Faculty of SET / School of Computer and Mathematical Sciences

COMP SCI 3007/7059/7659
Artificial Intelligence
Large Language Models

Acknowledgement of Country

We acknowledge and pay our respects to the Kaurna people, the traditional custodians whose ancestral lands we gather on.

We acknowledge the deep feelings of attachment and relationship of the Kaurna people to the country and we respect and value their past, present and ongoing connection to the land and cultural beliefs.

Large Language Models

concept, structure, learning

Xinyu Zhang

Research Fellow, AIML, The University of Adelaide (UoA)

29 May 2025

You've Probably Met an LLM!

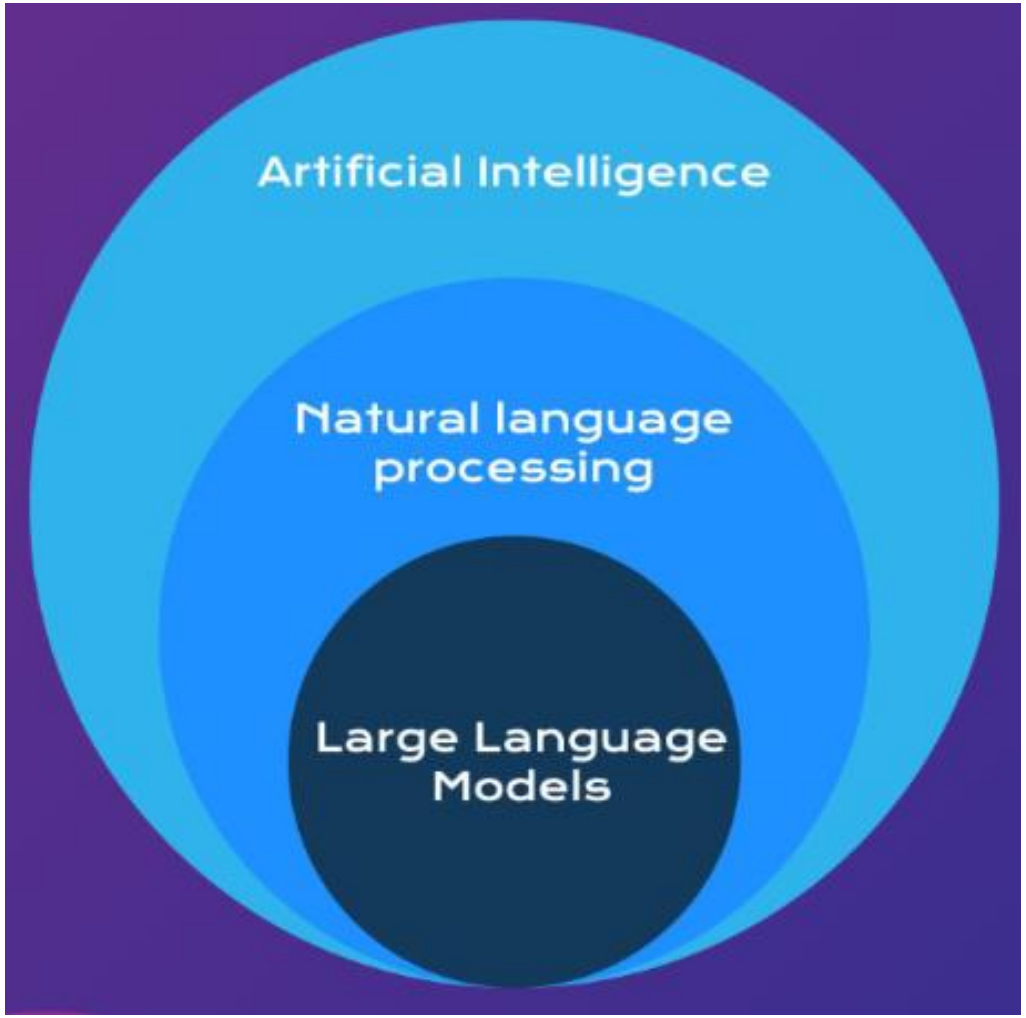
- ChatGPT (or other conversational AI)
- Grammarly (or other writing assistants)
- Google Search (AI-powered summaries)
- Translation tools (Google Translate, DeepL)

Outline

- What is LLMs?
- The Road to LLMs
- How do LLMs think?
  - Modeling: Tokenization; Embedding; Model Structure
  - Training: Pre-training; Fine-tuning; Prompting; Reinforce Learning with Human Feedback (RLHF)
- What LLMs can do?
- Conclusion and QA

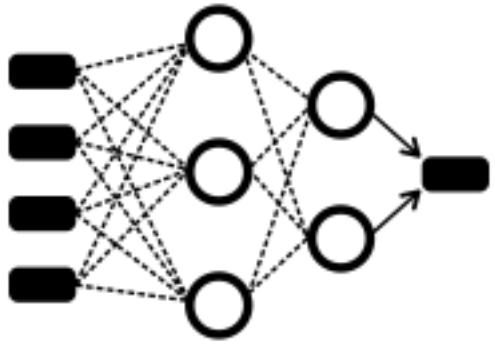
How does LLMs relate to AI?

- Artificial intelligence
  - Simulating human abilities
- Natural Language Processing (NLP)
  - Computational linguistics
  - Machine learning
- Large Language Models (LLMs)
  - Focused on understanding, interpreting and generating **human-like** language



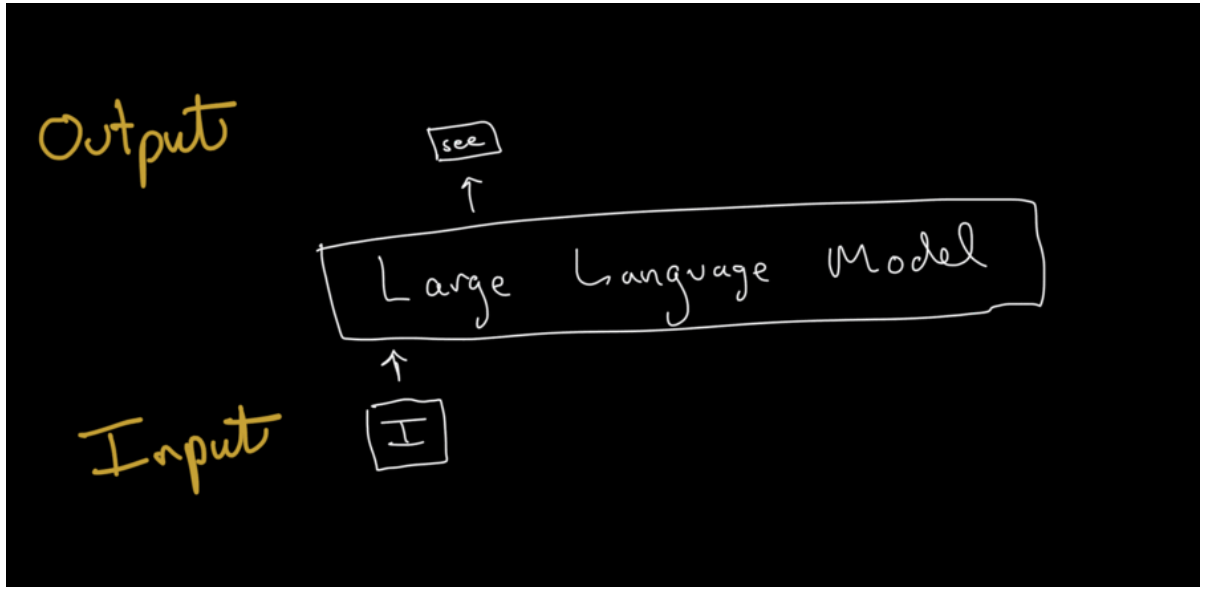
LLM = Large + Language + Model

- **Large**:
  - Trained on VAST amounts of text data (books, websites, articles – think libraries of information!).
  - Model maybe also large
- **Language**:
  - Understands, generates, and manipulates human language (text).
- **Model**:
  - An AI system, specifically a type of neural network, that learns patterns and relationships from data to make predictions.

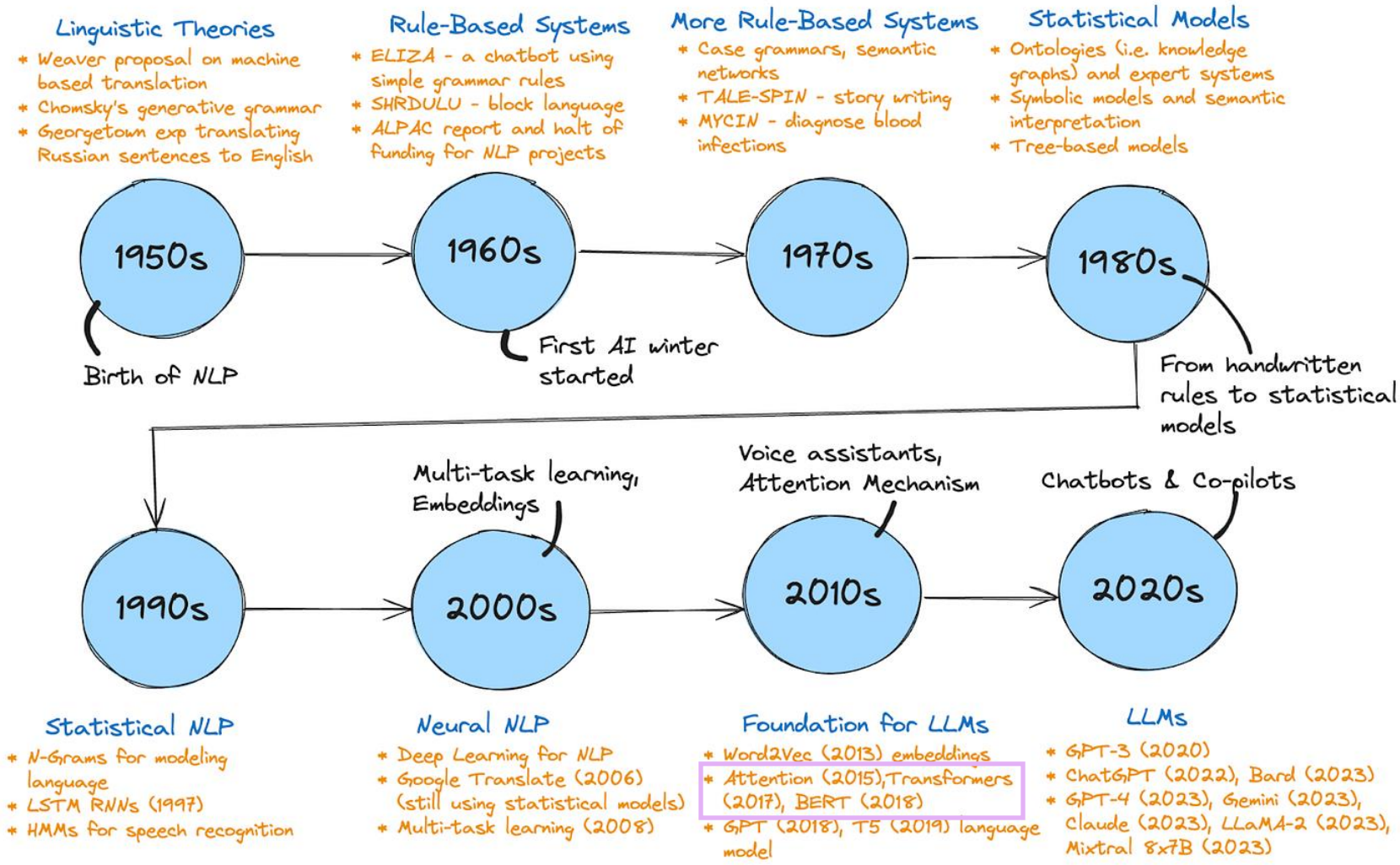


At its Heart: LLMs are Super-Powered Predictors

- An LLM's primary goal is to predict the next word (or token) in a sequence given the preceding words.
- Example: "The cat sat on the _____." (LLM predicts "mat", "floor", "chair", …)
- It's like a very, very sophisticated autocomplete, but on a massive scale and with a deep understanding of context.

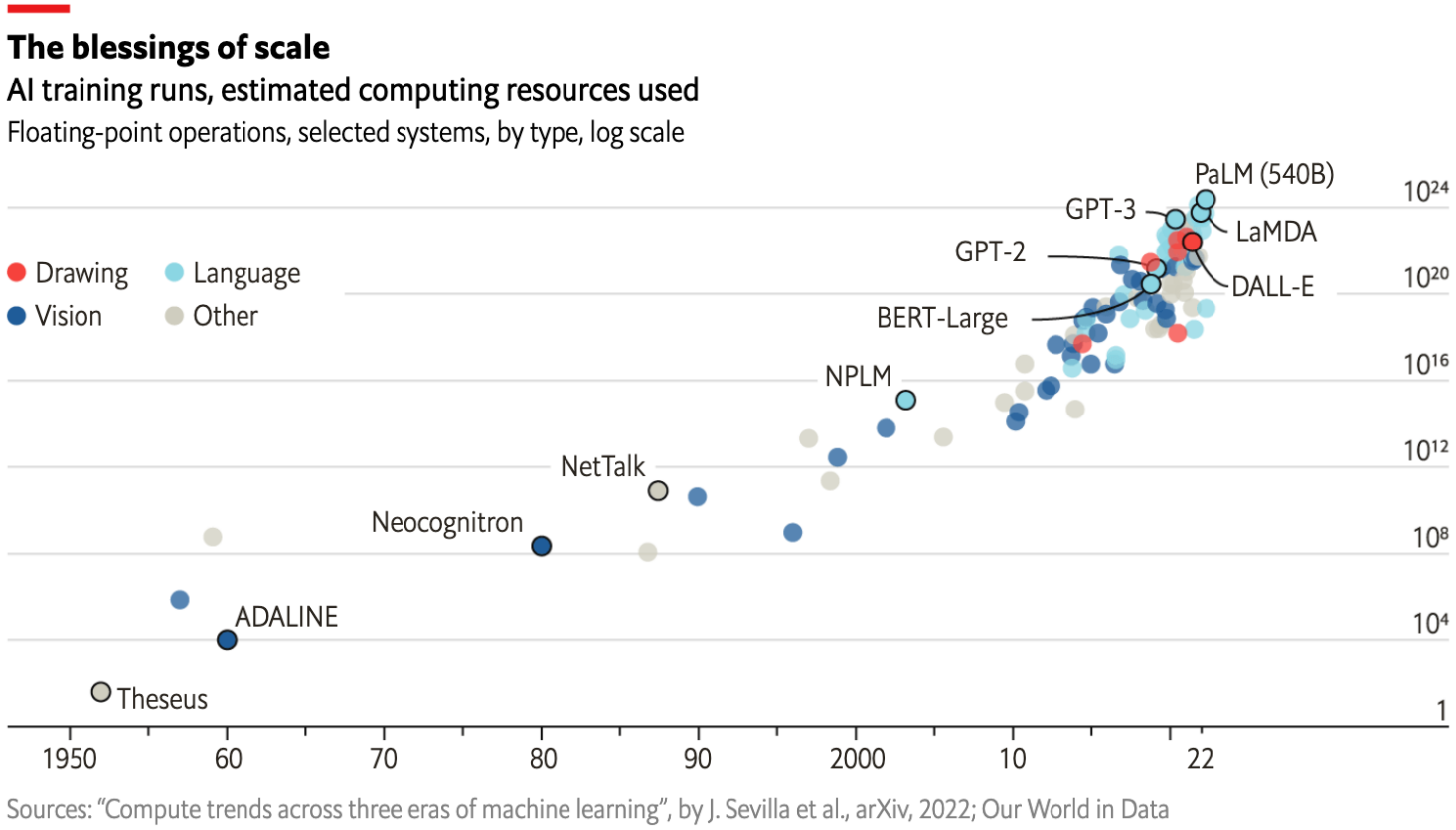


The Road to LLMs: A Quick History



Liu et al., Representation Learning for Natural Language Processing, Springer, 2020

