Attention Mechanism and Transformer Models

Agenda

- Key Questions
- What are Transformer Models?
- How does a Transformer model work?
- What are the components of the Encoder?
- How does the Self-Attention Mechanism work?
- What are the components of the Decoder?

Key Questions

What are Transformer Models?

How does a Transformer model work?

What are the components of the Encoder?

How does the Self-Attention Mechanism work?

What are the components of the Decoder?

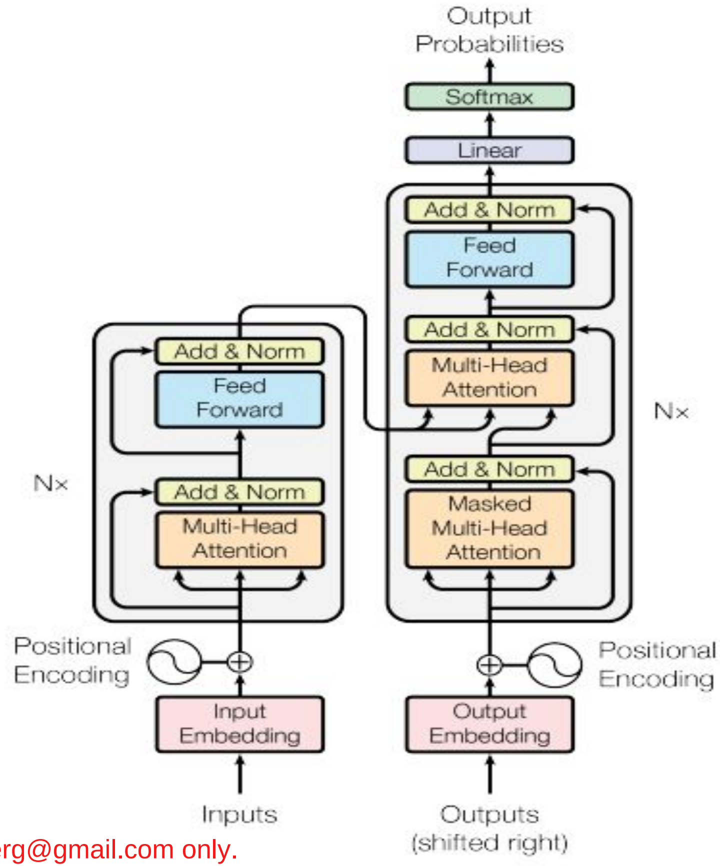
What are Transformer Models?

