Generative AI & Large Language Models

(LLMs)

GenAI

The Creative Shift: Traditional AI is a **Critic** (it identifies things); Generative AI is an **Artist** (it creates things from scratch).

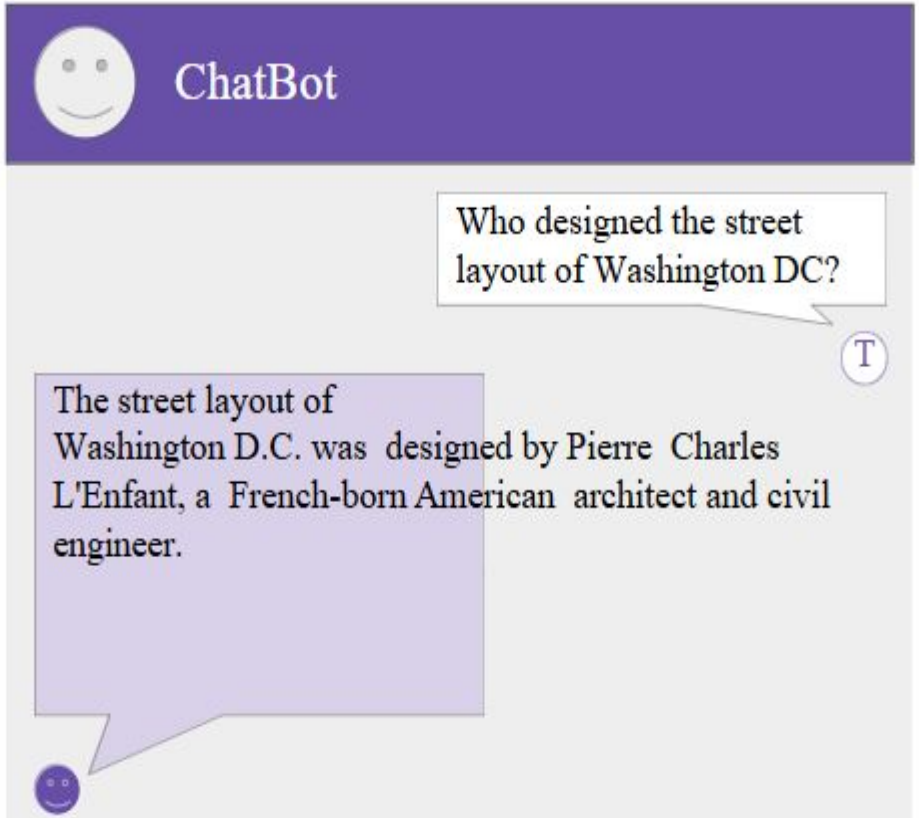
The Prediction Game: It doesn’t "know" facts like a person. It’s a super-powered calculator that predicts the **next most likely word** in a sentence.

The Power Source: It combines three ingredients: **Massive Knowledge** (the internet), **Raw Power** (fast computers), and **Context** (understanding how words relate).

A Universal Tool: One single AI can be your coder, your writer, and your brainstormer all at once—no special reprogramming needed.

Conversational: It changes computers from "tools we command" into "partners we talk to" using everyday language.

