Transformer Neural Network Explain.

_The state_of_the_art(unordered neural network)

Summarize the paper:

The paper titled "Attention Is All You Need" introduces a new network architecture called the Transformer, which is based solely on attention mechanisms and eliminates the need for recurrent or convolutional neural networks. The authors propose this architecture as an alternative to existing sequence transduction models. The Transformer model demonstrates superior quality, parallelizability, and reduced training time compared to traditional models.

The paper begins by discussing the prominence of recurrent neural networks (RNNs) in sequence modeling and transduction tasks such as language modeling and machine translation. However, RNNs suffer from sequential computation and limited parallelization, especially with longer sequences. Attention mechanisms have been used in conjunction with RNNs to model dependencies between positions in sequences. The authors propose the Transformer as a model architecture that relies solely on attention mechanisms, allowing for increased parallelization and improved performance.

The paper presents the Transformer model's architecture and its components, including the encoder and decoder stacks. The encoder consists of multiple layers with self-attention mechanisms and position-wise fully connected networks. The decoder, in addition to the encoder layers, incorporates multi-head attention over the encoder's output. The attention function used in the Transformer is described as scaled dot-product attention, which computes the compatibility between queries and keys to obtain weighted values.

The authors highlight the advantages of the Transformer model over existing approaches, such as reduced computational complexity and increased parallelization. They conduct experiments on machine translation tasks, demonstrating that the Transformer outperforms previous models in terms of translation quality. The Transformer achieves state-of-the-art results on the WMT 2014 English-to-German and English-to-French translation tasks while requiring significantly less training time.

Furthermore, the authors show that the Transformer model generalizes well to other tasks by successfully applying it to English constituency parsing with both large and limited training data. They provide detailed descriptions of the model's architecture, training process, and evaluation metrics.

In conclusion, the paper introduces the Transformer model as a novel architecture for sequence transduction tasks. By leveraging attention mechanisms and eliminating the reliance on recurrent or convolutional networks, the Transformer achieves improved performance, parallelizability, and training efficiency. The experimental results validate the effectiveness of the Transformer on machine translation tasks, setting new state-of-the-art results.

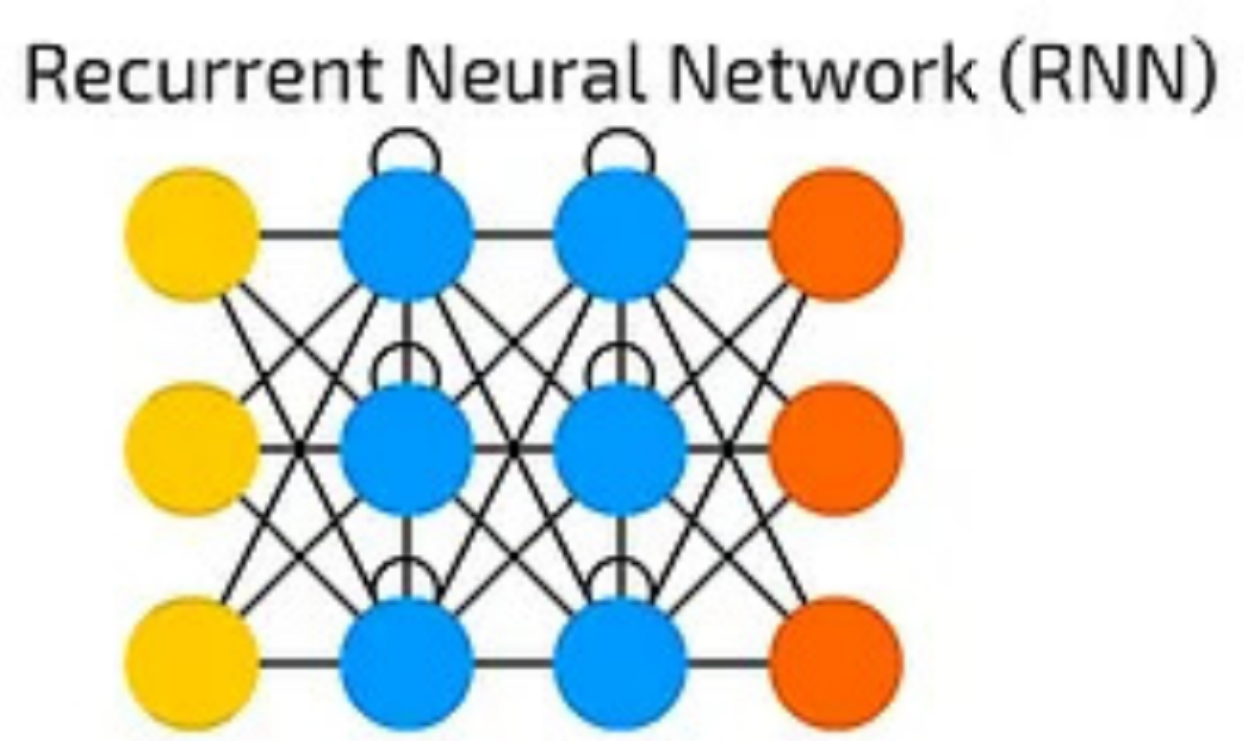
The Transformer neural network is a powerful deep learning model that was introduced in a landmark paper titled "attention is all you need..pdf" by Vaswani et al. in 2017. It revolutionized the field of natural language processing (NLP) and has since found applications in various other domains. The Transformer architecture is based on the concept of attention, enabling it to capture long-range dependencies and achieve state-of-the-art performance on a wide range of tasks.

