Home Work 8 - Scaled dot Product Attention

In [ ]:
```
import torch; from torch import nn; from torch.nn import functional as F

d = 4; B = 1; T = 5
Q, K, V = torch.randn(B, T, d), torch.randn(B, T, d), torch.randn(B, T, d)

# PyTorch way
MHA = nn.MultiheadAttention(d, num_heads=1, bias=False, batch_first=True)
Wi, Wo = MHA.in_proj_weight, MHA.out_proj.weight
MHA_output, MHA_attention = MHA(Q, K, V) # shapes B, T, d and B, T, T

# Manual way
Wi_q, Wi_k, Wi_v = Wi.chunk(3)
Q, K, V = Q.squeeze(0), K.squeeze(0), V.squeeze(0) # remove batch dim

# write code here to derive `manual_attention` and `manual_output`.

# Apply projection weights
Q_proj = Q @ Wi_q.T
K_proj = K @ Wi_k.T
V_proj = V @ Wi_v.T

# Compute attention scores
scaled_dot_product = Q_proj @ K_proj.T / d**0.5
manual_attention = F.softmax(scaled_dot_product, dim=-1)

# Compute manual output
manual_output = manual_attention @ V_proj @ Wo.T

# Compare the two
print("Output:")
print(torch.allclose(MHA_attention, manual_attention)) # Aim for True
print(torch.allclose(MHA_output, manual_output)) # Aim for True
```

```
Output:
True
True
```

Advanced Data Analysis
Homework Week 8

Aswin Vijay

June 17, 2023

Positional Encoding Scheme in ”Attention is all you need” Vaswani et al.

The transformer model architecture proposed in the paper does not contain any recurrence or convolutions to encode the order of sequence of the tokens. So to encode information about the relative or absolute positions of the input tokens ”positional encodings” are proposed. They are added to the input embedding layer in the encoder and decoder stacks (Figure 1). The positional encodings

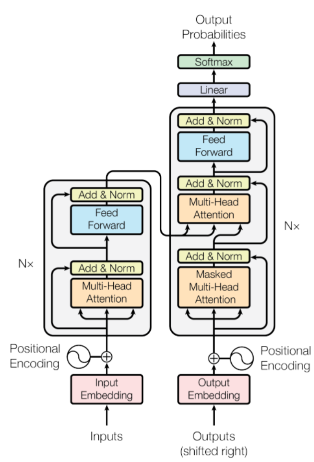

Figure 1: The Transformer - model architecture.

used are of the form,

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

Where d_{model} is embedding dimension, pos is the position and i is the dimension. The positional embedding consists of sine and cosine functions of different frequencies. The wavelenghts form a geometric progression and it was hypothesized that it would allow the model to learn to attend by relative positions as PE_{pos+k} is a linear function of PE_{pos}. It was also hypothesized that the chosen embedding would allow the model to extrapolate to longer sequence lenghts than that encountered during training.