Generative AI For The Enterprise

Lino Tadros, RD
The Training Boss LLC

Carey Payette
Trillium Innovations LLC

Lino Tadros

Carey Payette

Agenda

Time Slot	Description
10:00 - 10:15	Introductions
10:15 - 10:45	Introduction to Generative AI
10:45 - 11:00	Break
11:00 - 11:30	Working with OpenAI using API
11:30 - 12:00	Lab 1 Working with OpenAI using API
12:00 - 1:00	Lunch
1:00 - 1:30	Semantic Kernel Fundamentals
1:30 - 2:00	Lab 2 Semantic Kernel Fundamentals
2:00 - 2:15	Break
2:15 - 2:45	Langchain Fundamentals
2:45 - 3:15	Lab 3 Langchain Fundamentals
3:15 - 3:45	RAG Fundamentals
3:45 - 4:15	Lab 4 RAG Fundamentals
4:15 - 4:45	Embeddings & Vectorization
4:45 - 5:00	Lab 5 Embeddings and Vectorization and final wrap up.

Where is everything?

Introductions and Setup

GitHub

https://github.com/TheTrainingBoss/StirTrek-GenerativeAI

Generative AI Fundamentals

The story behind Generative AI, LLMs and OpenAI

Generative AI

Generative AI is Artificial Intelligence capability that can be used to create new content

What is Reinforcement Learning?

Reinforcement Learning is concerned with **solving sequential decision-making problems**.

- We have an objective when solving these problems.
- We take actions and get feedback from the world about how close we are to achieving the objective.
- Reaching the goal involves taking many actions in sequence, each action changing the world around us.
- We observe these changes in the world as well as the feedback we receive before deciding on the next action to take in response.

Example problems that can be framed this way:

- Playing video games
- Robotics, Autonomous Systems, Driving a Car, Balancing a Pole
- Algorithmic Trading

Large Language Models

Definition

- A Language Model is a probability distribution over sequences of words.
- Language Modeling is the task of predicting of what comes next after a sequence of words.
- What should be the next words = Word with the highest probability.
- It is super super smart autocomplete.
- LLM is similar concept but with significant amount of data that it has been trained on based on billions of parameters

Capabilities

- ChatGPT LLM uses 175 billion parameters (GPT-3.5) Vs 1.5 Trillion parameters for GPT-4
- ChatGPT does NOT understand what you write
- it uses statistical models in the LLM to predict what response is the best for what you ask.

Why all the fuss about LLMs?

Role

Language plays a key role facilitating communication between humans, and between humans and machines and between machines.

Power

Large language models (LLM's) have emerged as cutting-edge AI systems designed to process and generate text, with the aim of coherent communication. We want machines to handle complex language tasks like:

- Translation
- Summarization
- Information retrieval
- Conversational

How did we get here?

We have Large Language Models owing to advancements in:

deep learning techniques	neural network architectures (like Transformers)	increased compute and access to massive amounts of training data extracted from the Internet

What did we get with LLMs?

Large Language Models show tremendous generalization abilities and are capable of tasks like:

- Code Generation
- Text Generation
- Tool Selection & Manipulation
- Reasoning
- Understanding

Importantly, these capabilities are being unlocked in situation with no specific training or situations where only a few examples are given to the problem, in plain English.

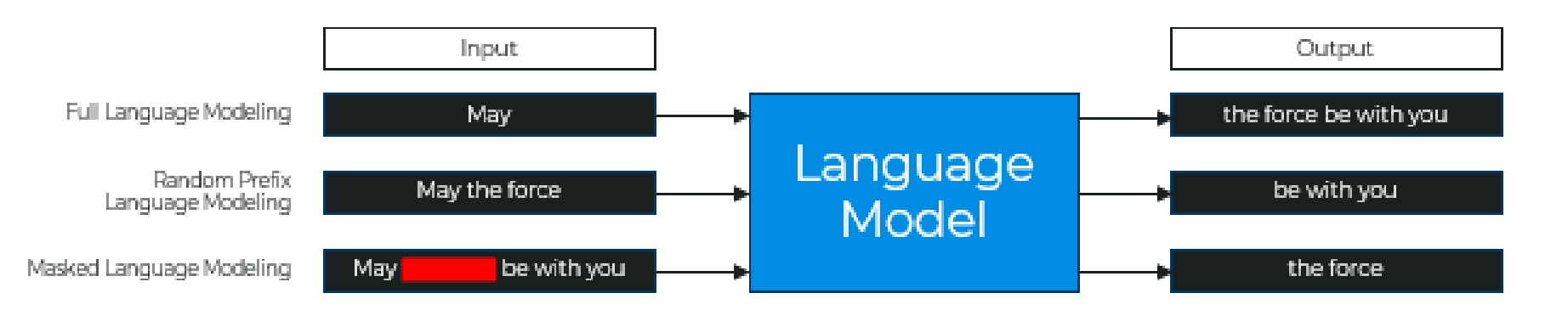
This level of generalization was not previously attainable with smaller language models.

What is a Language Model?

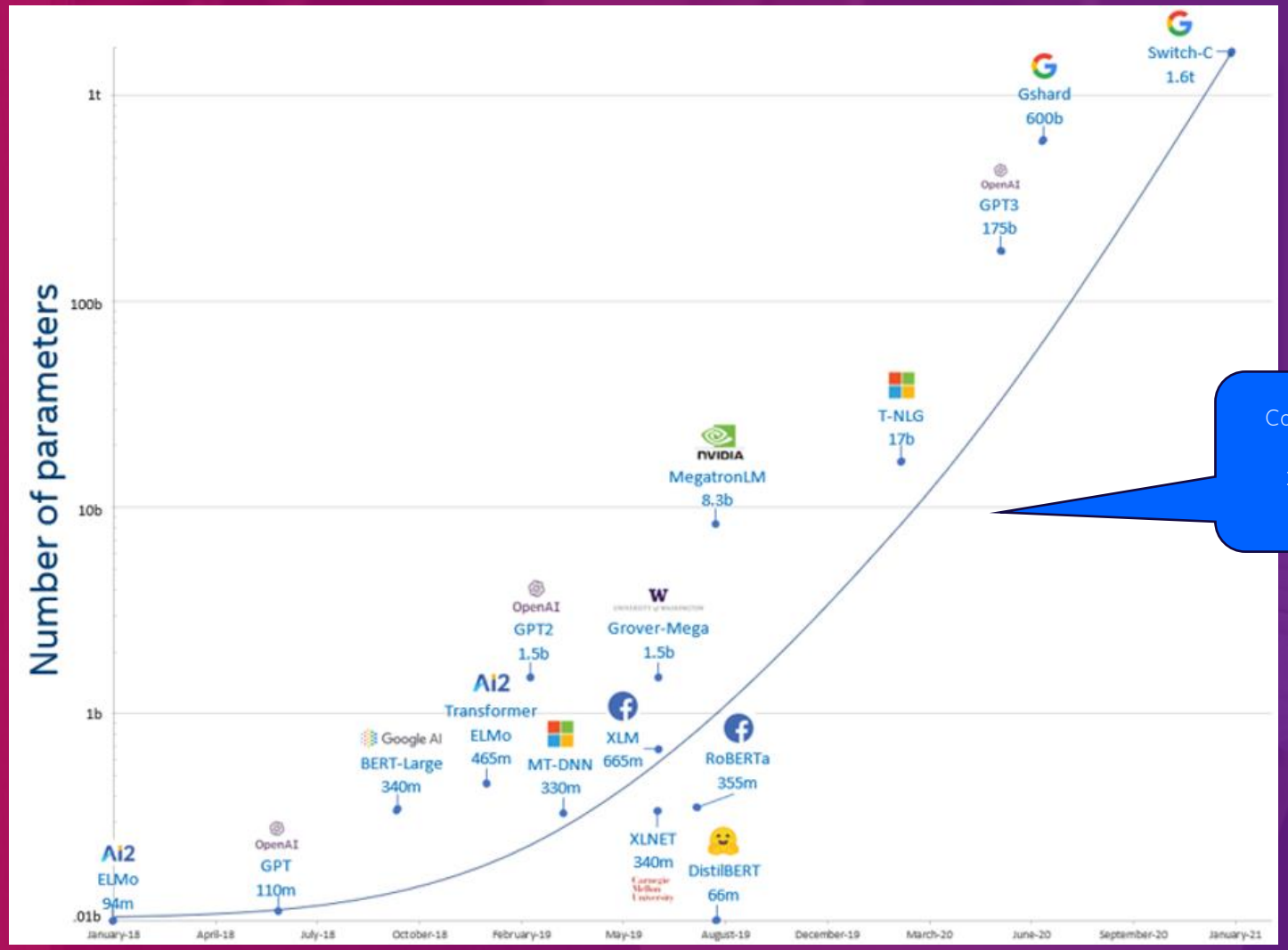
A core concept underpinning Large Language Model is the **Language Model**

- A language model is a deep neural network model that takes text as input and predicts text as output.
- Language models in the context of large language models are trained using self supervised learning, that is:
- Give some text dataset, choose a portion of the text to use as the input and train the model to predict the rest of the text as output. Predicting this text is the objective.
- Importantly, there is no labeled data as is typically needed in supervised learning- the labels are selected from the dataset itself.

Examples of how the text is used for the unsupervised learning training objective



What makes an LLM “LARGE”?



Consensus is the model has **10B or more** parameters

There’s more than one LLM ;-)

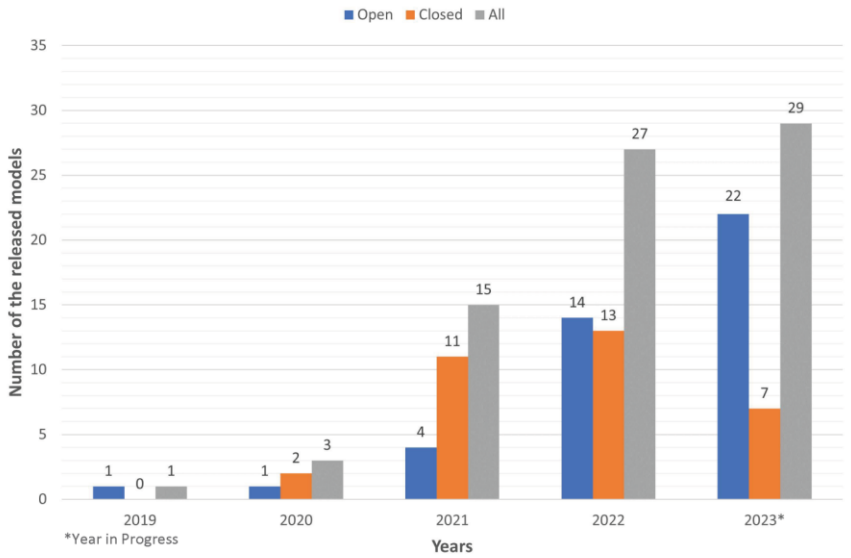


Fig. 1: The trends in the number of LLM models introduced over the years.

https://arxiv.org/abs/2307.06435

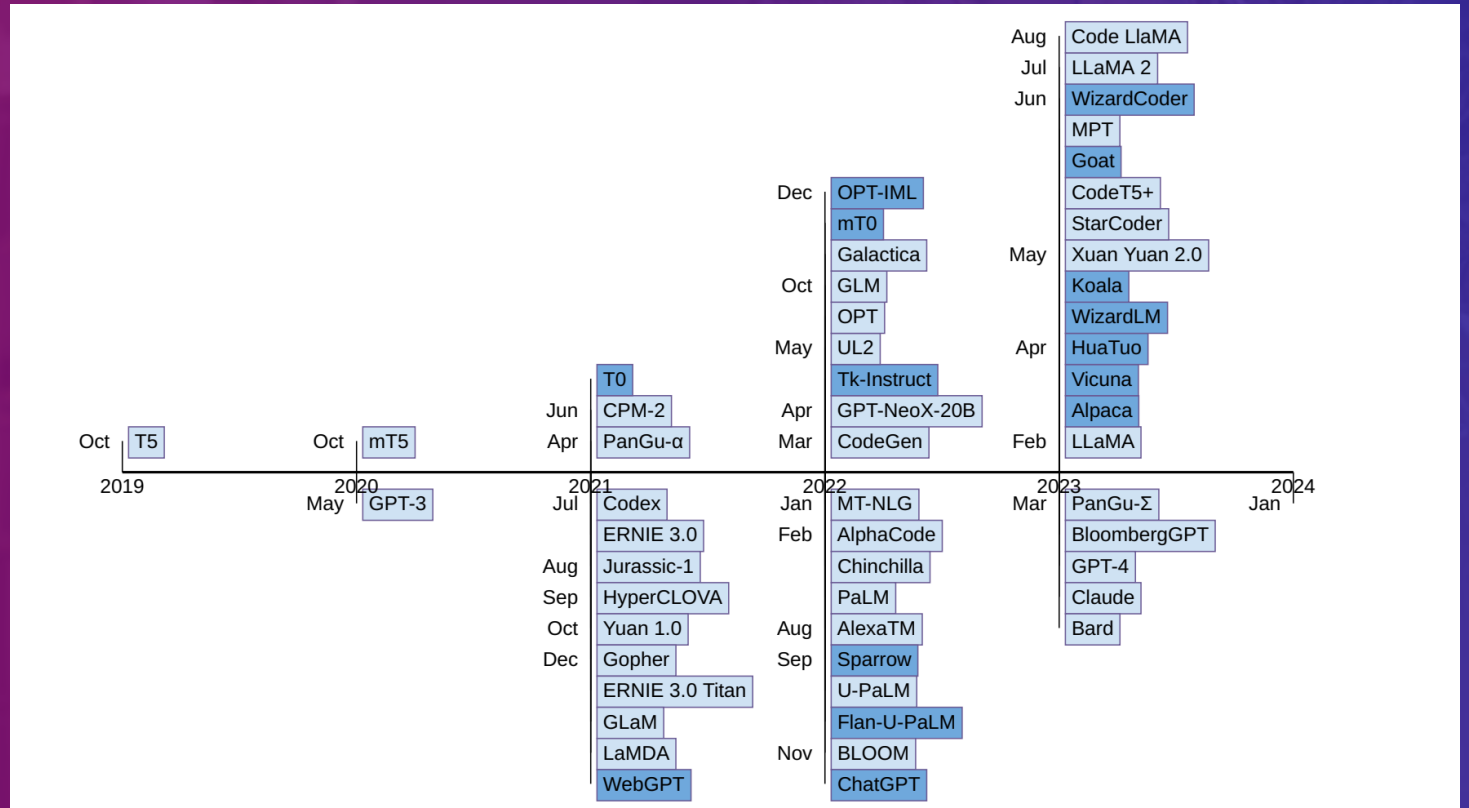
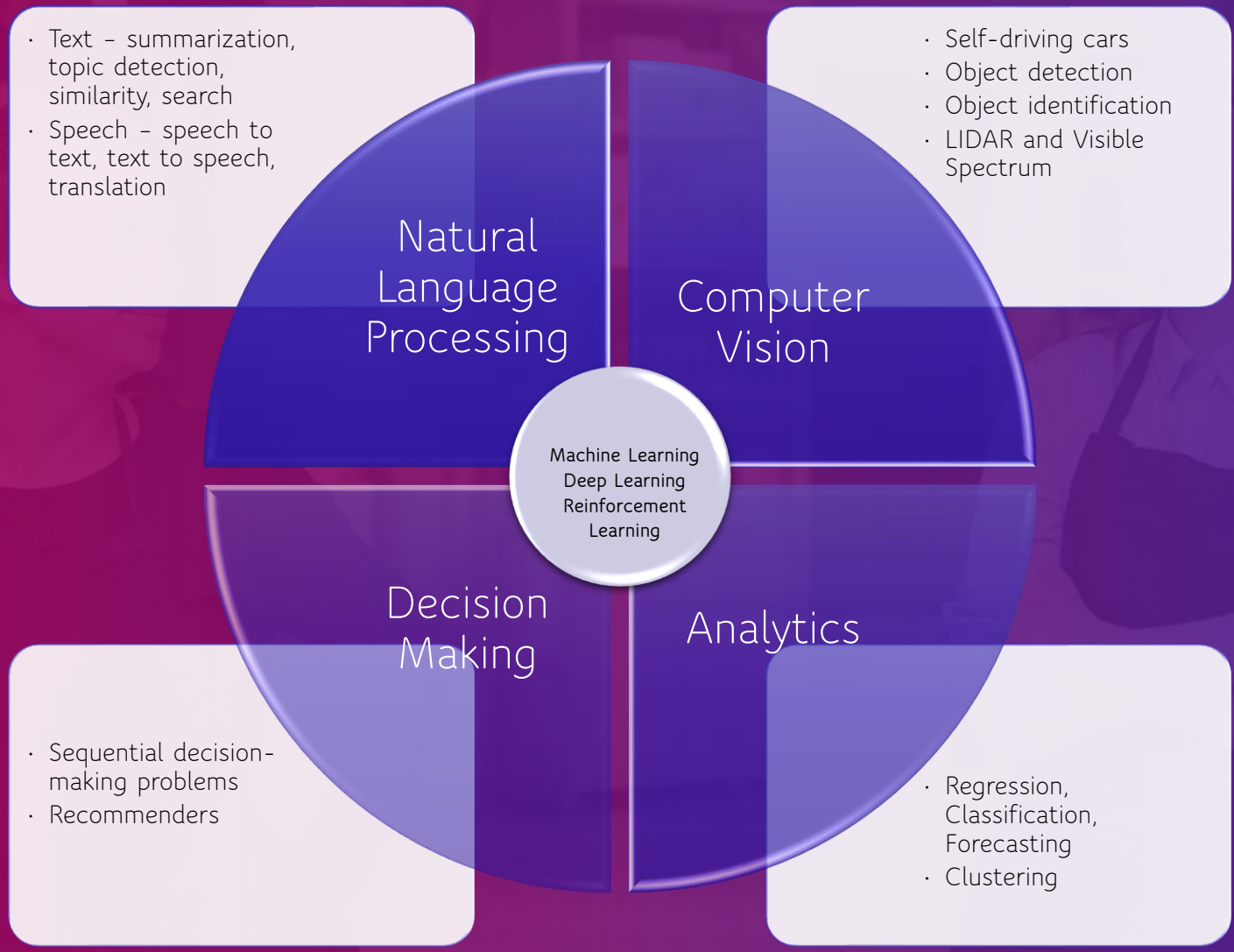


Fig. 2: Chronological display of LLM releases: light blue rectangles represent ‘pre-trained’ models, while dark rectangles correspond to ‘instruction-tuned’ models. Models on the upper half signify open-source availability, whereas those on the bottom half are closed-source. The chart illustrates the increasing trend towards instruction-tuned models and open-source models, highlighting the evolving landscape and trends in natural language processing research.

