Attention Mechanism and Transformer model

Introduction

NLP requires efficient sequence modeling for building state-of-the-art language model

RNN was tried in the beginning but it suffered from vanishing and exploding gradient

LSTM and GRU was tried later but that too could not find relationship between words which are quite distant apart

e.g.: Rama, being the eldest son of Dasaratha and also being the conqueror of Ravana's lanka was undoubtedly the best candidate for being the king and had actually become the king of Ayodhya

Attention is all you need...

The famous paper by Ashish Vaswani et at. (2017)

Language model should weigh different words in a sequence differently to understand the meaning properly

Attention model was built on the basis of the above philosophy to give attention to important words in a sequence

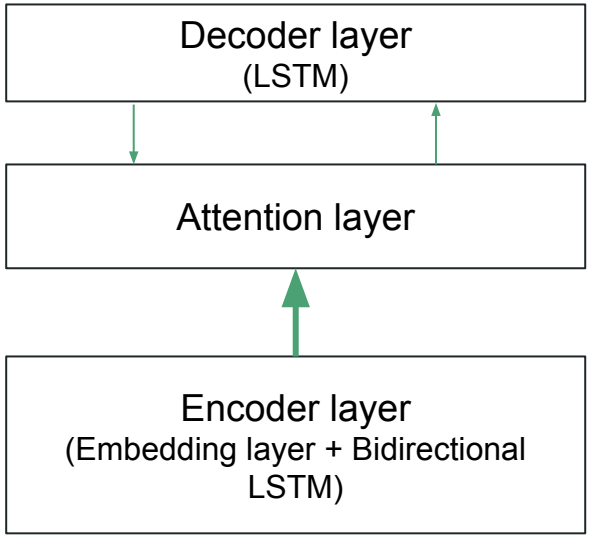
It is a neural network model that learns how to give attention to different words in a sequence

Attention model surpassed solely LSTM or solely GRU based models by large margin to claim the state-of-the-art-model recognition

The architecture

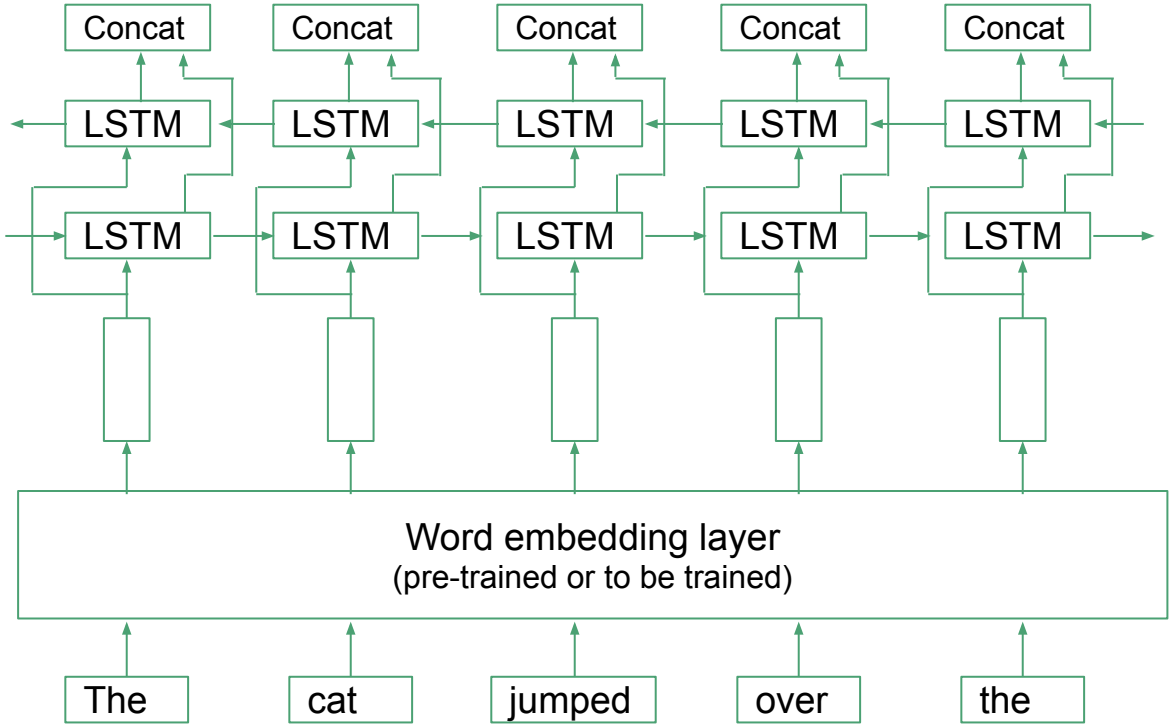
Attention model comprises of 3 different layers

