Transformers
An Intuitive Understanding

Sandeep Arora : Principal Architect
https://www.linkedin.com/in/sandeeparora/

Attention is All You Need

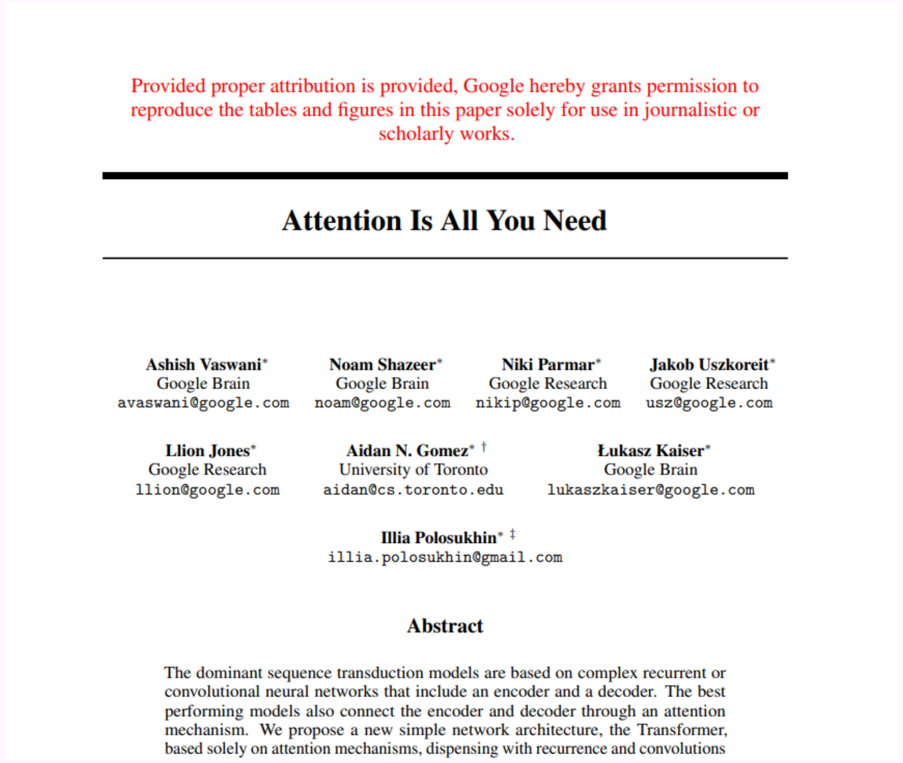
The Transformer model is based on the 2017 Google paper titled "Attention is all you need"

Transformer

The paper introduced a new **deep learning architecture** known as the **transformer**

So, Relax

It is just a complex software program. You don't need to understand every detail of its inner workings. To begin, get an intuitive, high-level understanding.



Provided proper attribution is provided, Google hereby grants permission to reproduce the tables and figures in this paper solely for use in journalistic or scholarly works.

Attention Is All You Need

Ashish Vaswani[*] Google Brain avaswani@google.com

Noam Shazeer[*] Google Brain noam@google.com

Niki Parmar[*] Google Research nikip@google.com

Jakob Uszkoreit[*] Google Research usz@google.com

Llion Jones[*] Google Research llion@google.com

Aidan N. Gomez[*] [†] University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser[*] Google Brain lukaszkaiser@google.com

Illia Polosukhin[*] [‡] illia.polosukhin@gmail.com

Abstract

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

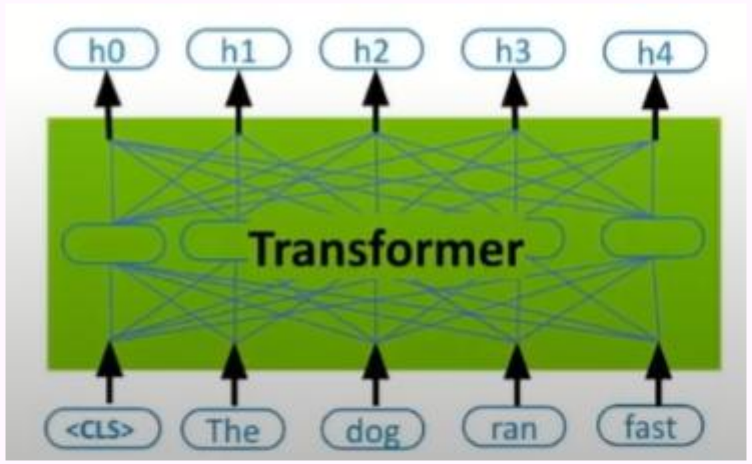
https://arxiv.org/pdf/1706.03762

Architecture

Transformer

With transformer architecture, now you can enter the entire sentence at once.

