Transformers Explained, really

Roger Lam

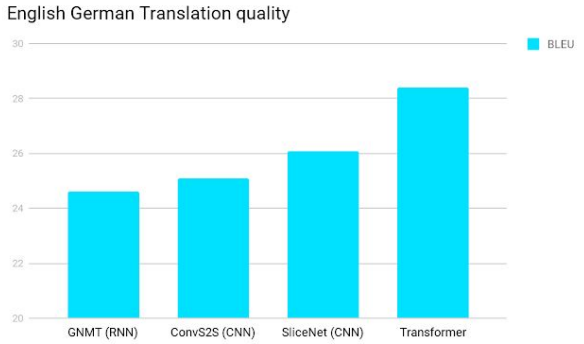
lamroger.com

Introduction

The Transformer architecture first led to breakthroughs in language translation.

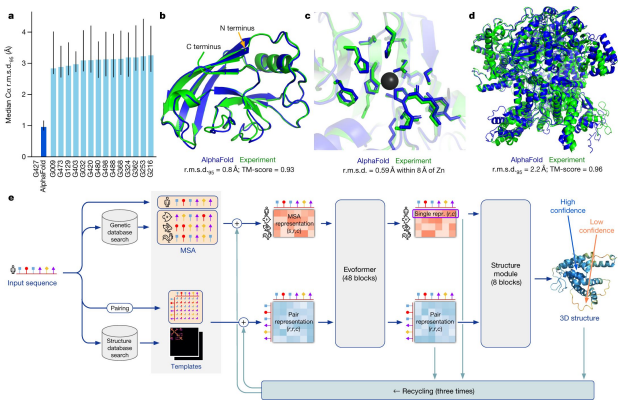
Soon after, Transformers were applied to other fields (Computer Vision, Reinforcement Learning, Science, and more) and generating similar breakthroughs.

How does the Transformer work?



BLEU scores (higher is better) of single models on the standard WMT newstest2014 English to German translation benchmark.

https://blog.research.google/2017/08/transformer-novel-neural-network.html



https://en.wikipedia.org/wiki/AlphaFold#AlphaFold_2,_2020

Building Blocks

Let’s go piece by piece.

- Embedding
- Positional Encoding
- Encoder
  - Multi-head Attention
  - Add & Norm
  - Feed Forward
- Decoder
  - Masked Multi-head Attention
  - Passing encoder results to Multi-head Attention
- Linear and Softmax

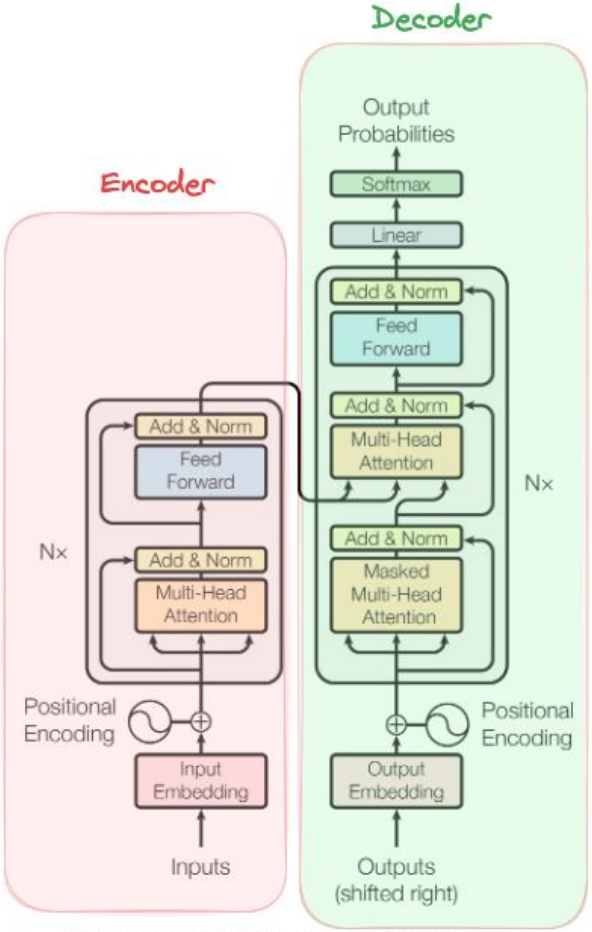


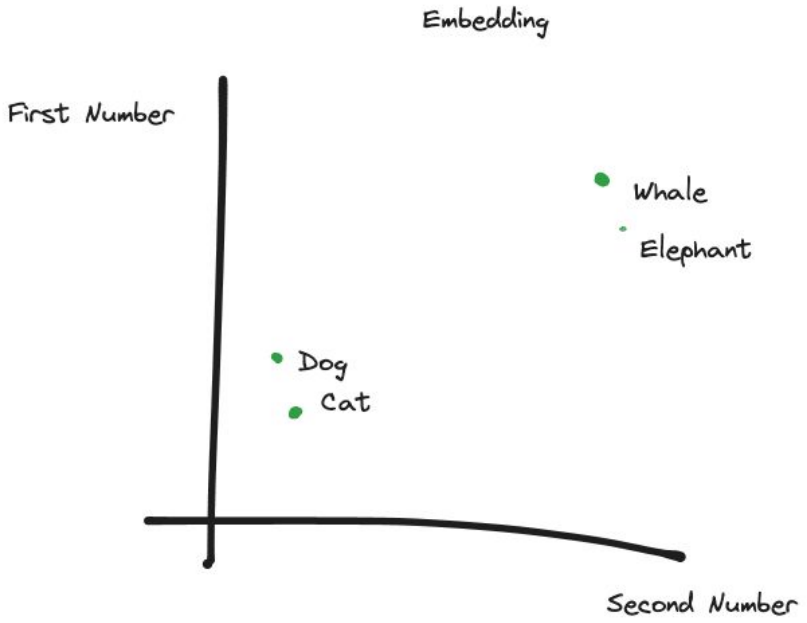
Figure 1: The Transformer - model architecture.