- The encoding layer
- The attention layer
- The decoder layer

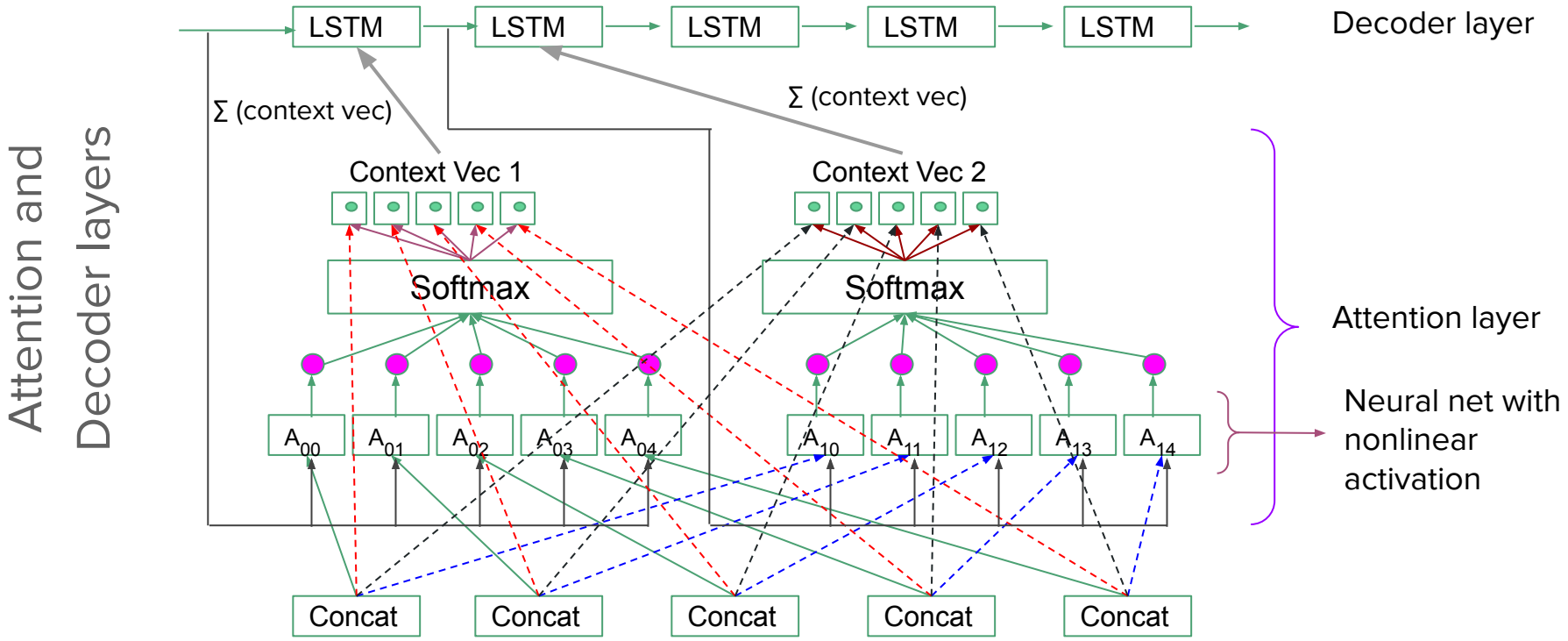


Getting Deeper...