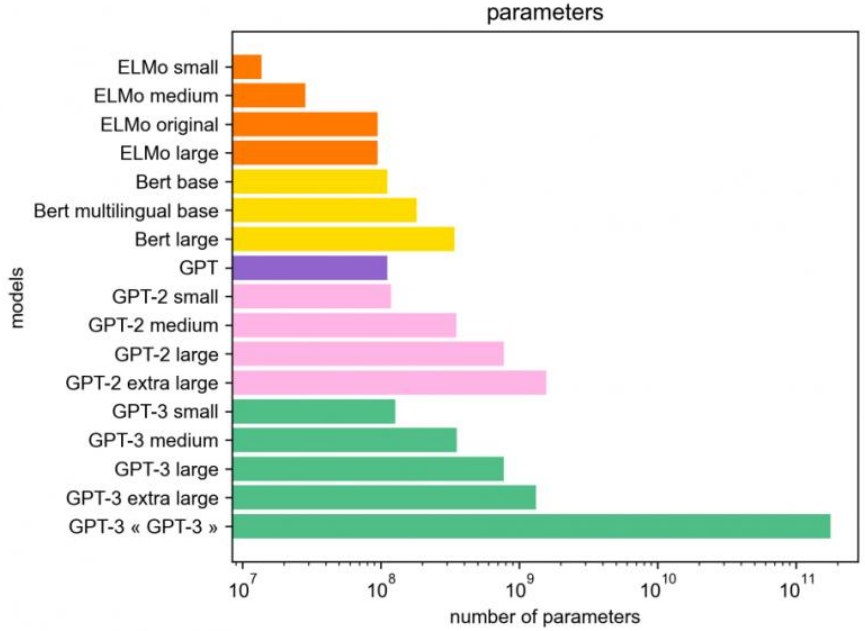
Larger and Larger Models

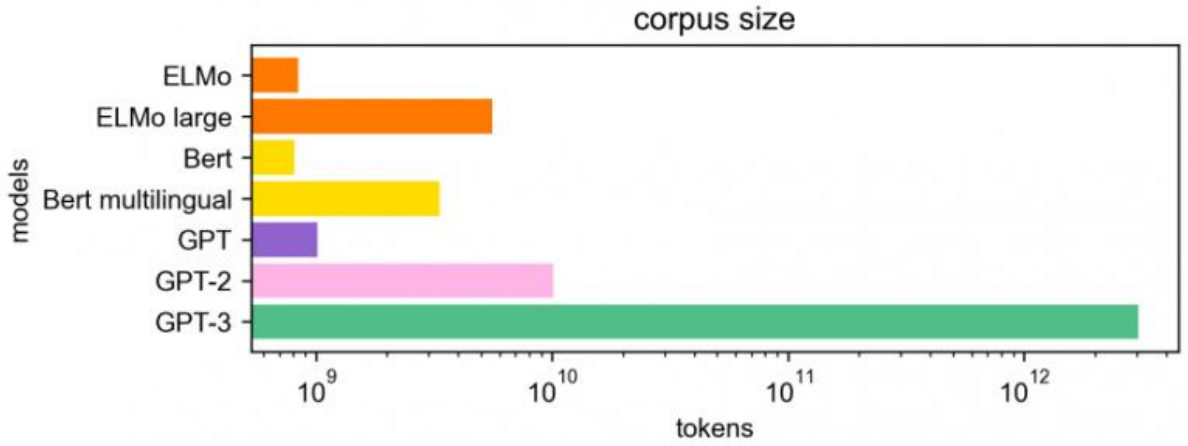


Larger model sizes → larger compute, more expensive during inference

How large are “large” LMs?

- Today, we mostly talk about two camps of models:
  - **Medium-sized models**: BERT/RoBERTa models (100M or 300M), T5 models (220M, 770M, 3B)
  - **“Very” large LMs**: models of 100+ billion parameters



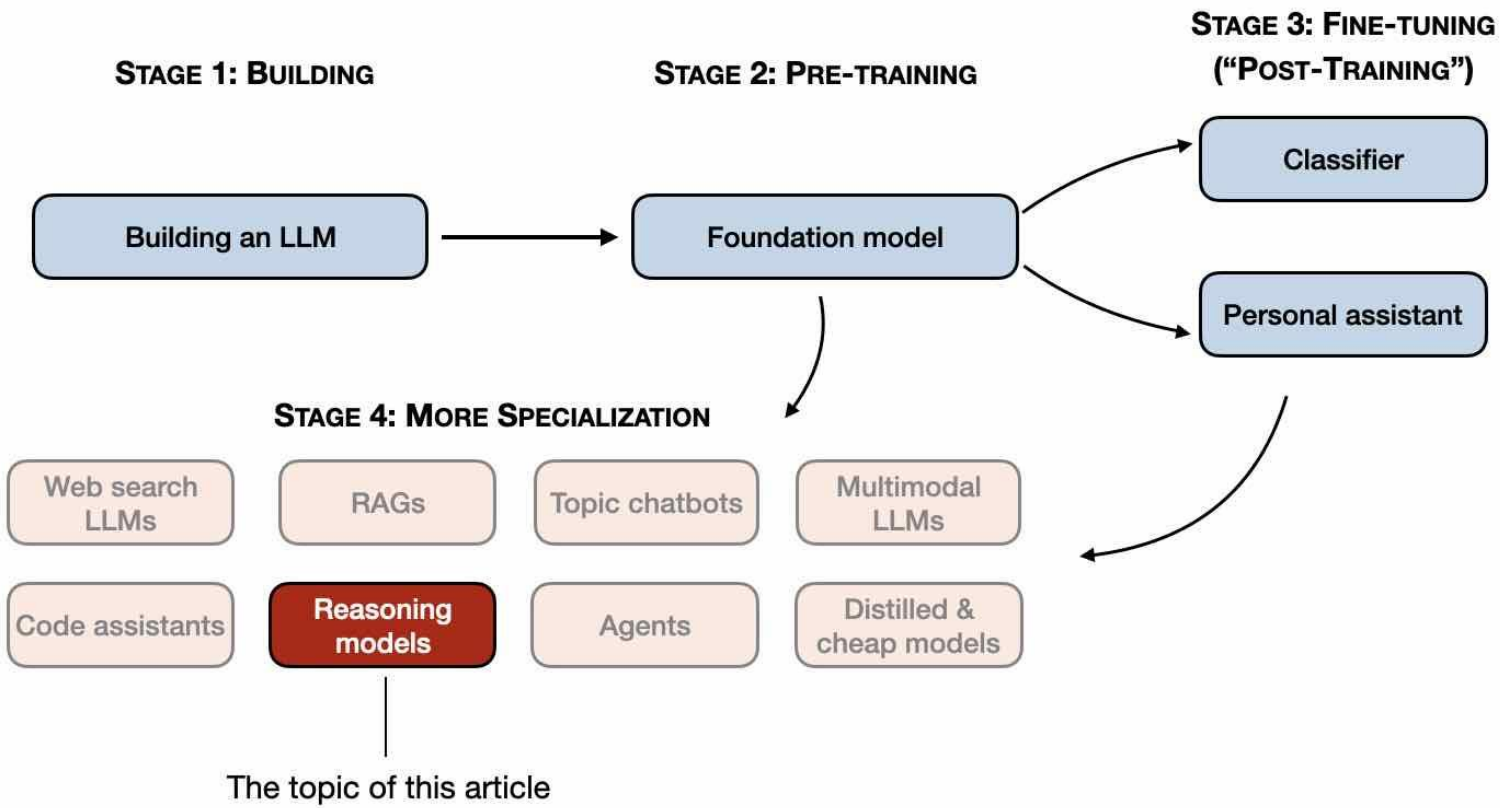


More recent models: PaLM (540B), OPT (175B), BLOOM (176B)...

https://huggingface.co/blog/large-language-models
Image source: https://hellofuture.orange.com/en/the-gpt-3-language-model-revolution-or-evolution/

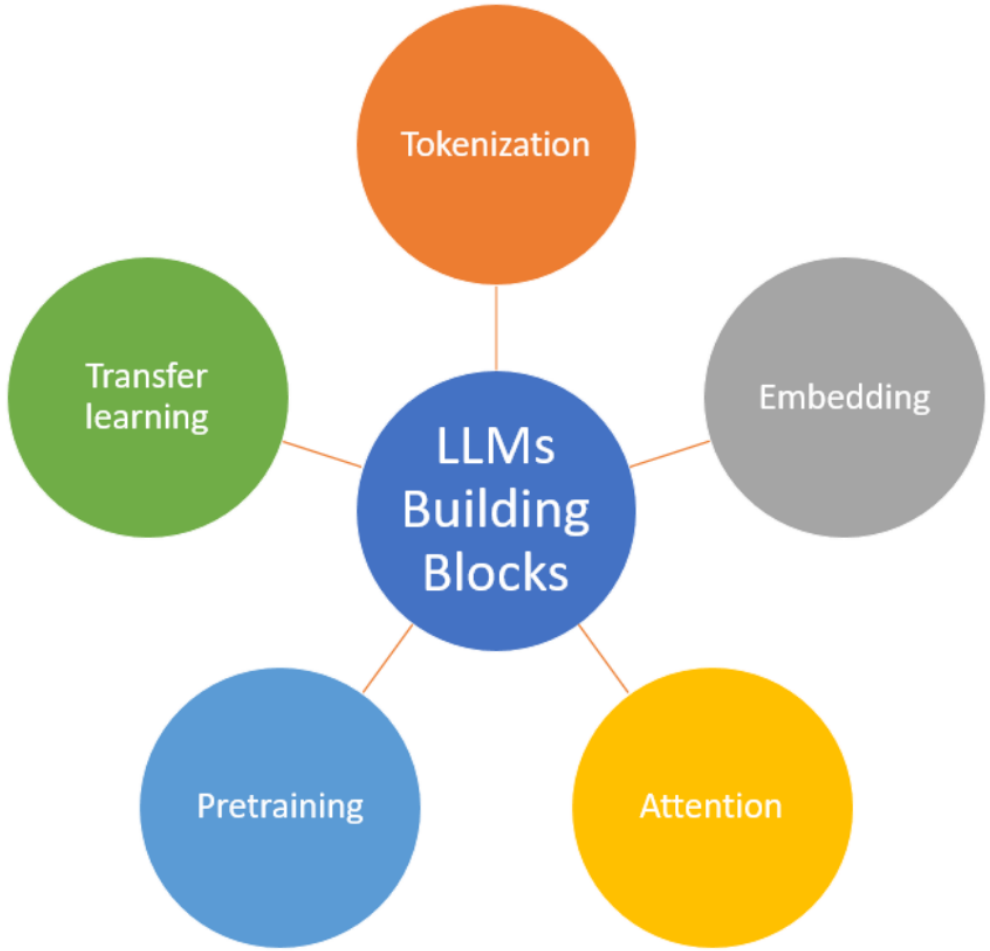
How Do LLMs "Think"?

- It's Not Magic!
- LLM = Large + Language + Model



Build an LLM

- Tokenization
- Embedding
- Model Structure:
  - RNN & LSTM
  - Transformer (Attention)
- Training
  - Pretraining
  - Fine-tuning
  - Prompting
  - RLHF



Tokenization

- How do we represent an input text?
- **Tokenization** is the process of converting a sequence of text into smaller, manageable pieces called **tokens**.
- **Tokens** could be **words, subwords, or characters**, depending on the tokenization strategy
- The tokenizer then assigns each token a **unique token ID** based on a pre-defined vocabulary.

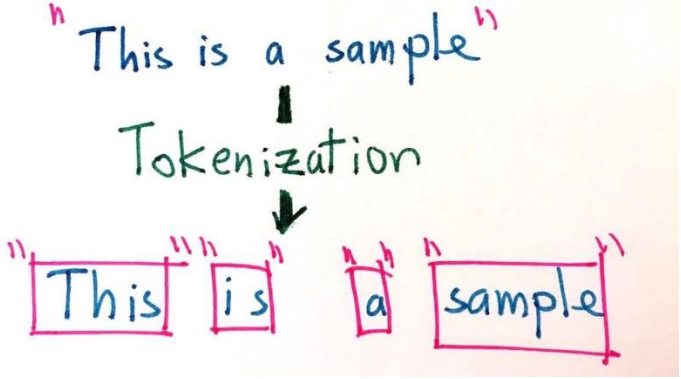


Image source: https://www.nomidl.com/natural-language-processing/what-is-tokenization-in-nlp/

Tokenization

- **World-level** tokenization
  - Different words have different IDs.
- **Character-level** tokenization
  - Split the text into individual characters.
- **Subword-level** tokenization
  - Breaks down words into smaller parts or subwords, meaningful pieces
  - “annoyingly” -> “annoying” and “ly”, both with meaning

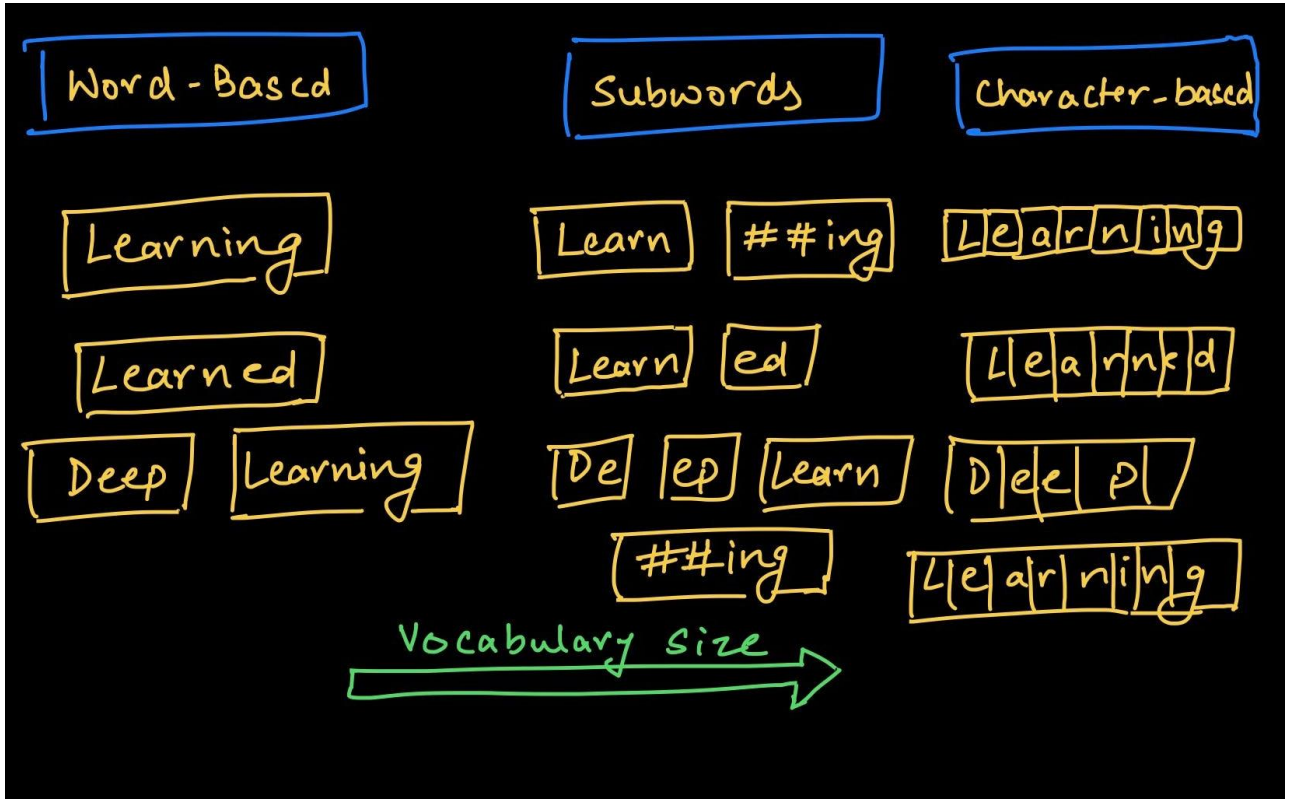


Image source: https://medium.com/data-science/the-evolution-of-tokenization-in-nlp-byte-pair-encoding-in-nlp-d7621b9c1186

Tokenization

- **World-level** tokenization problems
  - Similar words but with different IDs: “dog” and “dogs”
  - Huge vocabulary: predefined a limited vocabulary
  - Out-of-Vocabulary (OOV) : use [UNK] or <unk> as unknow token, losing information
- **Character-level** tokenization problems
  - Vocabulary is small: 256 different characters (letters, numbers, special characters)
  - Less meaningful problem: A character usually doesn’t carry any meaning or information as a word does.
  - Very large amount of tokens: one word will be split into 10 or more tokens
- **Subword-level** tokenization

