Transformers and Natural Language Understanding

Objective: To argue if transformers have the architectural expressiveness to perform the task of Natural Language Understanding.

Condensing down what Transformers are and what is a basic structure?

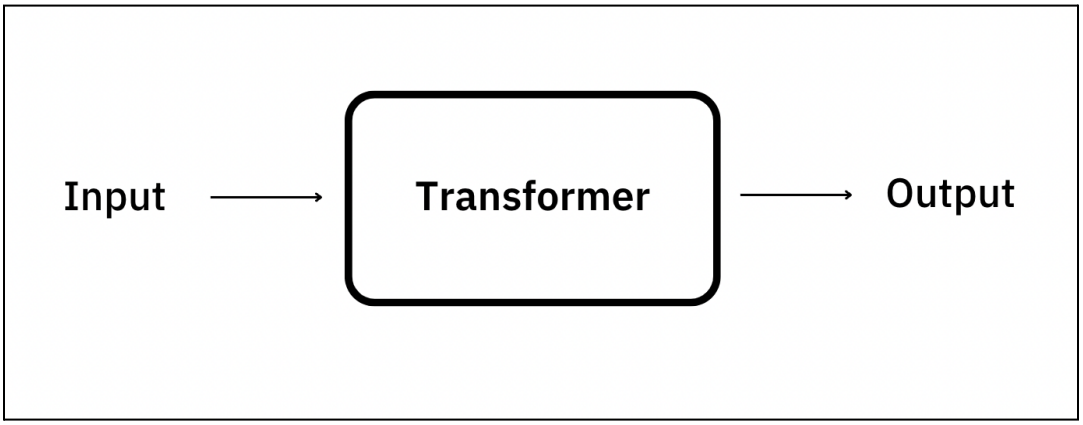


Fig 1. Schematic layout with Transformer as a Black Box

Tranformers are like blackboxes that are responsible for any form of transformations.

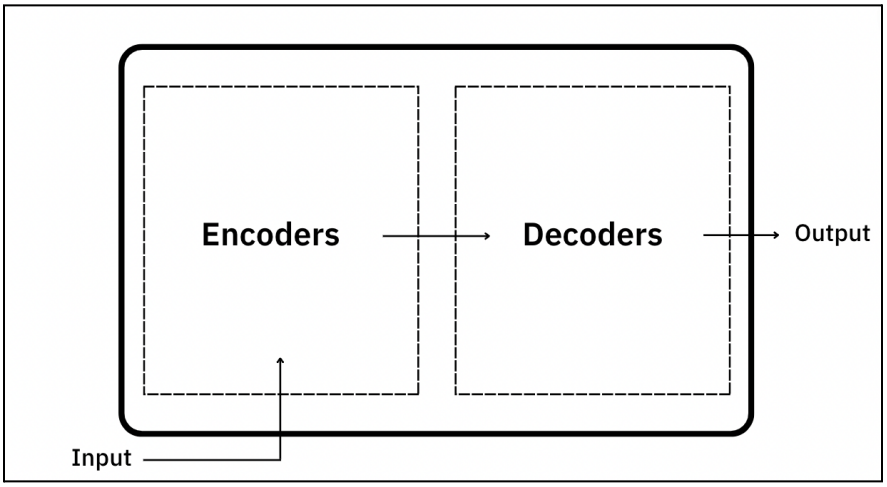


Fig 2. Internal Structure of a Transformer

The internal structure of transformers, involve the presence of an **encoder** alongside a **decoder.**

What does the encoder consist of?

The encoder is made up of two sub-units:

Self Attention ----> Feed Forward Neural Network

- The **Self Attention** Layer is present in the encoder, and it allows the inputs to interact with one another, in order to find out which particular input should be paid more attention to, through **attention scores.**
- The **Feed Forward Neural Network,** which is a simple neural network allowing only unidirectional movement of information.

