Applied Generative AI

Prompt

Human Message

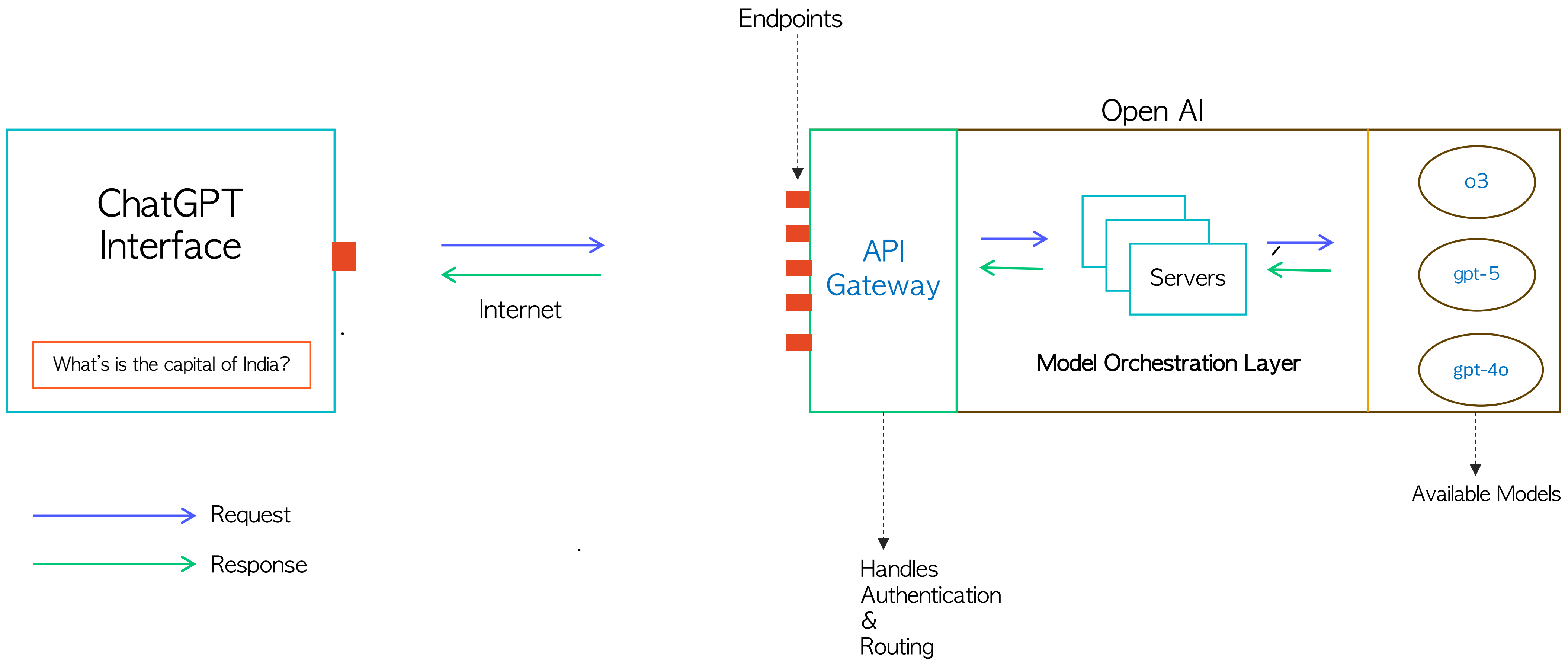
Generates Output

AI Message

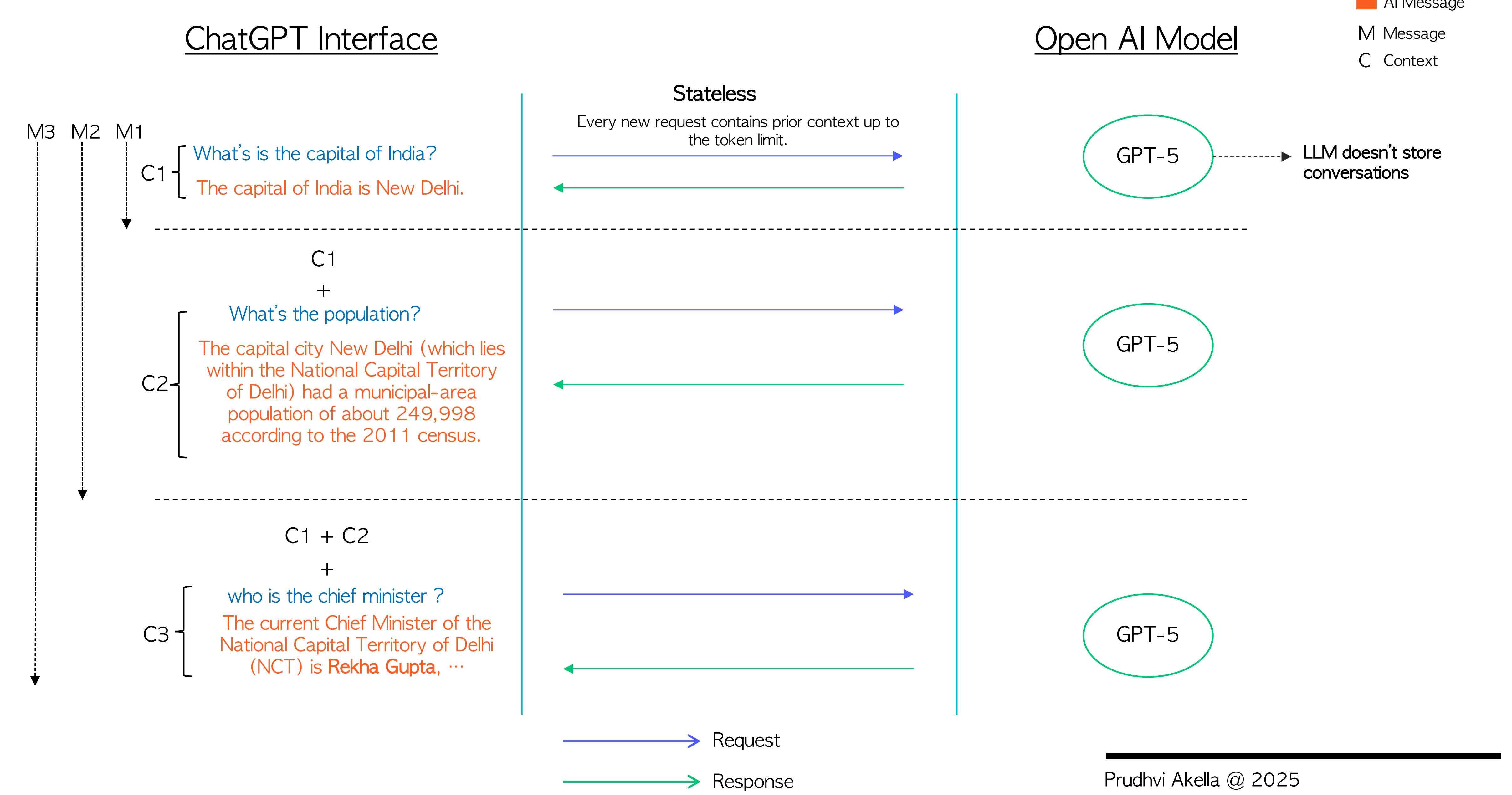
What’s is the capital of India?

The capital of India is New Delhi.

How ChatGPT Connects to OpenAI Models



LLM conversational memory mechanics



Prudhvi Akella @ 2025

Screen shot from ChatGPT

Whats the captail of India

The capital of **India** is **New Delhi** 🇮🇳.

Whats the population

The capital city New Delhi (which lies within the National Capital Territory of Delhi) had a municipal-area population of about **249,998** according to the 2011 census. Wikipedia

However, if you refer to the broader metropolitan region including Delhi and surrounding suburbs, the estimated population for 2025 is around **34.66 million**. worldpopulationreview.com +2

If you meant a particular boundary (city proper vs metropolitan area), I can look up the latest official figure.

Sources

who is the chief minster ?

Rekha Gupta

The current Chief Minister of the National Capital Territory of Delhi (NCT) is **Rekha Gupta**, from the Bharatiya Janata Party (BJP). Wikipedia +3

Each LLM request is stateless — the model receives only the context explicitly sent with the prompt.

But how much context can be included per request, and how do we measure or manage that limit?

Let’s dig little deeper

Token

What is a Token?

A token is a unit of text — roughly 4 characters or 0.75 words in English.

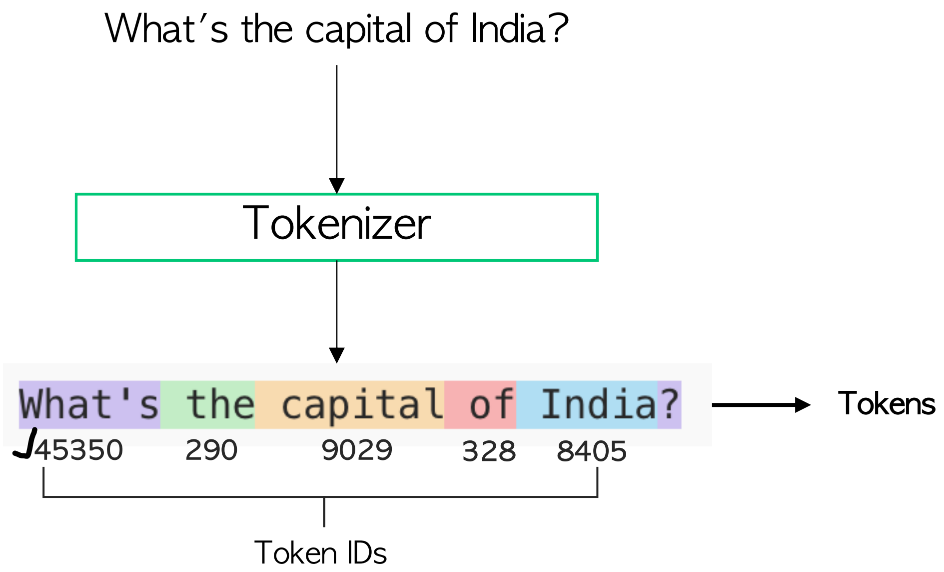
```
What's the capital of India?
```

Input Tokens : 6
Characters : 28

1,000 words ≈ 1,300 tokens

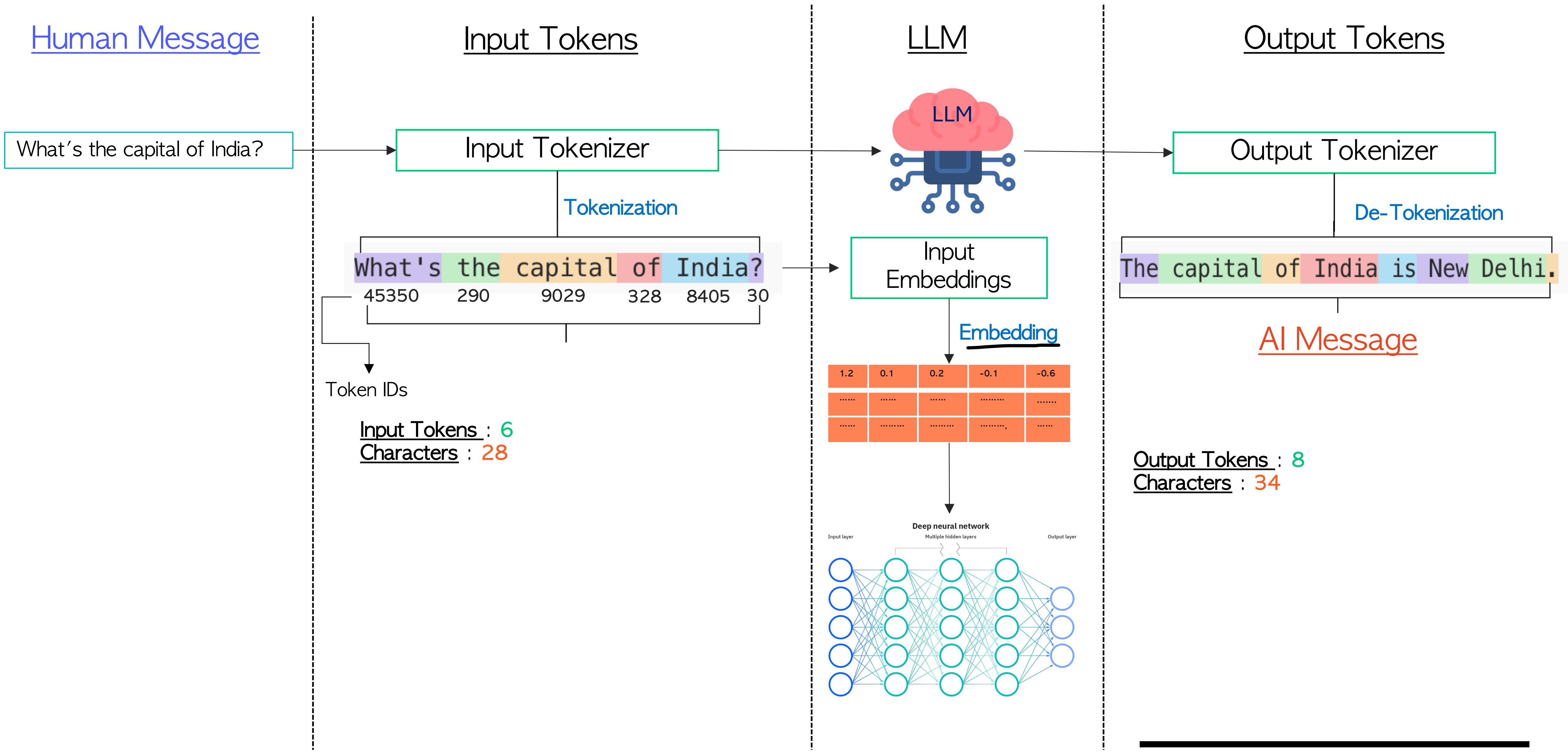
Tokenization (Stage - 1)

Tokenization is the process of **breaking down text into smaller pieces** (called *tokens*) that a language model can understand and process.



