The Transformer

Working through attention in deep learning

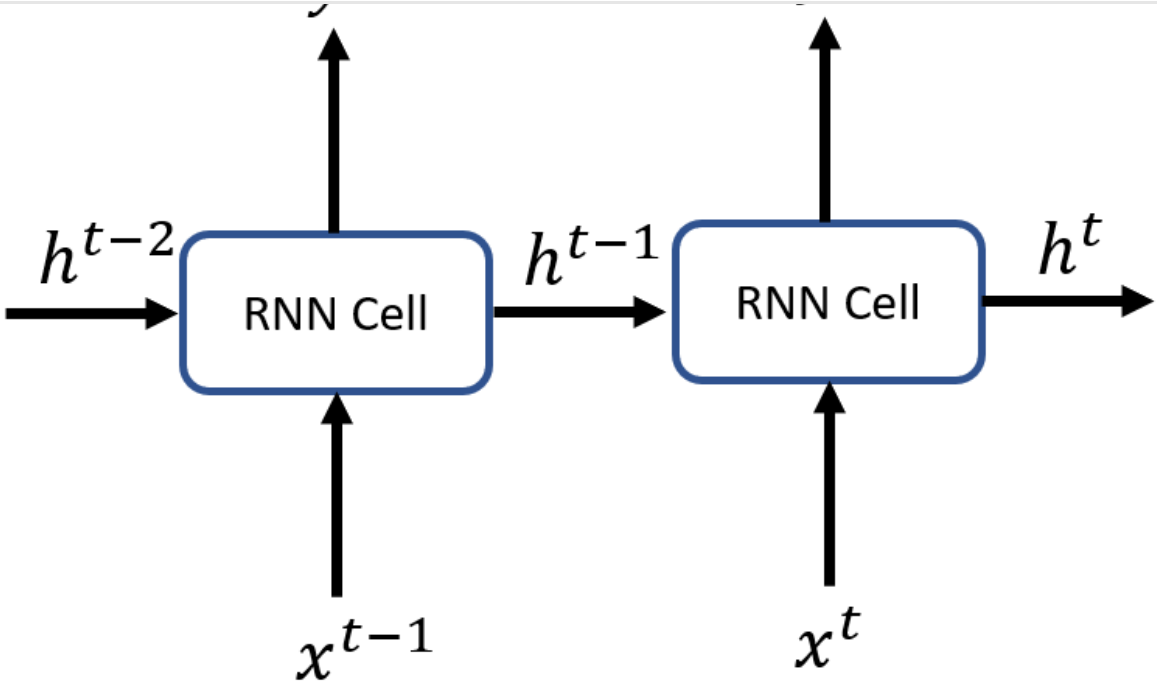
Tim Sullivan Jun 20 · 5 min read ★

I've started to go through classic papers in machine learning, inventions that shifted the state of the art or created an entirely new application. These are my notes on the **Transformer** introduced in Attention is All You Need. The transformer addressed problems with recurrent sequence modeling in natural language processing (NLP) but has since been applied to vision, reinforcement learning, audio, and other sequence tasks.

Why attention?

Recurrent models built from RNNs, LSTMs, or GRUs were developed to deal with sequence modeling in neural networks because they can include information from adjacent inputs as well as the current input. This has obvious relevance for data like language where the meaning of a word is partially or entirely defined in relation to surrounding words.

The problem occurs when relevant words occur far from the current input to the network. The gradient between these disconnected words has to travel through all of the recurrent cells because the model contains no direct connection between each word.



RNN cell sequence

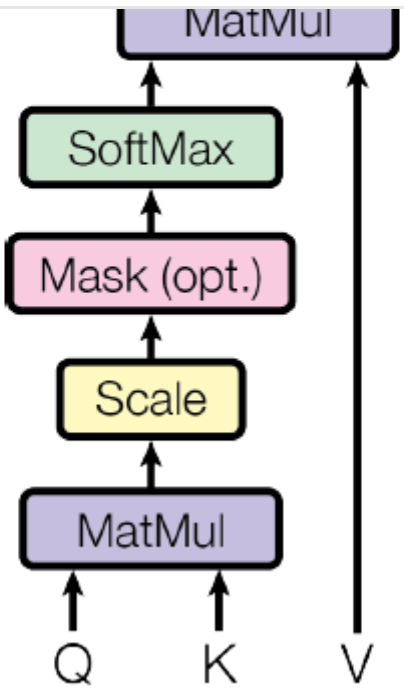
This sequential model also means the computation isn't as parallelizable as the computation for each cell depends on the results from the previous cell slowing down the training process.