Embedding

Embedding is converting a word into hundreds of numbers.

Combinations of numbers allow us to group words that are related for almost any reason (size, number of feathers, emotion, etc.).

Now we can understand the meaning behind words and how words relate to each other.



Building Blocks

- **Embedding - converting words to numbers/meaning**
- Positional Encoding
- Encoder
  - Multi-head Attention
  - Add & Norm
  - Feed Forward
- Decoder
  - Masked Multi-head Attention
  - Passing encoder results to Multi-head Attention
- Linear and Softmax

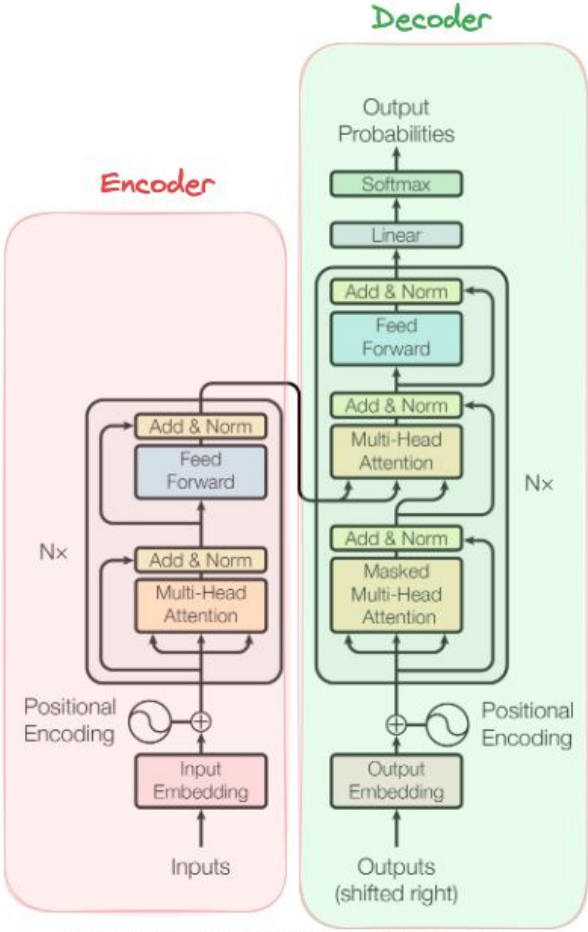


Figure 1: The Transformer - model architecture.

Positional Encoding

In sentences, it also matters in what order the word appears.

One way to keep track of position is to add something to the word's value depending on the position.

Building Blocks

- Embedding - converting words to numbers/meaning
- **Positional Encoding - adding numbers to account for their order**
- Encoder
  - Multi-head Attention
  - Add & Norm
  - Feed Forward
- Decoder
  - Masked Multi-head Attention
  - Passing encoder results to Multi-head Attention
- Linear and Softmax