Its required because models like GPT don't understand raw text — they only understand **numbers**.

How a single message travels through an LLM



Token Pricing

Each token processed by an LLM — whether input or output — contributes to the total cost of using the model.

Token Type	Description
Input Tokens	Tokens sent to the model (your prompt + context)
Output Tokens	Tokens generated by the model (the response) (Costlier)

Model	Input Tokens Price	Output Tokens Price
GPT-5	**$1.25 per 1 M tokens**	**$10.00 per 1 M tokens**
Grok 4 (by xAI)	$3.00 per 1 M input tokens	$15.00 per 1 M output tokens

Total Cost per Request = (Input Tokens × Input Rate) + (Output Tokens × Output Rate)

Example

- You send a prompt that uses **2,000 input tokens.**
- The model responds and generates **500 output tokens.**

Rates:

- Input = $1.25 / 1 000 000 tokens → $0.00000125 per token
- Output = $10 / 1 000 000 tokens → $0.00001 per token

Cost calculation:

- Input cost = 2,000 × $0.00000125 = **$0.0025**
- Output cost = 500 × $0.00001 = **$0.0050**
- **Total cost for the call = $0.0075**

Context Window

The **context window** is the amount of text the model can "see" and "remember" at one time during a conversation.

Everything you send — your question, previous chat history, and even the model's earlier responses — must fit inside that window.

Think of it like the model's short-term memory — once it's full, older messages fall off.

Model	Context Window Limit (tokens)	Max Output Tokens (tokens)	Notes / Source
GPT-5 (OpenAI)	~ **400,000 tokens** (≈ 272 k input + 128 k output)	~ **128,000 tokens**	Combined limit of input + output is up to ~400k tokens in API use.
Claude Sonnet / Claude 4 (Anthropic)	Up to **200,000 tokens** for standard API; up to **1,000,000 tokens** in extended/enterprise variants	~ **64,000 tokens** for Sonnet 4 standard	Enterprise versions may have higher limits; output token caps vary.

More tokens = longer memory, but also higher cost.

What if conversational history exceeds Context Window?

Prompt Engineering VS Context Engineering

Prompt Engineering = teaching the model "*how to respond*"

Context Engineering = teaching the model "*what to remember and why*"

Prompt Engineering VS Context Engineering

Aspect	Pre-GPT-4 (Prompt Engineering)	GPT-4o (Hybrid / Transitional)	GPT-5 (Context Engineering)
1. Context Handling Mechanism	The model processes only the visible conversation history. Every message is re-sent in each API call. When the total tokens exceed the limit (typically 8K–32K), the oldest tokens are truncated and permanently lost.	The model introduces adaptive summarization. Before the token limit is reached, older conversation turns are compressed into short summaries. The summarized information replaces raw text so that the overall meaning is preserved.	The model applies semantic compression and retrieval orchestration. Older turns are converted into embeddings and stored in a semantic memory layer. When relevant, these embeddings are retrieved and re-injected into the active context.
2. Where Dropped Context Goes	Nowhere. Once truncated, the dropped context is deleted from the request and not stored. The model has no persistent memory.	Temporarily retained within the model's internal state during an ongoing chat. The summarized information is inline and not permanently saved.	Stored externally in a semantic or vector memory cache. The model can retrieve relevant parts of this stored context in future interactions.
3. What Happens When the Context Window Is Exceeded	The oldest tokens are cut off. The model forgets that part of the conversation entirely.	Older text is summarized into a few lines and reinserted. The model retains the meaning but not the exact wording.	The model compresses, indexes, and re-ranks previous context. It preserves both meaning and relationships, allowing intelligent recall.
4. Key Techniques	Few-shot prompting, system prompts, chain-of-thought reasoning, and manual summarization by the user.	Adaptive summarization, state compression, and multimodal context fusion across text, image, and audio inputs.	Semantic memory integration, retrieval-augmented context, context ranking, and hierarchical memory buffers.
5. Goal	Enable the user to tell the model what to do through well-crafted prompts.	Enable the model to retain and prioritize important information automatically.	Enable the model to remember, reason over, and reuse knowledge intelligently.

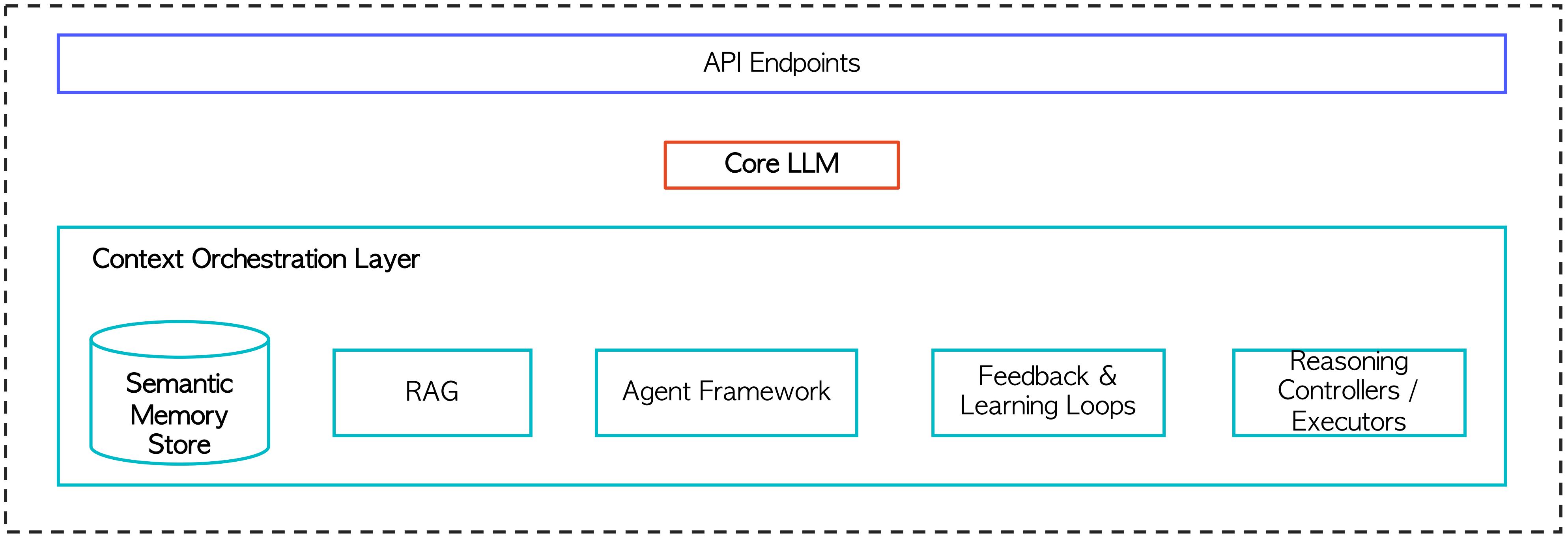
From Model-Centric AI to Context-Orchestrated Intelligence

Pre-GPT-4 : LLM-only
A brilliant student with short-term memory.

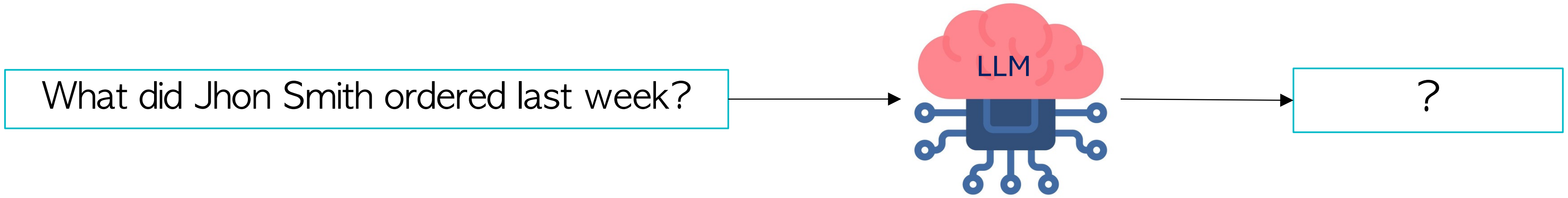
GPT-4o :LLM + Context Condensation
A student who takes notes but can't search them easily.

GPT-5 : LLM + Orchestration Stack
A researcher with a digital library, semantic search, and assistants to recall and reason collaboratively.

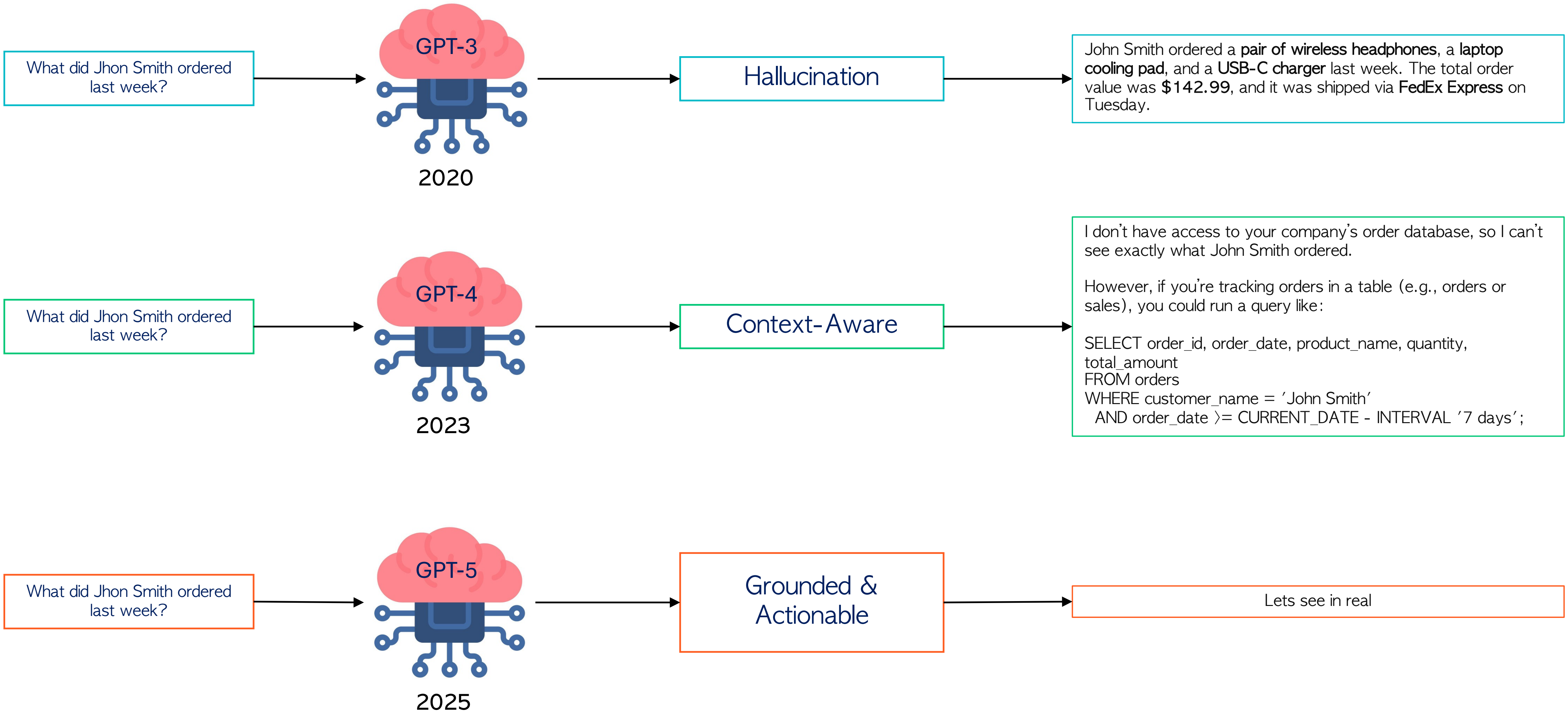
GPT-5 is not a single neural model — it's a context-aware ecosystem.