Encoder layer

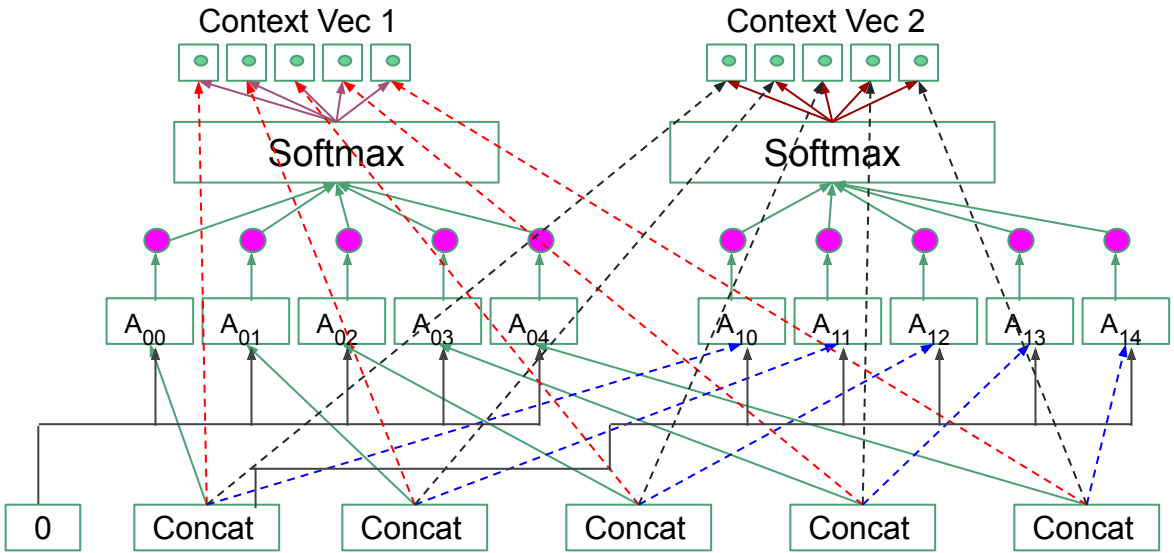


Getting Deeper...



Self Attention

No decoding layer



Attention without RNN

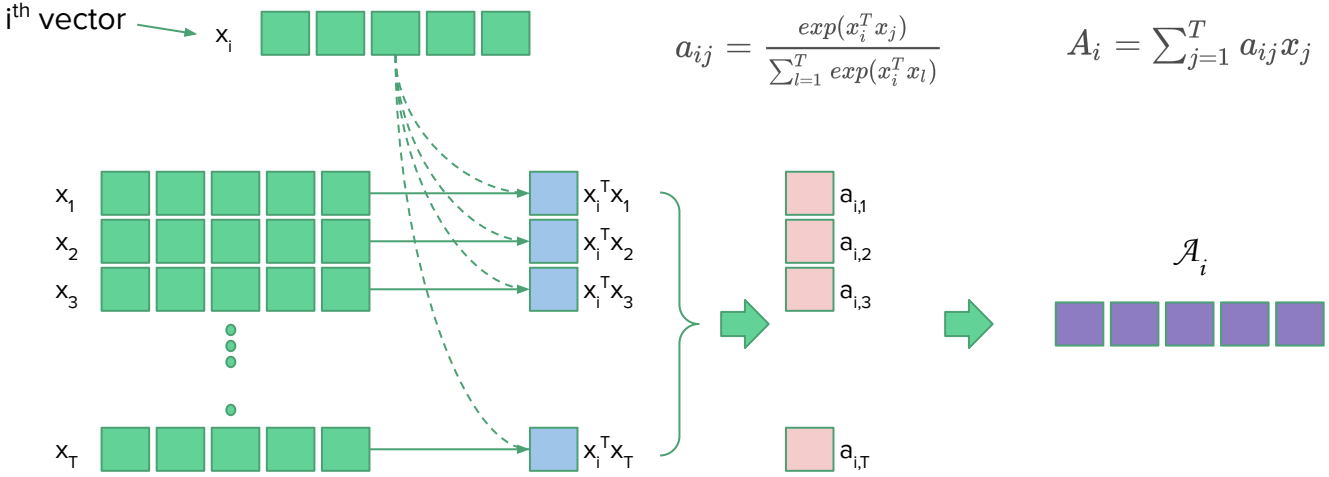
Step 1: Compute dot product

Step 2: Compute softmax

Step 3: Compute context vector

$$a_{ij} = \frac{exp(x_i^T x_j)}{\sum_{l=1}^{T} exp(x_i^T x_l)}$$

$$A_i = \sum_{j=1}^{T} a_{ij} x_j$$



Basic attention model without trainable parameters (not good..!)