Applications of the Transformer neural network:

1. **Machine Translation**: The Transformer has achieved impressive results in machine translation tasks, such as translating text from one language to another. Its ability to capture long-range dependencies and handle variable-length input sequences makes it well-suited for this task.
2. **Text Generation**: The Transformer can be used for generating coherent and contextually relevant text. It has been applied to tasks such as generating news articles, dialogue systems, and story generation.
3. **Summarization and Document Understanding**: The Transformer's attention mechanism enables it to focus on important parts of a document or text, making it effective for tasks like text summarization, document classification, and sentiment analysis.
4. **Speech Recognition**: The Transformer has also been applied to automatic speech recognition tasks, where it converts spoken language into written text. It has shown promising results in this domain as well.
5. **Question Answering**: The Transformer's ability to understand and generate text has made it useful in question answering systems. It can process a question and a context paragraph and generate relevant answers.
6. **Image Recognition**: While primarily designed for NLP tasks, the Transformer has also found applications in computer vision tasks. It can be adapted for image recognition tasks by treating images as sequences of patches.

Before the emergence of Transformer Neural Networks (TNNs), Recurrent Neural Networks (RNNs) were commonly employed for sequential processing tasks, including machine translation. However, RNNs were characterized by slow processing speeds, limited accuracy, and challenges in handling large datasets.

here how RNN woks:
is designed to process sequential data, where the current input not only depends on the current state but also on the previous inputs and states.



suppose we have this sentence "I work at the university.", and we want to translate it to Arabic "انا اعمل في الجامعة" .

In the translation task, the RNN analyzes each word ('I', 'work', 'at', 'the', 'university') one by one, updating the hidden state at each step. The output at each time step is influenced by the current word and the hidden state, which captures the historical information from previous words. The final output is a sequence of translated words ('انا', 'اعمل', 'في', 'الجامعة') in Arabic.

Indeed, RNNs tend to be slow and can struggle with handling large datasets, which can lead to potential confusion or difficulties in processing extensive data. However,

the Transformer Neural Network (TNN) introduced a breakthrough solution called "Self-Attention" in the paper "Attention is all you Need." This innovation addressed these issues and paved the way for subsequent advancements such as GPT, Bert, LLama, stable diffusion, and more.

In this section, I will delve into the details of the Transformer Neural Network as described in the paper:

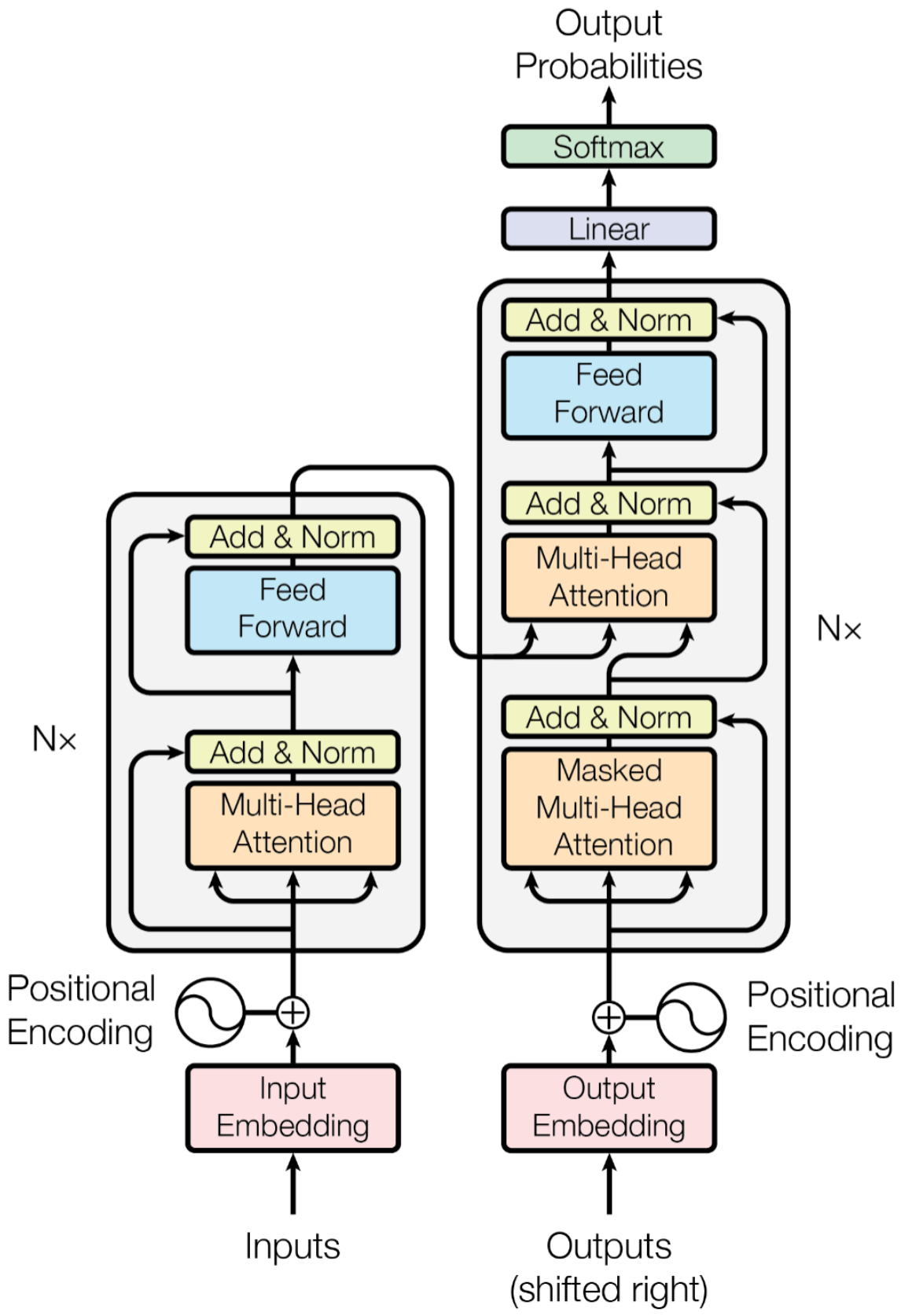
> The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data