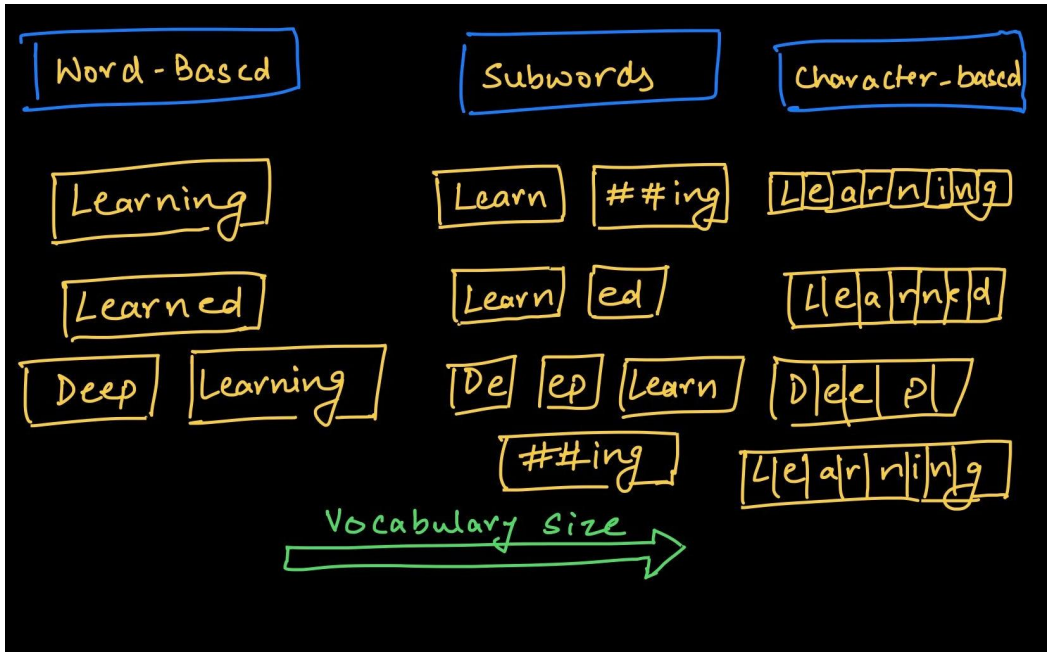


Image source: https://medium.com/data-science/the-evolution-of-tokenization-in-nlp-byte-pair-encoding-in-nlp-d7621b9c1186

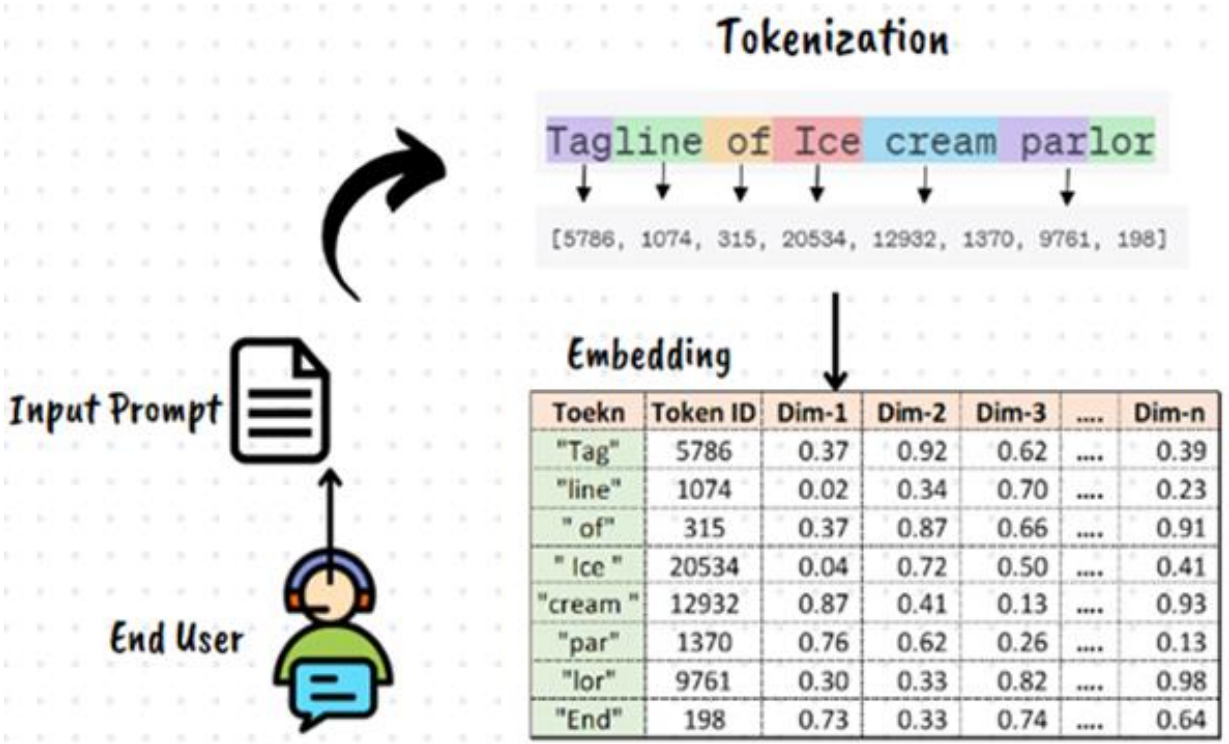
Subword tokenization

- Byte-Pair Encoding (BPE)
  - GPT-2, GPT-3, …
- WordPiece
  - BERT, DistilBERT, …
- SentencePiece
  - T5, XLNet, ALBERT, …

	Byte-level BPE	SentencePiece (unigram)	WordPiece
Lossless?*	• partial (non-GPT) • full (GPT)	• partial (e.g., English) • full (e.g., Chinese)	N
Pre-tokenization?	• Y (non-GPT) • N (GPT)	• Y (e.g., English) • N (e.g., Chinese)	Y
Base vocab	256	(Large!) all characters **and** most frequent substrings	Unicode characters in the training data
Rules	"Merge" by **frequency**	"Trim" by **minimizing likelihood loss**	"Merge" by **maximizing likelihood**
Used by	GPT, Roberta	T5, XLNet, Reformer	BERT, DistilBERT
Proposed in	(Sennrich et al., 2016). Initially proposed to translate **rare words** through **subword** units.	(Kudo & Richardson, 2018). Proposed to **sample** different tokenizations for the same string, which are used as "subword regularizers" for the model.	(Schuster & Nakajima, 2012). First subword tokenization model.

Embedding

- Tokens are **discrete** IDs
- How can models understand tokens?
- Token Embedding: transferring tokens to n-dimensional **continuous** vectors, which are numerical representations .
- Has meaning for models
- This layer is usually pre-defined



Toekn	Token ID	Dim-1	Dim-2	Dim-3	....	Dim-n
"Tag"	5786	0.37	0.92	0.62	....	0.39
"line"	1074	0.02	0.34	0.70	....	0.23
" of"	315	0.37	0.87	0.66	....	0.91
" Ice "	20534	0.04	0.72	0.50	....	0.41
"cream "	12932	0.87	0.41	0.13	....	0.93
"par"	1370	0.76	0.62	0.26	....	0.13
"lor"	9761	0.30	0.33	0.82	....	0.98
"End"	198	0.73	0.33	0.74	....	0.64

Image source: https://medium.com/epsilon-engineering-blog/tokenization-and-embedding-science-behind-large-language-model-3cafbaa78de1

Embeddings ≠ Final Features

- Embeddings are just the starting point.
- Embeddings capture **individual token meaning,** but not sentence structure or context.
- Example: "bank" in the following has **same embedding** → no context awareness!
  - "He sat by the river bank"
  - "She works at the bank"

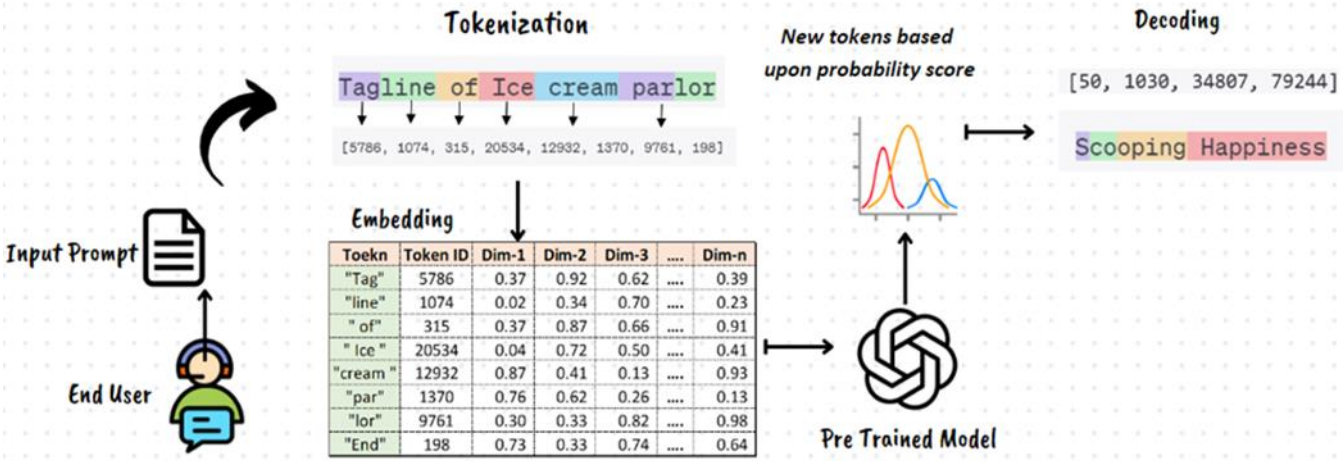
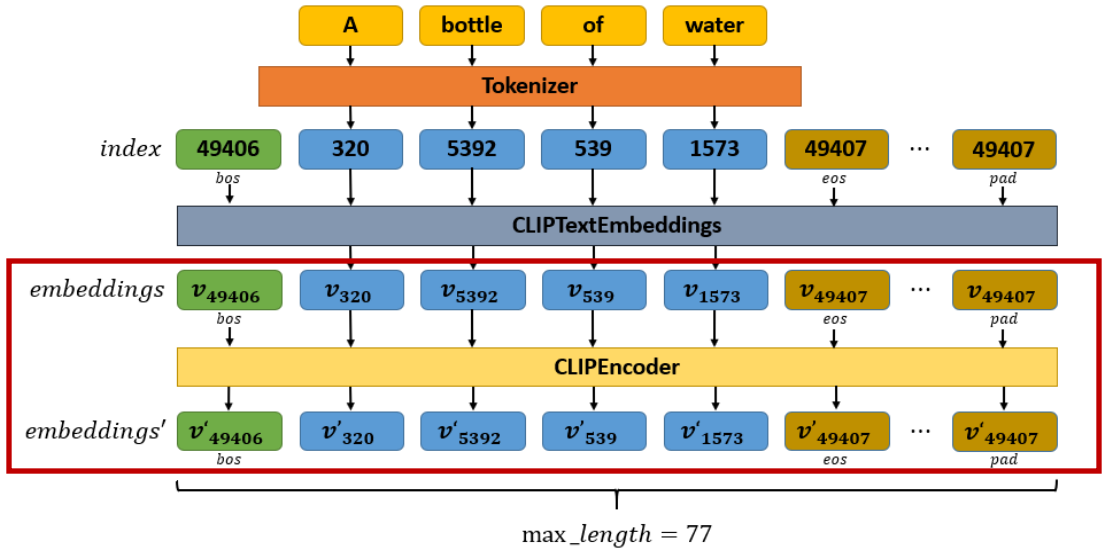


Image source: https://medium.com/epsilon-engineering-blog/tokenization-and-embedding-science-behind-large-language-model-3cafbaa78de1

Modeling

- A way to **model sequences**, **relationships**, and **context**
- That's where **RNNs, LSTMs, and Transformers** come in!
- **Embedding** = dictionary definition of each word
- **Mode/Architecture** = how the words form a meaningful sentence



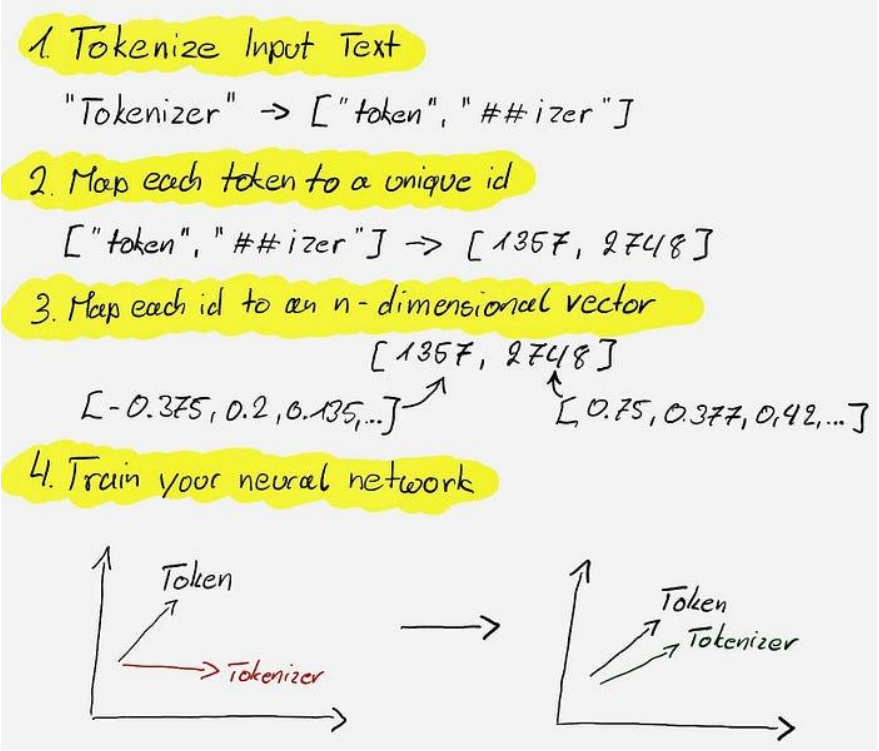


Image source: https://saschametzger.com/en/posts/what-are-tokens-vectors-and-embeddings-how-do-you-create-them

Recurrent Neural Network (RNN)

- Human language is **sequential**: the order of words matters.
- **RNNs were designed to process sequences** by having a "memory."
- **How it works (simplified):**
  - Processes input (tokens) **one by one**.
  - The output for a token is influenced by the current token **AND** the information (hidden state) remembered from previous tokens.
  - Think of it as reading a sentence word by word, trying to keep the context in mind.

```
x₁ → [RNN] → h₁
x₂ → [RNN] → h₂ (uses h₁)
...
```