Attention

Enter attention. Attention works by weighting the relational importance of inputs regardless of distance. The inputs to an attention module (also known as a head) are a set of queries (Q), keys (K), and values (V) expressed as embedded vectors passed through a fully-connected layer. By multiplying the queries and keys together the attention head determines which keys should be activated for a particular query. In language, this means identifying which words are relevant to the current word. Then, multiplying by the value network, the attention head determines how much importance to assign to each query-key combination.

You might think of this as a key activating pins in a lock. Given the correctly cut key for the pins in the tumbler, the lock will turn. The beauty of attention over other sequence models is the interactions are considered across the entire sequence.

Scaled Dot-Product Attention

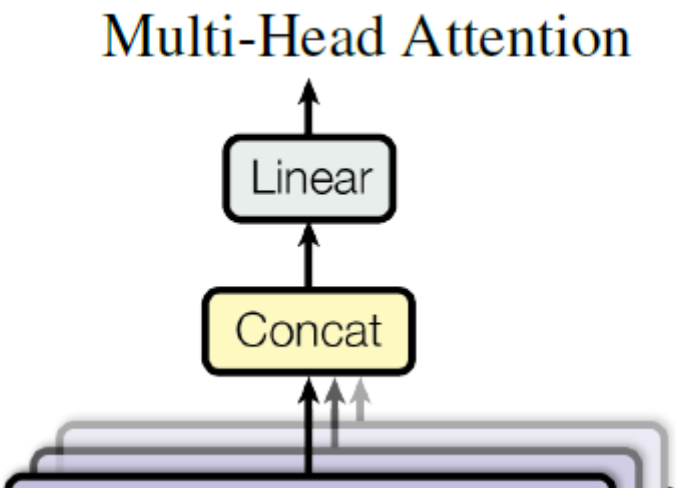


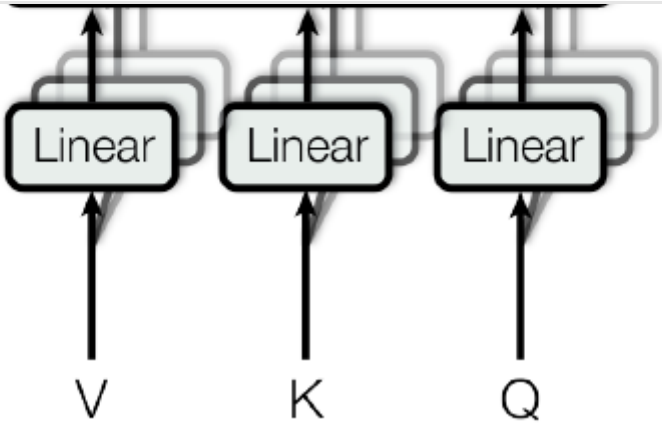
(source: Attention is All You Need)

All of this combines into a mechanism that can allow, “modeling of dependencies without regard to their distance in the input or output sequences.” [1]

