Deep Learning

16 Self-Attention & Transformers

Dr. Konda Reddy Mopuri
Dept. of AI, IIT Hyderabad
Jan-May 2023

Motivation

1. Why does one need to think beyond LSTMs?

Motivation

1. Why does one need to think beyond LSTMs?
2. Sequential processing doesn't allow parallelization
   - Path length $= \mathbb{O}(n)$
   - RNNs need at most $\mathbb{O}(n)$ sequential computations to access each element

Motivation

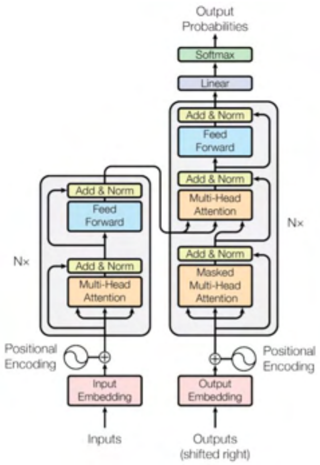
1. Despite the LSTM/GRU, RNNs need attention to deal with long-range dependencies

Motivation

1. Despite the LSTM/GRU, RNNs need attention to deal with long-range dependencies
2. Since attention enables accesses to any state, do we need RNNs?

Transformers

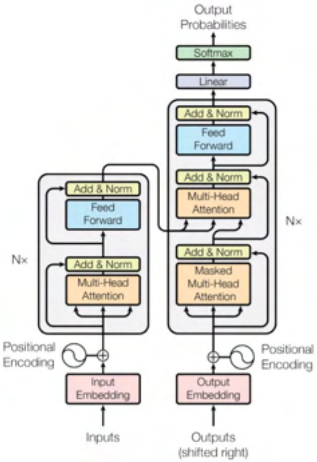
1. Introduced by Vaswani et al. NeurIPS 2017



Transformers

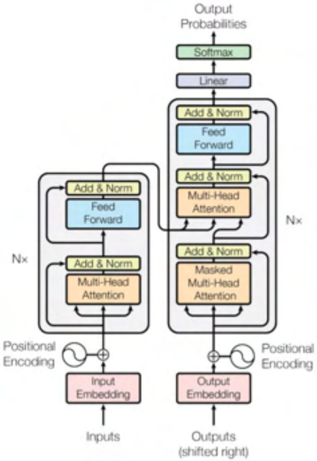
భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

1. Introduced by Vaswani et al. NeurIPS 2017
2. Sequnce to sequence modelling without RNNs



Transformers

1. Introduced by Vaswani et al. NeurIPS 2017
2. Sequnce to sequence modelling without RNNs
3. Transformer model is built on self-attention (no recurrent architectures!)