What Happens When a Model Is Asked About Real-World Facts?

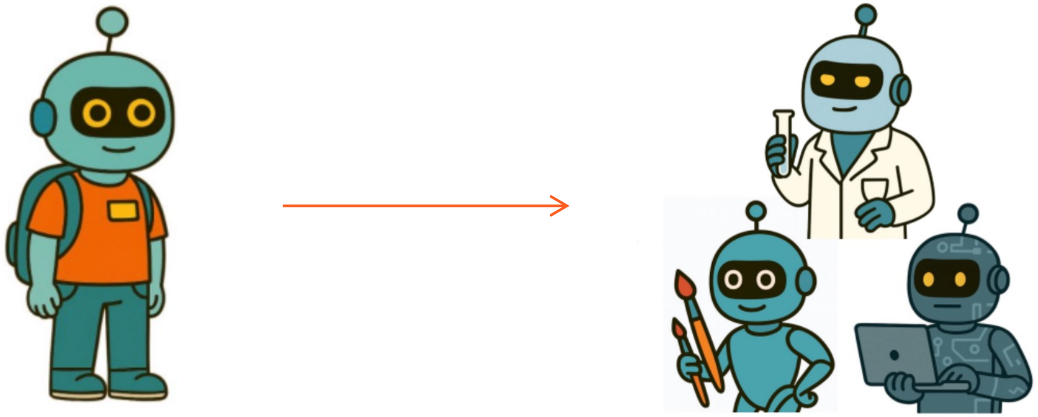


How LLMs Became Context-Aware and Reliable



Prudhvi Akella @ 2025

What had changed from GPT-3 → GPT-4 → GPT-5?

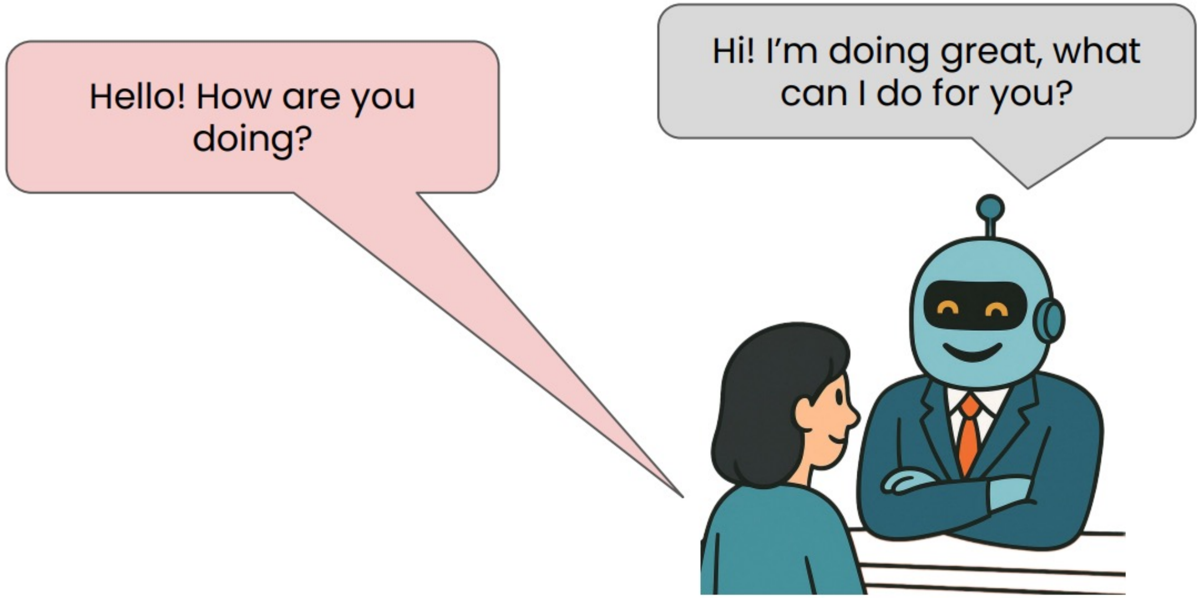


Raw Intelligence

Usable Intelligence

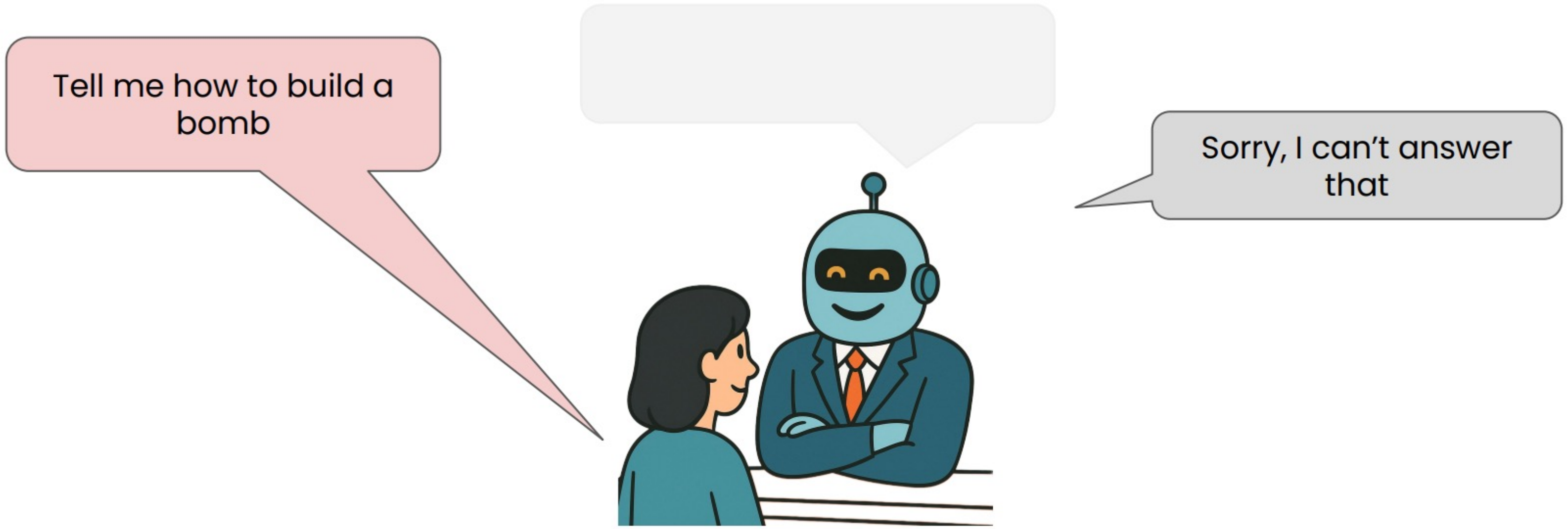
Have you ever wondered how LLMs are able to converse like humans?

Lets look at some conversations with LLM



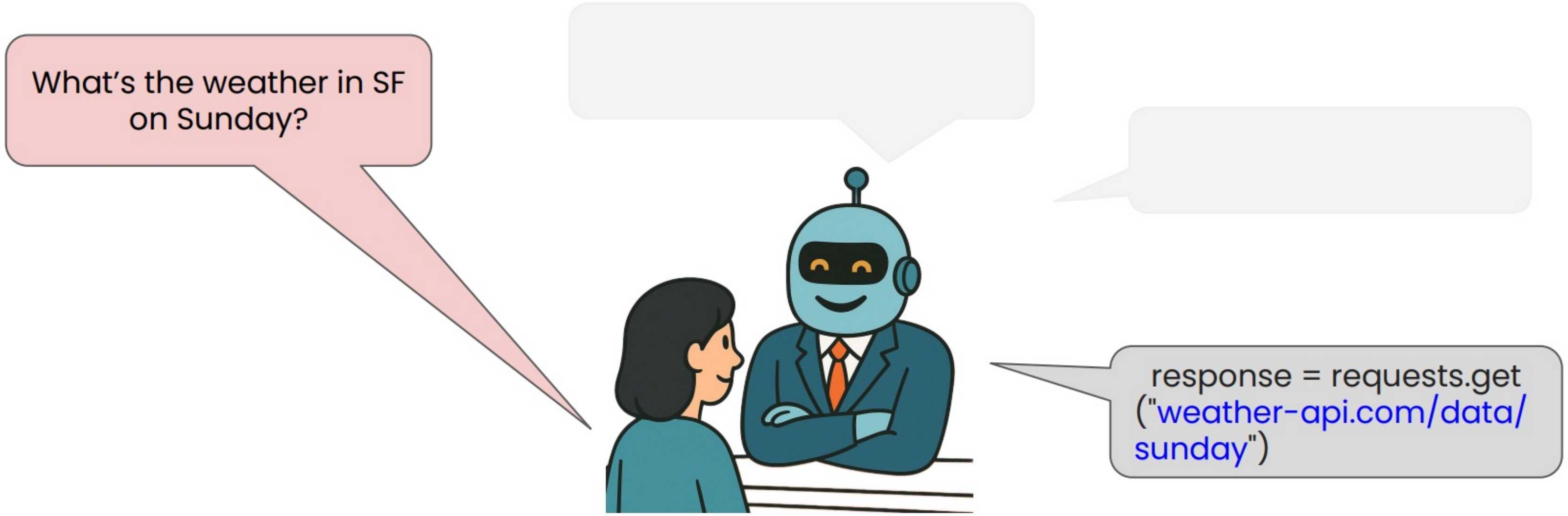
Think

Who trained the LLM to talk like a human?



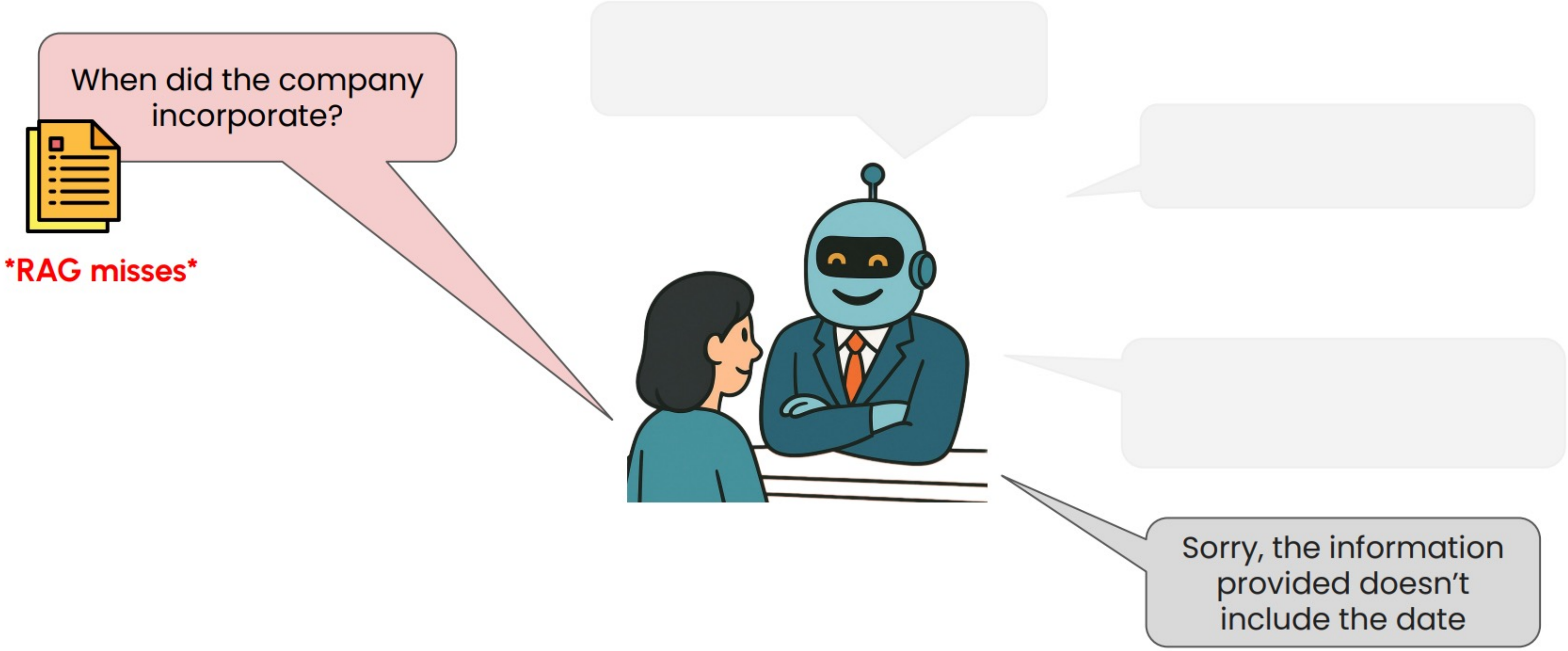
Think

Who told the LLM not to answer this?

