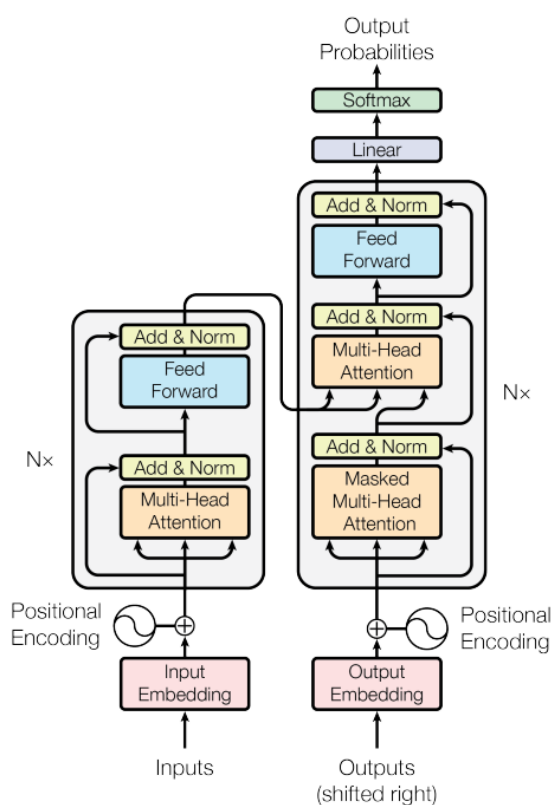


# 預訓練模型

## 背景知識

### Transformer:

由<<Attention Is All You Need>><sup>1</sup>這篇論文所提出，其使用 Encoder-decoder 結構:encoder 將輸入(input)的序列 $(x_1, \dots, x_n)$ 標註為連續的序列 $\mathbf{z} = z_1, \dots, z_n$ ，而 decoder 根據序列 $\mathbf{z}$ 生成輸出(output)序列 $(y_1, \dots, y_n)$ ，而模型的這些步驟皆擁有自回歸(auto-regressive)的特性，也就是將前個時間步 $t$ 的輸出 $y_t$ 作為下個時間步 $t + 1$ 的輸入。



The Transformer – 模型結構

<sup>1</sup> <https://arxiv.org/abs/1706.03762>

## Encoder:

在<<Attention Is All You Need>>這篇論文中，Transformer 的 Encoder 使用 $N = 6$ 層相同的編碼器(identical layers)組成，每層都有兩個子層(sub-layers)組成。第一子層是一個 Multi-Head Self-Attention，第二子層是一個簡單的 Position-wise Fully Connected Feed-forward Network，子層和子層間使用 Residual Connection 後接著 Layer Normalization，意味著每個子層的輸出為 $LayerNorm(x + Sublayer(x))$ 。為了使 Residual Connection 可行，所有層的輸出維度必須相同，因此固定 $d_{model} = 512$ 。

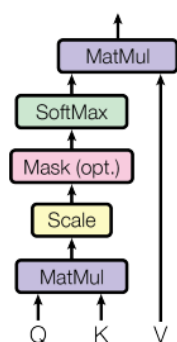
## Decoder:

在<<Attention Is All You Need>>這篇論文中，Transformer 的 Decoder 同樣使用 $N = 6$ 層相同的編碼器組成，每層除了原本的兩個子層，更加入了第三個子層 Multi-performs Attention，加入這層的原因是為了防止 Decoder 的子層提前注意到了還未解讀的內容，子層和子層間同樣使用 Residual Connection 後接著 Layer Normalization。

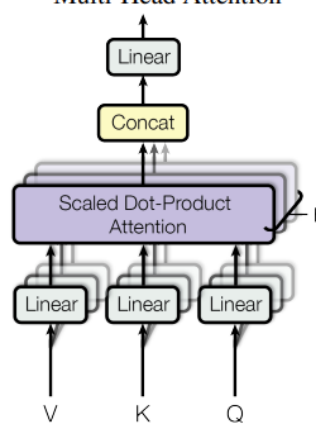
Attention:

這是一種函數，它會根據一個 query vector 和一組 key-value pairs，產生一個 output vector。Output vector 是通過 values 的加權總合得到，而每個值的加權由 compatibility function 得到。

Scaled Dot-Product Attention



Multi-Head Attention



Scaled Dot-Product Attention:

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

$d_k$  是 Key 向量的維度， $\sqrt{d_k}$  是以 query 和所有 keys 做 dot-product 得到的結果，矩陣 Q 是矩陣包裝的注意力函數，K 為 keys vector，V 為 values vector。

Multi-Head Attention:

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O$$

$$where head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

$h$  是計算注意力的次數，意味著它有  $h$  個 head，使這個模型可以關注到不同表現的資訊。

每個 head 都有自己的線性投影矩陣，將原始 $d_{model}$ 維的 query、key 和 value 映射到更小的維度 $(d_k, d_k, d_v)$ ，則以上過程可以用以下矩陣表示：

$$W_i^Q \in R^{d_{model} \times d_k}, W_i^K \in R^{d_{model} \times d_k}, W_i^V \in R^{d_{model} \times d_v}, \text{ and } W^O \in R^{hd_v \times d_{model}}$$

Position-wise Feed-Forward Networks with ReLU:

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

FNN 的輸入和輸出維度 $d_{model} = 512$ ，而內部隱藏層的維度 $d_{ff} = 2048$ 。

Embeddings and Softmax:

Embedding 是將 Input tokens 和 Output tokens 轉換成向量的技術，而 Softmax 則是將 Output vector 轉換成機率的函數。Transformer 採用了一樣的參數矩陣在 Embedding Matrix 和 Pre-Softmax Linear Transformation 中，並將 Embedding 的權重乘上 $\sqrt{d_{model}}$ 。

Positional Encoding

在<<Attention Is All You Need>>這篇論文中，因為 Transformer 模型沒有 RNN 或 CNN 結構，所以需要 PE 來讓模型理解序列的順序。於是 Transformer 模型在 Input embeddings 的最後加入了 Positional encodings，來讓模型知道位置的關係。在 Transformer 模型中使用 sine 和 cosine 函數達成這個目的：

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

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

pos 是位置，i 是維度。

## BERT

引用資料:

<https://haren.medium.com/paper-notes-bert-bert-%E6%9E%B6%E6%A7%8B%E7%90%86%E8%A7%A3-31c014d7dd63>

<https://arxiv.org/pdf/1810.04805>

全名:Bidirectional Encoder Representations from Transformers

使用 Masked LM(MLM) 和 Next Sentence Prediction(NSP) 捕捉 token-level 和 sentence-level 的特徵。也就是透過遮住某個詞語來讓模型預測被遮住的詞語。

## GPT

引用資料:

<https://reurl.cc/mxGN0A>

全名:Generative Pre-trained Transformer

GPT 僅採用 Transformer decoder 的部分，透過前面的文字序列來預測下一個單詞，或透過周圍的語境來推斷缺失單詞的意思。

## **PaLM**

引用資料:

<https://arxiv.org/pdf/2204.02311>

全名:Pathways Language Model

PaLM 採用完整的 Transformer 架構，並使用 Pathways 架構訓練，使其可以同時學習多種不同的任務。

## **LLaMA**

引用資料:

<https://ithelp.ithome.com.tw/m/articles/10366983>

全名:Large Language Model Meta AI

LLaMA 僅採用 Transformer decoder 的部分，相比於 Transformer 的 LayerNorm，它使用 RMSNorm，且使用更加平滑的 SwiGLU 函數作為激活函數。

# 訓練資料集

## Wikipedia Datasets

此資料集包含清楚的文章，並根據不同的語言將其分類，通常會使用英文的數據集進行訓練。