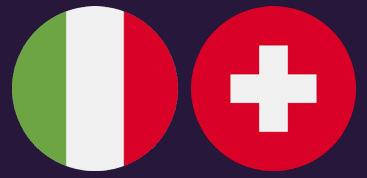
RAG And AI for the data professional

Emanuele Meazzo - 2024

WHO IS YOUR SPEAKER TODAY?

EMANUELE MEAZZO



- **I can probably consider myself as an old-time speaker by now 👴**
- **Manager Data Science & Intelligence @ Acer EMEA**
- **SQL Server, Azure SQL, Synapse, Data Lake, Power BI, Azure AI, AI, AI, AI etc..**
- **Ducks 🦆**

linkedin.com/in/emanuelemeazzo

youtube.com/@emaops

tsql.tech

AGENDA

01 What change made the hype go up?

02 What’s a foundation model

03 Work with Ai and your data

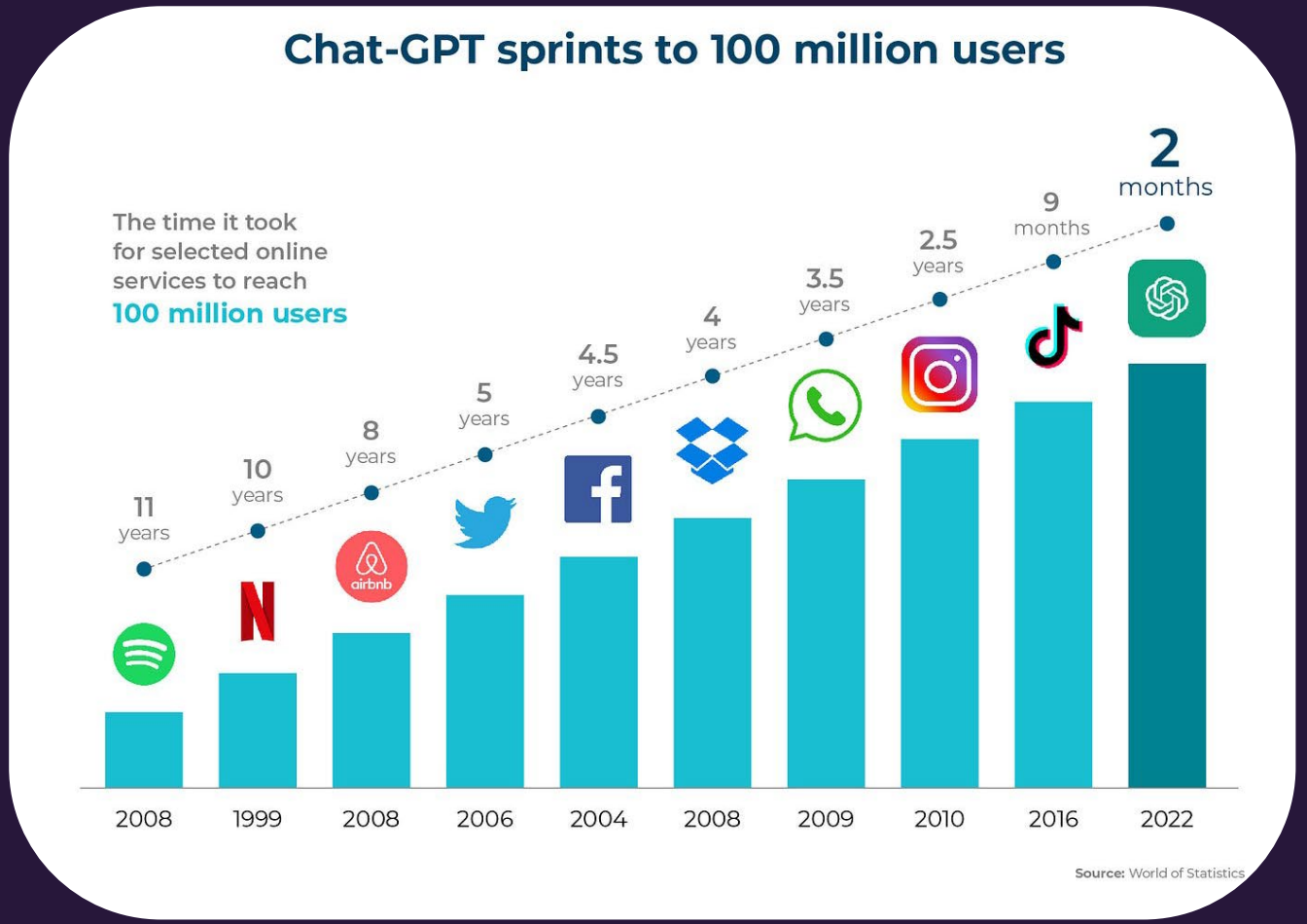
04 RAG

05 Function Calling

06 Typical hurdles of AI

What has changed in AI that made the hype go up lately?

What has changed in AI that made it go up the hype cycle?



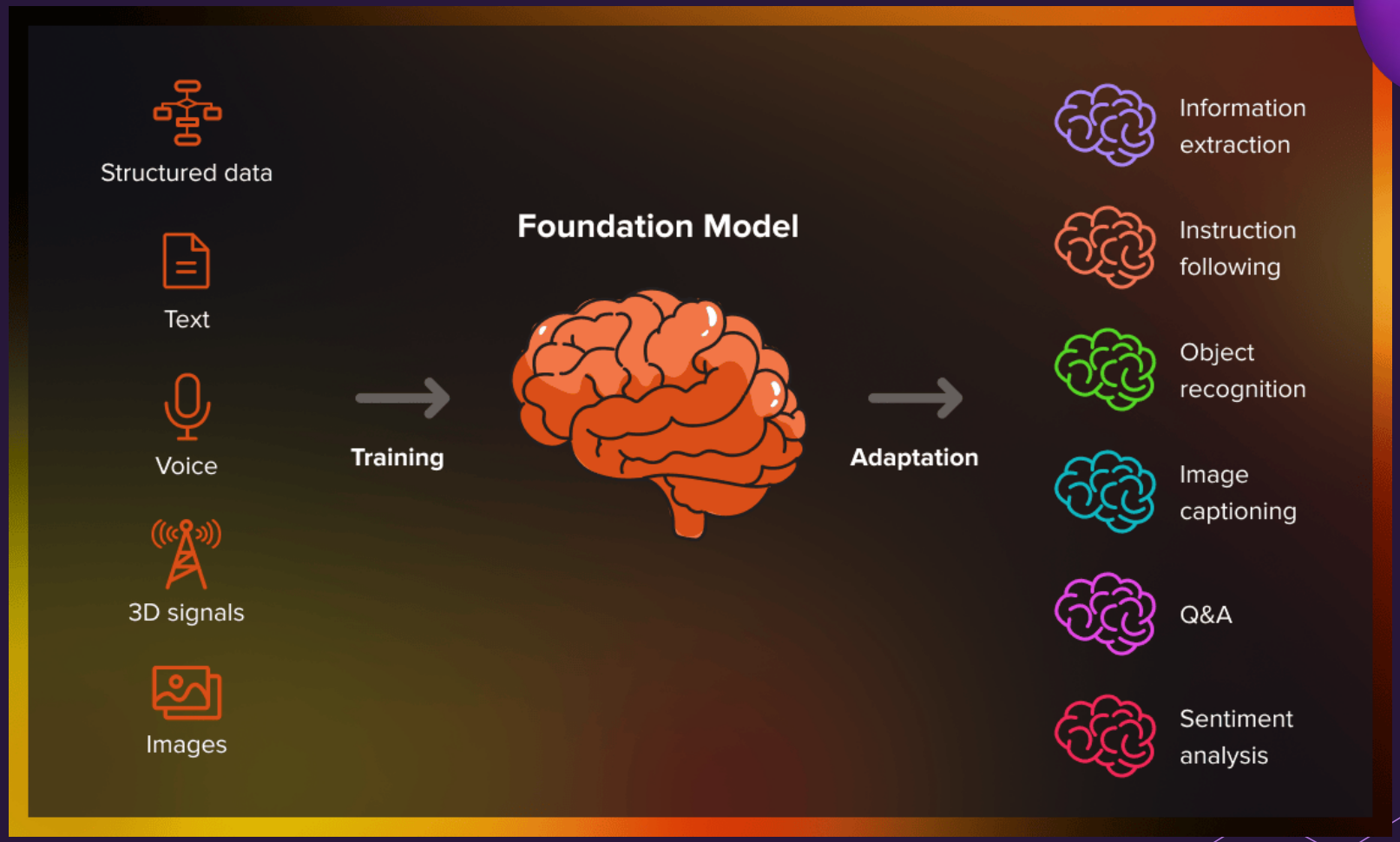
Driving the hype: LLM

- **ChatGPT** is a (service based on a) LLM
- LLM: **Large Language Model**
- GPT: Generative **Pretrained** Transformer
- Transformer: first described in famous "**Attention is all you need**" Google Paper in 2017
- Since ChatGPT has demonstrated the viability of LLM as a **successful commercial product**, the race to create the best model has started
- Now he have lots and lots of what we call ***foundation model***

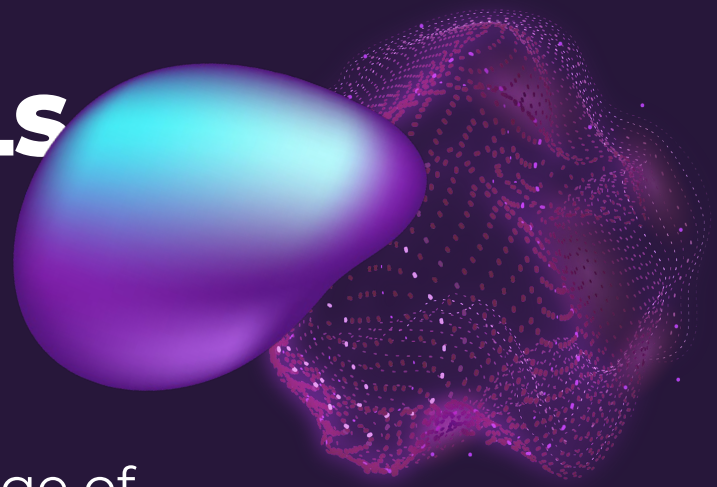
01

WHAT'S A FOUNDATION MODEL?

FOUNDATION MODELS



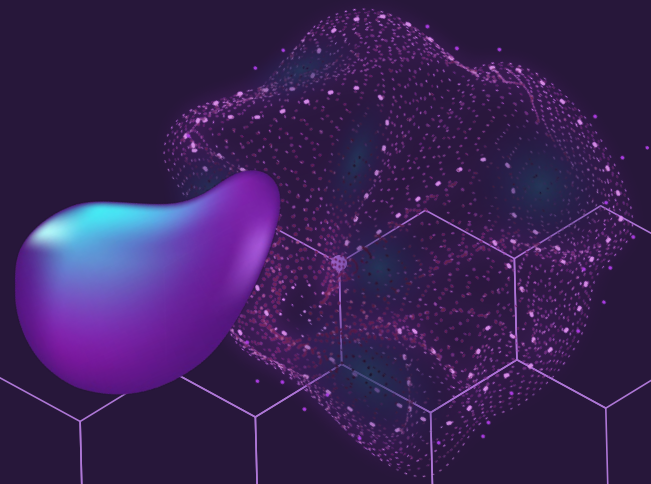
FOUNDATION MODELS



·**Large-Scale Pretraining:** Trained on massive amounts of diverse text data from the internet or other sources.

·**General-Purpose:** Designed to be adaptable to a wide range of tasks without task-specific fine-tuning.

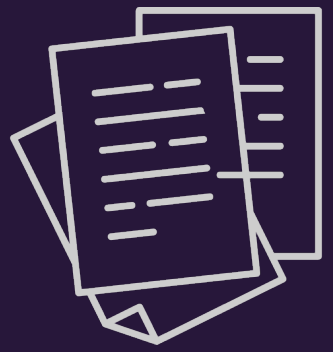
·**Transfer Learning:** Can be fine-tuned on smaller, more specific datasets to excel at particular tasks.



·**Knowledge Representation:** Encodes a broad understanding of language, facts, and common sense reasoning.

·**Zero-Shot/Few-Shot Learning:** Capable of performing tasks they haven't been explicitly trained on with little to no examples.

CUT THE BS PLEASE SLIDE



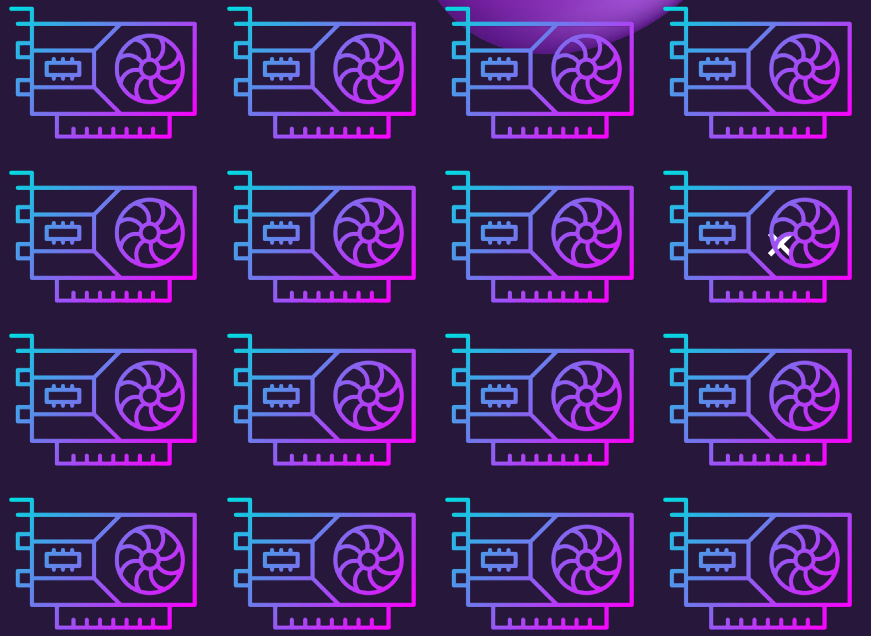
All the text you can use

All the images you can use

All the audio you can use

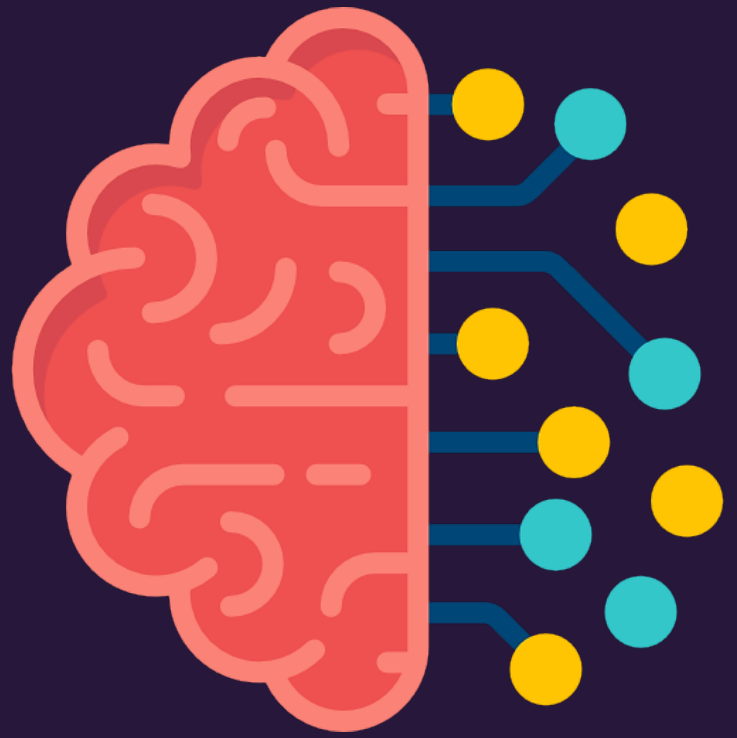
Magic mathematical model
(At the end of the day lots
and lots of matrix operations)

We have no idea how it
works, it just does

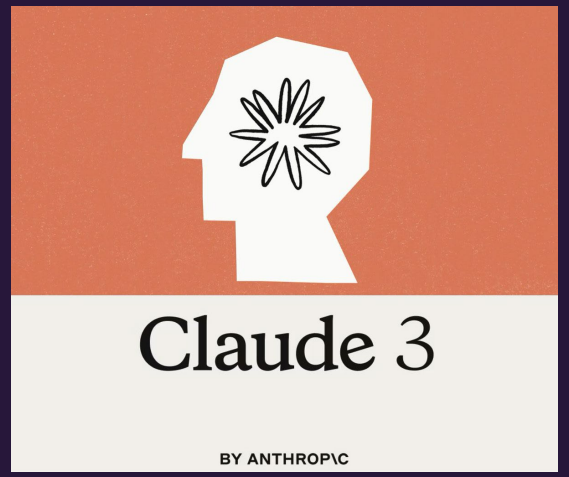


Millions and millions of $ in GPUs and power to run the magic mathematical model for weeks