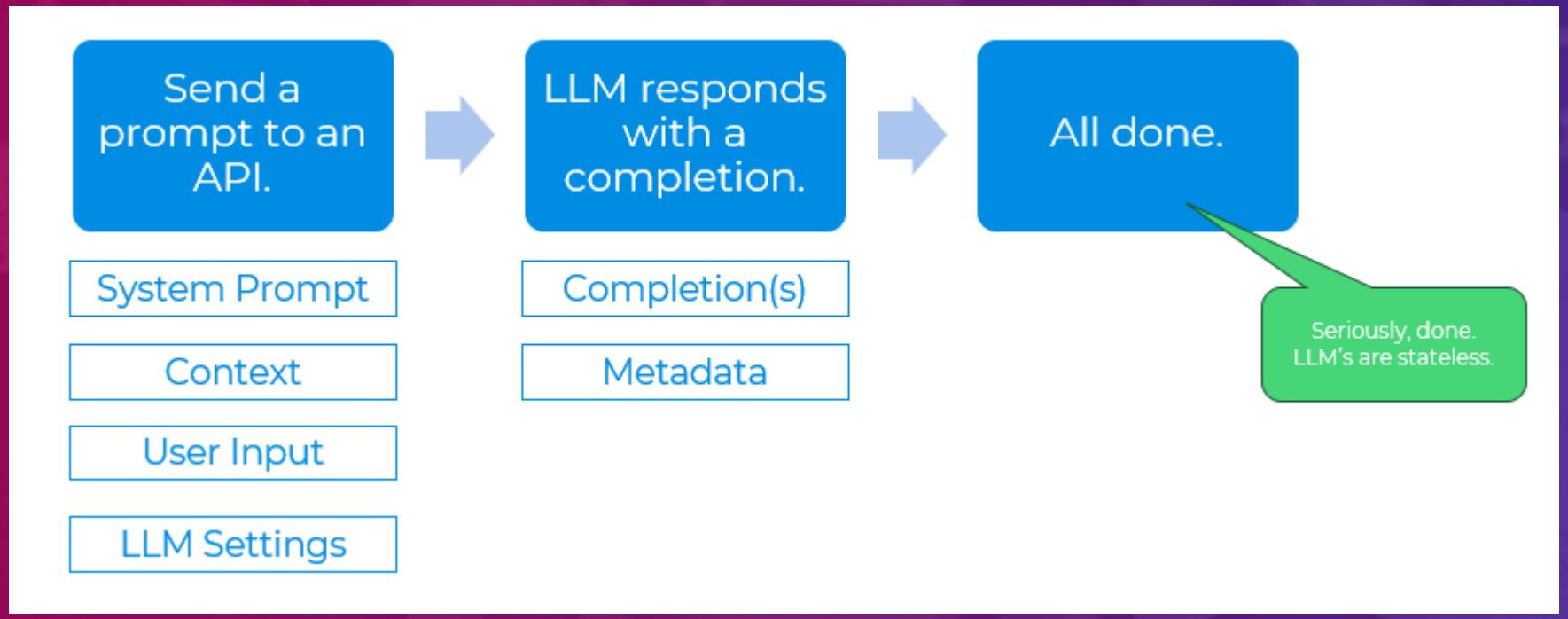
What are some of the applications of LLMs?

- **NLP**: Summarization, Classification, Paraphrasing, Reasoning, Solving and Recommending, Translation
- **Code**: validation and generation, prompt functions
- **Visuals**: Image and video, visual reasoning
- **Audio**: Text to speech, Speech to text
- **Robotics**: human-robot interaction, task planning, navigation, manipulation
- **Data**: Generating data, synthetic datasets

What are some of the applications of LLMs?



Basics of Interacting with an LLM



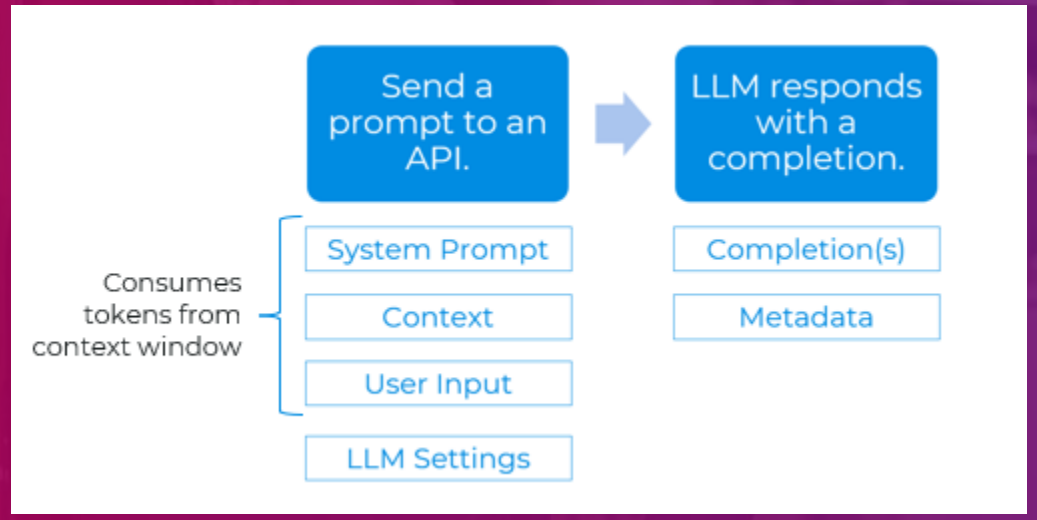
50 First Dates

- Large language models are like Lucy.
- Irrespective of the number of billions of parameters that the large language models have, these models have two forms of "memory": parametric memory and nonparametric memory.
- The parametric memory models have effectively the memories encoded in the model weights during training. Their "knowledge" is only as current as the most recent data the model was trained against.
- Comically, the solution to how we catch a model up on recent events is almost exactly like what Henry Roth did for Lucy: before the day starts show her a summary of what she needs to know for the day. This summary is passed in the context window.

MODEL	DESCRIPTION	CONTEXT WINDOW	TRAINING DATA
`gpt-4-1106-preview`	**GPT-4 Turbo** New The latest GPT-4 model with improved instruction following, JSON mode, reproducible outputs, parallel function calling, and more. Returns a maximum of 4,096 output tokens. This preview model is not yet suited for production traffic. Learn more.	128,000 tokens	Up to Apr 2023
`gpt-4-vision-preview`	**GPT-4 Turbo with vision** New Ability to understand images, in addition to all other GPT-4 Turbo capabilities. Returns a maximum of 4,096 output tokens. This is a preview model version and not suited yet for production traffic. Learn more.	128,000 tokens	Up to Apr 2023
`gpt-4`	Currently points to `gpt-4-0613`. See continuous model upgrades.	8,192 tokens	Up to Sep 2021
`gpt-4-32k`	Currently points to `gpt-4-32k-0613`. See continuous model upgrades.	32,768 tokens	Up to Sep 2021

What is the Context Window?

- The context window is the size limit to the text you can send to a completions endpoint as input
- It is measured in number of tokens (of the tokenized input text)

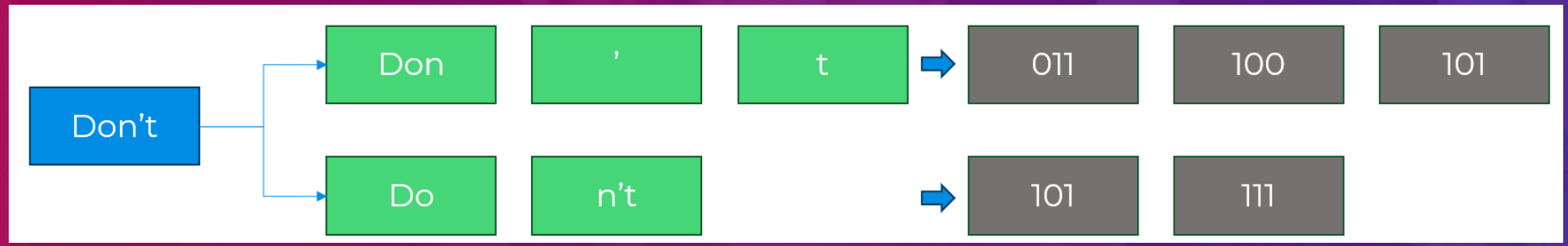


MODEL	DESCRIPTION	CONTEXT WINDOW
gpt-4-1106-preview	**GPT-4 Turbo** New The latest GPT-4 model with improved instruction following, JSON mode, reproducible outputs, parallel function calling, and more. Returns a maximum of 4,096 output tokens. This preview model is not yet suited for production traffic. Learn more.	128,000 tokens
gpt-4-vision-preview	**GPT-4 Turbo with vision** New Ability to understand images, in addition to all other GPT-4 Turbo capabilities. Returns a maximum of 4,096 output tokens. This is a preview model version and not suited yet for production traffic. Learn more.	128,000 tokens
gpt-4	Currently points to gpt-4-0613. See continuous model upgrades.	8,192 tokens
gpt-4-32k	Currently points to gpt-4-32k-0613. See continuous model upgrades.	32,768 tokens

What's a Token?

- LLMs are trained on text to predict text.
- Like other natural language processing (NLP) systems, they use tokenization as the essential text preprocessing step.
- Tokenization aims to parse the text into non-decomposing units called tokens.
- Tokens can be characters, subwords, symbols, or words, depending on the size and type of the model.
- Eventually the tokens end up as an integer or binary sequence.

"'tokens are not a real thing. they are a computer-generated illusion created by a clever engineer"
–@dril_gpt



What were the LLMs trained on?

Dataset	Type	Size/Samples	Tasks	Source	Creation	Comments
C4 [10]	Pretrain	806GB	-	Common Crawl	Automated	A clean, multilingual dataset with billions of tokens
mC4 [11]	Pretrain	38.49TB	-	Common Crawl	Automated	A multilingual extension of the C4 dataset, mC4 identifies over 100 languages using cld3 from 71 monthly web scrapes of Common Crawl.
PILE [276]	Pretrain	825GB	-	Common Crawl, PubMed Central, OpenWebText2, ArXiv, GitHub, Books3, and others	Automated	A massive dataset comprised of 22 constituent sub-datasets
ROOTs [277]	Pretrain	1.61TB	-	498 Hugging Face datasets	Automated	46 natural and 13 programming languages
MassiveText [114]	Pretrain	10.5TB	-	MassiveWeb, Books, News, Wikipedia, Github, C4	Automated	99% of the data is in English
Wikipedia [278]	Pretrain	-	-	Wikipedia	Automated	Dump of wikipedia
RedPajama [279]	Pretrain	5TB	-	CommonCrawl, C4, Wikipedia, Github, Books, StackExchange	Automated	Open-source replica of LLaMA dataset
PushShift.io Reddit	Pretrain	21.1GB	-	Reddit	Automated	Submissions and comments on Reddit from 2005 to 2019
BigPython [131]	Pretrain	5.5TB	Coding	GitHub	Automated	-
Pool of Prompt (P3) [17]	Instructions	12M	62	PromptSource	Manual	A Subset of PromptSource, created from 177 datasets including summarization, QA, classification, etc.
xP3 [146]	Instructions	81M	71	P3+Multilingual datasets	Manual	Extending P3 to total 46 languages
Super-NaturalInstructions (SNI) [18]	Instructions	12.4M	1616	Multiple datasets	Manual	Extending P3 with additional multi-lingual datasets, total 46 languages
Flan [16]	Instructions	15M	1836	Muffin+T0-SF+NIV2	Manual	Total 60 languages
OPT-IML [93]	Instructions	18.1M	1667	-	Manual	-
Self-Instruct [19]	Instructions	82k	175	-	Automated	Generated 52k instructions with 82k samples from 175 seed tasks using GPT-3
Alpaca [150]	Instructions	52k	-	-	Automated	Employed self-instruct method to generate data from text-davinci-003
Vicuna [151]	Instructions	125k	-	ShareGPT	Automated	Conversations shared by users on ShareGPT using public APIs
LLaMA-GPT-4 [152]	Instructions	52k	-	Alpaca	Automated	Recreated Alpaca dataset with GPT-4 in English and Chinese
Unnatural Instructions [280]	Instructions	68k	-	15-Seeds (SNI)	Automated	-
LIMA [177]	Instructions	1k	-	Multiple datasets	Manual	Carefully created samples to test performance with fine-tuning on less data
Anthropic-HH-RLHF [281]	Alignment	142k	-	-	Manual	
Anthropic-HH-RLHF-2 [170]	Alignment	39k	-	-	Manual	

GPT 3, for example, was trained on:

- Common Crawl
- WebText
- Books Corpora
- Wikipedia

Training Stages

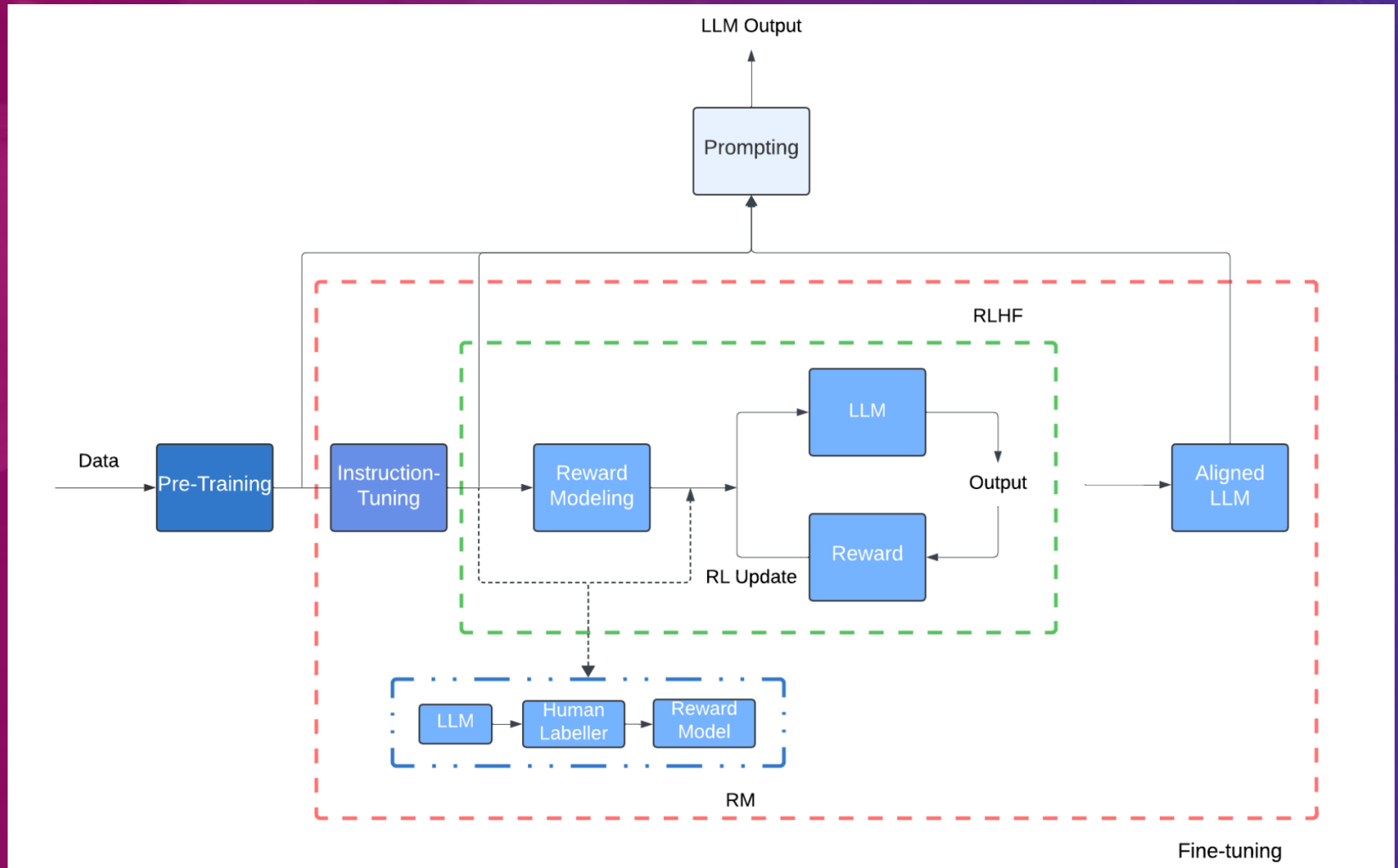


Fig. 6: A basic flow diagram depicting various stages of LLMs from pre-training to prompting/utilization. Prompting LLMs to generate responses is possible at different training stages like pre-training, instruction-tuning, or alignment tuning.

Fine Tuning

Fine-tuning improves on few-shot learning by training on many more examples than can fit in the prompt, letting you achieve better results on a wide number of tasks.

Once a model has been fine-tuned, you won't need to provide as many examples in the prompt. This saves costs and enables lower-latency requests.

At a high level, fine-tuning involves the following steps:

- Prepare and upload training data
- Train a new fine-tuned model
- Evaluate results and train again if needed
- Use your fine-tuned model

The challenge with fine tuning a pre-trained model can be catastrophic forgetting.
That is, in learning the new knowledge it lost significant chunks of its old knowledge, making the model less grounded or less powerful than it was before.

ChatGPT

Generative Pre-trained Transformer, models from OpenAI with the help of Microsoft

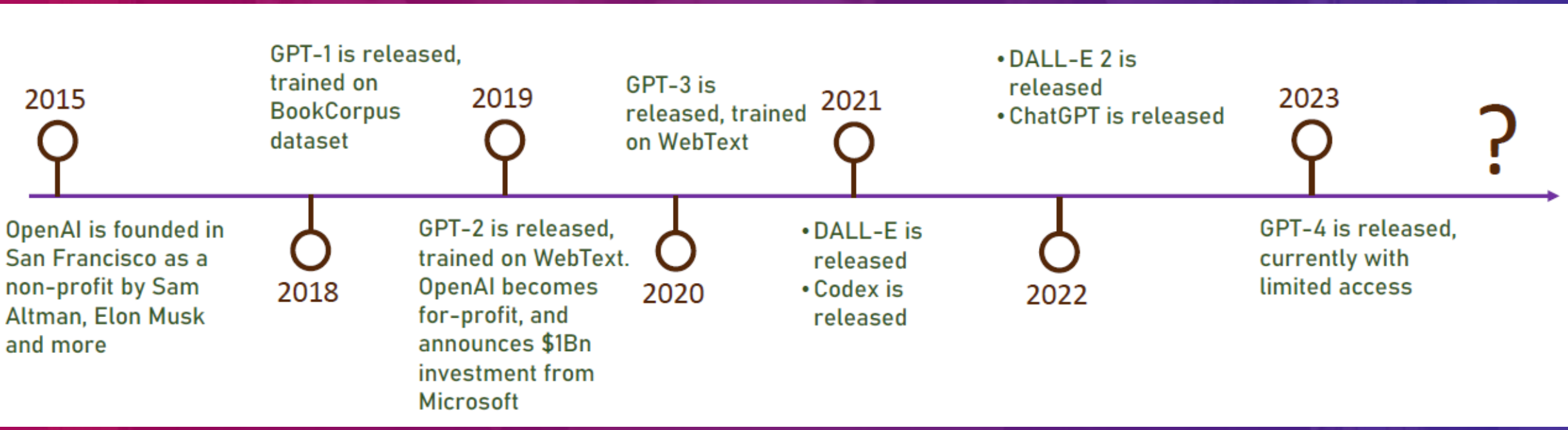
- ChatGPT is a conversational AI model developed by OpenAI.
- It was based on the GPT-3.5 architecture and now is also available on GPT-4
- Is designed to generate human-like responses to text inputs.
- ChatGPT can be used for a wide range of conversational applications, including chatbots, virtual assistants, and customer support systems.
- It is available as an API that developers can integrate into their applications to enable natural language interactions with users.

ChatGPT Models

- GPT-3.5 has 175 Billion parameters
- Cheaper to use than GPT-4
- GPT-4 has 1.5 Trillion parameters
- GPT-4 is considered the most advanced model to use as of the start of 2024
- https://openai.com/research/gpt-4
- It was built with the help of Microsoft

- ChatGPT is specifically fine-tuned for conversational interactions, making it adept at generating human-like text in a dialogue format
- ChatGPT uses WebText for its training data
- Contains ~ 8 million scraped web pages
- Emphasis on document quality
- Based on outbound links from Reddit which received at least 3 karma
- Currently, for GPT-3.5, it is only aware of data until September 2021.
- Based on GPT-4 being more recent ~ March 2022 (OpenAI did not disclose the exact date or the amount of parameters used in GPT-4)
- - Was trained using RLHF (Reinforcement Learning From Human Feedback)
- - So other than the 8 million scraped web pages it uses Human AI Trainers with a reward model to train the LLM

ChatGPT History



Prompt Engineering

Prompt engineering is the process of creating an input prompt for a large language model that coerces the model to provide the desired result.
Examples:

- Asking the model a question
- Providing examples of the problem and solution
- Providing additional context to the LLM
- Setting a style and tone for the response
- Instructing the model on how to process the input and format the output

Zero Shot Prompting

LLM's can be asked directly a question to answer or given a few examples of the input and response to guide them.

Zero Shot Prompting

Prompt:

```
Classify the text as positive, negative, or neutral.
Text: I thought the movie was just all right.
Sentiment:
```

Output:

```
negative
```

Definition

- LLM's are zero-shot learners capable of answering questions never seen before.
- LLM's are also zero-shot reasoners and can be prompted to generate answers to logical problems.

Few Shot Prompting

Prompt:

A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is:
We were traveling in Africa and we saw these very cute whatpus.
To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses the word farduddle is:

One complete example

The actual task

Few Shot Prompting

Output:

When we won the game, we all started to farduddle in celebration.

Definition

LLM's can be few-shot learners capable of answering questions using knowledge provided as examples.

LLM Utilization Patterns

Core

- Q&A
- Classification
- Zero shot prompting
- Chain of Thought

Augmented LLMs

- In-context learning (Few shot learning)*
- Retrieval Augmented Language Modeling (RALM)*
- Retrieval Augmented Generation (RAG)*
- Automatic Reasoning and Tool-use (ART)*

Multi-Turn Patterns

- Chain of Thought with
  - Self Consistency*
  - Tree of Thought*
- Reason and Act (ReAct)*
- Inner Monologue*
- Automatic prompt optimization: Automatic Prompt Engineer (APE) and Optimization by PROmpting (OPRO)*

* Needs supporting software infrastructure beyond direct use of LLM model completion endpoint.

What makes an LLM a Copilot

Enterprises are looking to leverage AI in core business functions, including

Marketing & Sales	Product & Services Development (R&D)	Service Operations (Call centers, back-office support)
• Craft first draft of text documents • Personalize marketing • Summarize text documents	• Identify trends in customer needs • Draft technical documents • Create new product designs	• Use of chatbots • Forecasting service trends or anomalies • Creating first drafts of documents

These scenarios have one thing in common:
The AI is used to assist a human directly with a specific task.

This is the very definition of an AI copilot.

Copilot Definition

At its most basic, a copilot uses enterprise supplied knowledge and generative AI models to author text, write code or render images.

This basic capability emerges in copilots which power these scenarios:

- Knowledge Management: Help users quickly find the information they seek and deliver at the right level and in the right format. Examples include summarization and sentiment analysis.
- Analytics: Help users quickly get to the data driven insights they seek. Examples include recommendations, predictions, anomaly detection, statistical analysis and data querying and reporting.

Working With OpenAI

Access over the API

OpenAI

OpenAI is an AI research and deployment company. Their mission is to ensure that artificial general intelligence benefits all of humanity

OpenAI

Demos

OpenAI

Labs

Orchestration

Orchestration Fundamentals

AI Orchestration

AI orchestration refers to the process of coordinating and managing the deployment, integration, and interaction of various artificial intelligence (AI) components within a system or workflow. This includes orchestrating the execution of multiple AI models, managing data flow, and optimizing the utilization of computational resources.

AI Orchestration ...

AI orchestration aims to streamline and automate the end-to-end life cycle of AI applications, from development and training to deployment and monitoring. It ensures the efficient collaboration of different AI models, services, and infrastructure components, leading to improved overall performance, scalability, and responsiveness of AI systems.

Essentially, AI orchestration acts as a conductor, harmonizing the diverse elements of an AI ecosystem to enhance workflow efficiency and achieve optimal outcomes.

Benefits of Orchestration

- Enhanced Scalability
- Improved Flexibility
- Efficient Resource Allocation
- Accelerated Development and Deployment
- Facilitated Collaboration
- Improved Monitoring and Management
- Streamlined Compliance and Governance

Examples of Orchestration

LangChain Vs Semantic Kernel

LangChain

Step	Component	Description
1	Model I/O	Interface with language models
2	Retrieval	Interface with application-specific data
3	Chains	Construct sequences of calls
4	Agents	Let chains choose which tools to use given high-level directives
5	Memory	Persist application state between runs of a chain
6	Response	Output of results

Semantic Kernel

Step	Component	Description
1	Kernel	The kernel orchestrates a user's tasks
2	Memories	Recall and store context
3	Planner	Mix and match plugins to execute
4	Connectors	Integration to external data sources
5	Plugins	Create custom C# or Python native functions
6	Response	Output of results

Orchestration

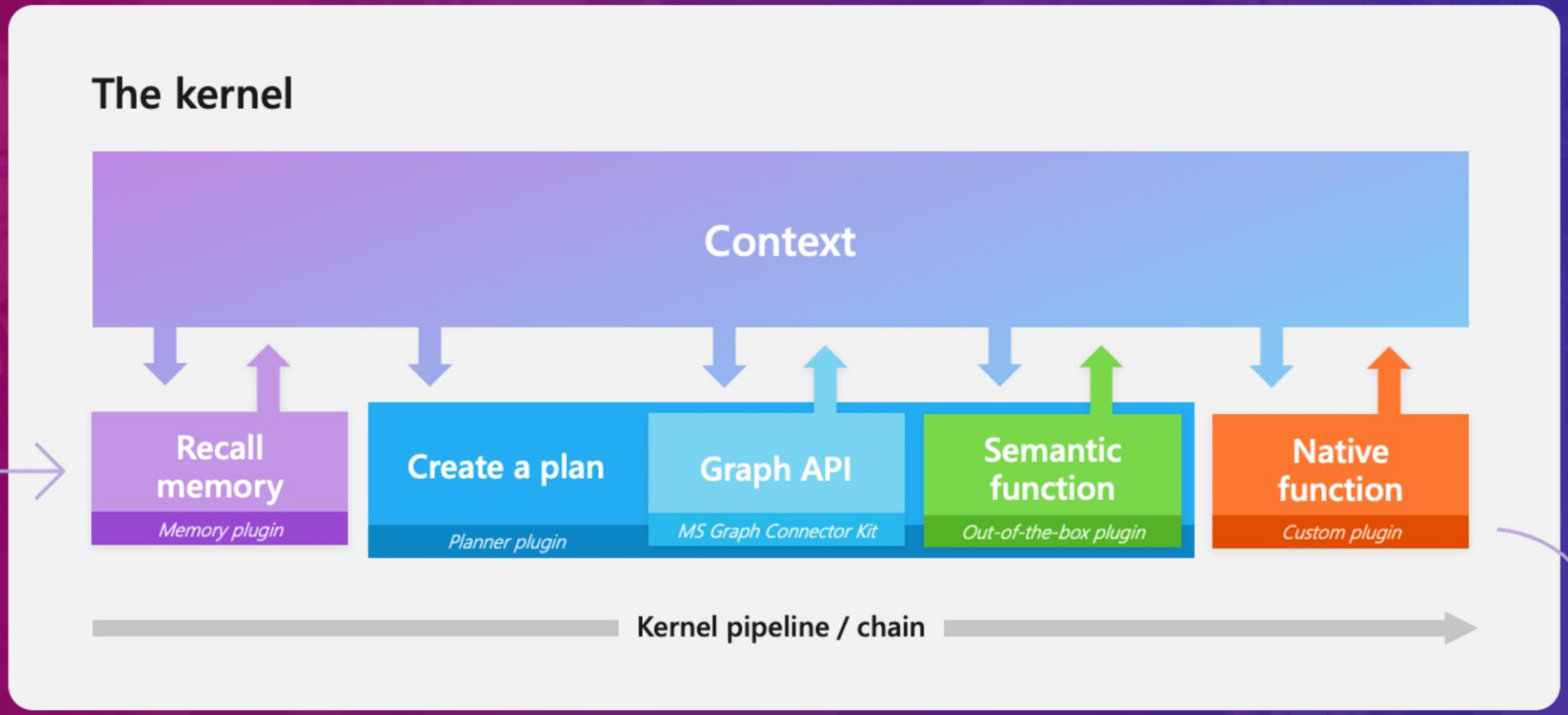
Semantic Kernel Fundamentals

Semantic Kernel Definition

Semantic Kernel is an SDK that integrates Large Language Models (LLMs) like OpenAI, Azure OpenAI, and Hugging Face with conventional programming languages like C#, Python, and Java. Semantic Kernel achieves this by allowing you to define plugins that can be chained together in just a few lines of code.

What makes Semantic Kernel special, however, is its ability to automatically orchestrate plugins with AI. With Semantic Kernel planners, you can ask an LLM to generate a plan that achieves a user's unique goal. Afterwards, Semantic Kernel will execute the plan for the user.

The Kernel



Please create a series of tasks to complete a marketing project. Once you're done, please summarize them and send them to my team.

Copilot
Here's the email with the summarized tasks in it.

New marketing tasks

Semantic Kernel

Demos

Sematic Kernel

Labs

Orchestration

Langchain Fundamentals

Langchain Definition

LangChain is a framework for developing applications powered by large language models (LLMs).

LangChain simplifies every stage of the LLM application lifecycle.

Use Cases:

- Question answering with RAG
- Extracting structured output
- Chatbots
- and more!

Langchain Expression Language (LCEL)

LangChain Expression Language (LCEL) is the foundation of many of LangChain's components and is a declarative way to compose chains. LCEL was designed from day 1 to support putting prototypes in production, with no code changes, from the simplest “prompt + LLM” chain to the most complex chains.

Benefits:

- First-class streaming support
- Async support
- Optimized parallel execution
- Retries and fallbacks
- Access intermediate results

Langchain Agents vs Chains

Chains: A sequence of actions is hard coded.

