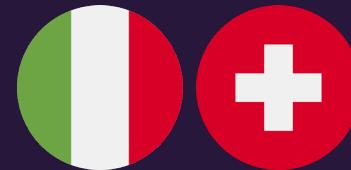


RAG And AI for the data professional

Emanuele Meazzo - 2024

WHO IS YOUR SPEAKER TODAY?

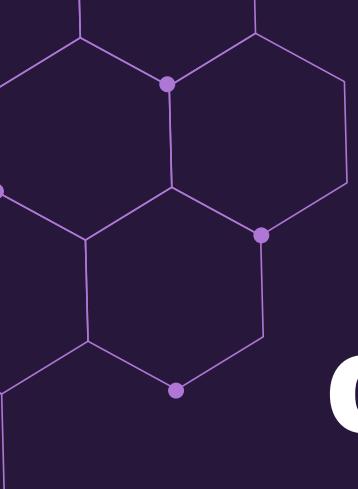
EMANUELE MEAZZO



- 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



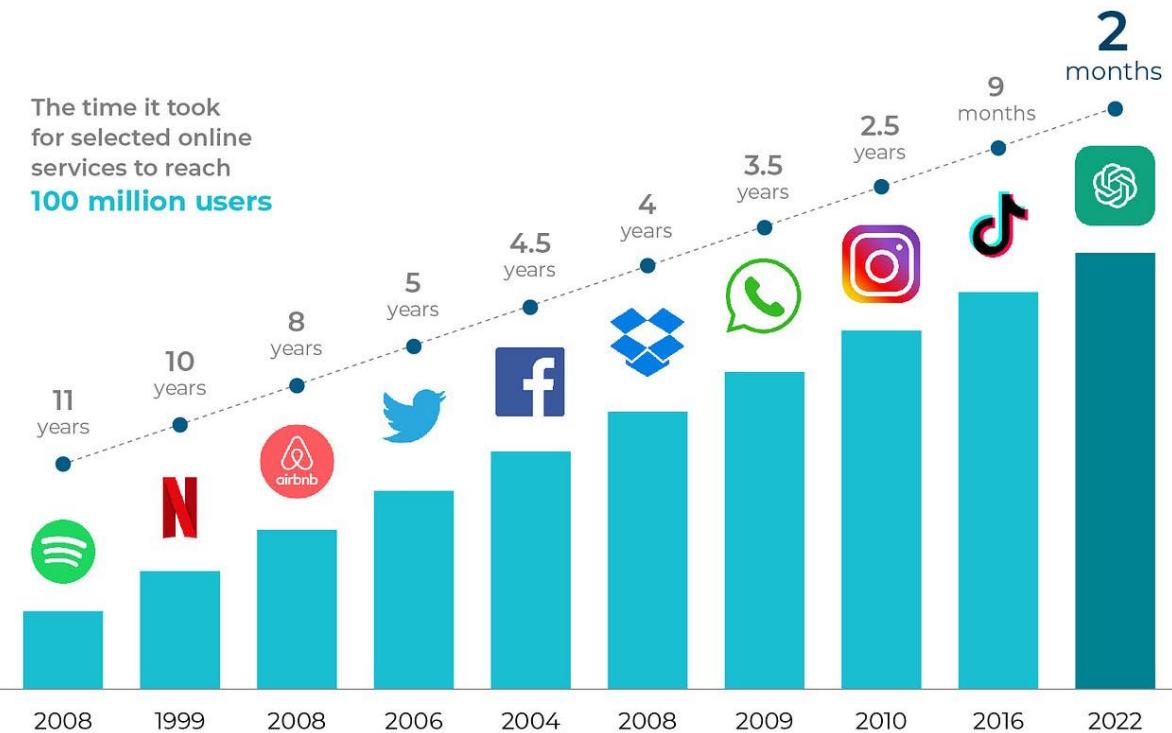
**What has changed in AI that
made the hype go up lately?**

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



Chat-GPT sprints to 100 million users

The time it took
for selected online
services to reach
100 million users

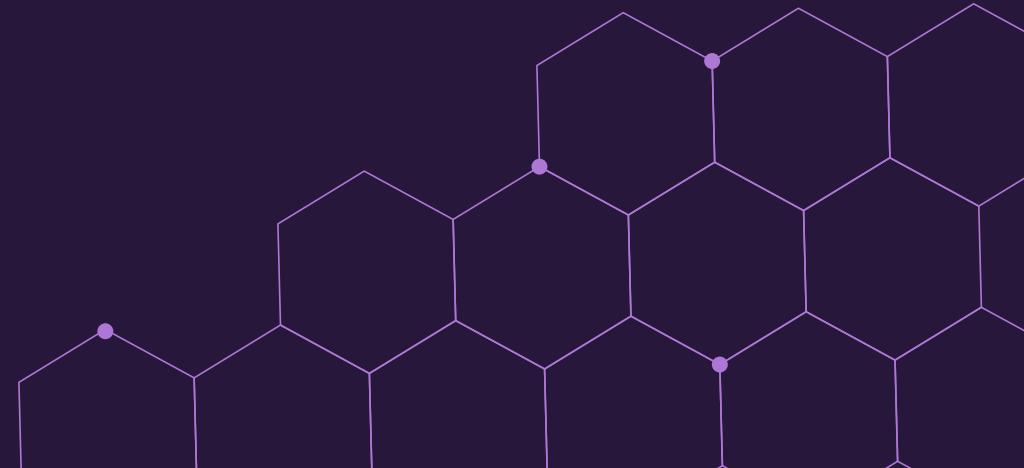
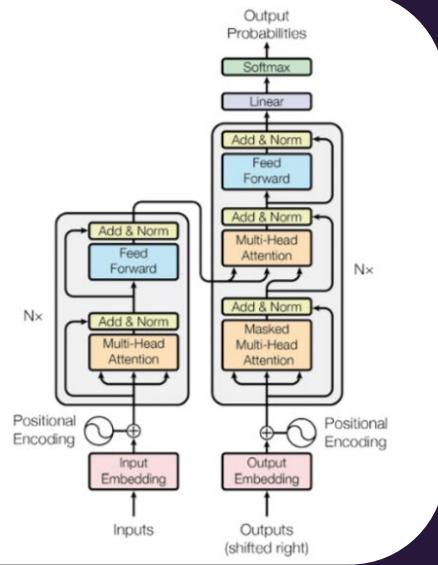


Source: World of Statistics

Driving the hype: LLM

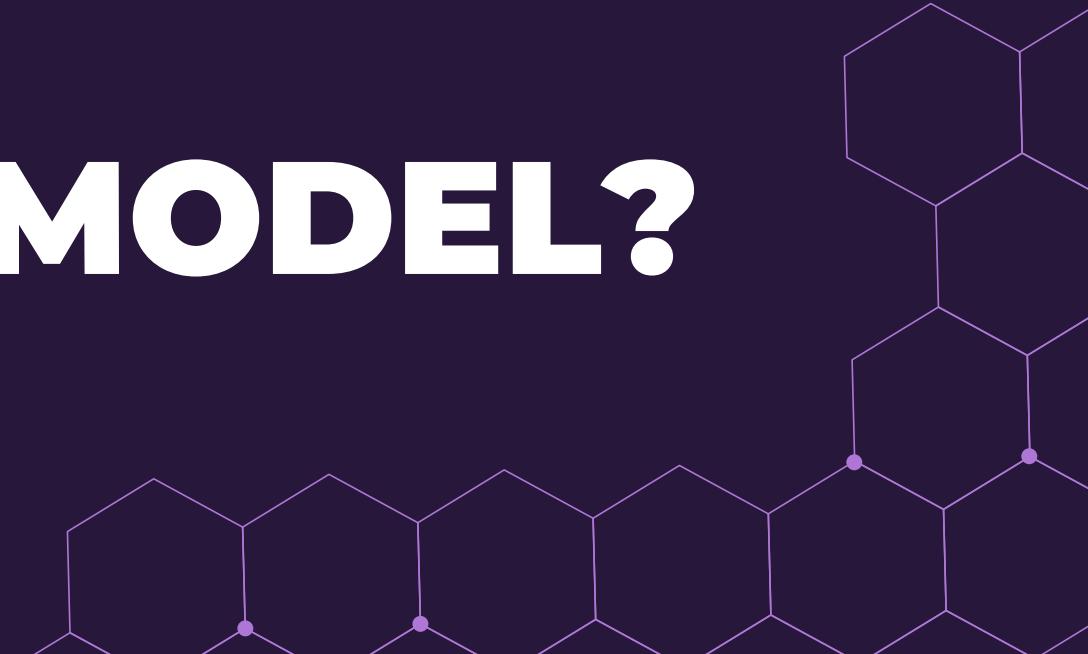
- ChatGPT is a (service based on a) LLM
- LLM: **Large Language Model**
- GPT: Generative **Pretrained** Transformer
- Transformer: first described in famous “**Attention is all you need**” Google Paper in 2017
- Since ChatGPT has demonstrated the viability of LLM as a **successful commercial product**,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

Transformer Attention Is All You Need

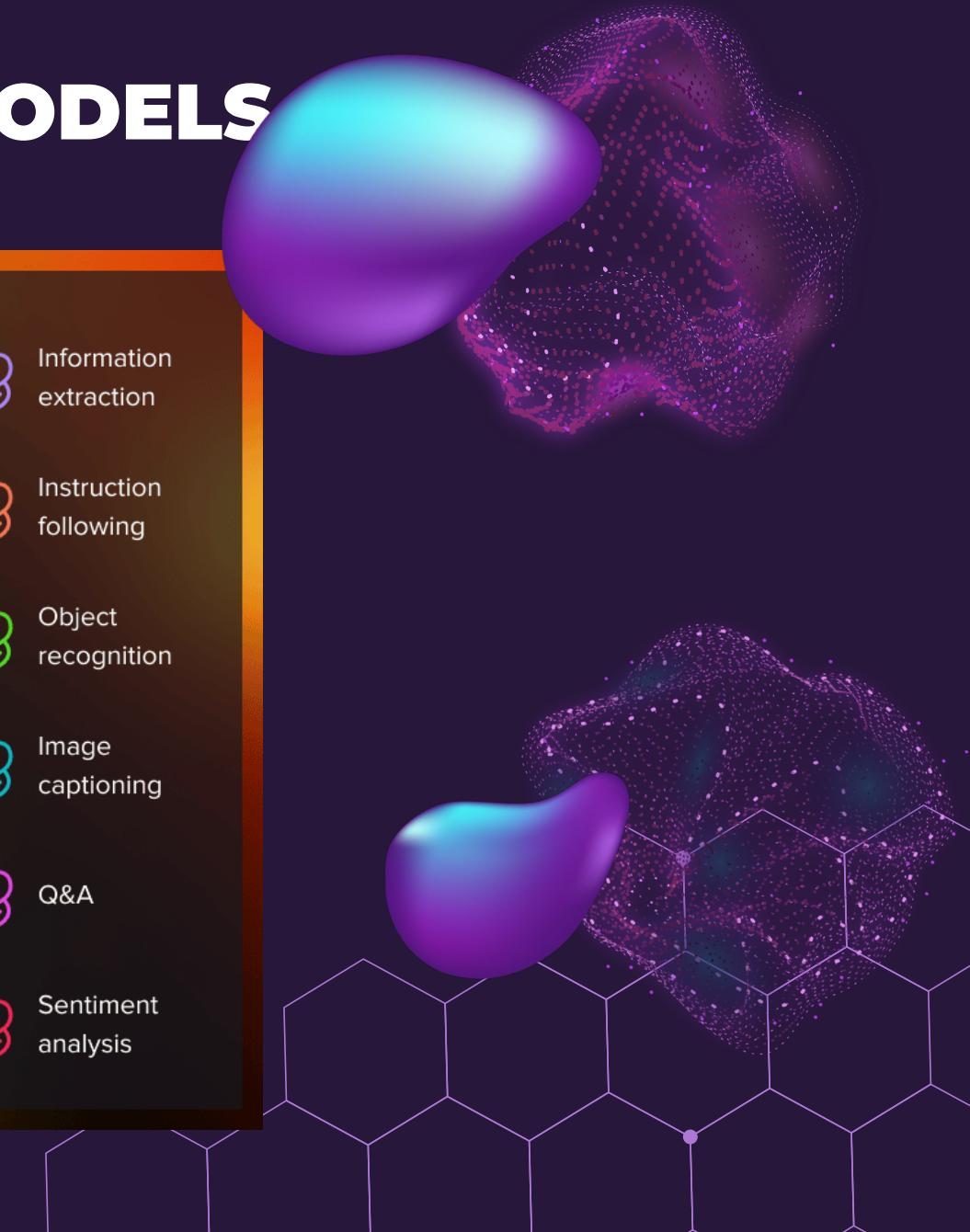
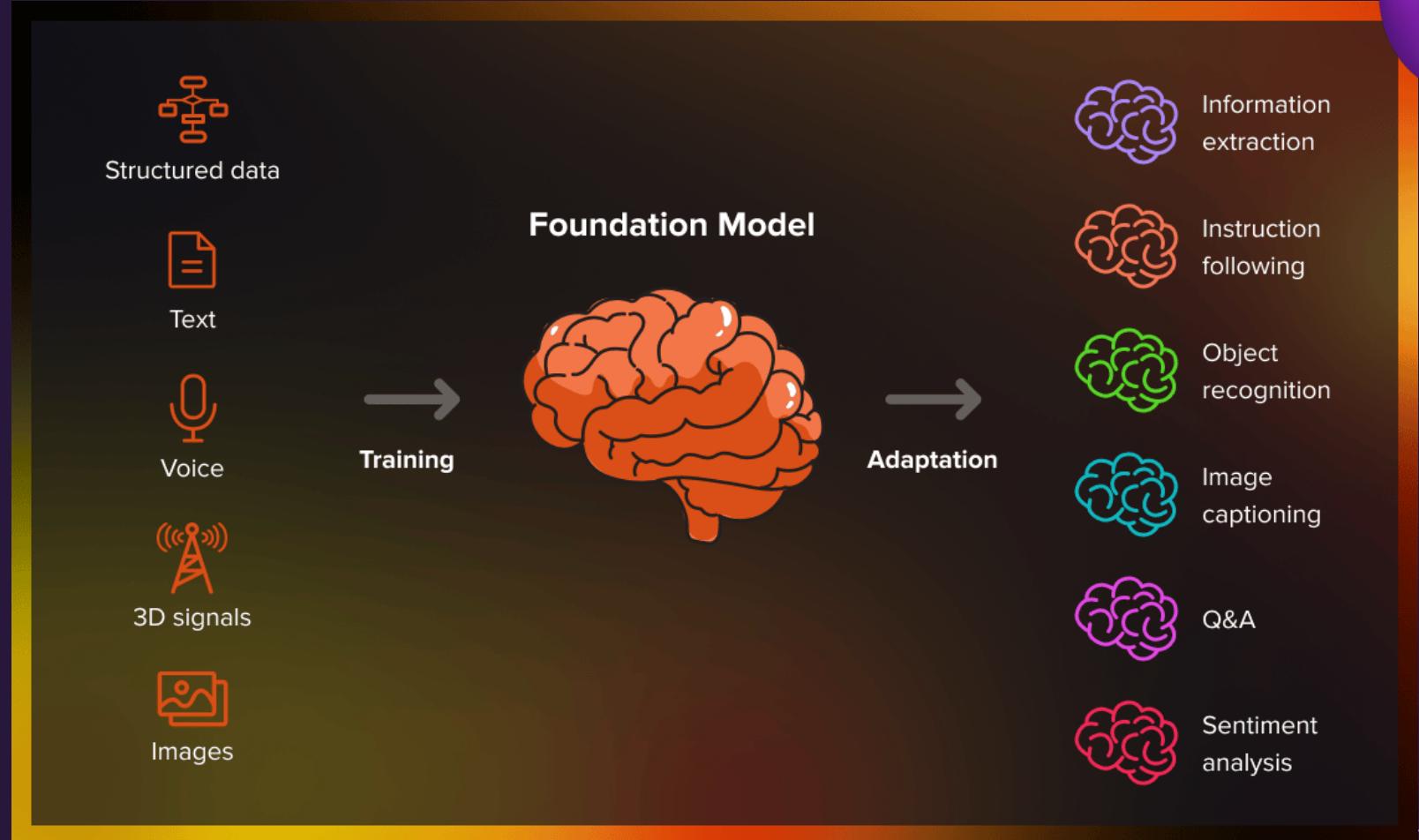


01

WHAT'S A FOUNDATION MODEL?