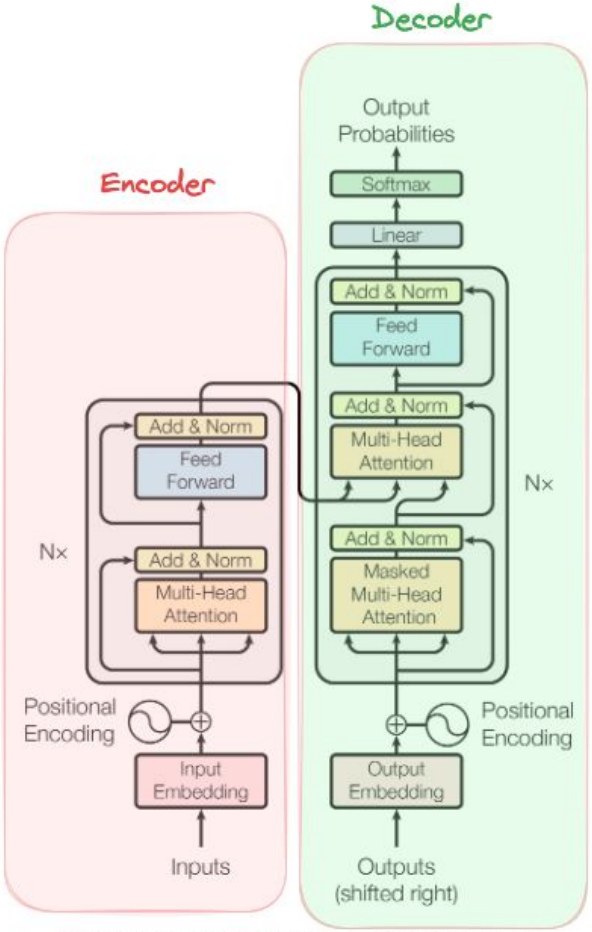


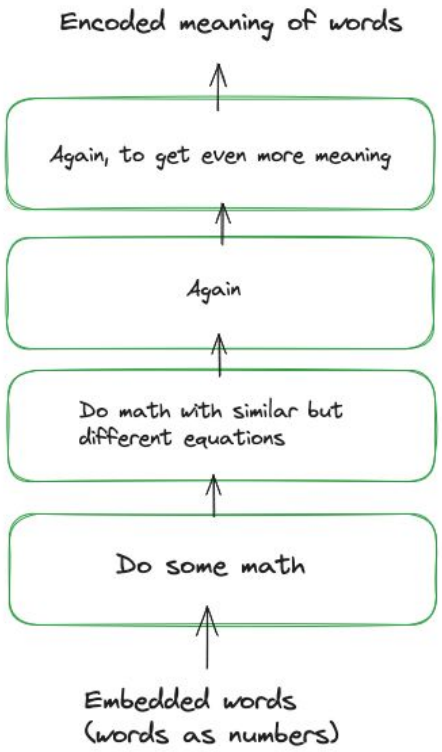
Figure 1: The Transformer - model architecture.

Encoders

Encoders convert the words (now numbers) into an even larger list of numbers (aka vector).

The extra numbers let you store the meaning of the whole sentence.

The meaning of the sentence is computed by a few methods that we'll get into.



Building Blocks

- Embedding - converting words to numbers/meaning
- Positional Encoding - adding numbers to account for their order
- **Encoder - represent the meaning of the sentence based on embedding and encoding**
  - Multi-head Attention
  - Add & Norm
  - Feed Forward
- Decoder
  - Masked Multi-head Attention
  - Passing encoder results to Multi-head Attention
- Linear and Softmax

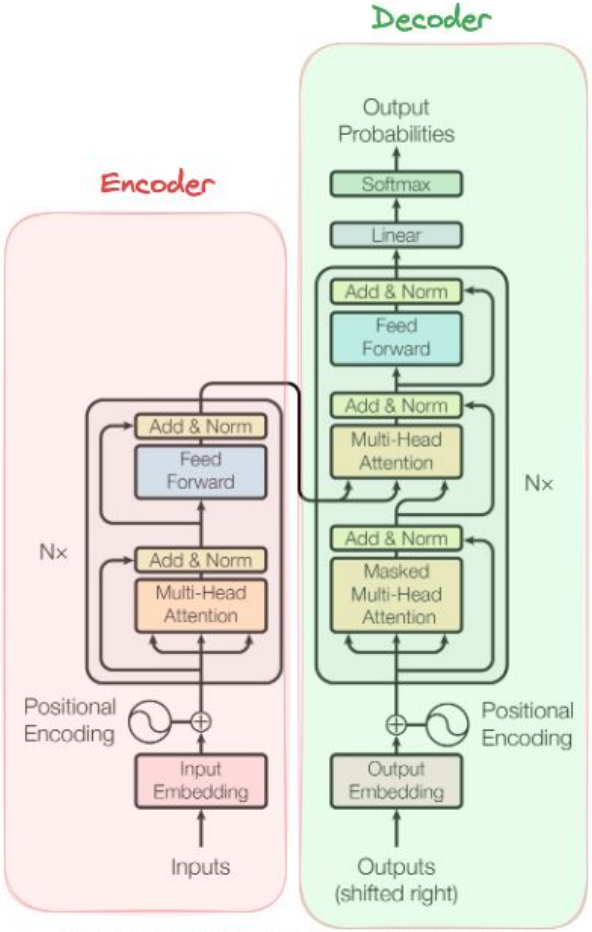


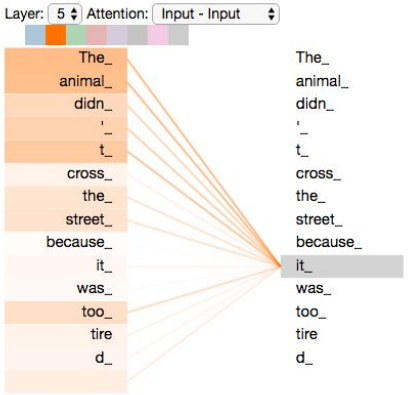
Figure 1: The Transformer - model architecture.

Attention

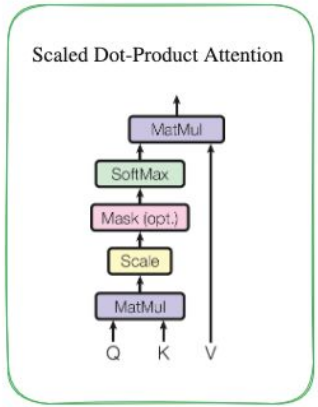
Attention is what calculates how much each word cares about each other word in the sentence.

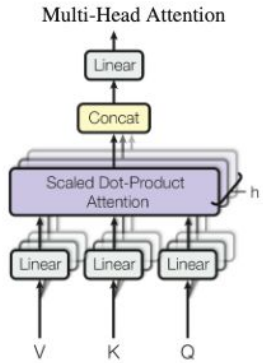
To do that, we use a Query, Key, Value method.

Think of Google. You send Google a phrase. Your phrase is understood and looks into its database using keys and return results, or values.



jalammar.github.io/illustrated-transformer/





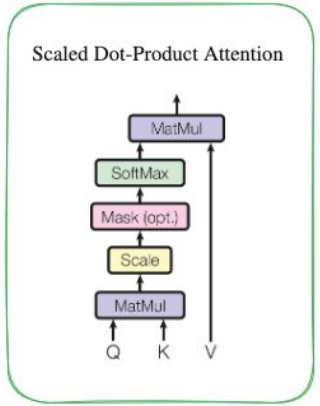
Attention

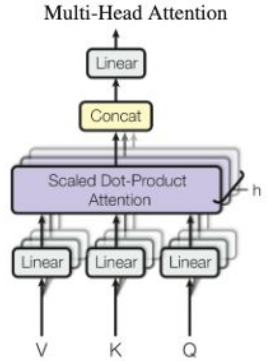
(For a deeper understanding of the math, check out Jay Alammar's Illustrated Transformer blog post.)