Transformers are a **type of neural network architecture**

Transformers were **introduced** in a paper by **Vaswani et al. in 2017**

Transformers are based on the idea of **self-attention**

Transformers consist of an **encoder** and a **decoder**

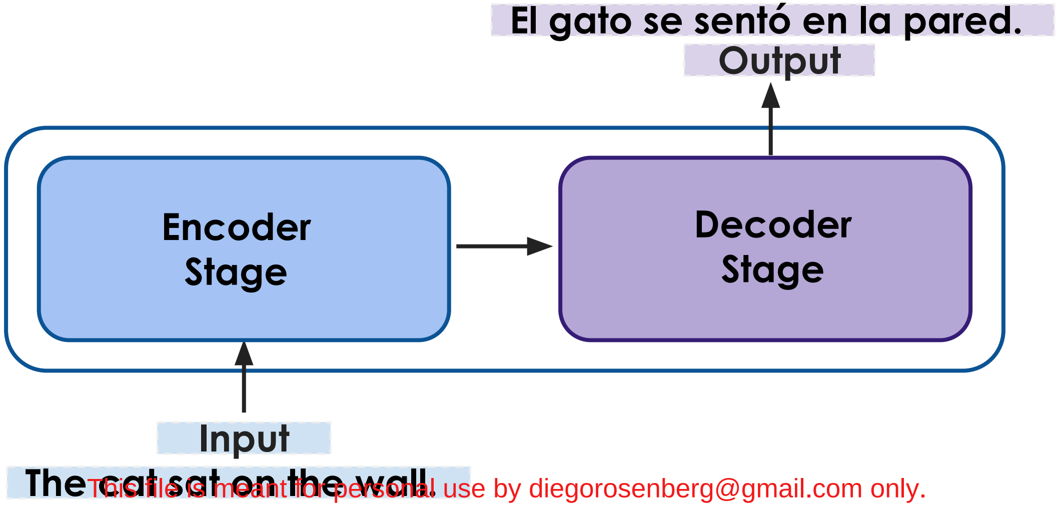


Source: Image from the original research paper [Attention Is All You Need]

How does a Transformer model work?

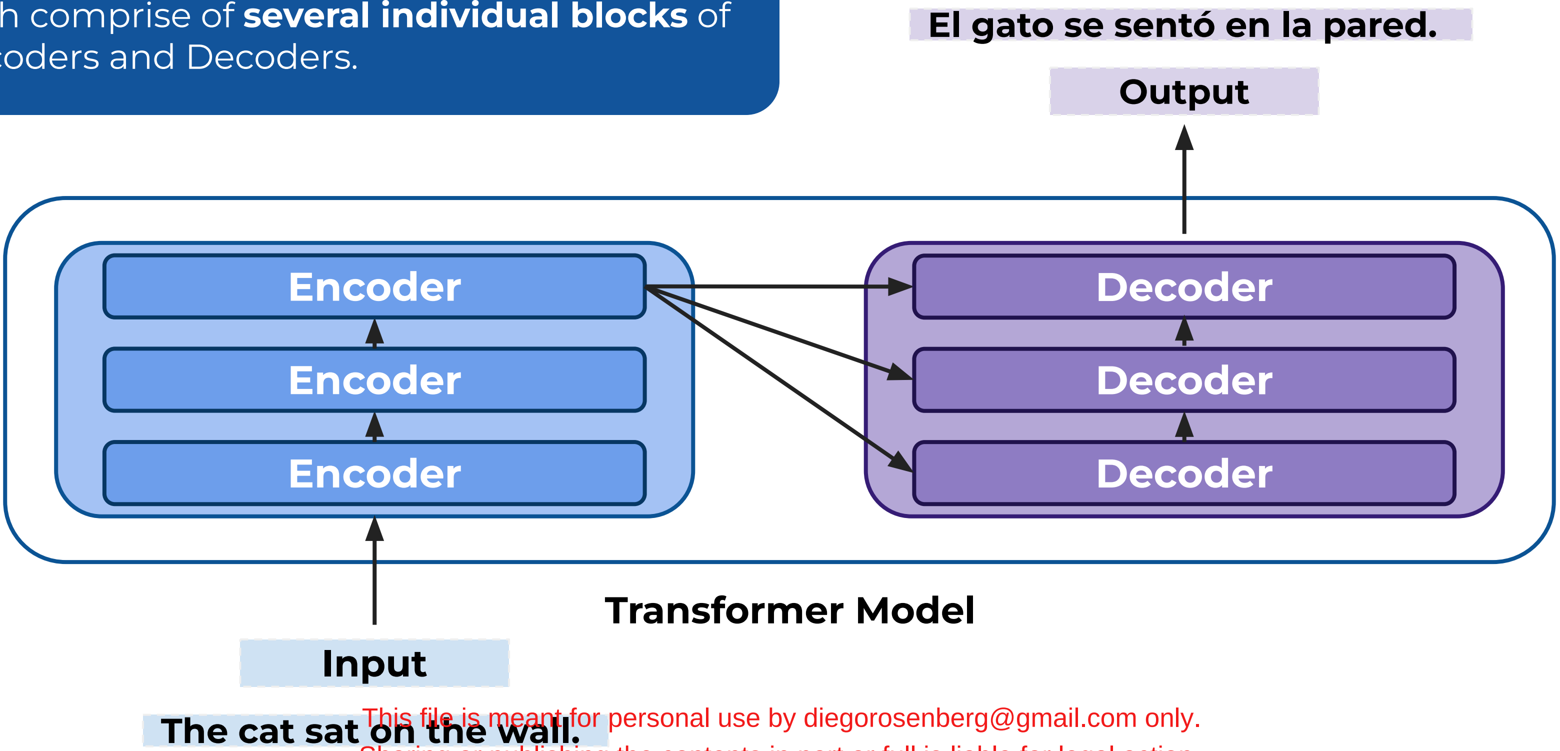
The **encoder** takes in a sequence of tokens (e.g. words or characters) and outputs a **latent representation**

