Lecture 13: Attention Mechanism

Why “**Attention”?**

Because the model is literally **deciding which tokens to “pay attention to"** when processing a given token.

It scores all other tokens (via queries·keys), figures out which ones matter more, and then **focuses** on those by giving them higher weights. The term is borrowed from human attention—prioritizing the most relevant information instead of treating everything equally.

There are 4 types of attention mechanism:

1. **Simplified self attention:** A simplified self attention technique to introduce the broad idea.
2. **Self attention:** Self attention with trainable weight, that form the basis of the mechanism used in LLMs.
3. **Casual attention:** A type of self attention used in LLMs that allows the model to only consider previous and current inputs in a sequence.
4. **Multi-head attention:** An extension of self attention and causal attention that enables the model to simultaneously attend to information from different representation subspaces.

The translation process requires contextual understanding and grammar alignment.

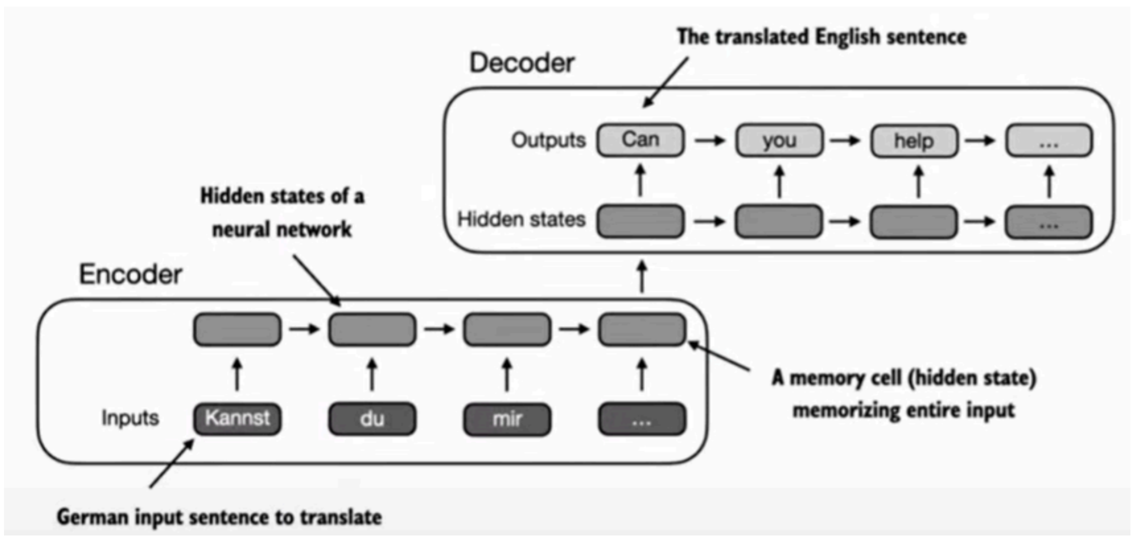
To address this issue that we cannot translate text word by word, it is common to use a neural network with two submodules:

1. **Encoder:** Reads and processes text
2. **Decoder:** Translates text

The encoder processes each item in the input sequence, it compiles the information it captures into a vector (called a context). After processing the entire input sequence, the encoder sends the context over to the decoder, which begins producing the output sequence item by item.

The decoder also maintains a hidden state that it passes from one time step to the next.

Before transformer came, RNNs were used to handle memory. Here’s how the encoder-decoder RNN works:


The translated English sentence
Decoder
Outputs
Can
you
help
...
Hidden states
...
Hidden states of a neural network
Encoder
A memory cell (hidden state) memorizing entire input
Inputs
Kannst
du
mir
...
German input sentence to translate

Then why attention mechanism?

Because, in RNNs, the decoder only has access to the last hidden state, it can not access the previous hidden states.

But why is it a big problem?

If we are working with long sequences, and the decoder only relies on one hidden state, that’s a lot of pressure on the decoder and leads to a **loss of context.** It usually fails for long sequences because it is very hard for it to contain a long information in one single hidden state.

This is where attention mechanism comes in, **to capture long range dependencies.**

Using attention mechanism, when we're translating a long sentence, we can have access to all of the input words, and we can decide how much attention to pay to one token. This importance is determined by the so called **Attention Weights.** With attention mechanism, we can gain access to words that come at the beginning of the long sentence, which wasn't possible in RNNs.

During translation, when the decoder is predicting a certain word, the attention mechanism allows it to focus on part of input that corresponds to the original word.

This **dynamic focus** on different parts of input sequence allows models to learn long range dependencies more effectively.

Note that, the model doesn't just mindlessly align the first word at the output with the first word from the input. It actually learned from the training phase how to align words in that language pair.

Self Attention: Self-attention lets each token look at **all the other tokens in the same sequence** to decide what information it needs when forming its own representation.

In Self Attention, the "self" refers to the mechanism's ability to compute attention weights by relating different positions in a single input sequence.

It learns the **relationship between various parts of the input itself**, such as words in a sentence.

This is in contrast to traditional attention mechanisms, where the focus is on relationships between elements of 2 different sequences.