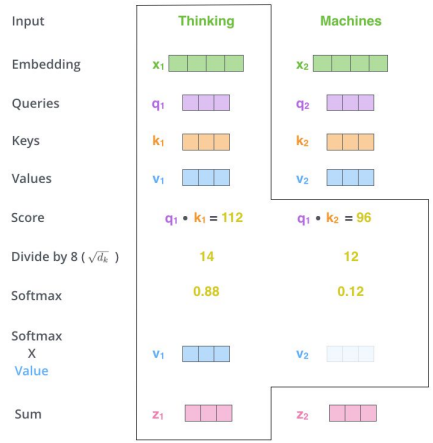
Similarly, the model has a bunch of questions (Query) to find out what's important.

Then, it sees how well each query matches with each key and gives more weight to better matching values.

This lets the architecture pay closer attention to more relevant information.







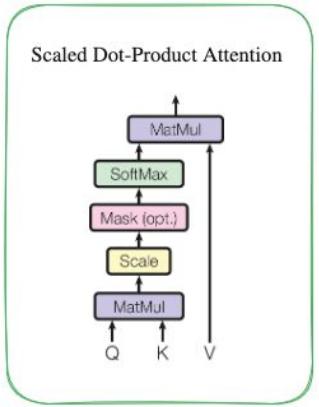
http://jalammar.github.io/illustrated-transformer/

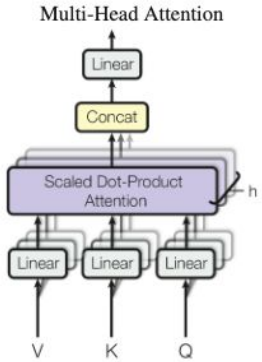
Multi-head Attention

Multi-headed attention allows for multiple Query, Key, and Value matrices.

This allows for more information to be encoded and performs better than just having one big set of Q, K, and V.

The Linear and Concat layers help split and merge results back together.



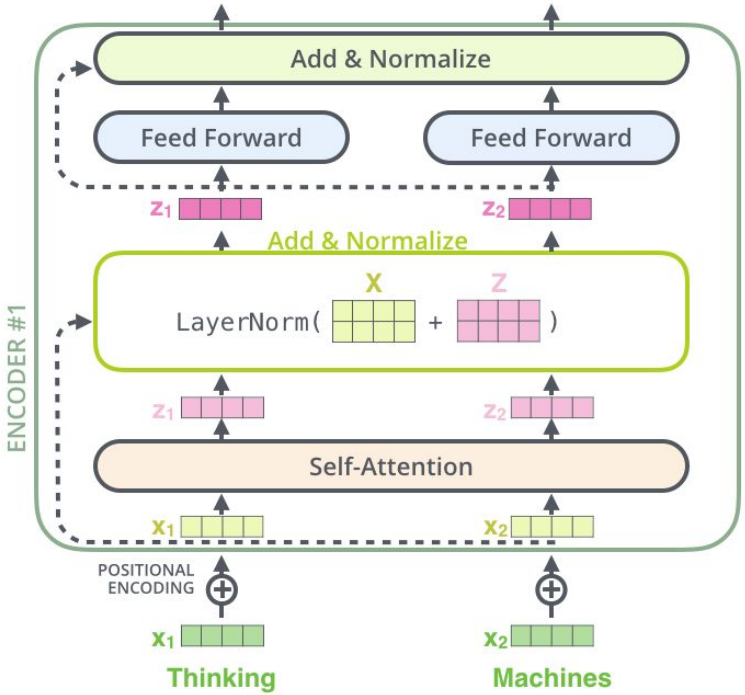


Add and Norm

After attention is calculated, we continue forward by adding the original input and the output and normalize the result.

Adding the original makes sure the original information isn't overwritten entirely.

Normalizing makes training easier and more stable.



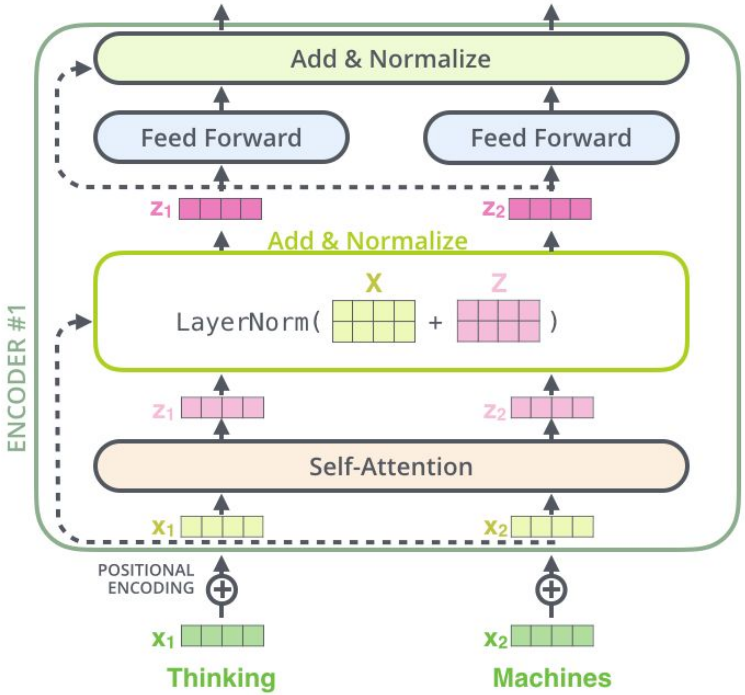
Feed Forward

Feed Forward layers are a standard way to organize neural networks.