The **decoder** then takes this latent representation as input and outputs a **sequence of tokens**



The Transformer Model - High-level Flow

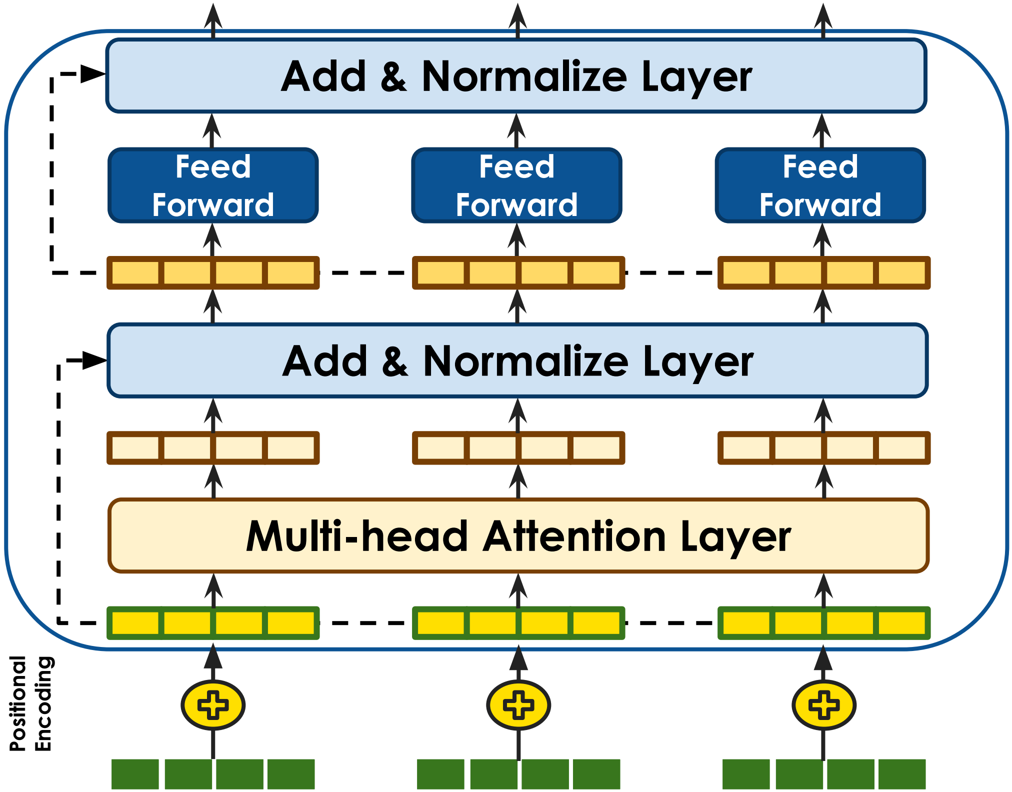
In reality, the **Encoder** and **Decoder** stage each comprise of **several individual blocks** of Encoders and Decoders.



What are the components of the Encoder?

The Encoder block of a Transformer architecture consists of the following components:

1. **Multi-head Attention**: A stack of self-attention layers that allows the Encoder to attend to different parts of the input sequence simultaneously.
2. **Feedforward Neural Network**: Processes the outputs of the Multi-head Attention layer using a standard fully connected neural network with activations like ReLU.
3. **Residual Connections and Layer Normalization**: Improves the flow of information through the Encoder and avoids the vanishing gradient problem. These are added after each sub-layer.
4. **Positional Encoding**: Typically added to the input embeddings of the Encoder to provide positional information for words, using a set of learned sinusoidal functions.