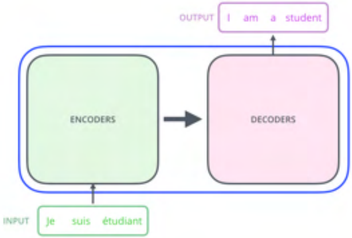


Transformers

Credits: Jay Alammar

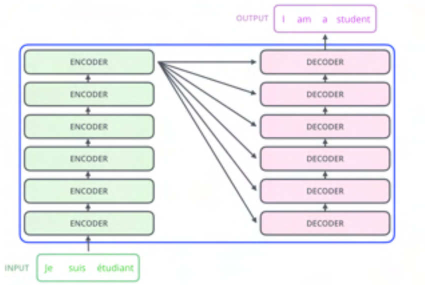
Transformers

Credits: Jay Alammar



Credits: Jay Alammar

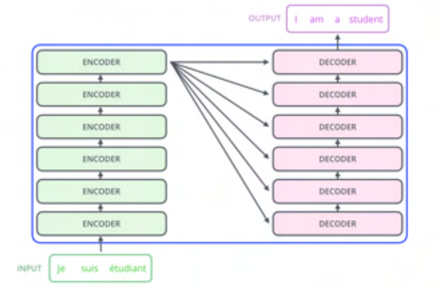
Transformers



Credits: Jay Alammar

1. Encoding module has a stack of encoders

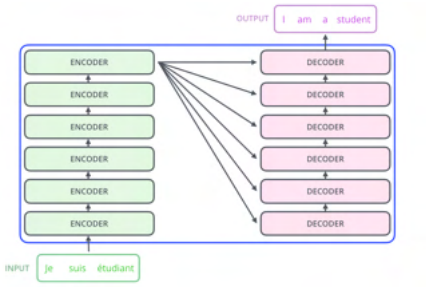
Transformers



Credits: Jay Alammar

1. Encoding module has a stack of encoders
2. Same structure different parameters

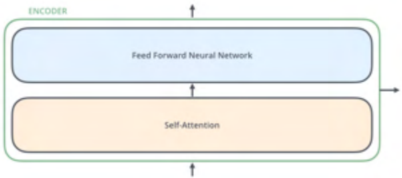
Transformers



Credits: Jay Alammar

1. Encoding module has a stack of encoders
2. Same structure different parameters
3. Similarly the decoding module (same number of components in the stack as encoder)

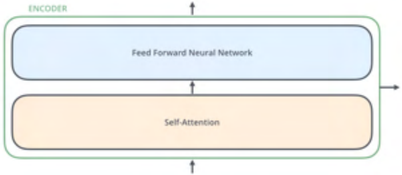
Transformers



Credits: Jay Alammar

1. Encoder first has a self-attention layer

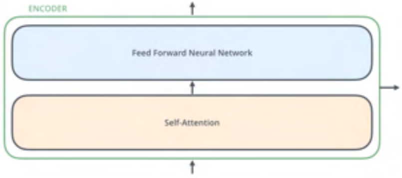
Transformers



Credits: Jay Alammar

1. Encoder first has a self-attention layer
2. Looks at the other words while encoding a specific word

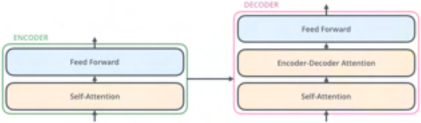
Transformers



Credits: Jay Alammar

1. Encoder first has a self-attention layer
2. Looks at the other words while encoding a specific word
3. Next a (same) feed-forward NN is applied at all positions

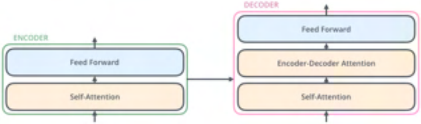
Transformers



Credits: Jay Alammar

1. Decoder also has both the layers

Transformers



Credits: Jay Alammar

1. Decoder also has both the layers
2. But, in the middle it has an encoder-decoder attention layer

Transformers-Encoding

1. Start with turning each word into a vector at the bottom-most encoder

Credits: Jay Alammar

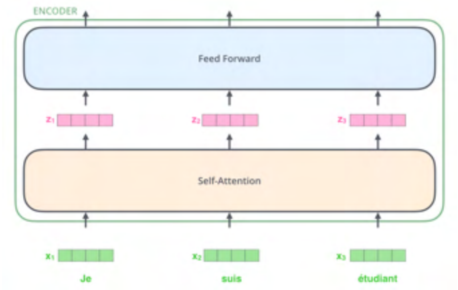
Transformers-Encoding

1. Start with turning each word into a vector at the bottom-most encoder
2. Others receive a list of vectors from the encoder immediately below

Credits: Jay Alammar

Transformers-Encoding

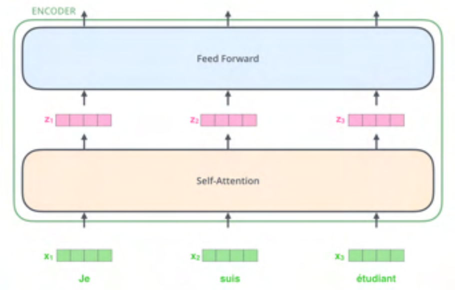
1. Each word flows through the two layers of the encoder through its own path