Attention without RNN (contd...)

The basic attention model without trainable parameters can be made powerful trainable attention model by incorporating three trainable matrices

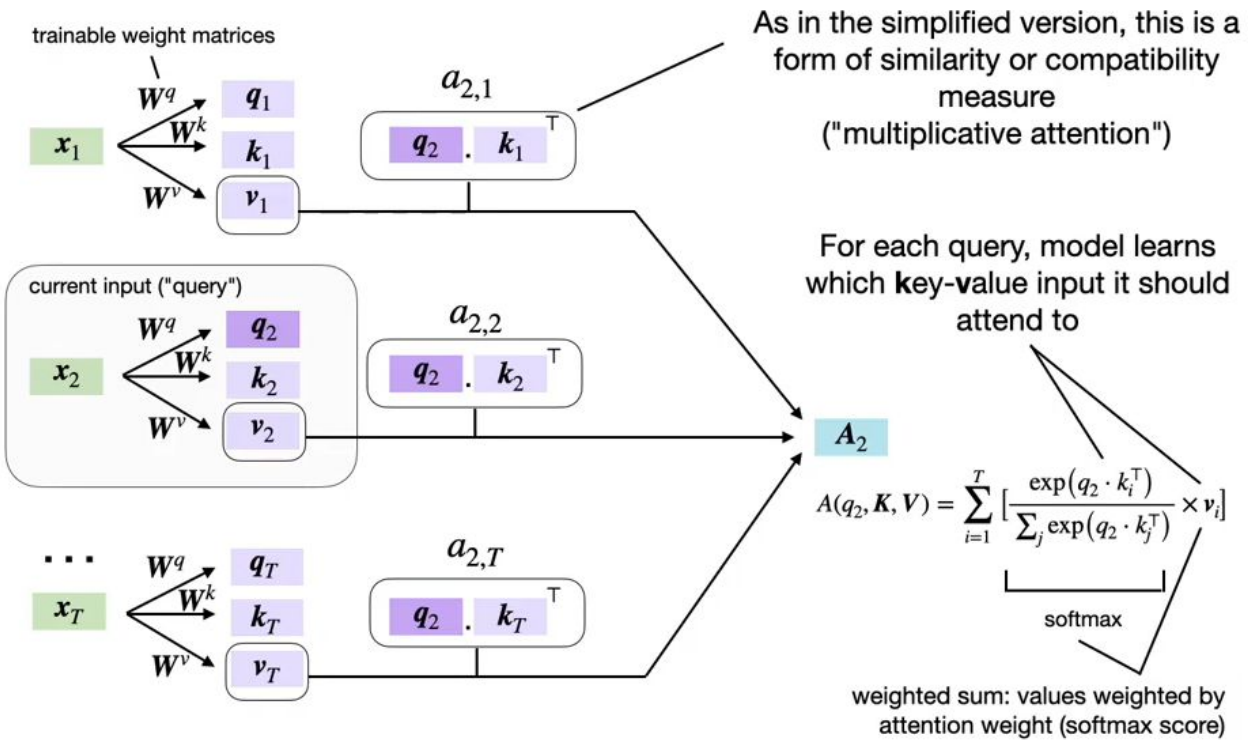
- Query matrix
- Key matrix
- Value matrix

$$\text{Query:} \quad W_q x_i$$

$$\text{Key:} \quad W_k x_i$$

$$\text{Value:} \quad W_v x_i$$

Attention without RNN (contd...)