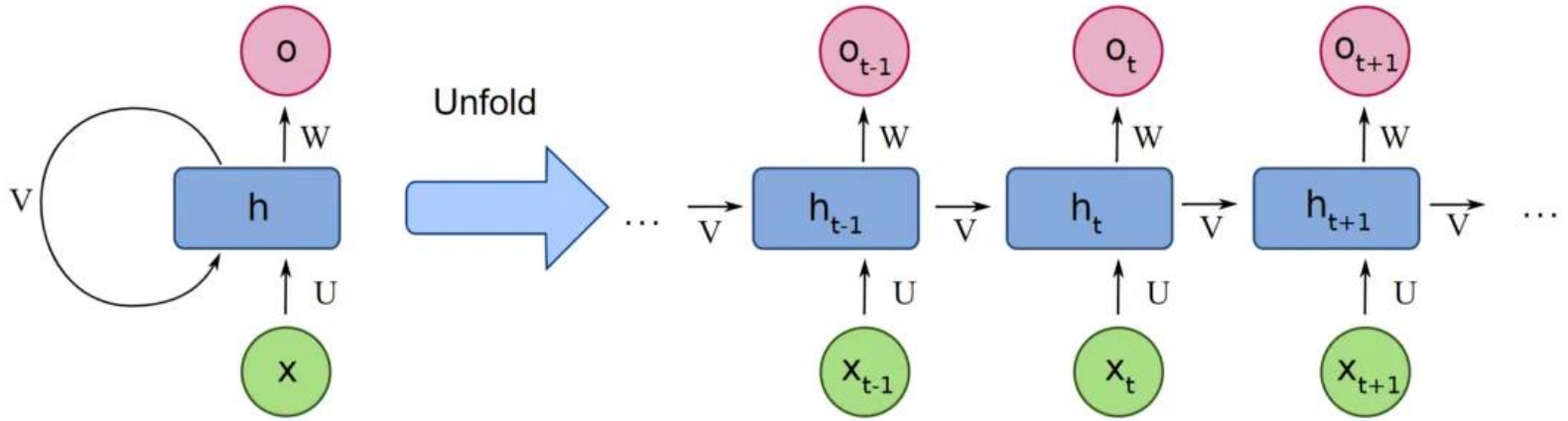
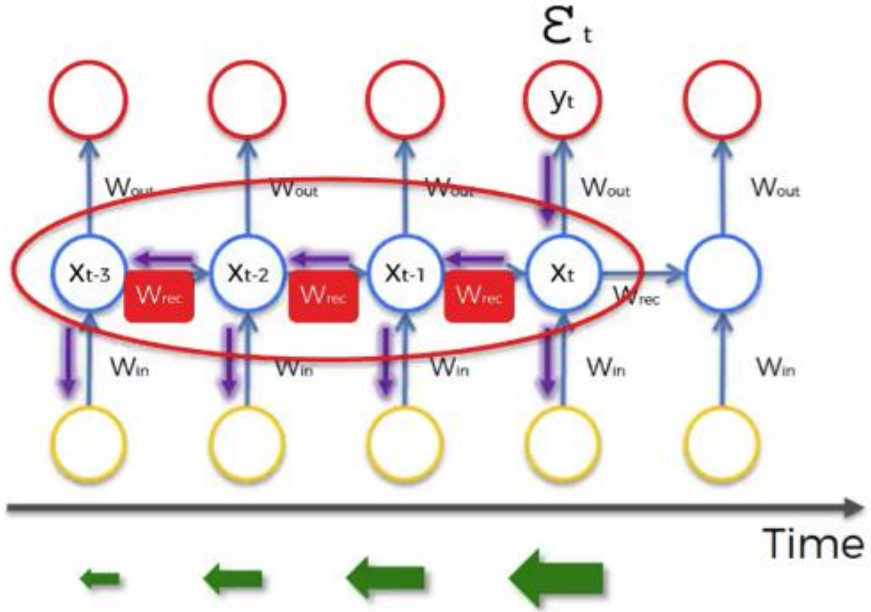


Image source: https://www.analyticsvidhya.com/blog/2022/03/a-brief-overview-of-recurrent-neural-networks-rnn/

Recurrent Neural Network (RNN)

- Limitation: The "Short-Term Memory" Problem.
  - Struggled to remember context over long sequences (vanishing/exploding gradient problem). Like someone forgetting the beginning of a very long sentence.
  - Processed sequentially, no parallelism during training, which can be slow.



$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \le t \le T} \frac{\partial \mathcal{E}_t}{\partial \theta} \quad (3)$$

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \right) \quad (4)$$

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \ge i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \ge i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1})) \quad (5)$$

W_{rec} ~ small ⇨ Vanishing

W_{rec} ~ large ⇨ Exploding

Formula Source: Razvan Pascanu et al. (2013)

Image source: https://www.analyticsvidhya.com/blog/2022/03/a-brief-overview-of-recurrent-neural-networks-rnn/
Razvan, Pascanu, On the difficulty of training recurrent neural networks. PMLR 2013

Recurrent Neural Network (RNN)

- Solving for vanishing or exploding gradients

  - Identity RNN with ReLU activation

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

-1 ⟶ 0

  - Gradient clipping

32 ⟶ 25

  - Skip connections

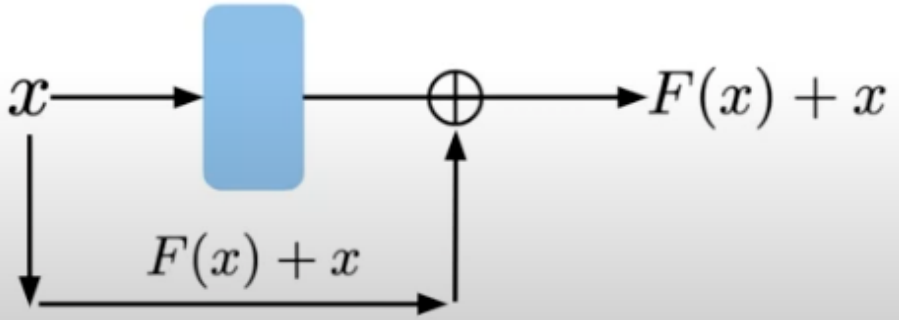


Image source: https://www.youtube.com/watch?v=NgxMUHTJYmU

Long Short-Term Memory (LSTM)

- **LSTMs are a special type of RNN**, solving the short-term memory problem.
- **Key Idea: "Gates"**
  - LSTMs have internal mechanisms called gates (input, forget, output gates).
  - These gates **control/choose** what information is kept, what is discarded, and what is added to the cell's memory.
  - This allows them to selectively remember or forget information over longer stretches of text.

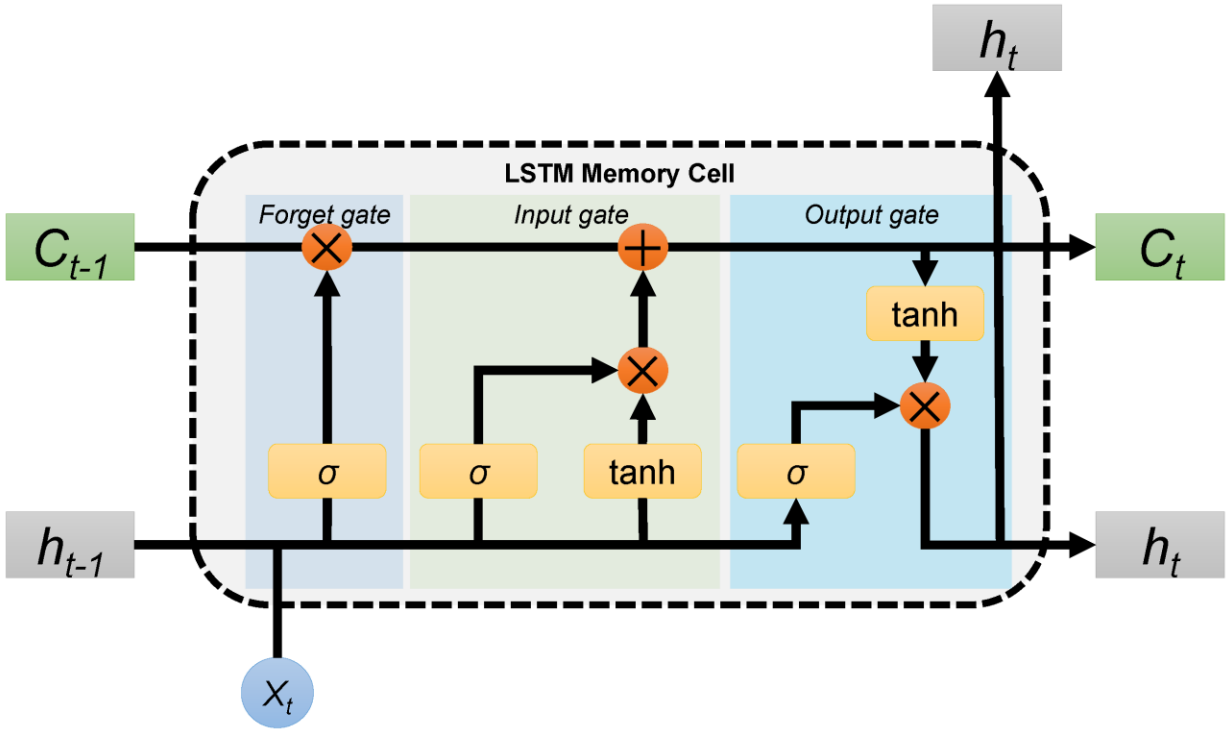


Image source: https://www.mdpi.com/2073-4441/12/1/175

Long Short-Term Memory (LSTM)

- **Forget Gate**: Decides what information to discard from the cell state.
- **Input Gate**: Determines which new information to add to the cell state.
- The **candidate** itself represents **what** the LSTM might add to the cell state.
- **Output Gate**: Controls the output based on the updated cell state.
- The **tanh** function ensures that the values of the candidate cell state range between -1 and 1.

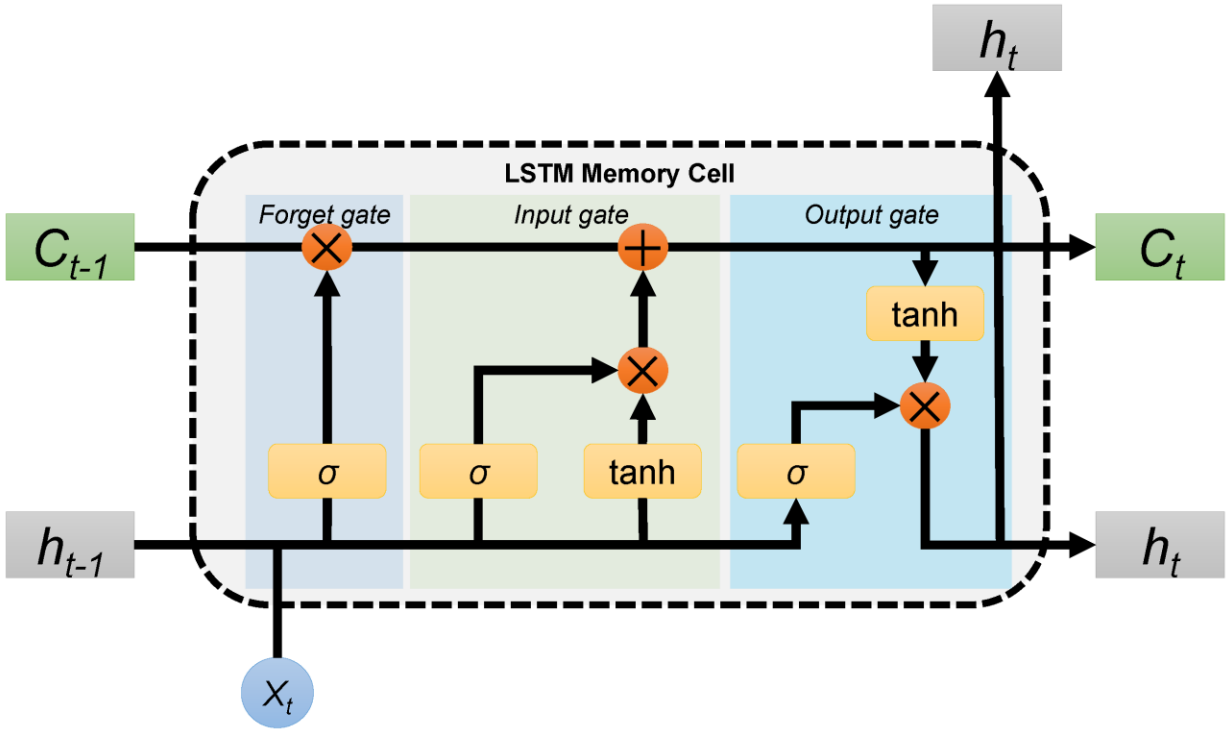


Image source: https://www.mdpi.com/2073-4441/12/1/175

Long Short-Term Memory (LSTM)

- **Better at:** Capturing "long-range dependencies" (e.g., connecting a pronoun to a noun much earlier in a paragraph).
- **Still:** Processes tokens sequentially, which can still be a bottleneck for **very long texts**.

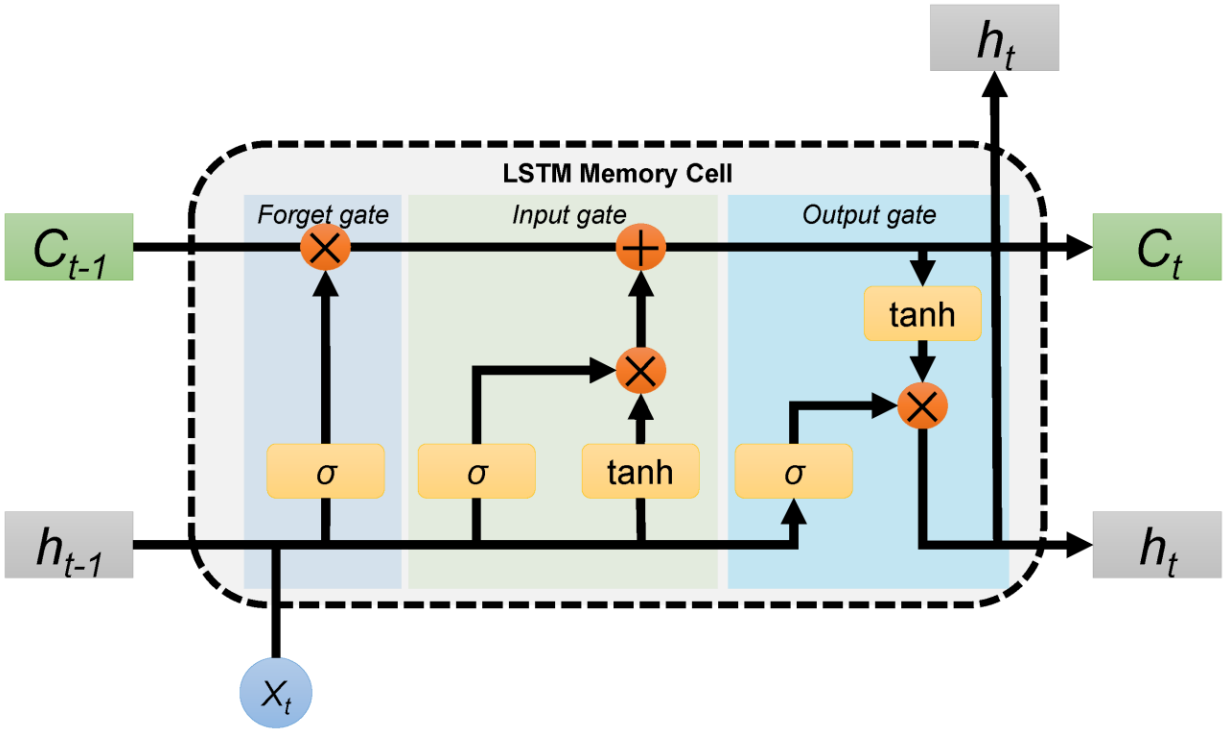
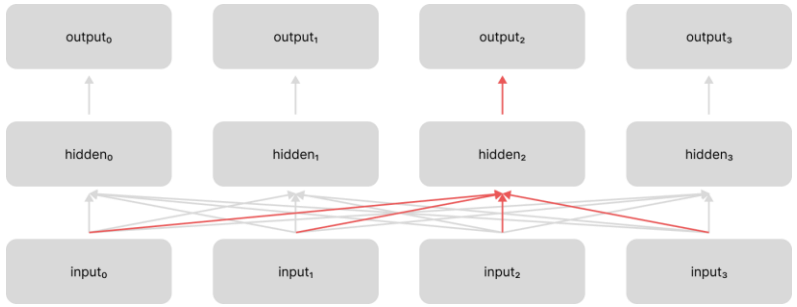


Image source: https://www.mdpi.com/2073-4441/12/1/175

Breakthrough: Transformer

- RNNs/LSTMs process tokens one by one
- **Transformers can process all tokens in a sequence simultaneously (parallel processing).**
- "**Attention Is All You Need**" (Google, 2017) – revolutionized the field.
- The key innovation that enables this parallel processing and deep understanding is **ATTENTION**.

```
x₁ → [RNN] → h₁
x₂ → [RNN] → h₂ (uses h₁)
...
```



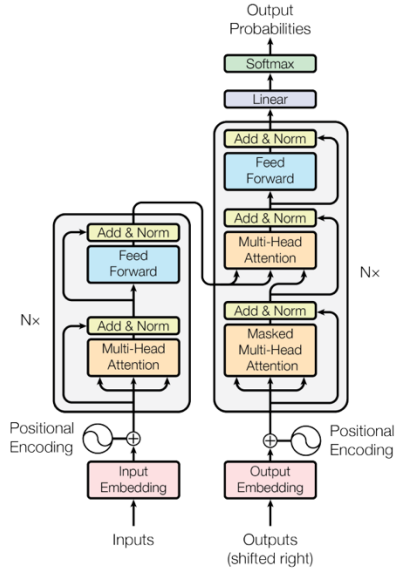


Figure 1: The Transformer - model architecture.

