**Transformer Architecture**

1. **Introduction**

RNN based encoder-decoder models struggle when processing large sequences. The encoder model is responsible for encoding this large sequence into a vector that represents the rich representation of the input sequence. Then, the decoder processes this vector to generate the output. However, the performance of this model decreases as the input sequence length increases. For instance, in a language translational task, the model struggles to translate a long paragraph. If a human were to translate this long sentence, they wouldn’t read the entire paragraph, memorize it, and then try to translate it. Instead, they would translate one sentence at a time. An attention network works similarly to a human, processing one part at a time. Transformer model utilizes these attention mechanisms to handle large sequences effectively.

Transformer network has revolutionized the field of Natural Language Processing (NLP). Many of the most effective algorithms for NLP today are based on the transformer architecture. Unlike RNN, which process the sequences sequentially, transformer allows for parallel processing of entire sequences. This allows the transformer to handle long range dependencies effectively. The key innovation in transformer architecture is attention mechanism combined with a Convolutional Neural Network (CNN) style of processing. This attention mechanism enables the model to focus on the most relevant parts of the input for each output. This allows the model to compute rich, useful representations of words in parallel, rather than sequentially. The transformer architecture is divided into two main sections: the Encoder and the Decoder. Both these sections include key components such as multi-head self-attention mechanism and a Feed Forward Neural Network.

1. **Self-Attention Mechanism**

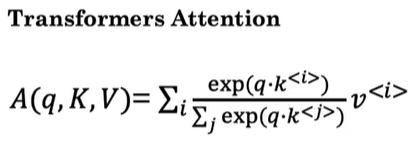
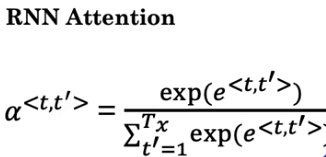
To understand the self-attention mechanism, let's consider an example from a language translation task. Take the French sentence “**Jane visite l’Afrique en septembre.**” This sentence needs to be translated into English. The main goal of the self-attention mechanism is to compute an attention-based vector representation for each word in the input sentence.



In our example, the sentence contains 5 words. Therefore, the self-attention mechanism will generate 5 vectors (A1 to A5), each corresponding to one word in the input sequence.

Let us explore how this mechanism computes A3 for the word l’Afrique.

* One way to represent the word "l’Afrique" is through word embeddings. However, this representation can vary depending on the context.
* Are we referring to "l’Afrique/Africa" as a site of historical interest, a holiday destination, or as the world's second-largest continent?
* Depending on the context, we need to represent "l’Afrique" differently.
* The A3 vector accomplishes this by examining the surrounding words to determine the context in which Africa is being discussed in the sentence and then finding the most appropriate representation for the word.



* Q represents query, K represents key, V represents Value. These vectors are the key inputs for computing the attention value for each word.
* Each word is associated with three variables: query, key, and value pairs. They were named using analogy to concept in databases where queries interact with key-value pairs. They are calculated as follows:
  + q<3> = WQ . x<3>
  + k<3> = WK . x<3>
  + v<3> = WV . x<3>

where WQ,WK, and WV are learnable parameters (weights) of this model.