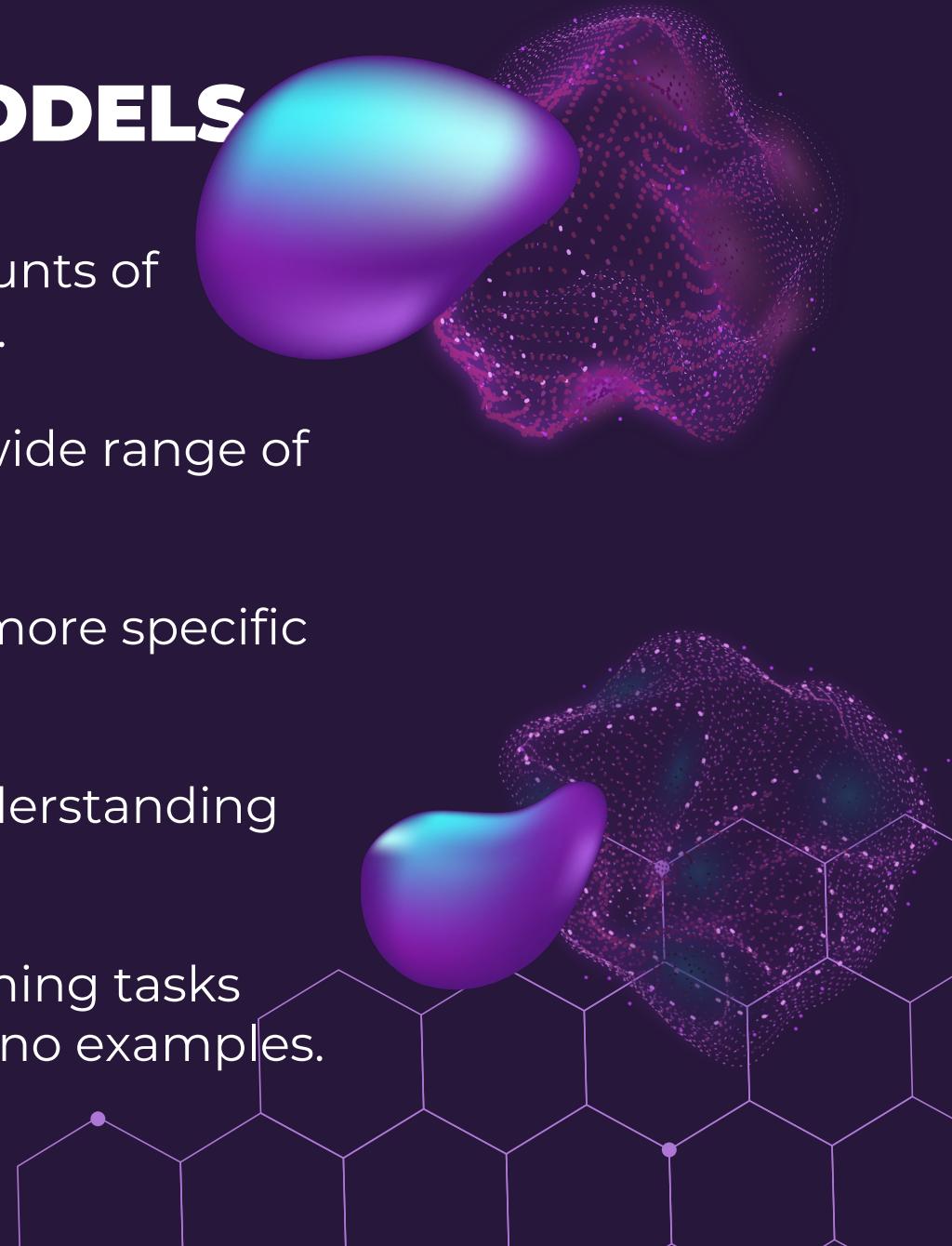


FOUNDATION MODELS



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



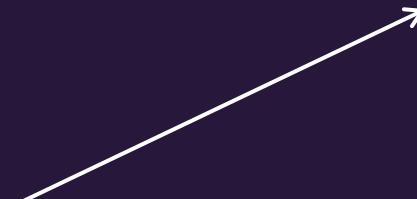
All the text you can use



All the images you can use



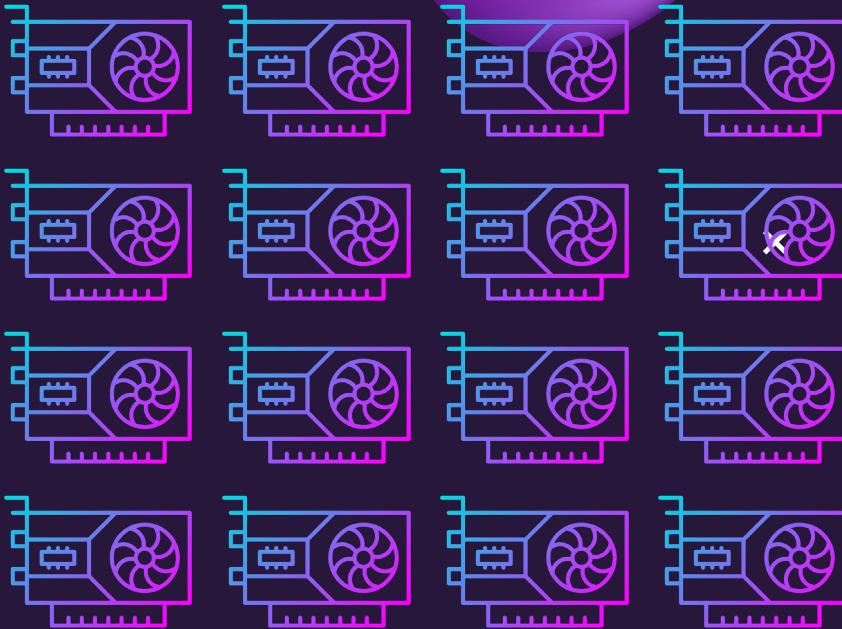
All the audio you can use



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

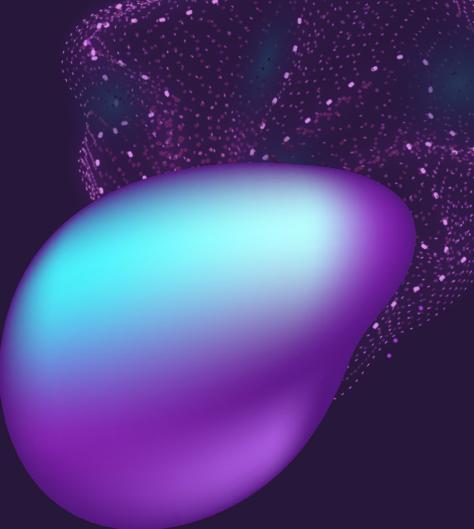
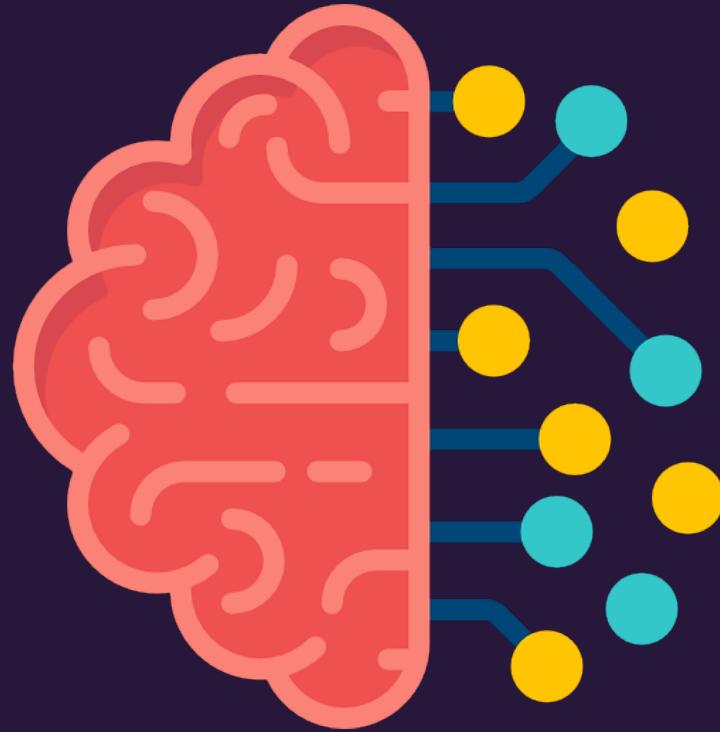
Magic mathematical model
(At the end of the day lots
and lots of matrix operations)

We have no idea how it
works, it just does

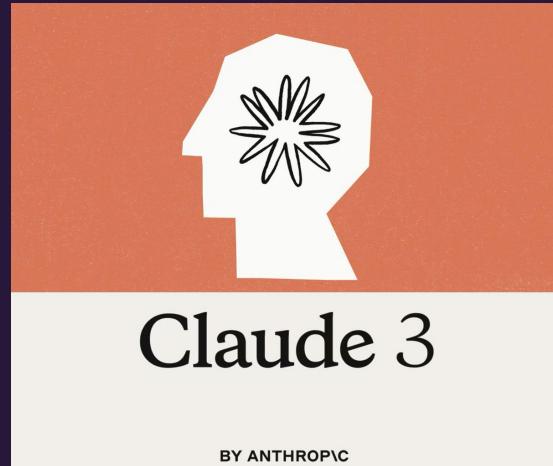


Millions and millions of \$ in GPUs and power to
run the magic mathematical model for weeks

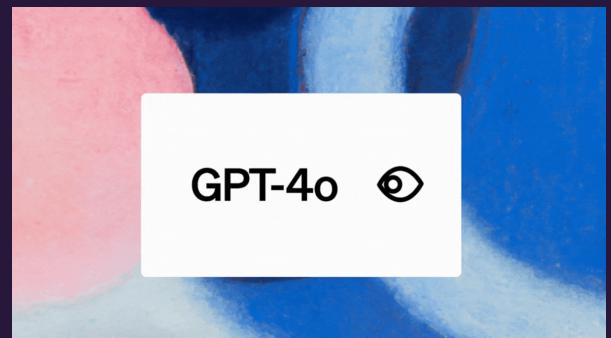
**YOU CAN THINK
ABOUT IT AS A
STATIC SNAPSHOT
OF A BRAIN
NEURON STATE
CONFIGURATION**



FAMOUS FOUNDATION MODELS



Gemini

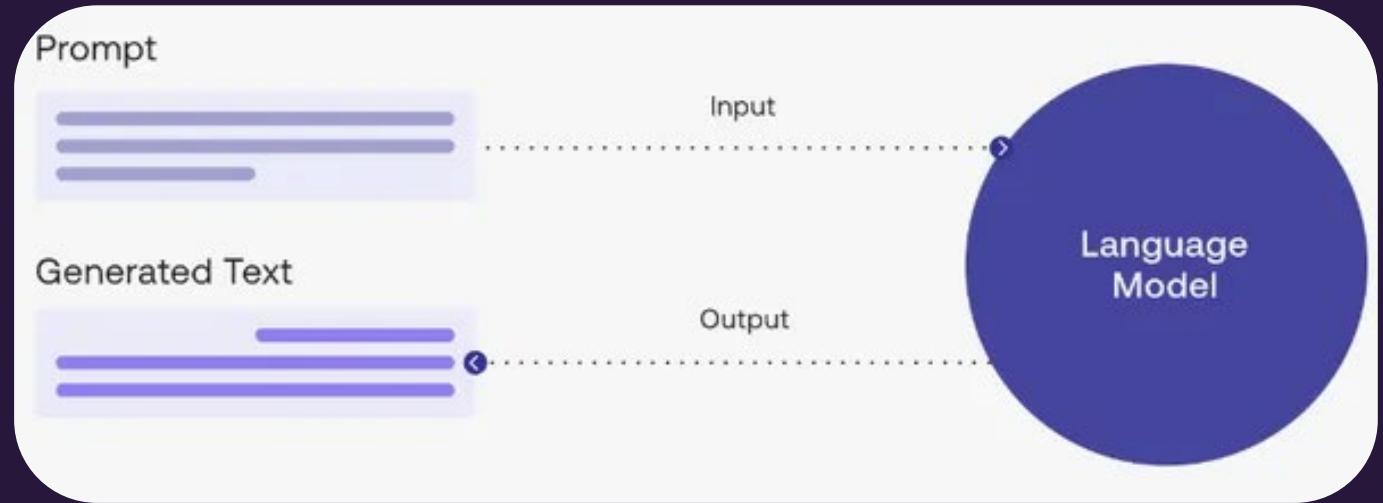




ALL THE BIG WORDS

Models 651,238 Filter by name Full-text search Sort: Trending

- meta-llama/Meta-Llama-3-8B**
Text Generation • Updated 3 days ago • 950k • 3.64k
- meta-llama/Meta-Llama-3-8B-Instruct**
Text Generation • Updated 2 days ago • 1.69M • 2.01k
- deepseek-ai/DeepSeek-V2**
Text Generation • Updated 3 days ago • 7.44k • 162
- apple/OpenELM**
Updated 11 days ago • 1.27k
- TheMistoAI/MistoLine**
Text-to-Image • Updated about 13 hours ago • 4.02k • 120
- gradientai/Llama-3-8B-Instruct-Gradient-1048k**
Text Generation • Updated 1 day ago • 18.5k • 557
- nvidia/Llama3-ChatQA-1.5-70B**
Text Generation • Updated 4 days ago • 2.28k • 227
- deepseek-ai/DeepSeek-V2-Chat**
Text Generation • Updated 3 days ago • 37.9k • 258
- nvidia/Llama3-ChatQA-1.5-8B**
Text Generation • Updated 4 days ago • 14.4k • 364
- google/timesfm-1.0-200m**
Updated 2 days ago • 154
- mlabonne/Meta-Llama-3-120B-Instruct**
Text Generation • Updated 3 days ago • 1.57k • 163
- mustafaaljadery/gemma-2B-10M**
Updated 3 days ago • 40k • 119
- refuelai/Llama-3-Refueled**
Text Generation • Updated 3 days ago • 535 • 103
- shenzhi-wang/Llama3-8B-Chinese-Chat**
Text Generation • Updated 2 days ago • 10.9k • 411



Big words #1: **PROMPT**

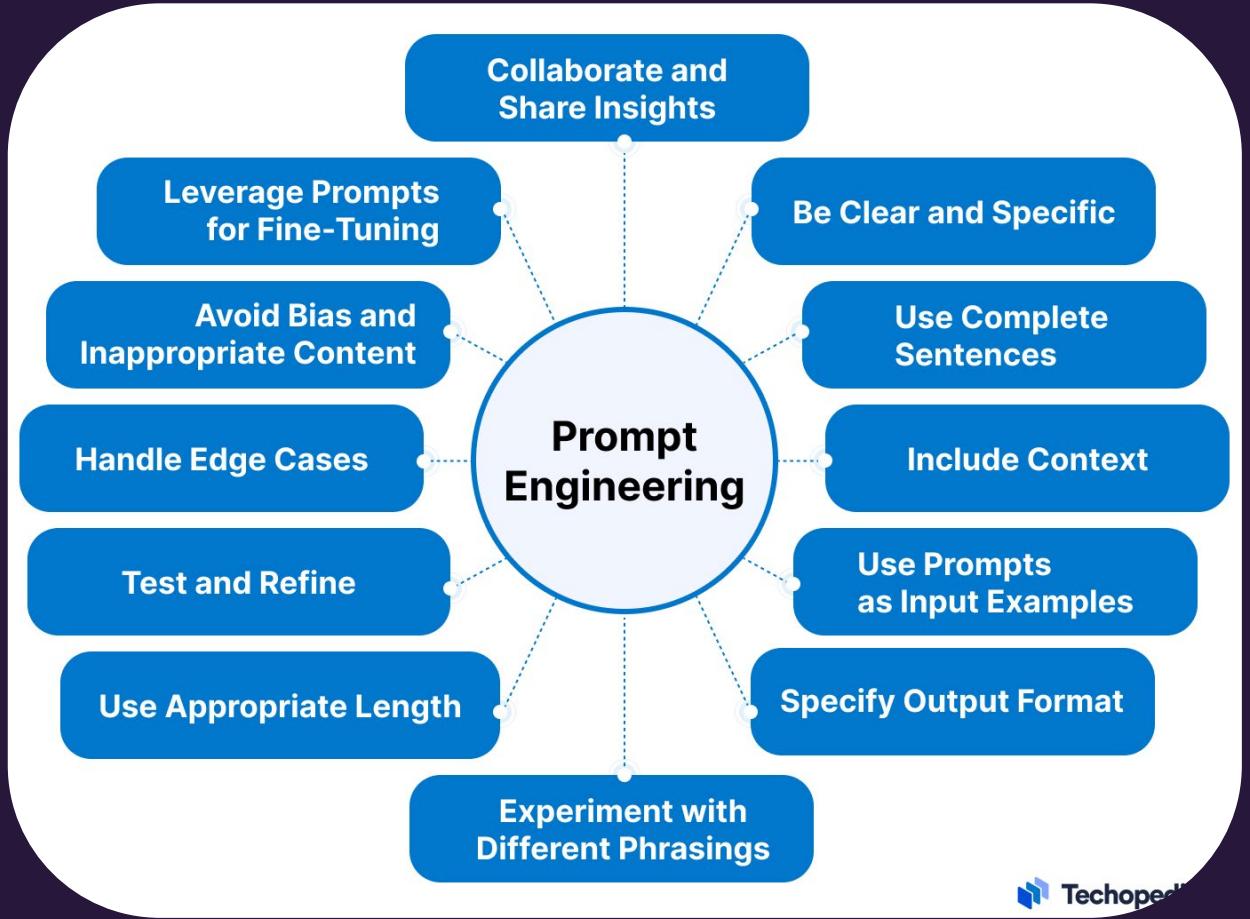
The prompt is the “command” you send to the LLM to get a response, it can be as simple as asking a question