> 1 Introduction Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [35, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [38, 24, 15]. Recurrent models typically factor computation along the symbol positions of the input and output sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states ht, as a function of the previous hidden state ht−1 and the input for position t. This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. Recent work has achieved significant improvements in computational efficiency through factorization tricks [21] and conditional computation [32], while also improving model performance in case of the latter. The fundamental constraint of sequential computation, however, remains. Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences [2, 19]. In all but a few cases [27], however, such attention mechanisms are used in conjunction with a recurrent network. In this work we propose the Transformer, a model architecture

> eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization and can reach a new state of the art in translation quality after being trained for as little as twelve hours on eight P100 GPUs.

Model Architecture:

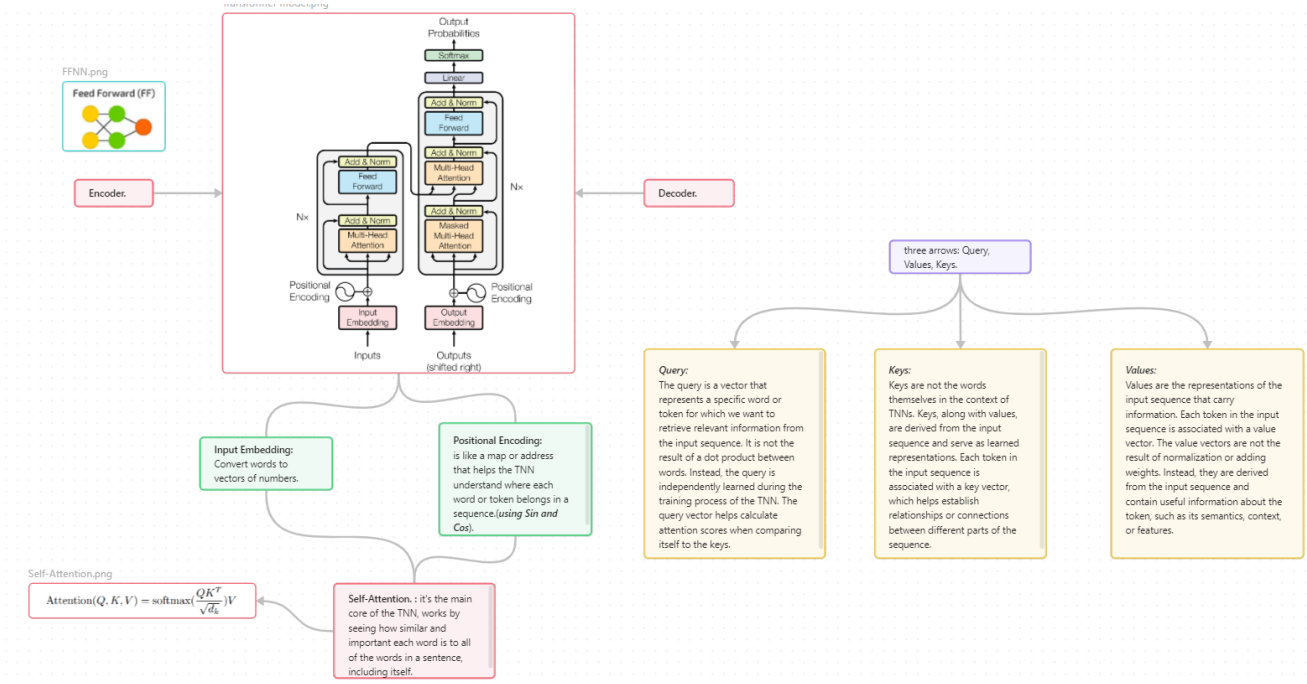
> Most competitive neural sequence transduction models have an encoder-decoder structure [5, 2, 35]. Here, the encoder maps an input sequence of symbol representations (x1, ..., xn) to a sequence of continuous representations z = (z1, ..., zn). Given z, the decoder then generates an output sequence (y1, ..., ym) of symbols one element at a time. At each step the model is auto-regressive [10], consuming the previously generated symbols as additional input when generating the next.

