"Attention Is All You Need" Paper Discussion

Mohammad Sabik Irbaz

Lead Machine Learning Engineer, Pioneer Alpha Ltd. (Full Time)
Lead Machine Learning Engineer, Omdena Inc. (Part Time)
Head Course Instructor, Amar iSchool (Part Time)

CSE Graduate, IUT

Zannatul Naim Shanto

EEE Graduate, BAIUST

NLP Reading Summer ‘21

TABLE OF CONTENTS

TABLE OF CONTENTS

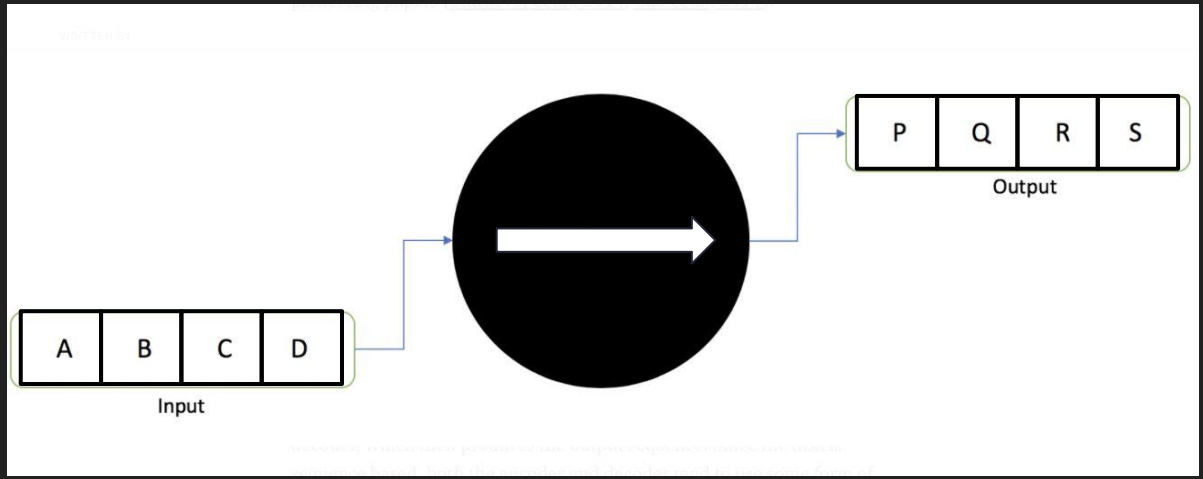
Introduction

- Transformer architecture was first introduced by Vaswani et. al. in “Attention Is All You Need” paper. It was a combined effort of Google Research and Google Brain.
- Right now in the deep learning research and development community transformer is one of the most popular models.
- There are many reasons why this architecture got so popular and was able to take down all the previous SotA Neural Machine Translation approaches.

Generic Translation Model

Sequence of words

Sequence of words



Generic Translation Model

Sequence of words

Sequence of words

Seq2Seq Model

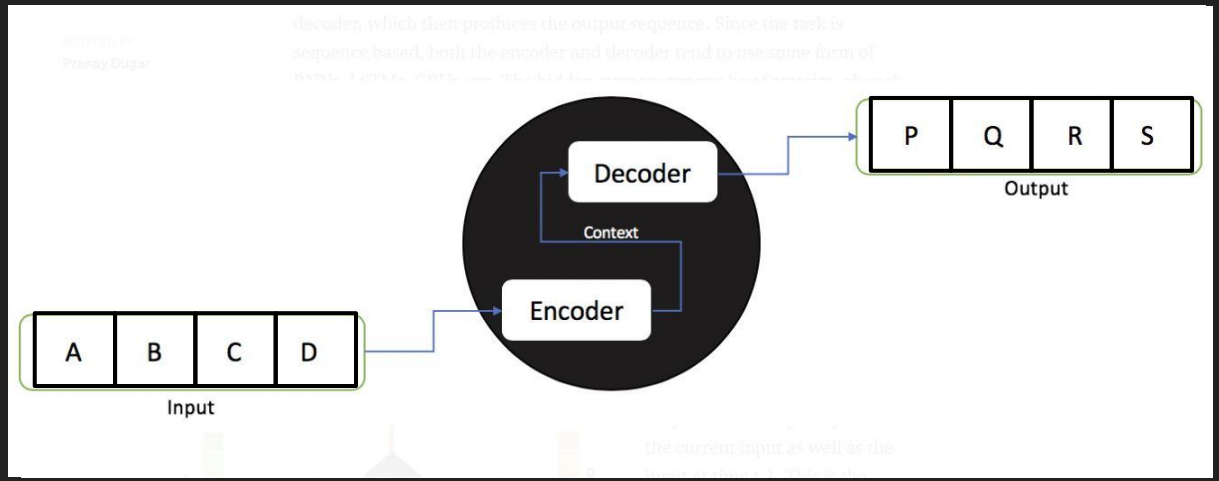


TABLE OF CONTENTS

Bottleneck

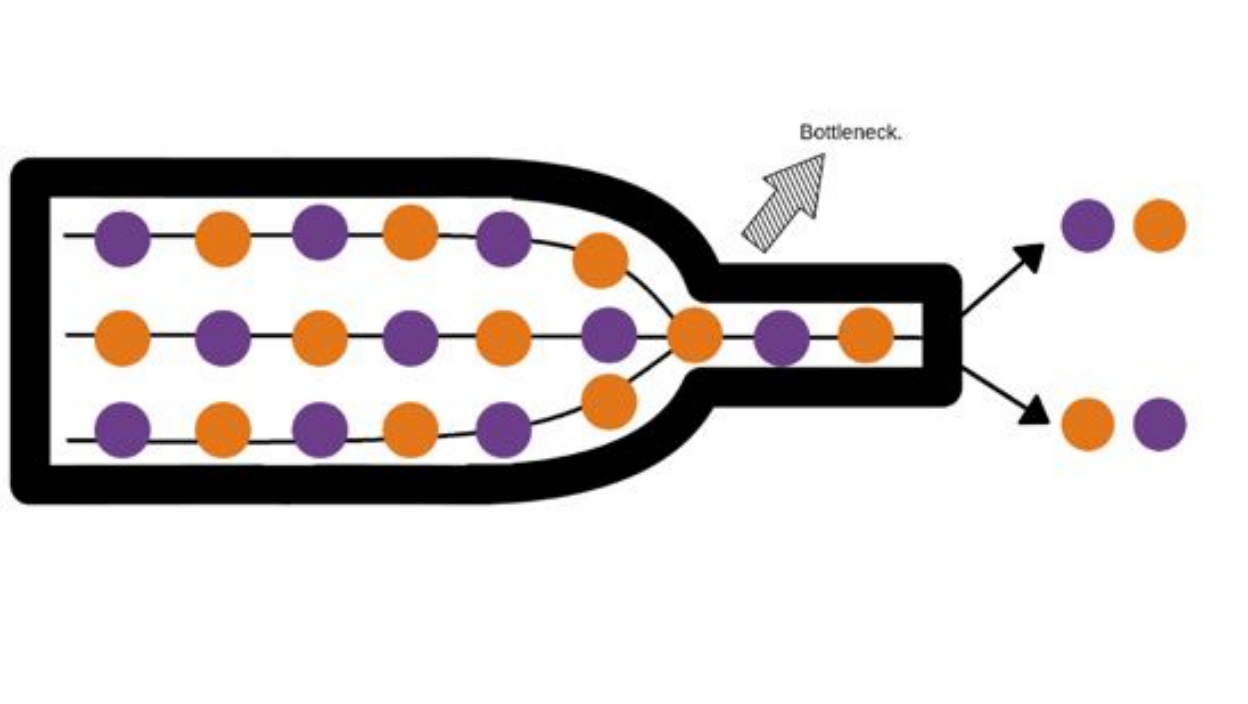
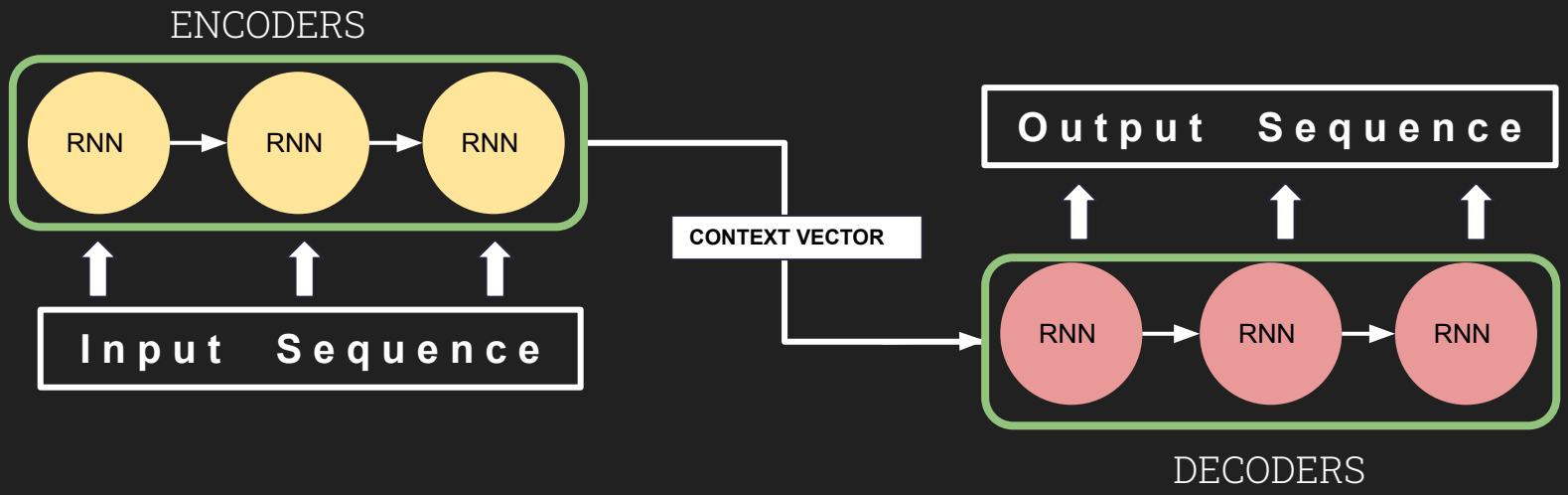