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

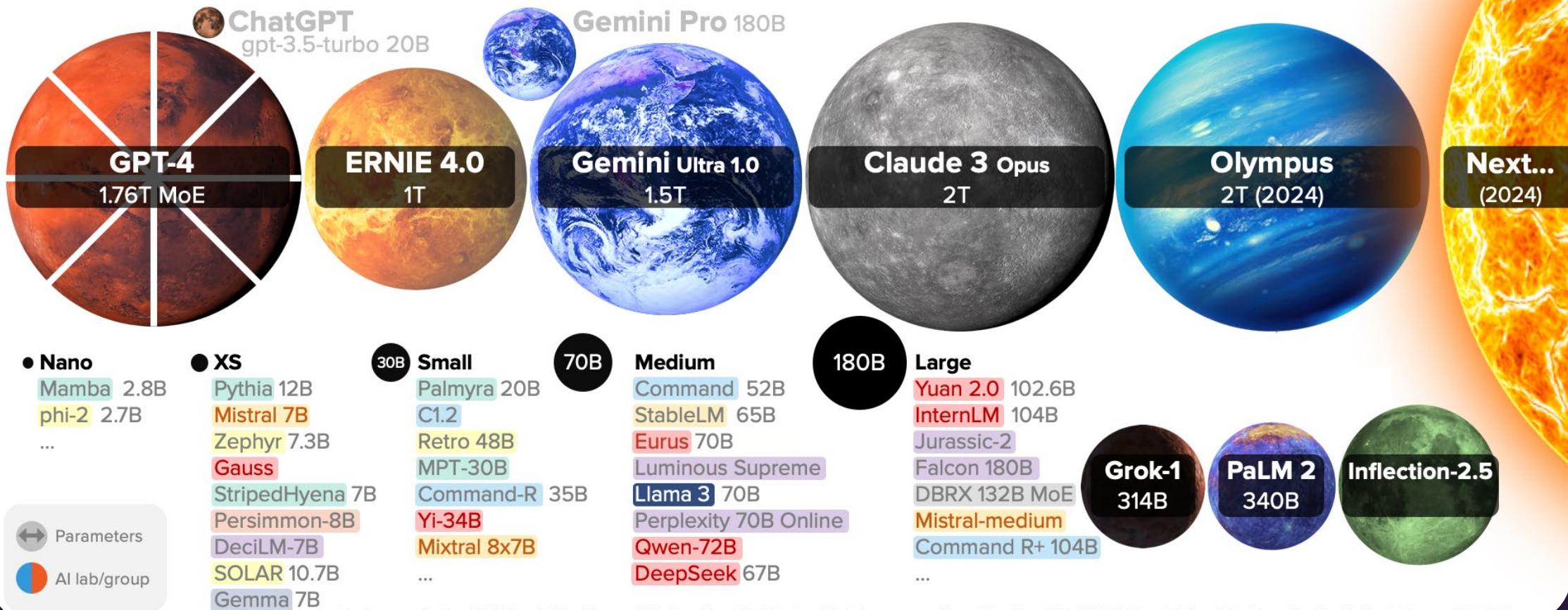


Techoped

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

LARGE LANGUAGE MODEL HIGHLIGHTS (APR/2024)



Big word #3: TOKENS

x

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.

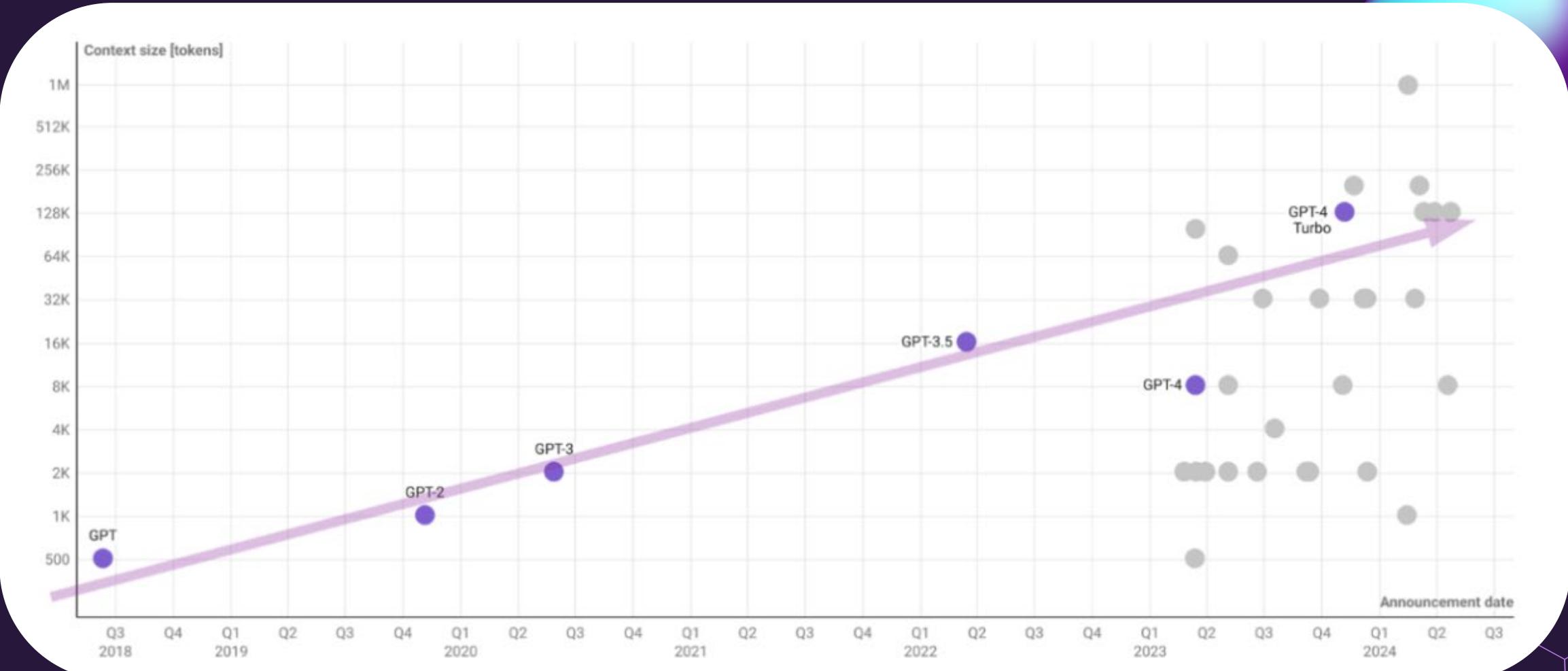
LLMs use tokenization to convert input text into a sequence of tokens that the model can understand.

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!?



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.

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.

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?



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

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





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

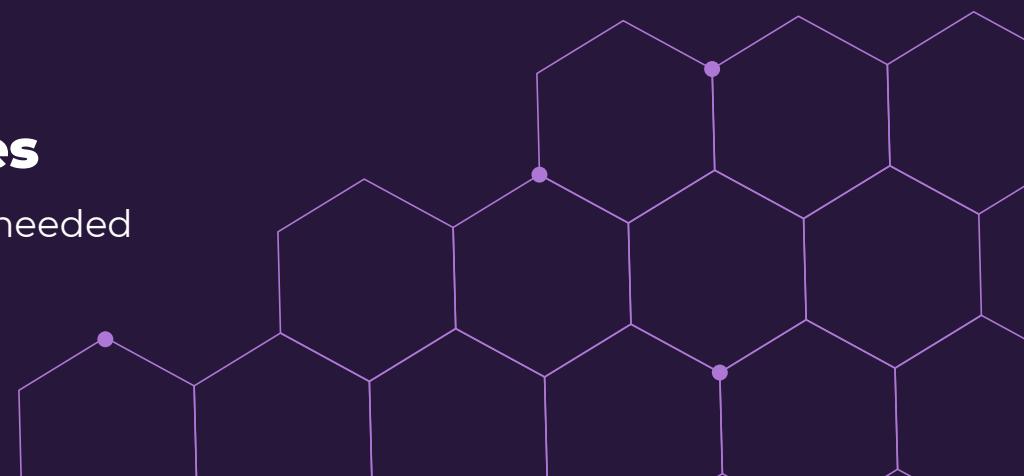
Only option for closed-source models

Many vendors will happily host open-source models too



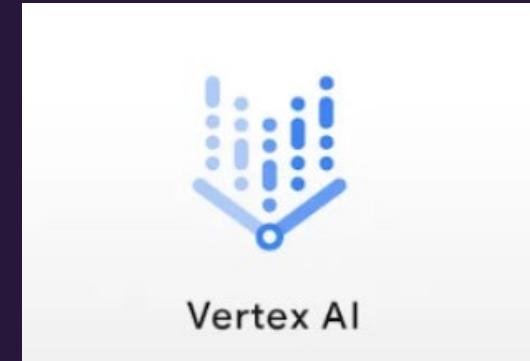
Self-host the model on your compute resources

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

Find the right model to build your custom AI solution

Show filters

Announcements

Mistral Small is now available!

Mistral AI's smallest yet highly efficient model, now available on Azure

[View models](#) [Read blog](#)



Phi-3 is now available

Microsoft's Phi-3-mini SLMs offer groundbreaking performance at a small size



Build the future of AI with Meta Llama 3

Serverless APIs for Meta-Llama-3-8B-Instruct and Meta-Llama-3-70B-Instruct models

[View models](#) [Read blog](#)



[All filters](#) [Collections](#) [Inference tasks](#) [Fine-tuning tasks](#) [Licenses](#)

Search

Models 1639

gpt-4 Chat completion	dall-e-3 Text to image	gpt-35-turbo-instruct Chat completion	davinci-002 Completions	text-embedding-ada-002 Embeddings	gpt-4-32k Chat completion	gpt-35-turbo-16k Chat completion
gpt-35-turbo Chat completion	babbage-002 Completions	mistralai-Mistral-7B-Instruct-v... Chat completion	mistral-community-Mixtral-8x... Text generation	mistralai-Mixtral-8x7B-Instruct... Chat completion	mistralai-Mistral-7B-Instruct-v01 Chat completion	mistralai-Mixtral-8x7B-v01 Text generation
mistralai-Mistral-7B-v01 Text generation	Mistral-small Chat completion	mistralai-Mixtral-8x22B-v0-1 Text generation	mistralai-Mixtral-8x22B-Instruc... Chat completion	Mistral-large Chat completion	Llama-2-7b-chat Chat completion	Llama-2-13b Text generation
CodeLlama-7b-hf Text generation	CodeLlama-7b-Python-hf Text generation	CodeLlama-34b-hf Text generation	CodeLlama-34b-Python-hf Text generation	CodeLlama-34b-Instruct-hf Text generation	CodeLlama-13b-hf Text generation	CodeLlama-13b-Instruct-hf Text generation
Llama-2-70b-chat Chat completion	Llama-2-70b Text generation	Meta-Llama-3-8B Text generation	CodeLlama-7b-Instruct-hf Text generation	CodeLlama-13b-Python-hf Text generation	Meta-Llama-3-70B Text generation	Llama-2-7b Text generation
Llama-2-13b-chat Chat completion	Meta-Llama-3-70B-Instruct Chat completion	Meta-Llama-3-8B-Instruct Chat completion	Nemotron-3-8B-Chat-SteerLM Text generation	Nemotron-3-8B-Chat-RLHF Text generation	Nemotron-3-8B-Chat-SFT Text generation	Nemotron-3-8B-QA-4k Text generation
Nemotron-3-8B-Base-4k Text generation	Phi-3-mini-4k-instruct Chat completion	Phi-3-mini-128k-instruct Chat completion	Cohere-embed-v3-multilingual Embeddings	Cohere-embed-v3-english Embeddings	Cohere-command-r-plus Chat completion	Cohere-command-r Chat completion
Deci-DeciLM-7B-instruct Text generation	Deci-DeciLM-7B Text generation	Deci-DeciCoder-1b Text generation	tiiuae-falcon-7b-instruct Text generation	tiiuae-falcon-7b Text generation	tiiuae-falcon-40b Text generation	tiiuae-falcon-40b-instruct Text generation
tiiuae-falcon-7b Text generation	tiiuae-falcon-40b Text generation	databricks-dolly-v2-12b Text generation	openai-whisper-large Speech recognition	openai-whisper-large-v3 Speech recognition	openai-clip-vit-large-patch14 Zero-shot image classification	openai-clip-vit-base-patch32 Zero-shot image classification



Probably even way too many models

Show filters



Build the future of AI with Meta Llama 3



Serverless APIs for Meta-Llama-3-8B-Instruct and Meta-Llama-3-70B-Instruct models

[View models](#)

[Read blog](#)

Models 1639

embedding-ada-002

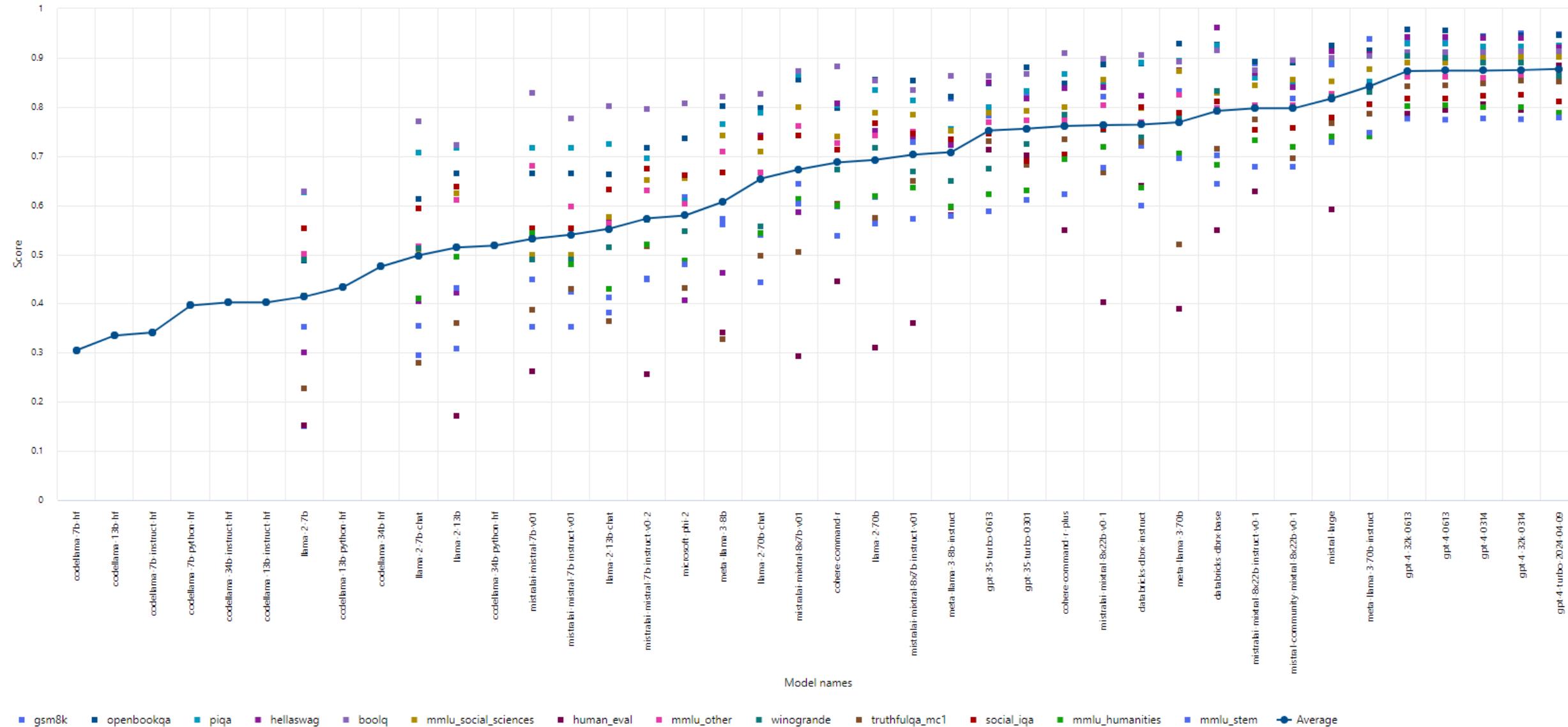
gpt-4-32k
Chat completion

gpt-35-turbo-16k
Chat completion

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

Meta-Llama-3-70B-Instruct PREVIEW

Overview Versions Artifacts

- Task: Chat completion
 - Languages: EN
 - License: custom
- Refresh Deploy View license

Description

Model Details

Meta developed and released the Meta Llama 3 family of large language models (LLMs), a collection of pretrained and instruction tuned generative text models in 8 and 70B sizes. The Llama 3 instruction tuned models are optimized for dialogue use cases and outperform many of the available open source chat models on common industry benchmarks. Further, in developing these models, we took great care to optimize helpfulness and safety.

