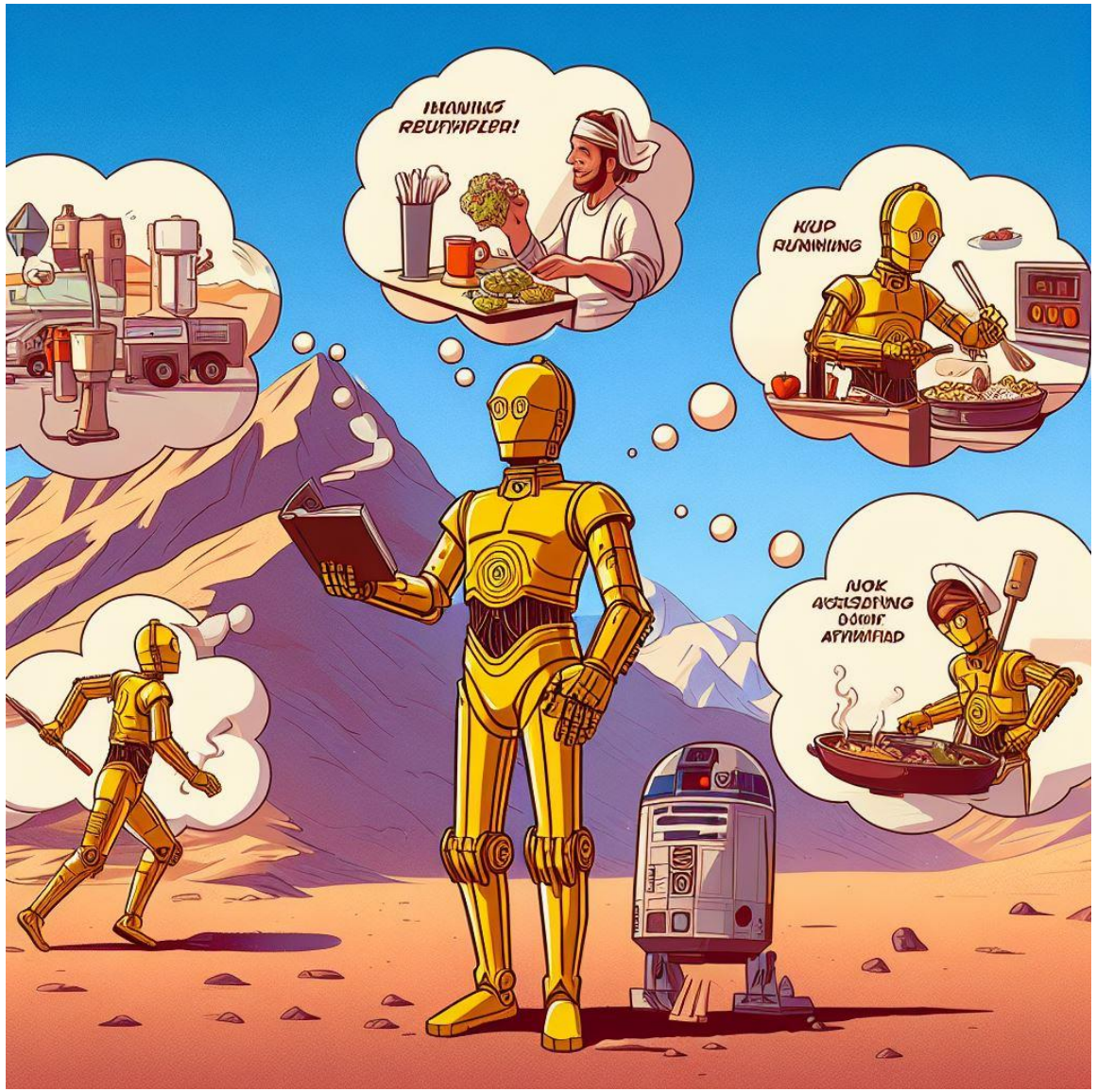
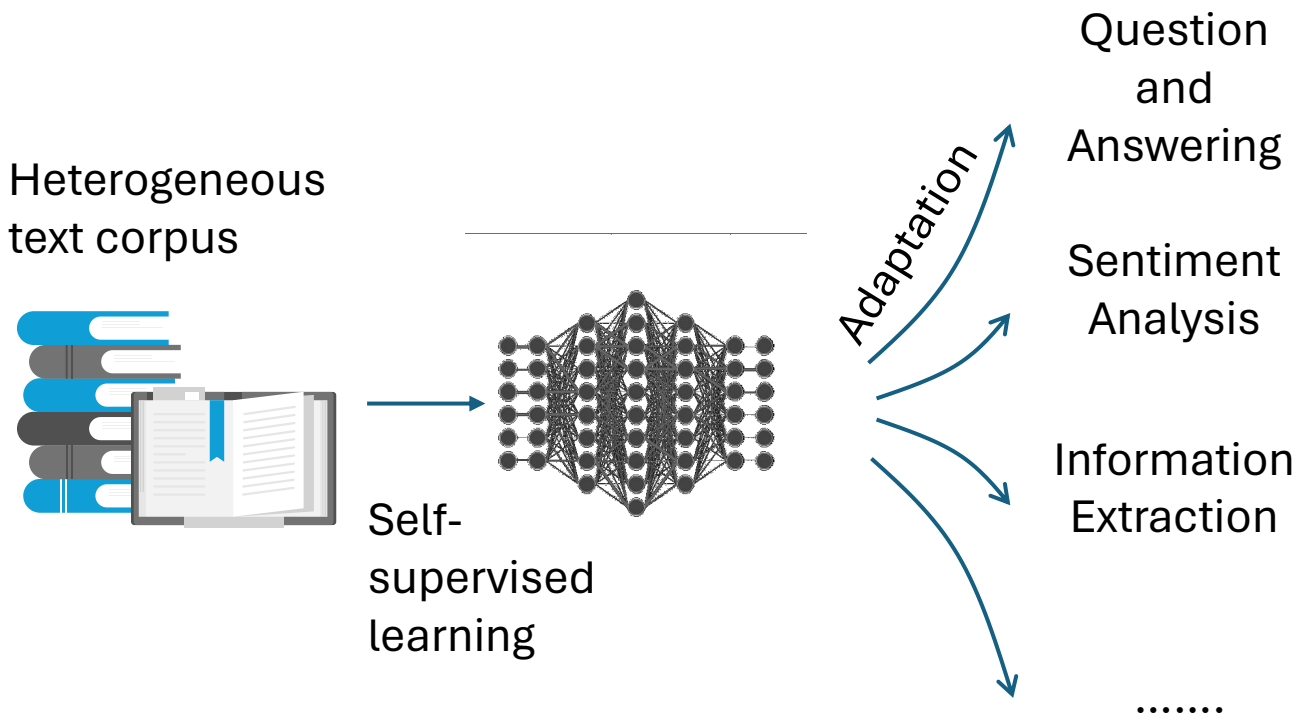
Powering Applications with Large Language Models