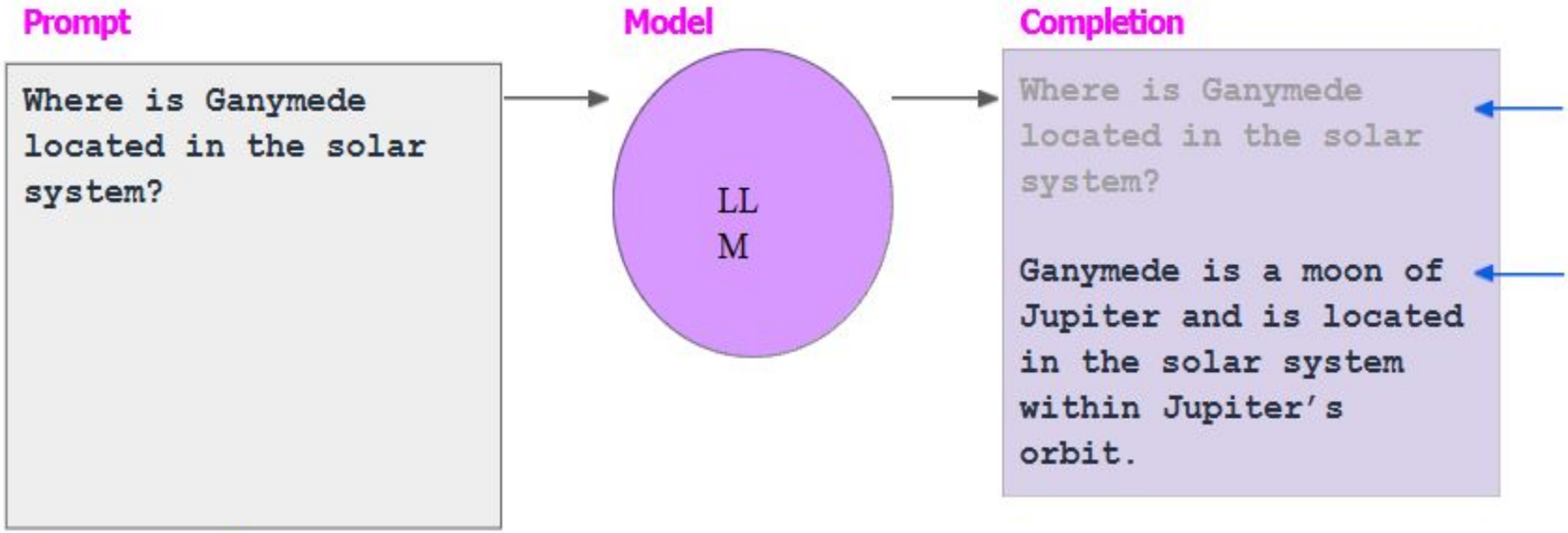
Generative AI



LLM

- **Closed vs. Open Models: Proprietary:** Like a "Secret Recipe" (e.g., GPT-4, Claude); you pay for access but cannot see the code.
- **Open-Source:** Like a "Community Cookbook" (e.g., Llama, Mistral); free to download, modify, and run privately.
- **The Training Journey: Base Models** are like students who read the whole library but don't follow orders.
- **Chat Models** are "Fine-tuned" assistants that have gone to school to learn how to answer questions and follow rules.
- **Reasoning Models ("The Thinkers"):** * Newer models (like OpenAI o1 or DeepSeek R1) are trained to "think before they speak." They solve problems step-by-step, making them masters of complex math and coding.
- **Size & Speed: Large Models** are massive, slow, and expensive brains for complex logic.
- **Small Models** are "pocket brains" that are fast, cheap, and can even run offline on your phone or laptop.

Prompts and completions



Context window

- typically a few 1000 words.

Transformer Architecture

Reading All at Once: Unlike older AI that read one word at a time (like a human), the Transformer reads **entire paragraphs instantly**. This allows it to understand how a word at the beginning of a page relates to one at the very end.

The "Attention" Highlighter: This is the AI's most important trick. It "highlights" the most relevant words in a sentence to get the meaning right.

- *Example:* In the sentence *"The **bank** of the river,"* the AI pays "attention" to **river** so it knows we aren't talking about a money bank.

The "Lego" Duo (Encoder & Decoder):

- **The Encoder (The Reader):** Digests and "understands" your input.
- **The Decoder (The Writer):** Takes that understanding and "writes" the response, one word at a time.

Massive Speed: Because it processes everything in parallel (all at once), we can train it on the entire internet using thousands of computers. This is why AI suddenly got so much smarter after 2017.

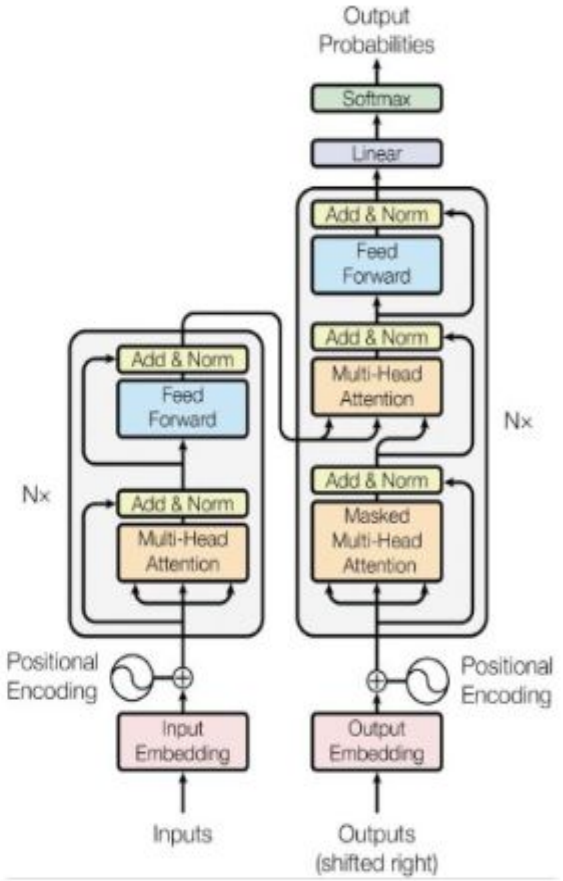
Transformers

Attention Is All You Need

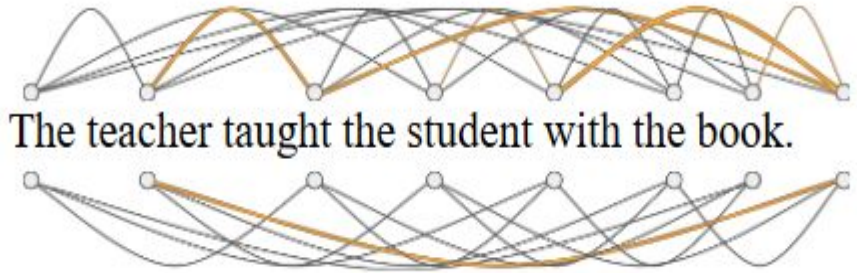
Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez*† University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin*‡ illia.polosukhin@gmail.com