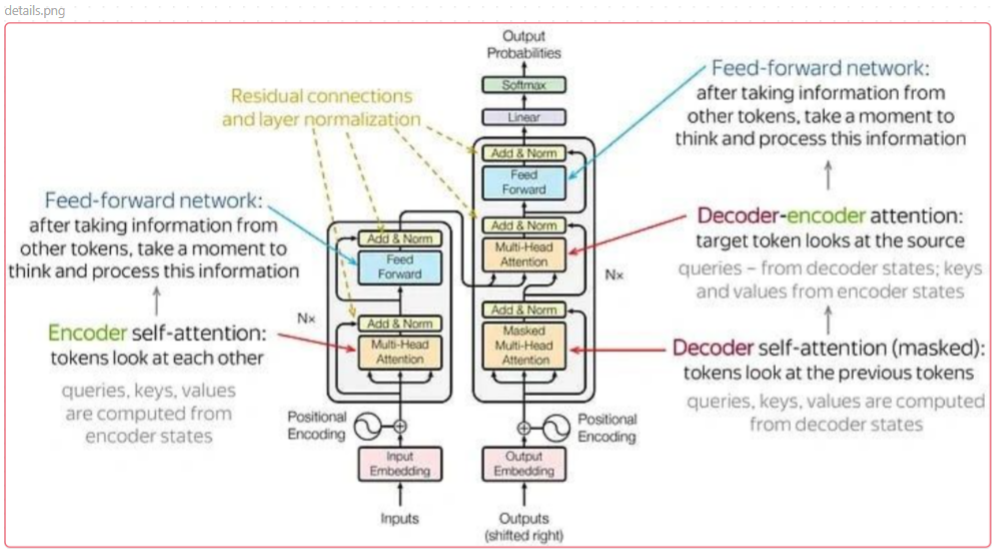


explanation:

Please check this canvas Transformer model. to observe how the peculiar words function, and then proceed to read the following section to understand how the model

itself operates.





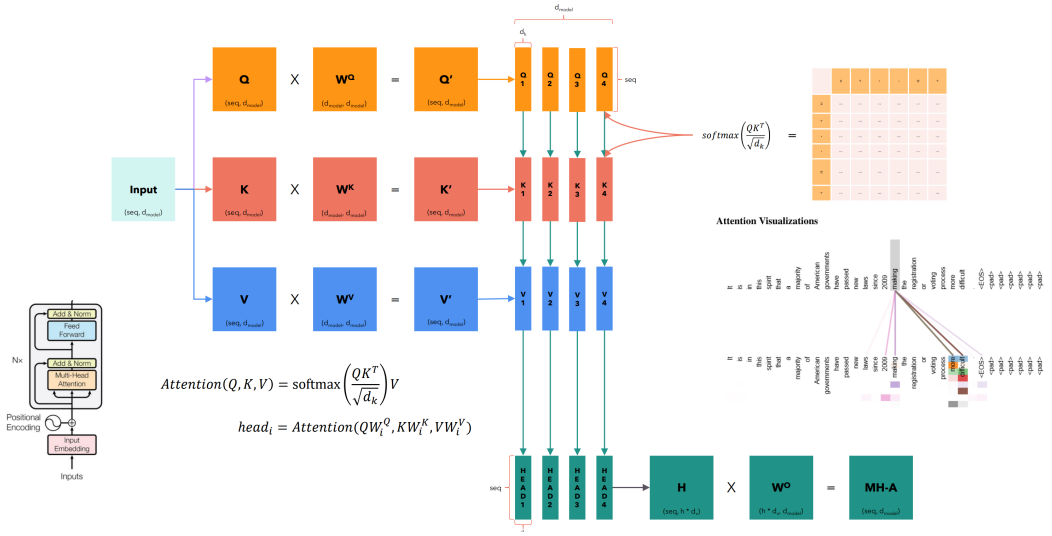
so first we have the left architecture which is the "encoder" and the right is the "decoder":

1. **Input Embeddings:**
   The input sequence is transformed into fixed-dimensional embeddings, typically composed of word embeddings and positional encodings. Word embeddings capture the semantic meaning of each word.
2. while **positional encodings** indicate the word's position in the sequence using the sin and cos waves.