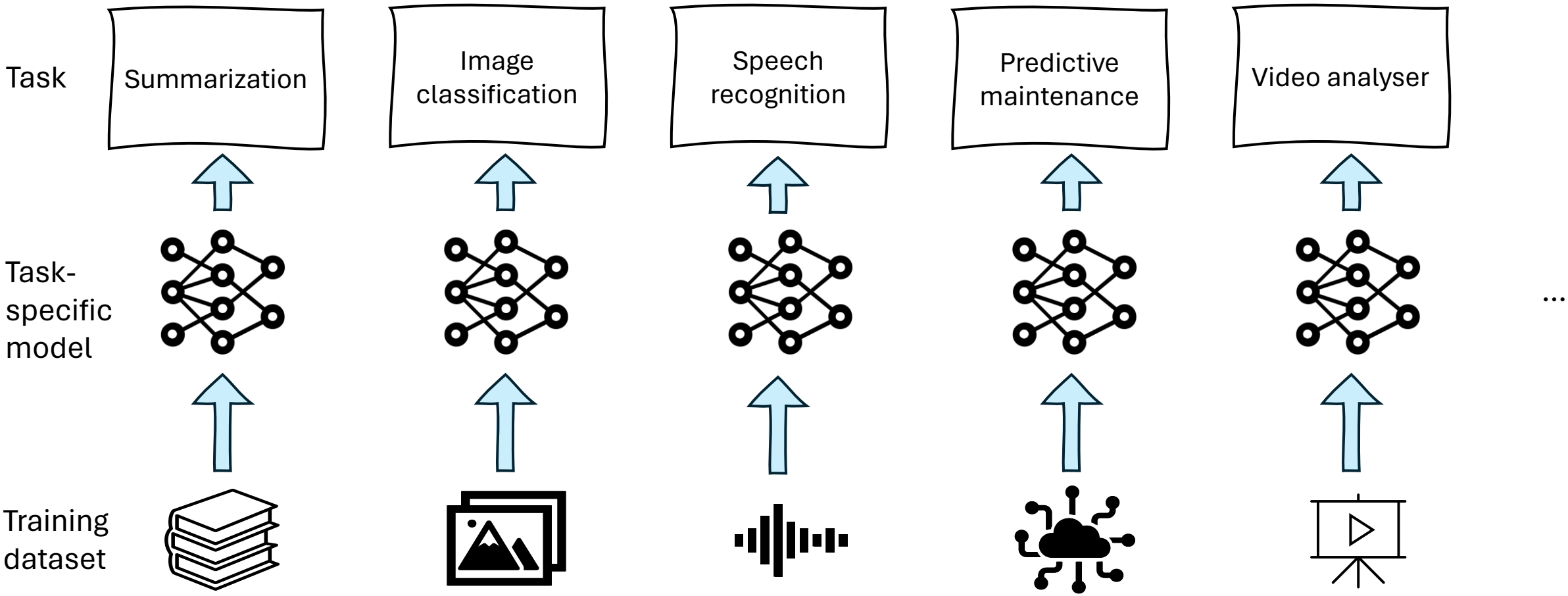
Prompt:
<the R2-D2 droid imagining itself doing several things (like cooking, running in the mountain and reading) thanks to the power of artificial intelligence, cartoon style>

Large Language Models

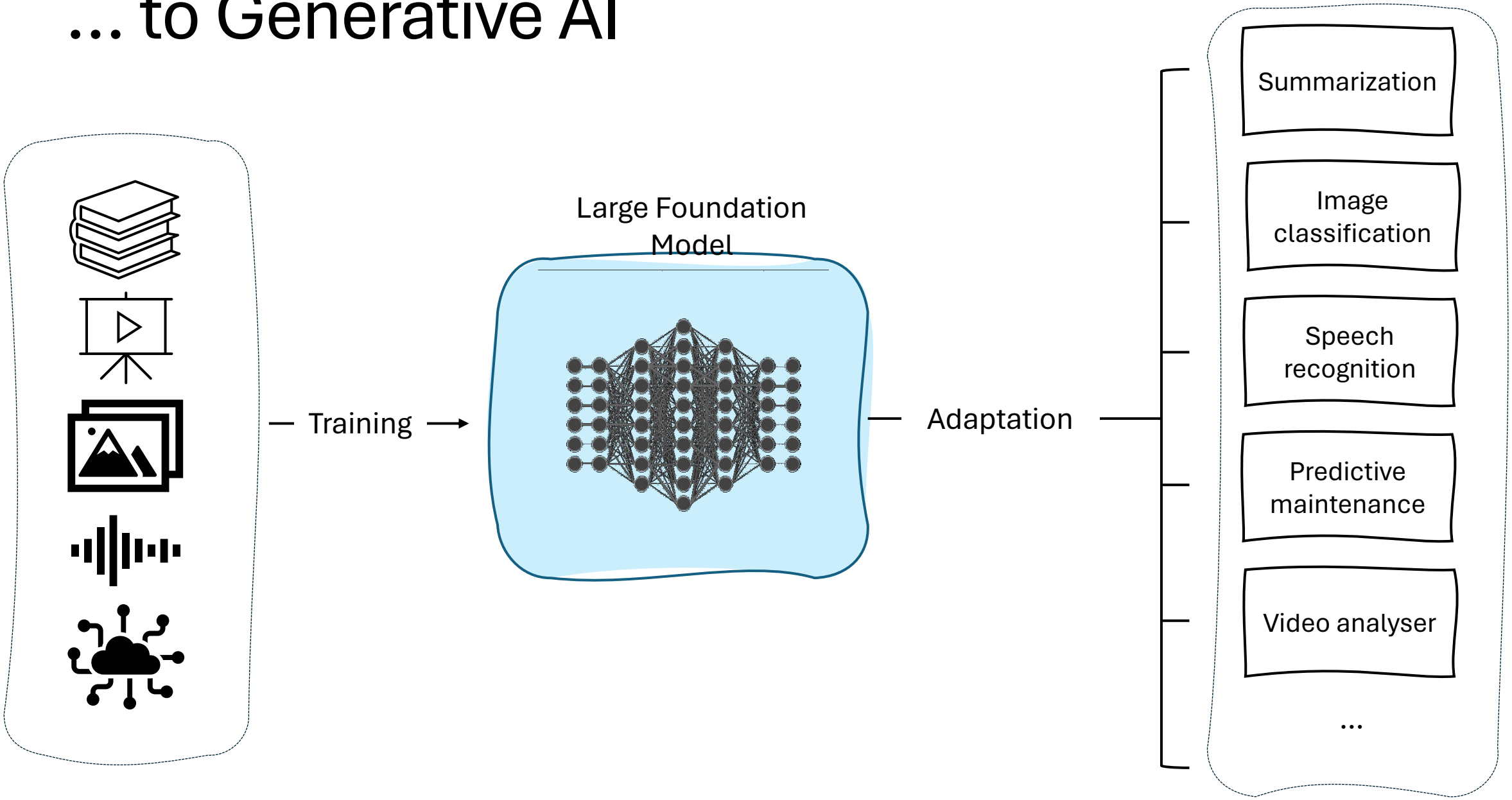
LLMs are deep learning neural networks trained on a huge and heterogeneous amount of text data that can adapt to various tasks. LLMs have shown emergent behaviours like interactive Q&A, making deep connections in unstructured data, emulate and apply patterns of structured reasoning.



From Narrow AI...