How does the Self-Attention Mechanism work?

The **self-attention mechanism** lies at the **core** of **transformer models**

Self attention allows us to generate a **context-aware representation** of **each token** in the input

The **context-aware representation** of **each token** is generated with respect to all other tokens in the input

The context-aware representation **focuses** on the **relevant parts of the input** for a given task

Self Attention - Computation

The **steps** to get the **context-aware representation** for **each** of the **token** in the **input** are

‘query’ a key

against all ‘keys’ to get **self attention**

combine with **self attention** to get **‘value’**

the **query** designates the **focused token**

the **keys** represents **all tokens** (including the query)

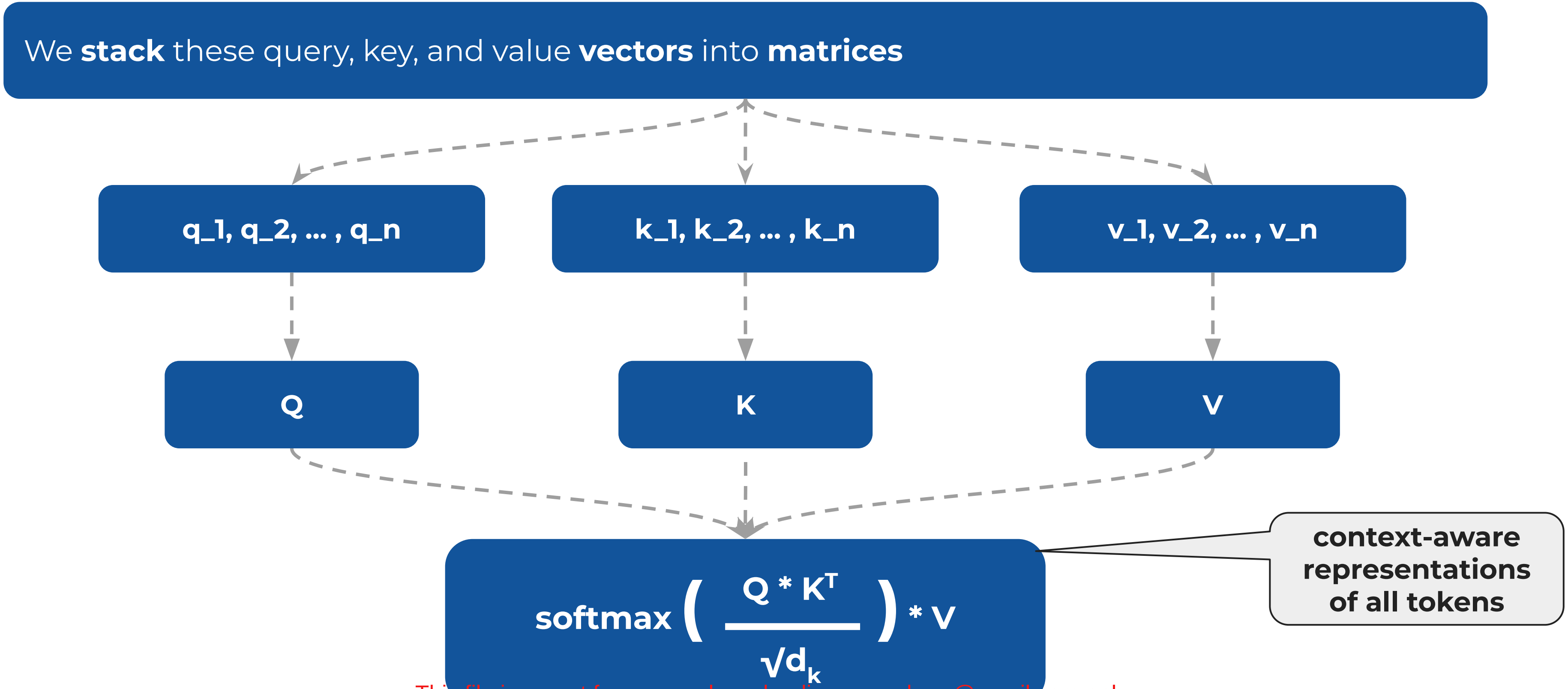
values retains the **original information**

query vector (q)

key vector (k)

value vetor (v)