Key Mechanism: Attention

- Not recurrence, but **parallel**
- Allows the model to weigh the importance of ***all other tokens*** in the input when processing a specific token.
- It doesn't just look at the immediately preceding ones.

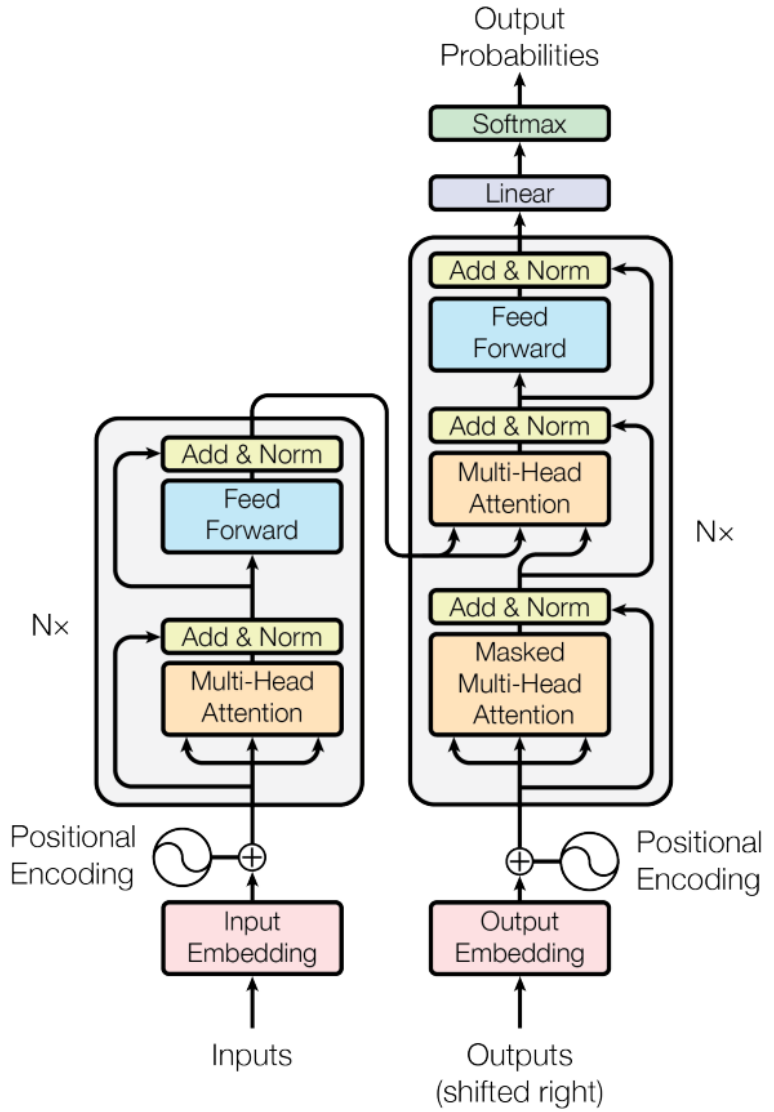


Figure 1: The Transformer - model architecture.

Image source: Ashish Vaswani, et al. Attention is all you need (Google Brain), 2017

Key Mechanism: Attention

- Each token creates three vectors: a **Query (Q)**, a **Key (K)**, and a **Value (V)**.
  - **Query:** "What am I looking for?"
  - **Key:** "What information do I hold?"
  - **Value:** "What information do I provide if relevant?"
- The model calculates "**attention scores**" by comparing the Query of one token with the Keys of all other tokens.
  - A high score means "this other token is very relevant to understanding me."
  - These scores then determine how much of each token's Value contributes to the representation of the current token.

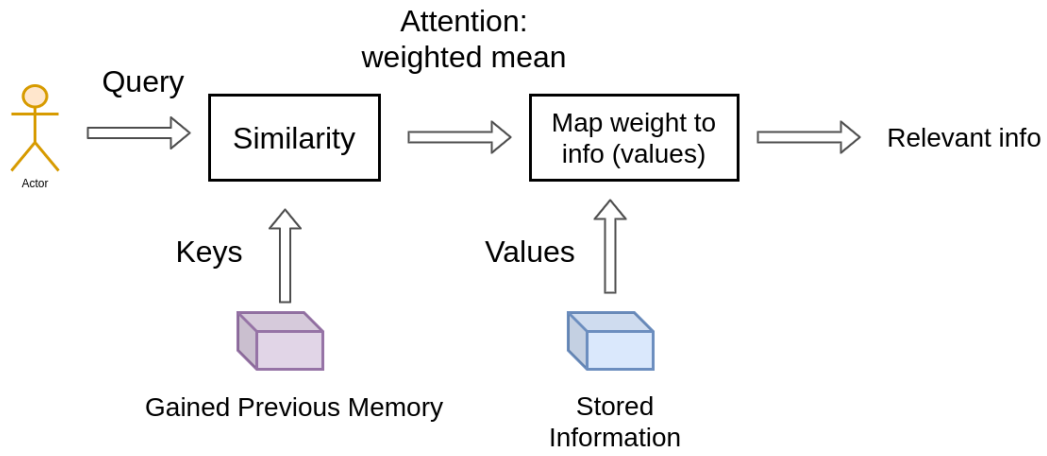


Image source: https://blog.sailor.plus/deep-learning/transformer-family/

Self-Attention

- Self-attention is an attention mechanism relating different positions of a **single sequence** in order to compute a representation of the sequence.
- Example: "The **animal** didn't cross the street because **it** was too tired."
  - When processing "**it**", self-attention helps the model assign a high score to "**animal**", understanding the link.

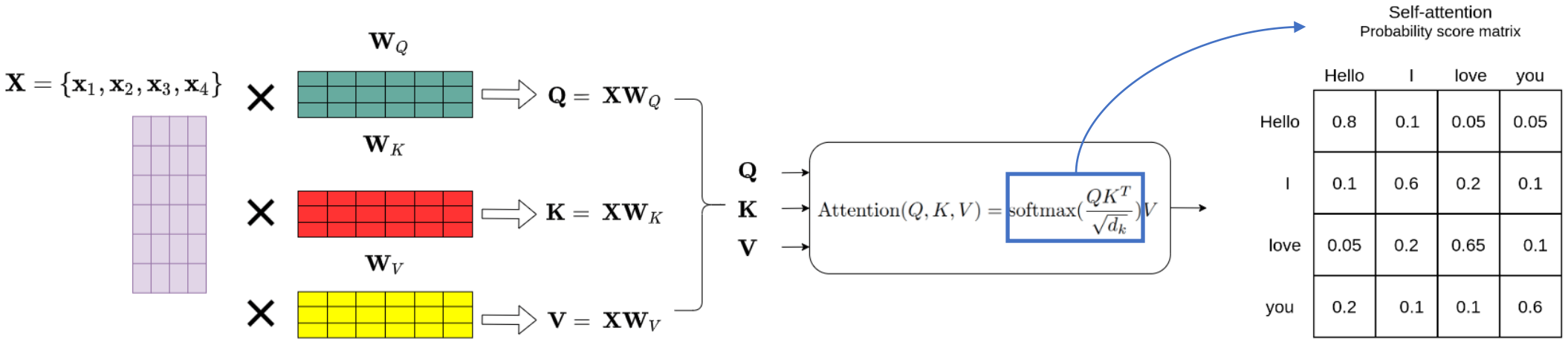
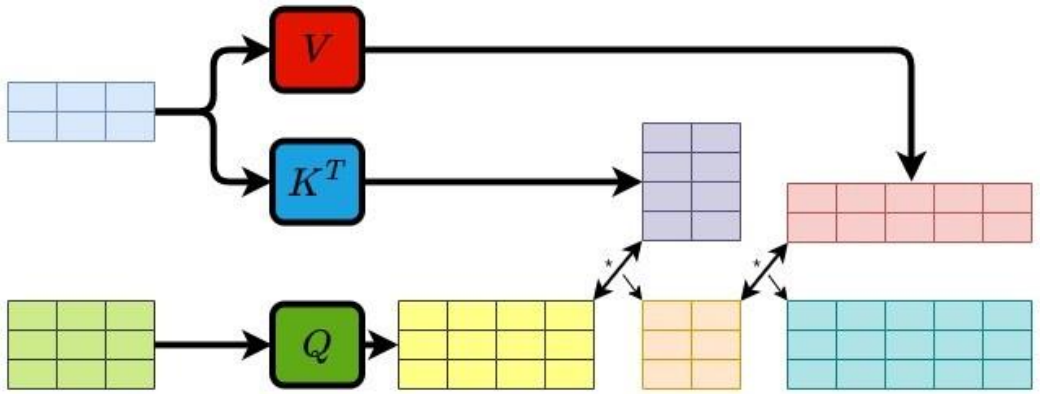


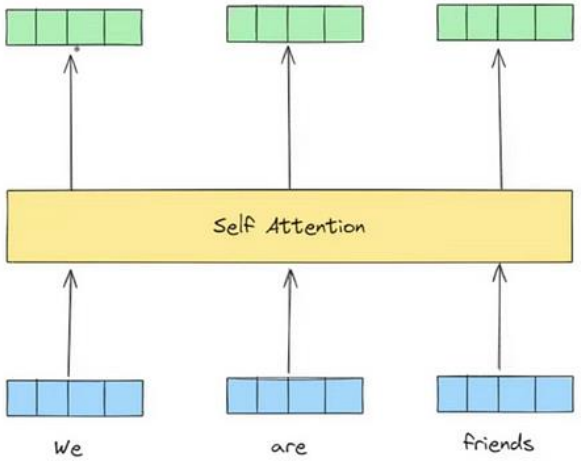
Image source: https://blog.sailor.plus/deep-learning/transformer-family/

Cross-Attention

- Identifies connections between **two sequences**, aligning words from different languages to help the model generate precise translations.



$$Cross\_attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{C/h}}) \cdot V$$



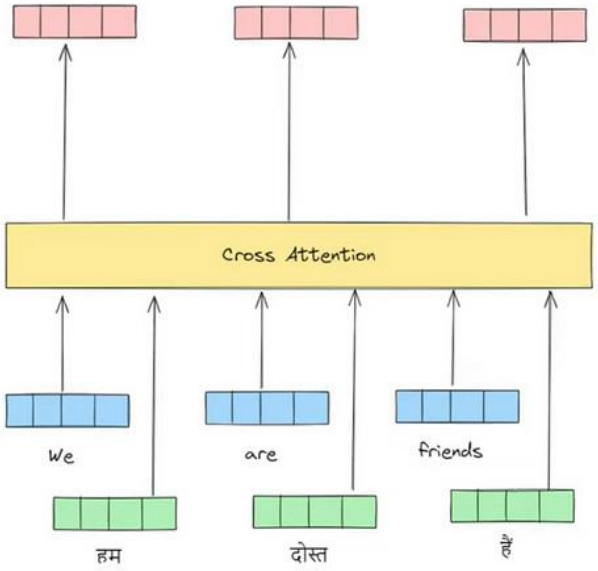
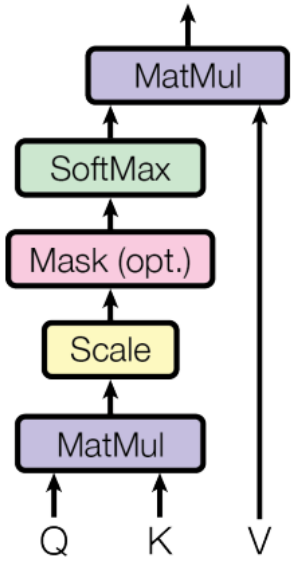


Image source: https://www.linkedin.com/pulse/decoding-attention-types-strategies-effective-nlp-models-swagat-panda-erhef/
https://medium.com/@sachinsoni600517/cross-attention-in-transformer-f37ce7129d78

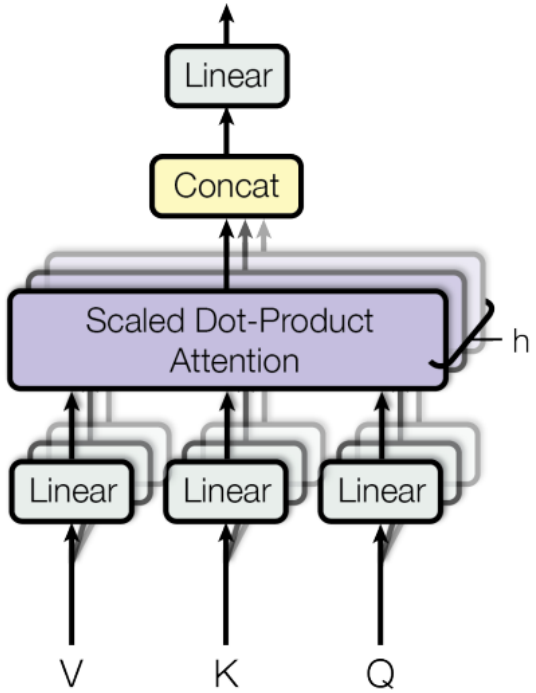
Multi-Head Attention

- Transformers use multiple attention mechanisms ("heads") in parallel.
- Each head can learn to focus on **different types of relationships** simultaneously (e.g., one head for grammatical relationships, another for semantic similarity).

Scaled Dot-Product Attention



Multi-Head Attention



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

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Image source: Ashish Vaswani, et al. Attention is all you need (Google Brain), 2017

Transformers

- Transformers use multiple layers of self-attention and cross-attention.

