<https://luv-bansal.medium.com/transformer-attention-is-all-you-need-easily-explained-with-illustrations-d38fdb06d7db>

**The transformer** is explained in the paper **Attention is All You Need** by **Google Brain**in **2017**. This paper came with evolution in the field of Natural Language Processing. Many State Of The Art models in NLP is built on top of transformers.

Transformers is one of the topics which people found most complicated and not able to understand.

In this blog, I will give the general intuition on the transformer and I hope by the end of this blog, you will be able to get intuition on how the transformer works.

Many large language models (LLMs) are constructed using these transformers. If you’re curious about open-source LLMs, you can visit my blog where I’ve listed them all.

## [List of All Open Sourced Large Language Models(LLMs)](https://luv-bansal.medium.com/list-of-all-open-sourced-large-language-models-llms-a9e3927e8da8?source=post_page-----d38fdb06d7db--------------------------------" \t "_blank)

### **[A Comprehensive Guide to all Open Source Foundational Large Language Models (LLMs) and Their Comparisons](https://luv-bansal.medium.com/list-of-all-open-sourced-large-language-models-llms-a9e3927e8da8?source=post_page-----d38fdb06d7db--------------------------------" \t "_blank)**

[luv-bansal.medium.com](https://luv-bansal.medium.com/list-of-all-open-sourced-large-language-models-llms-a9e3927e8da8?source=post_page-----d38fdb06d7db--------------------------------" \t "_blank)

# Attention is All You Need

A Transformer is a type of machine learning model, it’s an architecture of neural networks and a variant of transformer models architecture are introduced like BERT, GPT-2, GPT3, etc for several tasks that are built on the top of the transformer model.

In the original paper **Attention is All You Need,**thetransformer is introduced and explained with a **machine translation**with an **encoder-decoder transformer.**

## A High-level look

Let’s first take a high-level look at our machine translation model which takes input as a french sentence and converts it into English. And then step by step we will go deeper.

A white background with small dots

AI-generated content may be incorrect.

GIF by author

The **transformer** consists of an **encoder-decoder transformer** **block** where the **encoder block** takes input and an output translation sentence is generated by the **decoder block.**

A comparison of different types of objects

AI-generated content may be incorrect.

Image by author

In the paper **Attention is All You Need**, researchers used 6 encoders in the **encoding block** and the same number of decoders were used in the **decoder block**, where all the encoders and decoders blocks are identical.

A screenshot of a computer code

AI-generated content may be incorrect.

Image by author

Our input is a text sentence but the computer just understands the numbers. So first, we tokenize our input sentence and then convert it into a sequence of tokens. Each token of the sequence is then embedded into a vector of **size 512(**according to the original paper**)**and pre-trained **Word2Vec**embedding is used for vocabulary.

The sequence of embedded vectors will be taken as input to the first encoder.

A screen shot of a computer

AI-generated content may be incorrect.

Image by author

## Position Encoding

Irrespective of RNNs where word tokens are input one at a time into the model, whereas in the attention model, all the words are input at the same time i.e. all the words taken as input parallelly into the encoder model

The position and order of words are essential parts of any language. They define the grammar and thus the actual semantics of a sentence. So, we need to do something to maintain the order of sequence therefore, to maintain the order of words in the sequence we add **position encoding** in the embedding matrix.

A diagram of a computer

AI-generated content may be incorrect.

Image by author

For each word in the sequence, a vector of 512 values — each with a value between 1 and -1 is added to the word embedding vector to maintain the order of sequence

# The Encoder Block

Till now we have discussed the basic architecture of the Attention Machine Translation model. Now we will go deeper and look at each encoder and decoder block and we will see what happens inside these blocks which makes this model so accurate.

A diagram of a self-attention

AI-generated content may be incorrect.

Image by author

A simple encoder consist of 2 parts :

* **Self-Attention**: This self-attention is completely driven by the paper **Attention is all you Need** and this plays a very very important role in a transformer.
* **Feed-Forward Neural Network**: It is just like a simple Artificial Neural Network. The output of self-attention is passed to Feed -Forward network as input.

Now we will understand the most important component of the transformer in detail.

# Self-Attention

Self-Attention is used to relate each word of the sentence with every other word of the sentence so that each word can relate with every other word, and an output of **512 dimensions** will be produced for each word in a sentence which will relate every word in a sentence.

To compute self-attention, first, we need to create 3 vectors named **query, key, and values** for each word in the input sentence from its embedding vector. These vectors are generally smaller in size. In the **transformer model,** these 3 vectors are**64-dimensional vectors**.

## Query, Key, and Values

**Query, key, and values vector for each word calculated as follow:**

A table with white squares and white text

AI-generated content may be incorrect.

Image by author, 3 embedding matrix are the same

* we initialize 3 weight matrix(randomly) let’s named them Wq, Wk, and Wv of size (512,64 ) whose values will be updated while training.
* Query, key, and value vectors are calculated by matrix multiplication between the respective weight matrix and embedding vector.

A screenshot of a black background with white squares

AI-generated content may be incorrect.

Image by author

Note that these new vectors are smaller in dimension than the embedding vector. Their dimensionality is 64, while the embedding and encoder input/output vectors have a dimensionality of 512.

You might have questions in mind related to the query, key, and value and what these vectors are, and why we have initialized them. You will be getting all your answers by the end of this blog.