- The transformer takes the entire sentence as input
- It will understand it using the **encoder block**, and then generate the output using the **decoder**, capturing the full context of the sentence in a parallel and efficient way.



https://www.youtube.com/watch?v=PXc_SlnT2g0

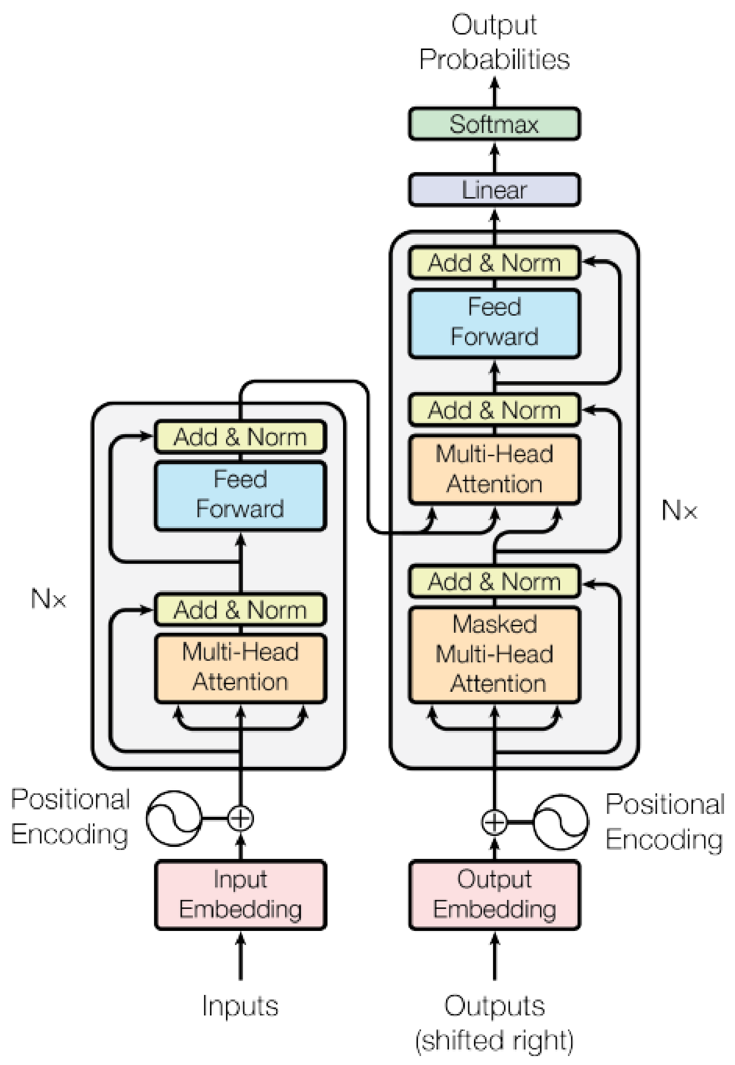


Figure 1: The Transformer - model architecture.

3.1 Encoder and Decoder Stacks

https://arxiv.org/pdf/1706.03762

Transformer

Intuitive Understandings

- Let us first learn how it works when you use (**inference**) it
- Then we will learn how it is **trained**
- Along the way we will learn important concepts

The model has two main parts:
Encoder:
Understands the input and creates feature representations.
Decoder:
Uses those features to generate the output sequence.

