goal is to find the context vector z(i) for each element in the input sequence $x_{(i)}$
- A *context vector* can be interpreted as enriched embedding vector.
- In the above figure the context vector $z(2)$ is for $x(2)$.
- Context vector plays a crucial role in the self-attention. Their purpose is to create a enriched representations for each element in the input sequence by incorporating the information from all other elements in the sequence.
- First we will understand the `context vector` without trainable weights.
- Next follow `pyare-ch03.ipynb`

pending :

1. https://github.com/rasbt/LLMs-from-scratch/tree/main/ch03 → 02,03 (buffer and exercise and bonus