Image taken from Google Photos

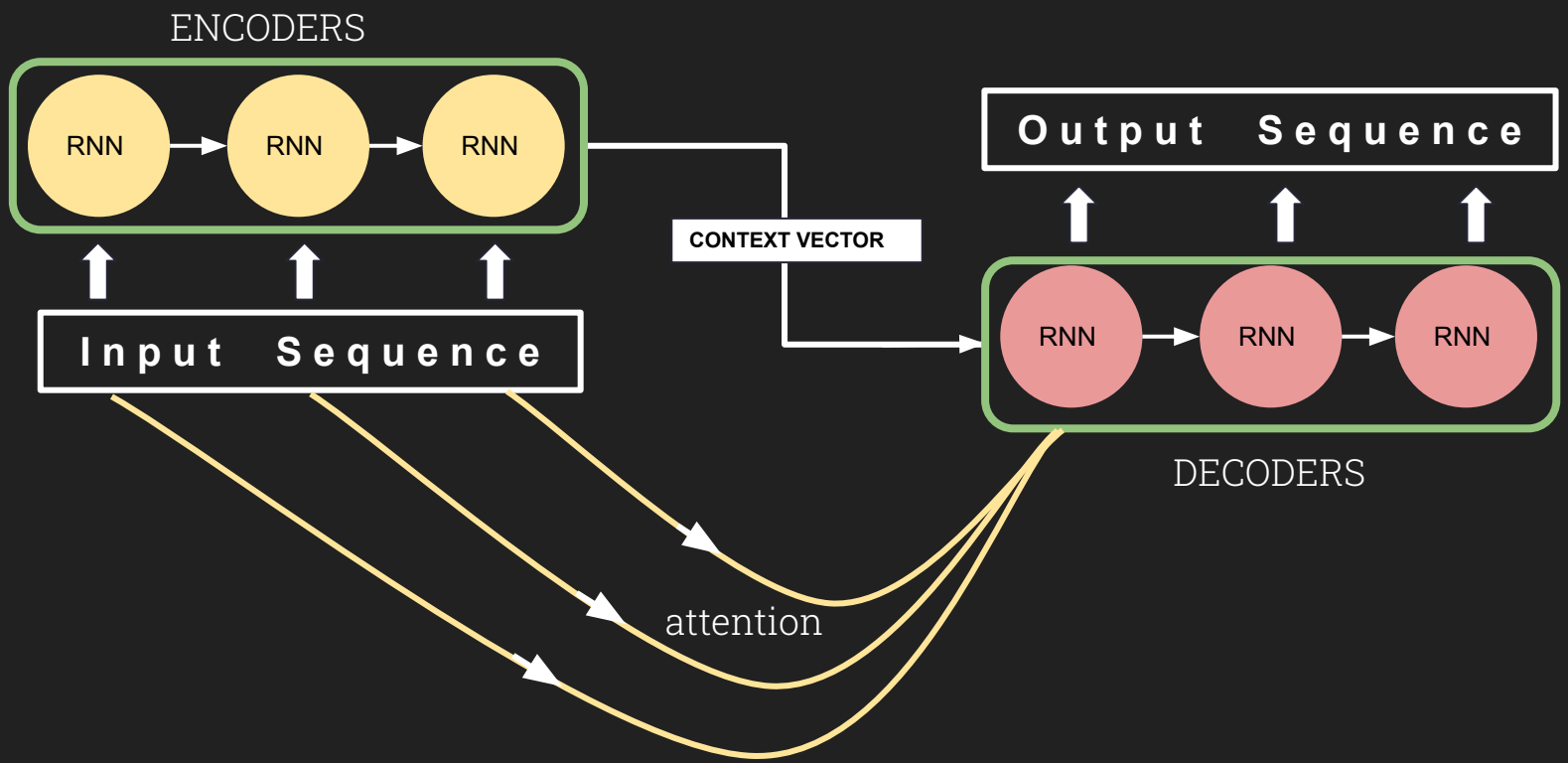
Encoder – Decoder [Sutskever et. al. 2014]



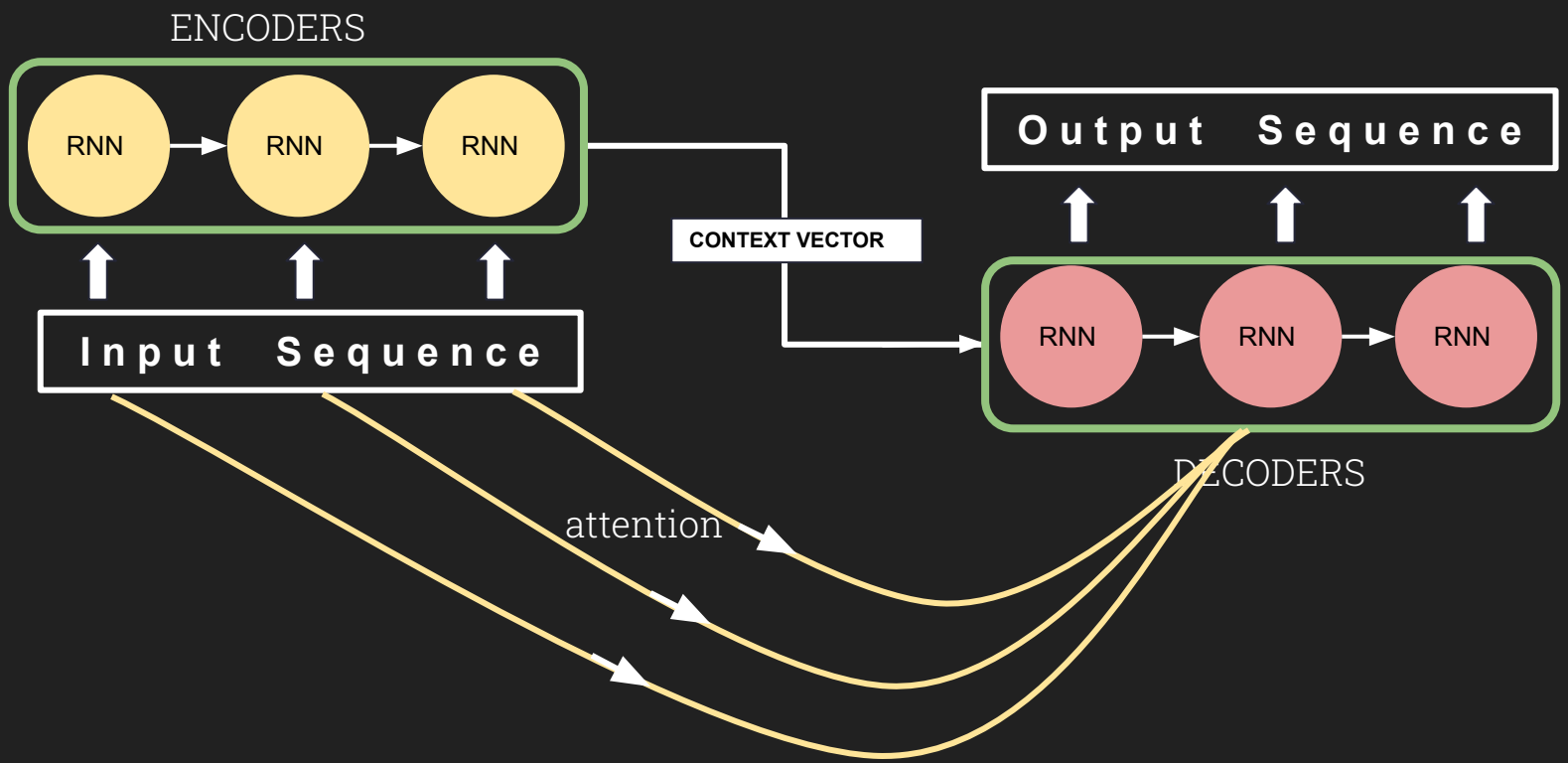
Drawbacks:

- The output sequence relies heavily on the context defined by the hidden state in the final output of the encoder
- In the case of long sequences, there is a high probability that the initial context has been lost by the end of the sequence.