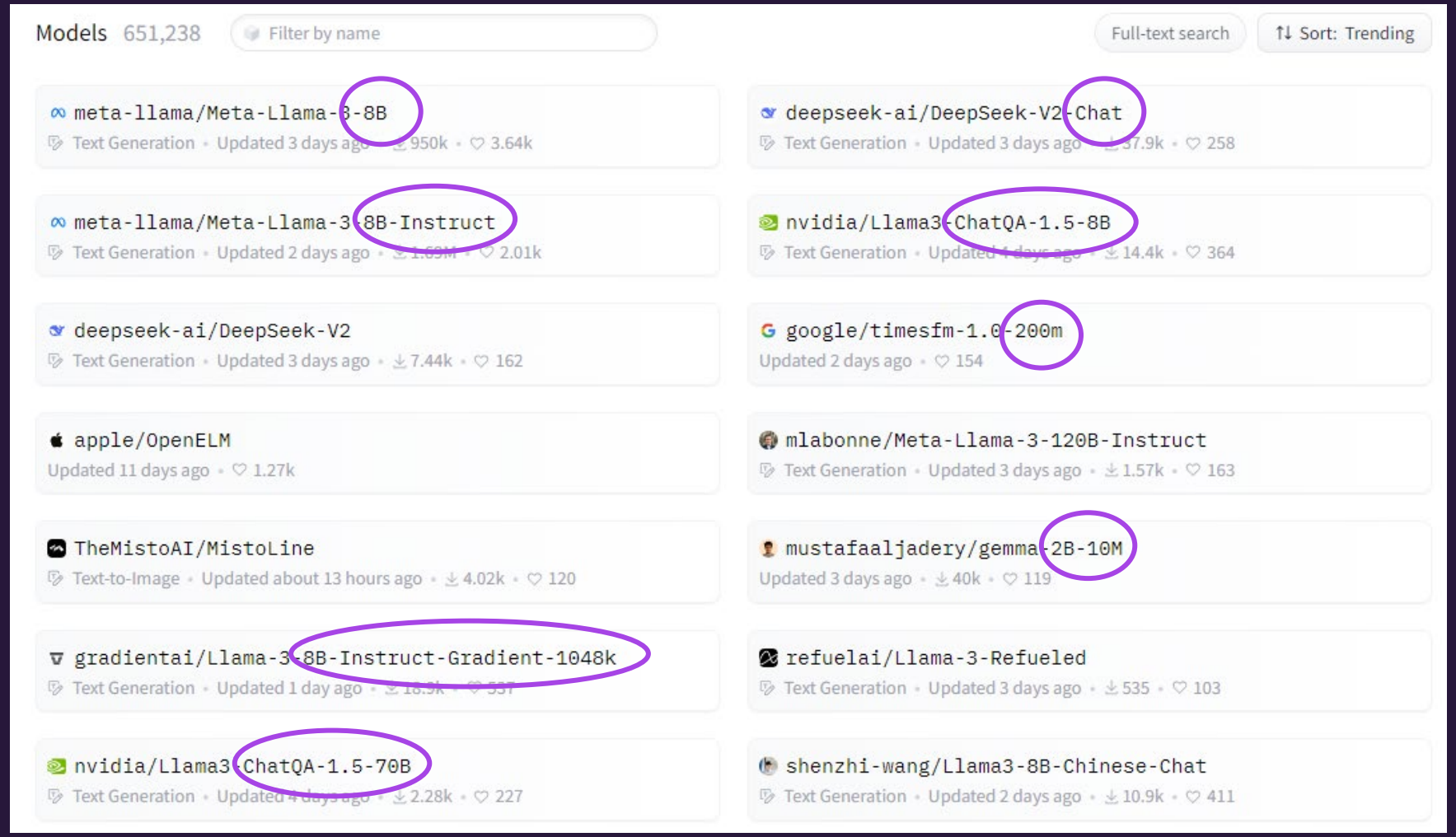
YOU CAN THINK ABOUT IT AS A STATIC SNAPSHOT OF A BRAIN NEURON STATE CONFIGURATION

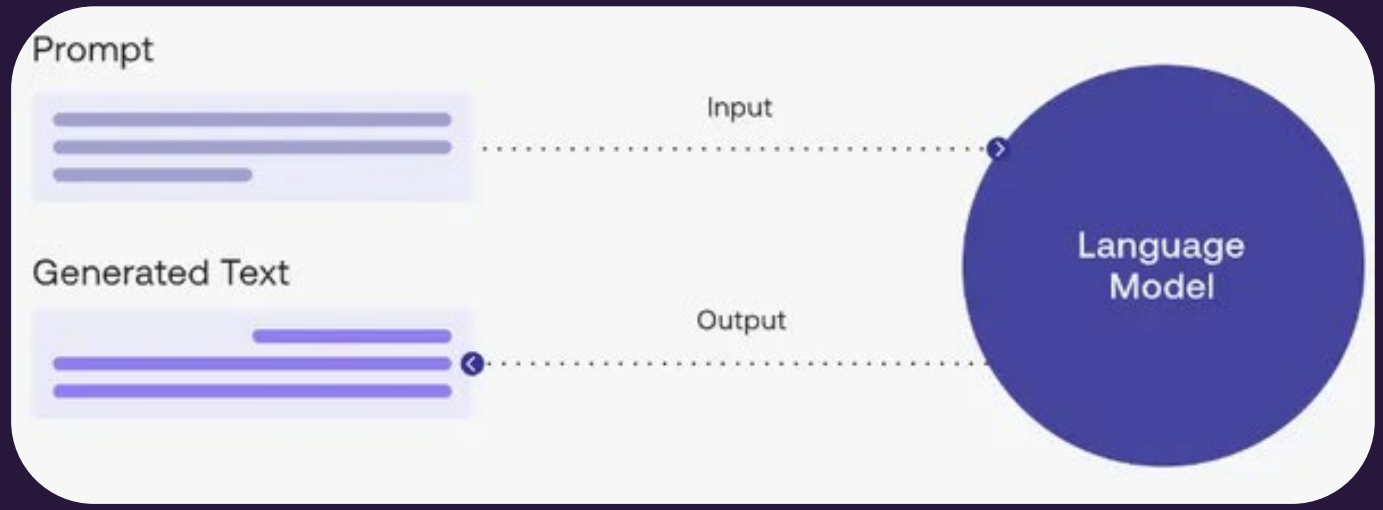


FAMOUS FOUNDATION MODELS



ALL THE BIG WORDS





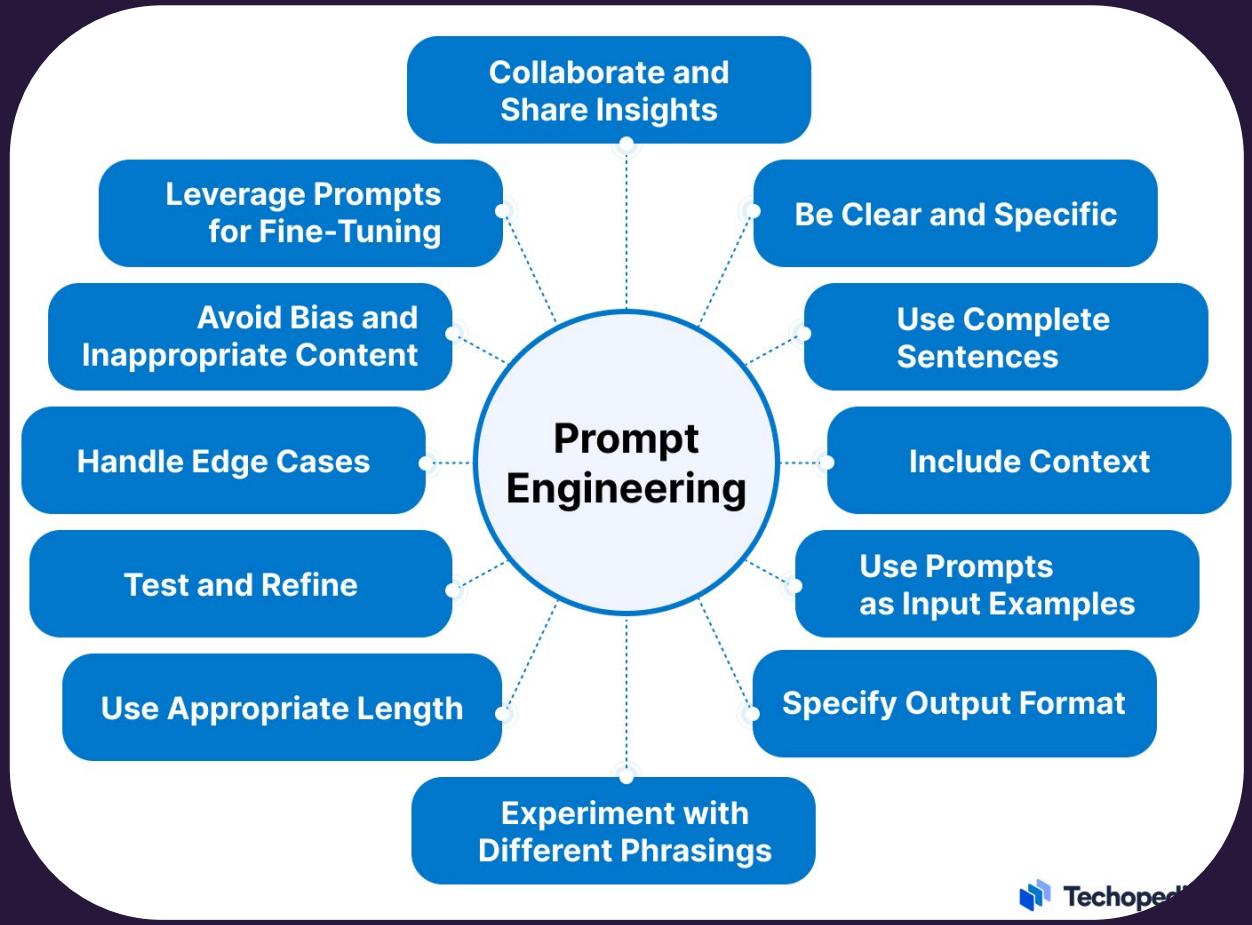
Big words #1: PROMPT

The prompt is the “command” you send to the LLM to get a response, it can be as simple as asking a question

The whole field of “Prompt Engineering” has evolved around the concept of prompt for LLMs, because it has been discovered that how the prompt is written and how much context is given to the LLM changes the output significantly

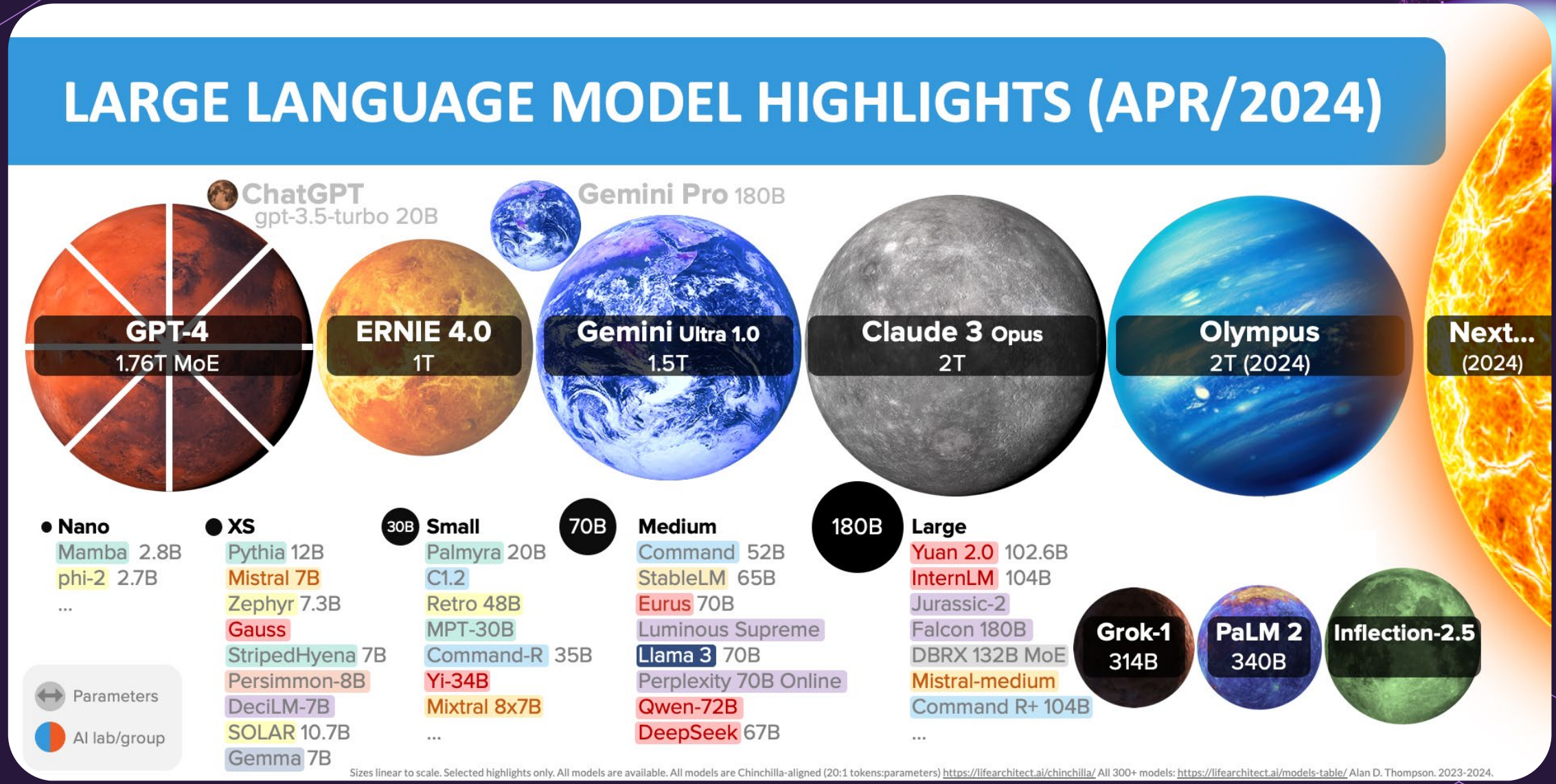
Big words #1: PROMPT

The prompt is the “command” you send to the LLM to get a response, it can be as simple as asking a question



The whole field of “Prompt Engineering” has evolved around the concept of prompt for LLMs, because it has been discovered that how the prompt is written and how much context is given to the LLM changes the output significantly

Big word #2: Parameter Size



Big word #3: TOKENS

GPT-3.5 & GPT-4 | GPT-3 (Legacy)

In the context of Large Language Models (LLMs), a token refers to the smallest unit of text that the model processes. These tokens can be individual words, punctuation marks, or even subword units. LLMs break down input text into tokens to analyze and generate meaningful responses.

Tokenization:
Tokenization is the process of splitting raw text into tokens.
LLMs use tokenization to convert input text into a sequence of tokens that the model can understand.
For example, the sentence “I love cats” might be tokenized into three tokens: “I,” “love,” and “cats.”

Clear | Show example

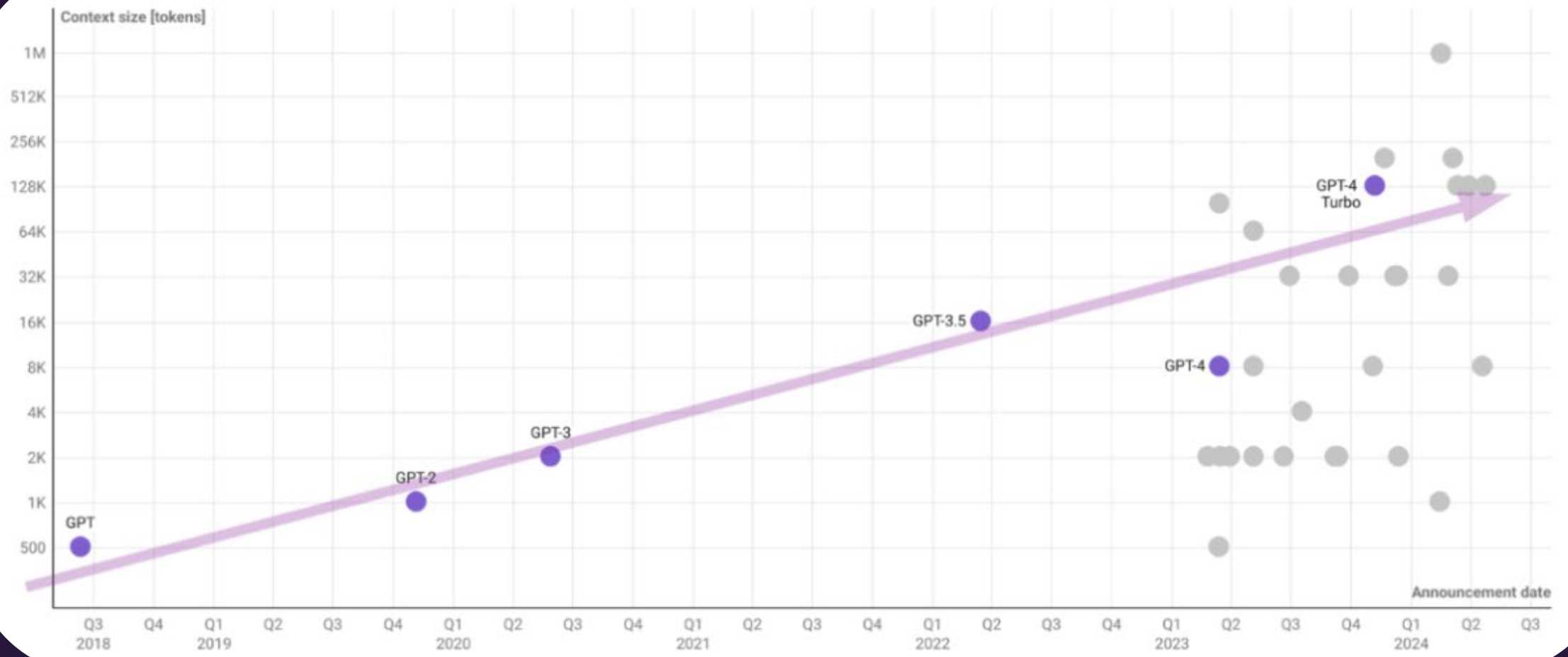
Tokens 120 **Characters** 564

In the context of Large Language Models (LLMs), a token refers to the smallest unit of text that the model processes. These tokens can be individual words, punctuation marks, or even subword units. LLMs break down input text into tokens to analyze and generate meaningful responses.