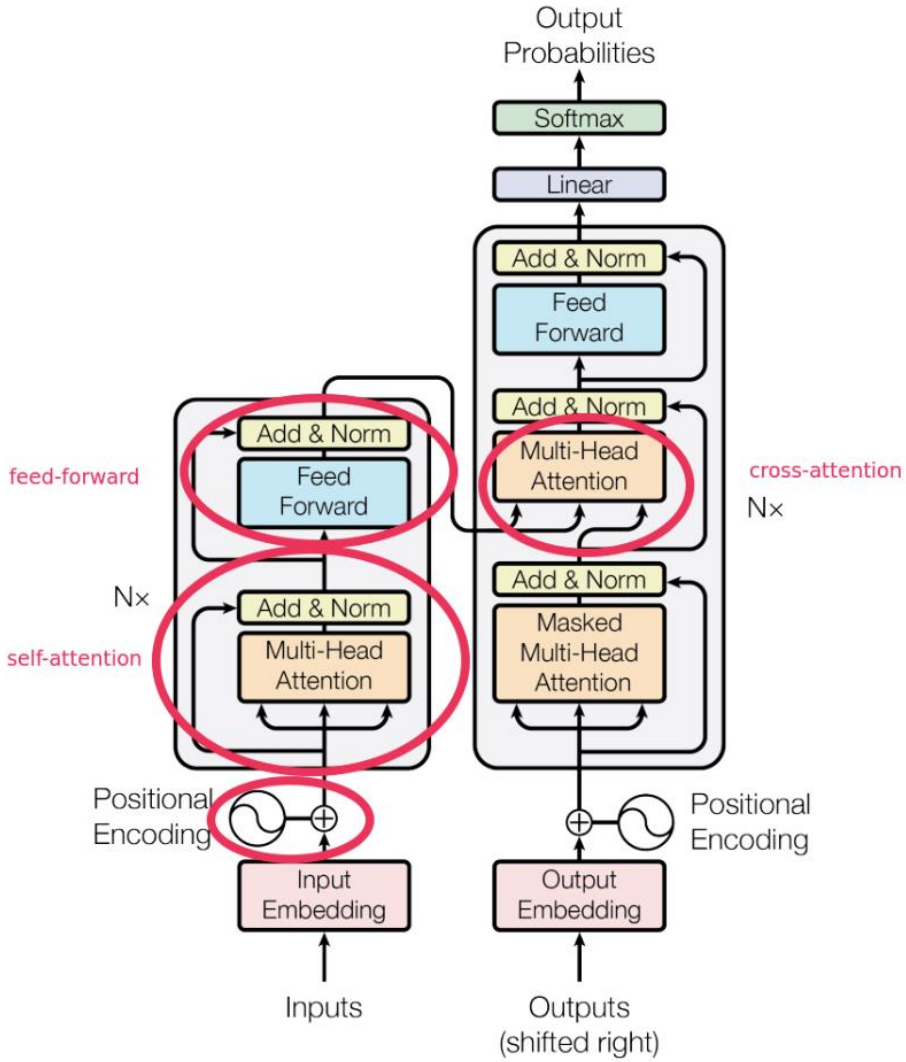


Image source: Ashish Vaswani, et al. Attention is all you need (Google Brain), 2017

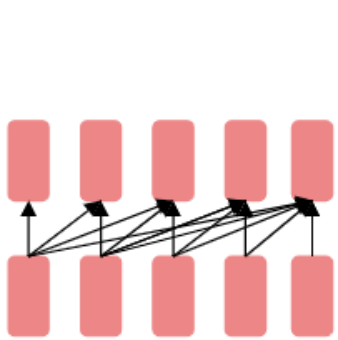
Transformer

how we apply the Transformer to machine translation

https://research.google/blog/transformer-a-novel-neural-network-architecture-for-language-understanding/

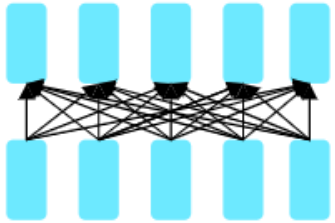
Transformer Architecture

- Encoder-Only Model
- Decoder-Only Model
- Encoder-Decoder Model



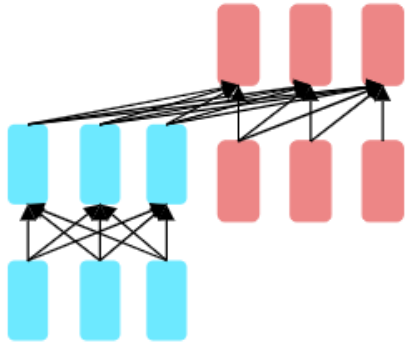
Decoders

GPT, Claude, Llama Mixtral



Encoders

BERT family, HuBERT

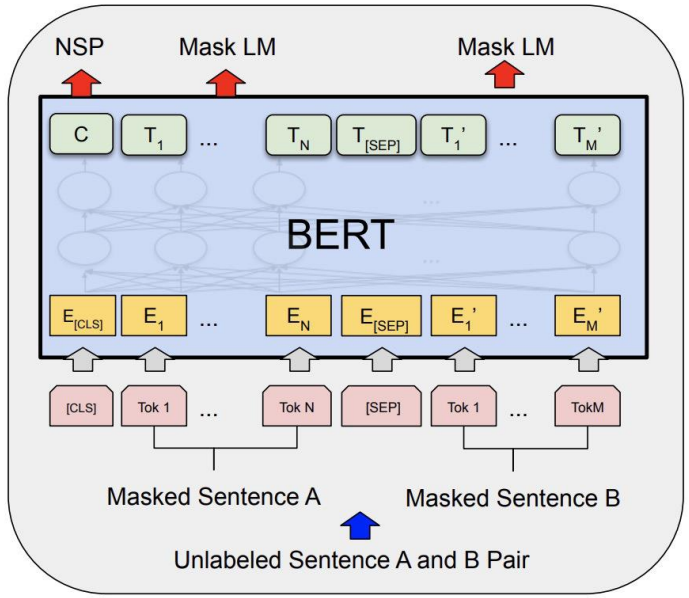


Encoder-decoders

Flan-T5, Whisper

Encoder-Only Model

- Popular: Masked Language Models (MLMs)
- BERT family: BERT, RoBERTa, ELECTRA
- Trained by predicting words from surrounding words on **both sides**
- Are usually finetuned (trained on supervised data) for classification tasks.



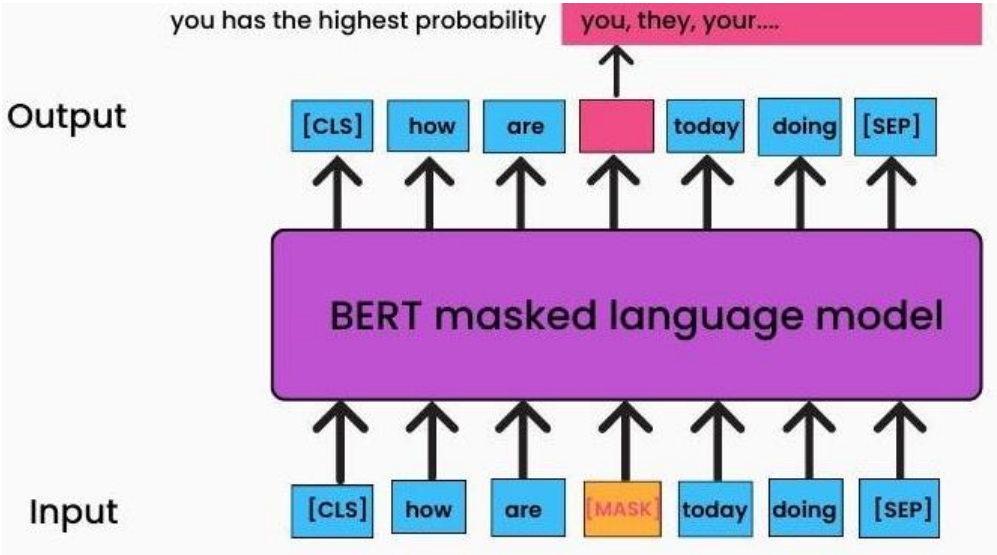


Image source: https://www.turing.com/kb/how-bert-nlp-optimization-model-works

Decoder-Only Model

- Also called:
  - Causal LLMs
  - Autoregressive LLMs
  - Left-to-right LLMs
- Predict words left to right (next-token)
- Example: GPT-x, Gemini, LLaMA
- Most of the really *large language models*

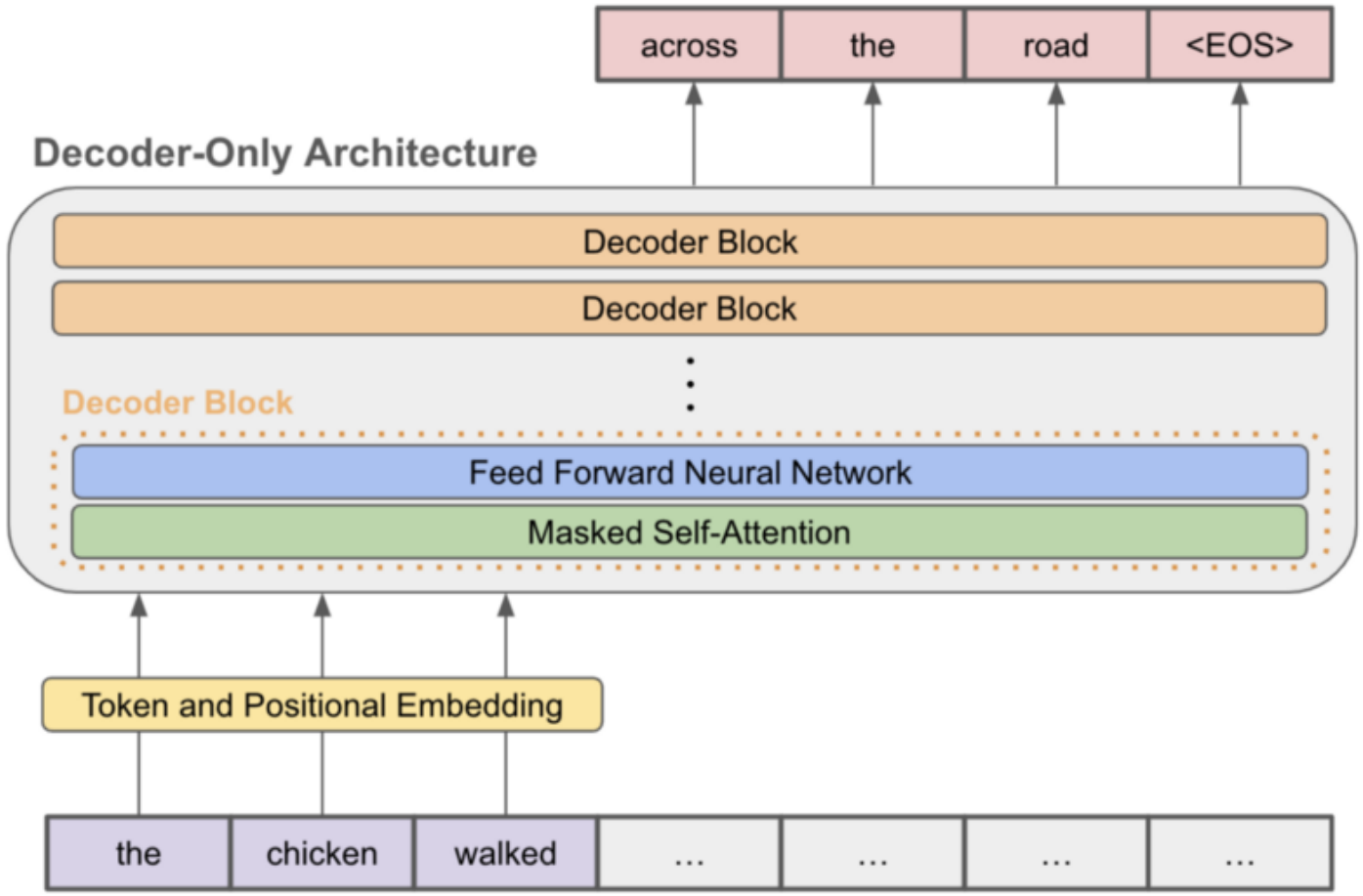
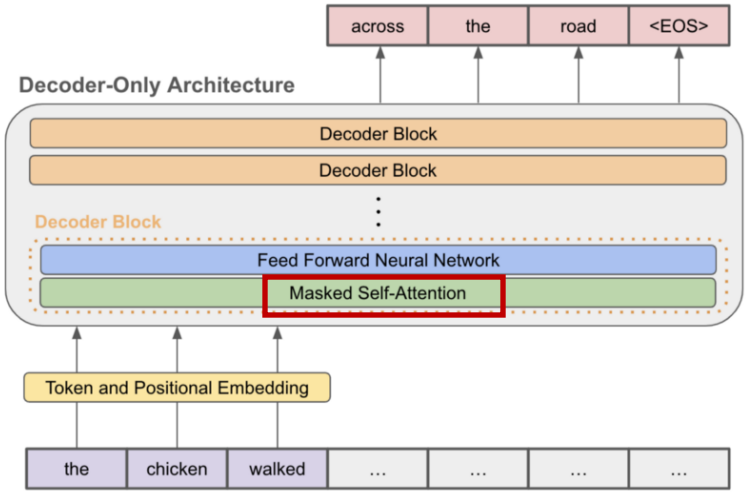


Image source: https://cameronrwolfe.substack.com/p/understanding-the-open-pre-trained-transformers-opt-library-193a29c14a15
Read: https://jalammar.github.io/illustrated-gpt2/

Decoder-Only Model

- Masked Self-Attention
- Stop the model from looking at the information we don't want it to look at



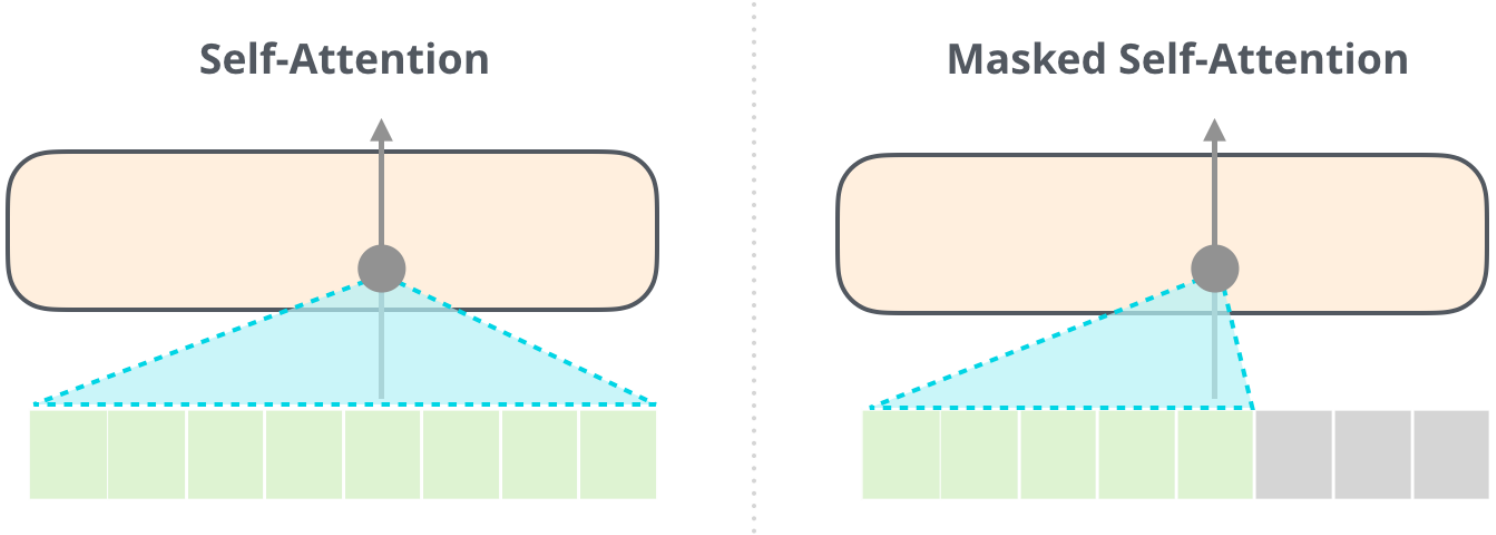
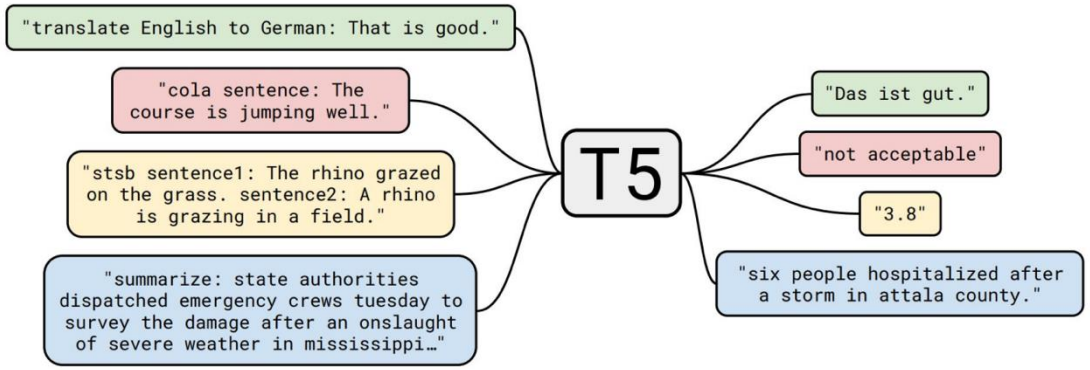
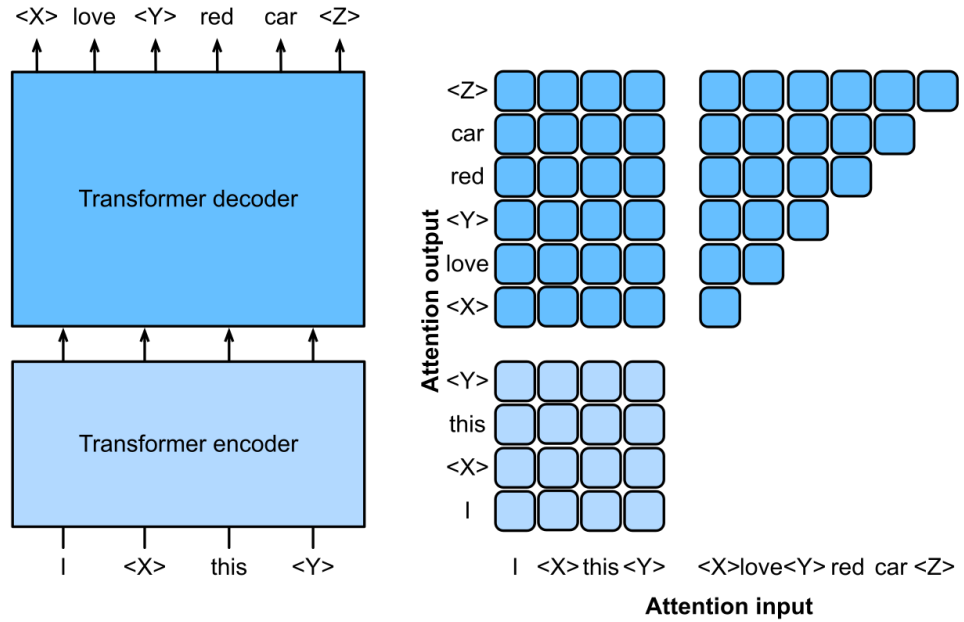


Image source: https://cameronrwolfe.substack.com/p/understanding-the-open-pre-trained-transformers-opt-library-193a29c14a15
Read: https://jalammar.github.io/illustrated-gpt2/

Encoder-Decoder Model

- Trained to map from one sequence to another
- The encoder processes the input text, and the decoder generates the output text.
- Very popular for:
  - machine translation (map from one language to another)
  - speech recognition (map from acoustics to words)
- Example: T5 (**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer)





Colin Raffel, et. al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, Google 2019

Question?