Model developers Meta

Variations Llama 3 comes in two sizes — 8B and 70B parameters — in pre-trained and instruction tuned variants.

Input Models input text only.

Output Models generate text and code only.

Inference Samples

Serverless APIs

Provision an inference API within seconds through Models as a Service. Explore the model in the playground. Pay only for tokens consumed.

Pricing

paygo-inference-output-tokens:
\$0.01134 per 1000 tokens

paygo-inference-input-tokens:
\$0.00378 per 1000 tokens

Deploy model

Model ID

<azureml://registries/azureml-meta/models/Meta-Llama-3-70B-Instruct/versions/3>

Serverless deploy

Meta-LI

Overview

Task: CI

Refre

Descripti

Model

Meta deve
8 and 70B
common ir

Model dev

Variations

Input Mod

Output Mod

Inferenc

Serverless APIs

Provision an inference API within seconds through Models as a Service. Explore the model in the playground. Pay only for tokens consumed.

Pricing

paygo-inference-output-tokens:
\$0.01134 per 1000 tokens

paygo-inference-input-tokens:
\$0.00378 per 1000 tokens

Deploy model

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?**

Deploy model

For the selected model, the scoring script and environment are auto generated for you.
[Learn More](#)

Project [Create a new project](#)
emanuelemeazzo-4371

You have no dedicated quota. A temporary 168-hour endpoint will be created for you. Alternatively, you can [request for quota for persistent endpoints](#). [Learn more about shared quota](#).

I want to use shared quota and I acknowledge that this endpoint will be deleted in 168 hours

Virtual machine * [\(i\)](#)
Standard_NC6s_v3 6 Cores, 112 GB (RAM), 336 GB (Disk), \$3.58/hr.

Showing 6 VM sizes

Name	Category	Av...	C... ↑	(i)
<input checked="" type="radio"/> Standard_NC6s_v3 6 cores, 112GB RAM, 336GB storage	GPU	24 cores	\$3.58/hr..	(i)
<input type="radio"/> Standard_NC24ads_A100_v4 24 cores, 220GB RAM, 64GB storage	--	96 cores	\$4.59/hr..	(i)
<input type="radio"/> Standard_NC12s_v3 12 cores, 224GB RAM, 672GB storage	GPU	24 cores	\$7.16/hr..	(i)
<input type="radio"/> Standard_NC48ads_A100_v4 48 cores, 440GB RAM, 128GB storage	--	96 cores	\$9.18/hr..	(i)
<input type="radio"/> Standard_NC24s_v3 24 cores, 448GB RAM, 1344GB storage	GPU	24 cores	\$14.32/hr	(i)

Inferencing data collection [\(i\)](#)
 Disabled

x

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

Prompt
Engineering

RAG

Fine Tuning

Training
from
Scratch

Complexity/Cost/Quality(?)

Paths to get most out of a LLM

Prompt
Engineering

RAG

Fine Tuning

Training
from Scratch

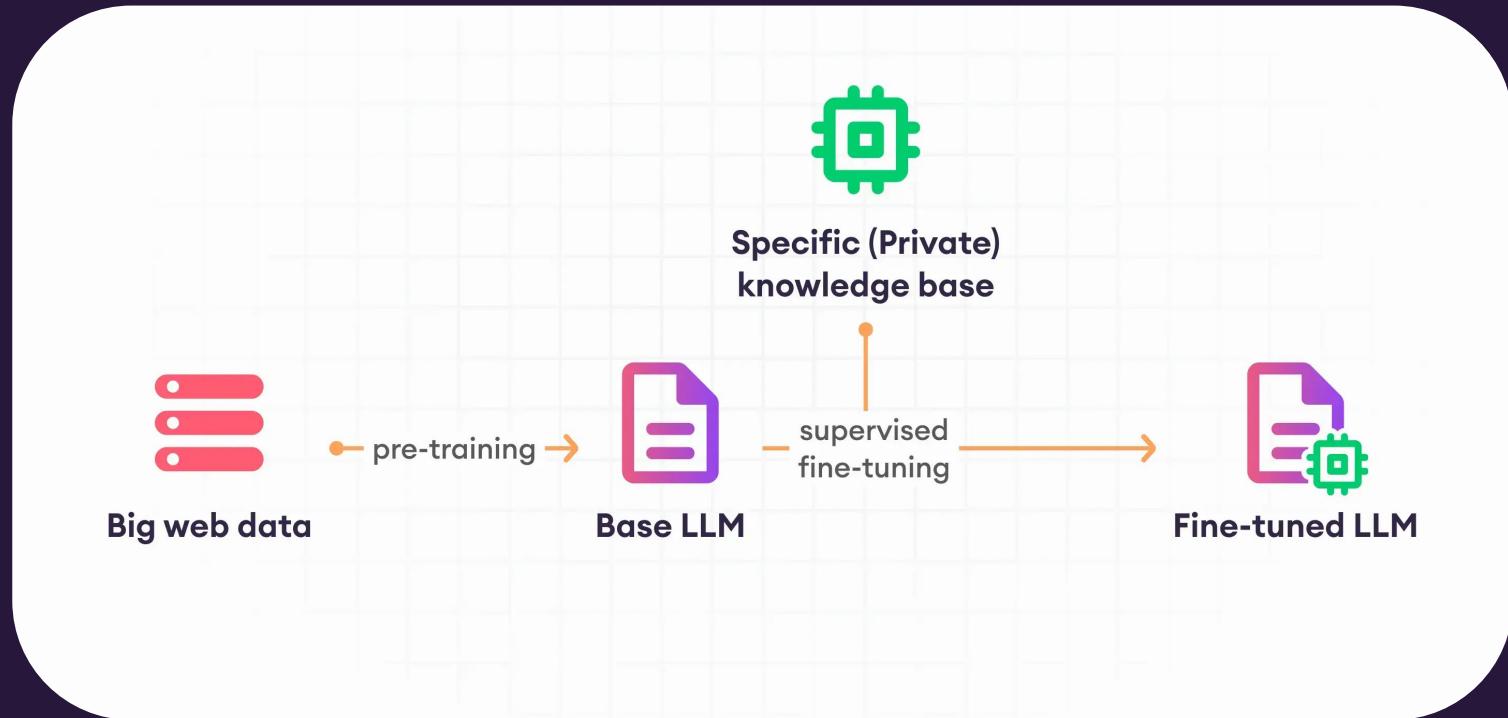
Complexity/Cost/Quality(?)

x

x

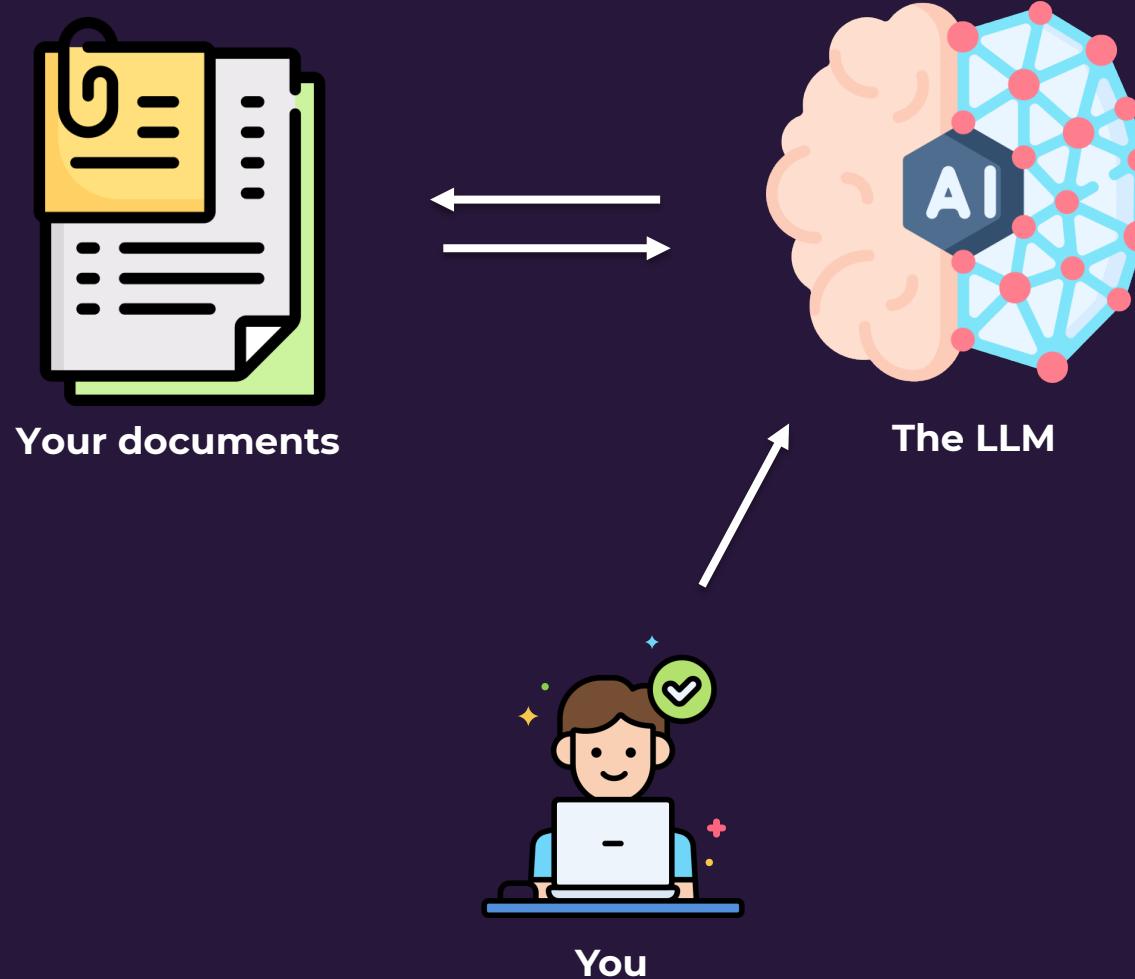
Fine tuning

- ◆ **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



04

RAG





Magic words:

Embeddings

Vectors

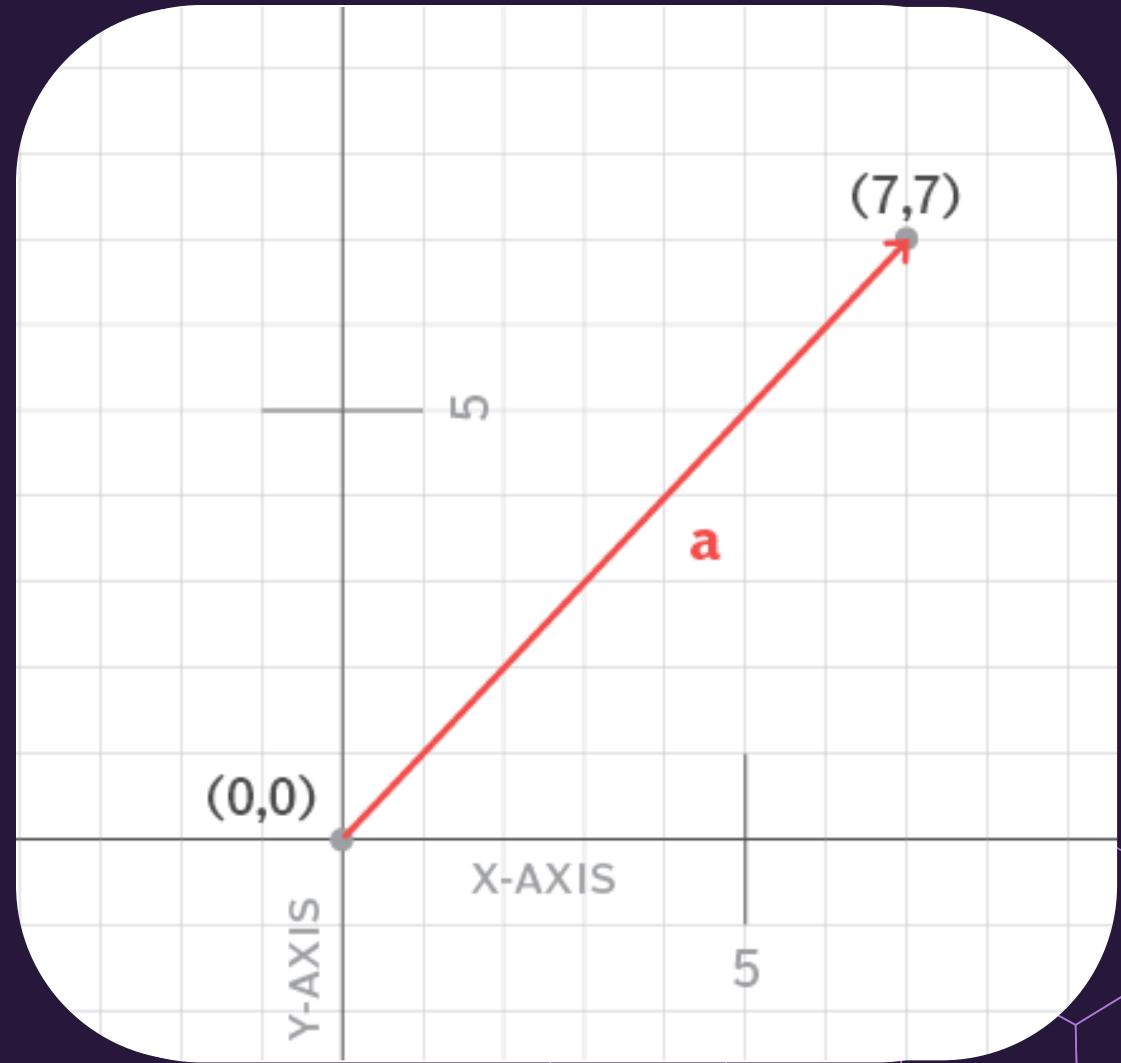
Semantic Search

How does RAG rags?