Multi-head attention

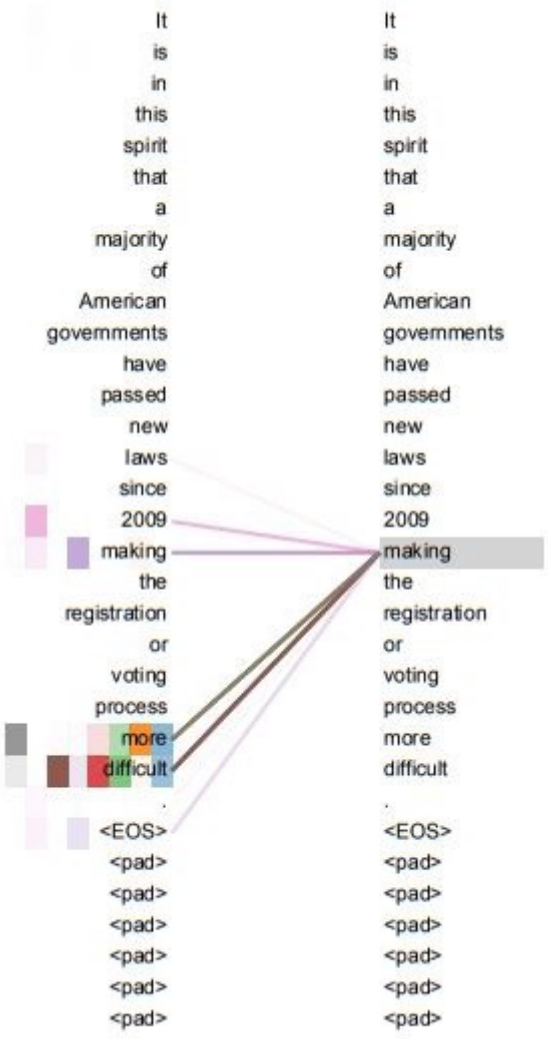
However, we’re not satisfied with just one attention head. Instead, the transformer developed in this paper contains multiple attention modules known as Multi-head Attention. By initializing separate attention heads with different weight matrices, the model computes different attention functions in parallel which are combined prior to passing through another fully-connected layer.





(source: Attention is All You Need)

You can see the results of the multi-head approach in the figure below where each head in the model assigns varying importance to different parts of the sentence given the embedded input word “making.”

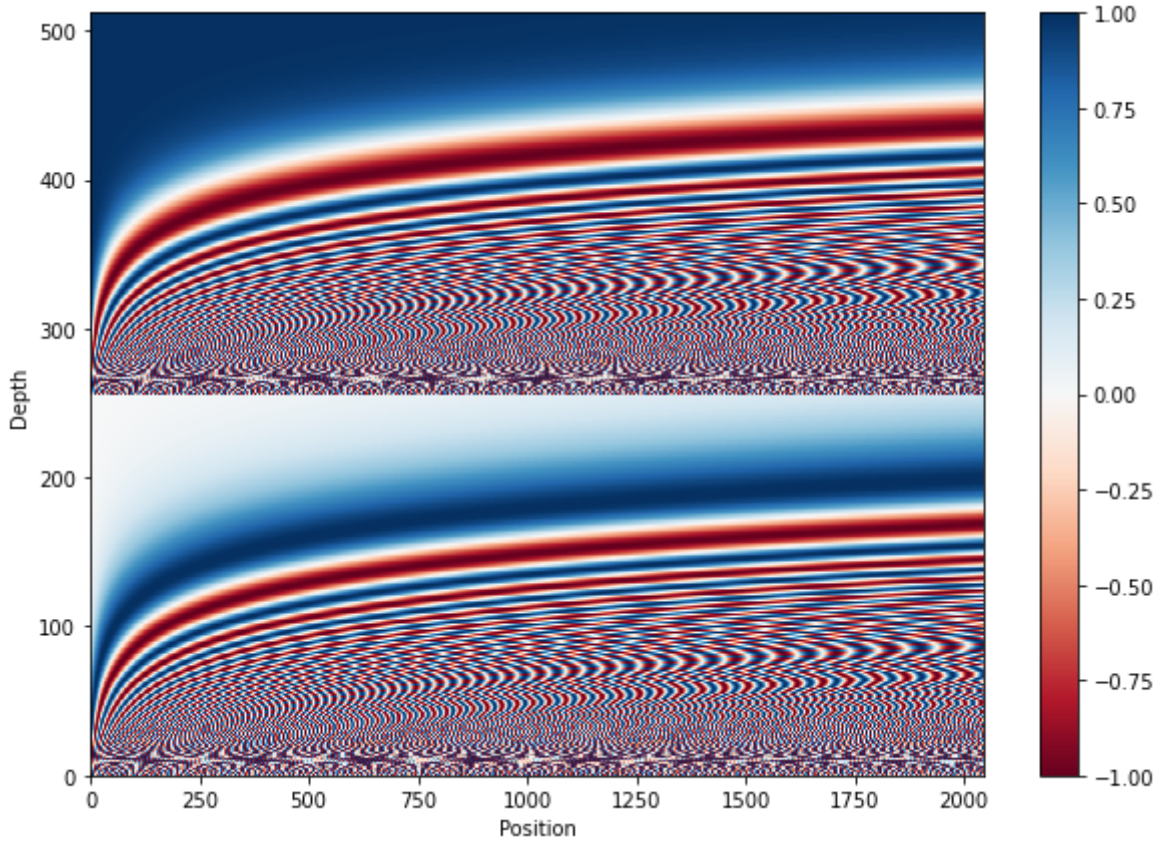


Position encoding

Of course, one benefit of using recurrent models is the encoding of position information based on which recurrent cells received each input. That information is valuable for understanding grammatical structures such as modifying words (adjectives that occur near the word they modify) or prepositional phrases. The transformer model would benefit from this information so a Positional Encoding layer is added as an input.

