**Demystifying Self Attention in the Transformer Neural Network Architecture**

# Introduction

This is the third blog in my ongoing series related to the Transformer Neural Network Architecture – the purpose of the whole series is to demystify the different units/components of the Transformer Neural Network Architecture in order to develop a finer and intuitive understanding of the overall working of Transformer. The first 2 articles corresponding to the series provided a deeper understanding of the Foundational principles of Deep Learning [<https://www.linkedin.com/pulse/foundational-principles-deep-learning-my-notes-ajay-taneja/?trackingId=ZYnWMcCuRG2A24%2B9SSRYSA%3D%3D>] and the evolution of language models [<https://www.linkedin.com/pulse/evolution-language-models-my-notes-ajay-taneja/?trackingId=ZYnWMcCuRG2A24%2B9SSRYSA%3D%3D>].

The article on the Foundational Principles of Deep Learning gradually built the understanding of the Neural Networks starting from the concept of the perceptron to a single layered neural network to multi layered neural network. The discussions through the article covered the concept of activation functions, regularization through dropouts in case of Neural Networks. The article on the evolution of language models brought about the hierarchical evolution of language models starting from n-gram models and their limitations and then moving into the Deep Learning era with Recurrent Neural Networks, Long Short-Term Memory Units (LSTMs) and the problems in these networks and then progressing into the Generative AI era – talking of Transformers with a detailed emphasis on the Attention Mechanism, talking at a higher level the architecture of the Transformer Architecture.

The Transformer Neural Network architecture has the Attention Mechanism as its key highlight – the purpose of this article is to cover – or uncover – more details of Self Attention, Multi Headed as well as code the mathematics in a Colab Notebook! The article is organized as follows:

* Firstly, we revisit the motivation behind Attention – laying emphasis on the concept of “Attention Scores” which very intuitively illustrate the related words in a sentence thus capturing the contextual / semantic meaning of the attention.
* In the next section of the article, we discuss an overview of the Transformer architecture at a very high level.
* And finally, the most important highlight of this article – we go into the Colab notebook wherein we code the mathematics involved in Self Attention and Multi-headed attention.

# Motivation behind Attention

## Recurrent Neural Networks and their successes with sequential modelling

Recurrent Neural Networks – as discussed in my article on the [evolution of language models](https://www.linkedin.com/pulse/evolution-language-models-my-notes-ajay-taneja/?trackingId=ZYnWMcCuRG2A24%2B9SSRYSA%3D%3D) were the state-of-the-art for problems involving sequent to sequence modelling. They have been used in many applications involving sequence modelling. Some examples of the problems involving sequence data where RNNs have been used include the following:

1. Text can be split into sequence of characters or words and each of the individual character or word can be thought of as a time step or sequence.
2. Audio like wave forms from a speech can be split into sequence of sound waves. RNNs have been used for problems involving speech recognition.
3. Problems involving prediction of abnormalities in electrocardiogram – reading of electrocardiogram is a problem involving sequential modelling.

A picture containing text, screenshot, font, diagram

Description automatically generated

**Figure: Recurrent Neural Networks for text prediction**

A blue sound wave on a white background

Description automatically generated with medium confidence

**Figure: Recurrent Neural Networks for speech recognition**

A close-up of a graph

Description automatically generated with medium confidence

**Figure: Recurrent Neural Networks for detecting abnormalities**

**in electrocardiograms**

## Problems with Recurrent Neural Networks

In-spite of the above applications wherein the RNNs were quite successfully applied, they suffered from limitations as discussed [earlier](https://www.linkedin.com/pulse/evolution-language-models-my-notes-ajay-taneja/?trackingId=7T4VcM%2FeT5qRVXYdSoSHNA%3D%3D):

* Firstly, RNNs were slow to train – the input data needed to be passed sequentially or serially one after the other. Inputs were required from the previous state to make any operation on the current state. Thus, it was not possible to make effective use of GPUs as the operations didn’t allow parallel processing.
* RNNs also couldn’t deal with long sequences very well and not were not able to context well in long sentences. Problems such as exploding gradient discussed earlier were resolved gradient clipping – however, one must remember that gradients carry information and gradient clipping will result in loss of information – hence loss of context.