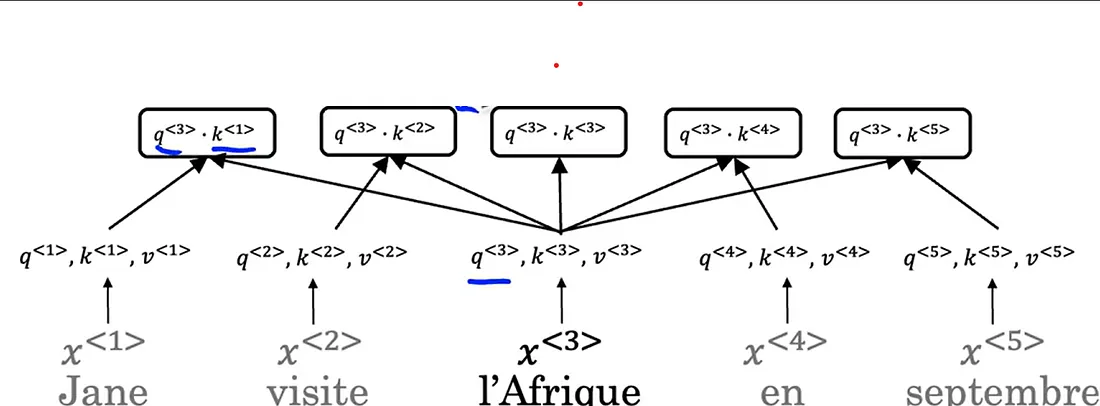
* q<3> represents a question you want to ask about **“l’Afrique”**. For example, it may represent a question like **“what’s happening there?”**
* To find the answer to this question, we compute:
* q<3> . k<1>

This computation will indicate how good "Jane" answers the question about "l’Afrique."

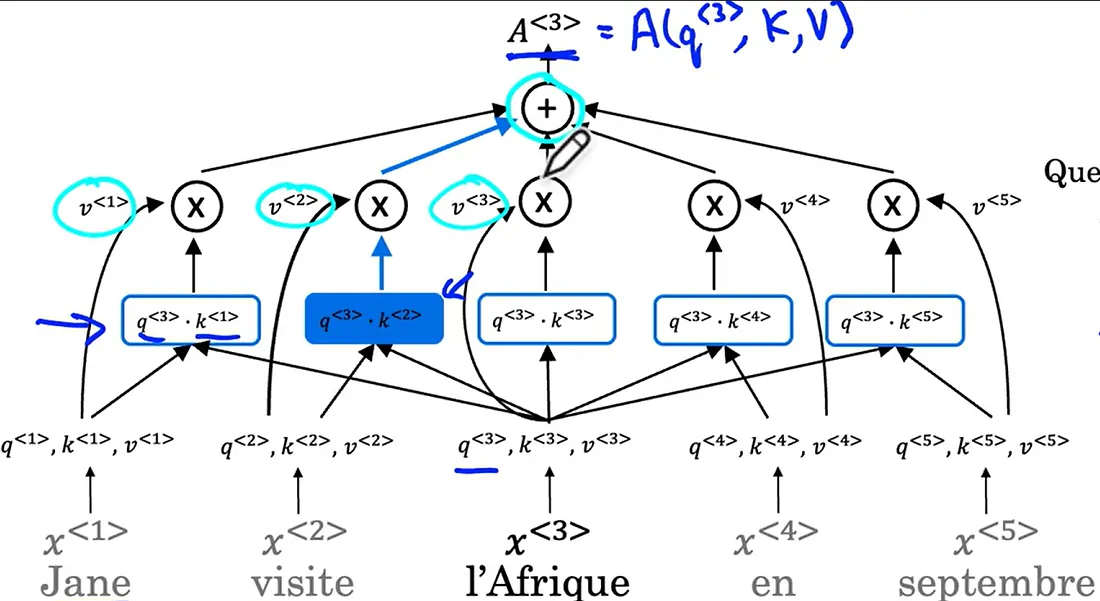
* Similarly, we compute:
  + q<3> . k<2>

This computation will indicate how good "visite" answers the question about “l’Afrique”. Similarly, we compute for all other words.

* The goal of this computation is to pull up the most information that's needed to compute the most useful representation A<3>.
* For intuition, consider that k<1> represents that the word "Jane" is a person, and k<2> represents that the word "visite" is an action.
* The inner product **q<3>*⋅* k*<2>***will have the largest value, indicating that "visite" provides the most relevant context for understanding "l’Afrique" as a destination for a visit.



* Then, we compute Softmax over these 5 inner product results.
* Multiply the Softmax values with v<1>, v<2>, etc., which are the values corresponding to the respective words in the input sentence.
* Sum all these resultant values to get final self-attention score A<3> for the word “l’Afrique”.