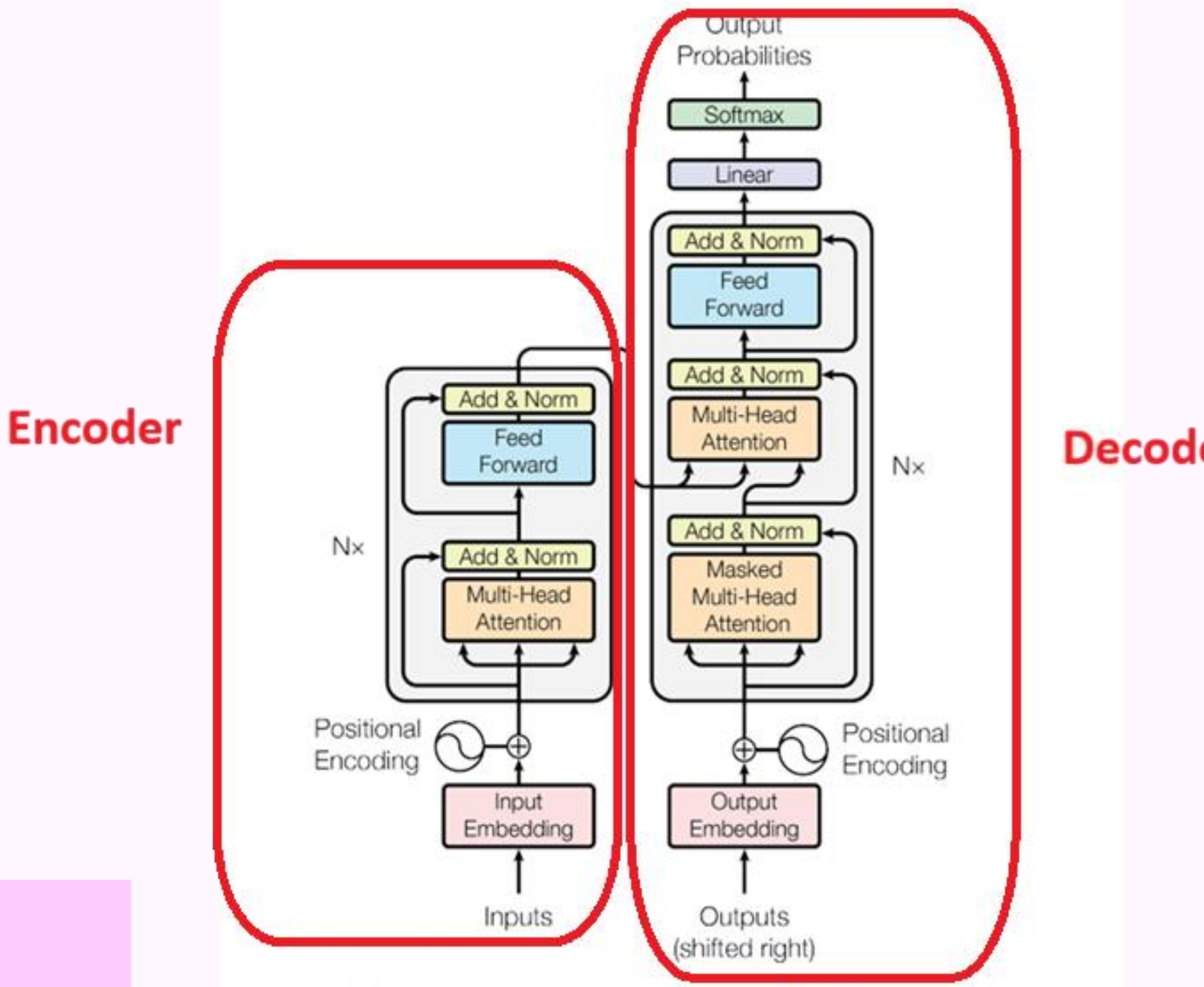


Figure 1: The Transformer - model architecture.

Transformer -Encoder

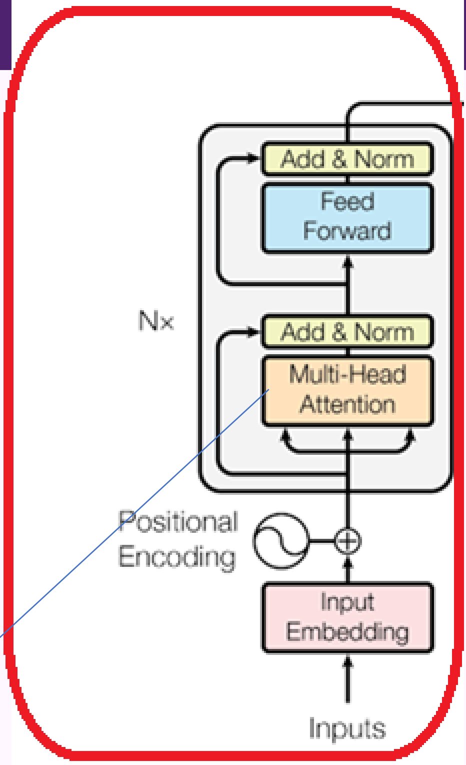
*What is the **Encoder**? What does it do?*