$$PE(pos, 2i) = sin(pos/100002i/dmodel)$$

$$PE(pos, 2i+1) = cos(pos/100002i/dmodel)$$

3. **Encoder and Decoder:**
   The Transformer model consists of an encoder and a decoder. Both the encoder and decoder are composed of multiple layers. Each layer has two sub-layers: a multi-head self-attention mechanism and a feed-forward neural network.
   - ***Encoder:*** The encoder takes the input sequence and processes it through multiple layers of self-attention and feed-forward networks. It captures the contextual information of each word based on the entire sequence.
   - ***Decoder:*** The decoder generates the output sequence word by word, attending to the encoded input sequence's relevant parts. It also includes an additional attention mechanism called "encoder-decoder attention" that helps the model focus on the input during decoding.
4. **Self-Attention Mechanism:**
   - **firs what is self attention:** it is the core of the Transformer model is the self-attention mechanism. It allows each word in the input sequence to attend to all other words, capturing their relevance and influence, works by seeing how similar and important each word is to all of the words in a sentence, including itself.
   - **Second the Mechanism:**
     - ***Multi-head attention in the encoder block*:** plays a crucial role in capturing different types of information and learning diverse relationships between words. It allows the model to attend to different parts of the input sequence simultaneously and learn multiple representations of the same input.



     - ***Masked Multi-head attention in the decoder block*:** the same as Multi-head attention in the encoder block but this time for the translation sentence, is used to ensure that during the decoding process, each word can only attend to the words before it. This masking prevents the model from accessing future information, which is crucial for generating the output sequence step by step.

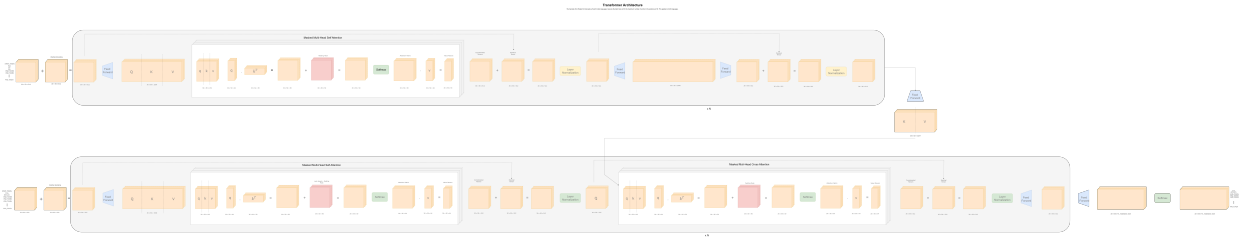
- ***Multi-head attention in the decoder block*:** do the same as the Multi-head attention in the encoder block but between the input sentence and the translation sentence, is employed to capture different relationships between the input sequence and the generated output sequence. It allows the decoder to attend to different parts of the encoder's output and learn multiple representations of the context.

5. **Feed Forward in two blocks:** it is just feed forward neural network but in this paper the neurons are 2048.
6. **Add & Normalization.**

$$Z_{\text{norm}}^{(i)} = Z^{(i)} - \frac{\mu}{\sqrt{\sigma^2 + \epsilon}}$$

Optional using Learnable parameters:

$$Z^{\tilde{(i)}} = \gamma Z_{\text{norm}}^{(i)} + \beta$$



---

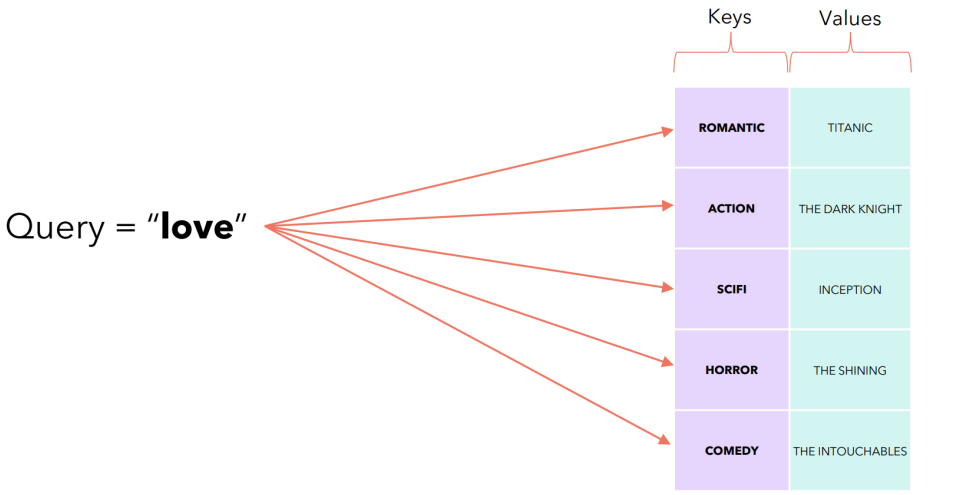
self attention mechanism:

The core of the Transformer model is the self-attention mechanism. It allows each word in the input sequence to attend to all other words, capturing their relevance and influence. Self-attention computes three vectors for each word: Query, Key, and Value.