Abstract

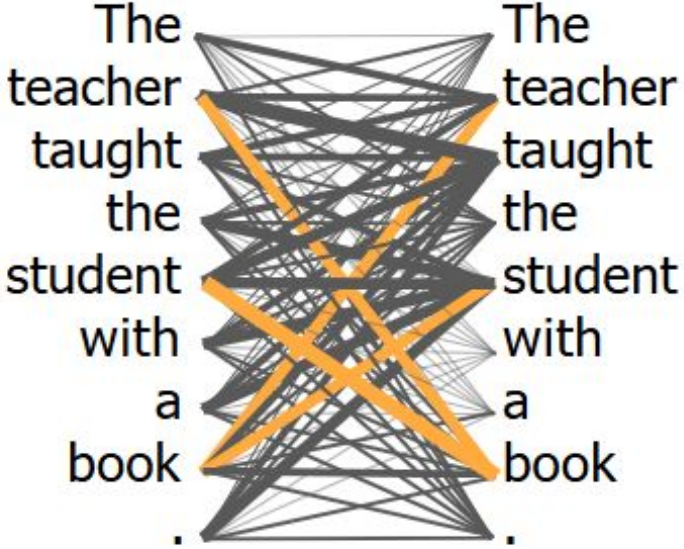
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to



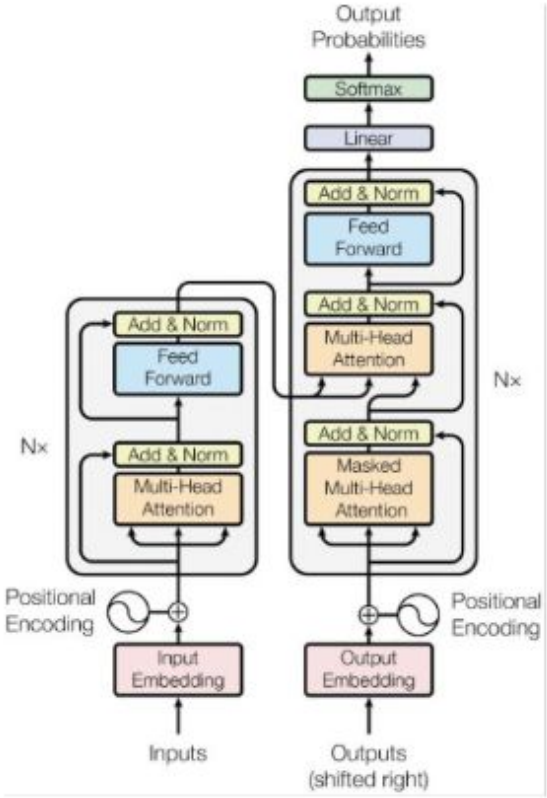
Transformers



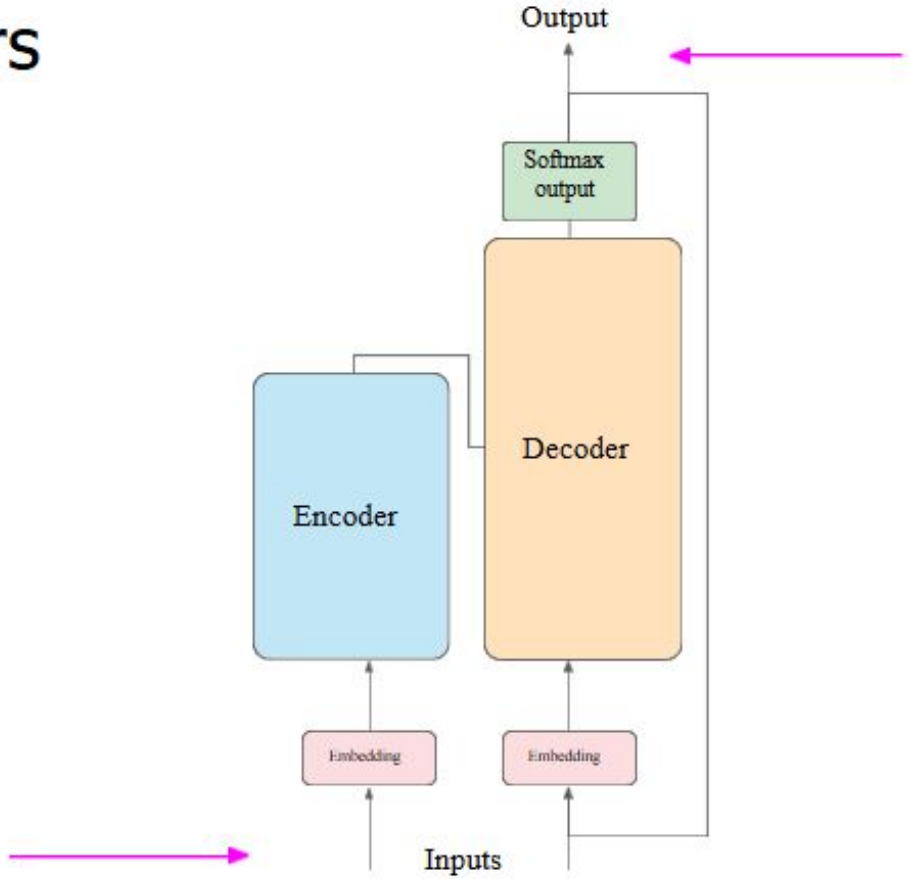
Self-attention



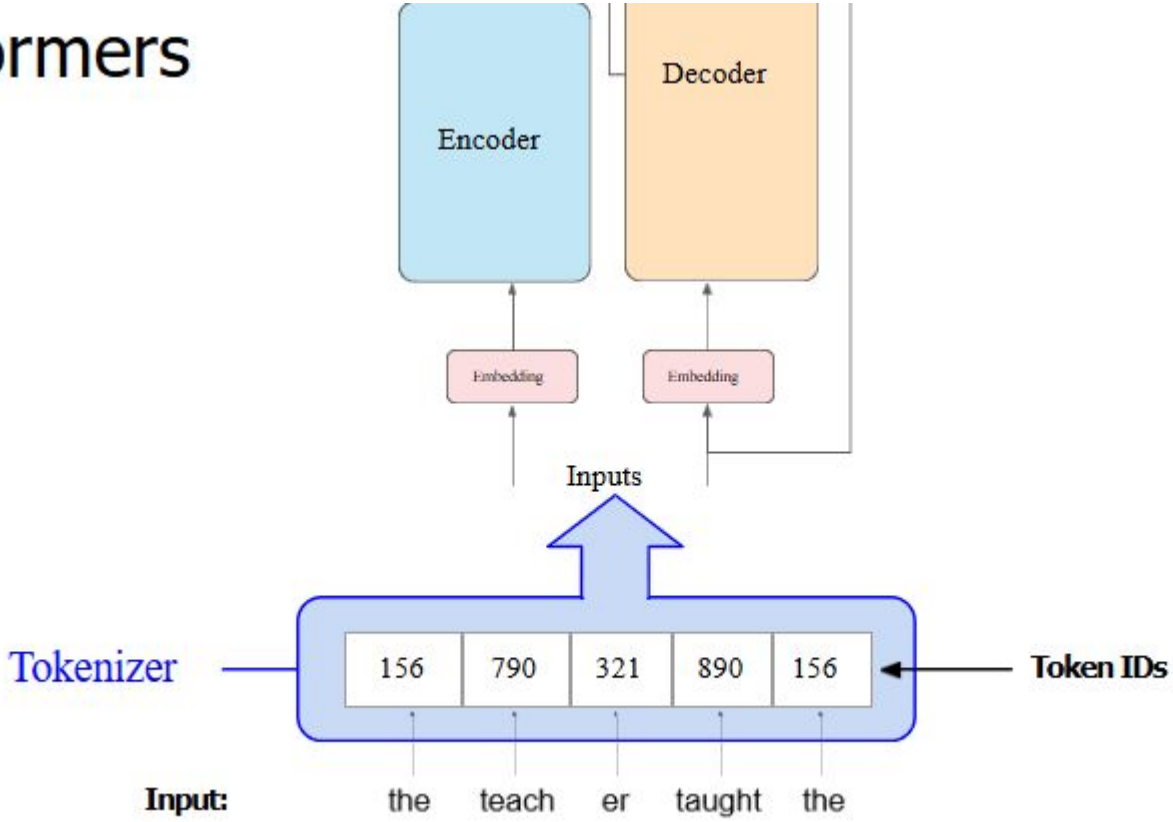
Transformers



Transformers

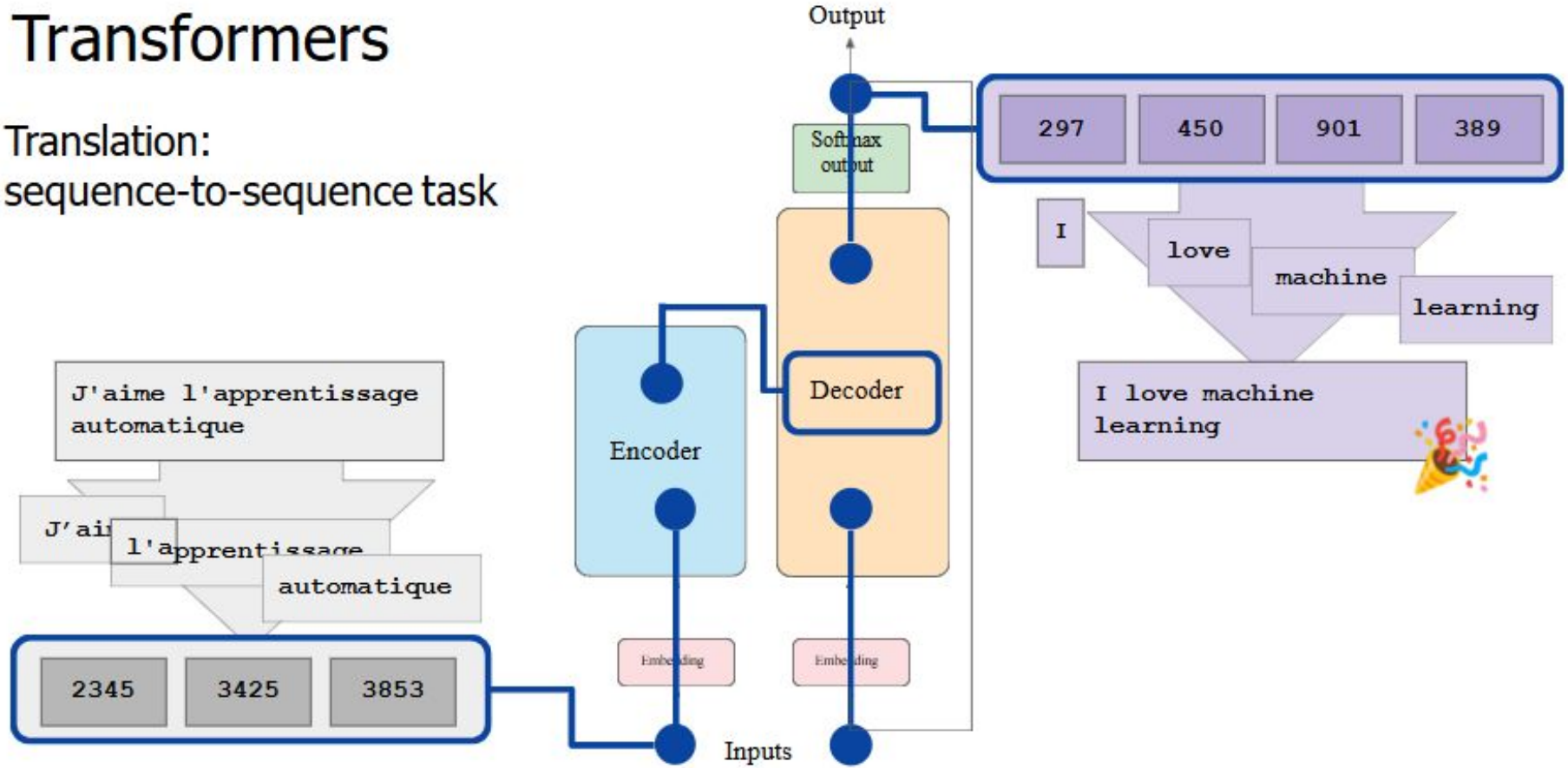


Transformers



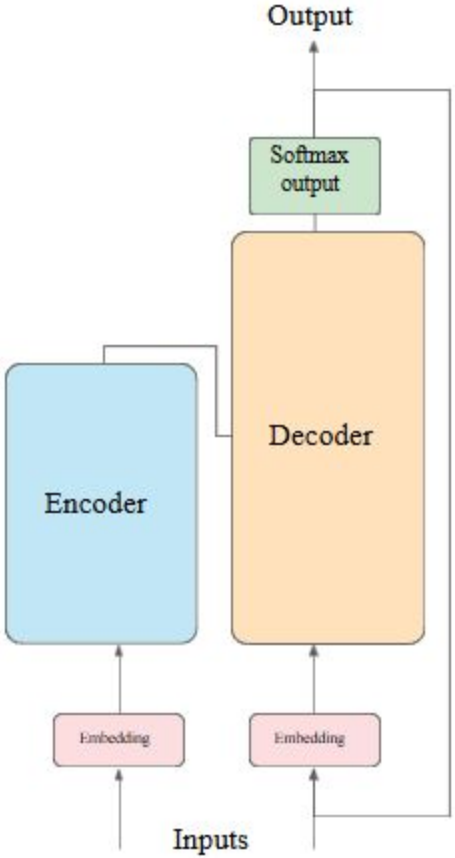