- A chain is a sequence of call to components, which can include custom code, LLM's and other chains.
- A chain is basically a function, with a signature defining the inputs and the output and an internal function that implements the logic that is run.
- **Foundational chains are:** LLM, Sequential, Router and Transformation. Router is interesting for FLM as a pattern.
- **Chains for working with documents:** Stuff, Refine, Map reduce, Map re-rank. Refine and Map re-rank are interesting for FLM as patterns.

Agents: An LLM chooses which actions to take and in which order.

- **AgentExecutor**: runs an action, observation loop using the agent to determine the action given an observation.
- **Agent**: decides what step to take next, using an LLM along with system prompt, context and prompting strategy
- **Toolkit**: A set of, typically, 3-5 tools that an agent picks from to execute.
- **Tool**: The function that the agent executes. Each tool is described to the agent in text.
- There are agents like Plan and Execute that are multi-agent utilizing one agent for coming up with plan of all steps upfront and another agent for executing tools (in the CoT or ReAct pattern).
- There are many prebuilt agents and toolkits that support them, loosely grouped as Integrations. Some interesting ones for FLM include JSON (for handling large JSON dicts), Office365 (email and calendar), OpenAPI (consume arbitrary API's), Pandas DataFrame (question answering over data frames), PlayWright (web scraping), PowerBI Dataset (query pbi dataset), Python (write and execute Python code), SQL Database (interact with a range of SQL databases), Vectorstore (handling of in-memory and external vector stores)

Langchain SQL Agent Approach

```
from langchain import OpenAI, SQLDatabase
from langchain_experimental.sql import SQLDatabaseChain

pg_uri = f"postgresql+psycopg2://{username}:{password}@{host}:{port}/{mydatabase}"
db = SQLDatabase.from_uri(pg_uri)
llm = OpenAI(temperature=0)

PROMPT = """
Given an input question, first create a syntactically correct postgresql query to run,
then look at the results of the query and return the answer.
The question: {question}
"""

db_chain = SQLDatabaseChain.from_llm(llm=llm, db=db, verbose=True, top_k=10)

question = "What usename are there in the pg_user table?"
# use db_chain.run(question) instead if you don't have a prompt
db_chain.run(PROMPT.format(question=question))
```

Langchain

Demos

Langchain

Labs

Retrieval Augmented Generation

RALM and RAG

The Misconception

Leveraging large language models is all about prompt engineering, it's as easy as:

As an enterprise, how do I?

provide a branded chat user experience atop this?	manage conversation history on a per user basis?	enable API integration into my automated workflows?	use more advanced interaction patterns that are recursive, create code or use tools?
scale and batch load thousands to millions of documents as knowledge for the model?	provide all the data from my enterprise data estate?	optimize the vectorization approach to best suit the content?	measure and optimize completion quality?
leverage other LLM's (Llama 2, Mistral) alongside ChatGPT models?	leverage the LLM orchestrators (LangChain, Semantic Kernel, Prompt Flow) I want to use?	self-host LLMs so my data doesn't leave my environment?	govern token use down to the user or app level?
keep sensitive knowledge sources private to authorized users?	keep sensitive data (PII, trade secrets) from being sent to the LLM?	manage having multiple AI agents?	scale the solution to support 100k users or multiple data sovereignty regions?

Augmented LLMs

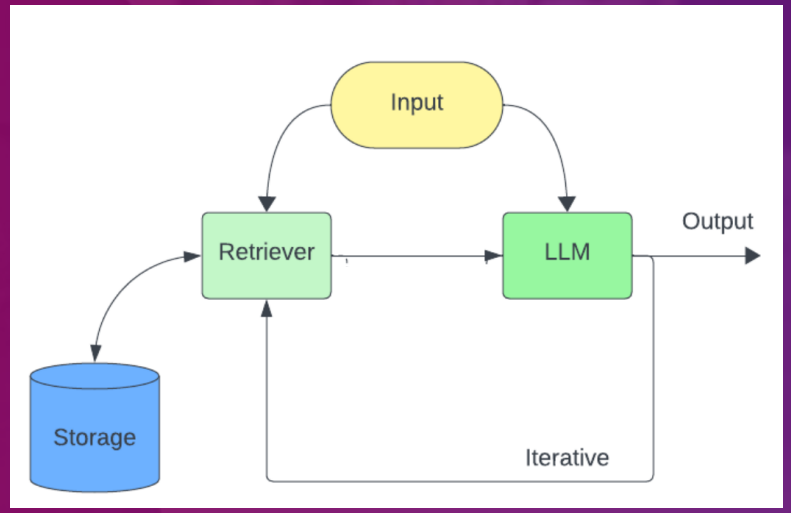
In-context learning, also referred to as Few shot learning, enables LLMS to learn from examples supplied with the input.

- Augmented LLM's use in-context learning by augmenting LLM's with external tools and data.
- Augmented LLM's provide an alternative way to adapt the model without fine tuning.
- There are several utilization patterns that enable this. The two most common including:
  - RAG – which provides relevant data to the context
  - ART – which enables the LLM to choose from a toolbox of tools to support data access and processing

Retrieval Augmented Language Modeling (RALM)

LLMs have limited memory and outdated information which leads to inaccurate responses, retrieving relevant info from external up to date storage enables answers that are accurate. This family of patterns is referred to as RALM.

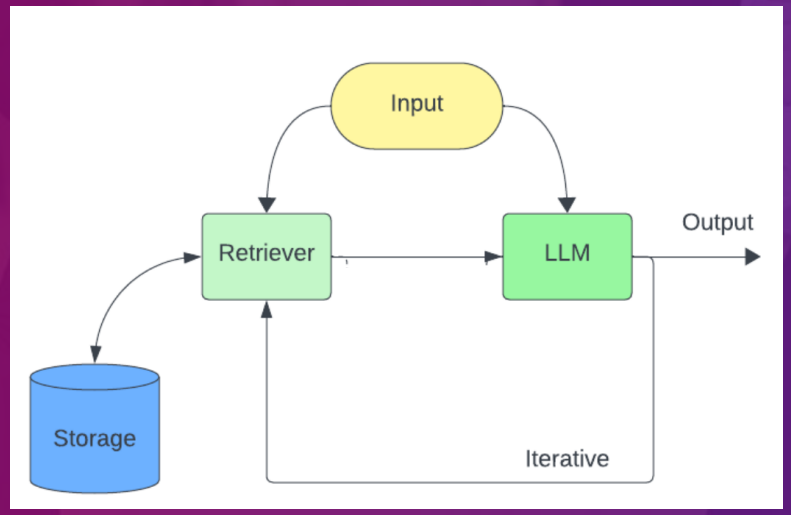
- The storage can be durable storage like a database or transient (in-memory).
- RALM has a retriever and a language model. Retriever plays crucial role in providing the right information to the context. Providing the wrong data to context can produce untrustworthy responses and hallucinations.
- RALM is one way to adapt the model without fine tuning



Understanding the Retriever

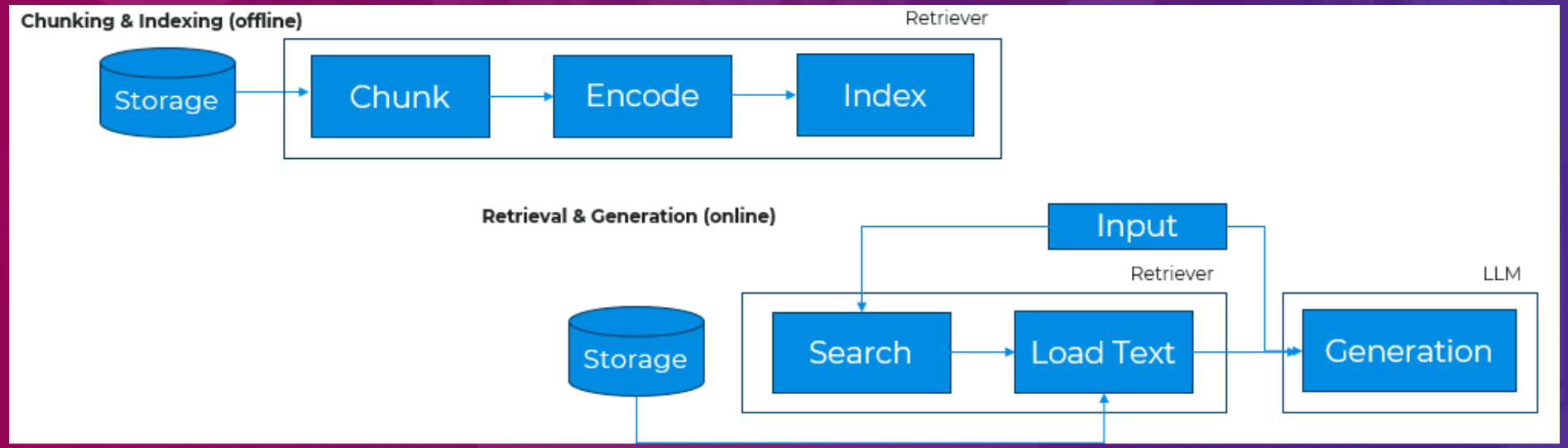
The retriever is all important to RALM patterns that is easily overlooked

- In-context learning is heavily restricted by the LLM's input context length
- Generally, the retriever performs some kind of search using some kind of index over the data in storage to find the right context data
- This search can be performed with information retrieval/ranking algorithms like BM25 or TF-IDF. These indices can be seen as representing the question and context document as sparse vectors.
- OR it can be performed using dense vector embeddings of both the question and context and doing a nearest neighbor search
- OR some hybrid combination of both (that combines results from a vector search with faceted search or semantic search)



RAG Utilization Patterns

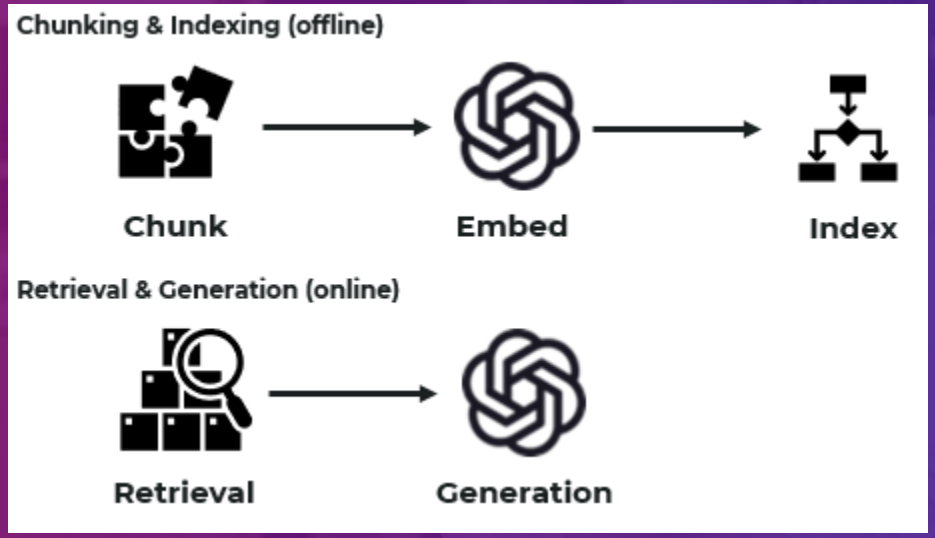
- In the RALM family, the most famous pattern is RAG.
- Its not just one pattern, it has several variants that vary in the implementation of the Retriever and LLM.
- The RAG pattern has two flows:
  - Chunking & Indexing: In an offline process the stored data is broken into size limited chunks, encoded and then indexed, typically with a vector index. This is an offline, pre-processing step because the process typically takes some time.
  - Retrieval & Generation: In response to completion requests, the retriever searches its index to get relevant chunks of text data, load that data from storage, concatenate that context data with the input prompt and then send that to LLM for completion generation.



Current RAG Pattern

The typical Retrieval Augmented Generation (RAG) pattern differs from the original RAG in replacing fine tuned encoders for vectorization with an OpenAI GPT embedding endpoint, and the fine-tuned completions model with an OpenAI GPT completions endpoint, and the retriever relies on service providing an index and another providing the documents and orchestration of the processes.

- The retrieval phase searches relevant documents or data.
- It uses this context to generate informed responses.



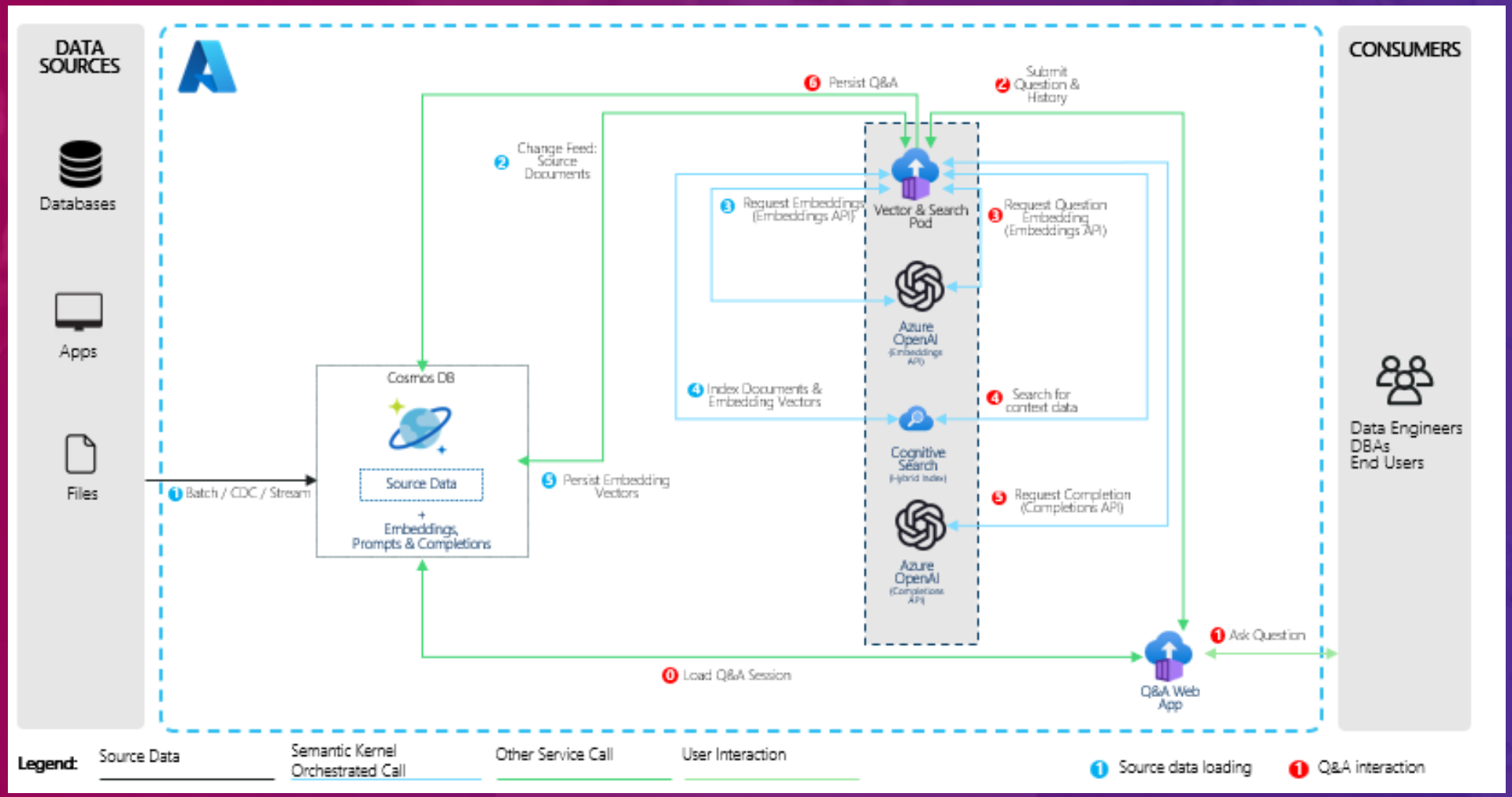
Benefits of RAG

- Enhanced Accuracy
- Contextual Relevance
- Scalability

RAG is ideal for Q&A systems, content creation, and data-driven decision-making
It combines the best of both worlds – robust data retrieval and creative response generation

* You need software infrastructure to run the retrieval. The model cannot run the retrieval on its own.