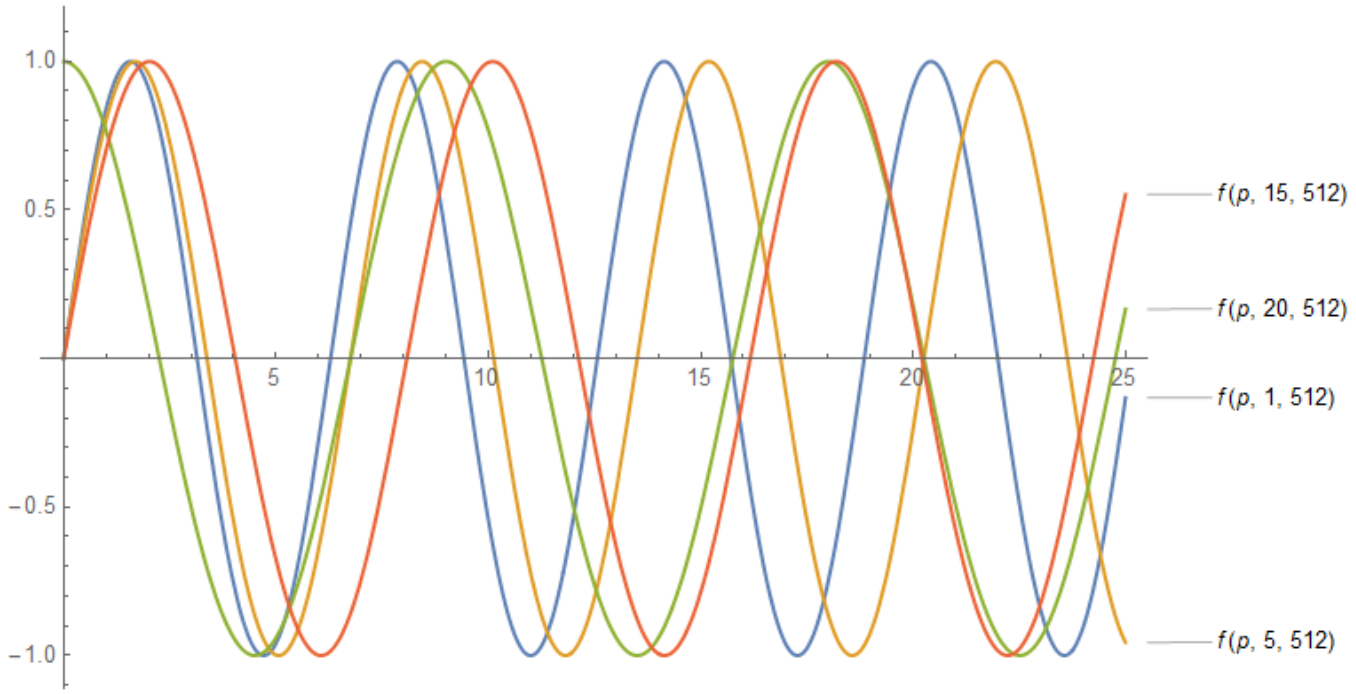
The position encoding is computed using a pair of sinusoids with varying wavelengths based on the depth of the model and the position of the input in the sequence. This position information is added to the embedding vector and together they form the input to the encoder and decoder modules of the transformer.

The TensorFlow (TF) example documentation on the transformer shows this plot of the positional encoding according to depth and position, but it's a bit difficult to interpret.



Complete plot of position embedding values (source: TF Docs)

Instead, consider this slice of the embedding showing the embedding value for different word positions and depths. Words receive higher or lower positional



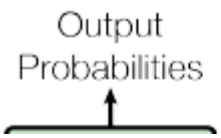
Positional embedding value for different model layers where the value f is a function of position, p, layer {1,5,15,20} and total model depth (512).