Back to high school

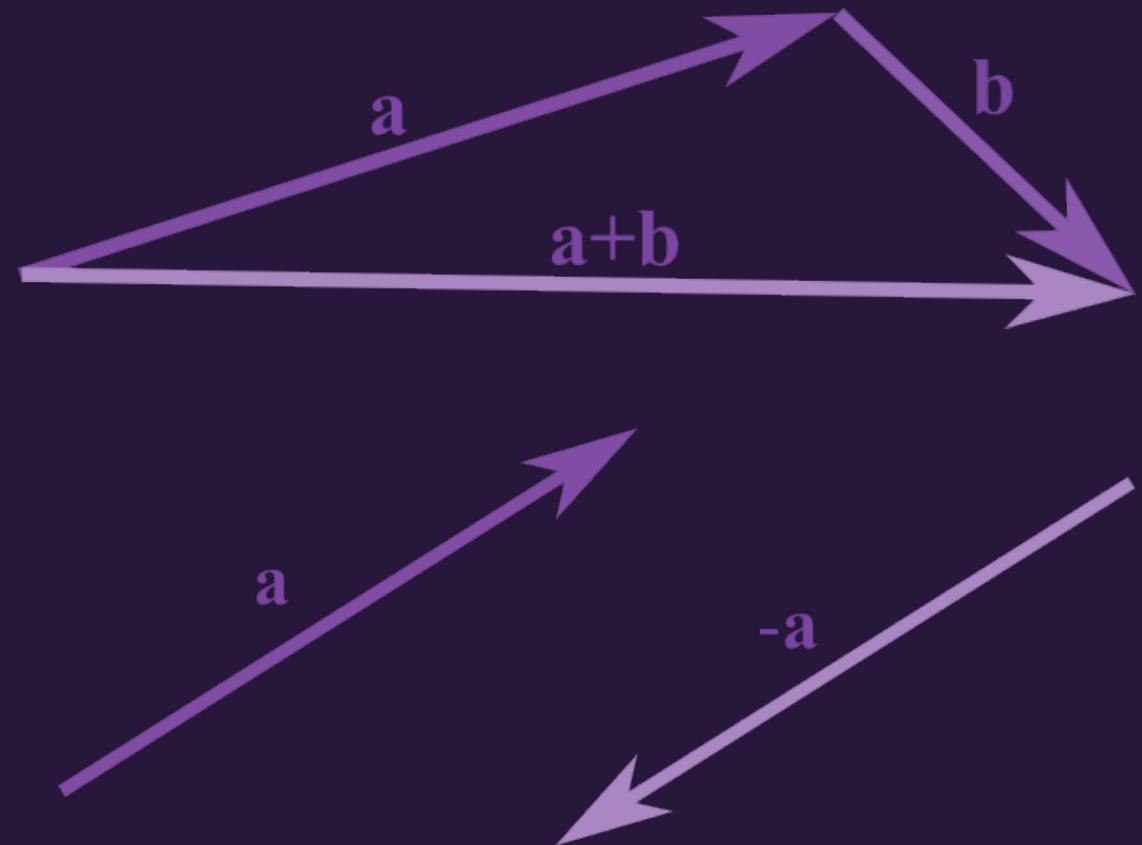
Vectors

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



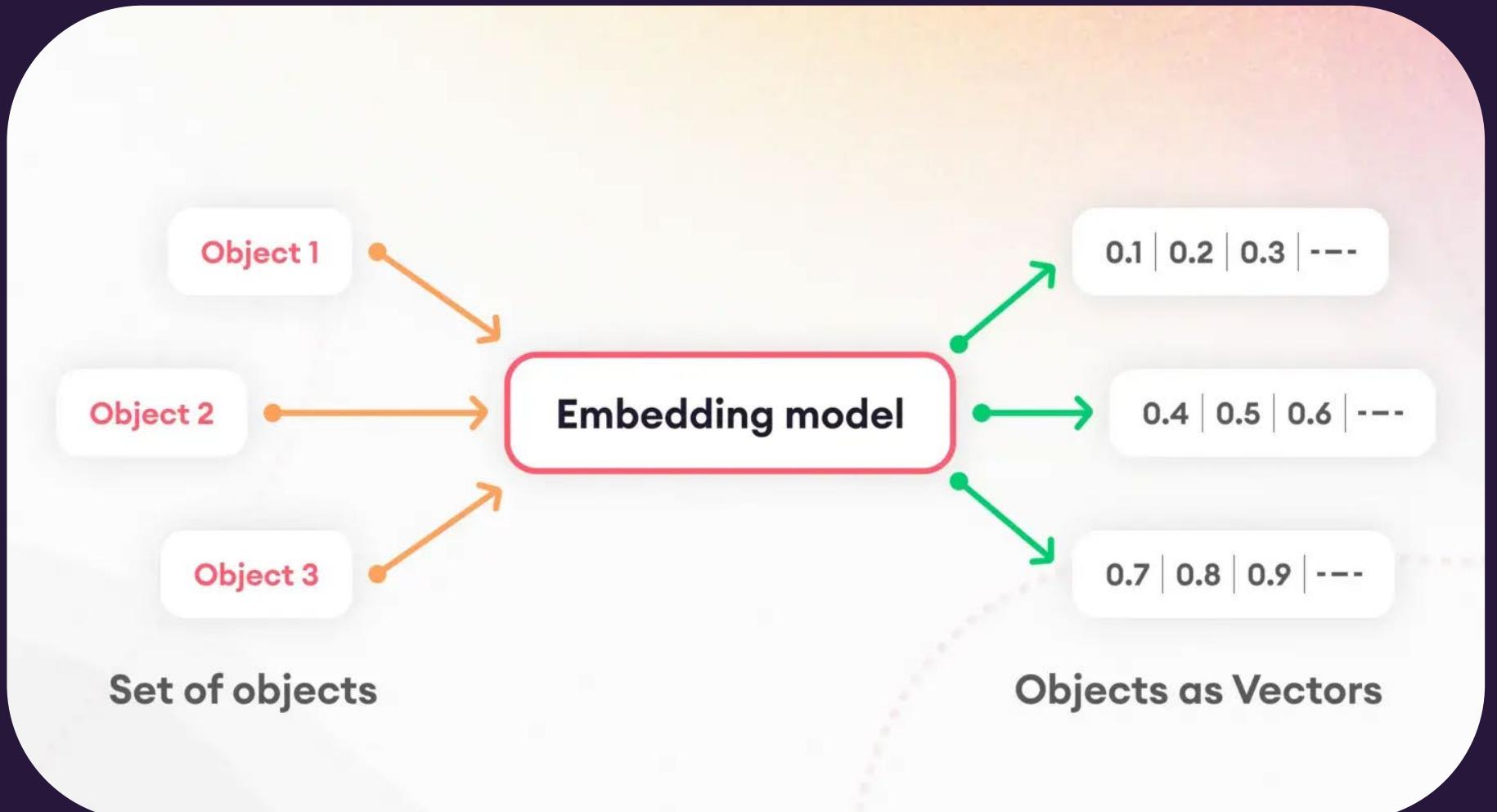


Back to high school



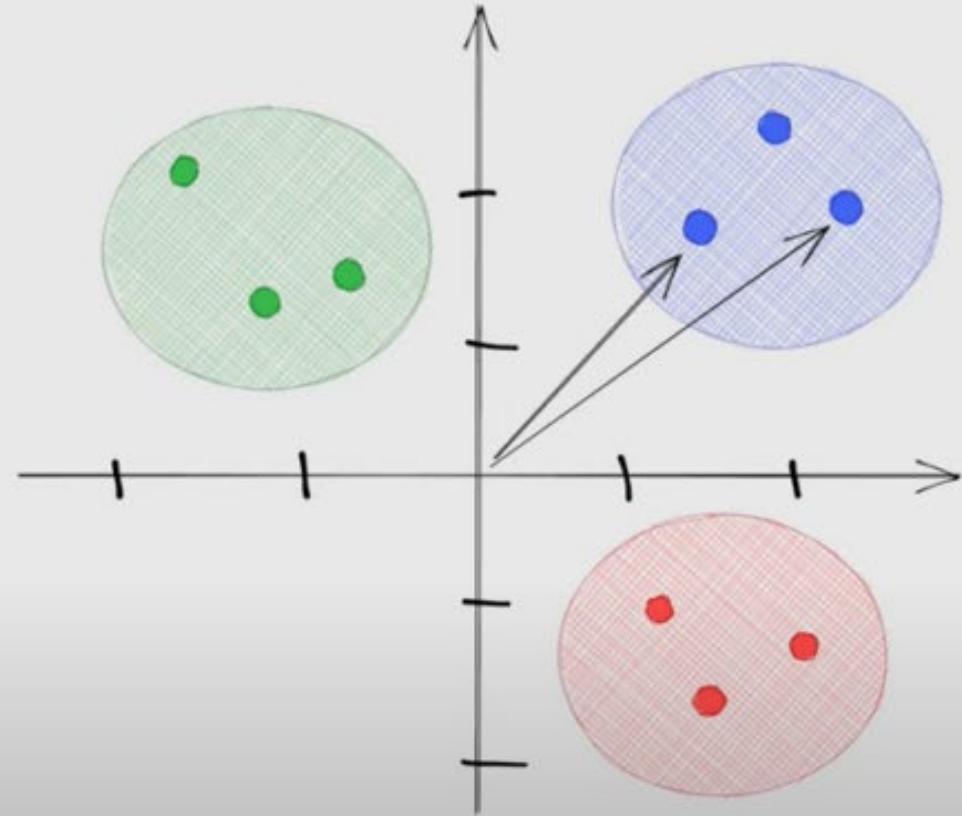
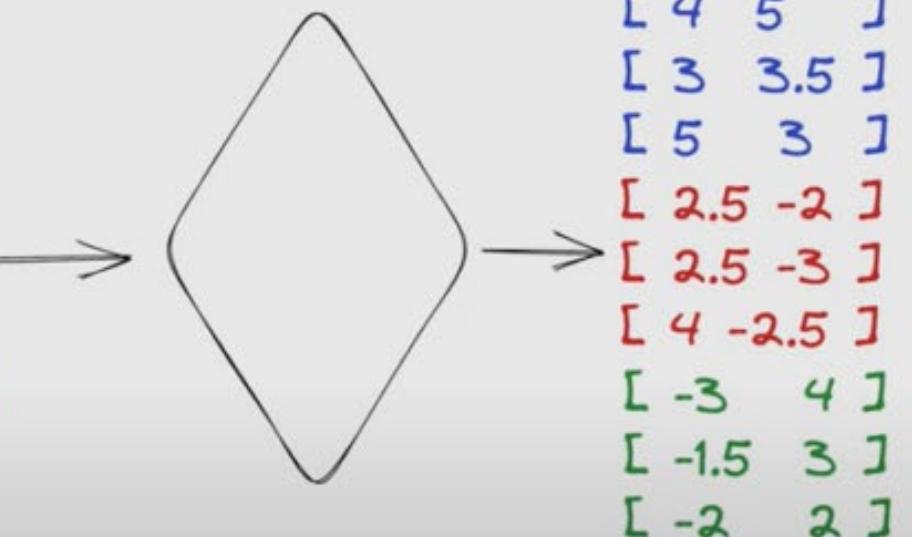
$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ \vdots \\ v_n \end{bmatrix}$$

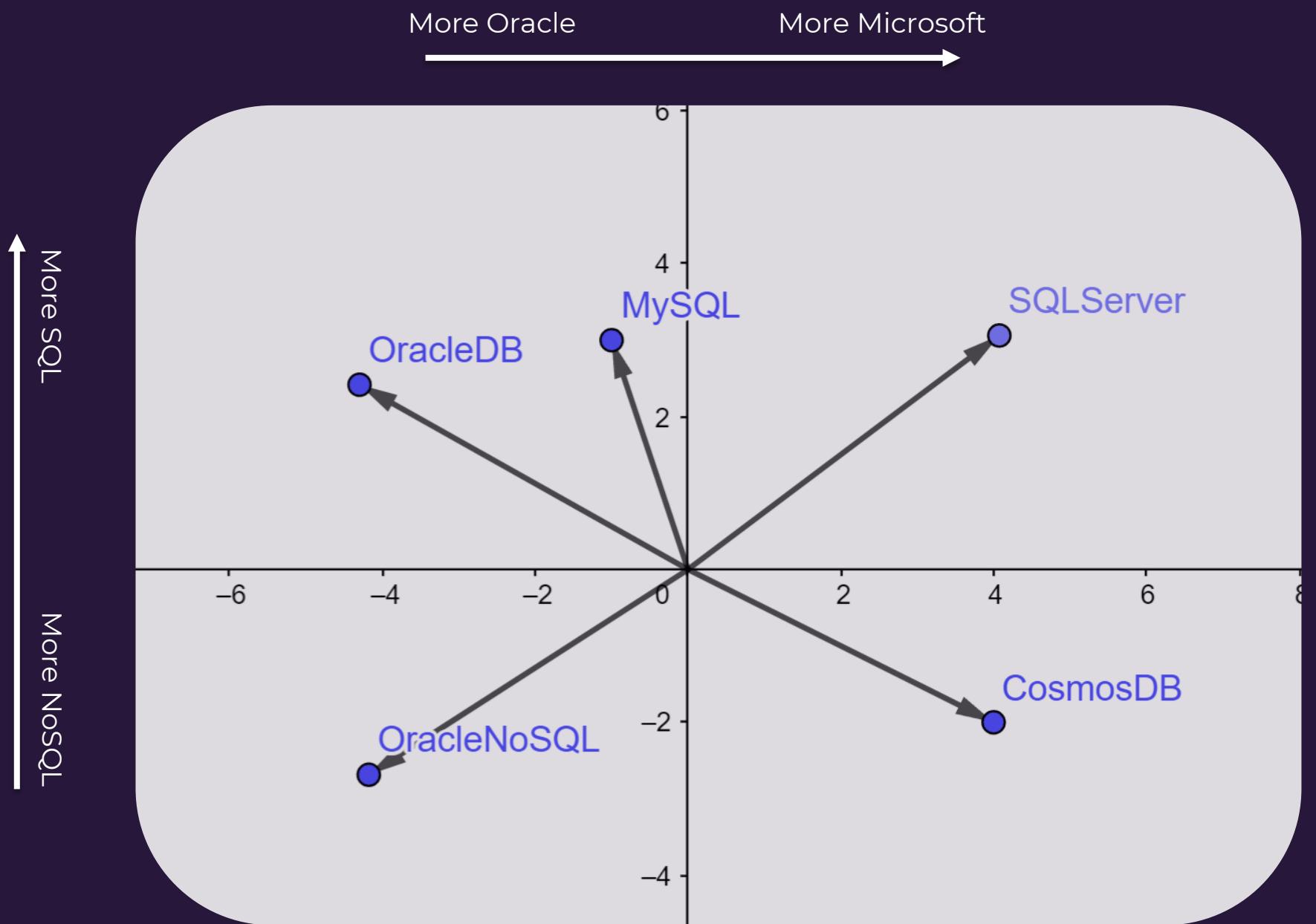
Embeddings



vector embeddings (2D example)

king
man
woman
apple
banana
orange
football
golf
tennis



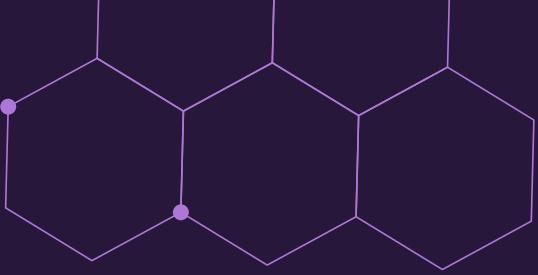




EMBEDDING MODELS

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



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

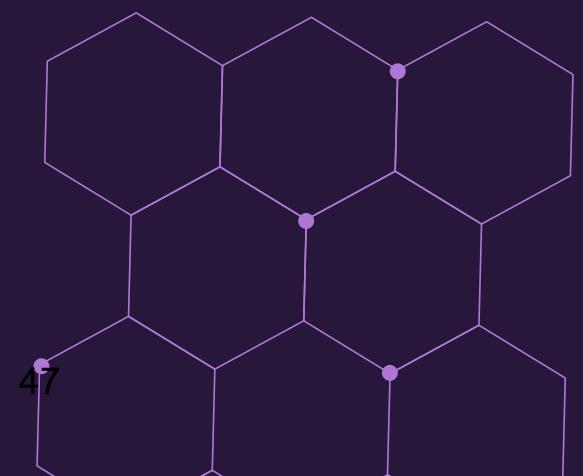


OUTPUT
DIMENSION

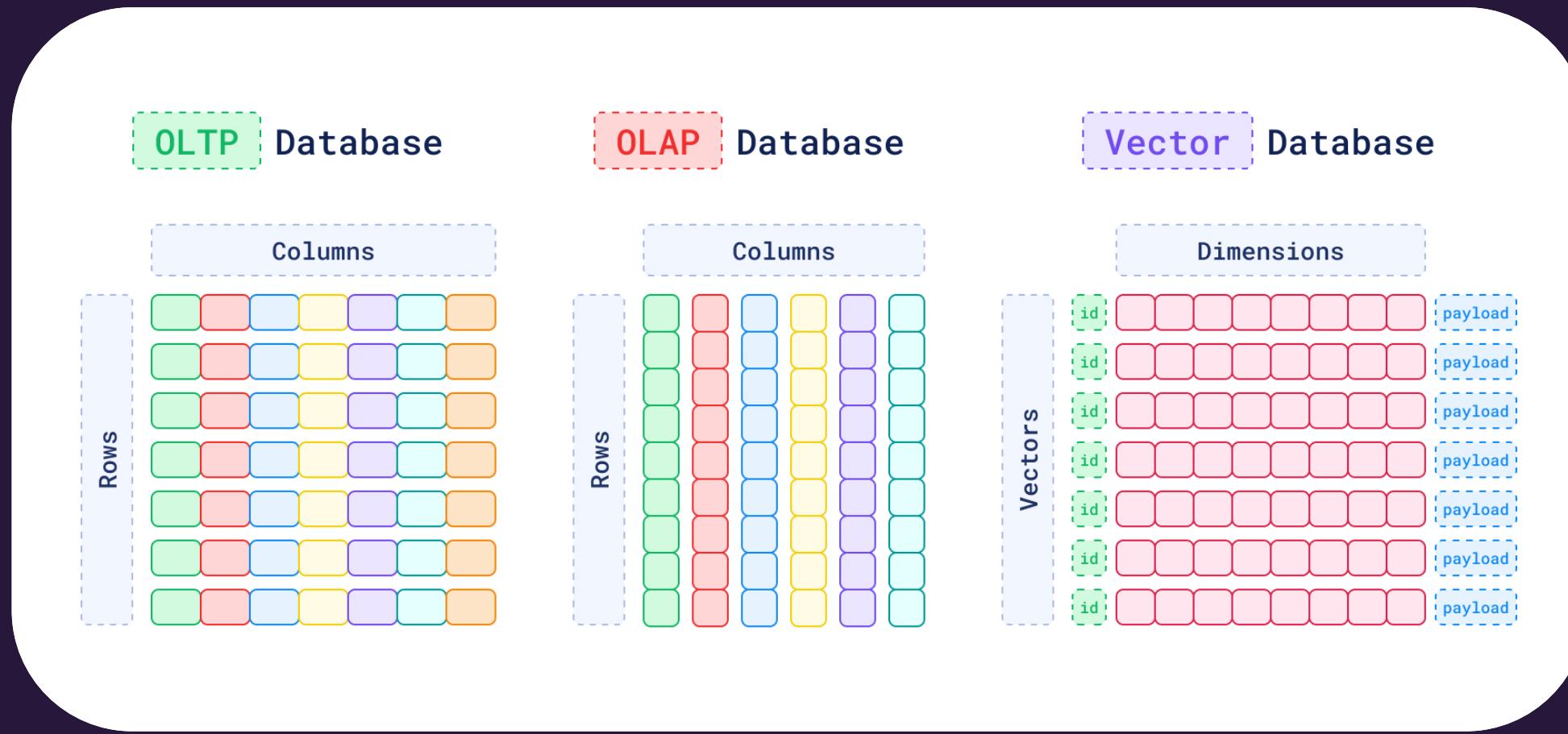
3,072

1,536

1,536



Where are you going to put all of these vectors?