Transformers

Translation:
sequence-to-sequence task



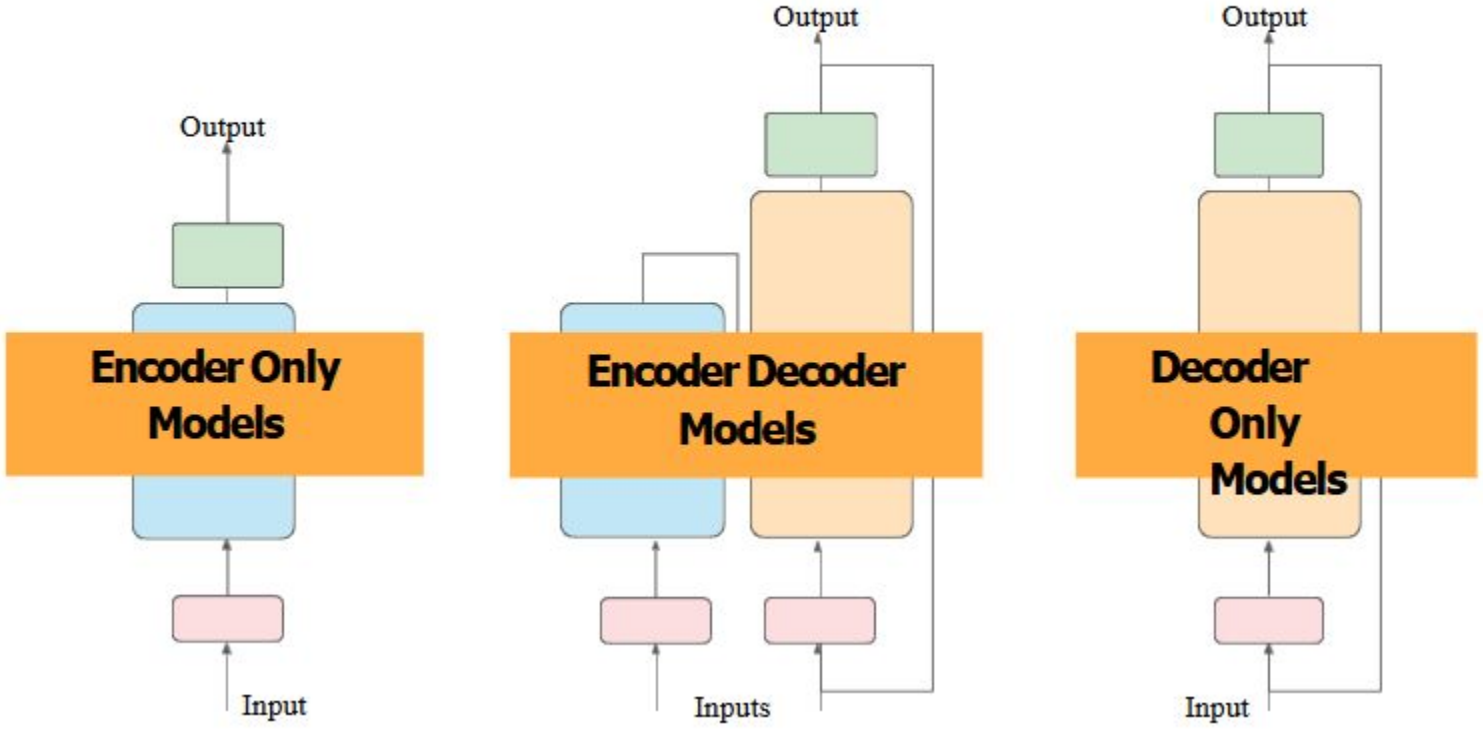
Transformers

Encoder
Encodes inputs ("prompts") with contextual understanding and produces one vector per input token.



Decoder
Accepts input tokens and generates new tokens.

Transformers



Words, Tokens, & Embeddings

Tokens (The Lego Bricks): AI doesn’t read words; it breaks them into chunks called "Tokens."