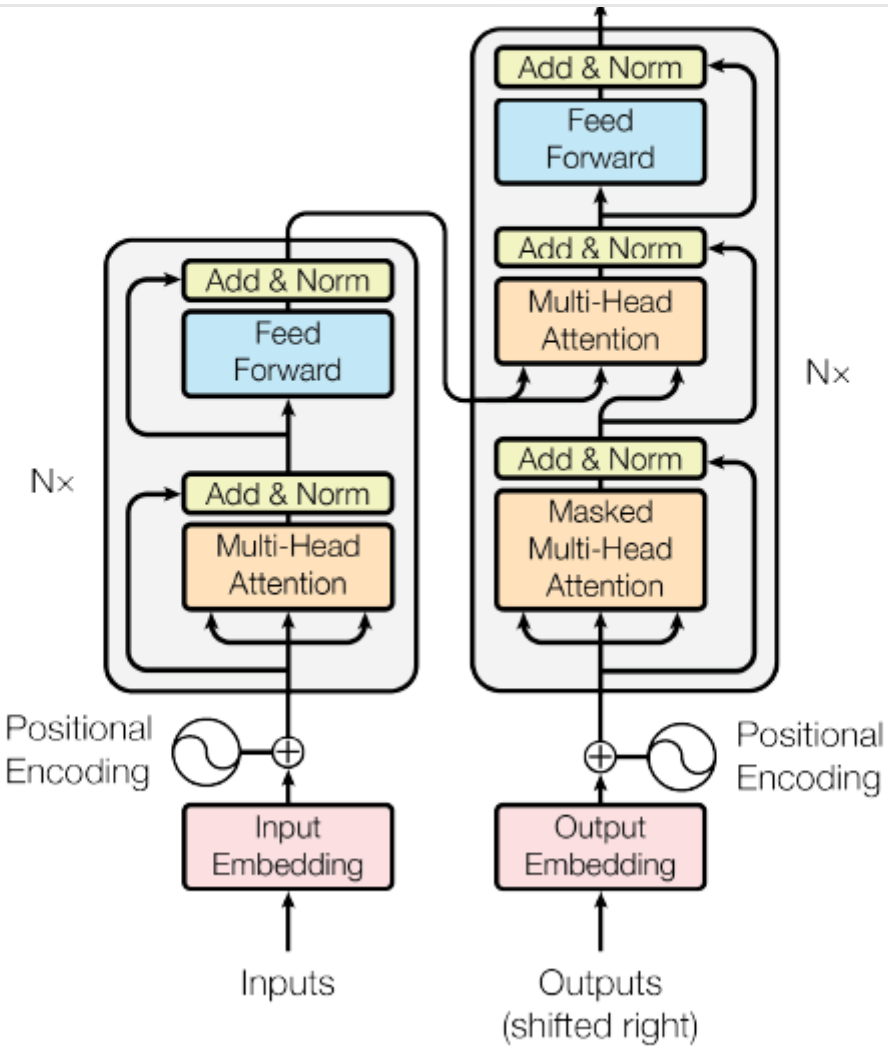
Transformer

The objective of the transformer is to take an encoded input sequence and its own previous output to predict the next output by computing attention on both. The Encoder module of the transformer takes input from a word embedding layer and the position embedding and applies the multi-head attention mechanism described earlier. The Decoder module receives the encoder output as values and keys in the second multi-head attention block and the output from the first decoder masked multi-head attention block as the query.

The inputs to the first decoder attention block are masked so the decoder cannot see the word it is trying to predict. Skip connections are included at each block.

During training, the transformer learns to decode the correct current word based on the encoder output and all other *correct* output words in the training data. When testing, the decoder output *replaces* the correct output words.





The full transformer (source: Attention is All You Need)

Visualizing

So what does multi-head attention look like in practice? Below are a few plots from the TF docs Portuguese to English translation problem where each subplot is a separate attention head. For the most part, each head activates for the translation of the Portuguese input to the English equivalent. However, sometimes the transformer computes higher attention for a single output given multiple inputs or vice-versa. This makes sense as sometimes one language will use two words to express information while other languages use only one.

Additional reading and references

1. Attention Is All You Need

2. Visualizing A Neural Machine Translation Model

4. Transformer model for language understanding

5. My notebook