We got the query, key, and values for each word in the sequence, now we will calculate the score of how much each word is related to every other word in a sentence by using the query, key, and values.

We will understand this further by doing all operations in one word and the same operations will be done on all the words of the sentence.

A diagram of a chart

AI-generated content may be incorrect.

Image by author

**The first step** in calculating **self-attention** is to calculate the score, for any word in a sentence. We calculate a **score**(how much a word is related to every word in sequence) by the dot product between **query**(q1) of a word to the **key**(k) of every word in the sentence.

The **second step** is to divide the scores by 8 (the square root of the dimension of the key vectors). This leads to having more stable gradients.

In the **third step**, **softmax**for the**scores**is calculated and according to the definition of softmax, all the values add up to 1.

A table with numbers and symbols

AI-generated content may be incorrect.

Image by author

This softmax score determines how much each word will be expressed at this position.

The **fourth and fifth step** is to **multiply** **softmax**valuesby **values vector(v)** of each word and **add** them to produce a single vector of **64- dimension.**

A screenshot of a computer

AI-generated content may be incorrect.

Image by author

*z1 = 0.86\*v1 + 012\*v2 + 0.06\*v3*

This self-attention for a single word has taken all the relevant information from all the words in the sentence.

All the step I have explained before is for self-attention of a single word, the same steps would be repeated to compute self-attention of all the words in the sentence.

To make the calculations faster and more computation efficient, all the calculations are performed in the matrix.

A white background with colorful squares and text

AI-generated content may be incorrect.

Image by author

***shape of matrix Z****= (length of sentence, dimension of V)*

## Multi-Head Attention

Above, we have discussed **single-head attention** i.e. only one weight matrix (Wq, Wk, and Wv) is randomly initialized to produce a single matrix each for query, key, and values, whereas**in the transformer**model **multi-head attention**is used i.e. instead of one weight matrix(Wq, Wk, and Wv), multiple weight matrixes are randomly initialized to produce multiple queries, key, and values matrix. And now for multiple queries, key, and value matrices, the same above operations would be repeated multiple times to produce multiple final Z matrices.

In the transformer model, 8 Multi-head attention is used.

A screenshot of a computer

AI-generated content may be incorrect.

Image by author

For each of the query, key, and value matrix, one Z matrix **(attention head),**and finally in total **8 attention-head** will be obtained.

A table of attention numbers

AI-generated content may be incorrect.

Image by author

Then all the attention-head matrices are concatenated and multiplied with other weight matrices to give the final **Z matrix**.

A screenshot of a computer

AI-generated content may be incorrect.

Image by [Jay Alammar](http://jalammar.github.io/illustrated-transformer/)

Now the output of **self-attention**i.e Z matrix feeds into **Feed-Forward Neural Network.**

A diagram of a self-attention

AI-generated content may be incorrect.

Image by author

***Feed-Forward Neural Network****output shape=****(length of sentence, 512)***

The output of the **Feed-Forward Neural Network**is then passed as input to another encoder.

All encoders and decoders are identical therefore, their working is also the same.

## The Residuals

One detail in the architecture of the encoder that we need to mention is that each sub-layer (self-attention, Feed-Forward Neural Network) in each encoder has a residual connection around it(This residual connection is the same as Resnet Residual connection), and is followed by a normalization step.

A diagram of a program

AI-generated content may be incorrect.

Image by author

Till now we have understood how the attention mechanism works in transformers. And I hope you get the intuition behind the attentions.

Many SOTA models like BERT and variant of BERT are built on **encoder transformer** and are used for many predicting various tasks.

# The Decoder

Finally, on passing the sentence into the **encoder transformer,** in the end, we will get a vector for each word(a matrix of shape **(length of sentence, 512)**) and now this matrix will be taken as input in the **encoder-decoder block** of the **decoder side**.

A diagram of a code

AI-generated content may be incorrect.

Image by author

The decoder has a self-attention at the beginning and encoder-decoder attention and feed-forward henceforth. The decoder input will be shifted to the right by one position and use the start of the word token as the first character **<EOS> token**and pass in the target sequence of words encoded in embedding along with the positional encodings.

The decoder’s self-attention block generates the attention vectors for the target sequence to find out how much each word in the target sequence is related to other words in the sequence. In the decoder, the self-attention layer is only allowed to attend to earlier positions in the output sequence. This is done by masking future positions (setting them to -inf) before the softmax step in the self-attention calculation. This ensures that while generating the attention vectors for the target sequence, we can use all the words from the input sequence, but only the previous word of the target sequence.

The decoder has an additional **multi-head attention** **block**as shown below that takes the embeddings from both the input sequence and the target sequence to determine how each word in the input sequence is related to each word in the target sequence.

A diagram of a process

AI-generated content may be incorrect.

Image from [Attention is all you need paper](https://arxiv.org/abs/1706.03762)

The second attention layer’s output is sent to an FFN layer, which is similar to the FFN layer of the encoder block with similar functionality.

Finally, in the end, we have a linear layer, which is just another FFN and a softmax function to get the probability distribution of all the next words and as such, the next predicted word with the highest probability score.

A screen shot of a computer

AI-generated content may be incorrect.

Image by author

This process is executed multiple times until the end of the sentence token is generated for the sequence.

I hope you get the intuition of how transformers actually work. If you don’t understand something then read it again and I suggest you try to visualize with the shape of matrices.