Tokenization:
Tokenization is the process of splitting raw text into tokens.
LLMs use tokenization to convert input text into a sequence of tokens that the model can understand.
For example, the sentence “I love cats” might be tokenized into three tokens: “I,” “love,” and “cats.”

Text | Token IDs

Big word #4: context window



Big words #5: Shots? But I don't drink!? 🤷

Zero Shot

The model makes predictions about a task without seeing any examples of it during training.
It relies solely on its pre-training phase.
For instance, if a model is asked to translate English to Italian but was never trained on English-Italian translation pairs, it's a zero-shot task.

Few Shots

The model is given a small number of examples for a specific task. For example, if a model is trained to recognize a specific type of bird and is then given a few images of a new bird species to learn from, it's a few-shot task.

Many Shots

The model is fine-tuned on a large number of examples for a specific task. This is the traditional machine learning approach where you have a large labeled dataset for your task of interest.

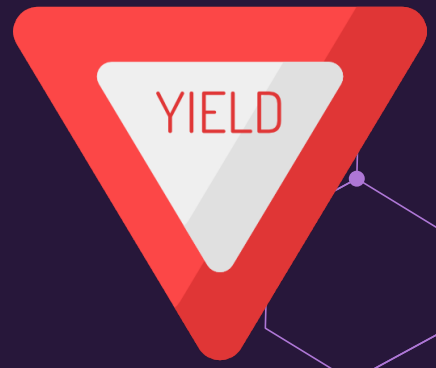
Big words #6: instruct? Chat?

Instruct

Variant of the model that has been fine-tuned to follow specific instructions provided in the user's prompt. This mode is designed for tasks that require the model to produce a particular output based on the instructions, such as translating sentences, summarizing articles, or generating code snippets

Chat

Variant of LLMs that is fine-tuned for conversational contexts. In this mode, the model is expected to engage in a dialogue with users, providing responses that are natural and engaging



02

Deploying a foundation model

How to work with a foundation model

It’s either:

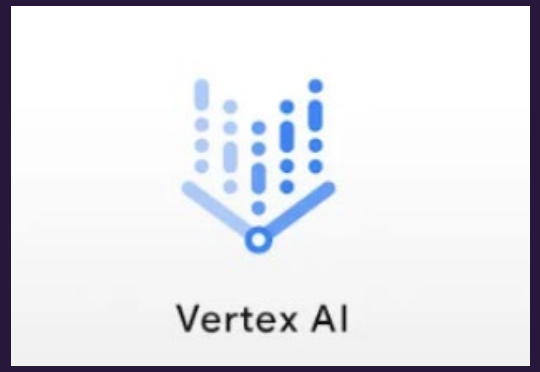
The model is deployed by your vendor of choice, and you access it via REST API

Only option for closed-source models
Many vendors will happily host open-source models too

Self-host the model on your compute resources

For decent performances, a GPU is needed

Anyone will gladly host your foundation models



Region	gpt-4, 1106-Preview	gpt-4, 0125-Preview	gpt-4, vision-preview	gpt-4, turbo-2024-04-09	gpt-4o, 2024-05-13	gpt-4-32k, 0613	gpt-35-turbo, 0301	gpt-35-turbo, 0613	gpt-35-turbo-instruct, 0914	text-embedding-ada-002, 2	text-embedding-3-small, 1	text-embedding-3-large, 1	dall-e-3, 3.0
australiaeast	✅	✅	-	✅	-	-	✅	-	✅	✅	-	✅	-
brazilsouth	-	-	-	-	-	-	-	-	-	-	-	-	-
canadaeast	✅	✅	-	-	-	-	✅	-	✅	✅	✅	✅	-
eastus	-	-	✅	-	-	✅	-	✅	✅	-	-	✅	✅
eastus2	-	✅	-	-	✅	✅	-	-	✅	-	-	✅	-
francecentral	✅	✅	-	-	-	-	✅	✅	✅	✅	-	✅	-
japaneast	-	-	-	✅	-	-	-	-	✅	-	-	✅	-
northcentralus	-	-	✅	-	-	✅	-	-	✅	-	✅	✅	-
norwayeast	-	✅	-	-	-	-	-	-	-	-	-	-	-
southafricanorth	-	-	-	-	-	-	-	-	-	-	-	-	-
southcentralus	-	-	✅	-	-	✅	-	✅	-	-	✅	-	-
southindia	-	✅	-	-	-	-	-	-	-	✅	-	-	-
swedencentral	✅	✅	-	✅	✅	-	✅	-	✅	✅	-	✅	✅
switzerlandnorth	✅	-	-	✅	-	-	✅	-	✅	-	-	✅	-
uksouth	-	✅	✅	-	-	-	-	✅	✅	✅	-	✅	-
westeurope	-	-	-	-	-	-	-	✅	-	-	-	-	-
westus	-	✅	-	✅	-	✅	-	-	-	✅	-	-	-
westus3	-	✅	-	-	-	✅	-	-	-	-	-	-	-

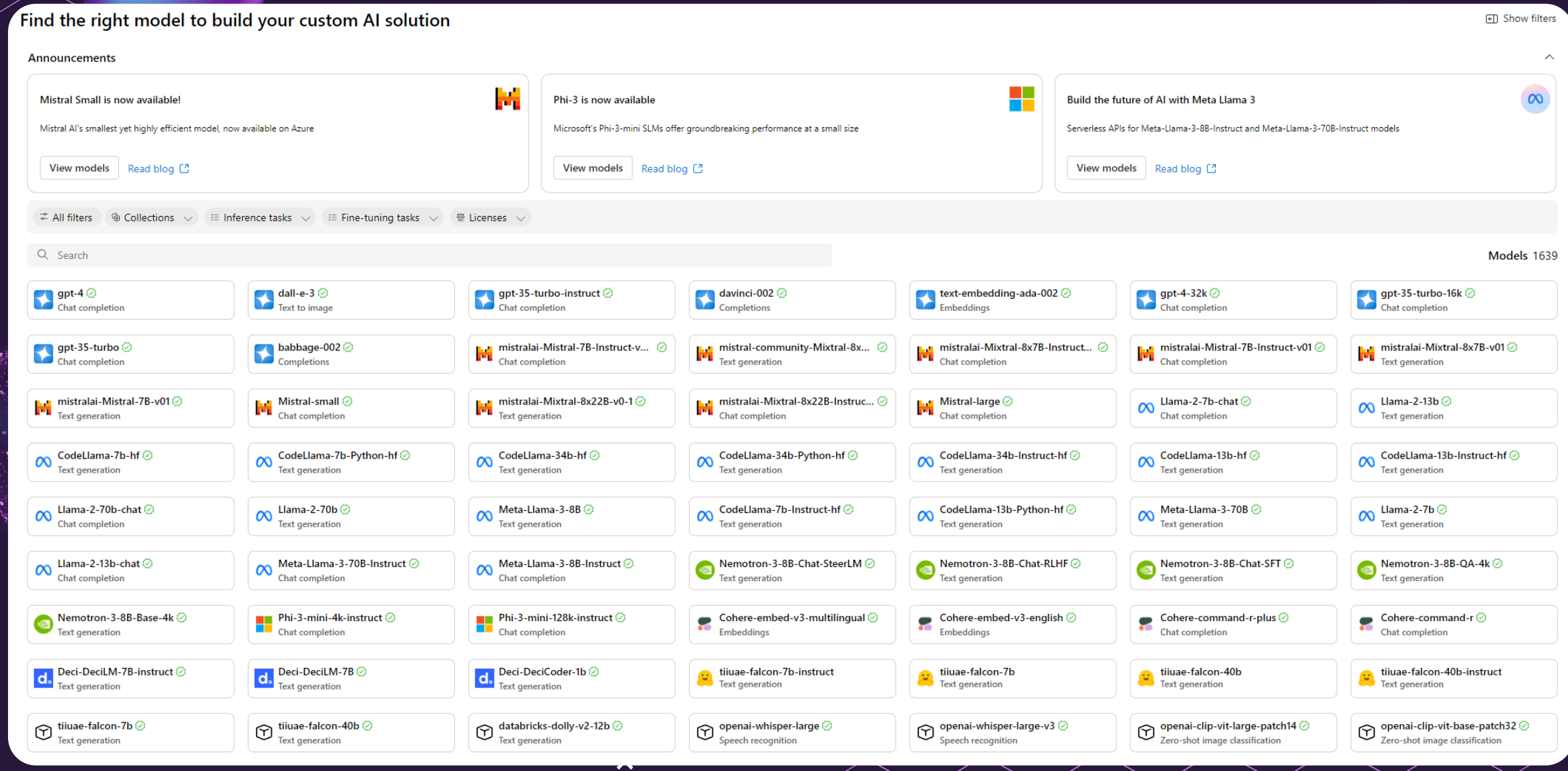
Azure OpenAI API Pricing Example

Pricing details:

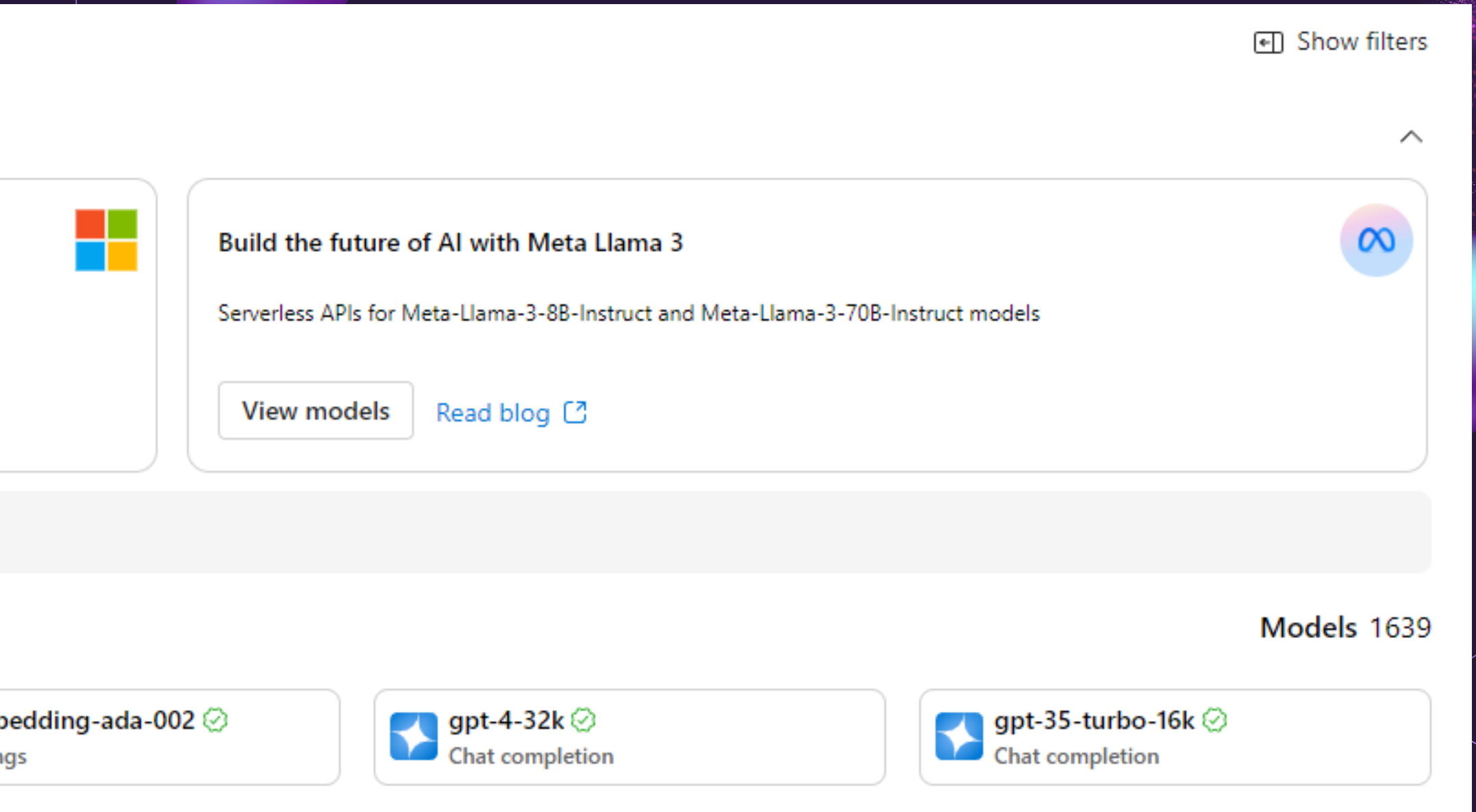
Language models

Models	Context	Input (Per 1,000 tokens)	Output (Per 1,000 tokens)
GPT-3.5-Turbo-0125	16K	€0.0005	€0.0015
GPT-3.5-Turbo-Instruct	4K	€0.0015	€0.002
GPT-4-Turbo	128K	€0.010	€0.029
GPT-4-Turbo-Vision	128K	€0.010	€0.029
GPT-4	8K	€0.029	€0.057
GPT-4	32K	€0.057	€0.113