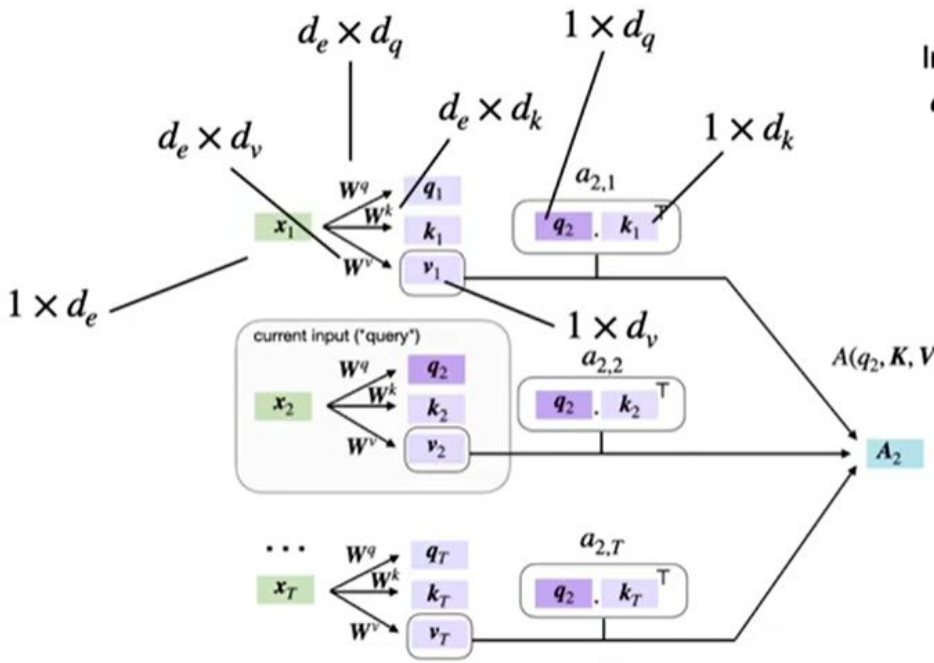


Attention without RNN (contd...)

d_e = embedding size (original transformer = 512)

where $d_q = d_k$

In original transformer, $d_q = d_v$ as well

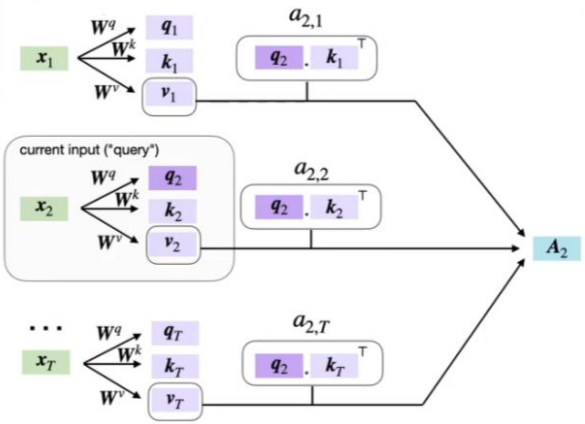


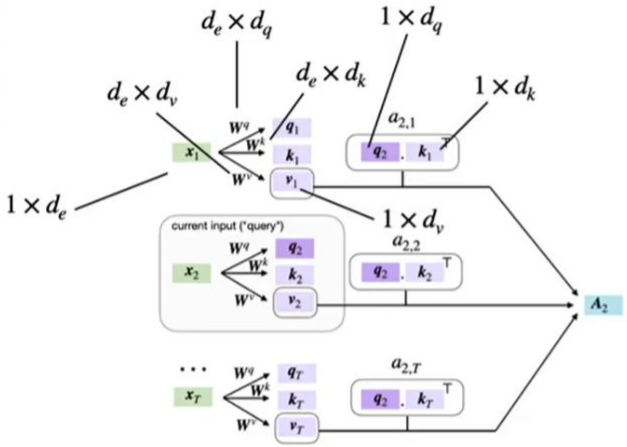
$$A(q_2, K, V) = \sum_{i=1}^{T} \left[\underbrace{\frac{\exp(q_2 \cdot k_i^T)}{\sum_j \exp(q_2 \cdot k_j^T)}}_{\text{softmax}} \times v_i \right]$$

1×1

$1 \times d_v$

Attention without RNN (Matrix form)





$$Q \in \mathbb{R}^{T \times d_q}$$
$$K \in \mathbb{R}^{T \times d_k}$$
$$V \in \mathbb{R}^{T \times d_v}$$

"attention matrix"

$$A(Q,K,V) = \overbrace{\text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)}^{T \times T} V$$

$T \times d_v$

"attention-based embedding"

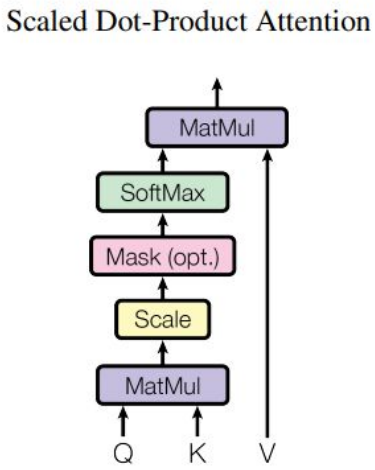
This model was proposed by Ashish Vaswani et at.