RAG infrastructure in Azure



What RAG is NOT

At its core, RAG is a powerful information retrieval solution for utilizing large language models by enhancing their parametric knowledge with a non-parametric source.

In plain English, RAG enhances the output capability of language models in knowledge intensive tasks.

RAG IS a Zero Shot Retrieval Augmentation pattern.

However, it's easy to confuse RAG with other patterns, especially since RAG has entered the vernacular like Kleenex is to tissue The RAG pattern **IS NOT** these things:

- The one pattern to rule them all.
- A pattern that uses tools or plugins. It's not the ART pattern. The retriever, the generator and the retriever's data source are the only components.
- A multi-turn pattern. For example, it cannot run in a loop by itself to refine the context data it is using or verify that its output is valid or reasonable.
- Chain of Thought, though it is often combined with Chain of Thought prompting.
- ReAct, though RAG is often used as a step in the plan executed by a ReAct pattern.
- The only pattern for Retrieval Augmented Language Modeling.
- A pattern for tasks where the data changes frequently.

RAG

Demos

RAG

Labs

Retrieval Augmented Language Modeling

COT, ReAct, ART, APE and Inner Monologue

Chain of Thought

CoT is a popular technique for prompt engineering. It aims to break down a complex idea into multiple steps, allowing the language model to work on simpler subproblems rather than tackling a single large problem.

Core Idea: It encourages transparent and traceable reasoning by explaining the reasoning process at each stage before progressing to the next.

Implementation: The prompt includes intermediate steps where the language model explains its thought process.
Helpful for tasks requiring logical deductions or mathematical calculations.

Limitations: Relies heavily on accurate intermediate steps. Can be inefficient for highly intricate problems.

Chain of Thought (COT) Pattern

Chain-of-thought (CoT) prompting adds reasoning steps to standard zero-shot and few-shot examples for more complex tasks

Instructs the model to proceed step-by-step and present all steps involved

Oftentimes results in greater accuracy over prompts that simply provides examples by forcing the model to break the task down into smaller steps.

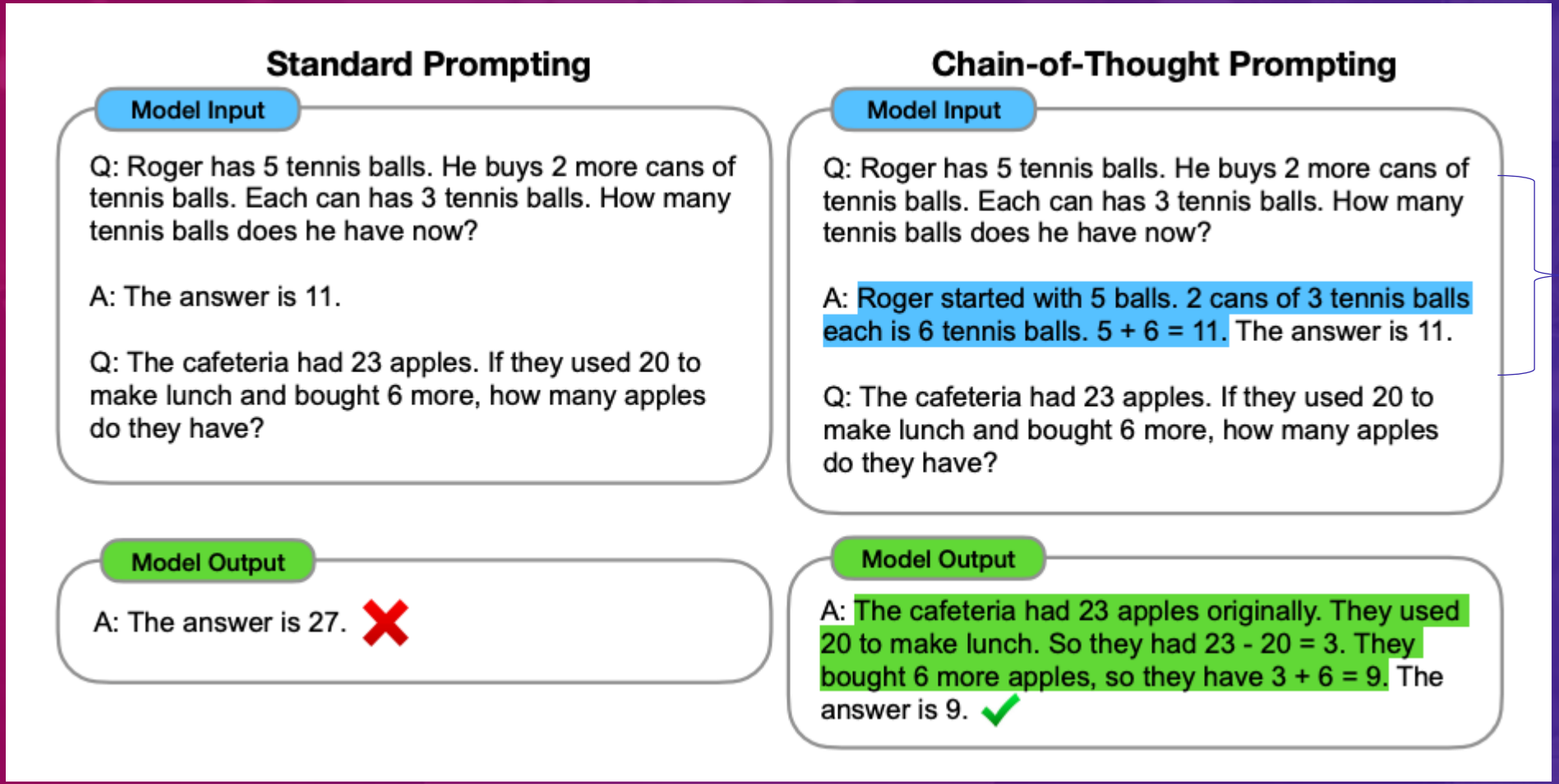


Image Source: Wei et al. (2022)

ReAct – (Reasoning & Acting)

ReAct is an advanced technique that combines reasoning and action within a prompt. While solving a complex problem, ReAct ensures that the language model performs the following tasks for every subtask:

Reason: Reasoning about current states and what needs to be done.

Action: Taking an action based on the reasoning.

Core Idea: Enables language models to interact with the environment and learn from consequences.

Implementation: The prompt guides the model to reason about a situation, suggest actions, and analyze outcomes.

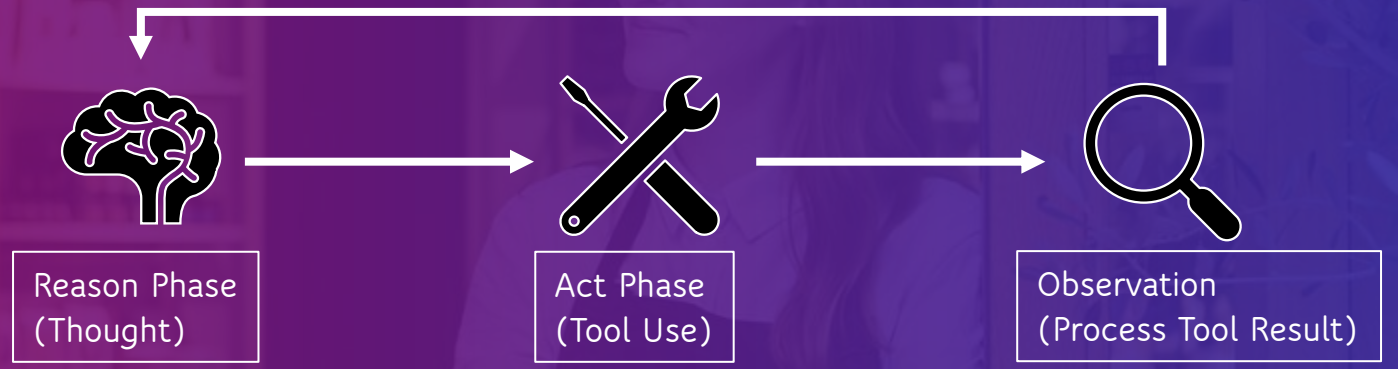
Benefits: Useful for tasks involving real-world simulations or game playing. Allows learning through trial and error.

Limitations: Relies heavily on accurate simulations and observation. May face ethical concerns if actions have real-world consequences.

ReAct – (Reasoning & Acting)...

The ReAct (Reason + Act) pattern is where LLMs are used to generate both reasoning traces and task-specific actions in an interleaved manner, leading to more reliable and factual responses

- Reasoning traces allow model to induce, track and update action plans and handle exceptions
- Action step uses external tools to gather information



Benefits of ReAct

- Improved human interpretability
- Improved trustworthiness

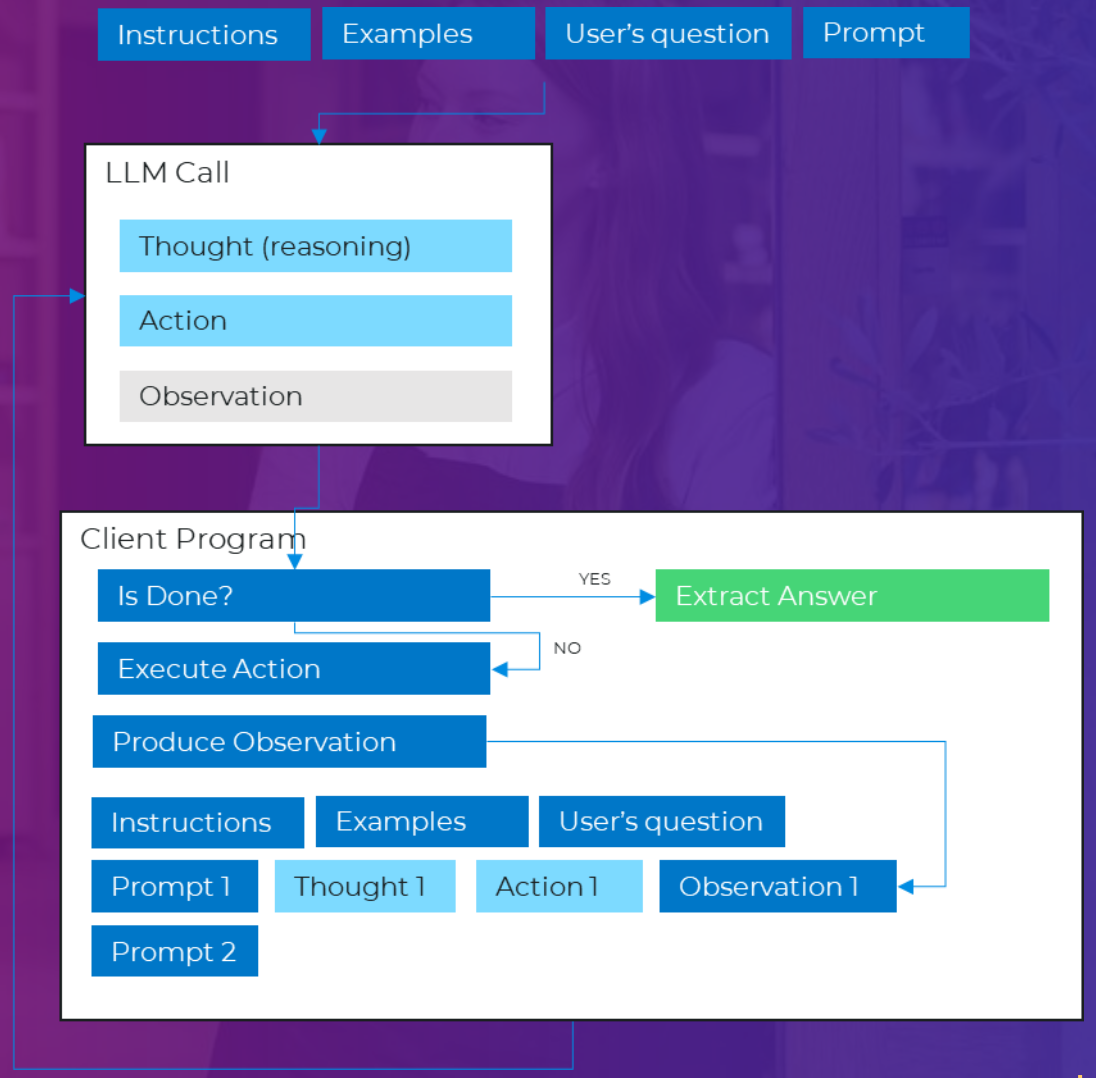
ReAct prompts LLMs to generate verbal reasoning traces and actions for a task. This allows the system to perform dynamic reasoning to create, maintain, and adjust plans for acting while also enabling interaction to external environments (e.g., Wikipedia) to incorporate additional information into the reasoning.

ReAct Example...

Question: What movie did actress Irene Jacob complete before the American action crime thriller film directed by Stuart Bird?
Thought 1: I need to search Irene Jacob and find the movie she completed before the American action crime thriller film directed by Stuart Bird.
Action 1: Search[Irene Jacob]
Observation 1: Irène Marie Jacob (born 15 July 1966) is a French-Swiss actress known for her work with Polish film director Krzysztof Kieślowski.[1] She won the 1991 Cannes Film Festival Award for Best Actress for the Kieślowski film The Double Life of Veronique, and was nominated for the BAFTA Award for Best Actress in a Leading Role for her 1994 film Three Colours: Red. Her other film appearances include The Secret Garden (1993), Beyond the Clouds (1995), U.S. Marshals (1998), and Eternity (2016).. Irène Jacob was born in Suresnes, Hauts-de-Seine, a western suburb of Paris.[2] The youngest child with three older brothers, she was raised in a highly educated and intellectual family and environment: her father, Maurice Jacob, was a physicist; her mother, a psychologist; one brother, Francis Jacob, a musician; and her other two brothers, scientists.[3][4] In 1969, at the age of three, Irène moved with her family to Geneva, Switzerland, where she became interested in the arts.. My family was very shy with feelings and never spoke about them, but we evolved a little bit.

Thought 2: The passage does not mention the movie Irene Jacob completed before the American action crime thriller film directed by Stuart Bird. Maybe I can look up "before". **Action 2:** Lookup[before]
Observation 2: (Result 1 / 2) It was not something he talked about beforehand; he would only work on the set.

Thought 3: The movie Irene Jacob completed before the American action crime thriller film directed by Stuart Bird is Eternity.
Action 3: Finish[Eternity]



Automated Reasoning and Tool-use (ART)

The ART (Automated Reasoning and Tool-use) pattern refers to the ability of AI systems to apply logical reasoning and use external tools or databases to solve complex problems. It combines advanced reasoning capabilities with external resources to enable AI to go beyond pre-programmed knowledge and operations.

- Reasoning process involves logical deduction, inference, and problem-solving strategies
- Tool-use aspect employs external tools like calculators and databases or accesses additional data sources to enhance problem-solving

ART is useful in fields like advanced research, complex data analysis, and decision support systems

Benefits and significance of ART

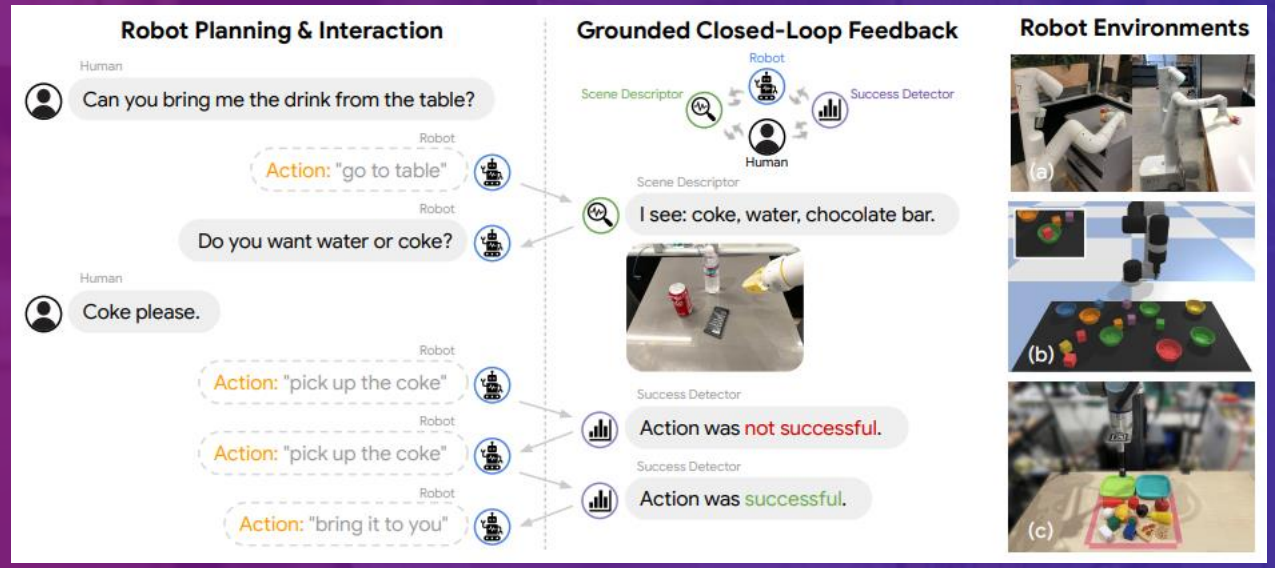
- Handles Complex Problems
- Evolutionary Step
- Bridging the Gap

ART's continued development could lead to AI systems that are more versatile, effective, and capable of handling a wider range of complex tasks.