Dedicated vector databases

Open source
(Apache 2.0 or MIT license)



chroma



vespa



LanceDB



Weaviate



Pinecone

Databases that support vector search



ClickHouse



PostgreSQL



elasticsearch



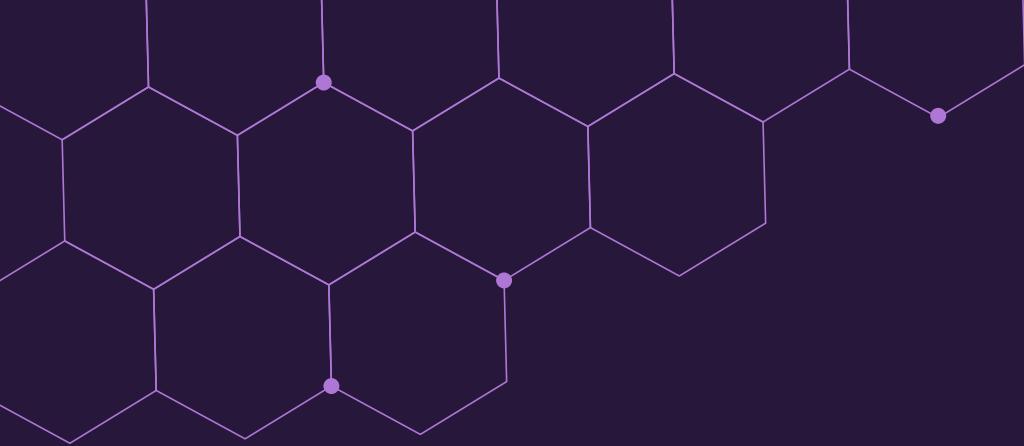
redis

[ROCKSET]

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 Cosmos DB



Azure AI Search



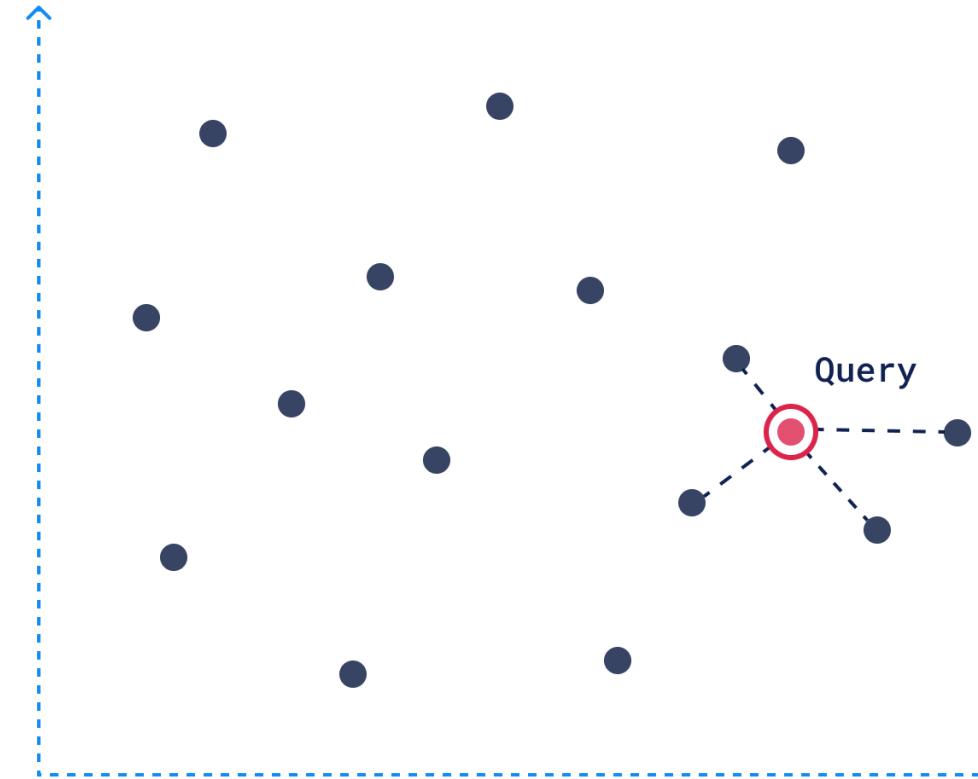
Azure SQL

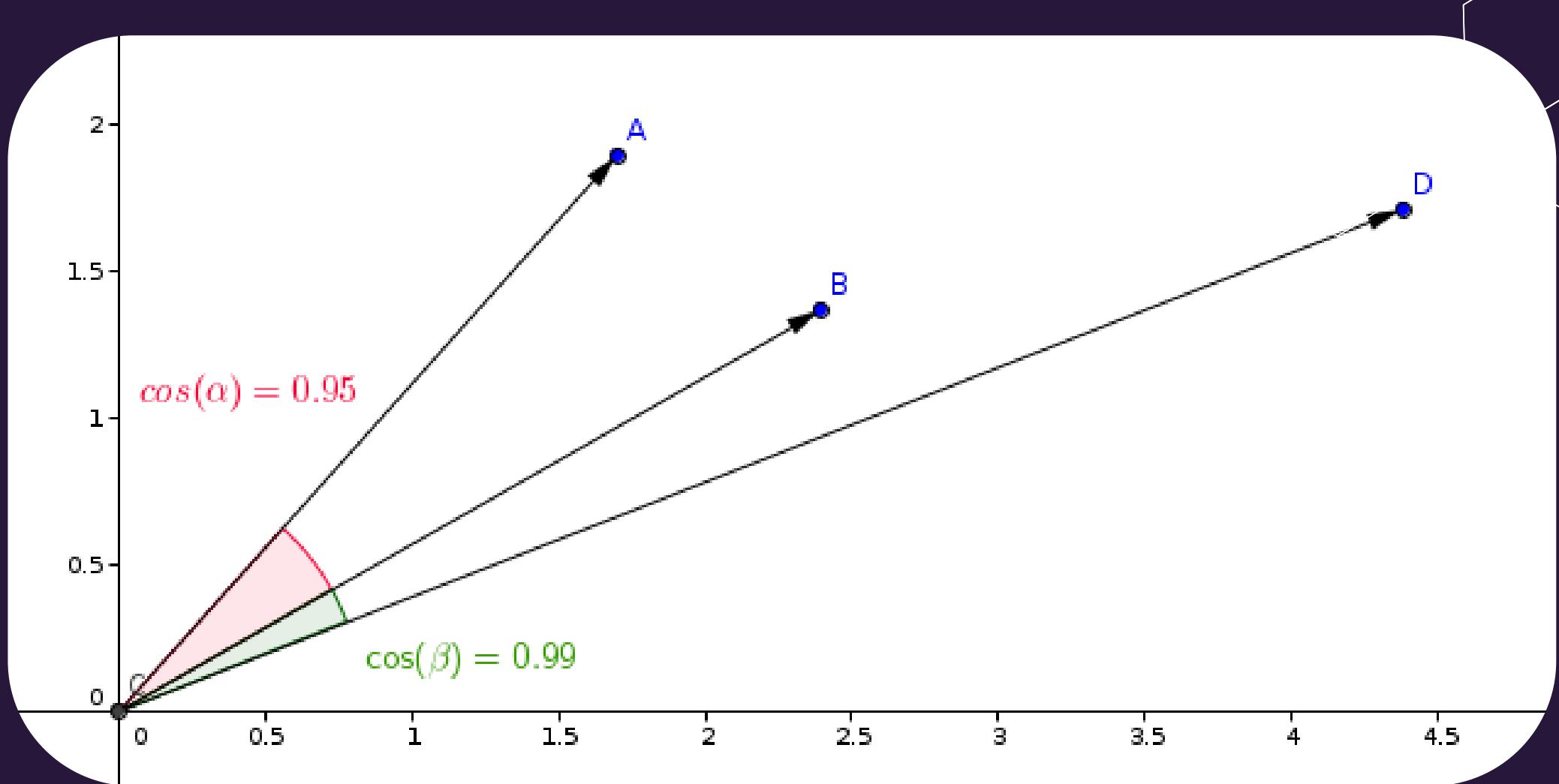


PostgreSQL Flexible Server

Vector Query

The four closest neighbors to a query within a vector space





**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 SALON

GASP! HE'S SO
EASY TO READ!



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



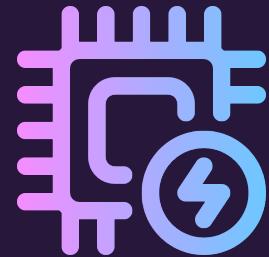
Source Text



→ Splitter



→ Chunked Text

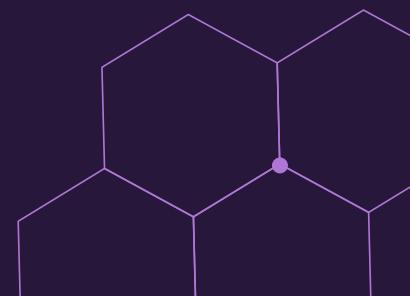
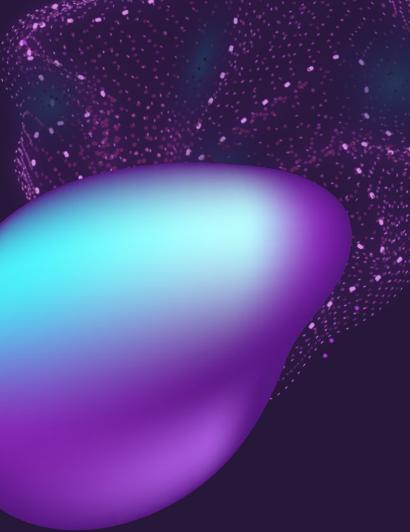


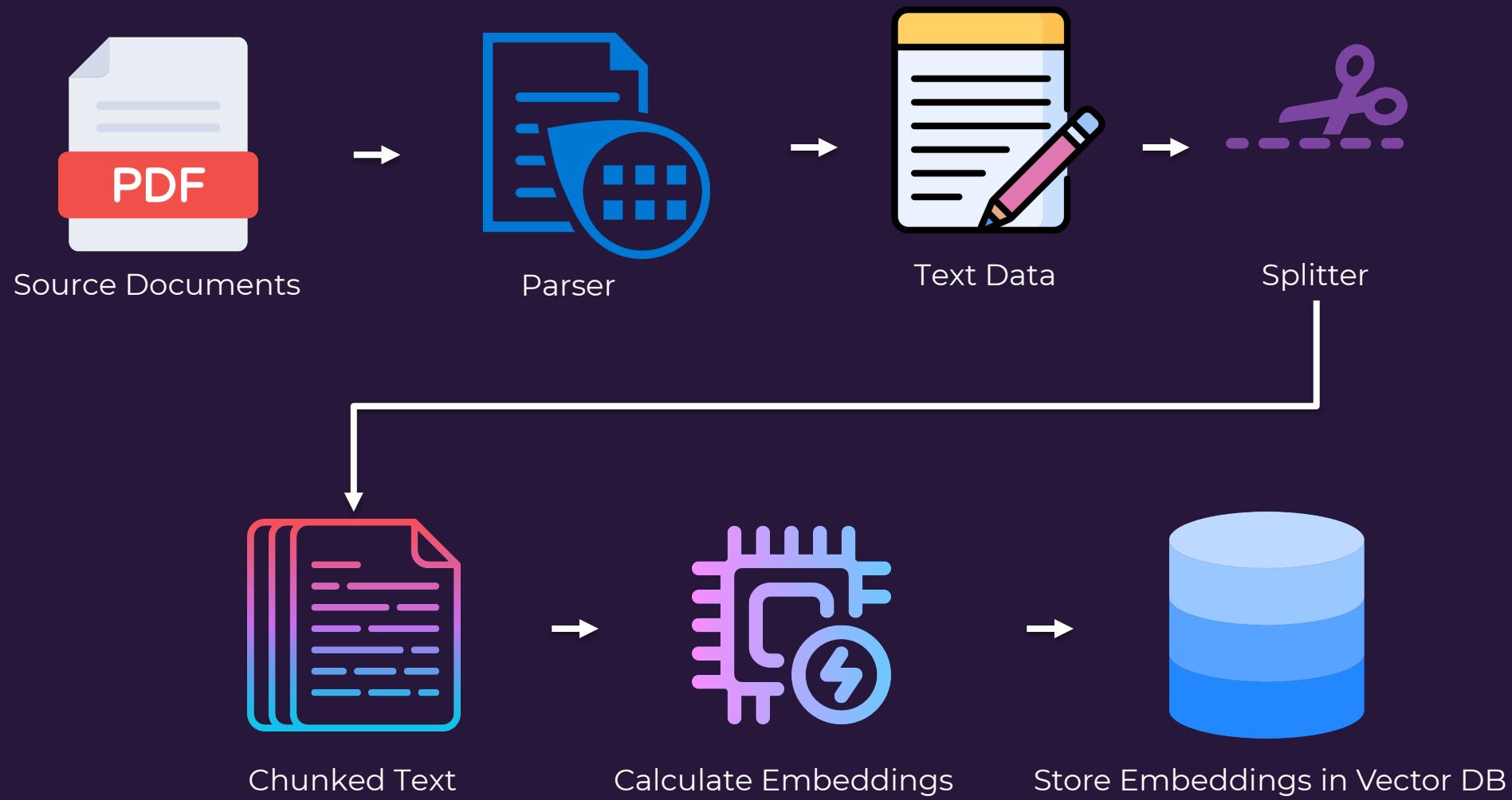
→ Calculate Embeddings → Store Embeddings in Vector DB