Is self-attention only used in the Encoder?

Is cross-attention only used in the Decoder?

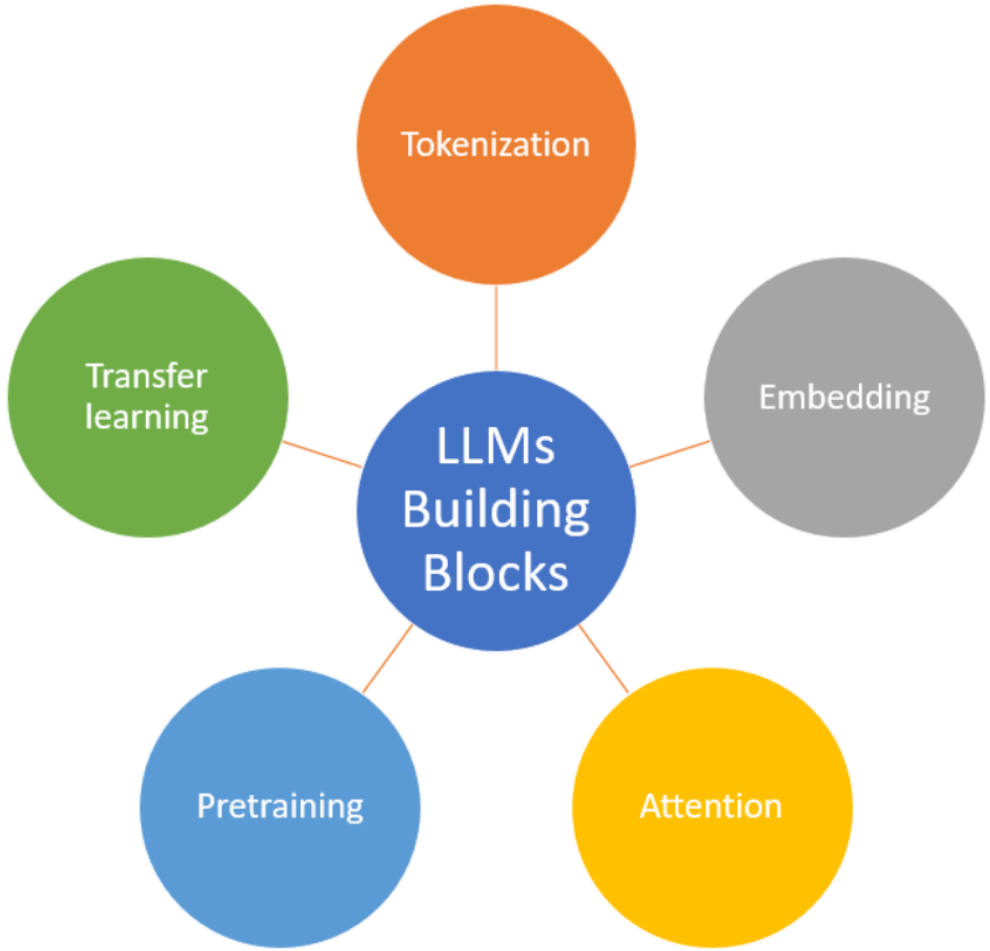
Self-attention is used in **both** the encoder and the decoder.

Cross-attention is used **only** in the decoder.

Mechanism	Encoder	Decoder	Purpose
Self-Attention	✅ Yes	✅ Yes (masked)	Allows each token to attend to others in the **same sequence**
Cross-Attention	❌ No	✅ Yes	Allows decoder tokens to attend to the **encoder's output**

Build an LLM

- Tokenization
- Embedding
- Model Structure:
  - RNN & LSTM
  - Transformer (Attention)
- Training
  - Pretraining
  - Fine-tuning
  - Prompting
  - RLHF



Training an LLM

- LLM Training = Pre-training + Fine-tuning (Optional)

[Massive Text Data] → Pretraining → [**General** LLM]

↓

+ Task-specific data → Fine-tuning → [**Specialized** LLM]

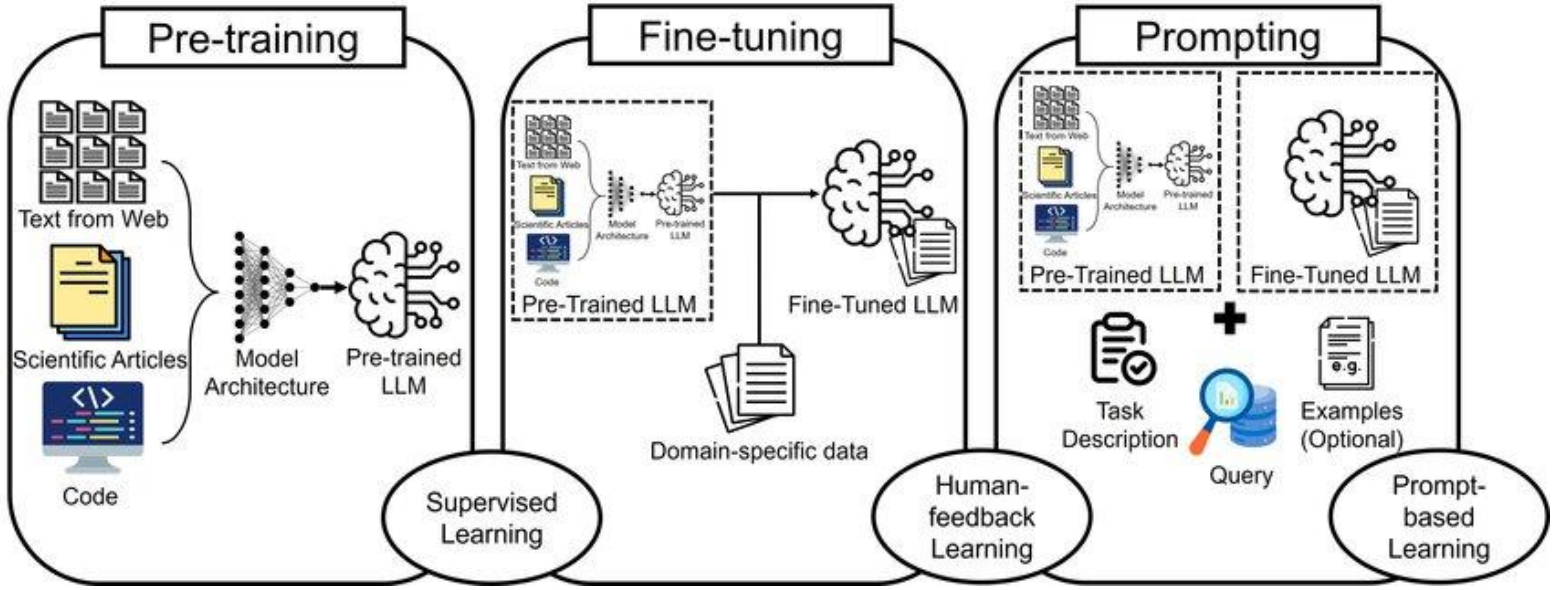


Image source: https://arxiv.org/pdf/2405.20585v1

Pre-training

- Train the model on **huge amounts of raw text** (e.g., books, websites, code).
- Objective: Learn **language patterns, syntax, facts, reasoning**.
- Common tasks:
  - Causal Language Modeling (CLM) → e.g., GPT
  - Masked Language Modeling (MLM) → e.g., BERT
- 📌 **Self-supervised learning** or **unsupervised learning**:
  - “Reading the internet without specific tasks.”

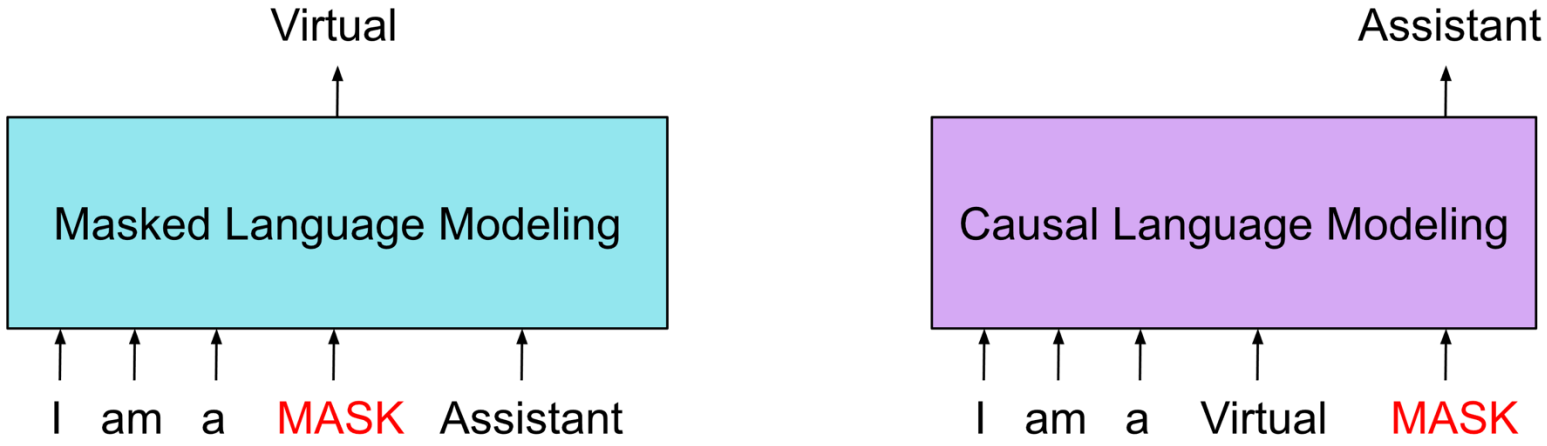
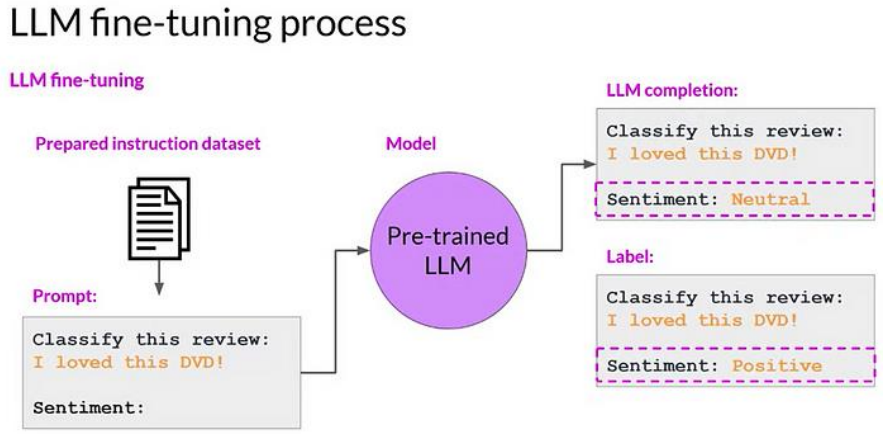


Image source: https://www.holisticai.com/blog/from-transformer-architecture-to-prompt-engineering

Fine-tuning

- Adapt the pretrained model to **specific tasks or domains**.
- Use **labeled data** or **instruction-style data**.
- Examples:
  - Sentiment classification
  - Legal/medical document analysis
  - Chatbots (e.g., ChatGPT-style)
- 📌 **Supervised** or **Reinforcement Learning**



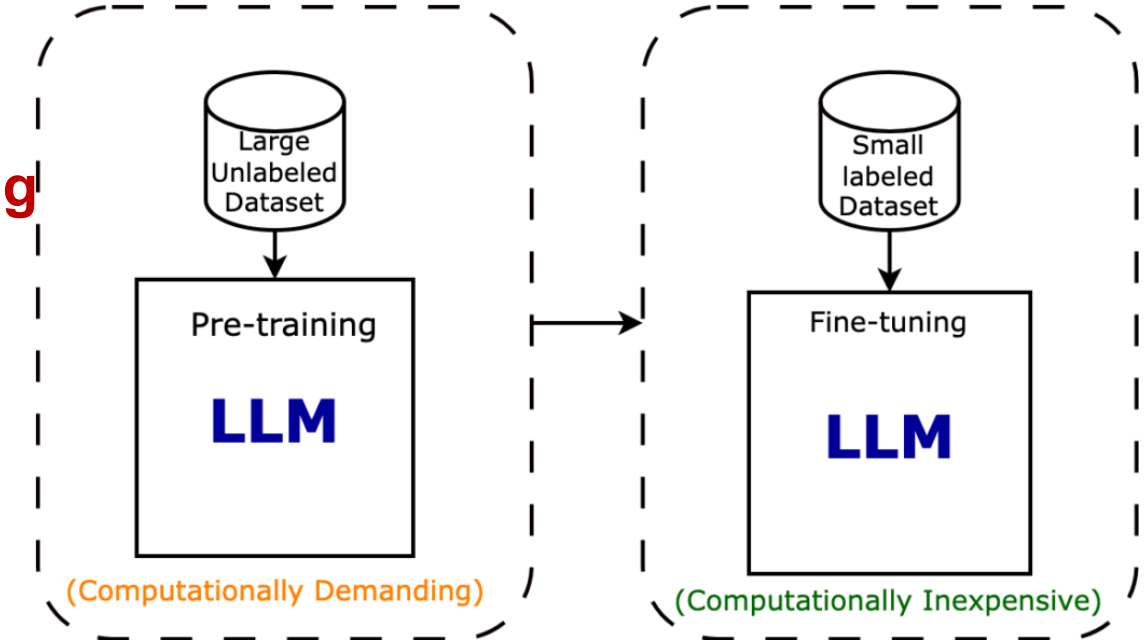


Image source: https://www.holisticai.com/blog/from-transformer-architecture-to-prompt-engineering
https://www.coursera.org/learn/generative-ai-with-llms

Prompting

- Given a prompt, an LLM responds incrementally with “tokens” (groups of letters, numbers, punctuation etc.) that it thinks are the best way to complete the prompt.

```
Translate English to French: cheese =>
```

The LLM would ideally calculate that “fromage” is the best completion.

- *N*-shot prompts provide N examples of the desired type of completion. (Few-shot just means N >= 2.)