The encoder takes the entire input sentence and builds a rich understanding of it. The understanding is stored in a **matrix** format

*How does it work? Say I enter "**When was Rome Built**?"*

- The input sentence is broken into tokens → ["When", "was", "Rome", "built", "?"]
- Each token is converted into a vector using a word embedding
- A **positional embedding** is added to each token to tell the model the order (e.g., "Rome" came after "was")
- The combined vectors go into a stack of **Nx encoder blocks** (Nx = repeated N times, like 6 or 12 layers)
- In each block, the model uses **multi-head attention** and **feed-forward networks** to build a deep understanding of each word in context
- The final output of the **encoder layer** is a set of **context-rich embeddings** (vectors) — one for each input token — capturing the meaning of each word in relation to the whole sentence.

Encoder



https://arxiv.org/pdf/1706.03762

Transformer -Encoder

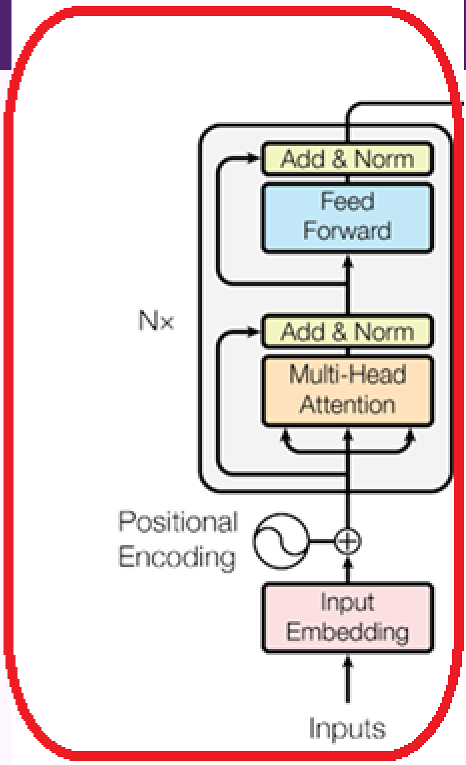
*What is the **Encoder**? What does it do?*

The encoder takes the entire input sentence and builds a rich understanding of it. The understanding is stored in a **matrix** format

*How does it work? Say I enter "**When was Rome Built**?"*

- The input sentence is broken into tokens → ["When", "was", "Rome", "built", "?"]
- Each token is converted into a vector using a word embedding
- A **positional embedding** is added to each token to tell the model the order (e.g., "Rome" came after "was")
- The combined vectors go into a stack of **Nx encoder blocks** (Nx = repeated N times, like 6 or 12 layers)
- In each block, the model uses **multi-head attention** and **feed-forward networks** to build a deep understanding of each word in context
- The final output of the **encoder layer** is a set of **context-rich embeddings** (vectors) — one for each input token — capturing the meaning of each word in relation to the whole sentence.