Credits: Jay Alammar

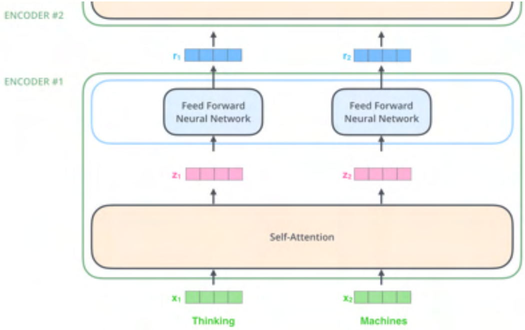
Transformers-Encoding

1. Each word flows through the two layers of the encoder through its own path
2. Self-attention layer has dependencies among them, however, the path length is $\mathbb{O}(1)$



Credits: Jay Alammar

Transformers-Encoding



Credits: Jay Alammar

Self-Attention

1. The animal didn't cross the street because it was too tired
2. The animal didn't cross the street because it was too wide

Self-Attention

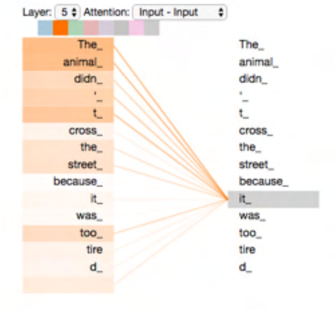
1. `The animal didn't cross the street because it was too tired`
2. `The animal didn't cross the street because it was too wide`
3. What does 'it' refer to?

Self-Attention

1. `The animal didn't cross the street because it was too tired`
2. `The animal didn't cross the street because it was too wide`
3. What does 'it' refer to?
4. Easy for humans, but not so much for the traditional Seq2Seq models

Self-Attention

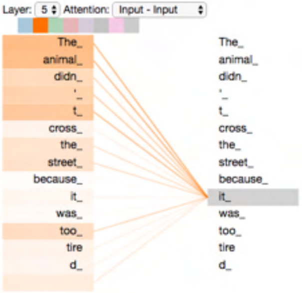
1. As the model processes each word, self-attention attends other positions in the i/p sequence to encoder better



Credits: Jay Alammar

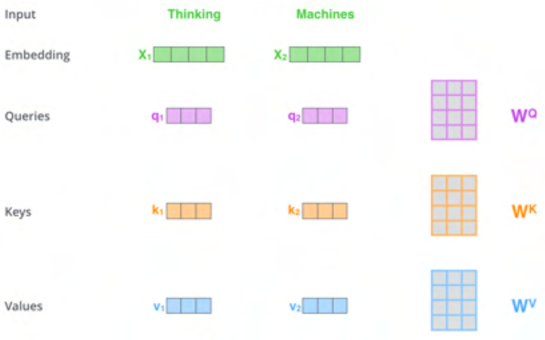
Self-Attention

1. As the model processes each word, self-attention attends other positions in the i/p sequence to encoder better
2. Unlike RNNs, here we don't keep hidden states from previous positions!