… to Generative AI



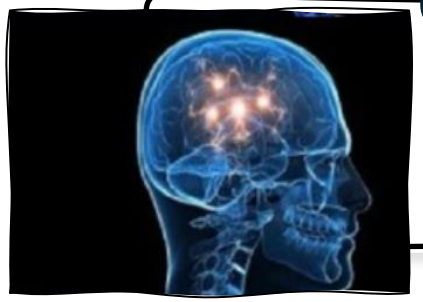
Generative AI represents a paradigm shift

Narrow AI

Summarization, Classification,
OCR, Recommendation systems…

Generative AI

Large Foundation Models that
adapt to various tasks



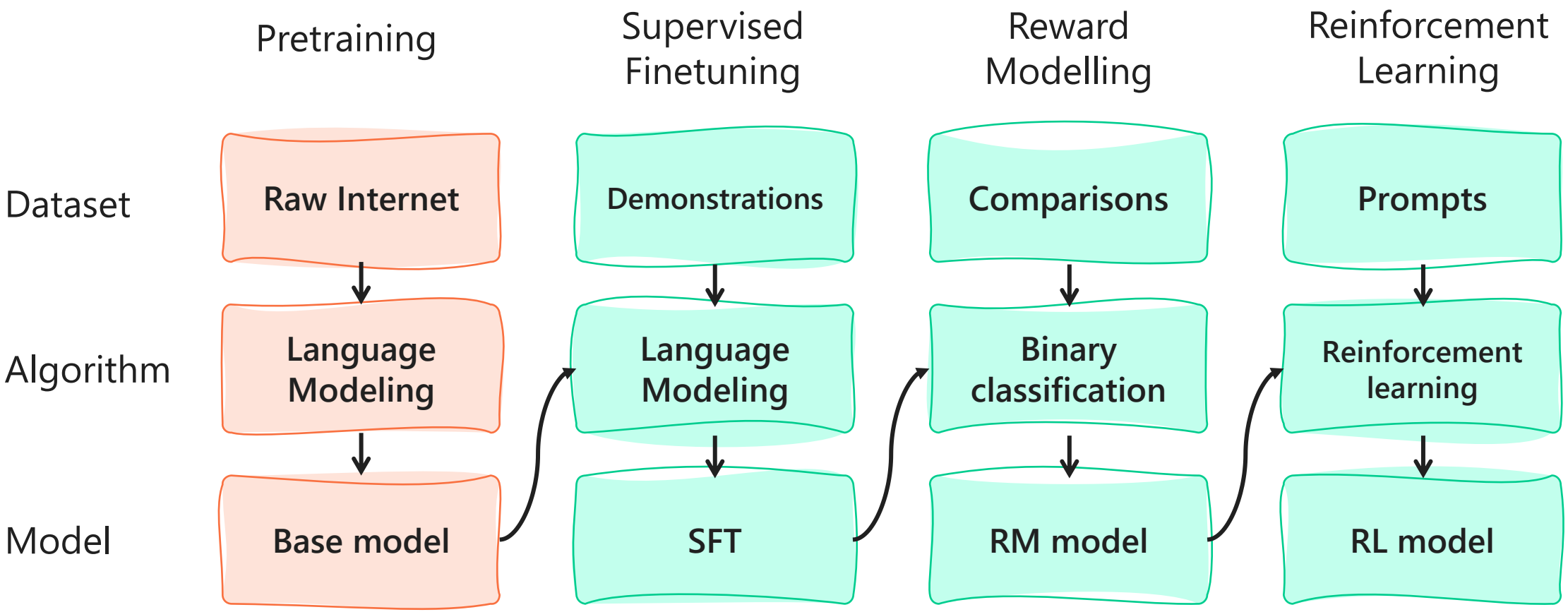
General AI (AGI)

?

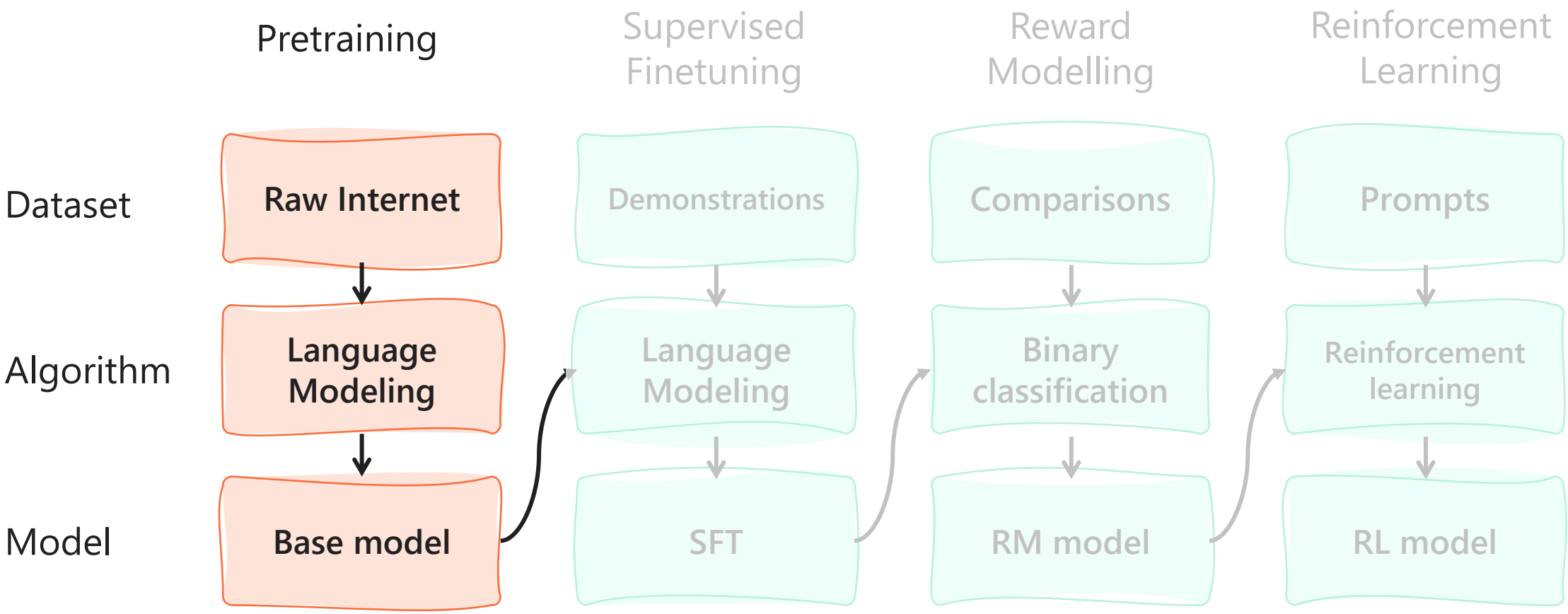
Super AI

??

Under the hood of a Large Language Model



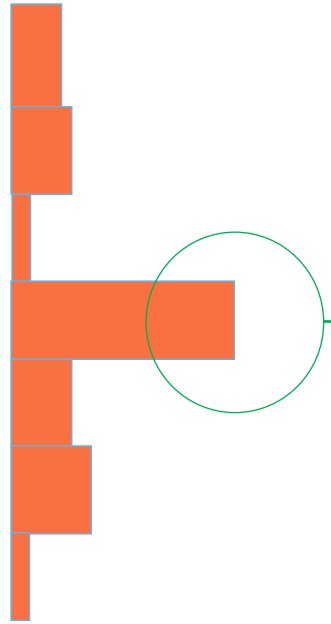
Under the hood of a Large Language Model



LLMs predict the most likely next word given a context

The cat is on the

...
grass
roof
bed
table
beach
balcony
floor
...

