# Transformers And Large Langauge Models

## Introduction

In the last section we covered natural language processing with recurrent neural networks, including the limitations of neural networks. In recent years, the field of natural language processing has moved away from recurrent neural networks and towards two modern architectures, both of which as still neural network based:

- **Transformers**: a deep learning, neural network-based architecture that is designed to handle sequential data. However, unlike recurrent neural networks, transformers rely on a self-attention mechanism that processes all tokens in a sequence simultaneously, allowing each token to dynamically relate to every other token, capturing both short-range and long-range dependencies among the tokens. A transformer model consists of stacked layers of multi-head self-attention layers, feedforward neural network layers, residual connections, and normalization. Transformers are the foundation for many model machine learning applications including large language models and computer vision.
- **Large Language Models (LLMs):** advanced models built on the transformer architecture which contain billions or millions of parameters that are used to model and generate human language. A large language model is made from many stacked transformers and are trained on a massive amount of text. LLMs are able to capture relationships between words, phrases, and concepts across a wide variety of contexts.

## Transformer Architecture

I recommend that you read the paper *Attention is All You Need*in the process of completing the relevant assignments as it is the basis for transformers and therefore the basis for all large language models.

Transformers use self-attention to model all relationships in a sequence of tokens at once, but before that occurs we must first create a set of positional encoding vectors.

The input sequence to the model is first embedded into a continuous feature space and then positional embeddings are added to encode the information between the sequences and tokens

due to their orders. If X is the sequence of tokens that are given as an input to the model, then $X' = X + PE$ is the initial input to the model after the positional encoding vectors (PE) are added. For a visualization on word vectors and encoding, check out this video.

### Self Attention

The basis of the success of transformers is self-attention. Self-attention is a mechanism that allows each token in a sequence of tokens to selectively focus on other tokens when training. Given a sequence of tokens which are passed as the input to the transformer, each token generates a **query vector (Q)** which describes what the token is looking for in the other tokens, a **key vector (K)** which describes how the token describes itself, and a **value vector (V)** which describes the content the current token can offer given the other tokens. The vectors can be computed from the input, X, using the following equations, where $W^Q$, $W^K$, and $W^V$ are weight matrices which are fit:

$$Q = XW^Q \quad K = XW^K \quad V = XW^v$$

Each token is then assigned an attention score by taking the dot product between the query and key vectors. The dot product is then scaled and modified by the softmax function before being multiplied by the values vector. Mathematically, the self-attention process performs the following operation, where Q is the query vector, K is the key vector, V is the values vector, and d is the dimensionality of the keys vector:

$$Attention(Q, K, V) = softmax(\frac{(}{Q}K^T/d)V$$

Conceptually, using self-attention each token gathers information from the entire token sequence and weighs the information by relevance. This means that the model can capture both short-range relationships like the relationships between adjacent words and it can capture long-range relationships like rhyming across multiple lines of a poem. Self-attention differs from recurrent neural networks because the self-attention mechanism does not process the tokens sequentially. This means that transformers can find relationships that recurrent neural networks cannot but also that transformers can run in parallel, reducing the training time even if the number of parameters has increased.

### Multi-Head Attention

The performance of a transformer can increase using **multi-head attention** which splits the query, key, and value vectors into multiple subspaces (heads) and then applies the attention function separately to each head and the results are concatenated at the end. This method allows the model to capture different types of relationships between tokens in parallel.

### Feedforward Neural Network

After the self-attention process the output flows through a feedfward neural network which is applied independently to each token. The network has the following form:

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

Giving it one hidden layer with a RELU activation function. $W_1$ and $b_1$ are the weights and biases of the hidden layer and $W_2$ and $b_2$ are the weights and biases of the output layer. Other sublayers, like residual connection or normalization, can be added before the self-attention, after the self-attention, or after the feedforward neural network to improve the stability of the network and prevent the vanishing gradient.

A single transformer is made of a self-attention layer and a feedforward neural network layer. Stacking multiple layers of this type on top of each other forms a deep transformer network.