It combines linear and nonlinear equations to save complex patterns and relationships.

This layer doesn't use and of the input data so it helps ground the model.

You can think of it as additional processing.



Building Blocks

- Embedding - converting words to numbers/meaning
- Positional Encoding - adding numbers to account for their order
- Encoder - represent the meaning of the sentence based on embedding and encoding
  - **Multi-head Attention - calculate meaning using embedding and positioning, multiple times**
  - **Add & Norm - maintain some input info and stability**
  - **Feed Forward - add more complexity**
- Decoder
  - Masked Multi-head Attention
  - Passing encoder results to Multi-head Attention
- Linear and Softmax

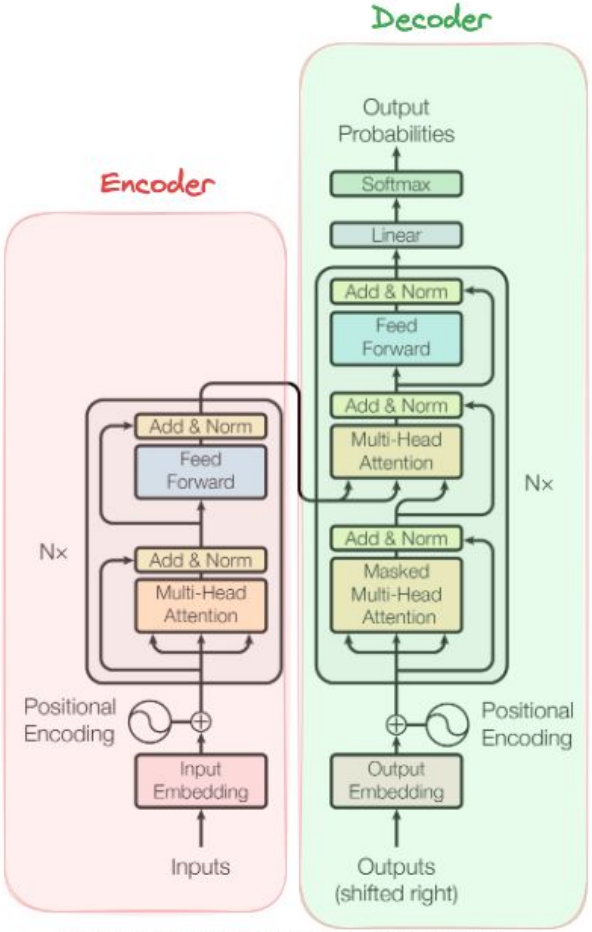
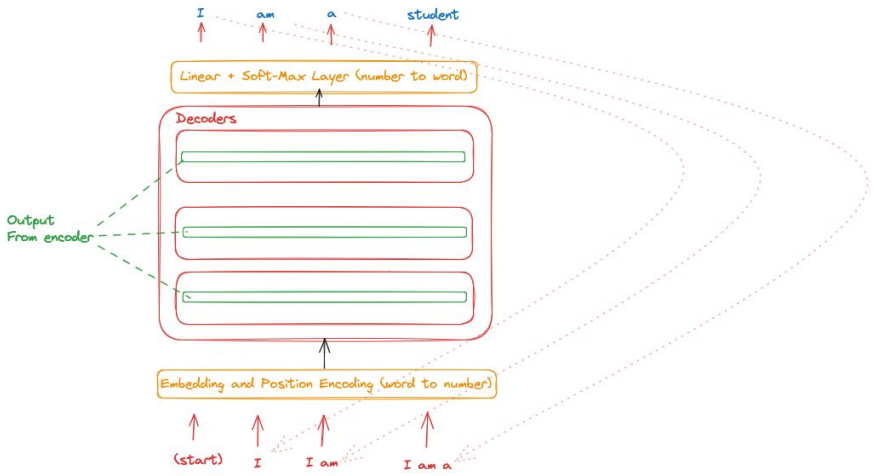


Figure 1: The Transformer - model architecture.

Decoders

Decoders generate a response word by word, passing previous words back in for context.

Within the generation, Decoders uses the output from the Encoder to hone in on what it thinks is the best response.



Building Blocks

- Embedding - converting words to numbers/meaning
- Positional Encoding - adding numbers to account for their order
- Encoder - represent the meaning of the sentence based on embedding and encoding
  - Multi-head Attention - calculate meaning using embedding and positioning, multiple times
  - Add & Norm - maintain some input info and stability
  - Feed Forward - add more complexity
- **Decoder - generates output word by word**
  - Masked Multi-head Attention
  - Passing encoder results to Multi-head Attention
- Linear and Softmax