Probably even way too many models



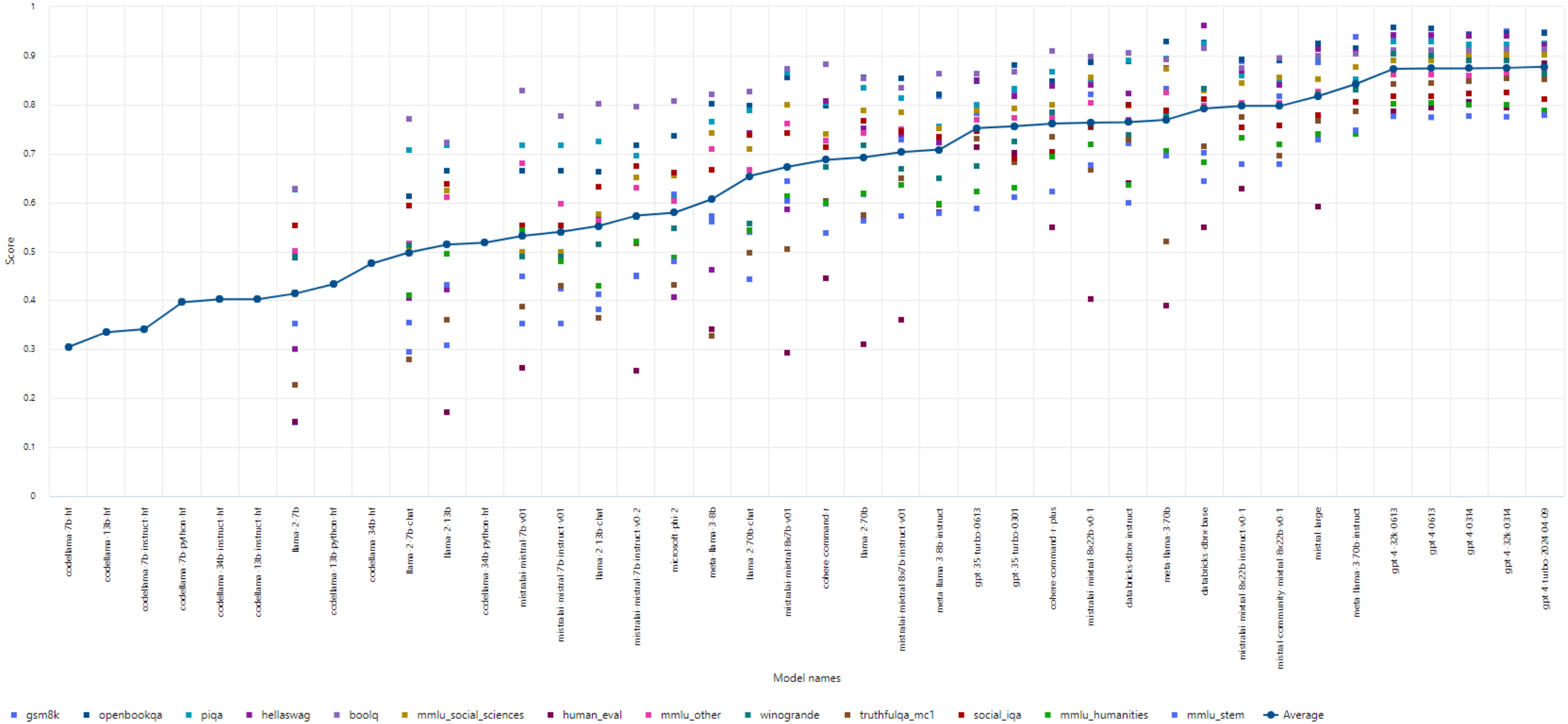
Probably even way too many models



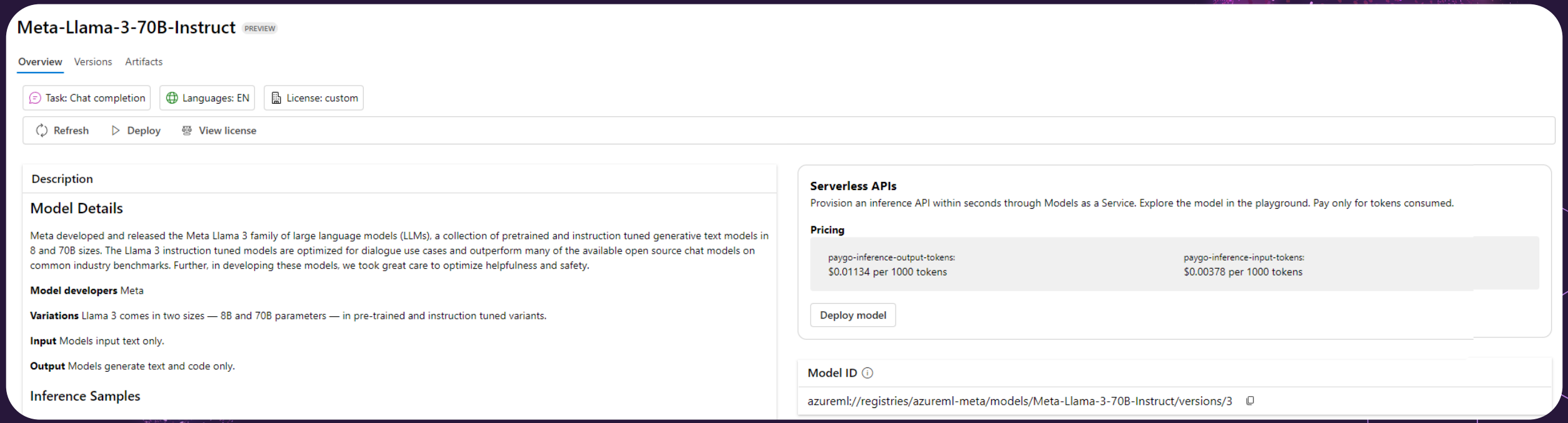
Benchmarks to go

Model accuracy

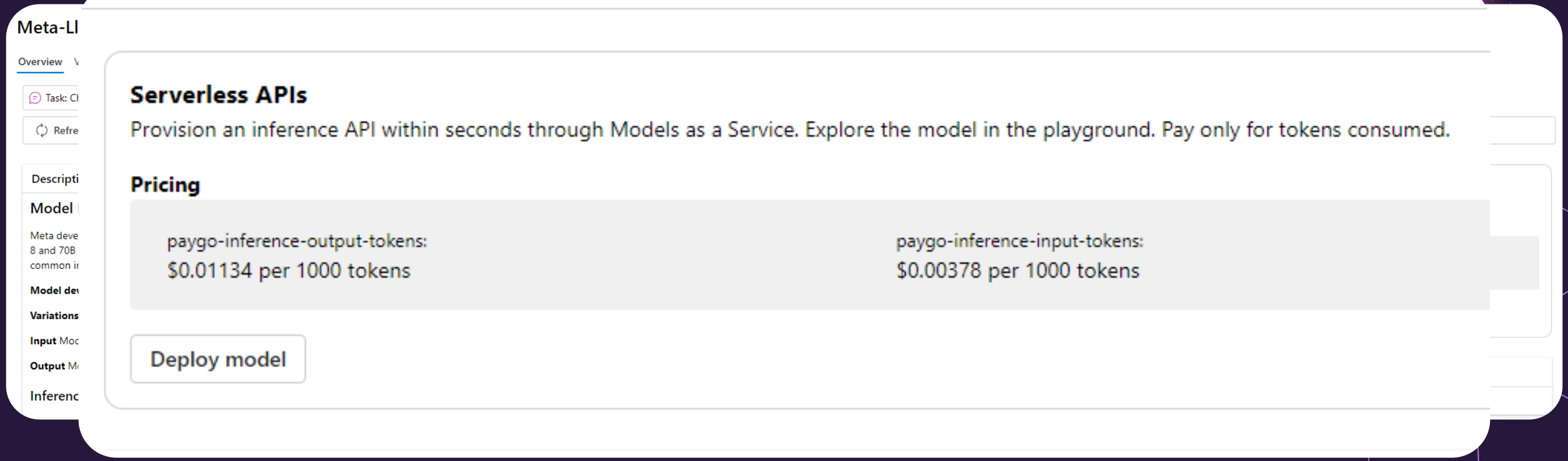
Accuracy scores are presented at the dataset and the model levels. At the dataset level, the score is the average value of an accuracy metric computed over all examples in the dataset. Learn more about accuracy.



Serverless deploy

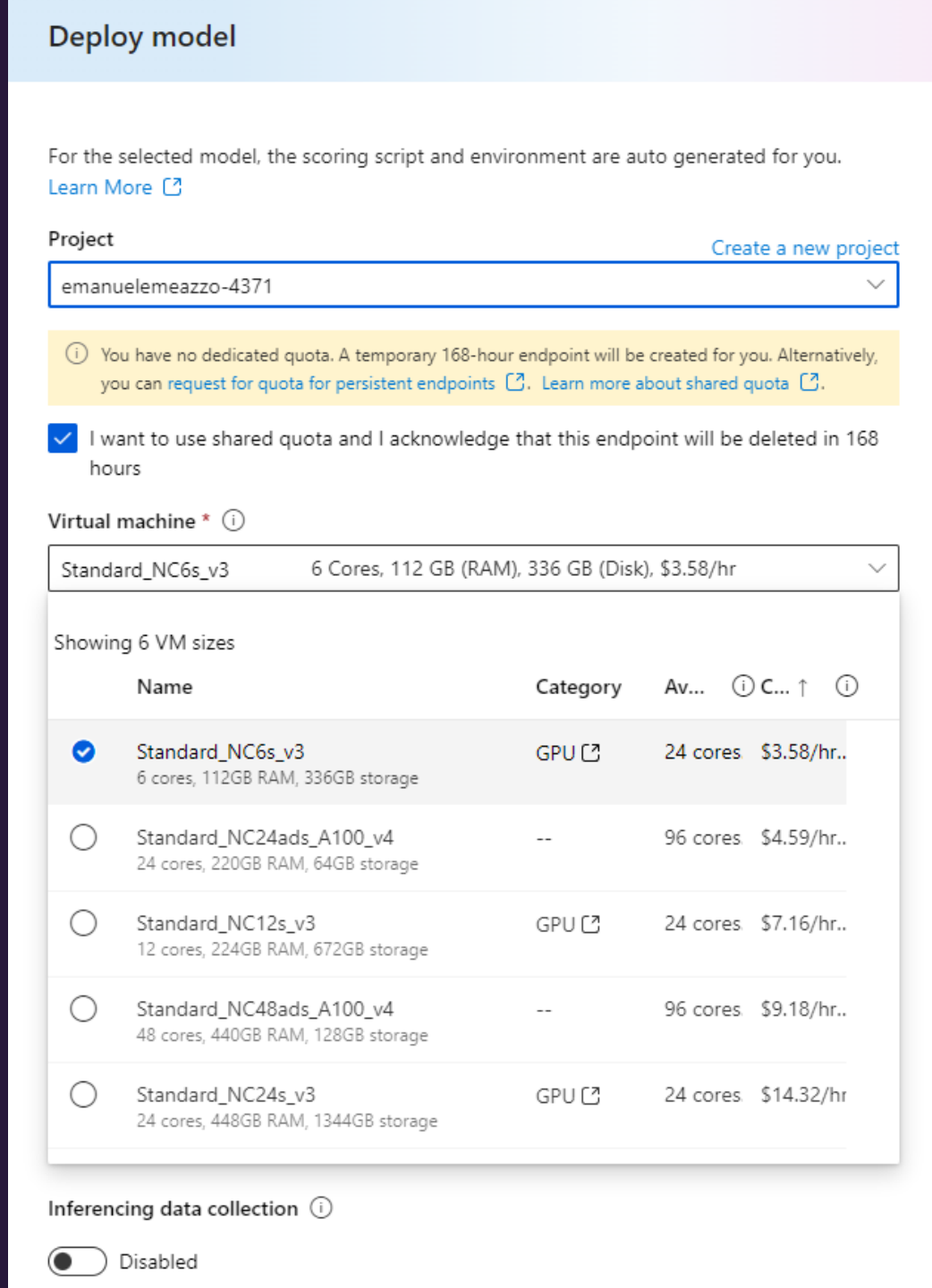


Serverless deploy



VM deploy

- The models that are not offered with a serverless endpoint are **deployable in a VM**
- Of course, for decent performances you'll need a VM with a **GPU**, which makes the pricing not trivial
- But if everything it's in a VM, isn't it better to **control it yourself**?



Run it yourself!

Ollama

GPT4All

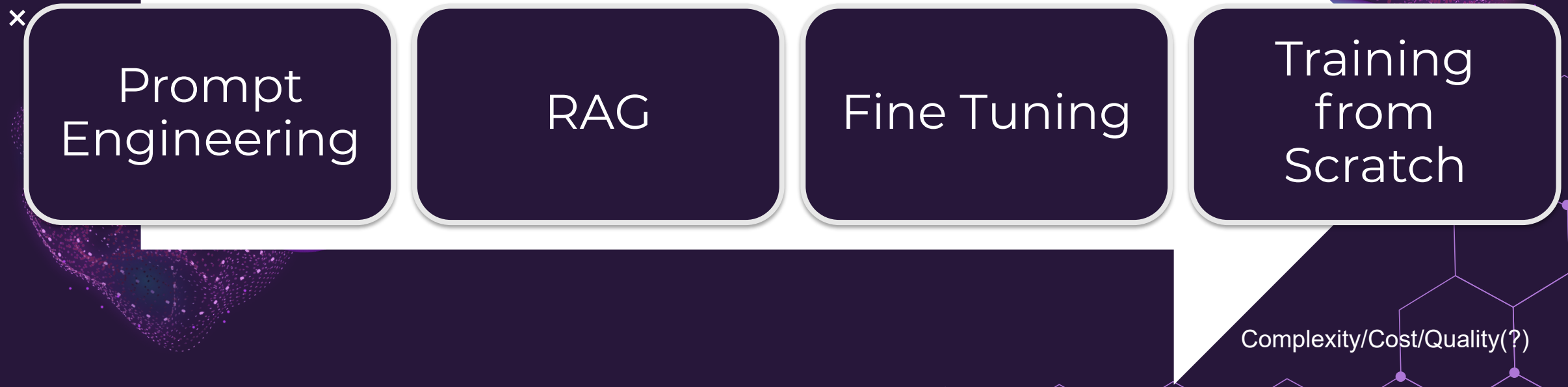
Jan

LM Studio

LLAMA.cpp

Two ways to work with your data and LLM

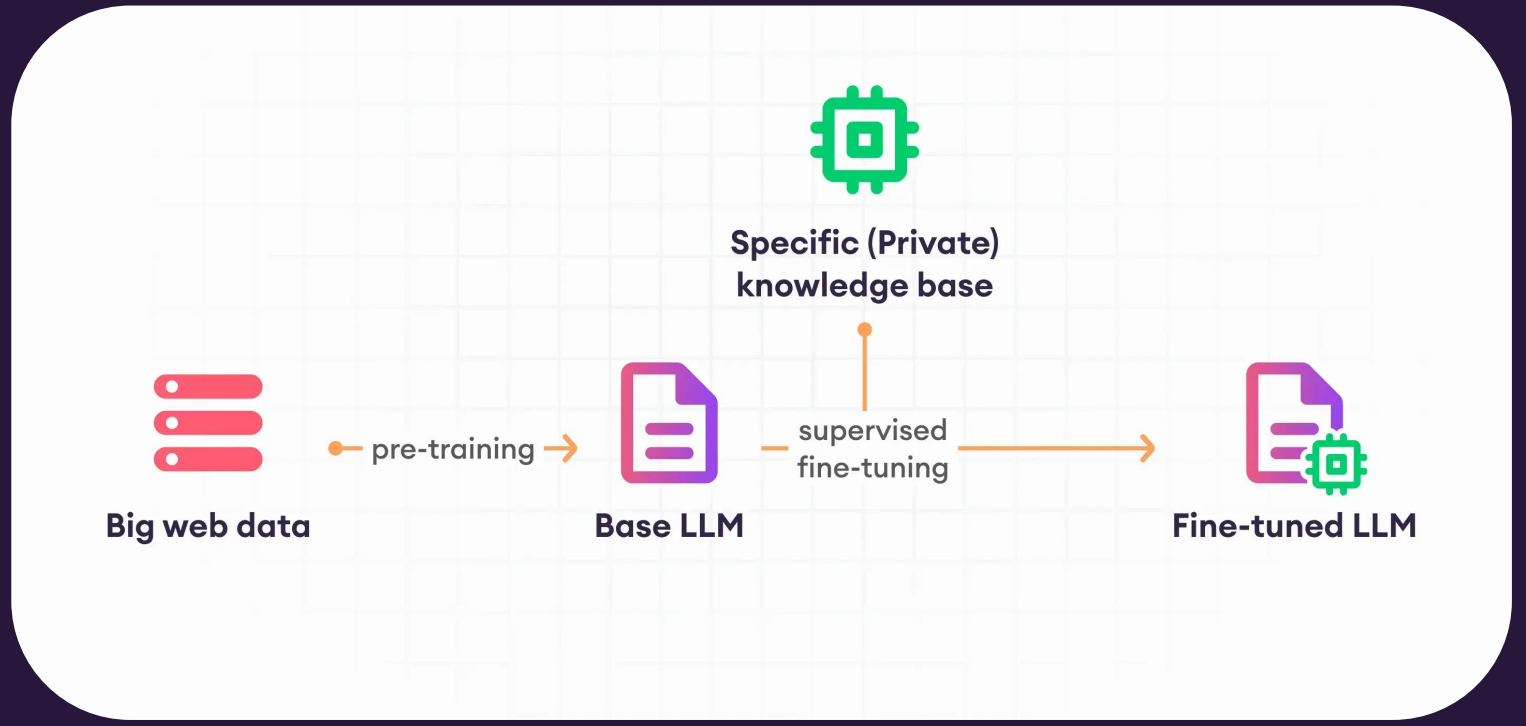
Paths to get most out of a LLM



Paths to get most out of a LLM

Fine tuning

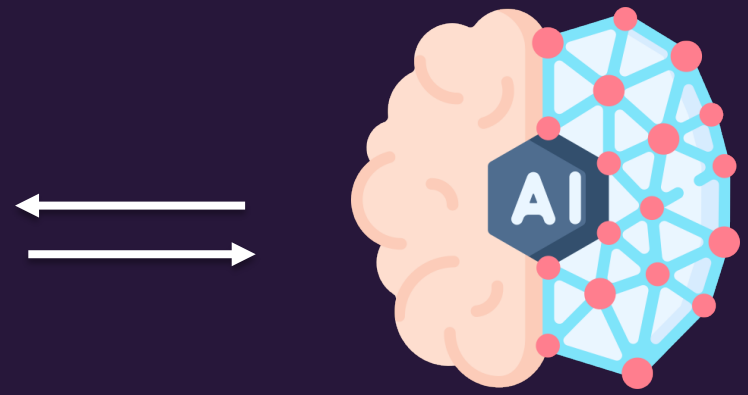
- Probably will never be your job
- Probably out of budget for your company
- Changes the behavior of the LLM directly updating the neural network weights inside the LLM
- Requires:
  - Curated Training Dataset
  - Days/Weeks of compute
  - ML Team Expertise
  - Some vendor-provided models can be fine-tuned to for an extra price in training and serving



RAG (Retrieval Augmented generation)

- With RAG you inject your knowledge in the LLM
- It mixed information retrieval with text generation
- It allows to use a foundation model as-is and have it work with your private data without the need of training a custom model

Your documents



The LLM

You

04
RAG

How does RAG rags?