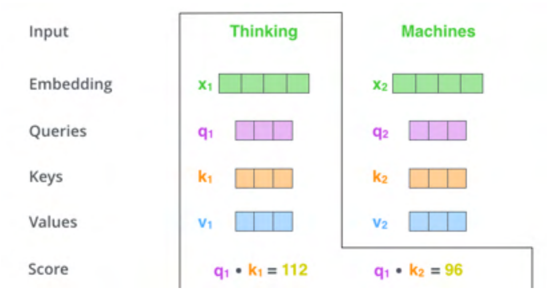
Credits: Jay Alammar

Self-Attention



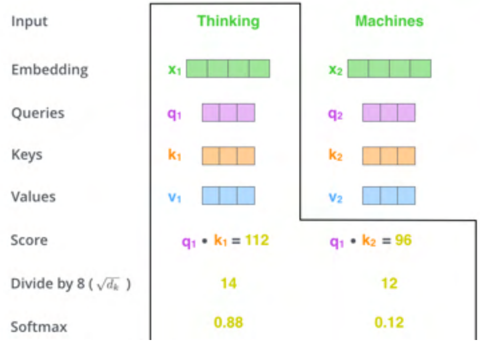
Credits: Jay Alammar

Self-Attention



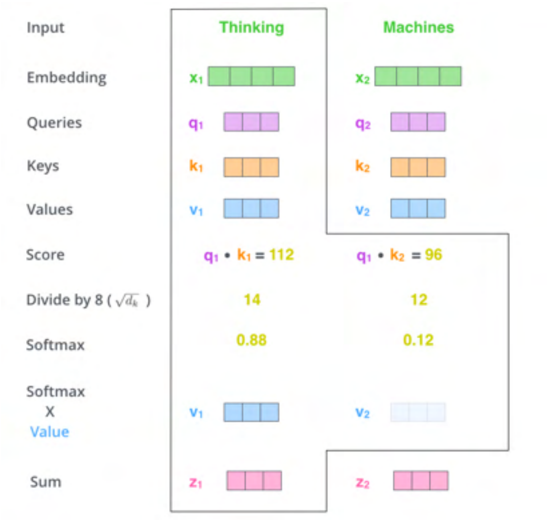
Credits: Jay Alammar

Self-Attention



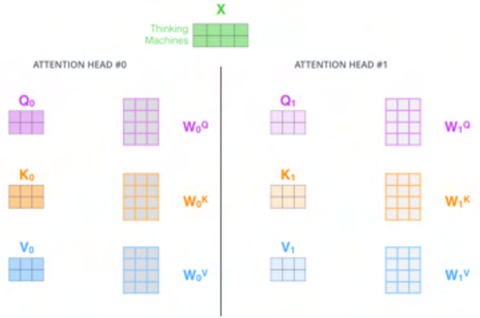
Credits: Jay Alammar

Self-Attention



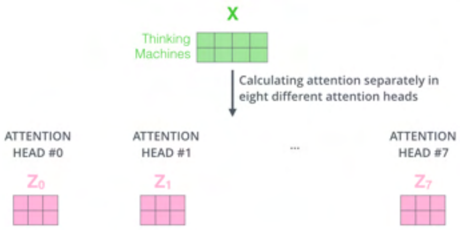
Credits: Jay Alammar

Multi-headed Self-Attention



Credits: Jay Alammar

Multi-headed Self-Attention



Credits: Jay Alammar

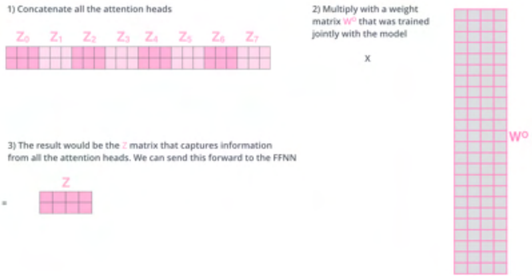
Multi-headed Self-Attention

1. Expands the model's ability to focus on different relevant positions in the i/p

Multi-headed Self-Attention

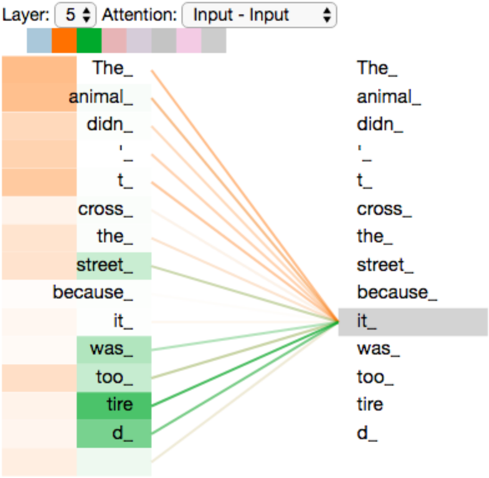
1. Expands the model's ability to focus on different relevant positions in the i/p
2. Enables different 'representational subspace'