A close-up of a label

Description automatically generated with low confidence

**Figure: The Vanishing and Exploding Gradients Problems in RNNs**

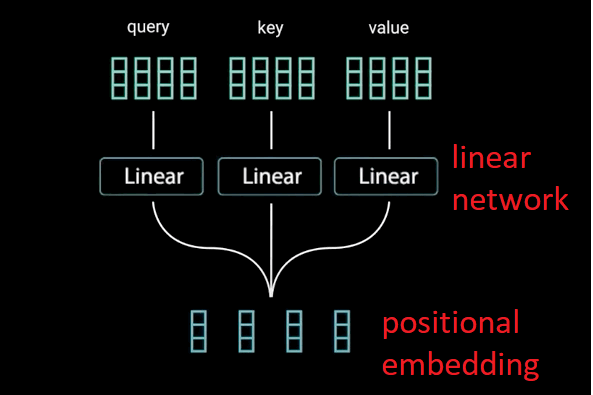
A picture containing design

Description automatically generated

**Figure: RNNs – loss of context for long sentences**

## The Attention Mechanism

The primary requirement in case of language modelling is to make the word vectors of very word carry contextual information – thus being of ‘high’ quality. And this was possible through the Attention mechanism. As explained in my last article [], the word embeddings refined through positional encoding were passed through 3 separate linear layers resulting in the Query, Key and Value vectors.

****

**Figure: Query, Key, Value vectors**

**from positional embedding**

The Attention matrix as illustrated below resulted from a dot product of the Query and the Key matrices. As it may be noticed, the Attention matrix clearly illustrated which word should each word in a sentence should focus on to incorporate in its own feature vector.

For example, given a sentence:

“I Love Deep Learning”.

Its Attention matrix should qualitatively look like as below:

As it may be noticed – observing the above matrix –

Firstly, every word has to focus on itself, therefore, the attention score of each word is the highest for the same word. Thus, the attention score of the word “I” is the highest for “I”, the attention score of “Love” is the highest for the same word “Love” and so on.

Obviously, the word Deep and Learning are closely related, and this is reflected in the Attention Matrix by a higher attention score for Deep in the fourth row. Therefore, the vector that corresponds to the word “Learning” will incorporate more context with respect to the word “Deep”.

The above is Self-Attention because we are attending on the same sentence that we are using as input. It should be underscored that Attention has many other forms and can be used in other applications such as Computer Vision.

Thus, through the Attention mechanism the original word embeddings were transformed into contextually aware ‘high quality’ word embeddings.

# An Overview of the Transformer Neural Network Architecture

As highlighted in my earlier article on the Evolution of Language Models - the Transformer Neural Network Architecture was first introduced in 2017 in the paper Attention Is All You Need – the paper dealt with the specific task of Language Translation.

[Figure]

The Transformer architecture consists of 2 parts: Encoder and the Decoder. During training, we pass the English words (Assume that we’re doing a language translation from English to French) simultaneously into the encoder portion of the Transformer neural network architecture. The output of the encoder portion will be a set of word embeddings which are “contextually” aware because of the Attention mechanism in the encoder portion of the architecture. Thus, suppose, we’re translating from English-to-French the sentence “We love this movie”, we will get a set of word embeddings which are contextually aware as output from the encoder portion of the Transformer architecture as shown in the figure below.

[Figure]

Next to the Decoder portion, we pass the English – contextually aware embeddings – along with a start token denoting the start of the sentence and start generating the French words one after the other. At the nth time step – to obtain the nth French word – we pass the English words that were obtained after processing through the encoder unit of the architecture plus the French words hat were generated upto the (n-1)th time step.

## Coding the Mathematics in Self Attention [Colab Notbook]