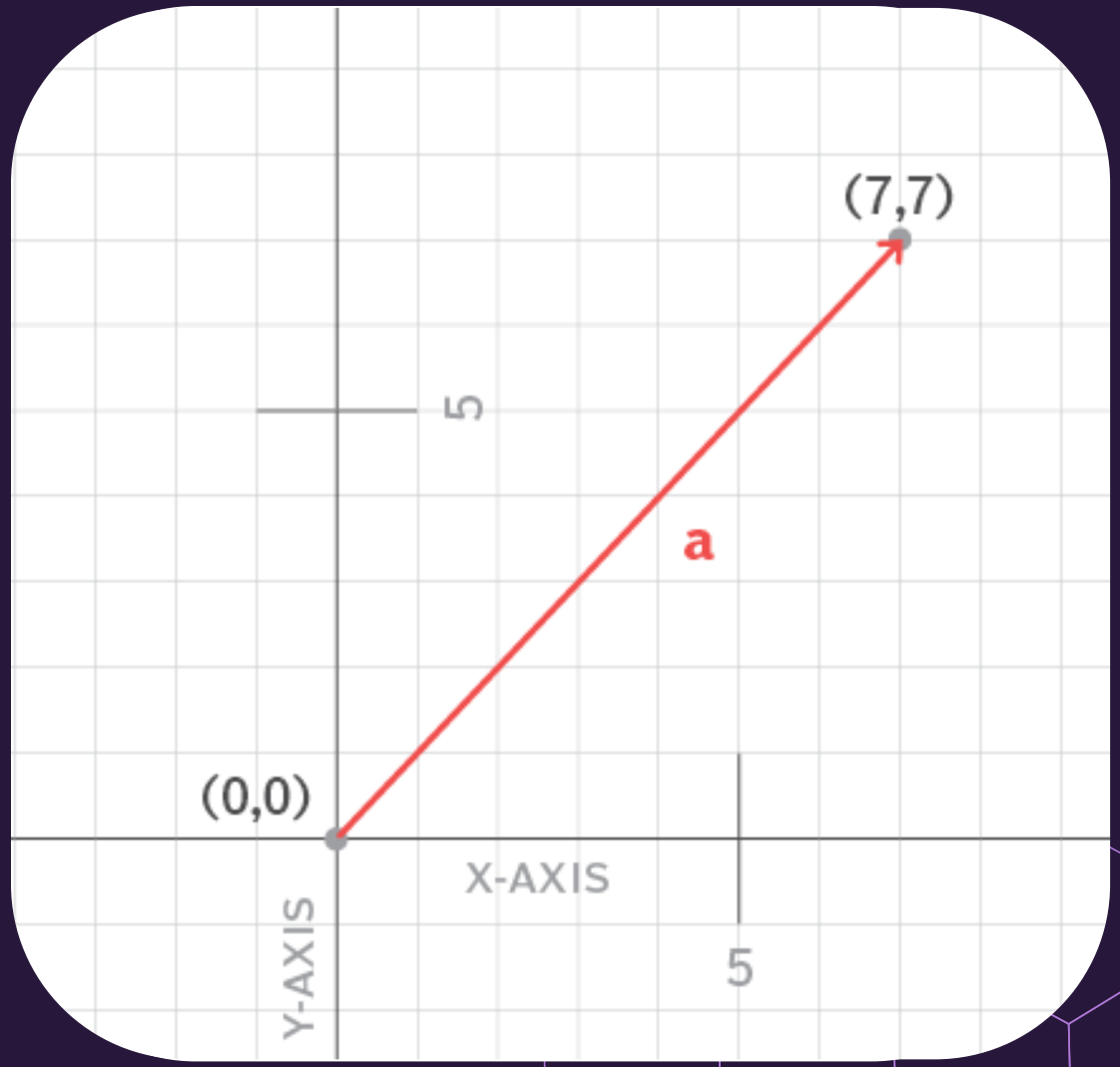
Magic words:

Embeddings
Vectors
Semantic Search

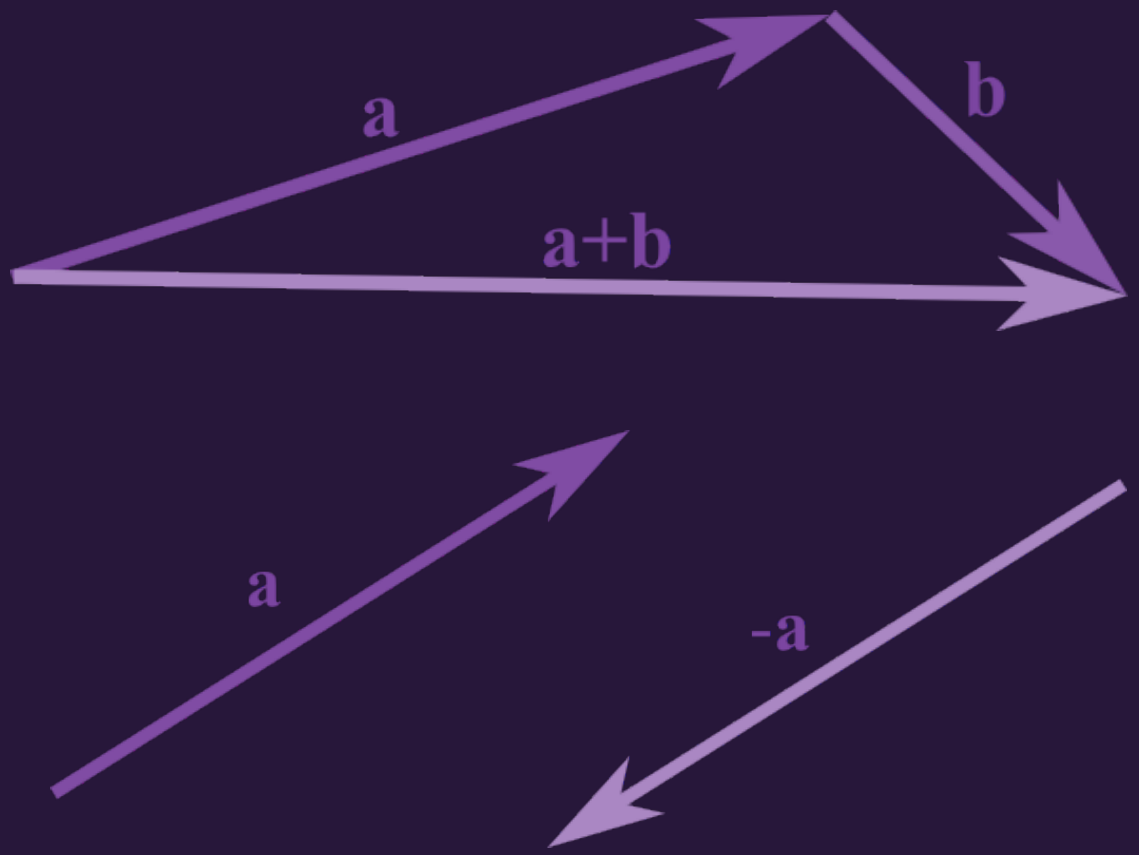
Back to high school

Vectors

$$v = \begin{bmatrix} 7 \\ 7 \end{bmatrix}$$

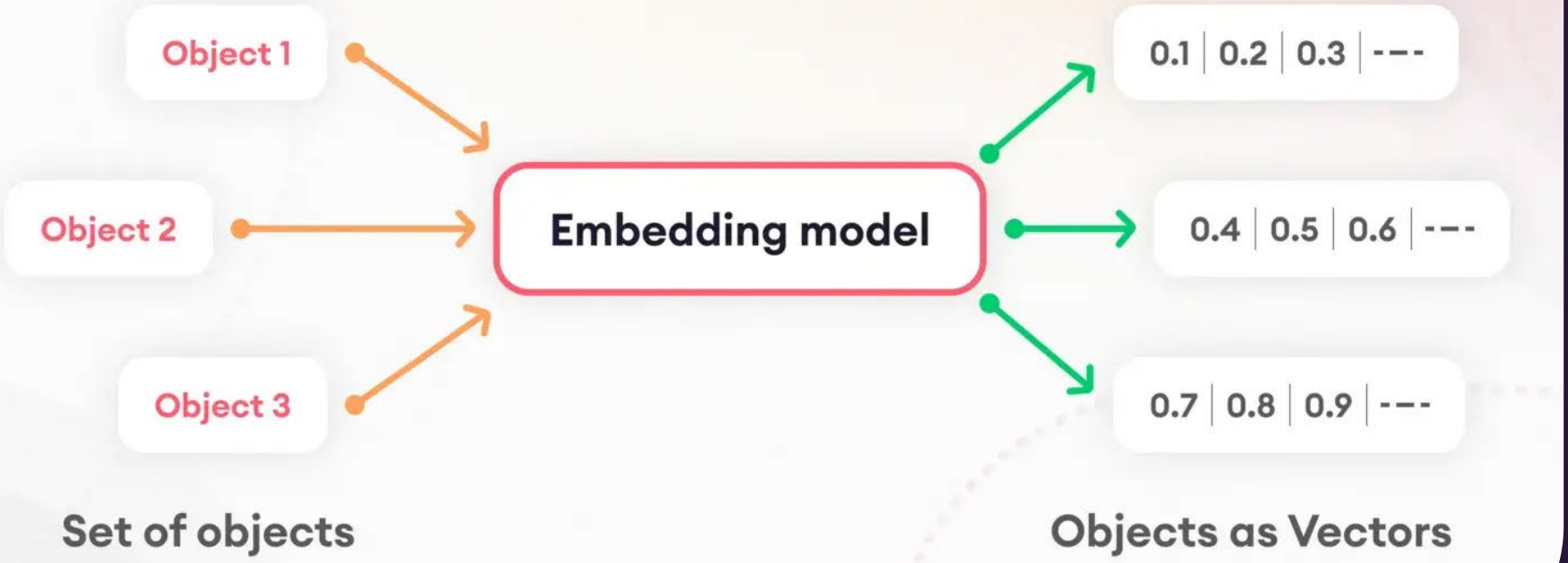


Back to high school

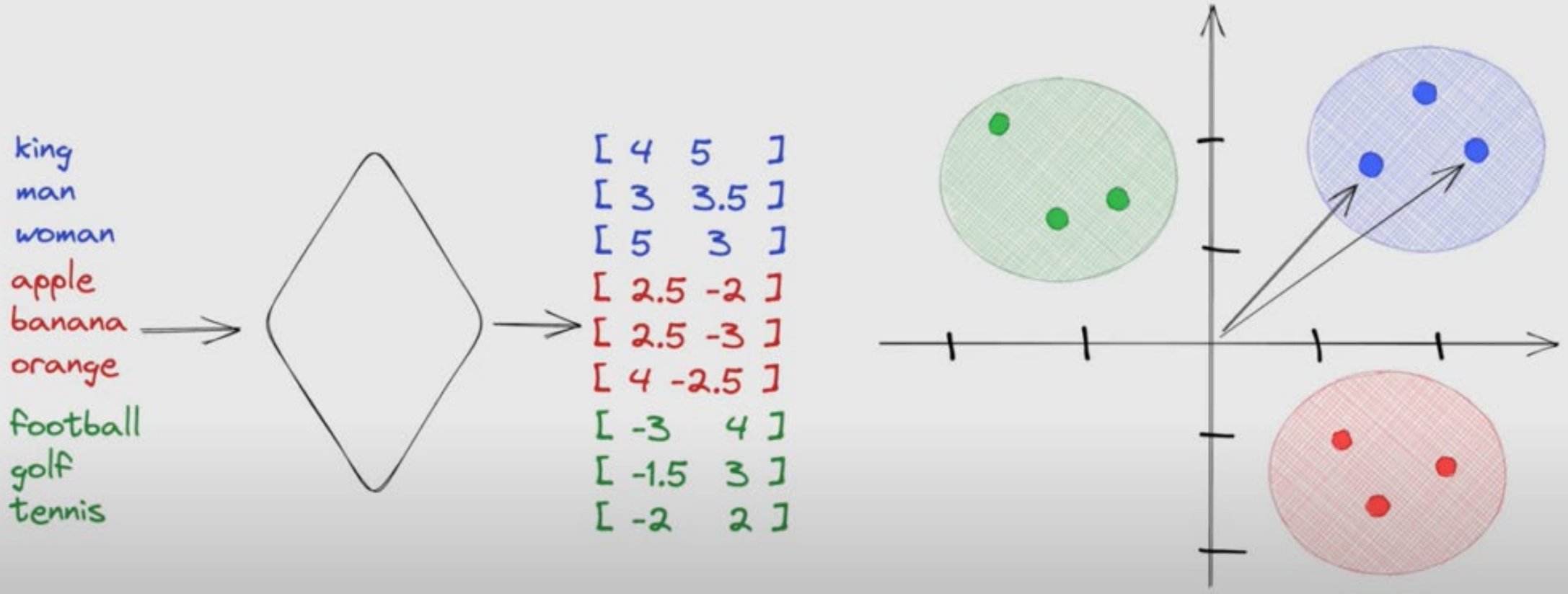


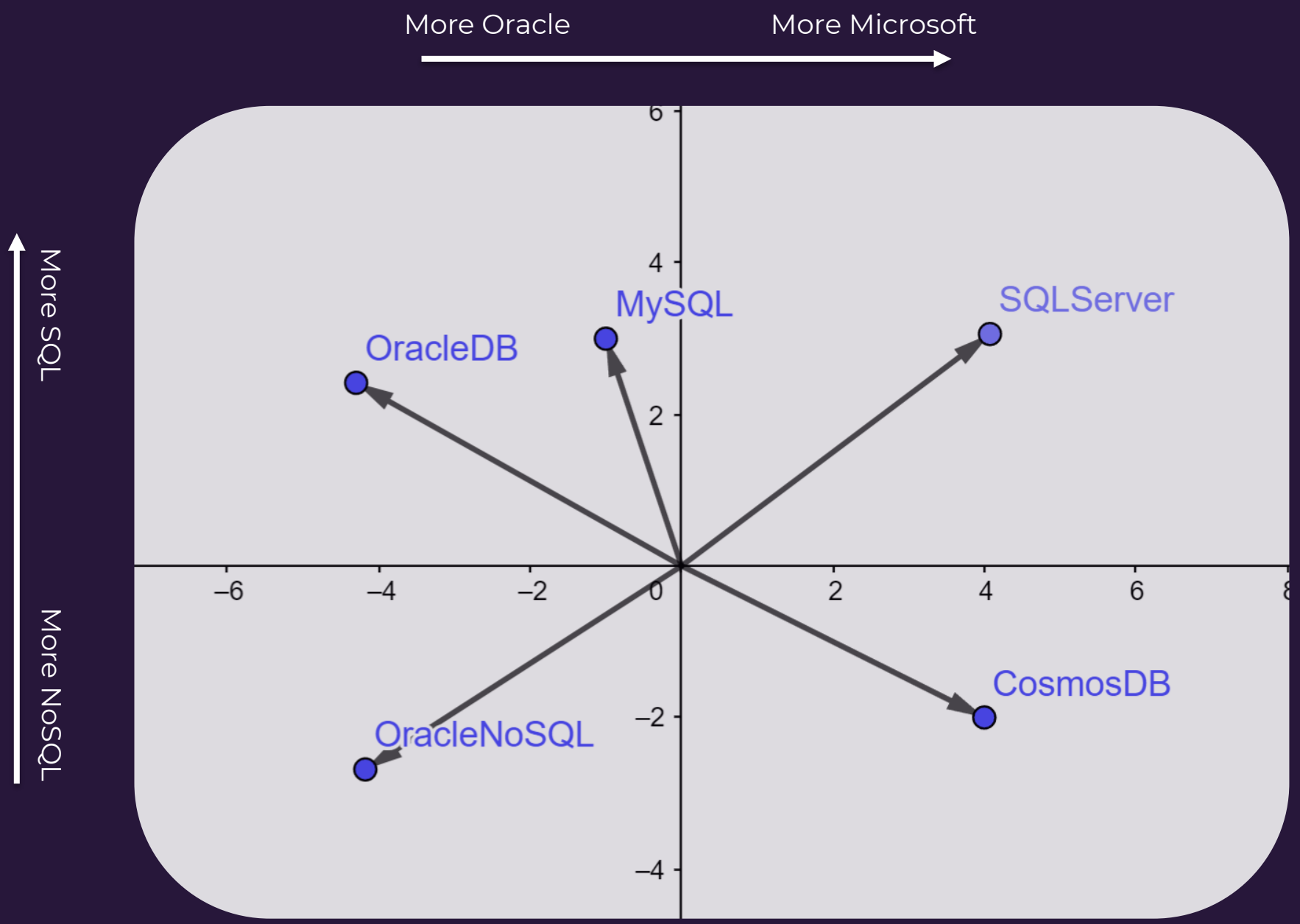
$$v = \begin{bmatrix} v_1 \\ v_2 \\ \cdots \\ \cdots \\ v_n \end{bmatrix}$$

Embeddings


Object 1
Object 2
Object 3
Embedding model
0.1 | 0.2 | 0.3 | ---
0.4 | 0.5 | 0.6 | ---
0.7 | 0.8 | 0.9 | ---
Set of objects
Objects as Vectors

vector embeddings (2D example)




More Oracle
More Microsoft
More SQL
More NoSQL
MySQL
SQLServer
OracleDB
CosmosDB
OracleNoSQL
6
4
2
−2
−4
−6
−4
−2
0
2
4
6

EMBEDDING MODELS

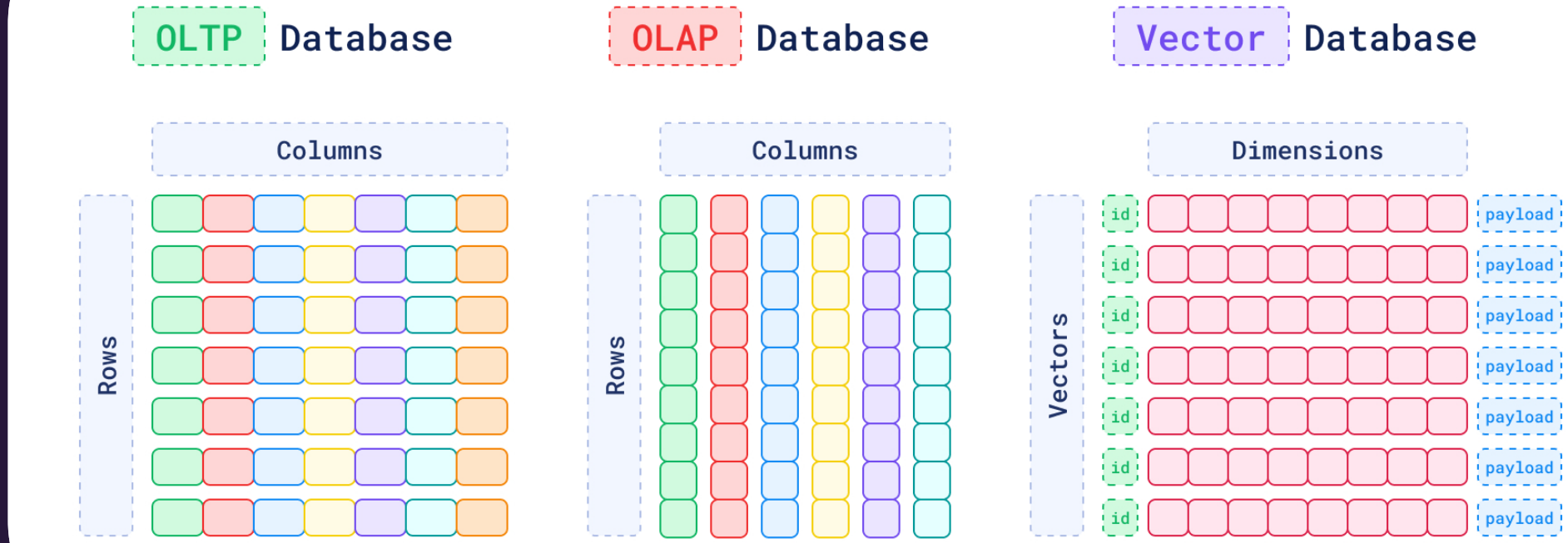
MODEL	DESCRIPTION	OUTPUT DIMENSION
`text-embedding-3-large`	New **Embedding V3 large** Most capable embedding model for both english and non-english tasks	3,072
`text-embedding-3-small`	New **Embedding V3 small** Increased performance over 2nd generation ada embedding model	1,536
`text-embedding-ada-002`	Most capable 2nd generation embedding model, replacing 16 first generation models	1,536