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1 Translate English to French:      ← task description
2 cheese =>                         ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1 Translate English to French:      ← task description
2 sea otter => loutre de mer        ← example
3 cheese =>                         ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1 Translate English to French:      ← task description
2 sea otter => loutre de mer        ← examples
3 peppermint => menthe poivrée      ←
4 plush girafe => girafe peluche    ←
5 cheese =>                         ← prompt
```

Image source: Tom B. Brown, et al. Language Models are Few-Shot Learners. NeurIPS 2020.

Chain-of-thought (CoT) prompting

- Similar to few-shot prompting
- Reasoning: The completions contain **detailed** thought processes, which are particularly relevant for tasks like solving arithmetic problems.

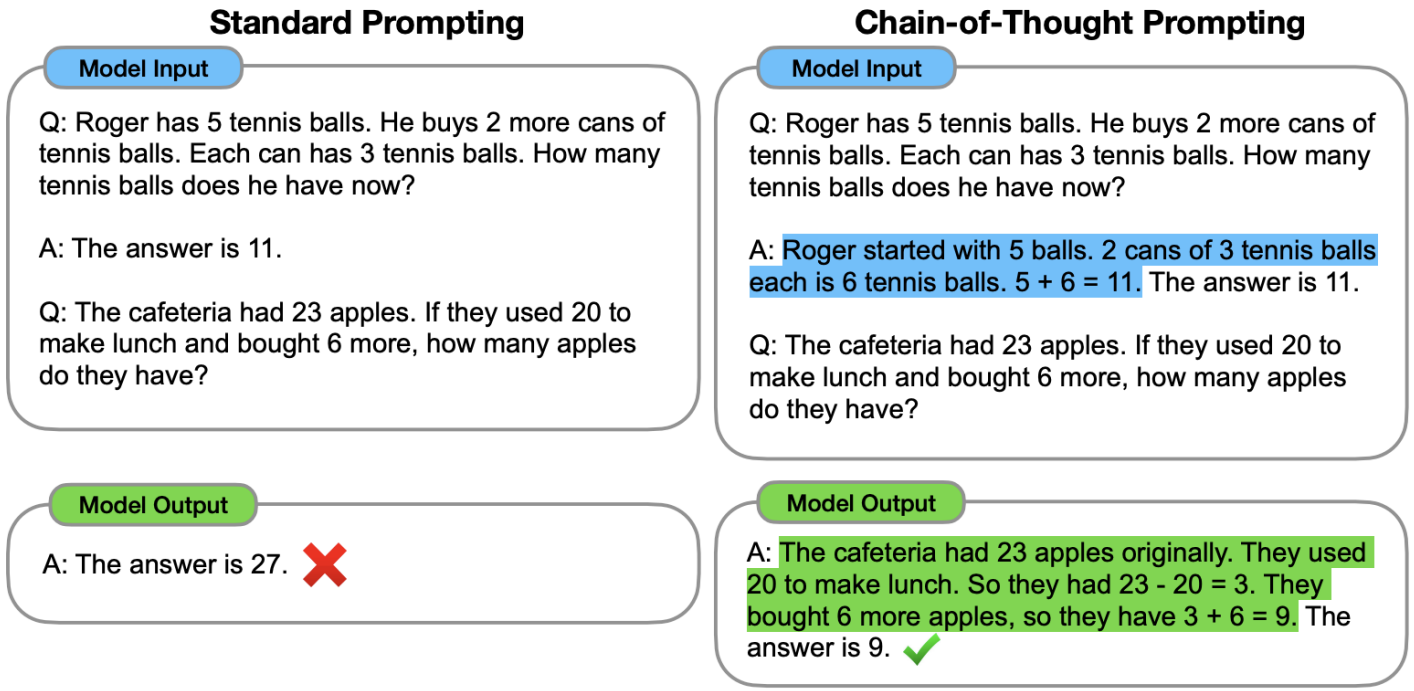
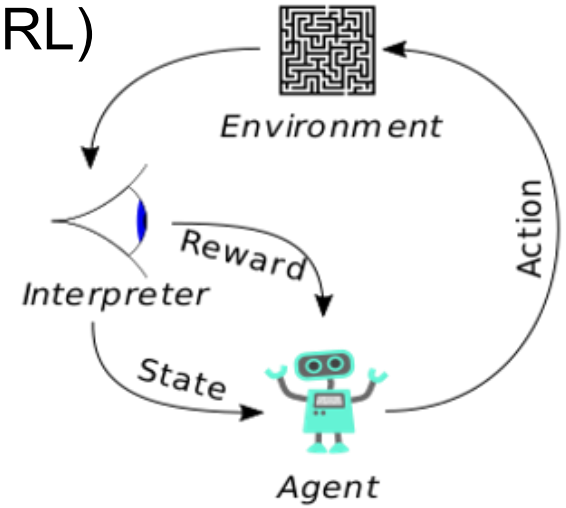


Image source: Wei, Jason, et al. "Chain-of-thought prompting elicits reasoning in large language models.", NeurIPS 2022

Reinforcement learning human feedback (RLHF)

- Recap: Reinforcement learning (RL)



  - State: observation of the current environment
  - Action: reaction of an agent
  - Reward: returns from the environment
  - Objective: **maximize cumulative reward**
  - Similar with human: observing by eyes ---> judging by brain ---> taking actions by hands

Reinforcement learning human feedback (RLHF)

- Agent: LLMs
- State: Current context
- Action: Next token, depends on the prompt text in the context and the probability distribution over the vocabulary space.
- Reward: is assigned based on how closely the completions align with human preferences
  - human expert
  - additional model called reward model to classify outputs of the LLM.

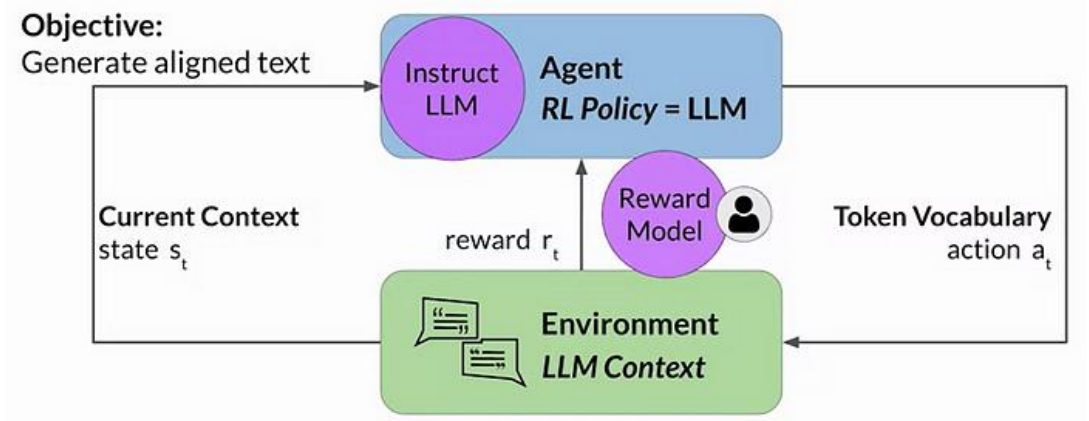


Image source: https://medium.com/@aydinKerem/everything-to-learn-about-large-language-models-part-2-3-pre-training-and-fine-tuning-5fc68700701a

What LLMs Can Do?

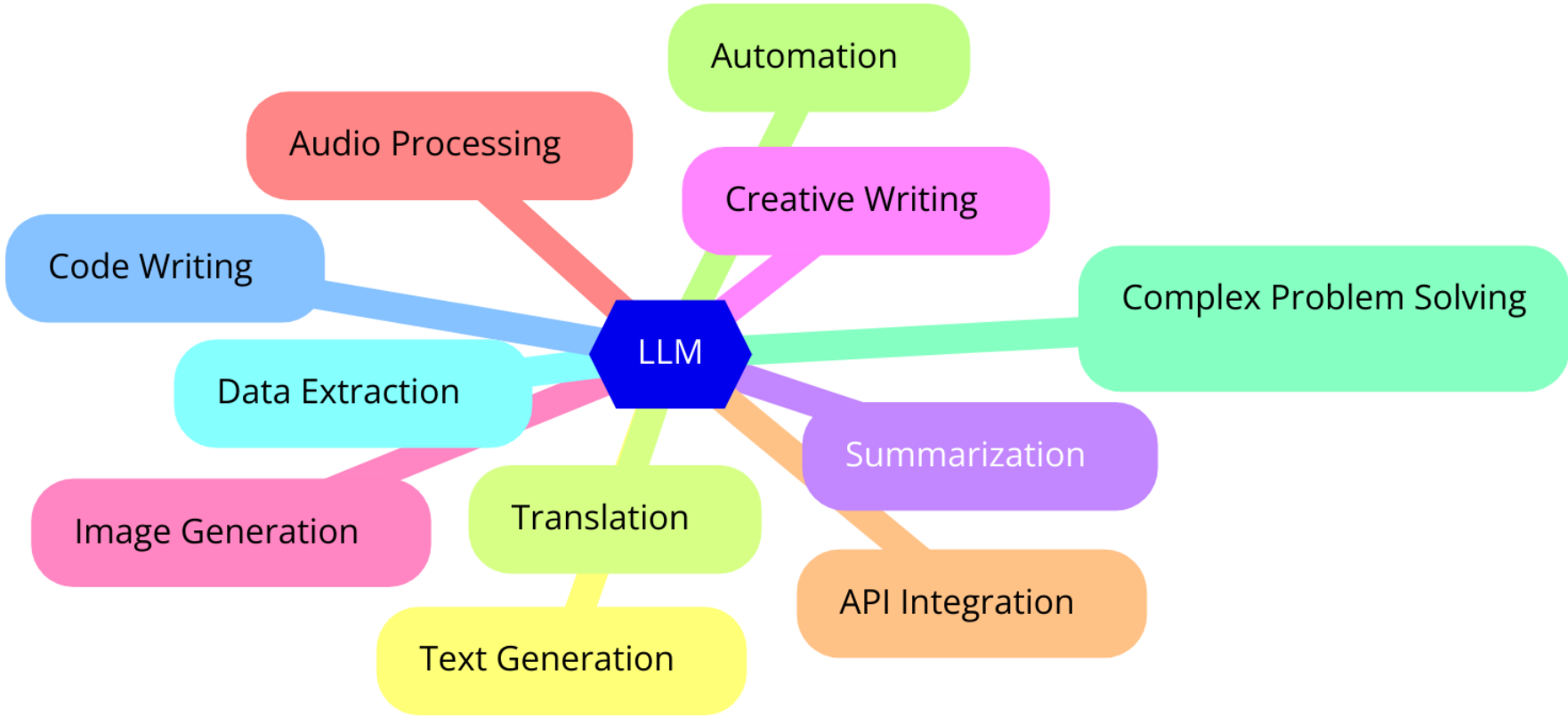


Image source: https://medium.com/@aydinKerem/everything-to-learn-about-large-language-models-part-2-3-pre-training-and-fine-tuning-5fc68700701a

Multimodal Large Language Models (MLLMs)

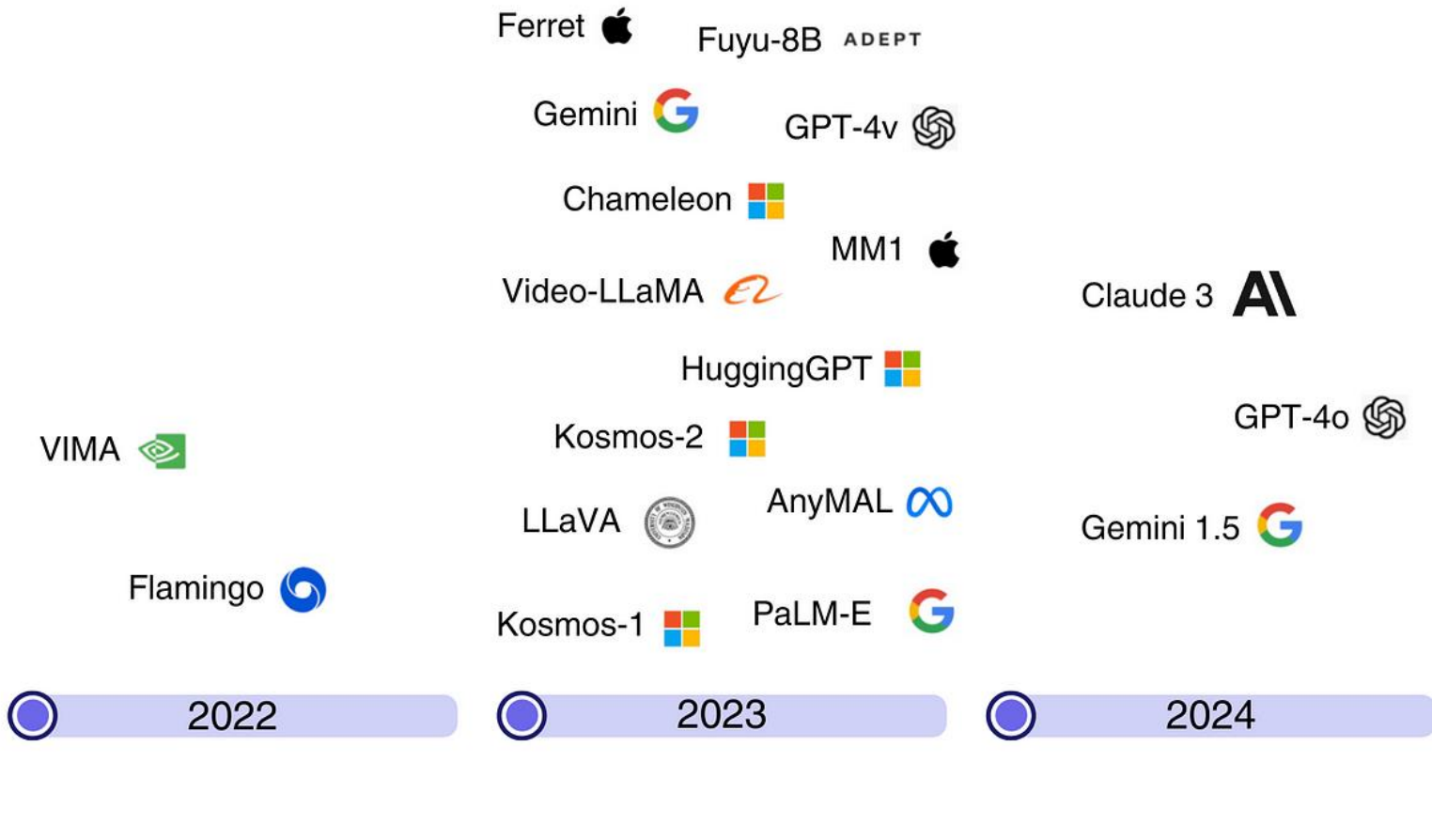


Image source: https://medium.com/@aydinKerem/everything-to-learn-about-large-language-models-part-2-3-pre-training-and-fine-tuning-5fc68700701a

The Limitations of LLMs

- Computational constraints – LLMs can't process everything at once
- Hallucinations – sometimes LLMs make stuff up
- Limited knowledge – LLMs can't update it's knowledgebase
- Lack of long-term memory and learning
- Limited reasoning – LLMs struggle complex multistep problems
- Inconsistency – LLMs can contradict themselves
- Lack of true understanding – LLMs don't really "get" subtext
- Difficulty with certain linguistic elements
- Bias and stereotyping – LLMs can perpetuate prejudices

Developer	Model	Context length
OpenAI	GPT-3.5 Turbo	16k tokens
OpenAI	GPT-4 Turbo	128k tokens
Anthropic	Claude 3 Haiku	200k tokens
Anthropic	Claude 3 Sonnet	200k tokens
Anthropic	Claude 3 Opus	200k tokens
Google	Gemini Pro	128k tokens
Google	Gemini 1.5	128k or 1m tokens

Developer	Model	Training data cut-off
OpenAI	GPT-3.5 Turbo	Sep 2021
OpenAI	GPT-4 Turbo	Dec 2023
Anthropic	Claude 3 Haiku	Aug 2023
Anthropic	Claude 3 Sonnet	Aug 2023
Anthropic	Claude 3 Opus	Aug 2023
Google	Gemini Pro	Early 2023
Google	Gemini 1.5	Early 2023

Image source: https://promptdrive.ai/llm-limitations/

Conclusion

- Introduction of LLMs
  - What
  - History
- How do LLMs think?
  - Tokenization; Embedding;
  - Model Structure: RNN, LSTM, Transformer
  - Training: Pre-training; Fine-tuning; Prompting; Reinforce Learning with Human Feedback (RLHF)
- What LLMs can do
  - MLLMs
  - Limitations

CRICOS PROVIDER 00123M

Thank you!

Email: xinyu.zhang02@adelaide.edu.au
Office: LG.03.C I AIML Building | Lot Fourteen