Encoder

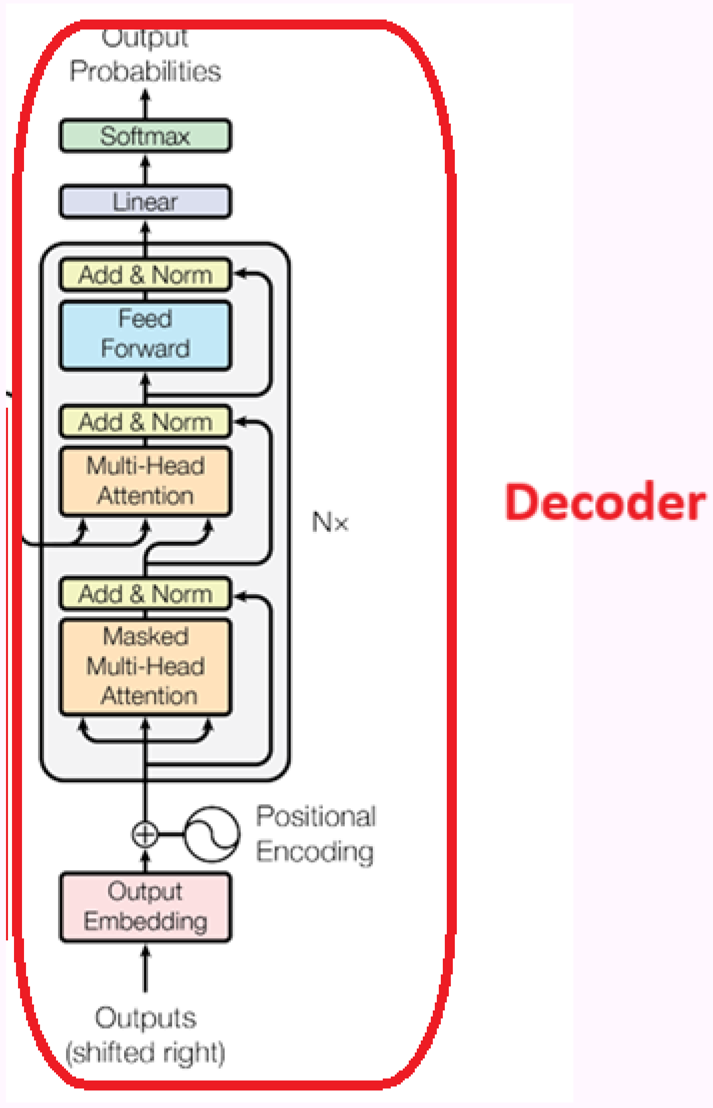


https://arxiv.org/pdf/1706.03762

Transformer -Decoder

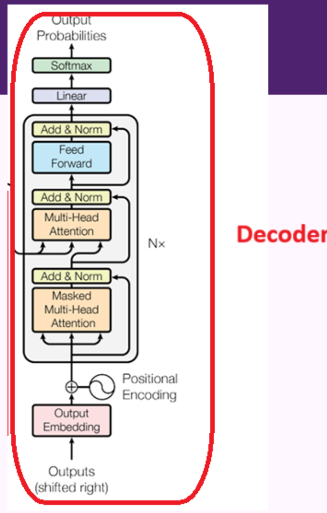
*What is the **Decoder**? What does it do?*

A **decoder** is the part of the Transformer that **generates the output** (like a translated sentence or answer), one token at a time.



https://arxiv.org/pdf/1706.03762

Transformer -Decoder



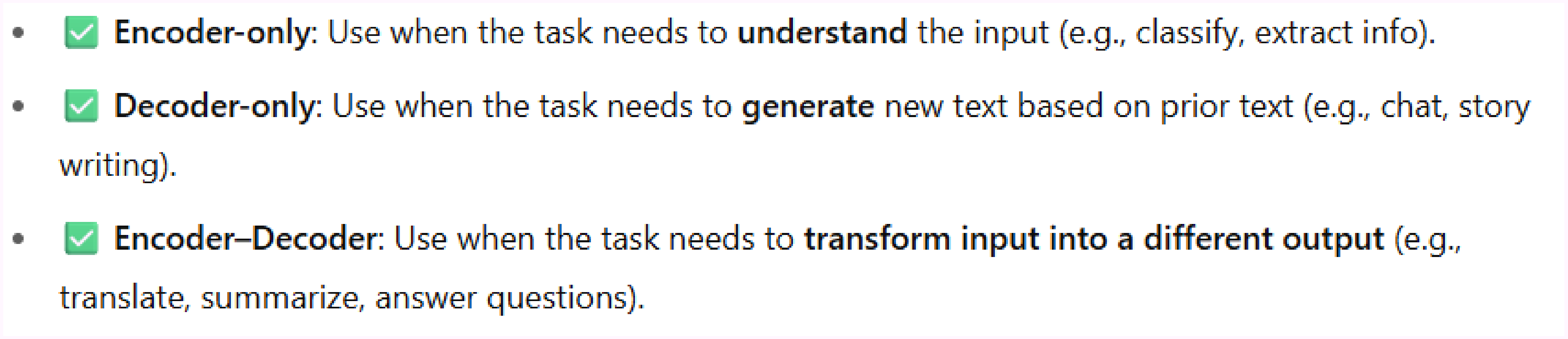
*How does the decoder work? Say I enter "**When was Rome Built**?"*

- The **encoder** processes the input and outputs context-rich embeddings (one per input token).
- The **decoder** starts with a special start token <START> as input.
- It uses masked **multi-head attention** to focus only on tokens already generated, not future ones.
- Then it performs **cross-attention** to read relevant info from the **encoder output** (e.g., focus on "Rome" and "built").
- Passes through layers that help it understand what to say next based on **input from encoder** and previous output.
- It predicts the next word (like "Rome"), adds it to the sentence, and uses that as input to guess the next one.
- Then it predicts the next word (like "was") based on what it's generated so far — and continues this process until the answer is complete.