- *Example:* The word **"wonderful"** is broken into `won` + `der` + `ful`.
- **Why?** This helps the AI understand new or complex words by looking at their smaller, familiar pieces.

ID Numbers (The Serial Codes): Since computers only speak "number," every Lego brick (token) gets a unique ID.

- *Example:* `won` might be **#101**, and `der` might be **#502**.

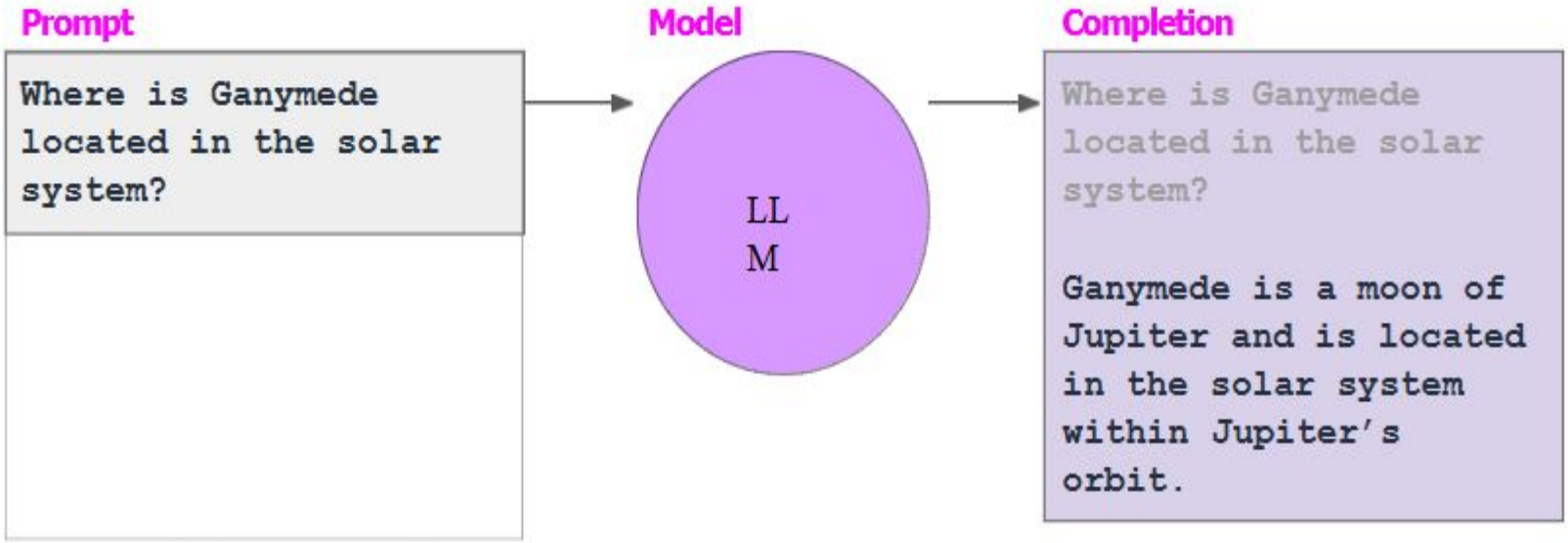
Embeddings (The GPS Map): An "Embedding" is a secret code that tells the AI where a word belongs on a **"Meaning Map."**

- Words that are similar (like **"Cup"** and **"Mug"**) are placed right next to each other on the map.
- Words that are different (like **"Cup"** and **"Spaceship"**) are placed miles apart.