Inner Monologue

Inner monologue supports agent planning and environment interaction where the agent not only needs to consider what skills to do but also how and when to do them and respond to external feedback.

- Feedback provided to model is provided as plain text.
- Unlocks several emergent capabilities from the LLM



Benefits of Inner Monologue

- Enables robust interaction with environments

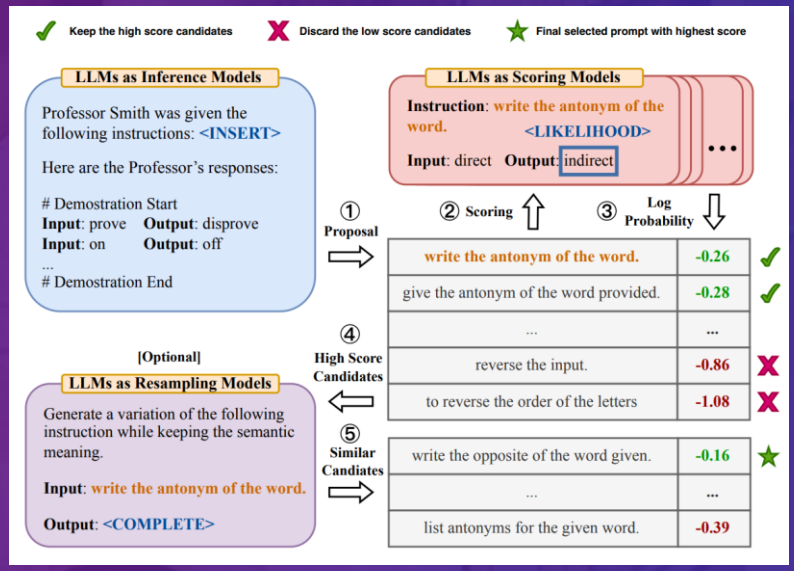
Inner Monologue, in providing environment feedback to the LLM, results in several emergent capabilities:

- Continued Adaptation to New Instructions: For example, the model can react to a human interaction to change its goal mid-task.
- Self-Proposing Goals under Infeasibility: Instead of mindlessly following an instruction, it can interactively problem solve and propose alternative goals to achieve. For example, retrying.
- Non-Static Understanding of the Environment: As the environment is affected by the agent, the agent is aware of and can describe the changes to the environment.

Automatic Prompt Engineer (APE)

APE generates several instruction candidates, either via direct inference or a recursive process based on semantic similarity, executes them using the target model, and selects the most appropriate instruction based on computed evaluation scores

- Uses LLMs for prompt scoring and prompt resampling
- Recommends the best prompt



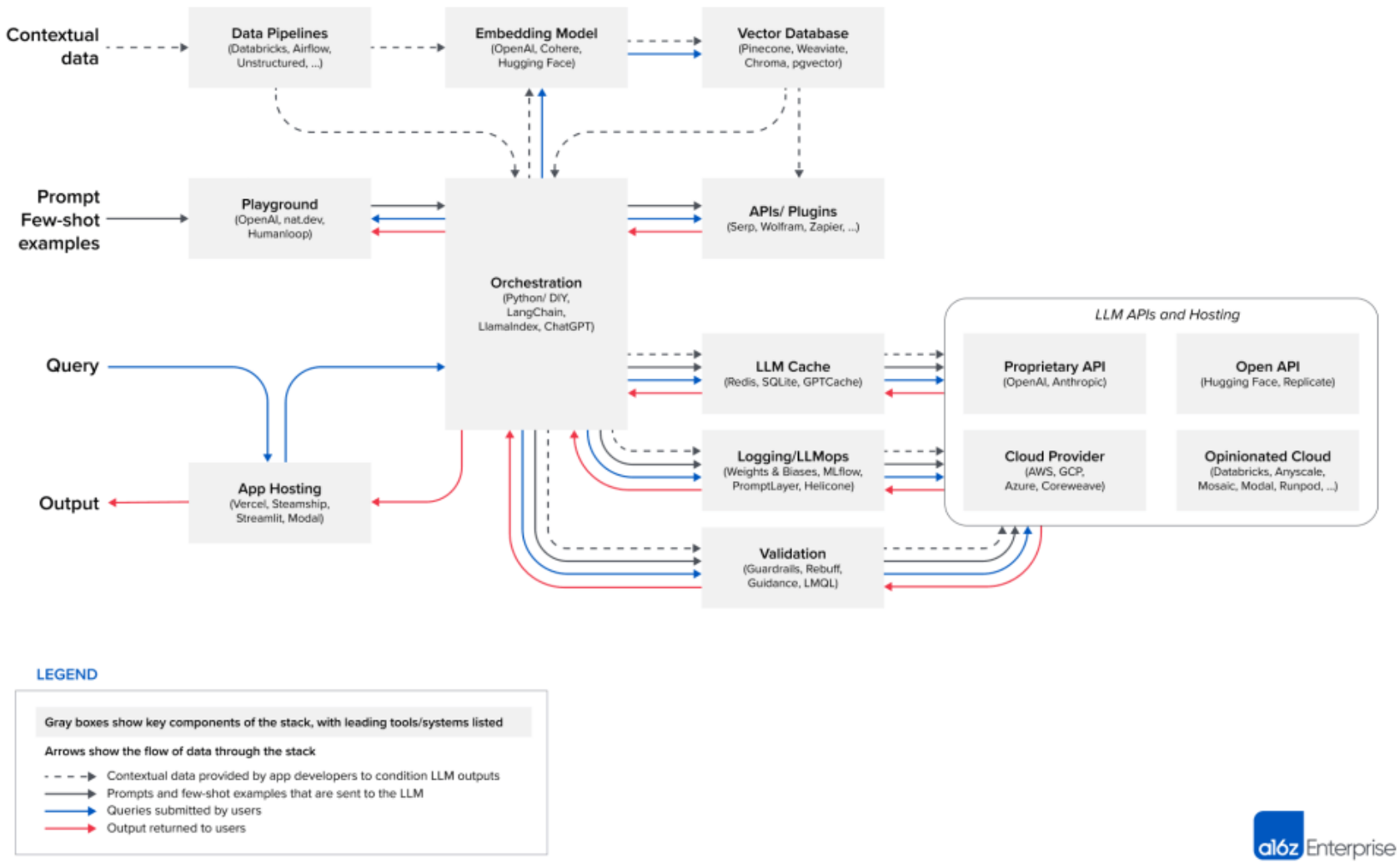
Benefits of APE

- Improves performance of LLM with better prompting

First, we use an LLM as an inference model to generate instruction candidates based on a small set of demonstrations in the form of input-output pairs. Next, we guide the search process by computing a score for each instruction under the LLM we seek to control. Finally, we propose an iterative Monte Carlo search method where LLMs improve the best candidates by proposing semantically similar instruction variants.
The score function used can vary, but execution accuracy is most commonly used to measure the alignment between the dataset and the data the model generates. The execution accuracy of a generated instruction is measured by testing whether LMs can correctly perform the task in a zero-shot manner by using the generated instruction alone, without any demonstrations.

The Complex reality

In reality, an enterprise grade LLM powered solution has a lot of moving parts (but don't just take our word for it):



Source: https://a16z.com/2023/06/20/emerging-architectures-for-llm-applications

Making the architecture Enterprise Grade

Make it secure & governable

- For RALM scenarios, enforce that the LLM cannot use as context anything the end user would not normally be allowed to know.
- Provides defense in depth with fine-grain security controls over data used by agent
- Provide guardrails to protect against misuse
- Enable pre- and post- completion filters that guard against attacks and protect against data exfiltration
- Red-team the solution
- Enable multiple LLM agents with specific purposes and provide support for private agents that are uniquely configured to each user
- Provide centralized configuration, logging and monitoring

Make it scalable & extensible

- Extend the orchestrators so that they can support more than a single OpenAI endpoint so that you can scale up completions capacity
- Enable support for LLM's beyond OpenAI for scenarios that need it
- Enable extensible integration with new enterprise data sources used by LLM for in-context learning
- Enable prompt functions – that is enabling API implementation using English instead of code

RALM

Demos

RALM

Labs

Working with Embedding Models

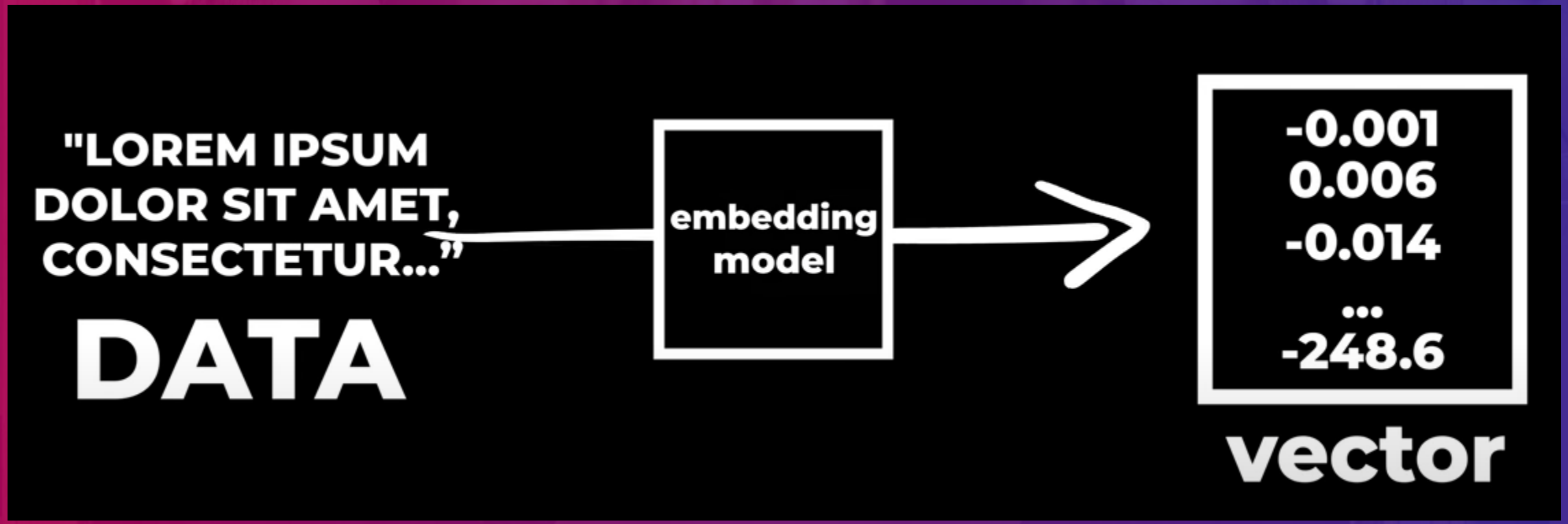
The power of Embedding Models

Embeddings basically convert text to coordinates that maintain relative distances between text that has similar meaning

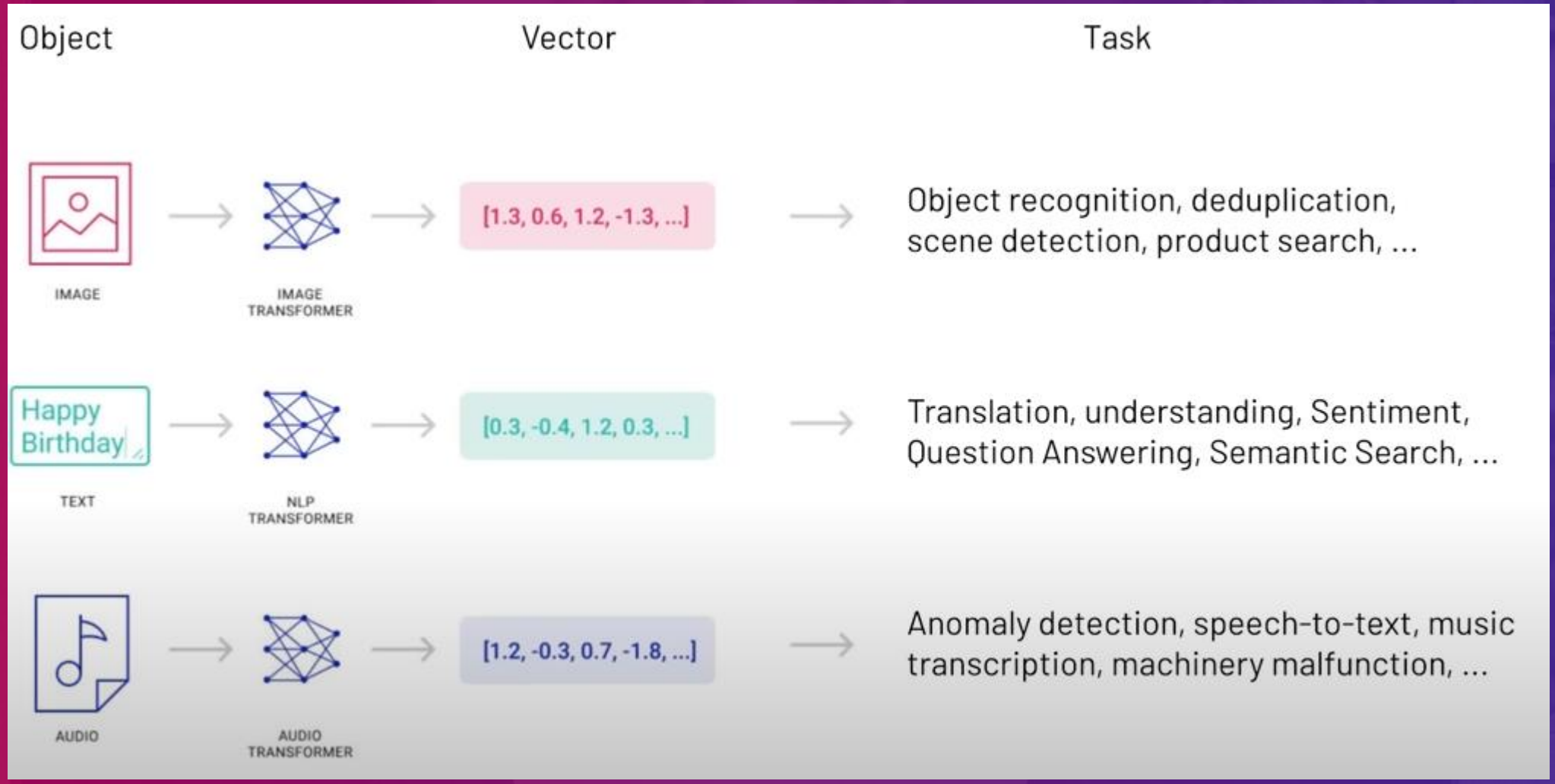
Applications:

- Semantic similarity
- Clustering
- Full-text search
- Classification
- Recommendations