Multi-headed Self-Attention



Credits: Jay Alammar

Multi-headed Self-Attention



Credits: Jay Alammar

Positional Encoding

1. Unlike RNN and CNN encoders, attention encoder outputs don't depend on the order

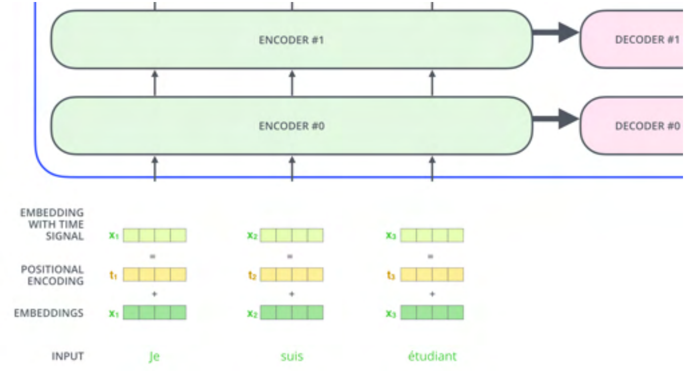
Positional Encoding

1. Unlike RNN and CNN encoders, attention encoder outputs don't depend on the order
2. However, order the sequence conveys vital information in some applications

Positional Encoding

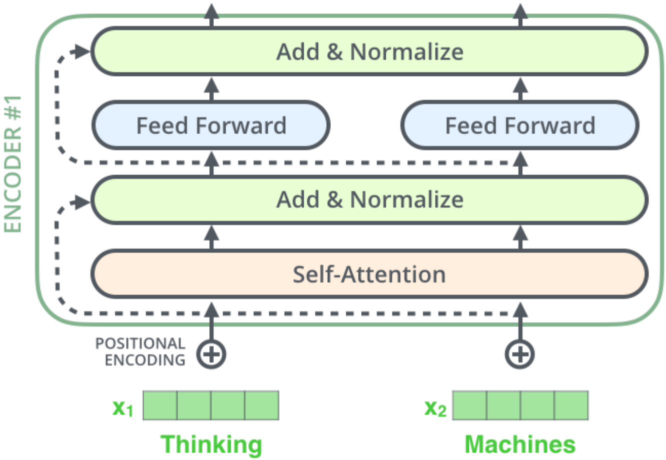
1. Unlike RNN and CNN encoders, attention encoder outputs don't depend on the order
2. However, order the sequence conveys vital information in some applications
3. Solution: Add positional information of the i/p words into their embedding vectors