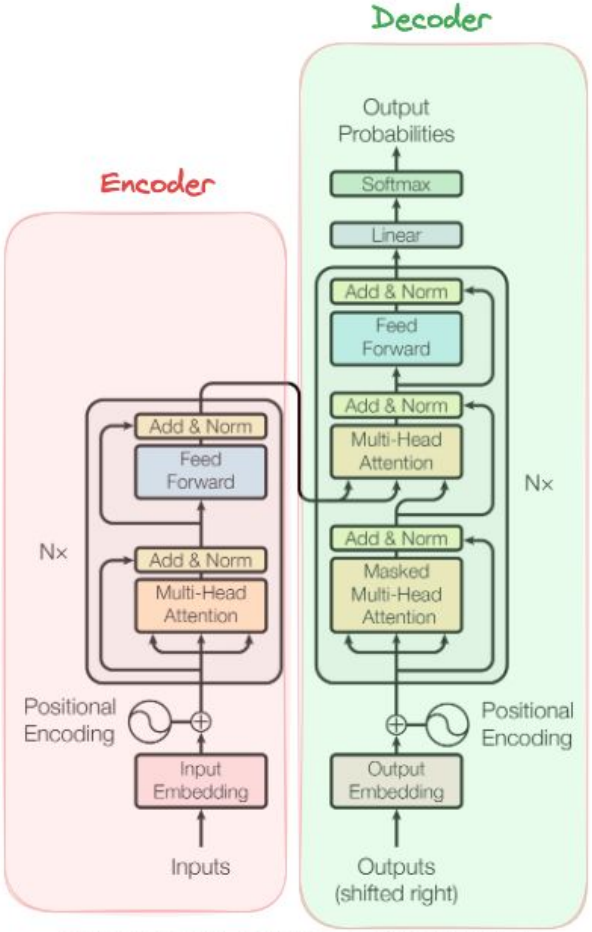


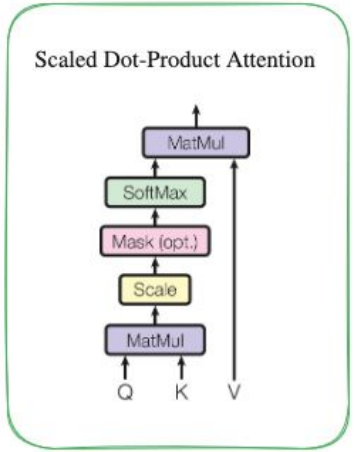
Figure 1: The Transformer - model architecture.

Masked Multi-head Attention

When calculating attention, the mask zeros out the any relation to words after the word we're looking at.

Instead of being able to look at the whole sentence like in encoding, we can only pay attention to words we've seen when generating word by word.

This helps with training our decoder to be the best at predicting.



Building Blocks

- Embedding - converting words to numbers/meaning
- Positional Encoding - adding numbers to account for their order
- Encoder - represent the meaning of the sentence based on embedding and encoding
  - Multi-head Attention - calculate meaning using embedding and positioning, multiple times
  - Add & Norm - maintain some input info and stability
  - Feed Forward - add more complexity
- Decoder - generates output word by word
  - **Masked Multi-head Attention - practice generating words by hiding knowledge about words that come after**
  - Passing encoder results to Multi-head Attention
- Linear and Softmax

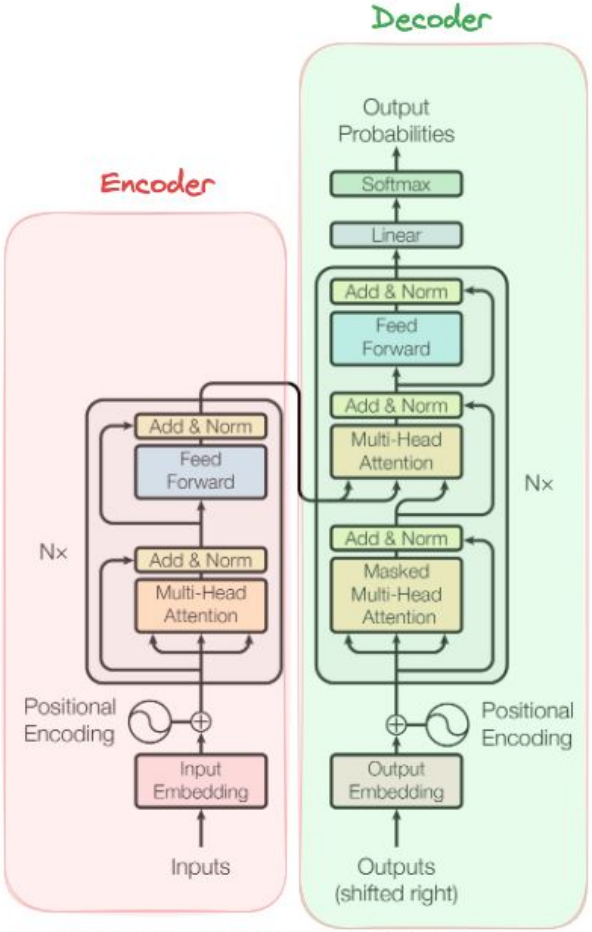
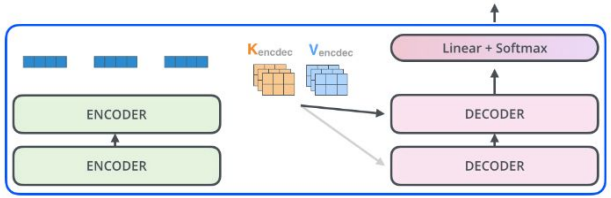


Figure 1: The Transformer - model architecture.

Passing encoder results to Decoder

By passing the output of the Encoder to the Decoder, it can attend to all positions in the input sequence.

There are also pre-trained weights that transform the encoder output into Keys and Values for the decoder.