Encoder – Decoder with Attention [Bahdanau et. al., 2015]



Encoder – Decoder with Attention [Bahdanau et. al., 2015]



Encoder – Decoder with Attention [Bahdanau et. al., 2015]

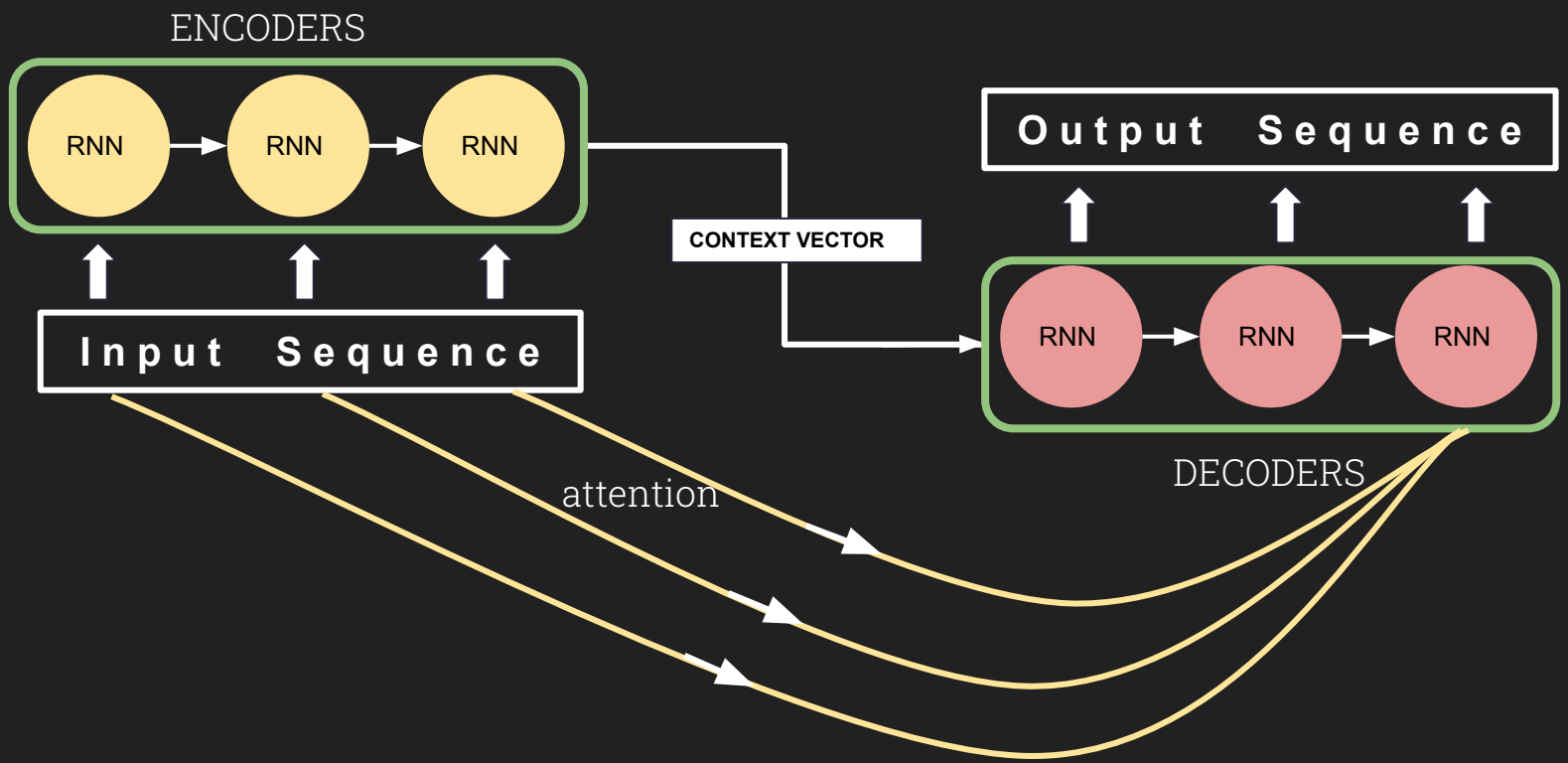
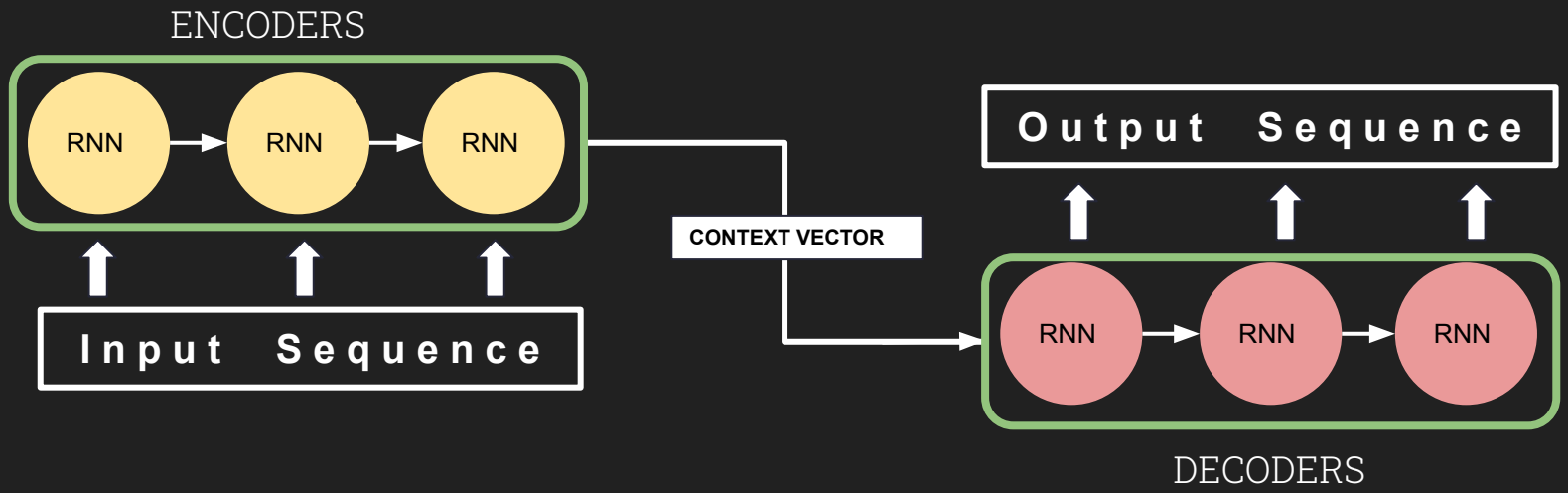


TABLE OF CONTENTS

Data Parallelization Issue

To work with GPU, we need parallel processing.

But, RNNs or LSTMs work sequentially which prevents parallelization.



Transformer [Vaswani et. al., 2017]

Reasons why Transformer got popular :

- ➔ Better long-range dependency modeling.
- ➔ Gets rid of recurrent nets which make the model highly parallelizable and enables faster GPU computation.
- ➔ Multi-head Attention.

Do we need RNNs - LSTMs ?

We do not need RNNs because:

- Now we have transformers which works with much better accuracy for all tasks that RNNs - LSTMs did.

- We can scale up the models using GPUs since parallelization is possible.

Do we need RNNs - LSTMs ?

We do not need RNNs because:

- Now we have transformers which works with much better accuracy for all tasks that RNNs - LSTMs did.
- We can scale up the models using GPUs since parallelization is possible.

But still we need RNNs - LSTMs for

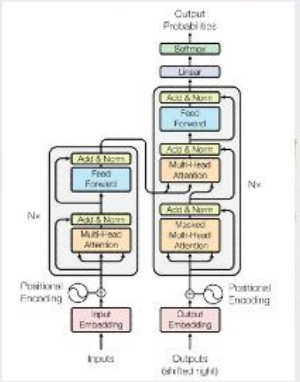
- Real-time use.
- Most of the organizations do not have the infrastructure to train and work with Transformers.