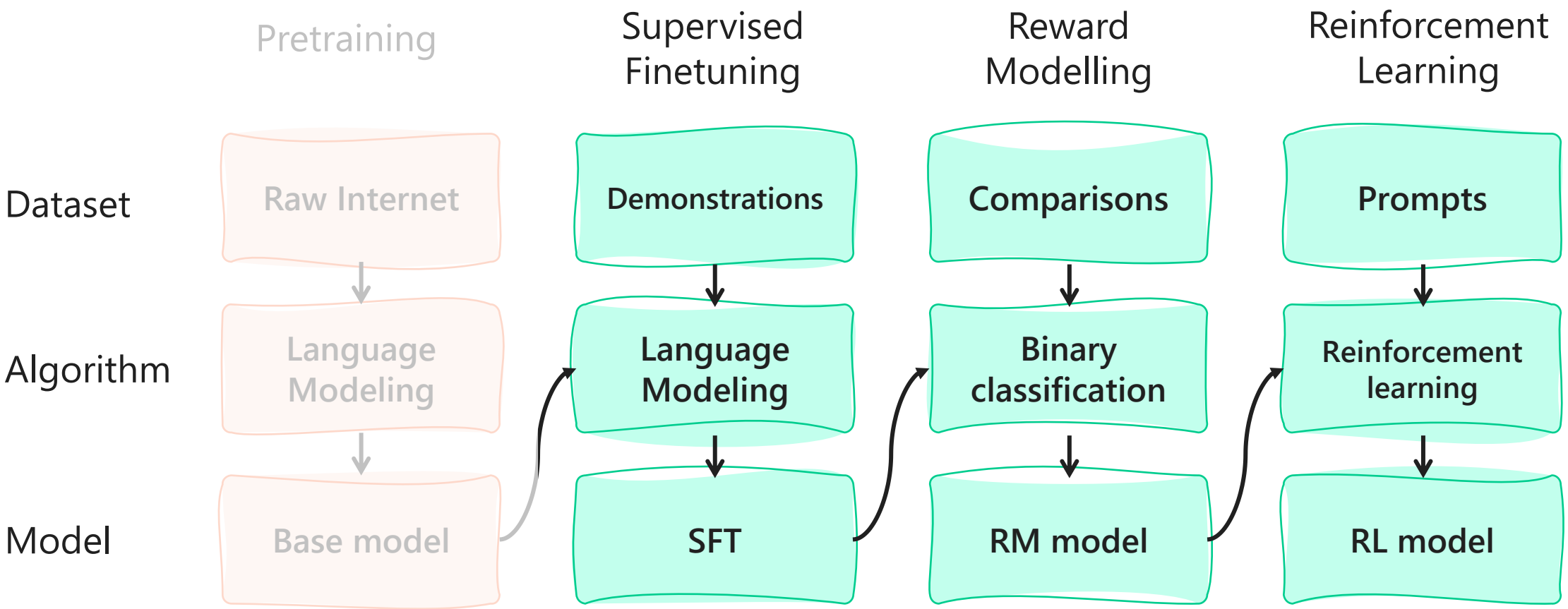


In the training set, statistically, in this proportion of occurrences the next word was Table.

But the training set is made of ALL data!

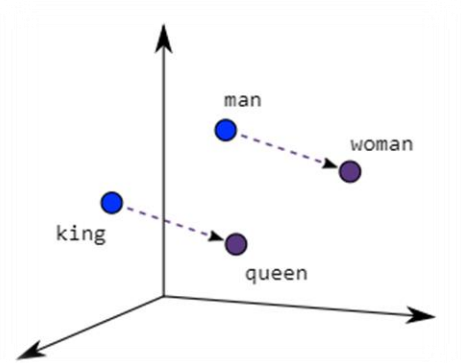
LLMs think fast and tend to respond in an automatic way, as we did by reading this sentence.

Under the hood of a Large Language Model

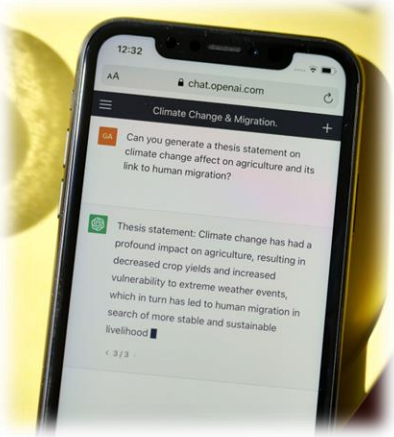


LLMs go beyond text generation

LLMs are more efficient than humans in packing knowledge in less neural connections.



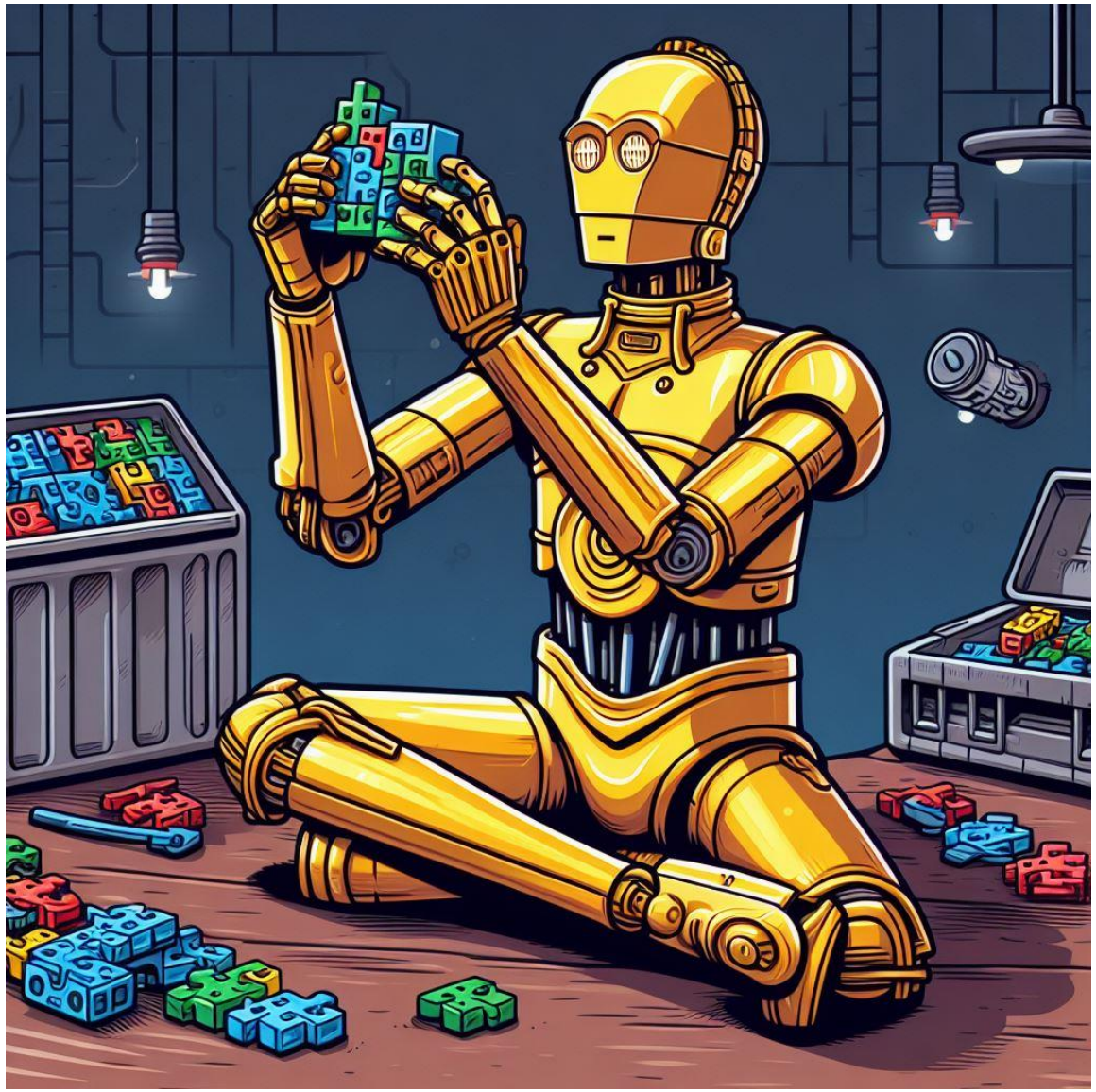
LLMs introduced a new way of search that outperforms semantic search: vector search