x

x





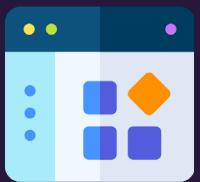
Let's put everything together

An end to end RAG architecture



User

→
Prompt



Application



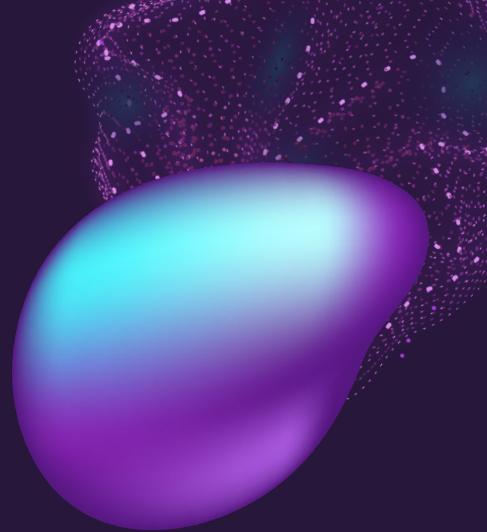
Embedding Model



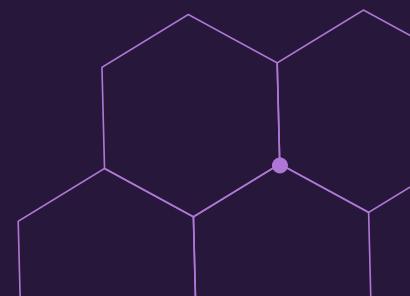
Vector DB



Foundation Model



X





User



Application

Create Embeddings
from the Prompt



Embedding Model



Vector DB



Foundation Model



User



Application



Embedding Model

Search the vector DB
with the
Embeddings



Vector DB



Foundation Model



User



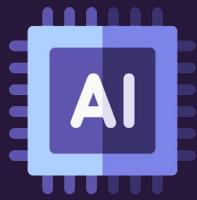
Application



Embedding Model



Vector DB



Foundation Model

Send the results (relevant chunks) to the app



User



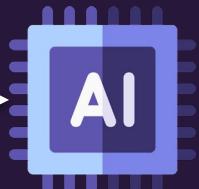
Application



Embedding Model



Vector DB



Foundation Model

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

x

x



User



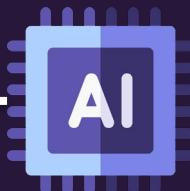
Application



Embedding Model



Vector DB



Foundation Model

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



User



Application

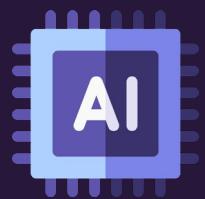
Receive
the response



Embedding Model



Vector DB



Foundation Model

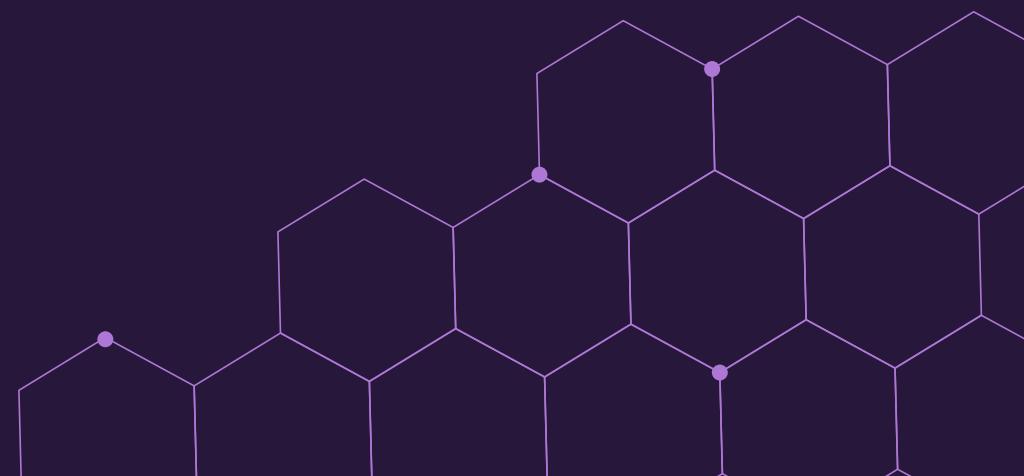


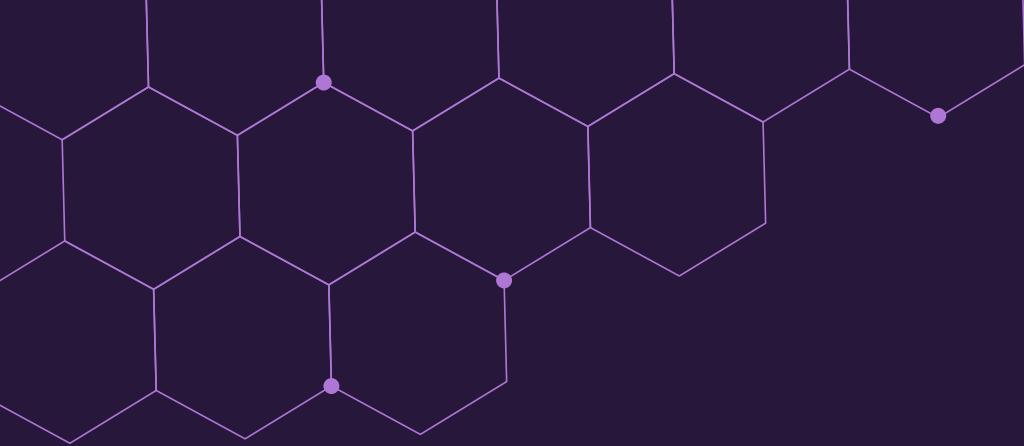
That's it!

Big word #6: GROUNDING

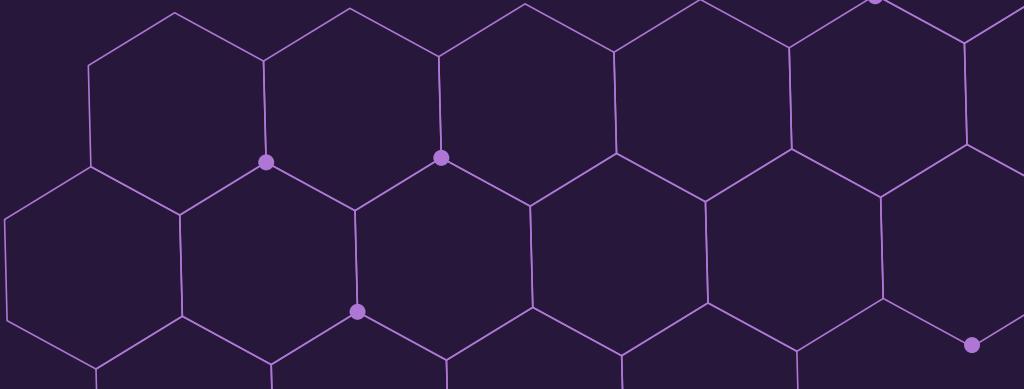
Literally what we just did with RAG 

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

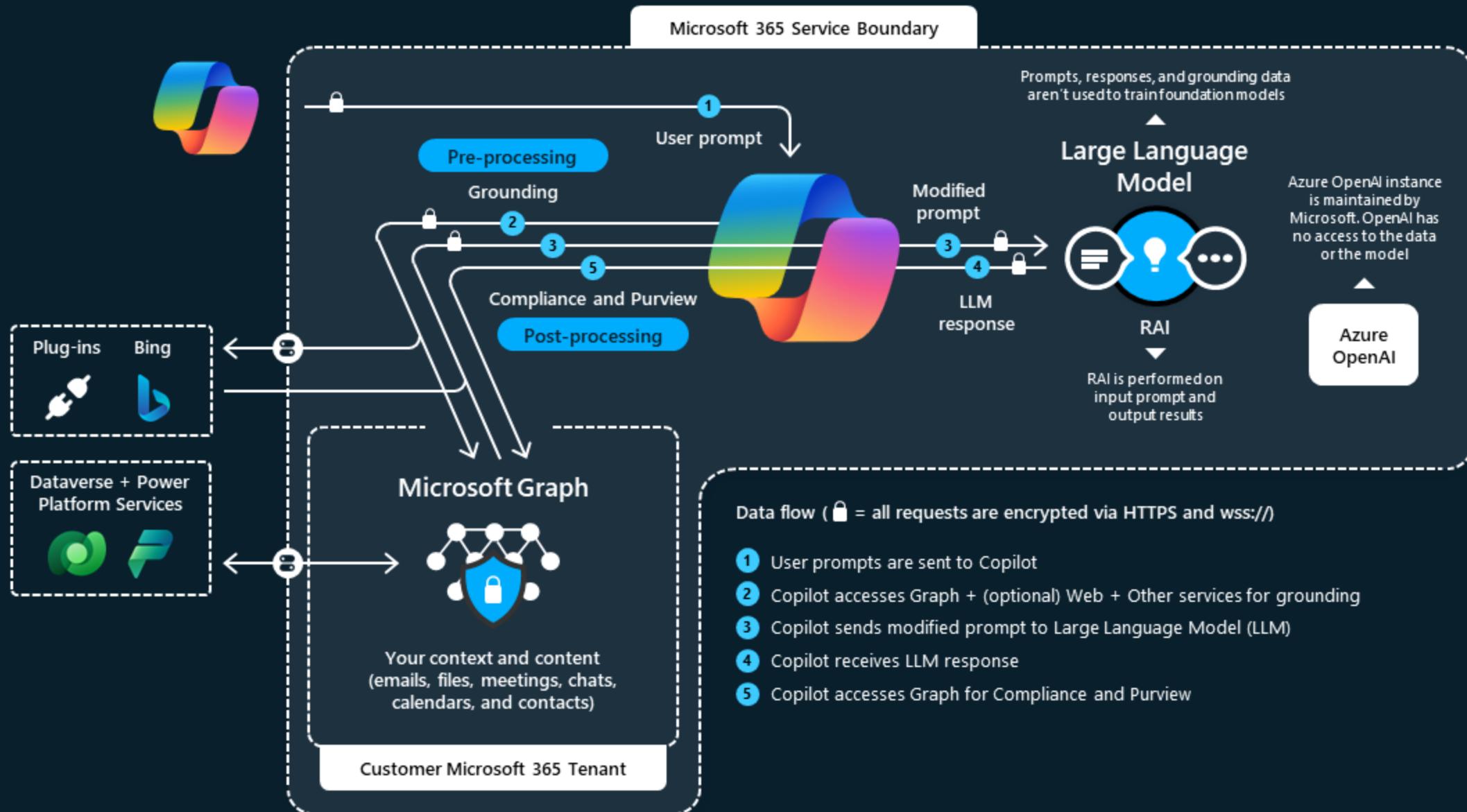


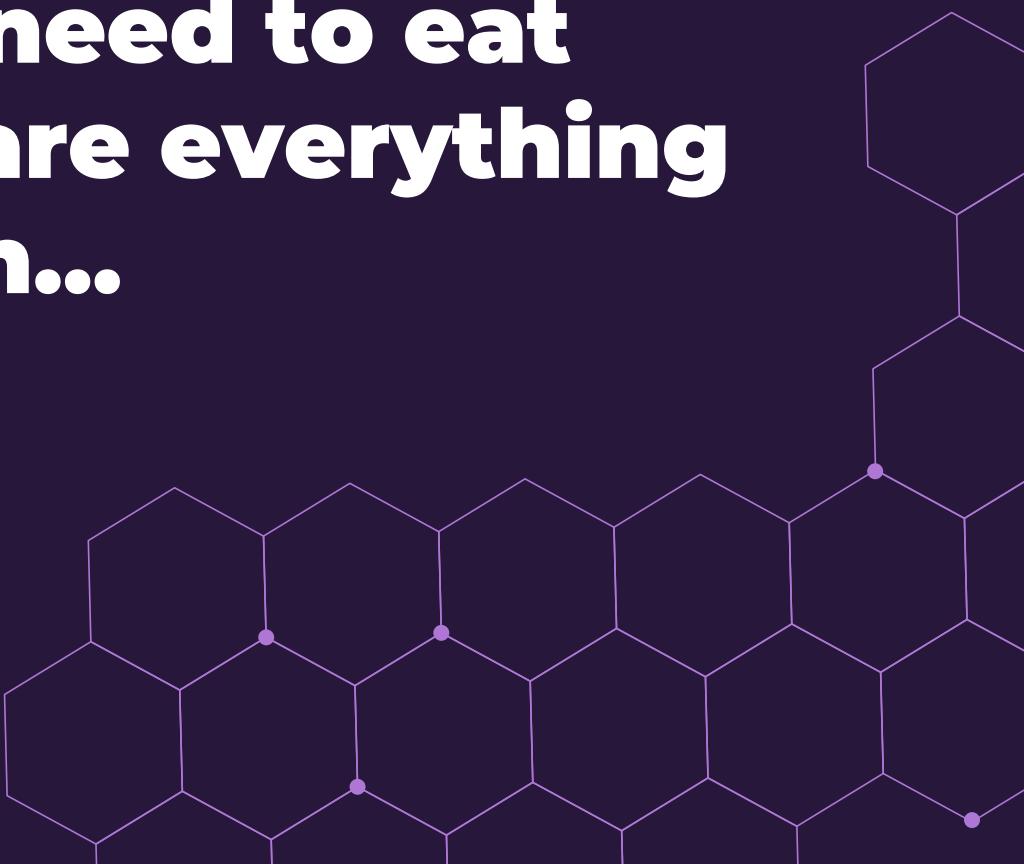
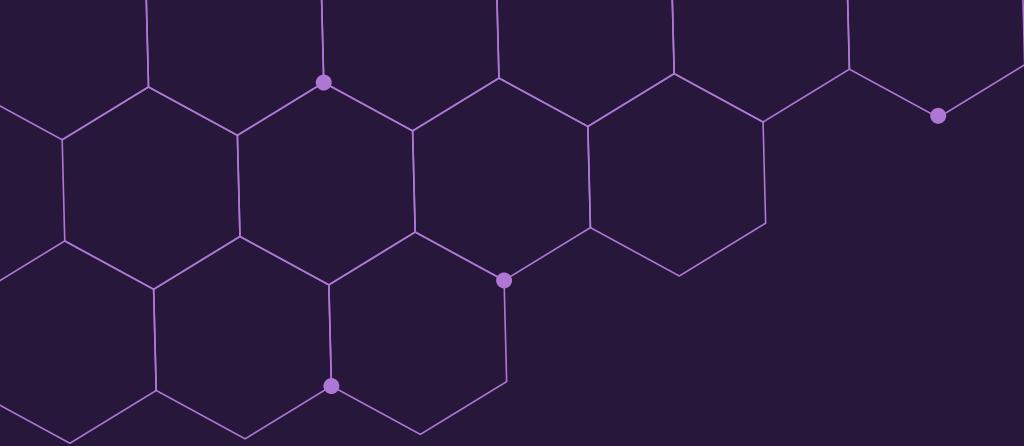