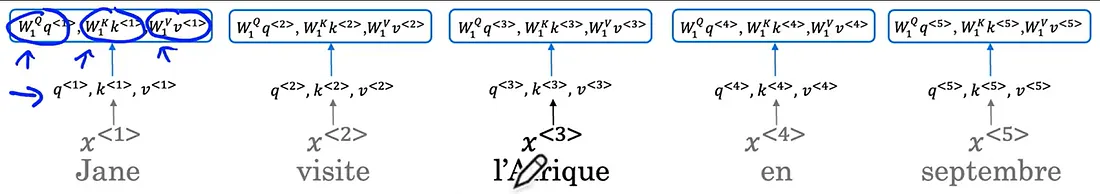


* The key advantage of this representation is that "l’Afrique" isn't a fixed word embedding. Instead, the self-attention mechanism recognizes "l’Afrique" as the destination of a visit, computing a richer, more useful representation for this word.
* The same computation process can be carried out for all the words in the sentence to get similarly rich representations for “Jane”, “visite”, “en”, and “septembre”.
* These computations can be summarized as follows:



1. **Multi-Head Attention Mechanism**

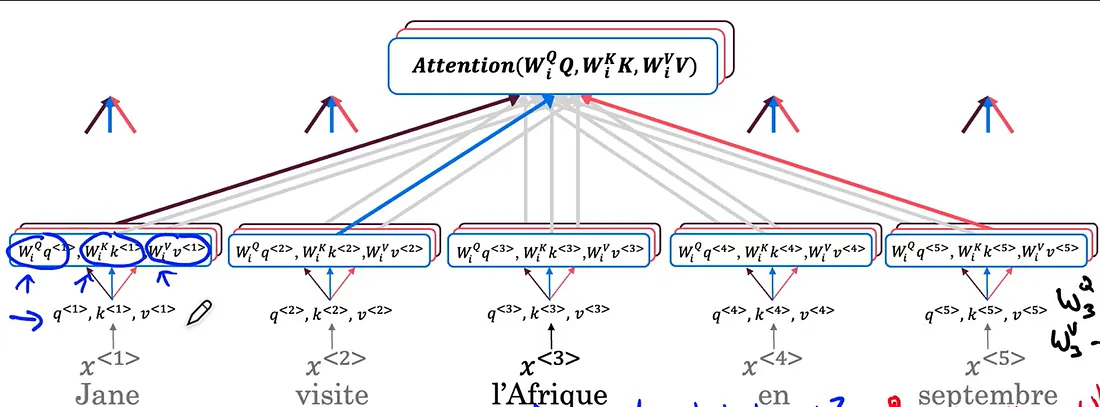
* The term "multi-head attention" means performing the self-attention process multiple times. This is like running a for loop over self-attention to get more specific contextual information in the sentence.
* Each time we calculate self-attention for a sequence is called **head.**



* If h=8, where “h” denotes the number of heads, the self-attention computation is repeated 8 times for the input sequence.
* Each head uses a different set of weight matrices, such as (W1Q, W1K, W1V) for head 1, (W2Q, W2K, W2V) for head 2, and so on, to ask and answer different questions in each head.
* For example, when considering the word “l’Afrique”, in the first head of multi-head attention, the query (q<3>) might be “What’s happening there?” In the second head, the question could be “When is something happening?” In the third head, the question could be “Who is involved?”
* Each head computes its own set of attention values, and the results are concatenated to form the final output of the multi-head attention mechanism.



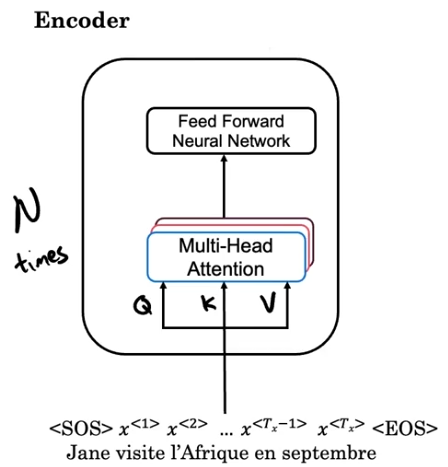
* This concatenated output is then multiplied by a matrix WO.