TABLE OF CONTENTS

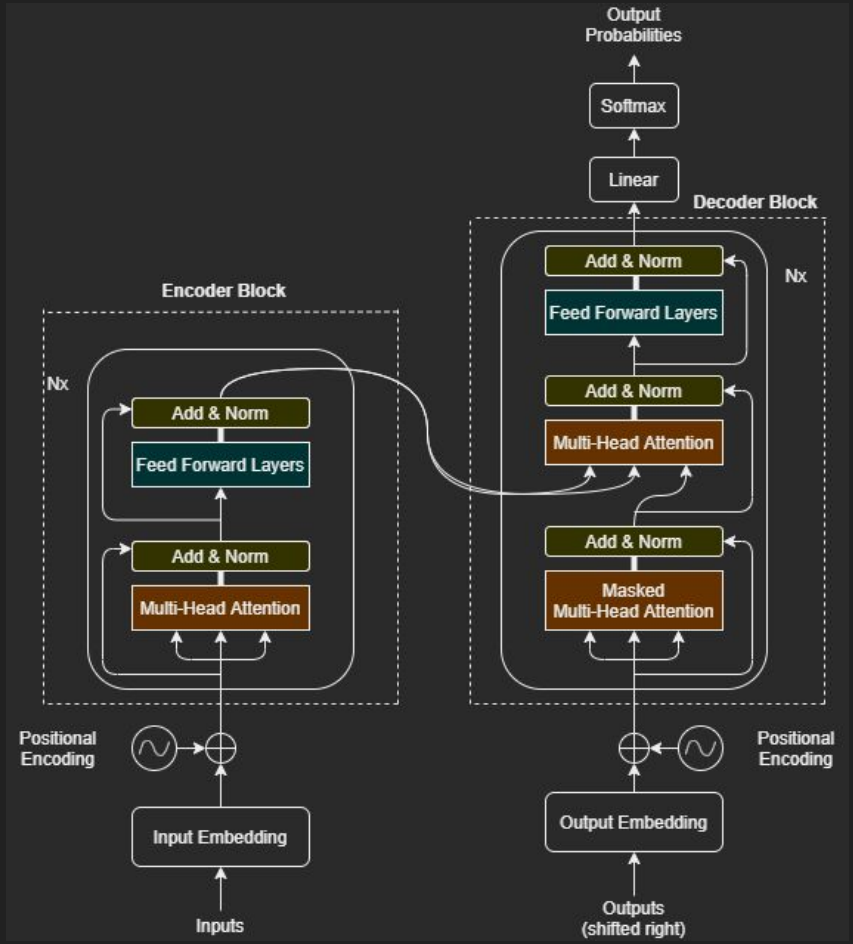
- Me: Mom can we have at home

- Mom: But we already have at home

-  at home:



Transformer Architecture

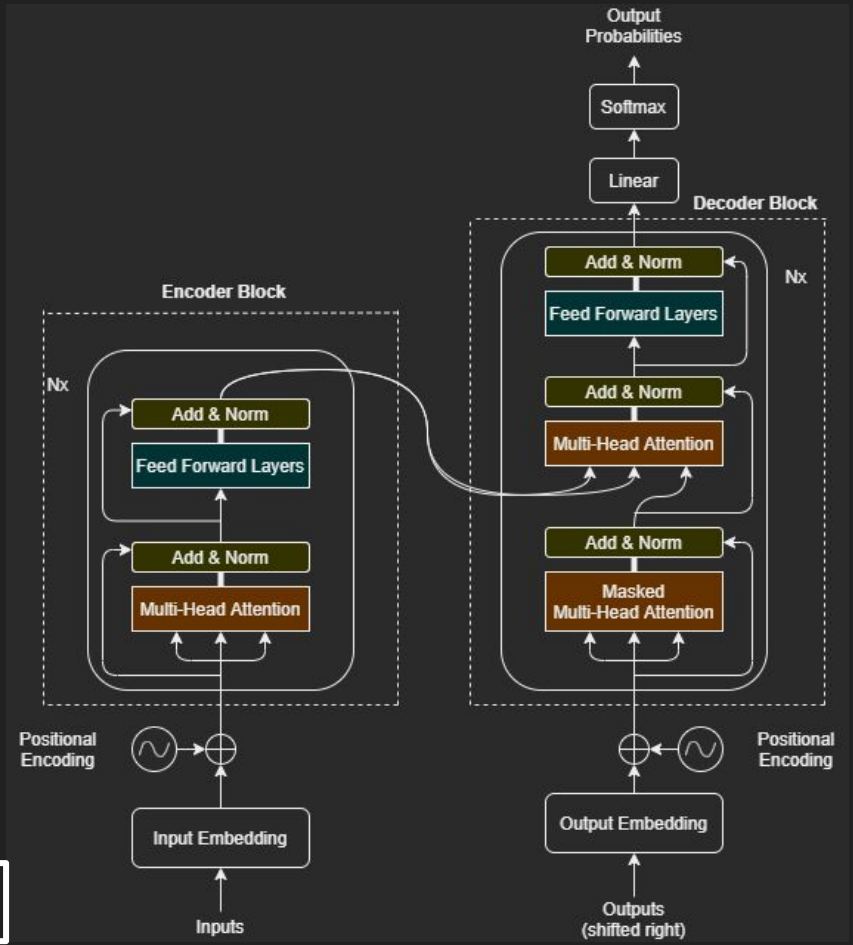


Transformer Architecture

Encoder Representation

Encoder Block

Input Sequence

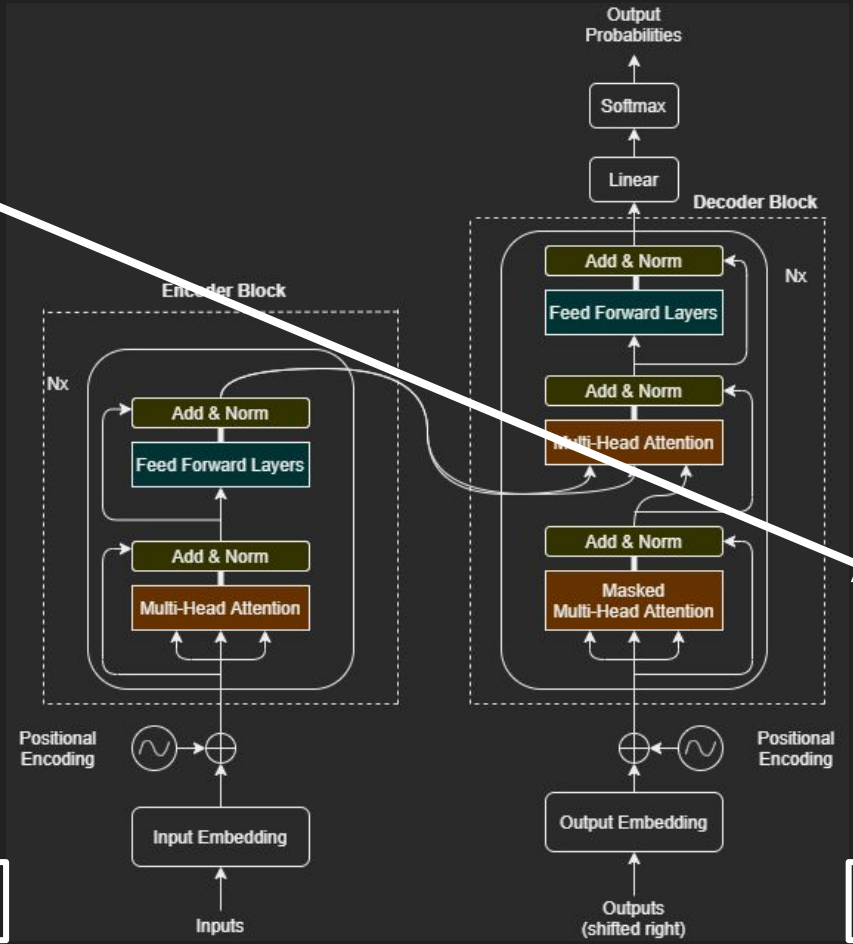


Transformer Architecture

Encoder Representation

Encoder Block

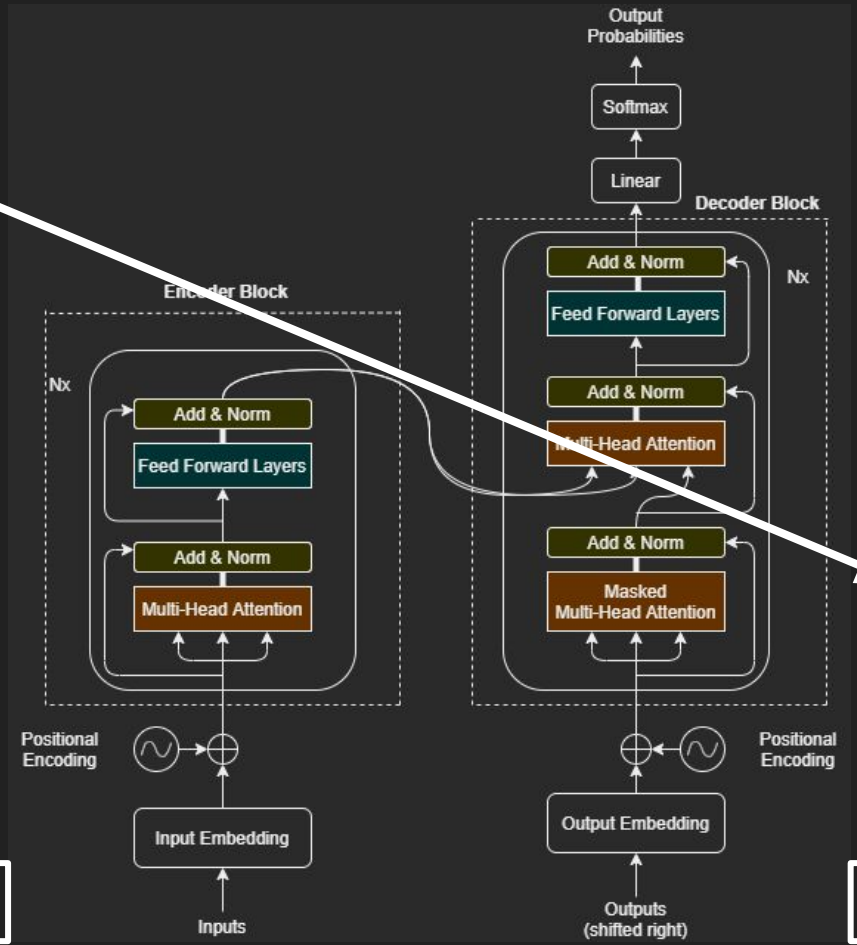
Input Sequence



Decoder Block

Output Sequence

Transformer Architecture



Encoder Representation

Encoder Block

Input Sequence

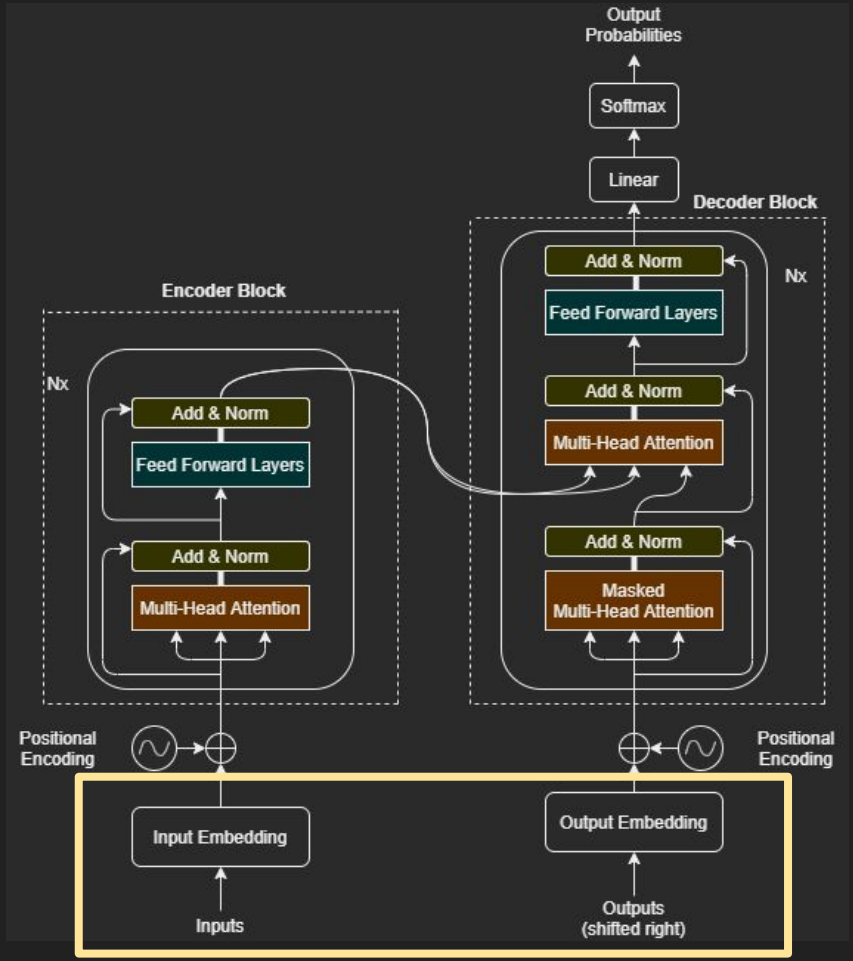
Output Probabilities

Decoder Block

Output Sequence

TABLE OF CONTENTS

Word Embedding



Word2Vec Embedding

Machines do not understand the text but they can understand numbers. For this, we go through tokenization and numericalization processes. After these preprocessing steps, we get the representation of the text using Word Embedding techniques.

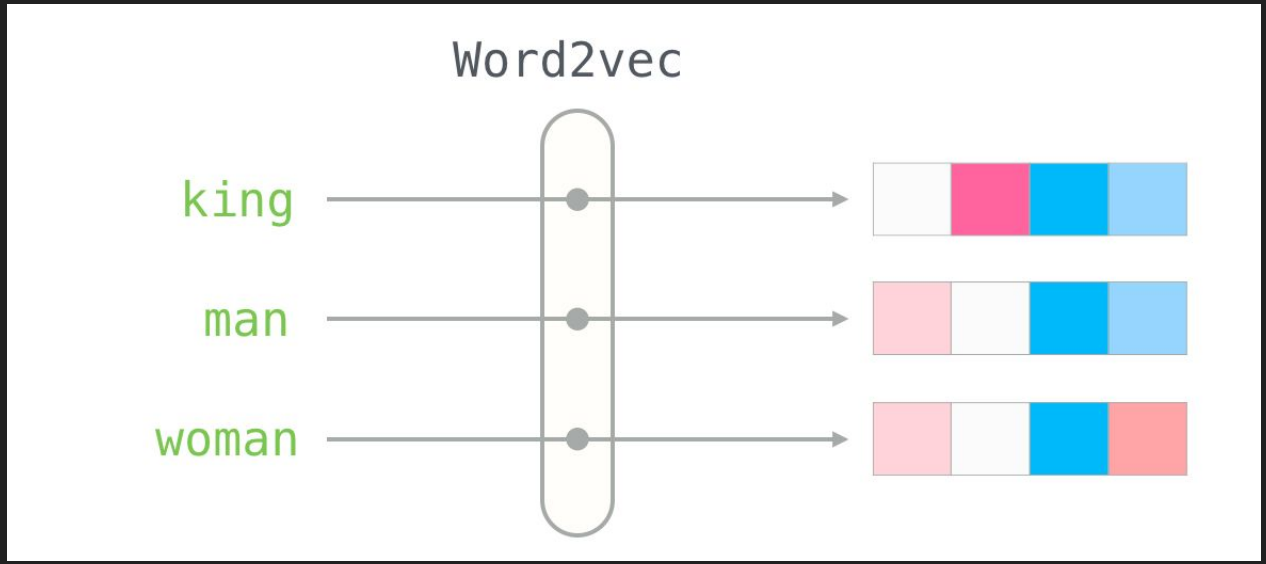
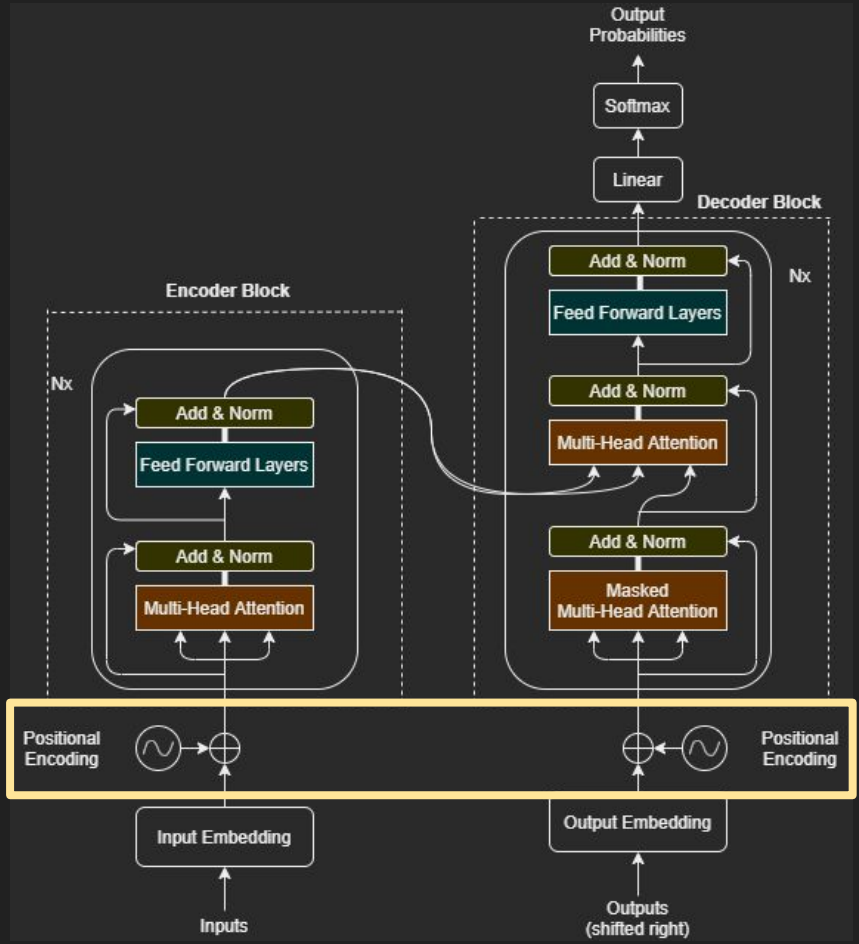