- Query (Q): Each word serves as a query to compute the attention scores.
  - Q: what I am looking for.
- Key (K): Each word acts as a key to determine its relevance to other words.
  - K: what I can offer.
- Value (V): Each word contributes as a value to the attention-weighted sum.

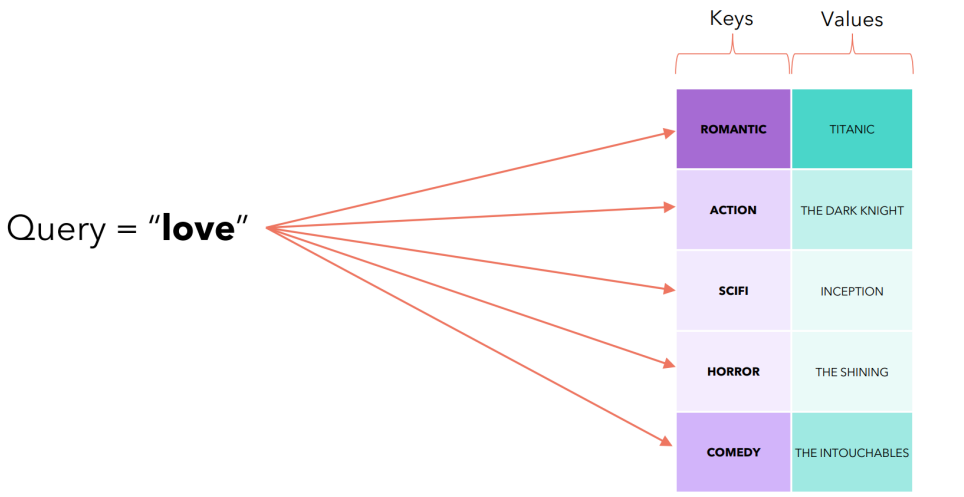
- what I actually offer.

The Internet says that these terms come from the database terminology or the Python-like dictionaries.



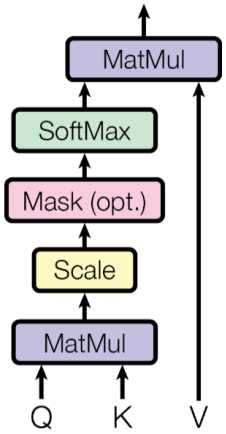
* this could be a Python dictionary or a database table.

The Internet says that these terms come from the database terminology or the Python-like dictionaries.

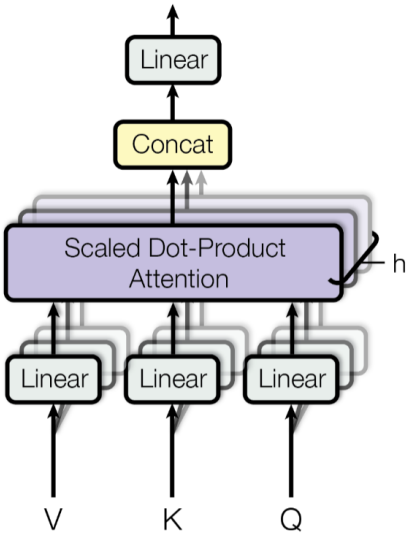


* this could be a Python dictionary or a database table.

Scaled Dot-Product Attention



Multi-Head Attention



Attention vector for every word using this formula:

$$Z = \text{softmax}\left(\frac{QK^T}{\sqrt{\text{Dimension of vector } Q, K \text{ or } V}}\right)V$$

Self-attention is calculated by taking the dot product of the query and key, scaled by a factor, and applying a softmax function to obtain attention weights. These attention weights determine the importance of each word's value for the current word.

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

$$\text{self attention} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_{\text{model}}}} + M\right)$$

Output:

The final layer of the decoder is a linear projection followed by a softmax activation function. It produces a probability distribution over the vocabulary, allowing the model to generate the output word by sampling from this distribution.

Softmax:

The softmax function is a mathematical function that converts a vector of K real numbers into a probability distribution of K possible outcomes. It is a generalization of the logistic function to multiple dimensions, and used in multinomial logistic regression. The softmax function is often used as the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output classes. The formula for the standard (unit) softmax function is as follows:

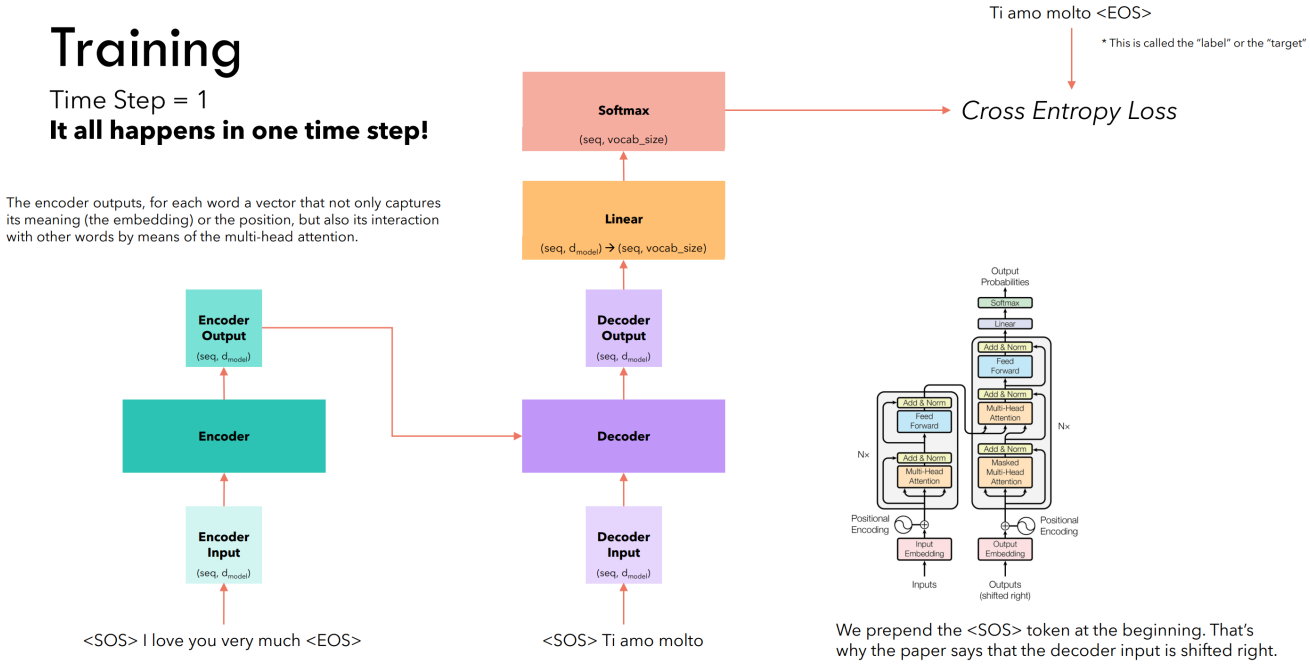
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Linear

it just has weights not biases.

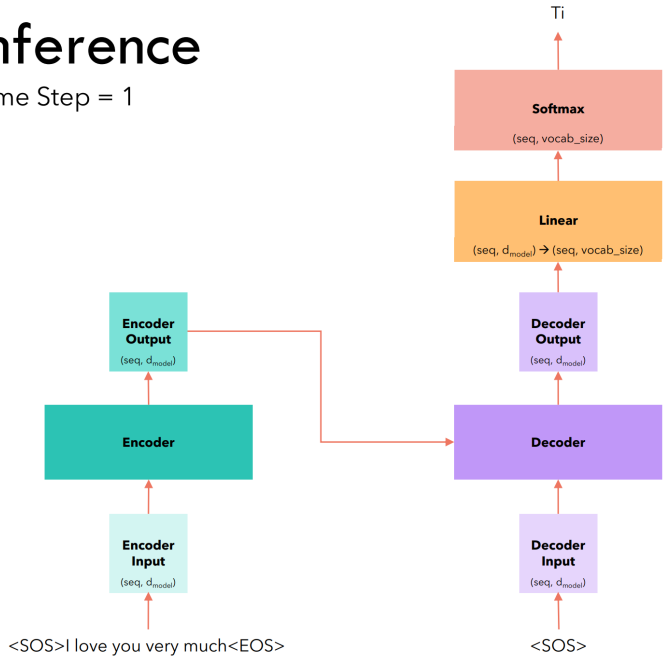
Training:

$$H(P^*|P) = -\sum P^*(i)\log(P(i))$$



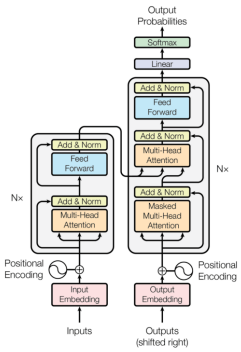
Inference

Time Step = 1



We select a token from the vocabulary corresponding to the position of the token with the maximum value.

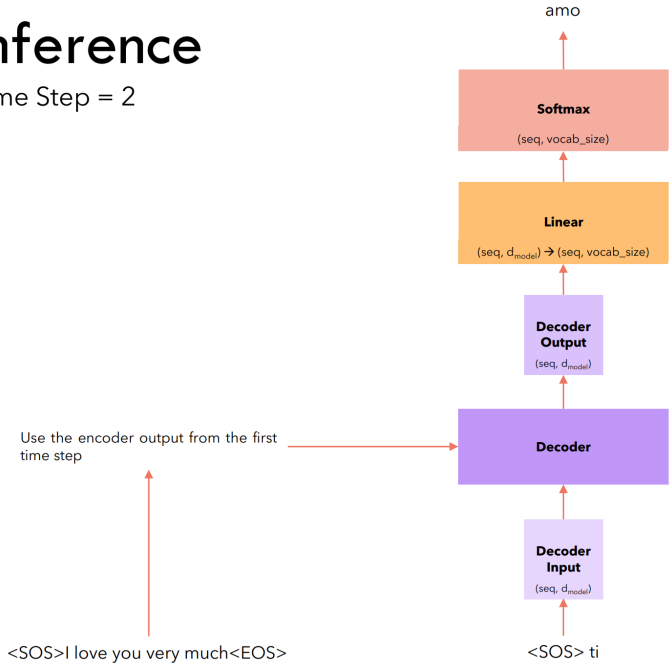
The output of the last layer is commonly known as **logits**