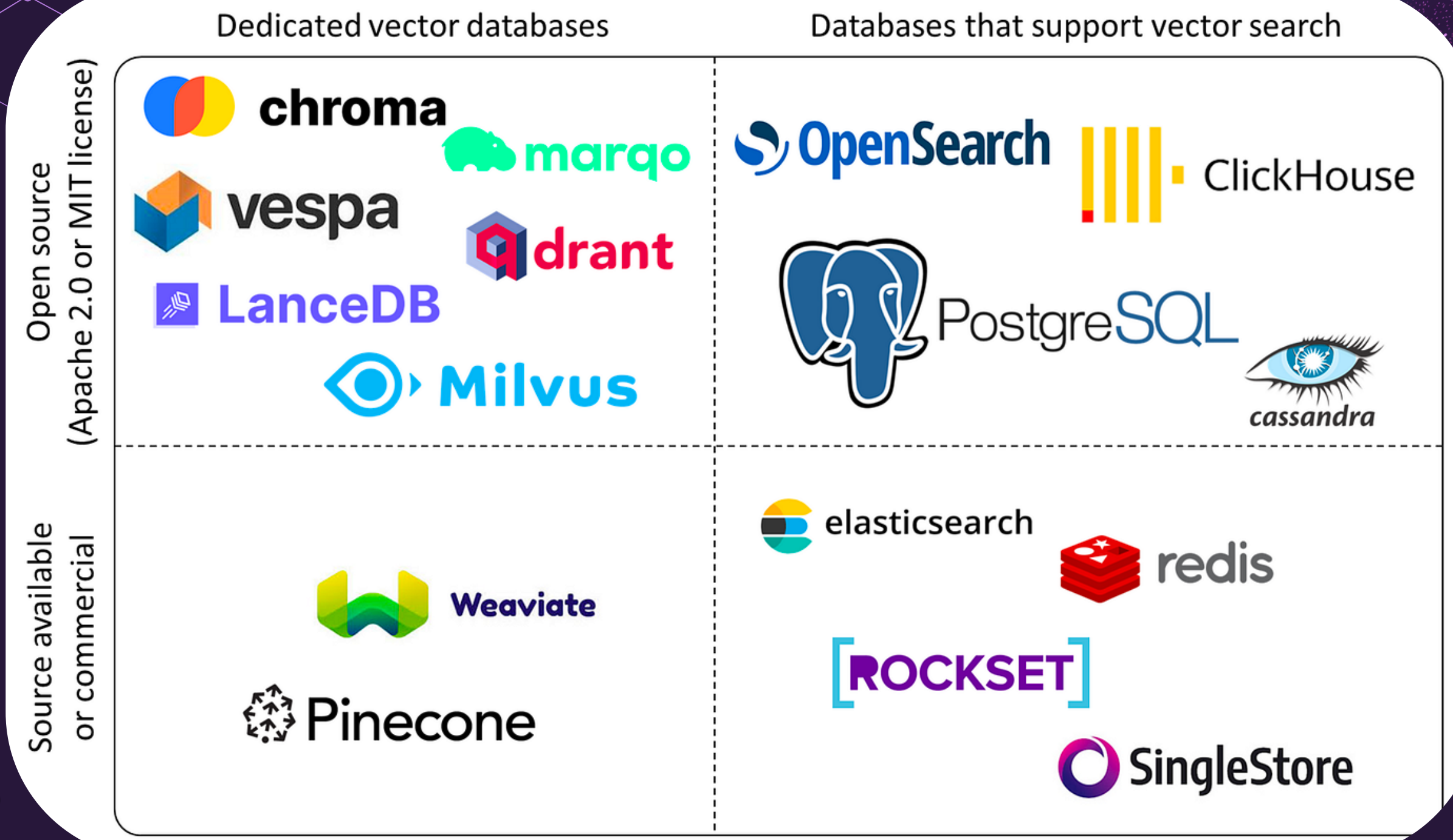
EMBEDDING VECTORS

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ \dots \\ v_{3071} \\ v_{3072} \end{bmatrix}$$

OUTPUT DIMENSION
3,072
1,536
1,536

Where are you going to put all of these vectors?




Dedicated vector databases
Databases that support vector search
Open source
(Apache 2.0 or MIT license)
Source available
or commercial
chroma
marqo
vespa
drant
LanceDB
Milvus
OpenSearch
ClickHouse
PostgreSQL
cassandra
elasticsearch
redis
Weaviate
ROCKSET
Pinecone
SingleStore

MICROSOFT IS PUSHING A LOT TO ADD VECTOR FUNCTIONALITY ON EXISTING DATA SERVICES

MICROSOFT IS PUSHING A LOT TO ADD VECTOR FUNCTIONALITY ON EXISTING DATA SERVICES

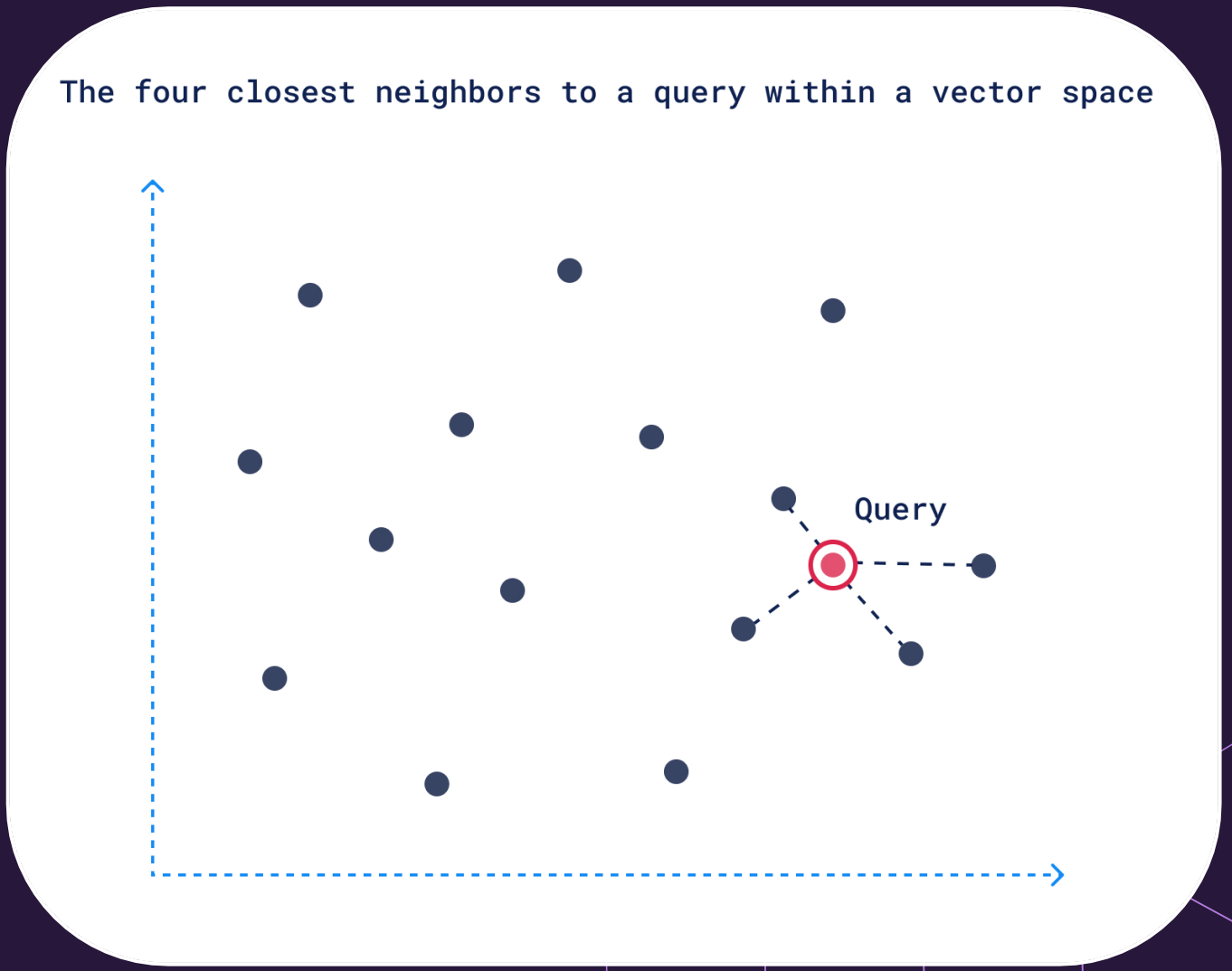
Azure AI Search

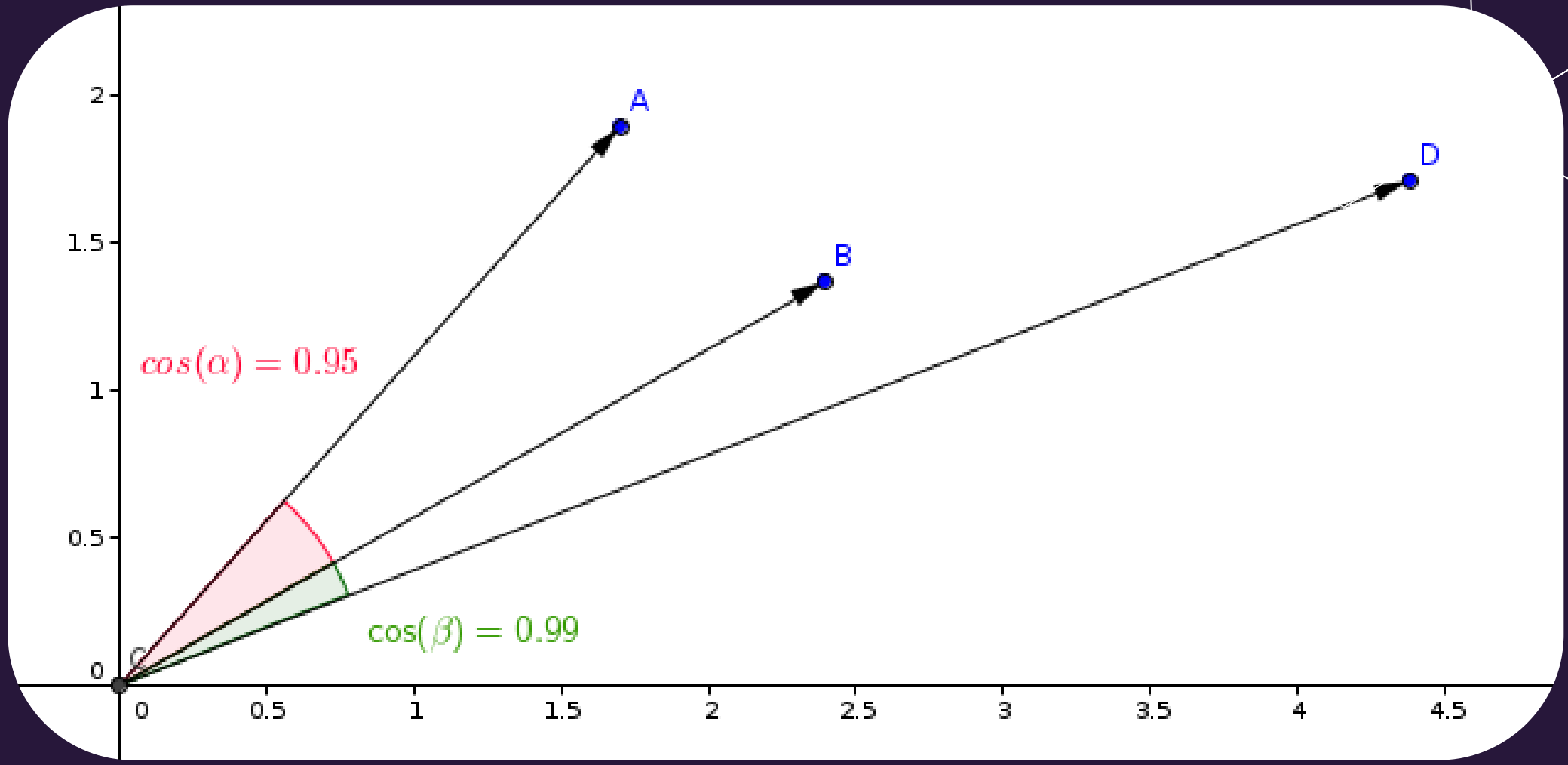
Azure SQL

Azure Cosmos DB

PostgreSQL Flexible Server

Vector Query





SEMANTIC SEARCH IS IMPLEMENTED VIA VECTOR SIMILARITY SEARCH

- Dot Product
- Cosine Similarity
- Euclidean Distance
- Manhattan Distance


WAIT A SECOND
DO YOU DUMP EVERYTHING?!
DUMP EVERYTHING!

CHUNKING

Copying from the human mind once again

Chunking: Fueling Efficient Information Retrieval

What is chunking?

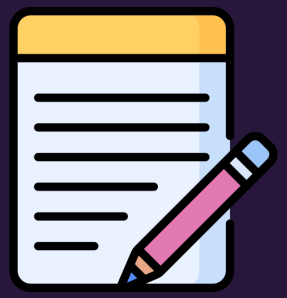
Breaking down text in smaller segments

Why Chunking Matters in RAG?

- Overcomes context window limits
- Enables targeted retrieval
- Boosts efficiency and speed

What's the best way to chunk?

- Ideally, splitting the corpus in self-contained concepts so when you're searching for it you get good semantic match
- In practice, it is difficult and depends on the nature of the content, more naive implementations like a fixed cutting window with some overlapping can be used at first

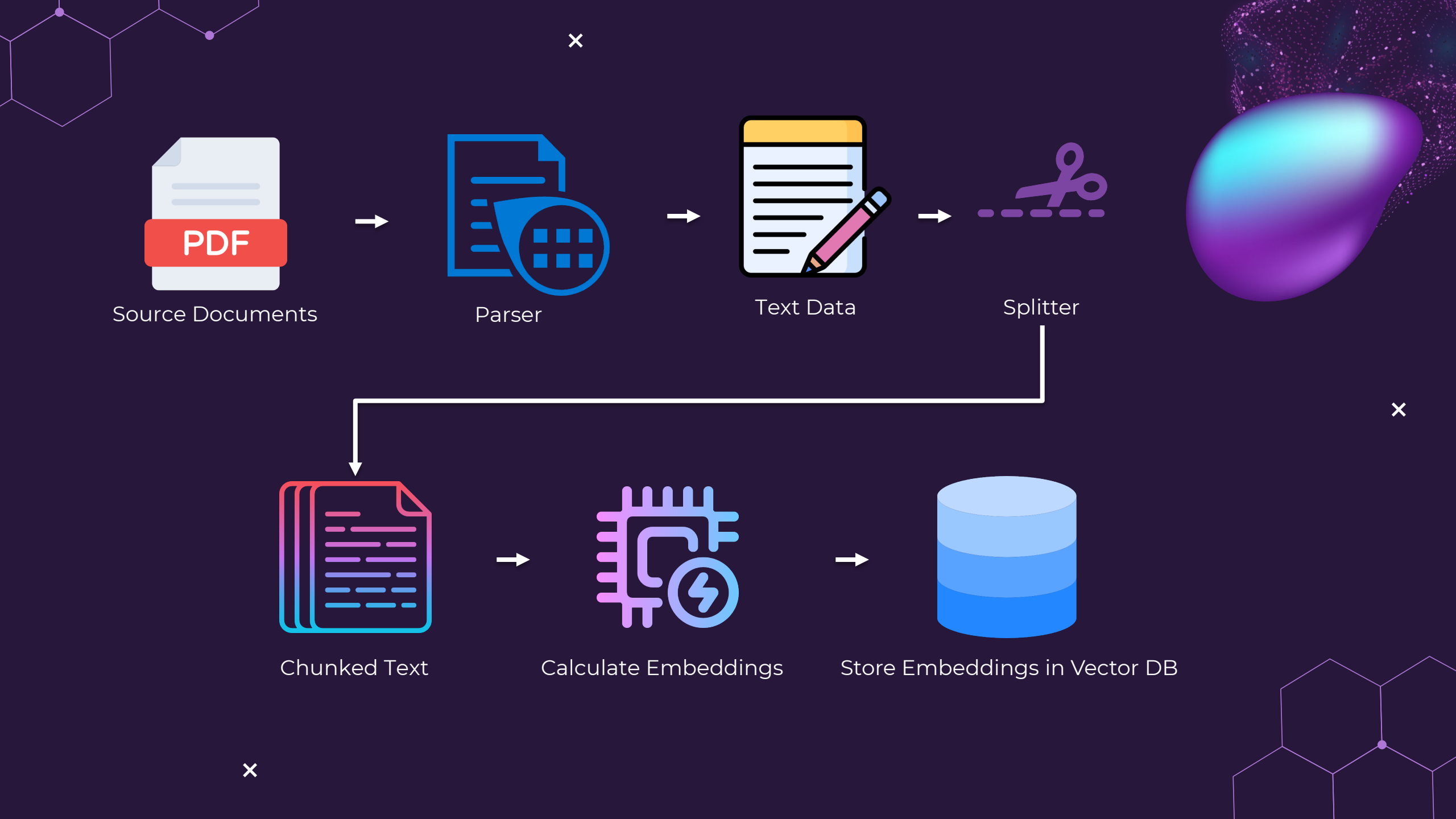

Source Text


Splitter


Chunked Text

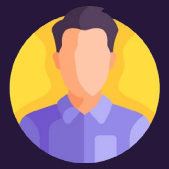

Calculate Embeddings


Store Embeddings in Vector DB


PDF
Source Documents
Parser
Text Data
Splitter
Chunked Text
Calculate Embeddings
Store Embeddings in Vector DB