Image taken from Jay Alammar's blog

TABLE OF CONTENTS

Positional Encoding



Positional Encoding (cont.)

Transformer architecture gets rid of recurrence. So, we need some way to understand order and position of words in a sentence.

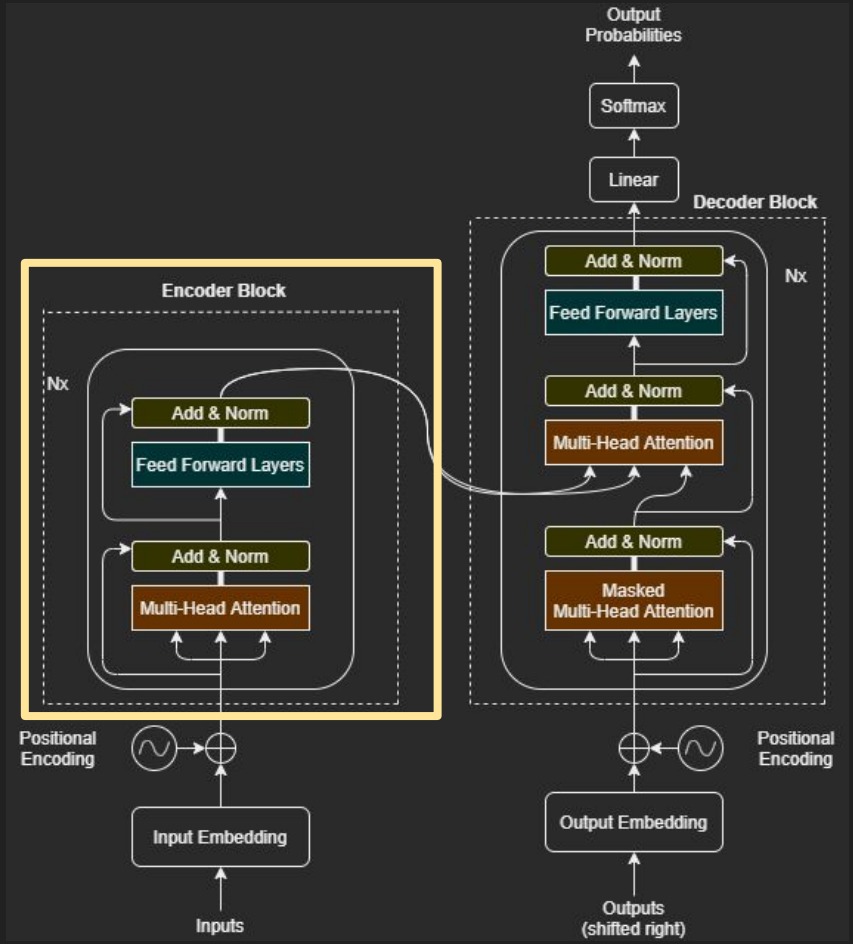
For every even position of words, we add the vector output from first equation. For every odd position of words, we add the vector output from second equation. After adding the positional encoding, we pass that to Encoder/Decoder Block.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

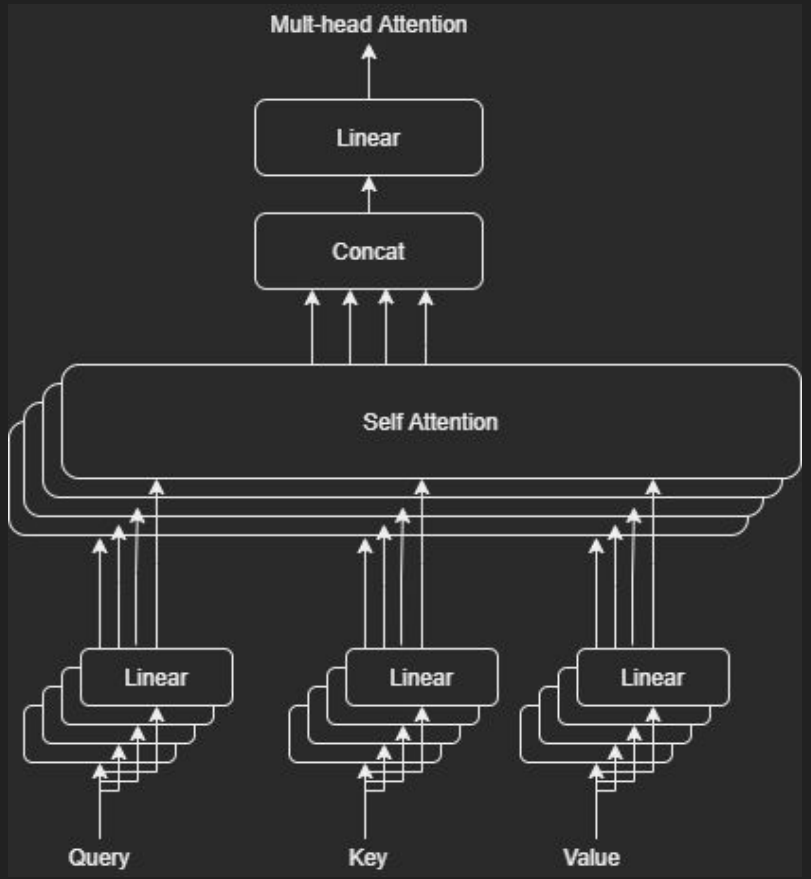
TABLE OF CONTENTS

Encoder Blocks



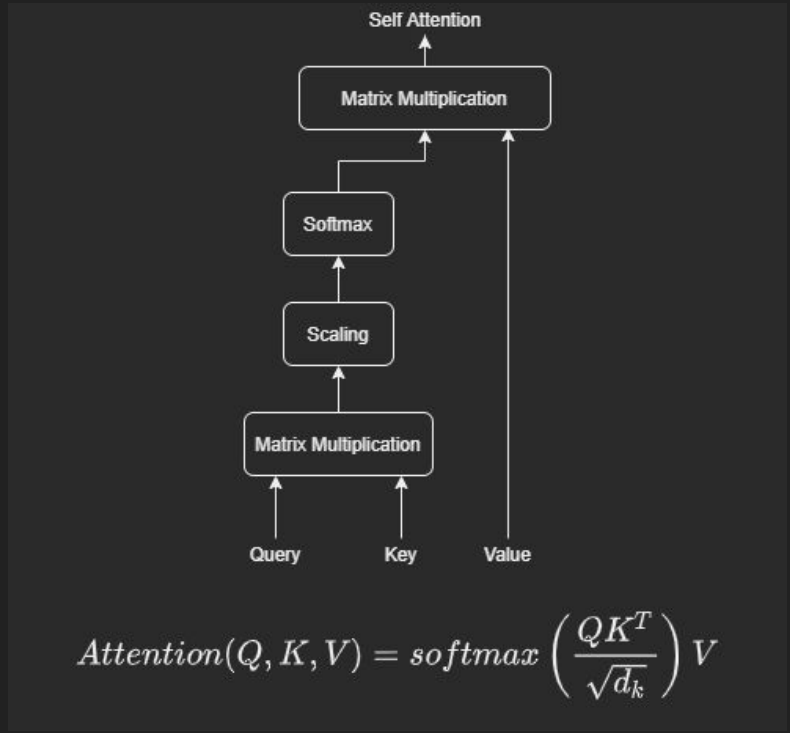
Multi-Head Attention

We divide the input into many portions i.e. heads. All of them, pass through a Linear layer and Self Attention layer. We concat the output from all self attention heads and pass them to a Linear layer.