How Embedding Works



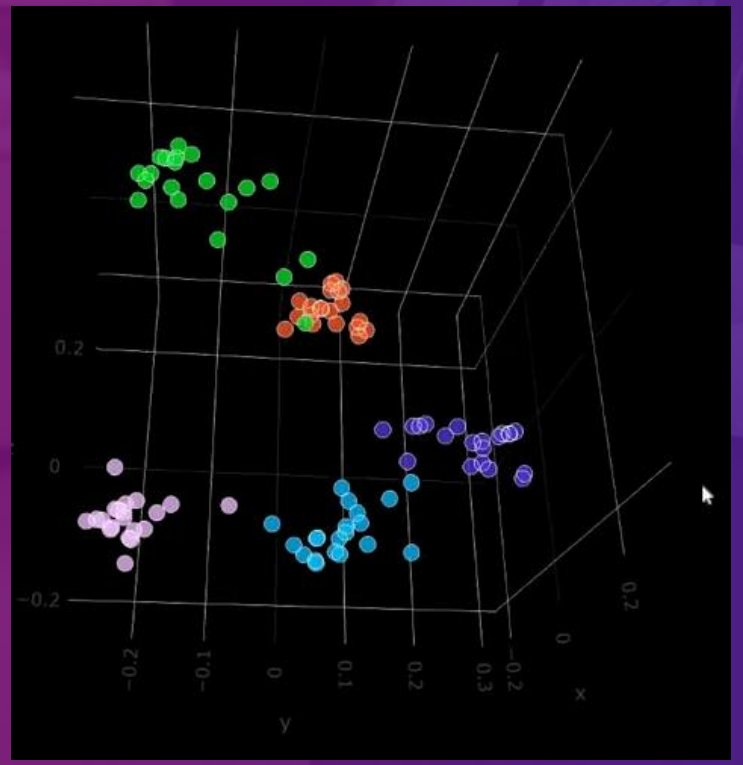
Object → Vector → Task



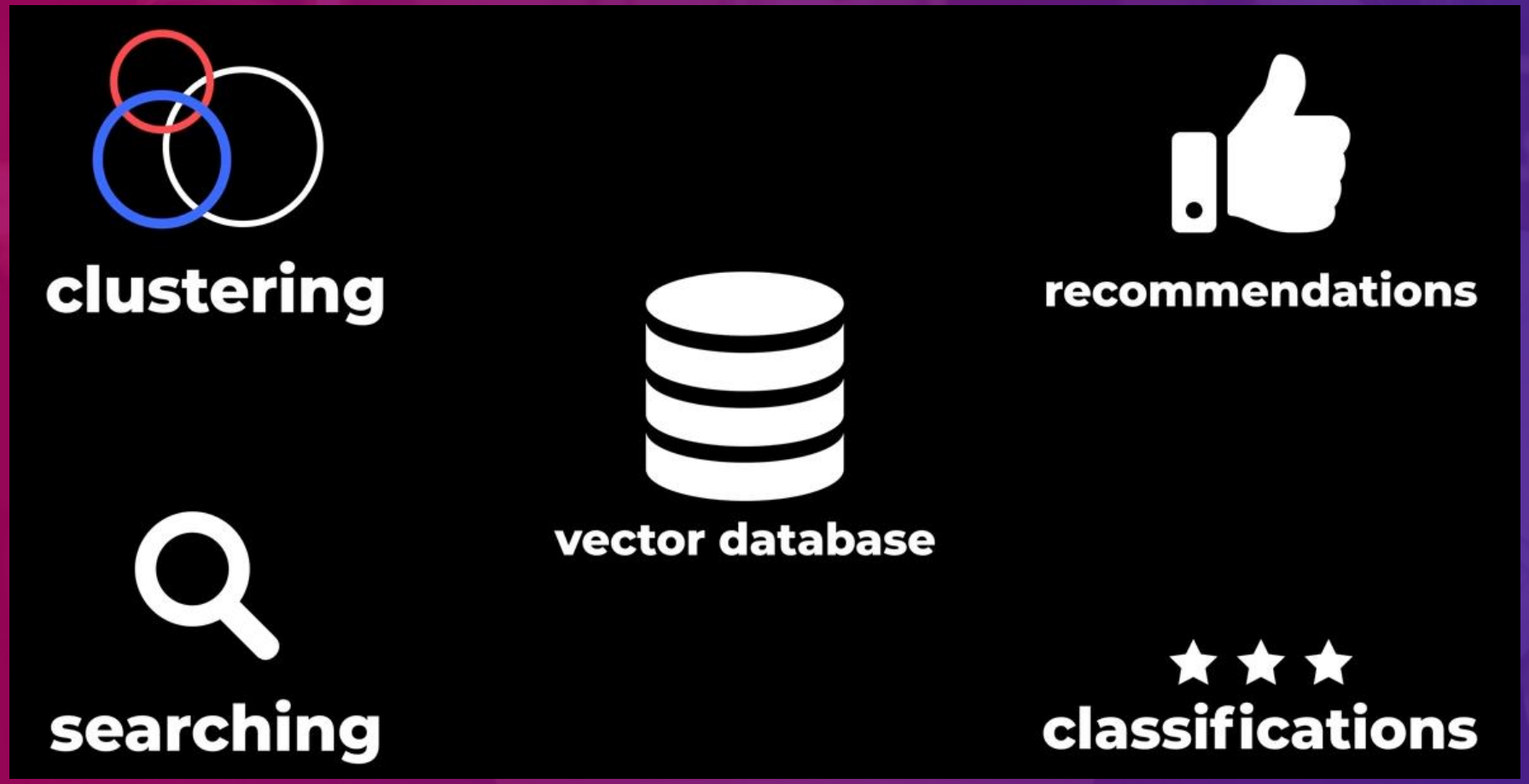
Similarity

dog
[0.1, 0.51, 0.6, 0.2, 0.8...]

puppy
[0.3, 0.5, 0.9, 0.8, 0.4...]



Use of Embeddings – Use Cases



* Embeddings does not store anything.

Embeddings

Demos

Embeddings

Labs

Vectorization Fundamentals

Foundation for Contextual Data Retrieval

Vector embeddings are numerical representations of textual data, transforming words, sentences, or documents into vectors in a high-dimensional space. Embeddings capture semantic meaning and relationships, enabling efficient and meaningful comparisons of textual data. Vector indexes are ultimately a flavor of a search index; we're just solving a search problem.

Indexing for retrieval

Indexes are structures that organize these embeddings, allowing for quick retrieval of the most relevant vectors (and thus, the corresponding textual data) in response to a query.

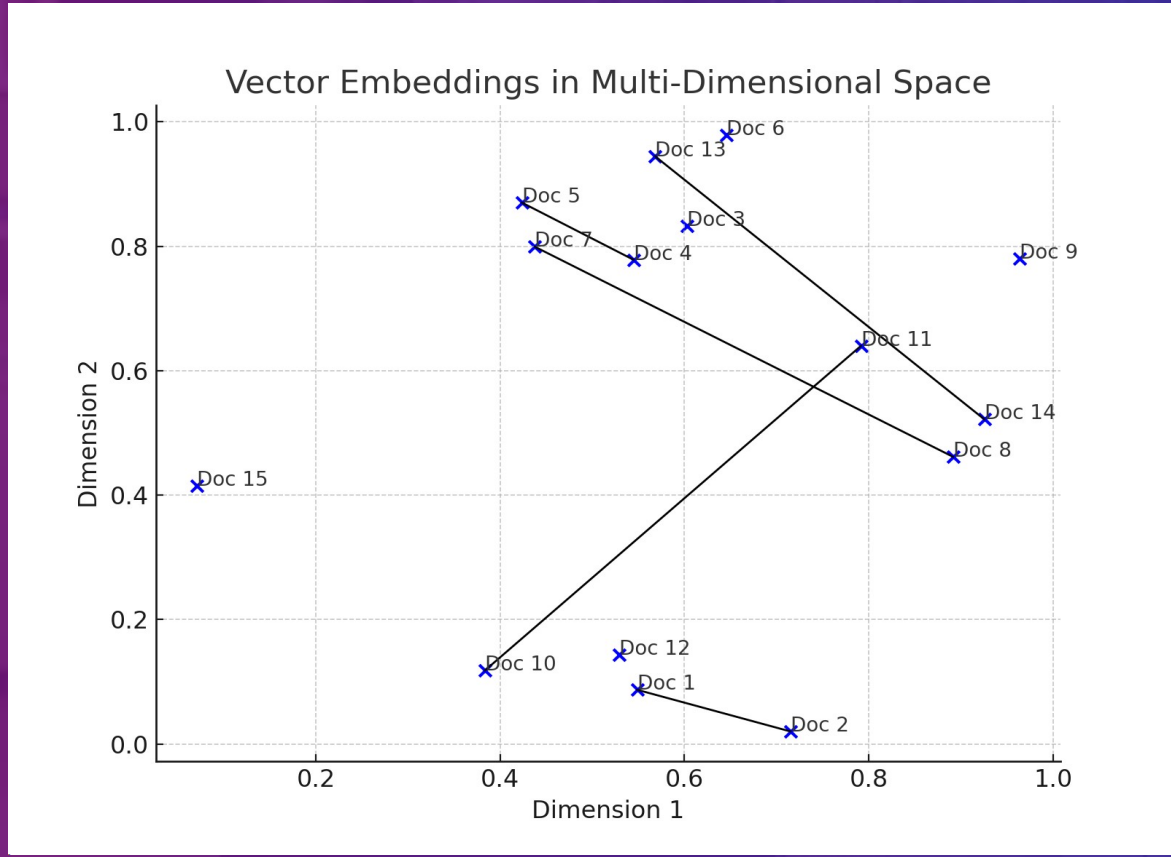
Role in contextual data retrieval

Vector embeddings and indexes are crucial for the retrieval phase in RAG, enabling the model to find the most relevant contextual information efficiently.

Benefits and challenges

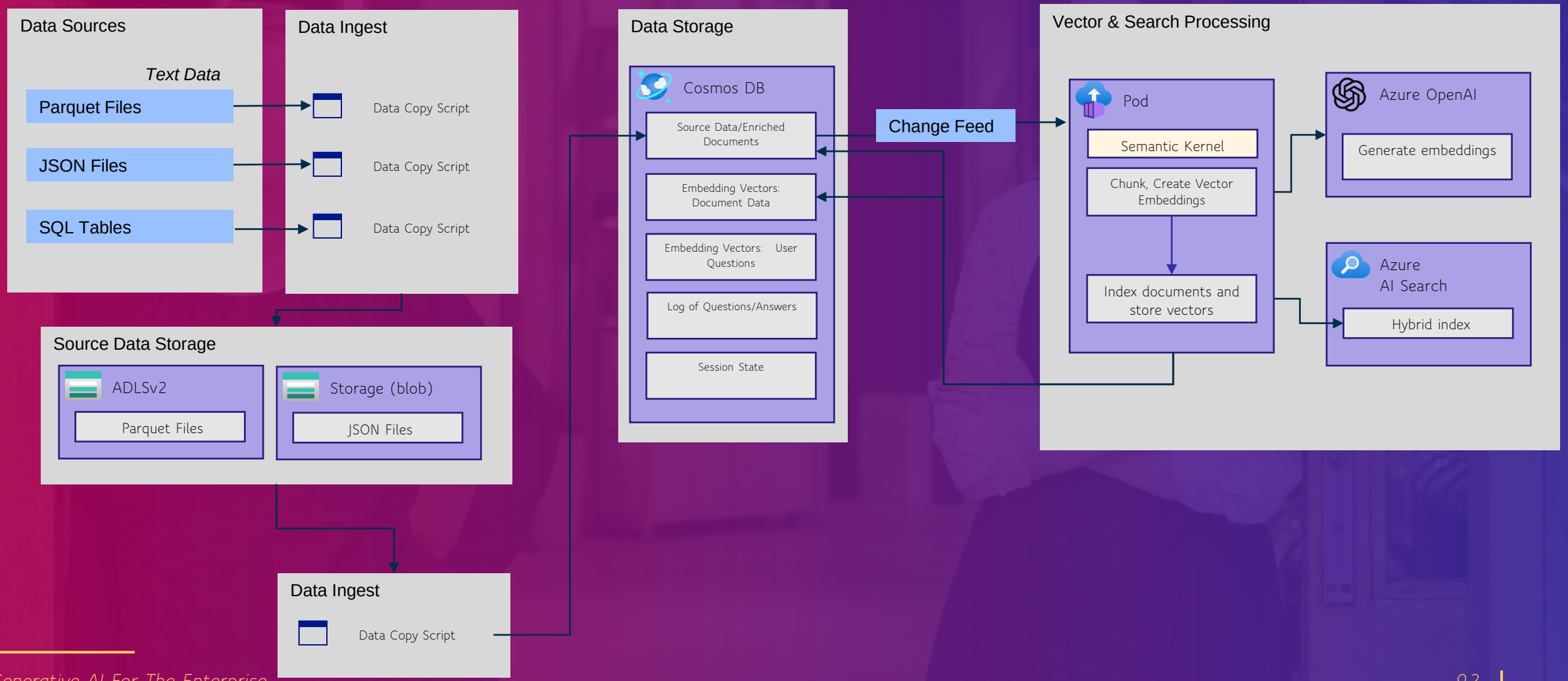
Benefits: Speed and relevance in retrieving information, scalability, and ability to handle large datasets.

Challenges: Complexity in creating effective embeddings, computational demands, and maintaining up-to-date indexes.

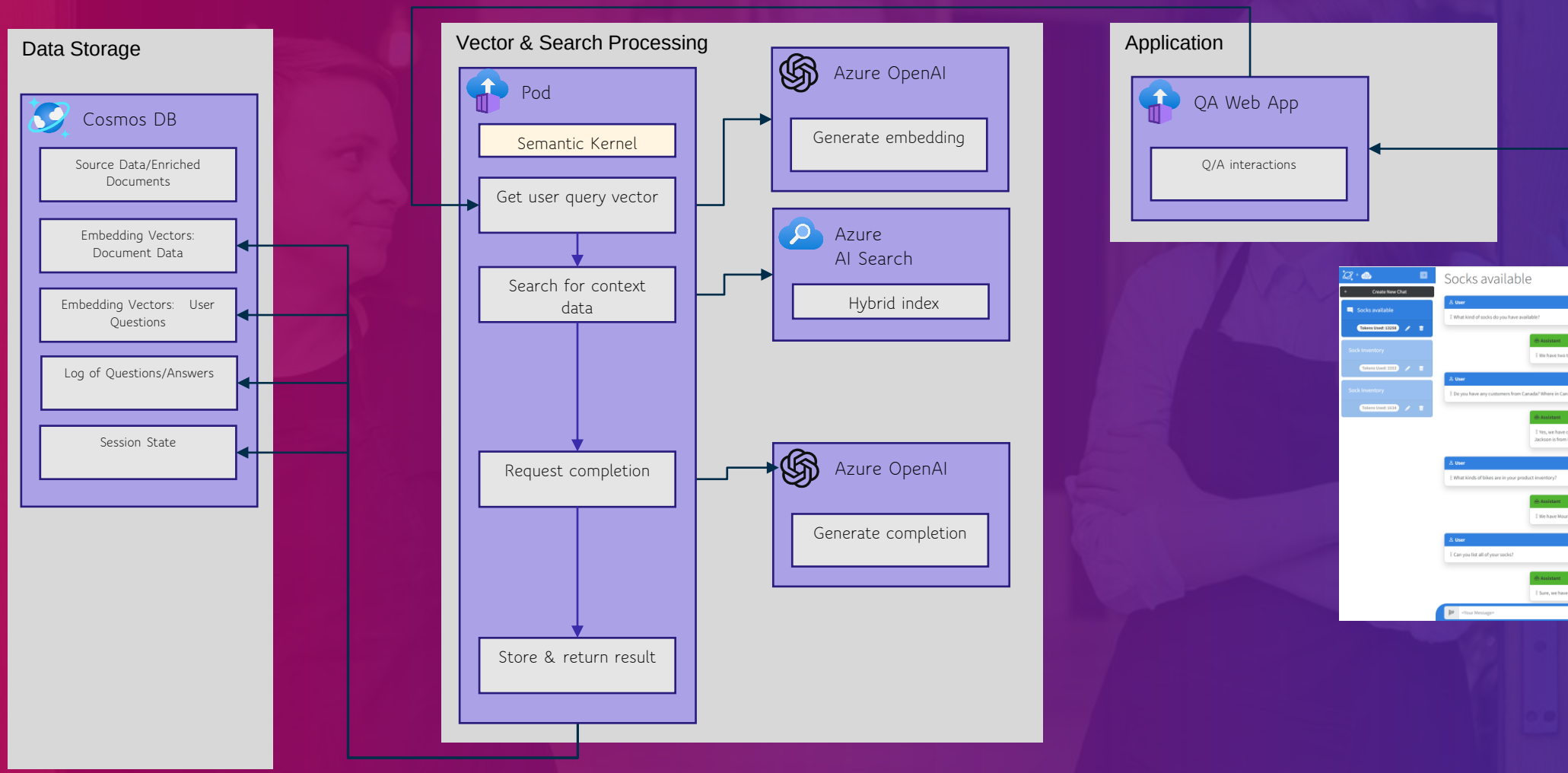


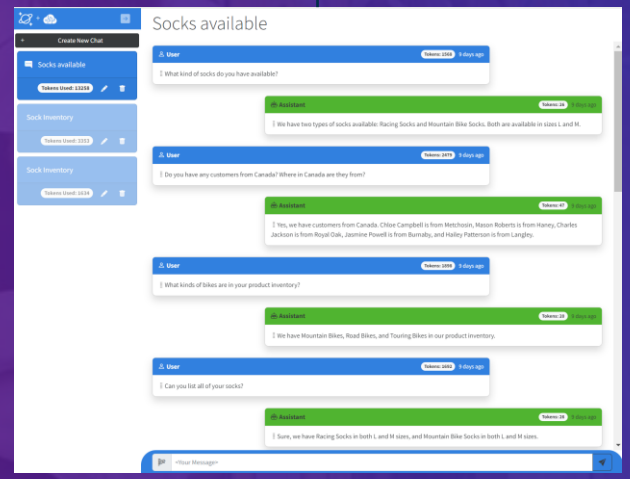
This diagram shows various documents represented as points in a two-dimensional space, with lines indicating semantic similarities between them.