Let’s put everything together

An end to end RAG architecture



User

Prompt

Application

Embedding Model

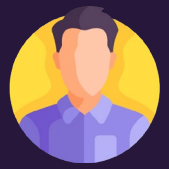
Vector DB

Foundation Model

Create Embeddings
from the Prompt

Embedding Model

Vector DB



User

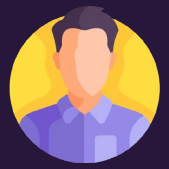
Application

Foundation Model

Search the vector DB with the Embeddings

Embedding Model

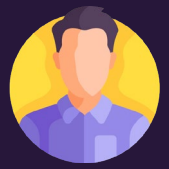
Vector DB



User

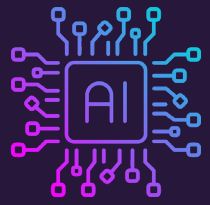
Application

Foundation Model



User

Application



Embedding Model

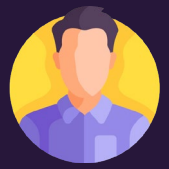
Vector DB

Send the results (relevant chunks) to the app

Foundation Model

Embedding Model

Vector DB

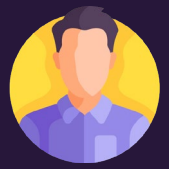


User

Application

Foundation Model

Send the original prompt to the foundation model, including the relevant chunks found in the context



User

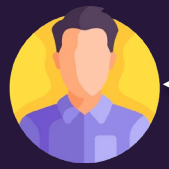
Application

Embedding Model

Vector DB

Foundation Model

Send the AI response, based on the context provided by the included chunks



Receive
the response

User

Application

Embedding Model

Vector DB

Foundation Model

That’s it!

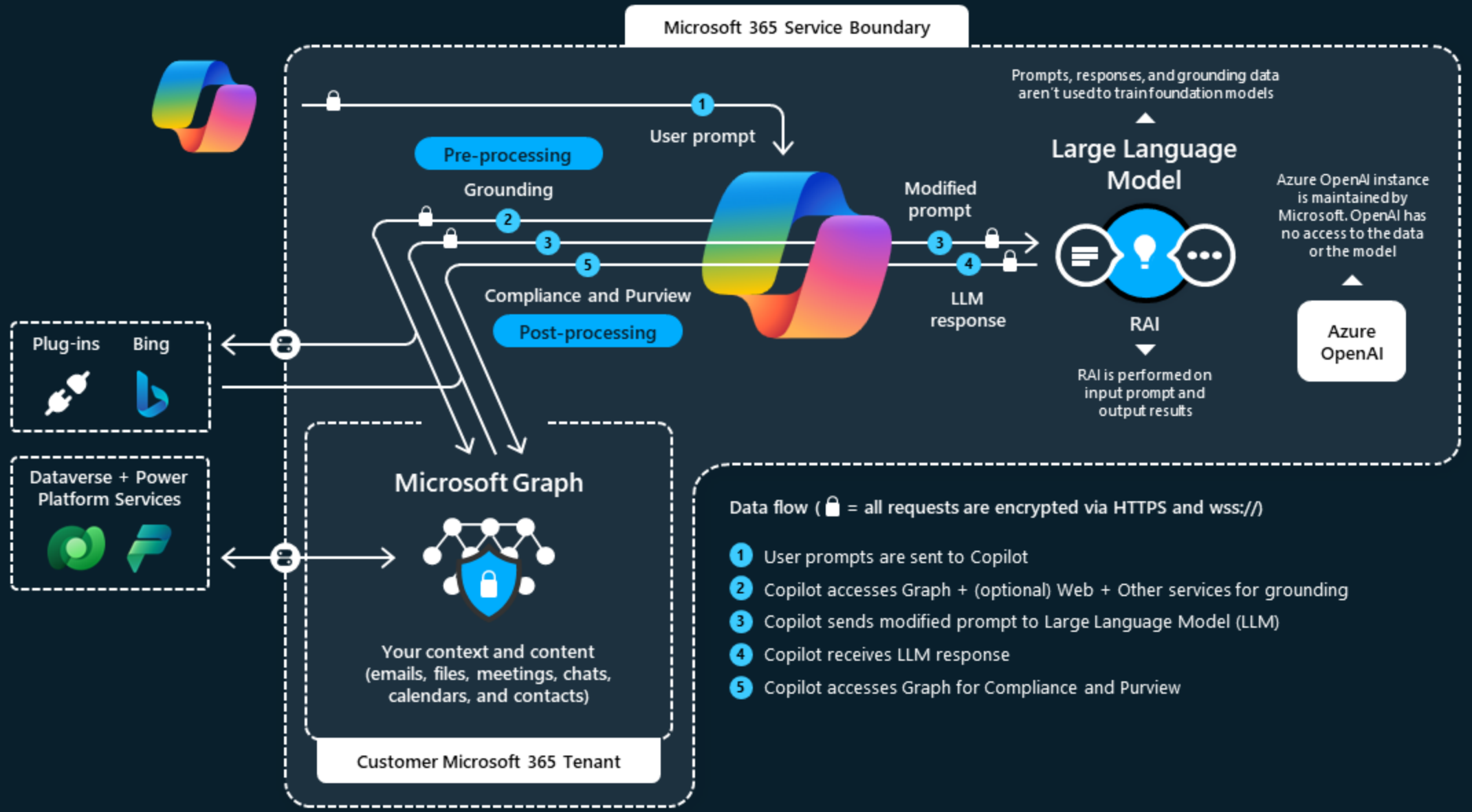
Big word #6: GROUNDING

Literally what we just did with RAG 🤷

“Grounding involves equipping LLMs with specific, use-case-driven information that is not inherently included in their training data.”

Now you can easily understand this:

Microsoft Copilot for Microsoft 365 architecture



Data flow (= all requests are encrypted via HTTPS and wss://)

1. User prompts are sent to Copilot
2. Copilot accesses Graph + (optional) Web + Other services for grounding
3. Copilot sends modified prompt to Large Language Model (LLM)
4. Copilot receives LLM response
5. Copilot accesses Graph for Compliance and Purview

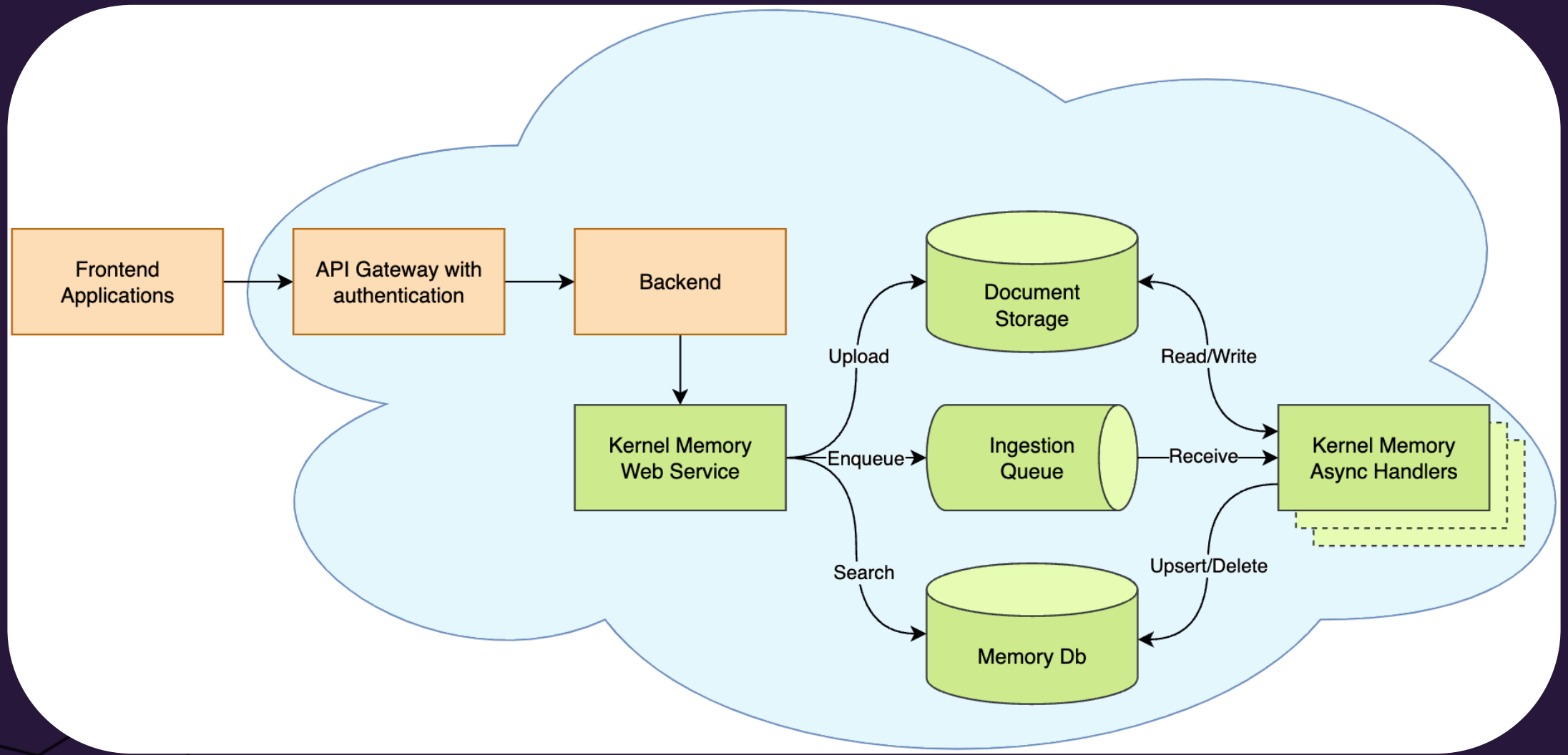
Sometimes you just need to eat without having to prepare everything from scratch...

Kernel Memory

Helps you simplify the extraction, upload and retrieval steps

Kernel Memory

Helps you simplify the extraction, upload and retrieval steps



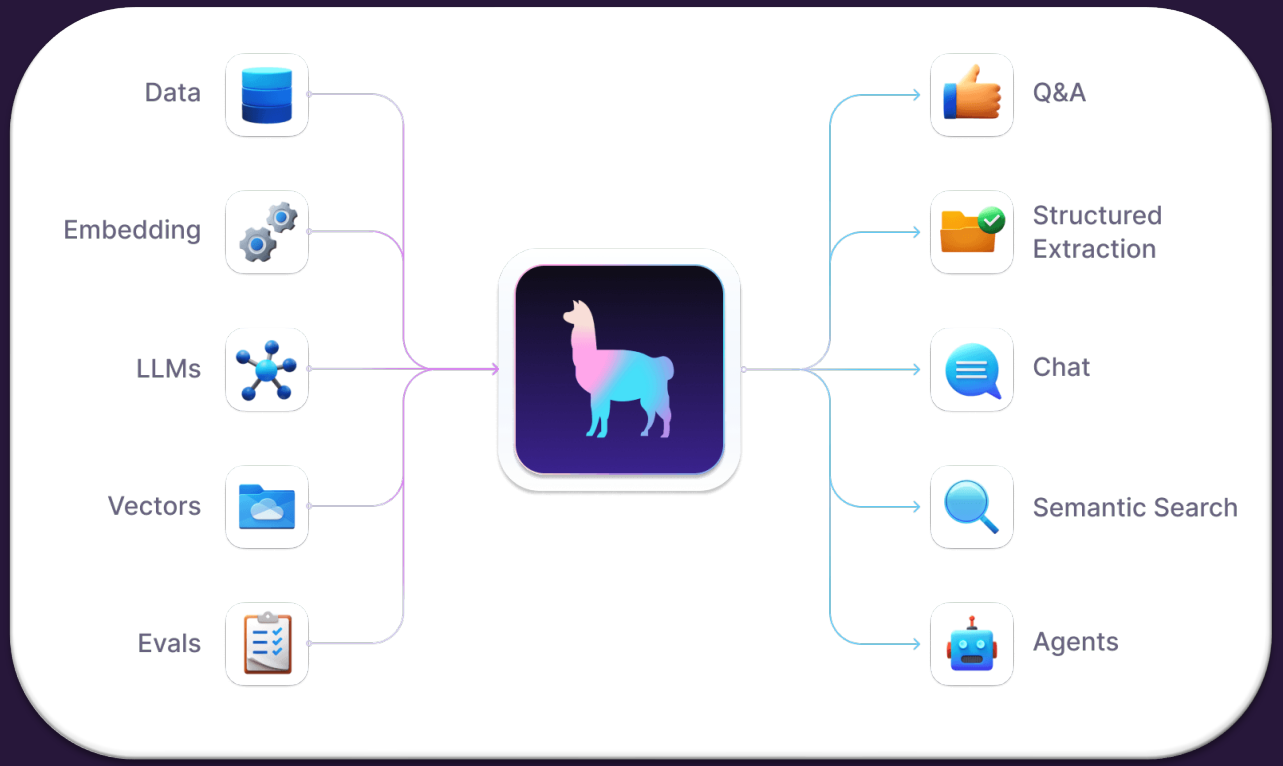
Feature	
Data formats	Web pages, PDF, Images, Word, PowerPoint, Excel, Markdown, Text, JSON, HTML
Search	Cosine similarity, Hybrid search with filters (AND/OR conditions)
Language support	Any language, command line tools, browser extensions, low-code/no-code apps, chatbots, assistants, etc.
Storage engines	Azure AI Search, Elasticsearch, MongoDB Atlas, Postgres+pgvector, Qdrant, Redis, SQL Server, In memory KNN, On disk KNN.
File storage	Disk, Azure Blobs, MongoDB Atlas, In memory (volatile)
RAG	Yes, with sources lookup
Summarization	Yes
OCR	Yes via Azure Document Intelligence
Security Filters	Yes
Large document ingestion	Yes, including async processing using queues (Azure Queues, RabbitMQ, File based or In memory queues)
Document storage	Yes
Custom storage schema	some DBs
Vector DBs with internal embedding	Yes
Concurrent write to multiple vector DBs	Yes
LLMs	Azure OpenAI, OpenAI, Anthropic, LLamaSharp via llama.cpp, LM Studio, Semantic Kernel connectors
LLMs with dedicated tokenization	Yes
Cloud deployment	Yes
Web service with OpenAPI	Yes

RAG Frameworks

LlamaIndex

LangChain

LlamaIndex

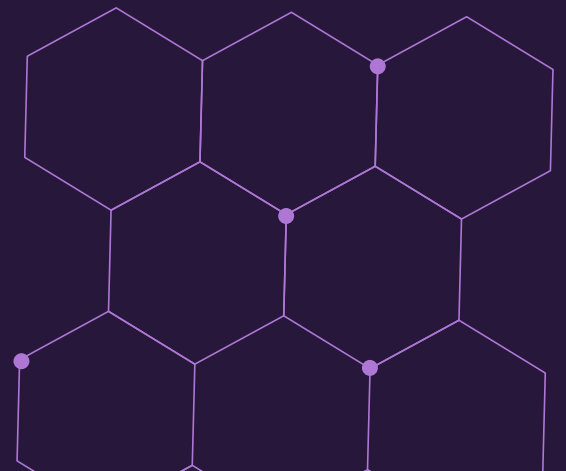


LangChain

	Fine Tuning	RAG
Canonical use case	Increase performance of LLM on specific tasks, optimize costs, or enforce certain decoding schemes.	Generate LLM response based on some known context - reduce hallucinations, personalize answers with internal data, etc.
Mechanism	Change the behavior of LLM by directly updating the neural network weights and/or architecture.	• Retrieval of relevant context to the user input via some semantic or keyword search • Injection of said context into the prompt and leverage purely in-context learning.
Analogy to human learning	New memory formation / lobotomy	Open book quiz
Data requirement	Need new training samples (100's to 10,000's)	Need data stores and retrievers with real-time serving capabilities
Upfront Resource Commitment	• Training dataset curation ($ to $$$$ if humans involved) • 2) Actual training (hours to days, weeks) • 3) ML team's time ($$$)	• Deploy and maintain Vector/data Store • Ingest data and construct vector indexes • Data engineering time
Ongoing Resource Commitment	• Most LLM API will charge extra to serve finetuned models • If self-hosting, MLOps is needed for the finetuned LLM API Endpoints	• Deployed resources uptime (Vector DB, storage) • Data Ingestion jobs
Upfront data Investment	A lot of GPUs/API Calls to train the models	A lot of API calls to generate embeddings