Building Blocks

- Embedding - converting words to numbers/meaning
- Positional Encoding - adding numbers to account for their order
- Encoder - represent the meaning of the sentence based on embedding and encoding
  - Multi-head Attention - calculate meaning using embedding and positioning, multiple times
  - Add & Norm - maintain some input info and stability
  - Feed Forward - add more complexity
- Decoder - generates output word by word
  - Masked Multi-head Attention - practice generating words by hiding knowledge about words that come after
  - **Passing encoder results to Multi-head Attention - gives attention to the input when generating outputs**
- Linear and Softmax

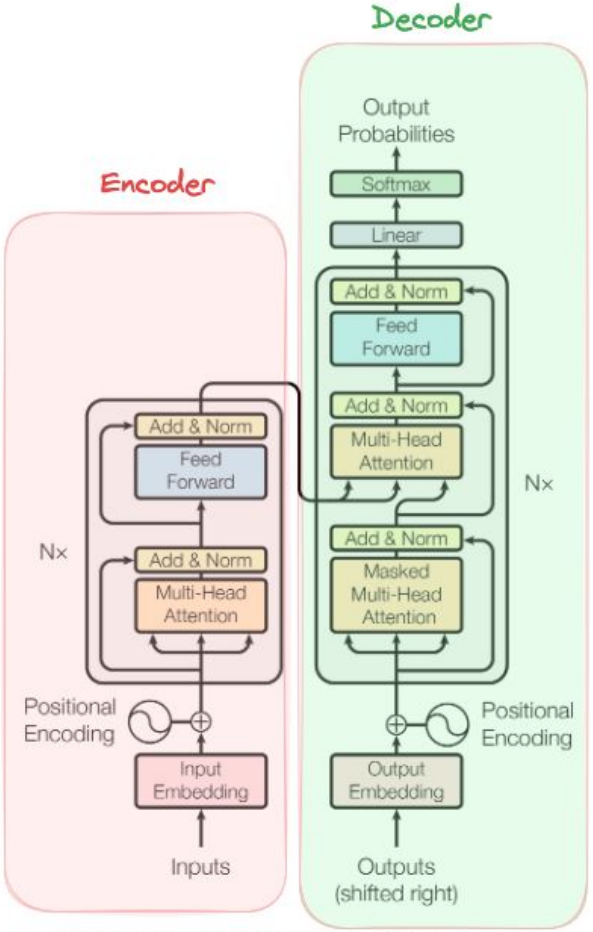


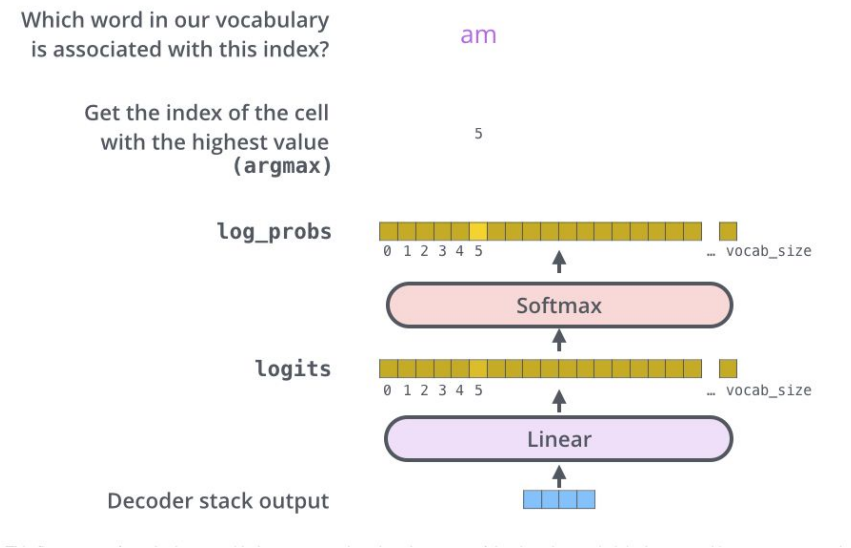
Figure 1: The Transformer - model architecture.

Linear and Softmax

The final two layers determine what word we output.

The Linear Layer maps the decoder result into a list of scores.

The Softmax layer turns the scores into probabilities and the highest one is chosen as the next word.



Building Blocks

- Embedding - converting words to numbers/meaning
- Positional Encoding - adding numbers to account for their order
- Encoder - represent the meaning of the sentence based on embedding and encoding
  - Multi-head Attention - calculate meaning using embedding and positioning, multiple times
  - Add & Norm - maintain some input info and stability
  - Feed Forward - add more complexity
- Decoder - generates output word by word
  - Masked Multi-head Attention - practice generating words by hiding knowledge about words that come after
  - Passing encoder results to Multi-head Attention - gives attention to the input when generating outputs
- **Linear and Softmax - determine what the next word is**