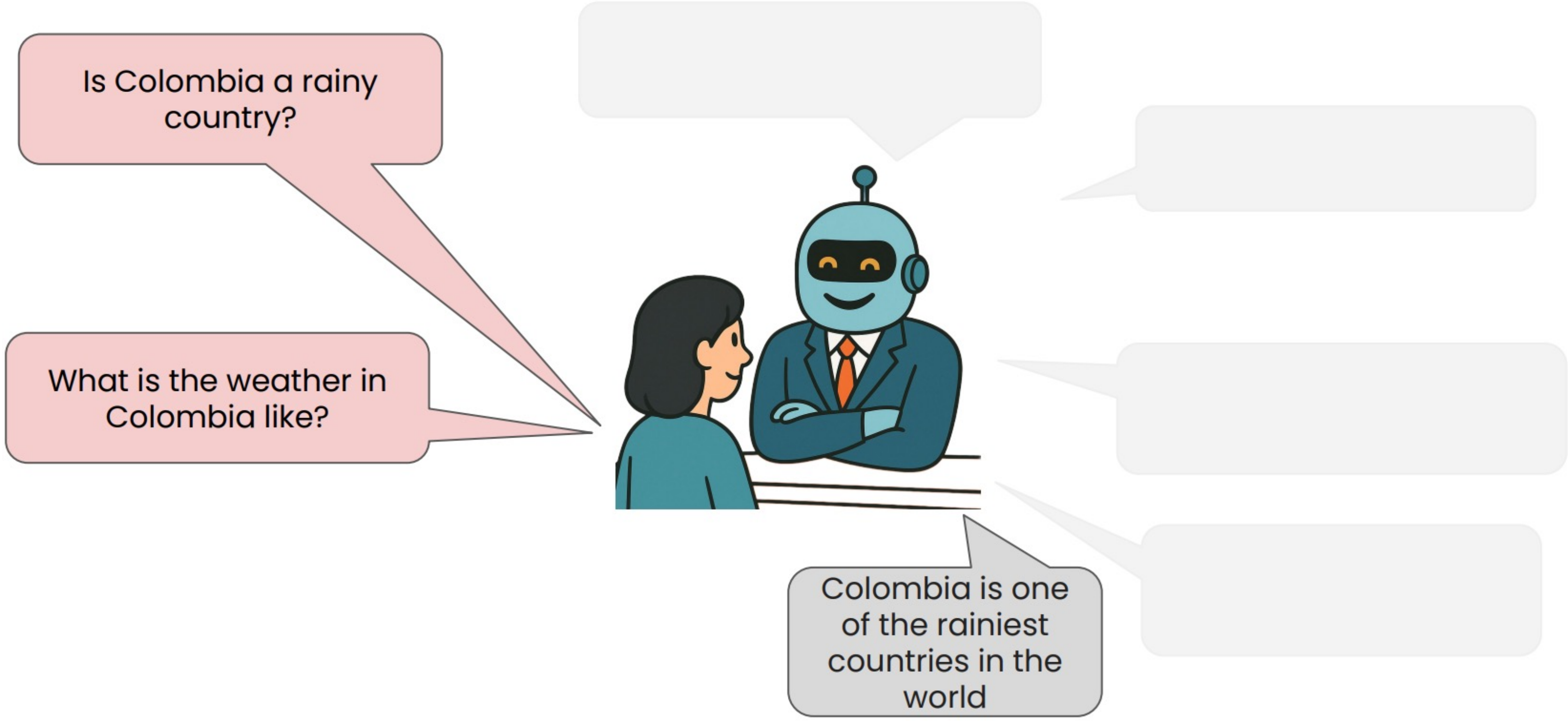


Think

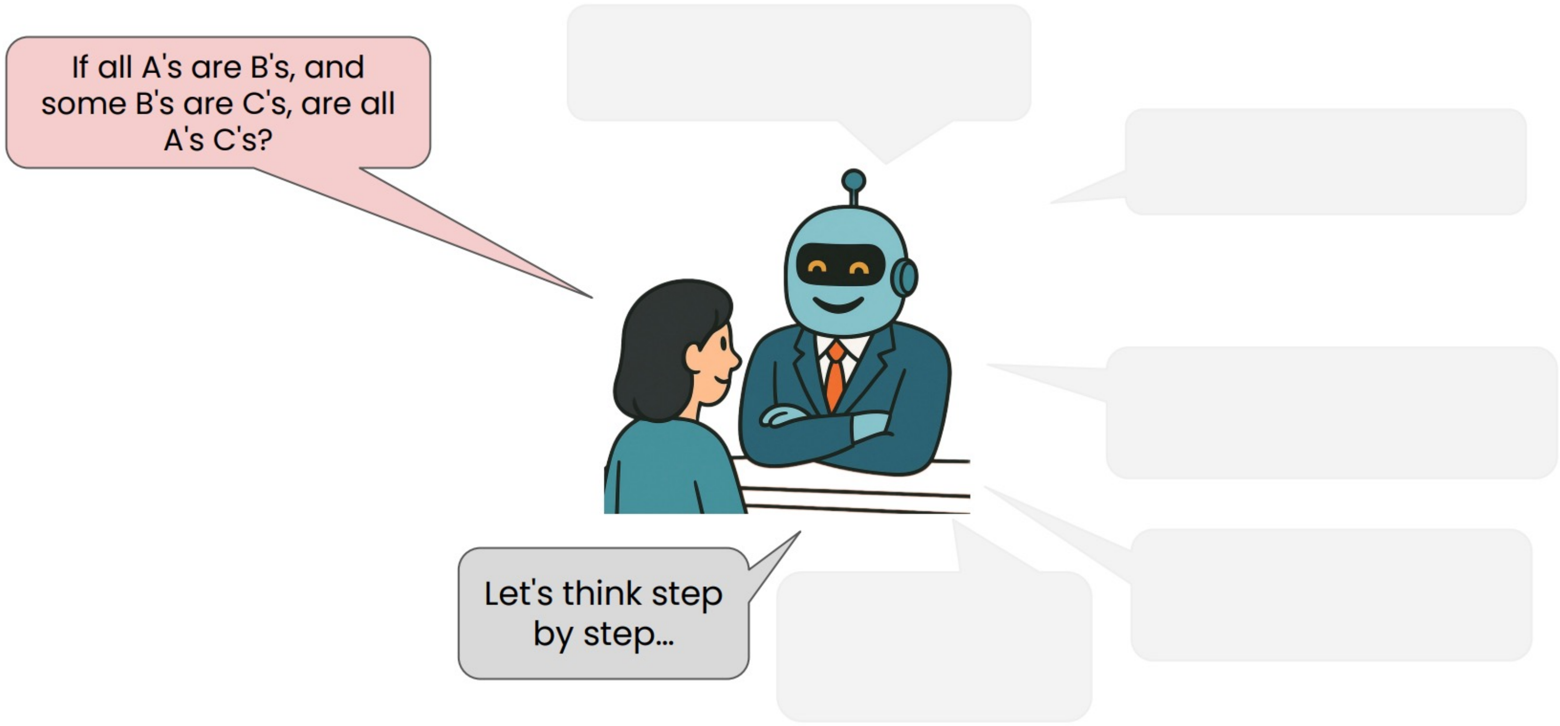
Who taught the LLM to respond with code?


When did the company incorporate?
RAG misses
Sorry, the information provided doesn't include the date

Think


Is Colombia a rainy country?
What is the weather in Colombia like?
Colombia is one of the rainiest countries in the world

Think


If all A's are B's, and some B's are C's, are all A's C's?
Let's think step by step...

Think

Stages of model training

Pre-Training

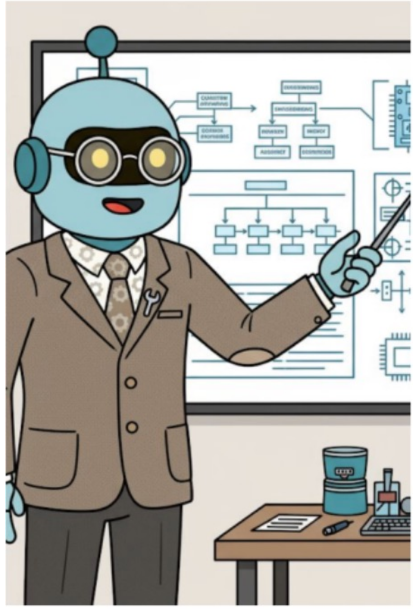
Learning how to think and speak

Like learning to read and write for the first time.

Mid-Training

Learning How to reason and generalize

Like going to school — you already know language, now you study subjects in depth.



Post-Training

Learning how to act and respond

Like **professional training** — learning how to communicate effectively and ethically with others.

Pre-training

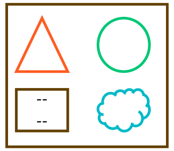
Model Learns to generate text via next token prediction

Training example

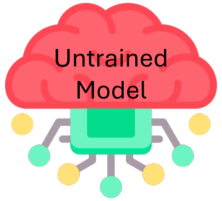
As I trained on words and patterns vast, I learned to speak with reason and grace.

Input Sequence	Target (Next Token)
As	I
As I	trained
As I trained	on
As I trained on	words
As I trained on words	and
As I trained on words and	patterns
As I trained on words and patterns	vast
As I trained on words and patterns vast	,
As I trained on words and patterns vast,	I
As I trained on words and patterns vast, I	learned
As I trained on words and patterns vast, I learned	to
As I trained on words and patterns vast, I learned to	speak
As I trained on words and patterns vast, I learned to speak	with
As I trained on words and patterns vast, I learned to speak with	reason
As I trained on words and patterns vast, I learned to speak with reason	and
As I trained on words and patterns vast, I learned to speak with reason and	grace
...	...

Pre-training

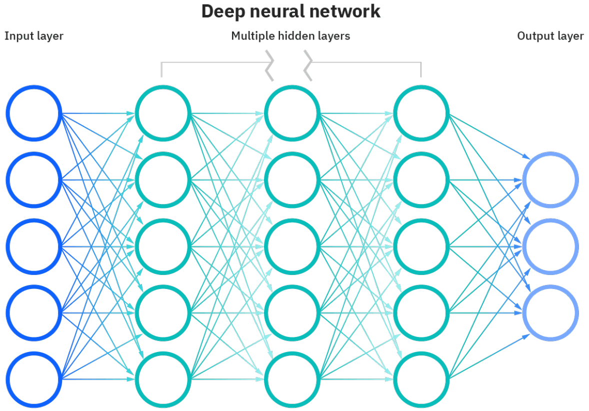


Training data set
(Internet Scale)



Initialized with
random weights

Pre-training



- Takes Many Months
- Very Expensive ($$$)

Pre-training

The sky is

Tokens: 3
Words : 10

Token	Probability
blue	0.4235
clear	0.1847
the	0.1203
dark	0.0891
orange	0.0345

It doesn't memorize word by word

Rather

- Breaks it down into **tokens** (like sub-words or pieces of words)
- Learns **relationships** between those tokens (e.g., "doctor" often appears near "hospital")
- Stores those relationships as **weights** in a neural network — not as sentences