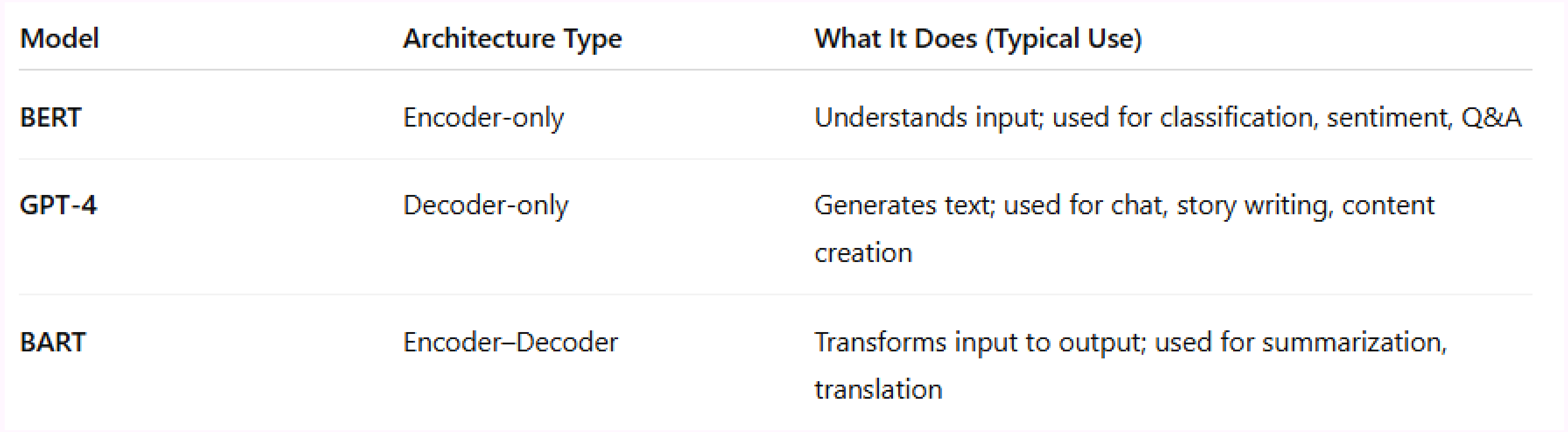
https://arxiv.org/pdf/1706.03762

Encoder or Decoder models

Generally speaking below apply but this is not fixed.