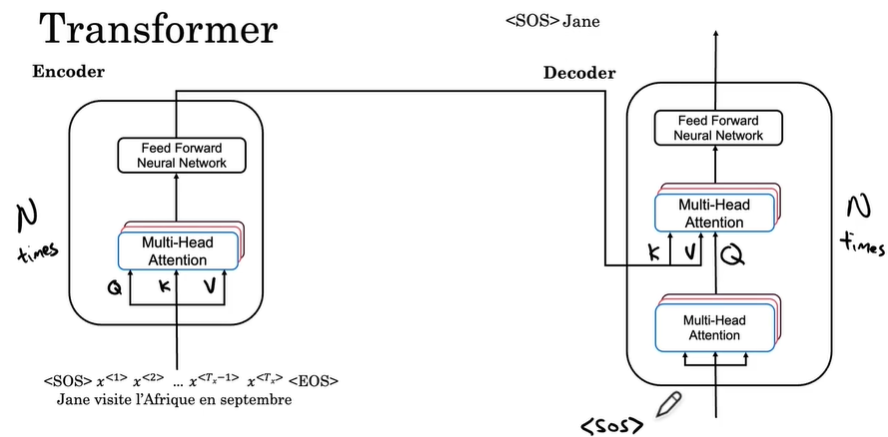
* Although conceptually you can think of this as a big for-loop, in practice, these computations can be done in parallel since the value of one head does not depend on the value of another.
* This parallel computation makes the process more efficient.

1. **Transformer**

* The transformer architecture is divided into two main sections: the Encoder and the Decoder.
* We'll use the example sentence "Jane visite l’Afrique en septembre" and its corresponding embeddings to illustrate how to translate the sentence from French to English. We can also include the start of sentence <SOS> and end of sentence <EOS> tokens.
* The words in the input sequence are represented by their corresponding embeddings.
* The first step in the transformer architecture is feed these embeddings into an encoder block, which contains a multi-head attention layer.
* This layer produces a matrix that is passed into a feed-forward neural network, helping to identify interesting features in the sentence.
* This encoding block is repeated “N” times, with a typical value for “N” being six.
* After passing through the encoder block multiple times, the output is fed into a decoder block.



* The decoder block's goal is to generate the English translation.



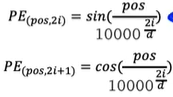
* The decoder block consists of two multi-head attention blocks.
* The first output will be the start of sentence token.
* At each step, the decoder block inputs the previously generated words of the translation.
* Initially, the only known input is the start of sentence <SOS> token.
* This token is fed into the first multi-head attention block, where it is used to compute Q, K, and V for this block.
* The output of this first block generates the Q matrix for the next multi-head attention block, while the output of the encoder generates K and V.

**Why is it structured this way?**

* The input to the decoder block <SOS> which is processed by the first attention block will generate a query “What is the start of sentence” for the second block.
* The second block then pulls context from K and V using the encoder’s output to decide the next word in the sequence to generate.
* The output of the second attention block is passed into the feed forward neural network to generate the next word.
* The model predicts the first word in the English translation as “Jane”.
* This word is then fed back into the input, and the next query is generated from SOS and “Jane”.
* The process continues, generating each subsequent word in the translation, such as “visit”, “Africa”, “in”, and “September” until the end of sentence token <EOS> is produced.
* These encoder and decoder blocks, and their combination to perform sequence-to-sequence translation tasks, are the main ideas behind the transformer architecture.
* This approach allows for simultaneous computation and efficient translation of input sentences into another language.
* Beyond these main ideas, there are additional features that enhance the transformer network's performance. One such feature is positional encoding of the input.