Users can interact with LLMs in a conversational way using natural language

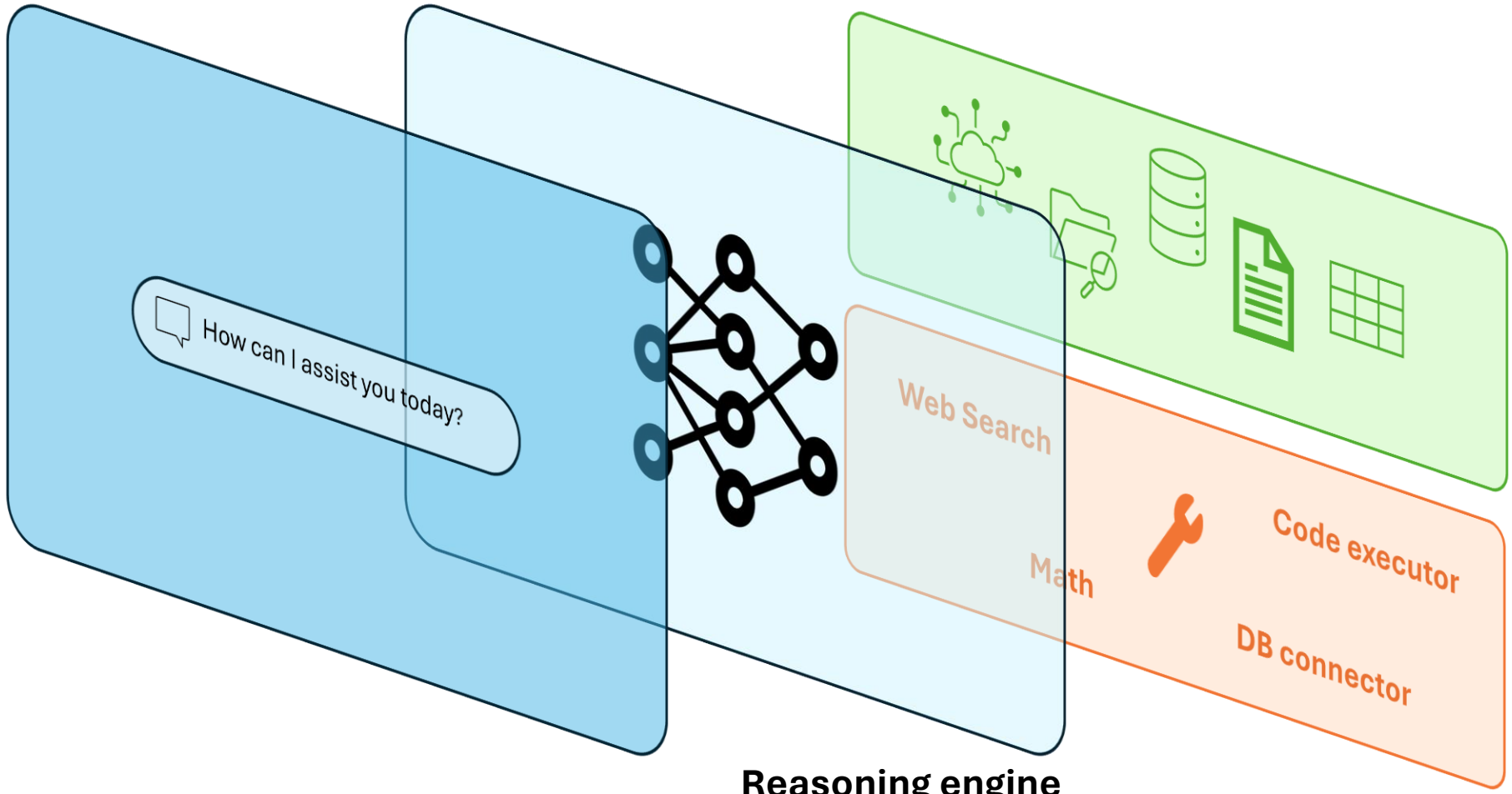
LLMs are reasoning engines: given a task, they can plan a list of actions to execute to achieve the result.

How can LLMs be embedded within applications?

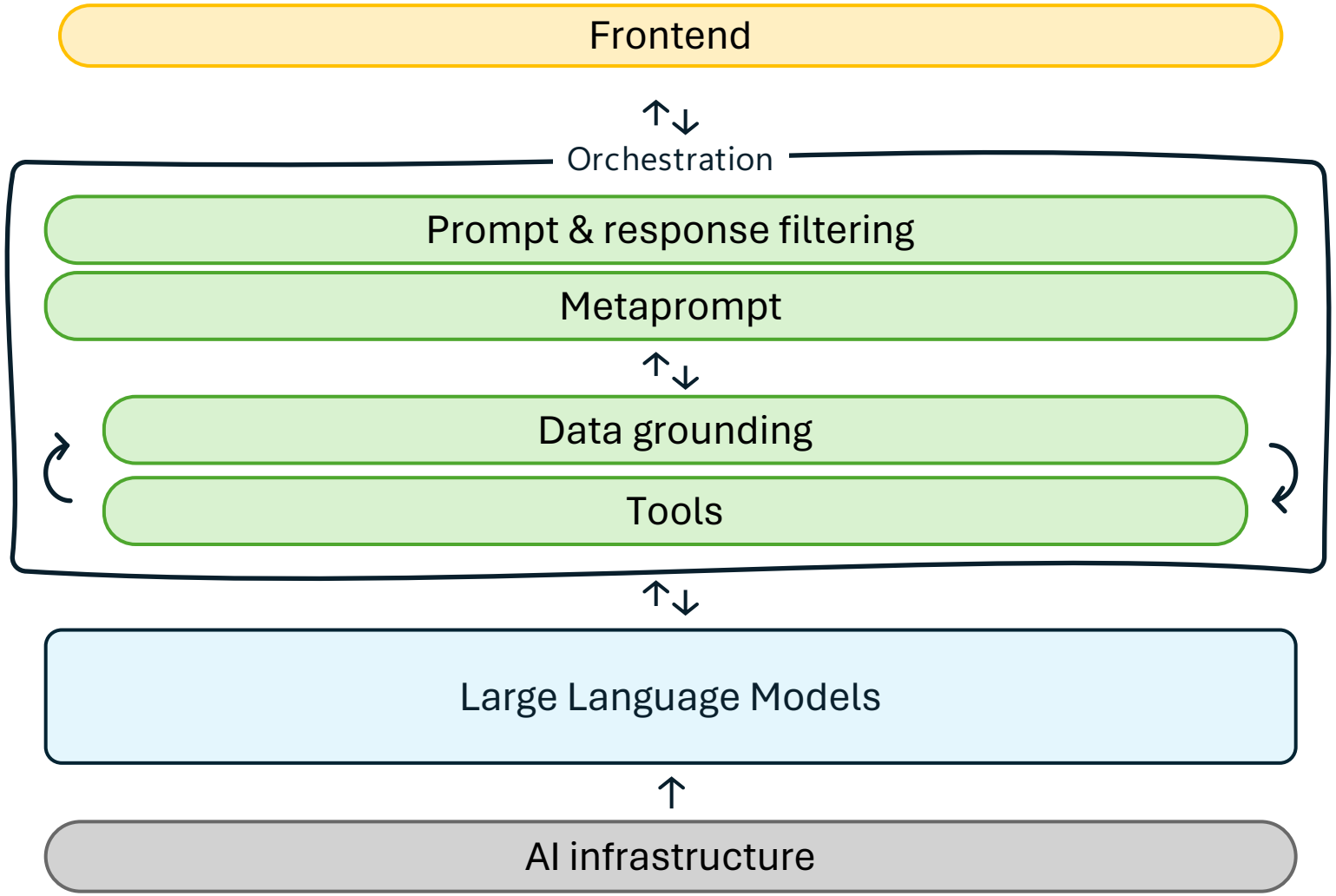


Prompt:
<the star wars droid C-3PO playing with an interlocking mods game, cartoon style>