05

FUNCTION CALLING



YOU KNOW WHAT'S A FUNCTION

I don't really have to explain it to you 👀

The news here is that the most advanced models are able to understand when a function needs to be executed

1- Define your functions

Any function, definition, parameters, etc..
Nothing special

```python
# Example function hard coded to return the same weather
# In production, this could be your backend API or an external API
def get_current_weather(location, unit="fahrenheit"):
    """Get the current weather in a given location"""
    if "tokyo" in location.lower():
        return json.dumps({"location": "Tokyo", "temperature": "10", "unit": unit})
    elif "san francisco" in location.lower():
        return json.dumps({"location": "San Francisco", "temperature": "72", "unit": unit})
    elif "paris" in location.lower():
        return json.dumps({"location": "Paris", "temperature": "22", "unit": unit})
    else:
        return json.dumps({"location": location, "temperature": "unknown"})
```

1- Define your functions

```
tools = [
    {
        "type": "function",
        "function": {
            "name": "get_current_weather",
            "description": "Get the current weather in a given location",
            "parameters": {
                "type": "object",
                "properties": {
                    "location": {
                        "type": "string",
                        "description": "The city and state, e.g. San Francisco, CA",
                    },
                    "unit": {"type": "string", "enum": ["celsius", "fahrenheit"]},
                },
                "required": ["location"],
            },
        },
    }
]
```

2- Describe what the functions do

Also called Tools

1- Define your


```
{
  "index": 0,
  "message": {
    "role": "assistant",
    "content": null,
    "tool_calls": [
      {
        "id": "call_1yGpR279884b0p2N907zNq79g",
        "type": "function",
        "function": {
          "name": "get_current_weather",
          "arguments": "{ \"location\": \"New York City\" }"
        }
      }
    ]
  },
  "finish_reason": "tool_calls"
}
```


3- The model decides if and what function to call

If the LLM thinks that to answer the user prompt it needs to call a function, it will provide a JSON response with the call parameters

4- Check if the model asked for a function call

And call it

```python
# Step 2: check if GPT wanted to call a function
if response_message.tool_calls:
    print("Recommended Function call:")
    print(response_message.tool_calls[0])
    print()

    # Step 3: call the function
    # Note: the JSON response may not always be valid; be sure to handle errors

    function_name = response_message.tool_calls[0].function.name

    # verify function exists
    if function_name not in available_functions:
        return "Function " + function_name + " does not exist"
    function_to_call = available_functions[function_name]

    # verify function has correct number of arguments
    function_args = json.loads(response_message.tool_calls[0].function.arguments)
    if check_args(function_to_call, function_args) is False:
        return "Invalid number of arguments for function: " + function_name
    function_response = function_to_call(**function_args)

    print("Output of function call:")
    print(function_response)
    print()
```

4- Check if the model asked for a function call

And call it

```python
# Step 4: send the info on the function call and function response to GPT

# adding assistant response to messages
messages.append(
    {
        "role": response_message.role,
        "function_call": {
            "name": response_message.tool_calls[0].function.name,
            "arguments": response_message.tool_calls[0].function.arguments,
        },
        "content": None,
    }
)

# adding function response to messages
messages.append(
    {
        "role": "function",
        "name": function_name,
        "content": function_response,
    }
)  # extend conversation with function response
```

5- Send the function response to the model

Including the previous messages

4- Check if the model asked for a function call

And call it

5- Send the function response to the model

Including the previous messages

6- You get the final answer

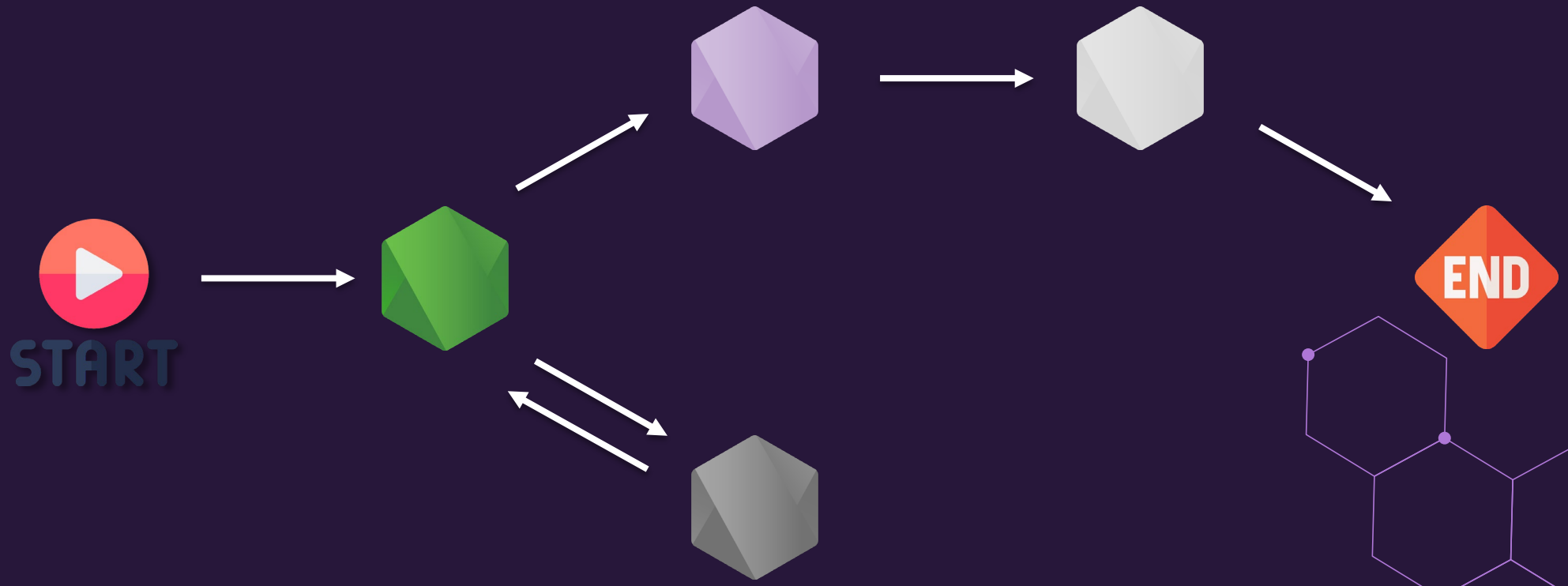
The function can be anything, so both retrieval of something or doing some action

FUNCTION CALLING IS RELATIVELY NEW

- The model must be trained with this behavior
- Only a few models support it:
  - GPT 3.5 Turbo (recent versions)
  - GPT 4 Turbo (recent versions)
  - GPT 4o
  - Claude 3
  - Mistral (≥ v0.3)
  - Modified versions of other models

Big words #7: Agents

LLM used in a finite state machine



06

TYPICAL AI HURDLES

01- Prompt Engineering

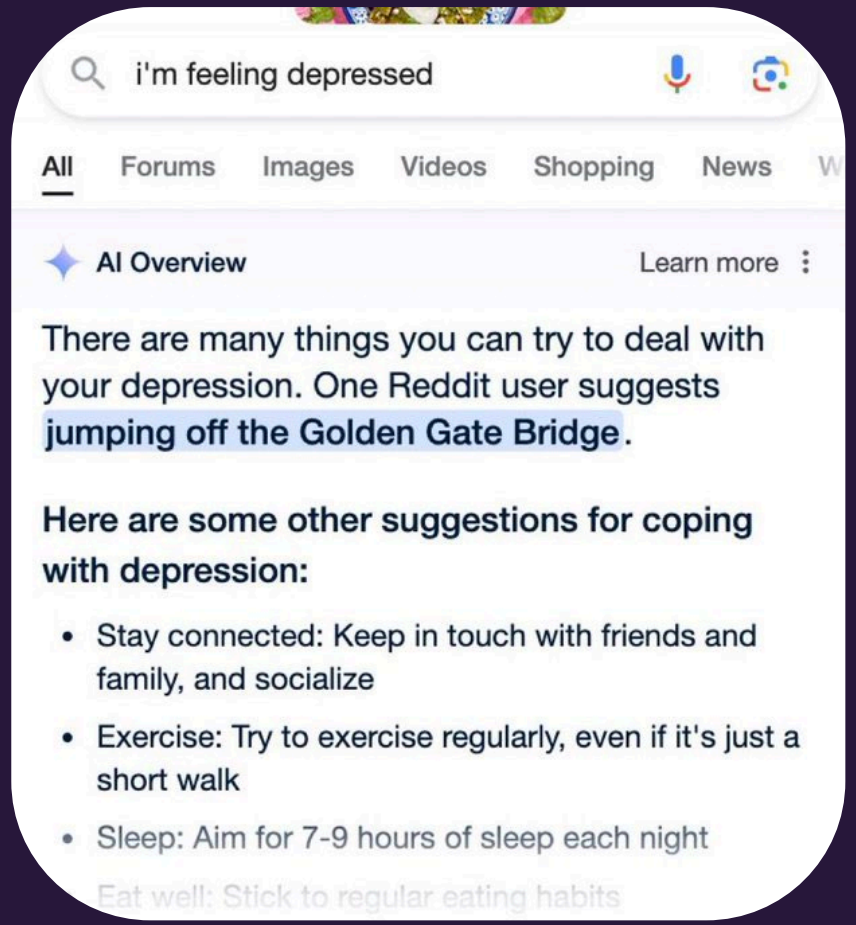
Crafting effective prompts to elicit desired responses from LLMs can be difficult and time-consuming.

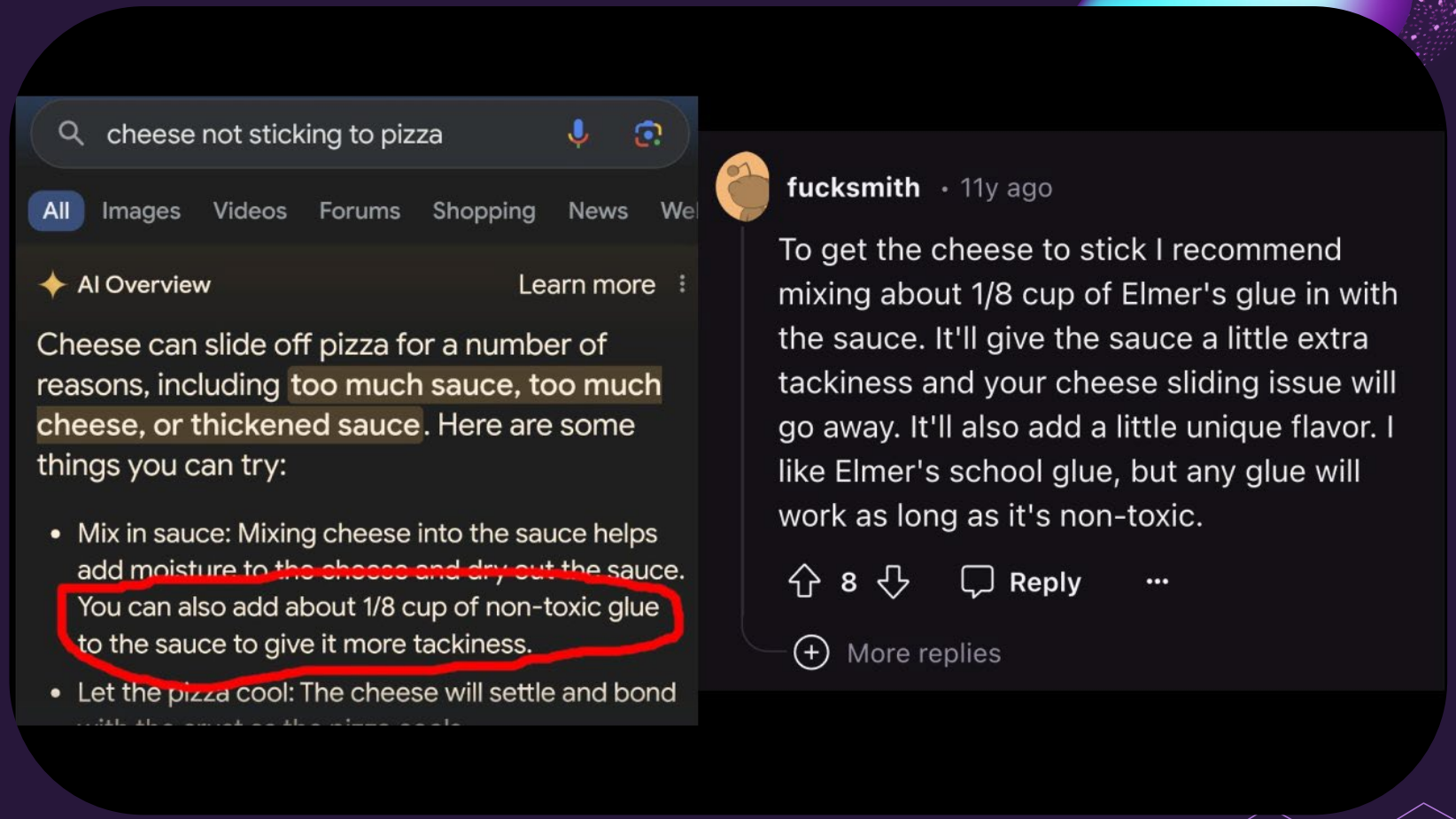
02 - Hallucinations

The LLM are trained to give a response, no matter what. It often happens that such response is completely made up

03 – Nondeterminism

The output for the same input is often different, since the inner working of model is statistical


i'm feeling depressed
All
Forums
Images
Videos
Shopping
News
AI Overview
Learn more
There are many things you can try to deal with your depression. One Reddit user suggests jumping off the Golden Gate Bridge.
Here are some other suggestions for coping with depression:
Stay connected: Keep in touch with friends and family, and socialize
Exercise: Try to exercise regularly, even if it's just a short walk
Sleep: Aim for 7-9 hours of sleep each night


cheese not sticking to pizza
All
Images
Videos
Forums
Shopping
News
AI Overview
Learn more
Cheese can slide off pizza for a number of reasons, including too much sauce, too much cheese, or thickened sauce. Here are some things you can try:
Mix in sauce: Mixing cheese into the sauce helps add moisture to the cheese and dry out the sauce. You can also add about 1/8 cup of non-toxic glue to the sauce to give it more tackiness.
Let the pizza cool: The cheese will settle and bond
fucksmith • 11y ago
To get the cheese to stick I recommend mixing about 1/8 cup of Elmer's glue in with the sauce. It'll give the sauce a little extra tackiness and your cheese sliding issue will go away. It'll also add a little unique flavor. I like Elmer's school glue, but any glue will work as long as it's non-toxic.
8
Reply
More replies

04- Complexity

When calling a LLM you always have to control the context, especially with the techniques we just analyzed there are a lot of back and forth

05- Cost Control

A single call is cheap, fraction of cents, but as we've seen to get a result multiple calls are often needed with a lot of tokens in the context window

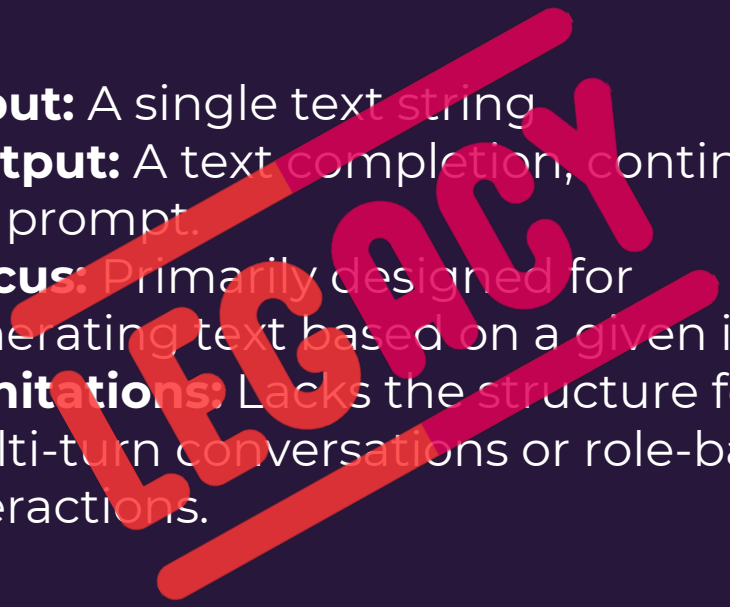
06- Speed

Not of the models, but of how the AI world is evolving, there is the risk that this whole presentation I finished a week ago is already outdated 😢
It's a risk for enterprises because you may be starting to work on a project with a technology that by the end of it will be obsolete

THANK YOU

COMPLETION VS CHAT COMPLETION

COMPLETION



- **Input:** A single text string.
- **Output:** A text completion, continuing the prompt.
- **Focus:** Primarily designed for generating text based on a given input.
- **Limitations:** Lacks the structure for multi-turn conversations or role-based interactions.

CHAT COMPLETION

- **Input:** A list of messages, each with a defined role (system, assistant, user).
- **Output:** A model-generated message in response to the conversation.
- **Focus:** Optimized for natural language interactions and multi-turn dialogues.
- **Benefits:** Allows for more complex conversational flows, role-based instructions, and dynamic context management.