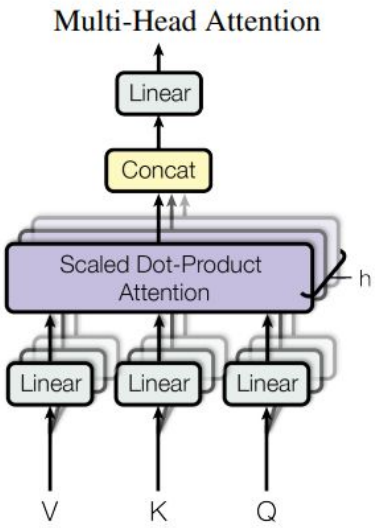
Scaling to prevent high values in softmax preventing saturation

Multi-head Attention

Use multiple sets of W_q, W_k and W_v (8 was used in the original paper)

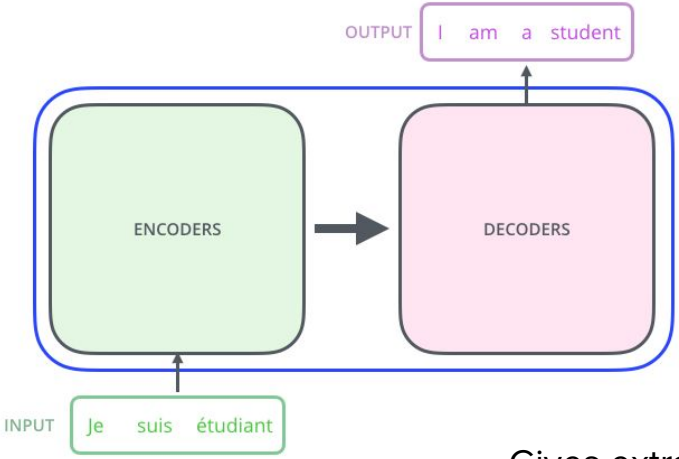
Each set learns its own set of parameters





Transformer model

A transformer is basically a combination of encoder decoder modules



$N_x = 6$

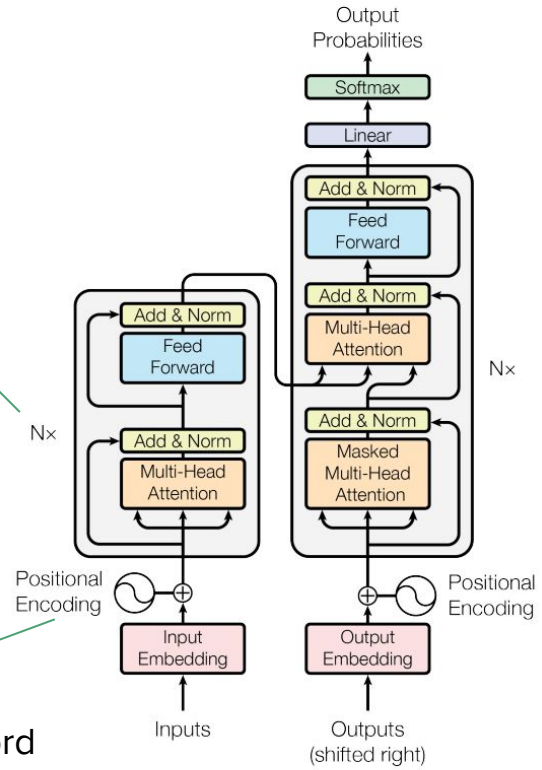


Figure 1: The Transformer - model architecture.

Gives extra piece of info regarding the position of a word in the sequence

Transformer model

The encoder component is essentially a stack of encoders having feed forward neural network and self-attention layer

Self-attention layer looks at the other words in the input sentence

Output of the last encoder is fed to the decoder component