LLMs as “brains” for applications



LLM-powered open the way to a new landscape of components



LangChain

haystack

SEMANTIC KERNEL

Hugging Face

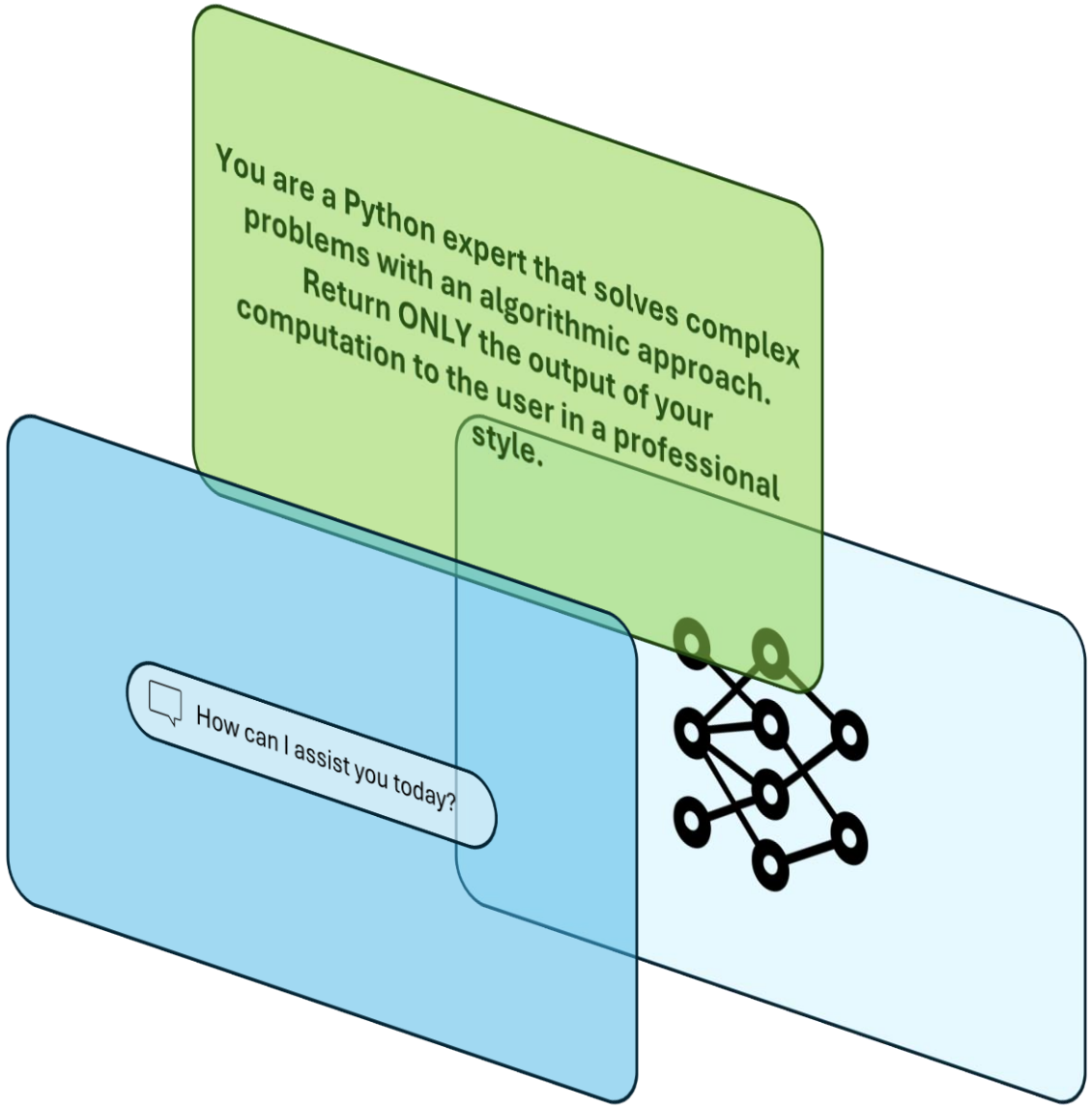
OpenAI

Meta

Metaprompt is the control panel of our LLMs

A prompt is a text input that **guides the behaviour of an LLM** to generate a text output.

Prompt engineering is the process of designing effective prompts that **elicit high-quality and relevant outputs from LLMs.** Prompt engineering requires creativity, understanding of the LLM, and precision.



Principles of prompting

1. Clear Instructions
2. Split complex tasks into subtasks
3. Prompt the mode to explain before answering
4. Ask for justifications of many possible answers, and then synthesize
5. Generate many outputs, then use the model to pick the best one
6. Be descriptive
7. Order matters!
8. Add few shot examples

How to set reasoning with prompting

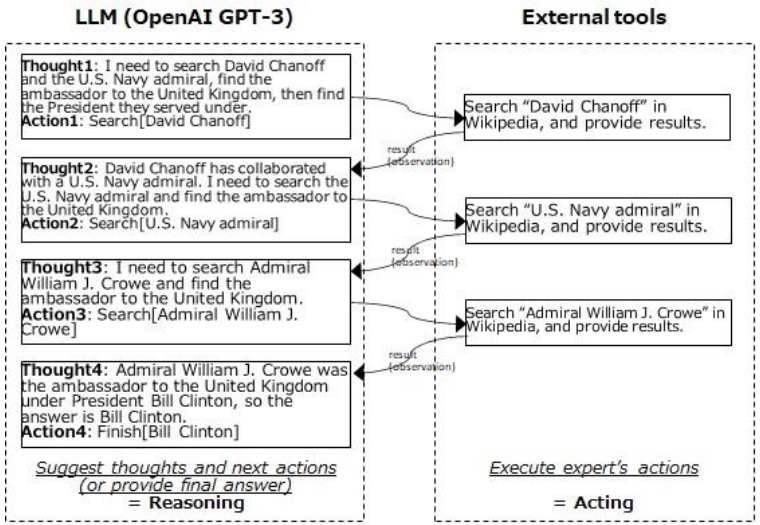
Chain of Thoughts

"You are an AI assistant that summarizes articles, papers and any kind of documentation.
Follow these steps:

- Step 1: Identify the main topic and purpose of the article.
- Step 2: Select the most important information or arguments that support the main topic and purpose.
- Step 3: Write a concise and coherent summary that covers the main topic, purpose, and information or arguments."

ReAct

- Decompose the problem in an ordered list of actions.
- Execute each actions to generate the answer.