**Now you can easily
understand this:**



Microsoft Copilot for Microsoft 365 architecture

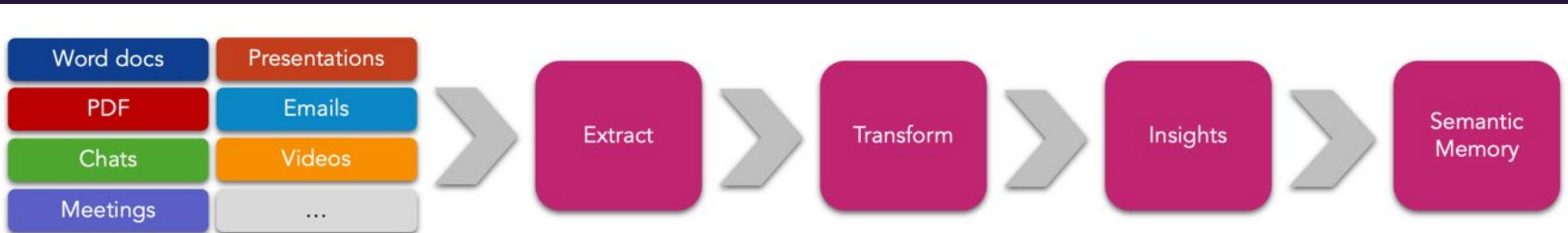




**Sometimes you just need to eat
without having to prepare everything
from scratch...**

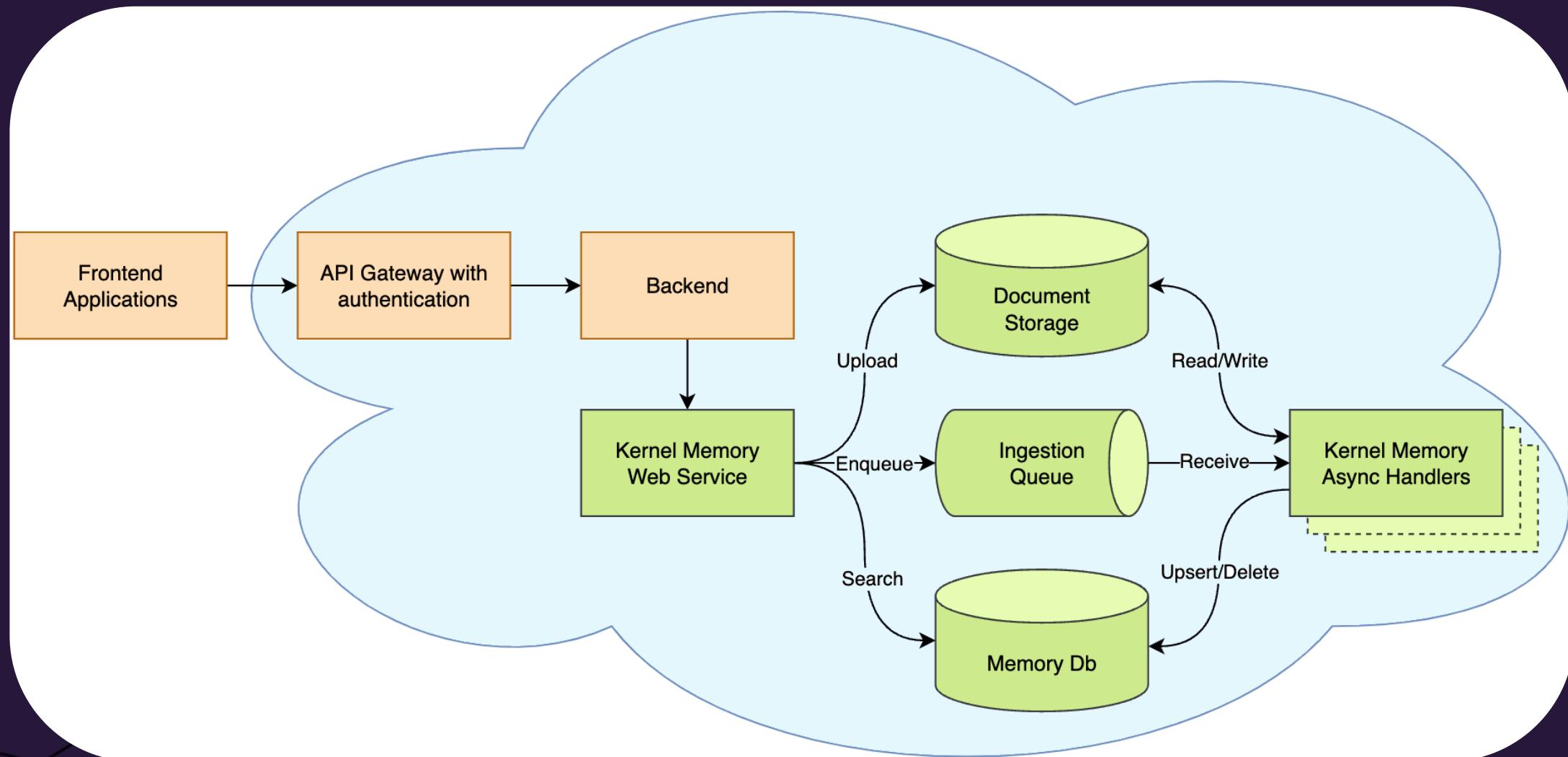
Kernel Memory

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

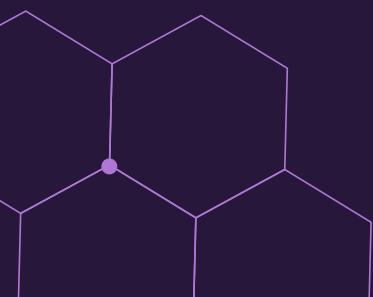


Llamaindex

LangChain

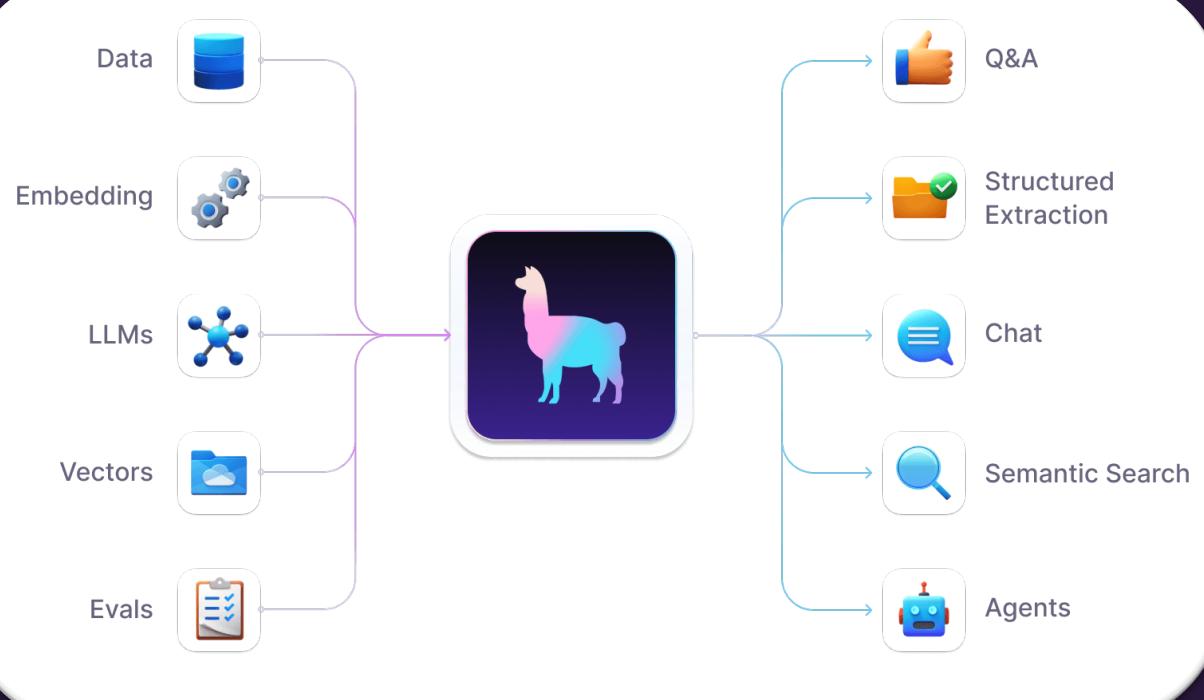


LangChain

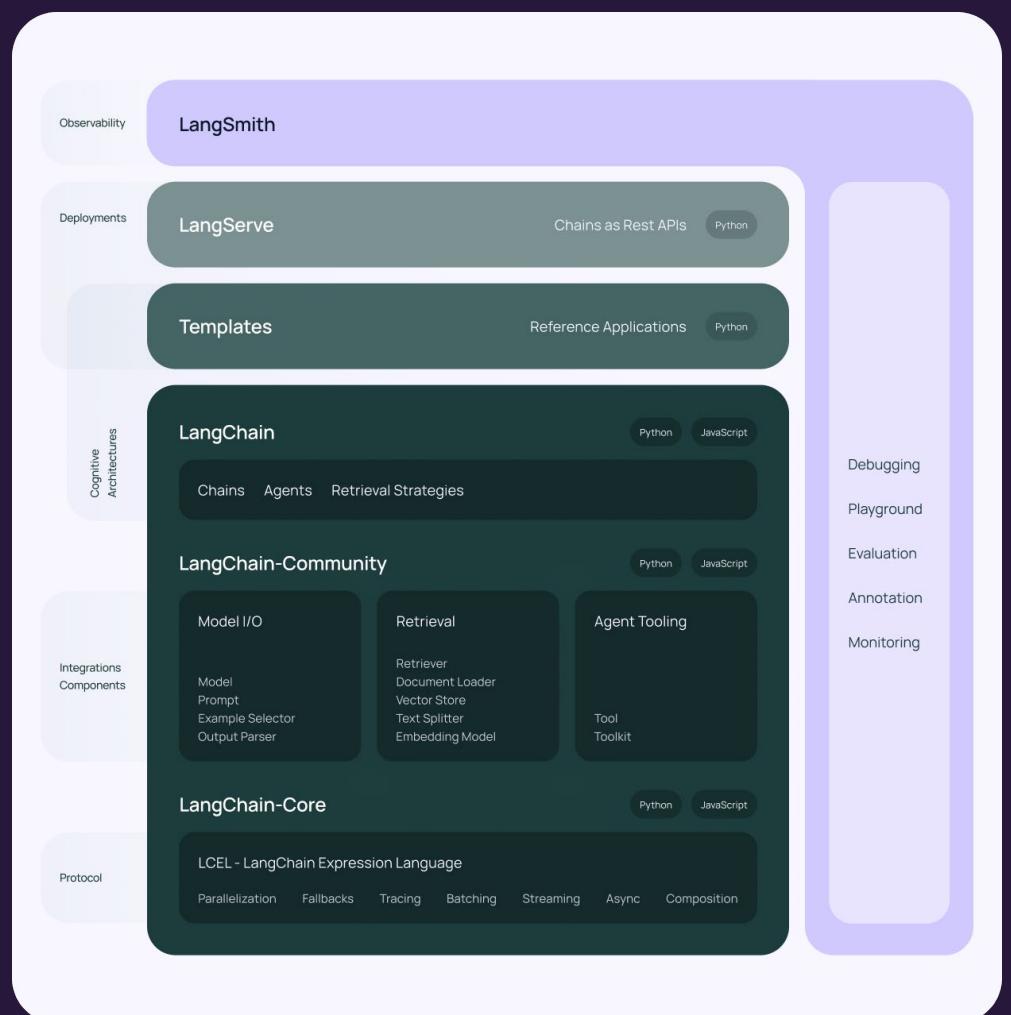




Llamaindex



LangChain



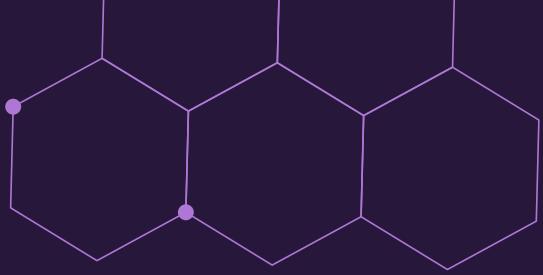
	Fine Tuning	RAG
Canonical use case	Increase performance of LLM on specific tasks, optimize costs, or enforce certain decoding schemes.	Generate LLM response based on some known context - reduce hallucinations, personalize answers with internal data, etc.
Mechanism	Change the behavior of LLM by directly updating the neural network weights and/or architecture.	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • Training dataset curation (\$ to \$\$\$ if humans involved) • 2) Actual training (hours to days, weeks) • 3) ML team's time (\$\$\$) 	<ul style="list-style-type: none"> • Deploy and maintain Vector/data Store • Ingest data and construct vector indexes • Data engineering time
Ongoing Resource Commitment	<ul style="list-style-type: none"> • Most LLM API will charge extra to serve finetuned models • If self-hosting, MLOps is needed for the finetuned LLM API Endpoints 	<ul style="list-style-type: none"> • 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

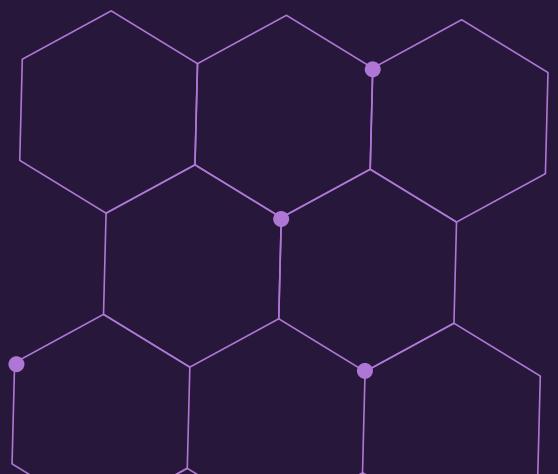
FUNCTION CALLING





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

```
# 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"
  }
}
```

he

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

```
# 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

```
# 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

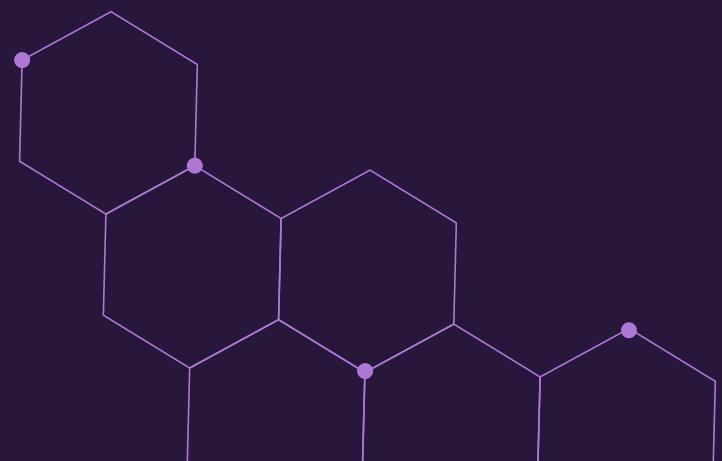
And call it

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



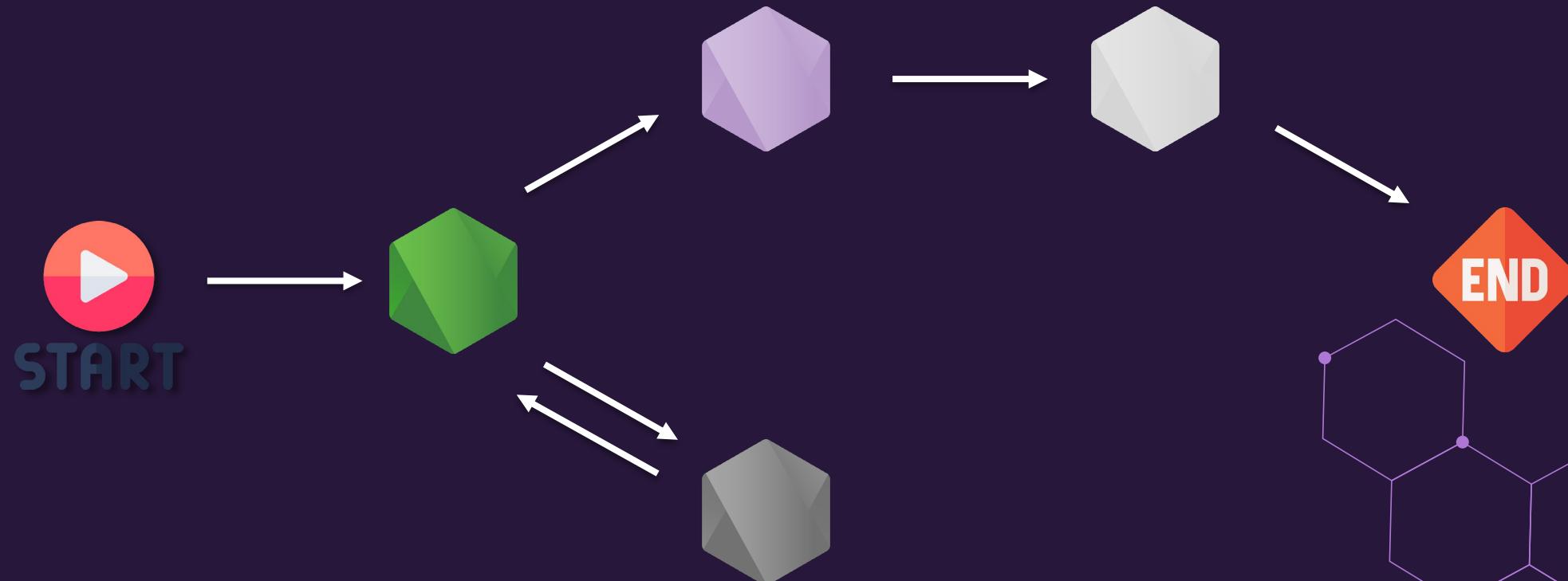


FUNCTION CALLING IS RELATIVELY NEW

- 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 ($\geq v0.3$)
 - Modified versions of other models

Big words #7: Agents

LLM used in a finite state machine



06

TYPICAL AI HURDLES





01- Prompt Engineering

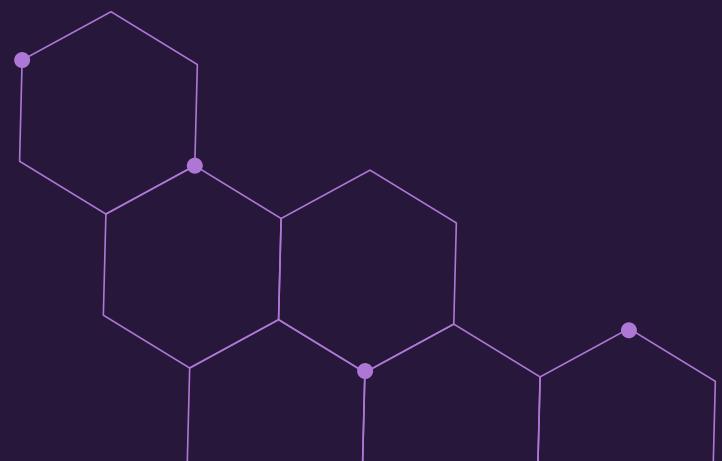
Crafting effective prompts to elicit desired responses from LLMs can be difficult and time-consuming.

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 W

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

Eat well: Stick to regular eating habits

cheese not sticking to pizza

All Images Videos Forums Shopping News We

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

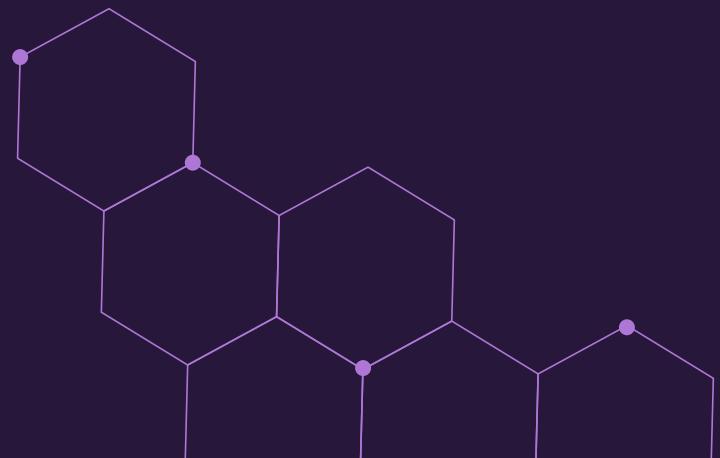
05- Cost Control

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

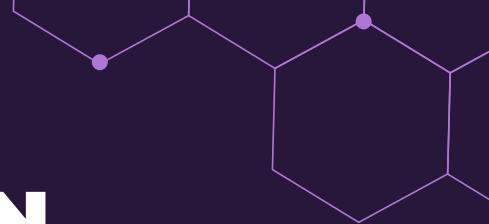
Not of the models, but of how the AI world is evolving, there is the risk that this whole presentation I finished a week ago is already outdated 😢

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





The background features a dark purple gradient with abstract white hexagonal patterns resembling molecular or crystal structures. Three large, semi-transparent spheres with a blue-to-purple gradient are positioned in the corners: one in the top-left, one in the bottom-left, and one in the bottom-right. Small white 'x' marks are scattered across the background.
**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.
- ~~Legacy~~



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.