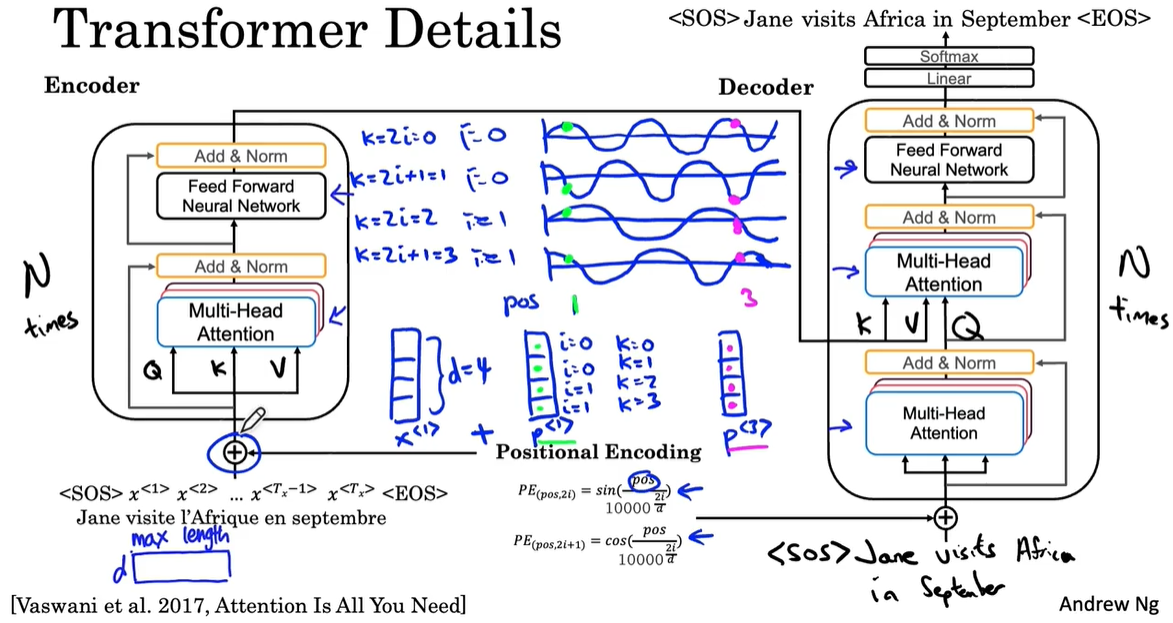
**4.1 Positional Encoding**

* The self-attention equations do not indicate the position of a word within a sentence, which is crucial information.
* Positional encoding uses sine and cosine functions to create unique positional vectors for each word, indicating their position in the sentence.
* These positional encodings are added to the word embeddings, ensuring that each word vector is influenced by its position in the sentence.
* The positional encoding is calculated as follows:



Where:

* “pos” represents the numerical position of the word in the input sequence
* “d” is the dimension of word embedding



* The value of “i” is repeated twice, because each value is used to encode two dimensions using sine and cosine.
* This positional encoding ensures that the output of the encoding block contains both contextual semantic embedding and positional encoding information.
* Additionally, residual connections, like those in ResNet, are used to pass positional information through the entire architecture.
* The transformer network also employs a layer like batch normalization, called add & norm, which helps speed up learning and is repeated throughout the architecture.
* At the output of the decoder block, a linear layer followed by a softmax layer predicts the next word one at a time.
* During training, mask multi-head attention is used to train a model. In this process, some part of the sentence is hided, and the model learns to predict the next word given a first part of the sentence.
* It pretends that the model has perfectly translated few words and hides remaining words to check whether the model can predict the next word in the sentence accurately.
* Since the publication of the “Attention Is All You Need” paper, many iterations of this model, such as BERT and DistilBERT, have been developed.
* These models build on the transformer architecture to achieve even better performance in various NLP tasks.