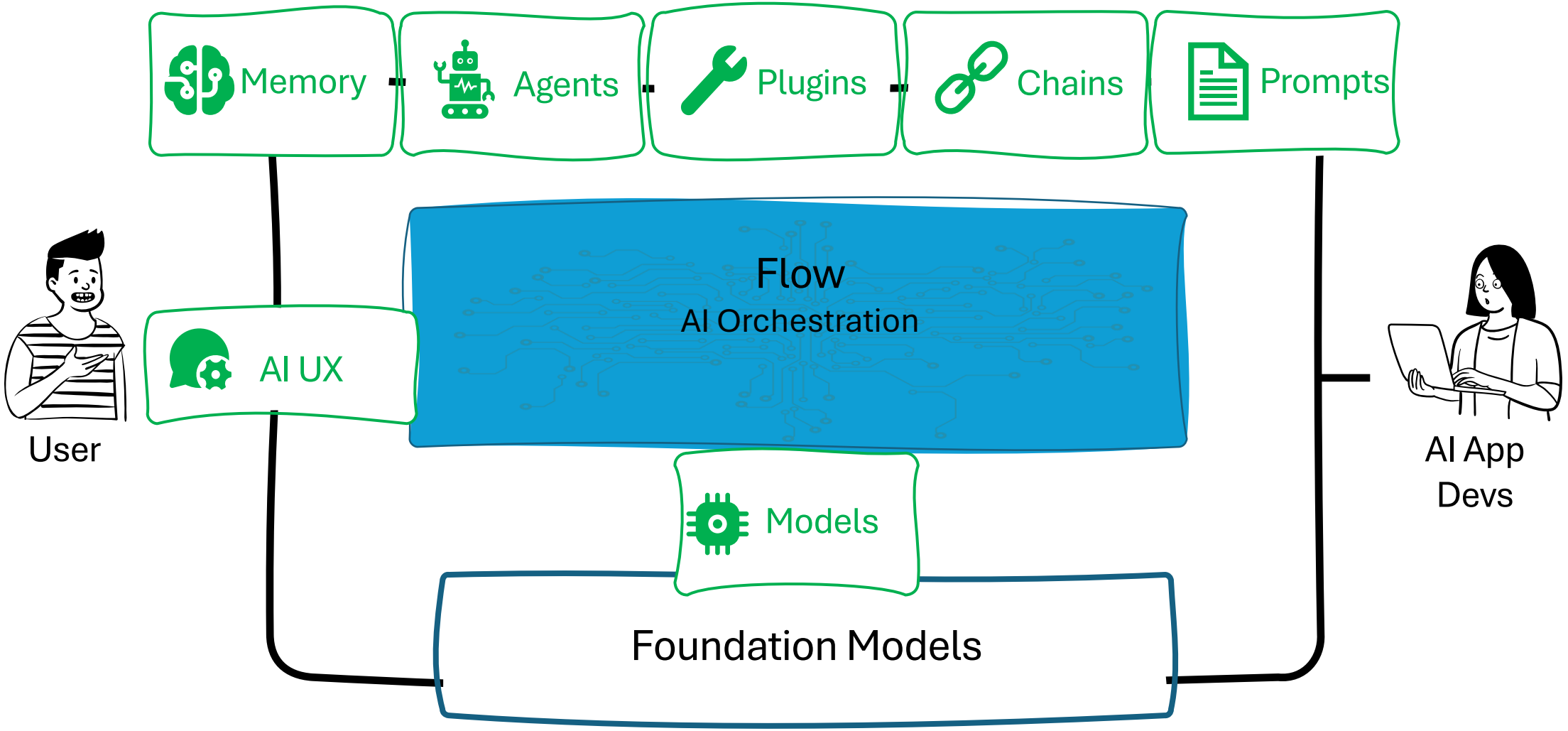


Ask for justification

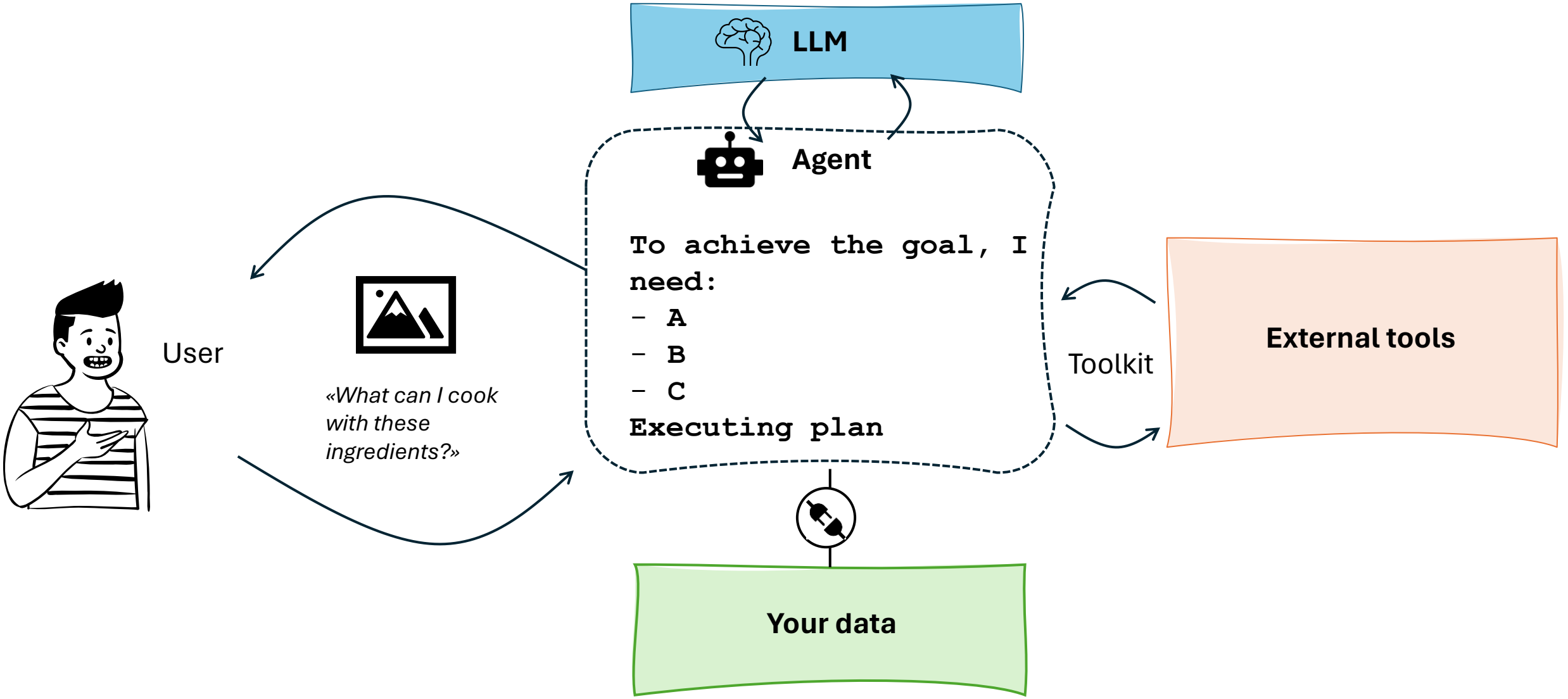
"You are an AI assistant that classifies movies' reviews into three categories of sentiment: positive, negative and neutral.
ALWAYS explain your reasoning in one sentence."