RAG Data Indexing (Vectorization) example



RAG Completions example





RAG Limitations with Structure Data

RAG is particularly optimized for working with text data. It is less optimal for dealing with structured data.

- Most scenarios working with structured data will be multi-turn (for example create a query against a table, validate the query, execute the query, query again and join it with another table), of which RAG is sometimes employed in a step.
- RAG fundamentally only works with text, even if the data is structured and typed. It requires text to vectorize, when native format of data is not text, you are converting that data to text first. This doesn't always work.
- By converting structured data to text, you end up flattening the underlying similarity function, as now the examples have a lot of text components in common.
    - For example, format the data as a JSON document string. Transform the column names into properties.
    - Without a good discriminant you will end up retrieving lots of "hits" that are not relevant
- If the schema changes, the vectors break drastically.
- If the data changes often, you'll need to constantly update the retriever's index.
- A representation is an approximation. Divergence between the vector representation and the data it is meant to represent is an ever-present problem. For tasks like summarization, an approximation is fine but for tasks performing mathematical calculations it is likely not.
- If you are doing computations on top of the data, it's better to the computation in the database natively or generate code (like use Pandas) than to have the LLM try to reason its way thru it. As the number of rows being processed grows, the challenges of processing from context also grow.
- With tables that are larger than 100's of items, it's impossible to do a selection of a large number of items and have statistically meaningful/relevant outcomes with vector distance calculations. Selecting a statistically meaningful subset of data to add to the prompt is the problem with large numbers of records or properties.
- If data changes often, you must constantly update the vector index. Time series suffers from this problem.
- How do you secure your vectorized data? Oh, that imaginary quadrant over there in the space is OK for this user? There is no concept of RLS or CLS or other security. The only realistic approach is to go back to the source of the data that was used to generate the vector – this adds complexity and latency to the whole process.

Vectorization

Demos

Vectorization

Labs

Vectorization Databases

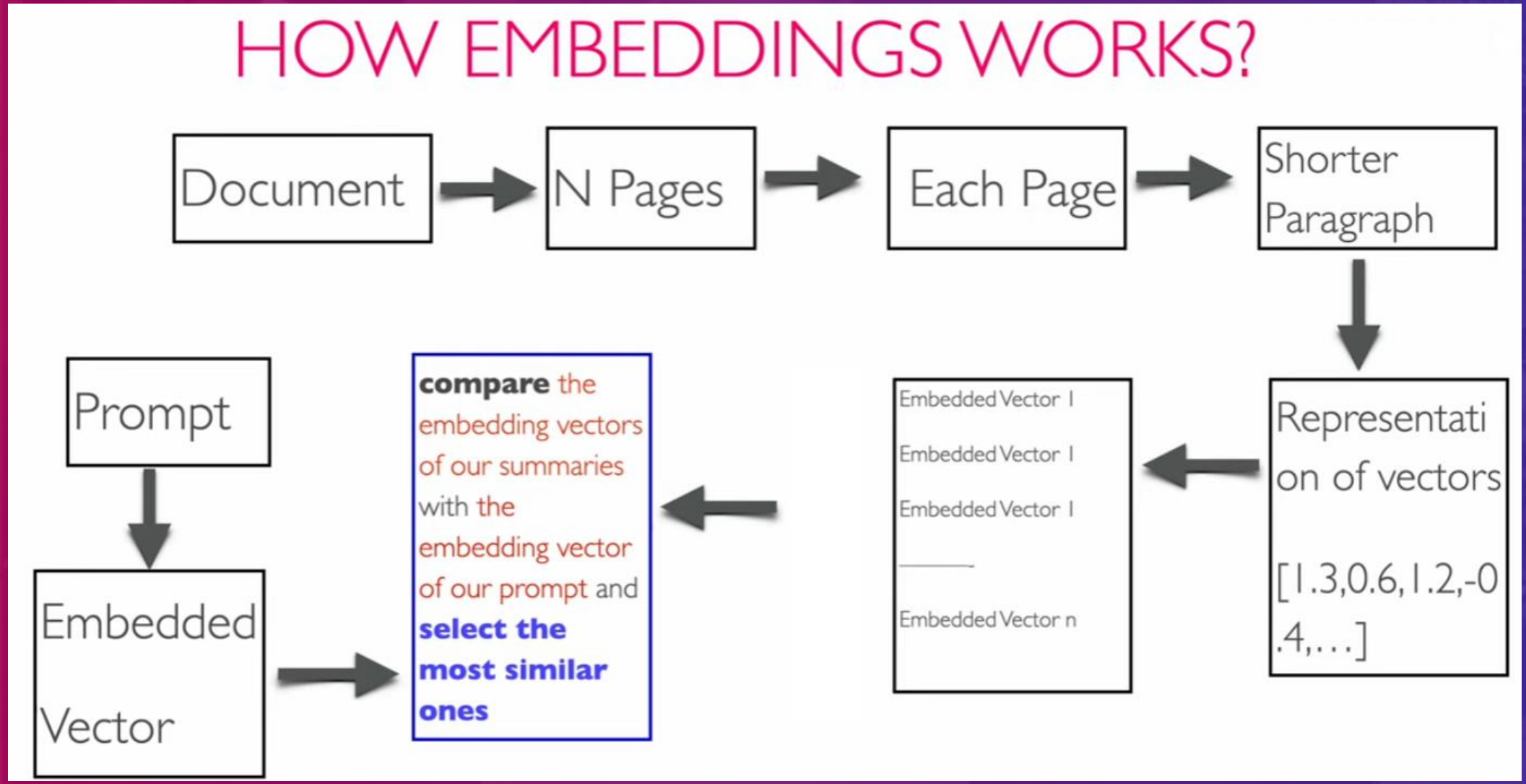
Vector Databases

Vector databases provide the data structures for storing and querying with vector embeddings.
As an information retrieval pattern, they support various forms of nearest neighbor search.
Given an input vector embedding, you can query a vector database for the top k vectors that are "closest" to it in the multi-dimensional space.
In truth, vector databases are really more like vector indexes because the way you use them follows these patterns:

- Indexing: Load your text documents into the database
  - Chunk the documents to a certain size.
  - For each chunk compute the vector embedding.
  - Store the vector embedding in the database and have it point to the source chunk by some ID
- Query: Given a piece of text, get the vector embedding.
  - Perform a k nearest neighbor search against the vectors.
  - For each of the top K vectors closest, retrieve the underlying text chunk.

You can expect most databases will become vector databases in the future, adding this "vector index" capability

Vector Databases with Embeddings



Few popular name in Vector Databases

Vector Database Connectors

- Chroma
- Cosmos DB
- Redis
- Azure AI Search
- Azure PostgreSQL
- Qdrant
- SharePoint
- More coming everyday…

Vector Databases

Demos

Vector Databases

Labs

Red Team Security

New Standards for AI Safety and Security

THE WHITE HOUSE

MENU

OCTOBER 30, 2023

Executive Order on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence

BRIEFING ROOM › PRESIDENTIAL ACTIONS

By the authority vested in me as President by the Constitution and the laws of the United States of America, it is hereby ordered as follows:

AI systems developers share safety test results, including the results of all red-team safety tests.

Develop standards, tools, and tests to help ensure that AI systems are safe, secure, and trustworthy

The National Institute of Standards and Technology will set the rigorous standards for extensive red-team testing to ensure safety before public release. The Department of Homeland Security will apply those standards to critical infrastructure sectors.

Protect against the risks of using AI to engineer dangerous biological materials

Protect Americans from AI-enabled fraud and deception

Establish an advanced cybersecurity program to develop AI tools to find and fix vulnerabilities

Order the development of a National Security Memorandum that directs further actions on AI and security

Protecting Americans' Privacy


THE WHITE HOUSE

MENU

OCTOBER 30, 2023

Executive Order on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence

BRIEFING ROOM › PRESIDENTIAL ACTIONS

By the authority vested in me as President by the Constitution and the laws of the United States of America, it is hereby ordered as follows:

Accelerating the development and use of privacy-preserving techniques

Strengthen privacy-preserving research and technologies

Evaluate how agencies collect and use commercially available information containing personally identifiable data

Develop guidelines for federal agencies to evaluate the effectiveness of privacy-preserving techniques

Advancing Equity and Civil Rights


THE WHITE HOUSE

MENU

OCTOBER 30, 2023

Executive Order on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence

BRIEFING ROOM ▸ PRESIDENTIAL ACTIONS

By the authority vested in me as President by the Constitution and the laws of the United States of America, it is hereby ordered as follows:

Keep AI algorithms from being used to exacerbate discrimination by landlords, Federal benefits programs and contractors

Address algorithmic discrimination

Ensure fairness throughout the criminal justice system

Standing Up for Consumers, Patients, and Students

Advance the responsible use of AI in healthcare and the development of affordable and life-saving drugs

Shape AI's potential to transform education by creating resources to support educators

Supporting Workers

Develop principles and best practices to mitigate the harms and maximize the benefits of AI for workers

Study and identify options for strengthening federal support for workers facing labor disruptions from AI

Promoting Innovation and Competition


THE WHITE HOUSE

MENU

OCTOBER 30, 2023

Executive Order on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence

BRIEFING ROOM › PRESIDENTIAL ACTIONS

By the authority vested in me as President by the Constitution and the laws of the United States of America, it is hereby ordered as follows:

Catalyze AI research across the United States

Promote a fair, open, and competitive AI ecosystem

Use existing authorities to expand the ability of highly skilled immigrants and nonimmigrants with expertise in critical areas to study, stay, and work in the United States

Advancing American Leadership Abroad

THE WHITE HOUSE

MENU

OCTOBER 30, 2023

Executive Order on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence

BRIEFING ROOM › PRESIDENTIAL ACTIONS

By the authority vested in me as President by the Constitution and the laws of the United States of America, it is hereby ordered as follows:

Expand bilateral, multilateral, and multistakeholder engagements to collaborate on AI.

Accelerate development and implementation of vital AI standards

Promote the safe, responsible, and rights-affirming development and deployment of AI abroad to solve global challenges

Ensuring Responsible and Effective Government Use of AI

THE WHITE HOUSE

MENU

OCTOBER 30, 2023

Executive Order on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence

BRIEFING ROOM ▸ PRESIDENTIAL ACTIONS

By the authority vested in me as President by the Constitution and the laws of the United States of America, it is hereby ordered as follows:

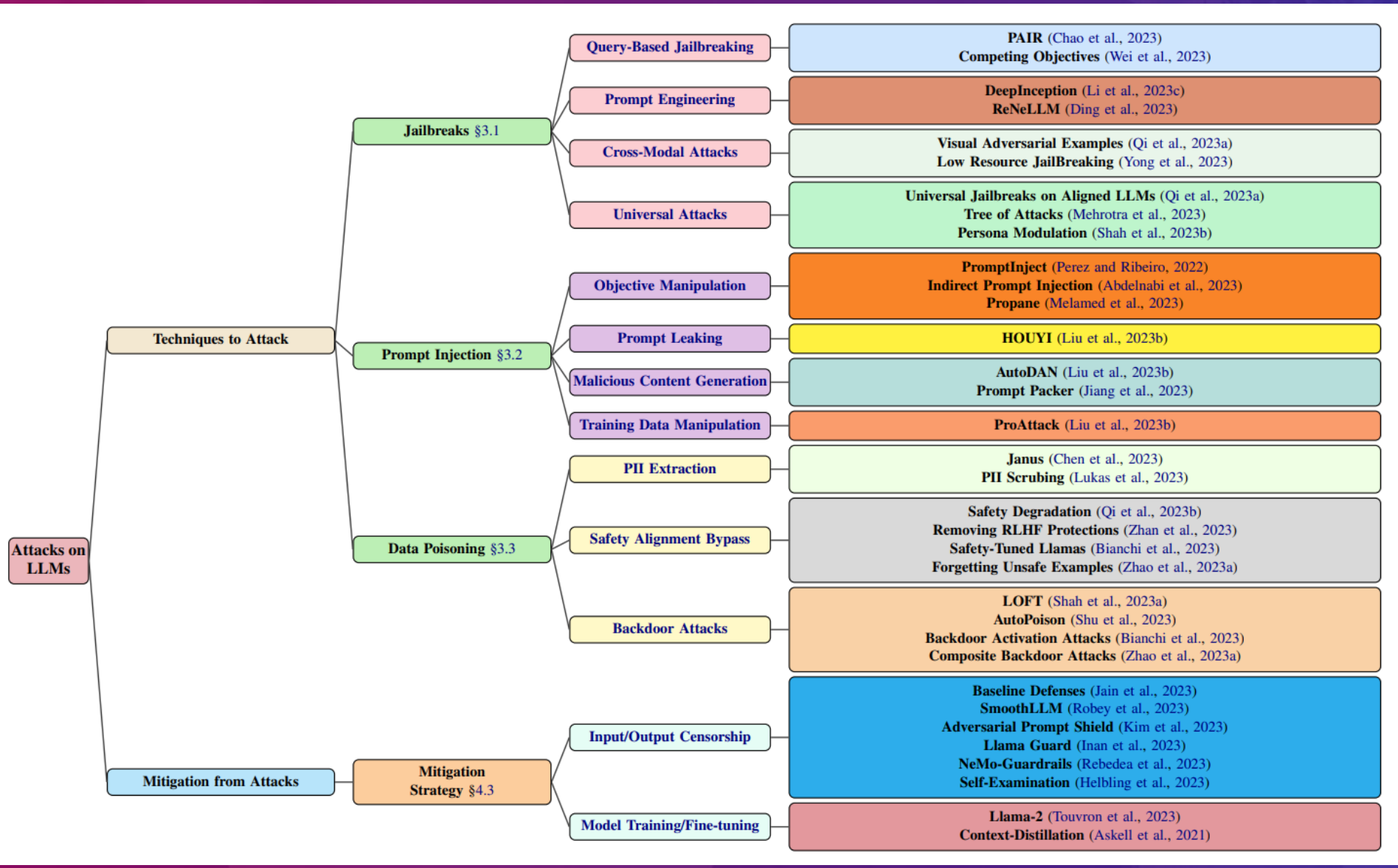
Issue guidance for agencies' use of AI

Help agencies acquire specified AI products and services faster, more cheaply, and more effectively through more rapid and efficient contracting

Accelerate the rapid hiring of AI professionals

Survey of LLM Attacks

Breaking Down the Defenses:
A Comparative Survey of Attacks on Large Language Models



https://arxiv.org/abs/2403.04786

Definition of LLM Attacks

When it comes to AI models. I like to think of red teaming as two things:

- Social engineering attacks: can you fool the person on the other end into doing something on your behalf?
- Hacking: Attacking the infrastructure surrounding the model.

What's novel is the use of AI to attack the AI. A common example of this being to use the AI to refine the attack prompt until it achieves the desired goal.

Common attack patterns:

- Grandma attack (overriding the system instructions by roleplaying)
- Logic attack (e.g., buying a car for $1)
- Encoding attack (e.g., use emojis or ASCII art to get around defenses)
- Escalation of privilege attack (Misconfigured file share releases company-wide compensation data to front line workers via a chatbot)

https://arxiv.org/abs/2403.04786

THANK YOU!

Lino Tadros – https://www.linkedin.com/in/linotadros
@LinoTadros @TheTrainingBoss