The decoder is made up of three sub-units:

Self Attention ----> Attention ----> Feed Forward Neural Network

- The **Self Attention** Layer does the same work as it does in the encoder.
- **Doubt: What is the decoder-encoder attention layer?**
- The **Feed Forward Neural Network,** which is a simple neural network allowing only unidirectional movement of information.

Self Attention in Depth

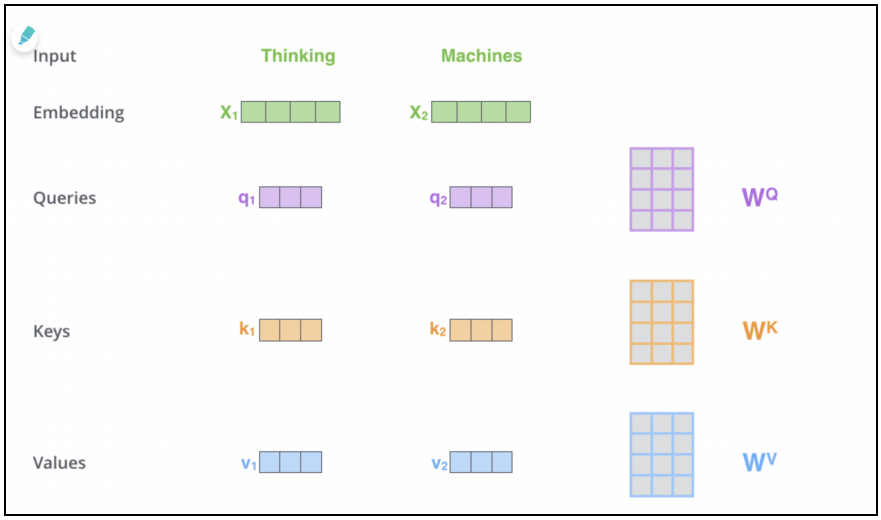


Fig 3. Dimensions of the involved vectors

- The first step of calculation involves the creation of an **embedding, query, key and a value** vector.
- **Doubt: Why is it an architectural choice to keep the dimensions of the query, key and value vector smaller?**

The Self Attention Score:
This score exists for every word of an input sentence, wherein for a given sentence:

word_1/ word_2/ word_3/ word_4/ word_5/ word_6/ word_7

- Self Attention score 1: Every word scored against **word_1**
- Self Attention score 2: Every word scored against **word_2**

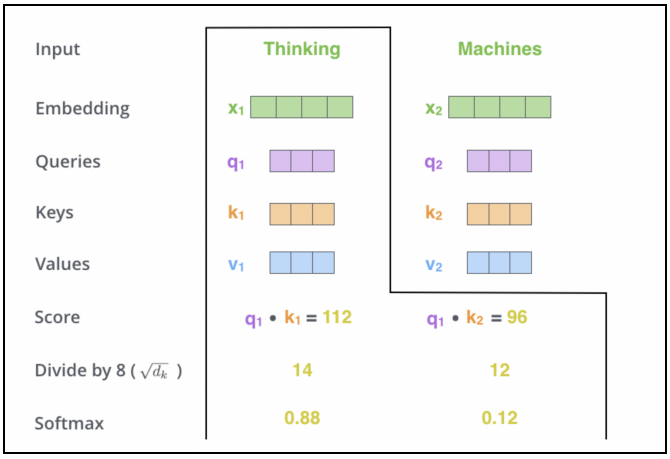
And so on and so forth.

Step 1:
The calculation of self attention is done by **dot product of query vector with key vector:**

- **Self_Attention_word_1:**
  - **self_attention_score1-** q1.k1
  - **self_attention_score2-** q1.k2

Step 2:
The self attention scores are divided by the **root of the dimensions** of the key vectors.
Step 3:

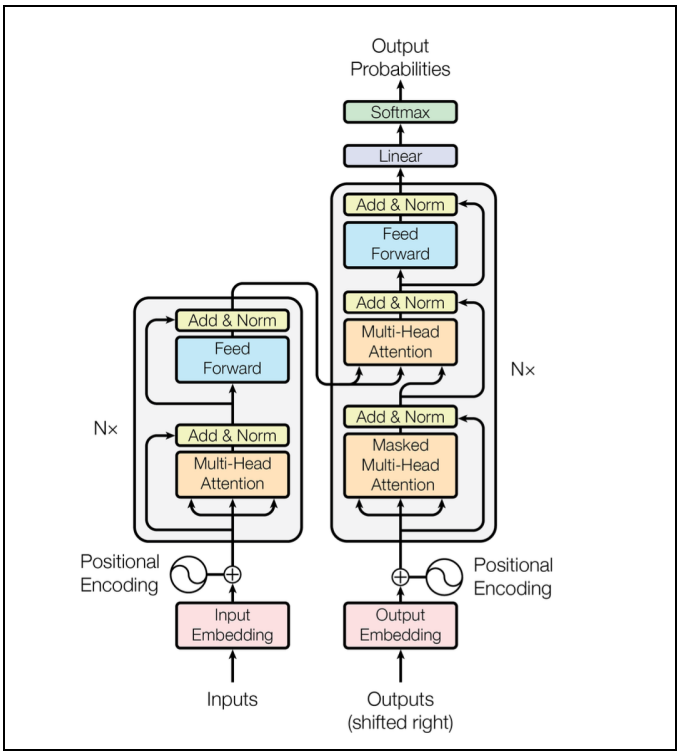
The values obtained from step 2 are passed through the **softmax activation function.**



The **greater** the softmax score, the greater each word is expressed at the position.

Step 4:
The **softmax scores** are then multiplied with the **value vectors**, to provide weight to the words whose importance is greater, while drowning out irrelevant words.

Step 5:
The concatenation of the weighted value vectors, gives the output of the self attention layer.



Additional Information - Multi-headed Attention Layer

To form a multi-headed attention layer, when **N** is the number of self-attention heads inside the layer, the **key**, **query** and **value vectors**, are divided into N vectors, and each yield an output through the self attention heads, following which they are then concatenated, and made to pass through the final linear layer.

Each Self-Attention Head can learn something different, which gives more representation to the model.

Residual Connection, Normalization and the Feed Forward NN

These are specifically used to refine the output more, allowing the decoder to focus its attention to only the most important segments of the input, in this case the most important words.

Additionally, when more encoders are stacked together, it allows the transformers to learn better, and thus making the processed input a more refined learning tool, to produce better outputs

The Decoder

Step 1:

The decoder, takes as input the previous **outputs(why?)** and allows it to be split into the **query, value and key vectors** which are passed through a multi-headed attention layer followed by normalization.

Additionally, to this Self-attention Layer, there can be additional mask applied which allows the model to restrict to certain conditions, making some values of the matrices equivalent to (-infinity), ultimately making them 0 under the softmax function.

Step 2:

The second **attention** layer, receives the outputs of the encoder as input along with the normalised outputs of the first self-attention layer of the detector. This is the **cross-attention layer.** This too undergoes, addition and normalization, and is mainly useful to allow the decoder to **contextualise** the encoder output.

Step 3:

Finally, the decoder output is passed through a classifier. A **linear classifier** having N classes, takes the input and forms a N-sized matrix, then passes it through a **softmax** function to obtain the **probability scores** of each class, and the highest probability score class is the predicted output.

Inference

Therefore, after an indepth study of the structure of transformers, we can argue that **Transformers have architectural sufficiency to perform the basic task of Natural Language Understanding**:

- They have an infinite memory, allowing them to take as input very large sentences without loss of any input.

- The self-attention allows priority to exist, meaning the model can focus on only important aspects(words) of the input.
- **The transformer decoder, is also influenced by its previous outputs which allows it learn and improve.**
- The cross-attention layer, allows the decoder to contextualise the encoder output, that is base its processing on the important aspects of the encoder output.
- Multiple encoders and decoders, allow the transformer to direct **attention** to different portions of the input and increase the representation in the model.