The Context Window (The Workbench): This is the AI's **Short-Term Memory.**

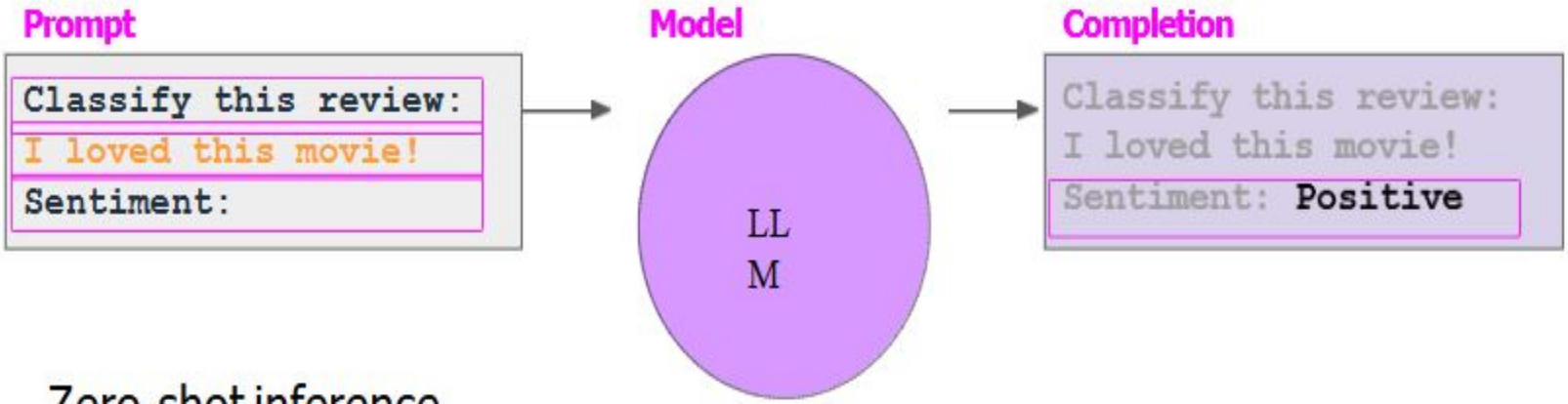
- It’s the size of the desk where the AI lays out its Lego bricks to build an answer.
- If the desk is too small, the AI has to "sweep off" the old bricks to make room for new ones, causing it to "forget" the start of your conversation.

Prompting and prompt engineering



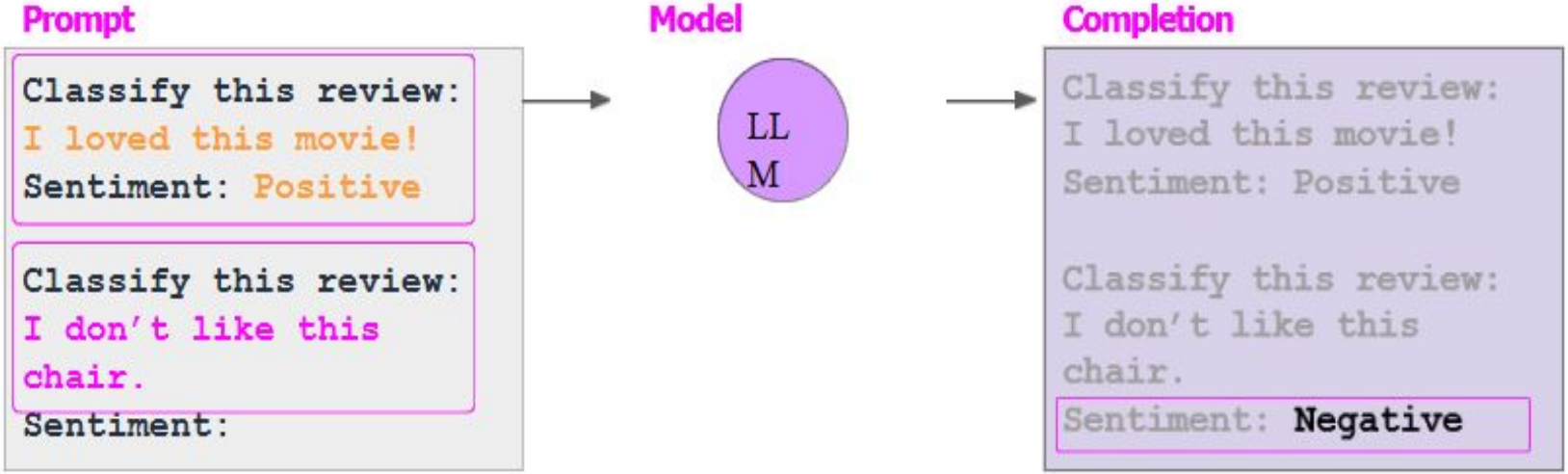
Context window: typically a few thousand words