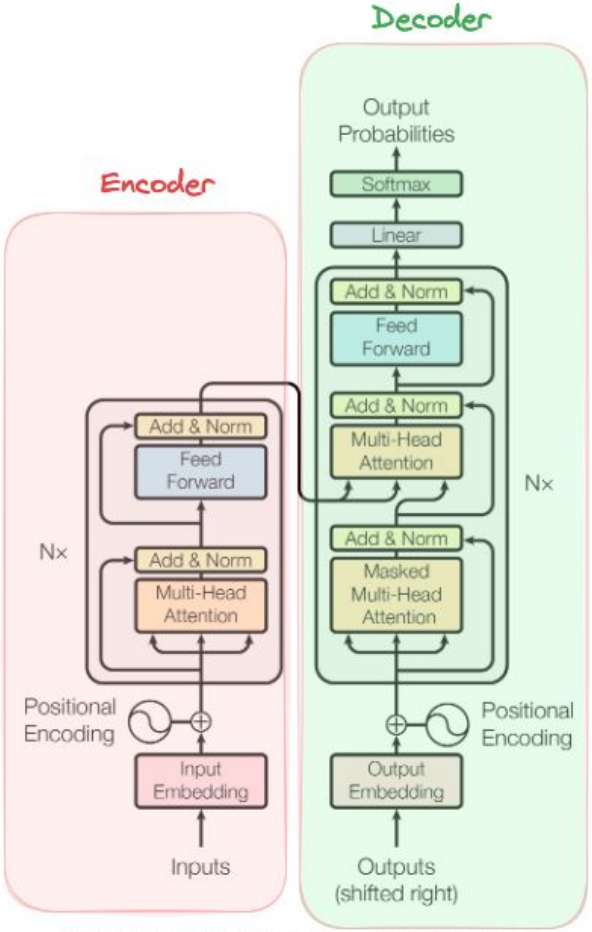


Figure 1: The Transformer - model architecture.

Hope it helped!

Sources:

- Attention Is All You Need Paper
- The Illustrated Transformer
- Introduction to Transformers w/ Andrej Karpathy
- ChatGPT

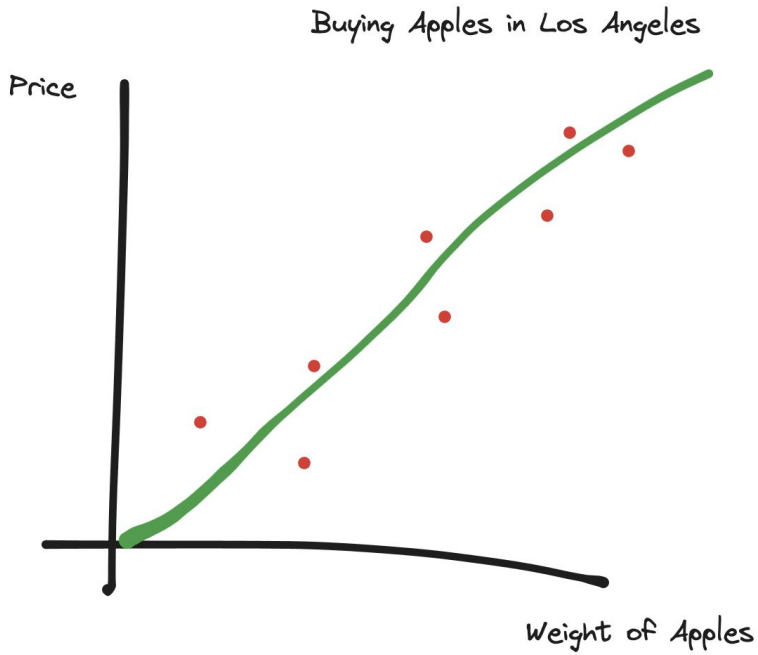
Roger Lam

https://www.linkedin.com/in/lam-roger/

Linearity

We use math equations to represent real life.

Sometimes things are simple and can be estimated by a straight line.

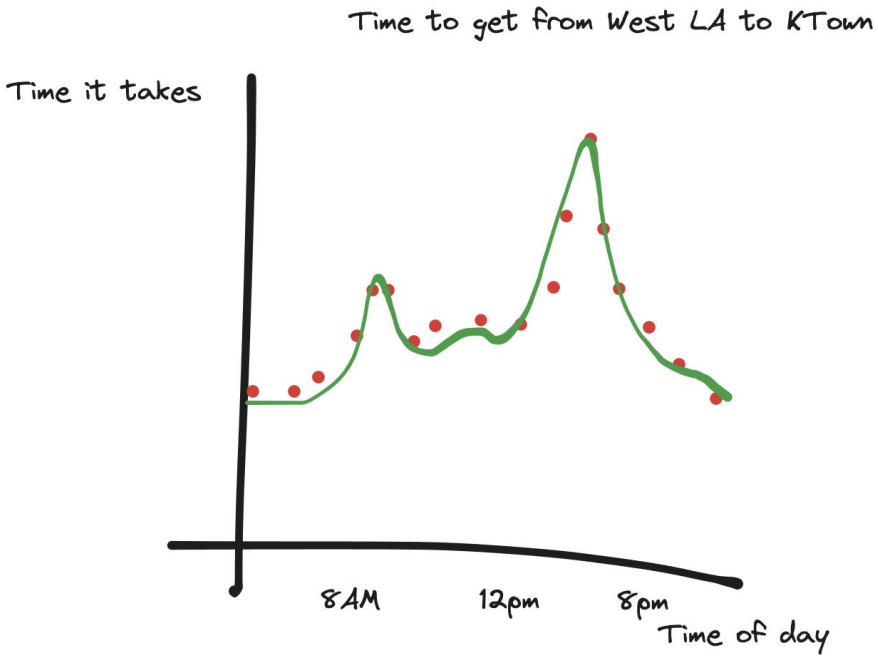


Non-linearity

Many times life is more complicated than just straight lines

You need curves.

The curves let you model more complex things.



Lines and Curves x 1,000

But instead of one line or curve, let's have a thousand (or a million or a billion) lines and curves.

We can estimate based on a bunch of things.

Also, we can let computers figure out what is important.