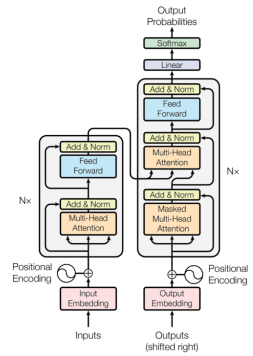
* Both sequences will have same length thanks to padding

Inference

Time Step = 2



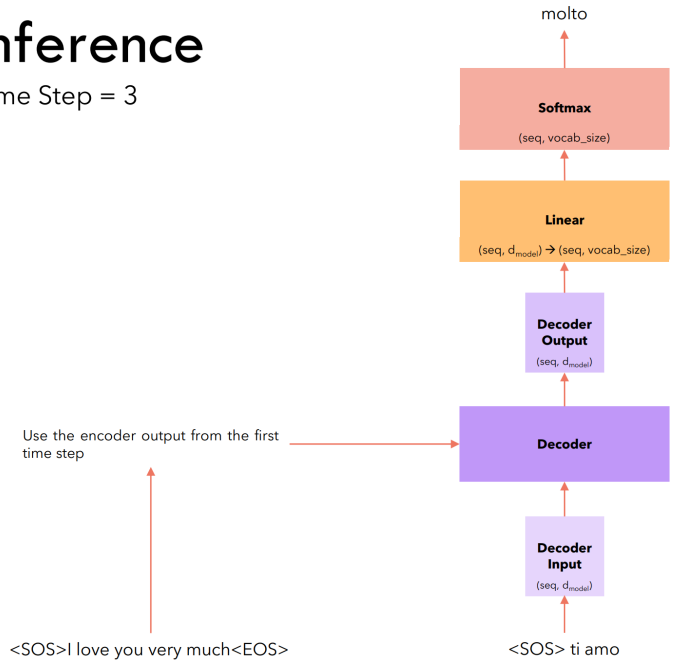
Since decoder input now contains **two** tokens, we select the softmax corresponding to the second token.



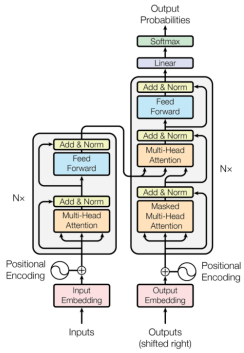
Append the previously output word to the decoder input

Inference

Time Step = 3



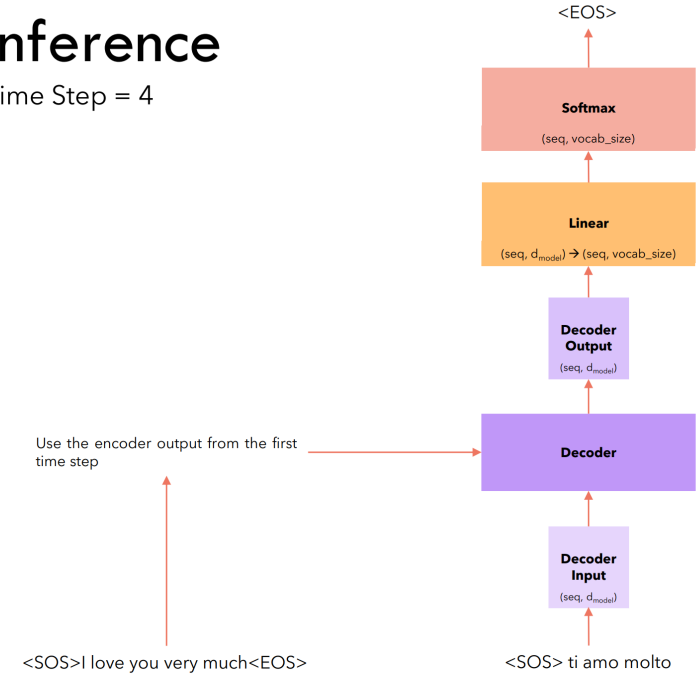
Since decoder input now contains **three** tokens, we select the softmax corresponding to the third token.



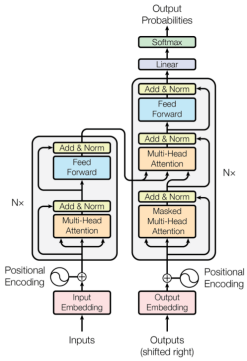
Append the previously output word to the decoder input

Inference

Time Step = 4



Since decoder input now contains **four** tokens, we select the softmax corresponding to the fourth token.



Append the previously output word to the decoder input