Self-Attention

```
Q : queries => the target text to find attention
K : keys => the source text to find attention
V : values => actual values of the text
d_k : dimension of keys <br/>
```



$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Feed Forward Layer

We feed the output from Layer Normalization to a Linear Layer with ReLU activation function.

Then again pass it to a Linear layer and a Dropout Layer. In between two Linear Layers.

We expand the size once and the again move back to original embedding size. This returns us with encoder representation which can be used for many downstream task.

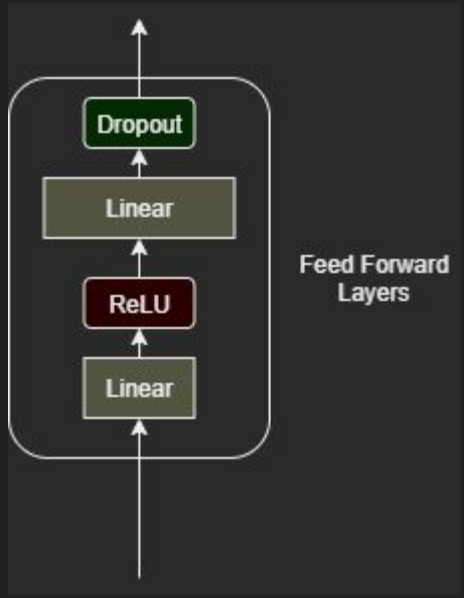
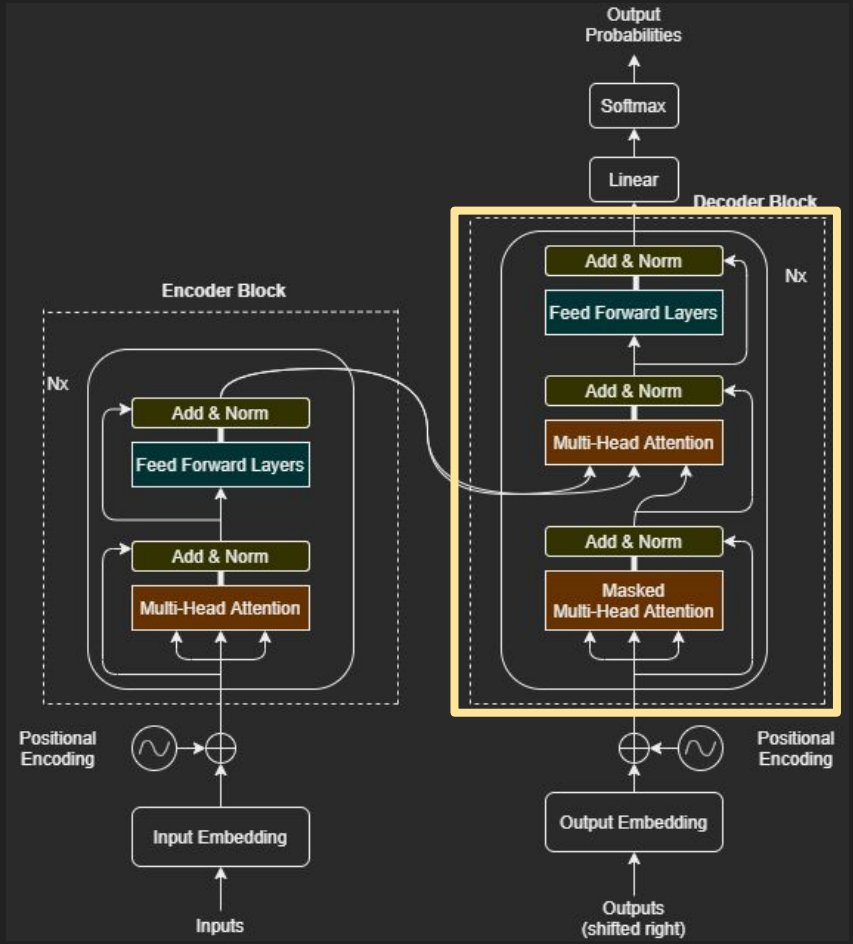


TABLE OF CONTENTS

Decoder Blocks

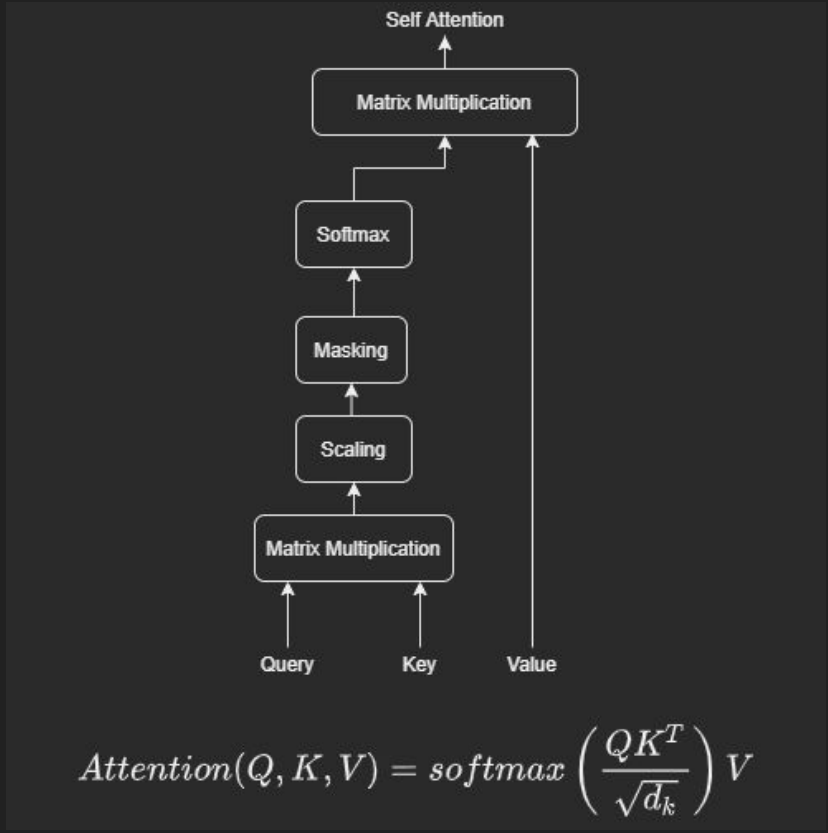


Masked Multi-Head Attention

Same as the Encoder Block but here we need to do masking.

While training the decoder, if it knows before training what comes next then the decoder cannot generalize well.

So, here we need to mask those terms. How we do that is explained in Look-Ahead Mask Section