Positional Encoding



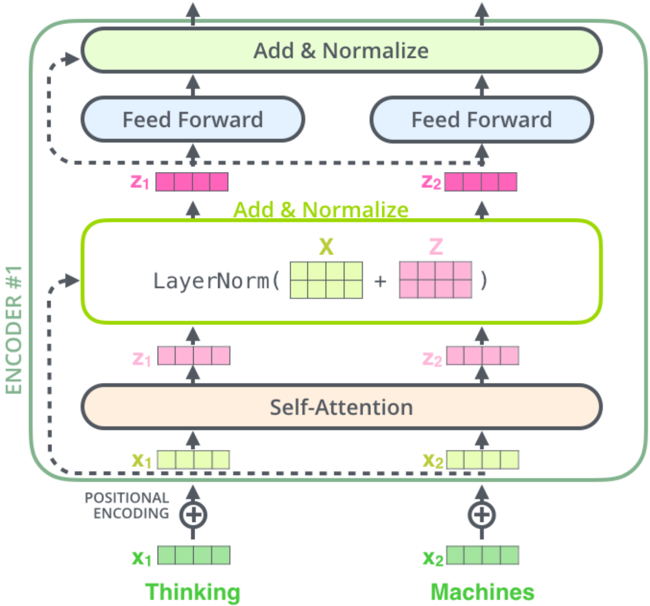
Credits: Jay Alammar

Residuals in the Encoder

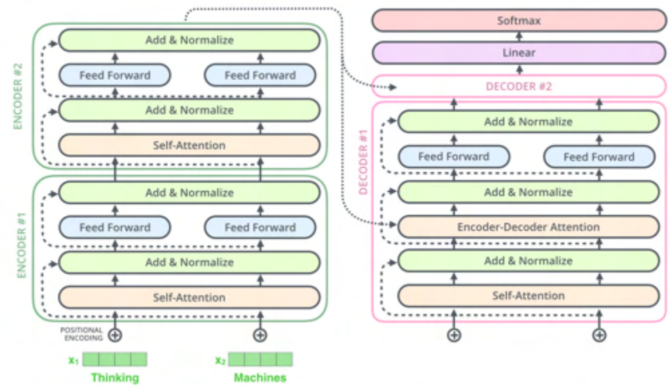


Credits: Jay Alammar

Residuals in the Encoder



Tranformer-Decoder



Credits: Jay Alammar

Transformer-Decoder

1. Self-attention here works in a slightly different way $\rightarrow$ masks the future positions

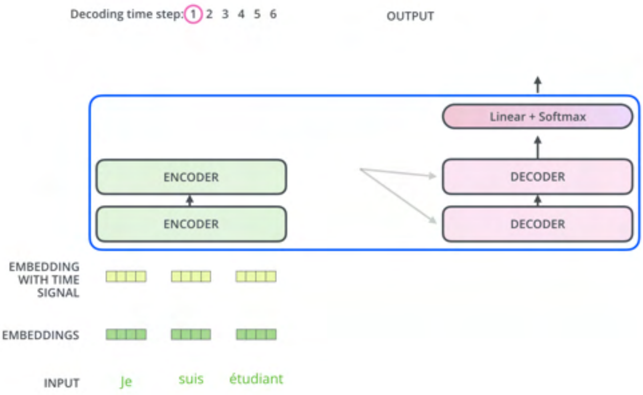
Transformer-Decoder

1. Self-attention here works in a slightly different way $\rightarrow$ masks the future positions
2. Uses the top encoder's K and V vectors for its' encoder-decoder attention

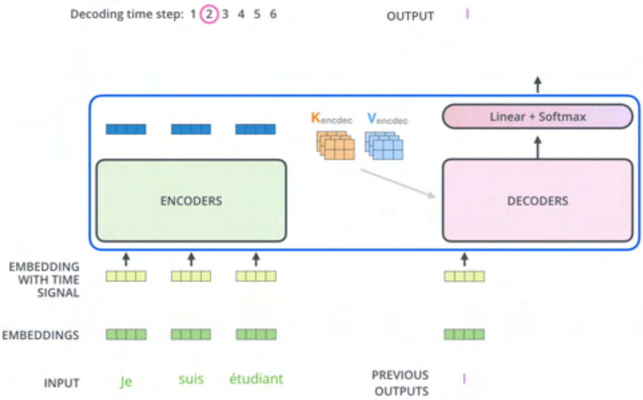
Transformer-Decoder

1. Self-attention here works in a slightly different way $\rightarrow$ masks the future positions
2. Uses the top encoder's K and V vectors for its' encoder-decoder attention
3. Encoder-decoder attention layer borrows the queries from the layer below it

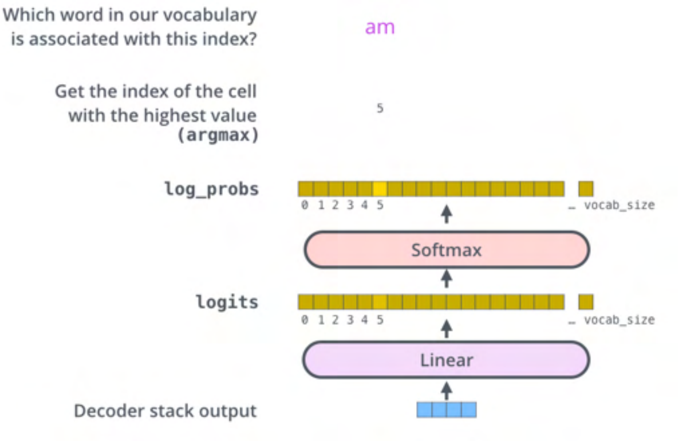
Transformer-Decoder



Transformer-Decoder



Final o/p



Credits: Jay Alammar