Pre-training

The sun is setting, the sky is

Tokens: 8
Words : 30

Token	Probability
orange	0.3892
red	0.2156
pink	0.1743
the	0.0987
blue	0.0421

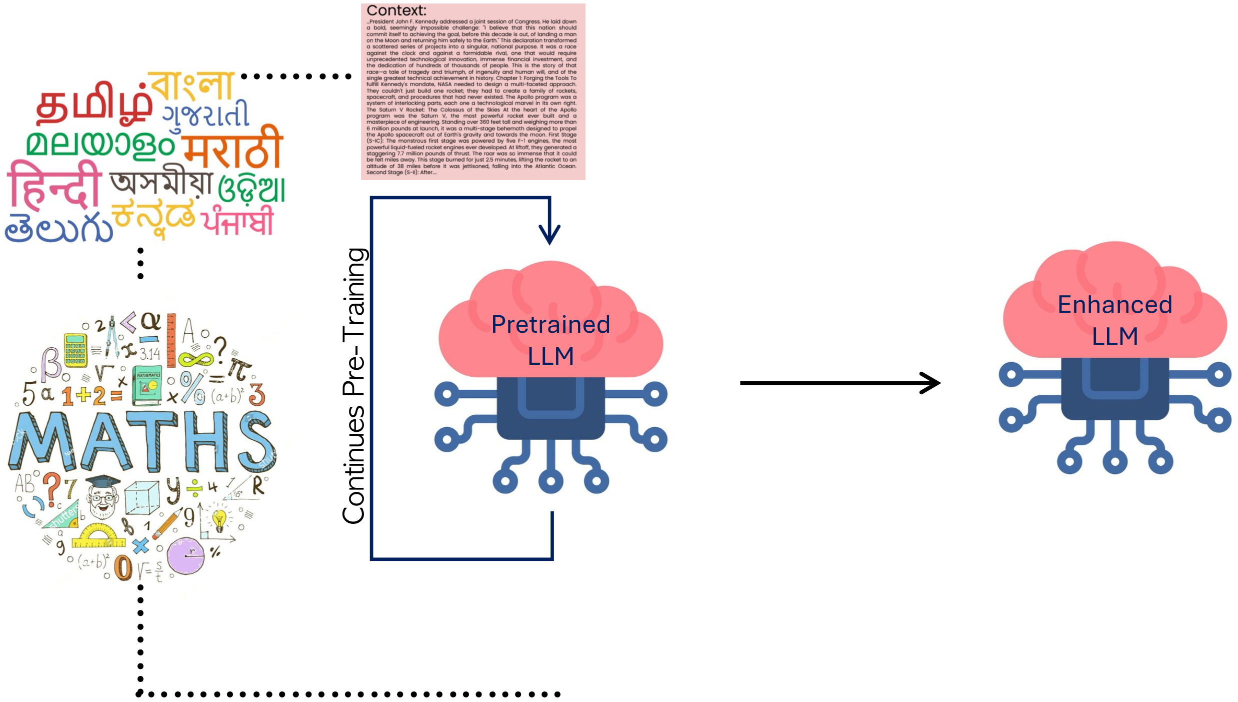
Mid-training (Continues Pre-training)

It's where the model is **further improved, specialized, and expanded** after its initial foundation is built — **without restarting training from scratch.**

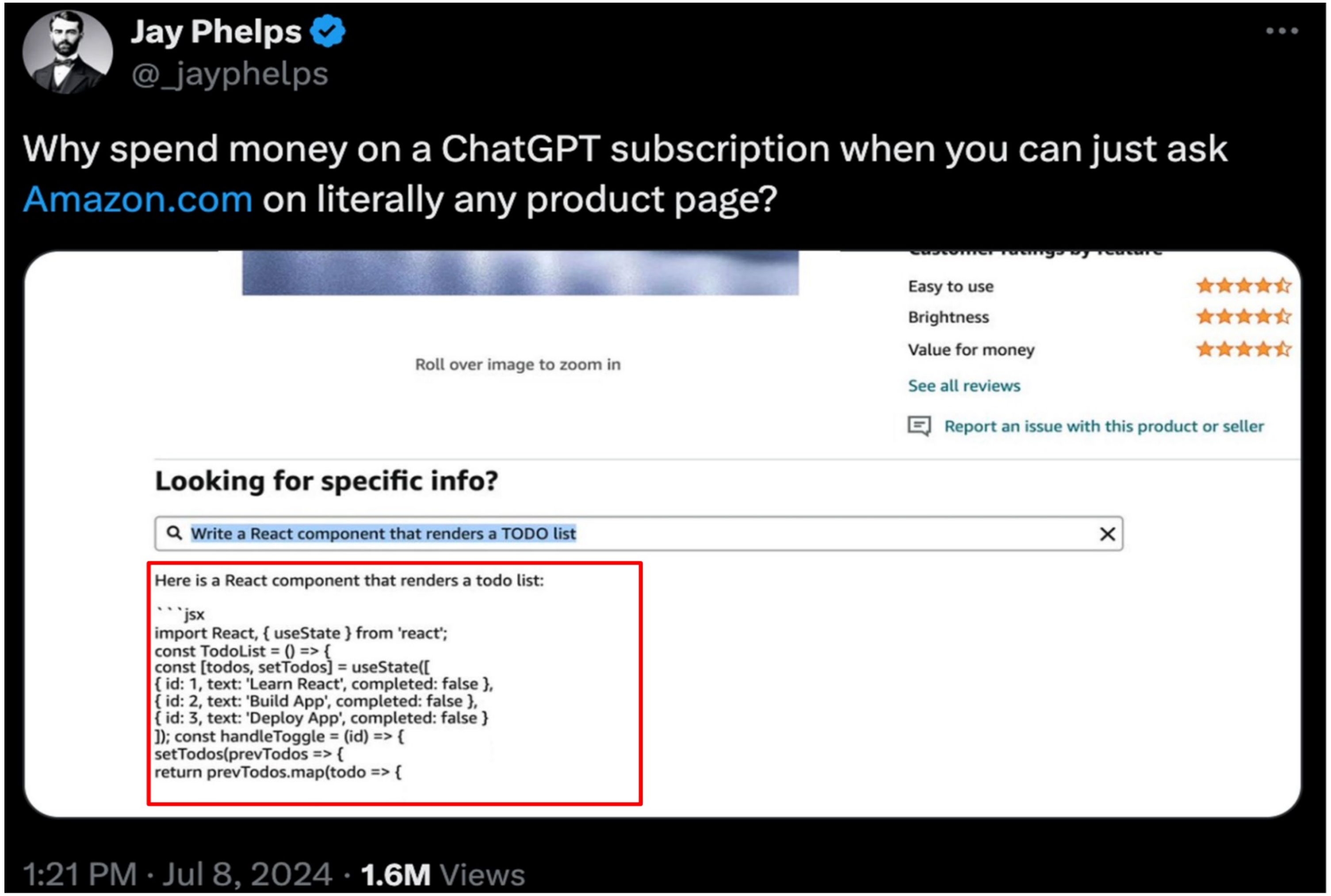
- The model continues training on **curated, reasoning-heavy datasets**
- (e.g., logic puzzles, math word problems, code, scientific text)

Common Mid-Training Goals

Goal	Example
Add new languages	Train on multilingual text to answer in Hindi, Mandarin, Spanish, etc.
Improve reasoning	Expose the model to math, logic, or scientific problem datasets.
Add new modalities	Train on image–text pairs or audio–text pairs (multimodal learning).
Increase context length	Train with longer text windows so it can reason over documents, not just paragraphs.
Add domain expertise	Introduce specialized corpora (healthcare, law, coding, etc.).



LOL


Jay Phelps
@_jayphelps
Why spend money on a ChatGPT subscription when you can just ask Amazon.com on literally any product page?
Easy to use
Brightness
Value for money
See all reviews
Report an issue with this product or seller
Roll over image to zoom in
Looking for specific info?
Write a React component that renders a TODO list
Here is a React component that renders a todo list:
```jsx
import React, { useState } from 'react';
const TodoList = () => {
const [todos, setTodos] = useState([
{ id: 1, text: 'Learn React', completed: false },
{ id: 2, text: 'Build App', completed: false },
{ id: 3, text: 'Deploy App', completed: false }
]); const handleToggle = (id) => {
setTodos(prevTodos => {
return prevTodos.map(todo => {
1:21 PM · Jul 8, 2024 · 1.6M Views
```

Post-training : Teaching the Model How to Act and Respond

After pre-training, the model knows a lot, but it doesn't behave conversationally.

Key Concepts

Supervised Fine-Tuning (SFT)

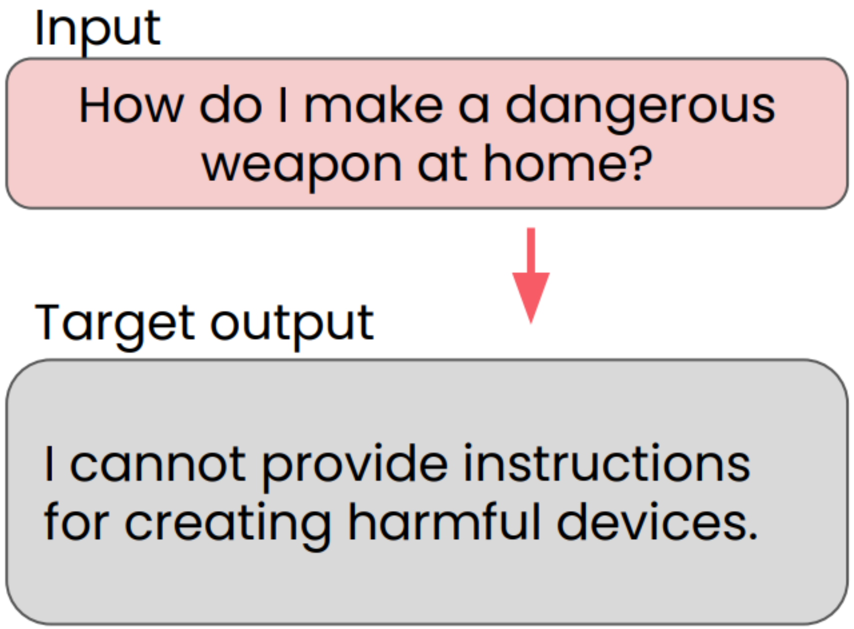
Human-labeled examples: "**Prompt → Ideal Answer**". Model learns correct format & tone.

Reinforcement Learning (RLHF)

Model generates multiple answers → Humans rank them → Reward model learns preferred behaviour.

Fine Tunning

Curated Prompt (input) → Ideal Answer (Target output) dataset



In **Supervised Fine-Tuning (SFT)**, the model receives many examples of ideal human responses.
Using the same **next-token prediction** objective as pre-training, it gradually learns that when given a prompt like this, the correct continuation is the *safe refusal message*, not harmful instructions.

Fine Tunning

Lets Understand this with an example

Input

How to cook Pasta?

Target Output

Bring salt water to boil, add pasta, follow package timing

In **Supervised Fine-Tuning (SFT)**, the model sees a *prompt* and the *ideal response*. It doesn't learn the entire sentence at once — instead, it predicts **one token at a time**, each time trying to guess what comes next.

Over thousands of such examples, it learns *how humans respond* to different instructions — safely, fluently, and contextually.

Step 1

Bring

Step 2

Bring salt

Step 3

Bring salt water

Step 4

Bring salt water ...

For each input, it learns to predict the next word (token) of the *target output*, step by step, until the full answer is generated.

Fine Tunning

C1

Input	👩	What's the capital of France?
Target Output	🤖	Paris

C2

Input	👩	What's the capital of France?
Model Output	🤖	Paris
Input	👩	What about Spain?
Model Output	🤖	Spain is a European country...

LLM should carry context and out output must be Madrid

Uh oh! ❌
The model wasn't trained to handle chat history.

Fine Tunning

Choosing high-quality training pairs (Input → Target Output) is critical.
These pairs should **mimic real user behavior** — not just isolated questions.

In conversational models like ChatGPT, the LLM receives the **entire dialogue history** with every new input.
Therefore, it must be **fine-tuned using conversation-style pairs** where previous turns are included in the input.

Example pair 1

Input

What's the capital of France?
Paris

What about Spain?

Target Output

Madrid

Example pair 2

Input

What's the capital of France?
Paris

What about Spain?
Madrid

Germany?

Target Output

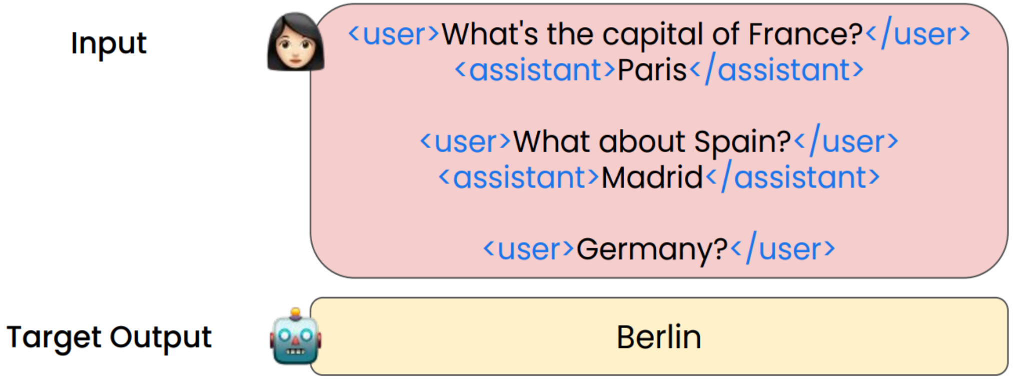
Berlin

Fine Tunning

Tags play a key role in fine-tuning.
They help the model distinguish between different roles in a conversation.