Look-Ahead Mask

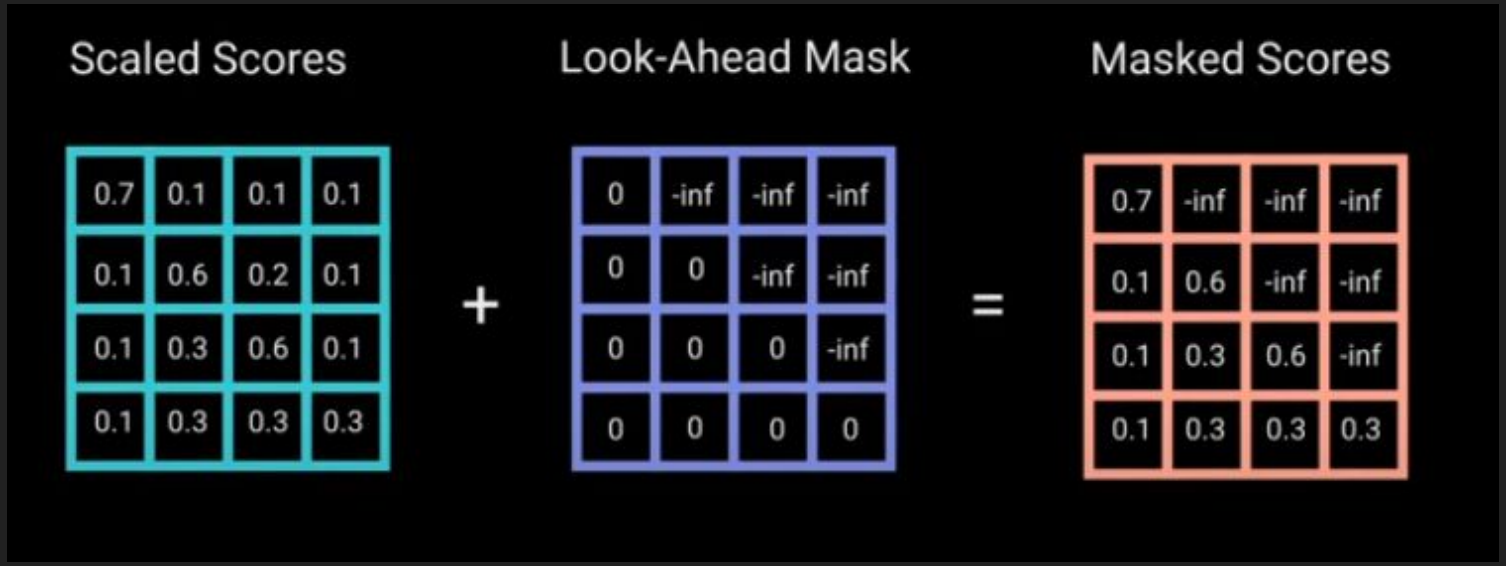
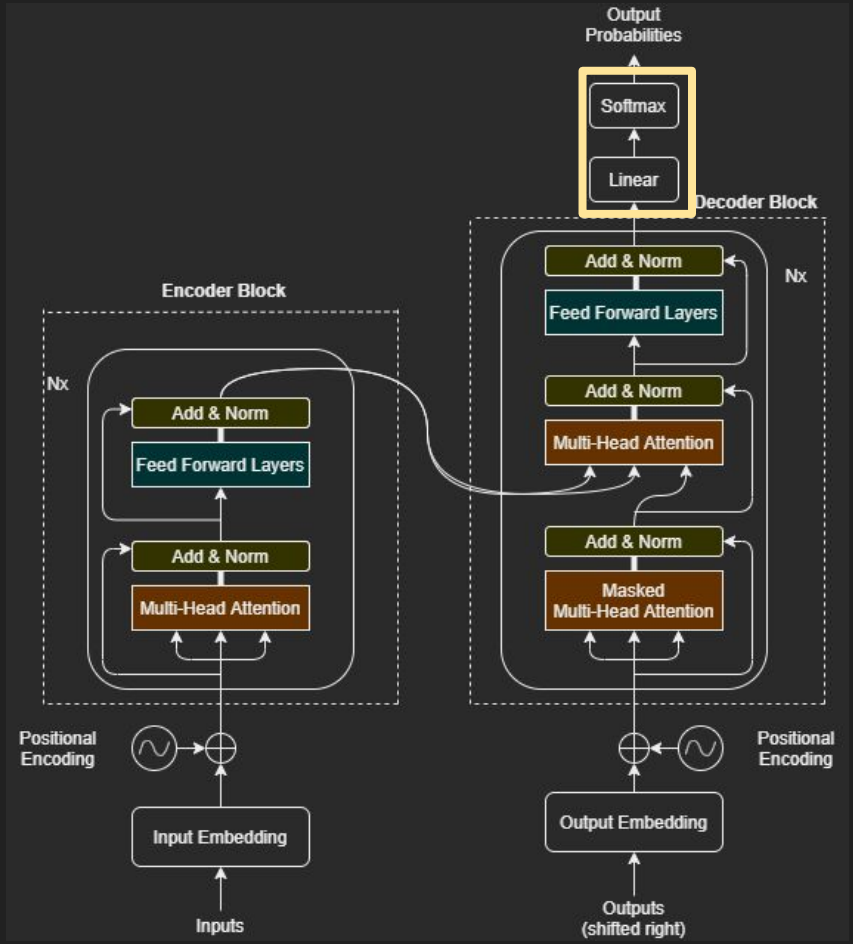


Image taken from Micheal Phi's blog

Linear Classifier

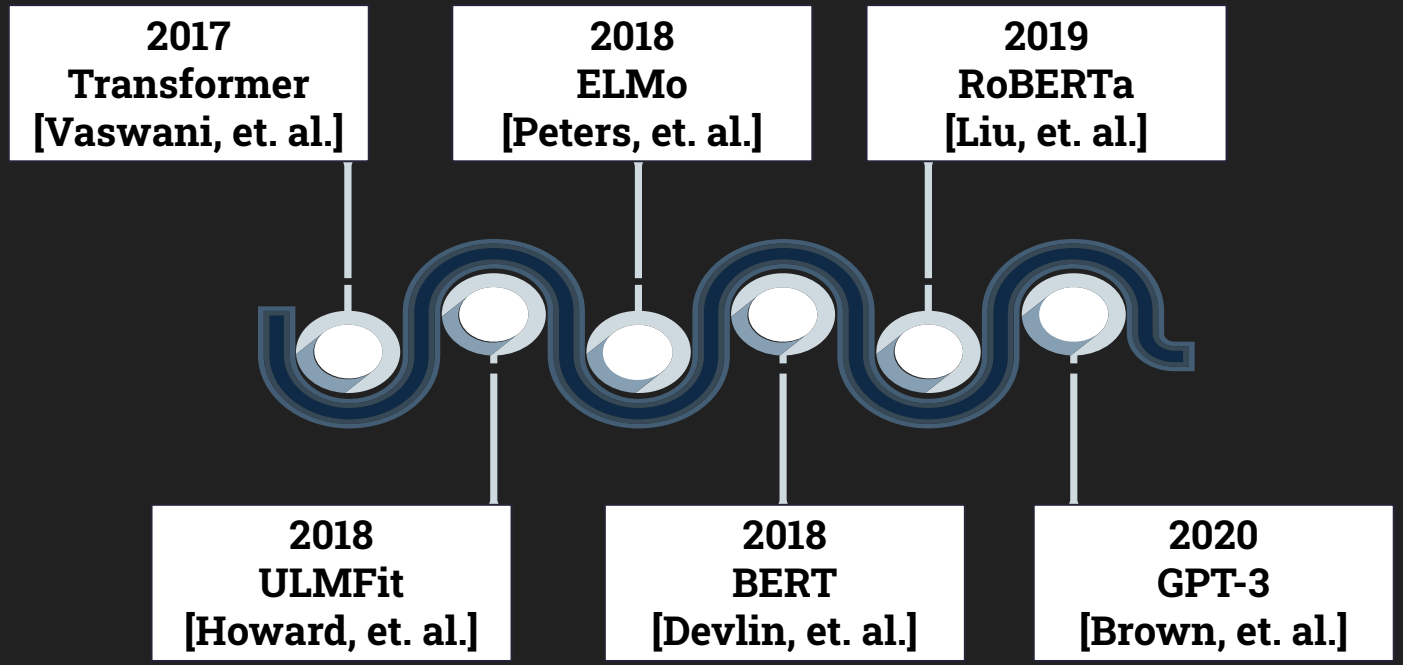


Linear Classifier (cont)

- There we have a Linear Layer and a Softmax Layer.
- The output size of the Linear Layer is equal to the vocab size where the softmax classifier maximizes the probability of the next work to come and depress others.
- After training, during inference time, we use different decoding techniques like greedy decoding or beam search.

TABLE OF CONTENTS

Natural Language Processing (NLP) Revolution [2017 – ongoing]



The Age of Transformers [ongoing]

Vision Transformer (ViT) [Dosovitskiy et. al. 2020]

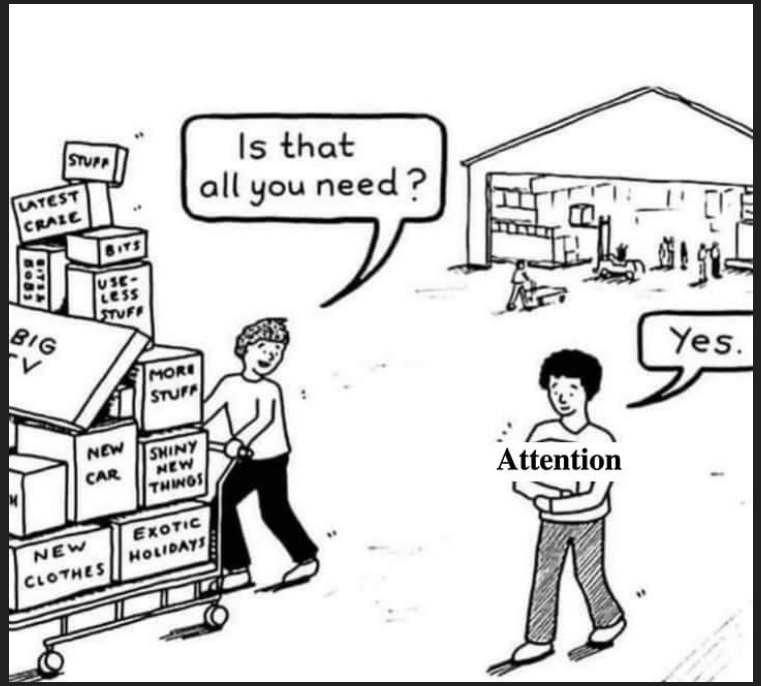
GANsformer [Hudson et. al. 2021]

Image Segmentation With Transformer [Karimi et. al. 2021]

DeTr for Object Detection [Carion et. al. 2020]

DALL-E [OpenAI 2021]

The End



For more details about transformers. Visit: https://github.com/msi1427/Original-Transformer-for-Bengali-Translation