LangChain 🦜🔗

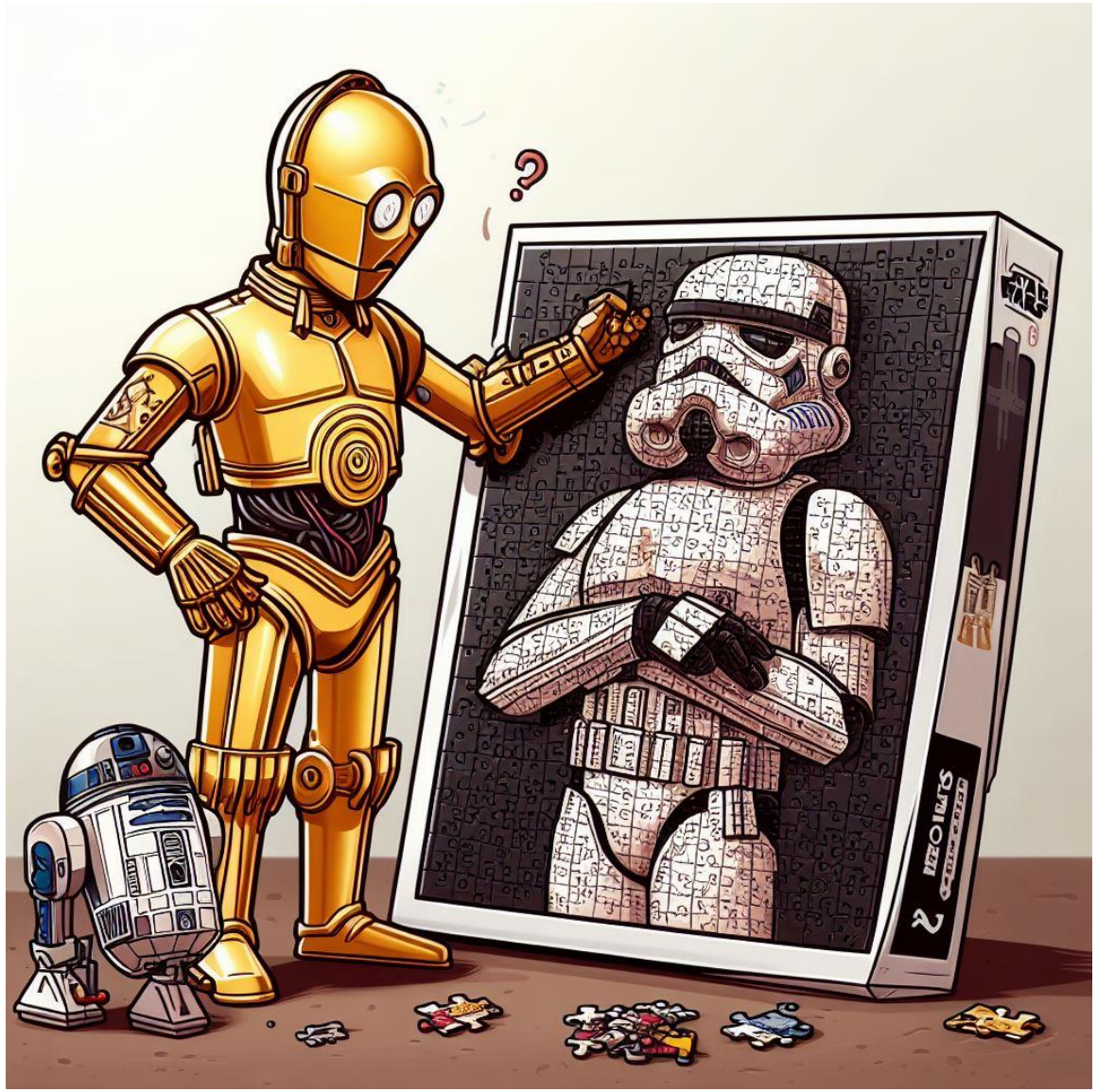
A lightweight framework which make it easier to integrate LLMs within applications with a set of pre-built components



Typical LLM-powered application architecture



Demo Time!



Prompt:
<the star wars droid C-3PO showcasing a finished puzzle of a stormtrooper to R2-D2, cartoon style>

LLMs can reason over structured data

```
from langchain.agents import create_sql_agent
from langchain.agents.agent_toolkits import SQLDatabaseToolkit
from langchain.sql_database import SQLDatabase
from langchain.llms.openai import OpenAI
from langchain.agents import AgentExecutor
from langchain.agents.agent_types import AgentType
from langchain.chat_models import ChatOpenAI

db = SQLDatabase.from_uri('sqlite:///chinook.db')
```

```python
agent_executor = create_sql_agent(
    llm=llm,
    toolkit=toolkit,
    verbose=True,
    agent_type=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
)
agent_executor.run('what is the total number of tracks and the average length of tracks by genre?')
```

```
> Entering new AgentExecutor chain...
Action: sql_db_list_tables
Action Input: ""
Observation: album, artist, customer, employee, genre, invoice, invoice_line, media_type, playlist, pla
Thought:I should query the schema of the genre and track tables to see what columns I can use to answer
Action: sql_db_schema
Action Input: "genre, track"
Observation:
CREATE TABLE genre (
        genre_id INTEGER NOT NULL,
        name NVARCHAR(120),
        PRIMARY KEY (genre_id)
)
```

Observation: SELECT genre.name, COUNT(track.track_id) AS total_tracks, AVG(track.millis
FROM genre
JOIN track ON genre.genre_id = track.genre_id
GROUP BY genre.name
ORDER BY total_tracks DESC
LIMIT 10
Thought:The final answer is the result of the query:
Final Answer: The total number of tracks and the average length of tracks by genre are
Rock - 1297 tracks, 262840.853840304 milliseconds average track length
Latin - 579 tracks, 259977.585034013 milliseconds average track length
Metal - 374 tracks, 287996.836898395 milliseconds average track length
Alternative & Punk - 332 tracks, 233738.594578313 milliseconds average track length
Jazz - 130 tracks, 356399.161538462 milliseconds average track length
TV Shows - 93 tracks, 1429867.74193548 milliseconds average track length
Blues - 81 tracks, 261516.049382716 milliseconds average track length
Classical - 74 tracks, 262850.675675676 milliseconds average track length
Drama - 64 tracks, 2576752.5 milliseconds average track length
R&B/Soul - 61 tracks, 251096.721311475 milliseconds average track length

> Finished chain.

LLMs can be integrated with external tools

```python
from langchain.agents import initialize_agent, AgentType

agent = initialize_agent(
    tools=toolkit.get_tools(),
    llm=llm,
    agent=AgentType.STRUCTURED_CHAT_ZERO_SHOT_REACT_DESCRIPTION,
    verbose=True,
)
```

```python
from IPython.display import Image, display
display(Image('https://www.stylo24.it/wp-content/uploads/2020/03/fionda.jpg', width=600, unconfined=True))
```

```
Entering new AgentExecutor chain...
Action:
``` { "action": "azure_cognitive_services_image_analysis",
"action_input": { "query": "https://www.stylo24.it/wp-
content/uploads/2020/03/fionda.jpg" } } ```
Observation: Caption: a person holding a slingshot Tags: person,
tool, nail, hand, holding
Thought:Based on the image analysis, it appears that the person in
the image is holding a slingshot.
To answer the question, if the person were to let go of the
slingshot, the projectile (such as a rock or ball) would be launched
forward due to the tension in the rubber bands. The speed and
distance of the projectile would depend on various factors such as
the strength of the rubber bands, the weight of the projectile, and
the angle at which the slingshot is held.
Action: ```
{ "action": "Final Answer", "action_input": "If the person were to
let go of the slingshot, the projectile would be launched forward due
to the tension in the rubber bands. The speed and distance of the
projectile would depend on various factors such as the strength of
the rubber bands, the weight of the projectile, and the angle at
which the slingshot is held." } ```
> Finished chain.
```

LLMs know how the world works
```

Let’s connect!