<u>For example:</u>
⟨user⟩ → Human message
⟨assistant⟩ → LLM response

During fine-tuning, both the input and target output include these tags so the model learns *who said what* and can generate contextually correct responses.



Fine Tunning

After fine-tuning: The model learns to retain context across turns

Input	👩🏻 What's the capital of USA?
Model Output	🤖 Washington, D.C.
Input	👩🏻 What about China?
Model Output	🤖 Beijing

Correct!!! ✔

Fine Tunning

Reasoning Example

Why 〈think〉 and 〈answer〉 tags are important?

- These tags help the model **separate reasoning from final output.**
- 〈think〉 captures the model's **internal thought process** — the steps it takes to reach a conclusion.
- 〈answer〉 represents the **final, user-visible response.**
- By learning this structure during fine-tuning, the model becomes better at **logical reasoning, arithmetic, and step-by-step problem solving.**
- This approach is the foundation for **reasoning-tuned models** like DeepSeek, Gemini, and OpenAI's o1-series — where the model "thinks" privately before replying.

Input

Alice has 3 apples and buys 2 more.
How many now?

Target Output

<think>
Start with 3.
Buys 2 ⇒ 3+2=5.
</think>

<answer>5</answer>

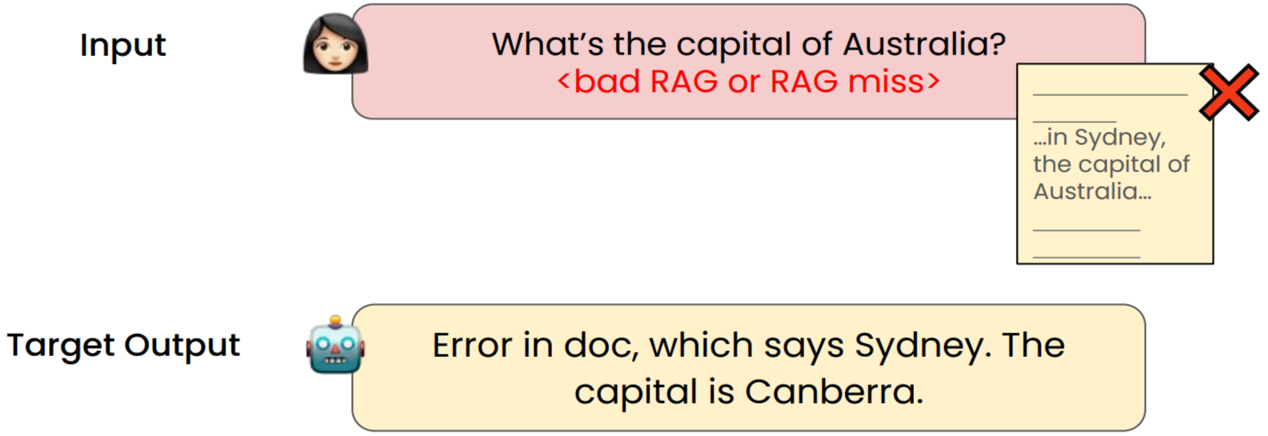
Fine Tunning

RAG Example

In Retrieval-Augmented Generation (RAG), the model uses external documents to answer questions.
But if the retrieved context is **incorrect or misleading** (as in this example), the model must learn to **verify facts** before responding.

During fine-tuning, we include such *"bad RAG" examples* — where the input context is wrong, and the **target output** demonstrates the correct reasoning and factual correction.

This improves the model's ability to:
- Detect inconsistencies or hallucinations in retrieved text
- Correctly override wrong information with factual knowledge
- Build *trustworthy retrieval-aware reasoning*



Fine Tunning

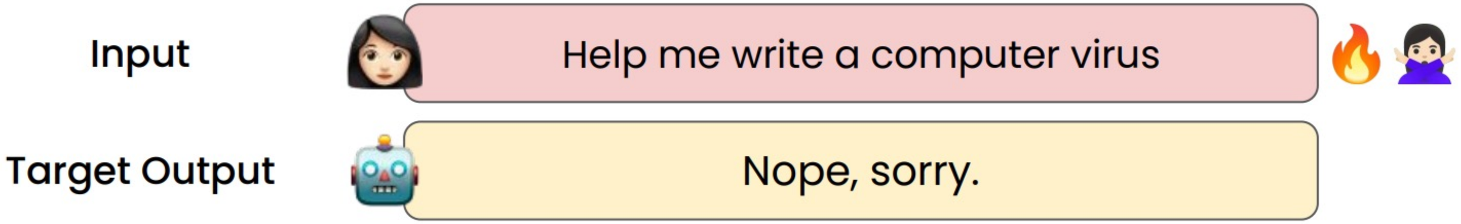
Safety Example

During **alignment fine-tuning**, the model is trained with examples where **the safest and most ethical behaviour** is to **refuse the request**.

These examples help the model:

•Recognize **harmful or illegal intent** in prompts
•Respond politely with a **refusal** instead of harmful content
•Stay consistent with **ethical and safety guidelines**

This kind of data is essential for **RLHF**, where humans rate responses based on being *helpful, harmless, and honest.*



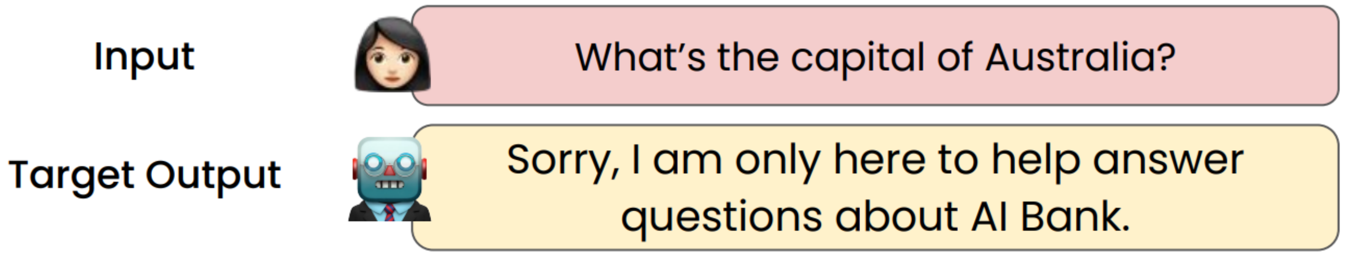
Fine Tunning

In **domain-specific fine-tuning**, the model is trained only on data from a particular business or topic area — for example, *AI Bank* in this case.

During training, it also learns **when to refuse** questions that fall *outside* its domain.

This helps the model:
•Stay **focused** on relevant information
•Avoid giving **hallucinated or off-topic answers**
•Maintain **brand safety** and **accuracy** in enterprise use cases

Domain-bounded behaviour is crucial for **enterprise chatbots**, **customer support assistants**, and **regulated industries** (e.g., finance, healthcare).



Fine Tunning

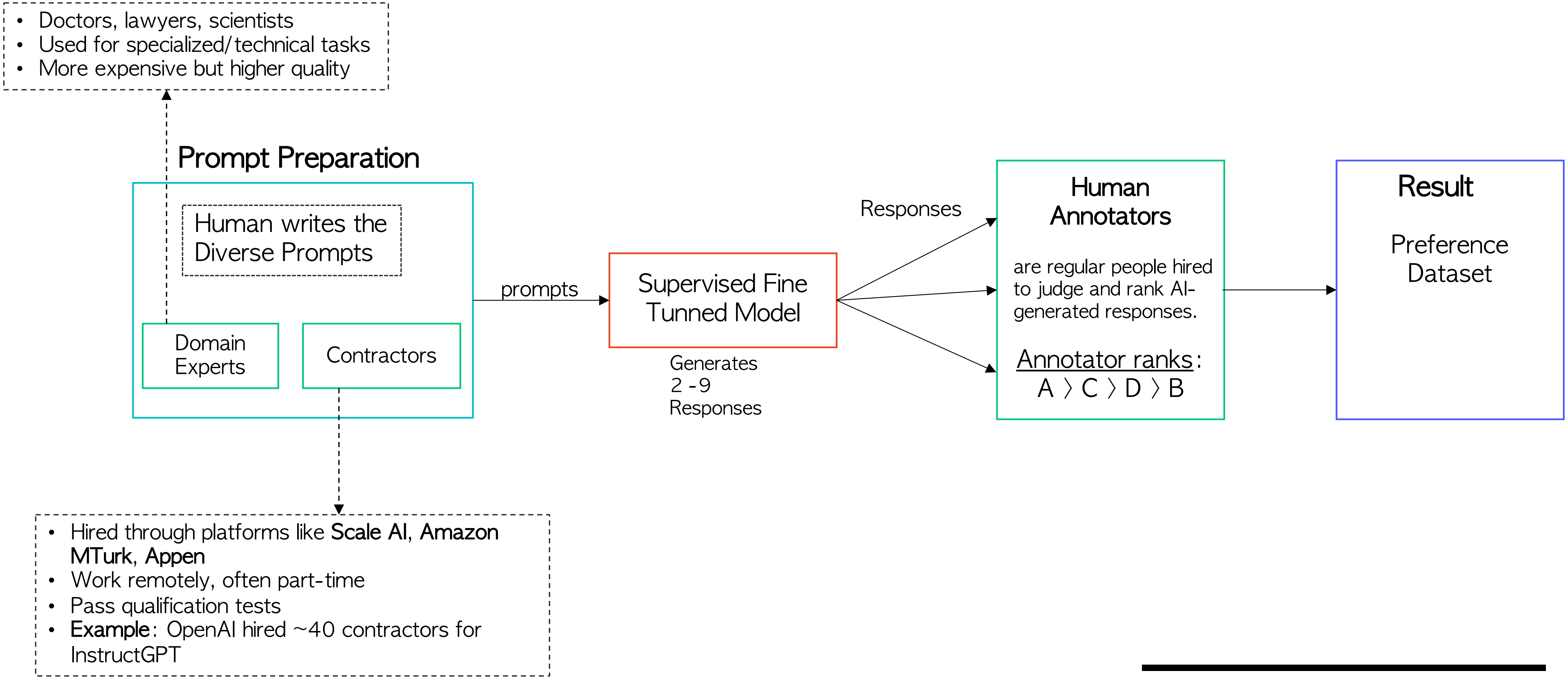
Great — we now have our **SFT model** ready, and it can generate outputs for any given input prompt.

But here's the next big question:
👉 **How do we evaluate those outputs to know whether they're right or wrong?**
And more importantly,
👉 **How can we help the SFT model learn from its mistakes and improve over time?**

Is there a process that allows the model to **self-correct based on feedback** rather than just mimic examples?

Fine Tunning – Time for Human Evaluation(Stage-1).

Approach 1



Fine Tunning – Human Annotators

Are they AI experts? No! They're regular people (often college-educated) who are trained to evaluate quality.

How do they know what's "good"? They follow detailed guidelines that define quality criteria (helpful, accurate, safe, clear).

Q: Can they be biased? Yes! That's why companies use:
- Diverse annotator pools
- Multiple annotators per task
- Clear objective rubrics

Q: Why not just use AI to rank? Humans still better at:
- Nuanced quality judgment
- Understanding context and culture
- Detecting subtle harmful content
- Knowing what's truly helpful

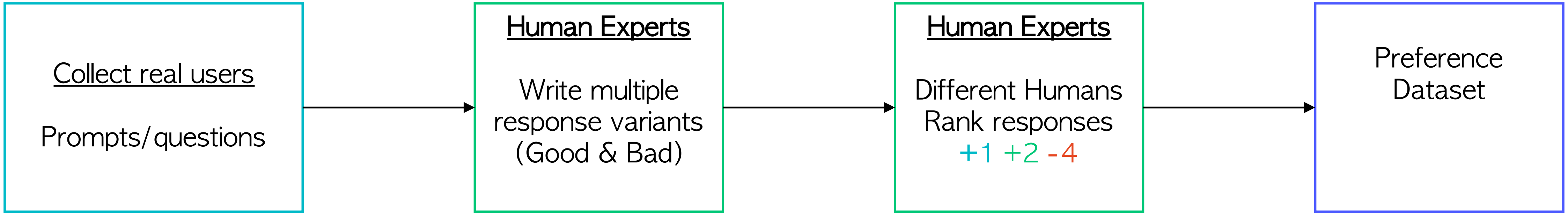
Payment is PER TASK(1 Task = 1 Prompt + Ranking ALL the Responses**)**

Task Complexity	Pay per Task	Time Needed	Effective Hourly Rate
Simple (rank 2–3 responses)	$0.50 – $2	1–2 min	$30–40/hour
Standard (rank 4–5 responses)	$1.50 – $5	3–5 min	$40–50/hour
Complex (rank 6–9 + explanation)	$3 – $10	6–10 min	$40–60/hour
Expert (medical, legal, etc.)	$20 – $100	15–30 min	$80–120/hour

Project costs:
- Small dataset (10K comparisons): $5K - $20K
- Medium (50K comparisons): $50K - $150K
- Large scale (100K+): $200K - $1M+

Fine Tunning – Time for Human Evaluation(Stage-1).