In-context learning (ICL) - zero shot inference



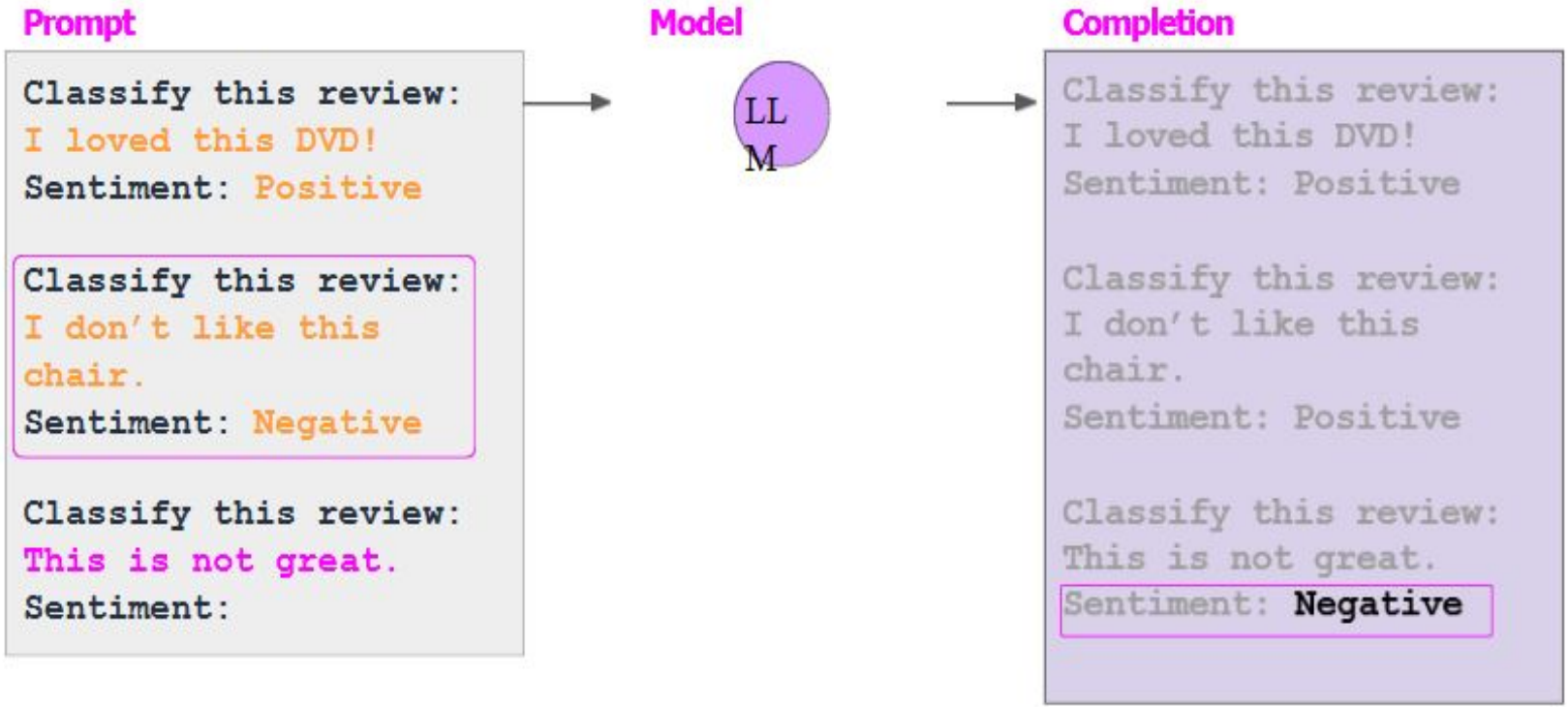
Zero-shot inference

In-context learning (ICL) - one shot inference



One-shot inference

In-context learning (ICL) - few shot inference



Summary of in-context learning (ICL)

Prompt / Zero Shot

```
Classify this review:
I loved this movie!
Sentiment:
```

Prompt / One Shot

```
Classify this review:
I loved this movie!
Sentiment: Positive

Classify this review:
I don't like this
chair.
Sentiment:
```

Prompt / Few Shot >5 or 6 examples

```
Classify this review:
I loved this movie!
Sentiment: Positive

Classify this review:
I don't like this
chair.
Sentiment: Negative

Classify this review:
Who would use this
product?
Sentiment:
```

Context Window
(few thousand words)

COT & Cognitive Verifier

- Chain of Thought (CoT): 'Let’s think step-by-step'
- Cognitive Verifier: Breaking down prompts into verified sub-questions

Programmatic Access & LangChain

- API vs Local (Ollama/Llama.cpp)
- LangChain: Framework for chaining LLMs, Prompts, and Memory
- Agents: Giving LLMs tools to use