Self Attention - Computation



What are the components of the Decoder?

Most of these operations in the Decoder are identical to the Encoder.

Self-Attention

Add & Normalize Layer

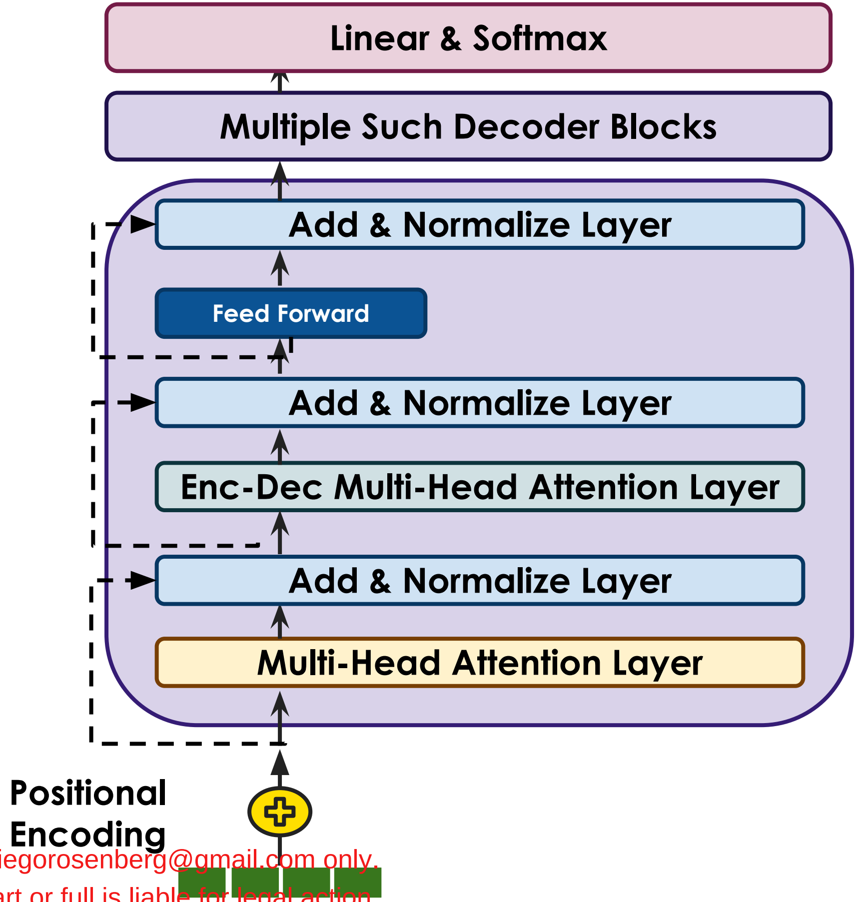
Feed Forward

But there are a few other operations **unique to the Decoder.**

Masking

Encoder-Decoder Attention Layer

Linear & Softmax



Components of the Decoder

Masking	Encoder-Decoder Attention	Linear & Softmax
Involves **hiding** (masking) **information** from the **future** to keep the model focused on the present and past during each run	**Aligns** the **decoder's output** with the **context** provided by the **encoder** by enabling selective attention to different segments of the input sequence	**Linear** layer **converts** contextual information into a **format suitable** for subsequent computations **Softmax** produces **probability scores**, facilitating the selection of the most likely output token

Great Learning

Happy Learning !