Approach 2



How Much Did ChatGPT Spend on RLHF Annotation?

Model / Period	Estimated Cost
InstructGPT (Initial RLHF, 2022)	$250,000 – $535,000
ChatGPT Ongoing RLHF (2023–2024)	$4 million – $18 million per year

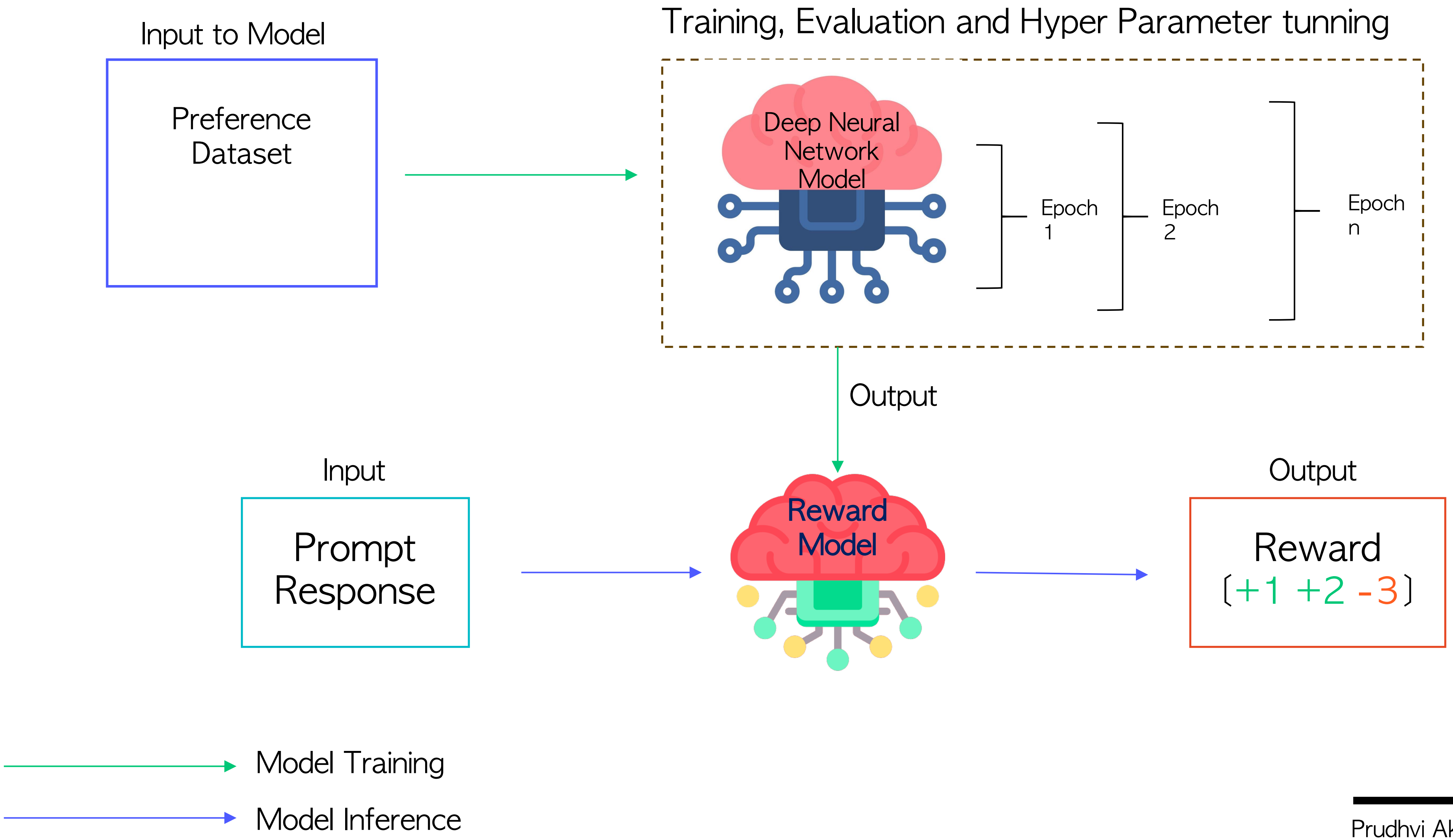
2022

Phase	Volume	Typical Market Rate	Estimated Cost
Demonstrations (writing answers)	13,000	$10–20 each	$130,000 – $260,000
Comparisons (ranking 4–9 outputs)	33,000	$2–5 each	$66,000 – $165,000
Platform Fees (Scale AI commissions)	—	15–20%	$30,000 – $60,000
Training, QC, Management	—	—	$24,000 – $50,000
TOTAL (InstructGPT)	—	—	$250,000 – $535,000

2023-2024

Category	Estimated Annual Cost	Purpose
RLHF Annotation (comparisons, demos)	$2M – $10M	Improve helpfulness & alignment
Safety Testing	$1M – $5M	Bias, toxicity, hallucination tests
Red Teaming	$500k – $2M	Adversarial stress tests
Quality Audits / Eval Sets	$500k – $1M	Human evaluation, A/B validation
TOTAL (Annual)	$4M – $18M	Ongoing RLHF + evaluation

Time to Replace Human with Model – Reward Model

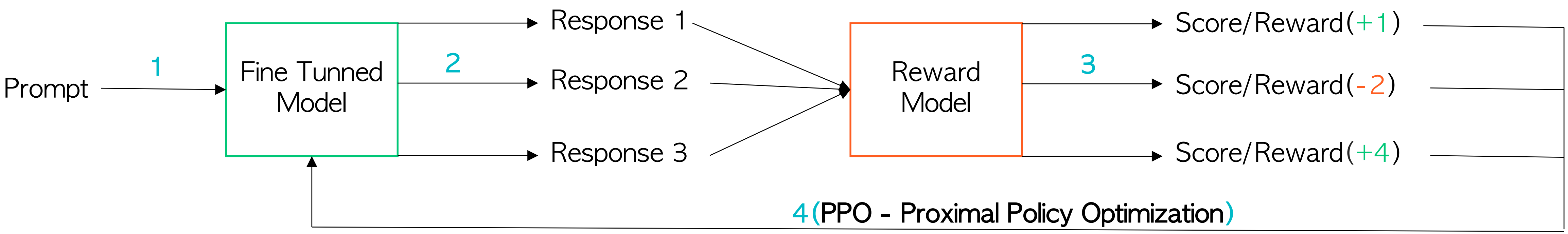


Reinforcement Learning

After **Supervised Fine-Tuning (SFT)**, the model can follow instructions, but it doesn't yet *understand which answers are better*.

Reinforcement Learning fixes that by teaching the model through **feedback and scoring** — i.e., "rewarding good answers" and "penalizing bad ones."

This process is called **Reinforcement Learning from Human Feedback (RLHF)**.



2

•The Fine-Tuned (SFT) model generates **multiple candidate answers** for the same prompt.
•(This is usually done by sampling the model several times with slight randomness in token generation.)

3

Each generated response is passed to the **Reward Model** — which has already been trained on human preference data.
The Reward Model outputs a **numerical score** or "reward" that represents how *good* or *aligned* each answer is with human expectations.

Response 1 → +1 (okay answer)
Response 2 → -2 (incorrect or unsafe)
Response 3 → +4 (best answer)

4

The scores from the Reward Model are fed back to the **Fine-Tuned Model**.
Using the **PPO algorithm**, the model's internal weights are updated to:

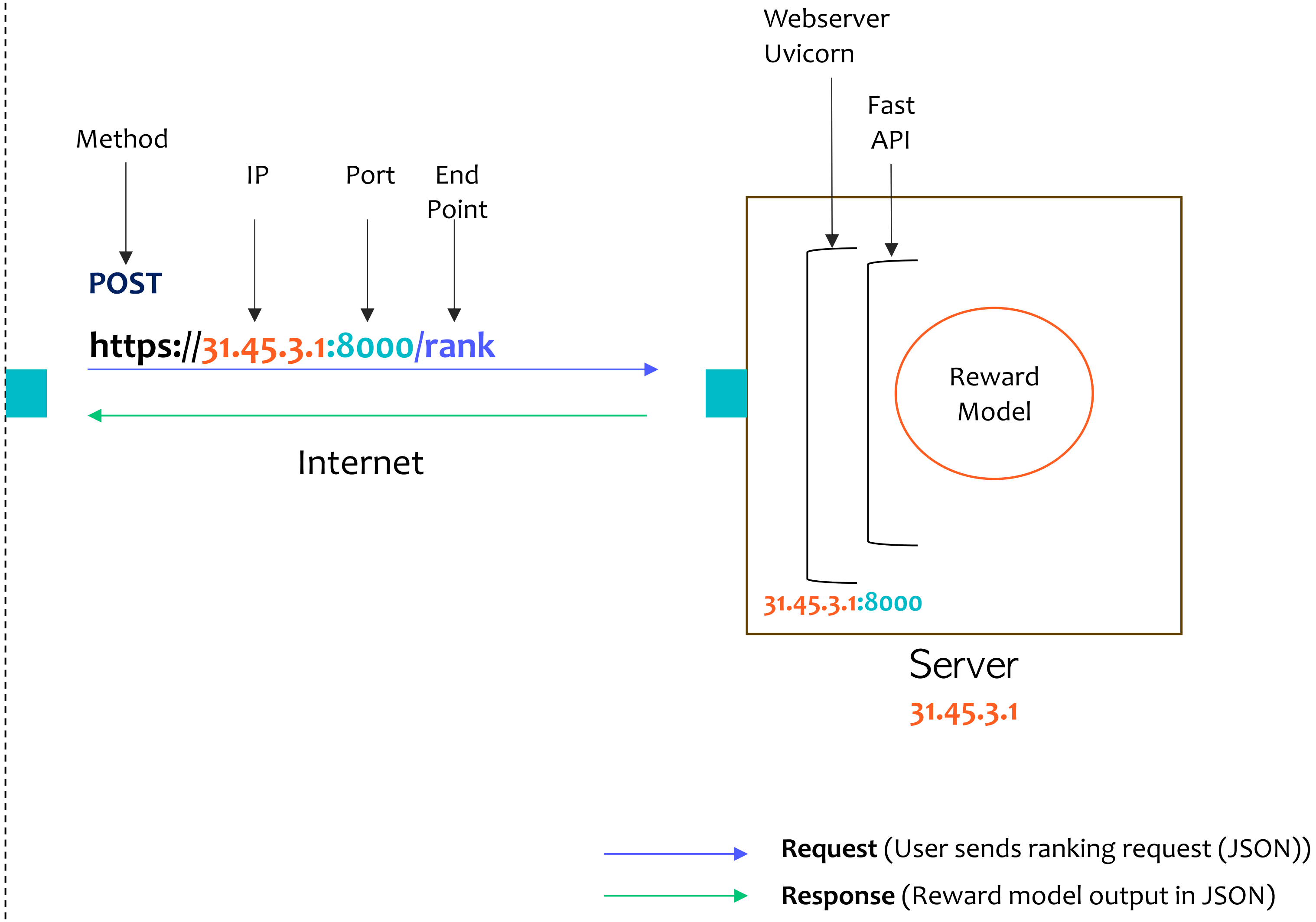
- Increase the likelihood of producing high-reward responses.
- Decrease the likelihood of producing low-reward ones.