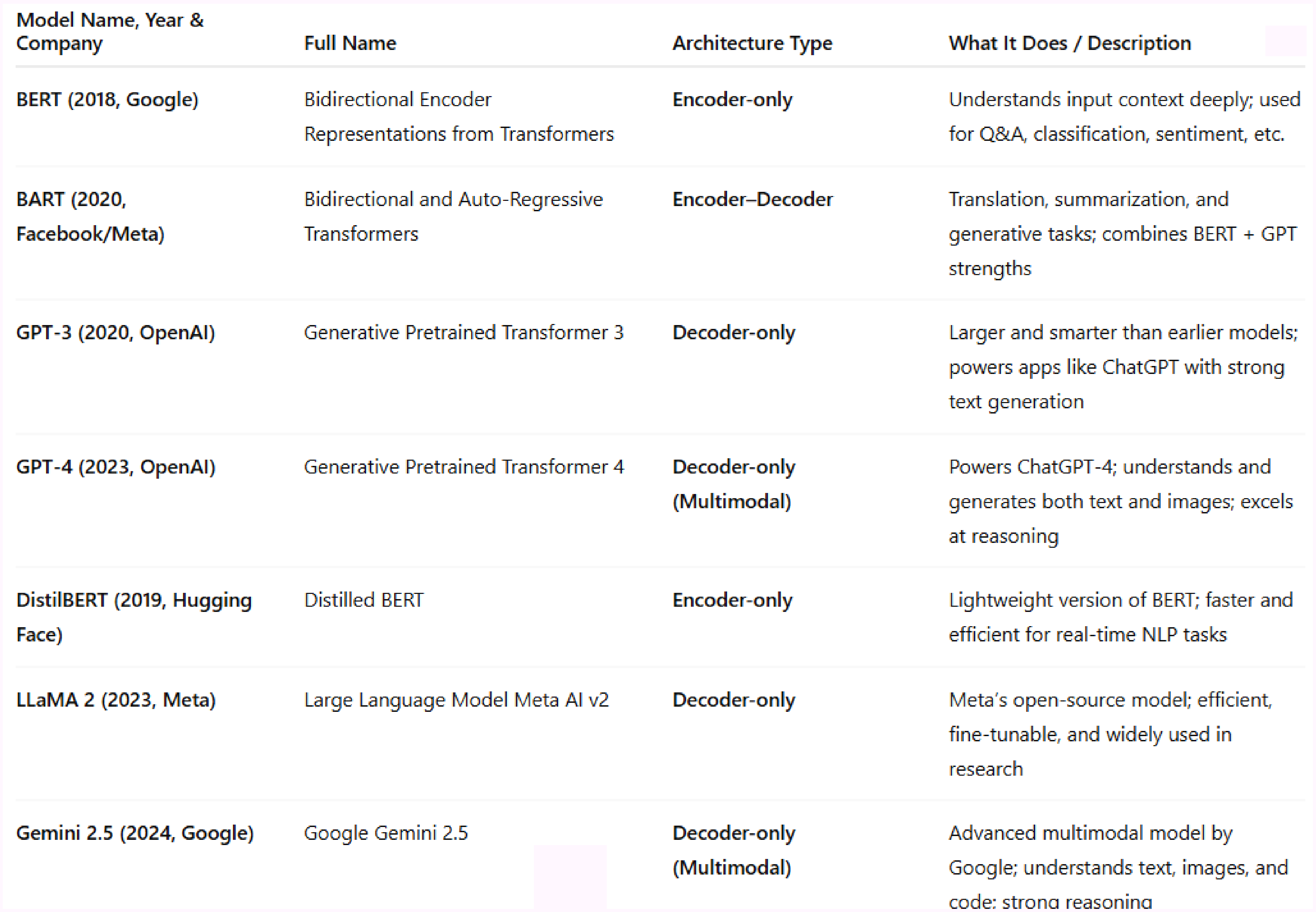


- ✅ **Encoder-only**: Use when the task needs to **understand** the input (e.g., classify, extract info).
- ✅ **Decoder-only**: Use when the task needs to **generate** new text based on prior text (e.g., chat, story writing).
- ✅ **Encoder–Decoder**: Use when the task needs to **transform input into a different output** (e.g., translate, summarize, answer questions).



Model	Architecture Type	What It Does (Typical Use)
BERT	Encoder-only	Understands input; used for classification, sentiment, Q&A
GPT-4	Decoder-only	Generates text; used for chat, story writing, content creation
BART	Encoder–Decoder	Transforms input to output; used for summarization, translation

Examples



Model Name, Year & Company	Full Name	Architecture Type	What It Does / Description
BERT (2018, Google)	Bidirectional Encoder Representations from Transformers	**Encoder-only**	Understands input context deeply; used for Q&A, classification, sentiment, etc.
BART (2020, Facebook/Meta)	Bidirectional and Auto-Regressive Transformers	**Encoder–Decoder**	Translation, summarization, and generative tasks; combines BERT + GPT strengths
GPT-3 (2020, OpenAI)	Generative Pretrained Transformer 3	**Decoder-only**	Larger and smarter than earlier models; powers apps like ChatGPT with strong text generation
GPT-4 (2023, OpenAI)	Generative Pretrained Transformer 4	**Decoder-only (Multimodal)**	Powers ChatGPT-4; understands and generates both text and images; excels at reasoning
DistilBERT (2019, Hugging Face)	Distilled BERT	**Encoder-only**	Lightweight version of BERT; faster and efficient for real-time NLP tasks
LLaMA 2 (2023, Meta)	Large Language Model Meta AI v2	**Decoder-only**	Meta's open-source model; efficient, fine-tunable, and widely used in research
Gemini 2.5 (2024, Google)	Google Gemini 2.5	**Decoder-only (Multimodal)**	Advanced multimodal model by Google; understands text, images, and code; strong reasoning

Appendix

https://community.openai.com/t/is-gpt-group-of-models-decoder-only-model/286586/2