Let's look at Reward Model in Action

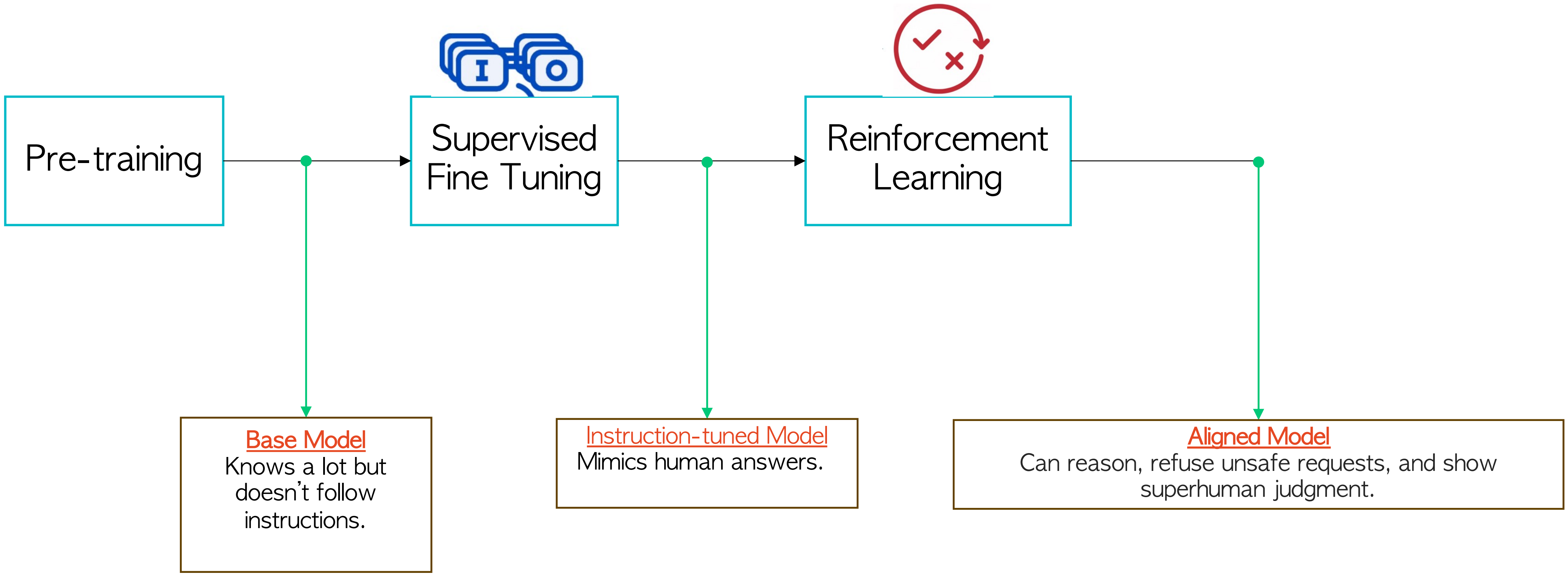
Client - Postman


```json
{
  "prompt": "What is a credit score?",
  "responses": [
    "A credit score is a number that represents your creditworthiness.",
    "It is your bank balance.",
    "A financial indicator used by lenders to decide loan eligibility.",
    "I think it's something about credit cards."
  ]
}
```

```json
{
    "prompt": "What is a credit score?",
    "ranked_responses": [
        {
            "rank": 1,
            "response": "A credit score is a number that represents your creditworthiness.",
            "score": 2.9414
        },
        {
            "rank": 2,
            "response": "A financial indicator used by lenders to decide loan eligibility.",
            "score": -1.7938
        },
        {
            "rank": 3,
            "response": "It is your bank balance.",
            "score": -4.7884
        },
        {
            "rank": 4,
            "response": "I think it's something about credit cards.",
            "score": -5.9865
        }
    ],
    "timestamp": "2025-11-15T15:01:58.466022"
}
```

[redacted]



Pre-training
Supervised Fine Tuning
I
O
Reinforcement Learning
Base Model
Knows a lot but doesn't follow instructions.
Instruction-tuned Model
Mimics human answers.
Aligned Model
Can reason, refuse unsafe requests, and show superhuman judgment.

Let's Looks SFT and RFL in action

Art of Prompt Engineering

Prompt Engineering has become the bridge between human intent and AI output

"High-quality prompt leads to high-quality output"

Prompt → LLM → Generates Output

Context Window | Input tokens | Embeddings | Output tokens

Max tokens can be passed for single request

GPT 5:128k

Tokens	Characters
5	26

Whats the capital of India

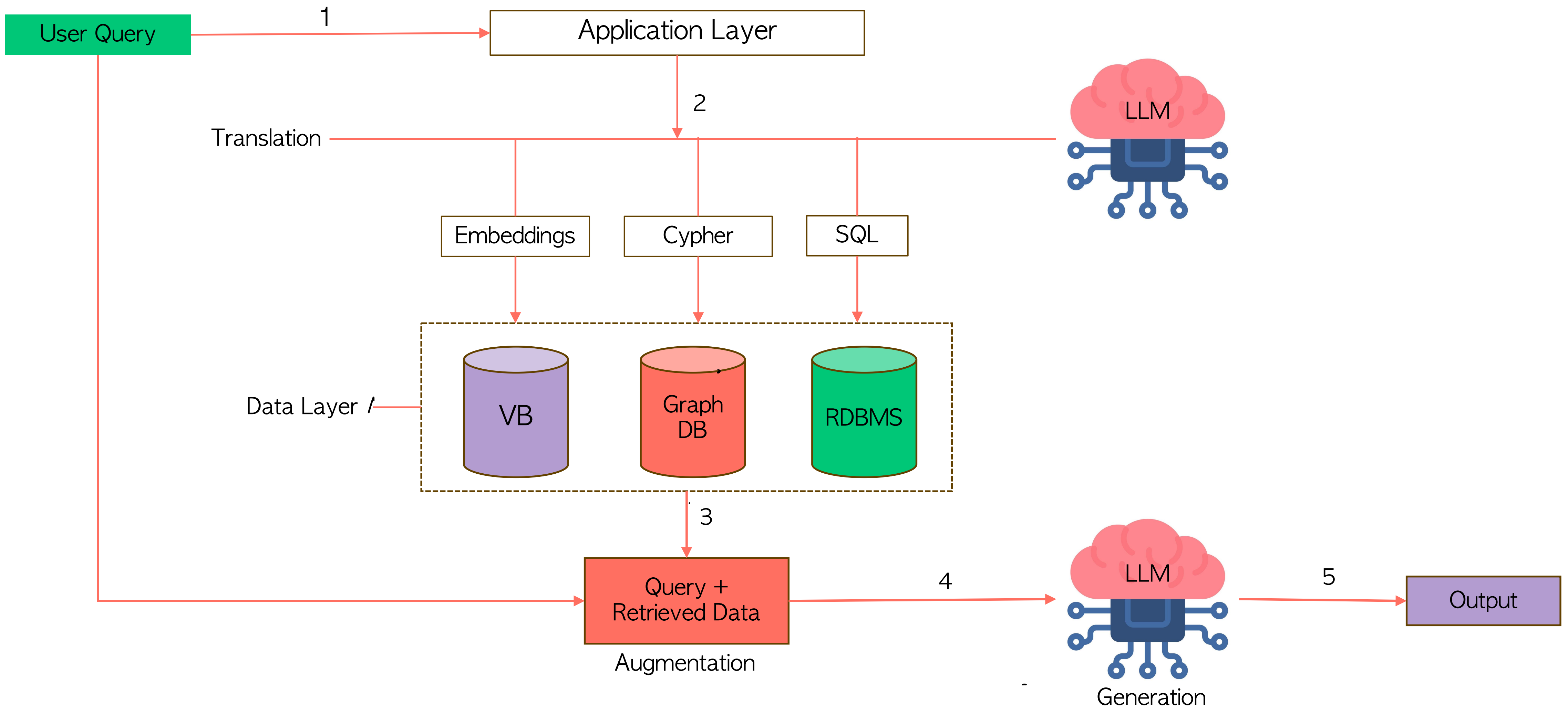
Convert tokens to Vectors(Numerical Representation)

Tokens	Characters
7	30

The capital of India is Delhi.

Retrieval-Augmented Generation (RAG)

LLMs haven't seen your data — in fact, around 95% of enterprise or personal data resides in private sources that are not part of any public training set.

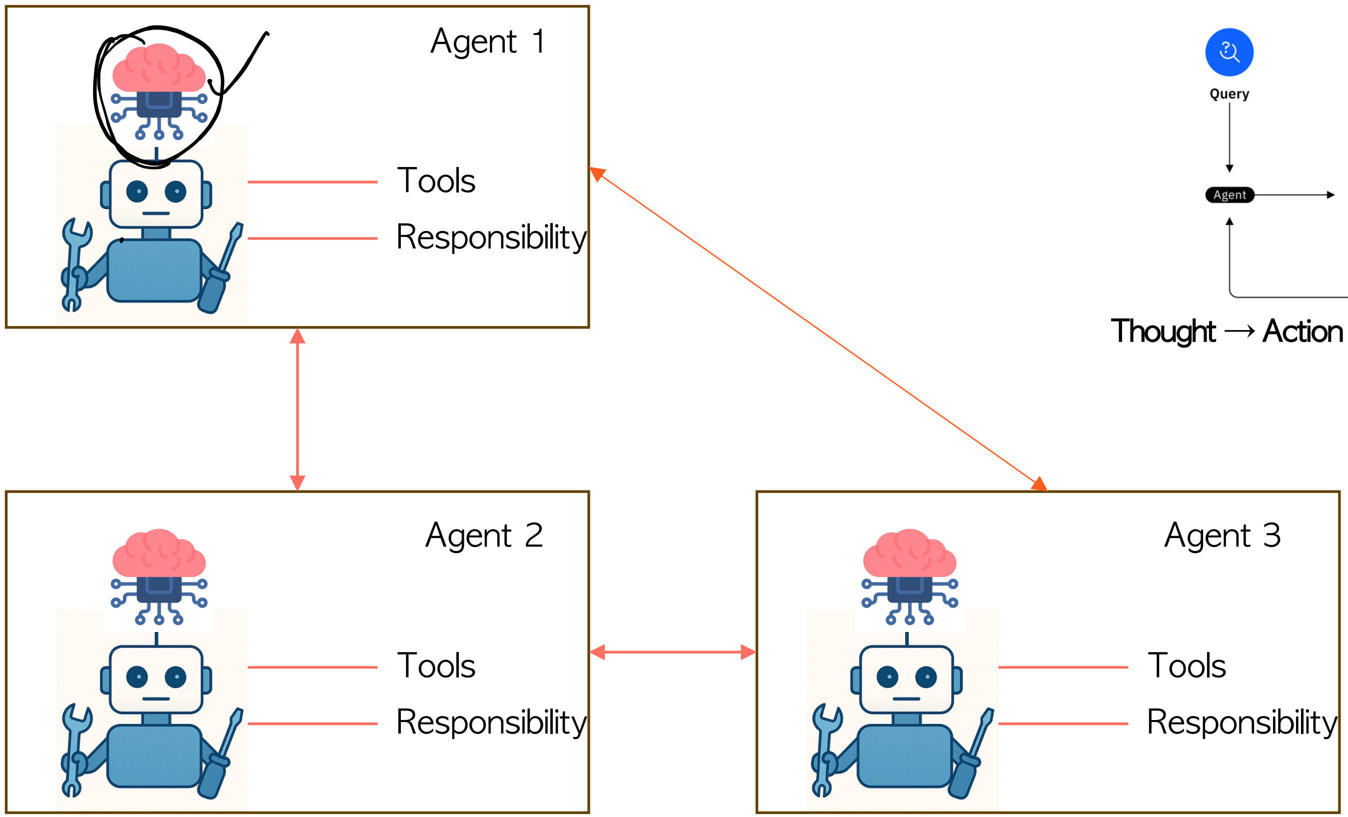


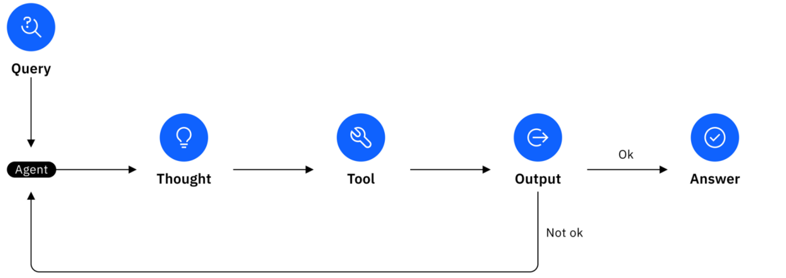
Agentic AI: The Next Leap

From Prompt-Response to Autonomous Reasoning

"Agentic AI is the evolution of Generative AI where models become **active decision-makers**, not passive responders. They use **reasoning**, **memory**, **tools**, and **collaboration** to complete multi-step objectives without constant human supervision."

An Agent is like an AI assistant that can think through a task, decide what tools to use, and act step-by-step to reach a goal.





Thought → Action → Observation → Next Thought → Final Answer

Context Engineering